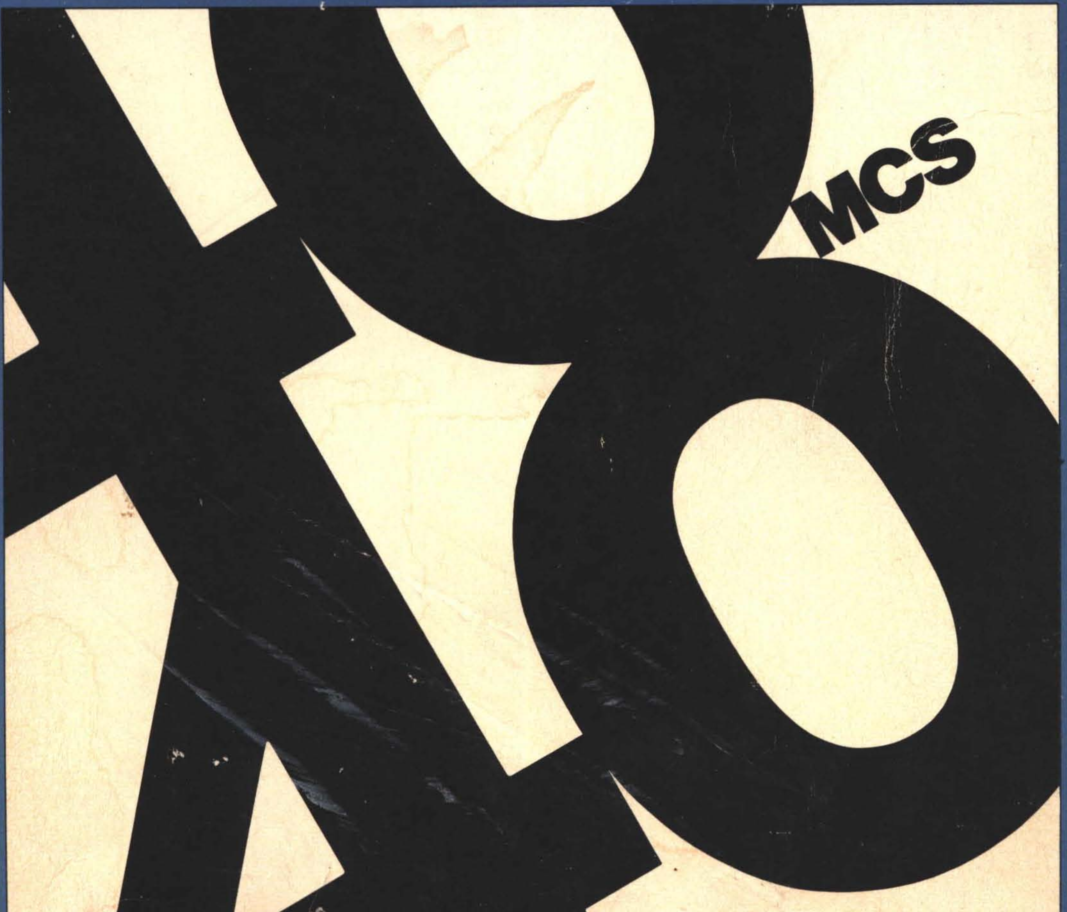
intel

MCS-40™ User's Manual For Logic Designers

November 1974



intel® MCS-40™ User's Manual For Logic Designers

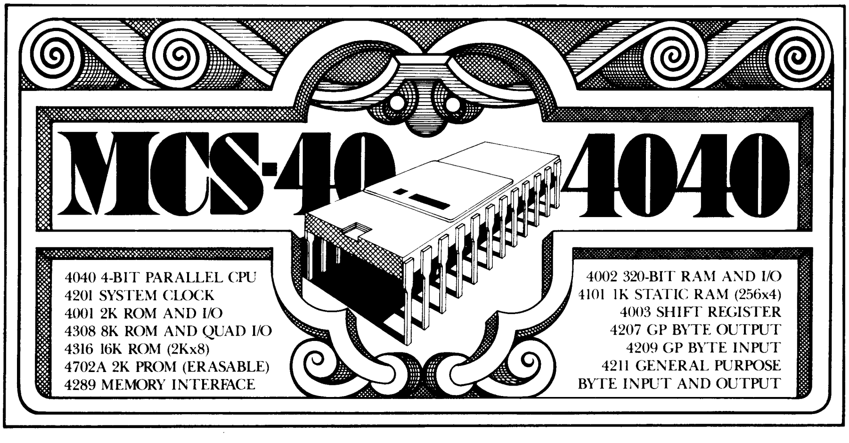
Intel Corporation has now doubled its line of four-bit microcomputer components with the introduction of the MCS-40 Microcomputer Device Family. The overwhelming growth trend to microcomputer design was established in 1971, by Intel's introduction of the first integrated microcomputer, the MCS-4™ system, now an industry standard. This extensive experience has allowed Intel to fill market demands with a consistent product delivery policy and produce the lowest cost usable microcomputer possible, the MCS-40 system.

The MCS-40 system provides the user with a new generation of components geared for random logic replacement and all designs requiring the unique advantages of a general purpose computer.

All Intel four-bit components are mutually compatible. Intel offers the user a comprehensive product development program consisting of Hardware and Software development aids and a network of regional application engineers. This User's Manual presents a complete description of the MCS-40 components.

Intel welcomes your comments and questions and looks forward to assisting you.

Howard A. Raphael
MCS-40 Product Manager



MCS-40 4040

4040 4-BIT PARALLEL CPU
4201 SYSTEM CLOCK
4001 2K ROM AND I/O
4308 8K ROM AND QUAD I/O
4316 16K ROM (2Kx8)
4702A 2K PROM (ERASABLE)
4289 MEMORY INTERFACE

4002 320-BIT RAM AND I/O
4101 1K STATIC RAM (256x4)
4003 SHIFT REGISTER
4207 GP BYTE OUTPUT
4209 GP BYTE INPUT
4211 GENERAL PURPOSE
BYTE INPUT AND OUTPUT

CONTENTS

LIST OF ILLUSTRATIONS

INTRODUCTION

Since their inception, digital computers have continuously become more efficient, expanding into new applications with each major technological improvement. The advent of minicomputers enabled the inclusion of digital computers as a permanent part of various process control systems. Unfortunately, the size and cost of minicomputers in "dedicated" applications has limited their use. Another approach has been the use of custom built systems made up of "random logic" (i.e., logic gates, flip-flops, counters, etc.). However, the huge expense and development time involved in the design and debugging of these systems has restricted their use to large volume applications where the development costs could be spread over a large number of machines.

Today, Intel offers the systems designer a new alternative . . . . the microcomputer. Utilizing the technologies and experience gained in becoming the world's largest supplier of LSI memory components, Intel has made the power of the digital computer available at the integrated circuit level.

ECONOMICS OF USING MICROCOMPUTERS

The MCS-40 T.M. microcomputers built around the 4040 CPU are a new extension of computer technology which offer users exciting possibilities for creating new products and services. Engineers are becoming more aware of the ways in which microcomputers can be applied to solve their problems. There are five basic reasons why many engineers have begun to use the MCS-40. These are:

1. Manufacturing costs of products can be significantly reduced.
2. Products can get to the market faster providing a company with the opportunity to increase product sales and market share.
3. Product capability is enhanced allowing manufacturers to provide customers with better products which can frequently command a higher price in the market place.
4. Development costs and time are reduced.
5. Product reliability is increased which leads to a corresponding reduction in both service and warranty costs.

Microcomputers simplify almost every phase of product development. The first step, as in any product design program, is to identify the various functions that the end system is expected to perform. These functions are then implemented by encoding suitable sequences of instructions (programs) in the memory elements. Data and certain types of programs will be stored in RAM circuits, while the basic program will be stored in ROM circuits. The microprocessor performs all of the system's functions by fetching the instructions in memory, executing them and communicating the results via the microcomputer's I/O ports. A single-chip microprocessor, executing the programmed logic stored in a single ROM element, can perform the same logical functions that have previously required many logic gates.

How Memory Replaces Random Logic

The MCS-40 T.M. microcomputer system replaces logic by storing program sequences in memory rather than implementing these sequences with gates and flip-flops. It can be said that 8 to 16 bits of memory are the logical equivalent of a single gate. Assuming that the type IC used today contains on the order of 10 gates, then one can conclude that logic can be stored in memory in a very cost effective fashion. The following table indicates the number of IC's which are replaced by a single ROM (Read Only Memory). The table was derived by using the assumptions that 8 to 16 bits of ROM replace a gate and that on the average an IC contains 10 gates.

ROM Memory Size Bits	Gates Replaced	IC's Replaced
2048	128-256	13-25
4096	256-512	25-50
8192	512-1024	50-100
16384	1024-2048	100-200

Table I. Number of IC's Replaced with a ROM (Read Only Memory)

REDUCING MANUFACTURING COSTS

If the burdened manufacturing cost of a digital electronic system is divided by the number of ICs, one generally finds that the system costs between $2 and $6 per IC to fabricate. The higher costs are generally associated with systems manufactured in volumes from 10 to 100 units annually. The table below presents a more detailed analysis of the source of these surprisingly high costs. The costs, themselves, are stated conservatively.

IC	.50
Incoming Inspection	.05
PC Card	.50
Fabrication	.05
Board Test and Rework	.10
Connector	.05
Discretes	.05
Wiring	.10
Power	.10
Cabinetry, Fans, Etc.	.10
	$1.60

Table II. System Manufacturing Costs Per IC

The ASP (average sale price) of an Integrated Circuit today is approximately 50¢. Incoming inspection and testing of these ICs costs the average company 5¢. However, many companies are now buying aged and tested circuits for their applications in order to increase system reliability. This adds about 15¢ to unit costs. Simple PC cards may cost as little as 25¢ an IC position, but the average cost in most applications for high quality cards is closer to 50¢. Sophisticated multilayer cards used in many high performance systems frequently cost *over a dollar a position.* When customers put ICs in sockets and then wire wrap cards, the cost per IC position quickly approaches $2. Customers with automatic IC insertion equipment and efficient flow soldering machines can fabricate a PC card for as low as 3¢ an IC position, though the average price is closer to 5¢. Board test and rework add another dime to system cost, while the cost of a connector divided by the number of ICs per printed circuit card frequently exceeds 5¢. In general, resistors, capacitors, power bus bars, etc., add a cost of 5¢ an IC position. Systems frequently average one wire or more per IC position and the wires put in with automatic equipment frequently cost over 10¢. Finally, the cost of power supplies and mechanical packaging add another 20¢ an IC position.

To determine the total savings in system manufacturing cost, the user must subtract the cost of implementing an equivalent system with a microcomputer. In moderate volumes, an MCS 40 with 16,384 bits of ROM, a processor, and a minimal amount of RAM can be purchased for under $50. This system has the potential of displacing between $150 and $600 of system manufacturing cost.

Reducing Development Time and Cost

The MCS-40$^{\text{T.M.}}$ system simplifies almost every phase of product development. Because of the extensive design aids and support supplied with microcomputers, it is relatively easy to develop application programs that tailor the device to the system. One of the most significant advantages of the MCS-40 system is modularity of the component family which aids in development. Development cycles can be cut by as long as six to twelve months. The table below tabulates a number of the steps in a development cycle and indicates how the MCS-40 system can affect them. Surprisingly, product definition is frequently speeded up once the decision has been made to use a microcomputer. This is because the incremental cost for adding features to the system is usually small and can be easily extimated. For example, added features such as automatic tax computation for an electronic cash register may only require the addition of a single ROM. The addition of one LSI chip has a minimal effect on total system cost, power and packaging requirements. On the other hand, the same function implemented with IC logic might require two or three fairly large PC cards filled with MSI and SSI.

System and logic design time is also reduced. Programming is a faster way to design than using logic diagrams. If you're used to logic design and the idea of designing with programmed logic seems like too radical a change, regardless of advantages, there's no need to worry because Intel has already done most of the groundwork for you. The INTELLEC® 4 Development Systems provide flexible, inexpensive and simplified methods for OEM product development. The INTELLEC® 4 system provides RAM program storage making program loading and modification easier, a display and control console for system monitoring and debugging, expandable memory and I/O, TTY interface, a PROM programming cap-

Development Steps	Conventional System	Programmed Logic
Product definition		Simplified because of ease of incorporating features
System and logic design	Done with logic diagrams	Can be programmed with design aids (compilers, assemblers, editors)
Debug	Done with conventional lab instrumentation	Software and hardware aids reduce time
PC card layout		Fewer cards to layout
Documentation		Less hardware to document
Cooling and packaging		Reduced system size and power consumption eases job
Power distribution		Less power to distribute
Engineering changes	Done with yellow wire	Change program in PROM

Table III. How Development Time and Cost are Reduced with Microcomputers

ability and a standard software package (System Monitor and Assembler). In addition to the standard software package available with the INTELLEC® 4 development system, Intel offers a cross-assembler and simulator written in FORTRAN IV and designed to run on any large scale computer. The programs may be procured directly from Intel or from a number of nation-wide computer time-sharing services. Intel's Microcomputer Systems Group and Regional Applications Engineers are available to provide assistance in every phase of your product development.

Intel provides complete documentation on all their hardware and software products. In addition to this User's Manual, there are the:

- MCS-4™ Assembly Language Programming Manual
- INTELLEC 4 Operator's Manual
- INTELLEC 4 Hardware and Microcomputer Modules Reference Manual

This User's Manual is intended as your primary source of information on the 4004 and 4040 processors and their family components.

The above aids also reduce the time for system debugging. PC card layout time is reduced simply because there are fewer cards to lay out. This reduction in hardware also reduces the load on the technical writers who must develop maintenance manuals. Parts lists become shorter, easing the task of transferring the product to manufacturing. Cooling, packaging, and power distribution problems frequently become trivial. Finally, engineering changes that are difficult to make and frequently tedious to document, become simple program changes. These can be made by changing the pattern in a ROM or PROM (Programmable Read Only Memory) such as Intel's 4702A.

Products Can Get to the Market Faster

The use of the MCS-40™ system allows the user to have a shorter development cycle, thus reducing development time and allows a faster response to the market from conception to production. This affords the MCS-40 user a definite advantage.

High Product Prices Due to Enhanced Product Capability

Product features can be easily added to microcomputer systems by simply adding more program storage. Many MCS 40 users have utilized this characteristic of microcomputers as a way of increasing the value of their product without significantly adding to the cost of the product. Examples of such easily added features are: putting automatic tax computations into a cash register by adding more ROM, adding automatic calibration features to instruments, and making traffic controllers that automatically sense traffic load and adjust the duration of the signals, etc. The MCS-40 microcomputer system offers the designer a way to add significant features to systems at trivial costs.

Reduced Complexity

Because the MCS-40 system eliminates many ICs and consequently the failures associated with these devices, it can significantly increase system reliability. Most of the failures in a digital system occur because an interconnect has failed. The use of a typical 16 pin IC will introduce approximately 36 interconnectors in a system. There are 16 interconnections from the chip to the lead frame, 16 from the lead frame to the PC card, and approximately 2 interconnections from the PC card to the back plane, and 2 interconnections from back plane point to back plane point per IC. If one ROM eliminates fifty ICs, then it eliminates approximately 1800 interconnections.

Conclusion

The MCS-40 microcomputer represents a new and exciting advantage in the state of the art. It has reduced the cost of putting basic computation into a device by a factor of 10 or more. As such it can bring to many new systems the benefits of using computers. Because of its small size and small cost, the MCS-40 system can be designed into many devices such as cash registers, scales, stop lights, instruments, etc., where the use of a computer was once unthinkable.

The benefits of putting an MCS-40 microcomputer into a system go far beyond the advantages of merely being able to include computation or decision making into the device being designed. As previously indicated, the use of an MCS-40 system can affect such basic things as manufacturing cost, market share, development costs and time, and system reliability and serviceability.

The following list summarizes the MCS-40 System Component's major features:

4040 – Central Processor Unit

- Instructions (60 total) including Logical Operations and Read Program Memory
- Large number of family devices
- 10.8 microsecond instruction cycle standard
- 2-phase dynamic operation
- Instruction set includes conditional branching, jump to subroutine and indirect fetching
- Logical instructions
- Binary and decimal arithmetic modes
- CPU directly compatible with MCS-4 ROMs and RAMs
- Unlimited number of input and output lines
- Interrupt capability
- Single step operation
- 8K byte memory addressing capability and up to 5120 bits of RAM
- 24 index registers
- Subroutine nesting to 7 levels

4004 – Central Processor Unit

- 4 bit parallel CPU with 46 instructions
- Can be used with standard memory components
- 2-phase dynamic operation
- Instruction set includes conditional branching, jump to subroutine and indirect fetching
- Binary and decimal arithmetic modes
- Unlimited number of input and output lines
- 10.8 microsecond instruction cycle standard
- 4K byte memory addressing capability
- 16 index registers

4201 – Clock Generator (Available First Quarter 1975)

- Provides 2-phase clock – TTL and MOS
- Crystal controlled oscillator
- Directly drives MCS-40$^{\text{T.M.}}$ set
- Generates power on reset for MCS-40
- Provides single step circuit for 4040

MEMORIES

4308 – Mask Programmable ROM

- 1K x 8 program storage
- Four independent 4 bit I/O ports
- Buffered inputs and outputs
- Equivalent of (4) 4001 ROMs (256 x 8)
- Directly TTL Compatible

4001 – Mask Programmable ROM

- 256 x 8 program storage
- Programmable 4 bit I/O port
- Directly TTL Compatible

4002 – RAM

- 320 bits organized 4 x 80
- Output 4 bit port
- Directly TTL compatible

INPUT/OUTPUT DEVICES

4207 – General Purpose Output Byte I/O

- 8 bits of data output
- 4 bits of status input
- 4 bits of control output
- Directly compatible to CPU
- Directly TTL compatible

4209 – General Purpose Input Byte I/O

- 8 bits of data input with strobe
- 4 bits of status input
- 4 bits of control output
- Directly compatible to CPU
- Directly TTL compatible

4211 – General Purpose Byte I/O

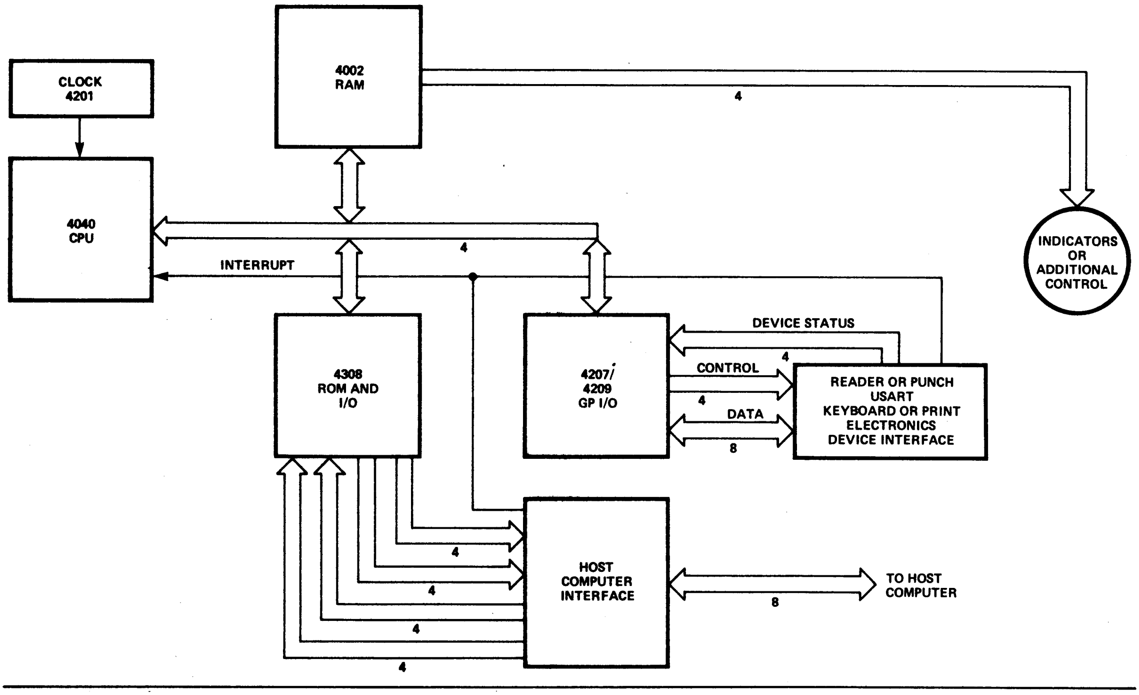
- 8 bits of data input with strobe
- 8 bits of data output
- Directly compatible with CPU
- Directly TTL compatible

4003 – Shift Register

- 10 bit serial in/parallel out
- Serial-output allows for expansion
- Asyncronous clock

4289 – Standard Memory Interface

- Direct interface to all standard memories: TTL, NMOS, PMOS, CMOS
- Allows READ and WRITE program memory
- Unlimited Input/Output expansion
- Single package equivalent of 4008/4009



Typical MCS-40 Application – Peripheral I/O Controller.

SUPPLEMENTAL DEVICES

The following devices are supplemental and are compatible with the 4289.

4702 – Erasable and Electrically Reprogrammable ROM

- 2048 bits, organized 256 x 8
- Fast programming – 2 minutes for 2048 bits
- Inputs and outputs TTL compatible
- Three-state outputs
- Alterable Program Memory in system development

4316 – Mask Programmable ROM

- 16,384 bits, organized 2048 x 8
- Fully decoded

4101 RAM

- 1024 bits – organized 256 x 4
- Static operation
- Fully decoded
- Directly TTL compatible
- Three-state outputs
- Used for writeable Program Memory

3216/3226 – 4 bit Bidirectional Bus

- Low input load current
- High output drive
- Three-state outputs
- Inverted (3226) and non-inverted (3216) operations

THE FUNCTIONS OF A COMPUTER

This section introduces certain basic computer concepts. It provides background information and definitions which will be useful in later sections of this manual. Those already familiar with computers may skip this material, at their option.

A Typical Computer System

A typical digital computer consists of:

a. A central processor unit (CPU)
b. A memory
c. Input/Output (I/O) ports

The program memory serves primarily as a place to store *instructions,* the coded pieces of data that direct the activities of the CPU. A group of logically related instructions stored in memory is referred to as a *program.* The CPU "reads" each instruction from memory in a logically determinate sequence, and uses it to initiate processing actions. If the program structure is coherent and logical, processing produces intelligible and useful results.

The data memory is used to store the data to be manipulated. The CPU can access any data stored in memory; but often the memory is not large enough to store the entire data bank required for a particular application. The program can be resolved by providing the computer with one or more *input ports.* The CPU can address these ports and input the data contained there. The addition of input ports enables the computer to receive information from external equipment (such as a paper tape reader) at high rates of speed and in large volumes.

Almost any computer requires one or more *output ports* that permit the CPU to communicate the result of its processing to the outside word. The output may go to a display, for use by a human operator, to a peripheral device that produces "hard-copy", such as a line-printer, to a peripheral storage device, such as a magnetic tape unit, or the output may constitute process control signals that direct the operations of another system, such as an automated assembly line. Like input ports, output ports are addressable. The input and output ports together permit the processor to interact with the outside world.

The CPU unifies the system. It controls the functions performed by the other components. The CPU must be able to fetch instructions from memory, decode their binary contents and execute them. It must also be able to reference memory and I/O ports as necessary in the execution of instructions. In addition, the CPU should be able to recognize and respond to certain external control signals, such as INTERRUPT and STOP requests. The functional units within a CPU that enable it to perform these functions are described below.

The Architecture of a CPU

A typical central processor unit (CPU) consists of the following interconnected functional units:

1. Registers
2. Arithmetic/Logic Unit (ALU)
3. Control Circuitry

Registers are temporary storage units within the CPU. Some registers, such as the program counter and instruction register, have dedicated uses. Other registers, such as the accumulator, are for more general purpose use.

ACCUMULATOR:

The accumulator usually stores one of the operands to be manipulated by the ALU. A typical instruction might direct the ALU to add the contents of some other register to the contents of the accumulator and store the result in the accumulator itself. In general, the accumulator is both a source (operand) and destination (result) register.

Often a CPU will include a number of additional general purpose registers that can be used to store operands or intermediate "scratch-pad" data.

PROGRAM COUNTER (JUMPS, SUBROUTINES AND THE STACK):

The instructions that make up a program are stored in the system's memory. The central processor examines the contents of the memory, in order to determine what action is appropriate. This means that the processor must know which location contains the next instruction.

Each of the locations in memory is numbered, to distinguish it from all other locations in memory. The number which identifies a memory location is called its *address.*

The processor maintains a counter which contains the address of the next program instruction. This register is called the *program counter.* The processor updates the program counter by adding "1" to the counter each time it fetches an instruction, so that the program counter is always current.

The programmer therefore stores his instructions in numerically adjacent addresses, so that the lower addresses contain the first instructions to be executed and the higher addresses contain later instructions. The only time the programmer may violate this sequential rule is when the last instruction in one block of memory is a *jump* instruction to another block of memory.

A jump instruction contains the address of the instruction which is to follow it. The next instruction may be stored in any memory location, as long as the programmed jump specifies the correct address. During the execution of a jump instruction, the processor replaces the contents of its program counter with the address embodied in the jump. Thus, the logical continuity of the program is maintained.

A special kind of program jump occurs when the stored program accesses or "branches" to a subroutine. In this kind of jump, the processor is logically required to "remember" the contents of the program counter at the time that the jump occurs. This enables the processor to resume execution of the main program when it is finished with the last instruction of the subroutine.

A *subroutine* is a program within a program. Usually it is a general-purpose set of instructions that must be executed repeatedly in the course of a main program. Routines which calculate the square, the sine, or the logarithm of a program variable are good examples of the functions often written as subroutines. Other examples might be programs designed for inputting or outputting data to a particular peripheral device.

The processor has a special way of handling subroutines, in order to insure an orderly return to the main program. When the processor receives a jump to subroutine instruction, it increments the program counter and stores the counter's contents in a register memory area known as the *stack.* The stack thus saves the address of the instruction to be executed after the subroutine is completed. Then the processor stores the address specified in the subroutine jump in its program counter. The next instruction fetched will therefore be the first step of the subroutine.

The last instruction in any subroutine is a branch back. Such an instruction need specify no address. When the processor fetches a branch back instruction, it simply replaces the current contents of the program counter with the address on the top of the stack. This causes the processor to resume execution of the program at the point immediately following the original branch.

Subroutines are often *nested;* that is, one subroutine will sometimes call a second subroutine. The second may call a third, and so on. This is perfectly acceptable, as long as the processor has enough capacity to store the necessary return addresses, and the logical provision for doing so. In other words, the maximum depth of nesting is determined by the depth of the stack itself. If the stack has space for storing three return addresses, then three levels of subroutines may be accommodated.

Processors have different ways of maintaining stacks. Most have facilities for the storage of return addresses built into the processor itself. The integral stack is usually more efficient, since fewer steps are involved in the execution of a call or a return.

INSTRUCTION REGISTER AND DECODER:

Every computer has a *word length* that is characteristic of that machine. A computer's word length is usually determined by the size of its internal storage elements and interconnecting paths (referred to as *buses*); for example, a computer whose registers and buses can store and transfer 4 bits of information has a characteristic word length of 4 bits and is referred to as a 4 bit parallel processor.

The characteristic eight bit field is sometimes referred to as a *byte*, a four bit field can be referred to as a *nibble.*

Each operation that the processor can perform is identified by a unique binary number known as an *instruction code.* An eight bit word used as an instruction code can distinguish among 256 alternative actions, more than adequate for most processors.

The processor fetches an instruction in two distinct operations. In the first, it transmits the address in its program counter to the memory. In the second, the memory returns the addressed byte to the processor. The CPU stores this instruction byte in a register known as the *instruction register,* and uses it to direct activities during the remainder of the instruction execution.

The mechanism by which the processor translates an instruction code into specific processing actions requires more elaboration than we can here afford. The concept, however, will be intuitively clear to any experienced logic designer. The eight bits stored in the instruction register can be decoded and used to selectively activate one of a number of output lines, in this case up to 256 lines. Each line represents a set of activities associated with execution of a particular instruction code. The enabled line can be combined coincidentally with selected timing pulses, to develop electrical signals that can then be used to initiate specific actions. This translation of code into action is performed by the *instruction decoder* and by the associated control circuitry.

An eight bit field is more than sufficient, in most cases, to specify a particular processing action. There are times, however, when execution of the instruction code requires more information than eight bits can convey.

One example of this is when the instruction references a memory location. The basic instruction code identifies the operation to be performed, but cannot specify the object address as well. In a case like this, a two-word instruction must be used. Successive instruction bytes are stored in sequentially adjacent memory locations, and the processor performs two fetches in succession to obtain the full instruction. The first byte retrieved from memory is placed in the processor's instruction register, and the second byte is placed in temporary storage, as appropriate. When the entire instruction is fetched, the processor can proceed to the execution phase.

ADDRESS REGISTER(S):

A CPU may use a register or register-pair to temporarily store the address of a memory location that is to be accessed for data. If the address register is *programmable,* (i.e., if there are instructions that allow the programmer to alter the contents of the register) the program can "build" an address in the address register prior to executing a *memory reference* instruction (i.e., an instruction that reads data from memory, writes data to memory or operates on data stored in memory).

ARITHMETIC/LOGIC UNIT (ALU):

All processors contain an arithmetic/logic unit, which is often referred to simply as the ALU. By way of analogy, the ALU may be thought of as a super adding machine with its keys commanded automatically by the control signals developed in the instruction decoder and the control circuitry. This is essentially how the first stored-program digital computer was conceived.

The ALU naturally bears little resemblance to a desk-top adder. The major difference is that the ALU calculates by creating an electrical analogy, rather than by mechanical analogy. Another important difference is that the ALU uses binary techniques — rather than decimal methods — for representing and manipulating numbers. In principle, however, it is convenient to think of the ALU as an electronically controlled calculator.

The ALU must contain an *adder* which is capable of combining the contents of two registers in accordance with the logic of binary arithmetic. This provision permits the processor to perform arithmetic manipulations on the data it obtains from memory and from its other inputs.

Using only the basic adder a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALUs provide other built-in functions, including hardware subtraction, boolean logic operations, and shift capabilities.

The ALU contains *flag bits* which register certain conditions that arise in the course of arithmetic manipulations. Flags typically include *carry* and *zero.* It is possible to program jumps which are conditionally dependent on the status of one or more flags. Thus, for example, the program may be designed to jump to a special routine, if the carry bit is set following an addition instruction. The presence of a carry generally indicates an overflow in the accumulator, and sometimes calls for special processing actions.

CONTROL CIRCUITRY:

The control circuitry is the primary functional unit within a CPU. Using clock inputs, the control circuitry maintains the proper sequence of events required for any processing task. After an instruction is fetched and decoded, the control circuitry issues the appropriate signals (to units both internal and external to the CPU) for initiating the proper processing action. Often the control circuitry will be capable of responding to external signals, such as an interrupt request. An *interrupt* request will cause the control circuitry to temporarily interrupt main program execution, jump to a special routine to service the interrupting device, then automatically return to the main program.

COMPUTER OPERATIONS

There are certain operations that are basic to almost any computer. A sound understanding of these basic operations is a necessary prerequisite to examining the specific operations of a particular computer.

TIMING:

The activities of the central processor are cyclical. The processor fetches an instruction, performs the operations required, fetches the next instruction, and so on. An orderly sequence of events like this requires timing, and the CPU therefore contains a free running oscillator clock which furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as a *machine cycle.* The portion of a cycle identified with a clearly defined activity is called a phase. And the interval between pulses of the timing oscillator is referred to as a *clock period.* As a general rule, one or more clock periods are necessary to the completion of a phase, and there are several phases in a cycle.

INSTRUCTION FETCH:

The first phase(s) of any machine cycle will be dedicated to fetching the next instruction. The CPU issues a read operation code and the contents of the program counter are sent to program memory, which responds by returning the next instruction word. The first word of the instruction is placed in the instruction register. If the instruction consists of more than one word, additional cycles are required to fetch each word of the instruction. When the entire instruction is present in the CPU, the program counter is incremented (in preparation for the next instruction fetch) and the instruction is decoded. The operation specified in the instruction will be executed in the ramaining phases of the machine cycle. The instruction may call for a memory read or write, an input or output and/or an internal CPU operation, such as a register-to-register transfer or an add-registers operation.

MEMORY READ:

The instruction fetched may then call for data to be read from data memory into the CPU. The CPU issues a read operation code and sends the proper memory address; memory responds by returning the requested word. The data received is placed in the accumulator (not the instruction register).

MEMORY WRITE:

A program memory write operation is similar to a read except for the direction of data flow. The CPU issues a write operation code, sends the proper memory address, then sends the data word to be written into the addressed memory location.

INPUT/OUTPUT:

Input and Output operations are similar to memory read and write operations with the exception that a peripheral I/O port is addressed instead of a memory location. The CPU issues the appropriate input or output command, sends the proper device address and either receives the data being input or sends the data to be outpout.

Data can be input/output in either parallel or serial form. All data within a digital computer is represented in binary coded form. A binary data word consists of a group of bits; each bit is either a one or a zero. *Parallel* I/O consists of transferring all bits in the word at the same time, one bit per line. *Serial* I/O consists of transferring one bit at a time on a single line. Naturally serial I/O is much slower, but it requires considerably less hardware than does parallel I/O.

INTERRUPTS:

Interrupt provisions are included on many central processors, as a means of improving the processor's efficiency. Consider the case of a computer that is processing a large volume of data, portions of which are to be output to a printer. The CPU can output a byte of data within a single machine cycle but it may take the printer the equivalent of many machine cycles to actually print the character specified by the data byte. The CPU will have to remain idle waiting until the printer can accept the next data byte. If an interrupt capability is implemented on the computer, the CPU can output a data byte then return to data processing. When the printer is ready to accept the next data byte, it can request an interrupt. When the CPU acknowledges the interrupt, it suspends main program execution and automatically branches to a routine that will output the next data byte. After the byte is output, the CPU continues with main program execution. Note that this is, in principle, quite similar to a subroutine call, except that the jump is initiated externally rather than by the program.

More complex interrupt structures are possible, in which several interrupting devices share the same processor. Interruptive processing is an important feature that enables maximum utilization of a processor's capacity.

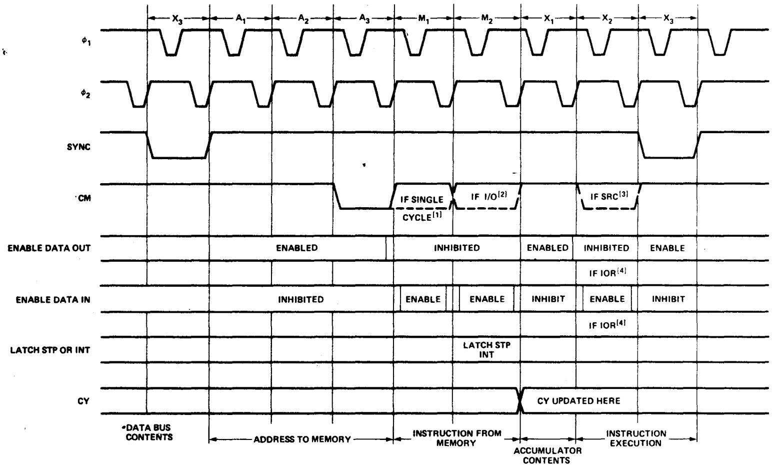
MCS-40™ SYSTEM OPERATION

The MCS-40 microcomputer, like all computer systems, contains a processor and memory. The processor (4040,4004) is contained in one dual in-line I.C. package. The memory can be either ROM program memory ro RAM data memory. In addition to information exchanges between the processor and memory, the processor also provides input and output (I/O) to external devices. The I/O faculties of the MCS-40 system are found in a separate I/O device or shared with memory elements, allowing for a reduction in package count. All MCS-40 inter-element communication transpires over a four bit data bus (D_0-D_3), D_3 being the most significant bus bit.

The data bus transfers information from the processor to the memory and I/O elements such as instruction addresses, operand addresses, operands, and I/O data. The processor receives instructions, operands, and I/O data back from the other elements. All traffic on the bus is contained within one instruction cycle for one cycle instructions. The instruction cycle is subdivided into equal time segments. Each segment is equivalent to one system clock period. Information on the data bus is altered from segment to segment. The first three time segments present a twelve bit (three groups of four bits) instruction address to the memory, least significant nibble first. The fourth and fifth segments provide the 8 bit instruction sequentially placed on the data bus by program memory. The sixth segment is utilized for instruction decode. The remaining two segments are used for program execution. Operands and I/O data can be found on the data bus during this time, depending on the instruction being executed.

Instructions requiring two cycles proceed in a similar manner as described above. Complete execution requires 16 clock periods which are partitioned into two eight segment cycles. During the first cycle the instruction will be fetched. Information fetched during this first cycle may also include a portion of the operand or an indirect address register. The second cycle will always fetch the operand and perform the execution.

Differentiation between one and two cycle instructions are performed by all elements on the data bus by decoding the unique instructions.



4040 Basic Timing Diagram

CHAPTER 1 PROCESSORS

MCS-40™ CENTRAL PROCESSOR UNITS

The MCS-40 family consists of two powerful central processor architectures, that of the 4040, and its predecessor, the 4004. The 4004 instruction set is a subset of the 4040 instruction set and programs generated for the 4004 will operate on the 4040. Hence, the 4040 is electrically and functionally upward compatible with the 4004. This mutual compatability is true for all family components.

The 4004 will be described first, followed by a description of the 4040 as an extension of the 4004. The last section is a detailed description of the instruction format and operation. The reader may find it useful to reference this section when reading the component descriptions.

The user will find the 4004/4040 Assembly Language Programming Manual useful as additional reference material.

4004 CENTRAL PROCESSOR UNIT

Introduction

The 4004 is a central processor unit designed to work in conjunction with the other members of the MCS 40 microcomputer 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 user peripheral devices through the RAM, ROM, GP I/O 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 ROM bank of up to 4K words of program memory.

Instructions

The instruction repertoire of the 4004 consists of:

a. 16 machine instructions (5 of which are double length)
b. 14 accumulator group instructions (Decimal/Binary operation)
c. 16 input/output instructions

The instruction set and its format will be described in detail in a subsequent section.

Hardware Description

The 4004 is packaged in a 16 pin DIP. The pin configuration is shown in the following figure. A brief functional description of each pin is given below:

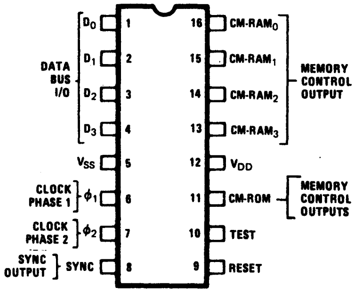


Figure 1-1. 4004 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1-4	D_0-D_3	Bidirectional data bus. All address and data communication between the processor and the RAM and ROM chips is handled by way of these 4 lines.
5	V_{SS}	Most positive supply voltage.
6-7	ϕ_1-ϕ_2	Non-overlapping clock signals which determine processor timing.
8	SYNC	SYNC output. Synchronization signal generated by the processor and sent to ROM and RAM chips. Indicates beginning of instruction cycle.

Pin No.	Designation	Description of Function
9	RESET	RESET input. A "1" level applied to this pin clears all flag and status flip-flops and forces the program counter to 0. To completely clear all of the address and index registers, RESET must be applied for 64 clock cycles (8 machine cycles).
10	TEST	TEST input. The logical state of this input can be examined with the JCN instruction.
11	CM-ROM	This pin enables a ROM bank, which can contain up to 4K words of program using 4001, 4308, or 4289. It also enables the GP I/O devices (4207, 4209, 4211) which are attached to the CM-ROM.
12	V_{DD}	Main supply voltage to the processor. Value must be V_{SS} -15.0V ±5%.
13-16	CM-RAM_0- CM-RAM_3	CM-RAM outputs. These outputs act as bank select signals for the 4002 RAM chips in the system.

Basic Timing

The MCS 40 system requires two non-overlapping clock phases, ϕ_1, ϕ_2 which are supplied by the 4201 clock generator. The 4004 will generate a SYNC signal every 8 clock periods and will send it to the other system components. The SYNC signal marks the beginning of each instruction cycle. The other MCS-40™ components will use the SYNC, ϕ_1, and ϕ_2 signals to generate external timing.

A typical machine cycle starts with the CPU sending a synchronization signal (SYNC) to the ROM's and RAM's. Next, 12 bits of ROM address are sent to the data bus using three clock cycles (@ .75 MHz). The address is then incremented by one and stored in the program counter. The selected ROM sends back 8 bits of instruction or data during the following 2 clock cycles. This information is stored in two registers: OPR and OPA. The next three clock cycles are used to execute the instruction. (See Basic Instruction Cycle.)

Basic Description of Major Circuit Blocks

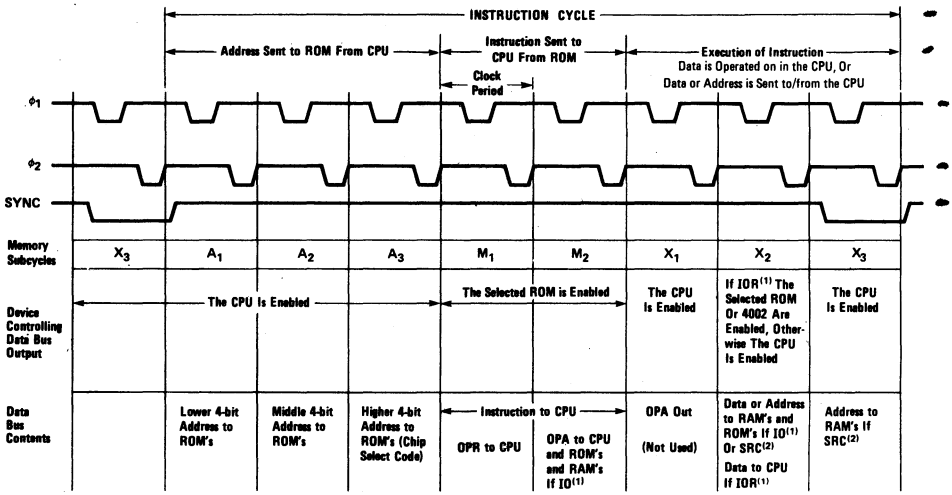
The 4004 block diagram shown contains the following functional blocks:

1. Address register (program counter and stack organized 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. 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


INSTRUCTION CYCLE
Address Sent to ROM From CPU
Instruction Sent to CPU From ROM
Clock Period
Execution of Instruction
Data is Operated on in the CPU, Or Data or Address is Sent to/from the CPU
ϕ_1
ϕ_2
SYNC
Memory Subcycles
X_3
A_1
A_2
A_3
M_1
M_2
X_1
X_2
X_3
Device Controlling Data Bus Output
The CPU Is Enabled
The Selected ROM is Enabled
The CPU Is Enabled
If IOR(1) The Selected ROM Or 4002 Are Enabled, Otherwise The CPU Is Enabled
The CPU Is Enabled
Data Bus Contents
Lower 4-bit Address to ROM's
Middle 4-bit Address to ROM's
Higher 4-bit Address to ROM's (Chip Select Code)
Instruction to CPU
OPR to CPU
OPA to CPU and ROM's and RAM's If IO(1)
OPA Out
(Not Used)
Data or Address to RAM's and ROM's If IO(1) Or SRC(2)
Data to CPU If IOR(1)
Address to RAM's If SRC(2)

(1) IO instructions control the flow of information between accumulator in CPU, I/O lines in ROM's and RAM's and RAM storage. IOR stands for IO Read. In this case the CPU will receive data from RAM storage locations or I/O input lines.

(2) The SRC instruction designates the chip number and address for a following IO instruction.

Figure 1-2. 4004 Basic Instruction Cycle.

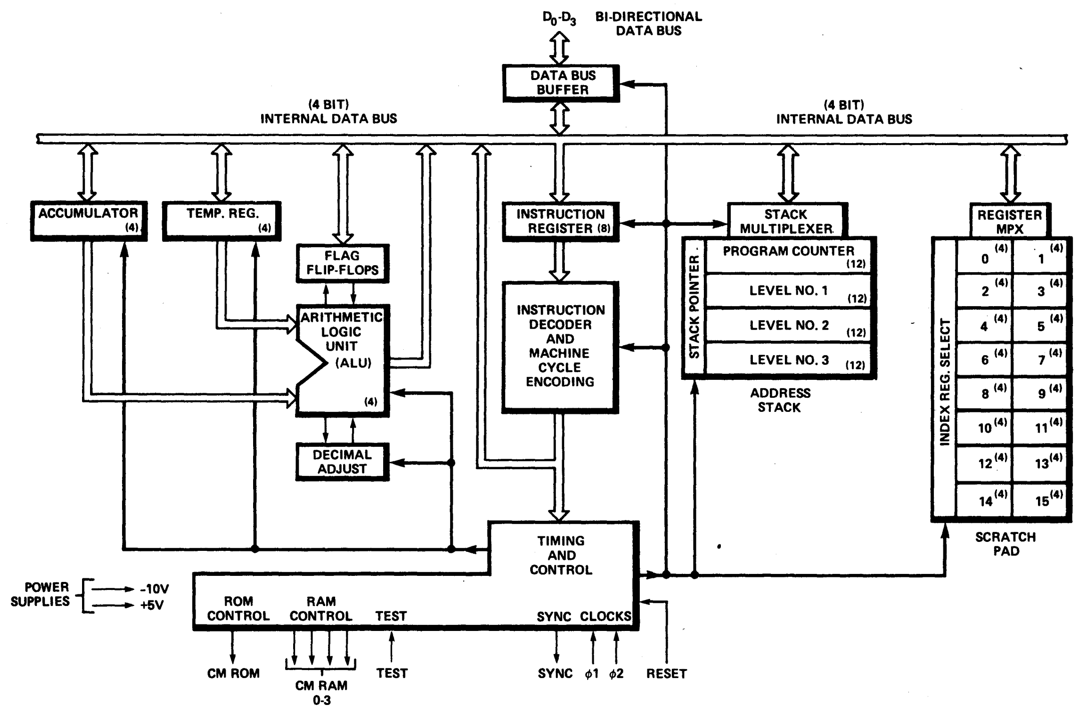


Figure 1-3. 4004 CPU Block Diagram.

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 nibbles. The address is incremented by a 4 bit carry look-ahead circuit (address incrementer) after each 4 bit nibble 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.

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.

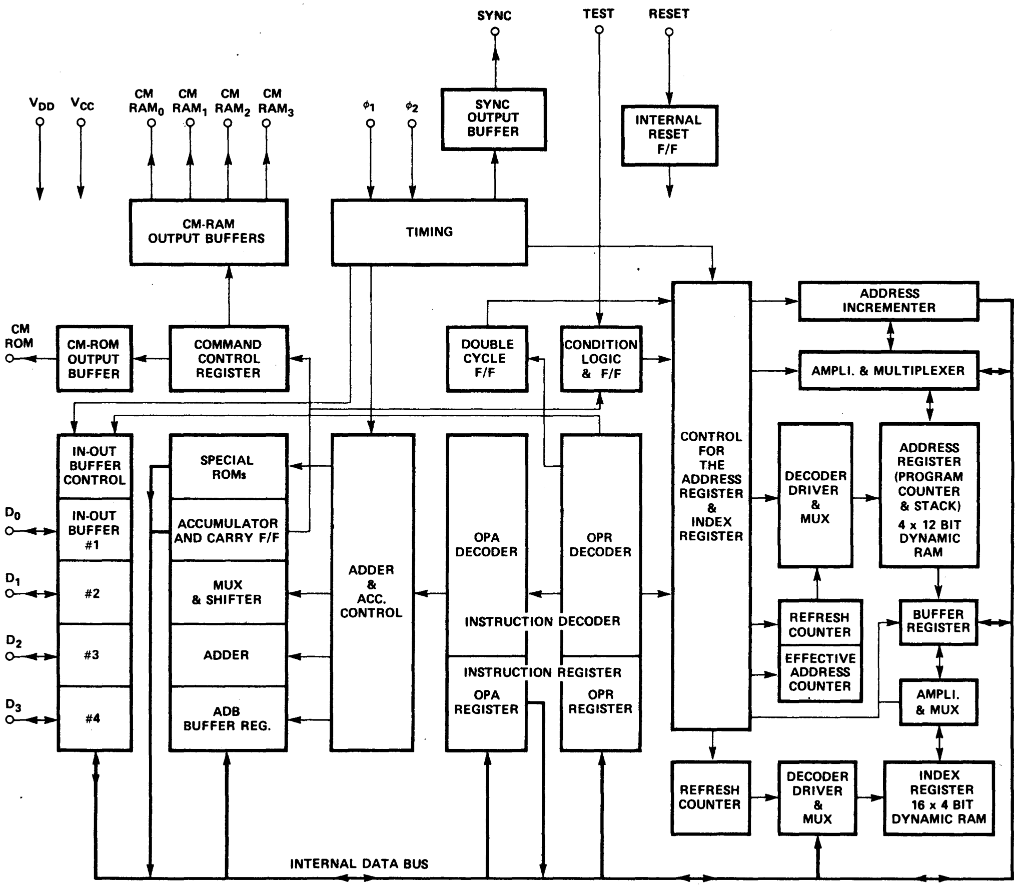


Figure 1-4. 4004 Detailed Block Diagram.

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 length 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-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-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.)

4040 CENTRAL PROCESSOR

Introduction

The 4040 is a single chip 4 bit parallel MOS central processor. It is intended as an enhanced version of the 4004 and as such retains all of the functional capability of that device. It does, however, provide several significant improvements in hardware and software. These are listed briefly here and will be described in detail in the body of this preliminary specification.

Extended Instruction Set

The 4040 software contains all of the 4004 instruction set and includes an additional 14 instructions, providing:

- Halt
- Logical operations
- Interrupt disable, enable functions
- ROM Bank switching
- Index register bank switching

This instruction set is described in detail in Definition of Instruction Set section.

Additional Features

The 4040 contains the necessary hardware to accept and process single level interrupts. The interrupt vectors program to location 003 while saving some key processor conditions.

The address stack has been increased from 4 x 12 bits to 8 x 12 bits, allowing up to seven levels of subroutine nesting.

The index register array has been increased from sixteen 4 bit registers to twenty-four 4 bit registers.

The 4040 is provided with a STOP control which allows the user to halt the processor at an instruction cycle. This feature allows the implementation of a 'single step' operation for program debugging (see STOP/HALT Mode Operation section and Appendix II for detailed descriptions).

The 4040 can address up to 8K x 8 words of ROM with no external logic required. This is implemented by having two 4K x 8 memory banks that can be toggled between.

More Flexible Interface and System Configurations

The 4040 is provided with separate power supply pins for the timing circuitry and for the output buffers. These features allow a low-power standby mode by shutting off the main power supply and operating only the timing. Since the output buffers have a separate supply they can be directly interfaced to other circuit types such as N-channel MOS or CMOS. For single-supply systems all three power supply pins can be tied together.

Hardware Description

The 4040 is packaged in a 24 pin DIP. The pin configuration is shown in the following figure. A brief functional description of each pin is given in the following Pin Description.

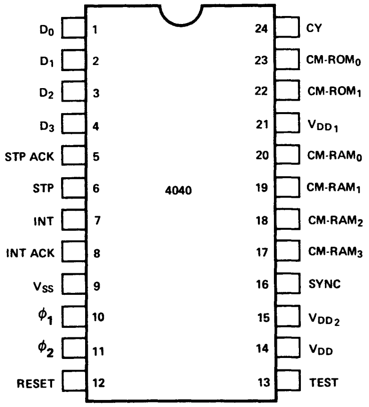


Figure 1-5. 4040 Pin Configuration.

Pin Description

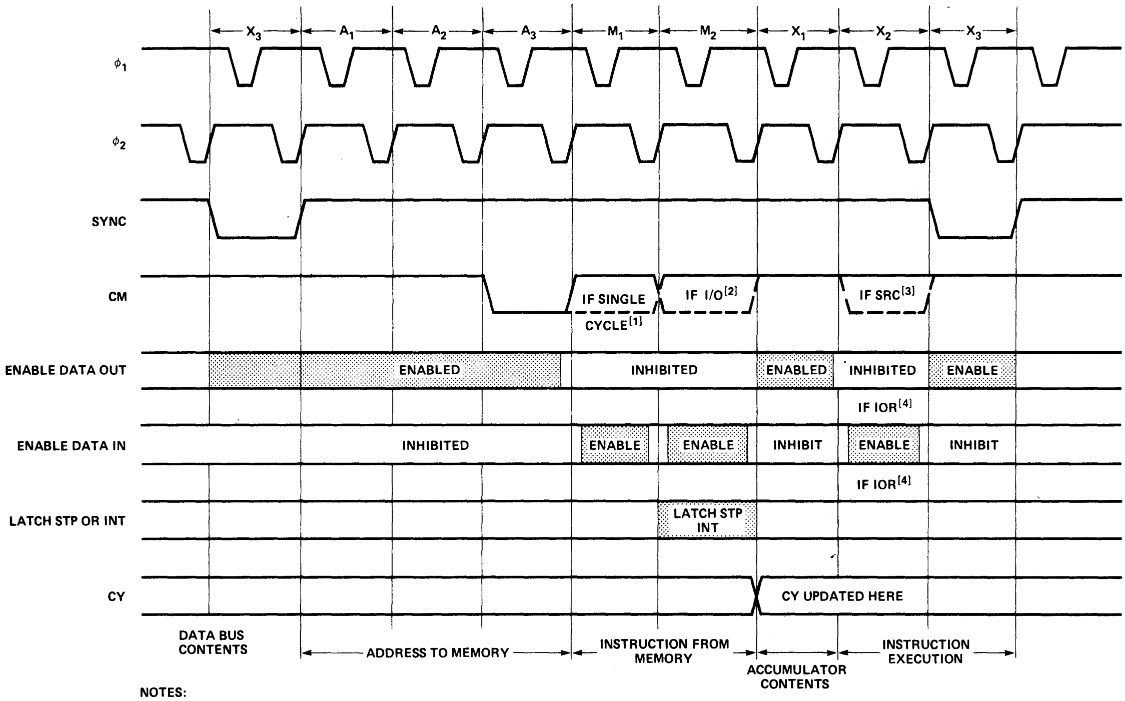
Pin No.	Designation	Description of Function
1-4	D_0-D_3	Bidirectional data bus. All address and data communication between the processor and the RAM and ROM chips is handled by way of these 4 lines.
5	STPA	STOP ACKNOWLEDGE output. This signal acknowledges that the processor has entered the stop mode. Output is "open drain" requiring pull-down resistor to V_{DD}.
6	STP	STOP input signal. A logic "1" level at this input causes the processor to enter the STOP mode.
7	INT	INTERRUPT input signal. A logic "1" level at this input causes the processor to enter the INTERRUPT mode.
8	INTA	INTERRUPT ACKNOWLEDGE output. This signal acknowledges receipt of an INTERRUPT command and prevents additional INTERRUPTs from entering the processor. INTERRUPT ACKNOWLEDGE remains active until cleared by the new BRANCH BACK and SRC (BBS) instruction. The output is "open drain", requiring a pull-down resistor to V_{DD}.
9	V_{SS}	Circuit GND potential – most positive supply voltage.
10-11	ϕ_1-ϕ_2	Non-overlapping clock signals which determine processor timing.
12	RESET	RESET input. A "1" level applied to this pin clears all flag and status flip-flops and forces the program counter to 0. To completely clear all of the address and index registers, RESET must be applied for 96 clock cycles(12 machine cycles).
13	TEST	TEST input. The logical state of this input can be examined with the JCN instruction.
14	V_{DD}	Main supply voltage to the processor. Value must be V_{SS} -15.0V ±5%.
15	V_{DD2}	Supply voltage for output buffers. May be varied depending on interface conditions.
16	$\overline{\text{SYNC}}$	SYNC output. Synchronization signal generated by the processor and sent to ROM and RAM chips. Indicates beginning of instruction cycle.
17-20	CM-RAM_0- CM-RAM_3	CM-RAM outputs. These outputs act as bank select signals for the 4002 RAM chips in the system.
21	V_{DD1}	Supply voltage for timing circuit. Value must be V_{SS} -15.0V ±5%. Allows low power standby operation. Only SYNC will be generated when this pin is the only active V_{DD}.
22-23	CM-ROM_0- CM-ROM_1	CM-ROM outputs. These outputs act as bank select signals for the ROM chips in the system.
24	CY	CARRY output buffer. The state of the CY flip-flop is presented at this output and is updated at X_1. The output is "open drain" requiring a pull-down resistor to V_{DD}.

Basic Circuit Timing

The basic system timing for the 4040 is identical to that used for the 4004, as shown in the following figure. Two non-overlapping clock signals, ϕ_1 and ϕ_2, are used to define the basic timing. The start of an instruction cycle is indicated by the SYNC signal, which is generated by the processor and sent to the various ROM and RAM or peripheral chips in the system. An instruction cycle consists of the following operations:

1. The 12 bit content of the program counter is sent out to the ROM chips in three 4 bit groups during A_1, A_2, A_3.
2. The 8 bit instruction or data from the addressed ROM location is received by the processor at M_1 and M_2 at which time the instruction is decoded.
3. Instruction execution occurs during X_1, X_2, and X_3. Data or address information may be sent to output ports or RAM chips; data may be received from input ports or RAM chips; or data may be operated on within the processor.

The data bus contents at the various times of the instruction cycle are defined just as for the 4004 with the exception of the data at X_1 and the carry output during X_3 of a No-Op instruction. The 4040 outputs the contents of the accumulator at X_1 for program debugging purposes, whereas the 4004 simply copies the data which it received at M_2. The data



NOTES:

1. CM-ROM, RAM SIGNALS WILL BE PRESENT AT M_1 FOR ANY SINGLE CYCLE INSTRUCTION OR FOR THE FIRST CYCLE OF A DOUBLE CYCLE INSTRUCTION.
2. CM-ROM, RAM SIGNALS WILL BE PRESENT AT M_2 FOR ANY OF THE SIXTEEN I/O GROUP INSTRUCTIONS.
3. CM-ROM, RAM SIGNALS WILL BE PRESENT AT X_2 DURING EXECUTION OF AN SRC INSTRUCTION.
4. IOR MEANS ONE OF THE I/O READ INSTRUCTIONS: SBM, ROM, RDR, ADM, RDφ, RD1, RD2, RD3.

Figure 1-6. 4040 Basic Timing Diagram.

bus contents at X_2 and X_3 depend on the instruction being executed; a listing for each individual instruction is contained in the Data Bus Activity section.

A timing feature not present in the 4004 occurs with the generation of the CM-ROM, CM-RAM signals at M_1. This will occur for all single cycle instructions and for the first cycle of all double cycle instructions. This feature allows external logic to distinguish between instruction information and address or data at M_1 and M_2 time.

Basic Description of Major Circuit Blocks

The following figure is a block diagram of the 4040 indicating the major circuit blocks and their interconnections.

The following major functional blocks are contained in the 4040:

1. Address register stack and address incrementer.
2. Index register array.
3. 4 bit adder/accumulator.
4. Instruction register/decoder and control logic.
5. Hardware interrupt and stop control.
6. Peripheral circuits for controlling timing and external communication.

A brief functional description of each of these major elements is given below.

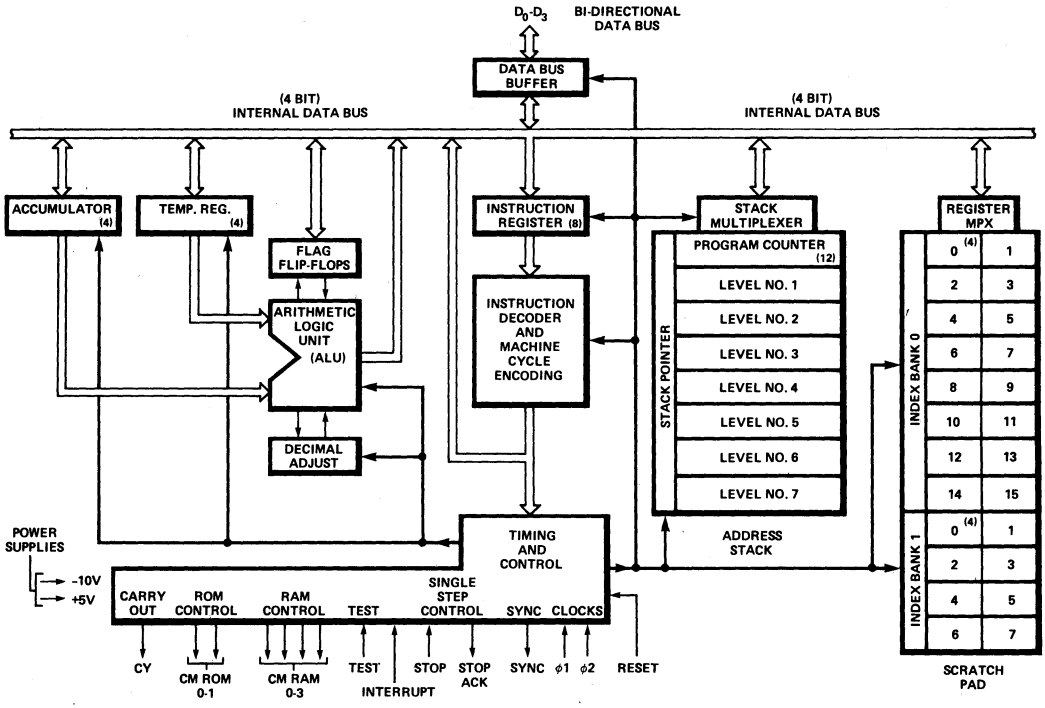


Figure 1-7. 4040 CPU Block Diagram.

Address Register and Address Incrementer

The address register is a dynamic RAM array of 8 x 12 bits operating as a push-down stack. One level of the stack is used to store the effective address, leaving seven levels available for subroutine calls and interrupt processing. The stack address is provided by the effective address counter to the decoder.

The contents of the selected address register are stored in the address buffer and multiplexed to the internal bus during A_1, A_2, and A_3 in 4 bit nibbles. The contents of the address buffer are incremented by a 4 bit carry-look-ahead circuit following the outputting of each 4 bit nibble. The incremented value is transferred back to the address buffer and written back into the selected address register.

Since the array is dynamic, provision is made for refreshing the stored data. A 3 bit refresh counter is multiplexed to the stack decoder for this purpose.

Index Register Array

The index register is a dynamic RAM array of 12 x 8 bits organized as three banks of 4 x 8 bits. Two of the banks have identical address locations and so must be individually selected with the SB0, SB1 instructions. The third bank is always available for use. Refer to the description in the Expanded Index Register Array organization in the next section.

Two modes of operation are possible for the index register array. In one mode the array provides 24 directly addressable 4 bit storage locations for intermediate computation or control purposes. In the second mode the array provides 12 pairs of register locations for addressing RAM, ROM and I/O ports or for storing data fetched from ROM.

Index register addressing is provided by the internal bus for normal read/write operations and by a refresh counter for refresh operation. The addresses are multiplexed to the array decoder.

The content of the selected register is stored in a temporary register and multiplexed to the internal bus. During write operations the internal bus contents are transferred to the temporary register and then to the selected index register.

SRC Register

The SRC register is an 8 bit dynamic latch which stores the contents of the designated index register pair during the execution of the SRC instruction. This 8 bit value is sent to the ROM and RAM chips as an address for any succeeding I/O instruction (see detailed description in Definition of Instruction Set section). The SRC register is used to hold this value in the case that an interrupt should occur, thus allowing the value to be automatically restored when a re-

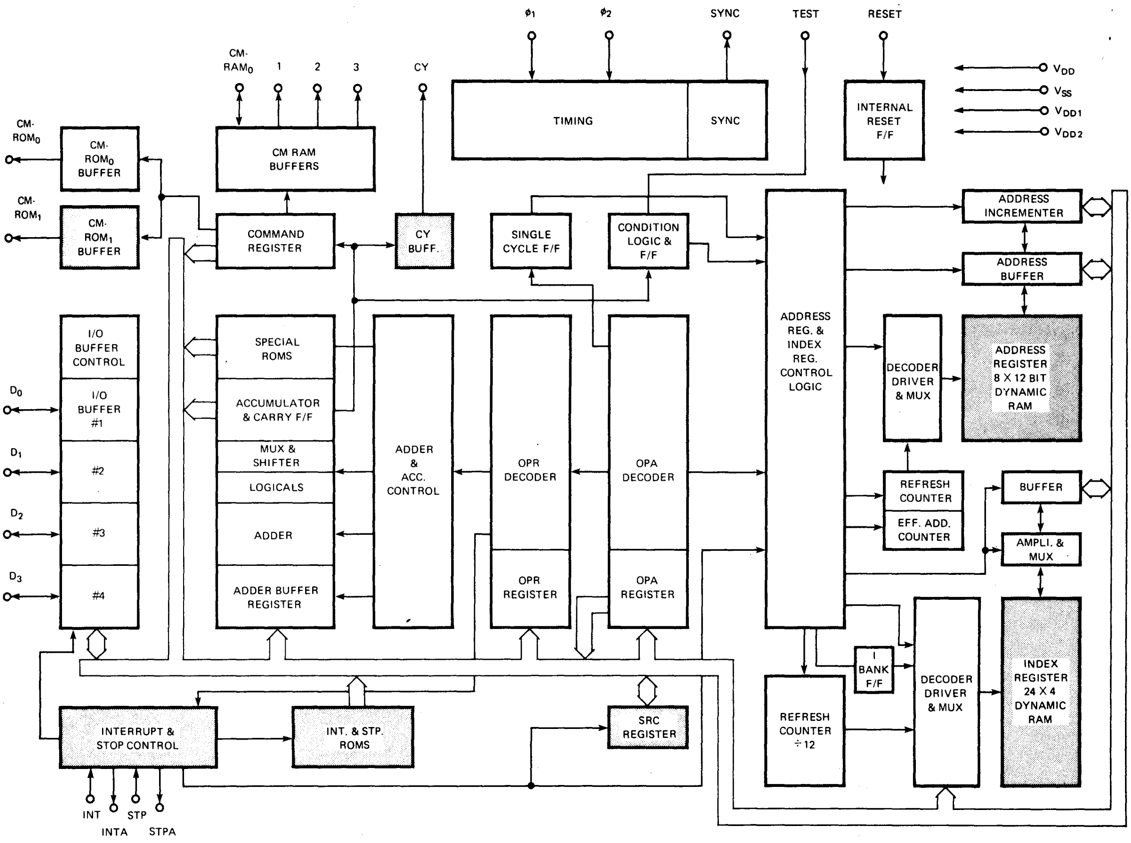


Figure 1-8. Detailed 4040 Block Diagram.

turn from interrupt is made. The SRC register is not loadable during an interrupt routine. However, an SRC may be executed during an Interrupt routine.

4 Bit Adder/Accumulator

The 4 bit adder is of the ripple-through carry type. One term of the addition comes from the "Adder Buffer" register which communicates with the internal bus on one side and can transfer to the adder data or $\overline{\text{data}}$. 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 shift right and shift left instructions. The accumulator communicates also with the command register, with special ROMs, with the condition logic, and with the internal bus. The command register holds a 3 bit code used for CM-RAM line switching and one bit for CM-ROM switching. The special ROMs perform a code conversion for DAA (decimal adjust accumulator) and KBP (process keyboard) instructions. The special ROMs 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.

Instruction Register/Decoder and Control Logic

The instruction register is loaded with the content of the internal bus at M_1 and M_2 during first instruction cycle through a multiplexer and holds the instruction fetched. 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 single cycle FF is reset from one of 5 double-length instructions. Double-length instructions are instructions that need two system cycles (16 clock cycles) for their execution. A condition FF controls JCN and ISZ instruction and is set by the condition logic.

INTERRUPT and STOP Control Logic

The 4040 is provided with hardware INTERRUPT and STOP controls which override the normal processor operation. The INTERRUPT logic detects and acknowledges the presence of an INTERRUPT signal and forces the processor to execute a JMS instruction to location 003. The INTERRUPT MODE section discusses in detail the timing and operation of the INTERRUPT control as well as the additional 4040 features which enhance its use.

The STOP control logic behaves in a similar manner by detecting and acknowledging the presence of a STOP signal. The processor is forced to execute a NOP instruction and will remain in the STOP condition until the STOP signal is removed. See STOP/HALT Mode Operation section for a detailed description of the STOP/HALT operating mode.

Peripheral Circuits

This includes:

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

b. Timing and SYNC generator.

c. 2 CM-ROM and 4 CM-RAM output buffers.

d. POWER-ON-CLEAR flip-flop.

During RESET, all RAMs and static FF's are cleared, and the data bus is set to "0". After RESET, program control will start from "0" and CM-ROM$_0$, CM-RAM$_0$ will be selected. In addition, the INTERRUPT logic will be disabled and INDEX register bank 0 will be selected.

NOTE: Reset will *not* clear the Accumulator in the Stop Mode. The Accumulator can be cleared with a reset in the Run Mode.

4040 Unique Operating Features

The following features will be described in detail:

- STOP/HALT mode logic
- INTERRUPT mode logic
- Extended ROM addressing capability

STOP/HALT Mode Operation

The normal processor cycle of the 4040 may be "stopped" at any point in an instruction sequence by one of two methods. A logic "1" level may be applied to the STOP input pin, in which case the processor will complete the current instruction and then enter the STOP mode. The timing for this operation is shown in the following figure and the sequence of events is outlined below:

a. During instruction cycle #1 the state of the STOP pin is gated into the internal STOP latch at M_2.

b At A_1 of the next single cycle instruction the STOP flip-flop will be set. In the example shown, the processor was executing a double cycle instruction when the STOP signal was first applied. It was allowed to complete the full instruction, hence the STOP flip-flop was not set until A_1 of instruction cycle #3. The buffered output of the STOP flip-flop is used as a STOP ACKNOWLEDGE signal.

c. During instruction cycle #3 and all succeeding instruction cycles, the content of the program counter is sent out at A_1, A_2, and A_3. The program counter is not allowed to increment, however, effectively "stopping" the processor at a given location. In addition the data bus input buffers are prevented from receiving information at M_1 and M_2 times and a NOP instruction is forced on the internal data bus.

d. The processor remains in this NOP loop until the external STOP signal is returned to a logic "0" level during instruction cycle N. The new information is gated into the STOP latch at M_2, allowing the STOP flip-flop to reset at X_3 of the instruction cycle N. Normal processor operation resumes at instruction cycle N + 1.

HALT Mode

Entry to the STOP mode may also be gained through the use of the HALT (HLT) instruction as shown in the following figure. In this case the processor executes the HLT instruction and causes the HALT and STOP flip-flops to be set at X_3. of instruction cycle #1. The processor is forced to execute

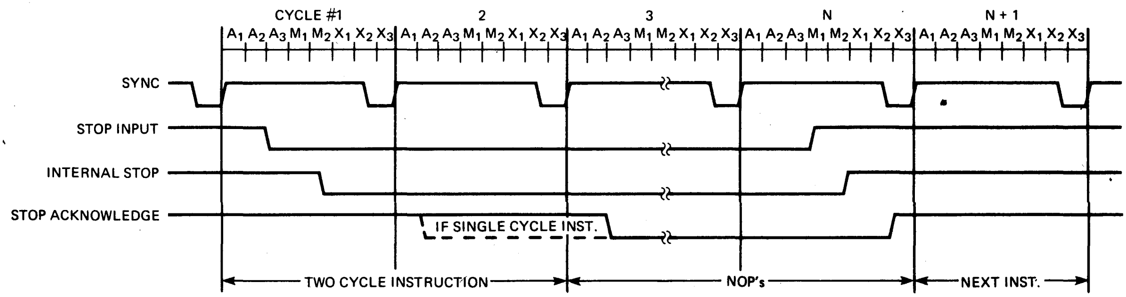


Figure 1-9. Stop Timing.

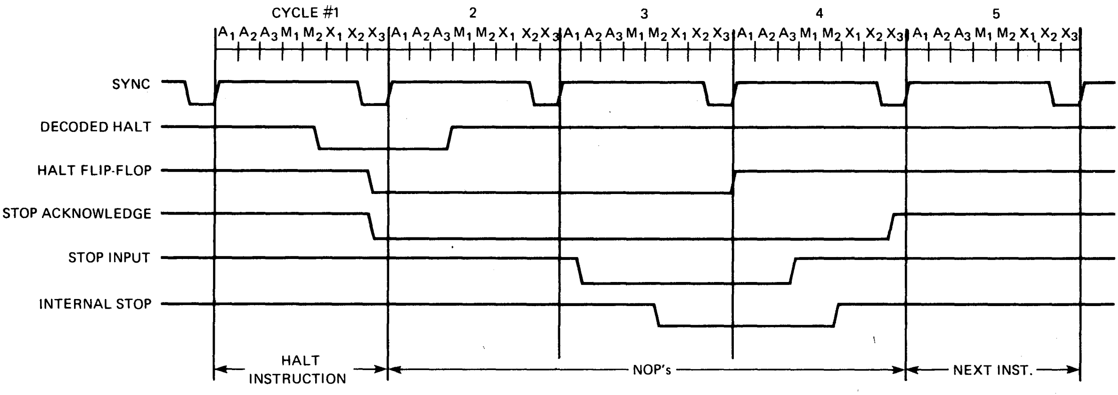


Figure 1-10. Halt Timing (Exit Using Stop Input).

NOP's at instruction cycle #2 and all successive cycle times until removed from the HALT mode.

Exit from the HALT mode can be gained in two ways, one of which is shown in the following figure. At instruction cycle #N, a logic "1" level is applied to the STOP input and is in turn latched by the STOP latch at M_2. A logic "1" in the STOP latch causes the HALT flip-flop to be reset at A_1 of cycle #N + 1. The processor is now in the normal STOP mode and can be released as described in (d) above.

The second means of exiting from the HALT condition is by way of the INTERRUPT input and will be described in the INTERRUPT Mode section.

DATA BUS Contents

The data bus contents during STOP/HALT mode are shown. For program debugging purposes the following information is available:

A_1,A_2,A_3 12 bit address from internal program counter.

M_1,M_2 8 bit instruction from addressed ROM location. Internally the processor executes NOP.

X_1 4 bit contents of ACCUMULATOR.

X_2,X_3 8 bit contents of internal SRC register which stores the value of the last SRC address. CM-ROM and CM-RAM signals are not present at X_2 in this case. (See INTERRUPT Mode section for complete description of operation of SRC register.)

Single Step Operation

The STOP control provides a convenient means of program debugging by allowing a "single step" operation.

INTERRUPT Mode

The 4040 is provided with an asynchronous INTERRUPT input and an INTERRUPT ACKNOWLEDGE output. The following figure presents the basic timing for the INTERRUPT mode. The sequence of events is as follows:

a. During instruction cycle #1 an INTERRUPT occurs and is gated into the INTERRUPT LATCH during M_2.

b. At A_1 of the next single cycle instruction the INTERRUPT flip-flop is set. As in the case of the STOP example, if the processor is executing a double cycle instruction it is allowed to complete it.

c. During instruction cycle #3 the program counter is prevented from incrementing and the data input buffers are inhibited at M_1 and M_2. A JUMP TO SUBROUTINE (JMS) instruction is forced on the internal data bus. The subroutine address is forced to be page 0, location 3. At X_3 the INTERRUPT ACKNOWLEDGE flip-flop is set and its buffered output is available on the INTERRUPT ACKNOWLEDGE pin. The instruction at location 0, 3 begins the interrupt processing routine.

d. The INTERRUPT ACKNOWLEDGE flip-flop remains set until the interrupt has been processed and the new BRANCH BACK and SRC (BBS) instruction has been executed (instruction cycle #N). No new INTERRUPT can be entered while INTERRUPT ACKNOWLEDGE is active. Note that the INTERRUPT signal may be removed after INTERRUPT ACKNOWLEDGE occurs.

Saving and Restoring Processor Status

To have an effective interrupt handling capability the processor must be capable of saving current program and status register values and restoring same when the interrupt pro-

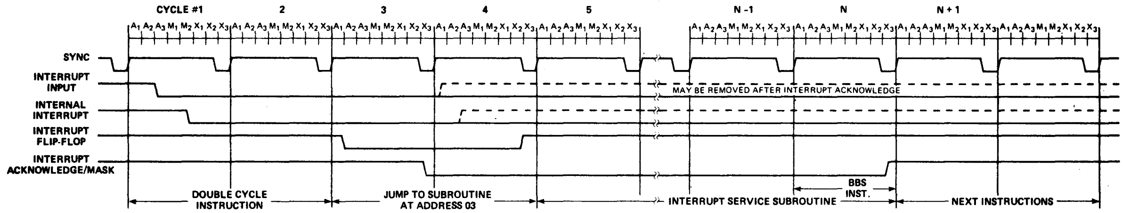


Figure 1-11. Interrupt Timing.

cessing is complete. In the 4040 the following values must be saved:

a. Content of ACCUMULATOR and CARRY flip-flop.
b. Content of COMMAND REGISTER.
c. Content of as many INDEX REGISTERS as required.
d. The value of the last SRC address sent out prior to interrupt.
e. Content of the PROGRAM COUNTER.
f. The current ROM bank (CM-ROM$_0$ + CM-RAM$_1$).

To facilitate the items listed, a number of new hardware features have been included in the 4040 and are described in the following paragraphs.

Expanded Index Register Array

Saving status values requires having temporary storage locations available in the index register array. For this reason the 4040 is provided with 8 additional index registers (4 register pairs) providing a total of 24 4 bit registers (12 8 bit register pairs). The array is organized into three 8 register banks. Bank 0 and Bank 1 are individually selectable by using the SELECT INDEX BANK (SB0, SB1) instructions. The upper bank, which contains registers 8 - 15, is always available for storage. Interrupt Processing section provides examples of interrupt routines and demonstrates the use of the index register banks as well as all other features listed here.

Note that both Bank 0 and Bank 1 contain the same individual address locations. This feature allows those instructions which reference specific register locations to be executed from either index register bank. Thus the JUMP INDIRECT (JIN) instruction and the logical instructions OR4, OR5, AN6, AN7 reference two different sets of registers, e.g., a JIN instruction can reference register pair #0 in Bank 0 or in Bank 1.

The BANK SELECT flip-flop is automatically saved and restored during interrupts. Thus a user may wish to operate in Bank 0 until interrupted then switch to Bank 1. When the BBS instruction is executed to return from interrupt, the previous Bank will automatically be selected for the next instruction.

After application of a RESET signal, Bank 0 will be selected.

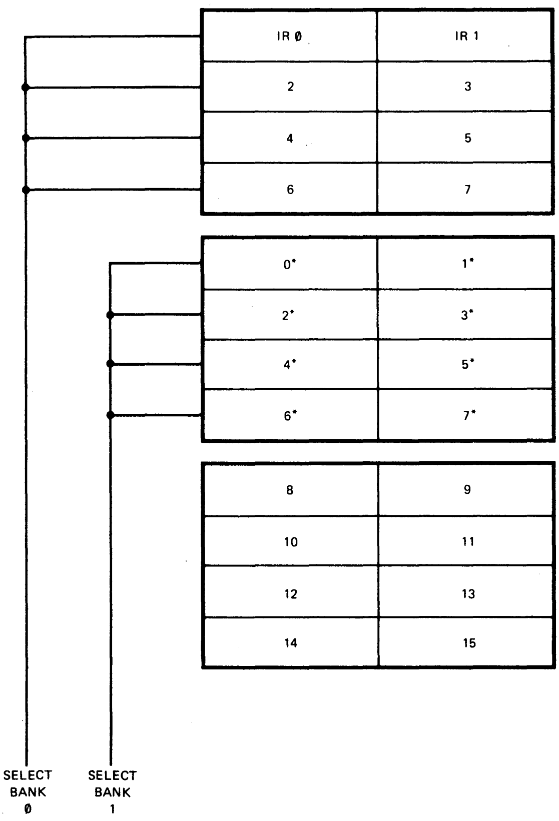


Figure 1-12. Index Register Organization.

SRC Register

When the 4040 executes an SRC instruction, the 4 bit values sent out at X_2 and X_3 are stored internally in the 8 bit SRC register. The SRC register is locked out during the interrupt routine by the INTERRUPT ACKNOWLEDGE flip-flop. Thus, any SRC instruction executed during an interrupt routine will not affect the value in the SRC register. The last instruction in the interrupt routine must be a BRANCH BACK and SRC (BBS), another new instruction which restores the program counter (see below) and sends out the contents of the SRC register at X_2 and X_3, and generates the appropriate CM-ROM and CM-RAM at X_2. This restores the ROM and RAM select logic to their pre-interrupt conditions.

Extended Address Register Stack

The address stack of the 4040 is an 8 x 12 bit array (compared to 4 x 12 bits in the 4004). One level of the stack is required for the program counter; hence seven levels of subroutines may be nested using the 4040.

When an interrupt occurs the program counter is not incremented, but is stored down one level in the stack by the execution of the forced JMS instruction. This value is restored by the execution of the BBS instruction at the end of the interrupt routine.

General Information Applying to Interrupt

The 4040 is capable of servicing one interrupt at a time. The INTERRUPT ACKNOWLEDGE signal is used internally to prevent a second interrupt from being entered until the first is completely serviced.

Two instructions, INTERRUPT ENABLE (EIN) and INTERRUPT DISABLE (DIN) are provided for protecting sequences of instructions from being interrupted. These are described in detail in the Definition of Instruction Set section.

The RESET signal disables interrupt. If the processor is started from a RESET condition, an EIN instruction must be performed before an interrupt will be recognized.

If an INTERRUPT and STOP signal occur such that they are both latched at M_2 of the same instruction cycle, the STOP logic will have priority, and must be cleared before the interrupt can be recognized.

As mentioned in the DATA BUS Contents section, the INTERRUPT control may be used to exit from a HALT condition. The timing for this is shown. The processor enters the HALT mode at instruction cycle #1 and remains in that mode until the INTERRUPT signal is recognized at instruction cycle #4. When the INTERRUPT flip-flop is set, it causes the HALT flip-flop to be reset. The processor is then in the INTERRUPT mode. In this way a processor could be used in a completely asynchronous control application in which the INTERRUPT signal would begin the processing routine. When the routine was complete the processor would execute the HLT instruction and wait for a new INTERRUPT.

Extended ROM Addressing Capability

The 4040 is equipped with two CM-ROM output buffers, each of which can be used to select a bank of sixteen 256 x 8 ROMs. A total of 8K x 8 bit words can be directly addressed. Bank switching is accomplished through the use of two new instructions, DESIGNATE BANK 0 (DB0) and DESIGNATE BANK 1 (DB1). Both of these instructions take effect on the third cycle following their execution. The Use DB0 & DB1 section provides example uses of the DB0, 1 instructions.

Since the INTERRUPT control logic will force a JMS to page 0, location 3, the first few instructions of the interrupt routine will have to be duplicated in both ROM banks (see Interrupt Processing section).

The fact that the bank switching operation requires three instruction cycles to be completed means that an INTERRUPT, Stop and Halt cannot occur during those three cycles. For this reason the INTERRUPT Stop and Halt logic is internally disabled during the execution of DB0 or DB1.

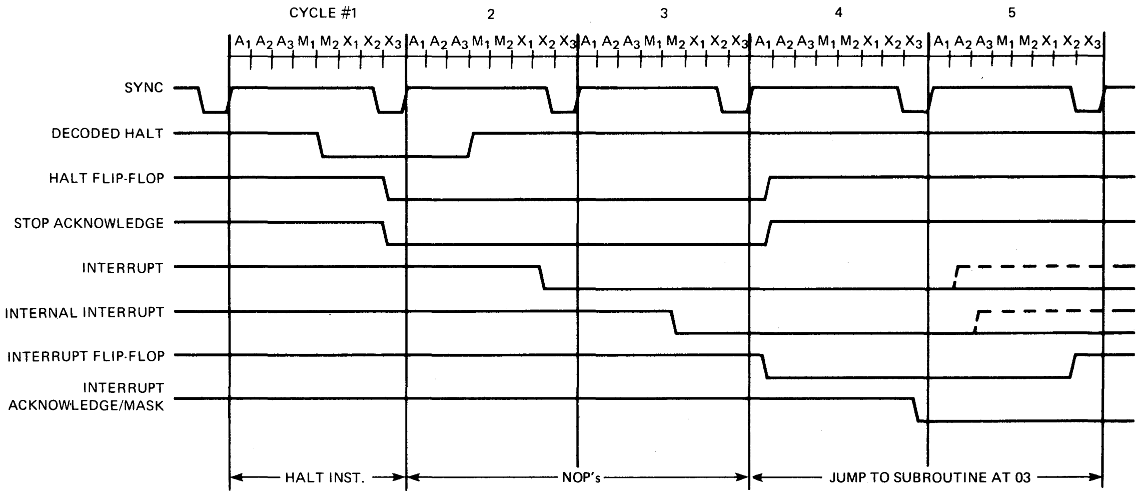


Figure 1-13. Halt Timing (Exit Using Interrupt).

Use of Designate ROM Bank Instructions

The DB instructions present a convenient method of switching from one ROM bank to another. As shown in the following examples, the bank switch is delayed until the 3rd instruction cycle after the DB is executed.

Example #1:

ROM Location*			Instruction	Comment
Bank /	Page /	Word		
0	2	107	X X X	
0	2	108	DB1	Designate Bank 1.
0	2	109	JUN 1	
0	2	110	27	During this instruction cycle a "1" is loaded in bit #3 of the command register.
1	1	27	X X X	JUN occurred to Bank 1 because CM-ROM$_1$ has been activated.
.			.	
.			.	
.			.	
1	1	63	DB0	Designate Bank 0.
1	1	64	ISZ 3	
1	1	65	151	
0	1	66	X X X	Program jumps here if (IR$_3$) = 0.
0	1	151	X X X	Program jumps here if (IR$_3$) ≠ 0.

Example #2

ROM Location			Instruction	Comment
Bank /	Page /	Word		
0	7	131	X X X	
0	7	132	DB1	Designate Bank 1.
0	7	133	JMS 2	
0	7	134	96	Address 7, 135 saved in stack.
1	2	96	X X X	JMS occurs to 1, 2, 96 since CM-ROM, is activated at this instruction cycle.
.		.	.	
.		.	.	
.		.	.	
1	2	170	DB0	Designate Bank 0.
1	2	171	XCH 7	
1	2	172	BBL	Address 7, 135 pulled from stack and placed in PC; branch back occurs to 7, 135 in Bank 0 because CM-ROM$_0$ is activated during this instruction cycle.
0	7	135	X X X	

* Bank #0, 1   Page #0 - 15   Word #0 - 255

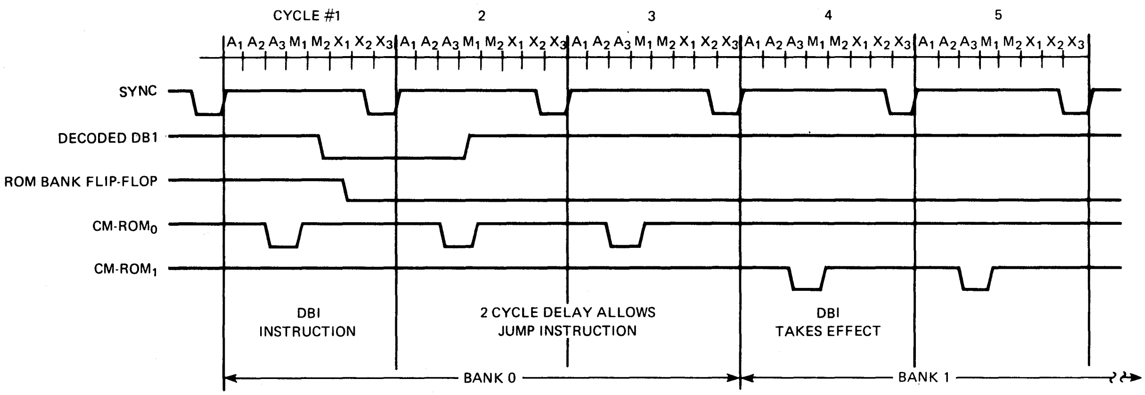


Figure 1-14. ROM Bank Switching.

EXTENDED BANK SWITCHING

The following describes how additional ROM banks can be added to the 4004 or how to further expand the capability of the 4040 to greater than two ROM banks.

When the ROM bank is to be switched, the RAM output port has one line either set or reset. The drawing assumes SET (logical "1", MCS-40 T.M. definition) to select BANK 1 and RESET (logical "0") to select BANK 0. The normal condition shall be BANK 0 selected.

The RAM port output basically steers the CM-ROM line to one of two separated output lines labeled CM-ROM BANK 0 and CM-ROM BANK 1. While the logic shown is a TTL implementation, an equivalent circuit at MOS levels may be built.

The circuit will functionally switch banks under software control by utilizing the WMP command for writing accumulator contents to the RAM port (A more detailed description of the programming sequence will be presented later). The next instruction after the WMP will execute from the bank selected. The bank selection remains in effect until another WMP for that port is issued (port value changed). Any subsequent instructions execute from the newly selected bank. The operation is similar to the DCL command for RAM BANK selection.

Because the program counter contents are incremented, execution of a command after the bank switch begins at PC + 1 but from the new ROM bank, not the ROM bank which issued the switch command.

I.E.

BANK	PC	PROGRAM
0	00	.
0	01	.
0	02	WMP ← SWITCH HERE
1	03	.
1	04	.
1	05	.
1	06	.
1	07	WMP ← SWITCH BACK
0	08	.
0	09	.
0	0A	.
0	0B	.

To avoid unnecessary gaps in either bank, an established switch point should be made. Since it is assumed that more than 16 ROM's are necessary, then BANK 0 ROM 15 location FF is always available and that BANK 1 ROM 0 location FF likewise is always available. More precisely, location FF of the last used ROM of BANK 1 is always available. Therefore, two "natural" switch points for bank switching exist. These "natural" points are easily seen if we assume 16 ROMs in each bank. Then a programming sequence might look like. . .

	BANK	PC	PROGRAM
	0	FFB	.
	0	FFC	.
	0	FFD	.
	0	FFE	.
Last byte, last ROM BANK 0	0	FFF	WMP ← SWITCH HERE
	1	000	.
	1	001	.
	1	002	.
	1	003	.
	1		.
	1	etc.	.
	1		.
	1	FFC	.
	1	FFD	.
	1	FFE	.
Last byte, last ROM BANK 1	1	FFF	WMP ← SWITCH BACK
	0	000	.
	0	001	.

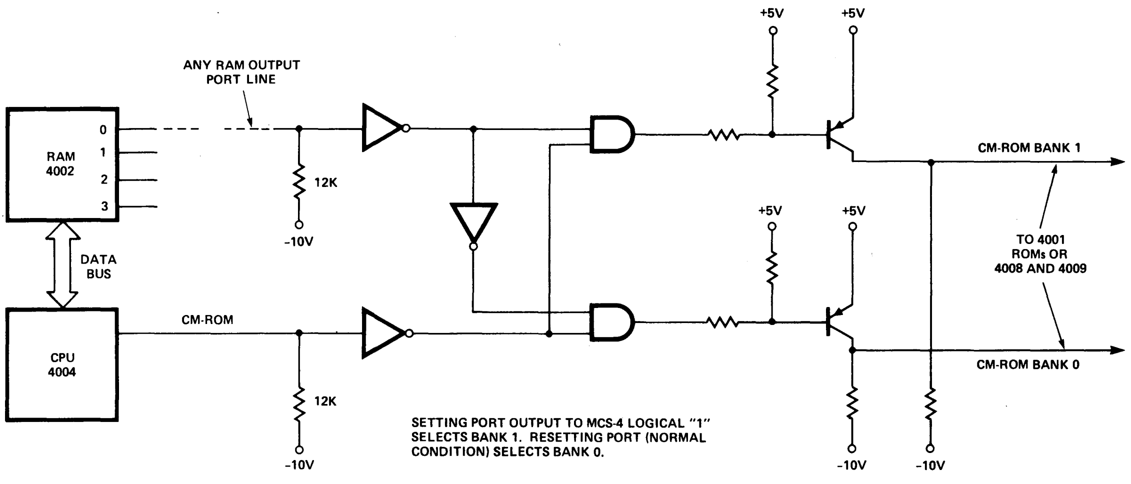

ANY RAM OUTPUT PORT LINE
RAM 4002
0
1
2
3
DATA BUS
CPU 4004
CM-ROM
12K
-10V
12K
-10V
+5V
+5V
+5V
+5V
-10V
-10V
CM-ROM BANK 1
CM-ROM BANK 0
TO 4001 ROMs OR 4008 AND 4009
SETTING PORT OUTPUT TO MCS-4 LOGICAL "1" SELECTS BANK 1. RESETTING PORT (NORMAL CONDITION) SELECTS BANK 0.

Figure 1-15. ROM Bank Selection Logic.

This is an "end around" switch and the beginning locations in ROM bank 0 must be reserved to "recognize" the switch back to bank 0 from bank 2. The recognition may be as simple as a jump back to the program beginning in BANK 0. For example:

```
          BANK    PC     PROGRAM
----------------------------------------------------
            1      :        :
            1     FFF      WMP ←──── SWITCH HERE
            0     000      JUN       START
            0     010                (address of
                                      START)
```

It may also cause a jump to any address within BANK 0.

If less than 16 ROM's are used in Bank 1, then as an example:

```
           BANK    PC     PROGRAM
----------------------------------------------------
             0     FFD       :
             0     FFE       :
             0     FFF      WMP ←──── SWITCH HERE
             1     000       .
             1     001       .
             1     002       .
             1     6FC       :
             1     6FD       :
Last byte,   1     6FE       :
last ROM     1     6FF      WMP ←──── SWITCH BACK
(only 0-6
used in      0     700      JUN       START
Bank 1)      0     010   (            (address of
             0     011                 START)
```

Here, the first locations of ROM 7 BANK 0 must be reserved to "recognize" the switch back and to get to the correct location within Bank 0.

The basic programming steps for the switch are as follows:

```
FFB SWITCH,      FIM 0, 0     ;RAM PORT 0 selected
FFD              SRC 0
FFE            * LDM 1        ;Load 0001 to ACC
FFF    ←──────── WMP
```

* LDM 1 for BANK 1, LDM 0 for BANK 0

The SWITCH may be called from any point *within* the ROM bank in which that SWITCH is located. For example:

```
BANK  ADDRESS              PROGRAM EXECUTION
------------------------------------------------
 0     7A2                        :
 0     7A3                        :
 0     7A4                    JUN SWITCH
 0     FFB      SWITCH        FIM OP 0
 0     FFD                    SRC OP
 0     FFE                  * LDM 1
 0     FFF   ←─────────────── WMP
 1     000                        :
 :      :                         :
 :      :                         :
```

The switching instructions as given can also be written as a subroutine with a BBL at the end. The program will switch to BANK 1 (for example) and execute from the address just following the address of the JMS (in Bank 0, for example) which called the switch.

When going from bank 0 to bank 1, the next instruction can be a resumption of program flow just prior to the switch, if the normal program would "overflow" into BANK 1. When going from BANK 1 to BANK 0, a determination must be made of where the program should resume executing from.

The first three instructions (FIM, SRC, LDM) can actually be anywhere in the program within one bank, and a JUN FFF will cause the program to switch at the end of ROM if the instruction at FF is WMP.

If the RAM port which is used for ROM Bank Selection is not used for anything else, the remaining three lines may be used to indicate where the program should go to within the newly selected bank. This is done by testing the accumulator contents *after* the switch. For example:

```
BANK  ADDRESS              PROGRAM EXECUTION
------------------------------------------------------
        :
 0     FFB   Switch        FIM 0, 0
 0     FFD                 SRC 0
 0     FFE              ** LDM N
 0     FFF                 WMP
 1     000   ←──────────── CLC
 1     001                 RAL
 1     002                 JC   SOMEWHERE     ;Simple
        :                  RAL                ;3 line
 :      :                  JC   SOMEPLACE     ;test &
 :      :                  RAL                ;branch
 :      :                  JC   ELSEWHERE
```

** LDM N where
N=DDDb

DDD=one of three branch conditions
B=Bank select code

Or, for more branch conditions, test the three DDD bits for 1 of 8 branch codes.

General Notes:

1. The use of the RAM port is advisable since addressing that port is possible from either bank, while ROM ports can only be addressed within the bank with which they are associated. It follows that the ROM port structures associated with bank 0 can be duplicated for bank 1 and used for entirely different purposes. This general scheme (RAM port addressing) can also be used as a means of ROM input and output port expansion for systems using the 4289.

2. It is advisable to keep all calls to subroutines within one bank. When switching ROM banks, no subroutine calls should be in effect, i.e., the stack for return addresses is empty. Pseudo-calls to subroutines can be made between banks by using JUN commands to the switch points in each bank and by testing the accumulator to determine which subroutine to go to. The return is a BBL to another

JUN command which switches you back to the previous bank. The use of the JMS and BBL after the switch has been made still leaves the stack empty when the return switch is made, and conforms with normal programming procedures involving subroutines.

3. The RAM port used should be one controlled by CM-RAM 0 since this group of RAM's is automatically selected after power on and reset, thereby assuring that the program begins in ROM bank 0.

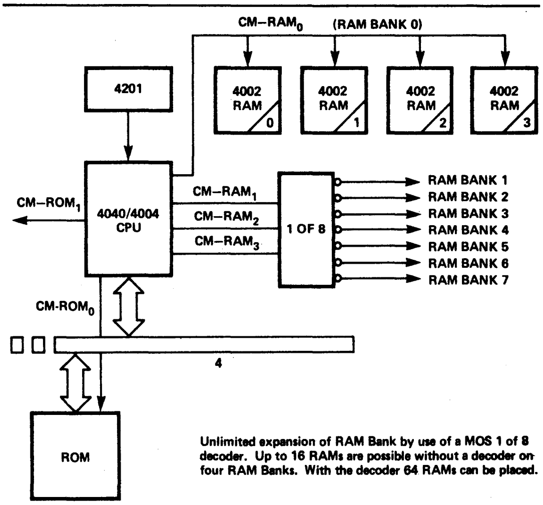
Interrupt Processing

An interrupt processing routine is, in general, composed of three parts:

a. The instructions required to save the current processor status.

b. A portion which determines and services the interrupting device.

c. The instructions required to restore program control to the pre-interrupt conditions.



Unlimited expansion of RAM Bank by use of a MOS 1 of 8 decoder. Up to 16 RAMs are possible without a decoder on four RAM Banks. With the decoder 64 RAMs can be placed.

Figure 1-16. CM-RAM Expansion.

In the first example, the processor is used with a single ROM bank, and Index Register (IR) Bank 1 is used to save status (accumulator/carry, Command Register (CR)). The six remaining registers in IR Bank 1 are available for interrupt servicing. In addition to being relatively simple, this scheme has the advantage of saving processor status with the fewest number of instructions. Note that since only one ROM bank is available, it is only necessary to save the lower three bits of CR. This allows saving the CR and CY to be merged in the same register location.

Example #3:

ROM Location			Instruction	Comment
Bank /	Page /	Word		
0	6	82	SRC 4	(IR 8,9) sent to ROM & RAM, Load SRC Reg.
0	6	83	INC 9	Interrupt occurs here.
0	6	84	(JMS 0)	Interrupt acknowledged, 6,84 saved in stack; instruction at 6,84 ignored.
0	6	84	(3)	
0	0	3	SB1	Select IR Bank 1.
0	0	4	XCH 7	(ACC) → IR7* – ACC saved.
0	0	5	LCR	(CR) → ACC
0	0	6	RAL	(CY) → Acc_0, Acc_0 → Acc_1 . . . Acc_3 → CY
0	0	7	XCH 6	(ACC) → IR6* CY, CR saved.
0	0	8		
0	0	9	.	Routine for determining and servicing interrupt is executed here.
	.		.	
	.		.	
	.		.	
0	P	n	XCH 6	(IR6*) → ACC
		n+1	RAR	ACC_0 → CY – CY restored
		n+2	DCL	ACC_0 → CR_0, ACC_1 → CR_1, ACC_2 → CR_2, CR restored.
		n+3	XCH 7	(IR7*) → ACC
		n+4	BBS	Address 6,84 loaded into PC; contents of SRC register sent out;
0	6	84	WRM	program restored.
			.	
			.	
			.	

*Index Register Bank is automatically restored with BBS.

INSTRUCTION SET

The 4040 is functionally compatible with the 4004 and therefore recognizes all 46 instructions valid for the 4004. In addition, the 4040 recognizes 14 new instructions giving a total of 60 instructions in the set. The instruction format is, of course, identical to that used in the 4004.

Four groups of instructions can be defined as follows:

a. Machine Instructions – This group of 16 instructions are designated by an OPR code of 0000 – 1101. Within this group is contained a second group which is designated supplemental group.

b. 4040 Group – This group of 14 instructions is designated by an OPR code of 0000 and an OPA code of 0001 – 1110. These are the new instructions which have been added to the 4040.

c. I/O Group – Designated by an OPR code of 1110, this group of 16 instructions is used for transferring data between the processor and the RAM chips or I/O circuits.

d. Accumulator Group – This group of 14 instructions is designated by an OPR code of 1111 and operates only on the accumulator/carry flip-flop, the special ROMs and the command register.

Instruction Set Format

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 nibbles. 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 first 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

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.

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.

Index Register Organization

The index register can be addressed in two modes at any one time:

a. By specifying 1 out of 16 possible locations with an OPA code of the form RRRR(1). The bank switch of the 4040 can access an additional 8.

b. By specifying 1 out of 8 pairs (12 pairs on 4040) with an OPA code of the form RRRX (2)

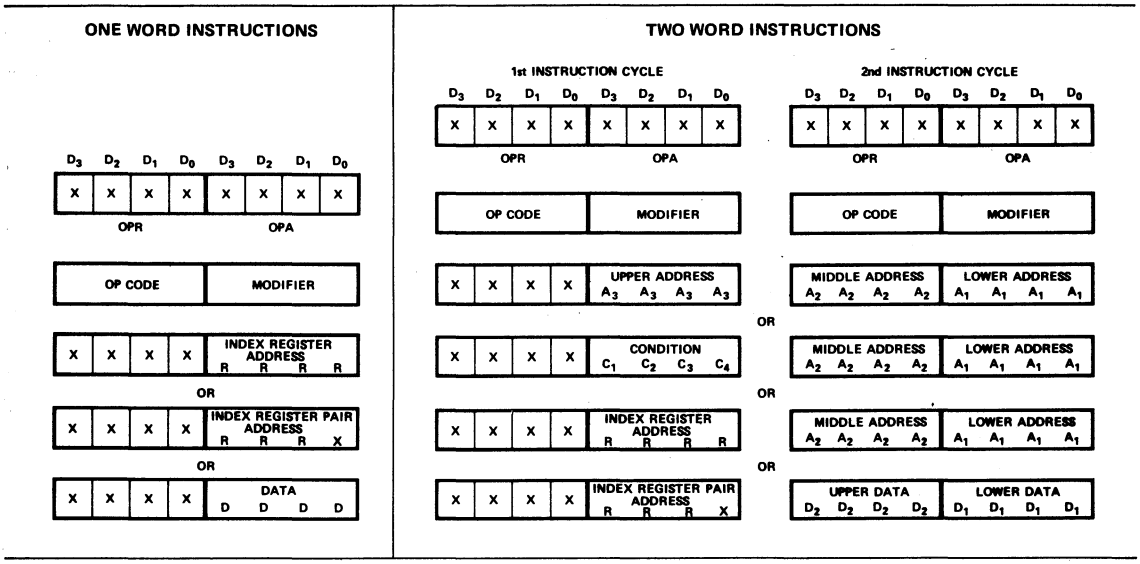


Figure 1-17. Machine Instruction Format.

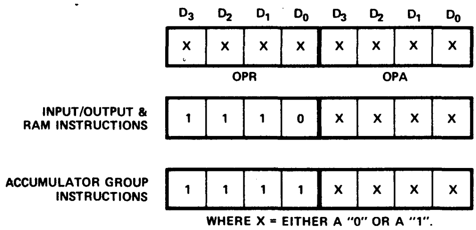

D3 D2 D1 D0 D3 D2 D1 D0
X X X X X X X X
OPR
OPA
INPUT/OUTPUT & RAM INSTRUCTIONS
1 1 1 0 X X X X
ACCUMULATOR GROUP INSTRUCTIONS
1 1 1 1 X X X X
WHERE X = EITHER A "0" OR A "1".

Figure 1-18. I/O and Accumulator Group Instruction Formats.

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.

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 three 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

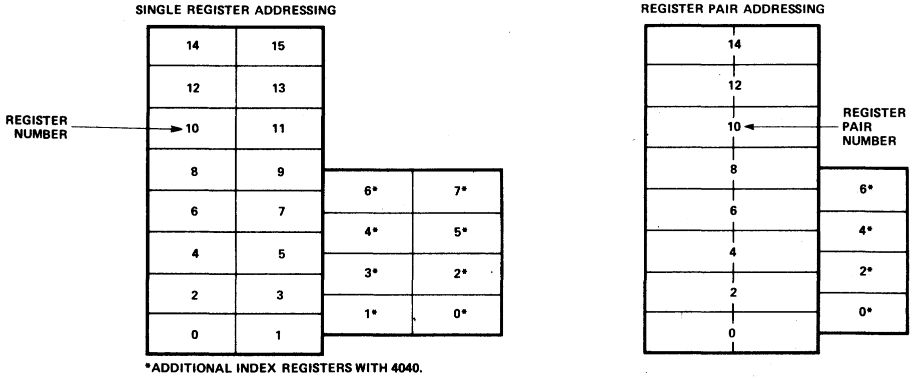


Figure 1-19. Index Register Organization.

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. The 4040 CPU can in addition to executing JMS/BBL combination to get to and from a subroutine, can also execute an Interrupt/BBS. In this case, the interrupt forces the program to location 003. The BBS returns the program counter to its previous value plus 1 and restores the SRC.

1. In this case of JMS the instruction is executed on the 4 bit content addressed by RRRR.
2. In this case the instruction is executed on the 8 bit content addressed by RRRX, where X is specified for each instruction.
3. Since the JMS instruction is a 2-word instruction the old effective address is incremented by 2 to correctly give the address of the next instruction to be executed after the return from JMS.

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 7 registers in the push down stack of the 4040, 7 return addresses may be saved. If an eighth JMS occurs, the deepest return address (the first one stored) is lost.

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 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.

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]

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,. . . .)

Note 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.

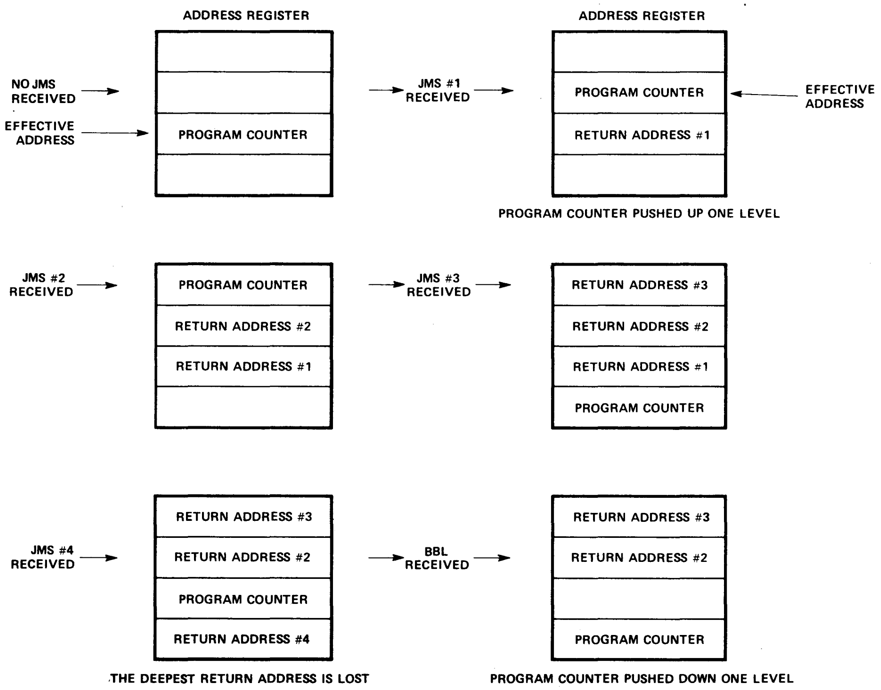


Figure 1-20. Operation of the Address Register on a Jump to Subroutine Instruction for 4004.

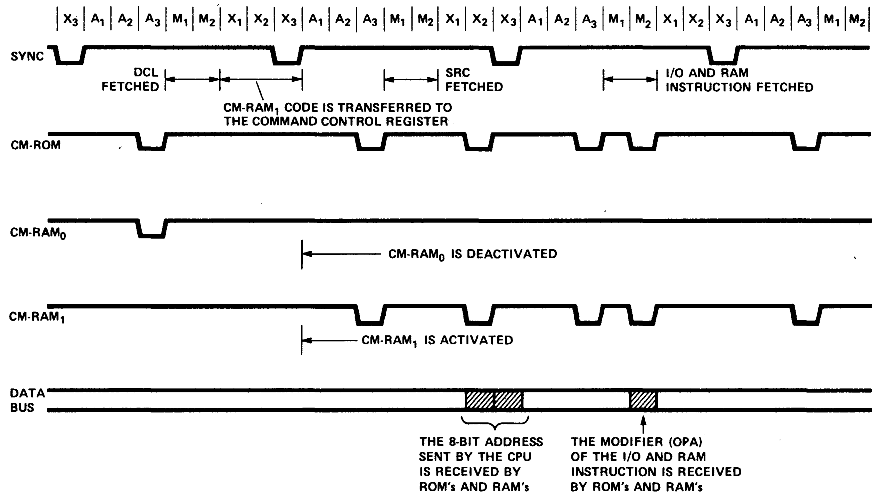

X3 A1 A2 A3 M1 M2 X1 X2 X3 A1 A2 A3 M1 M2 X1 X2 X3 A1 A2 A3 M1 M2 X1 X2 X3 A1 A2 A3 M1 M2
SYNC
DCL FETCHED
SRC FETCHED
I/O AND RAM INSTRUCTION FETCHED
CM-RAM1 CODE IS TRANSFERRED TO THE COMMAND CONTROL REGISTER
CM-ROM
CM-RAM0
CM-RAM0 IS DEACTIVATED
CM-RAM1
CM-RAM1 IS ACTIVATED
DATA BUS
THE 8-BIT ADDRESS SENT BY THE CPU IS RECEIVED BY ROM's AND RAM's
THE MODIFIER (OPA) OF THE I/O AND RAM INSTRUCTION IS RECEIVED BY ROM's AND RAM's

Figure 1-21. Operation of the Command Control Lines.

Following is a detailed explanation of each step.

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

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

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

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

BASIC INSTRUCTION SET

The basic instruction set of the 4040 and 4004 (CPU) are shown below. The following section 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.

NEW 4040 INSTRUCTIONS

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
HLT	0 0 0 0	0 0 0 1	Halt – inhibit program counter and data buffers.
BBS	0 0 0 0	0 0 1 0	Branch Back from Interrupt and restore the previous SRC. The Program Counter and send register control are restored to their pre-interrupt value.
LCR	0 0 0 0	0 0 1 1	The contents of the COMMAND REGISTER are transferred to the ACCUMULATOR.
OR4	0 0 0 0	0 1 0 0	The 4 bit contents of register #4 are logically "OR-ed" with the ACCUM.
OR5	0 0 0 0	0 1 0 1	The 4 bit contents of index register #5 are logically "OR-ed" with the ACCUMULATOR.
AN6	0 0 0 0	0 1 1 0	The 4 bit contents of index register #6 are logically "AND-ed" with the ACCUMULATOR
AN7	0 0 0 0	0 1 1 1	The 4 bit contents of index register #7 are logically "AND-ed" with the ACCUMULATOR.
DB0	0 0 0 0	1 0 0 0	DESIGNATE ROM BANK 0. CM-ROM_0 becomes enabled.
DB1	0 0 0 0	1 0 0 1	DESIGNATE ROM BANK 1. CM-ROM_1 becomes enabled.
SB0	0 0 0 0	1 0 1 0	SELECT INDEX REGISTER BANK 0. The index registers 0 - 7.
SB1	0 0 0 0	1 0 1 1	SELECT INDEX REGISTER BANK 1. The index registers 0* - 7*.
EIN	0 0 0 0	1 1 0 0	ENABLE INTERRUPT.
DIN	0 0 0 0	1 1 0 1	DISABLE INTERRUPT.
RPM	0 0 0 0	1 1 1 0	READ PROGRAM MEMORY.

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.
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).

DETAILED INSTRUCTION DESCRIPTION

A. Symbols and Abbreviations

The following symbols and abbreviations will be used throughout the next few sections:

SRCR	SRC Register
( )	the content of is transferred to
ACC	Accumulator (4 bit)
CY	Carry Flip-Flop
ACBR	Accumulator Buffer Register (4 bit)
RRRR	Index register address
RRR	Index register pair address
P_L	Low order program counter Field (4 bit)
P_M	Middle order program counter Field (4 bit)
P_H	High order program counter Field (4 bit)
a_i	Order i content of the accumulator
CM_i	Order i content of the command register
M	RAM main character location
M_{si}	RAM status character i
DB (T)	Data bus content at time T
Stack	The 3 or 7 registers in the address register other than the program counter.
CR	Command register
IE	Interrupt enable
RB0	Register bank 0 RRRR$_0$ – RRRR, enable
RB1	Register bank 1 RRRR$_0$ – RRRR, enable
V	Logical OR
Λ	Logical AND

Throughout the text "page" means a block of 256 instructions whose address differs only on the most significant 4 bits.

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

B. 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)

C. One Word Machine Instructions

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

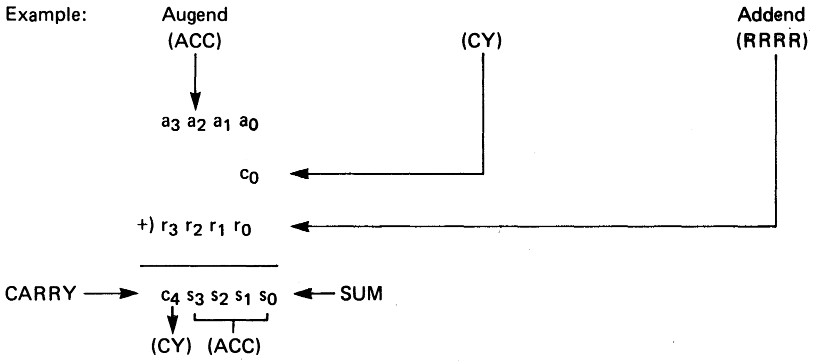
Mnemonic:	LDM (Load Data to Accumulator)
OPR OPA:	1101 DDDD
Symbolic:	DDDD → ACC
Description:	The 4 bits of data, DDDD stored in the OPA field of instruction word are loaded into the accumulator. The previous contents of the accumulator are lost. The carry/link bit is unaffected.

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.

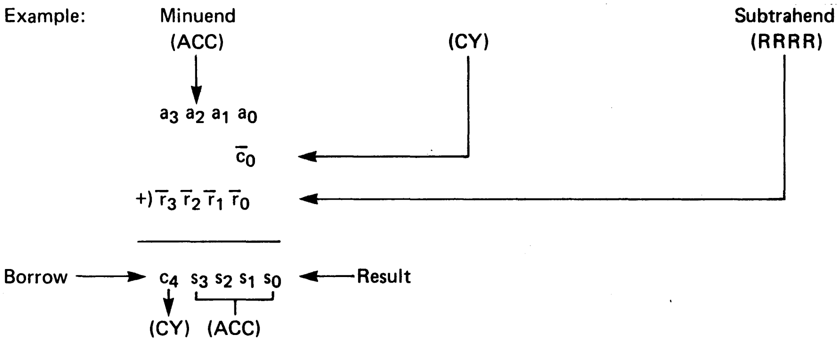
Mnemonic: ADD (Add index register to accumulator with carry)
OPR OPA: 1000 RRRR
Symbolic: (RRRR) + (ACC) + (CY) → ACC, CY
Description: The 4 bit content of the designated index register is added to the content of the accumulator with carry. The result is stored in the accumulator. The carry/link is set 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) + (CY) → ACC, CY
Description: The 4 bit content of the designated index register is complemented (ones complement) and added to content of the accumulator with borrow and the result is stored in the accumulator. If a borrow is generated, the carry bit is set to 0; otherwise, it is set to 1. The 4 bit content of the index register is unaffected.

Example:



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

Mnemonic: BBL (Branch back and load data to the accumulator)
OPR OPA: 1100 DDDD
Symbolic: (Stack) → P_L, P_M, P_H; DDDD → ACC
Description: The program counter (address stack) is pushed down one level. Program control transfers to the next instruction following the last jump to subroutine (JMS) instruction. The 4 bits of data DDDD stored in the OPA portion of the instruction are loaded to the accumulator. BBL is used to return from subroutine to main program.

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

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

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

Mnemonic: FIN (Fetch indirect from ROM)
OPR OPA: 0011 RRR0
Symbolic: (P_H) (0000) (0001) → ROM address
(OPR) → RRR0
(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).

Mnemonic: HLT

OPR OPA: 0000 0001

Symbolic: 1 → HALT 1 → STOP

Description: The processor sets the HALT and STOP flip-flops. Program counter incrementer and data input buffers are inhibited. The processor executes NOP continuously; continuation can occur by means of STOP or INTERRUPT control.

In this mode, the Program Counter + 1 is gated out at A_1, A_2, and A_3, times on the data bus. M_1, M_2 times will contain the addressed ROM instruction on the data bus. X_1, the 4 bit Accumulator contents, X_2 and X_3 will contain the 8 bit SRC register.

Mnemonic: BBS

OPR OPA: 0000 0010

Symbolic: (Stack → P_L, P_M, P_H;)
SRCR0 → DB(X2)
SRCR1 → DB(X3)

Description: This instruction is a combination of BRANCH BACK and SRC. The effective address counter is decremented and program control is returned to the location saved by the forced JMS which occurred at the beginning of the interrupt routine. In addition, the content of the SRC register is sent out at X_2 and X_3 of the instruction cycle, thus restoring the I/O port selection. This instruction will also turn off the INTA line re-enabling the CPU for Interrupt.

The previously selected Index register bank will also be restored during this instruction.

Mnemonic: LCR

OPR OPA: 0000 0011

Symbolic: (CR) → ACC

Description: The 4 bit contents of the COMMAND REGISTER are transferred to the ACCUMULATOR. This allows saving the command register values before processing the interrupt.

Mnemonic: OR4

OPR OPA: 0000 0100

Symbolic: ($RRRR_4$) V (ACC) → ACC

Description: The 4 bit contents of index register #4 are logically "OR-ed" with the ACCUMULATOR. The result is placed in the ACCUMULATOR and the CARRY flip-flop is unaffected.

Examples:

(ACC)	0101
($RRRR_4$)	1001
ACC	1101

(ACC)	0000
($RRRR_4$)	1000
ACC	1000

Mnemonic: OR5

OPR OPA: 0000 0101

Symbolic: ($RRRR_5$) V (ACC) → ACC

Description: The 4 bit contents of index register #5 are logically "OR-ed" with the ACCUMULATOR. Carry flip-flop is unaffected.

Mnemonic: AN6

OPR OPA: 0000 0110

Symbolic: $(RRRR_6)\ \Lambda\ (ACC) \rightarrow ACC$

Description: The 4 bit contents of index register #6 are logically "AND-ed" with the ACCUMULATOR. The result is placed in the ACCUMULATOR and the CARRY is unaffected.

Examples:

(ACC)	0110
$(RRRR_6)$	0100
ACC	0100
(ACC)	1111
$(RRRR_6)$	0001
ACC	0001

Mnemonic: AN7

OPR OPA: 0000 0111

Symbolic: $(RRRR_7)\ \Lambda\ (ACC) \rightarrow ACC$

Description: The 4 bit contents of index register #7 are logically "AND-ed" with the ACCUMULATOR. Carry flip-flop is unaffected.

Mnemonic: DB0

OPR OPA: 0000 1000

Symbolic: Enable → $CM\text{-}ROM_0$

Description: DESIGNATE ROM BANK 0. The most significant bit of the COMMAND REGISTER, CR_3, is reset. On the third instruction cycle following its execution, it causes $CM\text{-}ROM_0$ to be activated. This Bank is selected with reset.

Mnemonic: DB1

OPR OPA: 0000 1001

Symbolic: Enable → $CM\text{-}ROM_1$

Description: DESIGNATE ROM BANK 1. The most significant bit of the COMMAND REGISTER, CR_3, is set. On the third instruction cycle following its execution, it causes $CM\text{-}ROM_1$ to be activated.

Mnemonic: SB0

OPR OPA: 0000 1010

Symbolic: 1 → RB0, 0 → RB1

Description: SELECT INDEX REGISTER BANK 0. The index register bank select flip-flop is reset. Index registers 0 - 7, 8 - 15 will be available for program use. This bank is to be selected with a Reset.

Mnemonic: SB1

OPR OPA: 0000 1011

Symbolic: 0 → RB0 1 → RB1

Description: SELECT INDEX REGISTER BANK 1. The index register bank select flip-flop is set. Index registers 0* - 7*, 8 - 15 will be available for program use.

Mnemonic: RPM

OPR OPA: 0000 1110

Symbolic: (1111) (SRC) → ROM/RAM address
(DDDD) → ACC

Description: READ PROGRAM MEMORY. This instruction can be used only with the 4289 Standard Memory and I/O Interface Chip. The contents of the previously selected nibble of R/W Program Memory are transferred to the 4040 and loaded to the ACCUMULATÓR.

Mnemonic: EIN
OPR OPA: 0000 1100
Symbolic: 1 → IE
Description: ENABLE INTERRUPT. Internal interrupt detection logic is enabled.

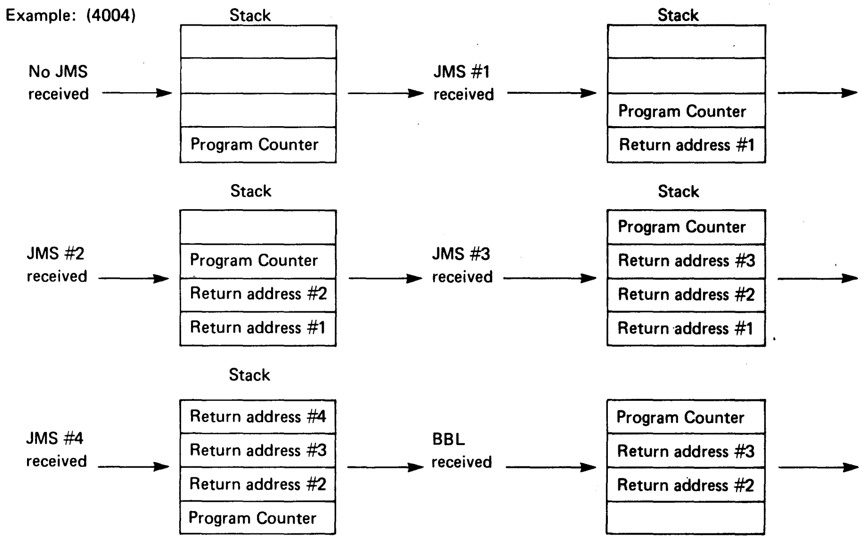
Mnemonic: DIN
OPR OPA: 0000 1101
Symbolic: 0 IE
Description: DISABLE INTERRUPT. Internal interrupt detection logic is disabled.

D. 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: $A_2\ A_2\ A_2\ A_2\quad A_1\ A_1\ A_1\ A_1$
Symbolic: $A_1\ A_1\ A_1\ A_1 \rightarrow P_L,\ A_2\ A_2\ A_2\ A_2 \rightarrow P_M,\ A_3\ A_3\ A_3\ A_3 \rightarrow P_H$
Description: Program control is unconditionally transferred to the instruction locator 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: $A_2\ A_2\ A_2\ A_2\quad A_1\ A_1\ A_1$
Symbolic: $(P_H, P_M, P_L + 2) \rightarrow$ Stack
$A_1\ A_1\ A_1\ A_1 \rightarrow P_L,\ A_2\ A_2\ A_2\ A_2 \rightarrow P_M,\ A_3\ A_3\ A_3\ A_3 \rightarrow P_H$
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 (8 registers in 4040). One of them is used as the program counter, therefore nesting of JMS can occur up to 3 levels (7 levels in the 4040).



The deepest return address is lost.

Mnemonic: JCN (Jump conditional)

1st word OPR OPA: 0001 C_1 C_2 C_3 C_4

2nd word OPR OPA: A_2 A_2 A_2 A_2 A_1 A_1 A_1 A_1

Symbolic: If C_1 C_2 C_3 C_4 is true, A_2 A_2 A_2 $A_2 \rightarrow P_M$
A_1 A_1 A_1 $A_1 \rightarrow P_L$, P_H unchanged
if C_1 C_2 C_3 C_4 is false,
$(P_H) \rightarrow P_H$, $(P_M) \rightarrow P_M$, $(P_L + 2) \rightarrow P_L$

Description: If the designated condition code is true, program control is transferred to the instruction located at the 8 bit address A_2 A_2 A_2 A_2, A_1 A_1 A_1 A_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.

C_X Condition Table for JCN Instruction

C_1	C_2	C_3	C_4	
0	0	0	0	NO OPERATION
0	0	0	1	Jump if test = 0 (High)
0	0	1	0	Jump if CY = 1
0	0	1	1	Jump if test = 0 or CY = 1
0	1	0	0	Jump if AC = 0
0	1	0	1	Jump if test = 0 or AC = 0
0	1	1	0	Jump if CY = 1 or AC = 0
0	1	1	1	Jump if test = 0 or CY = 1 or AC = 0
1	0	0	0	Jump Unconditionally
1	0	0	1	Jump if test = 1 (Low)
1	0	1	0	Jump if CY = 0
1	0	1	1	Jump if test = 1 and CY = 0
1	1	0	0	Jump if AC ≠ 0
1	1	0	1	Jump if test = 1 and AC ≠ 0
1	1	1	0	Jump if CY = 0 and AC ≠ 0
1	1	1	1	Jump if test = 1 and CY = 0 and AC ≠ 0

Example:

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

Several conditions can be tested simultaneously.

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

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

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

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: $A_2 A_2 A_2 A_2$ $A_1 A_1 A_1 A_1$
Symbolic: (RRRR) + 1 → RRRR, if result = 0
$(P_H) \rightarrow P_H$, $(P_M) \rightarrow P_M$, $(P_L + 2) \rightarrow P_L$:
if result ≠ 0 $(P_H) \rightarrow P_H$,
$A_2 A_2 A_2 A_2 \rightarrow P_M$, $A_1 A_1 A_1 A_1 \rightarrow P_L$
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 $A_2 A_2 A_2 A_2$, $A_1 A_1 A_1 A_1$ 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: $D_2 D_2 D_2 D_2$ $D_1 D_1 D_1 D_1$
Symbolic: $D_2 D_2 D_2 D_2 \rightarrow$ RRR0
$D_1 D_1 D_1 D_1 \rightarrow$ RRR1
Description: The 2nd word represents 8 bits of data which are loaded into the designated index register pair.

E. Input/Output Instructions

The following I/O instructions are described as they relate to ROM and RAM devices. These same instructions (mnemonics) can be redefined for devices other than ROM and RAM.

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: RD0 (Read RAM status character 0)
OPR OPA: 1110 1100
Symbolic: $(M_{S0}) \rightarrow$ 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: $(M_{S1}) \rightarrow$ ACC

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

Mnemonic: RD3 (Read RAM status character 3)
OPR OPA: 1110 1111
Symbolic: $(M_{S3}) \rightarrow$ 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 port 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) 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) → 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) → M_{S1}

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

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

Mnemonic: WRR (Write ROM port)
OPR OPA: 1110 0010
Symbolic: (ACC) → 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.) No operation is performed on I/O lines coded as inputs.

Mnemonic: WMP (Write memory port)
OPR OPA: 1110 0001
Symbolic: (ACC) → 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) → 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.

F. Accumulator Group Instructions

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

Mnemonic: CLC (Clear carry)
OPR OPA: 1111 0001
Symbolic: 0 → 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 → 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:

$$\begin{array}{c} (ACC) \\ \downarrow \\ a_3\ a_2\ a_1\ a_0 \\ +)\ \underline{1\ \ 1\ \ 1\ \ 1} \\ \begin{array}{c} C_4 \\ \downarrow \\ CY \end{array}\ \underbrace{S_3\ S_2\ S_1\ S_0}_{ACC} \end{array}$$

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: $0 \rightarrow ACC,\ (CY) \rightarrow a_0,\ 0 \rightarrow CY$
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 if 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 least one RESET). DCL remains latched until it is changed.

The selection is made according to the following truth table.

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

	DATA @ X_2	DATA @ X_3	
INSTRUCTION	$D_3 D_2 D_1 D_0$	$D_3 D_2 D_1 D_0$	COMMENTS
NOP	1 1 1 1	1 1 1 1	
JCN	1 1 1 1	1 1 1 1	
A_2, A_1	1 1 1 1	1 1 1 1	
FIM RRR0	(RRR0)	(RRR1)	The content of address
D_2 D_1	1 1 1 1	1 1 1 1	pair RRR
SRC RRR1	(RRR0)	(RRR1)	
FIN RRR0	(RRR0)	(RRR1)	
2nd cycle	1 1 1 1	1 1 1 1	
JIN RRR0	(RRR0)	(RRR1)	
JUN A_3	A_3	1 1 1 1	
A_2, A_1	A_3	1 1 1 1	
JMS A_3	A_3	1 1 1 1	
A_2, A_1	A_3	1 1 1 1	
INC RRRR	(RRRR)	(RRRR) +1	Content of register RRRR;
ISZ RRRR	(RRRR)	(RRRR) +1	Content +1 of RRRR
A_2, A_1	1 1 1 1	1 1 1 1	
ADD RRRR	(RRRR)	1 1 1 1	Content of register RRR
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,	(ACC)	1 1 1 (CY)	Content of accumulator;
WR2, WR3,			Content of CY F/F is
WPM, WMP, WRR	(ACC)	1 1 1 (CY)	present on D_0
RDM, RD0, RD1, RD2	(M) or (Input)	(M) or (INPUT)	Data fetched from RAM or
RD3, ADM, SBM, RDR			input
CLB, CLC, IAC, CMC	0 0 0 0	1 1 1 1	
CMA, RAL, PAR, TCC			
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	1 1 1 1	X_2 depends on ACC content
	0 1 1 0		
KBP	0000,0001,0010		X_2 depends on ACC content
	0011,0100,1111	1 1 1 1	

Figure 1-22. 4004 Data Bus Content During Execution of Each Instruction.

	DATA @ X_2	DATA @ X_3	
INSTRUCTION	D_3 D_2 D_1 D_0	D_3 D_2 D_1 D_0	COMMENTS
NOP	1 1 1 1	1 1 1 1	
HLT*	1 1 1 1	1 1 1 1	After execution of HLT, processor enters STOP mode.
BBS	(SRCH)	(SRCL)	(SRCH) means contents of 4 high order bits of SRC register.
LCR	(COM. REG)	1 1 1 1	
OR4	(0100)	1 1 1 1	
OR5	(0101)	1 1 1 1	
AN6	(0110)	1 1 1 1	
AN7	(0111)	1 1 1 1	
DB0	1 1 1 1	1 1 1 1	
DB1	1 1 1 1	1 1 1 1	
SB0	1 1 1 1	1 1 1 1	
SB1	1 1 1 1	1 1 1 1	
EIN	1 1 1 1	1 1 1 1	
DIN	1 1 1 1	1 1 1 1	
RPM	(P.M.)	(P.M.)	Program memory content

Figure 1-23. Summary of 4040 Data Bus Content During Instruction Execution.

CHAPTER 2
INTRODUCTION TO PROGRAMMING THE MCS-40™ MICROCOMPUTER SYSTEM

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-40 microcomputer is described in detail in the 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. When reviewing these examples refer frequently to the instruction definitions.

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
		$C_1 C_2 C_3 C_4$
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	
		$C_1 C_2 C_3 C_4$	
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 an 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, 20H	;Fetch immediate (direct) from ROM data (0010, 0000), to index register pair 0. (20H refers to 20 Hexidecimal.)
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 (4001).

```
RDR          ;Read the 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).

> *NOTE: All command statements are punctuated with ; at the initial statement Description entry.*

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; 0C0H; 192

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 a 0 into the appropriate port bit. Let us assume that we will only use the high order bit of the port on RAM #3 and that it is initially set at zero (so that the program does not have to reset it). Furthermore, let us assume that we do not care about the other three bits of the port.

First let us set the accumulator to 0

```
LDM    0      ;Set accumulator to 0
```

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

```
RAL          ;Rotate left (accumulator and carry)
CMC          ;Complement carry
RAR          ;Rotate right (accumulator and carry)
```

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

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

```
LDM   8      ;Load data DDDD (1000) to the
              accumulator.
XCH   15     ;Exchange contents of index register 15
              and accumulator
LDM   0      ;Load (0000) to accumulator
```

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

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

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

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

```
ADD   15     ;Add contents of register 15   ┐
              (1000 previously stored in    │
              the register) to accumulator  │  one clock
WMP          ;Write the contents of the     ├─ pulse
              accumulator into the          │  generated
              previously selected RAM       │
              output port                   │
ADD   15                                    │
WMP                                         ┘
```

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) to index
		12,	0	;register pair location 0
(6)		SRC	0	;Send address (contents of index register pair 0) to RAM
(7)		LDM	0	;Set accumulator to 0
(8)	LOOP	ADD	15	;Add contents of register 15 to accumulator
(9)		WMP		;Write contents of accumulator into RAM output ports
(10)		ADD	15	;Add contents of Register 15 to accumulator
(11)		WMP		;Write contents of accumulator into RAM output ports
(12)		ISZ	14,LOOP	;Increment contents of register 14. Go to ROM address
				;A_2, A_1 (called Loop) if result ≠ 0, otherwise skip.

Explanation of Program

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

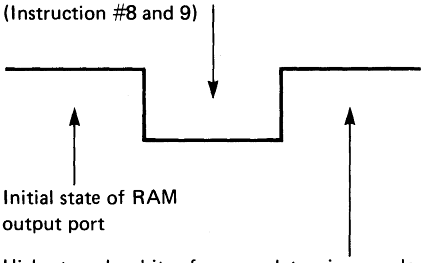
(b) Instruction #3 and #4 – Loads the number 6 (0110) into index register number 14 (1110)

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

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

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

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



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 are 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 are 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.

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.

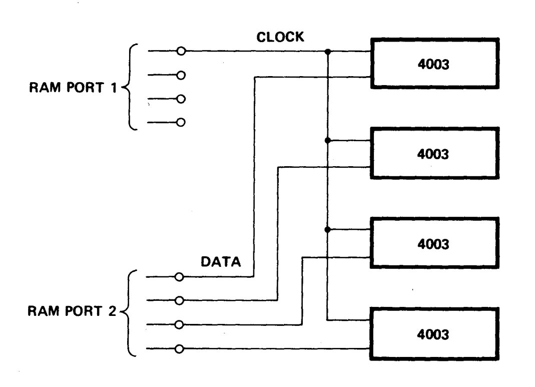


Figure 2-1. RAM Output Ports Driving Groups of Shift Registers

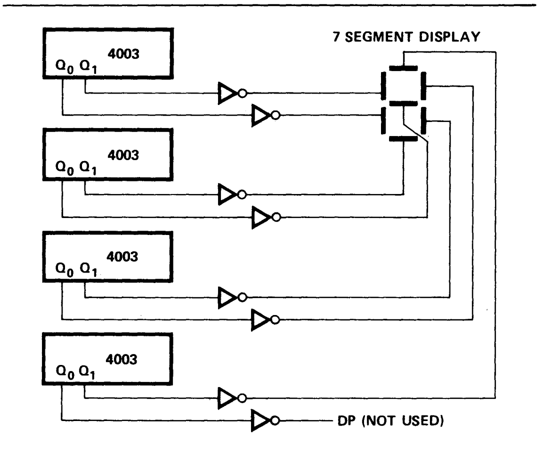


Figure 2-2. Shift Registers Driving Seven Segment LED Displays

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
             ;selected RAM character) into Port #2
SRC          ;Send address to RAM #1
LDM   0      ;Set accumulator to "0"
ADD  15
WMP
             ;Generate 1 clock pulse
ADD  15
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 4040 can be utilized to advantage to save these converters. Suppose the LED displays are wired as shown 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 initialized 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,DBGH    ;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            │
ADD      15    ├ ;generate one clock pulse
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           │
ADD     15    ├ ;generate one clock pulse
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.
```

*Operands are device dependent.

Example #6

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 (BBS in the 4040) 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 or BBS.

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

```
FIM     Reg Pair,     Date (Byte)
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 use 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

$$+\underbrace{.1372994157387406}_{\text{Mantissa (16 digits)}} \times 10^{-\underset{\underset{\text{Exponent (2 digits)}}{\uparrow}}{59}}$$

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.

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	Main Memory Character #
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	Status Character #
	0	1	0	1	1	
Exponent Sign – Neg.	0	0	0	1	2	
Mantissa Sign – Pos.	0	0	0	0	3	

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

	Mnemonic		Machine	Language
			OPR	OPA
Select Bank #2	LDM	2	1101	0010
	DCL		1111	1101
Select Chip #3, Register #1 Character #6	FIM	4	0010	1000
	13,	6	11 01	0110
			Chip #3 Register #1	Main memory Character #6
	SRC	4	0010	1001
Fetch the Mantissa sign From status Character #3 to Register #10 in the CPU	RD3		1110	1111
	XCH	10	1011	1010
Fetch the exponent sign From status Character #2 to Register #11 in the CPU	RD2		1110	1110
	XCH	11	1011	1011
Fetch the exponent from status Character #1 and #0 to Register #12 and #13 respectively	RD1		1110	1101
	SCH	12	1011	1100
	RDO		1110	1100
	SCH	13	1011	1101
Fetch the previously selected main memory Character #6 (which stored the decimal digit 7 to the accumulator	RDM		1110	1001

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-40T.M. microcomputer. The MCS-40 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. Refer to the Programmer's Manual for further details.

Example #9 – Interrupt Routine (4040)

The Interrupt signal, when armed and activated, causes the CPU to suspend normal program execution. The CPU is forced to execute a predetermined Interrupt subroutine starting from location 003. At the completion of the Interrupt routine, the CPU is returned to the normal program execution with a BBS instruction.

The Interrupt utilizes the address push down stack. When an Interrupt is executed, the program counter is pushed up one level and is reloaded with address 003. Execution will proceed from this location. The location may contain Jump (JIN, JUN) which allows the user to place the Interrupt routine anywhere in memory. A stack level should always be reserved for the interrupts to avoid overflowing the stack.

Since the Interrupt forces the CPU out of the normal instruction sequence, the state of the CPU's internal register values must be preserved and restored prior to returning from the Interrupt routine. The SRC and the Index Register Bank will automatically be saved. The designer should store the value of the *accumulator* and *carry flip-flop, command register, current ROM bank*, and *index registers* that will be used during the Interrupt execution.

The MCS-40 system has three groups of eight index registers organized into two banks. Registers 8-15 are common to both. Bank 0, Register 0-7 can be designated for normal program execution while Bank 1, Register 0-7 can be designated for interrupt execution. The designer need only switch banks to save the first seven register values. These will be restored automatically with the BBS.

The following is an example of a typical Interrupt subroutine. It illustrates entering and exiting the Interrupt routine, determining an Interrupt source from multiple requests and then directing the Interrupt routine to service the interrupting device.

The programmer must first enable the CPU to accept an Interrupt. This is done by executing the EIN instruction. When an Interrupt occurs, the program counter is forced to location 003 in whichever ROM bank it is executing.

```
Label   Code   Operand        Comment
Interrupt occurs
003     JUN    INTR     ;Branch to Interrupt
INTR    SB0             ;Switch to Bank 0
        XCH    5        ;Store accumulator (Acc) in
                         Register (Reg) 5
        TCC             ;Place carry (CY) in Acc
                         (a3 bit position)
        XCH    6        ;Store carry in Reg 6
        LCR             ;Load command Reg (CR)
                         into Acc
        XCH    7        ;Store CR in Reg 7
        SB1             ;Switch to Bank 1. This bank
                         could typically be used for
                         Interrupt work registers while
                         Bank 0 could be established
                         for normal program
                         execution. This is an
                         arbitrary definition.
TS      JUN    I0       ;Interrupt vector table
TS+1    JUN    I1       ;Eight branches, one for each
 .       .               possible source
 .       .
 .       .      .
 .       .      .
TS+7    JUN    I7
```

Assume there are eight sources of Interrupt. The Interrupt is a common line such that one or more sources can Interrupt simultaneously. The Interrupt Acknowledge should be daisy-chained so that only one device will respond to the Interrupt. This is one technique for arbitrating simultaneous interrupt requests.

Let us further assume that each of the eight sources has a 3 bit binary address. The interrupting device will place this address on ROM 2 (4308) Port 3. This port is designated an input, common to all Interrupt sources.

The program will interrogate the above port, taking the address of the interrupting device and adding it to a table of JUN (TS) instructions. The resulting source will cause the program to vector a unique routine to service the particular Interrupting source of interest.

```
Code   Operand   Comment
FIM    0,176     ;Load 1011,0000 into the index register
                  pair 0 (ROM 2, Port 3)
SRC    0         ;Select ROM 2, Port 3 for I/O transfer
RDR              ;Read in the 3 bit address to Acc. This
                  address will be multiplied by 2 and
                  added to JS. The result will access the
                  JUN of interest.
CLC              ;Clear Carry
RAL              ;Shift content of Acc toward MSB posi-
                  tion (multiple by 2) will allow the TS
                  table to be accessed by 2 (JUN being 2
                  bytes) on even boundaries.
FIM    8,TS      ;Fetch table starting address
CLC              ;Clear Carry
ADD    9         ;This adds the least significant nibble of
                  TS to the source address.
XCH    1         ;This stores the partial sum in Register 1
                  and loads zero into Acc.
ADD    8         ;This adds any previously generated
                  carry into the most significant nibble
                  position of TS.
XCH    0         ;Stores the results in Register 0. The
                  indirect address has been accumulated.
JIN    0         ;Indirect jump to a table location.
```

After the unique source interrupt routine has been initiated, a return must be generated. Prior to the return, the previous state of the CPU must be restored.

```
 SB0             ;Switch Bank 0
 XCH    7        ;Read back command register value
*RAL             ;Shift a3 (CM-ROM) bit into CY.
*JNC    $+2      ;Jump if CY is zero, hence select DB0
                  (CM-ROM0), otherwise switch banks to
                  DB1 (CM-ROM).
*DB1             ;Select CM-ROM.
*RAR             ;Shift back.
 DCL             ;Load command register.
 XCH    6        ;Fetch carry.
 RAR             ;Restore carry.
 XCH    5        ;Fetch accumulator.
 BBS             ;The program counter is restored to the
                  value prior to the interrupt. The SRC is
                  restored to the current PCL selection.
                  The index register bank selection is
                  restored. The Interrupt Acknowledge
                  line is cleared.
```

*These instructions may be omitted if only one ROM bank is being used.

The above example assumes that index registers did not contain information to be preserved before or after the Interrupt. If index registers are to be saved, they would be saved in RAM or protected index register zones. This is program dependent.

CHAPTER 3 SYSTEMS

INTERFACE DESIGN WITH THE MCS-40™ SYSTEM

MCS-40 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-40 system must include, as part of his design, suitable interface circuits and programs.

Devices to be operated or interrogated by an MCS-40 computer are attached to the system via the input and output data ports. The design of an interface consists of the following steps:

1. Establish I/O configuration by assigning peripheral device connections to port connections. If the number of available output ports is insufficient, 4003 output port expanders, 4207, 4209 or 4211 general purpose I/O may be used. Reducing the number of I/O lines can be accomplished by the use of multiplexers. These multiplexers can be controlled by output ports.
2. Develop the necessary level conditioning circuits for each signal. Port inputs and outputs are at MOS levels (logic 0 = 0V with a series output resistance of typically 150 ohms, 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, displays, etc. For TTL compatibility refer to next section.
3. Write the programs necessary to interpret inputs and generate the output levels necessary for proper operation of the peripherals. This is best done by use of a functional flow diagram.

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.

Keyboards

The MCS-40 microcomputer 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 4040 CPU may be interrogated to determine if a key has been pressed. If the Interrupt is used, the CPU will be placed into a subroutine to incorporate the keyboard.

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 40 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	KBP	0001
0010		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 simultaneously to determine if a scan is required.

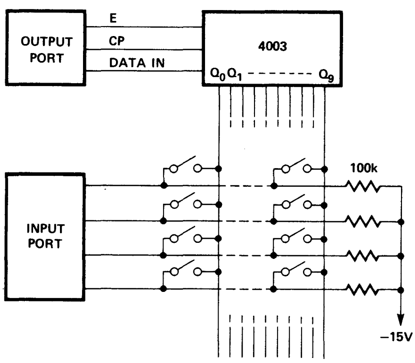


Figure 3-1. Keyboard Interface – (Scanned Array).

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

Display

Display devices such as Nixie$^{\text{T.M.}}$ tubes and LED arrays are easily interfaced to the MCS-40$^{\text{T.M.}}$ 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$^{\text{T.M.}}$ 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$^{\text{T.M.}}$ array, one set of (10 or 11) cathode drivers is used in combination with one anode driver for each Nixie$^{\text{T.M.}}$ 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.

NOTE: Nixie$^{\text{T.M.}}$ is a registered trademark of Burroughs Corporation.

Although the technique described above specifically mentioned Nixie$^{\text{T.M.}}$ 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.

Printer

Most standard printer interfaces are parallel in nature. The interface signals may be divided into two groups, data and control. The control section would typically consist of forms manipulation and printing handshaking. Due to their parallel nature, the general purpose I/O devices lend themselves well to this application, having 16 bits of I/O, printer functions can be assigned to the various bit lines and programmed accordingly.

Teletype Interface

The MCS-40 system is designed to interface with all types of terminal devices. Interface with teletype is a typical

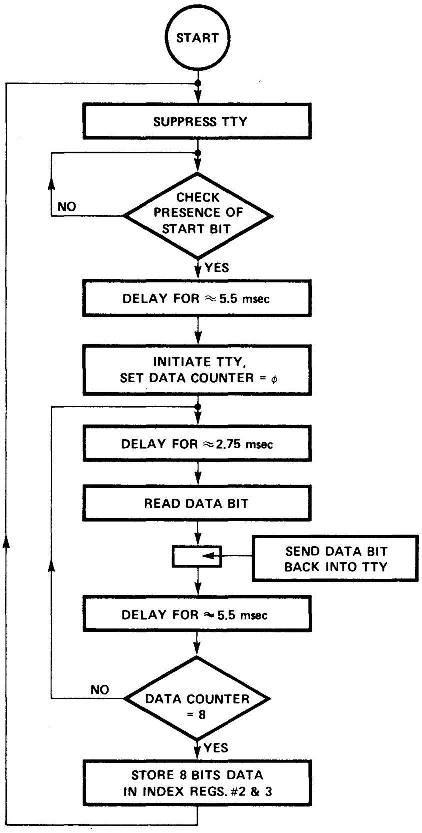

START
SUPPRESS TTY
CHECK PRESENCE OF START BIT
NO
YES
DELAY FOR ≈ 5.5 msec
INITIATE TTY, SET DATA COUNTER = φ
DELAY FOR ≈ 2.75 msec
READ DATA BIT
SEND DATA BIT BACK INTO TTY
DELAY FOR ≈ 5.5 msec
NO
DATA COUNTER = 8
YES
STORE 8 BITS DATA IN INDEX REGS. #2 & 3

Figure 3-2. Flow Chart for Teletype Interfaces.

example. The interface consists of three simple transistor circuits. (See figure below.) 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-40$^{T.M.}$ system is ideal for this timing control. The flow chart further explains the details of the program.

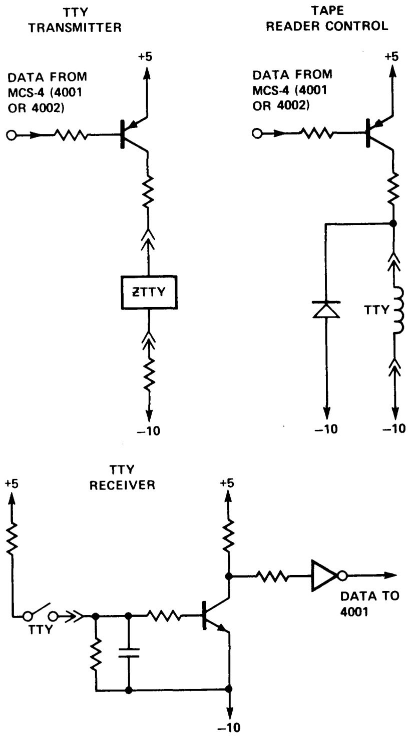


Figure 3-3. MCS-40 and Teletype Interface Circuits.

INTERFACE CHARACTERISTICS OF THE MCS-40 MICROCOMPUTER

The following pages provide the electrical characteristics for the MCS-40 system. For TTL compatibility, a resistor should be added between the output and V_{DD}. All outputs are push-pull MOS outputs.

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

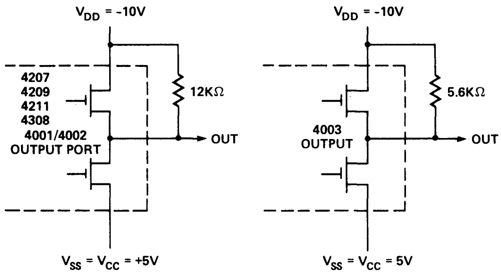


Figure 3-4. MCS-4 Output Configuration for TTL Compatibility.

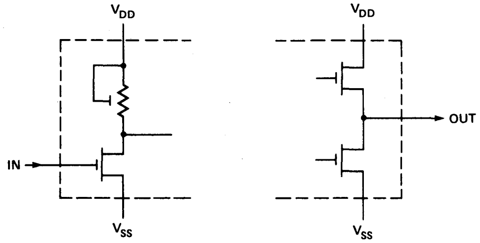


Figure 3-5. Typical MCS-4 Input and Output Circuitry.

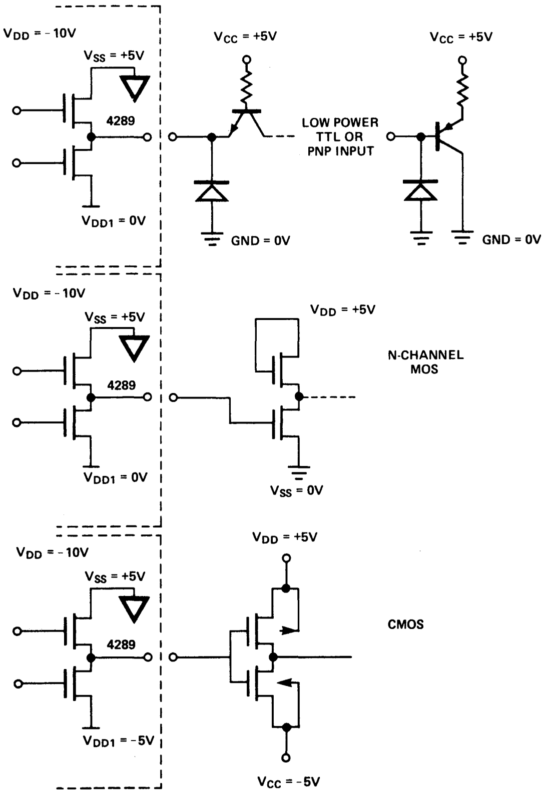


Figure 3-6. 4289 Interface Method by Using V_{DD1}.

SYSTEM CONFIGURATIONS

The following diagrams illustrate various configurations to which the MCS-40$^{\text{T.M.}}$ components can be applied. All of the block diagrams can be expanded to accommodate a specific application.

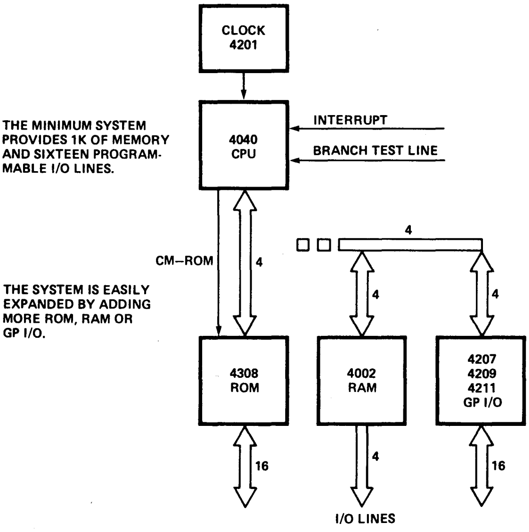
System Considerations

Although all MCS-40 components are designed to operate as a compatible family, there are certain system considerations that should be kept in mind when designing your system.

1. When determining the system clock time, the clock period should be set to the slowest component. If 4001 and 4308 are used in the same system, the maximum clock rate of the 4001 will determine the system time.

2. For normal operations, 4001 or 4308 ROMs may not be used in the same ROM bank with the 4289. This is because the 4289 will respond to all I/O and program references, presenting a data bus conflict with I/O and ROM components.

3. When using either 4207, 4209, or 4211 on a CM-RAM or CM-ROM in a system, the four most significant page



THE MINIMUM SYSTEM PROVIDES 1K OF MEMORY AND SIXTEEN PROGRAMMABLE I/O LINES.

THE SYSTEM IS EASILY EXPANDED BY ADDING MORE ROM, RAM OR GP I/O.

Figure 3-7. The Minimum System Provides 1K of Memory and Sixteen Programmable I/O Lines.

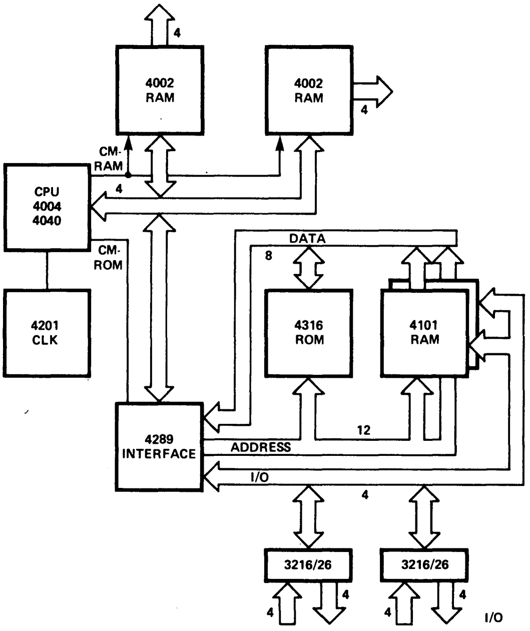


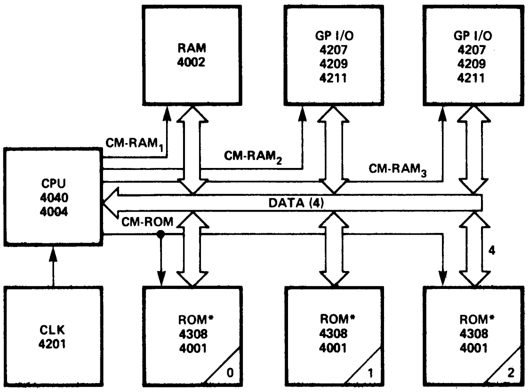
Figure 3-8. System Diagram Using Standard Memory Components.

numbers must be left vacant. ROM page 12, 13, 14, 15 (4308 chip select 3). This is done to avoid bus conflict.

4. If a system has two memory banks and the system utilizes the 4040 interrupt, location 003 of both memories should route the program to the interrupt routine.

5. A BBS will only restore a single SRC to the RAM bank enabled. Other RAM banks that would require SRC restoration after an Interrupt must be done under program control.

6. 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-RAM$_0$/CM-ROM$_0$ is selected. To completely clear all registers and RAM locations in the CPU the reset signal must be applied for at least 12 full instruction cycles (96 clock cycles) to allow the index register refresh counter to scan all locations in memory. (256 clock cycles for the 4002 RAM.)



*ROM ADDRESS 3 on Pages 12, 13, 14, 15 must be vacant for GP I/O on CM-RAM.

Figure 3-9. System Diagram Illustrating GP I/O on CM-RAM.

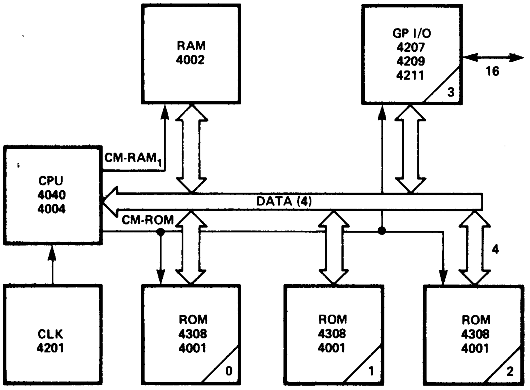


Figure 3-10. Typical System Using GP I/O.

7. Memory address, memory data, I/O bus, and control lines from 4289 are defined with respect to positive logic. The MCS-40$^{\text{T.M.}}$ data and control lines from the CPU are defined with respect to negative logic. As a result, in program memory used with the 4289, programs should be coded with logic "1"=high level and logic "0"=low level (i.e., NOP = 0000 0000 = NNNN NNNN).

A preferred method is to use negative logic program memory and place inverting buffers at the data inputs of the 4289. This allows program code to be consistent with that of the 4001 and 4308 mask-programmed ROMs and assures that 4289 input capacitance will not limit system speed when using several 4702A PROMs for program storage.

Note that programs are defined for the ROM 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.

8. Input and output data from the 4289 I/O bus is defined in terms of positive logic. If these interface devices are used for prototyping a 4001/4308 program memory, care should be taken to be sure that the I/O ports for the ROMs are defined consistent with the 4289 system.

9. An I/O port associated with the 4289 can have lines with both input and output capability. On the 4001/4308 each I/O line may have only a single function, either input or output.

10. Although RAM Program Memory can be used for data storage, it is not a direct substitute for the 4002 read/write data storage. They perform distinctly different functions.

11. CM-ROM and CM-RAM$_0$ cannot be used to control 4002s when CM-ROM is used for 4289 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 a 4289 CM-RAM$_0$ in absence of a DCL behaves exactly like CM-ROM. For 4002's with FPO numbers H7224 or higher this precaution is not necessary.

12. The 4207, 4209 and 4211 will not operate in the same ROM bank with a 4289.

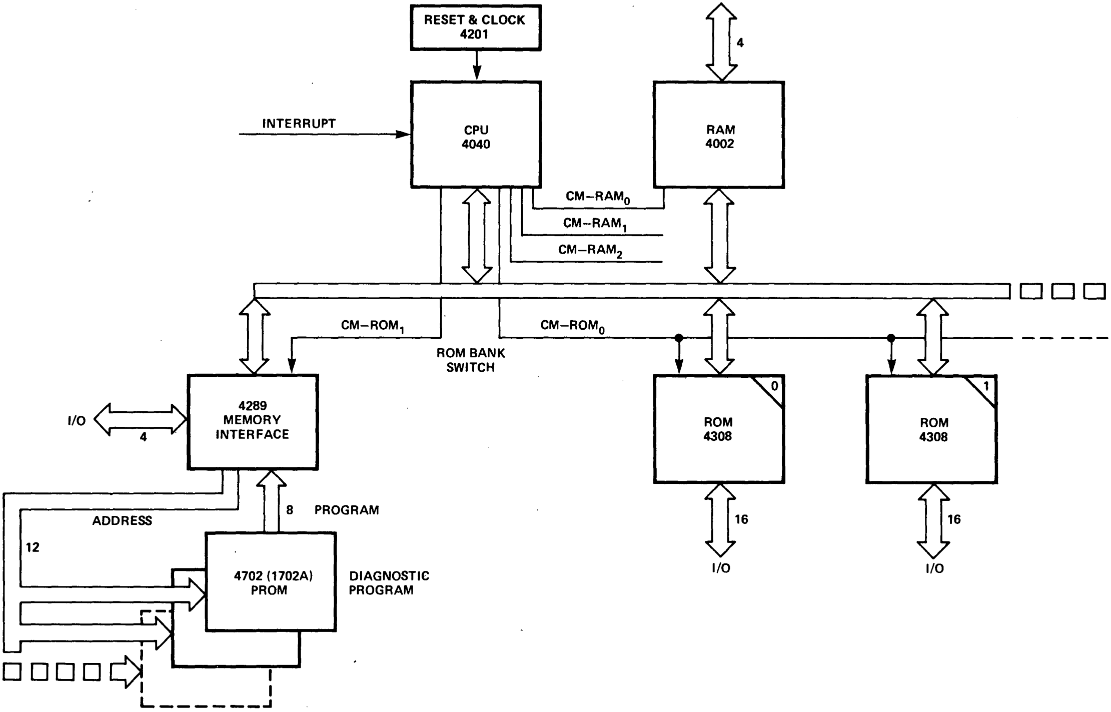


Figure 3-11. System Diagram Indicating the Dual Memory Environment and the Use of the 4289 and 4702.

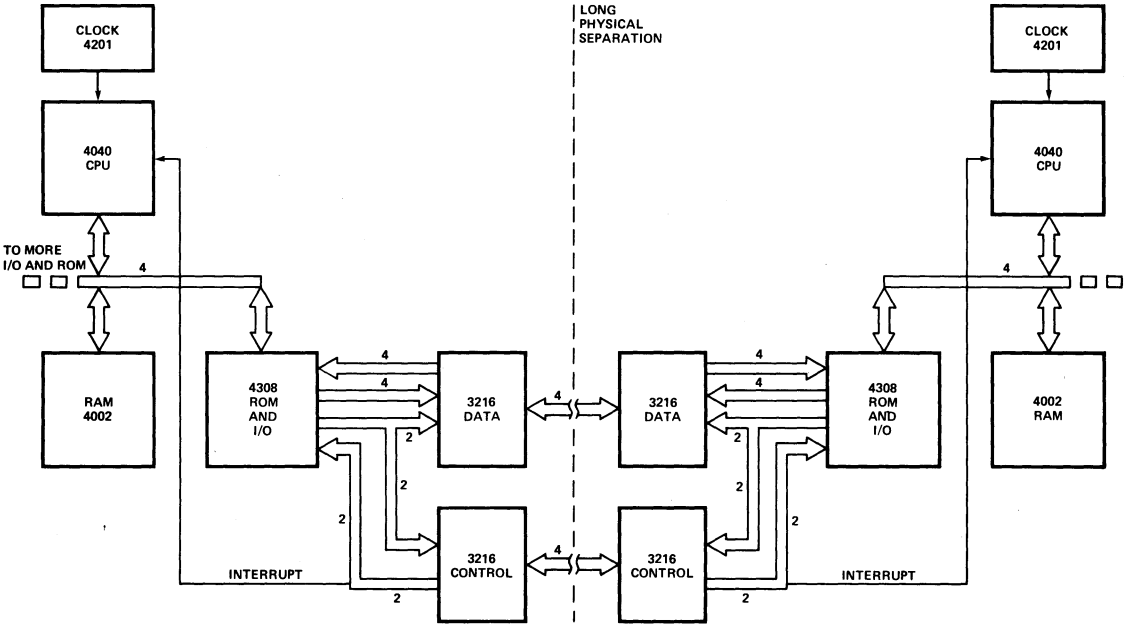


Figure 3-12. Dual Processor Communications Allows for Distributed Intelligence. This Interface can be Expanded to Provide any Width Word.

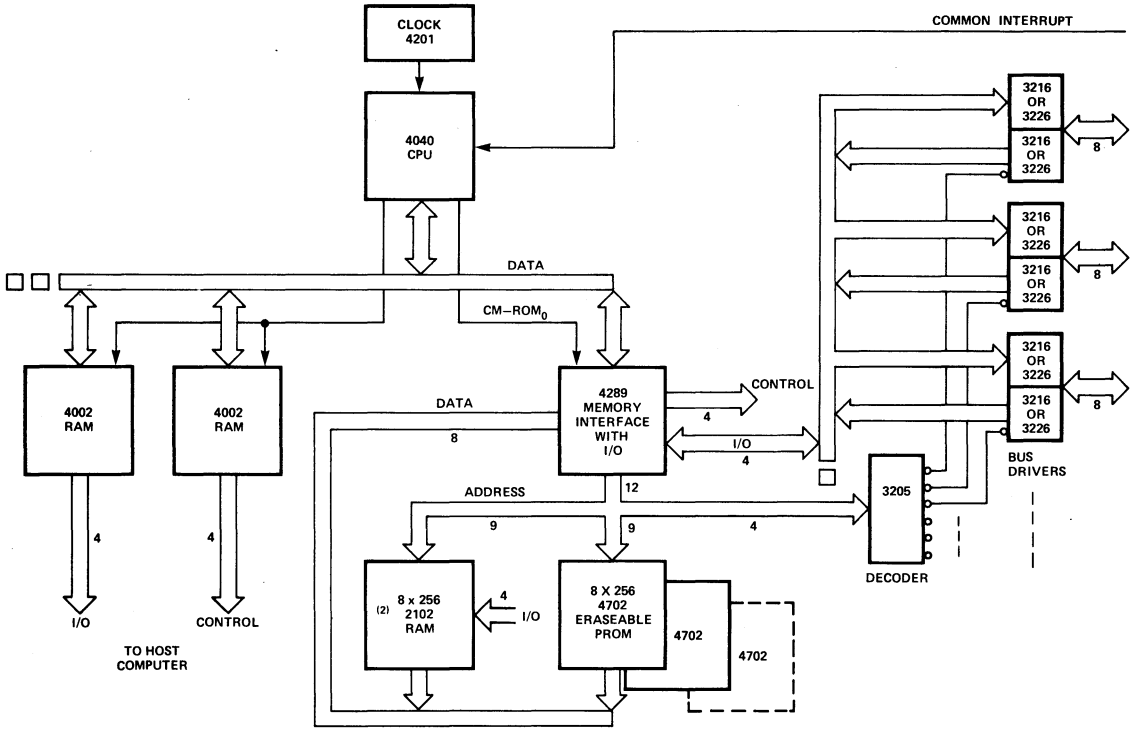


Figure 3-13. Multiple 8 Bit I/O Port with Writeable Program Memory.

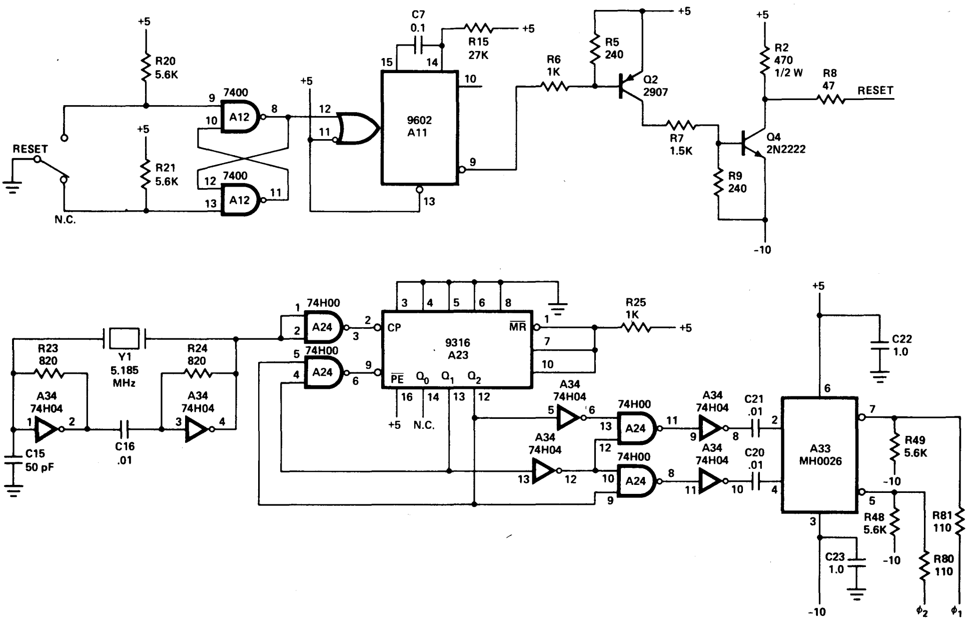


Figure 3-14. Suggested Clock and Reset Circuit.

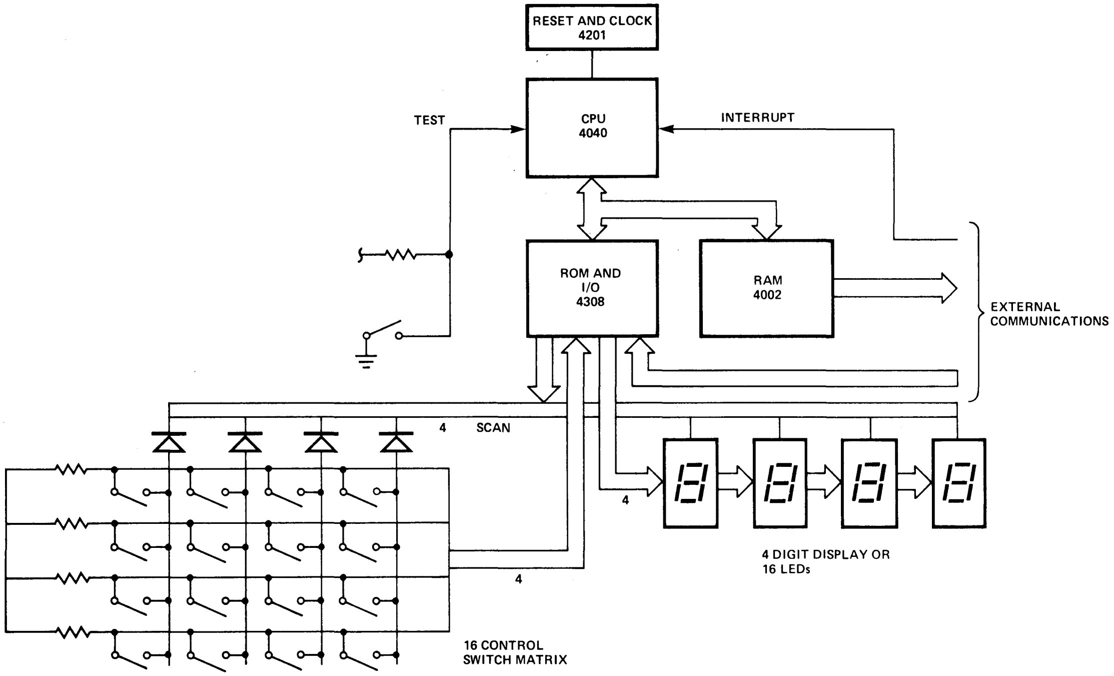


Figure 3-15. Control Panel Encoder with Display.

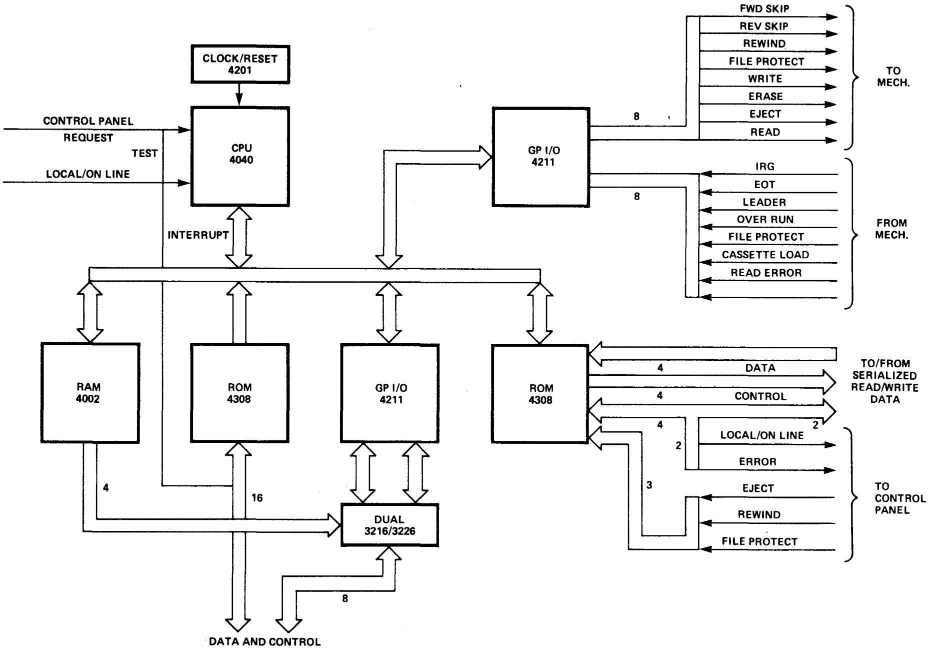


Figure 3-16. Cassette Controller.

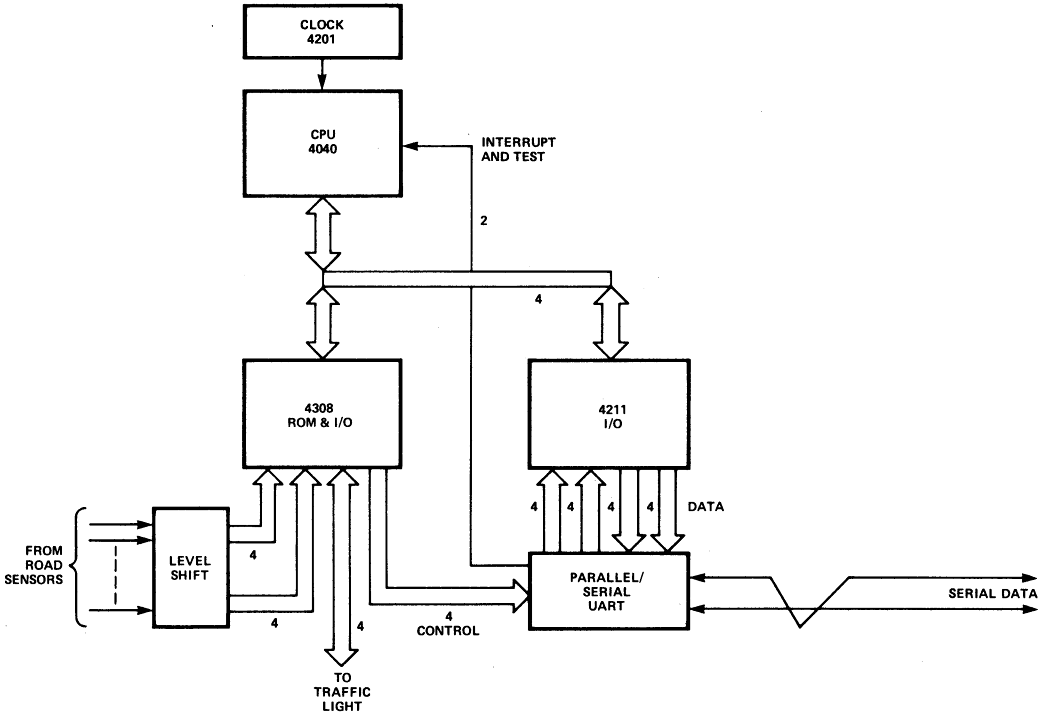


Figure 3-17. Traffic Light Control.

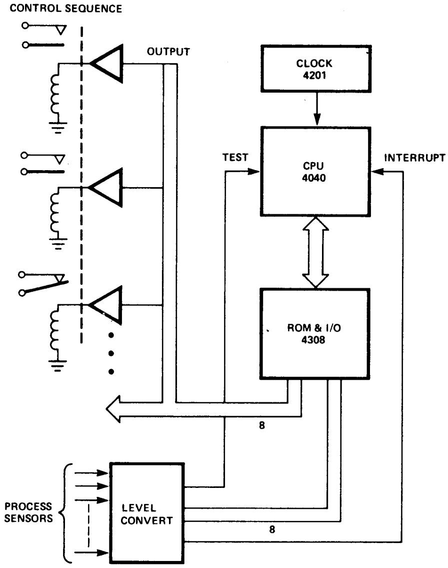


Figure 3-18. Basic Process Controller.

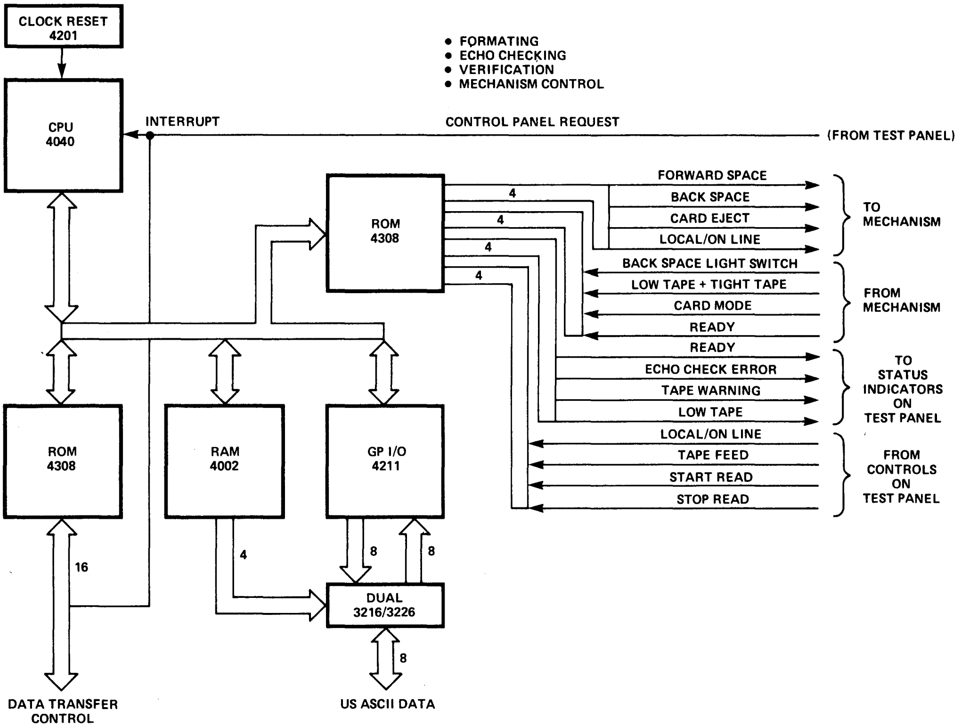


Figure 3-19. Paper Tape Reader and Punch Controller.

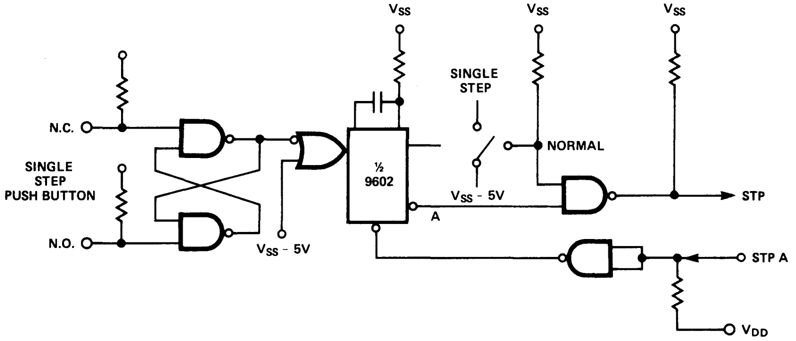


Figure 3-20. Suggested Single Step Feature Implementation.

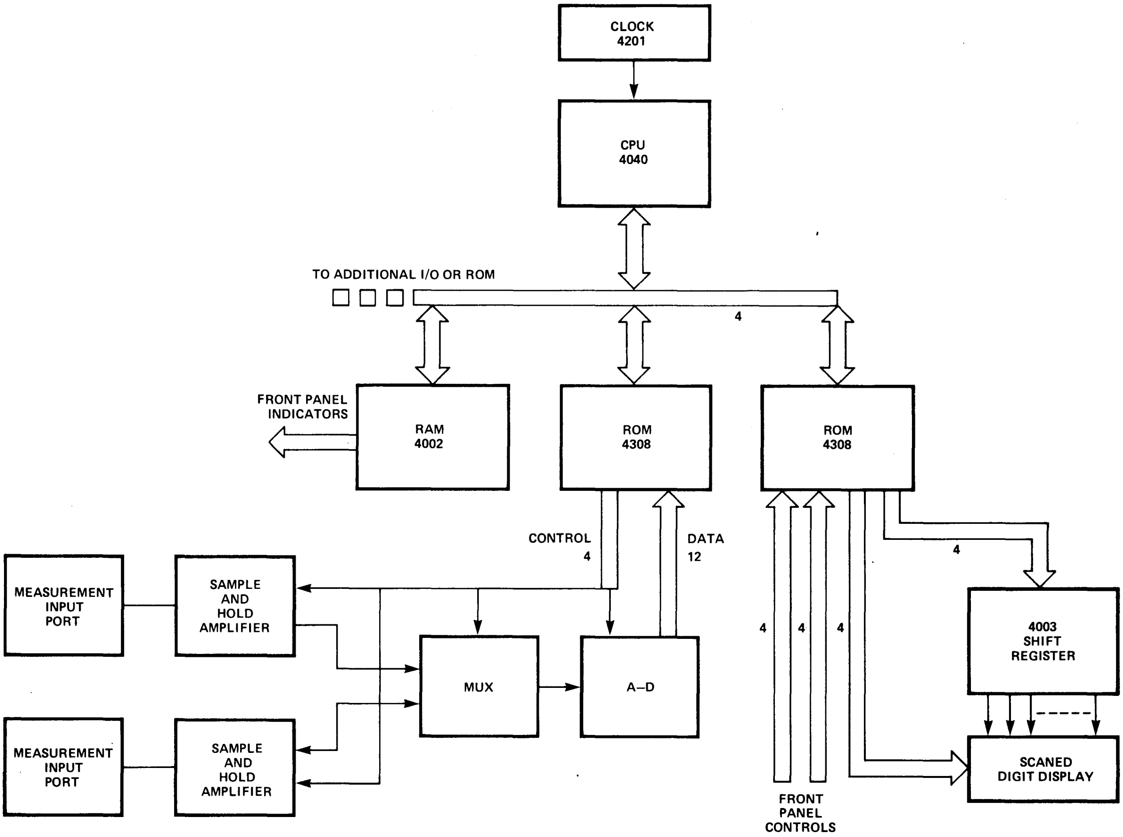


Figure 3-21. Typical A-D Instrument.

CHAPTER 4
SYSTEM COMPONENTS

4201*CLOCK GENERATOR

Introduction

The 4201 is a CMOS MSI integrated circuit designed to fill the clock requirements of the MCS-40™ microcomputer set. The 4201 contains a crystal controlled oscillator (XTAL external), clock generation circuitry, and both two phase MOS and TTL level clock driver circuits.

The 4201 also performs the power on reset function required by MCS-40 components and provides the logic necessary to implement the single-step function of the 4040 central processor unit.

Hardware Description

The 4201 is packaged in a 16 pin DIP. The pin configuration is shown in the following figure, and a functional description of each pin is given below:

*Available first quarter 1975.

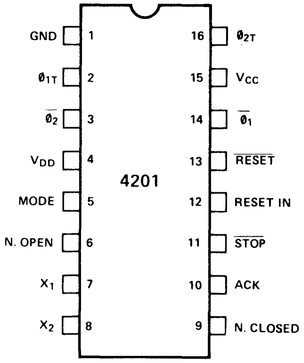


Figure 4-1. 4201 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1	GND	Circuit ground potential. This pin can be left floating for low power application. MOS clock output will be operative, TTL clock outputs will not.
2	ϕ1T	Phase 1 TTL level clock output.
3	$\overline{\phi 2}$	Phase 2 MOS level clock output. Directly drives all MCS 40 components.
4	V_{DD}	Main Power Supply Pin. $V_{DD} = V_{SS} - 15V \pm 5\%$.
5	MODE	Counter mode control pin. Determines whether counter divides basic frequency by 8 or 7. Mode 1 = V_{SS} Mode 2 = V_{DD}
6	N. OPEN	Input of single step circuitry to which normally open contact of SPDT switch is connected.
7	X1	External Crystal Connection
8	X2	External Crystal Connection
9	N. CLOSED	Input of single step circuitry to which normally closed contact of SPDT switch is connected.
10	ACK	Acknowledge input to single step circuitry normally connected to stop acknowledge output of 4040.
11	$\overline{STOP}$	Stop output of single step circuitry normally connected to stop input of 4040.

12	RESET IN	Input to which RC network is connected to provide power-on reset timing.
13	$\overline{\text{RESET}}$	Reset signal output which directly connects to all MCS 40 reset inputs.
14	$\overline{\phi 1}$	Phase 1 MOS level clock output. Directly drives all MCS 40 clock inputs.
15	V_{SS}	Circuit reference potential – most positive supply voltage.
16	ϕ2T	Phase 1 TTL level clock output.

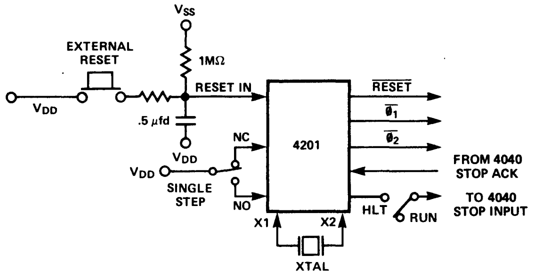


Figure 4-2. 4201 Implementation.

Functional Description

The 4201 consists of the following functional blocks:

CRYSTAL OSCILLATOR

The oscillator is a simple series mode crystal-type circuit consisting of two inverters biased in the active region, and a series crystal element.

PROGRAMMABLE SHIFT REGISTER

The shift register in the 4201 divides the master clock and generates the proper states for generating the desired two-phase clock. The circuit is a seven bit dynamic device which circulates a logical "1" through a field of zeroes. The output of the various stages are then combined to provide the proper clock waveforms.

In order to maintain the proper clock timing over the full operating frequency range of the MCS-40™ system, the shift register is programmable (using mode pin) as either a 7 bit or 4 bit device. When in the 4 bit mode the clock is divided by 2 and the relationship between the phases is equal; that is, ϕ_1 pulse width, ϕ_2 pulse width, ϕ_1 to ϕ_2 and ϕ_2 to ϕ_1 times are all equal.

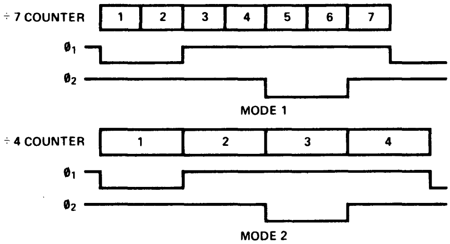


Figure 4-3. 4201 Shift Register Modes.

PHASE DECODER

A simple gate complex is used to decode the shift register outputs to provide phase 1 and phase 2 clock waveforms. This circuitry is controlled by the mode input to achieve the two sets of timing discussed in the previous section.

OUTPUT BUFFERS

There are two sets of output buffers for the 2 phase clock. One set is the MOS level drivers designed to directly drive a full complement of MCS-40 components. The second set provides TTL compatible outputs which can drive one standard TTL load.

RESET CIRCUIT

The reset circuit is simply a level detector and driver stage. An external RC network connected between V_{DD} and V_{SS} at the reset input pin of the 4201 provides the required power-on delay.

A reset pulse can be generated externally by discharging the external capacitor.

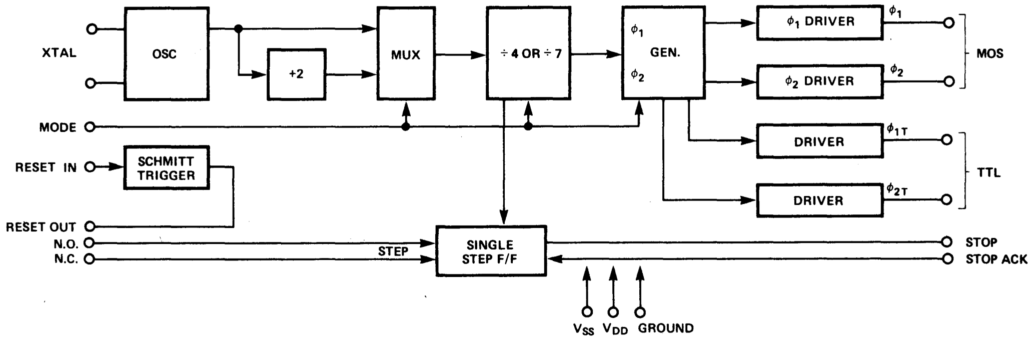


Figure 4-4. 4201 Block Diagram

In addition to driving members of the MCS-40™ system externally, the reset pulse is used internally for initializing the single step circuitry.

SINGLE STEP CONTROL

The 4201 contains the necessary circuitry for allowing the 4040 CPU to execute instructions one at a time. Using the stop input and stop acknowledge output of the 4040, the 4201 generates a pulse that allows the 4040 to perform only one instruction. The stop command can be provided by a SPDT pushbutton directly since debouncing is provided by the 4201. A SPST toggle switch, in series with the ACK line, provides the Run/Halt feature.

SYSTEM COMPONENTS

4001 – 256 X 8 MASK PROGRAMMABLE ROM AND 4 BIT I/O PORT

Introduction

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 4001 is a PMOS device, compatible with all other MCS-40T.M. devices.

Hardware Description

The 4001 is packaged in a 16 pin DIP. The pin configuration is shown in the following figure, and a functional description of each pin is given below.

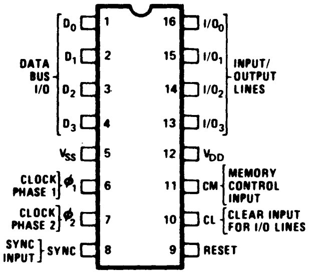


Figure 4-5. 4001 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1-4	D_0-D_3	Bidirectional data bus. All address and data communication between the processor and ROM is handled by these lines.
5	V_{SS}	Circuit GND potential – most positive supply voltage.
6-7	$\phi1, \phi2$	Non-overlapped clock signals which determine device timing.
8	SYNC	System synchronization signal generated by processor.
9	RESET	Reset input. A "1" level on this pin will clear internal flip-flops and buffers. The input buffers are not cleared by this signal.
10	CL	Clear input for I/O lines. A "1" level on this pin will clear this buffer.
11	CM-ROM	Chip enable generated by the processor.
12	V_{DD}	Main supply voltage value must be V_{SS} - 15.0V ± 5%.
13-16	I/O_0-I/O_3	A single I/O port consisting of 4 bidirectional and selectable lines which are selectively compatible to a host of logic families.

Timing Consideration

In the ROM mode of operation the 4001 will receive an 8 bit address during A_1 and A_2 time 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 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.

The 4001 can have a chip number via the metal option of from 0 – 15. The following table shows the chip number relationship between 4308 and 4001.

4308		4001	
Page No.	Chip No.	Page No.	Chip No.
0-3	(0)	0-15	0-15
4-7	(1)		
8-11	(2)		
12-15	(3)		

The above options for the 4308 are the only ones possible hence care should be taken not to have overlapping 4308 and 4001 page numbers when using the devices.

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 $X_2 \cdot \phi_2$ will be latched on the output flip-flops associated with the I/O lines.

Principles of Operation

Address and data are transferred in and out by time multiplexing on 4 data bus lines. Timing is internally generated using two clock signals, ϕ_1 and ϕ_2, and a SYNC signal supplied by the 4004. Addresses are received from the CPU on three time periods following SYNC, and select 1 out of 256 words and 1 out of 16 ROM's. For that purpose, each ROM is identified as #0, 1, 2, through 15, by metal option. A Command Line (CM) is also provided and its scope is to select a ROM bank (group of 16 ROM's).

During the two time periods (M_1 & M_2) following the addressing time, information is transferred from the ROM to the data bus lines.

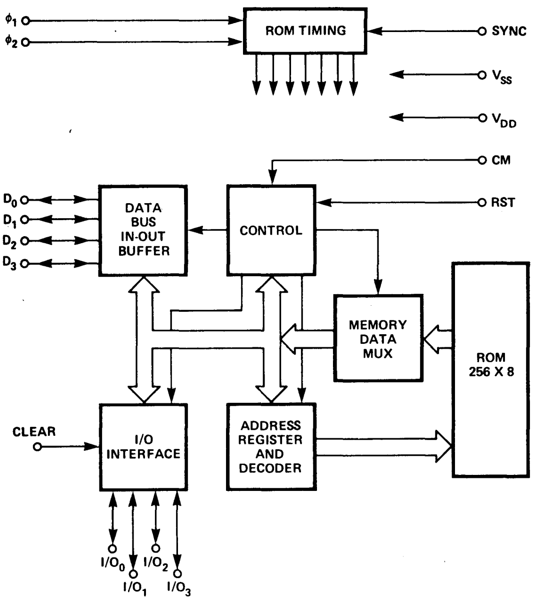


Figure 4-6. 4001 ROM Block Diagram.

A second mode of operation of the ROM is as an Input/Output control device. In that mode a ROM chip will route information to and from data bus lines in and out of 4 I/O external lines. Each chip has the capability to identify itself for an I/O port operation, recognize an I/O port instruction and decide whether it is an Input or an Output operation and execute the instruction. An external signal (CL) will asynchronously clear the output register during normal operation.

All internal flip flops (including the output register) will be reset when the RESET line goes low (negative voltage).

Each I/O pin can be uniquely chosen as either an input or output port by metal option. Direct or inverted input or output is optional. An on-chip resistor at the input pins, connected to either V_{DD} or V_{SS} is also optional.

Instruction Execution

The 4001 responds to the following instructions.

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 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. After an SRC only one CM-ROM and CM-RAM device will be selected.

2. **WRR – Write ROM Port**

   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.) No operation is performed on I/O lines coded as inputs.

3. **RDR – Read ROM Port**

   The data present at the input lines of the previously selected ROM chip is transferred to the accumulator.

   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.

   Given a port 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) 0
Input Data User can choose

I/O Options

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.

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 and blank copy can be found in Section 7.

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

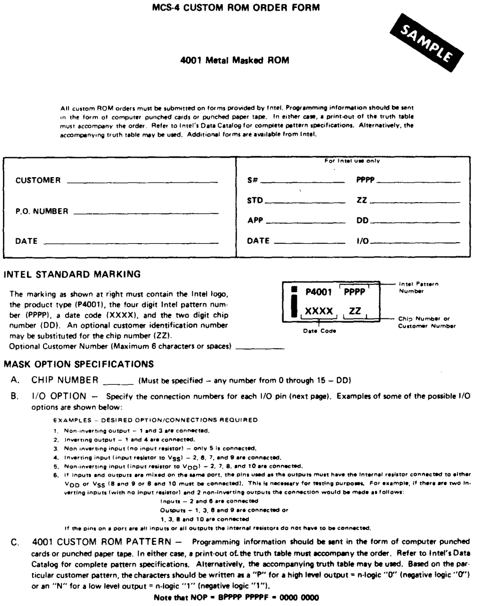
connection would be made as follows:

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.

It should be noted that all internal logic and processing is performed in negative logic, i.e., "1" equals V_{DD} and "0" equals V_{SS}. For positive logic conversion, the inverted options should be selected.

TTL Compatability is obtained by V_{DD} = 10V ± 5% and V_{SS} = 5V ± 5%. An external 12K resistor should be used on all outputs to insure the logic "0" state (V_{OL}).



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 ________
	STD ________	ZZ ________
P.O. NUMBER ____________	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:

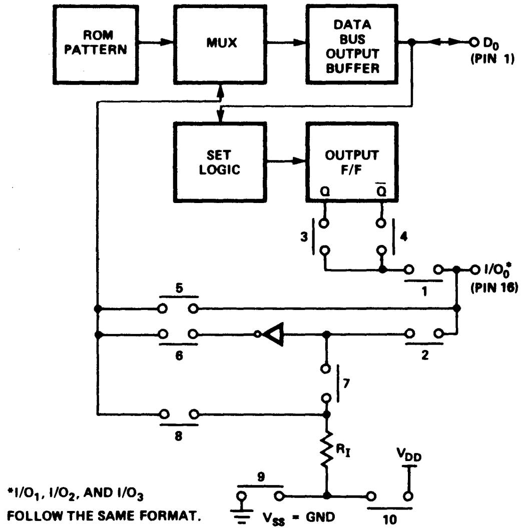
Inputs – 2 and 6 are connected
Outputs – 1, 3, 8 and 9 are connected or
1, 3, 8 and 10 are connected

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

Figure 4-7. 4001 Custom ROM Order Form.



*I/O_1, I/O_2, AND I/O_3 FOLLOW THE SAME FORMAT.

Figure 4-8. 4001 Available Metal Options for Each I/O Pin.

4308 READ ONLY MEMORY

Introduction

The 4308 is a 1024 x 8 bit word ROM memory with four I/O ports. It is designed for the MCS-40™ system, and is operationally compatible with all existing MCS-40 elements. The 4308 is functionally identical to four 4001 chips. It has 16 I/O lines arranged in four groups of four lines. Port selection and accessibility is accomplished as previously, with the 4001. In addition, 4308 has input I/O buffer storage with an optional strobe. A substitution of four 4001 programs can be incorporated in one 4308, including I/O, with no other consideration.

Hardware Description

The 4308 is packaged in a 28 pin DIP. The pin configuration is shown in the following figure and a functional description of each pin is given below.

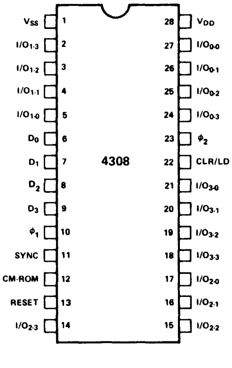


Figure 4-9. 4308 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1	VSS	Circuit GND potential – most positive supply. voltage.
2-5	I/O1$_3$-I/O1$_0$	Four I/O ports consisting of
6-9	D$_0$-D$_3$	Bi-directional data bus all information between processor and device are transmitted to these four pins.
14-17 18-21 24-27	I/O2$_3$-I/O2$_0$ I/O3$_3$-I/O3$_0$ I/O0$_3$-I/O0$_0$	4 bidirectional and selectable lines which are selectively compatible to a host of logic families.
10, 23	$\phi 1, \phi 2$	Non-overlapped clock signals which determine device timing.
11	SYNC	System synchronization signal generated by processor.
12	CM-ROM	Chip enable generated by the processor.
13	RESET	Reset input. A "1" level on this pin will clear internal flip-flops and buffers. The input buffers are not cleared by this signal.
22	CLR/LD	Clear/Load input. When this option is selected, a "0" to "1" state change will cause the output buffer to clear or the input buffer to load the contents of I/O bus. This bus is common to all I/O ports and is TTL compatible.
28	V$_{DD}$	Main supply voltage value must be V$_{SS}$ - 15.0V ± 5%.

Basic Timing

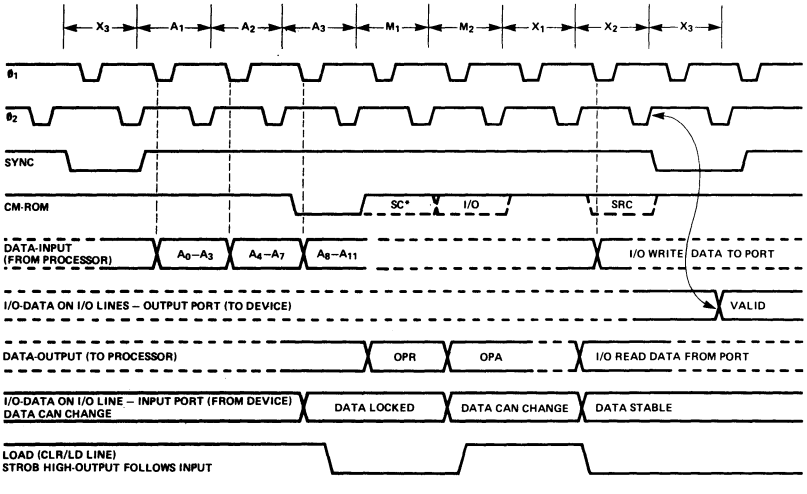
Referring to the timing diagrams, at the beginning of each instruction sequence, a SYNC pulse is generated externally to synchronize the processor with the various components of the system. This pulse, along with the clock inputs ϕ_1 and ϕ_2, is used in the 4308 as an input to a timing register.

During time A1, A2, and A3 the address is sequentially accepted from the data bus and decoded. During time A3, the CM-ROM line will be active, and if the 2 bit (highest order) chip select matches the metal pre-programmed chip select option, the ROM will respond to the current address.

At time M1, M3, the instruction OPR, OPA will be placed on the data bus for the processor. The 4308 responds to 3 I/O instructions in the 4004/4040 instruction set – SRC, RDR, and WRR.

The SRC or Send Register Control instruction is used to designate a set of 4 I/O lines (1 port) on a particular ROM which are to be used for subsequent ROM I/O operations. When this instruction is executed by the processor, the processor sends a 4 bit code to the ROM during X$_2$, and CM-ROM goes to a "1". The first two bits (D$_3$, D$_4$) of this code select a group of 1 out of 4 possible 4308, and the last two bits select a particular port (1 of 4 ports). This port remains selected until the next SRC instruction is executed.

As in the 4001, the WRR (write to ROM port) instruction will cause the contents of the accumulator to be transferred to a previously selected ROM port. As in both the RDR and WRR operations, the CM-ROM line will become active during time M2, and if the ROM has a previously selected I/O port it will respond to the I/O in two ways. For a WRR accumulator, data will be transferred to an internal ROM selected output port flip-flops during X2. Data will be available on the I/O line from time $X3 \cdot \overline{\phi}_2$. The data will remain on the bus until a new WRR occurs, a reset occurs, or a clear (CLR/LD line) is generated. The RDR instruction will transfer information from the input port flip-flops of a previously selected port. Prior to RDR instruction the user should insure that the input flip-flops have been loaded via the CLR/LD strobe if the load strobe is specified. If the load strobe is not specified, information on the input lines will be loaded into the accumulator at the time of the RDR. Input ports or lines are TTL compatible as well as the CLR/LD line. Outputs require external pull-up and +5V, -10V supply.


X3
A1
A2
A3
M1
M2
X1
X2
X3
Ø1
Ø2
SYNC
CM-ROM
SC*
I/O
SRC
DATA-INPUT (FROM PROCESSOR)
A0–A3
A4–A7
A8–A11
I/O WRITE DATA TO PORT
I/O-DATA ON I/O LINES – OUTPUT PORT (TO DEVICE)
VALID
DATA-OUTPUT (TO PROCESSOR)
OPR
OPA
I/O READ DATA FROM PORT
I/O-DATA ON I/O LINE – INPUT PORT (FROM DEVICE) DATA CAN CHANGE
DATA LOCKED
DATA CAN CHANGE
DATA STABLE
LOAD (CLR/LD LINE) STROB HIGH-OUTPUT FOLLOWS INPUT

*REFER TO 4040 SPECIFICATION

Figure 4-10. 4308 External Timing.

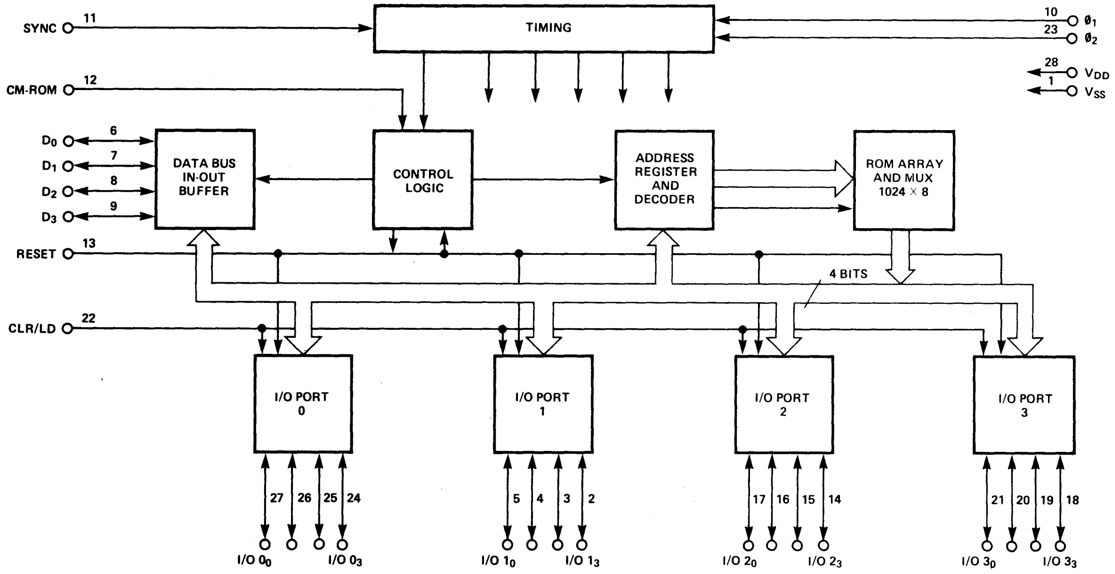


Figure 4-11. 4308 1024 X 8 Bit Mask Programmable ROM and 4 I/O Port, 4 Bits Each.

Principles of Operation

The 4308 ROM memory is arrayed 1024 x 8 bit words. As in the 4001, the A1 – A3 time periods are used to address the ROM contents. The 4308 decodes the first ten bits of the address to select 1 out of 1024 words, 8 bits wide. The remaining two bits select a package number, one of four combinations are possible per ROM bank. The package (chip select) number is selected by a metal mask option. As with the 4001, instruction information is available in two 4-bit segments during M_1 and M_2 time periods. A 4004 system can accommodate up to four 4308's while a 4040 system can utilize up to eight devices. Since the 4308 is compatible with all components of the MCS-40™ system, 4308 and 4001 can be mixed on one memory bank as long as the chip select addresses are mutually exclusive.

As an I/O device, the 4308 ROM recognizes the same class of I/O instructions (SRC, WRR, and RDR) as that of the 4001 ROM. Each of the four I/O ports are program selectable. Each of the four lines can be specified as either inputs or outputs. This is done via the metal mask option. A complete description of the I/O option capabilities are given below. The 4308 has an input storage buffer for utilization with those I/O pins designated as inputs. A common strobe line allows the asynchronous loading of data from the I/O lines. The same strobe line can also serve as a clear to the I/O output port buffers when designated. This line is common to all ports on a 4308 and when toggled, will effect those I/O lines connected by the metal mask option. For an input line, if the strobe line is left unconnected, or if it is pulled up (V_{SS}), then the output of the buffer will follow the input.

I/O Options

The 4308 offers all of the feasibilities found in the 4001. In addition, there are enhancements over the 4001:

1. The user can programmably clear any or all output ports with the CL/LD signal.
2. When operated as in input port, both the inverting and non-inverting inputs paths are TTL compatible.
3. By selecting the LD option, the input buffer becomes an input flip-flop and the CL/LD signal becomes an asynchronous clock for loading data.

Referring to the block diagram of the single I/O pin illustrating the various options, it should be noted that the option numbering differs from that of 4001. Certain pin combinations are mutually exclusive and should not be specified together. There are also certain invalid combinations. The following combinations should be avoided:

8, 9
5, 6
3, 4
10, 11

Examples of some common desired option/connections are:

a. *I/O pin inputs**

non-inverting	11, 2, 5, 7, 9 (TTL) – 2, 5, 7, 8
inverting	11, 2, 6, 7, 9 (TTL) – 2, 6, 7, 8

b. *I/O pin outputs*

non-inverting	3, 1 (10 optional)
inverting	4, 1 (10 optional)

Other combinations exist and should be used with caution.

**NOTE: Option 11 need not be specified if an unbuffered input is desired. This is equivalent to a 4001 input.*

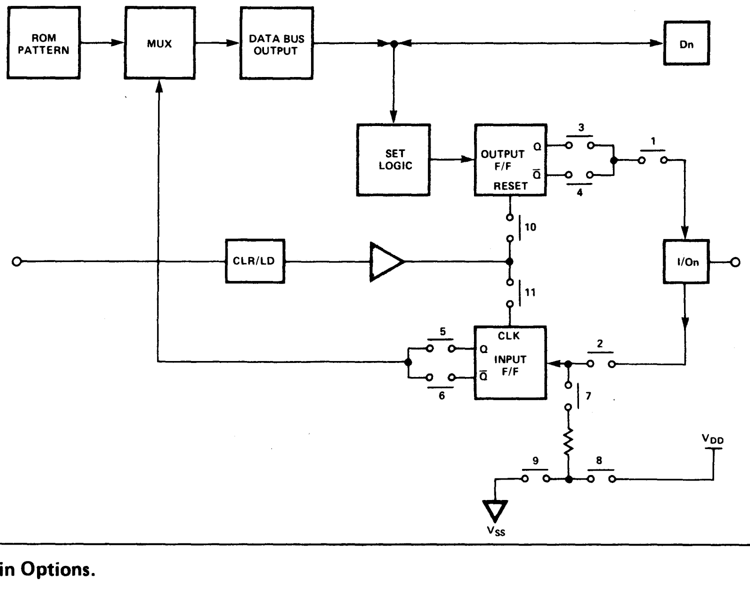


Figure 4-12. 4308 I/O Pin Options.

For TTL compatibility on the I/O lines, the supply voltage should be $V_{DD} = -10V \pm 5\%$, $V_{SS} = +5V \pm 5\%$. External pull-up is required for outputs.

All 4308 metal masked ROM orders must be submitted on forms provided by Intel. Programming information should be provided in the form of computer punched cards or punched tape. In either case, a print-out of the truth table must accompany the order. Refer to the Appendix for complete pattern specification and a sample 4308 ROM form.

intel®

4308 CUSTOM ROM ORDER FORM

4308
ROM

4308 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 the 4308 Data Catalog for complete pattern specifications. Additional forms are available from Intel.

CUSTOMER ______________

ADDRESS ______________

____

P.O. NUMBER ______________

DATE ______________

Intel use

S# ________ PPPP ________

STD ________ ZZ ________

APP ________ DD ________

DATE ________ I/O ________

INTEL STANDARD MARKING

The marking as shown at right must contain the Intel logo, the product type (P4308), 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). ___ ___

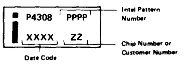
Optional Customer Number (Maximum 6 characters or spaces)

ROM DESCRIPTION

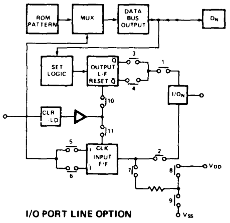
Chip Select Value (0 3) _ _ . . ____ (Must be specified.)

In the table below, select the connections which should be made for each of the 16 I/O port input lines. Avoid the use of illegal options—refer to the 4308 Data Catalog.

Mark the appropriate box for an option connection. Leave blank for a no connection.



PIN		OPTION										
I/O 0_0	27	1	2	3	4	5	6	7	8	9	10	11
I/O 0_1	26	1	2	3	4	5	6	7	8	9	10	11
I/O 0_2	25	1	2	3	4	5	6	7	8	9	10	11
I/O 0_3	24	1	2	3	4	5	6	7	8	9	10	11
I/O 1_0	5	1	2	3	4	5	6	7	8	9	10	11
I/O 1_1	4	1	2	3	4	5	6	7	8	9	10	11
I/O 1_2	3	1	2	3	4	5	6	7	8	9	10	11
I/O 1_3	2	1	2	3	4	5	6	7	8	9	10	11
I/O 2_0	17	1	2	3	4	5	6	7	8	9	10	11
I/O 2_1	16	1	2	3	4	5	6	7	8	9	10	11
I/O 2_2	15	1	2	3	4	5	6	7	8	9	10	11
I/O 2_3	14	1	2	3	4	5	6	7	8	9	10	11
I/O 3_0	21	1	2	3	4	5	6	7	8	9	10	11
I/O 3_1	20	1	2	3	4	5	6	7	8	9	10	11
I/O 3_2	19	1	2	3	4	5	6	7	8	9	10	11
I/O 3_3	18	1	2	3	4	5	6	7	8	9	10	11



I/O PORT LINE OPTION

Figure 4-13. 4308 Custom ROM Order Form.

4001 & 4308 ROM PROGRAMMING INSTRUCTIONS

To insure optimum handling of ROM programs and avoid delays, programs should be specified in the following format.

Paper Tape Format*

A 1" wide paper tape using 8 bit ASCII code, such as a model 33ASR teletype produces:

A. Preamble

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

2. Included in the tape before the leader, and preceded by another leader, should be the customer's complete telex or twx number and if more than one pattern is being transmitted, the ROM pattern number.

3. The first ROM pattern preamble field is the device type number or ROM number. The field should be framed by an "I" and "–"

   I4308– or I4001–

   This should be followed by the chip select information encoded in decimal (two digits), and enclosed by "C" and "S", as in

   "ChhS"

   The valid select digits for the 4308 are 0–3

   "C0S" – "C3S"

   The valid select digits for the 4001 are 0–15

   "C0S" – "C15S"

   Finally, the I/O options would be specified on a port-by-port basis with the connections to be made separated by commas, and enclosed in parentheses:

   "(n1, n2, n3 . . .)".

   where (n1, n2 . . .) are the option numbers associated with one I/O line. Hence, for a 4001 there will be four bracketed collections of I/O options and for the 4308 there will be sixteen.

   Each I/O pin has a series of possible connections*, 11 for 4308 and 10 for 4001. These connections are consecutively numbered from 1=10(11). It is these numbers that should be in parentheses for each I/O pin. *The connection numbers do not correspond for the two devices.

   Example: "( )" indicates no connection
   "(1)" indicates only #1
   "(2,5,7)" indicates connections #2, 5 and 7.

   I/O options should be placed on the tape sequentially for the 4308, from I/O0_0 – I/O3_3(16) and for the 4001 from I/O0 – I/O3(4). Always avoid illegal combinations (Refer to device description).

B. ROM Code

The format requirements are as follows:

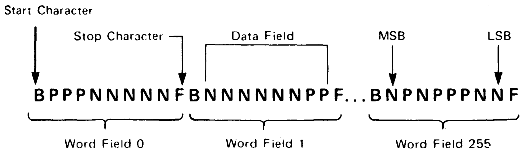
1. All word fields are to be punched in consecutive order, starting with word field 0 (all addresses low). There must be exactly N word fields for the N x 8 ROM organization.

2. Each word field must begin with the start character B and end with the stop character F. There must be exactly 8 data characters between the B and F. Within the word field, a P results in a high

NOTE: Cards may also be submitted.

level output and a N results in a low level output (Positive logic).

Example of 256 x 8 format (N=256):



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

4. Within the ROM pattern words a character, "X", may be used. Where "P" and "N" indicate a "1" and "0" setting, an "X" will indicate a single bit — "Don't Care" setting. This allows the optimum default bit values to be selected by Intel. The bit value will be fixed to allow for testing. The values will be specified to the user on the Verification Listing tape.

In the place of a standard BPNF word, a "B*nF" word may be used. This indicates that the data in the last BPNF word encountered is to be repeated for the next n words ($1 \leq n \leq 1023$). Note that if a repeat count of 4 is given in word position 10, then words 10, 11, 12, and 13 will be repeats of word 9 (except for Don't Care bits which might conceivably have different assigned values).

To indicate that an entire block (such as the remainder of a ROM) is not used (i.e., Don't Care), a word of Don't Care data can be followed by the remaining word count in a repeat count form.

4002 – 320 BIT RAM AND 4 BIT OUTPUT PORT

Introduction

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 4002 is a PMOS device and is compatible with all MCS 40 components.

Hardware Description

The 4002 is packaged in a 16 pin DIP. The pin configuration is shown in the following figure and a functional description of each pin is given below.

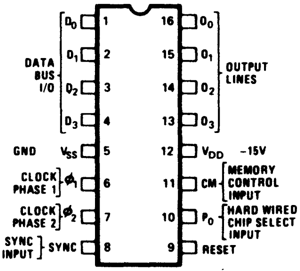


Figure 4-14. 4002 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1-4	D_0-D_3	Bidirectional data bus. All address, instruction and data communication between processor and the RAM MEMORY or the output port is transmitted on these 4 pins.
5	V_{SS}	Circuit GND potential; most positive supply voltage.
6-7	ϕ_1-ϕ_2	Non-overlapping clock signals which are used to generate the basic chip timing.
8	SYNC	Synchronization input signal driven by SYNC output of processor.
9	RESET	RESET input. A logic "1" level 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).
10	Po	For chip selection, the 4002 is available in two metal options, 4002-1 and 4002-2. An external pin, P_0 is also available for chip selection. The chip number is assigned as follows:

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

Pin No.	Designation	Description of Function
11	CM	Command input driven by CM-RAM output of processor. Used for enabling the device during the decoding SRC and instructions.
12	V_{DD}	Main power supply pin. Value must be V_{SS} - 15V ± 5%.
13-16	0_3-0_0	Four bit output port used for transferring data from the CPU to the users system. The outputs are buffered and data remains stable after the port has been loaded. This port can be ınade TTL compatible.

Timing Considerations

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.

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.

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 $X_2 \cdot 0_2$ will set the output flip-flops associated with the I/O pins. That information will be available until next WMP for peripheral devices control.

In the RAM mode, the operation is as follows: When the CPU receives an SRC instruction it will send out the content

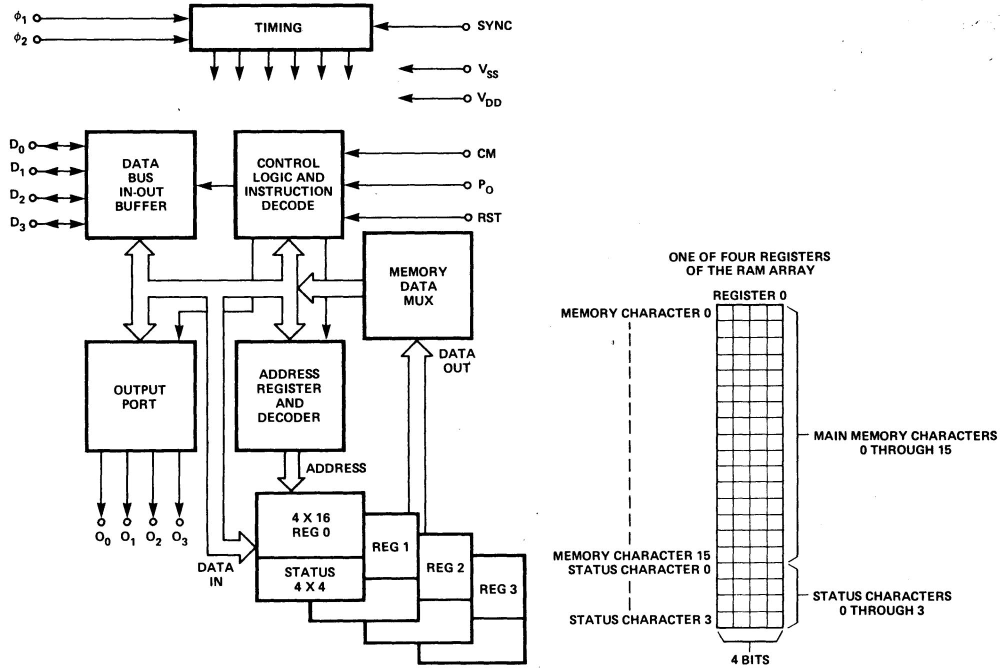


Figure 4-15. 4002 RAM Block Diagram.

of the designated index register pair during X_2 and X_3 and will activate one CM-RAM line at X_2 for the previously (1) selected RAM bank.

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

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

All RAM mode instructions will be executed during the X_2 and X_3. The instruction decoding is performed during the M_2 time when the OPA portion of the instruction is decoded. The CM-RAM of the selected Bank is enabled at that time.

Principles of Operation

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.

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.

The RAM Registers Locations Status Characters and Output Port are select and accessed with a corresponding RAM Instruction. 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 = 1^3 for IOR, D_3 = 0 for IOW.

Instr. Mnem.	4207, 4209,4211, 4308,4001 I/O Oper.	4002 I/O Oper.	4002 RAM Op.	ROM Data Bus Output Buffer Enabled	4002 Data Bus Output Buffer Enabled	CPU Data Bus Output Buffer Enabled
WRM			X			X
WMP		X				X
WRR	X					X
WR0			X			X
WR1			X			X
WR2			X			X
WR3			X			X
SBM			X		X	
RDM			X		X	
RDR	X			X		
ADM			X		X	
RD0			X		X	
RD1			X		X	
RD2			X		X	
RD3			X		X	

There can be up to four RAMS per RAM Bank (CM-RAM). There can be four RAM banks per system without decoding or 8 with decoding.

Bank switching is accomplished by the CPU after receiving a "DCL" (designated 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.

If no DCL is executed prior to SRC, the CM-RAM 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).

Instruction Execution

The following instructions are executed by the 4002.

1. RDM Read RAM character

   The content of the previously selected RAM main memory character is transferred to the accumulator. The 4 bit data in memory is unaffected.

2. RD0-3 Read RAM status characters 0-3

   The 4 bits of status characters 0-3 for the previously selected RAM register are transferred to the accumulator.

3. WRM Write accumulator into RAM character

   The accumulator content is written into the previously selected RAM main memory character location.

4. WR0-3 Write accumulator into RAM status characters 0-3

   The content of the accumulator is written into the RAM status characters 0-3 of the previously selected RAM register.

5. WMP Write memory port

   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.)

6. ADM Add from memory with carry

   The content of the previously selected RAM main memory character is added to the accumulator with carry. The RAM character is unaffected.

7. SBM Subtract from memory with borrow

   The content of the previously selected RAM character is subtracted from the accumulator with borrow. The RAM character is unaffected.

4207, 4209 and 4211 I/O DEVICES

Introduction

The 4207, 4209, 4211 are three PMOS devices designed to greatly expand the I/O capability of the MCS-40T.M. family. These devices are appended to the CPU via the four bit internal data bus on one side, and contain a 16 bit TTL compatible I/O interface on the other side. The above devices replace one 4308 or four 4001's on the CM-ROM line. The chips are designated 4308 ROM address 3. The above devices can also reside on a CM-RAM line along with four 4002s per line.

Hardware Description

The 4207, 4209, and 4211 are packaged in a 28 pin DIP. The pin configurations are shown in the following figures and a functional description of each pin is given below:

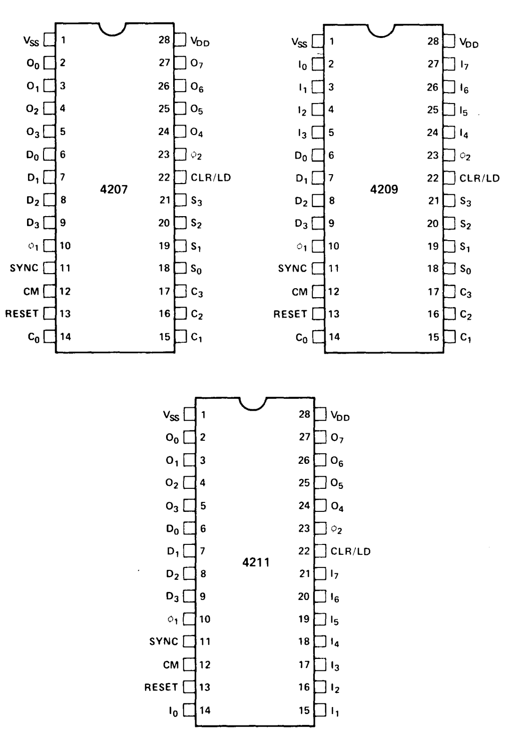


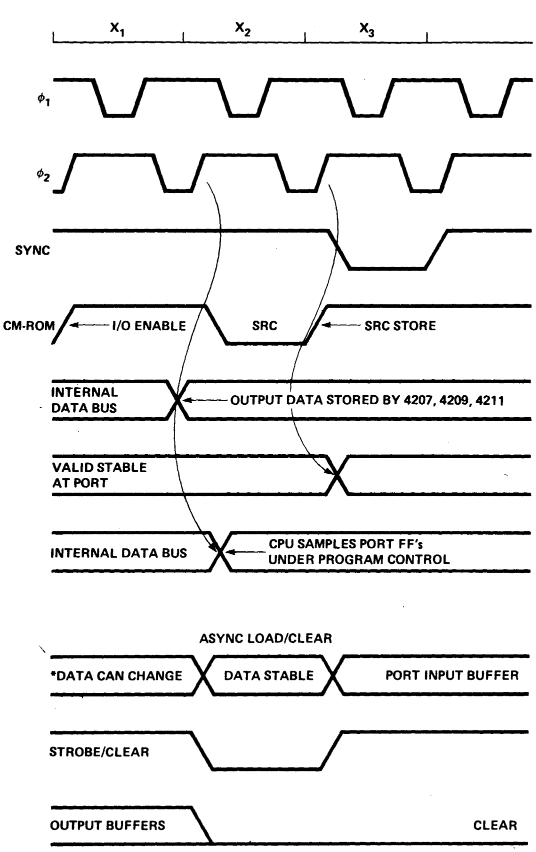
Figure 4-16. 4207, 4209, 4211 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1	V_{SS}	Circuit GND potential – most positive supply voltage (5v for TTL operation)
2-5 24-27	O_0-O_7 I_0-I_7 O_0-O_7	a. 4207 Output ports that can be cleared with CLR/LD line b. 4209 Input ports that can be strobed with CLR/LD line c. 4211 – Output ports
6-9	D_0-D_3	Bi-directional data bus all information between processor and device are transmitted to these four pins.
14-17	C_0-C_3 I_0-I_3	a. 4207, 4209 – Output port used for control functions b. 4211 – Input port; strobe with CLR/LD line
18-21	S_0-S_3 I_4-I_7	a. 4207, 4209 – Input port used for status input b. 4211 – Input port; strobe with CLR/LD line
10, 23	ϕ_1-ϕ_2	Non-overlapped clock signals which determine device timing.
11	SYNC	System synchronization signal generated by the processor.
12	CM	Chip enable generated by the processor.
13	RESET	Reset input; a "1" level on this pin will clear all internal flip-flops and buffers.
22	CLR/LD	Clear/Load Input. A "0" to "1" state change will cause the output buffer to go to a "0" state when this pin is designated as a clear. When this pin is designated a load strobe, "0" to "1" state change will cause the I/O input data to be locked in to the Input buffer. When Load is a "0", the buffer outputs will follow the I/O inputs.
28	V_{DD}	Main supply voltage value must be V_{SS} - 15.0V ± 5% (-10v for TTL operation)

Timing

Referring to the timing diagram, the SRC or Send Register Control instruction is used to designate one of the four I/O ports associated with 4207, 4209, or 4211 to which the subsequent I/O is intended. When an SRC is executed, the



*OUTPUT OF INPUT BUFFERS FOLLOW INPUT LINE WHEN STROBE IS HIGH.

Figure 17. 4207, 4209, 4211 Timing.

processor sends a 4 bit code to the GPI/O during X_2, and CM goes to a "1". The first two bits are always 11, the second two bits select the port of interest.

00 selects port 0
01 selects port 1
10 selects port 2
11 selects port 3

A port remains selected until a new SRC is generated.

The GPI/O responds to the following instructions: SRC, WRR, and RDR. In both the RDR and WRR operations, the CM line will become active during the M2 time. The GPI/O will respond to the I/O in two ways. For a WRR Accumulator data will be transferred to an internal GPI/O selected Output port flip-flops during X2. Data will be available on the I/O line from time $X3-\overline{\phi}_2$. The data will remain on the Output port bus until a new WRR occurs, a reset occurs, or a clear (CLR/LD) line is generated. The RDR instruction will transfer information from the Input port flip-flops of a previously selected port to the accumulator. Prior to the RDR instruction, the user should insure that the input flip-flops have been loaded via the CLR/LD strobe. If the CLR/LD line is connected to V_{SS}, the strobe will be inactive and the flip-flop outputs will follow the inputs. The I/O lines as well as the CLR/LD line are TTL compatible.

Any port designated as an Output port can be interrogated by an RDR. This means that an output buffer can be viewed by the program. When a port is designated an Input Port, a WRR will be ignored.

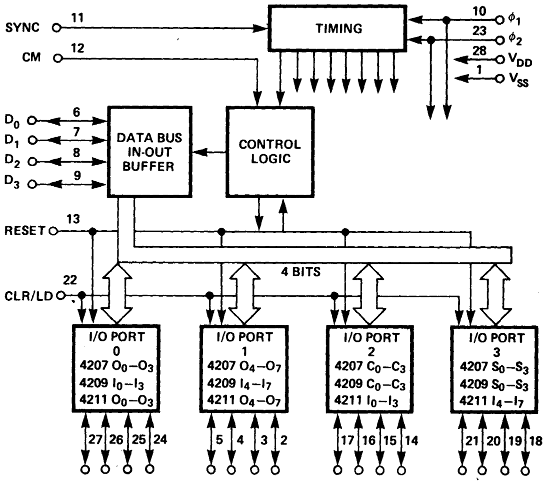


Figure 4-18. 4207,4209,4211 Functional Block Diagram.

In many applications it is desirable to place one GPI/O on the CM-RAM line. This allows the user up to 4 GPI/O per system, one for each DCL operand decode.

Whenever a GPI/O is in the system the 4308 ROM address 3 must be left vacant on the CM-ROM line. Whether the GPI/O is on a CM-RAM or CM-ROM, it will always respond to an SRC, RDR, and WRR command in the same way.

A logical "1" in the CPU accumulator (V_{DD} or negative logic) will be transferred to the I/O port pin as a "1" (V_{SS} or logical 1 positive logic). A positive logic "1" V_{SS} will be transferred to the accumulator as a negative logic "1" (V_{DD}).

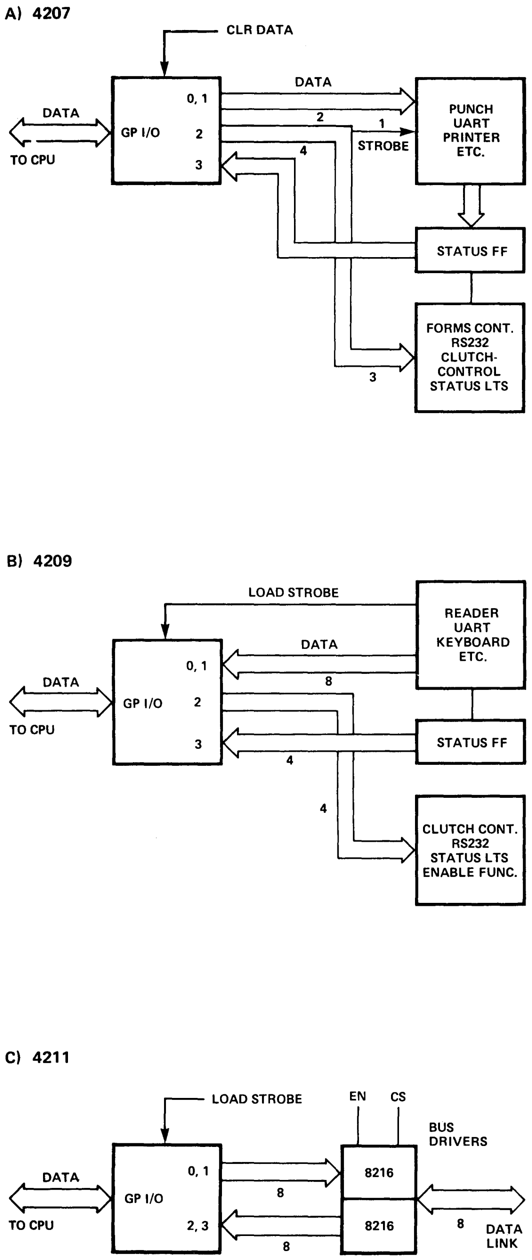


Figure 4-19. Typical GP I/O Applications.

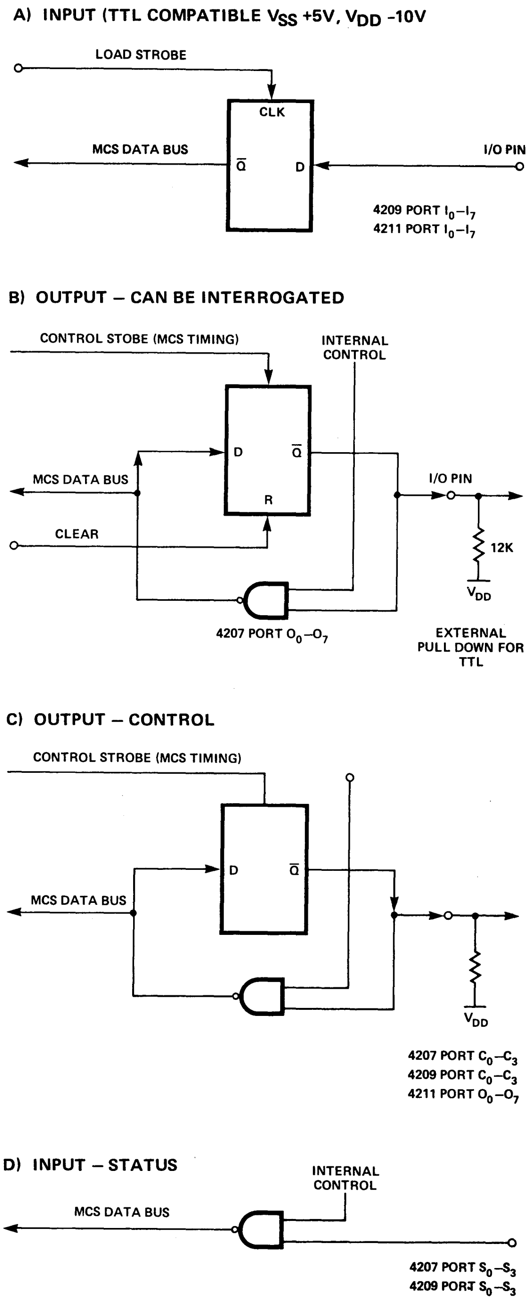


Figure 4-20. 4207, 4209, 4211 Equivalent Circuits.

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

Introduction

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.

Hardware Description

The 4003 is packaged in a 16 pin DIP. The pin configuration is shown in the following figure, and a functional description of each pin is given.

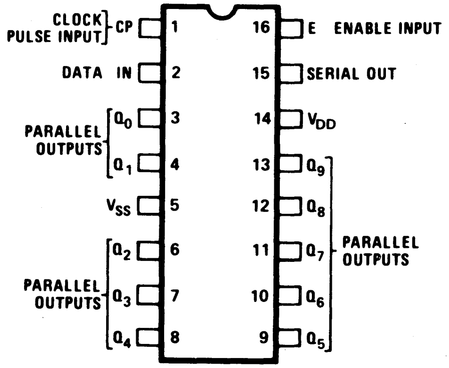


Figure 4-21. 4003 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1	CP	The clock pulse input A"0" to "1" transition will shift data in.
2	DATA IN	Serial data input line.
3	O_0	Parallel data output lines, when enabled.
4	O_1	
6	O_2	
7	O_3	
8	O_4	
9	O_5	
10	O_6	
11	O_7	
12	O_8	
13	O_9	
5	V_{SS}	Circuit GND potential — most positive supply voltage.
14	Serial out	Serial data output.
16	E	Enable, when E="0" the output lines contain valid data. When E="1" the output lines are at V_{SS}.

Timing Considerations

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.

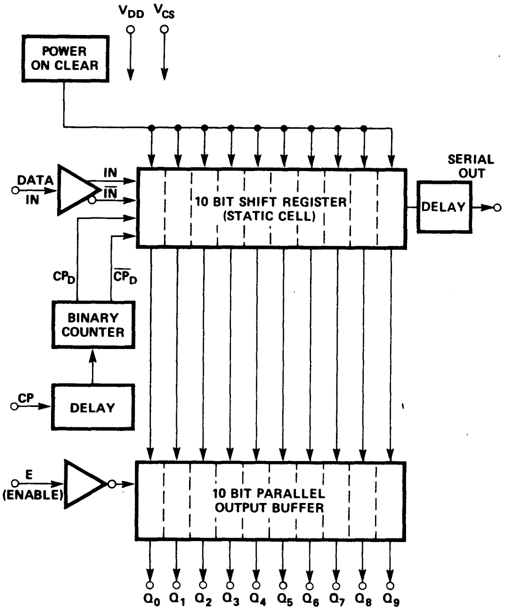


Figure 4-22. 4003 Shift Register Block Diagram.

Functional Description

The 4003 is designed to be typically appended to an MCS 40 I/O port. This can be the I/O port of a 4001, 4002, 4289, 4308, 4207/9/11. One I/O line is assigned to be the Enable (E), another the Clock (CP), and still another the Serial Data-Input. For example, to access the 4003 a subroutine of sequential outputs consisting of Data, clock pulse on, Enable — followed by an output of clock pulse off and Enable, will serially load the 4003.

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 are 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 to allow longer word cascading.

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 ($Qi = V_{SS}$) between the application of the supply voltage and the first CP signal.

The 4003 output buffers are 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.

Another typical application of the 4003 is for Keyboard or Display Scanning where a single bit of Logic " " is shifted through the 4003 and is used to activate the various digits, keyboard rows, etc. (Refer to the interface design for the MCS-40T.M. section.)

4289 STANDARD MEMORY INTERFACE

Introduction

The4289 standard memory interface and I/O interface enables the CPU devices to utilize standard memory components for use as program data memory. Noteably, PROMs (4702A), RAMs (2102) and ROMs can be arranged in a memory array to facilitate system development. Programs generated using the 4289 and interface can be committed to MCS-40$^{T.M.}$ ROMs (4308 and 4001) with no change to software.

The 4289 also contains a 4 bit bi-directional I/O bus and necessary steering logic to multiplex a host of I/O sources to the CPU. The Read and Write Program Memory instruction allows the user to store data and modify program memory. The device directly addresses 4K of program memory. The address is obtained sequentially during A_1-A_3. The eight bit instruction is presented to the CPU during M_1 and M_2 via the four bit data bus.

The 4289 stores the SRC instruction operand as an I/O address and responds to the ROM I/O instructions (WRR and RDR) by reading or writing data to and from the processor and 4289 I/O bus.

Description of Basic Hardware

The 4289 is packaged in a 40 pin DIP. The pin configuration and a brief functional description of each pin is given below:

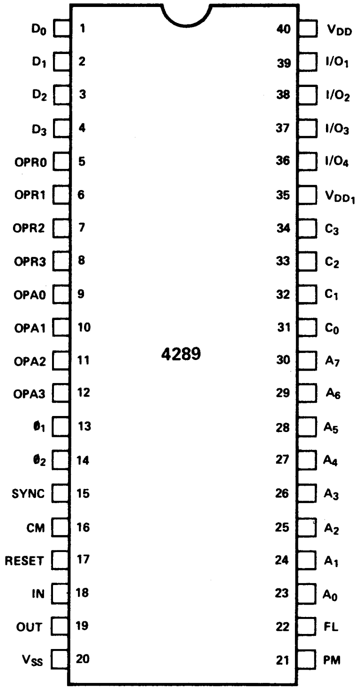


Figure 4-23. 4289 Pin Configuration.

Pin Description

Pin No.	Designation	Description of Function
1-4	D_0-D_3	Bidirectional data bus. All address, instruction and data communication between processor and the PROGRAM MEMORY or I/O ports is transmitted on these 4 pins.
5-8	OPR_0-OPR_3	The high order 4 bits (OPR) of the instruction or data (RPM) from the PROGRAM MEMORY are transferred to the 4289 on these pins.
9-12	OPA_0-OPA_3	The low order 4 bits (OPA) of the instruction or data (RPM) are transferred to the 4289.
13-14	ϕ_1-ϕ_2	Non-overlapping clock signals which are used to generate the basic chip timing.
15	SYNC	Synchronization input signal driven by SYNC output of processor.
16	CM	Command input driven by CM-ROM output of processor. Used for decoding SRC and I/O instructions.
17	RESET	RESET input. A logic "1" level applied to this input resets the CM flip-flop and FIRST/LAST flip-flop.
18	IN	Output signal generated by 4289 when the processor executes an RDR or RPM instruction.
19	OUT	Output signal generated by the 4289 when the processor executes a WRR or WPM instruction.
20	V_{SS}	Circuit Reference potential; most positive supply voltage.
21	PM	Output signal generated by the 4289 when the processor executes an RPM or WPM instruction.
22	F/L	Output signal generated by the 4289 to indicate which half-byte of PROGRAM MEMORY is to be operated on.
23-30	A_0-A_7	Address output buffers. The demultiplexed address values generated by the 4289 from the address data supplied by the processor at A_1 and A_2.

31-34	C_0-C_3	Chip select output buffers. The address data generated by the processor at A_3, or during an SRC are transferred here.
35	V_{DD1}	Supply voltage for address and chip select buffers.
36-39	I/O_0-I/O_3	Bidirectional I/O data bus. Data to and from I/O ports or data to write PROGRAM MEMORY are transferred via these pins.
40	V_{DD}	Main power supply pin. Value must be V_{SS} - 15V ± 5%.

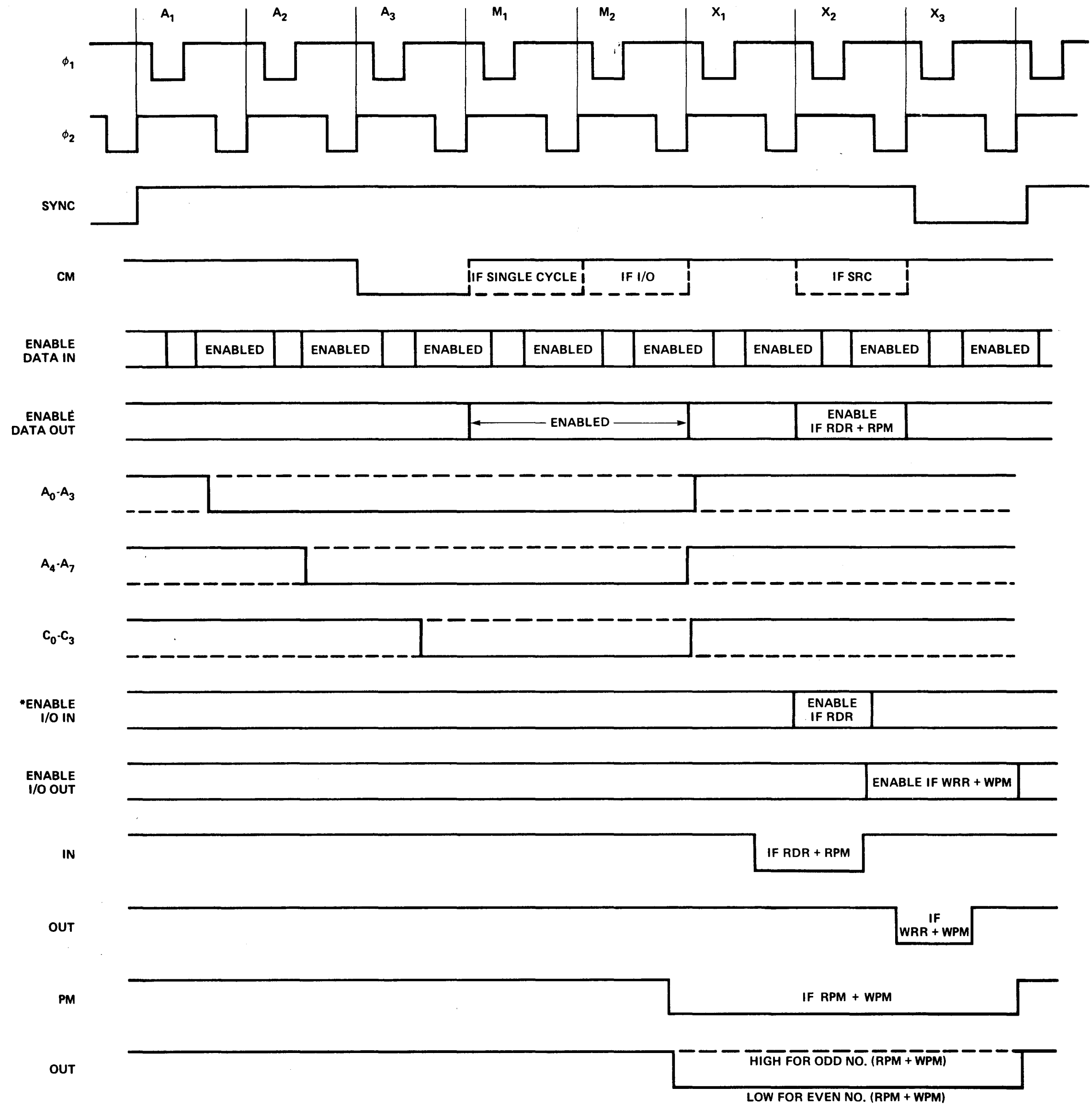


Figure 4-24. 4289 Timing Diagram.

SYSTEM COMPONENTS

Basic Circuit Timing

The basic timing of the 4289 which is generated by the two non-overlapping clock pulses ϕ_1 and ϕ_2. The 4289 is synchronized to the processor by the SYNC signal generated by the processor and sent out at the beginning of each instruction cycle.

During a typical instruction cycle, the 4289 follows the sequence of events outlined below:

a. The device latches the address information sent by the processor at A_1, A_2, and A_3, and presents the 12 bit parallel address on pins A_0-A_7 and C_0-C_3. A_0-A_7 select 1 out of 256 eight bit words and C_0-C_3 enable 1 out of 16 pages of PROGRAM MEMORY.

b. The 8 bit contents of the selected memory location are transferred to the 4289 on pins OPR_0-OPR_3 and OPA_0-OPA_3. They are then multiplexed and transferred to the processor in two 4 bit groups at M_1 and M_2.

c. The 8 bit contents of the internal SRC register are transferred to the ADDRESS and CHIP SELECT buffers at X_1. This value is used as an address for an ensuing I/O or PROGRAM MEMORY operation.

d. The special control signals IN, OUT, PROGRAM MEMORY and FIRST/LAST are generated by the 4289. They are required for performing the I/O and PROGRAM MEMORY operations and directing the information flow.

Major Circuit Blocks

The major functional blocks which make up the chip are:

a. Data bus input-output buffer and multiplex circuitry.

b. Address and chip select output buffers and multiplex circuitry.

c. 8 bit SRC register.

d. I/O input-output buffers.

e. Timing and control logic.

A brief description of each of these major circuit blocks is given below.

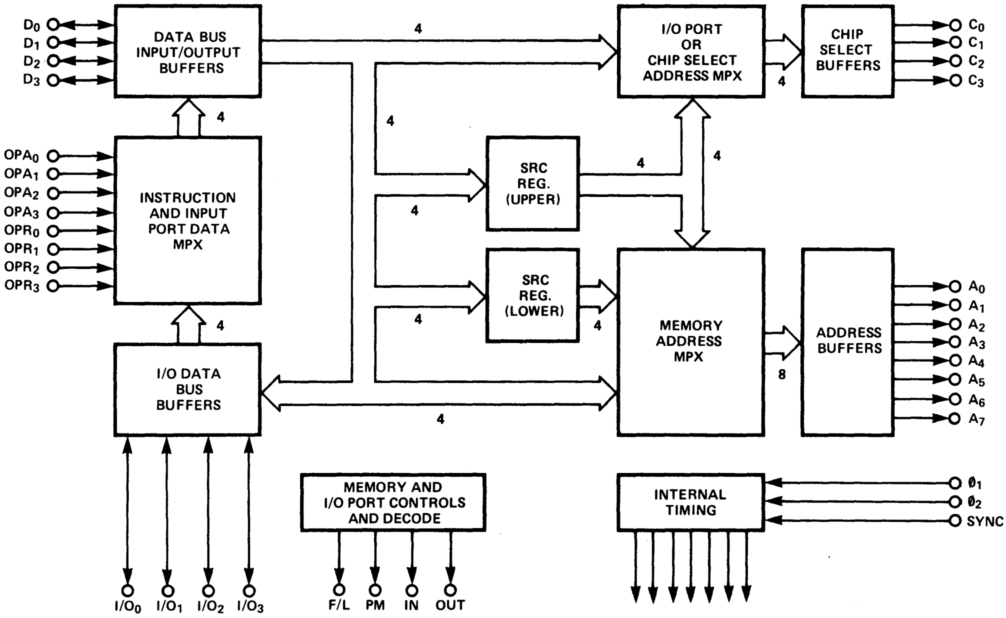


Figure 4-25. 4289 Block Diagram.

Data Bus Input-Output Buffer

The bidirectional data bus buffers provide a communication channel to the processor. The associated multiplex circuitry allows the 4289 to send PROGRAM MEMORY or I/O port information to the processor.

Address and Chip Select Output Buffers

The address and chip select output buffers provide a 12 bit parallel address to the PROGRAM MEMORY for instruction fetch operations. At X_1 the contents of the internal SRC register is transferred to the address and chip select buffers to be used as an address for an I/O or PROGRAM MEMORY operation. The high order 4 bits (SRCRH) are transferred to C_0-C_3 and to A_4-A_7. The low order 4 bits (SRCRL) are transferred to A_0-A_3.

SRC Register

The 8 bit SRC register stores the address value sent out by the processor at X_2 and X_3 of an SRC instruction. The contents of this register are transferred to the address and chip select buffers at X_1 of every instruction cycle, as described above.

I/O Input-Output Buffers

The I/O buffers are used to transfer data between the processor and the input/output ports or from the processor to the PROGRAM MEMORY in the case of a WRITE PROGRAM MEMORY (WPM) instruction.

Timing and Control Circuitry

This includes:

a. The internal timing register.

b. Instruction decoding for the following instructions: WRR, RDR, WPM, RPM.

c. FIRST/LAST flip-flop.

d. The logic required for generating the external control signals, IN, OUT, PM, and F/L.

Detailed Description of Operating Modes

As stated in the Introduction, the 4289 enables the 4 bit microprocessor chip (4004 or 4040) to interface to standard memory components. This allows construction of prototype or small volume systems using electrically programmable ROM's or RAM's in place of 4001 mask programmable ROM's. Since 4001's also contain mask programmable I/O ports, the 4289 has provisions for directly addressing 16 channels of 4 bit I/O ports. In its role as a Memory and I/O interface device, the 4289 provides three different types of operation, namely;

a. Interface to Program Memory for instruction fetch operations.

b. Interface to Input/Output ports for storing or fetching data using WRR, RDR instructions.

c. Interface to R/W Program Memory for storing or fetching data or for program alteration using WPM, RPM instructions. This last feature allows the use of standard R/W RAM to be used for data storage.

These three basic operations will be discussed in detail in the following paragraphs.

Instruction Execution

The contents of the data bus at A_1, A_2, and A_3 are latched by the 4289 and transferred to the address and chip select output buffers. The low order address at A_1 is transferred to A_0-A_3 outputs, the middle order address at A_2 is transferred to A_4-A_7 outputs and the high order address at A_3 is transferred to C_0-C_3 outputs. These 12 output lines provide the necessary address and chip select signals to interface to a 4K x 8 bit Program Memory.

The 8 bit word selected by A_0-A_7 and C_0-C_3 is transferred to the processor via the $OPR_{0\text{-}3}$, $OPA_{0\text{-}3}$ input lines and the data output buffer. The high order bits (OPR) are transferred at M_1 and the low order 4 bits (OPA) are transferred at M_2.

The 4289 has been designed to work equally well with either the 4004 or 4040 processor elements. Since the 4040 is provided with two CM-ROM controls which allow it to directly address up to 8K x 8 bits of Program Memory (4K x 8 bits selected by each CM-ROM control), two 4289's would be required for full memory capability. In this case, one 4289 would be controlled by $CM\text{-}ROM_0$ and the other by $CM\text{-}ROM_1$. The 4289 which receives CM at A_3 would be enabled to transfer data at M_1 and M_2.

It should be noted that the two CM-ROM controls permit the simultaneous use of 4001, 4308, and 4289 in the same system. The ROM's 4001 and 4308 can be mixed and assigned to one CM-ROM control line while a single 4289 can be assigned to the other. Within one CM-ROM control line, 4289, 4001, and 4308 cannot be mixed. The reason being that bus contention will arise between the devices.

I/O Port Operation

When the processor executes an I/O port instruction (WRR or RDR), a previously selected I/O port is enabled to receive or transmit 4 bits of data. In the case of WRR, the selected output port receives the 4 bit contents of the processor accumulator, and in the case of RDR, the selected input port transmits 4 bits of data to the processor accumulator. In either case, the port selection must be made by means of a previous SRC instruction. The 4 bit value sent out at X_2 is used as the port address. The 4289 must therefore be capable of storing the SRC address sent by the processor and presenting that address to the I/O port selection logic. To accomplish this, the 4289 behaves as follows:

a. When the processor executes an SRC instruction, the 4289 stores the address sent out by the processor at X_2 and X_3. The contents of this SRC register are transferred during every X_1 time to the address and chip select are available for subsequent I/O instructions. The high order 4 bits are presented at C_0-C_3 and will select the I/O port.

b. When the processor then executes a WRR instruction, the 4289 latches the data sent out by the processor at X_2 and transfers this data to the I/O output buffer. This buffer is enabled during X_3 and transmits the data to the selected output port. So that external port logic may be enabled to receive the data, the 4289 generates the OUT strobe signal.

c. When the processor executes an RDR instruction, the 4289 generates the IN strobe. This enables the selected input port to transmit its data to the I/O bus, where it is latched by the 4289 and transferred to the processor at X_2.

Read/Write Program Memory Operations

If the 4289 is used in conjunction with the 4040, both the WRITE and READ PROGRAM MEMORY (WPM/RPM) functions are directly available. To accomplish these operations, the following are required:

a. A program memory address.

b. The proper control signals.

c. A means of transmitting the data to be stored or fetched.

The 4289 provides all of these as described in the following paragraphs.

Program Memory Address

The address for an RPM or WPM operation is provided by the 8 bit contents of the SRC register. At X_1 of every instruction cycle this 8 bit value is transferred to the address output buffers A_0-A_7. These addresses will select 1 out of 256 program memory words. During execution of WPM or RPM, the 4289 does not transfer the high order 4 bits of the SRC register to C_0-C_3. Instead, it forces all 4 chip select output buffers to a logic "1" state (positive true logic). If only one page of R/W memory is required the 1111 condition on C_0-C_3 can be used to enable that page. If more than one page is required, an output port will be necessary to store the 1 out of 16 page select address.

Since the program memory is organized as 8 bit words, and since RPM and WPM are transmitting only 4 bit words, it is also necessary to specify either the upper or lower half-byte of program memory.

This is done automatically by the FIRST/LAST flip-flop in the 4289. The state of this flip-flop is used to generate the control signal F/L which determines the proper half-byte of program memory. If F/L is a logic "1" state, the first or lower half-byte is selected, and when F/L is a logic "0", the last or upper half-byte is selected. The user can directly reset the FIRST/LAST flip-flop in the 4289 by applying a RESET signal.

Starting from a "reset" condition the FIRST/LAST flip-flop automatically toggles after executing either an RPM or WPM instruction. Hence, odd numbered program memory operations select the 'first' half-byte and even numbered program memory operations select the 'last' half-byte.

Program Memory Control Signals

When the processor executes either WPM or RPM, the 4289 generates the following control signals:

a. F/L – As indicated above.

b. PM – This output signal is generated whenever a program memory operation is to be performed. This signal allows external logic to differentiate between a program memory operation and an I/O operation.

c. OUT – This strobe signal is generated only during WPM and WRR instructions. The combination of PM and OUT is used as a WRITE ENABLE signal for the program memory.

The system examples shown demonstrate the use of these control signals.

Program Memory Data Paths

When the processor executes the WPM instruction, the 4289 latches the data sent out at X_2 by the processor and transfers it via the I/O output buffers to the I/O bus. The I/O bus must be connected to the data input pins of the R/W memory chips (Refer to the following Applications).

If the processor (4040) executes the RPM instruction, then the entire 8 bit program memory word is transferred to the OPR_0-OPR_3 and OPA_0-OPA_3 inputs of the 4289. Depending on the state of F/L either the first (OPA) or last (OPR) half-byte is automatically selected by the 4289.

Applications

The 4289 can be used to form systems of widely varying complexity. Simple systems containing only one page (256 x 8) of PROGRAM MEMORY and few I/O ports, or more complex systems requiring as many as 32 pages (8K x 8) of memory and 32 I/O ports can readily be implemented. Several examples will be described here.

1. Basic Microcomputer System. This system contains:

   a. 1K x 8 bits of PROGRAM MEMORY (4702A ROM)

   b. 1280 bits of DATA MEMORY (4002 RAM) organized as 16 20-character registers

   c. 4 RAM output ports (4002)

   d. 4 I/O ports

   This system used 3205's to decode the IN and OUT strobes for the I/O ports. Since the I/O buffers of the 4289 can sink one full TTL load, no additional buffering is required.

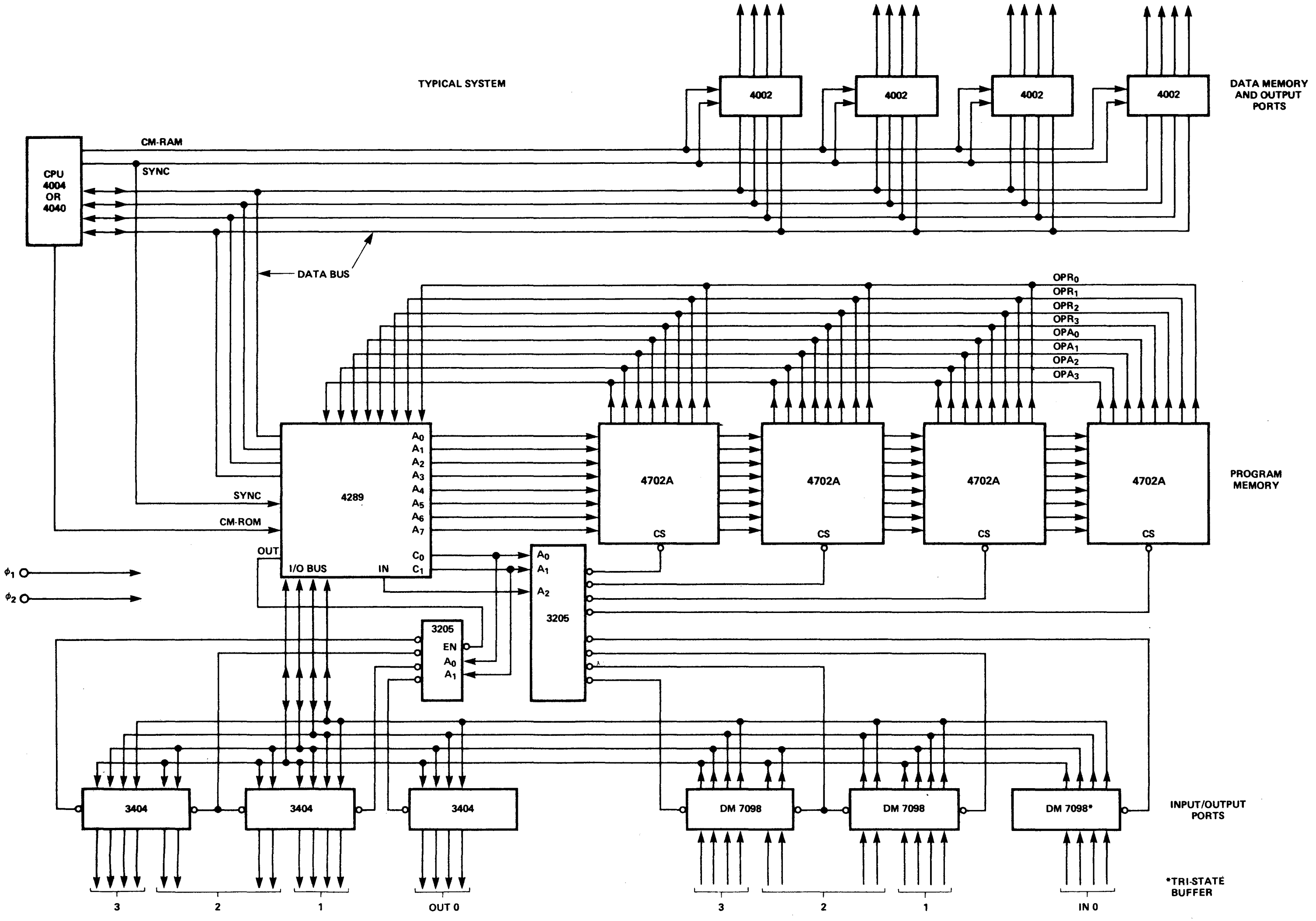


Figure 4-25. 4289 Typical System Implementation.

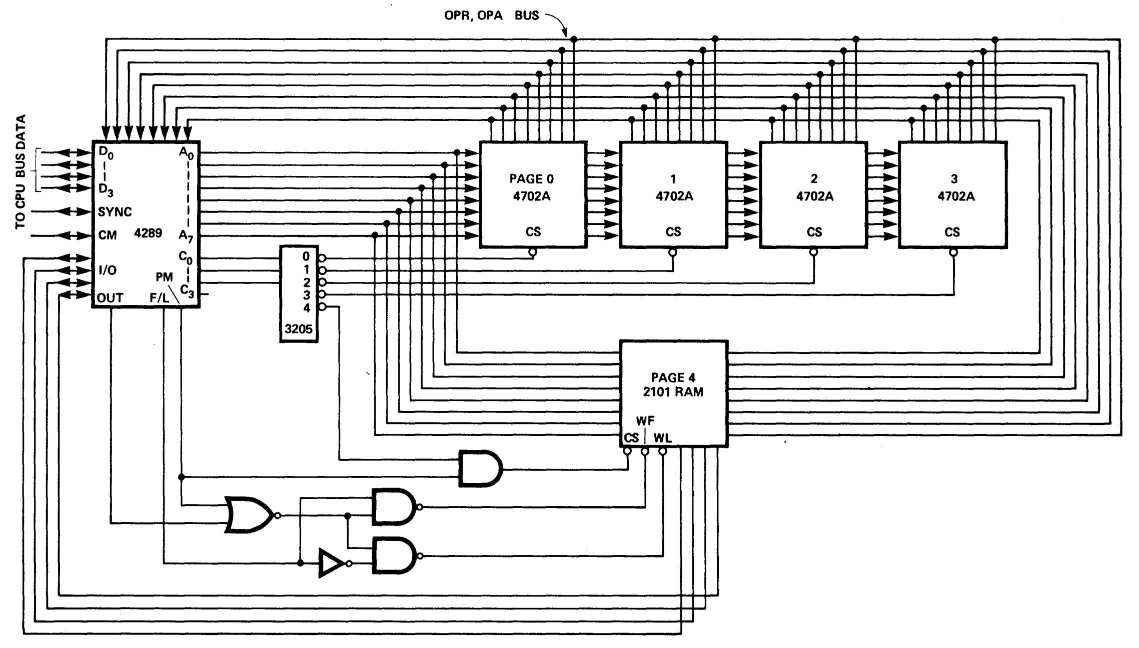


Figure 4-27. RAM and PROM Program Memory Organization.

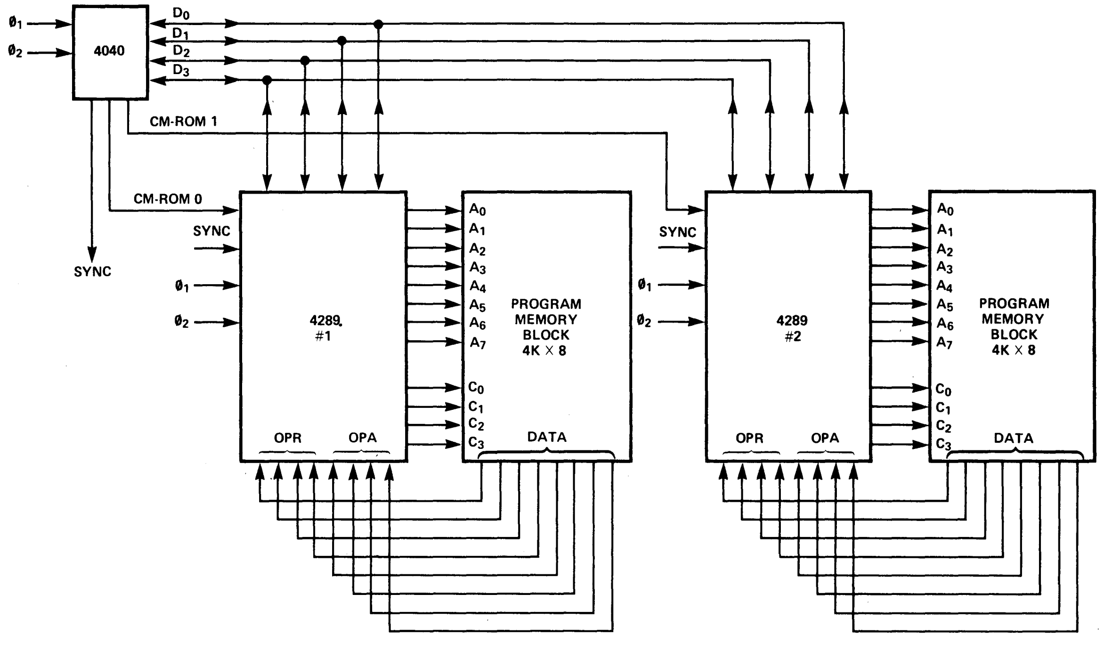


Figure 4-28. Dual Memory Mode.

2. This system again contains 4 pages of PROM storage but, in addition, has one page of RAM storage which can be used for either PROGRAM or DATA storage by using the WPM/RPM instructions. (The RPM instruction is valid only with the 4040). The RAM storage has been implemented with 2101's (256 x 4 static RAM). Notice that separate WRITE ENABLE signals must be generated for the upper and lower half-bytes of RAM.

3. Systems using two 4289's can be designed as shown here. In this case each 4289 is controlled from a separate CM-ROM control signal. The CM-ROM0 and CM-ROM1 lines are generated by the 4040. This system cannot be implemented with the 4004.

As mentioned above, a separate supply pin, V_{DD1}, has been provided for the ADDRESS and CHIP SELECT output buffers. The section on Interfaces shows examples of the use of V_{DD1} when interconnecting the 4289 with other circuit families.

ADDITIONAL PARTS DESIGNED TO BE USED WITH THE 4289 STANDARD MEMORY INTERFACE – 4702A, 4101, 4316, and 3216/3226

4702A – 2048 BIT ERASABLE AND ELECTRICALLY REPROGRAMMABLE READ ONLY MEMORY

- **Access Time – 1.7 μsec Max.**
- **Fast Programming – 2 Minutes for All 2048 Bits**
- **Fully Decoded, 256 x 8 Organization**
- **Static MOS – No Clocks Required**
- **Inputs and Outputs TTL Compatible**
- **Three-State Output – OR-Tie Capability**
- **Simple Memory Expansion Chip Select Input Lead**
- **Compatible with the 4289**

The 4702A is a 256 word by 8 bit electrically programmable ROM ideally suited for microcomputer system development where fast turn-around and pattern experimentation are important. The 4702A undergoes complete programming and functional testing on each bit position prior to shipment, thus insuring 100% programmability.

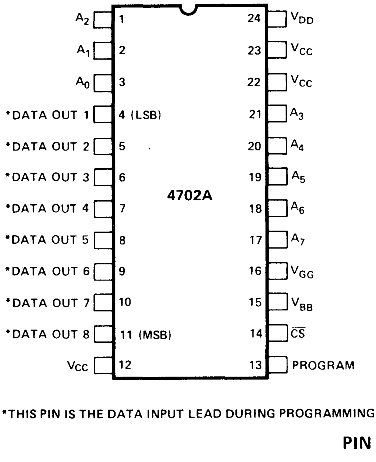
The 4702A is packaged in a 24 pin dual-in line package with a transparent quartz lid. The transparent quartz lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written into the device. This procedure can be repeated as many times as required.

The circuitry of the 4702A is entirely static; no clocks are required.

A pin-for-pin metal mask programmed ROM, the Intel 1302 or 4316, is ideal for large volume production runs of systems initially using the 4702A.

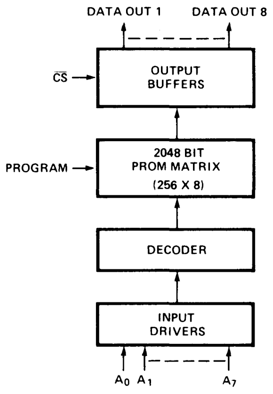
The 4702A is fabricated with silicon gate technology. This low threshold technology allows the design and production of higher performance MOS circuits and provides a higher functional density on a monolithic chip than conventional MOS technologies.

PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD DURING PROGRAMMING.

BLOCK DIAGRAM



PIN NAMES

A_0-A_7	ADDRESS INPUTS
$\overline{CS}$	CHIP SELECT INPUT
DO_1- DO_2	DATA OUTPUTS

4101 – 1024 BIT (256 x 4) STATIC MOS RAM WITH SEPARATE I/O

- 256 x 4 Organization to Meet Needs for Small System Memories
- Access Time – 1 μsec Max.
- Single +5V Supply Voltage
- Directly TTL Compatible – All Inputs and Output
- Statis MOS – No Clocks or Refreshing Required
- Simple Memory Expansion – Chip Enable Input
- Compatible with the 4289
- Inputs Protected – All Inputs Have Protection Against Static Charge
- Low Cost Packaging – 22 Pin Plastic Dual-In-Line Configuration
- Low Power – Typically 150mW
- Three-State Output – OR-Tie Capability
- Output Disable Provided for Ease of Use in Common Data Bus Systems

The Intel 4101 is a 256 word by 4 bit static random access memory element using normally off N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data.

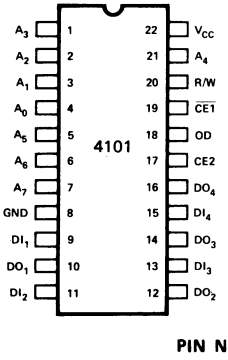
The 4101 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. Two chip-enables allow easy selection of an individual package when outputs are OR-tied. An output disable is provided so that data inputs and outputs can be tied for common I/O systems. Output disable is then used to eliminate any bidirectional logic.

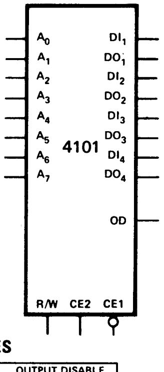
The Intel 4101 is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

Intel's silicon gate technology also provides excellent protection against contamination. This permits the use of low cost silicone packaging.

PIN CONFIGURATION



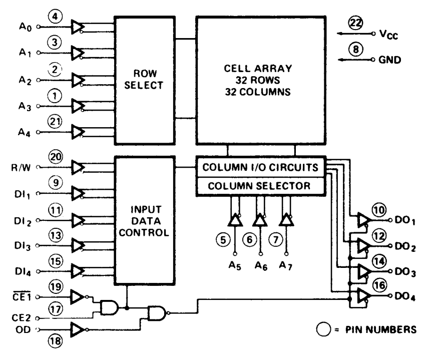
LOGIC SYMBOL



PIN NAMES

D_{IN}	DATA INPUT	OD	OUTPUT DISABLE
$A_0 - A_7$	ADDRESS INPUTS	D_{OUT}	DATA OUTPUT
R/W	READ/WRITE INPUT	V_{CC}	POWER (+5V)
$\overline{CE1}$, CE2	CHIP ENABLE		

BLOCK DIAGRAM



4316 – 16,384 BIT STATIC MOS READ ONLY MEMORY

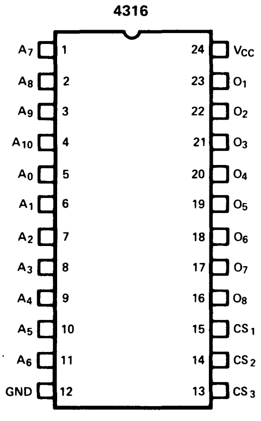
- **Organization – 2048 Words x 8 Bits**
- **Single +5 Volts Power Supply Voltage**
- **Directly TTL Compatible – All Inputs and Outputs**
- **Low Power Dissipation of 10.7 μW/Bit Maximum**
- **Three Programmable Chip Select Inputs for Easy Memory Expansion**
- **Three-State Output – OR-Tie Capability**
- **Fully Decoded – On Chip Address Decode**
- **Inputs Protected – All Inputs Have Protection Against Static Charge**
- **Compatible with the 4289**

The Intel 4316 is a 16,384 bit static MOS read only memory organized as 2048 words by 8 bits. This ROM is designed for microcomputer memory applications where high performance, large bit storage, and simple interfacing are important design objectives.

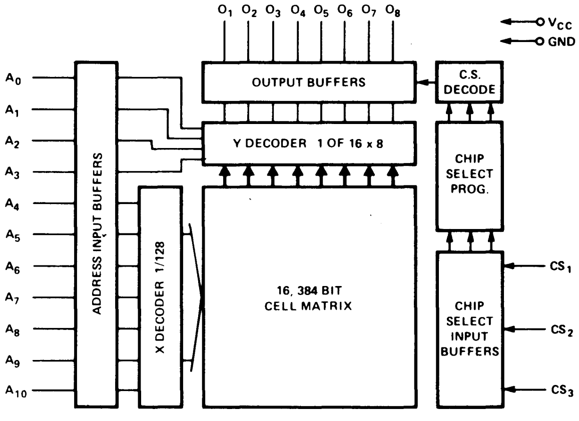
The inputs and outputs are fully TTL compatible. This device operates with a single +5V power supply. The three chip select inputs are programmable. Any combination of active high or low level chip select inputs can be defined and the desired chip select code is fixed during the masking process. These three programmable chip select inputs, as well as OR-tie compatibility on the outputs, facilitate easy memory expansion.

The 4316 read only memory is fabricated with N-channel silicon gate technology. This technology provides the designer with high performance, easy-to-use MOS circuits. Only a single +5V power supply is needed and all devices are directly TTL compatible.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0- A_{10}	ADDRESS INPUTS
O_1- O_8	DATA OUTPUTS
CS_1- CS_3	PROGRAMMABLE CHIP SELECT INPUTS

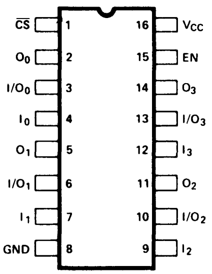
3216/3226* — 4 BIT PARALLEL BIDIRECTIONAL BUS DRIVER

- **Data Bus Buffer Driver for MCS-40 I/O**
- **Low Input Load Current — .25 mA Maximum**
- **High Output Drive Capability for Driving System Data Bus**
- **3.5V Output High Voltage for Direct Interface to MCS-40**
- **Three State Outputs**
- **Reduces System Package Count**

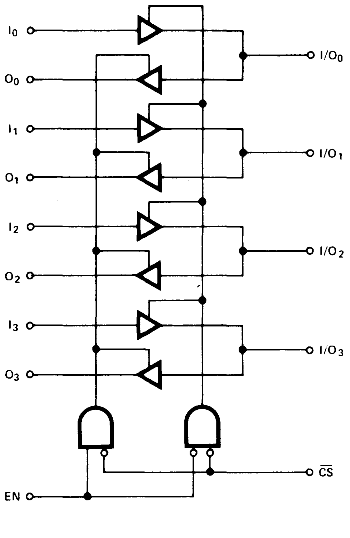
The 3216/3226 is a 4-bit bi-directional bus driver/receiver.

All inputs are low power TTL compatible. For driving MOS, the O outputs provide V_{OH} (3.5V), and for high capacitance terminated bus structures, the I/O outputs provide a higher I_{OL} (25 mA) capability.

PIN CONFIGURATION

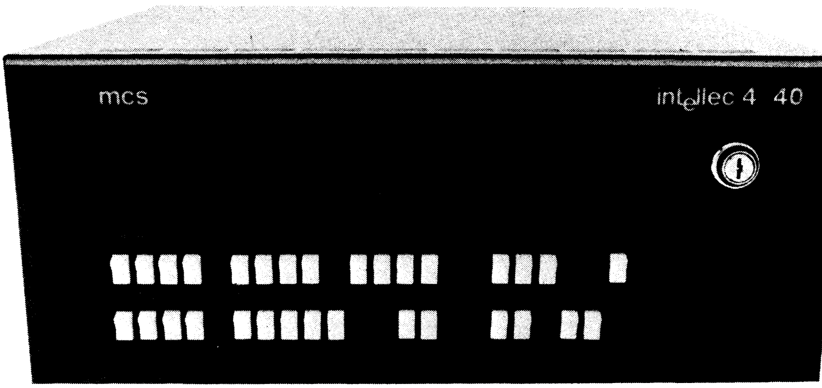


LOGIC DIAGRAM



*The 3226 provides inverted I/O.

CHAPTER 5
DEVELOPMENT AIDS



INTELLEC® 4 /MOD 40 MICROCOMPUTER DEVELOPMENT SYSTEM

- Complete hardware/software development system for the design and implementation of 4040 CPU based microcomputer systems.
- TTY interface, front panel designer's console, and high speed paper tape reader interface, in conjunction with PROM resident system monitor provide complete program loading, punching, monitoring, interrogation, and alteration capabilities.
- Program RAM (4K 8 bit bytes) provides a program development medium which lends itself to rapid and facile program monitoring and alteration.
- Data RAM (320 4 bit bytes expandable to 2560 bytes) provides data storage capacity.
- Program PROM (expandable to 4K 8 bit bytes) in conjunction with the resident PROM programmer provide capability of simulating final ROM resident program.
- PROM resident system monitor and RAM resident macroassembler included in standard systems software.
- Includes such standard program development features as program single step, address search (and pass count), next instruction indication, program flow verification.
- I/O expandable to 16 4 bit input ports and 48 4 bit output ports (all TTL compatible) allowing "hands-on" simulation of entire user system (processor and peripheral devices).
- RESET, STOP, INTERRUPT control signals available to user via back panel.
- Modular design with expansion capability provided for up to eleven optional or user designed modules.

The Intellec 4/MOD 40 (imm 4-44A) system is a complete, self-contained microcomputer development system designed specifically to support the development and implementation of 4040 CPU based microcomputer systems. Its modular design provides the flexibility to adapt to any size user system and the resident software greatly facilitates program development.

The basic Intellec 4/MOD 40 system consists of a 4 microcomputer modules (CPU, RAM, MEMORY CONTROL, and PROM PROGRAMMER), power supplies, I/O connectors, console, and displays. The heart of the system is the imm 4-43 central processor module built around Intel's high performance 4 bit 4040 CPU on a single chip. The imm 4-43 is a complete microcomputer system containing the system clock, 1K 8 bit bytes of PROM memory, 320 4 bit bytes of data RAM memory, 3 4 bit input ports and 8 4 bit output ports. The imm 6-28 program RAM memory module contains a 4K x 8 memory array composed of Intel 2102 static random access memory elements. The imm 4-72 control module contains the circuitry required to interface the central processor module to the program RAM module. The imm 6-76 PROM programmer module provides the capability of programming Intel 1702A PROMs in conjunction with the front panel PROM socket and system monitor. All I/O ports are TTL compatible and accessible from the back panel 37 pin connectors. The front panel designer's console provides a means of monitoring and controlling system operation.

The modular design of the Intellec allows great design system flexibility. Program PROM can be expanded to 4K 8 bit bytes using imm 6-26 or imm 4-22 optional modules. Data RAM can be expanded to 2560 4 bit bytes using imm 4-24 modules. I/O capability can be expanded to 16 4 bit input and 48 4 bit output ports using optional imm 4-60 modules. The universal prototype card (imm 6-70) in conjunction with the eleven optional card sockets (which contain all essential system signals) provide the capability for interfacing custom designed modules.

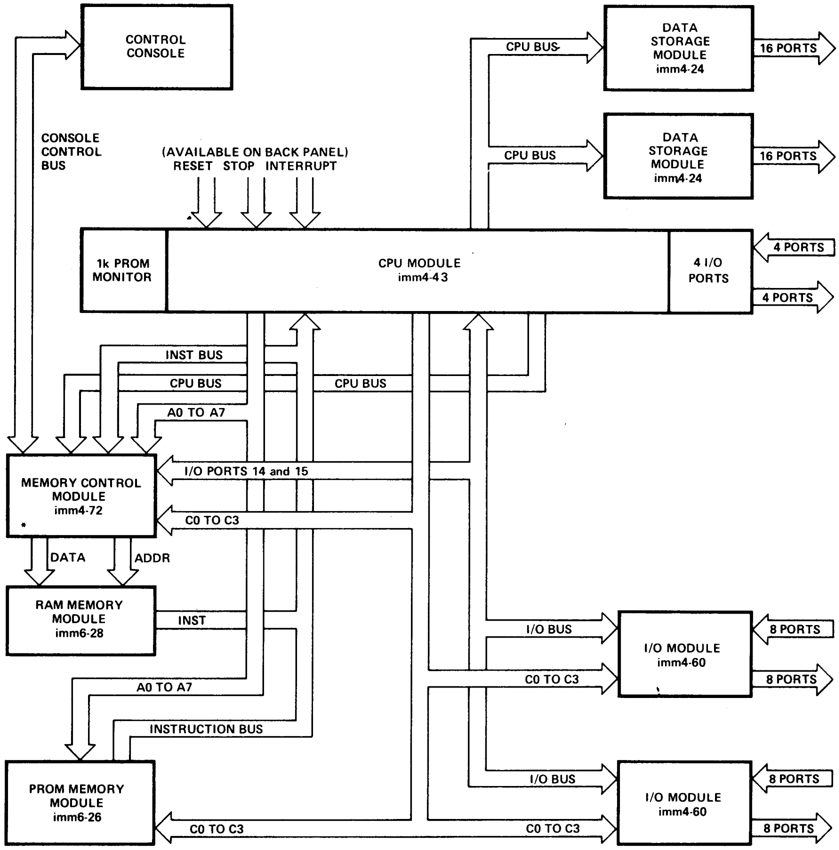
The user RESET IN/OUT, STOP/STOP ACKNOWLEDGE, and INTERRUPT/INTERRUPT ACKNOWLEDGE control signals are all available at the back panel. Hence, the user can interrupt, halt, and reset the resident CPU via his own interface.

Program interrogation and alteration can be accomplished by using any desired combination of the front panel designer's console, a teletype, the imm 4-90 high speed paper tape reader, and other Intellec® compatible peripherals. The front panel designer's console provides the capability of manually writing data into memory and displaying memory contents, monitoring CPU bus contents during each processor subcycle, "freezing" system status after execution of a predefined instruction after a specified number of passes, single-stepping the program and verifying program flow. The teletype and reader serve as vehicles to input and output paper tapes and execute the system monitor.

Every Intellec® 4/MOD 40 system comes with two systems software products — the PROM resident system monitor and the RAM resident assembler. The assembler has a paper-tape editor feature. The systems software is a powerful application program development tool.

The system monitor provides the capability of displaying and modifying memory contents, reading and punching object tapes, dynamically assigning system peripherals, programming and verifying PROMs, and performing other functions which significantly reduce program debug and development time.

The Intellec 4/MOD 40 RAM resident assembler translates source code into object code which will execute on the Intellec 4/MOD 40 or any MCS-40 T.M. system. The assembler collects information from the source program, builds an internal symbol table, outputs a listing of the assembled program including error messages, and punches an object program tape.



*Memory control module selects MONITOR, PROM or RAM for EXECUTION.

Figure 5-1. System Block Diagram.

Specifications

Word Size

Data: 4 bits
Instructions: 8 bits/16 bits

Memory Size

5K bytes expandable to 12K bytes (combination of PROM, Data RAM, Program RAM) in three 4K byte memories selectable for execution from the front panel.

Instruction Set

60; including conditionals, binary and decimal arithmetic, and I/O

Machine Cycle Time

10.8 microseconds

System Clock

Crystal-controlled at nominal 5.185 MHz

I/O Channels

All ports are 4-line TTL
3 input ports expandable to 16
8 output ports expandable to 48

Interrupt

Available at back panel

Console Memory Access

Standard via control console

Memory Access Time

1 μs with standard memory modules

Environmental Characteristics

Operating temperature: 0°C to 55°C

Electrical Characteristics

DC power supplies: $V_{CC} = 5V \pm 5\%$
$I_{CC} = 12A$
$V_{DD} = -10V \pm 5\%$
$I_{DD} = 1.8A$

AC power supplies: Mod 40: 60 Hz, 115 VAC @ 200 W
Mod 40/220: 50 Hz, 230 VAC @ 200 W

Physical Characteristics

Intellec 4/40: 7" x 17 1/8" x 12 1/4" (table top only; optional rack mount available)
Weight: 30 lb. (13.61 kg.)

Optional Modules

Available for the Intellec 4/MOD 40:

imm 4-22 Instruction/Data Storage Module
imm 4-24 Data Storage Module
imm 4-60 Input/Output Module
imm 6-26 PROM Memory Module
RAM Memory Modules (Additional)
imm 6-36 Rack Mounting Kit
imm 6-70 Universal Prototype Module
imm 6-72 Module Extender

Equipment Supplied

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

Complete Hardware Software

Programmers Manual
Operators Manual
Hardware Reference Manual
Module Schematics

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$^{T.M.}$ users. It was designed as an MCS-4 development tool and for convenient field service of microcomputer systems. 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).

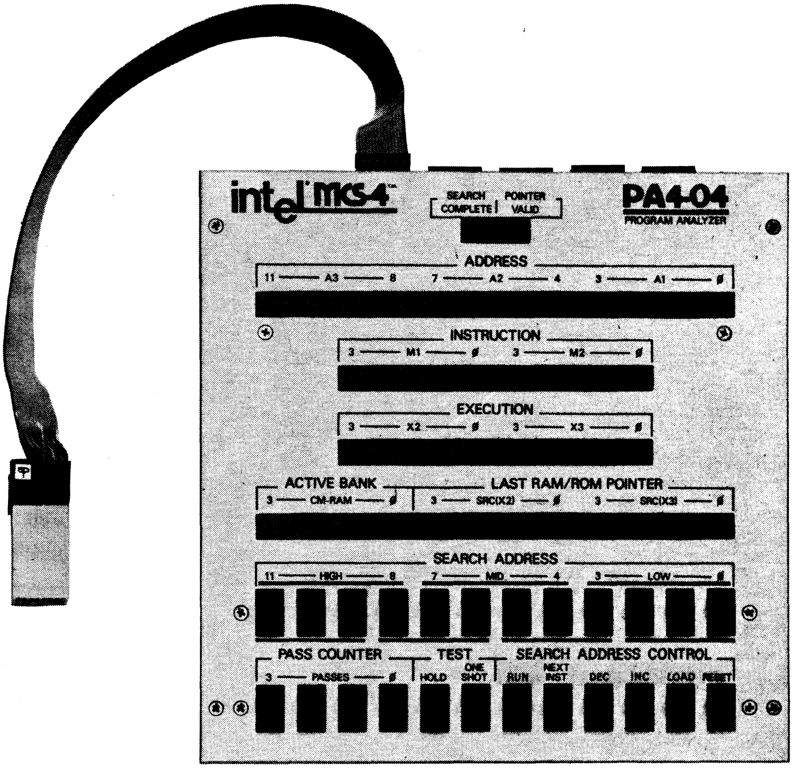
While the display of the search address is latched, the next instruction can be examined by hitting the NEXT INSTRUCTION switch. Pushing the INCREMENT button will increment the program one more count and this can be continued indefinitely. The previous instruction can be examined by using the DECREMENT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



MCS-40$^{T.M.}$ SOFTWARE SYSTEM DEVELOPMENT TOOLS

Programming for the MCS-40 microcomputer can be done easily and quickly using Intel's new cross macro assembler, MAC4. This powerful assembler translates three letter mnemonics representing each MCS-40 instruction into a numeric representation that may be loaded directly into an Intellec 4 development system or programmed to ROM. Advanced MAC4 features provide full macro capability and conditional assembly capability. All output is in hexadecimal for easy interpretation.

MAC4 is written in ANSI standard FORTRAN IV and is designed to run on any large scale computer system (32 bit word size or larger) with little or no modification. The FORTRAN source program for MAC4 is available on magnetic tape directly from Intel. In addition MAC4 may be used on either TYMSHARE, UNITED COMPUTING SYSTEMS, or GENERAL ELECTRIC worldwide timesharing services and may also be used on TIMESHARING LTD in Europe. Contact these services directly for further information.

MCS-40 USER'S LIBRARY

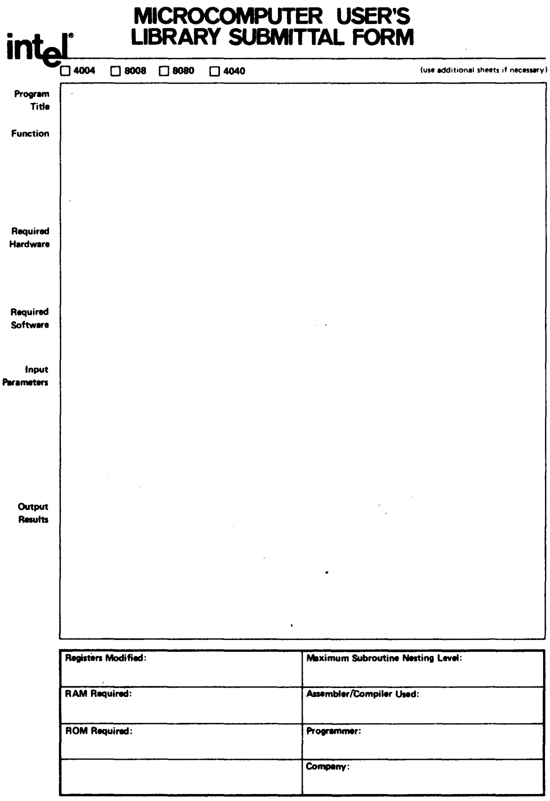
The MCS-40 User's Library is a collection of programs written by users of the 4004 and 4040 CPU chips. These programs have been contributed to the user's library for the benefit of all MCS-40 users. Intel will make source listings of all programs and detailed instructions on their use available to all members of the MCS-40 User's Library. To become a member simply:

1. Submit a program to the library with detailed documentation and a completed user's library submittal form, or
2. Pay a yearly membership fee.

For more information, contact your local Intel representative. Some of the current programs in the library include:

- Cross Assembler for PDP 8
- BNPF Tape Generator for PDP 8
- MCS-40 Simulator for PDP 8
- Chebyshev Approximation Functions
  64 bit Addition, Subtraction, Multiplication, Division
  SIN
  COS
  EXP
  LOG
  TAN
- Parity Checker/Generator
- Parity Generator, ASCII Character
- ASCII to EBCDIC Code Conversion
- Delay Subroutines
- Cross Assembler for NOVA
- Bit Manipulation Routine

(User's Library submittal forms are in back of manual.)



intel®

MICROCOMPUTER USER'S
LIBRARY SUBMITTAL FORM

□ 4004 □ 8008 □ 8080 □ 4040 (use additional sheets if necessary)

Program Title

Function

Required Hardware

Required Software

Input Parameters

Output Results

Registers Modified:	Maximum Subroutine Nesting Level:
RAM Required:	Assembler/Compiler Used:
ROM Required:	Programmer:
	Company:

Figure 5-2. Microcomputer User's Library Submittal Form.

DEVELOPMENT

CHAPTER 6
FAMILY COMPONENT CHARACTERISTICS

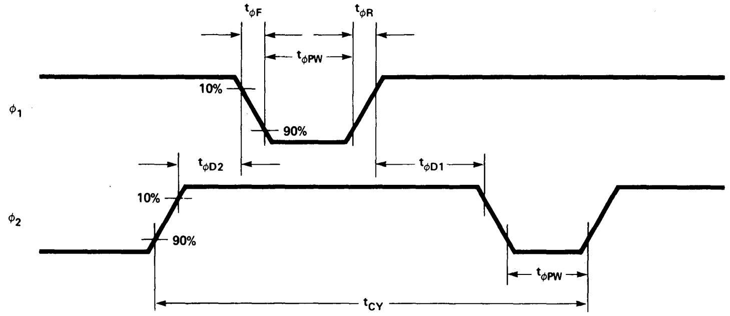
Detailed characteristics on all MCS-40 components will be supplied during the first quarter of 1975.

4207, 4209, 4211, 4308, 4289, 4040, 4001, 4002, 4004 GENERAL SYSTEM CHARACTERISTICS

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

Product	Symbol	Test	Limit		Unit	Conditions
			Min.	Max.		
MCS 40 All	t_{CY}	Clock Period	1.35	2	μs	
	$t_{\phi R}$ $t_{\phi F}$	Clock Rise and Fall Times		50	ns	
	$t_{\phi PW}$	Clock Width	380	480	ns	
	$t_{\phi D1}$	Clock Delay From ϕ_1 to ϕ_2	400	550	ns	
	$t_{\phi D2}$	Clock Delay From ϕ_2 to ϕ_1	150		ns	

TIMING DIAGRAM



* V_{DD} = V_{SS} -15V ±5%. For TTL compatability use V_{DD} = -10V, V_{SS} = 5V.

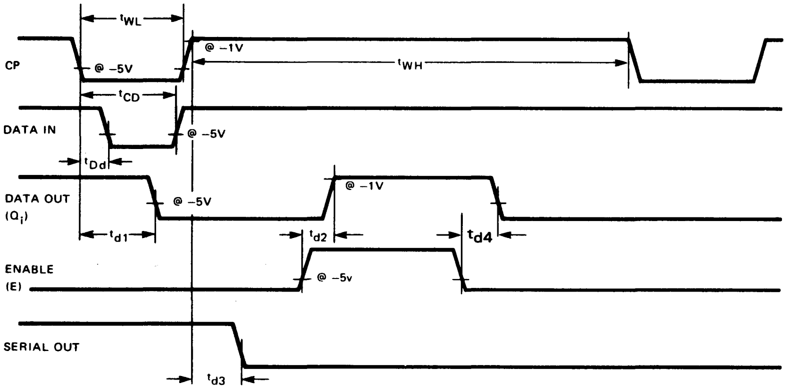
4003 A.C. Characteristics

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
t_{d4}	ENABLE TO DATA OUT DELAY		1.0	μsec	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.

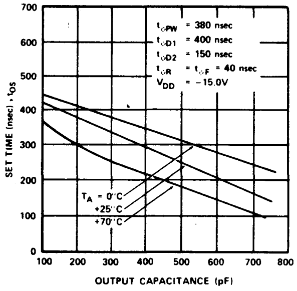
PRODUCT	SYMBOL	TEST	LIMIT (pF) TYP.	LIMIT (pF) MAX.
4001/2/3/4	C_{IN}	INPUT(1) CAPACITANCE	5	10
4001/2	$C_{\phi1}, C_{\phi2}$	CLOCK INPUT CAPACITANCE	8	15
4004	$C_{\phi1}, C_{\phi2}$	CLOCK INPUT CAPACITANCE	14	20

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

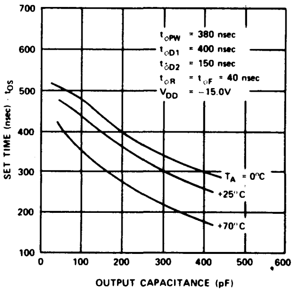
NOTE: (1) Refers to all input pins except data bus I/O and ϕ_1 and ϕ_2.

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)



CHAPTER 7 ORDERING INFORMATION

COMPONENT AVAILABILITY

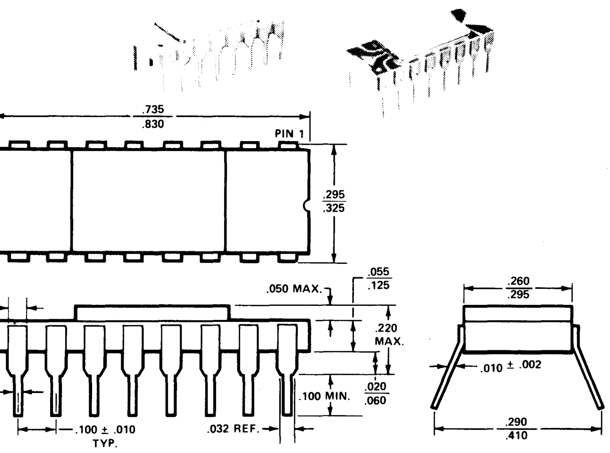
DEVICE	LEADS
4040	24
4004	16
4308	28
4001	16
4002-1	16
4002-2	16
4201	16
4003	16
4207	28
4209	28
4211	28
4289	40
4702A	24
4316	24
4101	22
3216/3226	16

Refer to the above description when ordering.

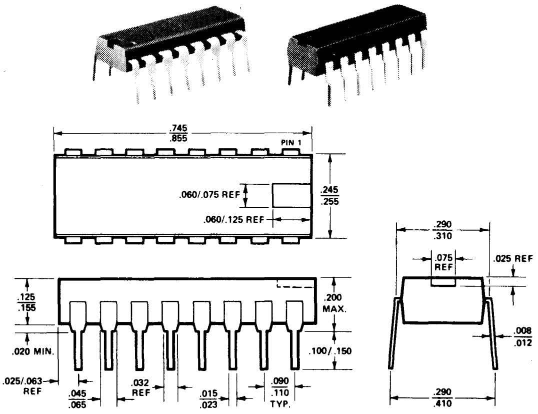
When ordering ROMs 4308 and 4001, the chip number, I/O options and custom pattern information must be contained on input tape or cards. The chip number and I/O options must be duplicated on the ROM order form. Hand written custom pattern truth tables will not be accepted. All input must be in the BPNF input format, on tape or cards.

PACKAGING INFORMATION

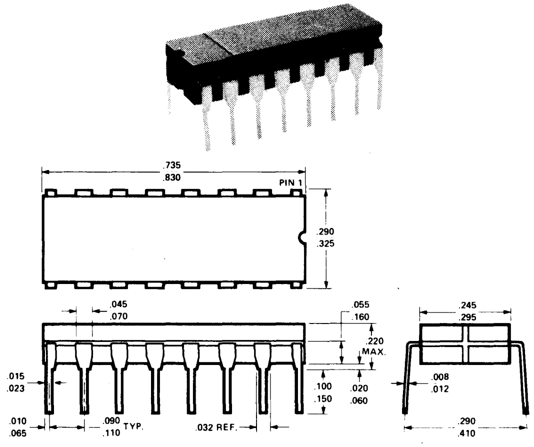
16-LEAD CERAMIC DUAL IN-LINE PACKAGE OUTLINE



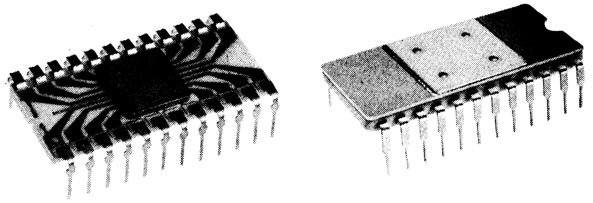
16-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE

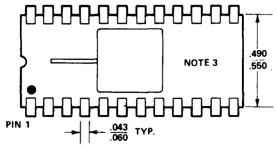


16-LEAD CerDIP (GLASS SEAL) PACKAGE OUTLINE (D)



24-LEAD CERAMIC DUAL IN-LINE PACKAGE OUTLINE (C)

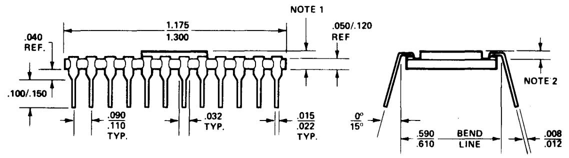




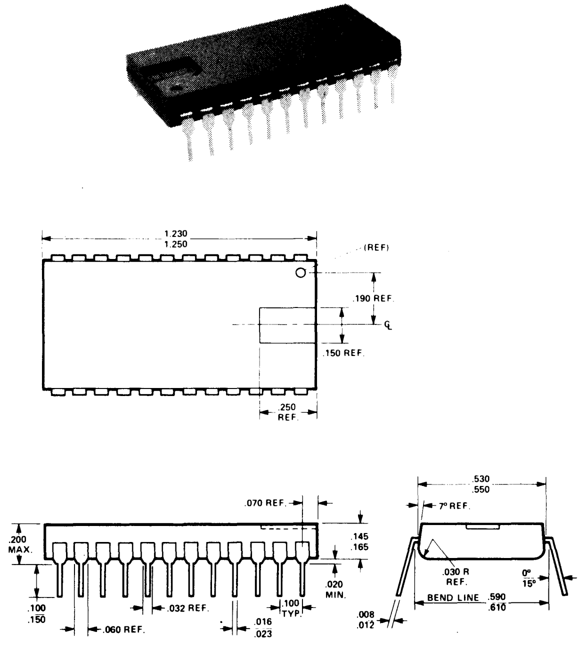
NOTE 1. 0.125 MAX. FOR 1602A
0.150 MAX. FOR 1702A

NOTE 2. 0.050 MAX. FOR 1602A
0.075 MAX. FOR 1702A

NOTE 3. THE 1702A HAS AN ADHESIVE LABEL FOR IDENTIFYING THE CUSTOMER ROM PATTERN NUMBER. THIS ALLOWS THE ROM PATTERN NUMBER TO BE EASILY CHANGED AFTER ERASING.

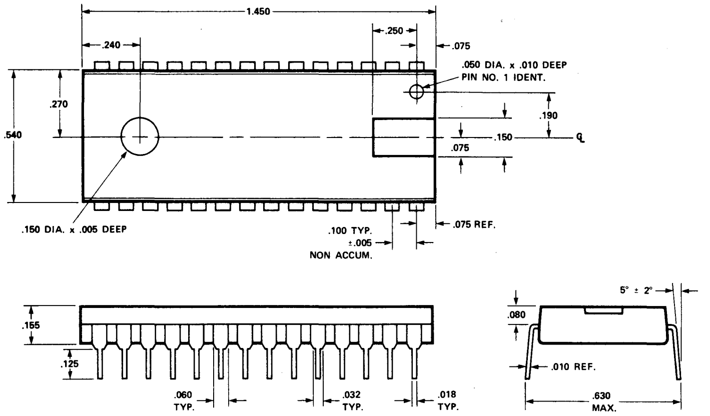


24-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE

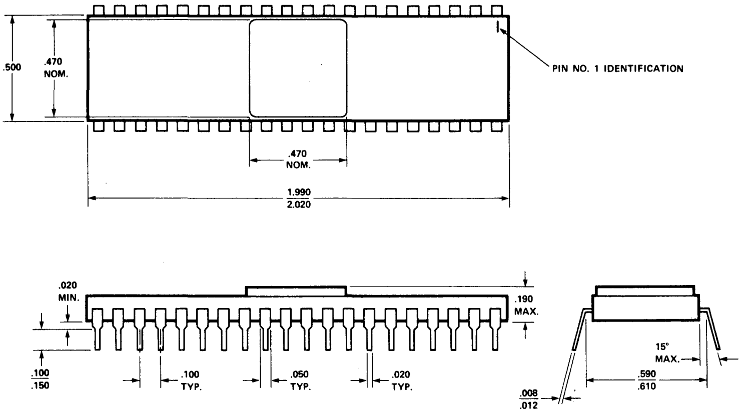


PACKAGING INFORMATION (Con't.)

28-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE



40-LEAD PLASTIC DUAL IN-LINE PACKAGE OUTLINE



4308 CUSTOM ROM ORDER FORM

4308 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 the 4308 Data Catalog for complete pattern specifications. Additional forms are available from Intel.

CUSTOMER ____________________

ADDRESS ____________________

____

P.O. NUMBER ____________________

DATE ____________________

Intel use

S# ____________ PPPP ____________

STD ____________ ZZ ____________

APP ____________ DD ____________

DATE ____________ I/O ____________

INTEL STANDARD MARKING

The marking as shown at right must contain the Intel logo, the product type (P4308), 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).____________

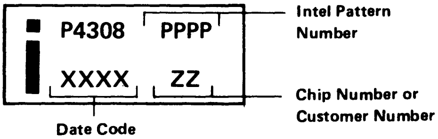
Optional Customer Number (Maximum 6 characters or spaces)

ROM DESCRIPTION

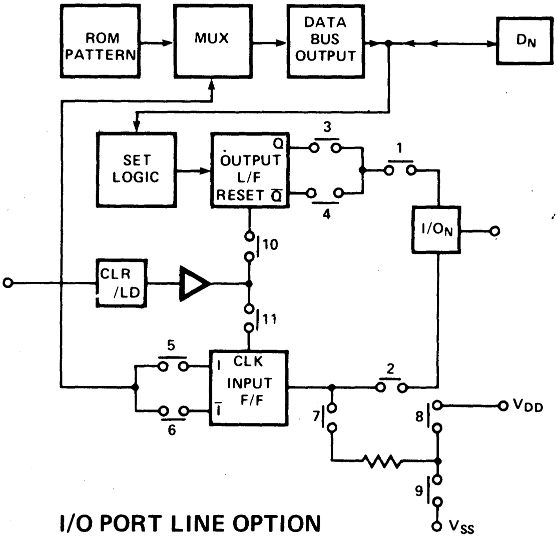
Chip Select Value (0–3) ____________ (Must be specified.)

In the table below, select the connections which should be made for each of the 16 I/O port input lines. Avoid the use of illegal options—refer to the 4308 Data Catalog.

Mark the appropriate box for an option connection. Leave blank for a no connection.



PIN		OPTION										
I/O 0_0	27	1	2	3	4	5	6	7	8	9	10	11
I/O 0_1	26	1	2	3	4	5	6	7	8	9	10	11
I/O 0_2	25	1	2	3	4	5	6	7	8	9	10	11
I/O 0_3	24	1	2	3	4	5	6	7	8	9	10	11
I/O 1_0	5	1	2	3	4	5	6	7	8	9	10	11
I/O 1_1	4	1	2	3	4	5	6	7	8	9	10	11
I/O 1_2	3	1	2	3	4	5	6	7	8	9	10	11
I/O 1_3	2	1	2	3	4	5	6	7	8	9	10	11
I/O 2_0	17	1	2	3	4	5	6	7	8	9	10	11
I/O 2_1	16	1	2	3	4	5	6	7	8	9	10	11
I/O 2_2	15	1	2	3	4	5	6	7	8	9	10	11
I/O 2_3	14	1	2	3	4	5	6	7	8	9	10	11
I/O 3_0	21	1	2	3	4	5	6	7	8	9	10	11
I/O 3_1	20	1	2	3	4	5	6	7	8	9	10	11
I/O 3_2	19	1	2	3	4	5	6	7	8	9	10	11
I/O 3_3	18	1	2	3	4	5	6	7	8	9	10	11



I/O PORT LINE OPTION

intel®

MCS-4 CUSTOM ROM ORDER FORM

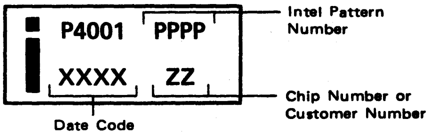
4001 Metal Masked ROM

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:

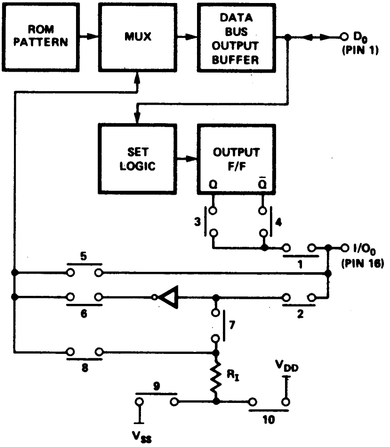
   Inputs – 2 and 6 are connected
   Outputs – 1, 3, 8 and 9 are connected or
   1, 3, 8 and 10 are connected

   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

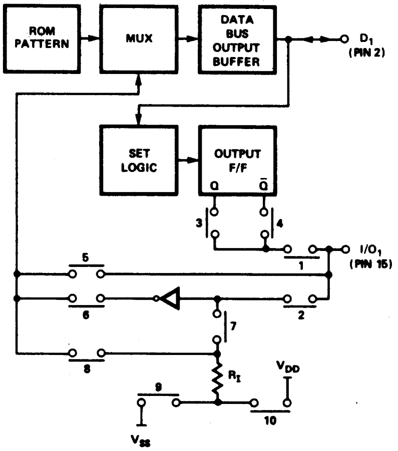


I/O$_0$ (PIN 16)

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

a. For T^{2}L 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).

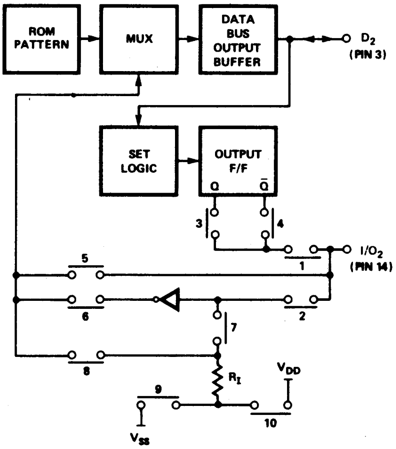


I/O$_1$ (PIN 15)

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

a. For T^{2}L 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).

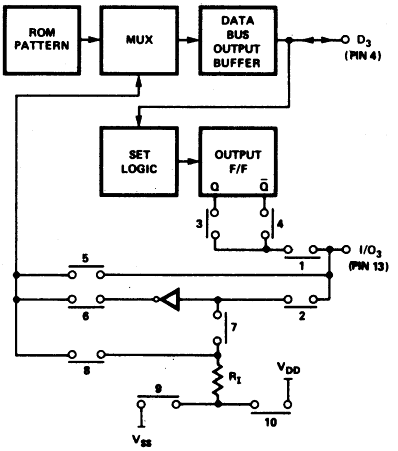


I/O$_2$ (PIN 14)

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

a. For T^{2}L 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^{2}L 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).

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

intel®

□ 4004 □ 8008 □ 8080 □ 4040

(use additional sheets if necess

Program Title

Function

Required Hardware

Required Software

Input Parameters

Output Results

Registers Modified:	Maximum Subroutine Nesting Level:
RAM Required:	Assembler/Compiler Used:
ROM Required:	Programmer:
	Company:

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
   a. Processor (check appropriate box)
   b. Program title: Name or brief description of program function
   c. Function: Detailed description of operations performed by the program
   d. Required hardware:
      For example: TTY on port 0 and 1
      Interrupt circuitry
      I/O Interface
      Machine line and configuration for cross products
   e. Required software:
      For example: TTY routine
      Floating point package
      Support software required for cross products
   f. Input parameters: Description of register values, memory areas or values accepted from input ports
   g. Output results: Values to be expected in registers, memory areas or on output ports
   h. Program details (for resident products only)
      1. Registers modified
      2. RAM required (bytes)
      3. ROM required (bytes)
      4. Maximum subroutine nexting level
   i. Assembler/Compiler Used:
      For example: PL/M
      Intellec 8 Macro Assembler
      IBM 370 Fortran IV
   j. Programmer and company

2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should **not** be included.

3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

U.S. SALES AND MARKETING OFFICES

U.S. MARKETING HEADQUARTERS
3065 Bowers Avenue
Santa Clara, California 95051*
Tel: (408) 246-7501
TWX: 910-338-0026
TELEX: 34-6372

NATIONAL SALES MANAGER
Hank O'Hara
3065 Bowers Avenue
Santa Clara, California 95051*
Tel: (408) 246-7501
TWX: 910-338-0026
TELEX: 34-6372

U.S. REGIONAL SALES MANAGERS' OFFICES

WESTERN
William T. O'Brien
1651 E. 4th St.
Suite 228
Santa Ana, California 92701*
Tel: (714) 835-9642
TWX: 910-595-1114

MID-AMERICA
Mick Carrier
6350 L.B.J. Freeway
Suite 178
Dallas, Texas 75240*
Tel: (214) 661-8829

GREAT LAKES REGION
Hank Josefczyk
856 Union Rd.
Englewood, Ohio 45322*
Tel: (513) 836-2808

EASTERN
Jim Saxton
2 Militia Drive
Suite 4
Lexington, Massachusetts 02173*
Tel: (617) 861-1136
TELEX: 92-3493

MID-ATLANTIC
John Kitzrow
520 Pennsylvania Ave.
Fort Washington, Pennsylvania 19034*
Tel: (215) 542-9444

U.S. SALES OFFICES

ARIZONA
Sales Engineering, Inc.
7155 E. Thomas Road, No. 6
Scottsdale 85252
Tel: (602) 994-3230
TWX: 910-950-1288

CALIFORNIA
3065 Bowers Avenue
Santa Clara 95051*
Tel: (408) 246-7501
TWX: 910-338-0026

1651 E. 4th Street
Suite 228
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Earle Associates, Inc.
4433 Convoy Street
Suite A
San Diego 92111
Tel: (714) 278-5441
TWX: 910-335-1585

COLORADO
1341 South Lima Street
Aurora 80010*
Tel: (303) 758-2505

FLORIDA
1090 N.E. 27th Terrace
Pompano Beach 33062
Tel: (305) 781-7450

ILLINOIS
Mar-Con Associates, Inc.
4836 Main Street
Skokie 60076
Tel: (312) 675-6450

1701 Carmen Dr.
Elk Grove Village 60007*
Tel: (312) 439-3764
TWX: 910-222-2710

MARYLAND
Barnhill and Associates
1931 Greenspring Drive
Timonium 21093
Tel: (301) 252-5610

Barnhill and Associates
P.O. Box 251
Glen Arm 21057
Tel: (301) 252-5610

MASSACHUSETTS
2 Militia Drive
Suite 4
Lexington 02173*
Tel: (617) 861-1136
TELEX: 92-3493

Datcom
55 Moody Street
Waltham 02154
Tel: (617) 891-4600
TELEX: 92-3462

MICHIGAN
Sheridan Associates, Inc.
33708 Grand River Road
Farmington 48024
Tel: (313) 477-3800

MINNESOTA
800 Southgate Office Plaza
5001 West 78th Street
Bloomington 55437*
Tel: (612) 835-6722

MISSOURI
Sheridan Associates, Inc.
110 S. Highway 140
Suite 10
Florissant 63033
Tel: (314) 837-5200

NEW JERSEY
Addem
Post Office Box 231
Keasbey 08832
Tel: (516) 567-5900

NEW YORK
Ossmann Components Sales Corp.
395 Cleveland Drive
Buffalo 14215
Tel: (716) 832-4271

Addem
45 Cambridge Street
Deer Park 11729
Tel: (516) 567-5900

Ossmann Components Sales Corp.
280 Metro Park
Rochester 14623
Tel: (716) 442-3290

Ossmann Components Sales Corp.
1911 Vestal Parkway E.
Vestal 13850
Tel: (607) 785-9949

Ossmann Components Sales Corp.
132 Pickard Building
Syracuse 13211
Tel: (315) 454-4477

Ossmann Components Sales Corp.
140 Pine Street
Kingston 12401
Tel: (914) 338-5505

NORTH CAROLINA
Barnhill and Associates
6030 Bellow Street
Raleigh 27602
Tel: (919) 787-5774

OHIO
Sheridan Associates, Inc.
23224 Commerce Park Rd.
Beachwood 44122
(216) 831-0130

Sheridan Associates, Inc.
Shiloh Bldg.
Suite 250
5045 North Main Street
Dayton 45405
Tel: (513) 277-8911

Sheridan Associates, Inc.
10 Knollcrest Drive
Cincinnati 15237
Tel: (513) 761-5432
TWX: 810-461-2670

CANADA
Multilek, Inc.
4 Barran Street
Ottawa, Ontario K2J 1G2
Tel: (613) 825-4695
TELEX: 053-4585

PENNSYLVANIA
Vantage Sales Company
21 Bala Avenue
Bala Cynwyd 19004
Tel: (215) 667-0990

Sheridan Associates, Inc.
1717 Penn Avenue
Suite 5009
Pittsburgh 15221
Tel: (412) 244-1640

TENNESSEE
Barnhill and Associates
206 Chicasaw Drive
Johnson City 37601
Tel: (615) 928-0184

TEXAS
Evans & McDowall Associates
13777 N. Central Expressway
Suite 405
Dallas 75231
Tel: (214) 238-7157
TWX: 910-867-4763

VIRGINIA
Barnhill and Associates
P.O. Box 1104
Lynchburg 24505
Tel: (703) 846-4624

WASHINGTON
SDR² Products and Sales
14040 N.E. 8th Street
Bellevue 98007
Tel: (206) 747-9424
TWX: 910-443-2483

*Direct Intel Offices

EUROPEAN MARKETING OFFICES

EUROPEAN MARKETING HEADQUARTERS

BELGIUM
Tom Lawrence
Intel Office
216 Avenue Louise
Brussels B1050
Tel: (02) 649-20-03
TELEX: 24814

FRANCE
Bernard Giroud
Intel Office
Cidex R-141
94534 Rungis
Tel: (1) 677-60-75
TELEX: 27475

DENMARK
John Johansen
Intel Office
Lyngbyvej 32 2nd Fl.
2100 Copenhagen East
Tel: (01) 18 20 00
TELEX: 19567

EUROPEAN MARKETING OFFICES

ENGLAND
Keith Chapple
Intel Office
Broadfield House
4 Between Towns Road
Cowley, Oxford
Tel: (0865) 771431
TELEX: 837203

GERMANY
Erling Holst
Intel Office
Wolfratshauserstrasse 169
D8 Munchen 71
Tel: (089) 798923
TELEX: 5-212870

INTERNATIONAL DISTRIBUTORS

AUSTRALIA
A. J. Ferguson (Adelaide) PTY, Ltd.
125 Wright Street
Adelaide 5000
Tel: (51) 6895

AUSTRIA
Bacher Elektronische Gerate GmbH
Meidlinger Hauptstrasse 78
A 1120 Vienna
Tel: (0222) 83-63-96
TELEX: (01) 1532

BELGIUM
Inelco Belgium S.A.
Avenue Val Duchesse, 3
B-1160 Bruxelles
Tel: (02) 660 00 12
TELEX: 25441

DENMARK
Scandinavian Semiconductor
Supply A/S
20, Nannasgade
DK-2200 Copenhagen N
Tel: (1) 93-50-60
TELEX: 19037

FINLAND
Havulinna Oy
P.O. Box 468
SF 00100 Helsinki 10
Tel: (90) 661451
TELEX: 12426

FRANCE
Tekelec Airtronic
Cite des Bruyeres
Rue Carle Vernet
92 Sevres
Tel: (1) 626-02-35
TELEX: 25997

GERMANY
Alfred Neye Enatachnik GmbH
Schillerstrasse 14
2085 Quickborn-Hamburg
Tel: (041) 06/612-1
TELEX: 02-13590

ISRAEL
Telsys Ltd.
54, Jabotinsky Road
Ramat - Gan 52 464
Tel: (3) 739865
TELEX: TSEE-IL 32392

ITALY
Eledra 3S
Via Ludovico da Viadana 9
20122 Milano
Tel: (02) 86-03-07
TELEX: 32027

SPAIN
Interface
Ronda General Mitre #7
Barcelona 17,
Spain
Tel: (093) 203-53-30
TELEX: 54713

NETHERLANDS
Joan Muyskenweg 22
NL1006
Amsterdam
Tel: (020) 93-48-24
TELEX: 14622

NORWAY
Nordisk Elektronik (Norge) A/S
Mustads Vei 1
Oslo 2
Tel: (02) 553893
TELEX: 16963

SOUTH AFRICA
Electronic Building Elements
P.O. Box 4609
Pretoria
Tel: (78) 9221
TELEX: 30181 SA

SWEDEN
Nordisk Electronik AB
Fack
S-103 Stockholm 7
Tel: 08-24-83-40
TELEX: 10547

SWITZERLAND
Industrade AG
Gemsenstrasse 2
Postcheck 80 - 21190
8021 Zurich
Tel: 01-60-22-30
TELEX: 56788

UNITED KINGDOM
Walmore Electronics Ltd.
11-15 Betterton Street
Drury Lane
London WC2H 9BS
Tel: (01) 836-1228
TELEX: 28752

ORIENT MARKETING OFFICES

ORIENT MARKETING HEADQUARTERS

JAPAN
Y. Magami
Intel Japan Corporation
Kasahara Bldg.
1-6-10, Uchikanda
Chiyoda-ku
Tokyo 101
Tel: (03) 295-5441
TELEX: 781-28426

ORIENT DISTRIBUTORS

JAPAN
Pan Elektron Inc.
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8321
TELEX: 781-4773

4040 AND 4004 BASIC INSTRUCTION SET

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

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

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

NEW 4040 INSTRUCTIONS

MNEMONIC	OPR $D_3 D_2 D_1 D_0$	OPA $D_3 D_2 D_1 D_0$	DESCRIPTION OF OPERATION
HLT	0 0 0 0	0 0 0 1	Halt – inhibit program counter and data buffers.
BBS	0 0 0 0	0 0 1 0	Branch Back from Interrupt and restore the previous SRC. The Program Counter and send register control are restored to their pre-interrupt value.
LCR	0 0 0 0	0 0 1 1	The contents of the COMMAND REGISTER are transferred to the ACCUMULATOR.
OR4	0 0 0 0	0 1 0 0	The 4 bit contents of register #4 are logically "OR-ed" with the ACCUM.
OR5	0 0 0 0	0 1 0 1	The 4 bit contents of index register #5 are logically "OR-ed" with the ACCUMULATOR.
AN6	0 0 0 0	0 1 1 0	The 4 bit contents of index register #6 are logically "AND-ed" with the ACCUMULATOR
AN7	0 0 0 0	0 1 1 1	The 4 bit contents of index register #7 are logically "AND-ed" with the ACCUMULATOR.
DB0	0 0 0 0	1 0 0 0	DESIGNATE ROM BANK 0. CM-ROM_0 becomes enabled.
DB1	0 0 0 0	1 0 0 1	DESIGNATE ROM BANK 1. CM-ROM_1 becomes enabled.
SB0	0 0 0 0	1 0 1 0	SELECT INDEX REGISTER BANK 0. The index registers 0 - 7.
SB1	0 0 0 0	1 0 1 1	SELECT INDEX REGISTER BANK 1. The index registers 0* - 7*.
EIN	0 0 0 0	1 1 0 0	ENABLE INTERRUPT.
DIN	0 0 0 0	1 1 0 1	DISABLE INTERRUPT.
RPM	0 0 0 0	1 1 1 0	READ PROGRAM MEMORY.

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.
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_1 = 0	Not invert jump condition
C_2 = 1	Jump if accumulator is zero
C_3 = 1	Jump if carry/link is a 1
C_4 = 1	Jump if test signal is a 0

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

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

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

intel®

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

Printed in U.S.A./MCS-335-1174/27.5K