intel®

PRELIMINARY

MCS-86™ USER'S MANUAL

July 1978

9800722A

MCS-86™ User's Manual
Table of Contents

CHAPTER 5 (Continued)

Peripherals** (Continued)

Static RAMs***

ROMs/EPROMs***

CHAPTER 6

DEVELOPMENT AIDS

APPENDIX

*This section contains partial data sheets. For complete specifications refer to the Intel MCS-85 User's Manual.

**This section contains partial data sheets. For complete specifications refer to the Intel Peripheral Design Handbook.

***This section contains partial data sheets. For complete specifications refer to the 1978 Intel Data Catalog.

CHAPTER 1

Introduction

CHAPTER 1
INTRODUCTION

The Intel® 8086, a new microcomputer, extends the mid-range 8080 family into the 16-bit arena. The chip has attributes of both 8- and 16-bit processors. By executing the full set of 8080A/8085 8-bit instructions plus a powerful new set of 16-bit instructions, it enables a system designer familiar with existing 8080 devices to boost performance by a factor of as much as 10 while using essentially the same 8080 software package and development tools.

The goals of the 8086 architectural design were to extend existing 8080 features symmetrically, across the board, and to add processing capabilities not to be found in the 8080. The added features include 16-bit arithmetic, signed 8- and 16-bit arithmetic (including multiply and divide), efficient interruptible byte-string operations, and improved bit manipulation. Significantly, they also include mechanisms for such minicomputer-type operations as reentrant code, position-independent code, and dynamically relocatable programs. In addition, the processor may directly address up to 1 megabyte of memory and has been designed to support multiple-processor configurations.

How It Is Done

The 8086's improved performance stems from a combination of process and architectural enhancements. It is the first microcomputer to be fabricated with the newly developed silicon-gate H-MOS process, which gives the device 4-micrometer scaled-down metal-oxide-semiconductor transistors and on-chip biasing that make it operate faster and more reliably.

With this high-performance MOS technique, typical on-chip gate propagation delays of 2 nanoseconds are as short as those obtained from costly Schottky transistor-transistor logic. This results in extremely fast internal clocking rates: 5 megahertz (200ns). That is faster than any one-chip central processing unit now available. Since four CPU clock cycles correspond to approximately one memory cycle, the 8086 is much more efficient in accessing memory. Indeed, memory chip selection for the 8086 requires devices that cycle in 500ns to 800ns and access data (address to data-in valid) in a matter of 295 to 460ns.

The H-MOS process also produces denser circuitry. The entire 16-bit data and microprogrammed control structures use 29,000 transistors integrated onto a die about 225 mils square. Many less complex peripheral chips that use large-scale integration are larger. The smallness of its die means that the high-performance 8086 will decline in cost as production experience with it grows, just as happened with the 8080, 8080A, and 8085.

An Enhanced Architecture

The architectural enhancements of the 8086 stem from a powerful register structure, increased memory address capability, almost unlimited levels of interrupts, and powerful input and output interface circuitry.

Unlike the 8080/8085 CPUs, the 8086's registers can process 16-bit as well as 8-bit data. A general register file provides operands for the 16-bit arithmetic/logic instructions. It contains four 16-bit general data registers that are also addressable as eight 8-bit registers, two 16-bit memory base pointer registers, and two 16-bit index registers. All data manipulation instructions apply to all registers; certain addressing modes imply specific registers. All told, twice as many general-purpose data registers are provided as on 8-bit CPU chips. Complex arithmetical capability, very flexible memory addressing, and high computational throughput are the result.

A second bank of registers is designated the segment register file and extends the chip's addressing capabilities. It can address 1 megabyte of memory, in contrast to the 65,536-byte capacity of the8080/8085. In this file, address relocation values for up to four 64-kilobyte program or data segments are stored in four 16-bit segment registers. The chip can control a full 64 kilobytes of address space for input/output ports.

The Instruction Set

The 8086 instruction set can address operands in several different ways. In general, operands in memory may be addressed either directly, with the 16-bit offset address, or indirectly, with base and/or index registers added to an optional 8- or 16-bit displacement constant.

A two-operand instruction format technique allows memory or any register to serve as one operand and either a register or a constant within an instruction to serve as the other operand. In these cases, the results of the two-operand operation may be directed to either of the source operands, unless one is an in-line (immediate) constant. On the other hand, single-operand operations are applicable uniformly to any operand in register or memory, with the same exception for immediate constants.

Within this instruction format, the 8086 supplies several variations of the four basic arithmetical operations (add, subtract, multiply, and divide). Both 8- and 16-bit arithmetical operations and both signed and unsigned arithmetic are provided, standard 2's complement representation being used to represent signed values. In this context, addition and subtraction can be both signed and unsigned, with flag settings to distinguish between the signed and unsigned operations.

With the aid of its correction operations, the 8086 can do this arithmetic directly on unpacked ASCII coded representations of decimal digits or on packed decimal representations. Standard logical operations, shift, move data, etc., are available to both 8- and 16-bit operands. Other instructions support the movement of 32-bit address objects called pointers, which consist of a 16-bit offset address and a 16-bit segment base address.

Also provided is a group of one-byte instructions that, besides performing various primitive operations to manipulate byte and word strings, can each be performed repeatedly when provided with a special prefix. The single-operation forms can also be combined to form complex strings of operations, with their repetition controlled by special iterative operations. The effect is to create tight, efficient loops for performing complex string functions.

For handling program flow, two basic varieties of calls, jumps, and returns are provided — one that transfers control within the current code segment, and one that transfers control to an arbitrary code segment, which then becomes the current code segment. The 8086 supports direct and indirect transfers, both of which make use of the standard addressing modes. Intrasegment calls and jumps specify a self-relative displacement, thus allowing position-independent code.

In all, 16 conditional jumps are provided. Both signed and unsigned relationships can be tested, as well as parity, overflow, zero, and sign conditions.

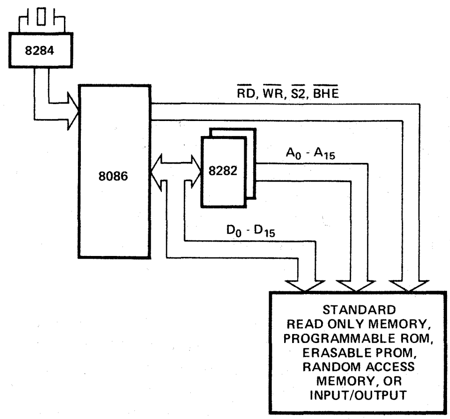
Some Sample Systems

In a minimum mode system of the kind shown in Figure 1a, the 8086 generates the control signals used by the memory and I/O devices for interacting with the address and data buses, as well as a timing signal for latching the addresses. In this configuration, as noted earlier, the access times required of memory and I/O devices for the 5MHz 8086 are roughly 430ns from receipt of address and 205ns from receipt of read or write enable.

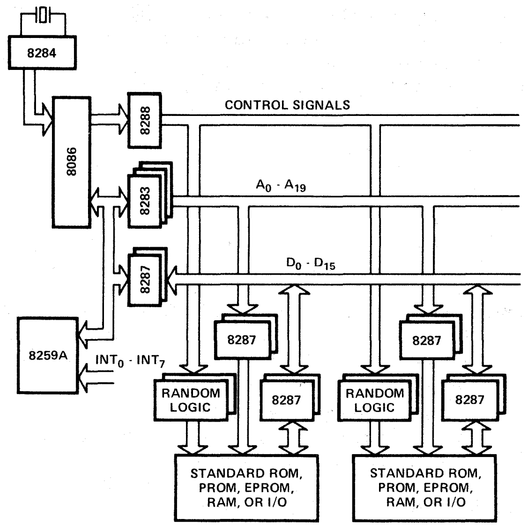
In the larger buffered configuration (Figure 1b), the 8086, in its maximum mode, generates coded status information on only some of the pins needed for producing minimum-mode control signals. Here, all that is needed is enough coded information to push an 8288 bus controller into generating Multibus-compatible control signals and timing signals for addressing latches and data transceivers. The minimum-mode control-signal pins can now take on additional functions, such as extra direct-memory-access control and bus-locking capabilities for use in multiprocessor operations.

In buffered systems the required memory and I/O access times for 5MHz CPU operation are 395ns from receipt of address and 290ns from receipt of read command.

The bottom line of all system design is performance. Initial studies have indicated that on average an 8086-based system will perform about an order of magnitude better than an 8080A-based one. Depending on program type, execution speeds 7 to 12 times faster than 8080A speeds can be expected. At the same time, the program is typically 10% to 25% shorter. However, while there exists an 8086 instruction mapping for every 8080 instruction, and while 8080 programs may be easily transferred to the 8086, maximum 8086 efficiency does require the rewriting of certain routines.



(a)



(b)

Figure 1. Systems. When operating in the minimum mode, the 8086 needs only 11 components to form a complete system, including clock, 2 kilobytes of RAM, and 4 kilobytes of ROM (a). By adding support components, very large systems (b) can be configured.

CHAPTER 2

Functional Description

CHAPTER 2
FUNCTIONAL DESCRIPTION

2.1 What The 8086 Is

The 8086 is a complete microprocessor for use in general-purpose computer systems of widely varying levels of complexity. At its simplest, an 8086-based system might be comparable in complexity to an 8080-based system, but is several times faster. At its upper limit, it can be a multiple-processor system, each of whose processors is capable of accessing up to 1 megabyte of memory.

Its memory is organized in 8-bit bytes, but memory transfers are handled in 16-bit words equally as easily as in bytes. Bit, byte, word, and block (string) operations are accommodated in the instruction set. The 8086 performs signed arithmetic and interruptible string operations, and can make use of relocatable subroutines, reentrant programming and multiprocessing. The device requires a single, 5-volt power source. It is contained in a standard, 40-pin dual in-line package (DIP).

Accomplishing the range of functions the 8086 is designed to perform within the standard package size is done in two ways. First, external communication to memory and peripherals is accomplished through a 20-bit time-multiplexed address and data bus (described in detail in Section 2.3.2.). Second, internal configuration switching is used to adapt the processor to the level of system complexity you desire. In simple systems, the 8086 generates its own bus control; in more complex systems, bus control is assumed by an 8288 bus controller, and eight of the 8086's physical leads are switched to perform the needed coordinating functions. Pin 33 of the 8086 (MN/MX) performs the switching: you connect it permanently to ground or to +5 V via a trace on your circuit board, depending on whether or not you are using the 8288.

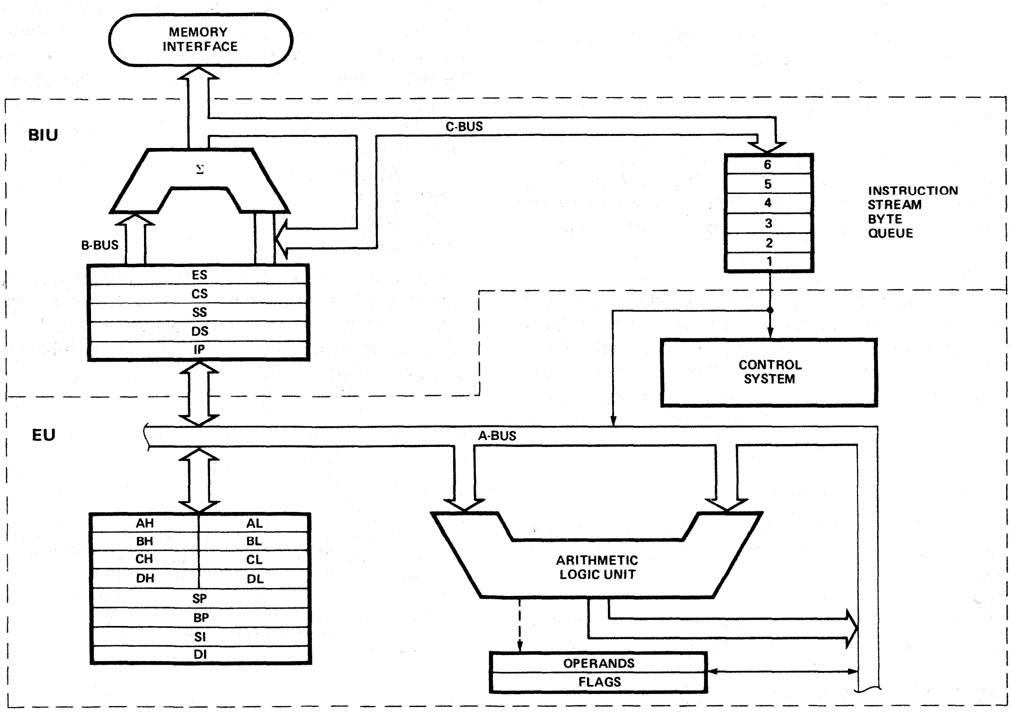


Figure 2-1. 8086 Functional Block Diagram.

The 8086 family of devices is summarized in the table below:

8086 cpu
8284 clock driver
8282 eight-line latch
8283 inverting eight-line latch
8286 eight-line three-state transceiver
8287 inverting eight-line three-state transceiver
8288 bus controller
8259A interrupt controller

The bus structure of the MCS-86 systems is compatible with MCS-80 and MCS-85 peripherals. This allows you to utilize pre-existing devices and hardware designs. Existing 8080 system software also is adaptable for use in MCS-86 systems. As much of your 8080 program as is independent of instruction execution time, size, memory location, or specific code (e.g., code that functions as a mask) can be machine-translated at the assembly-language level, and assembled into 8086 code without modification, since the 8080 registers represent a subset of the 8086 register set. (See Figure 2-5.) Time- and space-dependent subroutines, written for the MCS-80, would have to be redesigned to use a different scheme. In some cases, internal bus contention may cause the execution time of an instruction in the 8086 to vary, depending upon its position, but any given sequence is repeatable. In addition, vectored interrupts are handled in a different way in the 8086, which requires adaptation of 8080 interrupt software.

2.2 What's In The 8086

The internal functions of the 8086 microprocessor are divided into two major functional areas, as follows:

- Execution/Control Unit (EU)
- Bus Interface Unit (BIU)

Figure 2-1 shows the interrelations between these two units and the further breakdown of processing functions within each. They interact directly, but for the most part perform their separate functions independent of each other.

2.2.1 Execution Unit (EU)

The EU performs the basic processing functions, as it contains the data registers and the arithmetic-logic unit (ALU). It accepts prefetched instructions from the BIU and returns unrelocated operand addresses to it. (See paragraph 2.3.1 for discussion of memory addressing.) It then receives memory operands via the BIU, processes them, and passes the results to the BIU for storage.

2.2.2 Bus Interface Unit (BIU)

The purpose of the BIU is to maximize bus bandwidth utilization, as this is a major factor limiting processor speed. It does this in two ways. First, it prefetches instructions before they are required by the EU. It buffers them in a queue that can contain up to six bytes of instruction stream, awaiting their decoding and execution. The EU therefore need not wait for completion of a bus cycle before taking in a new instruction. Second, the BIU provides the functions related to operand fetch and store, address relocation, and bus control, all in parallel with EU processing.

2.2.3 Registers

The 8086 processor contains three sets of four 16-bit registers and a set of nine one-bit flags. The three sets of registers are the general registers, the pointer and index register set, and the segment registers. There is a 16-bit instruction pointer which is not directly accessible; rather it is manipulated with control transfer instructions. (See Figure 2-2.)

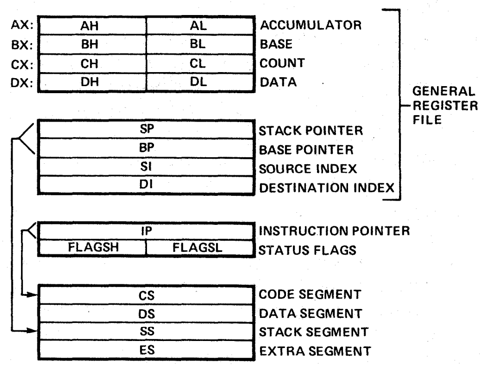


Figure 2-2. 8086 Register Format.

The (AX, BX, CX, DX) register set is called the General Register, or HL group. (See Figure 2-3.) The general registers can participate in the arithmetic and logic operations of the 8086 without constraint. Some of the other 8086 operations (such as the string operations) dedicate certain of the general registers to specific uses. These uses are indicated by the following mnemonic phrases:

AX: Accumulator
BX: Base
CX: Count
DX: Data

The general registers have a property that distinguishes them from the other registers, namely that their upper and lower halves are separately addressable. Thus the general registers can be thought of as two sets of four 8-bit registers. These are called H and L.

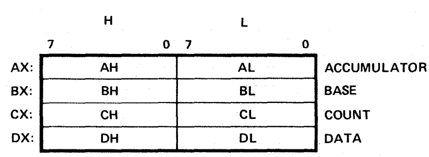


Figure 2-3. General Registers.

The accumulator is distinguished in another way: you get more compact programs by using it as the target of your data transfer, arithmetic, and logic instructions than when you use the general-registers. (See Section 2.2.4 for details.) The remainder of the registers in the 8086 processor are indivisible, and must be accessed as if containing 16-bit words, whether or not both their high-order bytes and their low-order bytes are utilized.

The (SP, BP, SI, DI) register set is called the Pointer and Index Register (P and I group; see Figure 2-4.) The registers in this group are similar in that they generally contain offset addresses used for addressing within a segment. Like the general registers, the pointer and index registers can participate in the 16-bit arithmetic and logical operations of the 8086.

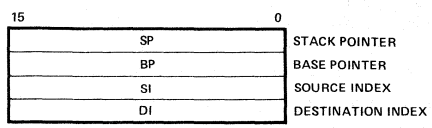


Figure 2-4. Pointer and Index Registers.

They are also similar in that they can enter into address computations. There are some differences, however, which result in dividing this set into two groups, the P, or Pointer group (SP, BP) and the I, or Index group (SI, DI). The difference is that the Pointers are by default assumed to contain offset addresses within the current stack segment, and the Indexes are by default assumed to contain offset addresses within the current data segment (except for string operations). The mnemonics associated with these registers are:

SP: Stack Pointer
BP: Base Pointer
SI: Source Index
DI: Destination Index

Certain of the 8086 registers are similar in function to the registers found in the 8080 series processors. (See Figure 2-5.) The BH and BL registers in the 8086 correspond to the H and L registers in the 8080. The CH and CL registers correspond to the B and C, and the DH and DL to the D and E. The SP and PC translate directly from the 8080 to SP and IP in the 8086, but their flag registers are a little different.

The (AF, CF, DF, IF, OF, PF, SF, TF, ZF) register set is called the Flag Register or F group. The flags in this group are all one bit in size, and are used to record processor status information and to control processor operation. The flag register mnemonics are:

AF:	Auxiliary-carry	OF:	Overflow	ZF:	Zero
CF:	Carry	PF:	Parity		
DF:	Direction	SF:	Sign		
IF:	Interrupt-enable	TF:	Trap		

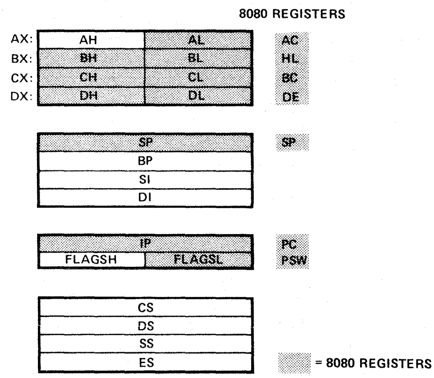


Figure 2-5. 8080 Registers as a Subset of 8086 Registers.

The AF, CF, PF, SF, and ZF flags are equivalent to 8080 flags, generally reflecting the status of the latest arithmetic or logical operation. The ZF flag is set if the result of an instruction is zero. The SF flag is set if a 1 appears as the most significant bit in a result. A PF flag set to 1 indicates even parity. A carry or borrow out of the high-order bit sets the CF flag, and a carry out of bit 3 into bit 4 or borrow from bit 4 into bit 3 sets the AF flag. The OF flag joins this group, reflecting the signed arithmetic overflow condition. The DF, IF, and TF flags are used to control certain aspects of the processor. The DF flag controls the direction of the string manipulation instructions (auto-incrementing or auto-decrementing). The IF flag enables or disables external interrupts. The TF flag puts the processor into a single-step mode for program debugging.

The flag registers are illustrated in Figure 2-6 in the format in which they are stored by push-flags operations. Figure 2-7 shows the equivalence of 8080 flags to those of 8086. Note that in the MCS-80 family, the PSW transferred to and from stack includes not only flags but also the content of the accumulator. A further discussion of the 8086 flags and their functions is contained in Chapter 4.

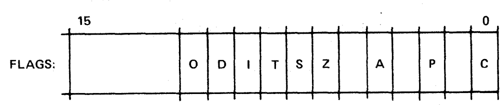


Figure 2-6. Flag Registers.

The (CS,DS, SS, ES) register set is called the Segment Register File, or S group. The segment registers play an important role in that they are used in all memory address computations. (See Paragraph 2.3.1.) The segment register mnemonics are:

CS: Code
DS: Data
SS: Stack
ES: Extra

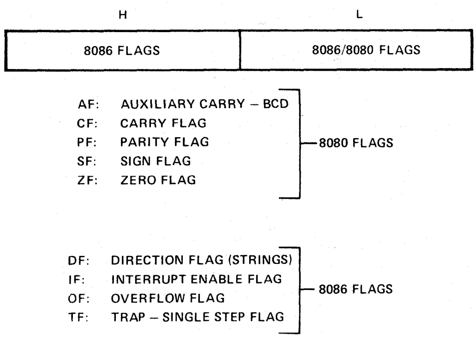


Figure 2-7. 8080/8086 Flag Register Equivalence.

The contents of the CS register define the current code segment. All instruction fetches are taken to be relative to CS, using the instruction pointer (IP) as an offset.

The contents of the DS register define the current data segment. All data references except those involving BP or SP, or DI in a string instruction, are taken by default to be relative to DS. Data references can be forced to be relative to one of the other three segment registers by preceding the instruction with a one-byte segment override prefix.

The contents of the SS register define the current stack segment. All data references which explicitly or implicitly involve SP or BP are taken by default to be relative to SS. This includes all push and pop operations, including those caused by call operations, interrupts, and return operations. Data references involving BP (but not SP) can be forced to be relative to one of the other three segment registers by using the special one-byte base prefix. (See Chapter 4.)

The contents of the ES register define the current extra segment. The extra segment is usually treated as an additional data segment. Data references in string instructions which use DI are taken to be relative to ES.

Programs which do not load or manipulate the segment registers are said to be dynamically relocatable. Such a program may be interrupted, moved in memory to a new location, and restarted with new segment register values.

The segment registers are illustrated in Figure 2-8.

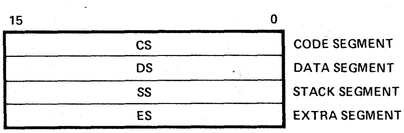


Figure 2-8. Segment Register File.

2.2.4 Organization of the Instruction Set

Instructions are described here in six functional groups:

Data transfer
Arithmetic
Logic
String manipulation
Control transfer
Processor control

Each of the first three groups mentioned in the preceding list is further subdivided into an array of codes that specify whether the instruction is to act upon immediate data, register or memory locations, whether 16-bit words or 8-bit bytes are to be processed, and what addressing mode is to be employed. All of these codes are listed and explained in detail in this book, but when you are writing assembly-language programs you do not have to code each one individually. The context of your program automatically causes the assembler to generate the correct code. There are three general categories of instructions within each of the three functional groups mentioned:

Register or memory space to or from register
Immediate data to register or memory
Accumulator to or from registers, memory, or ports

The assembly-language programming manual describes the syntax of the 8086 instruction set.

DATA TRANSFER

Data transfer operations are divided into four classes:

- general purpose
- accumulator-specific
- address-object
- flag

None affect flag settings except SAHF and POPF.

General Purpose Transfers. Four general purpose data transfer operations are provided. These may be applied to most operands, though there are specific exceptions. The general purpose transfers (except XCHG) are the only operations which allow a segment register as an operand.

—MOV performs a byte or word transfer from the source operand to the destination operand.
—PUSH decrements the SP register by two and then transfers a word from the source operand to the stack element currently addressed by SP.
—POP transfers a word operand from the stack element addressed by the SP register to the destination operand and then increments SP by 2.
—XCHG exchanges the byte or word source operand with the destination operand. The segment registers may not be operands of XCHG.

Accumulator-Specific Transfers. Three accumulator-specific transfer operations are provided:

—IN (or INW) transfers a byte (or word) from an input port to the AL register (or AX register for INW). The port is specified either with an inline data byte, allowing fixed access to ports 0 through 255, or with a port number in the DX register, allowing variable access to 64K input ports.
—OUT (or OUTW) is similar to IN (or INW) except that the transfer is from the accumulator to the output port.

—XLAT performs a table lookup byte translation. The AL register is used as an index into a 256-byte table addressed by the BX register. The byte operand so selected is transferred to AL.

Address-Object Transfers. Three address-object transfer operations are provided:

—LEA (load effective address) transfers the offset address of the source operand to the destination operand. The source operand must be a memory operand and the destination operand must be a 16-bit general, pointer, or index register.

—LDS (load pointer into DS) transfers a "pointer-object" (i.e., a 32-bit object containing an offset address and a segment address) from the source operand (which must be a memory operand) to a pair of destination registers. The segment address is transferred to the DS segment register. The offset address must be transferred to a 16-bit general, pointer, or index register.

—LES (load pointer into ES) is similar to LDS except that the segment address is transferred to the ES segment register.

Flag Register Transfers. Four flag register transfer operations are provided:

—LAHF (load AH with flags) transfers the flag registers SF, ZF, AF, PF, and CF (the 8080 flags) into specific bits of the AH register.

—SAHF (store AH into flags) transfers specific bits of the AH register to the flag registers, SF, ZF, AF, PF, and CF.

—PUSHF (push flags) decrements the SP register by two and transfers all of the flag registers into specific bits of the stack element addressed by SP.

—POPF (pop flags) transfers specific bits of the stack element addressed by the SP register to the flag registers and then increments SP by two.

ARITHMETIC

The 8086 provides the four basic mathematical operations in a number of different varieties. Both 8- and 16-bit operations and both signed and unsigned arithmetic are provided. Standard twos complement representation of signed values is used. The addition and subtraction operations serve as both signed and unsigned operations. In these cases the flag settings allow the distinction between signed and unsigned operations to be made (see Conditional Transfer). Correction operations are provided to allow arithmetic to be performed directly on unpacked decimal digits or on packed decimal representations.

Flag Register Settings. Six flag registers are set or cleared by arithmetic operations to reflect certain properties of the result of the operation. They generally follow these rules:

—CF is set if the operation results in a carry out of (from addition) or a borrow into (from subtraction) the high-order bit of the result; otherwise CF is cleared.

—AF is set if the operation results in a carry out of (from addition) or a borrow into (from subtraction) the low-order four bits of the result; otherwise AF is cleared.

—ZF is set if the result of the operation is zero; otherwise ZF is cleared.

—SF is set if the high-order bit of the result of the operation is set; otherwise SF is cleared.

—PF is set if the modulo 2 sum of the low-order eight bits of the result of the operation is 0 (even parity); otherwise PF is cleared (odd parity).

—OF is set if the operation results in a carry into the high-order bit of the result but not a carry out of the high-order bit, or vice versa; otherwise OF is cleared.

Addition. Five addition operations are provided:

—ADD performs an addition of the two source operands and returns the result to one of the operands.

—ADC (add with carry) performs an addition of the two source operands, adds one if the CF flag is found previously set, and returns the result to one of the operands.

—INC (increment) performs an addition of the source operand and one, and returns the result to the operand.

—AAA (unpacked BCD (ASCII) adjust for addition) performs a correction of the result in AL of adding two unpacked decimal operands, yielding an unpacked decimal sum.

—DAA (decimal adjust for addition) performs a correction of the result in AL of adding two packed decimal operands, yielding a packed decimal sum.

Subtraction. Seven subtraction operations are provided:

—SUB performs a subtraction of the two source operands and returns the result to one of the operands.

—SBB (subtract with borrow) performs a subtraction of the two source operands, subtracts one if the CF flag is found previously set, and returns the result to one of the operands.

—DEC (decrement) performs a subtraction of one from the source operand and returns the result to the operand.

—NEG (negate) performs a subtraction of the source operand from zero and returns the result to the operand.

—CMP (compare) performs a subtraction of the two source operands causing the flags to be affected but does not return the result.

—AAS (unpacked BCD (ASCII) adjust for subtraction) performs a correction of the result in AL of subtracting two unpacked decimal operands, yielding an unpacked decimal difference.

—DAS (decimal adjust for subtraction) performs a correction of the result in AL of subtracting two packed decimal operands, yielding a packed decimal difference.

Multiplication. Three multiplication operations are provided:

—MUL performs an unsigned multiplication of the accumulator (AL or AX) and the source operand, returning a double length result to the accumulator and its extension (AL and AH for 8-bit operation, AX and DX for 16-bit operation). CF and OF are set if the top half of the result is non-zero.

—IMUL (integer multiply) is similar to MUL except that it performs a signed multiplication. CF and OF are set if the top half of the result is not the sign-extension of the low half of the result.

—AAM (unpacked BCD (ASCII) adjust for multiply) performs a correction of the result in AX of multiplying two unpacked decimal operands, yielding an unpacked decimal product.

Division. Three division operations are provided and two sign-extension operations to support signed division:

—DIV performs an unsigned division of the accumulator and its extension (AL and AH for 8-bit operation, AX and DX for 16-bit operation) by the source operand and

returns the single length quotient to the accumulator (AL or AX), and returns the single length remainder to the accumulator extension (AH or DX). The flags are undefined. Division by zero generates an interrupt of type 0.
—IDIV (integer division) is similar to DIV except that it performs a signed division.
— AAD (unpacked BCD (ASCII) adjust for division) performs a correction of the dividend in AL before dividing two unpacked decimal operands, so that the result will yield an unpacked decimal quotient.
—CBW (convert byte to word) performs a sign extension of AL into AH.
—CWD (convert word to double word) performs a sign extension of AX into DX.

LOGIC

The 8086 provides the basic logic operations for both 8- and 16-bit operands.

Single-Operand Operations. Three single-operand logical operations are provided:
—NOT forms the ones complement of the source operand and returns the result to the operand. Flags are not affected.
—Shift operations of four varieties are provided for memory and register operands, SHL (shift logical left), SHR (shift logical right), SAL (shift arithmetic left), and SAR (shift arithmetic right). Single bit shifts, and variable bit shifts with the shift count taken from the CL register are available. The CF flag becomes the last bit shifted out; OF is defined only for shifts with count of 1, and set if the final sign bit value differs from the previous value of the sign bit; and PF, SF, and ZF are set to reflect the result value.
—Rotate operations of four varieties are provided for memory and register operands, ROL (rotate left), ROR (rotate right), RCL (rotate through CF left), and RCR (rotate through CF right). Single bit rotates, and variable bit rotates with the rotate count taken from the CL register are available. The CF flag becomes the last bit rotated cut; OF is defined only for shifts with count of 1, and is set if the final sign bit value differs from the previous value of the sign bit.

Two-Operand Operations. Four two-operand logical operations are provided. The CF and OF flags are cleared on all operations; SF, PF, and ZF reflect the result.
—AND performs the bitwise logical conjunction of the two source operands and returns the result to one of the operands.
—TEST performs the same operations as AND causing the flags to be affected but does not return the result.
—OR performs the bitwise logical inclusive disjunction of the two source operands and returns the result to one of the operands.
—XOR performs the bitwise logical exclusive disjunction of the two source operands and returns the result to one of the operands.

STRING MANIPULATION

One-byte instructions perform various primitive operations for the manipulation of byte and word strings (sequences of bytes or words). Any primitive operation can be performed repeatedly in hardware by preceding its instruction with a repeat prefix. The single-operation forms may be combined to form complex string operations with repetition provided by iteration operations.

Hardware Operation Control. All primitive string operations use the SI register to address the source operands, which are assumed to be in the current data segment. The DI register is used to address the destination operands, which reside in the current extra segment. If the DF flag is cleared the operand pointers are incremented after each operation, once for byte operations and twice for word operations. If the DF flag is set the operand pointers are decremented after each operation. See Processor Control for setting and clearing DF.

Any of the primitive string operation instructions may be preceded with a one-byte prefix indicating that the operation is to be repeated until the operation count in CX is satisfied. The test for completion is made prior to each repetition of the operation. Thus, an initial operation count of zero will cause zero executions of the primitive operation.

The repeat prefix byte also designates a value to compare with the ZF flag. If the primitive operation is one which affects the ZF flag, and the ZF flag is unequal to the designated value after any execution of the primitive operation, the repetition is terminated. This permits the scan operation to serve as a scan-while or a scan-until.

During the execution of a repeated primitive operation the operand pointer registers (SI and DI) and the operation count register (CX) are updated after each repetition, whereas the instruction pointer will retain the offset address of the repeat prefix byte (assuming it immediately precedes the string operation instruction). Thus, an interrupted repeated operation will be correctly resumed when control returns from the interrupting task.

You should try to avoid using the two other prefix bytes with a repeat-prefixed string instruction. One overrides the default segment addressing for the SI operand (Section 2.3.1), and one locks the bus to prohibit access by other bus masters. (See Section 2.3.4.) Execution of the repeated string operation will not resume properly following an interrupt if more than one prefix is present preceding the string primitive. Execution will resume one byte before the primitive (presumably where the repeat prefix resides), thus ignoring the additional prefixes.

Primitive String Operations. Five primitive string operations are provided:
—MOVB (or MOVW) transfers a byte (or word) operand from the source operand to the destination operand. As a repeated operation this provides for moving a string from one location in memory to another.
—CMPB (or CMPW) subtracts the destination byte (or word) operand from the source operand and affects the flags but does not return the result. As a repeated operation this provides for comparing two strings. With the appropriate repeat prefix it is possible to determine after which string element the two strings become unequal, thereby establishing an ordering between the strings.
—SCAB (or SCAW) subtracts the destination byte (or word) operand from AL (or AX) and affects the flags but does not return the result. As a repeated operation this provides for scanning for the occurrence of, or departure from a given value in the string.

—LODB (or LODW) transfers a byte (or word) operand from the source operand to AL (or AX). This operation ordinarily would not be repeated.

—STOB (or STOW) transfers a byte (or word) operand from AL (or AX) to the destination operand. As a repeated operation this provides for filling a string with a given value.

In all cases above, the source operand is addressed by SI and the destination operand is addressed by DI.

Software Operation Control. The repeat prefix provides for rapid iteration in a hardware-repeated string operation. The iteration control operations (see Iteration Control, Section 3.6.3) provide this same control for implementing software loops to perform complex string operations. These iteration operations provide the same operation count update, operation completion test, and ZF flag tests that the repeat prefix provides.

By combining the primitive string operations and iteration control operations with other operations, it is possible to build sophisticated yet efficient string manipulation routines. One instruction that is particularly useful in this context is XLAT; it permits a byte fetched from one string to be translated before being stored in a second string, or before being operated upon in some other fashion. The translation is performed by using the value in the AL register as an index into a table pointed at by the BX register. The translated value obtained from the table then replaces the value initially in the AL register.

As an example of the use of the primitive string operations and iteration control operations to implement a complex string operation, consider the following application: An input driver must translate a buffer of EBCDIC characters into ASCII, and transfer characters until one of several different EBCDIC control characters is encountered. The transferred ASCII string is to be terminated with an EOT character. To accomplish this, SI is initialized to point to the beginning of the EBCDIC buffer, DI is initialized to point to the beginning of the buffer to receive the ASCII characters, BX is made to point to an EBCDIC to ASCII translation table, and CX is initialized to contain the length of the EBCDIC buffer (possibly empty). The translation table contains the ASCII equivalent for each EBCDIC character, perhaps with ASCII NULs for illegal characters. The EOT code is placed into those entries in the table corresponding to the desired EBCDIC stop characters. The 8086 instruction sequence to implement this example is the following:

```
         JCXZ    Empty   ;skip if input buffer empty
  Next:  LODB    Ebcbuf  ;fetch next EBCDIC character
         XLAT    Table   ;translate it to ASCII
         CMP     AL,EOT  ;test for the EOT
         STOB    Ascbuf  ;transfer ASCII character
         LOOPNE  Next    ;continue if not EOT
         •
         •
         •
  Empty:
```

The body of this loop requires seven bytes of code.

CONTROL TRANSFER

Four classes of control transfer operations may be distinguished: calls, jumps, and returns; conditional transfers; iteration control; and interrupts.

All control transfer operations cause, perhaps upon a certain condition, the program execution to continue at some new location in memory, possibly in a new code segment.

Calls, Jumps, and Returns. Two basic varieties of calls, jumps, and returns are provided—those which transfer control within the current code segment, and those which transfer control to an arbitrary code segment, which then becomes the current code segment. Both direct and indirect transfers are supported; indirect transfers make use of the standard addressing modes as described in Section 2.3.1.

The three transfer operations are described below:

—CALL pushes the offset address of the next instruction onto the stack (in the case of an inter-segment transfer the CS segment register is pushed first) and then transfers control to the target operand.

—JMP transfers control to the target operand.

—RET transfers control to the return address saved by a previous CALL operation, and optionally may adjust the SP register so as to discard stacked parameters.

Intra-segment direct calls and jumps specify a self-relative direct displacement, thus allowing position independent code. A shortened jump instruction is available for transfers within ± 128 bytes about the instruction for code compaction.

Conditional Jumps. The conditional transfers of control perform a jump contingent upon various Boolean functions of the flag registers. The destination must be within a 256-byte range centered about the instruction. Table 2-1 shows the available instructions, the conditions associated with them, and their interpretation.

Iteration Control. The iteration control transfer operations perform leading- and trailing-decision loop control. The destination of iteration control transfers must be within a 256-byte range centered about the instruction. These operations are particularly useful in conjunction with the string manipulation operations.

There are four iteration control transfer operations provided:

—LOOP decrements the CX ("count") register by one and transfers if CX is not zero.

—LOOPZ (also called LOOPE) decrements the CX register by one and transfers if CX is not zero and the ZF flag is set (loop while zero or loop while equal).

—LOOPNZ (also called LOOPNE) decrements the CX register by one and transfers if CX is not zero and the ZF flag is cleared (loop while not zero or loop while not equal).

—JCXZ transfers if the CX register is zero.

Interrupts. Program execution control may be transferred by means of operations similar in effect to that of external interrupts (see Section 2.4.2). All interrupts perform a transfer by pushing the flag registers onto the stack (as in PUSHF), and then performing an indirect call (of the intersegment variety) through an element of an interrupt transfer vector located at absolute locations 0 through 3FFH. This vector contains a four-byte element for each of up to 256 different interrupt types.

There are three interrupt transfer operations provided:

—INT pushes the flag registers (as in PUSHF), clears the TF and IF flags, and transfers control with an indirect call through any one of the 256 vector elements. A one-byte

form of this instruction is available for interrupt type 3.
- —INTO pushes the flag registers (as in PUSHF), clears the TF and IF flags, and transfers control with an indirect call through vector element 4 if the OF flag is set (trap on overflow). If the OF flag is cleared no operation takes place.
- —IRET transfers control to the return address saved by a previous interrupt operation and restores the saved flag registers (as in POPF).

See Section 2.3.7 for further details on interrupt operations.

PROCESSOR CONTROL

Various instructions and mechanisms are provided for control and operation of the processor and its interaction with its environment.

Flag Operations. There are seven operations provided which operate directly on individual flag registers:
- —CLC clears the CF flag.
- —CMC complements the CF flag.
- —STC sets the CF flag.
- —CLD clears the DF flag, causing the string operations to auto-increment the operand pointers.
- —STD sets the DF flag, causing the string operations to auto-decrement the operand pointers.
- —CLI clears the IF flag, disabling external interrupts (except for the non-maskable external interrupt).
- —STI sets the IF flag, enabling external interrupts after the execution of the next instruction.

Processor Halt. The HLT instruction causes the 8086 processor to enter its halt state. The halt state is cleared by an enabled external interrupt or RESET.

Processor Wait. The WAIT instruction causes the processor to enter a wait state if the signal on its TEST pin is not asserted. The wait state may be interrupted by an enabled external interrupt. When this occurs the saved code location is that of the WAIT instruction, so that upon return from the interrupting task the wait state is reentered. The wait state is cleared and execution resumed when the TEST signal is asserted. Execution resumes without allowing external interrupts until after the execution of the next instruction. This instruction allows the processor to synchronize itself with external hardware.

Processor Escape. The ESC instruction provides a mechanism by which other processors may receive their instructions from the 8086 instruction stream and make use of the 8086 addressing modes. The 8086 processor does no operation for the ESC instruction other than to access a memory operand.

Bus Lock. A special one-byte prefix may precede any instruction causing the processor to assert its bus-lock signal for the duration of the operation caused by that instruction. This has use in multiprocessing applications as discussed in Section 2.3.4.

Single Step. When the TF flag register is set the processor generates a type 1 interrupt after the execution of each instruction. During interrupt transfer sequences caused by any type of interrupt, the TF flag is cleared after the push-flags step of the interrupt sequence. No instructions are provided for setting or clearing TF directly. Rather, the flag register image saved on the stack by a previous interrupt operation must be modified, so that the subsequent interrupt return operation (IRET) restores TF set. This allows a diagnostic task to single-step through a task under test, while still executing normally itself.

If the single-stepped instruction itself clears the TF flag, the type 1 interrupt will still occur upon completion of the single-stepped instruction. If the single-stepped instruction generates an interrupt or if an enabled external interrupt occurs prior to the completion of the single-stepped instruction, the type 1 interrupt sequence will occur after the interrupt sequence of the generated or external interrupt, but before the first instruction of the interrupt service routine is executed.

2.2.5 Arithmetic-Logic Unit (ALU) (Figure 2-9)

The ALU contains the following registers:

Two internal 16-bit operand registers, not user-accessible.
The flag registers, containing nine flags. (See Section 2.2.3.)

Arithmetic, logic, and rotate operations are performed by the ALU.

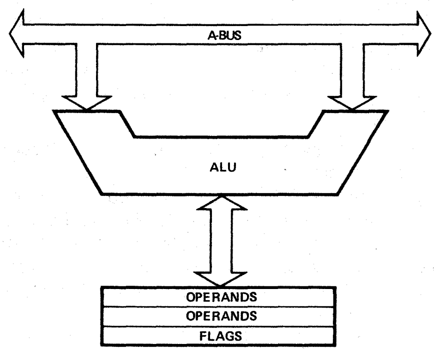


Figure 2-9. Arithmetic-Logic Unit

2.2.6 Memory Organization

The 8086 employs a 20-line address bus on which to locate a byte or word to be referenced in memory or I/O, and thus can access 2^{20} bytes (1 megabyte, or 1,048,576D bytes). Each location is an 8-bit space. Word (16-bit) operands consisting of consecutive bytes can fall on either even or odd address boundaries. The processor provides two signals, $\overline{BHE}$ and AO, to selectively enable an odd location, an even location, or both. For address and data operands, the least significant byte of the word will be stored in the lower valued address location and the most significant byte in the next higher address location. For maximum performance, 16-bit data should be located with the least significant byte in an even address. Otherwise two memory cycles will be run by the BIU to access the data. Except for the performance penalty, this double access is transparent. The instruction stream is fetched from memory as words and queued internally by the BIU at the byte level.

TABLE 2-1
8086 INSTRUCTION SET SUMMARY

DATA TRANSFER

MOV = **Move**:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate to register	1 0 1 1 w reg	data	data if w=1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		

PUSH = **Push**:

Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m
Register	0 1 0 1 0 reg	
Segment register	0 0 0 reg 1 1 0	

POP = **Pop**:

Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m
Register	0 1 0 1 1 reg	
Segment register	0 0 0 reg 1 1 1	

XCHG = **Exchange**:

Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m
Register with accumulator	1 0 0 1 0 reg	

IN/INW = **Input to AL/AX from**:

Fixed port	1 1 1 0 0 1 0 w	port
Variable port	1 1 1 0 1 1 0 w	

OUT/OUTW = **Output from AL/AX to**:

Fixed port	1 1 1 0 0 1 1 w	port
Variable port	1 1 1 0 1 1 1 w	
XLAT=Translate byte to AL	1 1 0 1 0 1 1 1	
LEA=Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m
LDS=Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m
LES=Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m
LAHF=Load AH with flags	1 0 0 1 1 1 1 1	
SAHF=Store AH into flags	1 0 0 1 1 1 1 0	
PUSHF=Push flags	1 0 0 1 1 1 0 0	
POPF=Pop flags	1 0 0 1 1 1 0 1	

ARITHMETIC

ADD = **Add**:

Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	

ADC = **Add with carry**:

Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	

INC = **Increment**:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m
Register	0 1 0 0 0 reg	
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1	
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1	

SUB = **Subtract**:

Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

SBB = **Subtract with borrow**

Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

DEC = **Decrement**:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG=Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		

CMP **Compare**:

Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS=ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS=Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL=Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL=Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM=ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV=Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV=Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD=ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW=Convert byte to word	1 0 0 1 1 0 0 0			
CWD=Convert word to double word	1 0 0 1 1 0 0 1			

LOGIC

NOT=Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m
SHL/SAL=Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m
SHR=Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m
SAR=Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m
ROL=Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m
ROR=Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m
RCL=Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m
RCR=Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m

AND = **And**:

Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	

TEST = **And function to flags, no result**:

Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	

OR = **Or**:

Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	

XOR = **Exclusive or**:

Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	

STRING MANIPULATION

REP=Repeat	1 1 1 1 0 0 1 z
MOVB/MOVW=Move byte/word	1 0 1 0 0 1 0 w
CMPB/CMPW=Compare oyte/word	1 0 1 0 0 1 1 w
SCAB/SCAW=Scan byte/word	1 0 1 0 1 1 1 w
LODB/LODW=Load byte/wd to AL/AX	1 0 1 0 1 1 0 w
STOB/STOW=Stor byte/wd frm AL/A	1 0 1 0 1 0 1 w

TABLE 2-1 (Cont.)
8086 INSTRUCTION SET SUMMARY

CONTROL TRANSFER

	76543210	76543210	76543210
CALL = Call:			
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	
JMP = Unconditional Jump:			
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within segment-short	1 1 1 0 1 0 1 1	disp	
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET = Return from CALL:			
Within segment	1 1 0 0 0 0 1 1		
Within seg. adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment, adding immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ=Jump on equal/zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE=Jump on less/not greater or equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG=Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE=Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA=Jump on below or equal/ not above	0 1 1 1 0 1 1 0	disp	
JP/JPE=Jump on parity/parity even	0 1 1 1 1 0 1 0	disp	
JO=Jump on overflow	0 1 1 1 0 0 0 0	disp	
JS=Jump on sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ=Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE=Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG=Jump on not less or equal/ greater	0 1 1 1 1 1 1 1	disp	

	76543210	76543210
JNB/JAE Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp
JNBE/JA=Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp
JNP/JPO=Jump on not par/par odd	0 1 1 1 1 0 1 1	disp
JNO Jump on not overflow	0 1 1 1 0 0 0 1	disp
JNS Jump on not sign	0 1 1 1 1 0 0 1	disp
LOOP=Loop CX times	1 1 1 0 0 0 1 0	disp
LOOPZ/LOOPE=Loop while zero/equal	1 1 1 0 0 0 0 1	disp
LOOPNZ/LOOPNE Loop while not zero/equal	1 1 1 0 0 0 0 0	disp
JCXZ=Jump on CX zero	1 1 1 0 0 0 1 1	disp
INT = Interrupt		
Type specified	1 1 0 0 1 1 0 1	type
Type 3	1 1 0 0 1 1 0 0	
INTO=Interrupt on overflow	1 1 0 0 1 1 1 0	
IRET=Interrupt return	1 1 0 0 1 1 1 1	

PROCESSOR CONTROL

CLC=Clear carry	1 1 1 1 1 0 0 0	
CMC=Complement carry	1 1 1 1 0 1 0 1	
STC=Set carry	1 1 1 1 1 0 0 1	
CLD=Clear direction	1 1 1 1 1 1 0 0	
STD=Set direction	1 1 1 1 1 1 0 1	
CLI=Clear interrupt	1 1 1 1 1 0 1 0	
STI=Set interrupt	1 1 1 1 1 0 1 1	
HLT=Halt	1 1 1 1 0 1 0 0	
WAIT=Wait	1 0 0 1 1 0 1 1	
ESC=Escape (to external device)	1 1 0 1 1 x	mod x r/m
LOCK=Bus lock prefix	1 1 1 1 0 0 0 0	

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction

if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

2.2.7 Input/Output Organization

The 8086 provides 64k addressable input or output ports. I/O space is addressed as if it were a single memory segment, without the use of segment registers. Input/output physical addresses are 20-bits in length, but the high order four bits are always zero. Ports may be 8 or 16 bits in size, and 16-bit ports may be located at either odd or even addresses, but as with memory fetches, faster operation is achieved if the high and low bytes of 16-bit ports are aligned with the high and low lines of the bus. Even-addressed bytes are transferred on the D7-D0 bus lines and odd-addressed bytes on D15-D8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even. The M/IO line from the processor is used for bus switching. Variable I/O instructions which use register DX as a pointer have full address capability. Direct I/O instructions may directly address one or a pair of the first 256 I/O byte locations of the I/O address space.

2.3 How The 8086 Works

2.3.1 Memory Addressing Scheme

Memory addresses are logically subdivided into segments of 64k bytes each, which can be allocated to code, data, and stack. The boundaries of such segments must coincide with integral 16-byte intervals. Segments may be overlapped, within this constraint, but each segment must begin at an address which is evenly divisible by sixteen (i.e., the low-order four bits of a segment address are zero). At any given moment, the contents of four of these segments are immediately addressable. The four segments are called the current code segment, the current data segment, the current stack segment, and the current extra segment. These segments need not be unique and may overlap. The high-order sixteen bits of the address of each current segment are held in a dedicated 16-bit segment register, and are called the segment address.

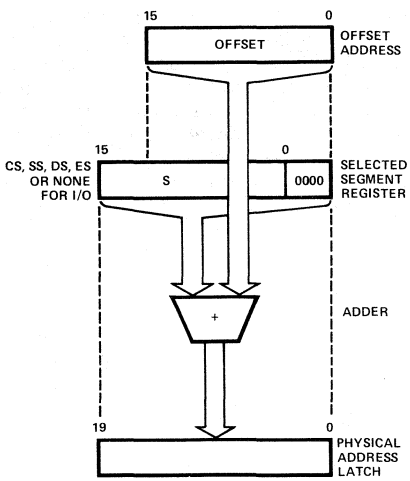


Figure 2-10. Address Generation with Segment Registers

USE OF SEGMENT REGISTERS

Bytes or words within a segment are addressed using 16-bit offset addresses, or effective addresses (EA), within the 64k byte segment. A 20-bit physical address is constructed by adding the 16-bit offset address to the 16-bit segment address with four low-order zero bits appended. That is, they are left-shifted four places. This is illustrated in Figure 2-10.

The four segment registers contain the current segment addresses:

- CS Code Segment
- DS Data Segment
- ES Extra Segment (alternate data location)
- SS Stack Segment

The segment register used when generating a data address (normally DS) can be overridden during the execution of most instructions if the instruction is preceded by the special one-byte segment override prefix. The prefix indicates that another segment register is to be used for all data references during the execution of that instruction.

The segment register used when generating a stack address (normally SS) can similarly be overridden with a segment override prefix providing the memory address is computed from the contents of BP; a stack address computed from the contents of SP cannot be overridden, and hence will always use SS. Code segment register CS is always used when generating a code address and cannot be overridden. Table 2-2 summarizes.

TABLE 2-2
USE OF SEGMENT OVERRIDE

Default	With Override Prefix
IP + CS = code address	Never
SP + SS = stack address	Never
BP + SS = stack address or stack marker	BP + DS or ES, or CS
EA + DS = data address	EA + ES, SS, or CS

2.3.2 Bus Multiplexing Scheme

The 8086 uses a time-multiplexed address and data bus. This technique provides the most efficient use of pins on the processor while permitting the use of a standard 40-lead package. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. Or, if your system design requires a nonmultiplexed bus the address/data bus can be demultiplexed at the processor with a single set of address latches.

2.3.3 Bus Cycle Timing (Figure 2-11)

Each processor bus cycle consists of at least four clock cycles. These are referred to as T1, T2, T3, and T4. The address is emitted from the processor during T1. Data transfer occurs on the bus during T3 and T4. T2 is the period during which the direction of the bus is changed for read operations. In the event that a "NOT READY" indication is given by the addressed device, WAIT states (TW) are inserted between T3 and T4. Each TW cycle inserted is one clock cycle in duration. Idle states (TI) can occur between 8086-driven bus cycles. These inactive CLK cycles are used by the processor for internal functions.

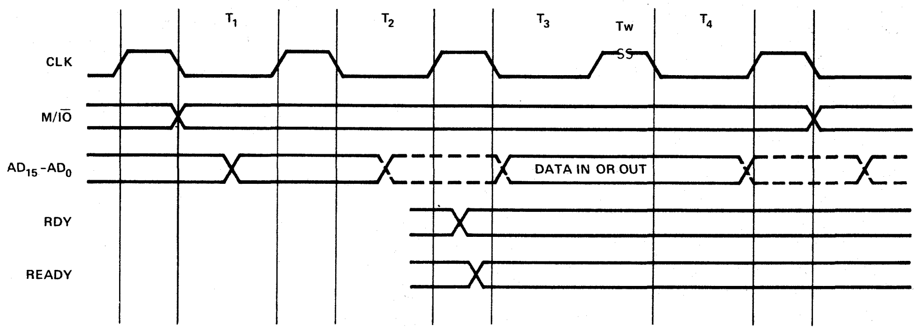


Figure 2-11. 8086 CPU Bus Timing

READ SEQUENCE (FIGURE 2-12)

The read cycle begins during T1, with the assertion of address latch enable signal ALE①. The trailing edge of ALE② is used to latch the address information, which is present on the local bus at this time③, into the 8282 or 8283 latch. The $\overline{BHE}$④ and A0 signals address the low-order byte, the high-order byte, or the whole word. From T1 to T4, the M/$\overline{I/O}$ signal⑤ selects memory or input-output (I/O) operation. At T2, the address is removed from the local bus and the processor bus drivers go to the high-impedance state⑥. The read control signal, ($\overline{RD}$)⑦, is also asserted during T2. $\overline{RD}$ causes the addressed device to enable its bus drivers to the now-released local bus. At some later time, valid data will become available on the bus⑧ and the addressed device will then drive the READY line high⑨. When the 8086 subsequently returns $\overline{RD}$ to the high level⑩, the addressed device will then tristate its bus drivers, relinquishing the bus again ⑪. If an 8286 or 8287 transceiver is used to buffer the local bus, it is serviced with DT/$\overline{R}$⑫ and $\overline{DEN}$⑬ signals by the 8086.

WRITE SEQUENCE (FIGURE 2-13)

The write cycle, like the read cycle, begins with the assertion of ALE① and the emission of an address②. And again the preconditioned M/$\overline{I/O}$ signal③ indicates either memory or I/O write operation is to occur. In the T2 that immediately follows the issuance of the address, the processor emits the data to be written into the addressed location④. This data remains valid on the bus until at least the middle of T4⑤. Write signal $\overline{WR}$ goes low at the beginning of T2 (somewhat earlier than $\overline{RD}$ would occur)⑥, and remains active throughout T2, T3, and TW.

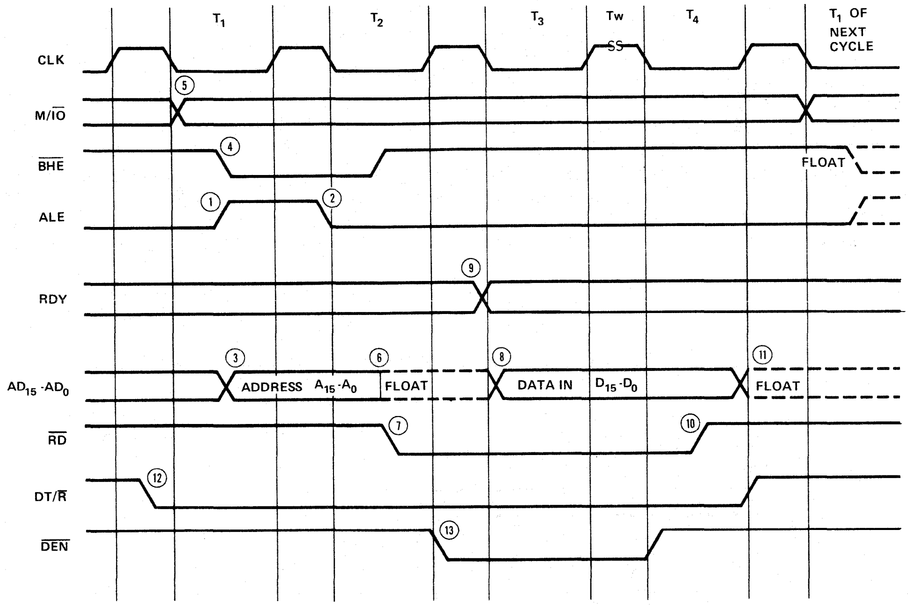


Figure 2-12. Read Cycle Timing

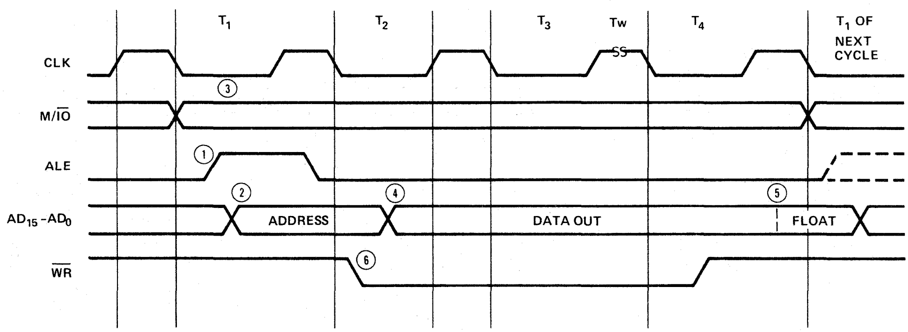


Figure 2-13. Write Cycle Timing

MEDIUM-COMPLEXITY SYSTEMS (FIGURE 2-14)

In medium-complexity systems (with MN/$\overline{MX}$ grounded to V_{SS}, and an 8288 bus controller in the system), signals ALE, $\overline{DEN}$, and DT/$\overline{R}$ are generated by the 8288 instead of by the 8086, but their timing remains the same as in the minimum system, in which they are available on 8086 pins. (See chapters 3 and 5 for circuit details.) The 8086 supplies status outputs $\overline{S0}$, $\overline{S1}$, and $\overline{S2}$ to the 8288, specifying read (code, data, or I/O), write (date or I/O), interrupt acknowledge, or software halt. The 8288, in addition to the control signals, generates two types of write strobes, normal and advanced.

2.3.4 Lock

When directly consecutive bus cycles are required for the execution of an instruction, the processor sends an active low on the $\overline{LOCK}$ line to external bus arbitration logic. This occurs in the clock cycle following the one in which the LOCK prefix instruction is found and decoded by the EU, and remaining through the instruction execution following the LOCK prefix. The bus arbitration logic, in turn, issues a $\overline{BUSY}$ to other processors on the bus. While $\overline{LOCK}$ is active, all interrupts are masked externally. During this period, any request to the 8086 on its $\overline{RQ}/\overline{GT}$ line will be received and latched in, but will not be answered until the end of the LOCK.

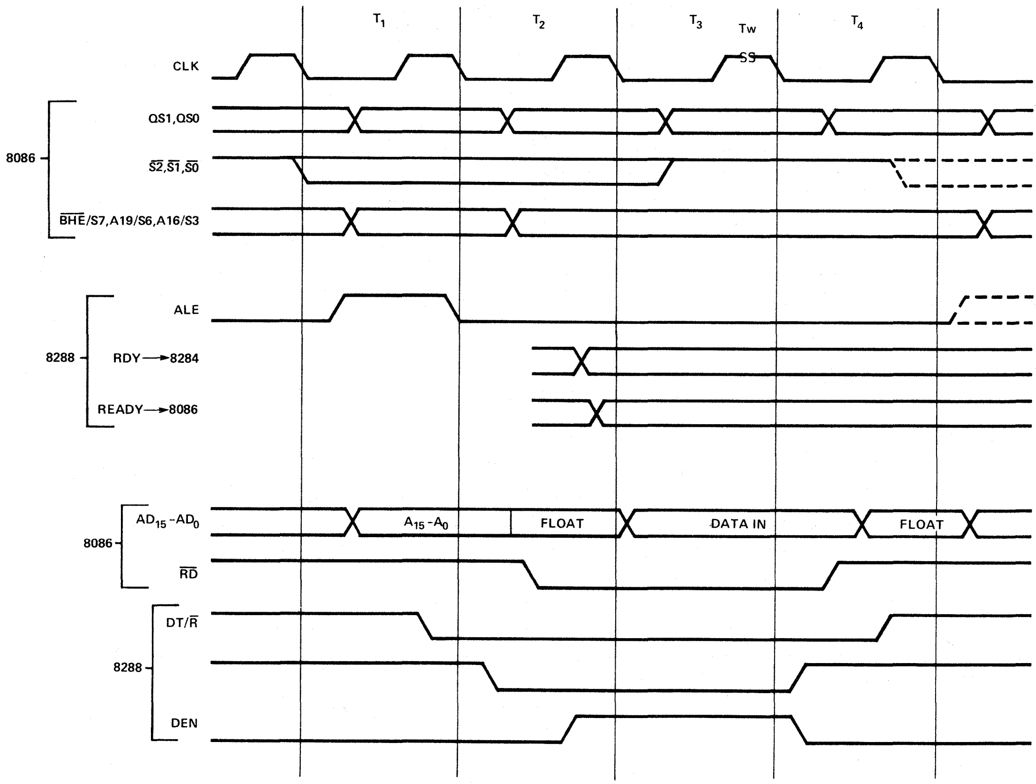


Figure 2-14. 8086 CPU Bus Timing, Medium Complexity System

2.3.5 Reset

When the 8086 is powered up, or at other times when it is desired to initialize the processor, its RESET pin 21 should be held high for a period of at least four clock cycles. Upon powering up, the +5 V source and RESET should be simultaneously applied to the chip with RESET held high for at least 50μs after the power stabilizes. Upon receipt of RESET, the processor ceases operation, and remains dormant for the duration of the pulse. The low-going transition then initiates a sequence which requires approximately 10 clock cycles to execute before normal operation commences. This sequence ends with registers initialized as follows:

Flags	= 0000H	(to disable interrupts and single-stepping)
CS	= FFFFH	(to begin execution at FFFF0H)
IP	= 0000H	
DS	= 0000H	
SS	= 0000H	
ES	= 0000H	

No other registers are acted upon during reset.

RESERVED MEMORY LOCATIONS

Intel Corporation reserves the use of memory locations FFFF0H through FFFFFH (with the exception of FFFF0H, mentioned in the preceding table) for Intel hardware and software products. If you use these locations for some other purpose, you may preclude compatibility of your system with certain of these products.

RESERVED INPUT/OUTPUT LOCATIONS

Intel Corporation reserves the use of input/output locations F8H through FFH for Intel hardware and software products. Users who wish to maintain compatibility with present and future Intel products should not use these locations.

2.3.6 Halt

When a software HALT instruction is executed the processor will issue status on $\overline{S2}$, $\overline{S1}$, and $\overline{S0}$ to indicate that it is in the HALT state. ALE is issued once, together with status, as the HALT state is entered. The 8086 will not leave the HALT state when a local bus "hold" is entered while in HALT. An interrupt request or RESET will force the 8086 out of the HALT state.

2.3.7 Interrupts

Interrupts may be software-initiated or hardware-initiated. Software-initiated interrupts are described in Chapter 4. Hardware interrupts may be of either of two types: nonmaskable or maskable.

NONMASKABLE INTERRUPT (NMI)

Interrupts which arrive on pin 17 of the 8086 have a higher priority than those which arrive on pin 18 (INTR) or are software-initiated. Activation of NMI occurs on the upward transition, and causes an unconditional jump to a subroutine in memory. It is not disabled by the clear interrupt instruction. NMI must remain high for greater than 2 clock cycles, but need not be synchronous with the clock. It is internally latched by the 8086, and is serviced at the end of any instruction in process when received, except in the case of block moves, which can be interrupted between whole moves. The greatest delay in response to NMI will occur during multiply, divide, and multibit-shift instructions. The downward transition of NMI may occur at any time after its 2-cycle latch time, but must be debounced to avoid retriggering.

INTERRUPT REQUEST (INTR)

The 8086 has an interrupt request line (pin 18) which can be masked internally by software instructions which reset interrupt flag IF. INTR is level-triggered at the leading edge of CLK, so should be present during the clock pulse preceding the end of an instruction or block move operation. During the interrupt sequence, further INTR signals, if present, will be ignored, i.e., CLI is invoked as a part of the response to any interrupt. The set of flags which is automatically pushed onto stack reflects the state of the processor prior to the interrupt; thus, when the flags are restored following the interrupt response routine, interrupts are normally enabled again automatically. It is therefore desirable to remove the INTR signal before the interrupt response completes.

During the response sequence the processor will execute two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the $\overline{LOCK}$ signal from T2 of the first bus cycle until T2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte of information is read from bus lines D7-D0 as supplied by the interrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into an interrupt vector look up table (located in system memory, see Section 4 for details). An INTR signal left HIGH will be continually responded to each time the IF flag is set by the INTERRUPT RETURN, IRET, instruction, which includes a stack pop and restores the flags.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. In the second of two successive INTA cycles.

RESERVED INTERRUPTS

Intel Corporation reserves the use of interrupts 0-31 (locations 00H through 7FH) for Intel hardware and software products. Users who wish to maintain compatibility with present and future Intel products should not use these locations.

Interrupts 0 through 4 (00H-13H) currently have dedicated hardware functions as defined below.

Interrupt	Location	Function
0	00H-03H	Divide by zero
1	04H-07H	Single step
2	08H-0BH	Non-maskable interrupt
3	0CH-0FH	One-byte interrupt instruction
4	10H-13H	Interrupt on overflow

CHAPTER 3

System Operation and Interfacing

CHAPTER 3
SYSTEM OPERATION AND INTERFACING

3.1 System Mode Configurations

The way you interface peripherals and memory to the 8086 can vary widely, depending upon the type and complexity of the system in which they will be used. The minimum-complexity system, as shown in Figure 3-1, consists of an 8086 cpu and 8284 clock generator, a pair of 8282 address latches, and memory and I/O ports. If the system you are dealing with does not exceed the specification current and capacitive loading requirement, the 8086 can drive the system bus without buffers, but if you need more than the loads it can handle, 8286 transceivers can be interposed on the data bus for additional drive capability. You would want to isolate with transceivers, in any case, if the bus must extend off the cpu board. A minimum system as shown can address up to 32K of memory and 32K of I/O mapped ports. If an additional 8282 address latch is used, the full 1 megabyte addressing capability of the 8086 can be used.

A step higher in versatility and in complexity is the fully buffered system as shown in Figure 3-2. By introducing an 8288 bus controller into the system and strapping the 8086 MN/MX to ground, you can isolate control, address and data buses. You can also separate I/O from memory and other system peripherals, and thereby utilize the full addressing capacity of the 8086, of 1 megabyte of memory space and 64 kilobytes of I/O, concurrently. Add to this an 8259A priority interrupt controller and the system flexibility is considerably greater.

The fully buffered MCS-86 bus system provides for an extensive and modular expansion capability. This bus is patterned after the Intel Multibus™. (See Intel Application Note AP-28, entitled "Intel Multibus™ Interface") A typical MCS-86 system with single 8086 CPU, and a buffered bus, has a number of uniquely addressed slave modules, all connected to the bus. In this system, the CPU can access any slave. The bus acts as the common point for all modules; its DC and timing parameters must be adhered to by the master and all slaves that use it. Its timing signals and their relationships are shown in the data sheet in Section 5.

At the heart of the MCS-86 expandable, buffered data bus is the 8288 bus controller. With the MN/MX pin of the 8086 strapped to ground, the CPU now sends status and control input signals to the 8288. (See 8086 and 8288 data sheets.) In this system, the CPU does not issue the device selects, the read/write commands, and the interrupt acknowledge. These signals are regenerated in the 8288, along with some further control requirements of Multibus™ operation.

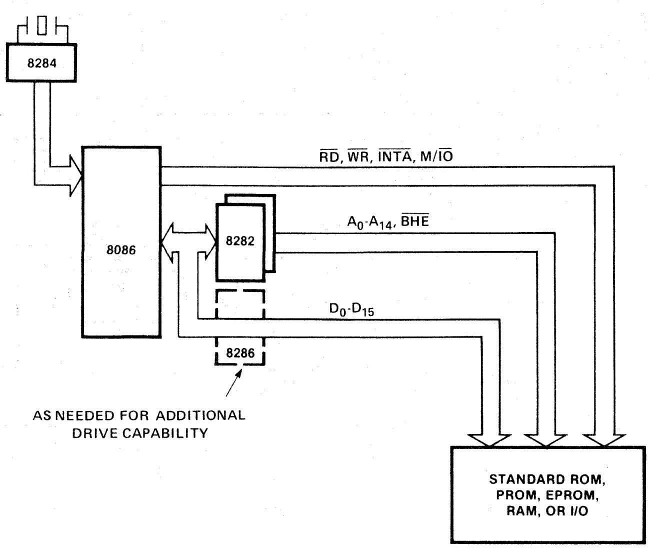


Figure 3-1. 8086 Minimum Mode System

3.2 System Nucleus

All MCS-86 systems have in common the 8284 system clock generator (See Figure 3-3.) This one-chip module provides the following functions:

- System clock output with MOS characteristics, at 1/3 of the frequency of either a quartz crystal or an external clock input (EFI) and a 33% duty cycle.
- One peripheral clock output with TTL characteristics, at 1/2 the frequency of the system clock.
- An oscillator output (at the crystal frequency) and a synchronizing input that allows multiple 8284's.
- RESET and READY signal conditioning for these 8086 inputs.

A strapping option, $F/\overline{C}$, allows the clock frequency to be established either by a quartz crystal or by an external frequency input signal. When $F/\overline{C}$ is low, an internal oscillator is enabled and its frequency is determined by a crystal connected across the two X pins. When $F/\overline{C}$ is high, EFI is enabled to the clock counters and the oscillator is gated off. A divide-by-three counter generates the frequency and duty cycle of CLK. PCLK is a TTL-level output, runs at half the frequency of CLK, and has a 50% duty cycle. The CSYNC input to the 8284 is used, in conjunction with its external frequency input, to cause two or more 8284s to start counting in phase. Its READY circuitry synchronizes to the clock the asynchronous events that generate READY.

The clock module accepts an active-low $\overline{RES}$ input, processes it through a Schmitt trigger so that an RC timer circuit can be used in conjunction with it to generate a power-on reset pulse of 50μs or four clock cycles, whichever is greater. Its RESET output is active high, as required by the 8086.

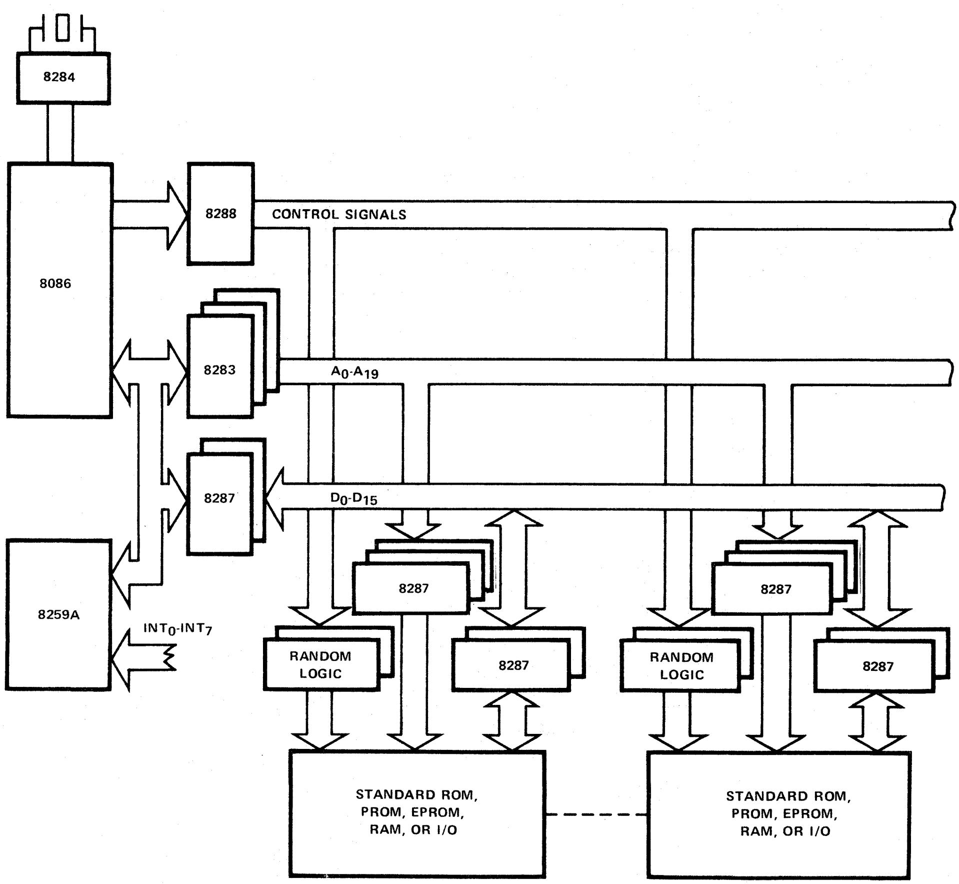


Figure 3.2 8086 Maximum Mode System

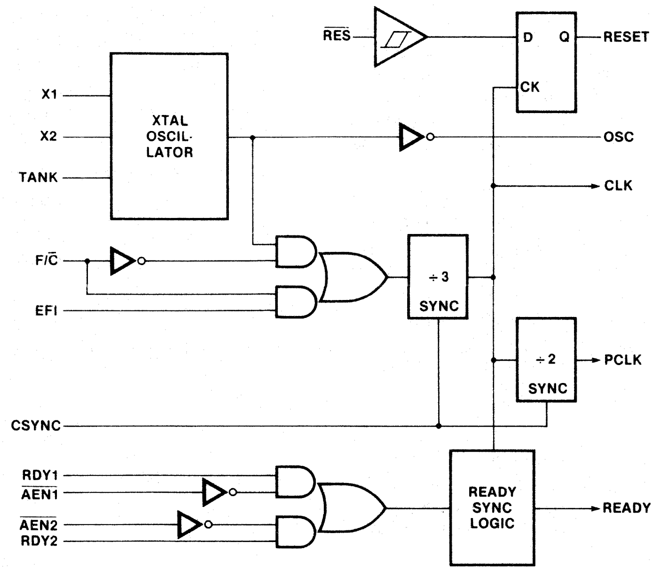


Figure 3-3. 8284 Bipolar Clock Generator

The only other essential system function common to all configurations is address latching. This is accomplished by means of 8282 or 8283 (inverting) 8-line latches (Figure 3-4). These are necessary in order to demultiplex the address/data bus so that there is a stable address for memory and I/O. The address is latched by the falling edge of the ALE signal during TI of the 8086 timing cycle. In order to speed the address transfer the latches are transparent during ALE. These bipolar devices provide a 32MA, 300pF drive capability for use on large system buses.

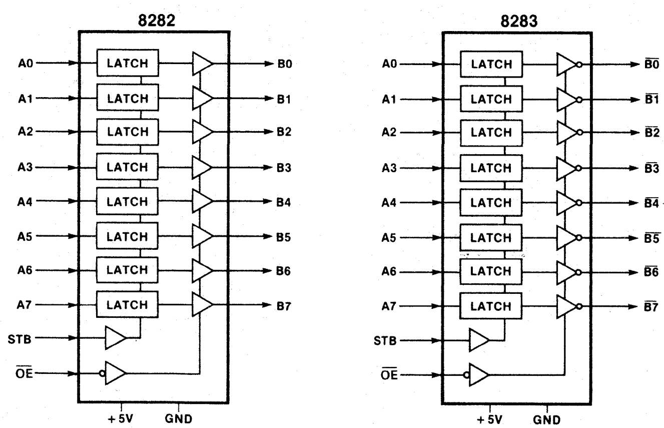


Figure 3-4. 8282 and 8283 Latches

3.3 System Nucleus Interface

3.3.1 Clock Generator and Latches

The most simplistic system configuration is obtained by strapping the MN/$\overline{MX}$ pin to +5V. With this minimum configuration the clock and latches are interconnected as shown in Figure 3.5. Using the two 8282 latches only 32K of address space is available since $\overline{BHE}$ must be demultiplexed and latched for system use. By adding another latch, for the upper five address bits the full 1M byte of memory is available.

A closer examination of the figure shows that the control signals ($\overline{RD}$, $\overline{WR}$, M/$\overline{IO}$) are generated by the 8086 and ALE is used to strobe the address into the latches. System timing is generated by the 8284 and a crystal (F/$\overline{C}$ is strapped to ground) with power on reset being generated by the RC network. Other functions, optionally used, are tied to the inactive state, however, the user is free to use these functions in order to optimize a system design.

3.3.2 Transceivers

If the minimum system requires buffering, or a need for additional drive, the 8286 bipolar transceivers should be used. The 8086 $\overline{DEN}$ signal is used to enable the data (8286 $\overline{OE}$) onto the system or local bus, as determined by the 8286 T input. When DT/$\overline{R}$ is high, data is sent to the system bus from the local bus. If DT/$\overline{R}$ is low, data is sent from the system bus to the local bus.

3.4 I/O Interface

The 8086 microprocessor interfaces to both memory and I/O by means of read and write machine cycles, the timing of which are essentially identical. During each machine cycle, the 8086 issues an address and a control signal, then either sends data out on the bus or reads it in from the bus. When the 8086 performs a read cycle, the source of the data it reads could be a ROM, a RAM, an I/O device, or nothing; the cpu has no mechanism for verifying the presence of valid data.

The only distinction between data read in from memory or I/O and an instruction opcode is the interpretation of the byte or word by the cpu. If the cpu expects an opcode, it will treat the input as an opcode. If an I/O port is expected, it will treat what is found on the bus as input data. The same is true for a write cycle. The 8086 issues an address, puts data on the bus, then issues a control signal. Unless it is requested to WAIT, by the use of the READY line, the cpu will complete the cycle and proceed to the next. Regardless of whether there is a device on the bus that will accept the data, the cpu executes one instruction at a time in sequence until requested to do otherwise. The program totally controls the sequence and nature of all machine cycles unless a hardware interrupt, hold, or wait occurs.

3.4.1 Memory-Mapped I/O

Although the 8086 has separate instructions to communicate with I/O devices, some users connect I/O devices to the memory control lines. This is referred to as memory-mapped I/O. As long as the I/O device responds like a memory device, the processor does not recognize the device as different. The advantage of using memory-mapped I/O is that you can use the large group of instructions that operate in the memory address space instead of the four (IN, INW, OUT, and OUTW) that transfer a byte or word between the accumulator and I/O port space. You can also perform arithmetic and logic operations on port data as well as move it between any of the internal registers and the I/O ports, between memory and ports, and from one port to another. Consider the meanings of the instructions in the list that follows:

MOV	reg, [BX]	Input 16 bit port to register (indirect address)
MOV	[BX],reg	Output register to 16 bit port (indirect address)
MOV	[BX],data	Output immediate data to 16 bit port (indirect address)
MOV	AX,addr	Input 16 bit port to accumulator (direct address)
MOV	addr,AX	Output accumulator to 16 bit port (direct address)
ADD	AX,[BX]	Add 16 bit port to accumulator (indirect address)
AND	AX,[BX]	And 16 bit port to accumulator (indirect address)

The list is not exclusive, since the 8086 instruction set has extensive memory addressing modes for many operations making memory mapped I/O very attractive in system design.

Although using memory instructions may increase the flexibility of a system, there are some drawbacks. Since memory-mapped I/O devices are addressed as memory, there are fewer addresses available for memory use. In systems with smaller addressing capabilities, this may be a serious problem. With MCS-86 systems, where there is a total of 1 megabyte of memory address space, it is less likely to impose limits on your design.

A further disadvantage of using memory mapped I/O is that it is slower than I/O mapped I/O when performing normal input, output functions. The IN or OUT instructions only require 10 clock cycles and as little as one byte of code when a variable port is specified. The speed and byte requirements when using the MOV instruction vary according to the addressing mode used as shown below.

TABLE 3.

Operation	Byte Count	Clock Cycles
16 bit immediate data to port (indirect)	4	10+EA
Accumulator to port (immediate)	3	15
Register to port (indirect)	2	9+EA
Port (immediate) to accumulator	3	14
Port (indirect) to register	2	8+EA

Where EA is the time required to calculate the effective address of the memory operand (in this case the port address).

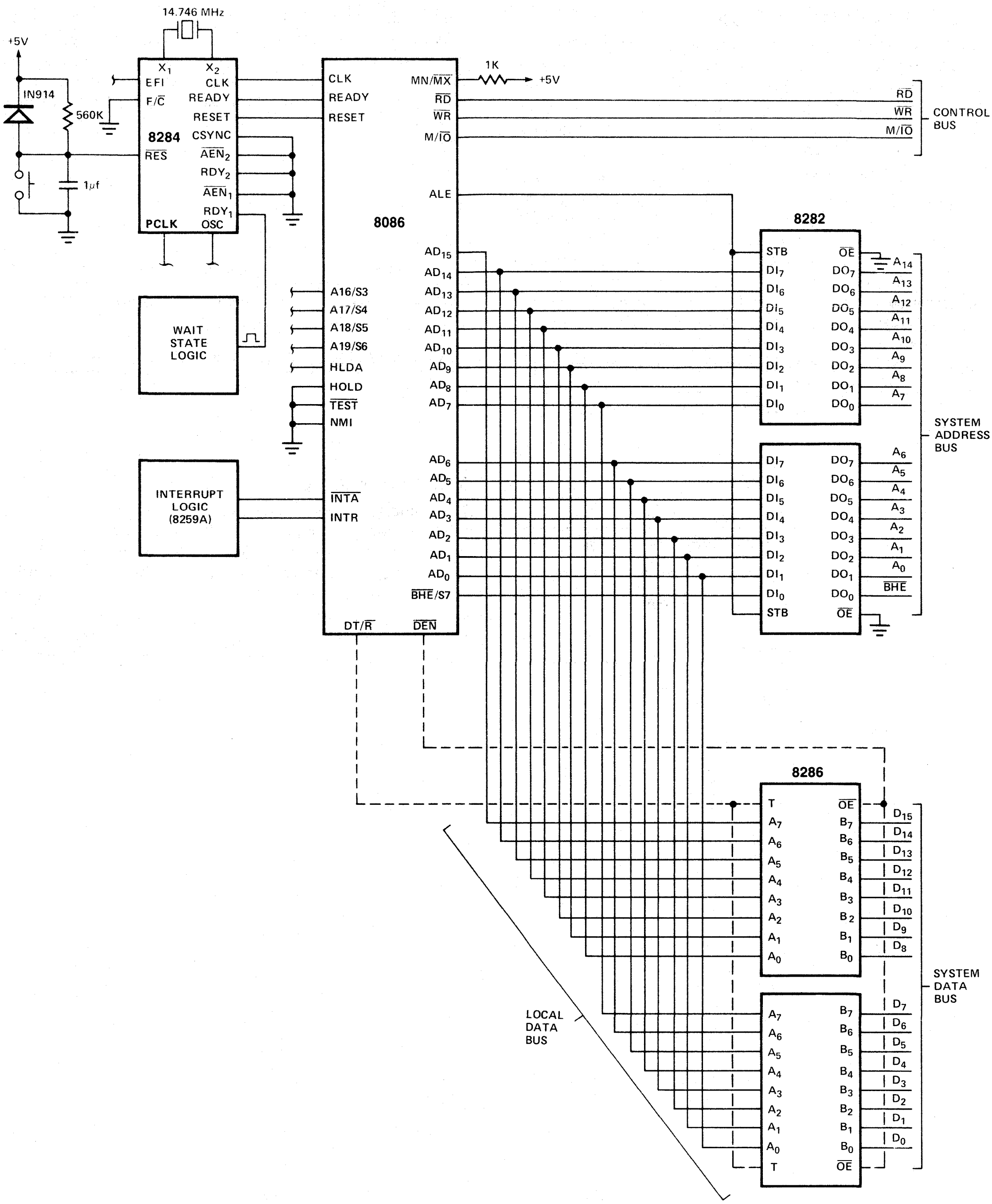


Figure 3-5. 8086 Minimum System

3.4.2 Interfacing Memory Mapped I/O

The 8086 memory space has dedicated address locations which prohibit the use of one specific bit for linear selection of a memory mapped I/O device. (The Vector Interrrupt code is in low memory and the Bootstrap code is in high memory.) This does not prevent a user from using a block of memory for I/O, even in a minimum system configuration.

Figure 3.6 shows how the 8255 can be used as memory mapped I/O ports. Using the 8205 to decode for the device chip selects, up to 24 programmable 16-bit ports, or 48 8-bit ports, can be selected. As the memory map in Figure 3.6b shows, the address range for these ports is 00400H to 00043FH. Since this is a two address latch minimum system, the upper five most significant bits are not used. Therefore A10 and A14 are used to select I/O. Whenever A14 is low and A10 is high, the 8205 will decode address bits A3, A4, and A5. The additional enable input is used to select high, low or both 8-bit data using $\overline{BHE}$ and A0 according to Table 3-2. This decoding scheme is reflected in the way a typical port is selected.

TABLE 3-2

$\overline{BHE}$	AO	FUNCTION
0	0	16 bit word from/to addressed devices
0	1	Upper 8 bits from/to odd addressed device
1	0	Lower 8 bits from/to even addressed device
1	1	No device selection

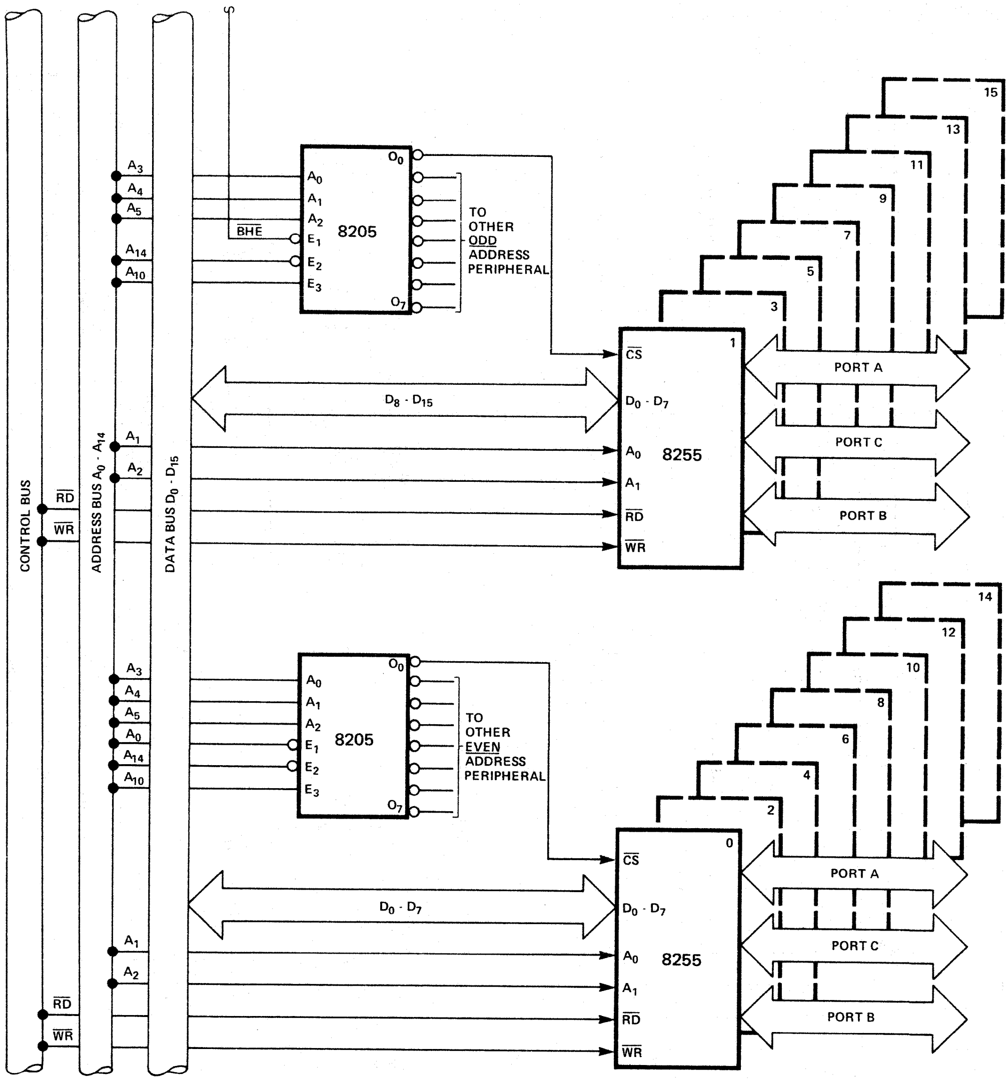
Figure 3.6b shows that the ports of device 0 are all even addresses. If 16-bit data needs to be transferred any memory address may be used, but odd locations will take two memory cycles and could inadvertently access the control port, as Table 3-3 shows. In order to avoid these conflicts, all ports should be addressed as *even* locations when transferring word data.

TABLE 3-3

INSTRUCTION	DATA TRANSFERRED
MOV AX, 402	16 bits from ports B_0 and B_1
MOV AX, 403	16 bits from ports B_1 and C_0
MOV AX, 405	16 bits from ports C_1 and Control 0

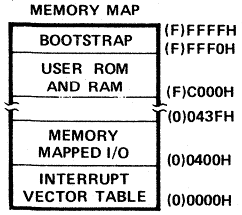
3.4.3 Interfacing I/O Mapped I/O

The primary difference between the design of I/O mapped I/O and memory mapped I/O is the necessity to distinguish between a memory or I/O bus cycle. To do this the $M/\overline{IO}$ signal is used to enable the 8205's for chip select decode. The third enable can be used to place the I/O into a specific location. Figure 3.7 shows this design for a minimum system. Even with the ports in I/O space, the same addressing considerations apply for this type of I/O as in memory mapped I/O.



a.

MEMORY MAP



TYPICAL PORT SELECTION

PORT	DEVICE	ADDRESS (8-bit data)	
A	0	00400H	EVEN ADDRESS
B	0	00402H	
C	0	00404H	
CONTROL	0	00406H	
A	1	00401H	ODD ADDRESS
B	1	00403H	
C	1	00405H	
CONTROL	1	00407H	

b.

Figure 3-6. Minimum System Memory Mapped I/O

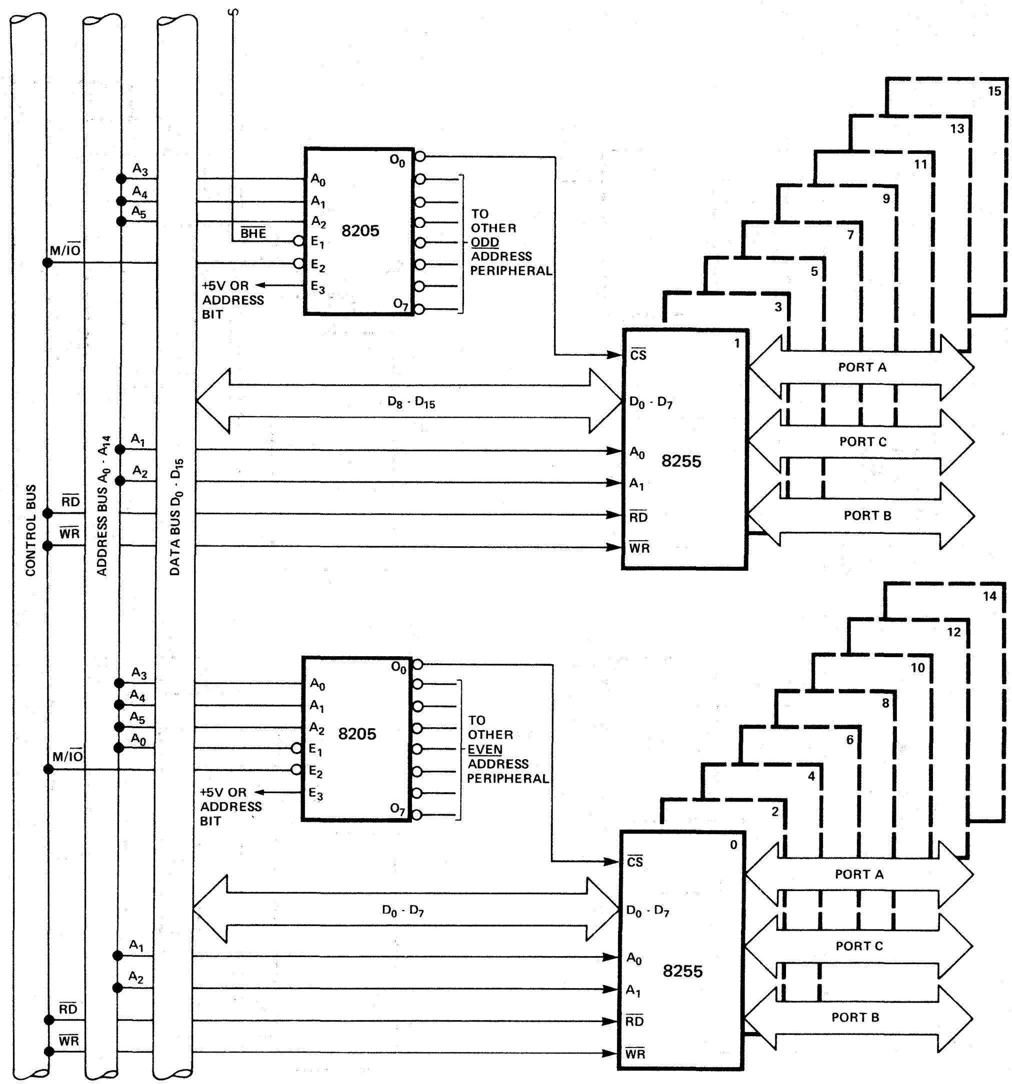


Figure 3-7. Minimum System I/O Mapped I/O

CHAPTER 4

The Instruction Set

CHAPTER 4
THE INSTRUCTION SET

4.1 What The Instruction Set Is

An instruction set is a set of codes that direct a computer to perform its operations. The ease of understanding an instruction set does not depend upon the structure of the machine codes that the computer recognizes so much as it depends upon the structure of the symbolic language that is associated with the computer. The description of the instruction set in this chapter is therefore based upon the syntax of the MCS-86™ assembly language, and each instruction is described in terms of its mnemonic, followed by the details of its binary codes and their meanings.

4.2 Symbols, Abbreviations, and Mnemonics

The terminology used in describing the MCS-86™ computer systems and their instruction set is in many respects similar with that which describes the MCS-80™, but there are some differences. In instances where an MCS-80 term is equivalent to that of the MCS-86™, it is included in the list that follows, with an indication of any significant differences.

Table 4-1. SYMBOLS

MCS-86 Symbol	MCS-80 Symbol	Meaning
AX	None	Accumulator (16-bit) (8080 Accumulator holds only 8-bits)
AH	None	Accumulator (high-order byte)
AL	A	Accumulator (low-order byte)
BX	HL	Register B (16-bit) (8080 register pair HL), which may be split and addressed as two 8-bit registers.
BH	H	High-order byte of register B
BL	L	Low-order byte of register B
CX	BC	Register C (16-bit) (8080 register pair BC), which may be split and addressed as two 8-bit registers
CH	B	High-order byte of register C
CL	C	Low-order byte of register C
DX	DE	Register D (16-bit) (8080 register DE) which may be split and addressed as two 8-bit registers
DH	D	High-order byte of register D
DL	E	Low-order byte of register D
SP	SP	Stack pointer (16-bit)
BP	None	Base pointer
IP	PC	Instruction pointer (8080 Program Counter)
Flags	Flags	16-bit register space, in which nine flags reside. (Not directly equivalent to 8080 PSW, which contains five flags and the contents of the accumulator).
DI	None	Data index register
SI	None	Stack index register
CS	None	Code segment register
DS	None	Data segment register
ES	None	Extra segment register
SS	None	Stack segment register
REG8		The name of an 8-bit CPU register location
REG16		The name of a 16-bit CPU register location
reg		In the description of an instruction a field which defines REG8 or REG16
EA		Effective address (16-bit)
r/m		A register name or memory address in an instruction, this 3-bit field defines EA, in conjunction with the mode and w fields.
mode		In an instruction, this 2-bit field defines addressing mode
w		A 1-bit field in an instruction, identifying byte instructions (w=0), and word instructions (w=1)
d		A 1-bit field identifying direction, i.e., which location is source and which is destination, in an instruction.
(...)		Parentheses enclosing a register name or the contents of register or memory location...
(BX)		Represents the contents of register BX, which could be used as the address where an 8-bit operand might be located.
((BX))		Means this 8-bit operand, the contents of the memory location pointed at by the contents of register BX.

MCS-86 Symbol	MCS-80 Symbol	Meaning
(BX) + 1, (BX)		Is the address of a 16-bit operand whose low-order 8-bits reside in the memory location pointed at by the contents of register BX and whose high-order 8-bits reside in the next sequential memory location, BX + 1.
((BX) + 1, (BX))		Is the 16-bit operand that resides there.
.		Field, e.g., (AL). low-nibble describes the low-nibble (least significant 4-bits) of the contents of register AL.
:		Concatenation, e.g., ((DX) + 1: (DX)) is a 16-bit word which is the concatenation of two 8-bit bytes, the low-order byte in the memory location pointed at by DX and the high-order byte in the next sequential memory location
addr		Address (16-bit) of a byte in memory
addr-low		Least significant byte of an address
addr-high		Most significant byte of an address
addr + 1: addr		Addresses of two consecutive bytes in memory, beginning at addr
data		Immediate operand (8-bit if w=0; 16-bit if w=1)
data-low		Least significant byte of 16-bit data word
data high		Most significant byte of 16-bit data word
disp		Displacement
disp-low		Least significant byte of 16-bit displacement
disp-high		Most significant byte of 16-bit displacement
<==		Assignment
+		Addition
—		Subtraction
*		Multiplication
/		Division
%		Modulo
&		And
|		Inclusive or
||		Exclusive or

4.3 Instruction and Data Formats

The formats described briefly here reflect the assembly language processed by the Intel-supplied assembler, ASM-86, used with the Intellec® development systems. (See Chapter 6.)

Assembly language instructions are written one per line. If a semicolon occurs other than in a string, then the remainder of that line is taken as a comment.

An instruction is made up of a series of tokens. Each token may be one of three types:

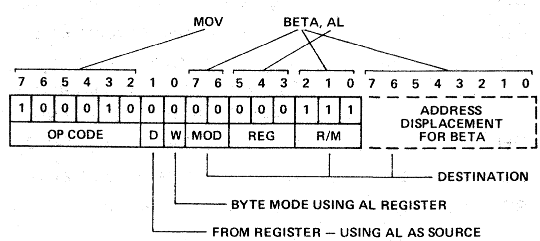
Name
Constant
Delimiter

If two consecutive tokens together might be interpreted as some other token, they must be separated by a space; if not, spaces have no meaning and may be omitted. However, extra spaces may be inserted if desired; the computer ignores them. Comments may be made any number of lines long, but a semicolon must start each line of a comment. The assembler ignores comments, blank lines may be inserted, too; the assembler ignores them. It does not distinguish between capitals and lower-case letters.

An exception to the above rules is the character string. The assembler recognizes all of the characters, spaces, and blanks that are contained within the string.

4.4 Instruction Set Encyclopedia

Five different lists of the 8086 instructions are found in this manual. The encyclopedia, located here, contains a complete description of every instruction and its function, arranged in the same six functional groups as in Section 2.2.4, where they are introduced in briefer form. Table 2-1 is a quick-reference summary of all instructions and their codes. It is repeated in the 8086 data sheets in Chapter 5. Table 4-3 is a translation table of 8080 instructions to 8086 instructions. Table 4-4 is an alphabetical index to this encyclopedia. In these lists, all instructions are referenced to the assembly-language mnemonics (Section 4.3). Although there is not a unique mnemonic for each instruction code, there is enough information in the instruction, source, and destination mnemonics for the assembler to identify the correct code. This means you don't have to keep in mind which of the 19 different MOV codes is needed when programming in assembly language. You write, for example,



Assembler takes care of:

D
W
MOD
Displacement

The assembler chooses the correct codes that perform your intended operation. This encyclopedia describes how those codes function, so that when you have written and are testing your program you can determine what the processor is doing by monitoring the bus.

4.4.1 Addressing Modes

The 8086 instruction set provides several different ways to address operands. Most two-operand instructions allow either memory or a register to serve as one operand, and either a register or a constant within the instruction to serve as the other operand. Operands in memory may be addressed directly with a 16-bit offset address, or indirectly with base (BX or BP) and/or index (SI or DI) registers added to an optional 8- or 16-bit displacement constant. The result of a two-operand operation may be directed to either of the source operands, with the exception, of course, of in-line immediate constants. Single-operand operations are applicable uniformly to any operand except immediate constants. Virtually all 8086 operations may specify either 8- or 16-bit operands.

Memory Operands. Operands residing in memory may be addressed in four ways:

- Direct 16-bit offset address
- Indirect through a base register, optionally with an 8- or 16-bit displacement
- Indirect through an index register, optionally with an 8- or 16-bit displacement
- Indirect through the sum of a base register and an index register, optionally with an 8- or 16-bit displacement

General register BX and pointer register BP may serve as base registers. When BX is the base the operand by default resides in the current data segment and the DS register is used to compute the physical address of the operand. When BP is the base the operand by default resides in the current stack segment and the SS segment register is used to compute the physical address of the operand. When both base and index registers are used the operand by default resides in the segment determined by the base register. When an index register alone is used, the operand by default resides in the current data segment.

The location of an operand in an 8086 register or in memory is specified by up to three fields in each instruction. These fields are the mode field (mod) the register field (reg), and the register/memory field (r/m). When used, they occupy the second byte of the instruction sequence. The mode field occupies the two most significant bits of the byte, and specifies how the r/m field is used in locating the operand, i.e., the register named in the r/m field either can be the location of the operand, or can point to the location of the operand in memory. The reg field occupies the next three bits following the mode field, and specifies either an 8-bit register or a 16-bit register to be the location of an operand.

Description: The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

if mod = 00 the DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16 bits, disp-high is absent
if mod = 10 then DISP = disp-high:disp-low
if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP

*except if mod = 00 and r/m = 110 then
EA = disp-high: disp-low

Instructions referencing 16-bit objects interpret EA as addressing the low-order byte; the word is addressed by EA+1,EA.

Encoding:

mod r/m	disp-low	disp-high

Segment Override Prefixes. When the effective address computation for a memory operand involves the BP register the SS segment register is used by default to compute the physical address. The physical address of most other memory operands is by default computed using the DS segment register (exceptions are noted below). These default segment register selections may be overridden by preceding the referencing instruction with a segment override prefix.

Description: The segment register selected by the reg field (see Section 2.3.1) is used to compute the physical address for the instruction this prefix precedes. This prefix may be combined with the LOCK and/or REP prefixes, although the latter has certain problems as discussed in Section 4.4.6.

Encoding:

0 0 1 reg 1 1 0

The physical addresses of all operands addressed by the SP register are computed using the SS segment register, which may not be overridden. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment register, which may not be overridden.

Register Operands: The four 16-bit general registers and the four 16-bit pointer and index registers may serve interchangeably as operands in nearly all 16-bit operations. Three exceptions to note are multiply, divide, and some string operations, which use the AX register implicitly. The eight 8-bit registers of the HL group may serve interchangeably in 8-bit operations. Multiply, divide, and some string operations use AL implicitly.

Description: Register operands may be indicated by a distinguished field, in which case REG will represent the selected register, or by an encoded field, in which case EA will represent the register selected by the r/m field. Instructions without a "w" bit always refer to 16-bit registers (if they refer to any register at all); those with a "w" bit refer to either 8- or 16-bit registers according to "w".

Encoding:

Distinguished Field:

| reg | or | reg |

for mod = 11 EA = r/m (a register):

| 11 reg |

Segment Register:

| reg |

REG = reg

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

Where X is undefined.

Immediate Operands. All two-operand operations except multiply, divide, and the string operations allow one source operand to appear within the instruction as immediate data. Sixteen-bit immediate operands having a high-order byte which is the sign extension of the low-order byte may be abbreviated to eight bits.

Three points about immediate operands should be made:

- Immediate operands always follow addressing mode displacement constants (when present) in the instruction.
- The 8-bit immediate operands of instructions with s:w = 11 are sign-extended to 16-bit values.
- The low-order byte of 16-bit immediate operands always precedes the high-order byte.

4.4.2 Instruction Set Timing

Opposite the expression for the operation contained in each listing in Sections 4.4.3 through 4.3.9 is a number that represents the number of clock cycles required for the execution of that instruction. If you are using a 5-MHz clock, one cycle is 200 nanoseconds; if you use an 8-MHz clock, it is 125 nanoseconds. The times stated are fixed in the case of register and immediate operands. In the case of operands in memory, you must add the time necessary to calculate an address to the execution time of the instruction. This is indicated in the listed times by the term, + EA. The amount of time needed for calculation varies, depending upon addressing mode (Section 4.4) and location. The list below shows calculation times for 8-bit operands or for 16-bit operands residing at even memory addresses. Add 4 clock cycles for 16-bit operands residing at odd memory addresses. Add 2 clock cycles for segment override.

- Direct 16-bit offset address — 6
- Indirect through base or index register — 5
- Indirect through base or index register with displacement constant — 9
- Indirect through sum of index-register plus base register — 7 or 8
- Indirect through sum of base register plus index register with displacement constant — 11 or 12

4.4.3 Data Transfer

Four classes of data transfer operations may be distinguished: general purpose, accumulator specific, address-object transfers, and flag transfers. None of the transfer instructions affect any flag settings except SAHF and POPF.

Four general purpose data transfer operations are provided. These may generally be applied to most operands, though there are specific exceptions. The general purpose transfers (except XCHG) are the only operations which allow a segment register as an operand:

MOV Move
PUSH Push
POP Pop
XCHG Exchange

Mnemonic: **MOV**

Description: MOV performs a byte or word transfer from the source operand to the destination operand.

Encoding:

Memory or Register Operand to/from Register Operand:

1 0 0 0 1 0 d w	mod reg r/m

if d = 1 then SRC = EA, DEST = REG
else SRC = REG, DEST = EA

Timing (clocks): register to register 2
memory to register 8+EA
register to memory 9+EA

Immediate Operand to Memory or Register Operand:

1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1

SRC = data, DEST = EA
Timing: 10+EA clocks

Immediate Operand to Register:

1 0 1 1 w reg	data	data if w=1

SRC = data, DEST = REG
Timing: 4 clocks

Memory Operand to Accumulator:

1 0 1 0 0 0 0 w	addr-low	addr-high

if w = 0 then SRC = addr, DEST = AL
else SRC = addr+1:addr, DEST = AX
Timing: 10 clocks

Accumulator to Memory Operand:

1 0 1 0 0 0 1 w	addr-low	addr-high

if w = 0 then SRC = AL, DEST = addr
else SRC = AX, DEST = addr+1:addr
Timing: 10 clocks

Memory or Register Operand to Segment Register:

1 0 0 0 1 1 1 0	mod 0 reg r/m

if reg ≠ 01 then SRC = EA, DEST = REG
else undefined operation
Timing (clocks): register to register 2
memory to register 8+EA

Segment Register to Memory or Register Operand:

1 0 0 0 1 1 0 0	mod 0 reg r/m

SRC = REG,DEST = EA

Timing (clocks): memory to register 9+EA
register to register 2

Operation:

(DEST) <== (SRC)

Flags Affected:

None

Mnemonic: **PUSH**

Description: PUSH decrements the SP register by two and then transfers a word from the source operand to the stack element currently addressed by SP.

Encoding:

Memory or Register Operand:

1 1 1 1 1 1 1 1	mod 1 1 0 r/m

SRC = EA
Timing (clocks): memory 16+EA
register 10

Register Operand:

0 1 0 1 0 reg

SRC = REG
Timing: 10 clocks

Segment Register:

0 0 0 reg 1 1 0

SRC = REG
Timing: 10 clocks

Operation:

(SP) <== (SP) - 2
((SP)+1:(SP)) <== (SRC)

Flags Affected:

None

Mnemonic: **POP**

Description: POP transfers a word operand from the stack element addressed by the SP register to the destination operand and then increments SP by 2.

Encoding:

Memory or Register Operand:

1 0 0 0 1 1 1 1	mod 0 0 0 r/m

DEST = EA

Timing (clocks): memory 17+EA
register 8

Register Operand:

0 1 0 1 1 reg

DEST = REG

Timing: 8 clocks

Segment Register:

0 0 0 reg 1 1 1

if reg ≠ 01 then DEST = REG
else undefined operation
Timing: 8 clocks

Operation:

(DEST) <== ((SP)+1:(SP))
(SP) <== (SP)+2

Flags Affected:

None

Mnemonic: **XCHG**

Description: XCHG exchanges the byte or word source operand with the destination operand. The segment registers may not be operands of XCHG.

Encoding:

Memory or Register Operand with Register Operand:

1 0 0 0 0 1 1 w	mod reg r/m

SRC = EA, DEST = REG

Timing (clocks): memory with register 17+EA
register with register 4

Register Operand with Accumulator:

1 0 0 1 0 reg

SRC = REG, DEST = AX

Timing: 3 clocks

Operation:

(Temp) <== (DEST)
(DEST) <== (SRC)
(SRC) <== (Temp)

Flags Affected:

None

Three Accumulator transfer operations are provided:

Accumulator Specific Transfers

IN	Input Byte	Single Operation
INW	Input Word	
OUT	Output Byte	Single Operation
OUTW	Output Word	
XLAT	Translate	

Mnemonic: **IN and INW**

Description: IN (or INW) transfers a byte (or word) from an input port to the AL register (or AX register for INW). The port is specified either with an inline data byte, allowing fixed access to ports 0 through 255, or with a port number in the DX register, allowing variable access to 64K input ports.

Encoding:

Fixed Port:

1 1 1 0 0 1 0 w	port

if w = 0 then SRC = port, DEST = AL
else SRC = port+1:port, DEST = AX
Timing: 10 clocks

Variable Port:

1 1 1 0 1 1 0 w

if w = 0 then SRC = (DX), DEST = AL
else SRC = (DX)+1:(DX), DEST = AX
Timing: 8 clocks

Operation:

(DEST) <== (SRC)

Flags Affected:

None

Mnemonic: **OUT and OUTW**

Description: OUT (or OUTW) transfers a byte (or word) from the AL register (or AX register for OUTW) to an output port. The port is specified either with an inline data byte, allowing fixed access to ports 0 through 255, or with a port number in the DX register, allowing variable access to 64K output ports.

Encoding:

Fixed Port:

1 1 1 0 0 1 1 w	port

if w = 0 then SRC = AL, DEST = port
else SRC = AX, DEST = port+1:port
Timing: 10 clocks

Variable Port:

1 1 1 0 1 1 1 w

if w = 0 then SRC = AL, DEST = (DX)
else SRC = AX, DEST = (DX)+1:(DX)
Timing: 8 clocks

Operation:

(DEST) <== (SRC)

Flags Affected:

None

Mnemonic: **XLAT**

Description: XLAT performs a table lookup byte translation. The AL register is used as an index into a 256-byte table addressed by the BX register. The byte operand so addressed is transferred to AL.

Encoding:

1 1 0 1 0 1 1 1

Timing: 11 clocks

Operation:

(AL) <== ((BX) + (AL))

Flags Affected:

None

Three Address-Object transfer operations are provided:

Address Object Transfers
- LEA Load Effective Address
- LDS Load Pointer into DS
- LES Load Pointer into ES

Mnemonic: **LEA**

Description: LEA (Load Effective Address) transfers the offset address of the source operand to the destination operand. The source operand must be a memory operand and the destination operand can be any 16-bit general, pointer, or index register.

Encoding:

1 0 0 0 1 1 0 1	mod reg r/m

if mod = 11 then undefined operation
Timing: 2+EA clocks

Operation:

(REG) <== EA

Flags Affected:

None

Mnemonic: **LDS**

Description: LDS (Load Pointer into DS) transfers a "pointer-object" (i.e., a 32-bit object containing an offset address and a segment address) from the source operand (which must be a memory operand) to a pair of destination registers. The segment address is transferred to the DS segment register. The offset address may be transferred to any 16-bit general, pointer, or index register.

Encoding:

1 1 0 0 0 1 0 1	mod reg r/m

if mod = 11 then undefined operation
Timing: 16+EA clocks

Operation:

(REG) <== (EA)
(DS) <== (EA + 2)

Flags Affected:

None

Mnemonic: **LES**

Description: LES (Load Pointer into ES) transfers a "pointer object" (i.e., a 32-bit object containing an offset address and a segment address) from the source operand (which must be a memory operand) to a pair of destination registers. The segment address is transferred to the ES segment register. The offset address may be transferred to a 16-bit general, pointer, or index register.

Encoding:

1 1 0 0 0 1 0 0	mod reg r/m

if mod = 11 then undefined operation
Timing: 16+EA clocks

Operation:

(REG) <== (EA)
(ES) <== (EA + 2)

Flags Affected:

None

Four Flag Register transfer operations are provided:

LAHF	Load AH with Flags
SAHF	Store AH into Flags
PUSHF	Push Flags
POPF	Pop Flags

Mnemonic: **LAHF**

Description: LAHF (Load AH with Flags) transfers the flag registers SF, ZF, AF, PF, and CF (which, when 8080 code is translated into 8086 code, are the 8080 flags) into specific bits of the AH register. The bits indicated "X" are unspecified.

Encoding:

1 0 0 1 1 1 1 1

Timing: 4 clocks

Operation:

(AH) <== (SF):(ZF):X:(AF):X:(PF):X:(CF)

Flags Affected:

None

Mnemonic: **SAHF**

Description: SAHF transfers specific bits of the AH register to the flag registers SF, ZF, AF, PF, and CF. The bits of AH indicated by "X" in the operation below are ignored.

Encoding:

1 0 0 1 1 1 1 0

Timing: 4 clocks

Operation:

(SF):(ZF):X:(AF):X:(PF):X:(CF) <== (AH)

Flags Affected:

AF, CF, PF, SF, ZF

Mnemonic: **PUSHF**

Description: PUSHF decrements the SP register by 2 and transfers all of the flag registers into specific bits of the word operand (stack element) addressed by SP.

Encoding:

1 0 0 1 1 1 0 0

Timing: 10 clocks

Operation:

(SP) <== (SP) - 2
((SP)+1:(SP)) <== Flags

Flags Affected:

None

Mnemonic: **POPF**

Description: POPF (pop flags) transfers specific bits of the stack element addressed by the SP register to the flag registers and then increments SP by two.

Encoding:

1 0 0 1 1 1 0 1

Timing: 8 clocks

Operation:

Flags <== ((SP)+1:(SP))
(SP) <== (SP) + 2

Flags Affected:

All

4.4.4 Arithmetic

The 8086 provides the four basic mathematical operations in a number of different varieties. Both 8- and 16-bit operations and both signed and unsigned arithmetic are provided. Standard two's complement representation of signed values is used. The addition and subtraction operations serve as both signed and unsigned operations. In these cases the flag settings allow the distinction between signed and unsigned operations to be made (see Conditional Transfer). Correction operations are provided to allow arithmetic to be performed directly on unpacked decimal digits or on packed decimal representations.

Six flag bits are set or cleared by most arithmetic operations to reflect certain properties of the result of the operation. They generally follow these rules:

- CF is set if the operation resulted in a carry out of (from addition) or a borrow into (from subtraction) the high-order bit of the result; otherwise CF is cleared.
- AF is set if the operation resulted in a carry out of (from addition) or a borrow into (from subtraction) the low-order four bits of the result; otherwise AF is cleared.
- ZF is set if the result of the operation is zero; otherwise ZF is cleared.
- SF is set if the high-order bit of the result of the operation is set; otherwise SF is cleared.
- PF is set if the modulo 2 sum of the low-order eight bits of the result of the operation is 0 (even parity); otherwise PF is cleared (odd parity).
- OF is set if the operation resulted in a carry into the high-order bit of the result but not a carry out of the high-order bit, or vice versa; otherwise OF is cleared.

Five addition operations are provided:

ADD Add
ADC Add with Carry
INC Increment
AAA Unpacked BCD (ASCII) Adjust for Addition
DAA Decimal Adjust for Addition

Mnemonic: **ADD**

Description: ADD performs an addition of the two source operands and returns the result to one of the operands.

Encoding:

Memory or Register Operand with Register Operand:

0 0 0 0 0 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA

Timing (clocks):		
	register to register	3
	memory to register	9+EA
	register to memory	16+EA

Immediate Operand to Memory or Register Operand:

1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01

LSRC = EA, RSRC = data, DEST = EA

Timing (clocks):		
	immediate to memory	17+EA
	immediate to register	4

Immediate Operand to Accumulator:

0 0 0 0 0 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing: 4 clocks

Operation:

(DEST) <== (LSRC) + (RSRC)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **ADC**

Description: ADC (add with carry) performs an addition of the two source operands, adds one if the CF flag is set, and returns the result to one of the operands.

Encoding:

Memory or Register Operand with Register Operand:

0 0 0 1 0 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA

Timing (clocks):		
	register to register	3
	memory to register	9+EA
	register to memory	16+EA

Immediate Operand to Memory or Register Operand:

1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01

LSRC = EA, RSRC = data, DEST = EA
Timing (clocks): immediate to memory 17+EA
immediate to register 4

Immediate Operand to Accumulator:

0 0 0 1 0 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing: 4 clocks

Operation:

if (CF) = 1 then (DEST) <== (LSRC) + (RSRC) + 1
else (DEST) <== (LSRC) + (RSRC)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **INC**

Description: INC (increment) performs an addition of the source operand and one, and returns the result to the operand.

Encoding:

Memory or Register Operand:

1 1 1 1 1 1 1 w	mod 0 0 0 r/m

DEST = EA
Timing (clocks): register 2
memory 15+EA

Register Operand:

0 1 0 0 0 reg

DEST = REG
Timing: 2 clocks

Operation:

(DEST) <== (DEST) + 1

Flags Affected:

AF, OF, PF, SF, ZF

Mnemonic: **AAA**

Description: AAA (Unpacked BCD (ASCII) adjust for addition) performs a correction of the result in AL of adding two unpacked decimal operands, yielding an unpacked decimal sum.

Encoding:

0 0 1 1 0 1 1 1

Timing: 4 clocks

Operation:

if ((AL) & OFH) > 9 or (AF) = 1 then
(AL) <== (AL) + 6
(AH) <== (AH) + 1
(AF) <== 1
(CF) <== (AF)
(AL) <== (AL) & OFH

Flags Affected:

AF, CF.
OF, PF, SF, ZF undefined

Mnemonic: **DAA**

Description: DAA (decimal adjust for addition) performs a correction of the result in AL of adding two packed decimal operands, yielding a packed decimal sum.

Encoding:

0 0 1 0 0 1 1 1

Timing: 4 clocks

Operation:

if ((AL) & OFH) > 9 or (AF) = 1 then
(AL) <== (AL) + 6
(AF) <== 1
if (AL) > 9FH or (CF) = 1 then
(AL) <== (AL) + 6OH
(CF) <== 1

Flags Affected:

AF, CF, PF, SF, ZF
OF undefined

Seven subtractions operations are provided:

SUB Subtract
SBB Subtract with Borrow
DEC Decrement
NEG Negate
CMP Compare
AAS Unpacked BCD (ASCII) Adjust for Subtraction
DAS Decimal Adjust for Subtraction

Mnemonic: **SUB**

Description: SUB performs a subtraction of the two source operands and returns the result to one of the operands.

Encoding:

Memory or Register Operand and Register Operand:

0 0 1 0 1 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA
Timing (clocks): register from register 3
memory from register 9+EA
register from memory 16+EA

Immediate Operand from Memory or Register Operand:

1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01

LSRC = EA, RSRC = data, DEST = EA
Timing (clocks): immediate from register 4
immediate from memory 17+EA

Immediate Operand from Accumulator:

0 0 1 0 1 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing (clocks): immediate from register 4

Operation:

(DEST) <== (LSRC) - (RSRC)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **SBB**

Description: SBB (subtract with borrow) performs a subtraction of the two source operands, subtracts one if the CF flag is set, and returns the result to one of the operands.

Encoding:

Memory or Register Operand and Register Operand:

0 0 0 1 1 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA
Timing (clocks): register from register 3
memory from register 9+EA
register from memory 16+EA

Immediate Operand from Memory or Register Operand:

1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01

LSRC = EA, RSRC = data, DEST = EA
Timing (clocks): immediate from register 4
immediate from memory 17+EA

Immediate Operand from Accumulator:

0 0 0 1 1 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing: (clocks): immediate from register 4

Operation:

if (CF) = 1 then (DEST) <== (LSRC) - (RSRC) - 1
else (DEST) <== (LSRC) - (RSRC)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **DEC**

Description: DEC (decrement) performs a subtraction of one from the source operand and returns the result to the operand.

Encoding:

Memory or Register Operand:

1 1 1 1 1 1 1 w	mod 0 0 1 r/m

DEST = EA
Timing (clocks): register 2
memory 15+EA

Register Operand:

0 1 0 0 1 reg

DEST = REG
Timing: 2 clocks

Operation:

(DEST) <== (DEST) - 1

Flags Affected:

AF, OF, PF, SF, ZF

Mnemonic: **NEG**

Description: NEG (negate) performs a subtraction of the source operand from zero and returns the result to the operand.

Encoding:

1 1 1 1 0 1 1 w	mod 0 1 1 r/m

if w = 0 then SRC = FFH
else SRC = FFFFH
Timing (clocks): register 3
memory 16+EA

Operation:

(EA) <== SRC - (EA)
(EA) <== (EA) + 1 (affecting flags)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **CMP**

Description: CMP (compare) performs a subtraction of the two source operands causing the flags to be affected but does not return the result.

Encoding:

Memory or Register Operand with Register Operand:

0 0 1 1 1 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA
else LSRC = EA, RSRC = REG
Timing (clocks): register with register 3
memory with register 9+EA
register with memory 16+EA

Immediate Operand with Memory or Register Operand:

1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01

LSRC = EA, RSRC = data
Timing (clock): immediate with register 4
immediate with memory 17+EA

Immediate Operand with Accumulator:

0 0 1 1 1 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data
else LSRC = AX, RSRC = data
Timing (clocks): immediate with register 4

Operation:

(LSRC) - (RSRC)

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **AAS**

Description: AAS (Unpacked BCD (ASCII) adjust for subtraction) performs a correction of the result in the AL register of subtracting two unpacked decimal operands, yielding an unpacked decimal difference.

Encoding:

0 0 1 1 1 1 1 1

Timing: 4 clocks

Operation:

if ((AL) & OFH) > 9 or (AF) = 1 then
(AL) <== (AL) - 6
(AH) <== (AH) - 1
(AF) <== 1
(CF) <== (AF)
(AL) <== (AL) & OFH

Flags Affected:

AF, CF.
OF, PF, SF, ZF undefined

Mnemonic: **DAS**

Description: DAS (decimal adjust for subtraction) performs a correction of the result in the AL register of subtracting two packed decimal operands, yielding a packed decimal difference.

Encoding:

0 0 1 0 1 1 1 1

Timing: 4 clocks

Operation:

if ((AL) & OFH) > 9 or (AF) = 1 then
(AL) <== (AL) - 6
(AF) <== 1
if (AL) > 9FH or (CF) = 1 then
(AL) <== (AL) - 6OH
(CF) <== 1

Flags Affected:

AF, CF, PF, SF, ZF.
OF undefined

Three multiplication operations are provided:

MUL Multiply
IMUL Integer Multiply
AAM Unpacked BCD (ASCII) Adjust for Multiplication

Mnemonic: **MUL**

Description: MUL (multiply) performs an unsigned multiplication of the accumulator (AL or AX) and the source operand, returning a double-length result to the accumulator and its extension (AL and AH for 8-bit operation, or AX and DX for 16-bit operation). CF and OF are set if the top half of the result is nonzero.

Encoding:

1 1 1 1 0 1 1 w	mod 1 0 0 r/m

if w = 0 then LSRC = AL, RSRC = EA, DEST = AX, EXT = AH
else LSRC = AX, RSRC = EA, DEST = DX:AX, EXT = DX
Timing (clocks): 8-bit 71+EA
16-bit 124+EA

Operation:

(DES) <== (LSRC) * (RSRC), where * is unsigned multiply
if (EXT) = 0 then (CF) <== 0
else (CF) <== 1;
(OF) <== (CF)

Flags Affected:

CF, OF.
AF, PF, SF, ZF undefined

Mnemonic: **IMUL**

Description: IMUL (integer multiply) performs a signed multiplication of the accumulator (AL or AX) and the source operand, returning a double-length result to the accumulator and its extension (AL and AH for 8-bit operation, or AX and DX for 16-bit operation). CF and OF are set if the top half of the result is not the sign-extension of the low half of the result.

Encoding:

1 1 1 1 0 1 1 w	mod 1 0 1 r/m

if w = 0 then LSRC = AL, RSRC = EA, DEST = AX, EXT = AH, LOW = AL
else LSRC = AX, RSRC = EA, DEST = DX:AX, EXT = DX, LOW = AX
Timing (clocks): 8-bit 90+EA
16-bit 144+EA

Operation:

(DEST) <== (LSRC) * (RSRC) where * is signed multiply
if (EXT) = sign-extension of (LOW) then (CF) <== 0
else (CF) <== 1;
(OF) <== (CF)

Flags Affected:

CF, OF.
AF, PF, SF, ZF undefined

Mnemonic: **AAM**

Description: AAM (Unpacked BCD (ASCII) adjust for multiply) performs a correction of the result in AX of multiplying two unpacked decimal operands, yielding an unpacked decimal product.

Encoding:

1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0

Timing: 83 clocks

Operation:

(AH) <== (AL) / OAH
(AL) <== (AL) % OAH

Flags Affected:

PF, SF, ZF.
AF, CF, OF undefined

Three division operations are provided, as well as two sign-extension operations which support signed division.

- DIV Divide
- IDIV Integer Divide
- AAD Unpacked BCD (ASCII) Adjust for Division
- CBW Convert Byte to Word
- CWD Convert Word to Double Word

Mnemonic: **DIV**

Description: DIV (divide) performs an unsigned division of the double-length NUMR operand, contained in the accumulator and its extension (AL and AH for 8-bit operation, or AX and DX for 16-bit operation) by the DIVR operand, contained in the specified source operand. It returns the single-length quotient (QUO operand) to the accumulator (AL or AX), and returns the single-length remainder (the REM operand) to the accumulator extension (AH for 8-bit operation or DX for 16-bit operation). If the quotient is greater than MAX (as when division by zero is attempted) then QUO and REM are undefined, and a type 0 interrupt is generated. Flags are undefined in any DIV operation. Nonintegral quotients are truncated to integers.

Encoding:

1 1 1 1 0 1 1 w	mod 1 1 0 r/m

if w = 0 then NUMR = AX, DIVR = EA, QUO = AL, REM = AH, MAX = FFH
else NUMR = DX:AX, DIVR = EA, QUO = AX, REM = DX, MAX = FFFFH

Timing: (clocks): 8-bit 90+EA
16-bit 155+EA

Operation:

(temp) <== (NUMR)
if (temp) / (DIVR) > MAX then the following, in sequence
(QUO), (REM) undefined
(SP) <== (SP) - 2
((SP)+1:(SP)) <== FLAGS
(IF) <== 0
(TF) <== 0
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (CS)
(CS) <== (2) i.e., the contents of memory locations 2 and 3
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (IP)
(IP) <== (0) i.e., the contents of locations 0 and 1
else
(QUO) <== (temp) / (DIVR), where / is unsigned division
(REM) <== (temp) % (DIVR), where % is unsigned modulo

Flags Affected:

AF, CF, OF, PF, SF, ZF undefined

Mnemonic: **IDIV**

Description: IDIV (integer divide) performs a signed division of the double-length NUMR operand, contained in the accumulator and its extension (AL and AH for 8-bit operation, or AX and DX for 16-bit operation) by the DIVR operand, contained in the specified source operand. It returns the single-length quotient (QUO operand) to the accumulator (AL or AX), and returns the single-length remainder (the REM operand) to the accumulator extension (AH for 8-bit operation or DX for 16-bit operation). If the quotient is positive and greater than MAX or if the quotient is negative and less than (0 - MAX - 1), (as when division by zero is attempted) then QUO and REM are undefined, and a type 0 interrupt is generated. Flags are undefined in any divide operation. IDIV truncates nonintegral quotients and returns a remainder with the same sign as the numerator.

Encoding:

1 1 1 1 0 1 1 w	mod 1 1 1 r/m

if w = 0 then NUMR = AX, DIVR = EA, QUO = AL, REM = AH, MAX = 7FH
else NUMR = DX:AX, DIVR = EA, QUO = AX, REM = DX, MAX = 7FFFH

Timing (clocks): 8-bit 112+EA
16-bit 177+EA

Operation:

(temp) <== (NUMR)
if (temp) / (DIVR) > 0 and (temp) / (DIVR) > MAX
or (temp) / (DIVR) < 0 and (temp) / (DIVR) < 0 - MAX - 1
then
(QUO), (REM) undefined
(SP) <== (SP) - 2
((SP)+1:(SP)) <== FLAGS
(IF) <== 0
(TF) <== 0
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (CS)
(CS) <== (2)
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (IP)
(IP) <== (0)
else
(QUO) <== (temp) / (DIVR), where / is signed division
(REM) <== (temp) % (DIVR), where % is signed modulo

Flags Affected:

AF, CF, OF, PF, SF, ZF undefined

Mnemonic: **AAD**

Description: AAD (Unpacked BCD (ASCII) adjust for division) performs an adjustment of the dividend in AL before dividing two unpacked decimal operands, so that the result of the division will be an unpacked decimal quotient.

Encoding:

1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0

Timing: 60 clocks

Operation:

(AL) <== (AH) * OAH + (AL)
(AH) <== 0

Flags Affected:

PF, SF, ZF.
AF, CF, OF undefined

Mnemonic: **CBW**

Description: CBW (convert byte to word) performs a sign extension of the AL register into the AH register.

Encoding:

1 0 0 1 1 0 0 0

Timing: 2 clocks

Operation:

if (AL) < 80H then (AH) <== 0 else (AH) <== FFH

Flags Affected:

None

Mnemonic: **CWD**

Description: CWD (convert word to double word) performs a sign extension of the AX register into the DX register.

Encoding:

1 0 0 1 1 0 0 1

Timing: 5 clocks

Operation:

if (AX) < 8000H then (DX) <== 0
else (DX) <== FFFFH

Flags Affected:

None

4.4.5 Logic

Nine single-operand logical instructions are provided:

NOT	Not
SHL	Shift Left
SAL	Shift Arithmetic Left
SHR	Shift Right
SAR	Shift Arithmetic Right
ROL	Rotate Left
ROR	Rotate Right
RCL	Rotate through Carry Left
RCR	Rotate through Carry Right

Mnemonic: **NOT**

Description: NOT forms the ones complement of (inverts) the source operand and returns the result to the operand. Flags are not affected.

Encoding:

1 1 1 1 0 1 1 w	mod 0 1 0 r/m

if w = 0 then SRC = FFH
else SRC = FFFFH

Timing (clocks):	register	3
	memory	16+EA

Operation:

(EA) <== SRC - (EA)

Flags Affected:

None

SHIFTS

Shift operations of four varieties are provided for memory and register operands, SHL (shift logical left), SHR (shift logical right), SAL (shift arithmetic left), and SAR (shift arithmetic right). Single bit shifts, and variable bit shifts with the shift count taken from the CL register are available. The CF flag becomes the last bit shifted out; OF is set if the final sign bit value of a single shift differs from the previous value of the sign bit; and PF, SF, and ZF are set to reflect the result value.

Mnemonic: **SHL and SAL**

Description: SHL (shift logical left) and SAL (shift arithmetic left) shift the source operand left by COUNT bits, shifting in low-order zero bits.

Encoding:

1 1 0 1 0 0 v w	mod 1 0 0 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)
Timing (clocks):

single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

(temp) <== COUNT
do while (temp) $\neq$ 0
  (CF) <== high-order bit of (EA)
  (EA) <== (EA) * 2
  (temp) <== (temp) - 1
if COUNT = 1 then
  if high-order bit of (EA) $\neq$ (CE) then (OF) <== 1
  else (OF) <== 0
else (OF) undefined

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

Mnemonic: **SHR**

Description: SHR (shift logical right) shifts the source operand right by COUNT bits, shifting in high-order zero bits.

Encoding:

1 1 0 1 0 0 v w	mod 1 0 1 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)
Timing (clocks):

single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

(temp) <== COUNT
do while (temp) $\neq$ 0
  (CF) <== low-order bit of (EA)
  (EA) <== (EA) / 2, where / is equivalent to unsigned division
  (temp) <== (temp) - 1
if COUNT = 1 then
  if high-order bit of (EA) $\neq$ next-to-high-order bit of (EA) then (OF) <== 1
  else (OF) <== 0
else (OF) undefined

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

Mnemonic: **SAR**

Description: SAR (shift arithmetic right) shifts the source operand right by COUNT bits, shifting in high-order bits equal to the original high-order bit of the operand (sign extension).

Encoding:

1 1 0 1 0 0 v w	mod 1 1 1 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)
Timing (clocks):

single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

(temp) <== COUNT
do while (temp) $\neq$ 0
  (CF) <== low-order bit of (EA)
  (EA) <== (EA) / 2, where / is equivalent to signed division, rounding down
  (temp) <== (temp) - 1
if COUNT = 1 then
  if high-order bit of (EA) $\neq$ next-to-high-order bit of (EA) then (OF) <== 1
else (OF) <== 0
else (OF) <== 0

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

ROTATES

Rotate operations of four varieties are provided for memory and register operands, ROL (rotate left), ROR (rotate right), RCL (rotate through CF left), and RCR (rotate through CF right). Single bit rotates, and variable bit rotates with the rotate count taken from the CL register are available. The CF flag becomes the last bit rotated out; OF is set if the final sign bit value differs from the previous value of the sign bit.

Mnemonic: **ROL**

Description: ROL (rotate left) rotates the source operand left by COUNT bits.

Encoding:

1 1 0 1 0 0 v w	mod 0 0 0 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)

Timing (clocks):	
single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

```
(temp) <== COUNT
do while (temp) ≠ 0
 (CF) <== high-order bit of (EA)
 (EA) <== (EA) * 2 + (CF)
 (temp) <== (temp) -1
if COUNT = 1 then
 if high-order bit of (EA) ≠ (CF) then (OF) <== 1
 else (OF) <== 0
else (OF) undefined
```

Flags Affected:

CF, OF

Mnemonic: **ROR**

Description: ROR (rotate right) rotates the source operand right by COUNT bits.

Encoding:

1 1 0 1 0 0 v w	mod 0 0 1 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)

Timing (clocks):	
single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

```
(temp) <== COUNT
do while (temp) ≠ 0
 (CF) <== low-order bit of (EA)
 (EA) <== (EA) / 2
 high-order bit of (EA) <== (CF)
 (temp) <== (temp) -1
if COUNT = 1 then
 if high-order bit of (EA) ≠ next-to-high-order bit of (EA)
   then (OF) <== 1
 else (OF) <== 0
else (OF) undefined
```

Flags Affected:

CF, OF

Mnemonic: **RCL**

Description: RCL (rotate through carry flag left) rotates the source operand left through the CF flag register by COUNT bits.

Encoding:

1 1 0 1 0 0 v w	mod 0 1 0 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)

Timing (clocks):	
single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

```
(temp) <== COUNT
do while (temp) ≠ 0
 (tmpcf) <== (CF)
 (CF) <== high-order bit of (EA)
 (EA) <== (EA) * 2 + (tmpcf)
 (temp) <== (temp) - 1
if COUNT = 1 then
 if high-order bit of (EA) ≠ (CF) then (OF) <== 1
 else (OF) <== 0
else (OF) undefined
```

Flags Affected:

CF, OF

Mnemonic: **RCR**

Description: RCR (rotate through carry flag right) rotates the EA operand right through the CF flag register by COUNT bits.

Encoding:

1 1 0 1 0 0 v w	mod 0 1 1 r/m

if v = 0 then COUNT = 1
else COUNT = (CL)

Timing (clocks):	
single-bit register	2
single-bit memory	15+EA
variable-bit register	8+4/bit
variable-bit memory	20+EA+4/bit

Operation:

(temp) <== COUNT
do while (temp) ≠ 0
 (tmpcf) <== (CF)
 (CF) <== low-order bit of (EA)
 (EA) <== (EA) / 2
 high-order bit of (EA) <== (tmpcf)
 (temp) <== (temp) - 1
if COUNT = 1 then
 if high-order bit of (EA) ≠ next-to-high-order bit of (EA)
  then (OF) <== 1
 else (OF) <== 0
else (OF) undefined

Flags Affected:

CF, OF

Four two-operand logic operations are provided:

AND	And
TEST	Test
OR	Or
XOR	Exclusive Or

Mnemonic: **AND**

Description: AND performs the bitwise logical conjunction of the two source operands and returns the result to one of the operands.

Encoding:

Memory or Register Operand with Register Operand:

0 0 1 0 0 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA

Timing (clocks):	
register to register	3
memory to register	9+EA
register to memory	16+EA

Immediate Operand to Memory or Register Operand:

1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1

LSRC = EA, RSRC = data, DEST = EA

Timing (clocks):	
immediate to register	4
immediate to memory	17+EA

Immediate Operand to Accumulator:

0 0 1 0 0 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing (clocks): immediate to register 4

Operation:

(DEST) <== (LSRC) & (RSRC)
(CF) <== 0
(OF) <== 0

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

Mnemonic: **TEST**

Description: TEST performs the bitwise logical conjunction of the two source operands, causing the flags to be affected but does not return the result.

Encoding:

Memory or Register Operand with Register Operand:

1 0 0 0 0 1 0 w	mod reg r/m

LSRC = REG, RSRC = EA
Timing (clocks): register with register 3
register with memory 9+EA

Immediate Operand with Memory or Register Operand:

1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1

LSRC = EA, RSRC = data
Timing (clocks): immediate with register 4
immediate with memory 10+EA

Immediate Operand with Accumulator:

1 0 1 0 1 0 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data
else LSRC = AX, RSRC = data
Timing (clocks): immediate with register 4

Operation:

(LSRC) & (RSRC)
(CF) <== 0
(OF) <== 0

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

Mnemonic: **OR**

Description: OR performs the bitwise logical inclusive disjunction of the two source operands and returns the result to one of the operands.

Encoding:

Memory or Register Operand with Register Operand:

0 0 0 0 1 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA
Timing (clocks): register to register 3
memory to register 9+EA
register to memory 16+EA

Immediate Operand to Memory or Register Operand:

1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1

LSRC = EA, RSRC = data, DEST = EA
Timing (clocks): immediate to register 4
immediate to memory 17+EA

Immediate Operand to Accumulator:

0 0 0 0 1 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing (clocks): immediate to register 4

Operation:

(DEST) <== (LSRC)|(RSRC)
(CF) <== 0
(OF) <== 0

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

Mnemonic: **XOR**

Description: XOR (exclusive Or) performs the bitwise logical exclusive disjunction of the two source operands and returns the result to one of the operands.

Encoding:

Memory or Register Operand with Register Operand:

0 0 1 1 0 0 d w	mod reg r/m

if d = 1 then LSRC = REG, RSRC = EA, DEST = REG
else LSRC = EA, RSRC = REG, DEST = EA
Timing (clocks): register to register 3
memory to register 9+EA
register to memory 16+EA

Immediate Operand to Memory or Register Operand:

1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1

LSRC = EA, RSRC = data, DEST = EA
Timing (clocks): immediate to register 4
immediate to memory 17+EA

Immediate Operand to Accumulator:

0 0 1 1 0 1 0 w	data	data if w=1

if w = 0 then LSRC = AL, RSRC = data, DEST = AL
else LSRC = AX, RSRC = data, DEST = AX
Timing (clocks): immediate to register 4

Operation:

(DEST) <== (LSRC) $\veebar$ (RSRC)
(CF) <== 0
(OF) <== 0

Flags Affected:

CF, OF, PF, SF, ZF.
AF undefined

4.4.6 String Manipulation

The 8086 provides a group of one-byte instructions which perform various primitive operations for the manipulation of byte and word strings (sequences of bytes or words). These primitive operations can be performed repeatedly by preceding the instruction with a special prefix. The single-operation forms may be combined to form complex string operations with repetition provided by special interation operations.

Hardware Operation Control. All primitive string operations use the SI register to address the source operands, which are assumed to be in the current data segment. The DI register is used to address the destination operands, which are assumed to reside in the current extra segment. If the DF flag is cleared the operand pointers are incremented after each operation, once for byte operations and twice for word operations. If the DF flag is set the operand pointers are decremented after each operation. See Processor Control for setting and clearing DF.

Any of the primitive string operation instructions may be preceded with a one-byte prefix indicating that the operation is to be repeated until the operation count in CX is decremented to zero. The test for completion is made prior to each repetition of the operation. Thus, an initial operation count of zero will cause zero executions of the primitive operation.

The repeat prefix byte also designates a value to compare with the ZF flag. If the primitive operation is one which affects the ZF flag, and the ZF flag is unequal to the designated value after any execution of the primitive operation, the repetition is terminated. This permits the scan operation to serve as a scan-while or a scan-until.

During the execution of a repeated primitive operation the operand pointer registers (SI and DI) and the operation count register (CX) are updated after each repetition whereas the instruction pointer will retain the offset address of the repeat prefix byte (assuming it immediately precedes the string operation instruction). Thus, an interrupted repeated operation will be correctly resumed when control returns from the interrupting task.

A caution is in order at this point: It is possible to include two other special prefix bytes with a repeat-prefixed string instruction, one which overrides the default segment addressing for the SI operand (Section 2.3.1), and one which locks the bus to prohibit access by other bus masters. Execution of the repeated string operation will not resume properly following an interrupt if more than one prefix is present preceding the string primitive. Execution will resume one byte before the primitive (presumably where the repeat prefix resides), thus ignoring the additional prefixes.

Software Operation Control. The repeat prefix provides for rapid iteration in a hardware-repeated string operation. The iteration control operations (Section 4.4.7) provide this same control for implementing software loops to perform complex string operations. These iteration operations provide the same operation count update, operation completion test, and ZF flag tests that the repeat prefix provides.

By combining the primitive string operations and iteration control operations with other operations, it is possible to build sophisticated yet efficient string manipulation routines. One instruction that is particularly useful in this context is XLAT; it permits a byte fetched from one string to be translated before being stored in a second string, or before being operated upon in some other fashion. The translation is performed by using the value in the AL register to index into a table pointed at by the BX register. The translated value obtained from the table then replaces the value initially in the AL register.

As an example of the use of the primitive string operations and iteration control operations to implement a complex string operation, consider the following application: An input driver must translate a buffer of EBCDIC characters into ASCII, and transfer characters until one of several different EBCDIC control characters is encountered. The transferred ASCII string is to be terminated with an EOT character. To accomplish this, SI is initialized to point to the beginning of the EBCDIC buffer, DI is initialized to point to the beginning of the buffer to receive the ASCII characters, BX is made to point to an EBCDIC to ASCII translation table, and CX is initialized to contain the length of the EBCDIC buffer (possibly empty). The translation table contains the ASCII equivalent for each EBCDIC character, perhaps with ASCII NULs for illegal characters. The EOT code is placed into those entries in the table corresponding to the desired EBCDIC stop characters. The 8086 instruction sequence to implement this example is the following:

```
      JCXZ    Empty   ;skip if input buffer empty
Next: LODB    Ebcbuf  ;fetch next EBCDIC
                      character (from SI)
      XLAT    Table   ;translate it to ASCII (from
                      BX)
      CMP     AL,EOT  ;test for the EOT
      STOB    Ascbuf  ;translate ASCII character
                      (to DI)
      LOOPNE  Next    ;continue if not EOT
      •
      •
      •
Empty:
```

The body of this loop requires seven bytes of code.

Mnemonic: **REP**

Description: REP (repeat) causes the succeeding primitive string operation to be performed repeatedly while (CX) is not zero. In the case of CMPB, CMPW, SCAB, and SCAW, if after any repetition of the primitive operation the ZF flag differs from the "z" bit of the repeat prefix, the repetition is terminated. This prefix may be combined with the segment override and/or LOCK prefixes, although this has certain problems.

Encoding:

1 1 1 1 0 0 1 z

Timing: 6 clocks/loop

Operation:

do while (CX) $\neq$ 0
  service pending interrupt (if any)
  execute primitive string operation in succeeding byte
  (CX) <== (CX) - 1
  if primitive operation is CMPB, CMPW, SCAB, or SCAW and (ZF) $\neq$ z then exit from while loop

Flags Affected:

None

Five primitive string operations are provided, each of which has both a byte instruction and a word instruction, as follows:

MOVB	Move Byte
MOVW	Move Word
CMPB	Compare byte
CMPW	Compare Word
SCAB	Scan Byte
SCAW	Scan Word
LODB	Load byte
LODW	Load Word
STOB	Store Byte
STOW	Store Word

Mnemonic: **MOVB and MOVW**

Description: MOVB (or MOVW) transfers a byte (or word) operand from the source operand addressed by SI to the destination operand addressed by DI, and adjusts the SI and DI registers by DELTA. As a repeated operation this provides for moving a string from one location in memory to another.

Encoding:

1 0 1 0 0 1 0 w

if w = 0 then SRC = (SI), DEST = (DI), DELTA = 1
else SRC = (SI)+1:(SI), DEST = (DI)+1:(DI), DELTA = 2
Timing: 17 clocks

Operation:

(DEST) <== (SRC)
if (DF) = 0 then
  (SI) <== (SI) + DELTA
  (DI) <== (DI) + DELTA
else
  (SI) <== (SI) - DELTA
  (DI) <== (DI) - DELTA

Flags Affected:

None

Mnemonic: **CMPB and CMPW**

Description: CMPB (or CMPW) subtracts the destination byte (or word) operand addressed by DI from the source operand addressed by SI and affects the flags but does not return the result. As a repeated operation this provides for comparing two strings. With the appropriate repeat prefix it is possible to determine after which string element the two strings become unequal, thereby establishing an ordering between the strings.

Encoding:

1 0 1 0 0 1 1 w

if w = 0 then LSRC = (SI), RSRC = (DI), DELTA = 1
else LSRC = (SI)+1:(SI), RSRC = (DI)+1:(DI), DELTA = 2
Timing: 22 clocks

Operation:

(LSRC) - (RSRC)
if (DF) = 0 then
 (SI) <== (SI) + DELTA
 (DI) <== (DI) + DELTA
else
 (SI) <== (SI) - DELTA
 (DI) <== (DI) - DELTA

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **SCAB and SCAW**

Description: SCAB (or SCAW) subtracts the destination byte (or word) operand addressed by DI from AL (or AX) and affects the flags but does not return the result. As a repeated operation this provides for scanning for the occurrence of, or departure from, a given value in a string.

Encoding:

1 0 1 0 1 1 1 w

if w = 0 then LSRC = AL, RSRC = (DI), DELTA = 1
else LSRC = AX, RSRC = (DI)+1:(DI), DELTA = 2
Timing: 15 clocks

Operation:

(LSRC) - (RSRC)
if (DF) = 0 then (DI) <== (DI) + DELTA
else (DI) <== (DI) - DELTA

Flags Affected:

AF, CF, OF, PF, SF, ZF

Mnemonic: **LODB and LODW**

Description: LODB (or LODW) transfers a byte (or word) operand from the source operand addressed by SI to accumulator AL (or AX) and adjusts the SI register by DELTA. This operation ordinarily would not be repeated.

Encoding:

1 0 1 0 1 1 0 w

if w = 0 then SRC = (SI), DEST = AL, DELTA = 1
else SRC = (SI)+1:(SI), DEST = AX, DELTA = 2
Timing: 12 clocks

Operation:

(DEST) <== (SRC)
if (DF) = 0 then (SI) <== (SI) + DELTA
else (SI) <== (SI) - DELTA

Flags Affected:

None

Mnemonic: **STOB and STOW**

Description: STOB (or STOW) transfers a byte (or word) operand from AL (or AX) to the destination operand addressed by DI and adjusts the DI register by DELTA. As a repeated operation this provides for filling a string with a given value.

Encoding:

1 0 1 0 1 0 1 w

if w = 0 then SRC = AL, DEST = (DI), DELTA = 1
else SRC = AX, DEST = (DI)+1:(DI), DELTA = 2
Timing: 10 clocks

Operation:

(DEST) <== (SRC)
if (DF) = 0 then (DI) <== (DI) + DELTA
else (DI) <== (DI) - DELTA

Flags Affected:

None

4.4.7 Control Transfer

Four classes of control transfer operations may be distinguished: calls, jumps, and returns; conditional transfers; iteration control; and interrupts.

All control transfer operations cause, perhaps upon a certain condition, the program execution to continue at some new location in memory, possibly in a new code segment.

NOTE: Queue reinitialization is not included in the timing information for transfer operations. To account for queue loading, add 4 clocks to timing numbers.

Calls, Jumps, and Returns. Two basic varieties of calls, jumps, and returns are provided — those which transfer control within the current code segment, and those which transfer control to an arbitrary code segment, which then becomes the current code segment. Both direct and indirect transfers are supported: indirect transfers make use of the standard addressing modes described in Section 4.4. Intra-segment direct calls and jumps specify a self-relative direct displacement, thus allowing position independent code. A shortened jump instruction is available for transfers within ±128 bytes from the instruction, using less code.

The three transfer operations are:

CALL Call
JMP Jump
RET Return

Mnemonic: **CALL**

Description: CALL pushes the offset address of the next instruction onto the stack (in the case of an inter-segment transfer the CS segment register is pushed first) and then transfers control to the target operand.

Encoding:

Intra-segment Direct:

1 1 1 0 1 0 0 0	disp-low	disp-high

DEST = (EA)
Timing: 13+EA clocks

Intra-Segment Indirect:

1 1 1 1 1 1 1 1	mod 0 1 0 r/m

DEST = (IP) + disp
Timing: 11 clocks

Inter-Segment Direct:

1 0 0 1 1 0 1 0	offset-low	offset-high
	seg-low	seg-high

DEST = offset, SEG = seg
Timing: 20 clocks

Inter-Segment Indirect:

1 1 1 1 1 1 1 1	mod 0 1 1 r/m

DEST = (EA), SEG = (EA + 2)
Timing: 29+EA clocks

Operation:

```
if Inter-Segment then
  (SP) <== (SP) - 2
  ((SP)+1:(SP)) <== (CS)
  (CS) <== SEG
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (IP)
(IP) <== DEST
```

Flags Affected:

None

Mnemonic: **JMP**

Description: JMP transfers control to the target operand.

Encoding:

Intra-Segment Direct:

1 1 1 0 1 0 0 1	disp-low	disp-high

DEST = (IP) + disp
Timing: 7 clocks

Intra-Segment Direct Short:

1 1 1 0 1 0 1 1	disp

DEST = (IP) + disp sign extended to 16-bits
Timing: **Timing: 2 clocks**

Intra-Segment Indirect:

1 1 1 1 1 1 1 1	mod 1 0 0 r/m

DEST = (EA)
Timing: 7+EA clocks

Inter-Segment Direct:

1 1 1 0 1 0 1 0	offset-low	offset-high
	seg-low	seg-high

DEST = offset, SEG = seg
Timing: 7 clocks

Inter-Segment Indirect:

1 1 1 1 1 1 1 1	mod 1 0 1 r/m

DEST = (EA), SEG = (EA + 2)
Timing: 16+EA clocks

Operation:

if Inter-Segment then (CS) <== SEG
(IP) <== DEST

Flags Affected:

None

Mnemonic: **RET**

Description: RET transfers control to the return address pushed by a previous CALL operation and optionally adds an immediate constant to the SP register so as to discard stack parameters.

Encoding:

Intra-Segment:

1 1 0 0 0 0 1 1

Timing: 8 clocks

Intra-Segment and Add Immediate to Stack Pointer:

1 1 0 0 0 0 1 0	data-low	data-high

Timing: 12 clocks

Inter-Segment:

1 1 0 0 1 0 1 1

Timing: 18 clocks

Inter-Segment and Add Immediate to Stack Pointer:

1 1 0 0 1 0 1 0	data-low	data-high

Timing: 17 clocks

Operation:

(IP) <== ((SP)=1:(SP))
(SP) <== (SP) + 2
if Inter-Segment then
(CS) <== ((SP)+1:(SP))
(SP) <== (SP) + 2
if Add Immediate to Stack Pointer then (SP) <== (SP) + data

Flags Affected:

None

Conditional Transfers. The conditional transfers of control perform a jump contingent upon various Boolean functions of the flag registers. The destination must be within ±128 bytes distance from the instruction. Table 4-2 shows the available instructions, the conditions associated with them, and their interpretation.

TABLE 4-2.
8086 CONDITIONAL TRANSFER OPERATIONS

Instruction	Condition	Interpretation
JE or JZ	ZF = 1	"equal" or "zero"
JL or JNGE	(SF xor OF) = 1	"less" or "not greater or equal"
JLE or JNG	((SP xor OF) or ZF) = 1	"less or equal" or "not greater"
JB or JNAE	CF = 1	"below" or "not above or equal"
JBE or JNA	(CF or ZF) = 1	"below or equal" or "not above"
JP or JPE	PF = 1	"parity" or "parity even"
J0	OF = 1	"overflow"
JS	SF= 1	"sign"
JNE or JNZ	ZF = 0	"not equal" or "not zero"
JNL or JGE	(SF xor OF) = 0	"not less" or "greater or equal"
JNLE or JG	((SF xor OF) or ZF) = 0	"not less or equal" or "greater"
JNB or JAE	CF = 0	"not below" or "above or equal"
JNBE or JA	(CF or ZF) = 0	"not below or equal" or "above"
JNP or JPO	PF = 0	"not parity" or "parity odd"
JNO	OF = 0	"not overflow"
JNS	SF = 0	"not sign"

*"Above" and "below" refer to the relation between two unsigned values, while "greater" and "less" refer to the relation between two signed values.

Mnemonic: **JE and JZ**

Description: JE (or JZ) transfers control to the target operand on equal (or zero).

Encoding:

0 1 1 1 0 1 0 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (ZF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JL and JNGE**

Description: JL (or JNGE) transfers control to the target operand on less (or not greater or equal).

Encoding:

0 1 1 1 1 1 0 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (SF)¦¦(OF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JLE and JNG**

Description: JLE (or JNG) transfers control to the target operand on less or equal (or not greater).

Encoding:

0 1 1 1 1 1 1 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if [(SF)¦¦(OF)] 1(ZF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JB and JNAE**

Description: JB (or JNAE) transfers control to the target operand on below (or not above or equal).

Encoding:

0 1 1 1 0 0 1 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (CF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JBE and JNA**

Description: JBE (or JNA) transfers control to the target operand on below or equal (or not above).

Encoding:

0 1 1 1 0 1 1 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (CF)¦(ZF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JP and JPE**

Description: JP (or JPE) transfers control to the target operand on parity) (or parity even).

Encoding:

0 1 1 1 1 0 1 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (PF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JO**

Description: JO transfers control to the target operand on overflow.

Encoding:

0 1 1 1 0 0 0 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (OF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JS**

Description: JS transfers control to the target operand on sign.

Encoding:

0 1 1 1 1 0 0 0	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (SF) = 1 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNE and JNZ**

Description: JNE (or JNZ) transfers control to the target operand on not equal (or not zero).

Encoding:

0 1 1 1 0 1 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (ZF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNL and JGE**

Description: JNL (or JGE) transfers control to the target operand on not less (or greater or equal).

Encoding:

0 1 1 1 1 1 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (SF) ¦¦ (OF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNLE and JG**

Description: JNLE (or JG) transfers control to the target operand on not less or equal (or greater).

Encoding:

0 1 1 1 1 1 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if ((SP ¦¦ (OF)) ¦ (ZF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNB and JAE**

Description: JNB (or JAE) transfers control to the target operand on not below (or above or equal).

Encoding:

0 1 1 1 0 0 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (CF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNBE and JA**

Description: JNBE (or JA) transfers control to the target operand on not below or equal (or above).

Encoding:

0 1 1 1 0 1 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (CF) ¦ (ZF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNP and JPO**

Description: JNP (or JPO) transfers control to the target operand on not parity (or parity odd).

Encoding:

0 1 1 1 1 0 1 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (PF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNO**

Description: JNO transfers control to the target operand on not overflow.

Encoding:

0 1 1 1 0 0 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (OF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JNS**

Description: JNS transfers control to the target operand on not sign.

Encoding:

0 1 1 1 1 0 0 1	disp

Timing (clocks): Jump is taken 8
Jump is not taken 4

Operation:

if (SF) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Iteration Control. The iteration control transfer operations perform leading- and trailing-decision loop control. The destination of iteration control transfers must be within ±128 bytes distance from the instruction. These operations are particularly useful in conjunction with the string manipulation operations. (See Section 4.4.6)

Four iteration control transfer operations are provided:

LOOP	Loop
LOOPZ	Loop While Zero
(LOOPE	Loop While Equal)
LOOPNZ	Loop While Not Zero
(LOOPNE	Loop While not Equal)
JCXZ	Jump on CX Zero

Mnemonic: **LOOP**

Description: LOOP decrements the CX (count) register by 1 and transfers control to the target operand if CX is not zero.

Encoding:

1 1 1 0 0 0 1 0	disp

Timing (clocks): Jump is taken 9
Jump is not taken 5

Operation:

(CX) <== (CX) - 1
if (CX) ≠ 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **LOOPZ and LOOPE**

Description: LOOPZ, also called LOOPE (loop while zero or loop while equal) decrements the CX register by one and transfers if CX is not zero and if the ZF flag is set.

Encoding:

1 1 1 0 0 0 0 1	disp

Timing (clocks): Jump is taken 11
Jump is not taken 5

Operation:

(CX) <== (CX) - 1
if (ZF) = 1 and (CX) ≠ 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **LOOPNZ and LOOPNE**

Description: LOOPNZ, also called LOOPNE, (loop while not zero or loop while not equal) decrements the CX register by one and transfers if CX is not zero and the ZF flag is cleared.

Encoding:

1 1 1 0 0 0 0 0	disp

Timing (clocks): Jump is taken 11
Jump is not taken 5

Operation:

(CX) <== (CX) - 1
if (ZF) = 0 and (CX) ≠ 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Mnemonic: **JCXZ**

Description: JCXZ (jump on CX zero) transfers control to the target operand if the CX register is zero.

Encoding:

1 1 1 0 0 0 1 1	disp

Timing (clocks): Jump is taken 9
Jump is not taken 5

Operation:

if (CX) = 0 then
(IP) <== (IP) + disp (sign-extended to 16-bits)

Flags Affected:

None

Interrupts. Program execution control may be transferred by means of operations similar in effect to that of external interrupts (see Section 2.4.2). All interrupts perform a transfer by pushing the flag registers onto the stack (as in PUSHF), and then performing an indirect call (of the inter-segment variety) through an element of an interrupt transfer vector located at absolute locations 0 through 3FFH. This vector contains a four-byte element for each of up to 256 different interrupt types.

Three interrupt transfer operations are provided:

INT Interrupt
INTO Interrupt on Overflow
IRET Interrupt Return

Mnemonic: **INT**

Description: INT pushes the flag registers (as in PUSHF), clears the TF and IF flags, and transfers control with an indirect call through any one of the 256 vector elements. The one-byte form of this instruction generates a type 3 interrupt.

Encoding:

1 1 0 0 1 1 0 v	type if v=1

if v = 0 then TYPE = 3
else TYPE = type
Timing: 52 clocks

Operation:

(SP) <== (SP) - 2
((SP)+1:(SP)) <== FLAGS
(IF) <== 0
(TF) <== 0
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (CS)
(CS) <== (TYPE * 4 + 2)
(SP) <==)SP) - 2
((SP)+1:(SP)) <== (IP)
(IP) <== (TYPE * 4)

Flags Affected:

IF, TF

Mnemonic: **INTO**

Description: INTO pushes the flag registers (as in PUSHF), clears the TF and IF flags, and transfers control with an indirect call through vector element 4 (location 10H) if the OF flag is set (trap on overflow). If the OF flag is clear, no operation takes place.

Encoding:

1 1 0 0 1 1 1 0

Timing: 52 clocks

Operation:

if (OF) = 1 then
(SP) <== (SP) - 2
((SP)+1:(SP)) <== FLAGS
(IF) <== 0
(TF) <== 0
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (CS)
(CS) <== (12H)
(SP) <== (SP) - 2
((SP)+1:(SP)) <== (IP)
(IP) <== (10H)

Flags Affected:

None

Mnemonic: **IRET**

Description: IRET transfers control to the return address saved by a previous interrupt operation and restores the saved flag registers (as in POPF).

Encoding:

1 1 0 0 1 1 1 1

Timing: 24 clocks

Operation:

(IP) <== ((SP)+1:(SP))
(SP) <== (SP) + 2
(CS) <== ((SP)+1:(SP))
(SP) <== (SP) + 2
FLAGS <== ((SP)+1:(SP))
(SP) <== (SP) + 2

Flags Affected:

All

4.4.8 Processor Control

The 8086 provides various instructions and mechanisms for controlling the operation of the processor and its interaction with its environment.

Flag Operations: Seven operations are provided which operate directly on individual flag registers:

CLC Clear Carry
CMC Complement Carry
STC Set Carry
CLD Clear Direction
STD Set Direction
CLI Clear Interrupt
STI Set Interrupt

Mnemonic: **CLC**

Description: CLC clears the CF flag.

Encoding:

1 1 1 1 1 0 0 0

Timing: 2 clocks

Operation:

(CF) <== 0

Flags Affected:

CF

Mnemonic: **CMC**

Description: CMC complements the CF flag.

Encoding:

1 1 1 1 0 1 0 1

Timing: 2 clocks

Operation:

if (CF) = 0 then (CF) <== 1 else (CF) <== 0

Flags Affected:

CF

Mnemonic: **STC**

Description: STC sets the CF flag.

Encoding:

1 1 1 1 1 0 0 1

Timing: 2 clocks

Operation:

(CF) <== 1

Flags Affected:

CF

Mnemonic: **CLD**

Description: CLD clears the DF flag, causing the string operations to auto-increment the operand pointers.

Encoding:

1 1 1 1 1 1 0 0

Timing: 2 clocks

Operation:

(DF) <== 0

Flags Affected:

DF

Mnemonic: **STD**

Description: STD sets the DF flag, causing the string operations to auto-decrement the operand pointers.

Encoding:

1 1 1 1 1 1 0 1

Timing: 2 clocks

Operation:

(DF) <== 1

Flags Affected:

DF

Mnemonic: **CLI**

Description: CLI clears the IF flag, disabling maskable external interrupts, which appear on the INTR line of the 8086. (Nonmaskable interrupts, which appear on the NMI line, are not disabled).

Encoding:

1 1 1 1 1 0 1 0

Timing: 2 clocks

Operation:

(IF) <== 0

Flags Affected:

IF

Mnemonic: **STI**

Description: STI sets the IF flag, enabling maskable external interrupts after the execution of the next instruction.

Encoding:

1 1 1 1 1 0 1 1

Timing: 2 clocks

Operation:

(IF) <== 1

Flags Affected:

IF

Processor Halt

Mnemonic: **HLT**

Description: The HLT instruction causes the 8086 processor to enter its halt state. The halt state is cleared by an enabled external interrupt or reset.

Encoding:

1 1 1 1 0 1 0 0

Timing: 2 clocks

Operation:

None

Flags Affected:

None

Processor Wait

Mnemonic: **WAIT**

Description: The WAIT instruction causes the processor to enter a wait state if the signal on its $\overline{TEST}$ pin is not asserted. The wait state may be interrupted by an enabled external interrupt. When this occurs the saved code location is that of the WAIT instruction, so that upon return from the interrupting task the wait state is reentered. The wait state is cleared and execution resumed when the $\overline{TEST}$ signal is asserted. Execution resumes without allowing external interrupts until after the execution of the next instruction. This instruction allows the processor to synchonize itself with external hardware.

Encoding:

1 0 0 1 1 0 1 1

Timing: 3 clocks

Operation:

None

Flags Affected:

None

Processor Escape

Mnemonic: **ESC**

Description: The ESC instruction provides a mechanism by which other processors may receive their instructions from the 8086 instruction stream and make use of the 8086 addressing modes. The 8086 processor does no operation for the ESC instruction other than to access a memory operand and place it on the bus.

Encoding:

1 1 0 1 1 x	mod x r/m

Timing: 7+EA clocks

Operation:

if mod $\neq$ 11 then data bus <== (EA)

Flags Affected:

None

Bus Lock

Mnemonic: **LOCK**

Description: A special one-byte lock prefix may precede any instruction. It causes the processor to assert its bus-lock signal for the duration of the operation caused by that instruction. In multiple processor systems with shared resources it is necessary to provide mechanisms to enforce controlled access to those resources. Such mechanisms, while generally provided through software operating systems, require hardware assistance. A sufficient mechanism for accomplishing this is a locked exchange (also known as test-and-set-lock).

It is assumed that external hardware, upon receipt of that signal, will prohibit bus access for other bus masters during the period of its assertion.

The instruction most useful in this context is an exchange register with memory. A simple software lock may be implemented with the following code sequence:

```
Check:  MOV   AL1       ;set AL to 1 (implies locked)
  LOCK  XCHG  Sema,AL   ;test and set lock
        TEST  AL,AL     ;set flags based on AL
        JNZ   Check     ;retry if lock already set
        •
        •
        MOV   Sema,0    ;clear the lock when done
```

The LOCK prefix may be combined with the segment override and/or REP prefixes, although the latter has certain problems. (See Section 4.4.6.)

Encoding:

1 1 1 1 0 0 0 0

Timing: 2 clocks

Operation:

None

Flags Affected:

None

Single Step. When the TF flag register is set, the processor generates a type 1 interrupt after the execution of each instruction. During interrupt transfer sequences caused by any type of interrupt, the TF flag is cleared after the push-flags step of the interrupt sequence. No instructions are provided for setting or clearing TF directly. Rather, the flag register file image saved on the stack by a previous interrupt operation must be modified, so that the subsequent interrupt return operation (IRET) restores TF set. This allows a diagnostic task to single-step through a task under test, while still executing normally itself.

If the single-stepped instruction itself clears the TF flag, the type 1 interrupt will still occur upon completion of the single-stepped instruction. If the single-stepped instruction generates an interrupt or if an enabled external interrupt occurs prior to the completion of the single-stepped instruction, the type 1 interrupt sequence will occur after the interrupt sequence of the generated or external interrupt, but before the first instruction of that interrupt service routine is executed.

TABLE 4-3. 8080 TO 8086 INSTRUCTION MAPPING

8080	8086
STAX B	MOV DI,CX STOB [DI]
LDAX B	MOV SI,CX LODB [SI]
STAX D	MOV DI,DX STOB [DI]
LDAX D	MOV SI,DX LODB [SI]
SHLD addr	MOV addr,BX
LHLD addr	MOV BX,addr
STA addr	MOV addr,AL
LDA addr	MOV AL,addr
INR reg	INC reg (See Figure 2-7)
DCR reg	DEC reg
MVI reg,data	MOV reg,data
MVI M,data	MOV [BX],data
MOV reg,reg	MOV reg,reg
MOV M,A	MOV [BX],AL
MOV A,M	MOV AL,[BX]
LXI reg,data	MOV reg,data
DAD reg	LAHF ADD BX,reg RCR SI SAHF RCL SI or ADD BX,reg (unlike DAD—will affect AF, PF, SF and ZF)
INX reg	LAHF INC reg SAHF or INC reg (unlike INX—will affect AF, PF, SF, and ZF)
DCX reg	LAHF DEC reg SAHF or DEC reg (unlike DCX—will affect AF, PF, SF, and ZF)
POP reg	POP reg
PUSH reg	PUSH reg
POP PSW	POP AX SAHF
PUSH PSW	LAHF PUSH AX
PCHL	JMP BX
SPHL	MOV SP,BX
XTHL	POP SI XCHG BX,SI PUSH SI
XCHG	XCHG DX, BX
ADD reg	ADD AL,reg
ADC reg	ADC AL,reg
SUB reg	SUB AL,reg
SBB reg	SBB AL, reg
ANA reg	AND AL,reg
XRA reg	XOR AL,reg
ORA reg	OR AL,reg
CMP reg	CMP AL,reg
ADI data	ADD AL,data
ACI data	ADC AL,data
SUI data	SUB AL,data
SBI data	SBB AL,data

8080	8086
ANI data	AND AL,data
XRI data	XOR AL,data
ORI data	OR AL,data
CPI data	CMP AL,data
RLC	ROL AL
RRC	ROR AL
RAL	RCL AL
RAR	RCR AL
DAA	DAA
CMA	NOT AL
STC	STC
CMC	CMC
JMP addr	JMP addr*
CALL addr	CALL addr
RET	RET
RST n	CALL 8*n
JNZ addr	JNZ addr**
JZ addr	JZ addr
JNC addr	JNB addr
JC addr	JB addr
JPE addr	JPE addr
JPO addr	JPO addr
JP addr	JNS addr
JM addr	JS addr
CNZ addr	JZ next-inst CALL addr
CZ addr	JNZ next-inst CALL addr
CNC addr	JB next-inst CALL addr
CC addr	JNB next-inst CALL addr
CPE addr	JPO next-inst CALL addr
CPO addr	JPE next-inst CALL addr
CP addr	JS next-inst CALL addr
CM addr	JNS next-inst CALL addr
RNZ	JZ next-inst RET
RZ	JNZ next-inst RET
RNC	JB next-inst RET
RC	JNB next-inst RET
RPE	JPO next-inst RET
RPO	JPE next-inst RET
RP	JS next-inst RET
RM	JNS next-inst RET
OUT port	OUT port
IN port	IN port
DI	CLI
EI	STI
NOP	XCHG AX,AX
HLT	HLT

*Address on 8086 jumps and calls must be adjusted to be self-relative.

**Conditional jumps to a location out of the short self-relative range must be implemented by using a reversed-sense conditional jump around a normal jump to the location.

e.g., JNZ addr   becomes: JZnext-inst   JMP addr

TABLE 4-4. OPERATION INDEX

CHAPTER 5

Device Specifications

- **MCS-86™**
- **MCS-85™***
- **Peripherals****
- **Static RAMs*****
- **ROMs/EPROMs*****

*For complete specifications refer to the Intel MCS-85 User's Manual.

**For complete specifications refer to the Intel Peripheral Design Handbook.

***For complete specifications refer to the 1978 Intel Data Catalog.

intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8086
16-BIT HMOS MICROPROCESSOR

- **Direct Addressing Capability to 1 MByte of Memory**
- **Assembly Language Compatible with 8080/8085**
- **14 Word, By 16-Bit Register Set with Symmetrical Operations**
- **24 Operand Addressing Modes**
- **Bit, Byte, Word, and Block Operations**
- **8-and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide**
- **5 MHz Clock Rate**
- **MULTIBUS™ Compatible System Interface**

The Intel® 8086 is a new generation, high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The processor has attributes of both 8- and 16-bit microprocessors. It addresses memory as a sequence of 8-bit bytes, but has a 16-bit wide physical path to memory for high performance.

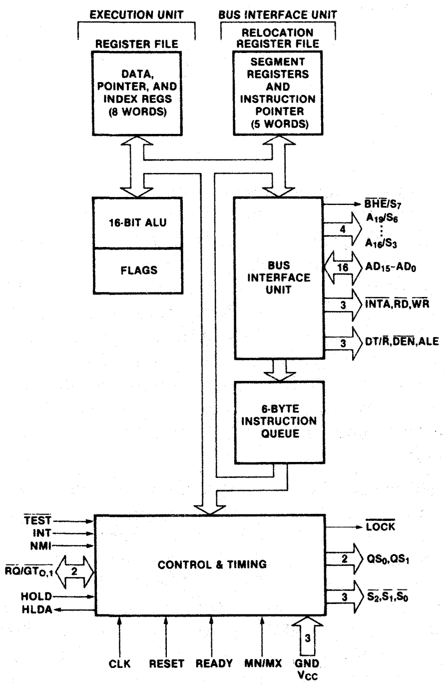


Figure 1. 8086 CPU Functional Block Diagram

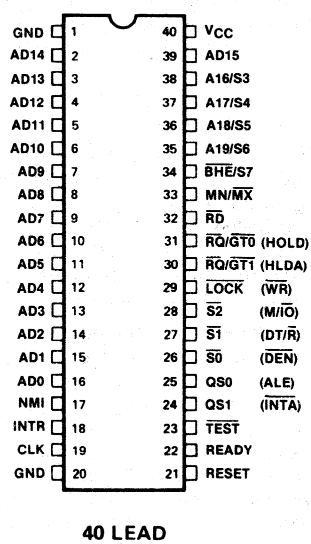


Figure 2. 8086 Pin Diagram

FUNCTIONAL DESCRIPTION

GENERAL OPERATION

The internal functions of the 8086 processor are partitioned logically into two processing units. The first is the Bus Interface Unit (BIU) and the second is the Execution Unit (EU) as shown in the block diagram of Figure 1.

These units can interact directly but for the most part perform as separate asynchronous operational processors. The bus interface unit provides the functions related to instruction fetching and queuing, operand fetch and store, and address relocation. This unit also provides the basic bus control. The overlap of instruction pre-fetching provided by this unit serves to increase processor performance through improved bus bandwidth utilization. Up to 6 bytes of the instruction stream can be queued while waiting for decoding and execution.

The instruction stream queuing mechanism allows the BIU to keep the memory utilized very efficiently. Whenever there is space for at least 2 bytes in the queue, the BIU will attempt a word fetch memory cycle. This greatly reduces "dead time" on the memory bus. The queue acts as a First-In-First-Out (FIFO) buffer, from which the EU extracts instruction bytes as required. If the queue is empty (following a branch instruction, for example), the first byte into the queue immediately becomes available to the EU.

The execution unit receives pre-fetched instructions from the BIU queue and provides un-relocated operand addresses to the BIU. Memory operands are passed through the BIU for processing by the EU, which passes results to the BIU for storage. See the Instruction Set description for further register set and architectural descriptions.

MEMORY ORGANIZATION

The processor provides a 20-bit address to memory which locates the byte being referenced. The memory is logically organized as a linear array of 1 million bytes, addressed as 00000(H) to FFFFF(H). The memory can be further logically divided into code, data, alternate data, and stack segments of up to 64K bytes each, with each segment falling on 16-byte boundaries. (See Figure 3a.)

Word (16-bit) operands can be located on even or odd address boundaries and are thus not constrained to even boundaries as is the case in many 16-bit computers. For address and data operands, the least significant byte of the word is stored in the lower valued address location and the most significant byte in the next higher address location. The BIU automatically performs the proper number of memory accesses, one if the word operand is on an even byte boundary and two if it is on an odd byte boundary. Except for the performance penalty, this double access is transparent to the software. This performance penalty does not occur for instruction fetches, only word operands.

Physically, the memory is organized as a high bank (D_{15}–D_8) and a low bank (D_7–D_0) of 512K 8-bit bytes addressed in parallel by the processor's address lines $A_{19}-A_1$. Byte data with even addresses is transferred on the D_7–D_0 bus lines while odd addressed byte data (A_0 HIGH) is transferred on the D_{15}–D_8 bus lines. The processor provides two enable signals, $\overline{BHE}$ and A_0, to selectively allow reading from or writing into either an odd byte location, even byte location, or both. The instruction stream is fetched from memory as words and is addressed internally by the processor to the byte level as necessary.

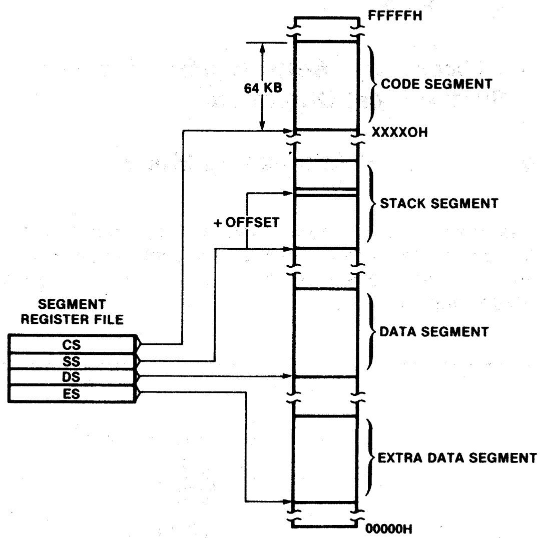


Figure 3a. Memory Organization

In referencing word data the BIU requires one or two memory cycles depending on whether or not the starting byte of the word is on an even or odd address, respectively. Consequently, in referencing word operands performance can be optimized by locating data on even address boundaries. This is an especially useful technique for using the stack, since odd address references to the stack may adversely affect the context switching time for interrupt processing or task multiplexing.

Certain locations in memory are reserved for specific CPU operations (see Figure 3b.) Locations from address FFFF0H through FFFFFH are reserved for operations including a jump to the initial program loading routine. Following RESET, the CPU will always begin execution at location FFFF0H where the jump must be. Locations 00000H through 003FFH are reserved for interrupt operations. Each of the 256 possible interrupt types has its service routine pointed to by a 4-byte pointer element consisting of a 16-bit segment address and a 16-bit offset address. The pointer elements are assumed to have been stored at the respective places in reserved memory prior to occurrence of interrupts.

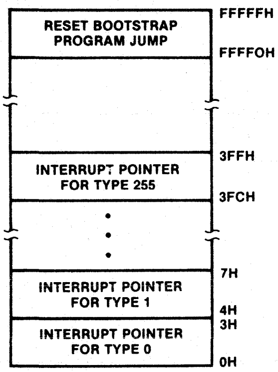


Figure 3b. Reserved Memory Locations

MINIMUM AND MAXIMUM MODES

The requirements for supporting minimum and maximum 8086 systems are sufficiently different that they cannot be done efficiently with 40 uniquely defined pins. Consequently, the 8086 is equipped with a strap pin (MN/$\overline{MX}$) which defines the system configuration. The definition of a certain subset of the pins changes dependent on the condition of the strap pin. When MN/$\overline{MX}$ pin is strapped to GND, the 8086 treats pins 24 through 31 in maximum mode. An 8288 bus controller interprets status information coded into $\overline{S_0}$,$\overline{S_1}$,$\overline{S_2}$ to generate bus timing and control signals compatible with the MULTIBUS™. When the MN/$\overline{MX}$ pin is strapped to V_{CC}, the 8086 generates bus control signals itself on pins 24 through 31, as shown in parentheses in Figure 2. Examples of minimum mode and maximum mode systems are shown in Figure 4.

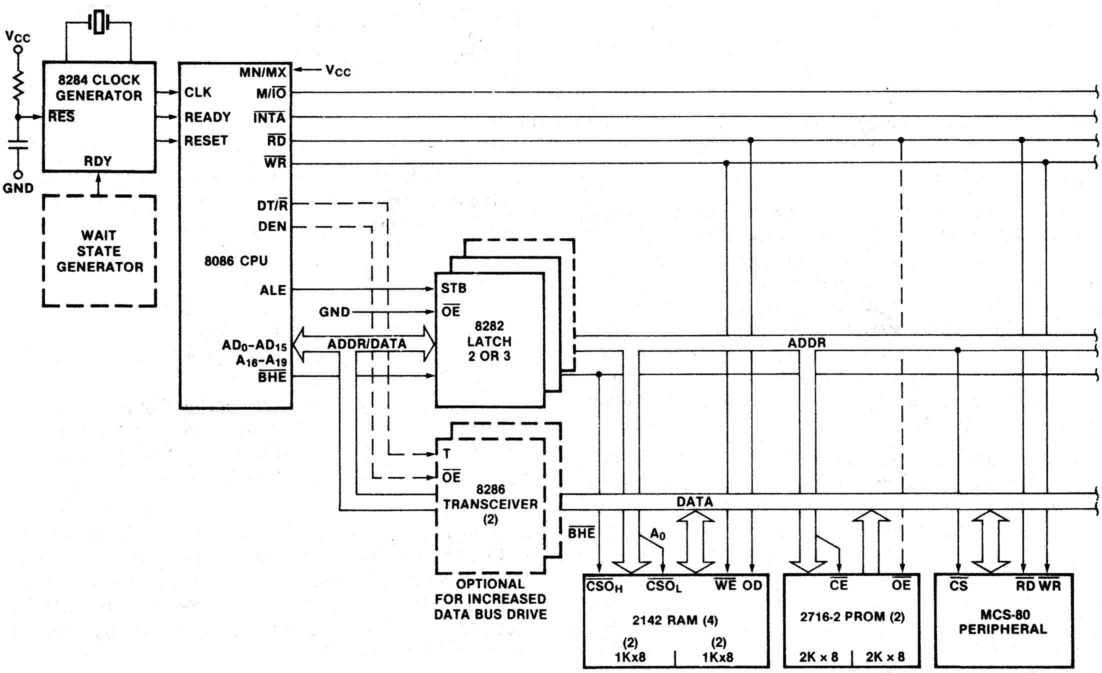


Figure 4a. Minimum Mode 8086 Typical System Configuration

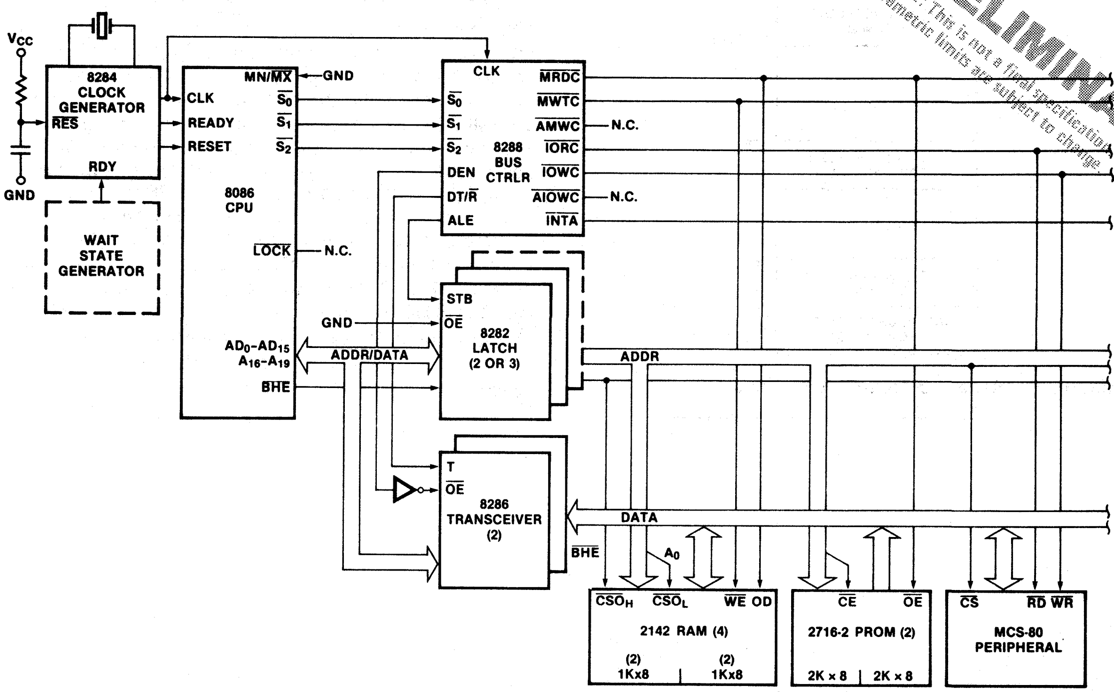


Figure 4b. Maximum Mode 8086 Typical System Configuration

BUS OPERATION

The 8086 has a combined address and data bus commonly referred to as a time multiplexed bus. This technique provides the most efficient use of pins on the processor while permitting the use of a standard 40-lead package. This "local bus" can be buffered directly and used throughout the system with address latching provided on memory and I/O modules. In addition, the bus can also be demultiplexed at the processor with a single set of address latches if a standard non-multiplexed bus is desired for the system.

Each processor bus cycle consists of at least four CLK cycles. These are referred to as T_1, T_2, T_3 and T_4 (see Figure 5). The address is emitted from the processor during T_1 and data transfer occurs on the bus during T_3 and T_4. T_2 is used primarily for changing the direction of the bus during read operations. In the event that a "NOT READY" indication is given by the addressed device, "Wait" states (T_W) are inserted between T_3 and T_4. Each inserted "Wait" state is of the same duration as a CLK cycle. Periods can occur between 8086 driven bus cycles. These are referred to as "Idle" states (T_I) or inactive CLK cycles. The processor uses these cycles for internal housekeeping.

During T_1 of any bus cycle the ALE (Address Latch Enable) signal is emitted (by either the processor or the 8288 bus controller, depending on the MN/$\overline{MX}$ strap). At the trailing edge of this pulse, a valid address and certain status information for the cycle may be latched.

Status bits $\overline{S_0}$, $\overline{S_1}$, and $\overline{S_2}$ are used, in maximum mode, by the bus controller to identify the type of bus transaction according to the following table:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt
1 (HIGH)	0	0	Instruction Fetch
1	0	1	Read Data from Memory
1	1	0	Write Data to Memory
1	1	1	Passive (no bus cycle)

Status bits S_3 through S_7 are multiplexed with high-order address bits and the $\overline{BHE}$ signal, and are therefore valid during T_2 through T_4. S_3 and S_4 indicate which segment register (see Instruction Set description) was used for this bus cycle in forming the address, according to the following table:

S_4	S_3	
0 (LOW)	0	Alternate Data (extra segment)
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_5 is a reflection of the PSW interrupt enable bit. $S_6 = 0$ and S_7 is a spare status bit.

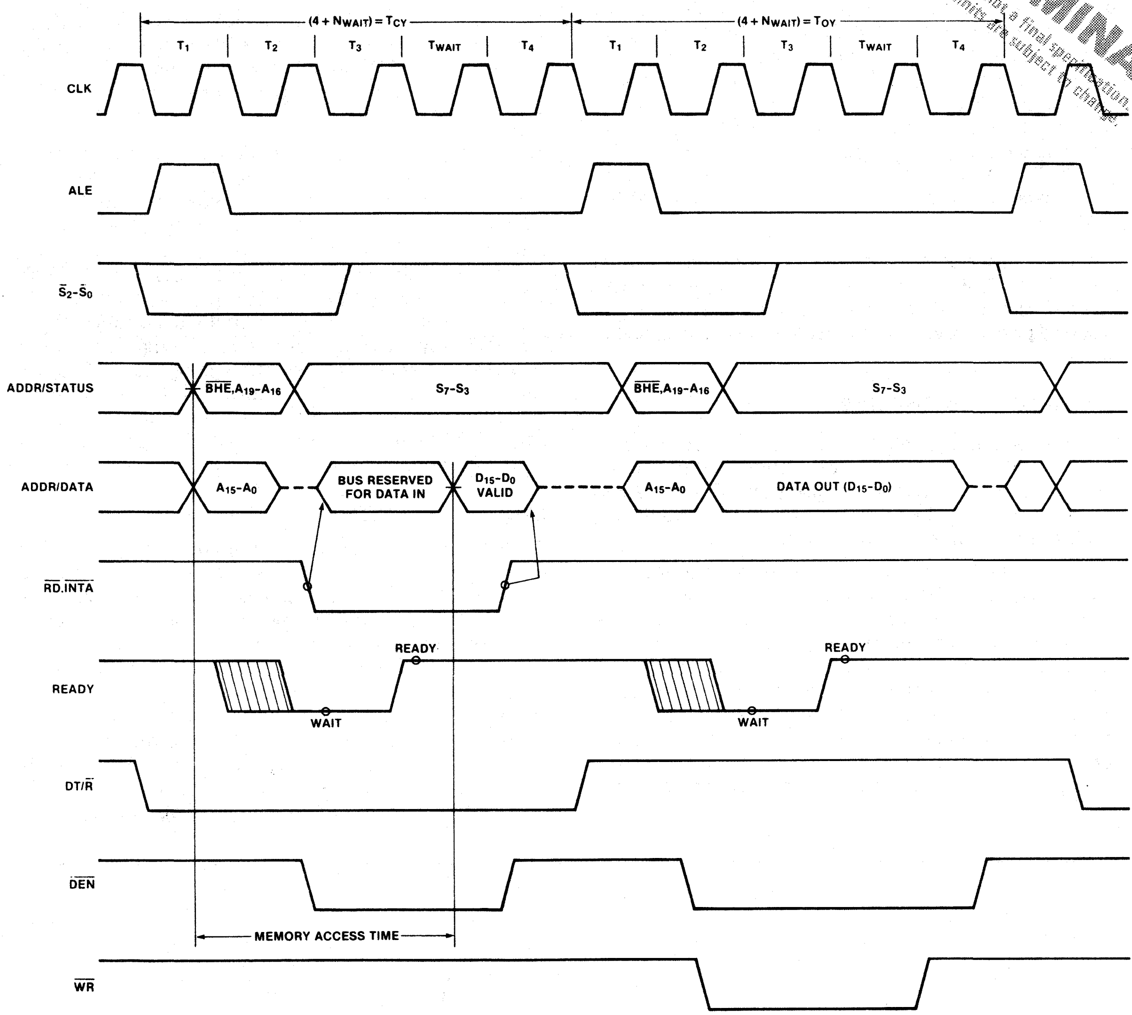


Figure 5. Basic System Timing

I/O ADDRESSING

In the 8086, I/O operations can address up to a maximum of 64K I/O byte registers or 32K I/O word registers. The I/O address appears in the same format as the memory address on bus lines A_{15}–A_0. The address lines A_{19}–A_{16} are zero in I/O operations. The variable I/O instructions which use register DX as a pointer have full address capability while the direct I/O instructions directly address one or two of the 256 I/O byte locations in page 0 of the I/O address space.

I/O ports are addressed in the same manner as memory locations. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8. Care must be taken to assure that each register within an 8-bit peripheral located on the lower portion of the bus be addressed as even.

EXTERNAL INTERFACE

PROCESSOR RESET AND INITIALIZATION

Processor initialization or start up is accomplished with activation (HIGH) of the RESET pin. The 8086 RESET is required to be HIGH for greater than 4 CLK cycles. The

8086 will terminate operations on the high-going edge of RESET and will remain dormant as long as RESET is HIGH. The low-going transition of RESET triggers an internal reset sequence for approximately 10 CLK cycles. After this interval the 8086 operates normally beginning with the instruction in absolute location FFFF0H (see Figure 3b). The details of this operation are specified in the Instruction Set description of the MCS-86 Users' Manual. The RESET input is internally synchronized to the processor clock. At initialization the HIGH-to-LOW transition of RESET must occur no sooner than 50 μs after power-up, to allow complete initialization of the 8086.

If INTR is asserted sooner than 9 CLK cycles after the end of RESET, the processor may execute one instruction before responding to the interrupt. NMI may not be asserted prior to the 2nd CLK cycle following the end of RESET.

INTERRUPT OPERATIONS

Interrupt operations fall into two classes; software or hardware initiated. The software initiated interrupts and software aspects of hardware interrupts are specified in the Instruction Set description. Hardware interrupts can be classified as non-maskable or maskable.

Interrupts result in a transfer of control to a new program location. A 256-element table containing address pointers to the interrupt service program locations resides in absolute locations 0 through 3FFH (see Figure 3b), which are reserved for this purpose. Each element in the table is 4 bytes in size and corresponds to an interrupt "type". An interrupting device supplies an 8-bit type number, during the interrupt acknowledge sequence, which is used to "vector" through the appropriate element to the new interrupt service program location.

NON-MASKABLE INTERRUPT (NMI)

The processor provides a single non-maskable interrupt pin (NMI) which has higher priority than the maskable interrupt request pin (INTR). A typical use would be to activate a power failure routine. The NMI is edge-triggered on a LOW-to-HIGH transition. The activation of this pin causes a type 2 interrupt. (See Instruction Set description.)

NMI is required to have a duration in the HIGH state of greater than two CLK cycles, but is not required to be synchronized to the clock. Any high-going transition of NMI is latched on-chip and will be serviced at the end of the current instruction or between whole moves of a block-type instruction. Worst case response to NMI would be for multiply, divide, and variable shift instructions. There is no specification on the occurrence of the low-going edge; it may occur before, during, or after the servicing of NMI. Another high-going edge triggers another response if it occurs after the start of the NMI procedure. The signal must be free of logical spikes in general and be free of bounces on the low-going edge to avoid triggering extraneous responses.

MASKABLE INTERRUPT (INTR)

The 8086 provides a single interrupt request input (INTR) which can be masked internally by software with the resetting of the interrupt enable FLAG status bit. The interrupt request signal is level triggered. It is internally synchronized during each clock cycle on the high-going edge of CLK. To be responded to, INTR must be present (HIGH) during the clock period preceding the end of the current instruction or the end of a whole move for a block-type instruction. During the interrupt response sequence further interrupts are disabled. The enable bit is reset as part of the response to any interrupt (INTR, NMI, software interrupt or single-step), although the

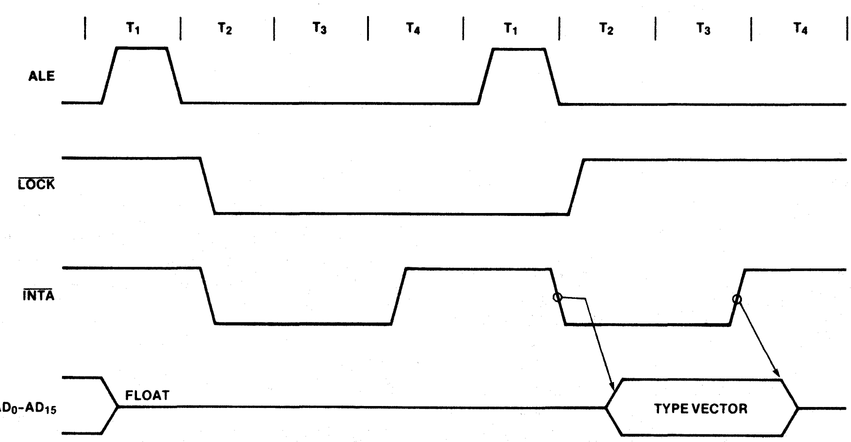


Figure 6. Interrupt Acknowledge Sequence

FLAGS register which is automatically pushed onto the stack reflects the state of the processor prior to the interrupt. Until the old FLAGS register is restored the enable bit will be zero unless specifically set by an instruction.

During the response sequence (figure 6) the processor executes two successive (back-to-back) interrupt acknowledge cycles. The 8086 emits the LOCK signal from T_2 of the first bus cycle until T_2 of the second. A local bus "hold" request will not be honored until the end of the second bus cycle. In the second bus cycle a byte is fetched from the external interrupt system (e.g., 8259A PIC) which identifies the source (type) of the interrupt. This byte is multiplied by four and used as a pointer into the interrupt vector lookup table. An INTR signal left HIGH will be continually responded to within the limitations of the enable bit and sample period. The INTERRUPT RETURN instruction includes a FLAGS pop which returns the status of the original interrupt enable bit when it restores the FLAGS.

HALT

When a software "HALT" instruction is executed the processor indicates that it is entering the "HALT" state in one of two ways depending upon which mode is strapped. In minimum mode, the processor issues one ALE with no qualifying bus control signals. In Maximum Mode, the processor issues appropriate HALT status on $\overline{S_2}\overline{S_1}\overline{S_0}$ and the 8288 bus controller issues one ALE. The 8086 will not leave the "HALT" state when a local bus "hold" is entered while in "HALT". In this case, the processor reissues the HALT indicator. An interrupt request or RESET will force the 8086 out of the "HALT" state.

READ/MODIFY/WRITE (SEMAPHORE) OPERATIONS VIA LOCK

The $\overline{\text{LOCK}}$ status information is provided by the processor when directly consecutive bus cycles are required during the execution of an instruction. This provides the processor with the capability of performing read/modify/write operations on memory (via the Exchange Register With Memory instruction, for example) without the possibility of another system bus master receiving intervening memory cycles. This is useful in multiprocessor system configurations to accomplish "test and set lock" operations. The $\overline{\text{LOCK}}$ signal is activated (forced LOW) in the clock cycle following the one in which the software "LOCK" prefix instruction is decoded by the EU. It is deactivated at the end of the last bus cycle of the instruction following the "LOCK" prefix instruction. While $\overline{\text{LOCK}}$ is active all interrupts are masked and a request on a RQ/$\overline{\text{GT}}$ pin will be recorded and then honored at the end of the $\overline{\text{LOCK}}$.

EXTERNAL SYNCHRONIZATION VIA TEST

As an alternative to the interrupts and general I/O capabilities, the 8086 provides a single software-testable input known as the $\overline{\text{TEST}}$ signal. At any time the program may execute a WAIT instruction. If at that time the $\overline{\text{TEST}}$ signal is inactive (HIGH), program execution becomes suspended while the processor waits for $\overline{\text{TEST}}$ to become active. It must remain active for at least 5 CLK cycles. The WAIT instruction is re-executed repeatedly until that time. This activity does not consume bus cycles. The processor remains in an idle state while waiting. All 8086 drivers go to 3-state OFF if bus "Hold"is entered. If interrupts are enabled, they may occur while the processor is waiting. When this occurs the processor fetches the WAIT instruction one extra time, processes the interrupt, and then re-fetches and re-executes the WAIT instruction upon returning from the interrupt.

8086 COMPARED WITH 8080/8085

While the 8086 has new instruction coding patterns to allow for the greatly expanded capabilities, all functions of the 8080/8085 may be performed by the 8086 with identical program semantics to their 8080/8085 versions. For every 8080/8085 instruction there is a corresponding 8086 instruction (or, in rare cases, a short sequence of instructions). Virtually all 8086 data manipulation instructions may be specified to operate on either the full set of 16-bit registers or on an 8-bit subset of them which corresponds to the 8080 register set. This relationship is shown in Figure 7 where the shaded registers (names in parentheses) represent the 8080 register set.

BASIC SYSTEM TIMING

Typical system configurations for the processor operating in minimum mode and in maximum mode are shown in Figures 4a and 4b, respectively. In minimum mode, the MN/$\overline{\text{MX}}$ pin is strapped to V_{CC} and the processor emits bus control signals in a manner similar to the 8085. In maximum mode, the MN/$\overline{\text{MX}}$ pin is strapped to V_{SS} and the processor emits coded status information which the 8288 bus controller uses to generate MULTIBUS™ compatible bus control signals. Figure 5 illustrates the signal timing relationships.

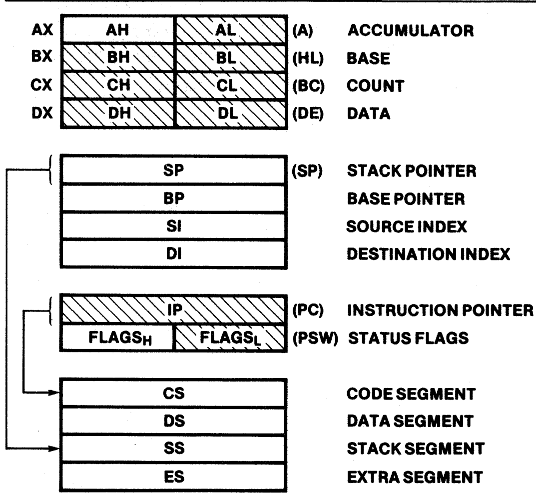


Figure 7. 8086 Register Model; (8080 Registers Shaded)

SYSTEM TIMING — MINIMUM SYSTEM

The read cycle begins in T_1 with the assertion of the Address Latch Enable (ALE) signal. The trailing (low-going) edge of this signal is used to latch the address information, which is valid on the local bus at this time, into the 8282/8283 latch. The $\overline{BHE}$ and A_0 signals address the low, high, or both bytes. From T_1 to T_4 the M/$\overline{IO}$ signal indicates a memory or I/O operation. At T_2 the address is removed from the local bus and the bus goes to a high impedance state. The read control signal is also asserted at T_2. The read ($\overline{RD}$) signal causes the addressed device to enable its data bus drivers to the local bus. Some time later valid data will be available on the bus and the addressed device will drive the READY line HIGH. When the processor returns the read signal to a HIGH level, the addressed device will again 3-state its bus drivers. If a transceiver (8286/8287) is required to buffer the 8086 local bus, signals DT/$\overline{R}$ and $\overline{DEN}$ are provided by the 8086.

A write cycle also begins with the assertion of ALE and the emission of the address. The M/$\overline{IO}$ signal is again asserted to indicate a memory or I/O write operation. In the T_2 immediately following the address emission the processor emits the data to be written into the addressed location. This data remains valid until the middle of T_4. During T_2, T_3, and T_W the processor asserts the write control signal. The write ($\overline{WR}$) signal becomes active at the beginning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float.

The $\overline{BHE}$ and A_0 signals are used to select the proper byte(s) of the memory/IO word to be read or written according to the following table:

$\overline{BHE}$	A0	
0	0	Whole word
0	1	Upper byte from/ to odd address
1	0	Lower byte from/ to even address
1	1	None

I/O ports are addressed in the same manner as memory location. Even addressed bytes are transferred on the D_7–D_0 bus lines and odd addressed bytes on D_{15}–D_8.

The basic difference between the interrupt acknowledge cycle and a read cycle is that the interrupt acknowledge signal ($\overline{INTA}$) is asserted in place of the read ($\overline{RD}$) signal and the address bus is floated. (See Figure 6.) In the second of two successive INTA cycles, a byte of information is read from bus lines D_7–D_0 as supplied by the interrupt system logic (i.e., 8259A Priority Interrupt Controller). This byte identifies the source (type) of the interrupt. It is multiplied by four and used as a pointer into an interrupt vector lookup table, as described earlier.

BUS TIMING — MEDIUM COMPLEXITY SYSTEMS

For medium complexity systems the MN/$\overline{MX}$ pin is connected to V_{SS} and the 8288 Bus Controller is added to the system as well as an 8282/8283 latch for latching the system address, and a 8286/8287 transceiver to allow for bus loading greater than the 8086 is capable of handling. Signals ALE, DEN, and DT/$\overline{R}$ are generated by the 8288 instead of the processor in this configuration although their timing remains relatively the same. The 8086 status outputs ($\overline{S}_2$, $\overline{S}_1$, and $\overline{S}_0$) provide type-of-cycle information and become 8288 inputs. This bus cycle information specifies read (code, data, or I/O), write (data or I/O), interrupt acknowledge, or software halt. The 8288 thus issues control signals specifying memory read or write, I/O read or write, or interrupt acknowledge. The 8288 provides two types of write strobes, normal and advanced, to be applied as required. The normal write strobes have data valid at the leading edge of write. The advanced write strobes have the same timing as read strobes, and hence data isn't valid at the leading edge of write. The 8286/8287 transceiver receives the usual T and $\overline{OE}$ inputs from the 8288's DT/$\overline{R}$ and DEN.

The pointer into the interrupt vector table, which is passed during the second INTA cycle, can derive from an 8259A located on either the local bus or the system bus. If the master 8259A Priority Interrupt Controller is positioned on the local bus, a TTL gate is required to disable the 8286/8287 transceiver when reading from the master 8259A during the interrupt acknowledge sequence and software "poll".

8086 FUNCTIONAL PIN DEFINITION

The following pin function descriptions are for 8086 systems in either minimum or maximum mode. The "Local Bus" in these descriptions is the direct multiplexed bus interface connection to the 8086 (without regard to additional bus buffers).

AD_{15}–AD_0 (INPUT/OUTPUT 3-STATE)

These lines constitute the time multiplexed memory/IO address (T_1) and data (T_2, T_3, T_W, T_4) bus. A_0 is analogous to $\overline{BHE}$ for the lower byte of the data bus, pins D_7–D_0. It is LOW during T_1 when a byte is to be transferred on the lower portion of the bus in memory or I/O operations. Eight-bit oriented devices tied to the lower half would normally use A_0 to condition chip select functions. (See table on page 8.) These lines are active HIGH and float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".

A_{19}/S_6, A_{18}/S_5, A_{17}/S_4, A_{16}/S_3 (OUTPUT 3-STATE)

During T_1 these are the four most significant address lines for memory operations. During I/O operations these lines are LOW. During memory and I/O operations, status information is available on these lines during T_2, T_3, T_W, and T_4. The status of the interrupt enable FLAG bit (S_5) is updated at the beginning of each CLK cycle. A_{17}/S_4 and A_{16}/S_3 are encoded as follows:

A_{17}/S_4	A_{16}/S_3	
0 (LOW)	0	Alternate Data
0	1	Stack
1 (HIGH)	0	Code or None
1	1	Data

S_6 is 0 (LOW

This information indicates which relocation register is presently being used for data accessing.

These lines float to 3-state OFF during interrupt acknowledge and local bus "hold acknowledge".

$\overline{BHE}/S_7$ (OUTPUT 3-STATE)

During T_1 the bus high enable signal ($\overline{BHE}$) should be used to enable data onto the most significant half of the data bus, pins D_{15}–D_8. Eight-bit oriented devices tied to the upper half of the bus would normally use $\overline{BHE}$ to condition chip select functions. $\overline{BHE}$ is LOW during T_1 for read, write, and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. (See table on page 8.) The S_7 status information is available during T_2, T_3, and T_4. The signal is active LOW, and floats to 3-state OFF in "hold". It is LOW during T_1 for the first interrupt acknowledge cycle.

$\overline{RD}$ (OUTPUT 3-STATE)

Read strobe indicates that the processor is performing a memory or I/O read cycle, depending on the state of the S_2 pin. This signal is used to read devices which reside on the 8086 local bus. $\overline{RD}$ is active LOW during T_2, T_3 and T_W or any read cycle, and is guaranteed to remain HIGH in T_2 until the 8086 local bus has floated.

This signal floats to 3-state OFF in "hold acknowledge".

READY (INPUT)

READY is the acknowledgement from the addressed memory or I/O device that it will complete the data transfer. The RDY signal from memory/IO is synchronized by the 8284 Clock Generator to form READY. This signal is active HIGH.

INTR (INPUT)

Interrupt request is a level triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt acknowledge operation. A subroutine is vectored to via an interrupt vector lookup table located in system memory. It can be internally masked by software resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.

$\overline{TEST}$ (INPUT)

The $\overline{TEST}$ input is examined by the "Wait For Test" instruction. If the $\overline{TEST}$ input is LOW execution continues, otherwise the processor waits in an "Idle" state. This input is synchronized internally during each clock cycle on the leading edge of CLK.

NMI (INPUT)

Non-maskable interrupt is an edge triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector lookup table located in system memory. NMI is not maskable internally by software. A transition from a LOW to HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.

RESET (INPUT)

RESET causes the processor to immediately terminate its present activity. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the Instruction Set description, when RESET returns LOW. RESET is internally synchronized.

CLK (INPUT)

The clock provides the basic timing for the processor and bus controller. It is asymmetric with a 33% duty cycle to provide optimized internal timing.

V_{CC}

V_{CC} is the $+5V \pm 10\%$ power supply pin.

GND

GND are the ground pins

The following pin function descriptions are for the 8086 minimum mode (i.e., MN/$\overline{\text{MX}}$ = V_{CC}). Only the pin functions which are unique to minimum mode are described; all other pin functions are as described above.

M/$\overline{\text{IO}}$ (OUTPUT 3-STATE)

This status line is logically equivalent to S_2 in the maximum mode. It is used to distinguish a memory access from an I/O access. M/$\overline{\text{IO}}$ becomes valid in the T_4 preceding a bus cycle and remains valid until the final T_4 of the cycle (M = HIGH, IO = LOW). M/$\overline{\text{IO}}$ floats to 3-state OFF in local bus "hold acknowledge".

$\overline{\text{WR}}$ (OUTPUT 3-STATE)

Write strobe indicates that the processor is performing a write memory or write I/O cycle, depending on the state of the M/$\overline{\text{IO}}$ signal. $\overline{\text{WR}}$ is active for T_2, T_3 and T_W of any write cycle. It is active LOW, and floats to 3-state OFF in local bus "hold acknowledge".

$\overline{\text{INTA}}$ (OUTPUT 3-STATE)

$\overline{\text{INTA}}$ is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T_2, T_3 and T_W of each interrupt acknowledge cycle. $\overline{\text{INTA}}$ floats to 3-state OFF in "hold acknowledge".

ALE (OUTPUT)

Address latch enable is provided by the processor to latch the address into the 8282/8283 address latch. It is a HIGH pulse active during T_1 of any bus cycle. Note that ALE is never floated.

DT/$\overline{\text{R}}$ (OUTPUT 3-STATE)

Data transmit/receive is needed in minimum system that desires to use an 8286/8287 data bus transceiver. It is used to control the direction of data flow through the transceiver. Logically DT/$\overline{\text{R}}$ is equivalent to $\overline{S_1}$ in the maximum mode, and its timing is the same as for M/$\overline{\text{IO}}$.(T = HIGH, R = LOW.) This signal floats to 3-state OFF in local bus "hold acknowledge".

$\overline{\text{DEN}}$ (OUTPUT 3-STATE)

Data enable is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. $\overline{\text{DEN}}$ is active LOW during each memory and I/O access and for INTA cycles. For a read or INTA cycle it is active from the middle of T_2 until the middle of T_4, while for a write cycle it is active from the beginning of T_2 until the middle of T_4. $\overline{\text{DEN}}$ floats to 3-state OFF in local bus "hold acknowledge".

HOLD (INPUT), HLDA (OUTPUT)

HOLD indicates that another master is requesting a local bus "hold". To be acknowledged, HOLD must be active HIGH. The processor receiving the "hold" request will issue HLDA (HIGH) as an acknowledgement in the middle of T_4 or T_1. Simultaneous with the issuance of HLDA the processor will float the local bus and control lines. After HOLD is detected as being LOW, the processor will LOWer HLDA, and when the processor needs to run another cycle, it will again drive the local bus and control lines. (See Figure 13.)

The following pin function descriptions are for the 8086/8288 system in maximum mode (i.e., MN/$\overline{MX}$ = V_{SS}). Only the pin functions which are unique to maximum mode are described; all other pin functions are as described above.

$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$ (OUTPUT 3-STATE)

These status lines are encoded as follows:

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	
0 (LOW)	0	0	Interrupt Acknowledge
0	0	1	Read I/O Port
0	1	0	Write I/O Port
0	0	0	Halt
1 (HIGH)	0	0	Code Access
1	0	1	Read Memory
1	1	0	Write Memory
1	1	1	Passive

Status is active during T_4, T_1, and T_2 and is returned to the passive state (1,1,1) during T_3 or during T_W when READY is HIGH. This status is used by the 8288 Bus Controller to generate all memory and I/O access control signals. Any change by $\overline{S_2}$, $\overline{S_1}$, or $\overline{S_0}$ during T_4 is used to indicate the beginning of a bus cycle, and the return to the passive state in T_3 or T_W is used to indicate the end of a bus cycle.

These signals float to 3-state OFF in "hold acknowledge".

$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ (INPUT/OUTPUT)

The request/grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional with $\overline{RQ}/\overline{GT_0}$ having higher priority than $\overline{RQ}/\overline{GT_1}$. $\overline{RQ}/\overline{GT}$ has an internal pull-up resistor so may be left unconnected. The request/grant sequence is as follows (see Figure 12):

1. A pulse of 1 CLK wide from another local bus master indicates a local bus request ("hold") to the 8086 (pulse 1).
2. During the CPU's next T_4 or T_1 a pulse 1 CLK wide, from the 8086 to the requesting master (pulse 2), indicates that the 8086 has allowed the local bus to float and that it will enter the "hold acknowledge" state at the next CLK. The CPU's bus interface unit is disconnected logically from the local bus during "hold acknowledge".
3. A pulse 1 CLK wide from the requesting master indicates to the 8086 (pulse 3) that the "hold" request is about to end and that the 8086 can reclaim the local bus at the next CLK. The CPU then enters T_4.

Each master-master exchange of the local bus is a sequence of 3 pulses. There must be one dead CLK cycle after each bus exchange. Pulses are active LOW.

$\overline{LOCK}$ (OUTPUT 3-STATE)

The $\overline{LOCK}$ output indicates that other system bus masters are not to gain control of the system bus while $\overline{LOCK}$ is active LOW. The $\overline{LOCK}$ signal is activated by the "LOCK" prefix instruction and remains active until the next instruction is extracted from the queue. This signal is active LOW, and floats to 3-state OFF in "hold acknowledge".

QS_1, QS_0 (OUTPUT)

QS_1 and QS_0 provide status to allow external tracking of the internal 8086 instruction queue.

QS_1	QS_0	
0 (LOW)	0	No Operation
0	1	First Byte of Op Code from Queue
1 (HIGH)	0	Empty the Queue
1	1	Subsequent Byte from Queue

The queue status is valid during the CLK cycle after which the queue operation is performed.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . 0 °C to 70 °C
Storage Temperature . . . . −65 °C to +150 °C
Voltage on Any Pin with
Respect to Ground . . . . −0.3 to +7V
Power Dissipation . . . . 2.5 Watt

******COMMENT:* Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS

8086: $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{IL}	Input Low Voltage	−0.5	+0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.5$	V	
V_{OL}	Output Low Voltage		0.45	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = 400\ \mu A$
I_{CC}	Power Supply Current		275	mA	
I_{LI}	Input Leakage Current		±10	μA	$V_{IN} = V_{CC}$
I_{LO}	Output Leakage Current		±10	μA	$0.45V \leq V_{OUT} \leq V_{CC}$
V_{CL}	Clock Input Low Voltage	−0.5	+0.6	V	
V_{CH}	Clock Input High Voltage	3.9	$V_{CC}+1.0$	V	
C_{IN}	Capacitance of Input Buffer (All input except AD_0-AD_{15}, RQ/GT)		10	pF	fc = 1 MHz
C_{IO}	Capacitance of I/O Buffer (AD_0-AD_{15}, RQ/GT)		20	pF	fc = 1 MHz

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

A.C. CHARACTERISTICS

8086: $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 10\%$

8086 MINIMUM COMPLEXITY SYSTEM (Figure 8) TIMING REQUIREMENTS

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	2000	ns	
TCLCH	CLK Low Time	(⅔TCLCL) – 15		ns	
TCHCL	CLK High Time	(⅓TCLCL) – 2		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		ns	
TCLDZ	Data In Hold Time	10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1,2)	50		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1,2)	0		ns	
TRHVCH	READY Setup Time into 8086	110		ns	
TCHRYX	READY Hold Time into 8086	30		ns	
TRYLCL	READY Inactive to CLK (See Note 3)	– 15		ns	
THVCH	Hold Setup Time	35		ns	
TIVCH	INTR, NMI, $\overline{TEST}$ Setup Time (See Note 2)	30		ns	

TIMING RESPONSES

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLAV	Address Valid Delay	15	110	ns	$C_L = 100$ pF
TCLAX	Address Hold Time	10		ns	$C_L = 20$ pF
TCLAZ	Address Float Delay	TCLAX	80	ns	
TLHLL	ALE Width	TCLCH – 20		ns	
TCLLH	ALE Active Delay		80	ns	
TCHLL	ALE Inactive Delay		85	ns	
TLLAZ	ALE Inactive to Address Float	TCHCL – 10		ns	
TCLDV	Data Valid Delay	15	110	ns	
TCHDZ	Data Float Delay	TCLAX	85	ns	
TWHDZ	Data Hold Time After $\overline{WR}$	TCLCH – 30		ns	
TCVCTV	Control Active Delay 1	10	110	ns	
TCHCTV	Control Active Delay 2	15	110	ns	
TCVCTX	Control Inactive Delay	10	110	ns	
TAZRL	Address Float to READ Active	0		ns	
TCLRL	$\overline{RD}$ Active Delay	10	165	ns	
TCLRH	$\overline{RD}$ Inactive Delay	10	150	ns	
TRHAV	$\overline{RD}$ Inactive to Next Address Active	TCLCL – 45		ns	

NOTES: 1. SIGNAL AT 8284 SHOWN FOR REFERENCE ONLY.
2. SETUP REQUIREMENT FOR ASYNCHRONOUS SIGNAL ONLY TO GUARANTEE RECOGNITION AT NEXT CLK.
3. APPLIES ONLY TO T2 STATE.

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

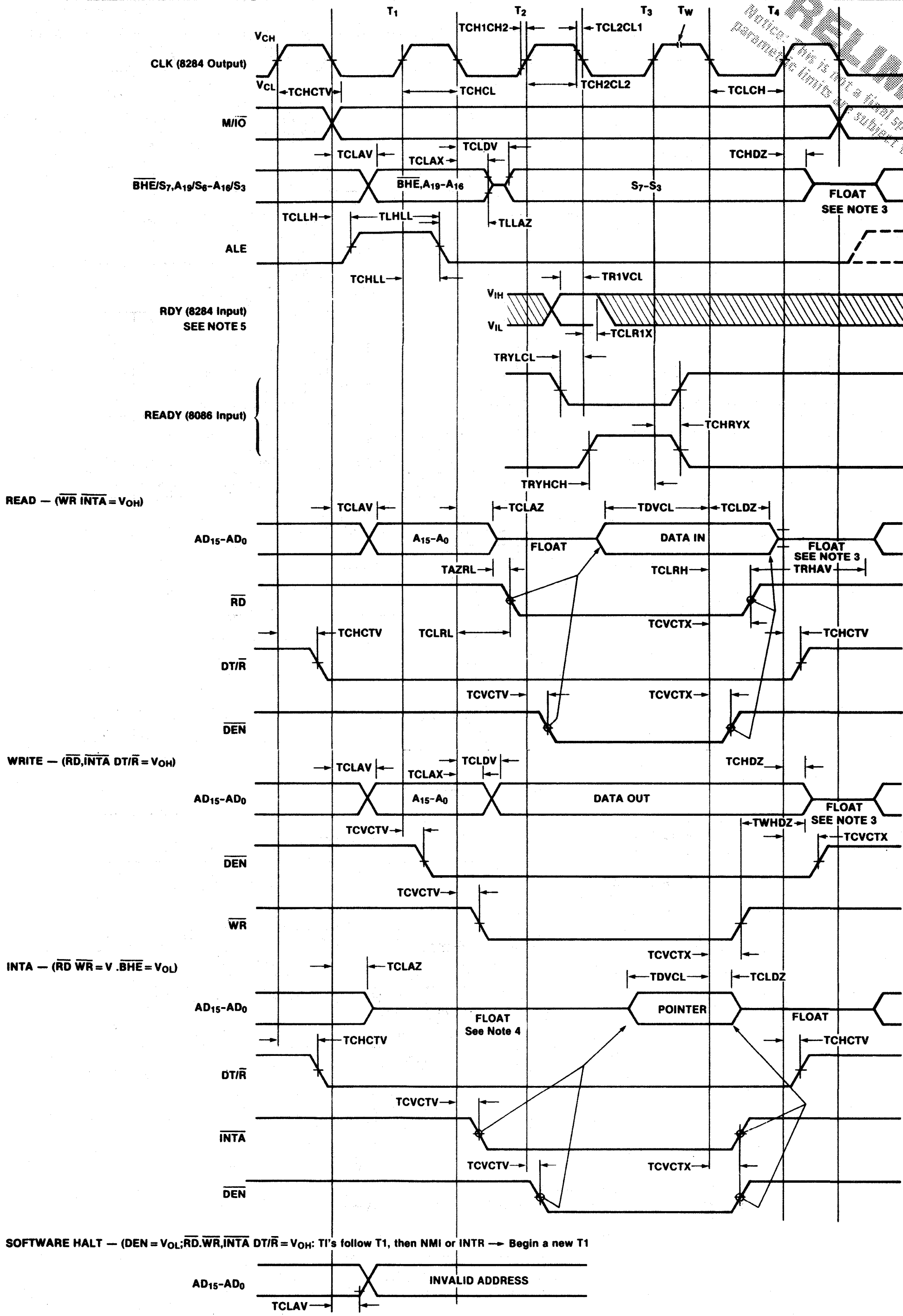
8086 MAX MODE SYSTEM (USING 8288 BUS CONTROLLER) (Figure 9) TIMING REQUIREMENTS

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLCL	CLK Cycle Period	200	2000	ns	
TCLCH	CLK Low Time	(⅔TCLCL) – 13		ns	
TCHCL	CLK High Time	(⅓TCLCL) – 2		ns	
TCH1CH2	CLK Rise Time		10	ns	From 1.0V to 3.5V
TCL2CL1	CLK Fall Time		10	ns	From 3.5V to 1.0V
TDVCL	Data In Setup Time	30		ns	
TCLDZ	Data In Hold Time	10		ns	
TR1VCL	RDY Setup Time into 8284 (See Notes 1,2)	50		ns	
TCLR1X	RDY Hold Time into 8284 (See Notes 1,2)	0		ns	
TRYHCH	READY Setup Time into 8086	110		ns	
TCHRYX	READY Hold Time into 8086	30		ns	
TRYLCL	READY Inactive to CLK (See Note 4)	– 15		ns	
TRYHSH	READY Active to Status Passive (See Note 3)		110	ns	
TINVCH	Setup Time for Recognition (INTR, NMI, $\overline{\text{TEST}}$)(See Note 2)	30		ns	
TGVCH	$\overline{\text{RQ}}/\overline{\text{GT}}$ Setup Time	35		ns	

TIMING RESPONSES

Symbol	Parameter	Min.	Max.	Units	Test Conditions
TCLML	Command Active Delay (See Note 1)	10	35	ns	$C_L = 80$ pF
TCLMH	Command Inactive Delay (See Note 1)	10	40	ns	
TCLHAV	HLDA Valid Delay (See Note 1)	10	160	ns	
TCHSV	Status Active Delay	10	110	ns	
TCLSH	Status Inactive Delay		130	ns	
TCLAV	Address Valid Delay	15	110	ns	
TCLAX	Address Hold Time	10		ns	
TCLAZ	Address Float Delay	TCLAX	80	ns	
TSVLH	Status Valid to ALE High (See Note 1)		15	ns	
TSVMCH	Status Valid to MCE High (See Note 1)		15	ns	
TCLLH	CLK Low to ALE Valid (See Note 1)		15	ns	
TCLMCH	CLK Low to MCE High (See Note 1)		15	ns	
TCHLL	ALE Inactive Delay (See Note 1)		15	ns	
TCLNCL	MCE Inactive Delay (See Note 1)		15	ns	
TCLDV	Data Valid Delay	15	110	ns	
TCHDZ	Data Float Delay	TCLAX	85	ns	
TCVNTV	Control Active Delay (See Note 1)	10	35	ns	
TCVNTX	Control Inactive Delay (See Note 1)	10	40	ns	
TAZRL	Address Float to Read Active	0		ns	
TCLRL	RD Active Delay	10	165	ns	
TCLRH	RD Inactive Delay	10	150	ns	
TRHAV	RD Inactive to Next Address Active	TCLCL – 45		ns	
TCHDTL	Direction Control Active Delay (SEE NOTE 1)		50	ns	
TCHDTH	Direction Control Inactive Delay (SEE NOTE 1)		30	ns	
TCVEV	Data Enable Active Delay (SEE NOTE 1)	5	45	ns	
TCVEX	Data Enable Inactive Delay (SEE NOTE 1)	10	45	ns	
TCLGL	GT Active Delay		85	ns	$C_L = 30$ pF
TCLGH	GT Inactive Delay		85	ns	$C_L = 30$ pF

NOTES: 1. SIGNAL AT 8284 OR 8288 SHOWN FOR REFERENCE ONLY.
2. SETUP REQUIREMENT FOR ASYNCHRONOUS SIGNAL ONLY TO GUARANTEE RECOGNITION AT NEXT CLK.
3. APPLIES ONLY TO T3 AND TW STATES.
4. APPLIES ONLY TO T2 STATE.



NOTES: 1. ALL SIGNALS SWITCH BETWEEN V_{OH} AND V_{OL} UNLESS OTHERWISE SPECIFIED.

2. RDY IS SAMPLED NEAR THE END OF T_2, T_3, T_W TO DETERMINE IF T_W MACHINES STATES ARE TO BE INSERTED.

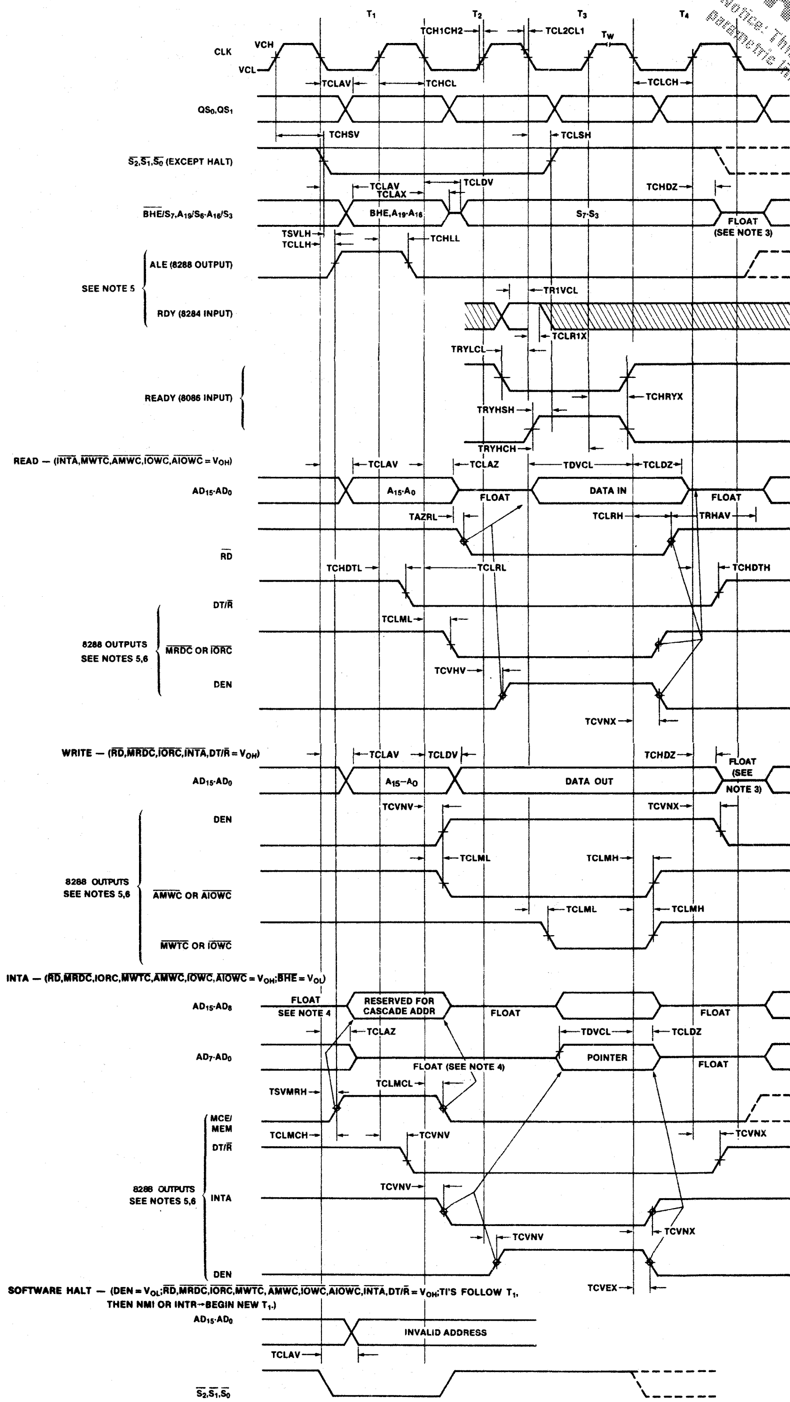
3. FOLLOWING A WRITE CYCLE DATA REMAINS VALID ON THE 8086 LOCAL BUS UNTIL A LOCAL BUS MASTER DECIDES TO RUN ANOTHER BUS CYCLE. THE LOCAL BUS IS FLOATED BY THE 8086 ONLY WHEN THE 8086 ENTERS A "HOLD ACKNOWLEDGE" STATE.

4. TWO INTA CYCLES RUN BACK-TO-BACK. THE 8086 LOCAL ADDR/DATA BUS IS FLOATING DURING THE SECOND INTA CYCLE.

5. SIGNALS AT 8284 OR 8288 ARE SHOWN FOR REFERENCE ONLY.

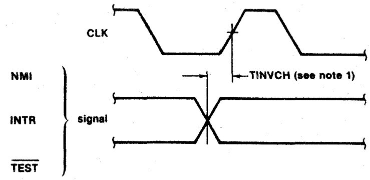
6. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

Figure 8. 8086 Bus Timing — Minimum Mode System



NOTES:
1. ALL SIGNALS SWITCH BETWEEN V_{OH} AND V_{OL} UNLESS OTHERWISE SPECIFIED.
2. RDY IS SAMPLED NEAR THE END OF T_2,T_3,T_W TO DETERMINE IF T_W MACHINES STATES ARE TO BE INSERTED.
3. FOLLOWING A WRITE CYCLE DATA REMAINS VALID ON THE 8086 LOCAL BUS UNTIL A LOCAL BUS MASTER DECIDES TO RUN ANOTHER BUS CYCLE. THE LOCAL BUS IS FLOATED BY THE 8086 ONLY WHEN THE 8086 ENTERS A "HOLD ACKNOWLEDGE" STATE.
4. TWO INTA CYCLES RUN BACK-TO-BACK. THE 8086 LOCAL ADDR/DATA BUS IS FLOATING DURING THE SECOND INTA CYCLE.
5. SIGNALS AT 8284 OR 8288 ARE SHOWN FOR REFERENCE ONLY.
6. THE ISSUANCE OF THE 8288 COMMAND AND CONTROL SIGNALS ($\overline{MRDC}$,$\overline{MWTC}$,$\overline{AMWC}$,$\overline{IORC}$,$\overline{IOWC}$,$\overline{AIOWC}$,$\overline{INTA}$, AND DEN) LAGS THE ACTIVE HIGH 8288 CEN.
7. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

Figure 9. 8086 Bus Timing — Maximum Mode System (Using 8288)



NOTE:
1. SETUP REQUIREMENTS FOR ASYNCHRONOUS SIGNALS ONLY TO GUARANTEE RECOGNITION AT NEXT CLK

Figure 10. Asynchronous Signal Recognition

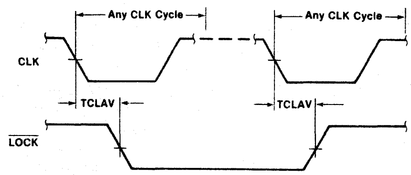
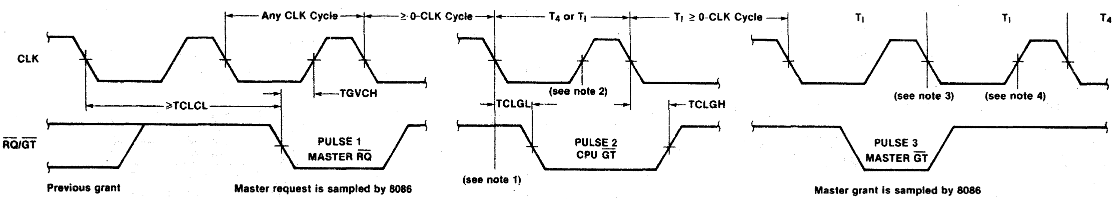
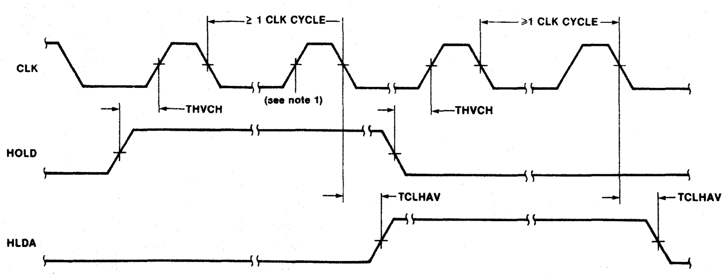


Figure 11. Bus Lock Signal Timing (Maximum Mode Only)



NOTES:
1. THE 8086 FLOATS $\overline{S_2}, \overline{S_1}, \overline{S_0}$ FROM 1.1.1 STATE ON THIS EDGE
2. THE 8086 FLOATS A_xD_x BUS $\overline{RD}$ AND $\overline{LOCK}$ ON THIS EDGE
3. THE OTHER MASTER FLOATS $\overline{S_2}, \overline{S_1}, \overline{S_0}$ FROM 1.1.1 STATE ON THIS EDGE
4. THE OTHER MASTER FLOATS A_xD_x $\overline{BUS}$, BHE, AND $\overline{LOCK}$ ON THIS EDGE

Figure 12. Request/Grant Sequence Timing (Maximum Mode Only)



NOTE:
1. BUS FLOATS ON THIS EDGE (SEE TCHDZ)

Figure 13. Hold/Hold Acknowledge Timing (Minimum Mode Only)

8086
INSTRUCTION SET SUMMARY

DATA TRANSFER

MOV = Move:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate to register	1 0 1 1 w reg	data	data if w=1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		
PUSH = Push:				
Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m		
Register	0 1 0 1 0 reg			
Segment register	0 0 0 reg 1 1 0			
POP = Pop:				
Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m		
Register	0 1 0 1 1 reg			
Segment register	0 0 0 reg 1 1 1			
XCHG = Exchange:				
Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m		
Register with accumulator	1 0 0 1 0 reg			
IN/INW = Input to AL/AX from:				
Fixed port	1 1 1 0 0 1 0 w	port		
Variable port	1 1 1 0 1 1 0 w			
OUT/OUTW = Output from AL/AX to:				
Fixed port	1 1 1 0 0 1 1 w	port		
Variable port	1 1 1 0 1 1 1 w			
XLAT=Translate byte to AL	1 1 0 1 0 1 1 1			
LEA=Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m		
LDS=Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m		
LES=Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m		
LAHF=Load AH with flags	1 0 0 1 1 1 1 1			
SAHF=Store AH into flags	1 0 0 1 1 1 1 0			
PUSHF=Push flags	1 0 0 1 1 1 0 0			
POPF=Pop flags	1 0 0 1 1 1 0 1			

ARITHMETIC

ADD = Add:				
Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	
ADC = Add with carry:				
Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	
INC = Increment:				
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m		
Register	0 1 0 0 0 reg			
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1			
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1			
SUB = Subtract:				
Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	
SBB = Subtract with borrow				
Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

DEC = Decrement:	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG=Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		
CMP = Compare:				
Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS=ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS=Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL=Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL=Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM=ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV=Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV=Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD=ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW=Convert byte to word	1 0 0 1 1 0 0 0			
CWD=Convert word to double word	1 0 0 1 1 0 0 1			

LOGIC

NOT=Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL=Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR=Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR=Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL=Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR=Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL=Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR=Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		
AND = And:				
Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	
TEST = And function to flags, no result:				
Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	
OR = Or:				
Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	
XOR = Exclusive or:				
Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	

STRING MANIPULATION

REP=Repeat	1 1 1 1 0 0 1 z
MOVB/MOVW=Move byte/word	1 0 1 0 0 1 0 w
CMPB/CMPW=Compare byte/word	1 0 1 0 0 1 1 w
SCAB/SCAW=Scan byte/word	1 0 1 0 1 1 1 w
LODB/LODW=Load byte/wd to AL/AX	1 0 1 0 1 1 0 w
STOB/STOW=Stor byte/wd frm AL/A	1 0 1 0 1 0 1 w

CONTROL TRANSFER

CALL = Call:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	

JMP = Unconditional Jump:

Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within segment-short	1 1 1 0 1 0 1 1	disp	
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	

RET = Return from CALL:

Within segment	1 1 0 0 0 0 1 1		
Within seg. adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment, adding immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ=Jump on equal/zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE=Jump on less/not greater or equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG=Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE=Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA=Jump on below or equal/not above	0 1 1 1 0 1 1 0	disp	
JP/JPE=Jump on parity/parity even	0 1 1 1 1 0 1 0	disp	
JO=Jump on overflow	0 1 1 1 0 0 0 0	disp	
JS=Jump on sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ=Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE=Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG=Jump on not less or equal/greater	0 1 1 1 1 1 1 1	disp	

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
JNB/JAE=Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp
JNBE/JA=Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp
JNP/JPO=Jump on not par/par odd	0 1 1 1 1 0 1 1	disp
JNO=Jump on not overflow	0 1 1 1 0 0 0 1	disp
JNS=Jump on not sign	0 1 1 1 1 0 0 1	disp
LOOP=Loop CX times	1 1 1 0 0 0 1 0	disp
LOOPZ/LOOPE=Loop while zero/equal	1 1 1 0 0 0 0 1	disp
LOOPNZ/LOOPNE=Loop while not zero/equal	1 1 1 0 0 0 0 0	disp
JCXZ=Jump on CX zero	1 1 1 0 0 0 1 1	disp

INT = Interrupt

Type specified	1 1 0 0 1 1 0 1	type
Type 3	1 1 0 0 1 1 0 0	
INTO=Interrupt on overflow	1 1 0 0 1 1 1 0	
IRET=Interrupt return	1 1 0 0 1 1 1 1	

PROCESSOR CONTROL

CLC=Clear carry	1 1 1 1 1 0 0 0	
CMC=Complement carry	1 1 1 1 0 1 0 1	
STC=Set carry	1 1 1 1 1 0 0 1	
CLD=Clear direction	1 1 1 1 1 1 0 0	
STD=Set direction	1 1 1 1 1 1 0 1	
CLI=Clear interrupt	1 1 1 1 1 0 1 0	
STI=Set interrupt	1 1 1 1 1 0 1 1	
HLT=Halt	1 1 1 1 0 1 0 0	
WAIT=Wait	1 0 0 1 1 0 1 1	
ESC=Escape (to external device)	1 1 0 1 1 x	mod x r/m
LOCK=Bus lock prefix	1 1 1 1 0 0 0 0	

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction

if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL)
x = don't care
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

0 0 1 reg 1 1 0

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

REG is assigned according to the following table:

16-Bit (w = 1)	8-Bit (w = 0)	Segment
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

intel®

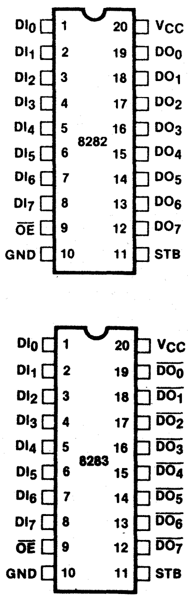
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8282/8283
8-BIT INPUT/OUTPUT PORTS

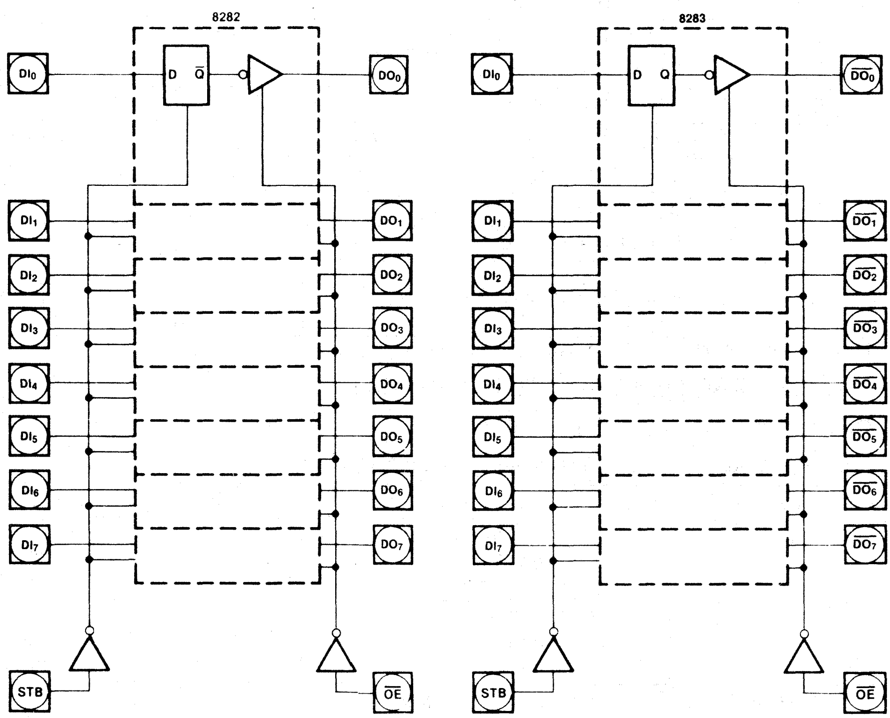
- **Fully Parallel 8-Bit Data Register and Buffer**
- **Transparent during Active Strobe**
- **Supports 8080, 8085, 8048, and 8086 Systems**
- **High Output Drive Capability for Driving System Data Bus**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

The 8282 and 8283 are 8-bit bipolar latches with 3-state output buffers. They can be used to implement latches, buffers, or multiplexers. The 8283 inverts the input data at its outputs while the 8282 does not. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with these devices.

PIN CONFIGURATIONS



LOGIC DIAGRAMS



PIN NAMES

DI_0-DI_7	DATA IN
DO_0-DO_7	DATA OUT
$\overline{OE}$	OUTPUT ENABLE
STB	STROBE

PIN DEFINITIONS

Pin	Description
STB	STROBE (Input). STB is an input control pulse used to strobe data at the data input pins (A_0–A_7) into the data latches. This signal is active HIGH to admit input data. The data is latched at the HIGH to LOW transition of STB.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal which when active LOW enables the contents of the data latches onto the data output pin (B_0–B_7). OE being inactive HIGH forces the output buffers to their high impedance state.
DI_0–DI_7	DATA INPUT PINS (Input). Data presented at these pins satisfying setup time requirements when STB is strobed and latched into the data input latches.
DO_0–DO_7 (8282) $\overline{DO_0}$–$\overline{DO_7}$ (8283)	DATA OUTPUT PINS (Output). When $\overline{CS}$ is true, the data in the data latches is presented as inverted (8283) or non-inverted (8282) data onto the data output pins.

OPERATIONAL DESCRIPTION

The 8282 and 8283 octal latches are 8-bit latches with 3-state output buffers. Data having satisfied the setup time requirements is latched into the data latches by strobing the STB line HIGH to LOW. Holding the STB line in its active HIGH state makes the latches appear transparent. Data is presented to the data output pins by activating the $\overline{OE}$ input line. When $\overline{OE}$ is inactive HIGH the output buffers are in their high impedance state. Enabling or disabling the output buffers will not cause negative-going transients to appear on the data output bus.

D.C. CHARACTERISTICS FOR 8282/8283

Conditions: $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $70°C$

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		−1	V	$I_C = -5$ mA
I_{CC}	Power Supply Current		160	mA	
I_F	Forward Input Current		−0.2	mA	$V_F = 0.45V$
I_R	Reverse Input Current		50	μA	$V_R = 5.25V$
V_{OL}	Output Low Voltage		0.50	V	$I_{OL} = 32$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -5$ mA
I_{OFF}	Output Off Current		50	μA	$V_{OFF} = 0.45$ to $5.25V$
V_{IL}	Input Low Voltage		0.8	V	$V_{CC} = 5.0V$ See Note 1
V_{IH}	Input High Voltage	2.0		V	$V_{CC} = 5.0V$ See Note 1
C_{IN}	Input Capacitance		12	pF	$F = 1$ MHz $V_{BIAS} = 2.5V$, $V_{CC} = 5V$ $T_A = 25°C$

Notes: 1. Output Loading $I_{OL} = 32$ mA, $I_{OH} = -5$ mA, $C_L = 300$ pF

A.C. CHARACTERISTICS FOR 8282/8283

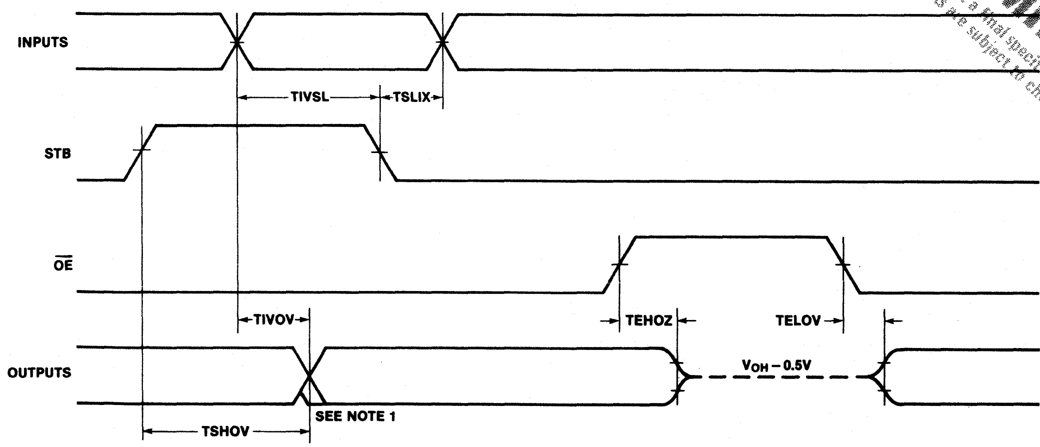
Conditions: $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $70°C$

Loading: Outputs — $I_{OL} = 32$ mA, $I_{OH} = -5$ mA, $C_L = 300$ pF

Symbol	Parameter	Min	Max	Units
TIVOV	Input to Output Delay Inverting Non-Inverting		 25 35	 ns ns
TSHOV	STB to Output Delay Inverting Non-Inverting		 45 55	 ns ns
TEHOZ	Output Disable Time		25	ns
TELOV	Output Enable Time	10	50	ns
TIVSL	Input to STB Setup Time	0		ns
TSLIX	Input to STB Hold Time	25		ns

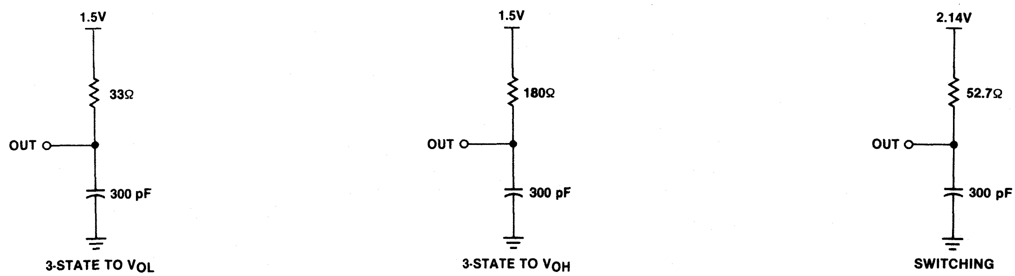
Notes: 1. See waveforms and test load circuit on following page.

8282/8283 TIMING



NOTE: 1. 8283 ONLY — OUTPUT MAY BE MOMENTARILY INVALID FOLLOWING THE HIGH GOING STB TRANSITION.
2. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED

OUTPUT TEST LOAD CIRCUITS



intel®

PRELIMINARY

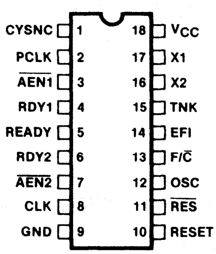
Notice: This is not a final specification. Some parametric limits are subject to change.

8284
CLOCK GENERATOR AND DRIVER FOR 8086 CPU

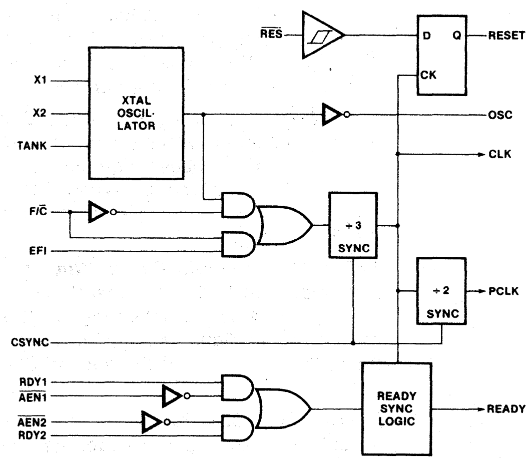
- Generates the System Clock for the 8086
- Uses a Crystal or a TTL Signal for Frequency Source
- Single +5V Power Supply
- 18-Pin Package
- Generates System Reset Output from Schmitt Trigger Input
- Provides Local Ready and MULTIBUS™ Ready Synchronization
- Capable of Clock Synchronization with other 8284's

The 8284 is a bipolar clock generator/driver designed to provide clock signals for the 8086 CPU and peripherals. It also contains READY logic for operation with two MULTIBUS™ systems and provides the 8086's required READY synchronization and timing. Reset logic with hysteresis and synchronization is also provided.

8284 PIN CONFIGURATION



8284 BLOCK DIAGRAM



8284 PIN NAMES

X1, X2	CONNECTIONS FOR CRYSTAL
TANK	USED WITH OVERTONE CRYSTAL
F/C̄	CLOCK SOURCE SELECT
EFI	EXTERNAL CLOCK INPUT
CSYNC	CLOCK SYNCHRONIZATION INPUT
RDY1, RDY2	READY SIGNAL FROM TWO MULTIBUS™ SYSTEMS
AEN1, AEN2	ADDRESS ENABLED QUALIFIERS FOR RDY1,2
RES	RESET INPUT
RESET	SYNCHRONIZED RESET OUTPUT
OSC	OSCILLATOR OUTPUT
CLK	MOS CLOCK FOR 8086
PCLK	TTL CLOCK FOR PERIPHERALS
READY	SYNCHRONIZED READY OUTPUT
V_{CC}	+5 VOLTS
GND	0 VOLTS

PIN DEFINITIONS

Pin	I/O	Definition
$\overline{AEN1}$, $\overline{AEN2}$	I	ADDRESS ENABLE. $\overline{AEN}$ is an active LOW signal. $\overline{AEN}$ serves to qualify its respective Bus Ready Signal (RDY1 or RDY2). $\overline{AEN1}$ validates RDY1 while $\overline{AEN2}$ validates RDY2. Two AEN signal inputs are useful in system configurations which permit the processor to access two Multi-Master System Busses. In non Multi-Master configurations the $\overline{AEN}$ signal inputs are tied true (LOW).
RDY1, RDY2	I	BUS READY (Transfer Complete). RDY is an active HIGH signal which is an indication from a device located on the system data bus that data has been received, or is available. RDY1 is qualified by $\overline{AEN1}$ while RDY2 is qualified by $\overline{AEN2}$.
READY	O	READY. READY is an active HIGH signal which is the synchronized RDY signal input. Since RDY occurs asynchronously with respect to the processor's clock (CLK) it is necessary for them to be synchronized before being presented to the processor. READY is cleared after the guaranteed hold time to the processor has been met.
X1, X2, TNK	I	CRYSTAL IN. X1 and X2 are the pins to which a crystal is attached with TNK (TANK) serving as the overtone input. The crystal frequency is 3 times the desired processor clock frequency.
F/$\overline{C}$	I	FREQUENCY/CRYSTAL SELECT. F/$\overline{C}$ is a strapping option. When strapped LOW, F/$\overline{C}$ permits the processor's clock to be generated by the crystal. When F/$\overline{C}$ is strapped HIGH, CLK is generated from the EFI input.
EFI	I	EXTERNAL FREQUENCY IN. When F/$\overline{C}$ is strapped HIGH, CLK is generated from the input frequency appearing on this pin. The input signal is a square wave 3 times the frequency of the desired CLK output.
CLK	O	PROCESSOR CLOCK. CLK is the clock output used by the processor and all devices which directly connect to the processor's local bus (i.e., the bipolar support chips and other MOS devices). CLK has an output frequency which is 1/3 of the crystal or EFI input frequency and a 1/3 duty cycle. An output HIGH of 4.5 volts (V_{CC} = 5V) is provided on this pin to drive MOS devices.
PCLK	O	PERIPHERAL CLOCK. PCLK is a TTL level peripheral clock signal whose output frequency is 1/2 that of CLK and has a 50% duty cycle.
OSC	O	OSCILLATOR OUTPUT. OSC is the TTL level output of the internal oscillator circuitry. Its frequency is equal to that of the crystal.
$\overline{RES}$	I	RESET IN. $\overline{RES}$ is an active LOW signal which is used to generate RESET. The 8284 provides a Schmitt trigger input so that an RC connection can be used to establish the power-up reset of proper duration.
RESET	O	RESET. Reset is an active HIGH signal which is used to reset the 8086 family processors. Its timing characteristics are determined by $\overline{RES}$.
CSYNC	I	CLOCK SYNCHRONIZATION. CSYNC is an active HIGH signal which allows multiple 8284's to be synchronized to provide clocks that are in phase. When CSYNC is HIGH the internal counters are reset. When CSYNC goes LOW the internal counters are allowed to resume counting. CSYNC needs to be externally synchronized to EFI. When using the internal oscillator CSYNC should be hardwired to ground.
GND		Ground
V_{CC}		+5V supply

FUNCTIONAL DESCRIPTION

GENERAL

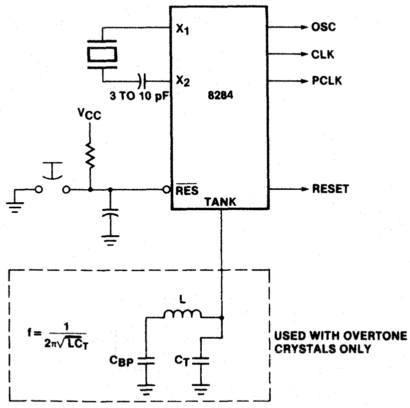
The 8284 is a single chip clock generator/driver for the 8086 CPU. The chip contains a crystal controlled oscillator, a "divide by three" counter, complete MULTI-BUS™ "Ready" synchronization and reset logic.

OSCILLATOR

The oscillator circuit of the 8284 is designed primarily for use with an external series resonant, fundamental mode, crystal from which the basic operating frequency is derived. However, overtone mode crystals can be used with a tank circuit as shown in Figure 1.

The crystal frequency should be selected at three times the required CPU clock. X_1 and X_2 are the two crystal input crystal connections.

The output of the oscillator is buffered and brought out on OSC so that other system timing signals can be derived from this stable, crystal-controlled source.



The tank input to the oscillator allows the use of overtone mode crystals. The tank circuit shunts the crystal's fundamental and high overtone frequencies and allows the third harmonic to oscillate. The external LC network is connected to the TANK input and is AC coupled to ground.

Figure 1

CLOCK GENERATOR

The clock generator consists of a synchronous divide-by-three counter with a special clear input that inhibits the counting. This clear input (CSYNC) allows the output clock to be synchronized with an external event (such as another 8284 clock). It is necessary to synchronize the CSYNC input to the EFI clock external to the 8284. This is accomplished with two Schottky flip-flops. (See Figure 2.) The counter output is a 33% duty cycle clock at one-third the input frequency.

The F/$\overline{C}$ input is a strapping pin that selects either the crystal oscillator or the EFI input as the clock for the ÷ 3 counter. If the EFI input is selected as the clock source, the oscillator section can be used independently for another clock source. Output is taken from OSC.

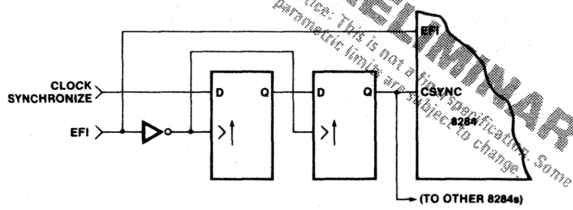


Figure 2. CSYNC Synchronization

CLOCK OUTPUTS

The CLK output is a 33% duty cycle MOS clock driver designed to drive the 8086 processor directly. PCLK is a TTL level peripheral clock signal whose output frequency is 1/2 that of CLK. PCLK has a 50% duty cycle.

RESET LOGIC

The reset logic provides a Schmitt trigger input ($\overline{RES}$) and a synchronizing flip-flop to generate the reset timing. The reset signal is synchronized to the falling edge of CLK. A simple RC network can be used to provide power on reset by utilizing this function of the 8284.

READY SYNCHRONIZATION

Two READY inputs (RDY1, RDY2) are provided to accomomodate two Multi-Master system busses. Each input has a qualifier ($\overline{AEN1}$ and $\overline{AEN2}$, respectively). The $\overline{AEN}$ signals validate their respective RDY signals. If a Multi-Master system is not being used the $\overline{AEN}$ pin should be tied LOW.

The READY output is an active HIGH signal which is the synchronized RDY1 or RDY2 input. Since RDY1 and RDY2 occur asynchronously with respect to the processor's clock (CLK), it is necessary to synchronize them before presenting them to the processor to insure they meet the required set-up time. The READY logic does this job and also guarantees the required hold time before clearing the READY signal.

D.C. CHARACTERISTICS FOR 8284

Conditions: $T_A = 0°C$ to $70°C$; $V_{CC} = 5V \pm 10\%$

Symbol	Parameter	Min	Max	Units	Test Conditions
I_F	Forward Input Current		−0.5	mA	$V_F = 0.45V$
I_R	Reverse Input Current		50	μA	$V_R = 5.25V$
V_C	Input Forward Clamp Voltage		−1.0	V	$I_C = -5$ mA
I_{CC}	Power Supply Current		140	mA	
V_{IL}	Input LOW Voltage		0.8	V	$V_{CC} = 5.0V$
V_{IH}	Input HIGH Voltage	2.0		V	$V_{CC} = 5.0V$
V_{IH_R}	Reset Input HIGH Voltage	2.6		V	$V_{CC} = 5.0V$
V_{OL}	Output LOW Voltage		0.45	V	5 mA
V_{OH}	Output HIGH Voltage CLK Other Outputs	4 2.4		V V	−1 mA −1 mA
V_{IH_R}-V_{IL_R}	$\overline{RES}$ Input Hysteresis	0.25		V	$V_{CC} = 5.0V$

A.C. CHARACTERISTICS FOR 8284

Conditions: $T_A = 0°C$ to $70°C$; $V_{CC} = 5V \pm 10\%$

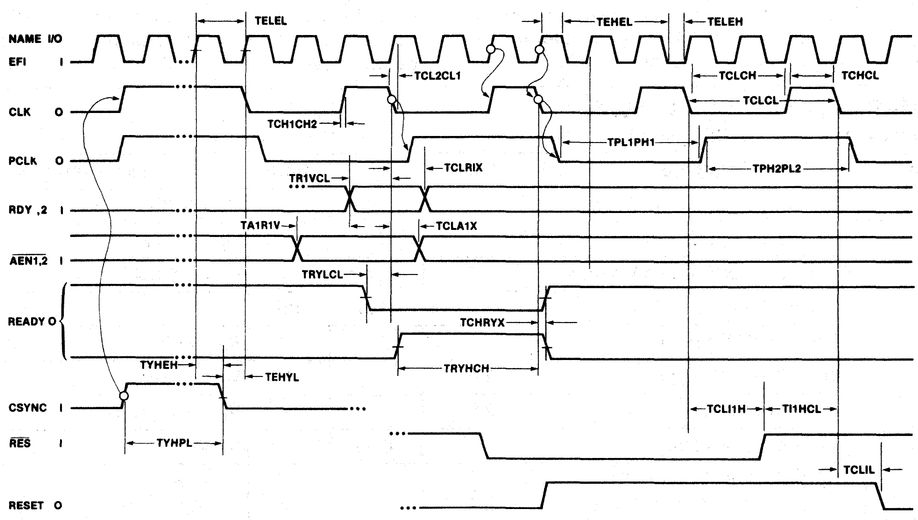
TIMING REQUIREMENTS

Symbol	Parameter	Min	Max	Units	Test Conditions
TEHEL	External Frequency High Time	20		ns	
TELEH	External Frequency Low Time	20		ns	
TELEL	EFI Period	TEHEL + TELEH + δ		ns	(Note 1)
	XTAL Frequency	12	25	MHz	
TR1VCL	RDY1, RDY2 Set-Up to CLK	45		ns	
TCLR1X	RDY1, RDY2 Hold to CLK	0		ns	
TA1VR1V	$\overline{AEN1}$, $\overline{AEN2}$ Set-Up to RDY1, RDY2	15		ns	
TCLA1X	$\overline{AEN1}$, $\overline{AEN2}$ Hold to CLK	0		ns	
TYHEH	CSYNC Set-Up to EFI	20		ns	
TEHYL	CSYNC Hold to EFI	20		ns	
TYHYL	CSYNC Width	2·TELEL		ns	
TCLI1H	$\overline{RES}$ Set-Up to CLK	65		ns	(Note 2)
TI1HCL	$\overline{RES}$ Hold to CLK	20		ns	(Note 2)

TIMING RESPONSES

Symbol	Parameter	Min	Max	Units	Test Conditions
TCLCL	CLK Cycle Period	125		ns	
TCHCL	CLK High Time	(⅓TCLCL) − 1.0		ns	
TCLCH	CLK Low Time	(⅔TCLCL) − 15.0		ns	
TCH1CH2 TCL2CL1	CLK Rise and Fall Time		10	ns	1.0V to 3.5V
TPHPL	PCLK High Time	TCLCL − 20		ns	
TPLPH	PCLK Low Time	TCLCL − 20		ns	
TRYLCL	Ready Inactive to CLK (See Note 4)	5-15		ns	
TRYHCH	Ready Active to CLK (See Note 3)	110		ns	
TCHRYX	CLK to Ready Invalid (See Note 3)	30		ns	
TCLIL	CLK to Reset Delay	40		ns	

Note: 1. δ = EFI rise + EFI fall.
2. Set up and hold only necessary to guarantee recognition at next clock.
3. Applies only to TS and TW states.
4. Applies only to T2 states.



ALL TIMING MEASUREMENTS ARE MADE AT 1.5 VOLTS, UNLESS OTHERWISE NOTED

Figure 3

intel®

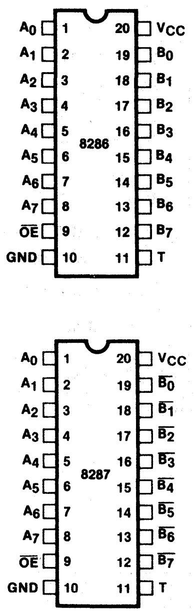
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8286/8287
8-BIT PARALLEL BIDIRECTIONAL BUS DRIVERS

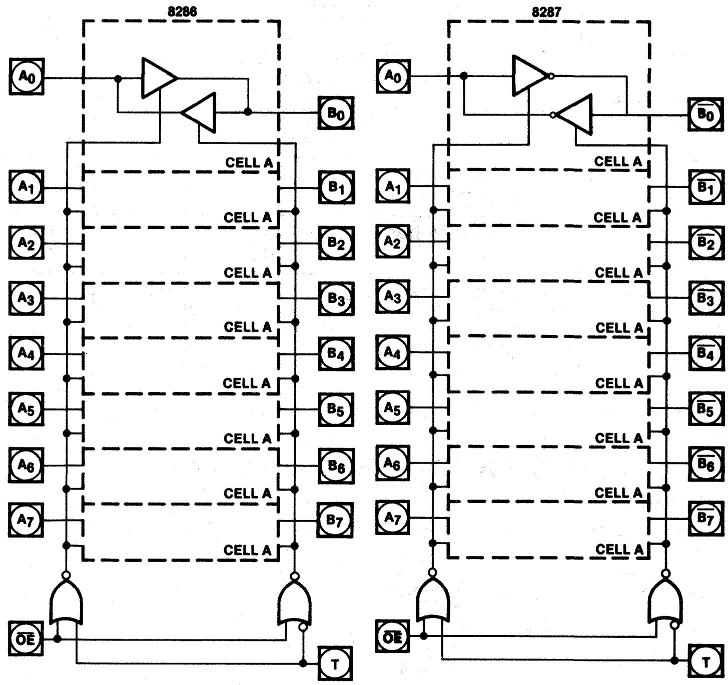
- **Data Bus Buffer Driver for MCS-86™, MCS-80™, MCS-85™, and MCS-48™**
- **High Output Drive Capability for Driving System Data Bus**
- **Fully Parallel 8-Bit Transceivers**
- **3-State Outputs**
- **20-Pin Package with 0.3" Center**
- **No Output Low Noise when Entering or Leaving High Impedance State**

The 8286 and 8287 are 8-bit bipolar transceivers with 3-state outputs. The 8287 inverts the input data at its outputs while the 8286 does not. Thus, a wide variety of applications for buffering in microcomputer systems can be met.

PIN CONFIGURATIONS



LOGIC DIAGRAMS



PIN NAMES

A_0-A_7	LOCAL BUS DATA
B_0-B_7	SYSTEM BUS DATA
$\overline{OE}$	OUTPUT ENABLE
T	TRANSMIT

PIN DEFINITIONS

Pin	Description
T	TRANSMIT (Input). T is an input control signal used to control the direction of the transceivers. When HIGH, it configures the transceiver's B_0–B_7 as outputs with A_0–A_7 as inputs. T LOW configures A_0–A_7 as the outputs with B_0–B_7 serving as the inputs.
$\overline{OE}$	OUTPUT ENABLE (Input). $\overline{OE}$ is an input control signal used to enable the appropriate output driver (as selected by T) onto its respective bus. This signal is active LOW.
A_0–A_7	LOCAL BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the processor's local bus depending upon the state of the T pin.
B_0–B_7 (8286) $\overline{B_0–B_7}$ (8287)	SYSTEM BUS DATA PINS (Input/Output). These pins serve to either present data to or accept data from the system bus depending upon the state of the T pin.

OPERATIONAL DESCRIPTION

The 8286 and 8287 transceivers are 8-bit transceivers with high impedance outputs. With T active HIGH and $\overline{OE}$ active LOW, data at the A_0–A_7 pins is driven onto the B_0–B_7 pins. With T inactive LOW and $\overline{OE}$ active LOW, data at the B_0–B_7 pins is driven onto the A_0–A_7 pins. No output low glitching will occur whenever the transceivers are entering or leaving the high impedance state.

D.C. CHARACTERISTICS FOR 8286/8287

Conditions: V_{CC} = 5V ±5%, T_A = 0°C to 70°C

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		-1	V	I_C = -5 mA
I_{CC}	Power Supply Current—8287 —8286		95 135	mA mA	
I_F	Forward Input Current		-0.2	mA	V_F = 0.45V
I_R	Reverse Input Current		50	μA	V_R = 5.25V
V_{OL}	Output Low Voltage —B Outputs —A Outputs		0.5 0.5	V V	I_{OL} = 32 mA I_{OL} = 10 mA
V_{OH}	Output High Voltage —B Outputs —A Outputs	2.4 2.4		V V	I_{OH} = -5 mA I_{OH} = -1 mA
I_{OFF} I_{OFF}	Output Off Current Output Off Current		IF IR		V_{OFF} = 0.45V V_{OFF} = 5.25V
V_{IL}	Input Low Voltage —A Side —B Side		0.8 0.9	V V	V_{CC} = 5.0V, See Note 1 V_{CC} = 5.0V, See Note 1
V_{IH}	Input High Voltage	2.0		V	V_{CC} = 5.0V, See Note 1
C_{IN}	Input Capacitance	12		pF	F = 1 MHz V_{BIAS} = 2.5V, V_{CC} = 5V T_A = 25°C

Note: 1. B Outputs — I_{OL} = 32 mA, I_{OH} = -5 mA, C_L = 300 pF A Outputs — I_{OL} = 16 mA, I_{OH} = -1 mA, C_L = 100 pF

A.C. CHARACTERISTICS FOR 8286/8287

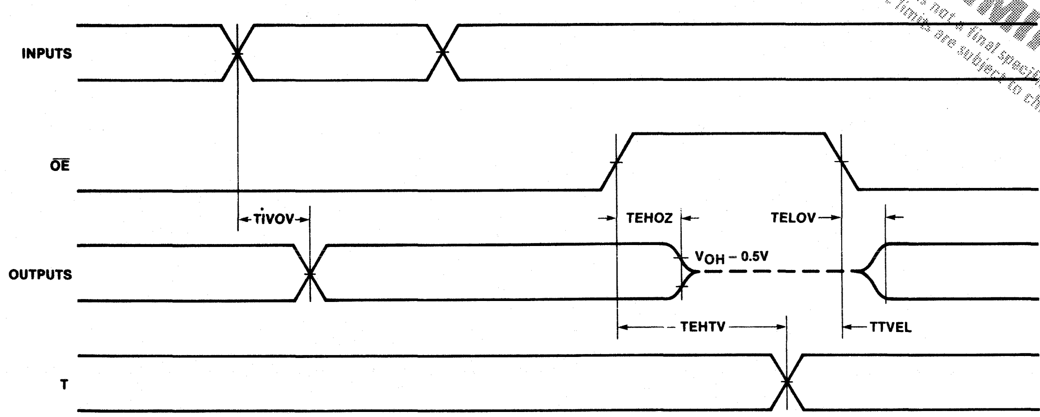
Conditions: V_{CC} = 5V ±5%, T_A = 0°C to 70°C

Loading: B Outputs — I_{OL} = 32 mA, I_{OH} = −5 mA, C_L = 300 pF
A Outputs — I_{OL} = 16 mA, I_{OH} = −1 mA, C_L = 100 pF

Symbol	Parameter	Min	Max	Units	Test Conditions
TIVOV	Input to Output Delay Inverting Non-Inverting		 25 35	 ns ns	(See Note 1)
TEHTV	Transmit/Receive Hold Time	TEHOZ		ns	
TTVEL	Transmit/Receive Setup	30		ns	
TEHOZ	Output Disable Time		25	ns	
TELOV	Output Enable Time	10	50	ns	

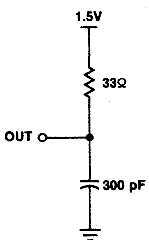
Note: 1. See waveforms and test load circuit on following page.

8286/8287 TIMING



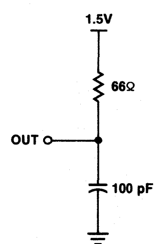
NOTE: 1. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS OTHERWISE NOTED.

TEST LOAD CIRCUITS



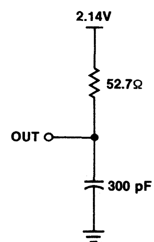
3-STATE TO V_{OL}

B OUTPUT



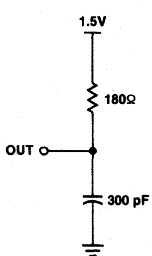
3-STATE TO V_{OL}

A OUTPUT



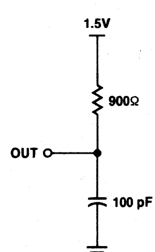
SWITCHING

B OUTPUT



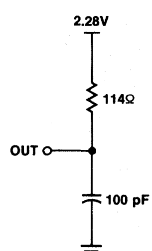
3-STATE TO V_{OH}

B OUTPUT



3-STATE TO V_{OH}

A OUTPUT



SWITCHING

A OUTPUT

intel®

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

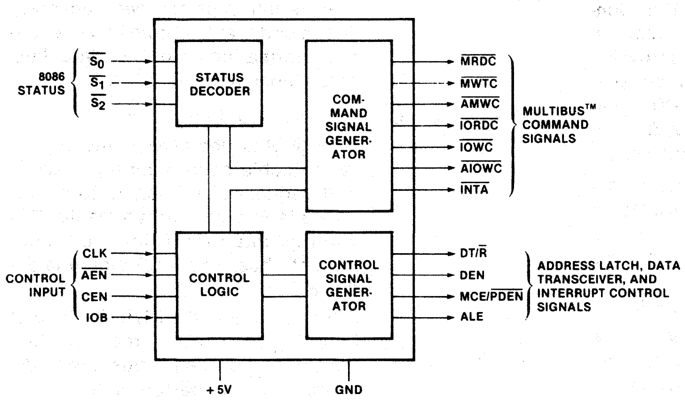
8288
BUS CONTROLLER FOR THE 8086 CPU

- Bipolar Drive Capability
- Provides Advanced Commands
- Provides Wide Flexibility in System Configurations
- 3-State Command Output Drivers
- Configurable for Use with an I/O Bus
- Facilitates Interface to One or Two Multi-Master Busses

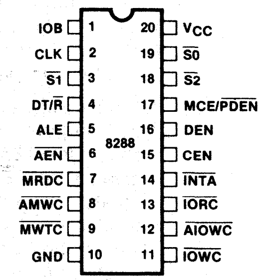
The Intel® 8288 Bus Controller is a 20-pin bipolar component for use with medium-to-large 8086 processing systems. The bus controller provides command and control timing generation as well as bipolar bus drive capability while optimizing system performance.

A strapping option on the bus controller configures it for use with a multi-master system bus and separate I/O bus.

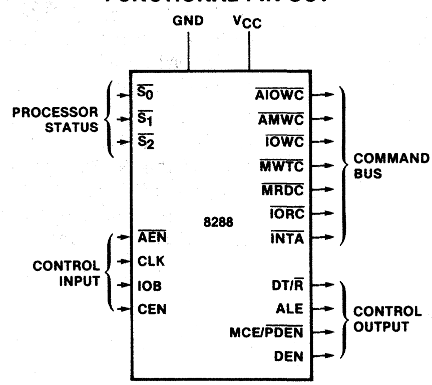
BLOCK DIAGRAM



PIN CONFIGURATION



FUNCTIONAL PIN-OUT



PIN DEFINITIONS

Name	I/O	Function
V_{CC}		+5V supply.
GND		Ground.
$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$	I	Status Input Pins: These pins are the status input pins from the 8086 processor. The 8288 decodes these inputs to generate command and control signals at the appropriate time. When these pins are not in use (passive) they are all HIGH. (See chart under Command and Control Logic.)
CLK	I	Clock: This is a clock signal from the 8284 clock generator and serves to establish when command and control signals are generated.
ALE	O	Address Latch Enable: This signal serves to strobe an address into the address latches. This signal is active HIGH and latching occurs on the falling (HIGH to LOW) transition. ALE is intended for use with transparent D type latches.
DEN	O	Data Enable: This signal serves to enable data transceivers onto either the local or system data bus. This signal is active HIGH.
DT/$\overline{R}$	O	Data Transmit/Receive: This signal establishes the direction of data flow through the transceivers. A HIGH on this line indicates Transmit (write to I/O or memory) and a LOW indicates Receive (Read).
$\overline{AEN}$	I	Address Enable: $\overline{AEN}$ enables command outputs of the 8288 Bus Controller at least 85 ns after it becomes active (LOW). $\overline{AEN}$ going inactive immediately 3-states the command output drivers. $\overline{AEN}$ does not affect the I/O command lines if the 8288 is in the I/O Bus mode (IOB tied HIGH).
CEN	I	Command Enable: When this signal is LOW all 8288 command outputs and the DEN and $\overline{PDEN}$ control outputs are forced to their inactive state. When this signal is HIGH, these same outputs are enabled.
IOB	I	Input/Output Bus Mode: When the IOB is strapped HIGH the 8288 functions in the I/O Bus mode. When it is strapped LOW, the 8288 functions in the System Bus mode. (See sections on I/O Bus and System Bus modes).
$\overline{AIOWC}$	O	Advanced I/O Write Command: The $\overline{AIOWC}$ issues an I/O Write Command earlier in the machine cycle to give I/O devices an early indication of a write instruction. Its timing is the same as a read command signal. $\overline{AIOWC}$ is active LOW.
$\overline{IOWC}$	O	I/O Write Command: This command line instructs an I/O device to read the data on the data bus. This signal is active LOW.
$\overline{IORC}$	O	I/O Read Command: This command line instructs an I/O device to drive its data onto the data bus. This signal is active LOW.
$\overline{AMWC}$	O	Advanced Memory Write Command: The $\overline{AMWC}$ issues a memory write command earlier in the machine cycle to give memory devices an early indication of a write instruction. Its timing is the same as a read command signal. $\overline{AMWC}$ is active LOW.
$\overline{MWTC}$	O	Memory Write Command: This command line instructs the memory to record the data present on the data bus. This signal is active LOW.
$\overline{MRDC}$	O	Memory Read Command: This command line instructs the memory to drive its data onto the data bus. This signal is active LOW.
$\overline{INTA}$	O	Interrupt Acknowledge: This command line tells an interrupting device that its interrupt has been acknowledged and that it should drive vectoring information onto the data bus. This signal is active LOW.
MCE/$\overline{PDEN}$	O	This is a dual function pin. MCE (IOB is tied LOW): Master Cascade Enable occurs during an interrupt sequence and serves to read a Cascade Address from a master PIC (Priority Interrupt Controller) onto the data bus. The MCE signal is active HIGH. $\overline{PDEN}$ (IOB is tied HIGH): Peripheral Data Enable enables the data bus transceiver for the I/O bus during I/O instructions. It performs the same function for the I/O bus that DEN performs for the system bus. $\overline{PDEN}$ is active LOW.

COMMAND AND CONTROL LOGIC

The command logic decodes the three 8086 CPU status lines ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) to determine what command is to be issued.

This chart shows the meaning of each status "word".

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	8086 State	8288 Command
0	0	0	Interrupt Acknowledge	$\overline{\text{INTA}}$
0	0	1	Read I/O Port	$\overline{\text{IORC}}$
0	1	0	Write I/O Port	$\overline{\text{IOWC}}$, $\overline{\text{AIOWC}}$
0	1	1	Halt	None
1	0	0	Code Access	$\overline{\text{MRDC}}$
1	0	1	Read Memory	$\overline{\text{MRDC}}$
1	1	0	Write Memory	$\overline{\text{MWTC}}$, $\overline{\text{AMWC}}$
1	1	1	Passive	None

The command is issued in one of two ways dependent on the mode of the 8288 Bus Controller.

I/O Bus Mode — The 8288 is in the I/O Bus mode if the IOB pin is strapped HIGH. In the I/O Bus mode all I/O command lines ($\overline{\text{IORC}}$, $\overline{\text{IOWC}}$, $\overline{\text{AIOWC}}$, $\overline{\text{INTA}}$) are always enabled (i.e., not dependent on $\overline{\text{AEN}}$). When an I/O command is initiated by the processor, the 8288 immediately activates the command lines using $\overline{\text{PDEN}}$ and DT/$\overline{\text{R}}$ to control the I/O bus transceiver. The I/O command lines should not be used to control the system bus in this configuration because no arbitration is present. This mode allows one 8288 Bus Controller to handle two external busses. No waiting is involved when the CPU wants to gain access to the I/O bus. Normal memory access requires a "Bus Ready" signal ($\overline{\text{AEN}}$ LOW) before it will proceed. It is advantageous to use the IOB mode if I/O or peripherals dedicated to one processor exist in a multi-processor system.

System Bus Mode — The 8288 is in the System Bus mode if the IOB pin is strapped LOW. In this mode no command is issued until 85 ns after the $\overline{\text{AEN}}$ Line is activated (LOW). This mode assumes bus arbitration logic will inform the bus controller (on the $\overline{\text{AEN}}$ line) when the bus is free for use. Both memory and I/O commands wait for bus arbitration. This mode is used when only one bus exists. Here, both I/O and memory are shared by more than one processor.

Command Outputs

The advanced write commands are made available to initiate write procedures early in the machine cycle. This signal can be used to prevent the 8086 CPU from entering an unnecessary wait state.

The command outputs are:

$\overline{\text{MRDC}}$ — Memory Read Command
$\overline{\text{MWTC}}$ — Memory Write Command
$\overline{\text{IORC}}$ — I/O Read Command
$\overline{\text{IOWC}}$ — I/O Write Command
$\overline{\text{AMWC}}$ — Advanced Memory Write Command
$\overline{\text{AIOWC}}$ — Advanced I/O Write Command
$\overline{\text{INTA}}$ — Interrupt Acknowledge

$\overline{\text{INTA}}$ (Interrupt Acknowledge) acts as an I/O read during an interrupt cycle. Its purpose is to inform an interrupting device that its interrupt is being acknowledged and that it should place vectoring information onto the data bus.

Control Outputs

The control outputs of the 8288 are Data Enable (DEN), Data Transmit/Receive (DT/$\overline{\text{R}}$) and Master Cascade Enable/Peripheral Data Enable (MCE/$\overline{\text{PDEN}}$). The DEN signal determines when the external bus should be enabled onto the local bus and the DT/$\overline{\text{R}}$ determines the direction of data transfer. These two signals usually go to the chip select and direction pins of a transceiver.

The MCE/$\overline{\text{PDEN}}$ pin changes function with the two modes of the 8288. When the 8288 is in the IOB mode (IOB HIGH) the $\overline{\text{PDEN}}$ signal serves as a dedicated data enable signal for the I/O or Peripheral System bus.

Interrupt Acknowledge and MCE

The MCE signal is used during an interrupt acknowledge cycle if the 8288 is in the System Bus mode (IOB LOW). During any interrupt sequence there are two interrupt acknowledge cycles that occur back to back. During the first interrupt cycle no data or address transfers take place. Logic should be provided to mask off MCE during this cycle. Just before the second cycle begins the MCE signal gates a master Priority Interrupt Controller's (PIC) cascade address onto the processor's local bus where ALE (Address Latch Enable) strobes it into the address latches. On the leading edge of the second interrupt cycle the addressed slave PIC gates an interrupt vector onto the system data bus where it is read by the processor.

If the system contains only one PIC, the MCE signal is not used. In this case the second Interrupt Acknowledge signal gates the interrupt vector onto the processor bus.

Address Latch Enable and Halt

Address Latch Enable (ALE) occurs during each machine cycle and serves to strobe data into the address latches. ALE also serves to strobe the status ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$) into a latch within the 8288. For this reason an ALE occurs when entering a halt state.

Command Enable

The Command Enable (CEN) input acts as a command qualifier for the 8288. If the CEN pin is high the 8288 functions normally. If the CEN pin is pulled LOW, all command lines are held in their inactive state (not 3-state). This feature can be used to implement memory partitioning and to eliminate address conflicts between system bus devices and resident bus devices.

D.C. CHARACTERISTICS FOR THE 8288

Conditions: V_{CC} = 5V ± 10%, T_A = 0°C to 70°C

Symbol	Parameter	Min	Max	Units	Test Conditions
V_C	Input Clamp Voltage		−1	V	I_C = −5 mA
I_{CC}	Power Supply Current		170	mA	
I_F	Forward Input Current		−0.7	mA	V_F = 0.45V
I_R	Reverse Input Current		50	μA	V_R = 5.25V
V_{OL}	Output Low Voltage				
	Command Outputs		.5	V	I_{OL} = 32 mA
	Control Outputs		.5	V	I_{OL} = 16 mA
V_{OH}	Output High Voltage				
	Command Outputs	2.4		V	I_{OH} = −5 mA
	Control Outputs	2.4		V	I_{OH} = −1 mA
V_{IL}	Input Low Voltage		0.8	V	
V_{IH}	Input High Voltage	2.0		V	
I_{OFF}	Output Off Current		100	μA	V_{OFF} = 0.4 to 5.25V

A.C. CHARACTERISTICS FOR THE 8288

Conditions: V_{CC} = 5V ± 10%, T_A = 0°C to 70°C

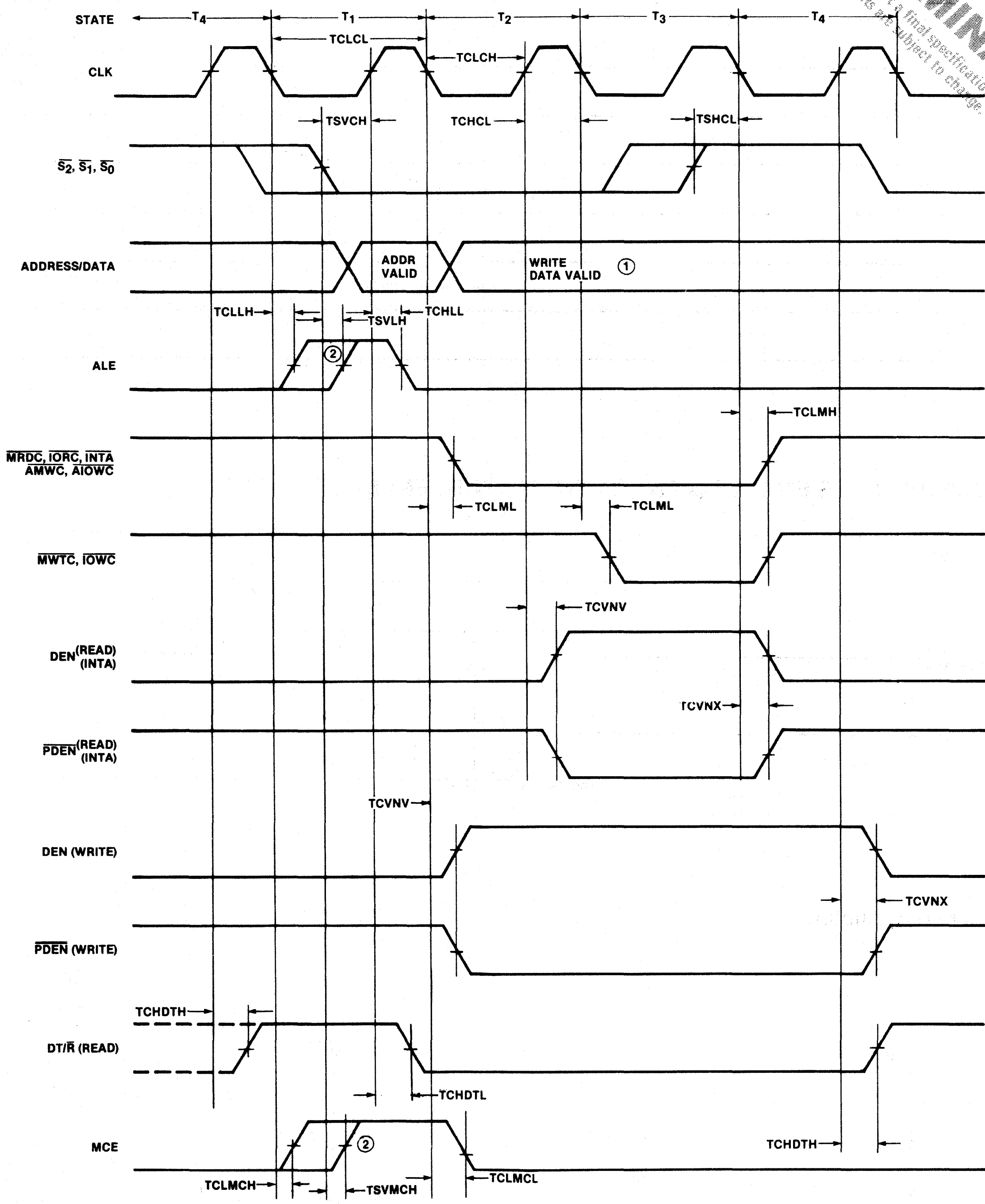
TIMING REQUIREMENTS

Symbol	Parameter	Min	Max	Unit	Loading
TCLCL	CLK Cycle Period	125		ns	
TCLCH	CLK Low Time	66		ns	
TCHCL	CLK High Time	40		ns	
TSVCH	Status Active Setup	65	TCLCL − 10	ns	
TSHCL	Status Inactive Setup	55	TCLCL − 10	ns	

TIMING RESPONSES

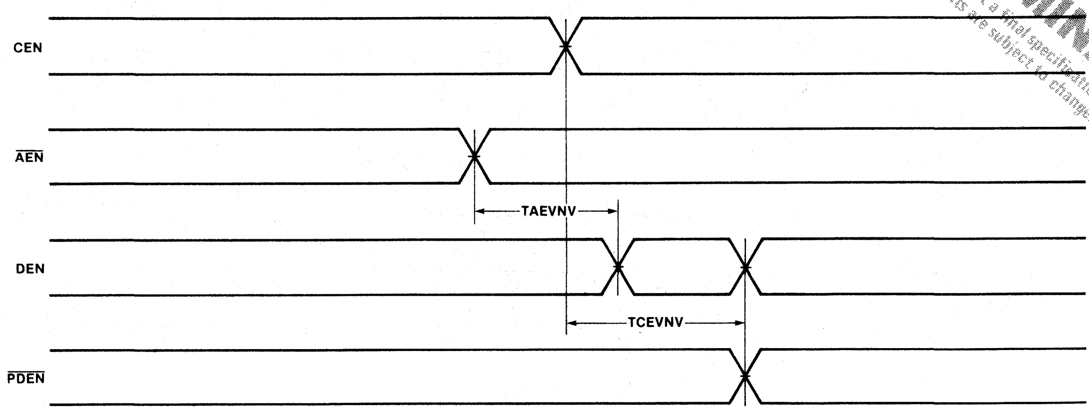
Symbol	Parameter	Min	Max	Unit	Loading
TCVNV	Control Active Delay	5	45	ns	
TCVNX	Control Inactive Delay	10	45	ns	
TCLLH,TCLMCH	ALE MCE Active Delay (from CLK)		15	ns	
TSVLH,TCLMCH	ALE MCE Active Delay (from Status)		15	ns	
TCHLL,TCLMCL	ALE MCE Inactive Delay		15	ns	$\overline{MRDC}$, $\overline{IORC}$, $\overline{MWTC}$, $\overline{IOWC}$, $\overline{INTA}$, $\overline{AMWC}$, $\overline{AIOWC}$: I_{OL} = 32 mA, I_{OH} = −2 mA, C_L = 300 pF
TCLML	Command Active Delay	10	35	ns	
TCLMH	Command Inactive Delay	10	40	ns	
TCHDTL	Direction Control Active Delay		50	ns	
TCLDTH	Direction Control Inactive Delay		30	ns	
TAELCH	Command Enable Time		40	ns	
TAEHCZ	Command Disable Time		40	ns	Other: I_{OL} = 16 mA, I_{OH} = −1 mA, C_L = 80 pF
TAELCV	Enable Delay Time	105	275	ns	
TAEVNV	$\overline{AEN}$ to DEN		20	ns	
TCEVNV	CEN to DEN, PDEN		20	ns	
TCELRH	CEN to Command		TCLML	ns	

8288 TIMING DIAGRAM

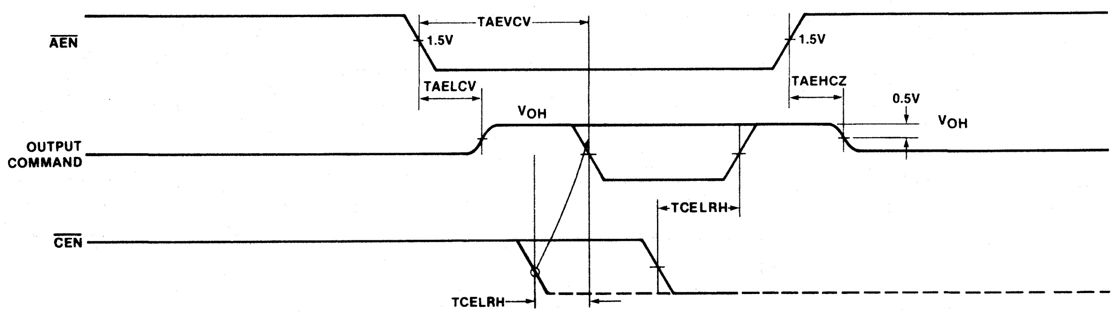


NOTES:
1. ADDRESS/DATA BUS IS SHOWN ONLY FOR REFERENCE PURPOSES.
2. LEADING EDGE OF ALE AND MCE IS DETERMINED BY THE FALLING EDGE OF CLK OR STATUS GOING ACTIVE, WHICHEVER OCCURS LAST.
3. ALL TIMING MEASUREMENTS ARE MADE AT 1.5V UNLESS SPECIFIED OTHERWISE.

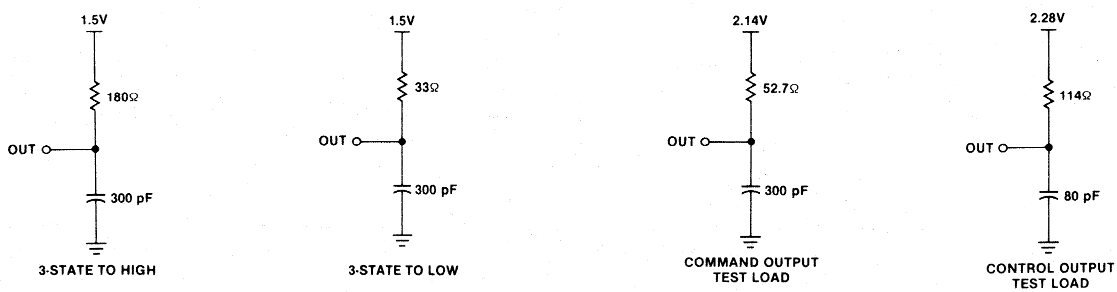
DEN, PDEN QUALIFICATION TIMING



8288 ADDRESS ENABLE (AEN) TIMING (3-STATE ENABLE/DISABLE)



TEST LOAD CIRCUITS



3-STATE COMMAND OUTPUT TEST LOAD

intel®

8259A
PROGRAMMABLE INTERRUPT CONTROLLER

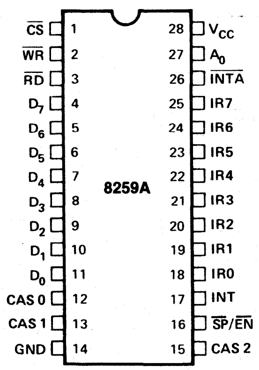
- **MCS-86™ Compatible**
- **MCS-80/85™ Compatible**
- **Eight-Level Priority Controller**
- **Expandable to 64 Levels**
- **Programmable Interrupt Modes**
- **Individual Request Mask Capability**
- **Single +5V Supply (No Clocks)**
- **28-Pin Dual-In-Line Package**

The Intel® 8259A Programmable Interrupt Controller handles up to eight vectored priority interrupts for the CPU. It is cascadable for up to 64 vectored priority interrupts without additional circuitry. It is packaged in a 28-pin DIP, uses NMOS technology and requires a single +5V supply. Circuitry is static, requiring no clock input.

The 8259A is designed to minimize the software and real time overhead in handling multi-level priority interrupts. It has several modes, permitting optimization for a variety of system requirements.

The 8259A is fully upward compatible with the Intel® 8259. Software originally written for the 8259 will operate the 8259A in all 8259 equivalent modes (MCS-80/85, Non-Buffered, Edge Triggered).

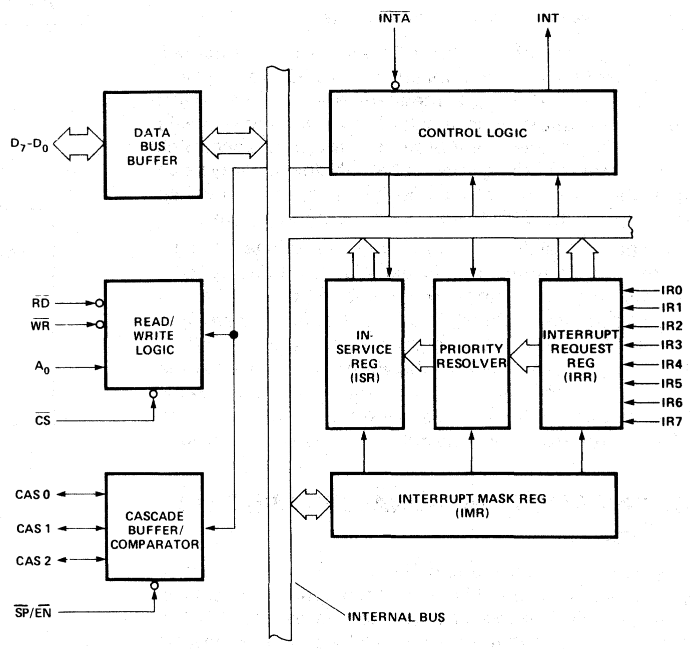
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (BI-DIRECTIONAL)
RD	READ INPUT
WR	WRITE INPUT
A_0	COMMAND SELECT ADDRESS
CS	CHIP SELECT
CAS2-CAS0	CASCADE LINES
SP/EN	SLAVE PROGRAM INPUT/ENABLE
INT	INTERRUPT OUTPUT
INTA	INTERRUPT ACKNOWLEDGE INPUT
IR0-IR7	INTERRUPT REQUEST INPUTS

BLOCK DIAGRAM



June 1978

INTERRUPTS IN MICROCOMPUTER SYSTEMS

Microcomputer system design requires that I/O devices such as keyboards, displays, sensors and other components receive servicing in an efficient manner so that large amounts of the total system tasks can be assumed by the microcomputer with little or no effect on throughput.

The most common method of servicing such devices is the *Polled* approach. This is where the processor must test each device in sequence and in effect "ask" each one if it needs servicing. It is easy to see that a large portion of the main program is looping through this continuous polling cycle and that such a method would have a serious, detrimental effect on system throughput, thus limiting the tasks that could be assumed by the microcomputer and reducing the cost effectiveness of using such devices.

A more desirable method would be one that would allow the microprocessor to be executing its main program and only stop to service peripheral devices when it is told to do so by the device itself. In effect, the method would provide an external asynchronous input that would inform the processor that it should complete whatever instruction that is currently being executed and fetch a new routine that will service the requesting device. Once this servicing is complete, however, the processor would resume exactly where it left off.

This method is called *Interrupt*. It is easy to see that system throughput would drastically increase, and thus more tasks could be assumed by the microcomputer to further enhance its cost effectiveness.

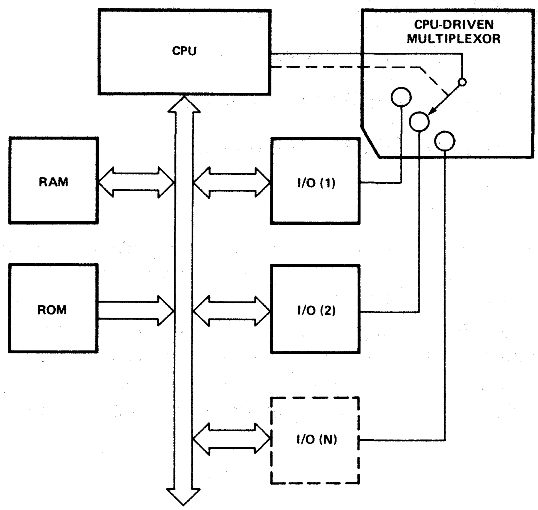
The Programmable Interrupt Controller (PIC) functions as an overall manager in an Interrupt-Driven system environment. It accepts requests from the peripheral equipment, determines which of the incoming requests is of the highest importance (priority), ascertains whether the incoming request has a higher priority value than the level currently being serviced, and issues an interrupt to the CPU based on this determination.

Each peripheral device or structure usually has a special program or "routine" that is associated with its specific functional or operational requirements; this is referred to as a "service routine". The PIC, after issuing an Interrupt to the CPU, must somehow input information into the CPU that can "point" the Program Counter to the service routine associated with the requesting device. This "pointer" is an address in a vectoring table and will often be referred to, in this document, as vectoring data.

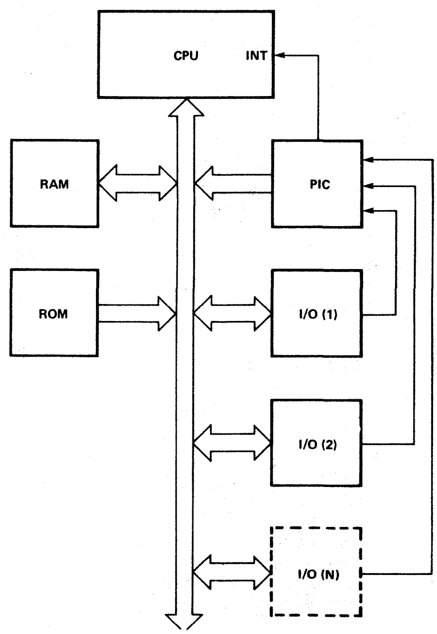
8259A BASIC FUNCTIONAL DESCRIPTION

GENERAL

The 8259A is a device specifically designed for use in real time, interrupt driven microcomputer systems. It manages eight levels or requests and has built-in features for expandability to other 8259A's (up to 64 levels). It is programmed by the system's software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by the 8259A can be configured to match his system requirements. The priority modes can be changed or reconfigured dynamically at any time during the main program. This means that the complete interrupt structure can be defined as required, based on the total system environment.



Polled Method



Interrupt Method

INTERRUPT REQUEST REGISTER (IRR) AND IN-SERVICE REGISTER (ISR)

The interrupts at the IR input lines are handled by two registers in cascade, the Interrupt Request Register (IRR) and the In-Service Register (ISR). The IRR is used to store all the interrupt levels which are requesting service; and the ISR is used to store all the interrupt levels which are being serviced.

PRIORITY RESOLVER

This logic block determines the priorities of the bits set in the IRR. The highest priority is selected and strobed into the corresponding bit of the ISR during $\overline{INTA}$ pulse.

INTERRUPT MASK REGISTER (IMR)

The IMR stores the bits which mask the interrupt lines to be masked. The IMR operates on the IRR. Masking of a higher priority input will not affect the interrupt request lines of lower priority.

INT (INTERRUPT)

This output goes directly to the CPU interrupt input. The V_{OH} level on this line is designed to be fully compatible with the 8080A, 8085A and 8086 input levels.

$\overline{INTA}$ (INTERRUPT ACKNOWLEDGE)

$\overline{INTA}$ pulses will cause the 8259A to release vectoring information onto the data bus. The format of this data depends on the system mode (μPM) of the 8259A.

DATA BUS BUFFER

This 3-state, bidirectional 8-bit buffer is used to interface the 8259A to the system Data Bus. Control words and status information are transferred through the Data Bus Buffer.

READ/WRITE CONTROL LOGIC

The function of this block is to accept OUTput commands from the CPU. It contains the Initialization Command Word (ICW) registers and Operation Command Word (OCW) registers which store the various control formats for device operation. This function block also allows the status of the 8259A to be transferred onto the Data Bus.

$\overline{CS}$ (CHIP SELECT)

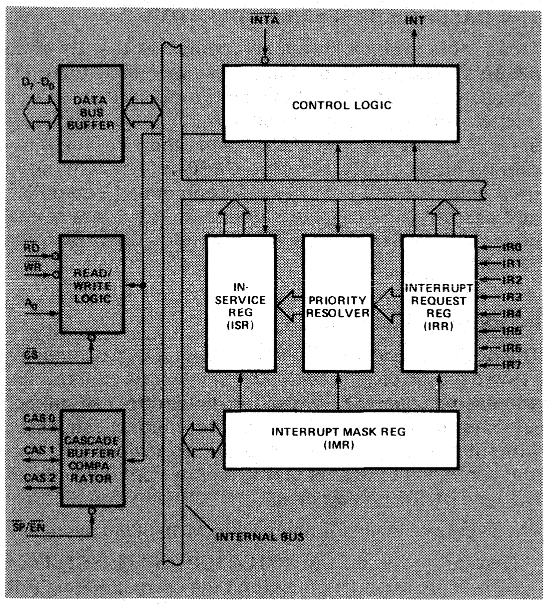
A LOW on this input enables the 8259A. No reading or writing of the chip will occur unless the device is selected.

$\overline{WR}$ (WRITE)

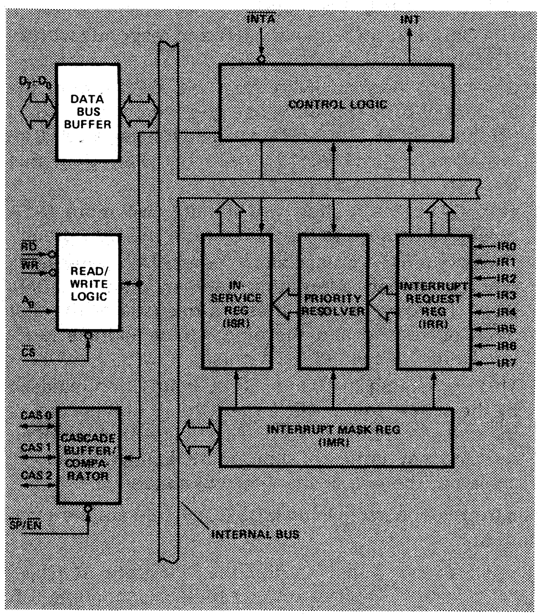
A LOW on this input enables the CPU to write control words (ICWs and OCWs) to the 8259A.

$\overline{RD}$ (READ)

A LOW on this input enables the 8259A to send the status of the Interrupt Request Register (IRR), In Service Register (ISR), the Interrupt Mask Register (IMR), or the Interrupt level onto the Data Bus.



8259A Block Diagram



8259A Block Diagram

A_0

This input signal is used in conjunction with $\overline{WR}$ and $\overline{RD}$ signals to write commands into the various command registers, as well as reading the various status registers of the chip. This line can be tied directly to one of the address lines.

THE CASCADE BUFFER/COMPARATOR

This function block stores and compares the IDs of all 8259A's used in the system. The associated three I/O pins (CAS0-2) are outputs when the 8259A is used as a master and are inputs when the 8259A is used as a slave. As a master, the 8259A sends the ID of the interrupting slave device onto the CAS0-2 lines. The slave thus selected will send its preprogrammed subroutine address onto the Data Bus during the next one or two consecutive $\overline{INTA}$ pulses. (See section "Cascading the 8259A".)

INTERRUPT SEQUENCE

The powerful features of the 8259A in a microcomputer system are its programmability and the interrupt routine addressing capability. The latter allows direct or indirect jumping to the specific interrupt routine requested without any polling of the interrupting devices. The normal sequence of events during an interrupt depends on the type of CPU being used.

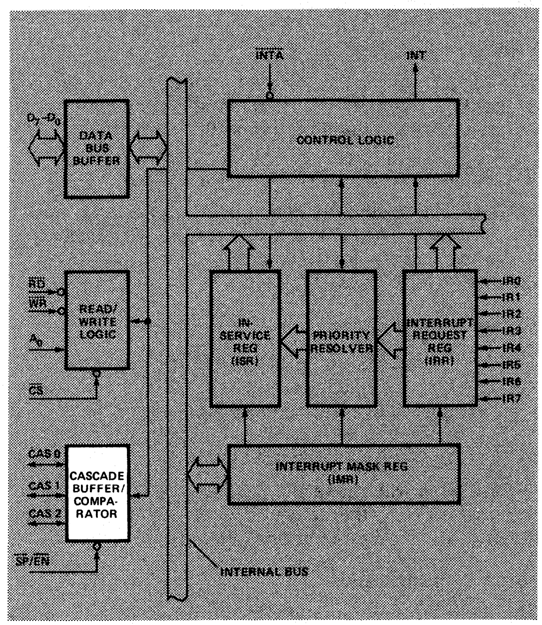
The events occur as follows in an MCS-80/85 system:

1. One or more of the INTERRUPT REQUEST lines (IR7-0) are raised high, setting the corresponding IRR bit(s).
2. The 8259A evaluates these requests, and sends an INT to the CPU, if appropriate.
3. The CPU acknowledges the INT and responds with an $\overline{INTA}$ pulse.
4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set, and the corresponding IRR bit is reset. The 8259A will also release a CALL instruction code (11001101) onto the 8-bit Data Bus through its D7-0 pins.
5. This CALL instruction will initiate two more $\overline{INTA}$ pulses to be sent to the 8259A from the CPU group.
6. These two $\overline{INTA}$ pulses allow the 8259A to release its preprogrammed subroutine address onto the Data Bus. The lower 8-bit address is released at the first $\overline{INTA}$ pulse and and the higher 8-bit address is released at the second $\overline{INTA}$ pulse.
7. This completes the 3-byte CALL instruction released by the 8259A. In the AEOI mode the ISR bit is reset at the end of the third $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt sequence.

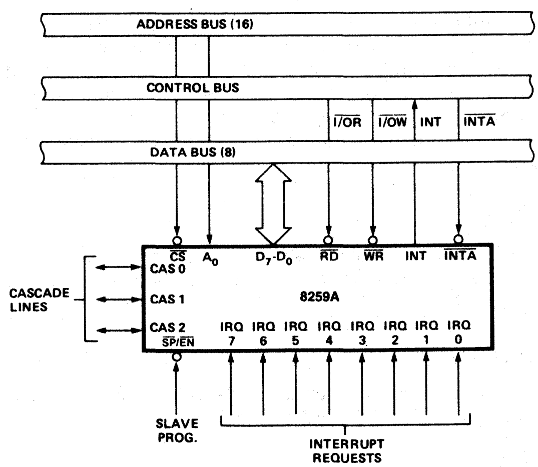
The events occurring in an MCS-86 system are the same until step 4.

4. Upon receiving an $\overline{INTA}$ from the CPU group, the highest priority ISR bit is set and the corresponding IRR bit is reset. The 8259A does not drive the Data Bus during this cycle.
5. The MCS-86 CPU will initiate a second $\overline{INTA}$ pulse. During this pulse, the 8259A releases an 8-bit pointer onto the Data Bus where it is read by the CPU.
6. This completes the interrupt cycle. In the AEOI mode the ISR bit is reset at the end of the second $\overline{INTA}$ pulse. Otherwise, the ISR bit remains set until an appropriate EOI command is issued at the end of the interrupt subroutine.

If no interrupt request is present at step 4 of either sequence (i.e., the request was too short in duration) the 8259A will issue an interrupt level 7. Both the vectoring bytes and the CAS lines will look like an interrupt level 7 was requested.



8259A Block Diagram



8259A Interface to Standard System Bus

INTERRUPT SEQUENCE OUTPUTS

MCS-80/85 SYSTEM

This sequence is timed by three $\overline{INTA}$ pulses. During the first $\overline{INTA}$ pulse the CALL opcode is enabled onto the data bus.

Content of First Interrupt Vector Byte

	D7	D6	D5	D4	D3	D2	D1	D0
CALL CODE	1	1	0	0	1	1	0	1

During the second $\overline{INTA}$ pulse the lower address of the appropriate service routine is enabled onto the data bus. When Interval = 4 bits A_5–A_7 are programmed, while A_0–A_4 are automatically inserted by the 8259A. When Interval = 8 only A_6 and A_7 are programmed, while A_0–A_5 are automatically inserted.

Content of Second Interrupt Vector Byte

IR	Interval = 4							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	A5	1	1	1	0	0
6	A7	A6	A5	1	1	0	0	0
5	A7	A6	A5	1	0	1	0	0
4	A7	A6	A5	1	0	0	0	0
3	A7	A6	A5	0	1	1	0	0
2	A7	A6	A5	0	1	0	0	0
1	A7	A6	A5	0	0	1	0	0
0	A7	A6	A5	0	0	0	0	0

IR	Interval = 8							
	D7	D6	D5	D4	D3	D2	D1	D0
7	A7	A6	1	1	1	0	0	0
6	A7	A6	1	1	0	0	0	0
5	A7	A6	1	0	1	0	0	0
4	A7	A6	1	0	0	0	0	0
3	A7	A6	0	1	1	0	0	0
2	A7	A6	0	1	0	0	0	0
1	A7	A6	0	0	1	0	0	0
0	A7	A6	0	0	0	0	0	0

During the third $\overline{INTA}$ pulse the higher address of the appropriate service routine, which was programmed as byte 2 of the initialization sequence (A_8–A_{15}), is enabled onto the bus.

Content of Third Interrupt Vector Byte

D7	D6	D5	D4	D3	D2	D1	D0
A15	A14	A13	A12	A11	A10	A9	A8

MCS-86 SYSTEM

MCS-86 mode is similar to MCS-80 mode except that only two Interrupt Acknowledge cycles are issued by the processor and no CALL opcode is sent to the processor. The first interrupt acknowledge cycle is similar to that of MCS-80/85 systems in that the 8259A uses it to internally freeze the state of the interrupts for priority resolution and as a master it issues the interrupt code on the cascade lines at the end of the $\overline{INTA}$ pulse. On this first cycle it does not issue any data to the processor and leaves its data bus buffers disabled. On the second interrupt acknowledge cycle in MCS-86 mode the master (or slave if so programmed) will send a byte of data to the processor with the acknowledged interrupt code composed as follows (note the state of the ADI mode control is ignored and A_5–A_{11} are unused in MCS-86 mode):

Content of Interrupt Vector Byte for MCS-86 System Mode

	D7	D6	D5	D4	D3	D2	D1	D0
IR7	A15	A14	A13	A12	A11	1	1	1
IR6	A15	A14	A13	A12	A11	1	1	0
IR5	A15	A14	A13	A12	A11	1	0	1
IR4	A15	A14	A13	A12	A11	1	0	0
IR3	A15	A14	A13	A12	A11	0	1	1
IR2	A15	A14	A13	A12	A11	0	1	0
IR1	A15	A14	A13	A12	A11	0	0	1
IR0	A15	A14	A13	A12	A11	0	0	0

PROGRAMMING THE 8259A

The 8259A accepts two types of command words generated by the CPU:

1. *Initialization Command Words (ICWs):* Before normal operation can begin, each 8259A in the system must be brought to a starting point — by a sequence of 2 to 4 bytes timed by $\overline{WR}$ pulses. This sequence is described in Figure 1.

2. *Operation Command Words (OCWs):* These are the command words which command the 8259A to operate in various interrupt modes. These modes are:
   a. Fully nested mode
   b. Rotating priority mode
   c. Special mask mode
   d. Polled mode

The OCWs can be written into the 8259A anytime after initialization.

INITIALIZATION

GENERAL

Whenever a command is issued with A0 = 0 and D4 = 1, this is interpreted as Initialization Command Word 1 (ICW1). ICW1 starts the initialization sequence during which the following automatically occur.

a. The edge sense circuit is reset, which means that following initialization, an interrupt request (IR) input must make a low-to-high transition to generate an interrupt.
b. The Interrupt Mask Register is cleared.
c. IR 7 input is assigned priority 7.
d. The slave mode address is set to 7.
e. Special Mask Mode is cleared and Status Read is set to IRR.
f. If IC4 = 0, then all functions selected in ICW4 are set to zero. (Non-Buffered mode*, no Auto-EOI, MCS-80/85 system).

***Note:** Master/Slave in ICW4 is only used in the buffered mode.

A_0	D_4	D_3	$\overline{RD}$	$\overline{WR}$	$\overline{CS}$	INPUT OPERATION (READ)
0			0	1	0	IRR, ISR or Interrupting Level → DATA BUS (Note 1)
1			0	1	0	IMR → DATA BUS
						OUTPUT OPERATION (WRITE)
0	0	0	1	0	0	DATA BUS → OCW2
0	0	1	1	0	0	DATA BUS → OCW3
0	1	X	1	0	0	DATA BUS → ICW1
1	X	X	1	0	0	DATA BUS → OCW1, ICW2, ICW3, ICW4 (Note 2)
						DISABLE FUNCTION
X	X	X	1	1	0	DATA BUS → 3-STATE
X	X	X	X	X	1	DATA BUS → 3-STATE

Notes: 1. Selection of IRR, ISR or Interrupting Level is based on the content of OCW3 written before the READ operation.
2. On-chip sequencer logic queues these commands into proper sequence.

8259A Basic Operation

INITIALIZATION COMMAND WORDS 1 AND 2 (ICW1, ICW2)

A_5–A_{15}: *Page starting address of service routines.* In an MCS 80/85 system, the 8 request levels will generate CALLs to 8 locations equally spaced in memory. These can be programmed to be spaced at intervals of 4 or 8 memory locations, thus the 8 routines will occupy a page of 32 or 64 bytes, respectively.

The address format is 2 bytes long (A_0–A_{15}). When the routine interval is 4, A_0–A_4 are automatically inserted by the 8259A, while A_5–A_{15} are programmed externally. When the routine interval is 8, A_0–A_5 are automatically inserted by the 8259A, while A_6–A_{15} are programmed externally.

The 8-byte interval will maintain compatibility with current software, while the 4-byte interval is best for a compact jump table.

In an MCS-86 system A_{15}–A_{11} are inserted in the five most significant bits of the vectoring byte and the 8259A sets the three least significant bits according to the interrupt level. A_{10}–A_5 are ignored and ADI (Address interval) has no effect.

LTIM: If LTIM = 1, then the 8259A will operate in the level interrupt mode. Edge detect logic on the interrupt inputs will be disabled.

ADI: CALL address interval. ADI = 1 then interval = 4; ADI = 0 then interval = 8.

SNGL: Single. Means that this is the only 8259A in the system. If SNGL = 1 no ICW3 will be issued.

IC4: If this bit is set — ICW4 has to be read. If ICW4 is not needed, set IC4 = 0.

INITIALIZATION COMMAND WORD 3 (ICW3)

This word is read only when there is more than one 8259A in the system and cascading is used, in which case SNGL = 0. It will load the 8-bit slave register. The functions of this register are:

a. In the master mode (either when $\overline{SP}$ = 1, or in buffered mode when M/S = 1 in ICW4) a "1" is set for each slave in the system. The master then will release byte 1 of the call sequence (for MCS-80/85 system) and will enable the corresponding slave to release bytes 2 and 3 (for MCS-86 only byte 2) through the cascade lines.

b. In the slave mode (either when $\overline{SP}$ = 0, or if BUF = 1 and M/S = 0 in ICW4) bits 2–0 identify the slave. The slave compares its cascade input with these bits and, if they are equal, bytes 2 and 3 of the call sequence (or just byte 2 for MCS-86) are released by it on the Data Bus.

INITIALIZATION COMMAND WORD 4 (ICW4)

SFNM: If SFNM = 1 the special fully nested mode is programmed.

BUF: If BUF = 1 the buffered mode is programmed. In buffered mode $\overline{SP}/\overline{EN}$ becomes an enable output and the master/slave determination is by M/S.

M/S: If buffered mode is selected: M/S = 1 means the 8259A is programmed to be a master, M/S = 0 means the 8259A is programmed to be a slave. If BUF = 0, M/S has no function.

AEOI: If AEOI = 1 the automatic end of interrupt mode is programmed.

μPM: Microprocessor mode: μPM = 0 sets the 8259A for MCS-80/85 system operation, μPM = 1 sets the 8259A for MCS-86 system operation.

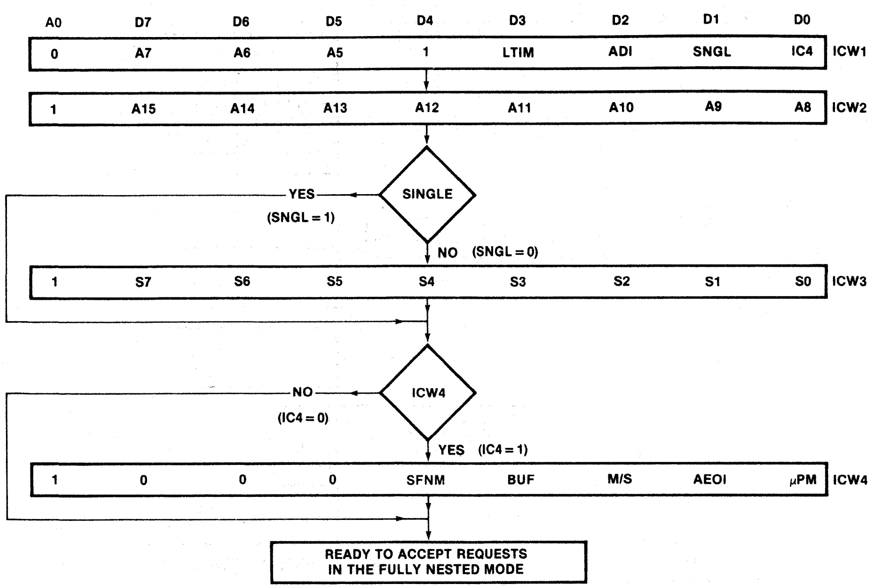
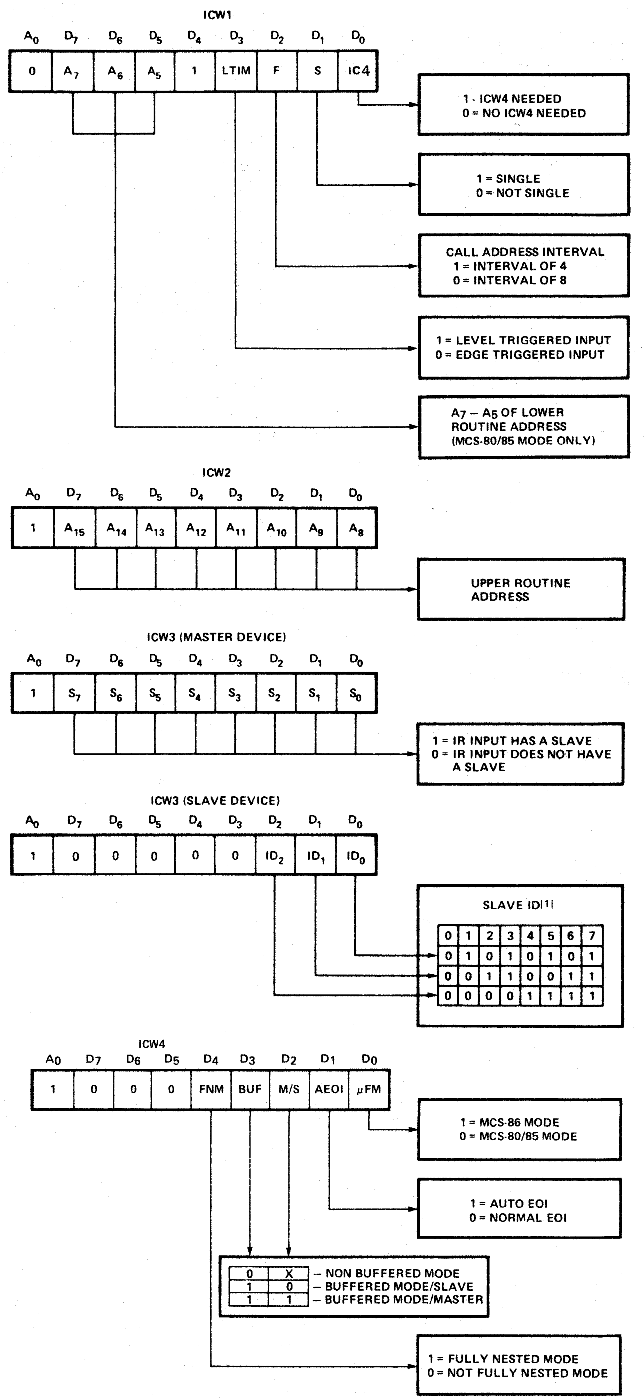


Figure 1. Initialization Sequence



NOTE 1: SLAVED ID IS EQUAL TO THE CORRESPONDING MASTER IR INPUT.
NOTE 2: X INDICATED "DON'T CARE".

Initialization Command Word Format

OPERATION COMMAND WORDS (OCWs)

After the Initialization Command Words (ICWs) are programmed into the 8259A, the chip is ready to accept interrupt requests at its input lines. However, during the 8259A operation, a selection of algorithms can command the 8259A to operate in various modes through the Operation Command Words (OCWs).

OPERATION CONTROL WORDS (OCWs)

OCW1

A0	D7	D6	D5	D4	D3	D2	D1	D0
1	M7	M6	M5	M4	M3	M2	M1	M0

OCW2

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	R	SEOI	EOI	0	0	L2	L1	L0

OCW3

A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	SSMM	SMM	0	1	P	SRIS	RIS

OPERATION CONTROL WORD 1 (OCW1)

OCW1 sets and clears the mask bits in the Interrupt Mask Register (IMR). M_7–M_0 represent the eight mask bits. M = 1 indicates the channel is masked (inhibited). M = 0 indicates the channel is enabled.

OPERATION CONTROL WORD 2 (OCW2)

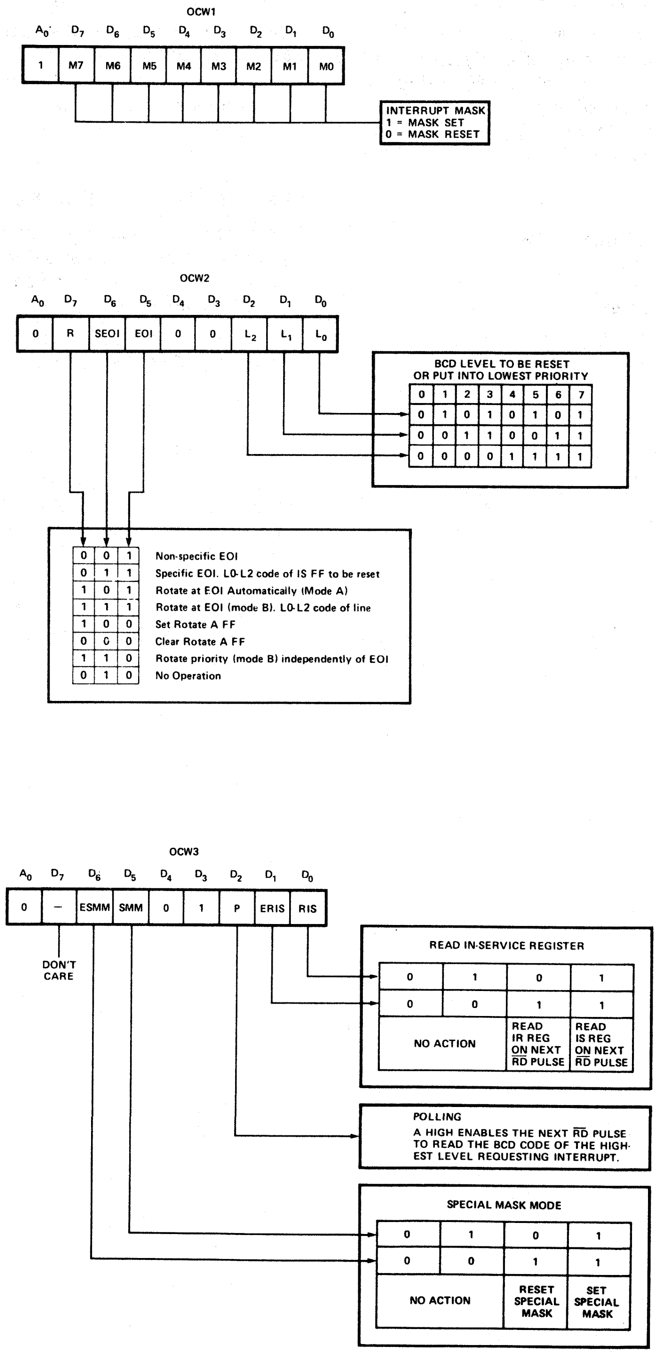
R, SEOI, EOI — These three bits control the Rotate and End of Interrupt modes and combinations of the two. A chart of these combinations can be found on the Operation Command Word Format.

L_2, L_1, L_0 — These bits determine the interrupt level acted upon when the SEOI bit is active.

OPERATION CONTROL WORD 3 (OCW3)

ESMM — Enable Special Mask Mode. When this bit is set to 1 it enables the SMM bit to set or reset the Special Mask Mode. When ESMM = 0 the SMM bit becomes a "don't care".

SMM — Special Mask Mode. If ESMM = 1 and SMM = 1 the 8259A will enter Special Mask Mode. If ESMM = 1 and SMM = 0 the 8259A will revert to normal mask mode. When ESMM = 0, SMM has no effect.



Operation Command Word Format

INTERRUPT MASKS

Each Interrupt Request input can be masked individually by the Interrupt Mask Register (IMR) programmed through OCW1. Each bit in the IMR masks one interrupt channel if it is set (1). Bit 0 masks IR0, Bit 1 masks IR1 and so forth. Masking an IR channel does not affect the other channels operation.

SPECIAL MASK MODE

Some applications may require an interrupt service routine to dynamically alter the system priority structure during its execution under software control. For example, the routine may wish to inhibit lower priority requests for a portion of its execution but enable some of them for another portion.

The difficulty here is that if an Interrupt Request is acknowledged and an End of Interrupt command did not reset its IS bit (i.e., while executing a service routine), the 8259A would have inhibited all lower priority requests with no easy way for the routine to enable them

That is where the Special Mask Mode comes in. In the special Mask Mode, when a mask bit is set in OCW1, it inhibits further interrupts at that level *and enables* interrupts from *all other* levels (lower as well as higher) that are not masked.

Thus, any interrupts may be selectively enabled by loading the mask register.

The special Mask Mode is set by OCW3 where: SSMM = 1, SMM = 1, and cleared where SSMM = 1, SMM = 0.

BUFFERED MODE

When the 8259A is used in a large system where bus driving buffers are required on the data bus and the cascading mode is used, there exists the problem of enabling buffers.

The buffered mode will structure the 8259A to send an enable signal on $\overline{SP}/\overline{EN}$ to enable the buffers. In this mode, whenever the 8259A's data bus outputs are enabled, the $\overline{SP}/\overline{EN}$ output becomes active.

This modification forces the use of software programming to determine whether the 8259A is a master or a slave. Bit 3 in ICW4 programs the buffered mode, and bit 2 in ICW4 determines whether it is a master or a slave.

NESTED MODE

This mode is entered after initialization unless another mode is programmed. The interrupt requests are ordered in priority from 0 through 7 (0 highest). When an interrupt is acknowledged the highest priority request is determined and its vector placed on the bus. Additionally, a bit of the Interrupt Service register (IS0-7) is set. This bit remains set until the microprocessor issues an End of Interrupt (EOI) command immediately before returning from the service routine, or if AEOI (Automatic End of Interrupt) bit is set, until the trailing edge of the last $\overline{INTA}$. While the IS bit is set, all further interrupts of the same or lower priority are inhibited, while higher levels will generate an interrupt (which will be acknowledged only if the microprocessor internal Interrupt enable flip-flop has been re-enabled through software).

After the initialization sequence, IR0 has the highest priority and IR7 the lowest. Priorities can be changed, as will be explained, in the rotating priority mode.

THE SPECIAL FULLY NESTED MODE

This mode will be used in the case of a big system where cascading is used, and the priority has to be conserved within each slave. In this case the fully nested mode will be programmed to the master (using ICW4). This mode is similar to the normal nested mode with the following exceptions:

a. When an interrupt request from a certain slave is in service this slave is not locked out from the master's priority logic and further interrupt requests from higher priority IR's within the slave will be recognized by the master and will initiate interrupts to the processor. (In the normal nested mode a slave is masked out when its request is in service and no higher requests from the same slave can be serviced.)

b. When exiting the Interrupt Service routine the software has to check whether the interrupt serviced was the only one from that slave. This is done by sending a non-specific End of Interrupt (EOI) command to the slave and then reading its In-Service register and checking for zero. If it is empty, a non-specific EOI can be sent to the master too. If not, no EOI should be sent.

POLL

In this mode the microprocessor internal Interrupt Enable flip-flop is reset, disabling its interrupt input. Service to devices is achieved by programmer initiative using a Poll command.

The Poll command is issued by setting P = "1" in OCW3. The 8259A treats the next $\overline{RD}$ pulse to the 8259A (i.e., $\overline{RD}$ = 0, $\overline{CS}$ = 0) as an interrupt acknowledge, sets the appropriate IS bit if there is a request, and reads the priority level. Interrupt is frozen from $\overline{WR}$ to $\overline{RD}$.

The word enabled onto the data bus during $\overline{RD}$ is:

D7	D6	D5	D4	D3	D2	D1	D0
I	—	—	—	—	W2	W1	W0

W0–W2: Binary code of the highest priority level requesting service.

I: Equal to a "1" if there is an interrupt.

This mode is useful if there is a routine command common to several levels so that the $\overline{INTA}$ sequence is not needed (saves ROM space). Another application is to use the poll mode to expand the number of priority levels to more than 64.

END OF INTERRUPT (EOI)

The In Service (IS) bit can be reset either automatically following the trailing edge of the last in sequence $\overline{INTA}$ pulse (when AEOI bit in ICW1 is set) or by a command word that must be issued to the 8259A before returning from a service routine (EOI command). An EOI command must be issued twice, once for the master and once for the corresponding slave if slaves are in use.

There are two forms of EOI command: Specific and Non-Specific. When the 8259A is operated in modes which preserve the fully nested structure, it can determine which IS bit to reset on EOI. When a Non-Specific EOI command is issued the 8259A will automatically reset the highest IS bit of those that are set, since in the nested mode the highest IS level was necessarily the last level acknowledged and serviced.

However, when a mode is used which may disturb the fully nested structure, the 8259A may no longer be able to determine the last level acknowledged. In this case a Specific End of Interrupt (SEOI) must be issued which includes as part of the command the IS level to be reset. EOI is issued whenever E = 1, in OCW2, where L0–L2 is the binary level of the IS bit to be reset. Note that although the Rotate command can be issued together with an EOI where E = 1, it is not necessarily tied to it.

It should be noted that an IS bit that is masked by an IMR bit will not be cleared by a non-specific EOI if the 8259A is in the Special Mask Mode.

AUTOMATIC END OF INTERRUPT (AEOI) MODE

If AEOI = 1 in ICW4, then the 8259A will operate in AEOI mode continuously until reprogrammed by ICW4. In this mode the 8259A will automatically perform a non-specific EOI operation at the trailing edge of the last interrupt acknowledge pulse (third pulse in MCS-80/85, second in MCS-86). Note that from a system standpoint, this mode should be used only when a nested multilevel interrupt structure is not required within a single 8259A.

To achieve automatic rotation (Rotate Mode A) within AEOI, there is a special rotate flip-flop. It is set by OCW2 with R = 1, SEOI = *0*, E = 0, and cleared with R = 0, SEOI = 0, E = 0.

ROTATING PRIORITY MODE A (AUTOMATIC ROTATION) FOR EQUAL PRIORITY DEVICES

In some applications there are a number of interrupting devices of equal priority. In this mode a device, after being serviced, receives the lowest priority, so a device requesting an interrupt will have to wait, in the worst case until each of 7 other devices are serviced at most *once*. For example, if the priority and "in service" status is:

Before Rotate (IR4 the highest priority requiring service)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	1	0	0	0	0
	Lowest Priority							Highest Priority
Priority Status	7	6	5	4	3	2	1	0

After Rotate (IR4 was serviced, all other priorities rotated correspondingly)

	IS7	IS6	IS5	IS4	IS3	IS2	IS1	IS0
"IS" Status	0	1	0	0	0	0	0	0
			Highest Priority		Lowest Priority			
Priority Status	2	1	0	7	6	5	4	3

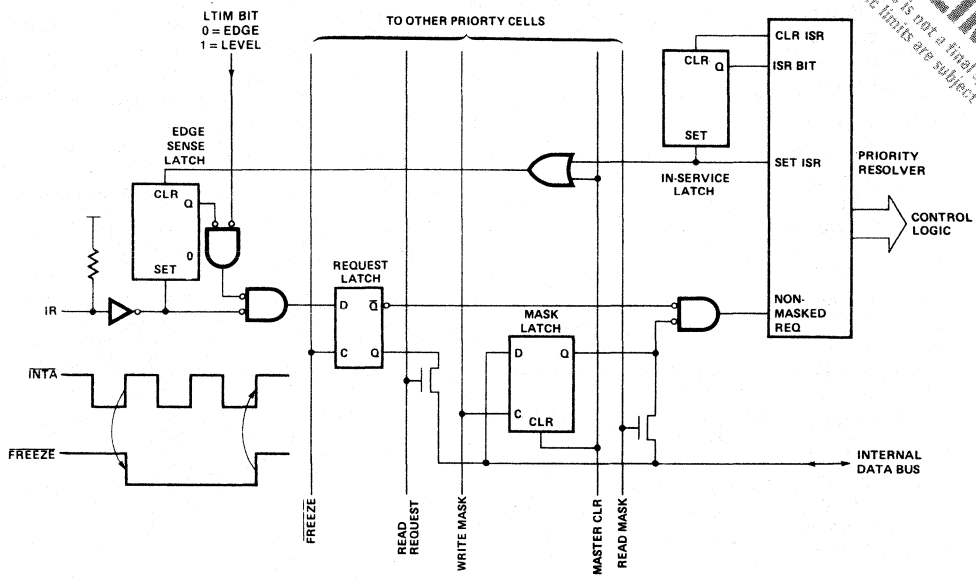
The Rotate command mode A is issued in OCW2 where: R = 1, E = 1, SEOI = 0. Internal status is updated by an End of Interrupt (EOI or AEOI) command. If R = 1, E = 0, SEOI = 0, a "Rotate-A" flip-flop is set. This is useful in AEOI, and described under Automatic End of Interrupt.

ROTATING PRIORITY MODE B (ROTATION BY SOFTWARE)

The programmer can change priorities by programming the bottom priority and thus fixing all other priorities; i.e., if IR5 is programmed as the bottom priority device, then IR6 will have the highest one.

The Rotate command is issued in OCW2 where: R = 1, SEOI = 1; L0–L2 is the binary priority level code of the bottom priority device.

Observe that in this mode internal status is updated by software control during OCW2. However, it is independent of the End of Interrupt (EOI) command (also executed by OCW2). Priority changes can be executed during an EOI command or independently.



NOTES
1. MASTER CLEAR ACTIVE ONLY DURING ICW1
2. FREEZE/ IS ACTIVE DURING $\overline{\text{INTA}}$/ AND POLL SEQUENCES ONLY
3. TRUTH TABLE FOR D-LATCH

C	D	Q	OPERATION
1	Di	Di	FOLLOW
0	X	Qn−1	HOLD

Priority Cell

LEVEL TRIGGERED MODE

This mode is programmed using bit 3 in ICW1.

If LTM = '1', an interrupt request will be recognized by a 'high' level on IR Input, and there is no need for an edge detection. The interrupt request must be removed before the EOI command is issued or the CPU interrupt is enabled to prevent a second interrupt from occurring.

The above figure shows a conceptual circuit to give the reader an understanding of the level sensitive and edge sensitive input circuitry of the 8259A. Be sure to note that the request latch is a transparent D type latch.

READING THE 8259A STATUS

The input status of several internal registers can be read to update the user information on the system. The following registers can be read by issuing a suitable OCW3 and reading with $\overline{\text{RD}}$.

Interrupt Mask Register: 8-bit register whose content specifies the interrupt request lines being masked. acknowledged. The highest request level is reset from the IRR when an interrupt is acknowledged. (Not affected by IMR.)

In-Service Register (ISR): 8-bit register which contains the priority levels that are being serviced. The ISR is updated when an End of Interrupt command is issued.

Interrupt Mask Register: 8-bit register which contains the interrupt request lines which are masked.

The IRR can be read when, prior to the $\overline{\text{RD}}$ pulse, a $\overline{\text{WR}}$ pulse is issued with OCW3 (ERIS = 1, RIS = 0.)

The ISR can be read in a similar mode when ERIS = 1, RIS = 1 in the OCW3.

There is no need to write an OCW3 before every status read operation, as long as the status read corresponds with the previous one; i.e., the 8259A "remembers" whether the IRR or ISR has been previously selected by the OCW3.

After initialization the 8259A is set to IRR.

For reading the IMR, no OCW3 is needed. The output data bus will contain the IMR whenever $\overline{\text{RD}}$ is active and A0 = 1.

Polling overrides status read when P = 1, ERIS = 1 in OCW3.

SUMMARY OF 8259A INSTRUCTION SET

Inst. #	Mnemonic	A0	D7	D6	D5	D4	D3	D2	D1	D0	Operation Description
1	ICW1 A	0	A7	A6	A5	1	0	1	1	0	Format = 4, single, edge triggered
2	ICW1 B	0	A7	A6	A5	1	1	1	1	0	Format = 4, single, level triggered
3	ICW1 C	0	A7	A6	A5	1	0	1	0	0	Byte 1 Initialization — Format = 4, not single, edge triggered
4	ICW1 D	0	A7	A6	A5	1	1	1	0	0	Format = 4, not single, level triggered
5	ICW1 E	0	A7	A6	0	1	0	0	1	0	No ICW4 Required — Format = 8, single, edge triggered
6	ICW1 F	0	A7	A6	0	1	1	0	1	0	Format = 8, single, level triggered
7	ICW1 G	0	A7	A6	0	1	0	0	0	0	Format = 8, not single, edge triggered
8	ICW1 H	0	A7	A6	0	1	1	0	0	0	Format = 8, not single, level triggered
9	ICW1 I	0	A7	A6	A5	1	0	1	1	1	Format = 4, single, edge triggered
10	ICW1 J	0	A7	A6	A5	1	1	1	1	1	Format = 4, single, level triggered
11	ICW1 K	0	A7	A6	A5	1	0	1	0	1	Byte 1 Initialization — Format = 4, not single, edge triggered
12	ICW1 L	0	A7	A6	A5	1	1	1	0	1	Format = 4, not single, level triggered
13	ICW1 M	0	A7	A6	0	1	0	0	1	1	ICW4 Required — Format = 8, single, edge triggered
14	ICW1 N	0	A7	A6	0	1	1	0	1	1	Format = 8, single, level triggered
15	ICW1 O	0	A7	A6	0	1	0	0	0	1	Format = 8, not single, edge triggered
16	ICW1 P	0	A7	A6	0	1	1	0	0	1	Format = 8, not single, level triggered
17	ICW2	1	A15	A14	A13	A12	A11	A10	A9	A8	Byte 2 initialization
18	ICW3 M	1	S7	S6	S5	S4	S3	S2	S1	S0	Byte 3 initialization — master
19	ICW3 S	1	0	0	0	0	0	S2	S1	S0	Byte 3 initialization — slave
20	ICW4 A	1	0	0	0	0	0	0	0	0	No action, redundant
21	ICW4 B	1	0	0	0	0	0	0	0	1	Non-buffered mode, no AEOI, MCS-86
22	ICW4 C	1	0	0	0	0	0	0	1	0	Non-buffered mode, AEOI, MCS-80/85
23	ICW4 D	1	0	0	0	0	0	0	1	1	Non-buffered mode, AEOI, MCS-86
24	ICW4 E	1	0	0	0	0	0	1	0	0	No action, redundant
25	ICW4 F	1	0	0	0	0	0	1	0	1	Non-buffered mode, no AEOI, MCS-86
26	ICW4 G	1	0	0	0	0	0	1	1	0	Non-buffered mode, AEOI, MCS-80/85
27	ICW4 H	1	0	0	0	0	0	1	1	1	Non-buffered mode, AEOI, MCS-86
28	ICW4 I	1	0	0	0	0	1	0	0	0	Buffered mode, slave, no AEOI, MCS-80/85
29	ICW4 J	1	0	0	0	0	1	0	0	1	Buffered mode, slave, no AEOI, MCS-86
30	ICW4 K	1	0	0	0	0	1	0	1	0	Buffered mode, slave, AEOI, MCS-80/85
31	ICW4 L	1	0	0	0	0	1	0	1	1	Buffered mode, slave, AEOI, MCS-86
32	ICW4 M	1	0	0	0	0	1	1	0	0	Buffered mode, master, no AEOI, MCS-80/85
33	ICW4 N	1	0	0	0	0	1	1	0	1	Buffered mode, master, no AEOI, MCS-86
34	ICW4 O	1	0	0	0	0	1	1	1	0	Buffered mode, master, AEOI, MCS-80/85
35	ICW4 P	1	0	0	0	0	1	1	1	1	Buffered mode, master, AEOI, MCS-86
36	ICW4 NA	1	0	0	0	1	0	0	0	0	Fully nested mode, MCS-80, non-buffered, no AEOI
37	ICW4 NB	1	0	0	0	1	0	0	0	1	ICW4 NB through ICW4 ND are identical to ICW4 B through ICW4 D with the addition of Fully Nested Mode
38	ICW4 NC	1	0	0	0	1	0	0	1	0	
39	ICW4 ND	1	0	0	0	1	0	0	1	1	
40	ICW4 NE	1	0	0	0	1	0	1	0	0	Fully Nested Mode, MCS-80/85, non-buffered, no AEOI
41	ICW4 NF	1	0	0	0	1	0	1	0	1	ICW4 NF through ICW4 NP are identical to ICW4 F through ICW4 P with the addition of Fully Nested Mode
42	ICW4 NG	1	0	0	0	1	0	1	1	0	
43	ICW4 NH	1	0	0	0	1	0	1	1	1	
44	ICW4 NI	1	0	0	0	1	1	0	0	0	
45	ICW4 NJ	1	0	0	0	1	1	0	0	1	
46	ICW4 NK	1	0	0	0	1	1	0	1	0	
47	ICW4 NL	1	0	0	0	1	1	0	1	1	
48	ICW4 NM	1	0	0	0	1	1	1	0	0	
49	ICW4 NN	1	0	0	0	1	1	1	0	1	
50	ICW4 NO	1	0	0	0	1	1	1	1	0	
51	ICW4 NP	1	0	0	0	1	1	1	1	1	
36	OCW1	1	M7	M6	M5	M4	M3	M2	M1	M0	Load mask register, read mask register
37	OCW2 E	0	0	0	1	0	0	0	0	0	Non-specific EOI
38	OCW2 SE	0	0	1	1	0	0	L2	L1	L0	Specific EOI. L0-L2 code of IS FF to be reset
39	OCW2 RE	0	1	0	1	0	0	0	0	0	Rotate at EOI Automatically (Mode A)
40	OCW2 RSE	0	1	1	1	0	0	L2	L1	L0	Rotate at EOI (mode B). L0-L2 code of line
41	OCW2 R	0	1	0	0	0	0	0	0	0	Set Rotate A FF
42	OCW2 CR	0	0	0	0	0	0	0	0	0	Clear Rotate A FF
43	OCW2 RS	0	1	1	0	0	0	L2	L1	L0	Rotate priority (mode B) independently of EOI
44	OCW3 P	0	0	0	0	0	1	1	0	0	Poll mode
45	OCW3 RIS	0	0	0	0	0	1	0	1	1	Read IS register

SUMMARY OF 8259A INSTRUCTION SET (Cont.)

Inst. #	Mnemonic	A0	D7	D6	D5	D4	D3	D2	D1	D0	Operation Description
46	OCW3 RR	0	0	0	0	0	1	0	1	0	Read request register
47	OCW3 SM	0	0	1	1	0	1	0	0	0	Set special mask mode
48	OCW3 RSM	0	0	1	0	0	1	0	0	0	Reset special mask mode

Note: 1. In the master mode SP pin = 1, in slave mode SP = 0

Cascading

The 8259A can be easily interconnected in a system of one master with up to eight slaves to handle up to 64 priority levels.

A typical MCS-80/85 system is shown in Figure 2. The master controls, through the 3 line cascade bus, which one of the slaves will release the corresponding address.

As shown in Figure 2, the slave interrupt outputs are connected to the master interrupt request inputs. When a slave request line is activated and afterwards acknowledged, the master will enable the corresponding slave to release the device routine address during bytes 2 and 3 of $\overline{INTA}$. (Byte 2 only for MCS-86).

The cascade bus lines are normally low and will contain the slave address code from the trailing edge of the first $\overline{INTA}$ pulse to the trailing edge of the third pulse. It is obvious that each 8259A in the system must follow a separate initialization sequence and can be programmed to work in a different mode. An EOI command must be issued twice: once for the master and once for the corresponding slave. An address decoder is required to activate the Chip Select ($\overline{CS}$) input of each 8259A.

The cascade lines of the Master 8259A are activated for any interrupt input, even if no slave is connected to that input.

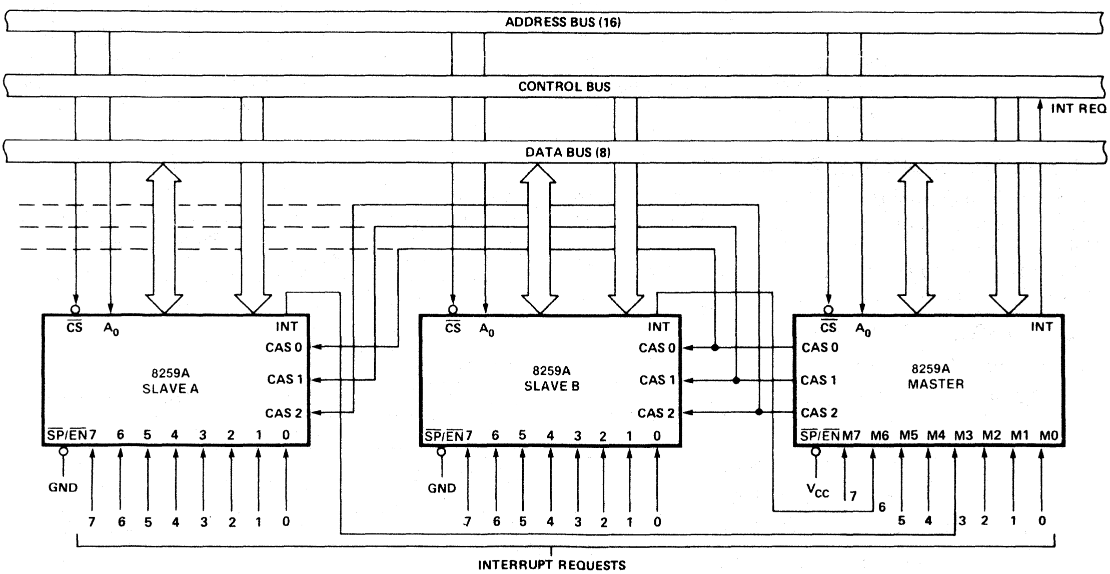


Figure 2. Cascading the 8259A

PIN FUNCTIONS

Name	I/O	Pin #	Function
V_{CC}		28	+5V supply.
GND		14	Ground.
D_{0-7}	I/O	11-4	Bidirectional data bus, used for: a) programming the mode of the 8259A (programming is done by software); b) the microprocessor can read the status of the 8259A; c) the 8259A will send vectoring data to the microprocessor when an interrupt is acknowledged.
IR_{0-7}	I	18-25	Interrupt Requests: These are asynchronous inputs. A positive-going edge will generate an interrupt request. Thus a request can be generated by raising the line and holding it high until acknowledged, or by a negative pulse. In level triggered mode, no edge is required. These lines are active HIGH.
$\overline{RD}$	I	3	Read (generally from 8228 in MCS-80 system or from 8086 in MCS-86 system).
$\overline{WR}$	I	2	Write (generally from 8228 in MCS-80 sytem or from 8086 in MCS-86 system).
$\overline{INTA}$	I	26	Interrupt Acknowledge (generally from 8228 in MCS-80 system, 8086 in MCS-86 system). The 8288 generates three distinct $\overline{INTA}$ pulses when a CALL is inserted, the 8086 produces two distinct $\overline{INTA}$ pulses during an interrupt cycle.
$\overline{CS}$	I	1	Chip Select: $\overline{RD}$ and $\overline{WR}$ are enabled by Chip Select, whereas Interrupt Acknowledge is independent of Chip Select.
A0	I	27	Usually the least significant bit of the microprocessor address output. When A0 = 1 the Interrupt Mask Register can be loaded or read. When A0 = 0 the 8259A mode can be programmed or its status can be read. $\overline{CS}$ is active LOW.
INT	O	17	Goes directly to the microprocessor interrrupt input. This output will have high V_{OH} to match the 8080 3.3V V_{IH}. INT is active HIGH.
C0-C2	I/O	12 13 15	Three cascade lines, outputs in master mode and inputs in slave mode. The master issues the binary code of the acknowledged interrupt level on these lines. Each slave compares this code with its own.
$\overline{SP}/\overline{EN}$	I/O	16	$\overline{SP}/\overline{EN}$ is a dual function pin. In the buffered mode $\overline{SP}/\overline{EN}$ is used to enable bus transceivers ($\overline{EN}$). In the non-buffered mode $\overline{SP}/\overline{EN}$ determines if this 8259A is a master or a slave. If $\overline{SP}$ = 1 the 8259A is master; $\overline{SP}$ = 0 indicates a slave.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias ..... −40°C to 85°C
Storage Temperature .... −65°C to +150°C
Voltage On Any Pin
With Respect to Ground .... −0.5V to +7V
Power Dissipation .... 1 Watt

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

D.C. CHARACTERISTICS

T_A = 0°C to 70°C V_{CC} = 5V ± 5% (8259A-8) V_{CC} = 5V ± 10% (8259A)

Symbol	Parameter	Min.	Max.	Units	Test Conditions
V_{IL}	Input Low Voltage	−.5	.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC} + .5V	V	
V_{OL}	Output Low Voltage		.45	V	I_{OL} = 2.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −400 μA
$V_{OH(INT)}$	Interrupt Output High Voltage	3.5 2.4		V V	I_{OH} = −100 μA I_{OH} = −400 μA
I_{LI}	Input Load Current		10	μA	V_{IN} = V_{CC} to 0V
I_{LOL}	Output Leakage Current		−10	μA	V_{OUT} = 0.45V
I_{LOH}	Output Leakage Current		10	μA	V_{OUT} = V_{CC}
I_{CC}	V_{CC} Supply Current		85	mA	

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8259A A.C. CHARACTERISTICS

$T_A = 0°C$ to $70°C$ $V_{CC} = 5V \pm 5\%$ (8259A-8) $V_{CC} = 5V \pm 10\%$ (8259A)

TIMING REQUIREMENTS

Symbol	Parameter	8259A-8 Min.	8259A-8 Max.	8259A Min.	8259A Max.	Units	Test Conditions
TAHRL	A0/$\overline{CS}$ Setup to $\overline{RD}/\overline{INTA}\downarrow$	50		0		ns	
TRHAX	A0/$\overline{CS}$ Hold after $\overline{RD}/\overline{INTA}\uparrow$	5		0		ns	
TRLRH	$\overline{RD}$ Pulse Width	420		235		ns	
TAHWL	A0/$\overline{CS}$ Setup to $\overline{WR}\downarrow$	50		0		ns	
TWHAX	A0/$\overline{CS}$ Hold after $\overline{WR}\uparrow$	20		0		ns	
TWLWH	$\overline{WR}$ Pulse Width	400		290		ns	
TDVWH	Data Setup to $\overline{WR}\uparrow$	300		240		ns	
TWHDX	Data Hold after $\overline{WR}\uparrow$	40		0		ns	
TJLJH	Interrupt Request Width (Low)	100		100		ns	See Note 1
TCVIAH	Cascade Setup to Second or Third $\overline{INTA}\downarrow$ (Slave Only)	55		55		ns	
TRHRL	End of $\overline{RD}$ to Next Command	160		160		ns	
TWHRL	End of $\overline{WR}$ to Next Command	190		190		ns	

Note: 1. This is the low time required to clear the input latch in the edge triggered mode.

TIMING RESPONSES

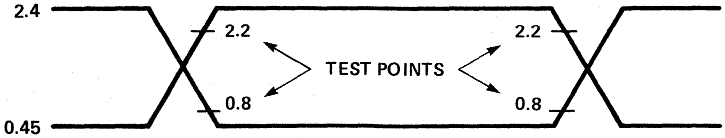
Symbol	Parameter	8259A-8 Min.	8259A-8 Max.	8259A Min.	8259A Max.	Units	Test Conditions
TRLDV	Data Valid from $\overline{RD}/\overline{INTA}\downarrow$		300		200	ns	C of Data Bus = 100 pF
TRHDZ	Data Float after $\overline{RD}/\overline{INTA}\uparrow$	20	200		100	ns	C of Data Bus
TJHIH	Interrupt Output Delay		400		350	ns	Max. test C = 100 pF Min. test C = 15 pF
TIAHCV	Cascade Valid from First $\overline{INTA}\downarrow$ (Master Only)		565		565	ns	C_{INT} = 100 pF
TRLEL	Enable Active from $\overline{RD}\downarrow$ or $\overline{INTA}\downarrow$		160		125	ns	$C_{CASCADE}$ = 100 pF
TRHEH	Enable Inactive from $\overline{RD}\uparrow$ or $\overline{INTA}\uparrow$		325		150	ns	
TAHDV	Data Valid from Stable Address		350		200	ns	
TCVDV	Cascade Valid to Valid Data		300		300	ns	

CAPACITANCE

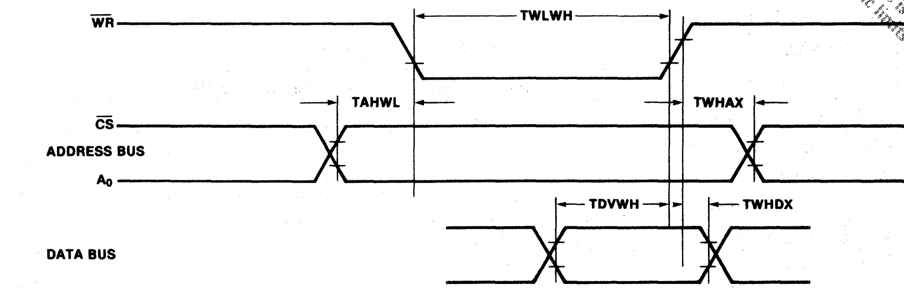
$T_A = 25°C$; $V_{CC} = GND = 0V$

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Conditions
C_{IN}	Input Capacitance			10	pF	fc = 1 MHz
$C_{I/O}$	I/O Capacitance			20	pF	Unmeasured pins returned to V_{SS}

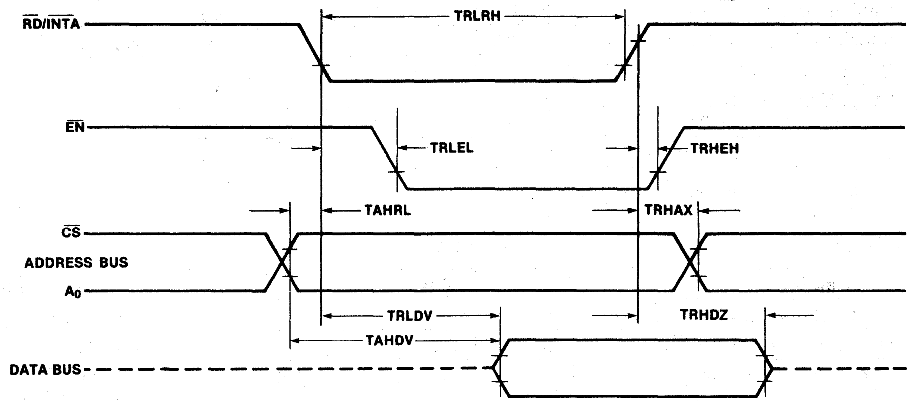
Input Waveforms for A.C. Tests



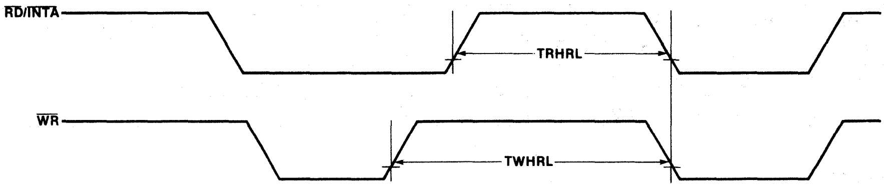
WRITE MODE



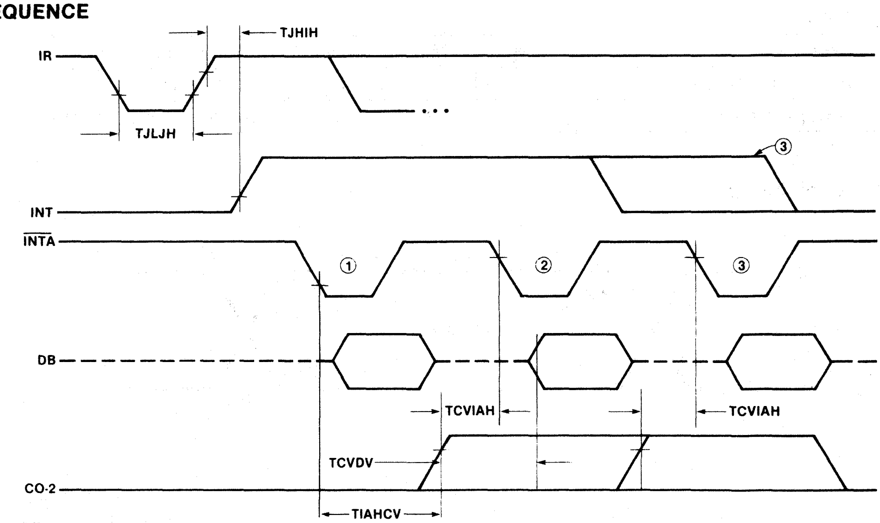
READ/INTA MODE



OTHER TIMING



INTA SEQUENCE



NOTE: Interrupt output must remain "HIGH" (at least) until leading edge of first INTA.
③ MCS 80/85 Systems only.
① Cycle 1 in MCS 86 Systems, the Data Bus is not active.

CHAPTER 5

Device Specifications

- MCS-86™
- **MCS-85™***
- Peripherals**
- Static RAMs***
- ROMs/EPROMs***

*For complete specifications refer to the Intel MCS-85 User's Manual.

**For complete specifications refer to the Intel Peripheral Design Handbook.

***For complete specifications refer to the 1978 Intel Data Catalog.

intel®

8085A/8085A-2
SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSORS

- **Single +5V Power Supply**
- **100% Software Compatible with 8080A**
- **1.3 μs Instruction Cycle (8085A); 0.8 μs (8085A-2)**
- **On-Chip Clock Generator (with External Crystal, LC or RC Network)**
- **On-Chip System Controller; Advanced Cycle Status Information Available for Large System Control**
- **Four Vectored Interrupt Inputs (One is non-Maskable) Plus an 8080A-compatible interrupt**
- **Serial In/Serial Out Port**
- **Decimal, Binary and Double Precision Arithmetic**
- **Direct Addressing Capability to 64k Bytes of Memory**

The Intel® 8085A is a complete 8 bit parallel Central Processing Unit (CPU). Its instruction set is 100% software compatible with the 8080A microprocessor, and it is designed to improve the present 8080A's performance by higher system speed. Its high level of system integration allows a minimum system of three IC's [8085A (CPU), 8156 (RAM/IO) and 8355/8755A (ROM/PROM/IO)] while maintaining total system expandability. The 8085A-2 is a faster version of the 8085A.

The 8085A incorporates all of the features that the 8224 (clock generator) and 8228 (system controller) provided for the 8080A, thereby offering a high level of system integration.

The 8085A uses a multiplexed data bus. The address is split between the 8 bit address bus and the 8 bit data bus. The on-chip address latches of 8155/8156/8355/8755A memory products allow a direct interface with the 8085A.

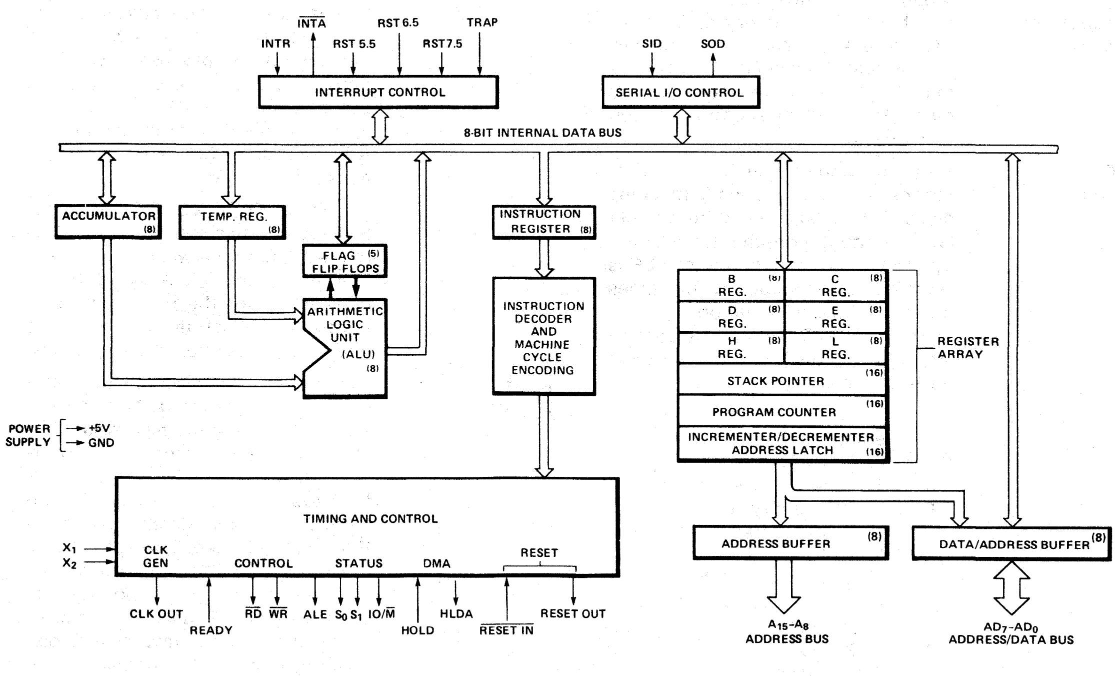


Figure 1. 8085A CPU Functional Block Diagram

July, 1978

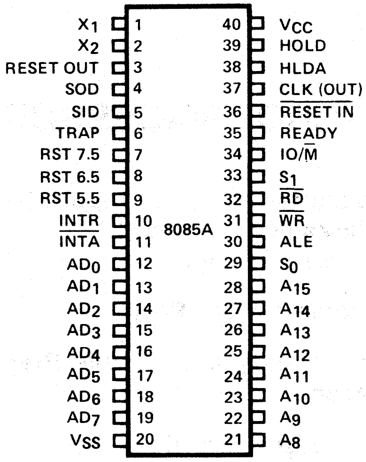


Figure 2. 8085A Pinout Diagram

8085A FUNCTIONAL PIN DEFINITION

The following describes the function of each pin:

Symbol	Function
A_8–A_{15} (Output, 3-state)	Address Bus: The most significant 8 bits of the memory address or the 8 bits of the I/O address, 3-stated during Hold and Halt modes and during RESET.
AD_{0-7} (Input/Output, 3-state)	Multiplexed Address/Data Bus: Lower 8 bits of the memory address (or I/O address) appear on the bus during the first clock cycle (T state) of a machine cycle. It then becomes the data bus during the second and third clock cycles.
ALE (Output)	Address Latch Enable: It occurs during the first clock state of a machine cycle and enables the address to get latched into the on-chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. The falling edge of ALE can also be used to strobe the status information. ALE is never 3-stated.
S_0, S_1, and IO/$\overline{M}$ (Output)	Machine cycle status: (see table below)

IO/$\overline{M}$	S_1	S_0	Status
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	Opcode fetch
1	1	1	Interrupt Acknowledge
*	0	0	Halt
*	X	X	Hold
*	X	X	Reset

* = 3-state (high impedance)
X = unspecified

Symbol	Function
	S_1 can be used as an advanced R/$\overline{W}$ status. IO/$\overline{M}$, S_0 and S_1 become valid at the beginning of a machine cycle and remain stable throughout the cycle. The falling edge of ALE may be used to latch the state of these lines.
$\overline{RD}$ (Output, 3-state)	READ control: A low level on $\overline{RD}$ indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer, 3-stated during Hold and Halt modes and during RESET.
$\overline{WR}$ (Output, 3-state)	WRITE control: A low level on $\overline{WR}$ indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set up at the trailing edge of $\overline{WR}$. 3-stated during Hold and Halt modes and during RESET.
READY (Input)	If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data. If READY is low, the cpu will wait an integral number of clock cycles for READY to go high before completing the read or write cycle.
HOLD (Input)	HOLD indicates that another master is requesting the use of the address and data buses. The cpu, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer. Internal processing can continue. The processor can regain the bus only after the HOLD is removed. When the HOLD is acknowledged, the Address, Data, $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ lines are 3-stated.
HLDA (Output)	HOLD ACKNOWLEDGE: Indicates that the cpu has received the HOLD request and that it will relinquish the bus in the next clock cycle. HLDA goes low after the Hold request is removed. The cpu takes the bus one half clock cycle after HLDA goes low.
INTR (Input)	INTERRUPT REQUEST: is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of an instruction and during Hold and Halt states. If it is active, the Program Counter (PC) will be inhibited from incrementing and an $\overline{INTA}$ will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

8085A FUNCTIONAL PIN DESCRIPTION (Continued)

Symbol	Function
$\overline{\text{INTA}}$ (Output)	INTERRUPT ACKNOWLEDGE: Is used instead of (and has the same timing as) $\overline{\text{RD}}$ during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.
RST 5.5 RST 6.5 RST 7.5 (Inputs)	RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. The priority of these interrupts is ordered as shown in Table 1. These interrupts have a higher priority than INTR. In addition, they may be individually masked out using the SIM instruction.
TRAP (Input)	Trap interrupt is a nonmaskable RESTART interrupt. It is recognized at the same time as INTR or RST 5.5-7.5. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt. (See Table 1.)
$\overline{\text{RESET IN}}$ (Input)	Sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. The data and address buses and the control lines are 3-stated during RESET and because of the asynchronous nature of RESET, the processor's internal registers and flags may be altered by RESET with unpredictable results. $\overline{\text{RESET IN}}$ is a Schmitt-triggered input, allowing connection to an R-C network for power-on RESET delay. The cpu is held in the reset condition as long as $\overline{\text{RESET IN}}$ is applied.
RESET OUT (Output)	Indicates cpu is being reset. Can be used as a system reset. The signal is synchronized to the processor clock and lasts an integral number of clock periods.
X_1, X_2 (Input)	X_1 and X_2 are connected to a crystal, LC, or RC network to drive the internal clock generator. X_1 can also be an external clock input from a logic gate. The input frequency is divided by 2 to give the processor's internal operating frequency.
CLK (Output)	Clock Output for use as a system clock. The period of CLK is twice the X_1, X_2 input period.
SID (Input)	Serial input data line. The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
SOD (Output)	Serial output data line. The output SOD is set or reset as specified by the SIM instruction.
V_{CC}	+5 volt supply.
V_{SS}	Ground Reference.

TABLE 1. INTERRUPT PRIORITY, RESTART ADDRESS, AND SENSITIVITY

Name	Priority	Address Branched To (1) When Interrupt Occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled.
RST 7.5	2	3CH	Rising edge (latched).
RST 6.5	3	34H	High level until sampled.
RST 5.5	4	2CH	High level until sampled.
INTR	5	See Note (2).	High level until sampled.

NOTES:

(1) The processor pushes the PC on the stack before branching to the indicated address.
(2) The address branched to depends on the instruction provided to the cpu when the interrupt is acknowledged.

FUNCTIONAL DESCRIPTION

The 8085A is a complete 8-bit parallel central processor. It is designed with N-channel depletion loads and requires a single +5 volt supply. Its basic clock speed is 3 MHz (8085A) or 5 MHz (8085A-2), thus improving on the present 8080A's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The cpu (8085A), a RAM/IO (8156), and a ROM or EPROM/IO chip (8355 or 8755A).

The 8085A has twelve addressable 8-bit registers. Four of them can function only as two 16-bit register pairs. Six others can be used interchangeably as 8-bit registers or as 16-bit register pairs. The 8085A register set is as follows:

Mnemonic	Register	Contents
ACC or A	Accumulator	8 bits
PC	Program Counter	16-bit address
BC,DE,HL	General-Purpose Registers; data pointer (HL)	8 bits x 6 or 16 bits x 3
SP	Stack Pointer	16-bit address
Flags or F	Flag Register	5 flags (8-bit space)

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8-bit Address Bus and the lower 8-bit Address/Data Bus. During the first T state (clock cycle) of a machine cycle the low order address is sent out on the Address/Data bus. These lower 8 bits may be latched externally by the Address Latch Enable signal (ALE). During the rest of the machine cycle the data bus is used for memory or I/O data.

The 8085A provides $\overline{RD}$, $\overline{WR}$, S_0, S_1, and IO/$\overline{M}$ signals for bus control. An Interrupt Acknowledge signal ($\overline{INTA}$) is also provided. HOLD, READY, and all Interrupts are synchronized with the processor's internal clock. The 8085A also provides Serial Input Data (SID) and Serial Output Data (SOD) lines for simple serial interface.

In addition to these features, the 8085A has three maskable, vector interrupt pins and one nonmaskable TRAP interrupt.

INTERRUPT AND SERIAL I/O

The 8085A has 5 interrupt inputs: INTR, RST 5.5, RST 6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080A INT. Each of the three RESTART inputs, 5.5, 6.5, and 7.5, has a programmable mask. TRAP is also a RESTART interrupt but it is nonmaskable.

The three maskable interrupts cause the internal execution of RESTART (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The nonmaskable TRAP causes the internal execution of a RESTART vector independent of the state of the interrupt enable or masks. (See Table 1.)

There are two different types of inputs in the restart interrupts. RST 5.5 and RST 6.5 are *high level-sensitive* like INTR (and INT on the 8080) and are recognized with the same timing as INTR. RST 7.5 is *rising edge-sensitive.*

For RST 7.5, only a pulse is required to set an internal flip-flop which generates the internal interrupt request. (See Section 2.2.7.) The RST 7.5 request flip-flop remains set until the request is serviced. Then it is reset automatically. This flip-flop may also be reset by using the SIM instruction or by issuing a $\overline{\text{RESET IN}}$ to the 8085A. The RST 7.5 internal flip-flop will be set by a pulse on the RST 7.5 pin even when the RST 7.5 interrupt is masked out.

The status of the three RST interrupt masks can only be affected by the SIM instruction and $\overline{\text{RESET IN}}$. (See SIM, Chapter 4.)

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP — highest priority, RST 7.5, RST 6.5, RST 5.5, INTR — lowest priority. This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt an RST 7.5 routine if the interrupts are re-enabled before the end of the RST 7.5 routine.

The TRAP interrupt is useful for catastrophic events such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both *edge and level sensitive.* The TRAP input must go high and remain high until it is acknowledged. It will not be recognized again until it goes low, then high again. This avoids any false triggering due to noise or logic glitches. Figure 3 illustrates the TRAP interrupt request circuitry within the 8085A. Note that the servicing of any interrupt (TRAP, RST 7.5, RST 6.5, RST 5.5, INTR) disables all future interrupts (except TRAPs) until an EI instruction is executed.

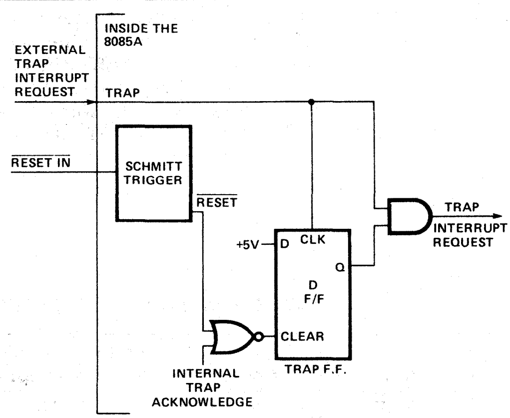


Figure 3. TRAP and $\overline{\text{RESET IN}}$ Circuit

The TRAP interrupt is special in that it disables interrupts, but preserves the previous interrupt enable status. Performing the first RIM instruction following a TRAP interrupt allows you to determine whether interrupts were enabled or disabled prior to the TRAP. All subsequent RIM instructions provide current interrupt enable status. Performing a RIM instruction following INTR, or RST 5.5–7.5 will provide current Interrupt Enable status, revealing that Interrupts are disabled. See the description of the RIM instruction in Chapter 4.

The serial I/O system is also controlled by the RIM and SIM instructions. SID is read by RIM, and SIM sets the SOD data.

DRIVING THE X_1 AND X_2 INPUTS

You may drive the clock inputs of the 8085A or 8085A-2 with a crystal, an LC tuned circuit, an RC network, or an external clock source. The driving frequency must be at least 1 MHz, and must be twice the desired internal clock frequency; hence, the 8085A is operated with a 6 MHz crystal (for 3 MHz clock), and the 8085A-2 can be operated with a 10 MHz crystal (for 5 MHz clock). If a crystal is used, it must have the following characteristics:

Parallel resonance at twice the clock frequency desired
C_L (load capacitance) $\leq$ 30 pf
C_s (shunt capacitance) $\leq$ 7 pf
R_s (equivalent shunt resistance) $\leq$ 75 Ohms
Drive level: 10 mW
Frequency tolerance: $\pm$.005% (suggested)

Note the use of the 20 pf capacitors between X_1, X_2 and ground. These capacitors are required with crystal frequencies below 4 MHz to assure oscillator startup at the correct frequency. A parallel-resonant LC circuit may be used as the frequency-determining network for the 8085A, providing that its frequency tolerance of approximately $\pm$10% is acceptable. The components are chosen from the formula:

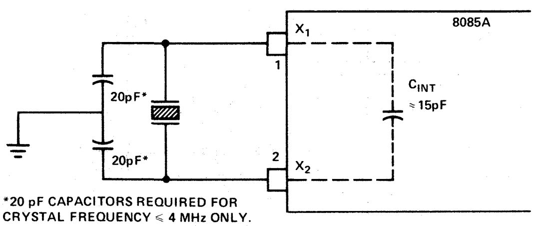
$$f = \frac{1}{2\pi\sqrt{L(C_{ext} + C_{int})}}$$

To minimize variations in frequency, it is recommended that you choose a value for C_{ext} that is at least twice that of C_{int}, or 30 pF. The use of an LC circuit is not recommended for frequencies higher than approximately 5 MHz.

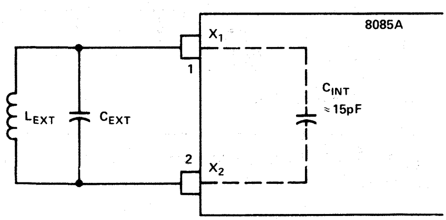
An RC circuit may be used as the frequency-determining network for the 8085A if maintaining a precise clock frequency is of no importance. Variations in the on-chip timing generation can cause a wide variation in frequency when using the RC mode. Its advantage is its low component cost. The driving frequency generated by the circuit shown is approximately 3 MHz. It is not recommended that frequencies greatly higher or lower than this be attempted.

Figure 4 shows the recommended clock driver circuits. Note in D and E that pullup resistors are required to assure that the high level voltage of the input is at least 4 V.

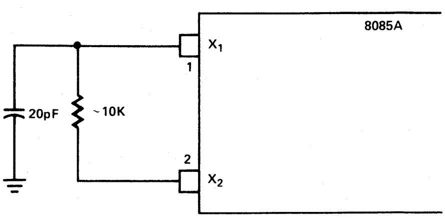
For driving frequencies up to and including 6 MHz you may supply the driving signal to X_1 and leave X_2 open-circuited (Figue 4D). If the driving frequency is from 6 MHz to 10 MHz, stability of the clock generator will be improved by driving both X_1 and X_2 with a push-pull source (Figure 4E). To prevent self-oscillation of the 8085A, be sure that X_2 is not coupled back to X_1 through the driving circuit.



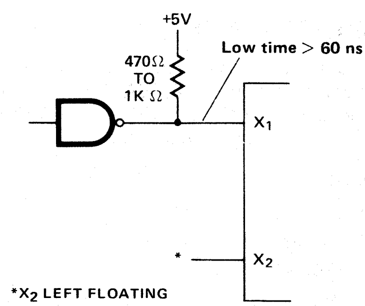
A. Quartz Crystal Clock Driver



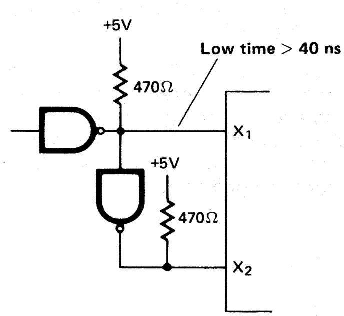
B. LC Tuned Circuit Clock Driver



C. RC Circuit Clock Driver



D. 1-6 MHz Input Frequency External Clock Driver Circuit



E. 1-10 MHz Input Frequency External Clock Driver Circuit

Figure 4. Clock Driver Circuits

GENERATING AN 8085A WAIT STATE

If your system requirements are such that slow memories or peripheral devices are being used, the circuit shown in Figure 5 may be used to insert one WAIT state in each 8085A machine cycle

The D flip-flops should be chosen so that

- CLK is rising edge-triggered
- CLEAR is low-level active.

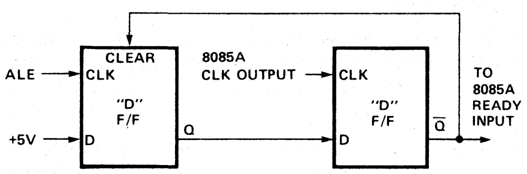


Figure 5. Generation of a Wait State for 8085A CPU

As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085A can be used with slow memory. HOLD causes the cpu to relinquish the bus when it is through with it by floating the Address and Data Buses.

SYSTEM INTERFACE

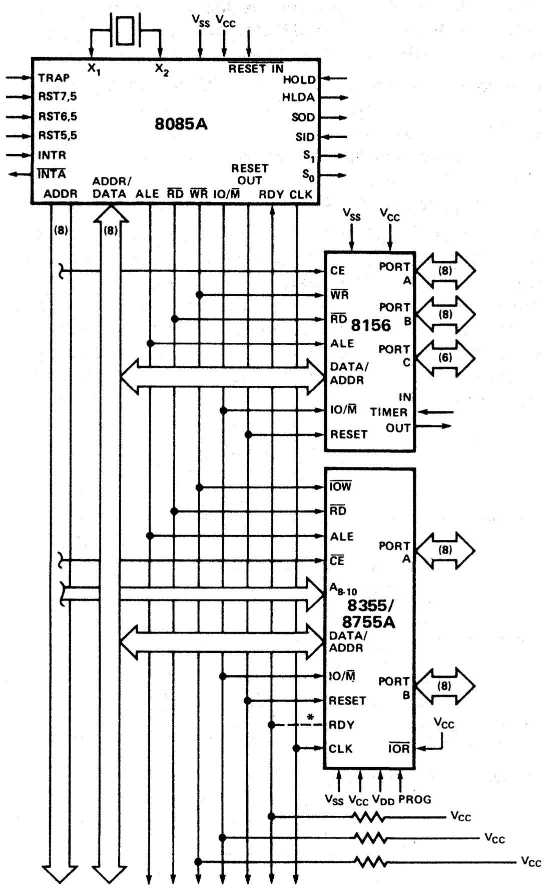
The 8085A family includes memory components, which are directly compatible to the 8085A cpu. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8-bit I/O Ports
- 1 6-bit I/O Port
- 4 Interrupt Levels
- Serial In/Serial Out Ports

This minimum system, using the standard I/O technique is as shown in Figure 6.

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. Figure 7 shows the system configuration of Memory Mapped I/O using 8085A.

The 8085A cpu can also interface with the standard memory that does *not* have the multiplexed address/data bus. It will require a simple 8212 (8-bit latch) as shown in Figure 8.



*NOTE: OPTIONAL CONNECTION

Figure 6. 8085A Minimum System (Standard I/O Technique)

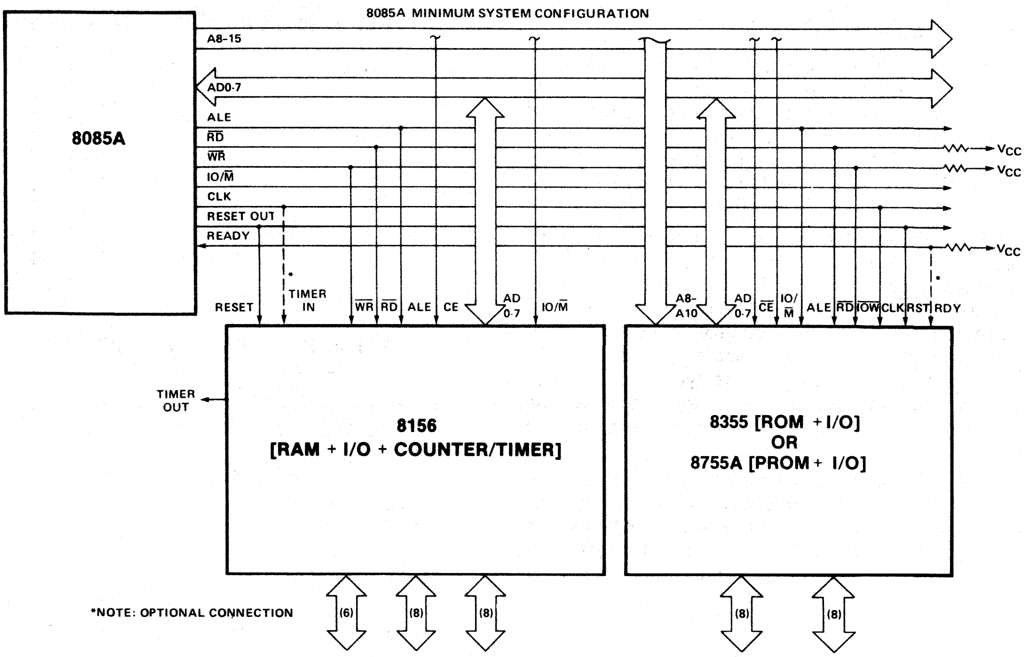


Figure 7. MCS-85™ Minimum System (Memory Mapped I/O)

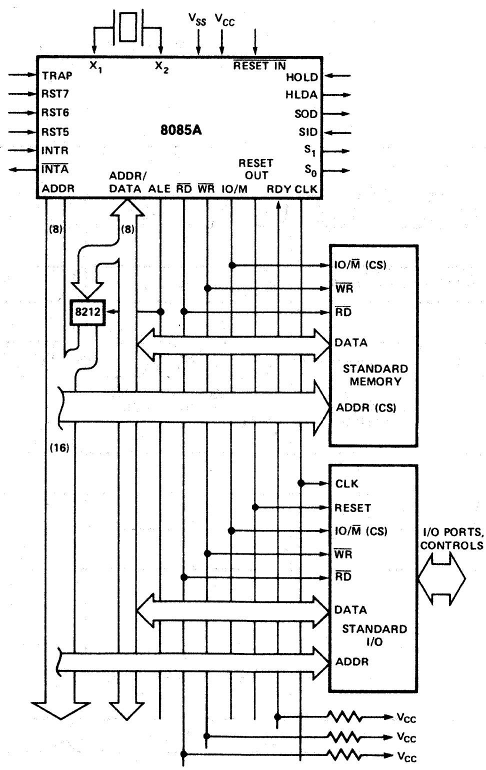


Figure 8. MCS-85™ System (Using Standard Memories)

BASIC SYSTEM TIMING

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure 9 shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines (IO/$\overline{M}$, S_1, S_0) and the three control signals ($\overline{RD}$, $\overline{WR}$, and $\overline{INTA}$). (See Table 2.) The status lines can be used as advanced controls (for device selection, for example), since they become active at the T_1 state, at the outset of each machine cycle. Control lines $\overline{RD}$ and $\overline{WR}$ become active later, at the time when the transfer of data is to take place, so are used as command lines.

A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of READY or HOLD inputs). Any T state must be one of ten possible states, shown in Table 3.

TABLE 2. 8085A MACHINE CYCLE CHART

MACHINE CYCLE		STATUS			CONTROL		
		IO/$\overline{M}$	S1	S0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
OPCODE FETCH	(OF)	0	1	1	0	1	1
MEMORY READ	(MR)	0	1	0	0	1	1
MEMORY WRITE	(MW)	0	0	1	1	0	1
I/O READ	(IOR)	1	1	0	0	1	1
I/O WRITE	(IOW)	1	0	1	1	0	1
ACKNOWLEDGE OF INTR	(INA)	1	1	1	1	1	0
BUS IDLE	(BI): DAD	0	1	0	1	1	1
	ACK. OF RST,TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

TABLE 3. 8085A MACHINE STATE CHART

Machine State	Status & Buses				Control		
	S1,S0	IO/$\overline{M}$	A_8-A_{15}	AD_0-AD_7	$\overline{RD}$,$\overline{WR}$	$\overline{INTA}$	ALE
T_1	X	X	X	X	1	1	1*
T_2	X	X	X	X	X	X	0
T_{WAIT}	X	X	X	X	X	X	0
T_3	X	X	X	X	X	X	0
T_4	1	0†	X	TS	1	1	0
T_5	1	0†	X	TS	1	1	0
T_6	1	0†	X	TS	1	1	0
T_{RESET}	X	TS	TS	TS	TS	1	0
T_{HALT}	0	TS	TS	TS	TS	1	0
T_{HOLD}	X	TS	TS	TS	TS	1	0

0 = Logic "0" TS = High Impedance
1 = Logic "1" X = Unspecified

* ALE not generated during 2nd and 3rd machine cycles of DAD instruction.
† IO/M = 1 during T_4-T_6 of INA machine cycle.

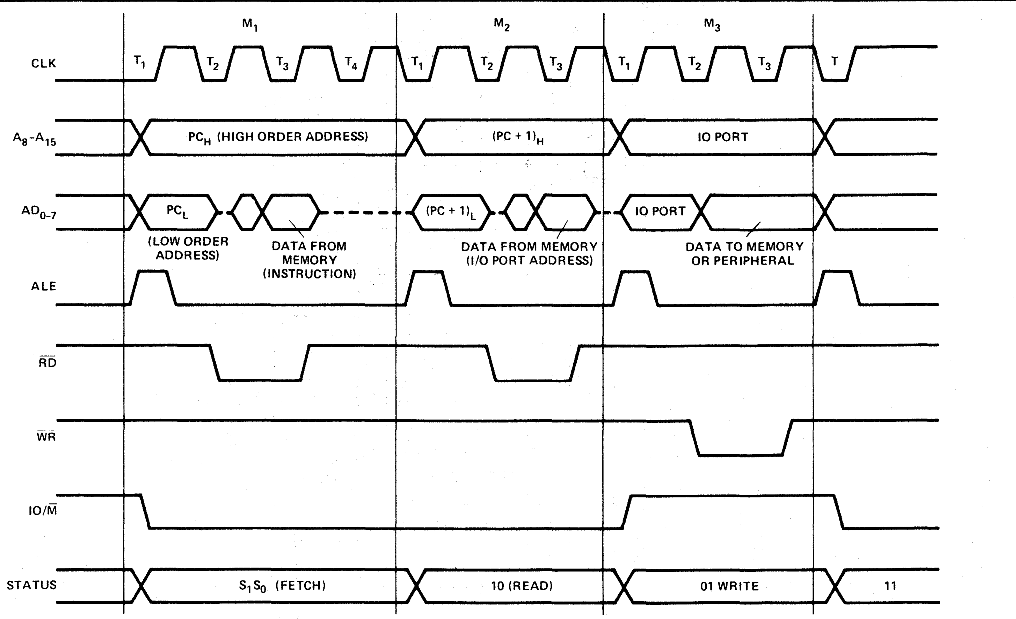


Figure 9. 8085A Basic System Timing

TABLE 8. INSTRUCTION SET SUMMARY

Mnemonic	Description	Instruction Code(1) D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock(2) Cycles
MOVE, LOAD, AND STORE										
MOVr1 r2	Move register to register	0	1	D	D	D	S	S	S	4
MOV M,r	Move register to memory	0	1	1	1	0	S	S	S	7
MOV r,M	Move memory to register	0	1	D	D	D	1	1	0	7
MVI r	Move immediate register	0	0	D	D	D	1	1	0	7
MVI M	Move immediate memory	0	0	1	1	0	1	1	0	10
LXI B	Load immediate register Pair B & C	0	0	0	0	0	0	0	1	10
LXI D	Load immediate register Pair D & E	0	0	0	1	0	0	0	1	10
LXI H	Load immediate register Pair H & L	0	0	1	0	0	0	0	1	10
LXI SP	Load immediate stack pointer	0	0	1	1	0	0	0	1	10
STAX B	Store A indirect	0	0	0	0	0	0	1	0	7
STAX D	Store A indirect	0	0	0	1	0	0	1	0	7
LDAX B	Load A indirect	0	0	0	0	1	0	1	0	7
LDAX D	Load A indirect	0	0	0	1	1	0	1	0	7
STA	Store A direct	0	0	1	1	0	0	1	0	13
LDA	Load A direct	0	0	1	1	1	0	1	0	13
SHLD	Store H & L direct	0	0	1	0	0	0	1	0	16
LHLD	Load H & L direct	0	0	1	0	1	0	1	0	16
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
STACK OPS										
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	12
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	12
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	12
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	12
POP B	Pop register Pair B & C off stack	1	1	0	0	0	0	0	1	10
POP D	Pop register Pair D & E off stack	1	1	0	1	0	0	0	1	10
POP H	Pop register Pair H & L off stack	1	1	1	0	0	0	0	1	10
POP PSW	Pop A and Flags off stack	1	1	1	1	0	0	0	1	10
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	16
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	6
JUMP										
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	7/10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	7/10
JZ	Jump on zero	1	1	0	0	1	0	1	0	7/10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	7/10
JP	Jump on positive	1	1	1	1	0	0	1	0	7/10
JM	Jump on minus	1	1	1	1	1	0	1	0	7/10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	7/10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	7/10
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	6
CALL										
CALL	Call unconditional	1	1	0	0	1	1	0	1	18
CC	Call on carry	1	1	0	1	1	1	0	0	9/18
CNC	Call on no carry	1	1	0	1	0	1	0	0	9/18
CZ	Call on zero	1	1	0	0	1	1	0	0	9/18
CNZ	Call on no zero	1	1	0	0	0	1	0	0	9/18
CP	Call on positive	1	1	1	1	0	1	0	0	9/18
CM	Call on minus	1	1	1	1	1	1	0	0	9/18
CPE	Call on parity even	1	1	1	0	1	1	0	0	9/18
CPO	Call on parity odd	1	1	1	0	0	1	0	0	9/18
RETURN										
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	6/12
RNC	Return on no carry	1	1	0	1	0	0	0	0	6/12
RZ	Return on zero	1	1	0	0	1	0	0	0	6/12
RNZ	Return on no zero	1	1	0	0	0	0	0	0	6/12
RP	Return on positive	1	1	1	1	0	0	0	0	6/12
RM	Return on minus	1	1	1	1	1	0	0	0	6/12
RPE	Return on parity even	1	1	1	0	1	0	0	0	6/12
RPO	Return on parity odd	1	1	1	0	0	0	0	0	6/12
RESTART										
RST	Restart	1	1	A	A	A	1	1	1	12
INPUT/OUTPUT										
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	10
INCREMENT AND DECREMENT										
INR r	Increment register	0	0	D	D	D	1	0	0	4
DCR r	Decrement register	0	0	D	D	D	1	0	1	4
INR M	Increment memory	0	0	1	1	0	1	0	0	10
DCR M	Decrement memory	0	0	1	1	0	1	0	1	10
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	6
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	6
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	6
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	6
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	6
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	6
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	6
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	6
ADD										
ADD r	Add register to A	1	0	0	0	0	S	S	S	4
ADC r	Add register to A with carry	1	0	0	0	1	S	S	S	4
ADD M	Add memory to A	1	0	0	0	0	1	1	0	7
ADC M	Add memory to A with carry	1	0	0	0	1	1	1	0	7
ADI	Add immediate to A	1	1	0	0	0	1	1	0	7
ACI	Add immediate to A with carry	1	1	0	0	1	1	1	0	7
DAD B	Add B & C to H & L	0	0	0	0	1	0	0	1	10
DAD D	Add D & E to H & L	0	0	0	1	1	0	0	1	10
DAD H	Add H & L to H & L	0	0	1	0	1	0	0	1	10
DAD SP	Add stack pointer to H & L	0	0	1	1	1	0	0	1	10
SUBTRACT										
SUB r	Subtract register from A	1	0	0	1	0	S	S	S	4
SBB r	Subtract register from A with borrow	1	0	0	1	1	S	S	S	4
SUB M	Subtract memory from A	1	0	0	1	0	1	1	0	7
SBB M	Subtract memory from A with borrow	1	0	0	1	1	1	1	0	7
SUI	Subtract immediate from A	1	1	0	1	0	1	1	0	7
SBI	Subtract immediate from A with borrow	1	1	0	1	1	1	1	0	7
LOGICAL										
ANA r	And register with A	1	0	1	0	0	S	S	S	4

TABLE 8. INSTRUCTION SET SUMMARY (Continued)

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
XRA r	Exclusive Or register with A	1	0	1	0	1	S	S	S	4
ORA r	Or register with A	1	0	1	1	0	S	S	S	4
CMP r	Compare register with A	1	0	1	1	1	S	S	S	4
ANA M	And memory with A	1	0	1	0	0	1	1	0	7
XRA M	Exclusive Or memory with A	1	0	1	0	1	1	1	0	7
ORA M	Or memory with A	1	0	1	1	0	1	1	0	7
CMP M	Compare memory with A	1	0	1	1	1	1	1	0	7
ANI	And immediate with A	1	1	1	0	0	1	1	0	7
XRI	Exclusive Or immediate with A	1	1	1	0	1	1	1	0	7
ORI	Or immediate with A	1	1	1	1	0	1	1	0	7
CPI	Compare immediate with A	1	1	1	1	1	1	1	0	7
ROTATE										
RLC	Rotate A left	0	0	0	0	0	1	1	1	4
RRC	Rotate A right	0	0	0	0	1	1	1	1	4

Mnemonic	Description	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
		Instruction Code[1]								
RAL	Rotate A left through carry	0	0	0	1	0	1	1	1	4
RAR	Rotate A right through carry	0	0	0	1	1	1	1	1	4
SPECIALS										
CMA	Complement A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Complement carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
CONTROL										
EI	Enable Interrupts	1	1	1	1	1	0	1	1	4
DI	Disable Interrupt	1	1	1	1	0	0	1	1	4
NOP	No-operation	0	0	0	0	0	0	0	0	4
HLT	Halt	0	1	1	1	0	1	1	0	5
NEW 8085A INSTRUCTIONS										
RIM	Read Interrupt Mask	0	0	1	0	0	0	0	0	4
SIM	Set Interrupt Mask	0	0	1	1	0	0	0	0	4

NOTES: 1. DDD or SSS: B 000. C 001. D 010. E 011. H 100. L 101. Memory 110. A 111.
2. Two possible cycle times. (6/12) indicate instruction cycles dependent on condition flags.

![intel logo]

8155/8156/8155-2/8156-2
2048 BIT STATIC MOS RAM WITH I/O PORTS AND TIMER

8085A	8085A-2	Compatible CPU / Chip Enable
8155	8155-2	ACTIVE LOW
8156	8156-2	ACTIVE HIGH

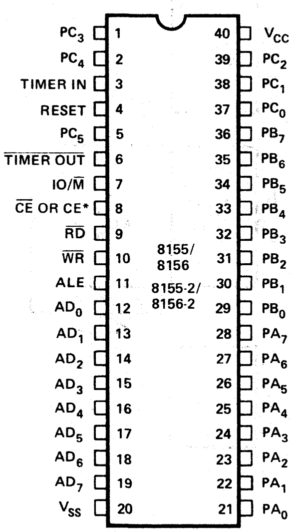
- **256 Word x 8 Bits**
- **Single +5V Power Supply**
- **Completely Static Operation**
- **Internal Address Latch**
- **2 Programmable 8 Bit I/O Ports**
- **1 Programmable 6-Bit I/O Port**
- **Programmable 14-Bit Binary Counter/ Timer**
- **Multiplexed Address and Data Bus**
- **40 Pin DIP**

The 8155 and 8156 are RAM and I/O chips to be used in the MCS-85™ microcomputer system. The RAM portion is designed with 2048 static cells organized as 256 x 8. They have a maximum access time of 400 ns to permit use with no wait states in 8085A CPU. The 8155-2 and 8156-2 have maximum access times of 330 ns for use with the 8085A-2.

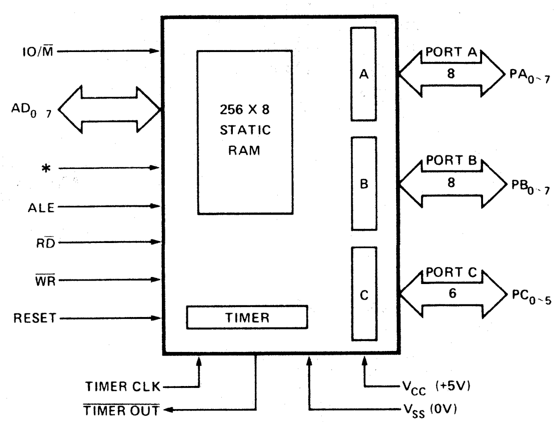
The I/O portion consists of three general purpose I/O ports. One of the three ports can be programmed to be status pins, thus allowing the other two ports to operate in handshake mode.

A 14-bit programmable counter/timer is also included on chip to provide either a square wave or terminal count pulse for the CPU system depending on timer mode.

PIN CONFIGURATION



BLOCK DIAGRAM



*: 8155/8155-2 = $\overline{CE}$, 8156/8156-2 = CE

July, 1978

intel®

ADVANCE INFORMATION
Characteristics are subject to change without notice

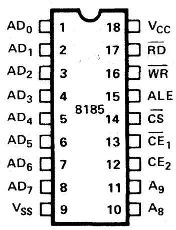
8185
1024 x 8-BIT STATIC RAM FOR MCS-85

- **Multiplexed Address and Data Bus**
- **Directly Compatible with 8085A Microprocessor**
- **Low Operating Power Dissipation**
- **Low Standby Power Dissipation**
- **Single +5V Supply**
- **High Density 18-Pin Package**

The Intel® 8185 is an 8192-bit static random access memory (RAM) organized as 1024 words by 8-bits using N-channel Silicon-Gate MOS technology. The multiplexed address and data bus allows the 8185 to interface directly to the 8085A microprocessor to provide a maximum level of system integration.

The low standby power dissipation minimizes system power requirements whent he 8185 is disabled.

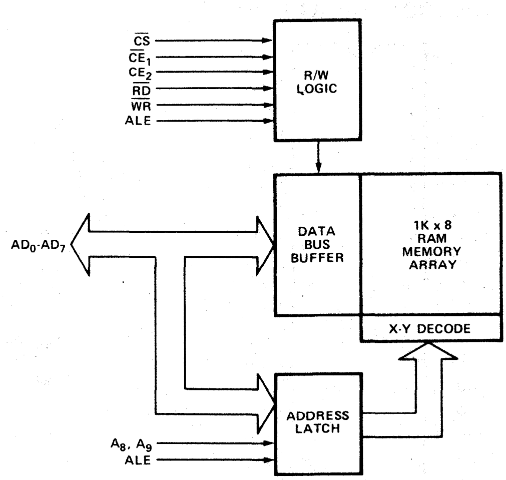
PIN CONFIGURATION



PIN NAMES

AD_0-AD_7	ADDRESS/DATA LINES
A_8, A_9	ADDRESS LINES
$\overline{CS}$	CHIP SELECT
$\overline{CE}_1$	CHIP ENABLE (IO/$\overline{M}$)
CE_2	CHIP ENABLE
ALE	ADDRESS LATCH ENABLE
$\overline{RD}$	READ ENABLE
$\overline{WR}$	WRITE ENABLE

BLOCK DIAGRAM



July, 1978

8355*/8355-2**
16,384-BIT ROM WITH I/O
*Directly Compatible with 8085A CPU
**Directly Compatible with 8085A-2

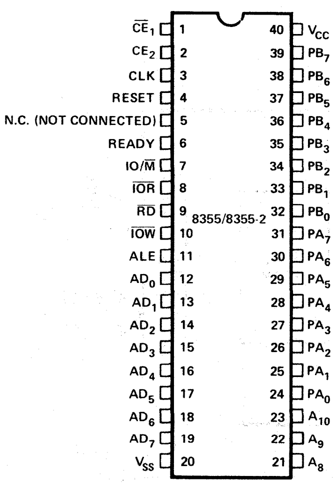
- **2048 Words × 8 Bits**
- **Single +5V Power Supply**
- **Internal Address Latch**
- **2 General Purpose 8-Bit I/O Ports**
- **Each I/O Port Line Individually Programmable as Input or Output**
- **Multiplexed Address and Data Bus**
- **40-Pin DIP**

The Intel® 8355 is a ROM and I/O chip to be used in the MCS-85™ microcomputer system. The ROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 400 ns to permit use with no wait states in the 8085A CPU.

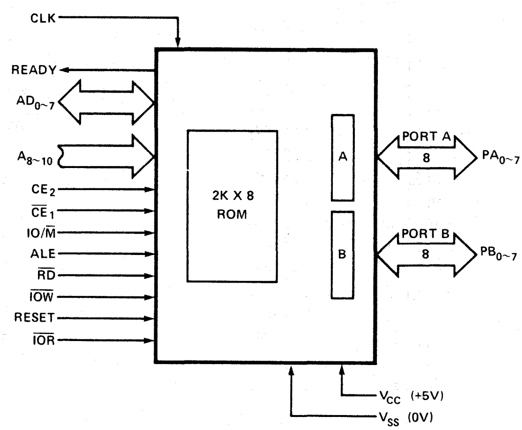
The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is indivdually programmable as input or output.

The 8355-2 has a 300ns access time for compatibility with the 8085A-2 microprocessor.

PIN CONFIGURATION



BLOCK DIAGRAM



July, 1978

intel®

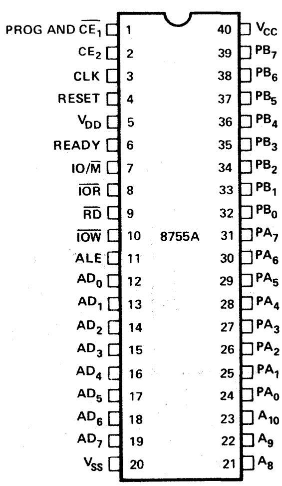
8755A
16,384-BIT EPROM WITH I/O
• Directly Compatible with 8085A CPU

- 2048 Words × 8 Bits
- Single +5V Power Supply (V_{CC})
- U.V. Erasable and Electrically Reprogrammable
- Internal Address Latch
- 2 General Purpose 8-Bit I/O Ports
- Each I/O Port Line Individually Programmable as Input or Output
- Multiplexed Address and Data Bus
- 40-Pin DIP

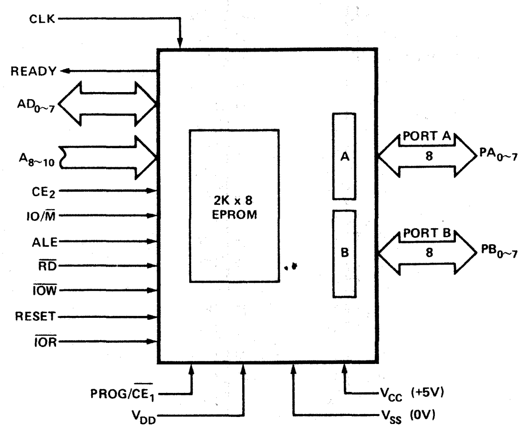
The Intel® 8755A is an erasable and electrically reprogrammable ROM (EPROM) and I/O chip to be used in the MCS-85™ microcomputer system. The EPROM portion is organized as 2048 words by 8 bits. It has a maximum access time of 450 ns to permit use with no wait states in an 8085A CPU.

The I/O portion consists of 2 general purpose I/O ports. Each I/O port has 8 port lines, and each I/O port line is individually programmable as input or output.

PIN CONFIGURATION



BLOCK DIAGRAM



July, 1978

CHAPTER 5

Device Specifications

- MCS-86™
- MCS-85™*
- **Peripherals****
- Static RAMs***
- ROMs/EPROMs***

*For complete specifications refer to the Intel MCS-85 User's Manual.

**For complete specifications refer to the Intel Peripheral Design Handbook.

***For complete specifications refer to the 1978 Intel Data Catalog.

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

8041/8741 UNIVERSAL PERIPHERAL INTERFACE 8-BIT MICROCOMPUTER

- **Fully Compatibnle with MCS-80™, MCS-85™ and MCS-48™ Microprocessor Families**
- **Single Level Interrupt**
- **8-Bit CPU plus ROM, RAM, I/O, Timer and Clock in a Single Package**
- **Single 5V Supply**
- **Alternative to Custom LSI**
- **Pin Compatible ROM and EPROM Versions**
- **1K × 8 ROM/EPROM, 64 × 8 RAM, 18 Programmable I/O Pins**
- **Asynchronous Data Register for Interface to Master Processor**
- **Expandable I/O**

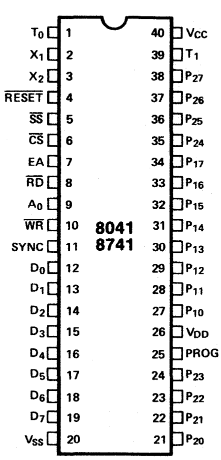
The Intel® 8041/8741 is a general purpose, programmable interface device designed for use with a variety of 8-bit microprocessor systems. It contains a low cost microcomputer with program memory, data memory, 8-bit CPU, I/O ports, timer/counter, and clock in a single 40-pin package. Interface registers are included to enable the UPI device to function as a peripheral controller in MCS-80™, MCS-85™, MCS-48™, and other 8-bit systems.

The UPI-41™ has 1K words of program memory and 64 words of data memory on-chip. To allow full user flexibility the program memory is available as ROM in the 8041 version or as UV-erasable EPROM in the 8741 version. The 8741 and the 8041 are fully pin compatible for easy transition from prototype to production level designs.

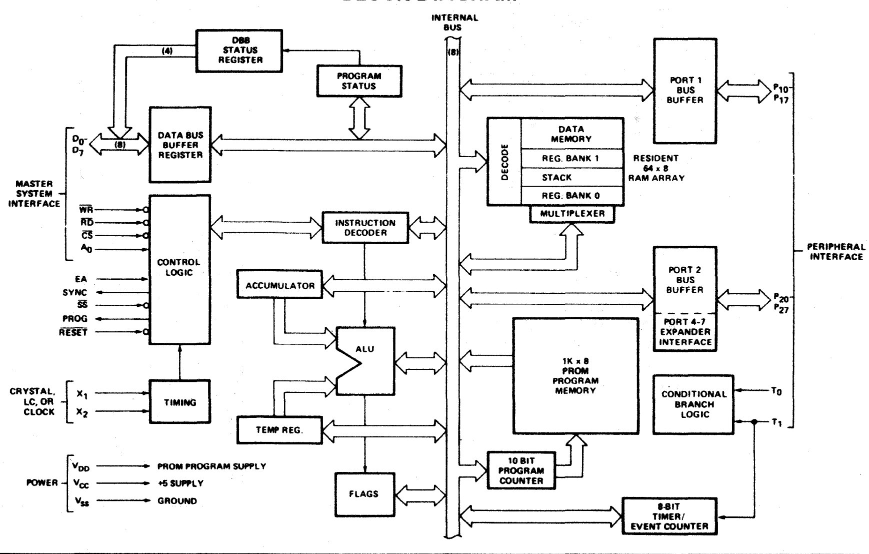
The device has two 8-bit, TTL compatible I/O ports and two test inputs. Individual port lines can function as either inputs or outputs under software control. I/O can be expanded with the 8243 device which is directly compatible and has 16 I/O lines. An 8-bit programmable timer/counter is included in the UPI device for generating timing sequences or counting external inputs. Additional UPI features include: single 5V supply, low power standby mode (in the 8041), single-step mode for debug (in the 8741), single level interrupt, and dual working register banks.

Because it's a complete microcomputer, the UPI provides more flexibility for the designer than conventional LSI interface devices. It is designed to be an efficient controller as well as an arithmetic processor. Applications include keyboard scanning, printer control, display multiplexing and similar functions which involve interfacing peripheral devices to microprocessor systems.

PIN CONFIGURATION



BLOCK DIAGRAM



intel®

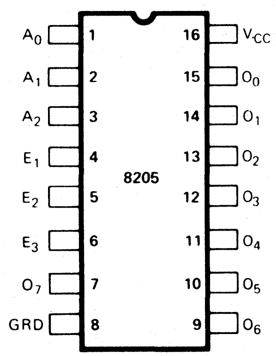
8205
HIGH SPEED 1 OUT OF 8 BINARY DECODER

- I/O Port or Memory Selector
- Simple Expansion — Enable Inputs
- High Speed Schottky Bipolar Technology — 18 ns Max Delay
- Directly Compatible with TTL Logic Circuits
- Low Input Load Current — 0.25 mA Max, 1/6 Standard TTL Input Load
- Minimum Line Reflection — Low Voltage Diode Input Clamp
- Outputs Sink 10 mA Min
- 16-Pin Dual In-Line Ceramic or Plastic Package

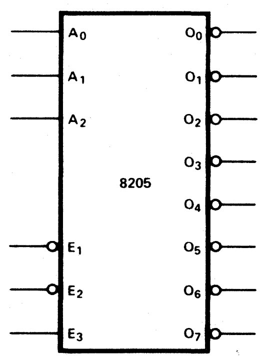
The Intel® 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its 8 outputs goes "low", thus a single row of a memory system is selected. The 3-chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

The 8205 is packaged in a standard 16-pin dual in-line package, and its performance is specified over the temperature range of 0°C to +75°C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffussion process.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

A_0-A_2	ADDRESS INPUTS
$\overline{E_1}$-E_3	ENABLE INPUTS
$\overline{O_0}$-$\overline{O_7}$	DECODED OUTPUTS

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

8212

8-BIT INPUT/OUTPUT PORT

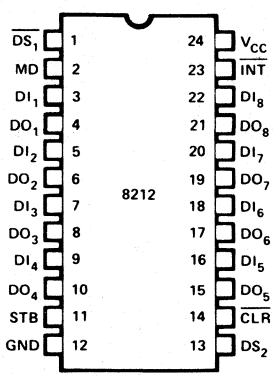
- **Fully Parallel 8-Bit Data Register and Buffer**
- **Service Request Flip-Flop for Interrupt Generation**
- **Low Input Load Current — 0.25 mA Max**
- **3-State Outputs**
- **Outputs Sink 15 mA**
- **3.65V Output High Voltage for Direct Interface to 8080 CPU or 8008 CPU**
- **Asynchronous Register Clear**
- **Replaces Buffers, Latches, and Multiplexers in Microcomputer Systems**
- **Reduces System Package Count**

The Intel® 8212 input/output port consists of an 8-bit latch with 3-state output buffers along with control and device selection logic. Also included is a service request flip-flop for the generation and control of interrupts to the microprocessor.

The device is multimode in nature. It can be used to implement latches, gated buffers or multiplexers. Thus, all of the principal peripheral and input/output functions of a microcomputer system can be implemented with this device.

*Note: The specifications for the 3212 are identical with those for the 8212.

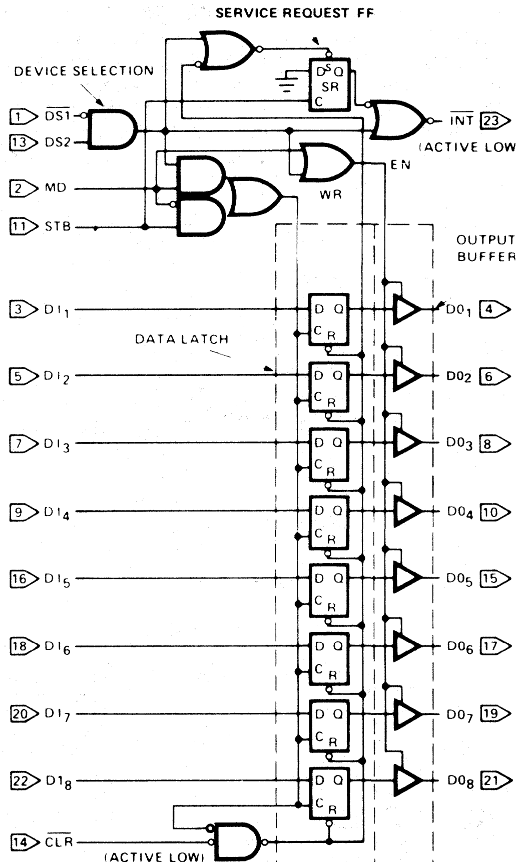
PIN CONFIGURATION



PIN NAMES

DI_1-DI_8	DATA IN
DO_1-DO_8	DATA OUT
$\overline{DS_1}$-DS_2	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

LOGIC DIAGRAM

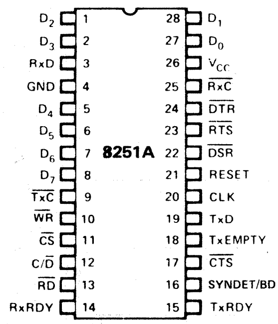


8251A
PROGRAMMABLE COMMUNICATION INTERFACE

- Synchronous and Asynchronous Operation
- Synchronous 5-8 Bit Characters; Internal or External Character Synchronization; Automatic Sync Insertion
- Asynchronous 5-8 Bit Characters; Clock Rate—1, 16 or 64 Times Baud Rate; Break Character Generation; 1, 1½, or 2 Stop Bits; False Start Bit Detection; Automatic Break Detect and Handling; 19.2K Baud.
- Baud Rate — DC to 64K Baud
- Full Duplex, Double Buffered, Transmitter and Receiver
- Error Detection — Parity, Overrun and Framing
- Fully Compatible with 8080/8085 CPU
- 28-Pin DIP Package
- All Inputs and Outputs are TTL Compatible
- Single +5V Supply
- Single TTL Clock

The Intel® 8251A is the enhanced version of the industry standard, Intel® 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART), designed for data communications with Intel's new high performance family of microprocessors such as the 8085. The 8251A is used as a peripheral device and is programmed by the CPU to operate using virtually any serial data transmission technique presently in use (including IBM "bi-sync"). The USART accepts data characters from the CPU in parallel format and then converts them into a continuous serial data stream for transmission. Simultaneously, it can receive serial data streams and convert them into parallel data characters for the CPU. The USART will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character for the CPU. The CPU can read the complete status of the USART at any time. These include data transmission errors and control signals such as SYNDET, TxEMPTY. The chip is constructed using N-channel silicon gate technology.

PIN CONFIGURATION

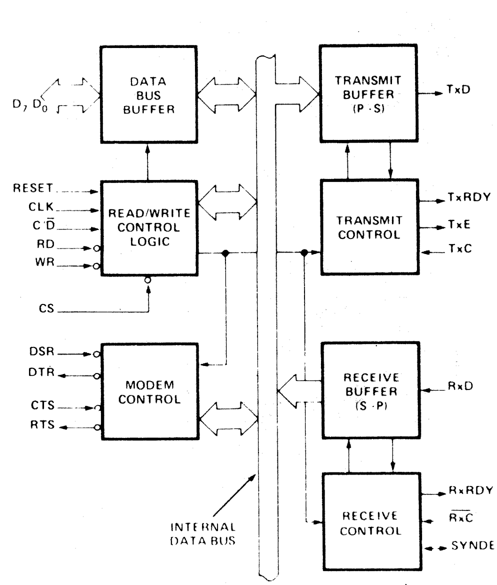


PIN NAMES

D_7-D_0	Data Bus (8 bits)
C/D	Control or Data is to be Written or Read
$\overline{RD}$	Read Data Command
$\overline{WR}$	Write Data or Control Command
$\overline{CS}$	Chip Enable
CLK	Clock Pulse (TTL)
RESET	Reset
$\overline{TxC}$	Transmitter Clock
TxD	Transmitter Data
$\overline{RxC}$	Receiver Clock
RxD	Receiver Data
RxRDY	Receiver Ready (has character for 8080)
TxRDY	Transmitter Ready (ready for char. from 8080)

$\overline{DSR}$	Data Set Ready
$\overline{DTR}$	Data Terminal Ready
SYNDET/BD	Sync Detect/ Break Detect
$\overline{RTS}$	Request to Send Data
$\overline{CTS}$	Clear to Send Data
TxE	Transmitter Empty
V_{CC}	+5 Volt Supply
GND	Ground

BLOCK DIAGRAM



intel®

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

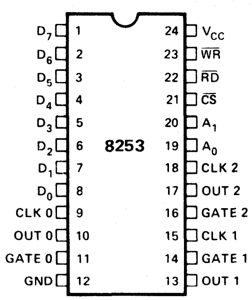
8253/8253-5
PROGRAMMABLE INTERVAL TIMER

- **MCS—85™ Compatible 8253-5**
- **3 Independent 16-Bit Counters**
- **DC to 2 MHz**
- **Programmable Counter Modes**
- **Count Binary or BCD**
- **Single +5V Supply**
- **24-Pin Dual In-Line Package**

The Intel® 8253 is a programmable counter/timer chip designed for use as an Intel microcomputer peripheral. It uses nMOS technology with a single +5V supply and is packaged in a 24-pin plastic DIP.

It is organized as 3 independent 16-bit counters, each with a count rate of up to 2 MHz. All modes of operation are software programmable.

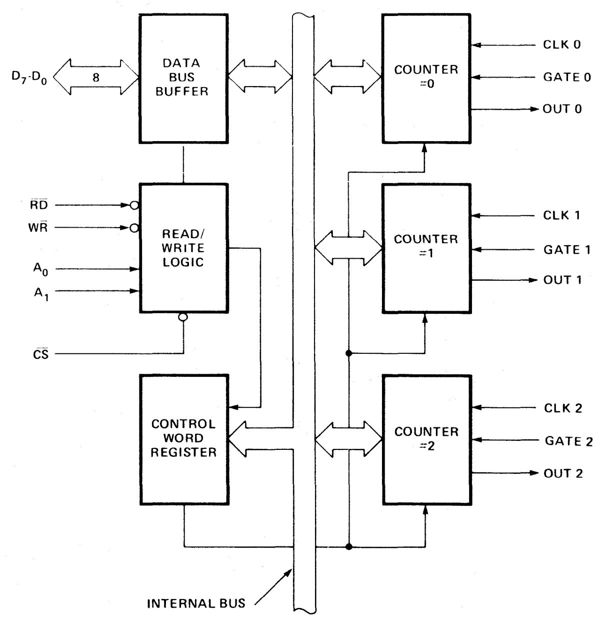
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS (8-BIT)
CLK N	COUNTER CLOCK INPUTS
GATE N	COUNTER GATE INPUTS
OUT N	COUNTER OUTPUTS
$\overline{RD}$	READ COUNTER
WR	WRITE COMMAND OR DATA
CS	CHIP SELECT
A_0-A_1	COUNTER SELECT
V_{CC}	+5 VOLTS
GND	GROUND

BLOCK DIAGRAM



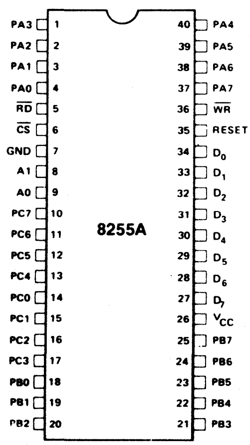
intel®

8255A/8255A-5 PROGRAMMABLE PERIPHERAL INTERFACE

- **MCS-85™ Compatible 8255A-5**
- **24 Programmable I/O Pins**
- **Completely TTL Compatible**
- **Fully Compatible with Intel® Microprocessor Families**
- **Improved Timing Characteristics**
- **Direct Bit Set/Reset Capability Easing Control Application Interface**
- **40-Pin Dual In-Line Package**
- **Reduces System Package Count**
- **Improved DC Driving Capability**

The Intel® 8255A is a general purpose programmable I/O device designed for use with Intel® microprocessors. It has 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. In the first mode (MODE 0), each group of 12 I/O pins may be programmed in sets of 4 to be input or output. In MODE 1, the second mode, each group may be programmed to have 8 lines of input or output. Of the remaining 4 pins, 3 are used for handshaking and interrupt control signals. The third mode of operation (MODE 2) is a bidirectional bus mode which uses 8 lines for a bidirectional bus, and 5 lines, borrowing one from the other group, for handshaking.

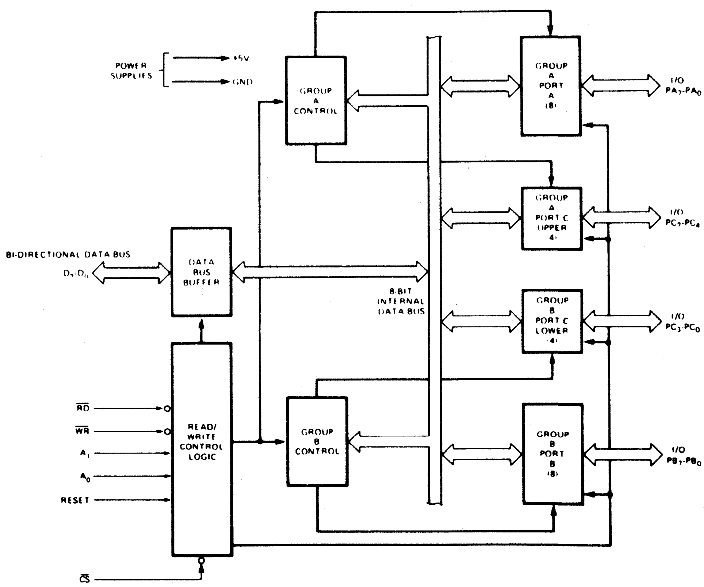
PIN CONFIGURATION



PIN NAMES

D_7–D_0	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
$\overline{CS}$	CHIP SELECT
$\overline{RD}$	READ INPUT
$\overline{WR}$	WRITE INPUT
A0, A1	PORT ADDRESS
PA7-PA0	PORT A (BIT)
PB7-PB0	PORT B (BIT)
PC7-PC0	PORT C (BIT)
V_{CC}	+5 VOLTS
GND	Ø VOLTS

8255A BLOCK DIAGRAM



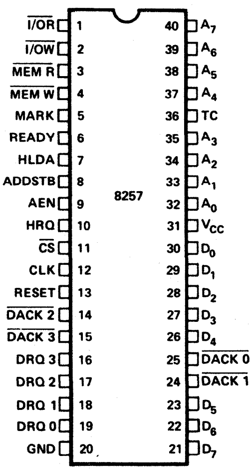
intel®

8257/8257-5 PROGRAMMABLE DMA CONTROLLER

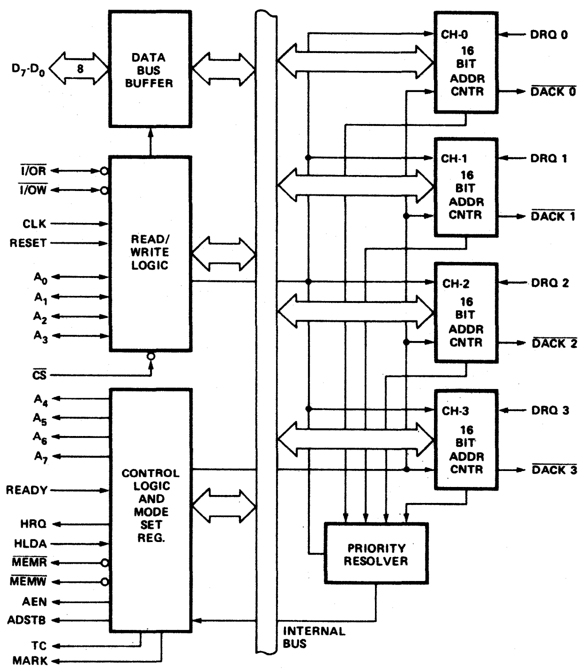
- MCS-85™ Compatible 8257-5
- 4-Channel DMA Controller
- Priority DMA Request Logic
- Channel Inhibit Logic
- Terminal Count and Modulo 128 Outputs
- Auto Load Mode
- Single TTL Clock
- Single +5V Supply
- Expandable
- 40-Pin Dual In-Line Package

The Intel® 8257 is a 4-channel direct memory access (DMA) controller. It is specifically designed to simplify the transfer of data at high speeds for the Intel® microcomputer systems. Its primary function is to generate, upon a peripheral request, a sequential memory address which will allow the peripheral to read or write data directly to or from memory. Acquisition of the system bus in accomplished via the CPU's hold function. The 8257 has priority logic that resolves the peripherals requests and issues a composite hold request to the CPU. It maintains the DMA cycle count for each channel and outputs a control signal to notify the peripheral that the programmed number of DMA cycles is complete. Other output control signals simplify sectored data transfers and expansion to other 8257 devices for systems that require more than 4 channels of DMA controlled transfer. The 8257 represents a significant savings in component count for DMA-based microcomputer systems and greatly simplifies the transfer of data at high speed between peripherals and memories.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

D_7-D_0	DATA BUS
A_7-A_0	ADDRESS BUS
I/OR	I/O READ
I/OW	I/O WRITE
MEMR	MEMORY READ
MEMW	MEMORY WRITE
CLK	CLOCK INPUT
RESET	RESET INPUT
READY	READY
HRQ	HOLD REQUEST (TO 8080A)
HLDA	HOLD ACKNOWLEDGE (FROM 8080A)

AEN	ADDRESS ENABLE
ADSTB	ADDRESS STROBE
TC	TERMINAL COUNT
MARK	MODULO 128 MARK
DRQ_3-DRQ_0	DMA REQUEST INPUT
$DACK_3$-$DACK_0$	DMA ACKNOWLEDGE OUT
CS	CHIP SELECT
V_{CC}	+5 VOLTS
GND	GROUND

February, 1978

intel®

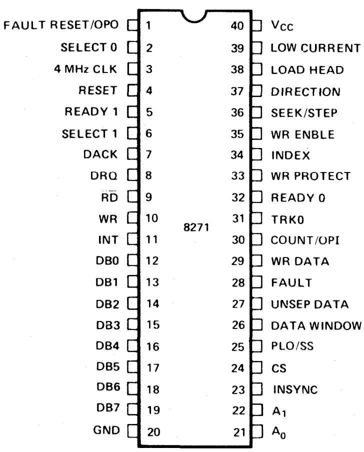
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8271
PROGRAMMABLE FLOPPY DISK CONTROLLER

- **IBM 3740 Soft Sectored Format Compatible**
- **Programmable Record Lengths**
- **Multi-Sector Capability**
- **Maintain Dual Drives with Minimum Software Overhead Expandable to 4 Drives**
- **Automatic Read/Write Head Positioning and Verification**
- **Internal CRC Generation and Checking**
- **Programmable Step Rate, Settle-Time, Head Load Time, Head Unload Ihdex Count**
- **Fully MCS-80 and MCS-85 Compatible**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8271 Programmable Floppy Disk Controller (FDC) is an LSI component designed to interface one to 4 floppy disk drives to an 8-bit microcomputer system. Its powerful control functions minimize both hardware and software overhead normally associated with floppy disk controllers.

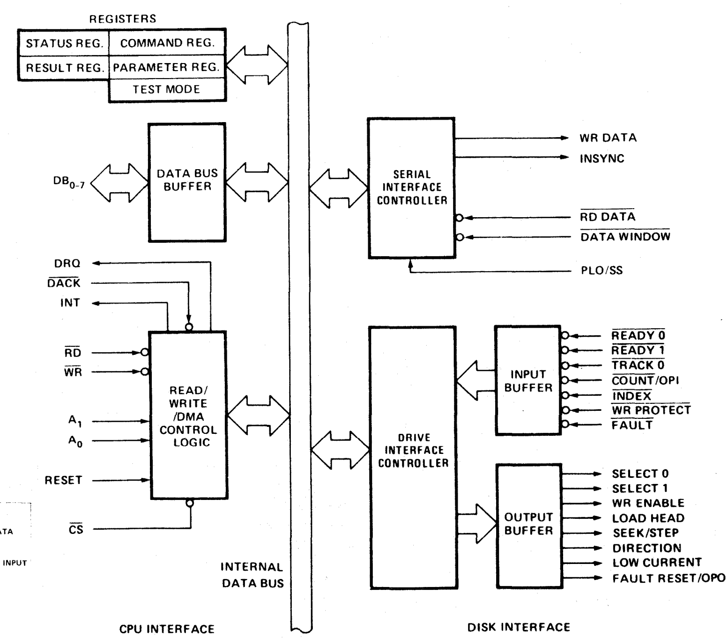
PIN CONFIGURATION



PIN NAMES

$DB_7 - DB_0$	DATA BUS (BI-DIRECTIONAL)	PLO/SS	PLO/SINGLE SHOT
CLK	CLOCK INPUT (TTL)	$\overline{\text{DATA WINDOW}}$	DATA WINDOW
SELECT 1, 0	SELECT 1, 0	$\overline{\text{UNSEP}}$ DATA	UNSEPARATED DATA
FAULT RESET/OPO	FAULT RESET/OPTIONAL OUTPUT	FAULT	FAULT
RESET	CHIP RESET	$\overline{\text{WR DATA}}$	WRITE DATA
$\overline{\text{READY 1}}$, 0	READY 1, 0	$\overline{\text{COUNT}}$/OPI	COUNT/OPTIONAL INPUT
$\overline{\text{DACK}}$	DMA ACKNOWLEDGE	$\overline{\text{TRK 0}}$	TRACK 0
DRQ	DMA REQUEST	$\overline{\text{WR}}$ PROTECT	WRITE PROTECT
$\overline{\text{RD}}$	CPU READ INPUT	INDEX	INDEX
$\overline{\text{WR}}$	CPU WRITE INPUT	WR ENABLE	WRITE ENABLE
INT	INTERRUPT	SEEK/STEP	SEEK/STEP
$A_{1,0}$	REGISTER SELECT	DIRECTION	DIRECTION
INSYNC	READ DATA INSYNC	LOAD HEAD	LOAD HEAD
$\overline{\text{CS}}$	CHIP SELECT	LOW CURRENT	LOW CURRENT

BLOCK DIAGRAM



April, 1978

intel®

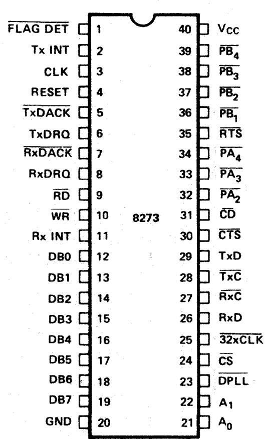
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8273
PROGRAMMABLE HDLC/SDLC PROTOCOL CONTROLLER

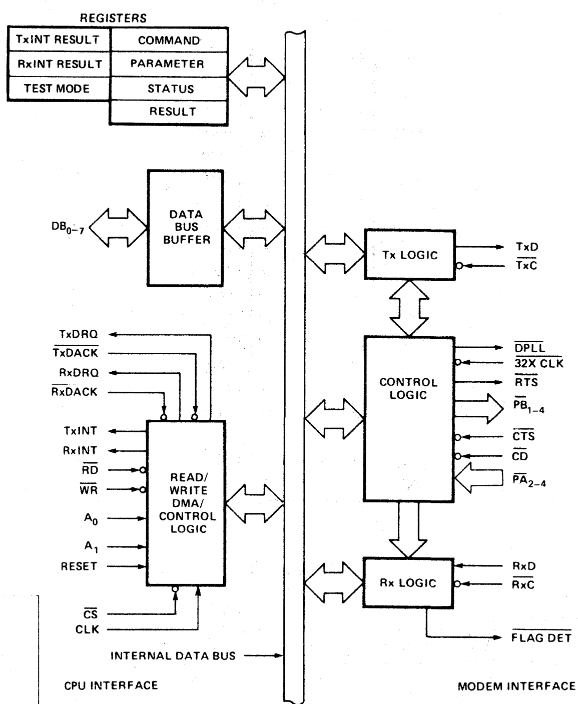
- **HDLC/SDLC Compatible**
- **Frame Level Commands**
- **Full Duplex, Half Duplex, or Loop SDLC Operation**
- **Up to 64K Baud Transfers**
- **Two User Programmable Modem Control Ports**
- **Automatic FCS (CRC) Generation and Checking**
- **Programmable NRZI Encode/Decode**
- **N-Bit Reception Capability**
- **Digital Phase Locked Loop Clock Recovery**
- **Minimum CPU Overhead**
- **Fully Compatible with 8080/8085 CPUs**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8273 Programmable HDLC/SDLC Protocol Controller is a dedicated device designed to support the ISO/CCITT's HDLC and IBM's SDLC communication line protocols. It is fully compatible with Intel's new high performance microcomputer systems such as the MCS-85™. A frame level command set is achieved by a unique microprogrammed dual processor chip architecture. The processing capability supported by the 8273 relieves the system CPU of the low level real-time tasks normally associated with controllers.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

DB0–DB7	DATA BUS (8 BITS)	CS	CHIP SELECT
FLAG DET	FLAG DETECT	32xCLK	32 TIMES CLOCK
TxINT	TRANSMITTER INTERRUPT	Rx D	RECEIVER DATA
CLK	CLOCK INPUT	Rx C	RECEIVER CLOCK
RESET	RESET	Tx C	TRANSMITTER CLOCK
Tx DACK	TRANSMITTER DMA ACKNOWLEDGE	Tx D	TRANSMITTER DATA
TxDRQ	TRANSMITTER DMA REQUEST	CTS	CLEAR TO SEND
RD	READ INPUT	CD	CARRIER DETECT
WR	WRITE INPUT	PA_2–PA_4	GP INPUT PORTS
Rx DACK	RECEIVER DMA ACKNOWLEDGE	PB_1–PB_4	GP OUTPUT PORTS
Rx DRQ	RECEIVER DMA REQUEST	RTS	REQUEST TO SEND
Rx INT	RECEIVER INTERRUPT	V_{CC}	+5 VOLT SUPPLY
A0–A1	COMMAND REGISTER SELECT ADDRESS	GND	GROUND
DPLL	DIGITAL PHASE LOCKED LOOP		

February, 1978

intel®

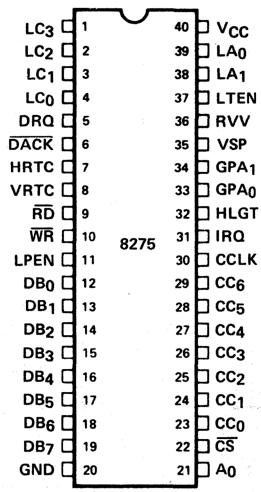
PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8275
PROGRAMMABLE CRT CONTROLLER

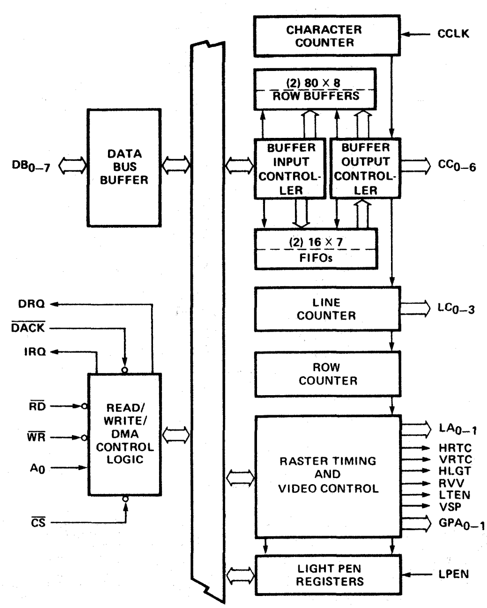
- **Programmable Screen and Character Format**
- **6 Independent Visual Field Attributes**
- **11 Visual Character Attributes (Graphic Capability)**
- **Cursor Control (4 Types)**
- **Light Pen Detection and Registers**
- **Fully MCS-80™ and MCS-85™ Compatible**
- **Dual Row Buffers**
- **Programmable DMA Burst Mode**
- **Single +5V Supply**
- **40-Pin Package**

The Intel® 8275 Programmable CRT Controller is a single chip device to interface CRT raster scan displays with Intel® microcomputer systems. Its primary function is to refresh the display by buffering the information from main memory and keeping track of the display position of the screen. The flexibility designed into the 8275 will allow simple interface to almost any raster scan CRT display with a minimum of external hardware and software overhead.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

DB_{0-1}	BI-DIRECTIONAL DATA BUS	LC_{0-3}	LINE COUNTER OUTPUTS
DRQ	DMA REQUEST OUTPUT	LA_{0-1}	LINE ATTRIBUTE OUTPUTS
$\overline{DACK}$	DMA ACKNOWLEDGE INPUT	HRTC	HORIZONTAL RETRACE OUTPUT
IRQ	INTERRUPT REQUEST OUTPUT	VRTC	VERTICAL RETRACE OUTPUT
$\overline{RD}$	READ STROBE INPUT	HLGT	HIGHLIGHT OUTPUT
$\overline{WR}$	WRITE STROBE INPUT	RVV	REVERSE VIDEO OUTPUT
A_0	REGISTER ADDRESS INPUT	LTEN	LIGHT ENABLE OUTPUT
CS	CHIP SELECT INPUT	VSP	VIDEO SUPPRESS OUTPUT
CCLK	CHARACTER CLOCK INPUT	GPA_{0-1}	GENERAL PURPOSE ATTRIBUTE OUTPUTS
CC_{0-6}	CHARACTER CODE OUTPUTS	LPEN	LIGHT PEN INPUT

intel®

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

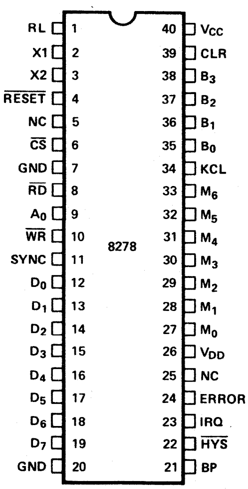
8278
PROGRAMMABLE KEYBOARD INTERFACE

- **Simultaneous Keyboard and Display Operations**
- **Interface Signals for Contract and Capacitive Coupled Keyboards**
- **128-Key Scanning Logic**
- **10.7 msec Matrix Scan Time for 128 Keys and 6 MHz Clock**
- **8-Character Keyboard FIFO**
- **N-Key Rollover with Programmable Error Mode on Multiple New Closures**
- **16- or 18-Character 7-Segment Display Interface**
- **Right or Left Entry Display RAM**
- **Depress/Release Mode Programmable**
- **Interrupt Output on Key Entry**

The Intel® 8278 is a general purpose programmable keyboard and display interface device designed for use with 8-bit microprocessors such the MDS-80™ and MCS-85™. The keyboard portion can provide a scanned interface to 128-key contact or capacitive-coupled keyboards. The keys are fully debounced with N-key rollover and programmable error generation on multiple new key closures. Keyboard entries are stored in an 8-character FIFO with overrun status indication when more than 8 characters are entered. Key entries set an interrupt request output to the master CPU.

The display portion of the 8278 provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric displays and simple indicators may be used. The 8278 has a 16X4 display RAM which can be loaded or interrogated by the CPU. Both right entry calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

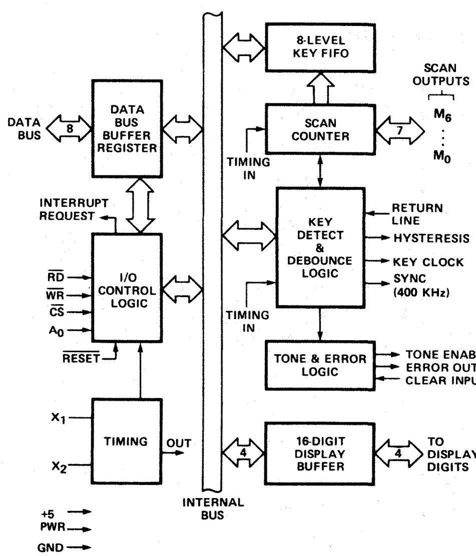
PIN CONFIGURATION



PIN NAMES

D_7-D_0	DATA BUS
$\overline{RD}$, $\overline{WR}$	READ, WRITE STROBES
$\overline{CS}$	CHIP SELECT
A_0	CONTROL/DATA SELECT
$\overline{RESET}$	RESET INPUT
X_1, X_2	FREQ. REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT CLOCK
RL	KEYBOARD RETURN LINE
CLR	CLEAR ERROR
KCL	KEY CLOCK
M_6–M_0	MATRIX SCAN LINES
B_3-B_0	DISPLAY OUTPUTS
ERROR	ERROR SIGNAL
IRQ	INTERRUPT REQUEST
$\overline{HYS}$	HYSTERESIS
BP	TONE ENABLE

BLOCK DIAGRAM



intel®

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

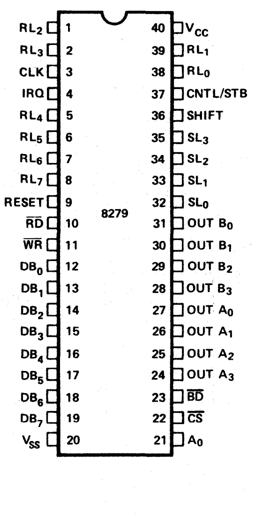
8279/8279-5
PROGRAMMABLE KEYBOARD/DISPLAY INTERFACE

- **MCS-85™ Compatible 8279-5**
- **Simultaneous Keyboard Display Operations**
- **Scanned Keyboard Mode**
- **Scanned Sensor Mode**
- **Strobed Input Entry Mode**
- **8-Character Keyboard FIFO**
- **2-Key Lockout or N-Key Rollover with Contact Debounce**
- **Dual 8- or 16-Numerical Display**
- **Single 16-Character Display**
- **Right or Left Entry 16-Byte Display RAM**
- **Mode Programmable from CPU**
- **Programmable Scan Timing**
- **Interrupt Output on Key Entry**

The Intel® 8279 is a general purpose programmable keyboard and display I/O interface device designed for use with Intel® microprocessors. The keyboard portion can provide a scanned interface to a 64-contact key matrix. The keyboard portion will also interface to an array of sensors or a strobed interface keyboard, such as the hall effect and ferrite variety. Key depressions can be 2-key lockout or N-key rollover. Keyboard entries are debounced and strobed in an 8-character FIFO. If more than 8 characters are entered, overrun status is set. Key entries set the interrupt output line to the CPU.

The display portion provides a scanned display interface for LED, incandescent, and other popular display technologies. Both numeric and alphanumeric segment displays may be used as well as simple indicators. The 8279 has 16X8 display RAM which can be organized into dual 16X4. The RAM can be loaded or interrogated by the CPU. Both right entry, calculator and left entry typewriter display formats are possible. Both read and write of the display RAM can be done with auto-increment of the display RAM address.

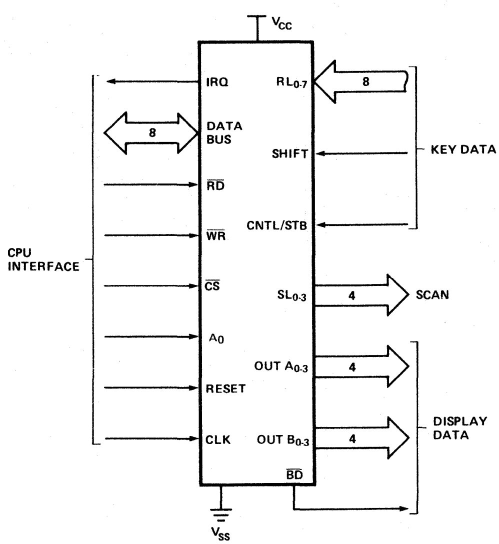
PIN CONFIGURATION



PIN NAMES

DB_{0-7}	I/O	DATA BUS (BI-DIRECTIONAL)
CLK	I	CLOCK INPUT
RESET	I	RESET INPUT
$\overline{CS}$	I	CHIP SELECT
$\overline{RD}$	I	READ INPUT
$\overline{WR}$	I	WRITE INPUT
A_0	I	BUFFER ADDRESS
IRQ	O	INTERRUPT REQUEST OUTPUT
SL_{0-3}	O	SCAN LINES
RL_{0-7}	I	RETURN LINES
SHIFT	I	SHIFT INPUT
CNTL/STB	I	CONTROL/STROBE INPUT
OUT A_{0-3}	O	DISPLAY (A) OUTPUTS
OUT B_{0-3}	O	DISPLAY (B) OUTPUTS
$\overline{BD}$	O	BLANK DISPLAY OUTPUT

LOGIC SYMBOL



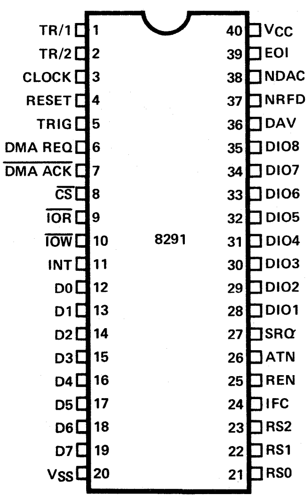
intel®

8291
GPIB TALKER/LISTENER

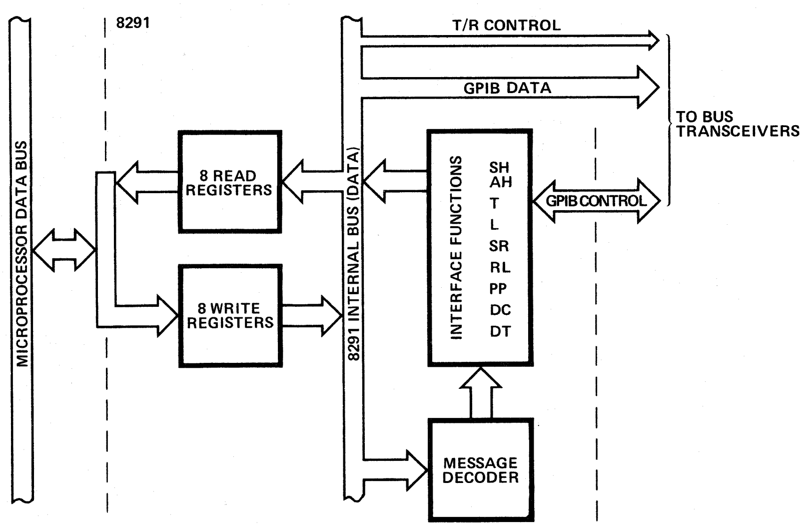
- Complete source and acceptor handshake.
- Complete talker and listener functions with extended addressing.
- Service request, parallel poll, device clear, device trigger, remote/local functions.
- Selectable Interrupts
- On chip primary and secondary address recognition.
- Automatic handling of addressing and handshake protocol.
- Provision for software implementation of additional features.
- Designed to interface 8-Bit Micropro-processors (e.g., 8080, 8085, 8048) to an IEEE Standard 488 Digital Interface Bus.
- 16 Registers (8 Read, 8 Write), 2 for Data Transfer, the Rest for Interface Function Control, Status, etc.
- Directly Interfaces to External Transceivers for Connection to the GPIB Bus.
- Provides Three Addressing Modes, Allowing the Chip to be Addressed Either as a Major or a Minor Talker/-Listener with Primary or Secondary Addressing.
- DMA Handshake Provision Allows for Bus Transfers without CPU Intervention.
- Trigger Output Pin Allows for Triggering of any Device in the System Without CPU Intervention.
- On Chip EOS Message Recognition Facilitates Handling of Multi-Byte Transfers.

The 8291 GPIB TALKER/LISTENER is a microprocessor-controlled chip designed to interface 8-bit microprocessors (e.g., 8080, 8085, 8048) to an IEEE Standard 488 Instrumentation Interface Bus. It implements all of the Standard's talker/listener interface functions.

PIN CONFIGURATION



BLOCK DIAGRAM



intel®

PRELIMINARY

Notice: This is not a final specification. Some parametric limits are subject to change.

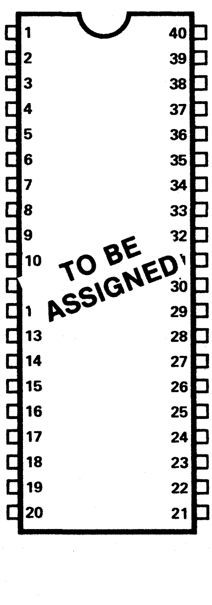
8292
GPIB CONTROLLER

FEATURES:

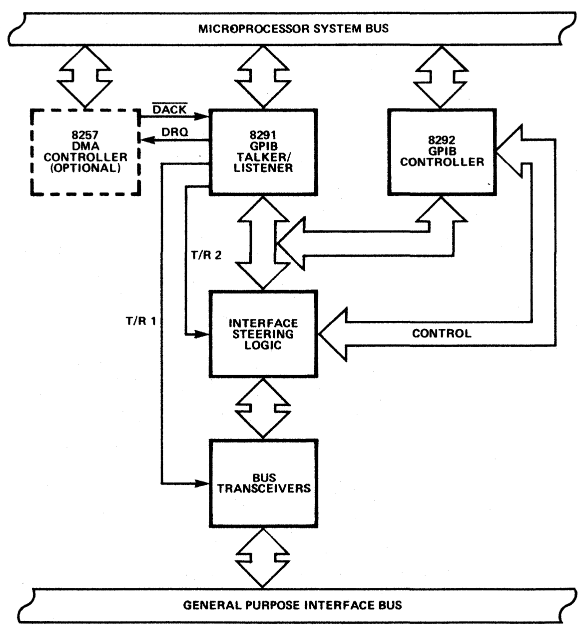
- **Complete IEEE Standard 488 Controller Function.**
- **Interface Clear (IFC) Sending Capability Allows for Seizure of Control and/or Initialization of the Bus.**
- **Responds to Service Requests (SRQ).**
- **Sends (REN), Allowing Instruments to Switch to Remote Control.**
- **Complete Implementation of Transfer Control Protocol.**
- **Synchronous Control Seizure Prevents the Destruction of any Data Transmission in Progress.**
- **Connects with the 8291 to Form a Complete IEEE Standard 488 Interface Talker/Listener/Controller.**

The 8292 GPIB CONTROLLER is a microprocessor-controlled chip designed to connect with the 8291 GPIB TALKER/LISTENER to implement the full IEEE Standard 488 controller function, including transfer control protocol.

PIN CONFIGURATION



8291, 8292 SYSTEM DIAGRAM



intel®

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

8294 DATA ENCRYPTION UNIT

- Certified by National Bureau of Standards
- 80-Byte/Sec Data Conversion Rate
- 64-Bit Data Encryption Using 56-Bit Key
- DMA Interface
- 3 Interrupt Outputs to Aid in Loading and Unloading Data
- 7-Bit User Output Port
- Single 5V ± 10% Power Supply
- Peripheral to MCS-85™, MCS-80™ and MCS-48™ Processors
- Implements Federal Information Processing Data Encryption Standard
- Encrypt and Decrypt Modes Available

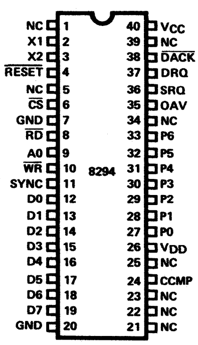
DESCRIPTION

The Intel® 8294 Data Encryption Unit (DEU) is a microprocessor peripheral device designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard. The DEU operates on 64-bit text words using a 56-bit user-specified key to produce 64-bit cipher words. The operation is reversible: if the cipher word is operated upon, the original text word is produced. The algorithm itself is permanently contained in the 8294; however, the 56-bit key is user-defined and may be changed at any time.

The 56-bit key and 64-bit message data are transferred to and from the 8294 in 8-bit bytes by way of the system data bus. A DMA interface and three interrupt outputs are available to minimize software overhead associated with data transfer. Also, by using the DMA interface two or more DEUs may be operated in parallel to achieve effective system conversion rates which are virtually any multiple of 120 bytes/second. The 8294 also has a 7-bit TTL compatible output port for user-specified functions.

Because the 8294 implements the NBS encryption algorithm it can be used in a variety of Electronic Funds Transfer applications as well as other electronic banking and data handling applications where data must be encrypted.

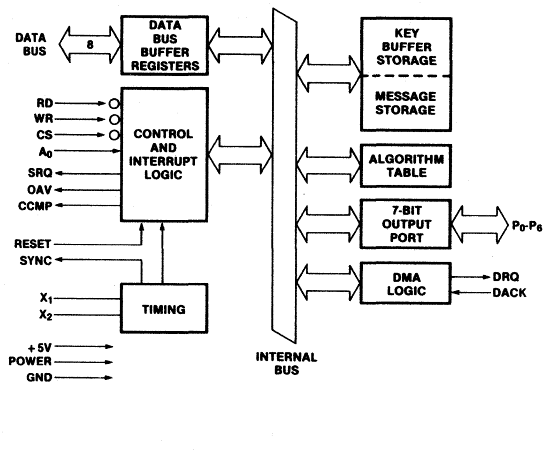
PIN CONFIGURATION



PIN NAMES

PIN NAME	FUNCTION
D_7-D_0	DATA BUS
RD, WR	READ, WRITE STROBES
CS	CHIP SELECT
A_0	CONTROL/DATA SELECT
RESET	RESET INPUT
X_1, X_2	FREQUENCY REFERENCE INPUT
SYNC	HIGH FREQUENCY OUTPUT
DRQ, DACK	DMA REQUEST, DMA ACKNOWLEDGE
SRQ, OAV, CCMP	INTERRUPT REQUEST OUTPUTS
P_6-P_0	OUTPUT PORT LINES
V_{CC}, V_{DD}, GND	+5V POWER, GND

BLOCK DIAGRAM



April 1978

ADVANCE INFORMATION
Characteristics are subject to change without notice.

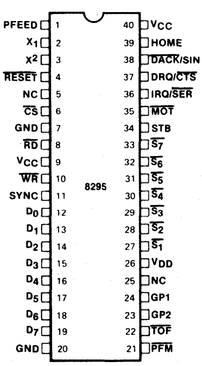
8295 DOT MATRIX PRINTER CONTROLLER

- Interfaces Dot Matrix Printers to MCS-48™, MCS-80™, MCS-85™ Systems
- 40 Character Buffer On Chip
- Serial or Parallel Communication with Host
- DMA Transfer Capability
- Programmable Character Density (10 or 12 Chararcters/Inch)
- Programmable Print Intensity
- Single or Double Width Printing
- Programmable Multiple Line Feeds
- 3 Tabulations
- 2 General Purpose Outputs

The Intel® 8295 Dot Matrix Printer Controller provides an interface for microprocessors to the LRC 7040 Series dot matrix impact printers. It may also be used as an interface to other similar printers.

The chip may be used in a serial or parallel communication mode with the host processor. Furthermore, it provides internal buffering of up to 40 characters and contains a 7×7 matrix character generator accommodating 64 ASCII characters.

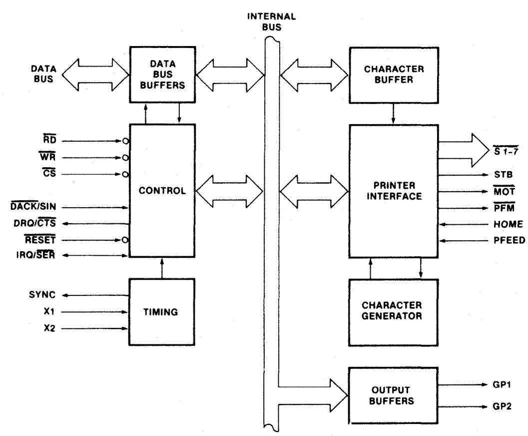
PIN CONFIGURATION



PIN NAMES

PIN NAME	FUNCTION
D_0-D_7	DATA BUS
$\overline{RD}$, $\overline{WR}$	READ, WRITE STROBES
$\overline{CS}$	CHIP SELECT
$\overline{RESET}$	RESET INPUT
X_1, X_2	FREQUENCY REFERENCE INPUTS
SYNC	HIGH FREQUENCY OUTPUT
$\overline{MOT}$, $\overline{PFM}$	MAIN, PAPER FEED MOTOR DRIVES
DRQ, $\overline{DACK}$	DMA REQUEST, ACKNOWLEDGE
SIN, $\overline{CTS}$	SERIAL INPUT, CLEAR-TO-SEND
IRQ/SER	INTERRUPT REQUEST, SERIAL GROUND
$\overline{S_1}$-$\overline{S_7}$	SOLENOID DRIVE OUTPUTS
PFEED	PAPER FEED INPUT
HOME, $\overline{TOF}$	HOME, TOP-OF-FORM INPUTS
STB	SOLENOID STROBE OUTPUT
GP1, GP2	GENERAL PURPOSE OUTPUTS
V_{CC}, V_{DD}, GND	+5V POWER, GND

BLOCK DIAGRAM



CHAPTER 5

Device Specifications

- MCS-86™
- MCS-85™*
- Peripherals**
- **Static RAMs*****
- ROMs/EPROMs***

*For complete specifications refer to the Intel MCS-85 User's Manual.
**For complete specifications refer to the Intel Peripheral Design Handbook.
***For complete specifications refer to the 1978 Intel Data Catalog.

intel®

2114
1024 X 4 BIT STATIC RAM

	2114-2	2114-3	2114	2114L2	2114L3	2114L
Max. Access Time (ns)	200	300	450	200	300	450
Max. Power Dissipation (mw)	525	525	525	370	370	370

- **High Density 18 Pin Package**
- **Identical Cycle and Access Times**
- **Single +5V Supply**
- **No Clock or Timing Strobe Required**
- **Completely Static Memory**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**
- **Pin-Out Compatible with 3605 and 3625 Bipolar PROMs**

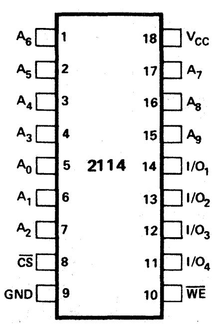
The Intel® 2114 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 2114 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives. The 2114 is placed in an 18-pin package for the highest possible density.

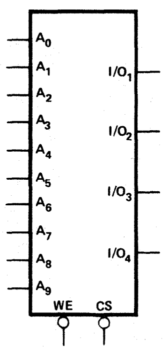
It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. A separate Chip Select ($\overline{CS}$) lead allows easy selection of an individual package when outputs are or-tied.

The 2114 is fabricated with Intel's N-channel Silicon-Gate technology — a technology providing excellent protection against contamination permitting the use of low cost plastic packaging.

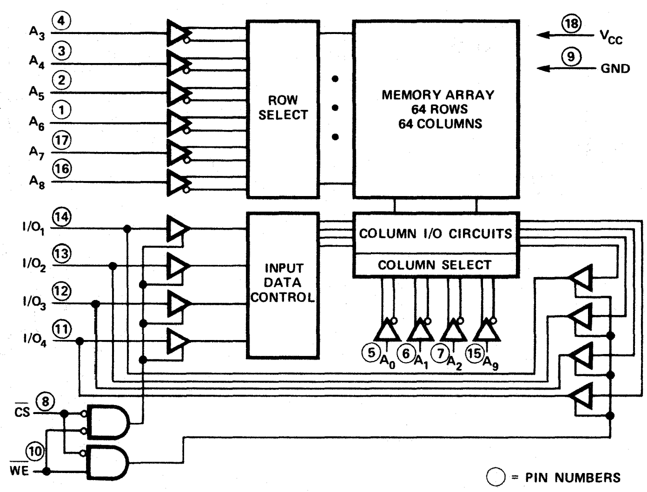
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
I/O_1–I/O_4	DATA INPUT/OUTPUT	

intel®

MILITARY TEMP. ADVANCE INFORMATION
Characteristics are subject to change without notice.

M2114
1024 X 4 BIT STATIC RAM

	M2114
Max. Access Time (ns)	450
Max. Power Dissipation (mW)	550

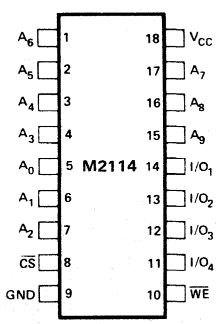
- **High Density 18 Pin Package**
- **Identical Cycle and Access Times**
- **Single +5V Supply**
- **No Clock or Timing Strobe Required**
- **Completely Static Memory**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**
- **Military Temperature Range -55°C to +125°C**

The Intel® M2114 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

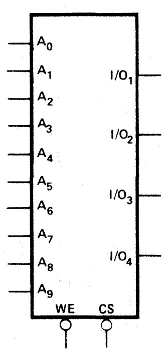
The M2114 is designed for memory applications where high performance, large bit storage, and simple interfacing are important design objectives. The M2114 is placed in an 18-pin package for the highest possible density.

It is directly TTL compatible in all respects: inputs, outputs, and a single +5V supply. A separate Chip Select ($\overline{CS}$) lead allows easy selection of an individual package when outputs are OR-tied.

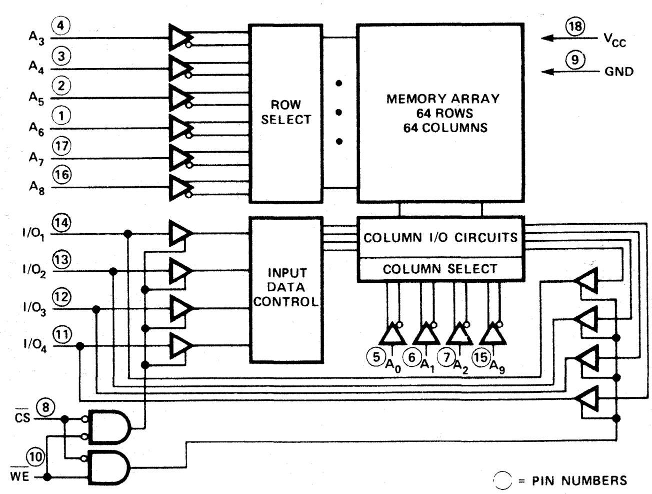
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

$A_0 - A_9$	ADDRESS INPUTS	V_{CC} POWER (+5V)
$\overline{WE}$	WRITE ENABLE	GND GROUND
$\overline{CS}$	CHIP SELECT	
$I/O_1 - I/O_4$	DATA INPUT/OUTPUT	

2142
1024 X 4 BIT STATIC RAM

	2142-2	2142-3	2142	2142L2	2142L3	2142L
Max. Access Time (ns)	200	300	450	200	300	450
Max. Power Dissipation (mw)	525	525	525	370	370	370

- **High Density 20 Pin Package**
- **Access Time Selections From 200-450ns**
- **Identical Cycle and Access Times**
- **Low Operating Power Dissipation .1mW/Bit Typical**
- **Single +5V Supply**
- **No Clock or Timing Strobe Required**
- **Completely Static Memory**
- **Directly TTL Compatible: All Inputs and Outputs**
- **Common Data Input and Output Using Three-State Outputs**

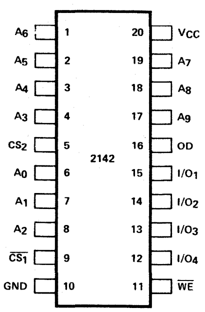
The Intel® 2142 is a 4096-bit static Random Access Memory organized as 1024 words by 4-bits using N-channel Silicon-Gate MOS technology. It uses fully DC stable (static) circuitry throughout — in both the array and the decoding — and therefore requires no clocks or refreshing to operate. Data access is particularly simple since address setup times are not required. The data is read out nondestructively and has the same polarity as the input data. Common input/output pins are provided.

The 2142 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.

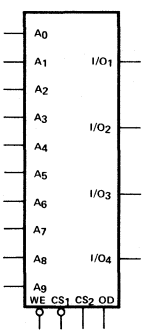
The 2142 is placed in a 20-pin package. Two Chip Selects ($\overline{CS}_1$ and CS_2) are provided for easy and flexible selection of individual packages when outputs are OR-tied. An Output Disable is included for direct control of the output buffers.

The 2142 is fabricated with Intel's N-channel Silicon-Gate technology — a technology providing excellent protection against contamination permitting the use of low cost plastic packaging.

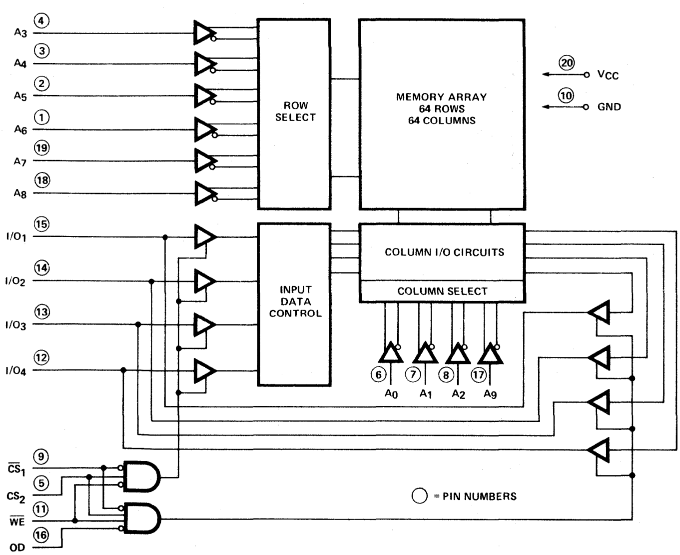
PIN CONFIGURATION



LOGIC SYMBOL



BLOCK DIAGRAM



PIN NAMES

A0–A9	ADDRESS INPUTS	OD	OUTPUT DISABLE
$\overline{WE}$	WRITE ENABLE	VCC	POWER (+5V)
$\overline{CS}_1$, CS_2	CHIP SELECT	GND	GROUND
I/O1–I/O4	DATA INPUT/OUTPUT		

CHAPTER 5

Device Specifications

- MCS-86™
- MCS-85™*
- Peripherals**
- Static RAMs***
- ROMs/EPROMs***

*For complete specifications refer to the Intel MCS-85 User's Manual.
**For complete specifications refer to the Intel Peripheral Design Handbook.
***For complete specifications refer to the 1978 Intel Data Catalog.

intel®

ADVANCE INFORMATION
Characteristics are subject to change without noti

2332
32K (4K x 8) ROM

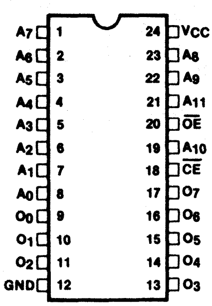
- Single +5V ± 10% Power Supply
- Pin Compatible to Intel® 2716 and 2732 EPROMs
- 300ns Max. Access Time
- Low Power Dissipation: 40mA Max. Average Current 15mA Max. Standby Current
- Edge Enabled With Static Array
- Inputs and Outputs TTL Compatible
- Three-State Output for Direct Bus Interface
- Output Enable for MCS-85™ and MCS-86™ Compatibility

The Intel® 2332 is a single +5V supply, 32,768-bit N-channel MOS read only memory organized as 4096 words by 8-bits. It has static memory cells and clocked peripheral circuitry, giving a fast device access time with low active power dissipation. The 2332 features an automatic standby power mode. When deselected by $\overline{CE}$, the active power dissipation is reduced from 40mA to 15mA, a 60% reduction.

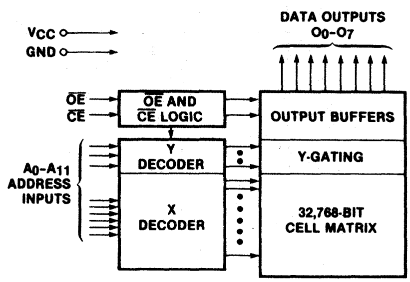
The 2332 is ideal for microprocessor systems, especially those with common input and output bus structures. The separate output control, $\overline{OE}$, eliminates bus contention. The 300ns access time, three-state outputs, address latches, and TTL input/output levels further simplify system design.

A cost effective system development program may be implemented by using the pin compatible Intel® 2732, 32K UV EPROM for prototyping and the 2332 ROM for volume production. The 2732 is fully compatible to the 2332 in all respects.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0-A_{10}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0-O_7	OUTPUTS

intel®

ADVANCE INFORMATION
Characteristics are subject to change without notice

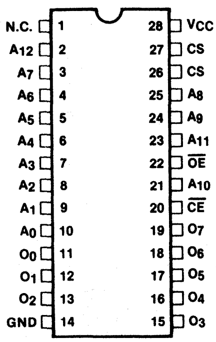
2364
64K (8K × 8) BIT ROM

- **Single +5V ±10% Power Supply**
- **Pin Compatible to Intel® 2732 EPROM**
- **Low Power Mode**
- **Inputs and Outputs TTL Compatible**
- **Three-State Output for Direct Bus Interface**
- **MCS-80 and MCS-85 Compatible**

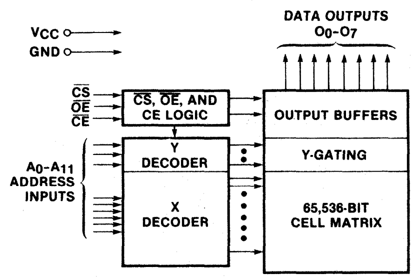
The Intel® 2364 is a single +5V, 65,536-bit N-channel MOS read only memory organized as 8192 words by 8 bits. Its high bit density is ideal for large, non-volatile data storage, such as program storage. The three-state outputs and TTL input/output levels allow for direct interface with common bus structures. The 2364 has a low power mode which reduces the active power dissipation by over 50%.

A cost-effective system development program may be implemented by using the Intel® 2732 32K UV EPROM for prototyping and the 2364 ROM for production. The lower 24 pins of the 2364 are the same as the 2732 to facilitate board designs in making the transition from EPROM to ROM.

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0-A_{12}	ADDRESSES
$\overline{OE}$	OUTPUT ENABLE
$\overline{CE}$	CHIP ENABLE
CS	CHIP SELECT
N.C.	NO CONNECTION

intel®

2616*
16K (2K × 8) FACTORY PROGRAMMABLE PROM

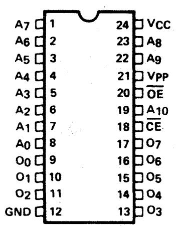
- **Single +5V Power Supply**
- **Low Power Dissipation**
  525 mW Max. Active Power
  132 mW Max. Standby Power
- **Pin Compatible to Intel® 2716 EPROM and 2316E ROM**
- **Fast Access Time — 450 ns Max.**
- **Inputs and Outputs TTL Compatible**
- **Completely Static**

The Intel® 2616 is a 16,384-bit, one-time factory-programmable MOS PROM organized as 2048 words by 8 bits. The 2616 operates from a single +5V power supply, has a static standby mode, and is TTL input/output compatible. It is specified over the 0°C to 70°C operating temperature with 5% power supply variation.

A cost-effective system development program may be implemented quickly into production by using the Intel® 2716 EPROM for pattern experimentation, the 2616 for fast first incremental 2316E ROM delivery, and the 2316E for volume production. The 2616 is fully compatible to the 2716 in all respects. The fast factory 2616 code pattern turnaround time gives rapid transition from EPROM to ROM for production.

The 2616 has a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW, while the maximum standby power dissipation is only 132 mW — a 75% saving.

PIN CONFIGURATION*



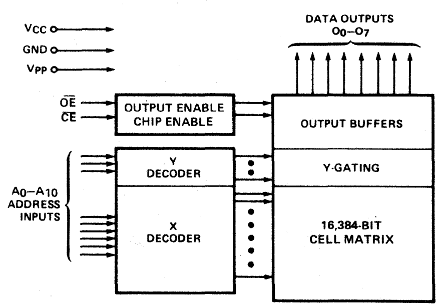
MODE SELECTION

MODE \ PINS	$\overline{CE}$ (18)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z

PIN NAMES

A_0–A_9	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS

BLOCK DIAGRAM



**Pin 18 and pin 20 have been named to conform with the entire family of 16K, 32K, and 64K EPROMs and ROMs.*

2716*
16K (2K × 8) UV ERASABLE PROM

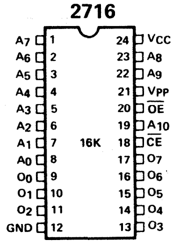
- **Fast Access Time**
  - **— 350 ns Max. 2716-1**
  - **— 390 ns Max. 2716-2**
  - **— 450 ns Max. 2716**
- **Single +5V Power Supply**
- **Low Power Dissipation**
  - **— 525 mW Max. Active Power**
  - **— 132 mW Max. Standby Power**
- **Pin Compatible to Intel® 5V ROMs (2316E, 2332, and 2364) and 2732 EPROM**
- **Simple Programming Requirements**
  Single Location Programming
  Programs with One 50 ms Pulse
- **Inputs and Outputs TTL Compatible during Read and Program**
- **Completely Static**

The Intel® 2716 is a 16,384-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2716 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical. For production quantities, the 2716 user can convert rapidly to Intel's pin-for-pin compatible 16K ROM (the 2316E) or the new 32K and 64K ROMs (the 2332 and 2364 respectively).

The 2716, with its single 5-volt supply and with an access time up to 350 ns, is ideal for use with the newer high performance +5V microprocessors such as Intel's 8085 and 8086. The 2716 is also the first EPROM with a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW while the maximum standby power dissipation is only 132 mW, a 75% savings.

The 2716 has the simplest and fastest method yet devised for programming EPROMs — single pulse TTL level programming. No need for high voltage pulsing because all programming controls are handled by TTL signals. Now, it is possible to program on-board, in the system, in the field. Program any location at any time — either individually, sequentially or at random, with the 2716's single address location programming. Total programming time for all 16,384 bits is only 100 seconds.

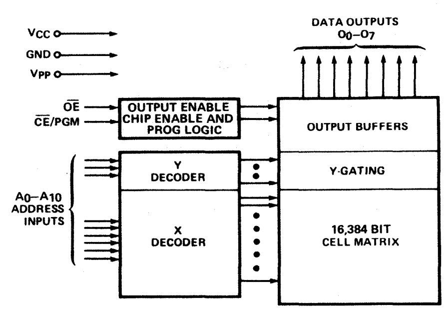
PIN CONFIGURATION*



MODE SELECTION

PINS / MODE	CE/PGM (18)	OE (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

BLOCK DIAGRAM



PIN NAMES

A_0–A_9	ADDRESSES
CE/PGM	CHIP ENABLE/PROGRAM
OE	OUTPUT ENABLE
O_0–O_7	OUTPUTS

**Pin 18 and pin 20 have been renamed to conform with the entire family of 16K, 32K, and 64K EPROMs and ROMs. The die, fabrication process, and specifications remain the same and are totally uneffected by this change.*

intel®

2758*
8K (1K × 8) UV ERASABLE LOW POWER PROM

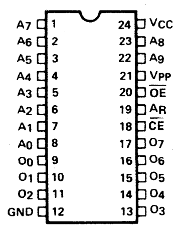
- **Single + 5V Power Supply**
- **Simple Programming Requirements**
  Single Location Programming
  Programs with One 50 ms Pulse
- **Low Power Dissipation**
  525 mW Max. Active Power
  132 mW Max. Standby Power
- **Fast Access Time: 450 ns Max. in Active and Standby Power Modes**
- **Inputs and Outputs TTL Compatible during Read and Program**
- **Completely Static**
- **Three-State Outputs for OR-Ties**

The Intel® 2758 is a 8192-bit ultraviolet erasable and electrically programmable read-only memory (EPROM). The 2758 operates from a single 5-volt power supply, has a static standby mode, and features fast single address location programming. It makes designing with EPROMs faster, easier and more economical. The total programming time for all 8192 bits is 50 seconds.

The 2758 has a static standby mode which reduces the power dissipation without increasing access time. The maximum active power dissipation is 525 mW, while the maximum standby power dissipation is only 132 mW, a 75% savings. Power-down is achieved by applying a TTL-high signal to the $\overline{CE}$ input.

A 2758 system may be designed for total upwards compatibility with Intel's 16K 2716 EPROM (see Applications Note 30). The 2758 maintains the simplest and fastest method yet devised for programming EPROMs — single pulse TTL-level programming. There is no need for high voltage pulsing because all programming controls are handled by TTL signals. Now it is possible to program on-board, in the system, in the field. Program any location at any time — either individually, sequentially, or at random, with the single address location programming.

PIN CONFIGURATION*



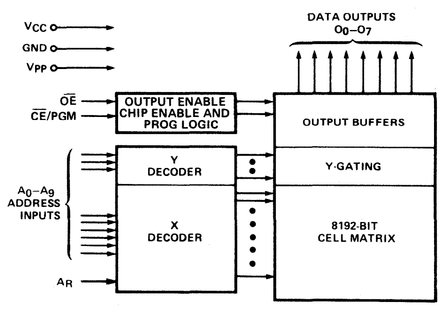
PIN NAMES

A_0–A_9	ADDRESSES
$\overline{CE}$/PGM	CHIP ENABLE/PROGRAM
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS
A_R	SELECT REFERENCE INPUT LEVEL

MODE SELECTION

MODE \ PINS	$\overline{CE}$/PGM (18)	A_R (19)	$\overline{OE}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Standby	V_{IH}	V_{IL}	Don't Care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IL}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IL}	V_{IH}	+25	+5	High Z

BLOCK DIAGRAM



**Pin 18 and pin 20 have been renamed to conform with the entire family of 16K, 32K, and 64K EPROMs and ROMs. The die, fabrication process, and specifications remain the same and are totally uneffected by this change.*

CHAPTER 6

Development Aids *

*For complete specifications on the Intellec Series II Development Systems contact the Intel Literature Department
3065 Bowers Ave.
Santa Clara, CA. 95051
(408) 987-6475

intel®

MODEL 230
INTELLEC® SERIES II
MICROCOMPUTER DEVELOPMENT SYSTEM

Complete Microcomputer Development Center for Intel MCS-80, MCS-85 and MCS-48 microprocessor families

Integral CRT with detachable upper/lower case "typewriter-style" full ASCII keyboard

64K bytes RAM memory

1 million bytes (expandable to 2.5M bytes) of diskette storage

LSI electronics board with CPU, RAM, ROM, I/O and interrupt circuitry

Built-in interfaces for High-Speed Paper Tape Reader/Punch, Printer and Universal PROM Programmer

Powerful ISIS-II Diskette Operating System Software with Relocating Macro Assembler, Linker and Locater

"Self-Test" Diagnostic capability

Standard MULTIBUS with multiprocessor and DMA capability

Eight-level nested, maskable priority interrupt system

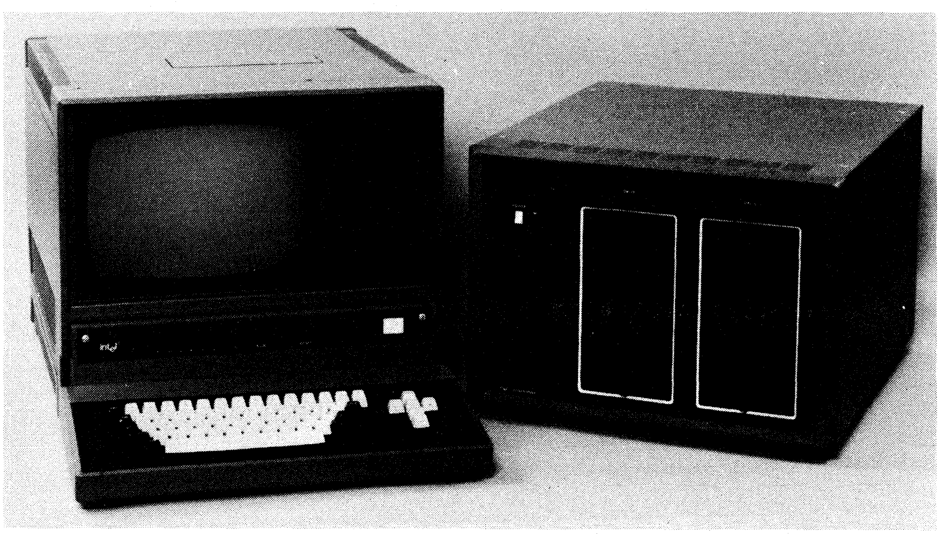
Compatible with standard Intellec/iSBC Expansion Modules

Software compatible with previous Intellec Systems

Supports PL/M and FORTRAN high level languages

The Intellec Series II Model 230 Microcomputer Development System is a complete center for the development of microcomputer-based products. It includes a CPU, 64K bytes of RAM, 4K bytes of ROM memory, a 2000-character CRT, detachable full ASCII keyboard and dual double-density diskette drives providing over 1 million bytes of on-line data storage.

Powerful ISIS-II Diskette Operating System software allows the Model 230 to be used quickly and efficiently for assembly and/or compilation and debugging of programs for Intel's MCS-80, MCS-85 or MCS-48 microprocessor families without the need for handling paper tape. ISIS-II performs all file handling operations for the user, leaving him free to concentrate on the details of his own application. When used in conjunction with an optional in-circuit emulator (ICE™) module, the Model 230 provides all the hardware and software development tools necessary for the rapid development of a microcomputer-based product.



intel®

MDS-311
PL/M-86 HIGH LEVEL PROGRAMMING LANGUAGE

Sophisticated new compiler design allows user to achieve maximum benefits of 8086 capabilities

Language is upward compatible from PL/M-80, assuring MCS™-80/85 design portability

Operation on Intellec® Microcomputer Development System and Intellec® Series II Microcomputer Development System permits MCS™-86 software development without significant hardware reinvestment and retraining

Supports 16-bit signed integer and 32-bit floating point arithmetic

Produces relocatable and linkable object code

Supports full extended addressing features of the 8086 microprocessor

Sophisticated code optimization assures efficient code generation and minimum application memory utilization

Like its counterpart for MCS™-80/85 program development, PL/M-86 is an advanced structured high level programming language. PL/M-86 is a new compiler created specifically for performing software development for the Intel® 8086 Microprocessor.

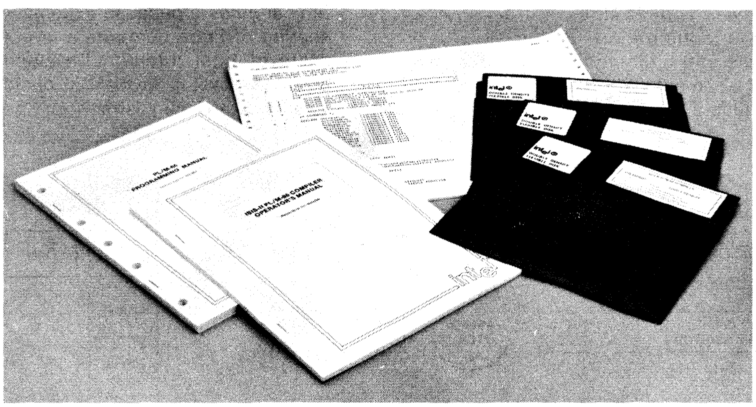
PL/M-86 has significant new capabilities over PL/M-80 that take advantage of the new facilities provided by the 8086 microprocessor, yet the PL/M-86 language remains downward compatible with PL/M-80.

With the exception of interrupts, hardware flags, and time-critical code sequences, PL/M-80 programs may be recompiled under PL/M-86 with little or no conversions required. PL/M-86, like PL/M-80, is easy to learn, facilitates rapid program development, and reduces program maintenance costs.

PL/M is a powerful, structured high level algorithmic language in which program statements can naturally express the program algorithm. This frees the programmer to concentrate on the system implementation without concern for burdensome details of assembly language programming (such as register allocation, meanings of assembler mnemonics, etc.).

The PL/M-86 compiler efficiently converts free-form PL/M language statements into equivalent 8086 machine instructions. Substantially fewer PL/M statements are necessary for a given application than if it were programmed at the assembly language or machine code level.

Since PL/M programs are implementation problem oriented and more compact, use of PL/M results in a high degree of engineering productivity during project development. This translates into significant reductions in initial software development and follow-on maintenance costs for the user.



June, 1978

FEATURES

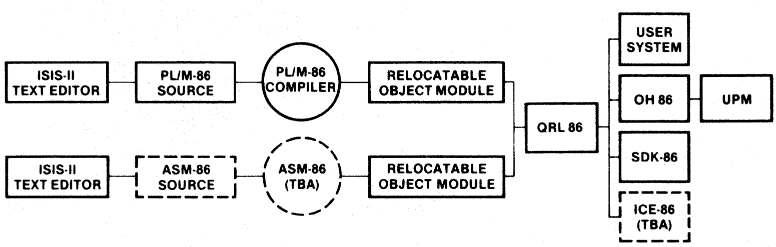
Major features of the Intel PL/M-86 compiler and programming language include:

- **Supports Five Data Types**
  - — Byte: 8-bit unsigned number
  - — Word: 16-bit unsigned number
  - — Integer: 16-bit signed number
  - — Real: 32-bit floating point number
  - — Pointer: 16-bit or 32-bit memory address indicator
- **Two Data Structuring Facilities**
  - — Array: Indexed list of same type data elements
  - — Structure: Named collection of same or different type data elements
  - — Combinations of Each: Arrays of structures or structures of arrays
- **Block Structure**
  - — Permits use of structured programming techniques
- **Relocatable and Linkable Object Code**
  - — Permits PL/M-86 programs to be developed and debugged in small modules. These modules can be easily linked with other modules and/or library routines to form a complete application system.
- **Built-In String Handling Facilities**
  - — Operates on byte strings or word strings
  - — Six Functions: MOVE, COMPARE, TRANSLATE, SEARCH, SKIP, and SET.
- **Automatic Support for 8086 Extended Addressing**
  - — Three compiler options for programs from 128K bytes to 1-Megabyte in size
  - — Language transparency for extended addressing
- **Support for ICE-86™ and Symbolic Debugging**
  - — Debug option for inclusion of symbol table in object modules for in-circuit emulation with symbolic debugging
- **Numerous Compiler Options**
  - — A host of 26 sets of compiler options including:
    - Conditional compilation
    - Included file or copy facility
    - Two levels of optimization
    - Intra-module and inter-module cross reference
    - Arbitrary placement of compiler and user files on any available combination of disk drives
- **Reentrant and Interrupt Procedures**
  - — May be specified as user options

BENEFITS

PL/M-86 is designed to be an efficient, cost-effective solution to the special requirements of MCS™-86 Microcomputer Software Development, as illustrated by the following benefits of PL/M-86 use:

- **Low Learning Effort —** PL/M-86 is easy to learn and to use, even for the novice programmer.
- **Earlier Project Completion —** Critical projects are completed much earlier than otherwise possible because PL/M-86, a structured high-level language, increases programmer productivity.
- **Lower Development Cost —** Increases in programmer productivity translate immediately into lower software development costs because less programming resources are required for a given programmed function.
- **Increased Reliability —** PL/M-86 is designed to aid in the development of reliable software (PL/M-86 programs are simple statements of the program algorithm). This substantially reduces the risk of costly correction of errors in systems that have already reached full production status, as the more simply stated the program is, the more likely it is to perform its intended function.
- **Easier Enhancements and Maintenance —** Programs written in PL/M tend to be self-documenting, thus easier to read and understand. This means it is easier to enhance and maintain PL/M programs as the system capabilities expand and future products are developed.
- **Simpler Project Development —** The Intellec® Development Systems offer a cost-effective hardware base for the development of MCS™-86 designs. PL/M-86 and other elements of ISIS-II and the MCS™-86 Software Development Package are all that is needed for development of software for the 8086 microcomputer. This further reduces development time and costs because expensive (and remote) time sharing of large computers is not required. Present users of Intel Intellec® Development Systems can begin to develop MCS™-86 designs without expensive hardware reinvestment or costly retraining.



SAMPLE PROGRAM

```
STATISTICS:  DO;

/* The procedure in this module computes the mean and variance of an array
of data, X, of length N+1, according to the method of Kahan and Parlett
(University of California, Berkeley, Memo no. UCB/ERL M77/21. */


STAT:  PROCEDURE(X$PTR,N,MEAN$PTR,VARIANCE$PTRO)  PUBLIC;

DECLARE  (X$PTR,MEAN$PTR,VARIANCE$PTRO)  POINTER,
         X  BASED X$PTR (1)  REAL,
         N  INTEGER,
         MEAN  BASED MEAN$PTR  REAL,
         VARIANCE  BASED VARIANCE$PTR  REAL,
         (M,Q,DIFF)  REAL,
         I  INTEGER;

M = X(0);
Q = 0.0;

DO I = 1 TO N;
   DIFF = X(I) - M;
   M = M + DIFF/FLOAT(I + 1);
   Q = Q + DIFF*DIFF*FLOAT(I)/FLOAT(I + 1);
END;

MEAN = M;
VARIANCE = Q/FLOAT(N);

END STAT;


END STATISTICS;
```

SPECIFICATIONS

Operating Environment

Required Hardware

Intellec® Microcomputer Development System
— MDS-800, MDS-888
— Series II

64K Bytes of RAM Memory

Dual Diskette Drives
— Single or Double Density*

System Console
— CRT or Hardcopy Interactive Device

Optional Hardware

Line Printer*
ICE-85™
ICE-86™

Required Software

ISIS-II Diskette Operating System
— Single or Double Density*

Optional Software

SDK-85™ Upload/Download Utility

Documentation Package

PL/M-86 Programming Manual (9800466)

ISIS-II PL/M-86 Compiler Operator's Manual (9800478)

MCS™-86 User's Manual (9800722)

Shipping Media

PL/M-86 is provided as part of the MCS™-86 Software Support Package (MDS-311)

Flexible Diskettes
— Single and Double Density*

*Recommended for optimum use.

PRELIMINARY
Notice: This is not a final specification. Some parametric limits are subject to change.

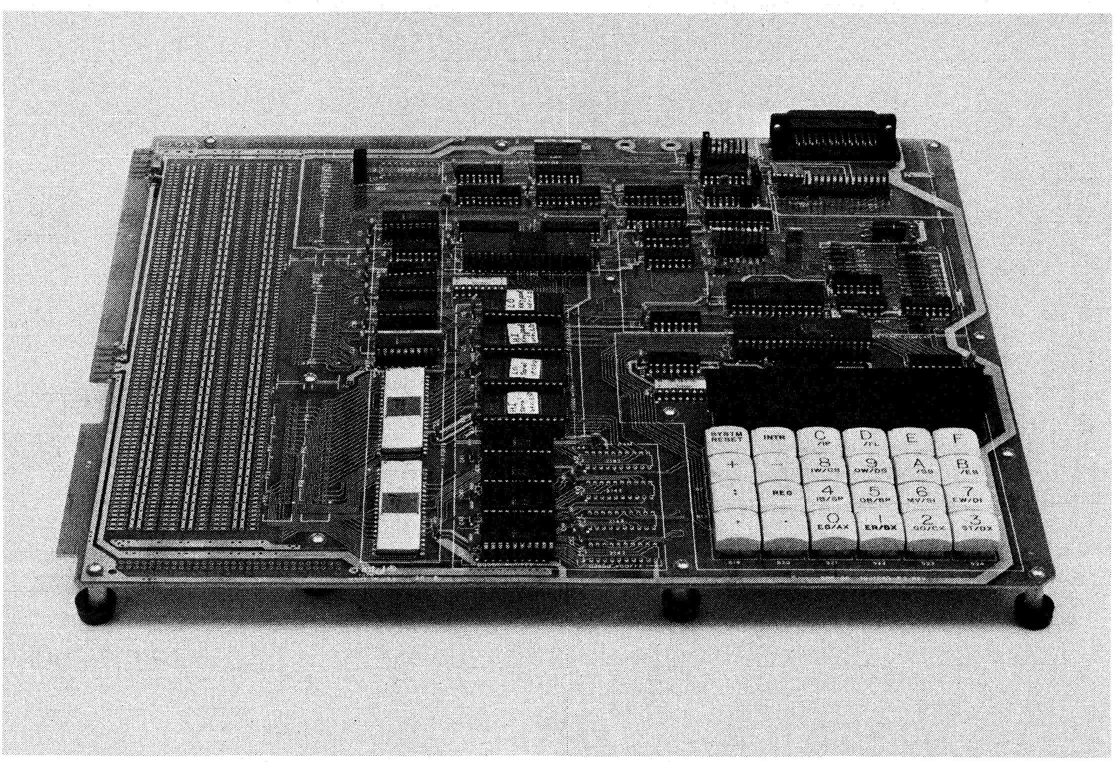
SDK-86
MCS-86™ SYSTEM DESIGN KIT

- **Complete Single Board Microcomputer System Including CPU, Memory and I/O**
- **Easy to Assemble Kit-Form**
- **High-Performance 8086 16-Bit CPU**
- **Interfaces Directly with TTY or CRT**
- **Interactive LED Display and Keyboard**
- **Wire-Wrap Area for Custom Interfaces**
- **Extensive System Monitor Software in ROM**
- **Comprehensive Design Library Included**

The MCS-86 System Design Kit (SDK-86) is a complete, single board 8086 microcomputer system in kit form. It contains all necessary components, including LED Display, Keyboard, resistors, caps, crystal and miscellaneous hardware to complete construction. Included are preprogrammed ROMs that contain the system monitor for general software utilities and system diagnostics.

The SDK-86 includes 8-digit LED display and a mnemonic 24-key keyboard for direct insertion, examination and execution of a user's program. In addition, it can be directly interfaced with a teletype terminal, CRT terminal, or the serial port of an Intellec® system.

The SDK-86 is a high-performance prototype system that has designed-in flexibility for simple interface to the user's application.



GENERAL

The SDK-86 is a complete MCS-86 microcomputer system on a single board, in kit form. It contains all necessary components to build a useful, functional system. Such items as resistors, caps, and sockets are included. Assembly time varies from 4 to 10 hours, depending on the skill of the user.

A compact but powerful system monitor is supplied with the SDK-86 to provide general software utilities and system diagnostics. If comes in preprogrammed ROMs.

The SDK-86 communicates with the outside world through either the on-board LED Display/Keyboard combination, the user's TTY or CRT terminal (Jumper Selectable); or a special mode in which an Intellec Development System can transport finished programs to and from the SDK-86. Memory can be easily expanded by simply soldering in additional devices in locations provided for this purpose. A large area of the board (22 sq. in.) is laid out as general purpose wire-wrap for the user's custom interfaces.

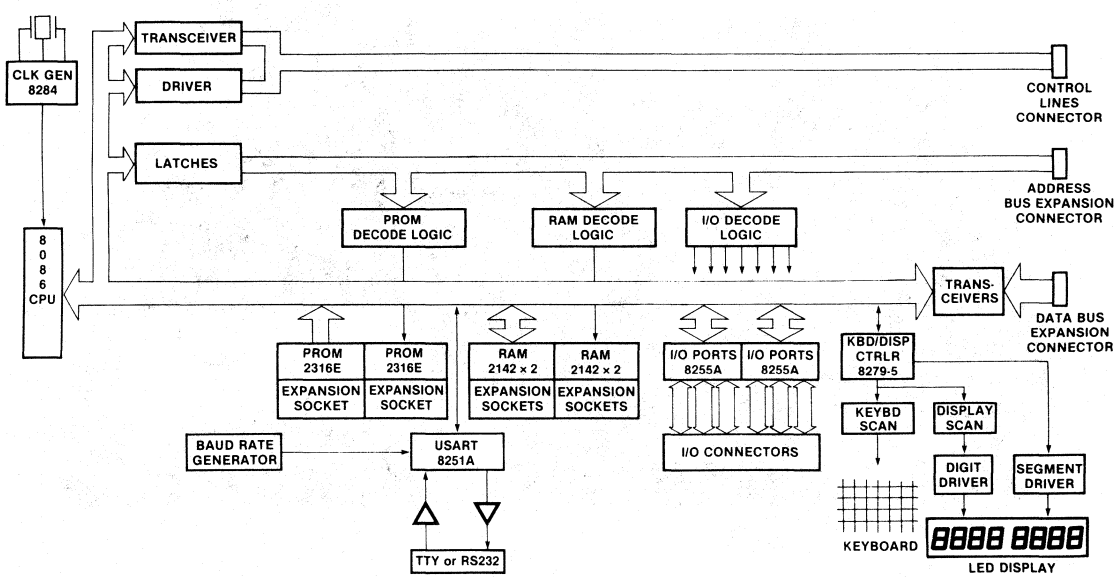
Only a few simple tools are required for assembly; soldering iron, cutters, screwdriver, etc. The SDK-86 Assembly Manual contains step-by-step instructions that make assembly easy, and minimize mistakes. Once construction is complete, the user connects his kit to a power supply and the SDK-86 is ready to go. The monitor starts immediately upon power-on or reset.

KEYBOARD MODE COMMANDS

- **Reset** — starts the monitor.
- **Execute with Breakpoint (GO)** — Allows you to execute a user program and cause it to halt at a predetermined program step — useful for debugging.
- **Single Step (ST)** — allows you to execute a user program one instruction at a time — useful for debugging.
- **Substitute Memory (EB, EW)** — allows you to examine and modify memory locations in byte or word mode.
- **Examine Register (ER)** — allows you to examine and modify the 8086's register contents.
- **Block Move (MV)** — allows you to relocate program and data portions in memory.
- **Input or Output (IB, IW, OB, OW)** — allows direct control of the SDK-86's I/O facilities in byte or word mode.

SERIAL PORT MODE COMMANDS

- All of the above keyboard mode commands via TTY or CRT terminal.
- **Dump Memory** — allows you to print or display large blocks of memory information larger than the amount
- **Start/Continue Display** — allows you to display blocks of memory information larger than the amount visible on the terminal's CRT display.
- **Punch/Read Paper Tape** — allows you to transmit finished programs into and out of the SDK-86 via the TTY paper tape punch.

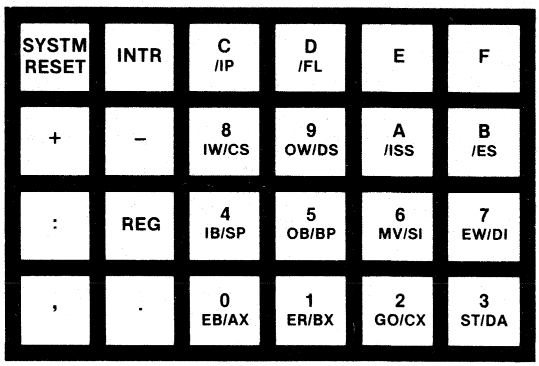


SDK-86 FUNCTIONAL BLOCK DIAGRAM

INTELLEC SLAVE MODE COMMANDS

- All of the above Keyboard Mode and Serial Port Mode commands via the console of an Intellec Development System.
- **Up/Download** — allows you to transport finished programs between the Intellec and the SDK-86, using a special utility program in the Intellec.

In addition to detailed information on using the monitors, the SDK-86 User's Manual provides circuit diagrams, a monitor listing, and a description of how the system works.

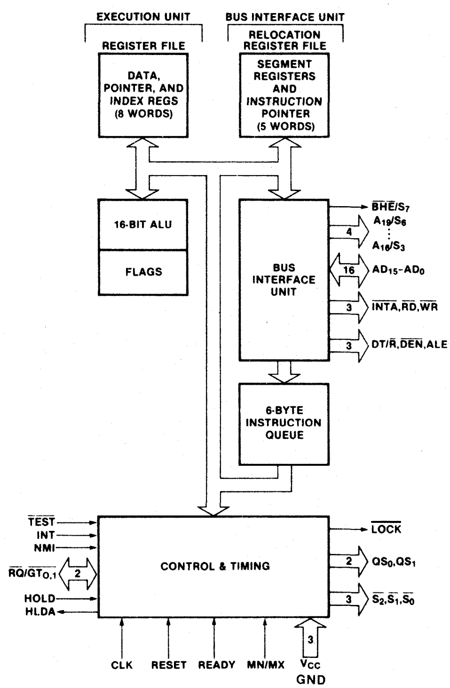


SDK-86 KEYBOARD

THE 8086

The Intel® 8086 is a new generation, high performance microprocessor implemented in N-channel, depletion load, silicon gate technology (HMOS), and packaged in a 40-pin CerDIP package. The processor has attributes of both 8- and 16-bit microprocessors. It addresses memory as a sequence of 8-bit bytes, but has a 16-bit wide physical path to memory for high performance.

- **Direct Addressing Capability to 1 MByte of Memory**
- **Assembly Language Compatible with 8080/8085**
- **14 Word, By 16-Bit Register Set with Symmetrical Operations**
- **24 Operand Addressing Modes**
- **Bit, Byte, Word, and Block Operations**
- **8- and 16-Bit Signed and Unsigned Arithmetic in Binary or Decimal Including Multiply and Divide**
- **5 MHz Clock Rate**
- **MULTIBUS™ Compatible System Interface**



8086 BLOCK DIAGRAM

8086 INSTRUCTION SET SUMMARY

DATA TRANSFER

MOV = Move:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory to/from register	1 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate to register	1 0 1 1 w reg	data	data if w=1	
Memory to accumulator	1 0 1 0 0 0 0 w	addr-low	addr-high	
Accumulator to memory	1 0 1 0 0 0 1 w	addr-low	addr-high	
Register/memory to segment register	1 0 0 0 1 1 1 0	mod 0 reg r/m		
Segment register to register/memory	1 0 0 0 1 1 0 0	mod 0 reg r/m		

PUSH = Push:

Register/memory	1 1 1 1 1 1 1 1	mod 1 1 0 r/m
Register	0 1 0 1 0 reg	
Segment register	0 0 0 reg 1 1 0	

POP = Pop:

Register/memory	1 0 0 0 1 1 1 1	mod 0 0 0 r/m
Register	0 1 0 1 1 reg	
Segment register	0 0 0 reg 1 1 1	

XCHG = Exchange:

Register/memory with register	1 0 0 0 0 1 1 w	mod reg r/m
Register with accumulator	1 0 0 1 0 reg	

IN/INW = Input to AL/AX from:

Fixed port	1 1 1 0 0 1 0 w	port
Variable port	1 1 1 0 1 1 0 w	

OUT/OUTW = Output from AL/AX to:

Fixed port	1 1 1 0 0 1 1 w	port
Variable port	1 1 1 0 1 1 1 w	
XLAT=Translate byte to AL	1 1 0 1 0 1 1 1	
LEA=Load EA to register	1 0 0 0 1 1 0 1	mod reg r/m
LDS=Load pointer to DS	1 1 0 0 0 1 0 1	mod reg r/m
LES=Load pointer to ES	1 1 0 0 0 1 0 0	mod reg r/m
LAHF=Load AH with flags	1 0 0 1 1 1 1 1	
SAHF=Store AH into flags	1 0 0 1 1 1 1 0	
PUSHF=Push flags	1 0 0 1 1 1 0 0	
POPF=Pop flags	1 0 0 1 1 1 0 1	

ARITHMETIC

ADD = Add:

Reg./memory with register to either	0 0 0 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 0 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 0 0 1 0 w	data	data if w=1	

ADC = Add with carry:

Reg./memory with register to either	0 0 0 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 s w	mod 0 1 0 r/m	data	data if s:w=01
Immediate to accumulator	0 0 0 1 0 1 0 w	data	data if w=1	

INC = Increment:

Register/memory	1 1 1 1 1 1 1 w	mod 0 0 0 r/m
Register	0 1 0 0 0 reg	
AAA=ASCII adjust for add	0 0 1 1 0 1 1 1	
DAA=Decimal adjust for add	0 0 1 0 0 1 1 1	

SUB = Subtract:

Reg./memory and register to either	0 0 1 0 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 1 0 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 1 0 1 1 0 w	data	data if w=1	

SBB = Subtract with borrow

Reg./memory and register to either	0 0 0 1 1 0 d w	mod reg r/m		
Immediate from register/memory	1 0 0 0 0 0 s w	mod 0 1 1 r/m	data	data if s:w=01
Immediate from accumulator	0 0 0 1 1 1 0 w	data	data if w=1	

DEC = Decrement:

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
Register/memory	1 1 1 1 1 1 1 w	mod 0 0 1 r/m		
Register	0 1 0 0 1 reg			
NEG=Change sign	1 1 1 1 0 1 1 w	mod 0 1 1 r/m		

CMP = Compare:

Register/memory and register	0 0 1 1 1 0 d w	mod reg r/m		
Immediate with register/memory	1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if s:w=01
Immediate with accumulator	0 0 1 1 1 1 0 w	data	data if w=1	
AAS=ASCII adjust for subtract	0 0 1 1 1 1 1 1			
DAS=Decimal adjust for subtract	0 0 1 0 1 1 1 1			
MUL=Multiply (unsigned)	1 1 1 1 0 1 1 w	mod 1 0 0 r/m		
IMUL=Integer multiply (signed)	1 1 1 1 0 1 1 w	mod 1 0 1 r/m		
AAM=ASCII adjust for multiply	1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0		
DIV=Divide (unsigned)	1 1 1 1 0 1 1 w	mod 1 1 0 r/m		
IDIV=Integer divide (signed)	1 1 1 1 0 1 1 w	mod 1 1 1 r/m		
AAD=ASCII adjust for divide	1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0		
CBW=Convert byte to word	1 0 0 1 1 0 0 0			
CWD=Convert word to double word	1 0 0 1 1 0 0 1			

LOGIC

NOT=Invert	1 1 1 1 0 1 1 w	mod 0 1 0 r/m		
SHL/SAL=Shift logical/arithmetic left	1 1 0 1 0 0 v w	mod 1 0 0 r/m		
SHR=Shift logical right	1 1 0 1 0 0 v w	mod 1 0 1 r/m		
SAR=Shift arithmetic right	1 1 0 1 0 0 v w	mod 1 1 1 r/m		
ROL=Rotate left	1 1 0 1 0 0 v w	mod 0 0 0 r/m		
ROR=Rotate right	1 1 0 1 0 0 v w	mod 0 0 1 r/m		
RCL=Rotate through carry flag left	1 1 0 1 0 0 v w	mod 0 1 0 r/m		
RCR=Rotate through carry right	1 1 0 1 0 0 v w	mod 0 1 1 r/m		

AND = And:

Reg./memory and register to either	0 0 1 0 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 0 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 0 0 1 0 w	data	data if w=1	

TEST = And function to flags, no result:

Register/memory and register	1 0 0 0 0 1 0 w	mod reg r/m		
Immediate data and register/memory	1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w=1
Immediate data and accumulator	1 0 1 0 1 0 0 w	data	data if w=1	

OR = Or:

Reg./memory and register to either	0 0 0 0 1 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w=1
Immediate to accumulator	0 0 0 0 1 1 0 w	data	data if w=1	

XOR = Exclusive or:

Reg./memory and register to either	0 0 1 1 0 0 d w	mod reg r/m		
Immediate to register/memory	1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w=1
Immediate to accumulator	0 0 1 1 0 1 0 w	data	data if w=1	

STRING MANIPULATION

REP=Repeat	1 1 1 1 0 0 1 z
MOVB/MOVW=Move byte/word	1 0 1 0 0 1 0 w
CMPB/CMPW=Compare byte/word	1 0 1 0 0 1 1 w
SCAB/SCAW=Scan byte/word	1 0 1 0 1 1 1 w
LODB/LODW=Load byte/wd to AL/AX	1 0 1 0 1 1 0 w
STOB/STOW=Stor byte/wd frm AL/A	1 0 1 0 1 0 1 w

CONTROL TRANSFER

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
CALL = **Call:**			
Direct within segment	1 1 1 0 1 0 0 0	disp-low	disp-high
Indirect within segment	1 1 1 1 1 1 1 1	mod 0 1 0 r/m	
Direct intersegment	1 0 0 1 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 0 1 1 r/m	
JMP = **Unconditional Jump:**			
Direct within segment	1 1 1 0 1 0 0 1	disp-low	disp-high
Direct within segment-short	1 1 1 0 1 0 1 1	disp	
Indirect within segment	1 1 1 1 1 1 1 1	mod 1 0 0 r/m	
Direct intersegment	1 1 1 0 1 0 1 0	offset-low	offset-high
		seg-low	seg-high
Indirect intersegment	1 1 1 1 1 1 1 1	mod 1 0 1 r/m	
RET = **Return from CALL:**			
Within segment	1 1 0 0 0 0 1 1		
Within seg. adding immed to SP	1 1 0 0 0 0 1 0	data-low	data-high
Intersegment	1 1 0 0 1 0 1 1		
Intersegment, adding immediate to SP	1 1 0 0 1 0 1 0	data-low	data-high
JE/JZ=Jump on equal/zero	0 1 1 1 0 1 0 0	disp	
JL/JNGE=Jump on less/not greater or equal	0 1 1 1 1 1 0 0	disp	
JLE/JNG=Jump on less or equal/not greater	0 1 1 1 1 1 1 0	disp	
JB/JNAE=Jump on below/not above or equal	0 1 1 1 0 0 1 0	disp	
JBE/JNA=Jump on below or equal/not above	0 1 1 1 0 1 1 0	disp	
JP/JPE=Jump on parity/parity even	0 1 1 1 1 0 1 0	disp	
JO=Jump on overflow	0 1 1 1 0 0 0 0	disp	
JS=Jump on sign	0 1 1 1 1 0 0 0	disp	
JNE/JNZ=Jump on not equal/not zero	0 1 1 1 0 1 0 1	disp	
JNL/JGE=Jump on not less/greater or equal	0 1 1 1 1 1 0 1	disp	
JNLE/JG=Jump on not less or equal/greater	0 1 1 1 1 1 1 1	disp	

	7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0
JNB/JAE=Jump on not below/above or equal	0 1 1 1 0 0 1 1	disp
JNBE/JA=Jump on not below or equal/above	0 1 1 1 0 1 1 1	disp
JNP/JPO=Jump on not par/par odd	0 1 1 1 1 0 1 1	disp
JNO=Jump on not overflow	0 1 1 1 0 0 0 1	disp
JNS=Jump on not sign	0 1 1 1 1 0 0 1	disp
LOOP=Loop CX times	1 1 1 0 0 0 1 0	disp
LOOPZ/LOOPE=Loop while zero/equal	1 1 1 0 0 0 0 1	disp
LOOPNZ/LOOPNE=Loop while not zero/equal	1 1 1 0 0 0 0 0	disp
JCXZ=Jump on CX zero	1 1 1 0 0 0 1 1	disp
INT = **Interrupt**		
Type specified	1 1 0 0 1 1 0 1	type
Type 3	1 1 0 0 1 1 0 0	
INTO=Interrupt on overflow	1 1 0 0 1 1 1 0	
IRET=Interrupt return	1 1 0 0 1 1 1 1	

PROCESSOR CONTROL

CLC=Clear carry	1 1 1 1 1 0 0 0	
CMC=Complement carry	1 1 1 1 0 1 0 1	
STC=Set carry	1 1 1 1 1 0 0 1	
CLD=Clear direction	1 1 1 1 1 1 0 0	
STD=Set direction	1 1 1 1 1 1 0 1	
CLI=Clear interrupt	1 1 1 1 1 0 1 0	
STI=Set interrupt	1 1 1 1 1 0 1 1	
HLT=Halt	1 1 1 1 0 1 0 0	
WAIT=Wait	1 0 0 1 1 0 1 1	
ESC=Escape (to external device)	1 1 0 1 1 x	mod x r/m
LOCK=Bus lock prefix	1 1 1 1 0 0 0 0	

Footnotes:

AL = 8-bit accumulator
AX = 16-bit accumulator
CX = Count register
DS = Data segment
ES = Extra segment
Above/below refers to unsigned value.
Greater = more positive;
Less = less positive (more negative) signed values
if d = 1 then "to"; if d = 0 then "from"
if w = 1 then word instruction; if w = 0 then byte instruction

if mod = 11 then r/m is treated as a REG field
if mod = 00 then DISP = 0*, disp-low and disp-high are absent
if mod = 01 then DISP = disp-low sign-extended to 16-bits, disp-high is absent
if mod = 10 then DISP = disp-high: disp-low

if r/m = 000 then EA = (BX) + (SI) + DISP
if r/m = 001 then EA = (BX) + (DI) + DISP
if r/m = 010 then EA = (BP) + (SI) + DISP
if r/m = 011 then EA = (BP) + (DI) + DISP
if r/m = 100 then EA = (SI) + DISP
if r/m = 101 then EA = (DI) + DISP
if r/m = 110 then EA = (BP) + DISP*
if r/m = 111 then EA = (BX) + DISP
DISP follows 2nd byte of instruction (before data if required)

*except if mod = 00 and r/m = 110 then EA = disp-high: disp-low.

if s:w = 01 then 16 bits of immediate data form the operand.
if s:w = 11 then an immediate data byte is sign extended to form the 16-bit operand.
if v = 0 then "count" = 1; if v = 0 then "count" in (CL)
x = don't care.
if v = 0 then "count" = 1; if v = 1 then "count" in (CL) register.
z is used for string primitives for comparison with ZF FLAG.

SEGMENT OVERRIDE PREFIX

001 reg 110

REG is assigned according to the following table:

16-Bit (w = 1)	**8-Bit (w = 0)**	**Segment**
000 AX	000 AL	00 ES
001 CX	001 CL	01 CS
010 DX	010 DL	10 SS
011 BX	011 BL	11 DS
100 SP	100 AH	
101 BP	101 CH	
110 SI	110 DH	
111 DI	111 BH	

Instructions which reference the flag register file as a 16-bit object use the symbol FLAGS to represent the file:

FLAGS = X:X:X:X:(OF):(DF):(IF):(TF):(SF):(ZF):X:(AF):X:(PF):X:(CF)

SDK-86 SPECIFICATIONS

Central Processor

CPU: 8086-4

May be operated at 2.5 MHz or 5 MHz, jumper selectable, for use with 8086.

Memory

ROM: 8K bytes 2316/2716

RAM: 2K bytes (expandable to 4K bytes) 2142

Addressing:

ROM: FE000–FFFFF

RAM: 0–7FF (800–FFF available with additional 2142's)

Note: The wire-wrap area of the SDK-86 PC board may be used for additional custom memory expansion.

Input/Output

Parallel: 48 lines (two 8255A's)

Serial: RS232 or current loop (8251A)

Baud Rate: selectable from 110 to 4800 baud.

Interfaces

Bus: All signals TTL compatible.

Parallel I/O: All signals TTL compatible.

Serial I/O: 20 mA current loop TTY or RS232.

Note: The user has access to all bus signals which enable him to design custom system expansions into the kit's wire-wrap area.

Interrupts (256 vectored)

Maskable
Non-maskable
TRAP

DMA

Hold Request: Jumper selectable. TTL compatible input

Software

System Monitor: Preprogrammed 2716 or 2316 ROMs

Addresses: FE000–FFFFF

Monitor I/O: Keyboard/Display or TTY or CRT (serial I/O)

Literature

Design Library (Provided with kit):

- SDK-86 User's Manual and Assembly Manual
- SDK-86 Monitor Listings
- MCS-86 User's Manual
- 8086 Assembly Language Reference Manual

Physical Characteristics

Width: 13.5 in.

Height: 12 in.

Depth: 1.75 in.

Weight: approx. 24 oz.

Electrical Characteristics (DC Power Required — Power Supply Not Included in Kit)

V_{CC} 5V ± 5%	3.5 Amps
V_{TTY} – 12V ± 10%	0.3 Amps (V_{TTY} required only if teletype is connected)

Environmental

Operating Temperature: 0–50°C

APPENDIX

Packaging Information

ORDERING INFORMATION

Semiconductor components are identified as follows:

Example:

M C 5 1 0 1 ___ L – 4

- 5 1 0 1 ___ : Four or five characters per device type
- L – 4 : Up to three character modifier for power, speed, processing, etc.

Package Type
B – Hermetic Package, Type B
C – Hermetic Package, Type C
D – Hermetic Package, Type D
M – Metal Can Package
P – Plastic Package
X – Unpackaged Device

M – Indicates Military Operating Temperature Range

Examples:

P5101L	CMOS 256 X 4 RAM, low power selection, plastic package, commercial temperature range.
C8080A2	8080A Microprocessor with 1.5 μs cycle time, hermetic package Type C, commercial temperature range.
MD3604/C	512 X 8 PROM, hermetic package Type D, military temperature range, MIL-STD-883 Level C processing.*
MC8080A/B	8080A Microprocessor, hermetic package Type C, military temperature range, MIL-STD-883 Level B processing.*

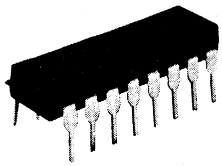
Kits, boards and systems may be ordered using the part number designations in this catalog.

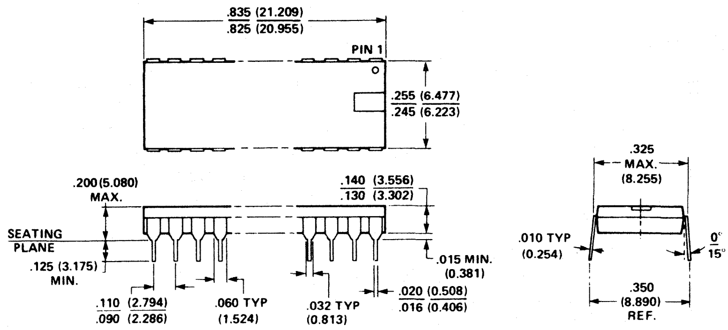
The latest Intel OEM price book should be consulted for availability of various options. These may be obtained from your local Intel representative or by writing directly to Intel Corporation, 3065 Bowers Avenue, Santa Clara, California 95051.

**On military temperature devices, B suffix indicates MIL-STD-883 Level B processing. Suffix C indicates MIL-STD-883 Level C processing. "S" number suffixes must be specified when entering any order for military temperature devices. All orders requesting source inspection will be rejected by Intel.*

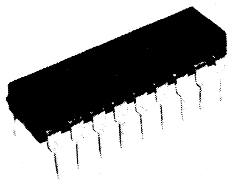
PLASTIC DUAL IN-LINE PACKAGE TYPE P

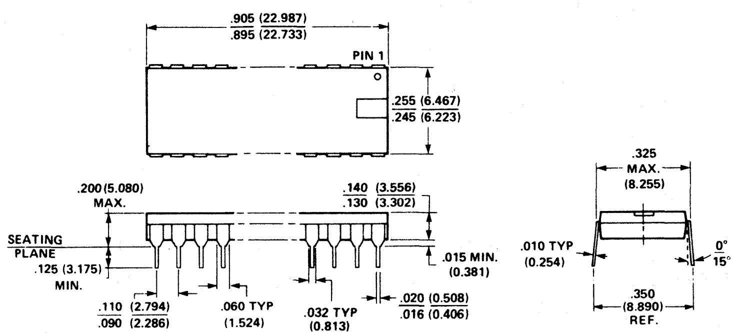
16-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



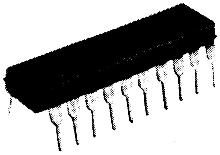


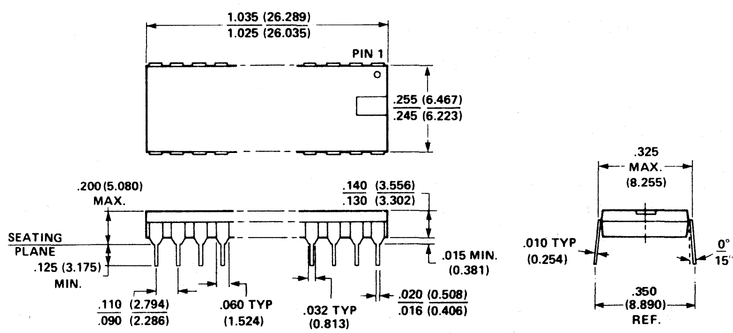
18-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



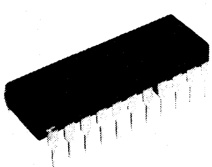


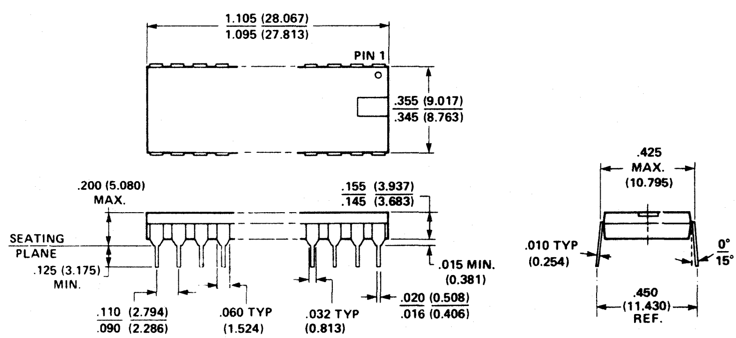
20-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P





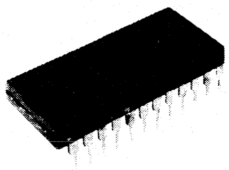
22-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P

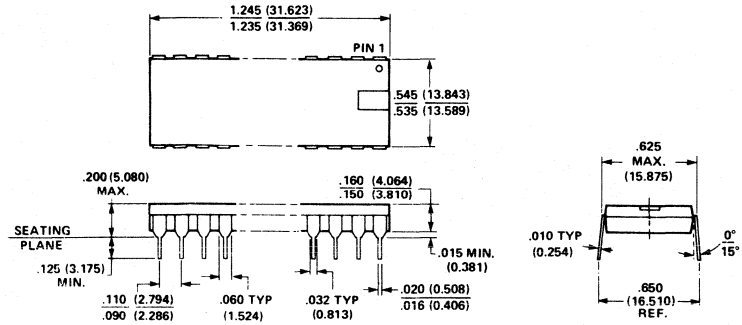




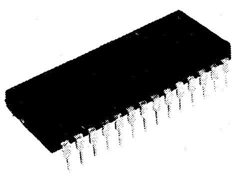
PLASTIC DUAL IN-LINE PACKAGE TYPE P

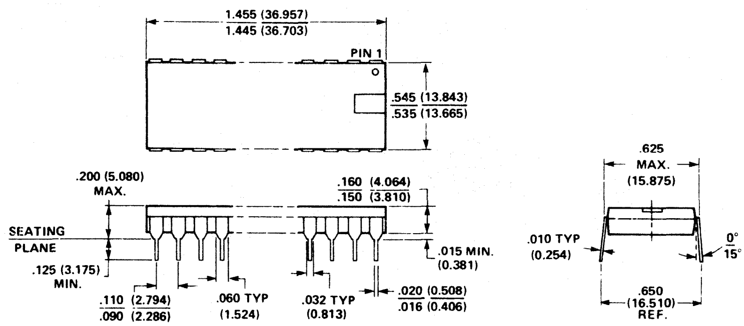
24-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



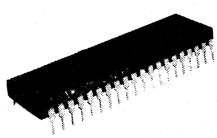


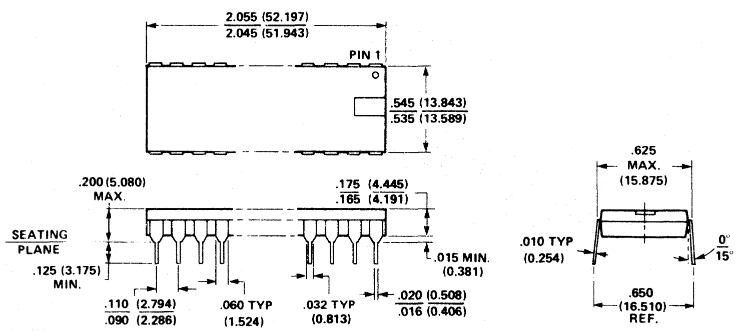
28-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P





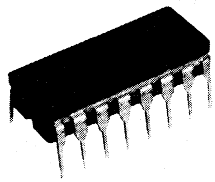
40-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P

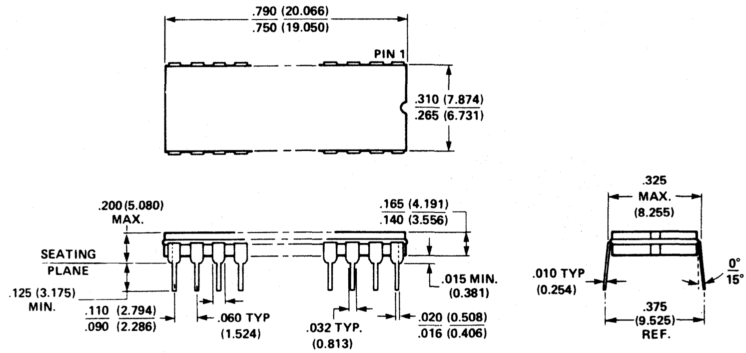




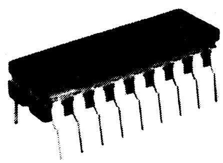
CERAMIC DUAL IN-LINE PACKAGE TYPE D

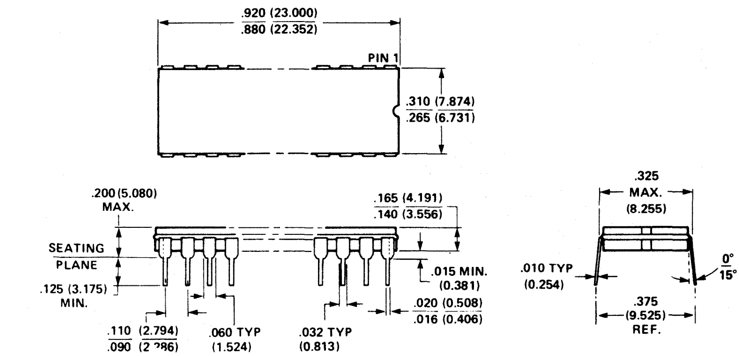
16-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



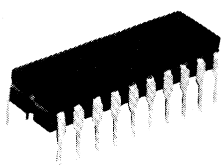


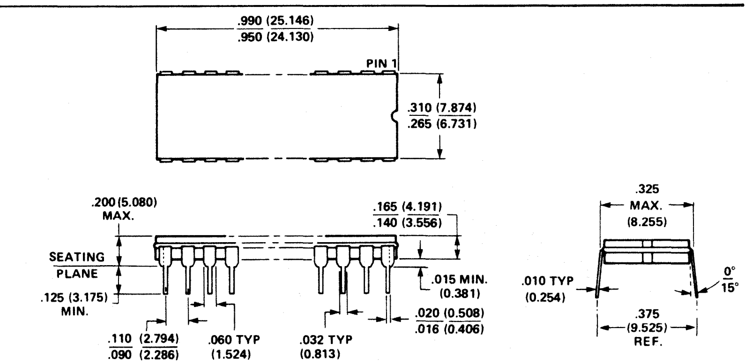
18-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



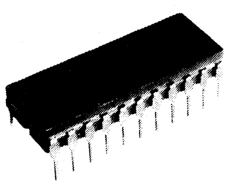


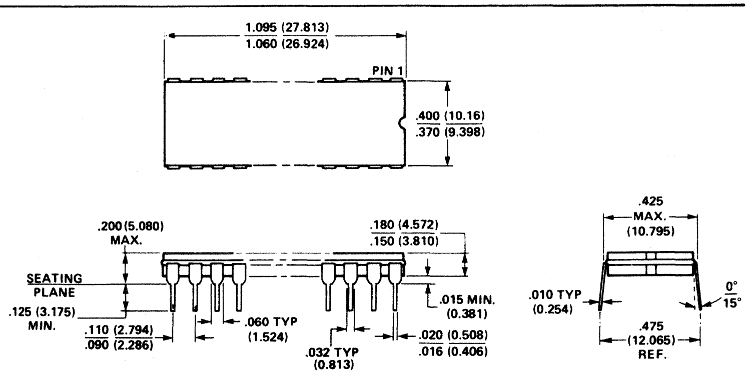
20-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D





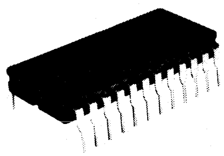
22-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D

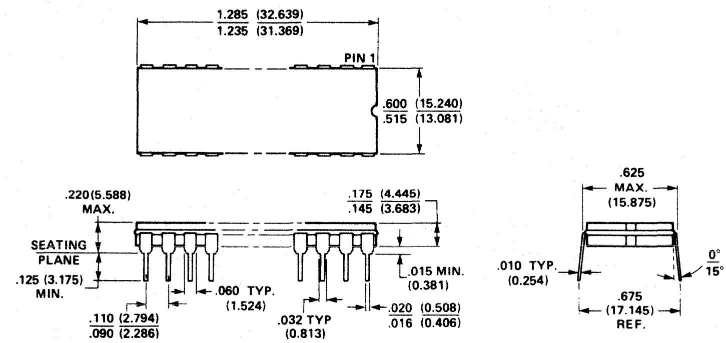




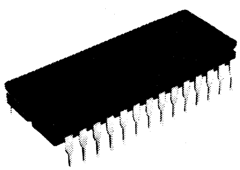
CERAMIC DUAL IN-LINE PACKAGE TYPE D

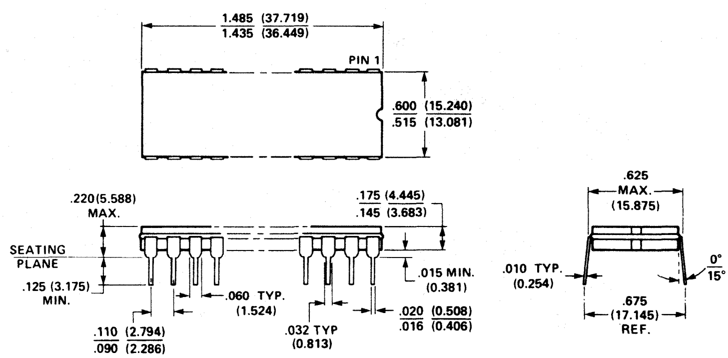
24-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D



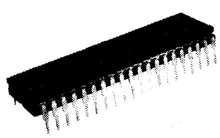


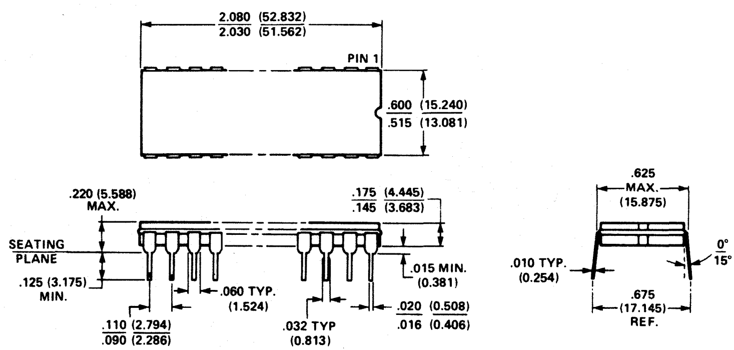
28-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D





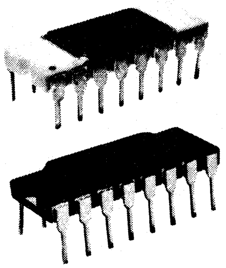
40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE D

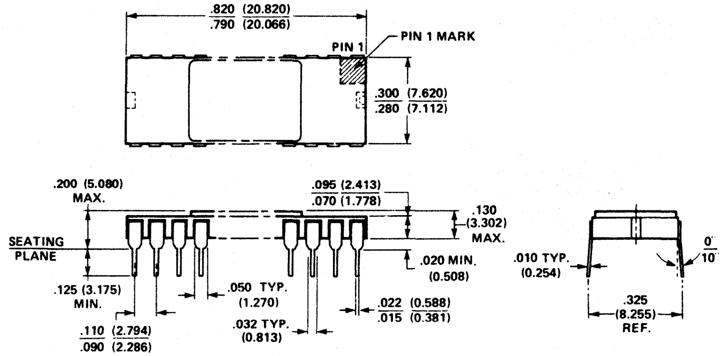




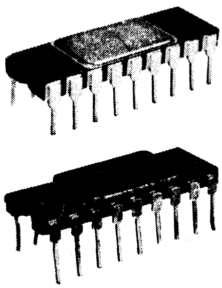
CERAMIC DUAL IN-LINE PACKAGE TYPE C

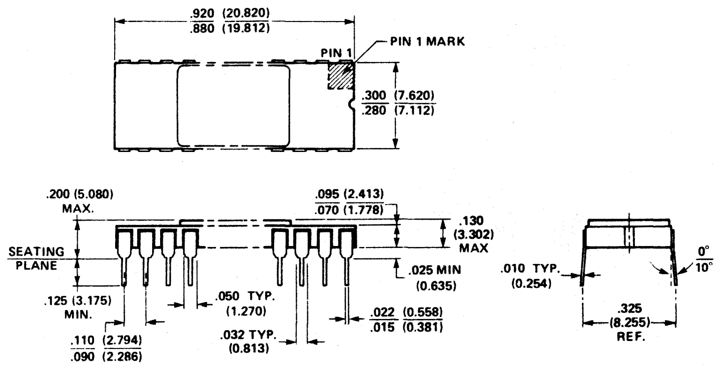
16-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C



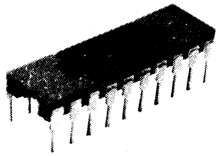


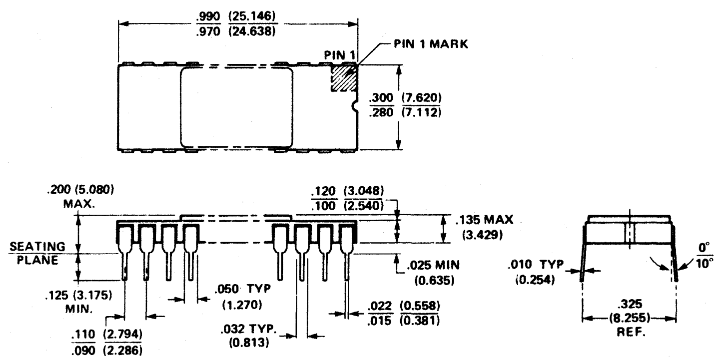
18-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C



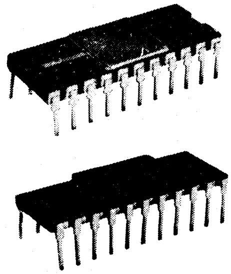


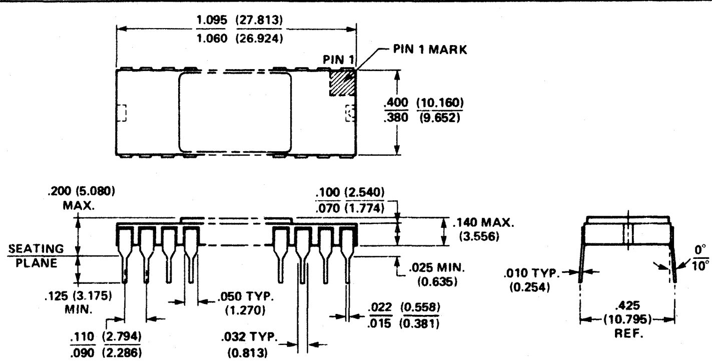
20-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C





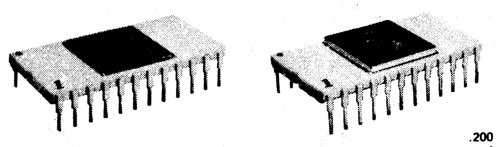
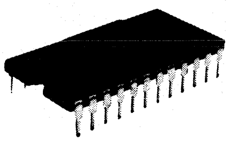
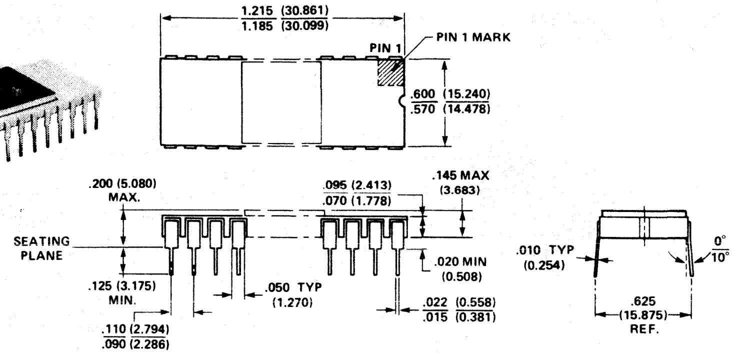
22-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C



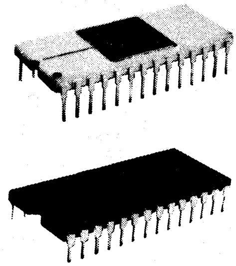
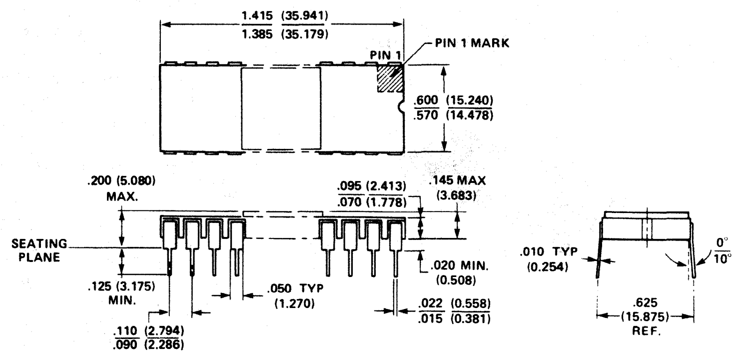


CERAMIC DUAL IN-LINE PACKAGE TYPE C

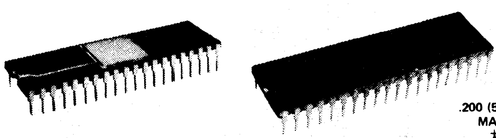
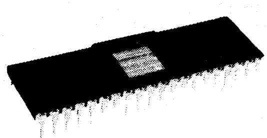
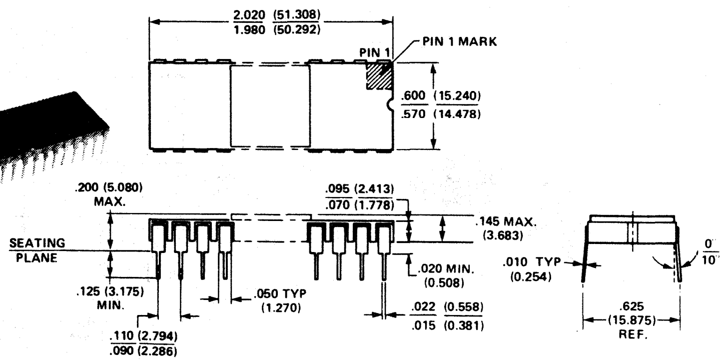
24-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C

28-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C

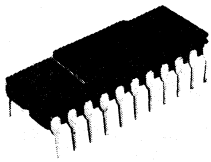



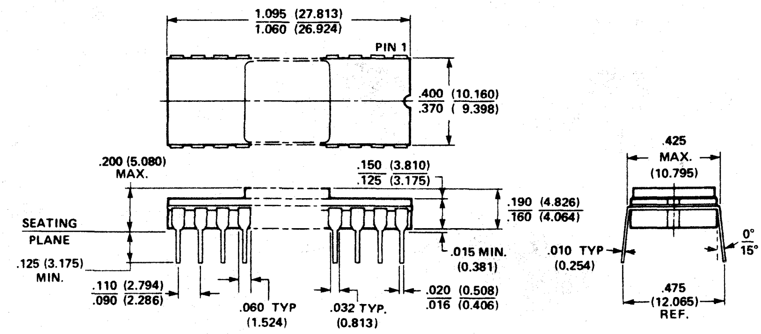
40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE C

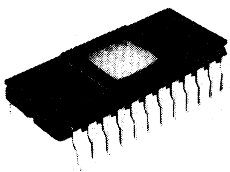
CERAMIC DUAL IN-LINE PACKAGE TYPE B

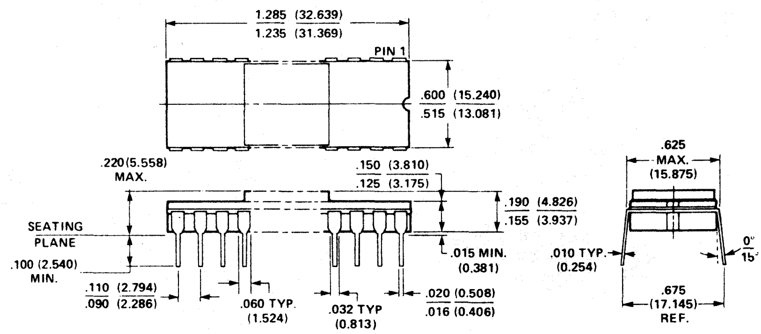
22-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B



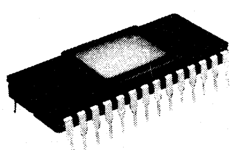


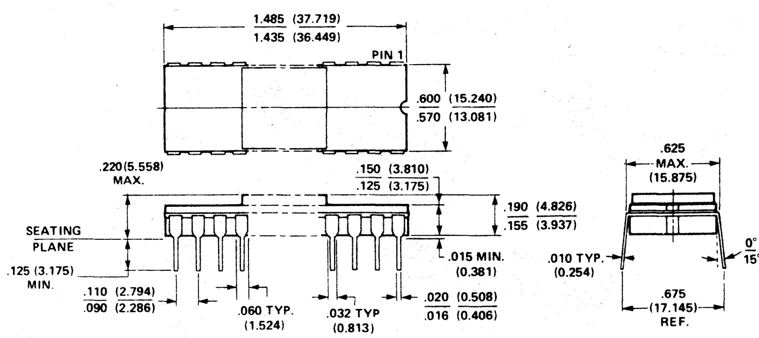
24-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B



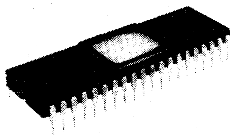


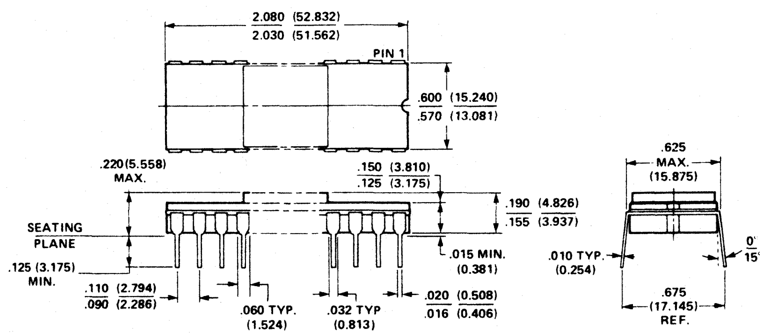
28-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B





40-LEAD HERMETIC DUAL IN-LINE PACKAGE TYPE B





NOTES

intel®

MICROCOMPUTER AND MEMORY COMPONENT SALES AND MARKETING OFFICES

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080**
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADIAN SALES OFFICES

ALABAMA
Glen White Associates
7844 Horseshoe Trail
Huntsville 35802
Tel: (205) 883-9394

ARIZONA
Intel Corp.
8650 N. 35th Avenue
Phoenix 85021
Tel: (602) 242-7205

CALIFORNIA
Intel Corp.
15335 Morrison
Suite 345
Sherman Oaks 91403
(213) 986-9510
TWX: 910-495-2045

Intel Corp.*
990 E. Arques Ave.
Suite 112
Sunnyvale 94086
Tel: (408) 738-3870
TWX: 910-339-9279
TWX: 910-338-0255

Mac-I
2576 Shattuck
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-I
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Earle Associates, Inc.
4805 Mercury Street
Suite L
San Diego 92111
Tel: (714) 278-5441
TWX: 910-335-1585

Mac-I
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-1
20121 Ventura Blvd.
Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900
TWX: 910-494-4966

Intel Corp.*
1651 East 4th Street
Suite 150
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

COLORADO
Intel Corp.
6000 East Evans Ave.
Bldg. 1, Suite 260
Denver 80222
Tel: (303) 758-8086
TWX: 910-931-2289

CONNECTICUT
Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA
Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 105
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

ILLINOIS
Intel Corp.*
900 Jorie Boulevard
Suite 220
Oakbrook 60521
Tel: (312) 325-9510
TWX: 910-651-5881

INDIANA
Electro Reps Inc.
941 E. 86th Street
Indianapolis 46240
Tel: (317) 255-4147
TWX: 810-341-3217

Electro Reps Inc.
3402 N. Anthony Blvd.
Ft. Wayne 46802
Tel: (219) 482-2388

IOWA
Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS
Technical Representatives, Inc.
8245 Nieman Road, Suite #100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

MARYLAND
Glen White Associates
57 West Timonium Road
Timonium 21093
Tel: (301) 252-6360

Intel Corp.*
57 West Timonium Road
Suite 307
Timonium 21093
Tel: (301) 252-7742
TWX: 710-232-1807

MASSACHUSETTS
Intel Corp.*
187 Billerica Road, Suite 14A
Chelmsford 01824
Tel: (617) 256-6567
TWX: 710-343-6333

MICHIGAN
Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

MINNESOTA
Intel Corp.
8200 Normandale Avenue
Suite 422
Bloomington 55437
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI
Technical Representatives, Inc.
Trade Center Bldg.
320 Brookes Drive, Suite 104
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY
Intel Corp.
1 Metroplaza Office Bldg.
505 Thornall St.
Edison 08817
Tel: (201) 985-9100
TWX: 710-480-6238

NEW YORK
Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

Intel Corp.
474 Thurston Road
Rochester, N.Y. 14619
Tel: (716) 328-7340
TWX: 510-253-3841

T-Squared
4054 Newcourt Ave.
Syracuse 13206
Tel: (315) 463-8592
TWX: 710 541-0554

T-Squared
642 Kreag Road
P.O. Box W
Pittsford 14534
Tel: (716) 248-5005
TELEX: 97-8289

NEW YORK (cont.)
Intel Corp.
85 Market Street
Poughkeepsie, New York 12601
Tel: (914) 473-2303
TWX: 510-248-0060

NORTH CAROLINA
Glen White Associates
3700 Computer Dr., Suite 330
Raleigh 27609
Tel: (919) 787-7016

OHIO
Intel Corp.*
8312 North Main Street
Dayton 45415
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
Chagrin-Brainard Bldg.
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

OREGON
ES/Chase Company
4095 SW 144th St.
Beaverton 97223
Tel: (503) 641-4111

PENNSYLVANIA
Intel Corp.*
275 Commerce Dr.
200 Office Center
Suite 212
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

TENNESSEE
Glen White Associates
Rt. #12, Norwood S/D
Jonesboro 37659
Tel: (615) 477-8850

Glen White Associates
2523 Howard Road
Germantown 38138
Tel: (901) 754-0483

Glen White Associates
6446 Ridge Lake Road
Hixon 37343
Tel: (615) 842-7799

TEXAS
Intel Corp.
6776 S.W. Freeway
Suite 550
Houston 77074
Tel: (713) 784-3400

Mycrosystems Marketing Inc.
13777 N. Central Expressway
Suite 405
Dallas 75243
Tel: (214) 238-7157
TWX: 910-867-4763

Mycrosystems Marketing Inc.
6610 Harwin Avenue, Suite 125
Houston 77036
Tel: (713) 783-2900

Intel Corp.*
2925 L.B.J. Freeway
Suite 100
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

VIRGINIA
Glen White Associates
P.O. Box 1104
Lynchburg 24505
Tel: (804) 384-6920

Glen White Associates
Rt. #1, Box 322
Colonial Beach 22443
Tel: (804) 224-4871

WASHINGTON
E.S./Chase Co.
P.O. Box 80903
Seattle 98108
Tel: (206) 762-4824
Twx: 910-444-2298

WISCONSIN
Intel Corp.
4369 S. Howell Ave.
Milwaukee 53207
Tel: (414) 747-0789

CANADA
Intel Corp.
70 Chamberlain Ave.
Ottawa, Ontario K1S 1V9
Tel: (613) 232-8576
TELEX: 053-4419

Multitek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 226-2365
TELEX: 053-4585

EUROPEAN MARKETING OFFICES

BELGIUM
Intel International*
Rue du Moulin à Papier
51-Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

FRANCE
Intel Corporation, S.A.R.L.*
5 Place de la Balance
Silic 223
94528 Rungis Cedex
Tel: (01) 687 22 21
TELEX: 270475

SCANDINAVIA
Intel Scandinavia A/S*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Denmark
Tel: (01) 18 20 00
TELEX: 19567

Intel Sweden AB*
Box 20092
S-16120 Bromma
Sweden
Tel: (08) 98 53 90
TELEX: 12261

ENGLAND
Intel Corporation (U.K.) Ltd.*
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel: (0865) 77 14 31
TELEX: 837203

Intel Corporation (U.K.) Ltd.
46-50 Beam Street
Nantwich, Cheshire CW5 5LJ
Tel: (0270) 62 65 60
TELEX: 36620

GERMANY
Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel: (089) 55 81 41
TELEX: 523 177

Intel Semiconductor GmbH
Abraham Lincoln Strasse 30
6200 Wiesbaden 1
Tel: (06121) 74855
TELEX: 04186183

Intel Semiconductor GmbH
Ernsthaldenstrasse 17
D-7000 Stuttgart 80
Tel: (0711) 7351506
TELEX: 7255346

Intel Vertriebsburo
Hindenburger Strasse 28/29
3000 Hannover
Tel: (0511) 852051
TELEX: 0923625

ORIENT MARKETING OFFICES

JAPAN
Intel Japan Corporation*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel: (03) 426-9261
TELEX: 781-28426

TAIWAN
Taiwan Automation Co.*
2nd Floor, 224
Nanking East Road
Section 3
Taipei
Tel: (02) 7710940-3
TELEX: 11942 TAIAUTO

HONG KONG
China Electronics
Sea Bird House, 9th Floor
22-28 Wyndham Street
Hong Kong

SINGAPORE
General Engineers Associates
37, Hill Street
Singapore 6

KOREA
Koram Digital
Sam Yung Bldg. #303
71-2 Bukchang - Dong Chung-Ku
Seoul 100

Leewood International, Inc.
C.P.O. Box 4046
112-25, Sokong-Dong
Chung-Ku, Seoul

INTERNATIONAL DISTRIBUTORS

ARGENTINA
S.I.E.S.A.
Av. Pte. Rogue Saenz Pena 1142 9B
1035 Buenos Aires
Tel: 35-6784

AUSTRALIA
A.J.F. Systems & Components
PTY. LTD.
44 Prospect Rd.
Prospect 5082
South Australia 17005
Tel: 269-1244
TELEX: 82635

A.J.F. Systems & Components
PTY. LTD.
29 Devlin St.
Ryde, N.S.W. 2112
Tel: (02) 807-6878
TELEX: 24906

A.J.F. Systems & Components
PTY. LTD.
310 Queen St.
Melbourne, Victoria 3000
Tel: (03) 679-702
TELEX: 30270

Warburton-Franki (Sydney) Pty. Ltd.
199 Parramatta Road
Auburn, N.S.W. 2114
Tel: 648-1711, 648-1381
TELEX: WARFRAN AA 22265

Warburton-Franki Industries
(Melbourne) Pty. Ltd.
220 Park Street
South Melbourne, Victoria 3205
Tel: 699-4999
TELEX: WARFRAN AA 31370

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 Brussels
Tel: (02) 660 00 12
TELEX: 25441

DENMARK
Lyngso Komponent A/S
Ostmarken 4
2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor
Supply A/S
Nannasgade 18
DK-2200 Copenhagen N
Tel: (01) 83 50 90
TELEX: 19037

FINLAND
Oy Fintronic AB
Loennrotinkatu 35D
SF 00180
Helsinki 18
Tel: (90) 664 451
TELEX: 12426

FRANCE
Celdis
53, Rue Charles Frerot
94250 Gentilly
Tel: 581 00 20—581 04 69
TELEX: 200 485 F

Metrologie
La Tour d'Asnieres
4, avenue Laurent Cely
92606 Asnieres
Tel: 791 44 44
TELEX: 611 448 F

Tekelec Airtronic*
Cite des Bruyeres
Rue Carle Vernet
92310 Sevres
Tel: (1) 027 75 35
TELEX: 250997

GERMANY
Alfred Neye Enatachnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel: (04106) 6121
TELEX: 02-13590

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Muenchen 80
Tel: (089) 434061
TELEX: 522561

Jermyn GmbH
Postfach 1146
D-6277 Kamberg
Tel: (06434) 6005
TELEX: 484426

INDIA
Electronics International
128 Mahatma Gandhi Road
Secunderabad
Tel: 73720
TELEX: 015-363
CABLE: GUNTICO

ISRAEL
Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel-Aviv
Tel: 475151
TELEX: 33638

ITALY
Eledra 3S S.P.A.*
Viale Elvezia, 18
20154 Milan,
Tel: (02) 3493041
TELEX: 39332

Eledra 3S S.P.A.*
Via Paolo Gaidano, 141 D
10137 Torino
TEL: (011) 30 97 097 - 30 97 114

Eledra 3S S.P.A.*
Via Giuseppe Valmarana, 63
00139 Rome, Italy
Tel: (06) 81 27 290 - 81 27 324
TELEX: 63051

JAPAN
Pan Electron
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8811
TELEX: 781-4773

Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chiyoda-ku, Tokyo 102
Tel: (03) 230-0041

NETHERLANDS
Inelco Nederland
AFD Elektronic
Joan Muyskenweg 22
NL-1006 Amsterdam
Tel: (020) 934824
TELEX: 14622

NEW ZEALAND
W. K. McLean Ltd.
103-5 Felton Matthew Avenue
Glenn Innes, Auckland
Tel: 587-037
TELEX: NZ2763

NORWAY
Nordisk Elektronik (Norge) A/S
Mustads Vei 1
N-Oslo 2
Tel: (02) 55 38 93
TELEX: 16963

PORTUGAL
Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: 119 45 313

SOUTH AFRICA
Electronic Building Elements
P.O. Box 4609
Pretoria
Tel: 78 92 21
TELEX: 30181

SPAIN
Interface*
Ronda San Pedro 22
Barcelona 10
Tel: 301 78 51
TELEX: 51508 IFCE E

SWEDEN
Nordisk Electronik AB
Fack
S-10380 Stockholm 7
Tel: (08) 248340
TELEX: 10547

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

UNITED KINGDOM
Rapid Recall, Ltd.
11-15 Betterton Street
Drury Lane
London WC2H 9BS
Tel: (01) 379-6741
TELEX: 28752

G.E.C. Semiconductors Ltd.
East Lane
Wembley HA9 7PP
Middlesex
Tel: (01) 904-9303
TELEX: 923429

Jermyn Industries
Vestry Estate
Sevenoaks, Kent.
Tel: (0732) 50144
TELEX: 95142

Sintrom Electronics Ltd.*
Arkwright Road
Reading, Berkshire
RG2 0LS
Tel: (0734) 85464

**Note New Telephone Number

***Field Application Location**

MICROCOMPUTER AND MEMORY SYSTEM SALES AND MARKETING OFFICES

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080**
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADA SALES OFFICES

ALABAMA
Pen-Tech Associates, Inc.
Holiday Office Center
3322 S. Memorial Pkwy.
Huntsville 35801
Tel: (205) 533-0090

Glen White Associates
7844 Horseshoe Trail
Huntsville 35802
Tel: (205) 883-9394

ARIZONA
BFA
4426 North Saddle Bag Trail
Scottsdale 85251
Tel: (602) 994-5400

Intel Corp.
8650 N. 35th Avenue
Phoenix 85021
Tel: (602) 242-7205

CALIFORNIA
Intel Corp.
15335 Morrison
Suite 345
Sherman Oaks 91403
(213) 986-9510
TWX: 910-495-2045

Intel Corp.*
990 E. Arques Ave.
Suite 112
Sunnyvale 94086
Tel: (408) 738-3870
TWX: 910-339-9279
TWX: 910-338-0255

Mac-I
2576 Shattuck
Suite 4B
Berkeley 94704
Tel: (415) 843-7625

Mac-I
P.O. Box 1420
Cupertino 95014
Tel: (408) 257-9880

Intel Corp.*
1651 East 4th Street
Suite 150
Santa Ana 92701
Tel: (714) 835-9642
TWX: 910-595-1114

Mac-I
P.O. Box 8763
Fountain Valley 92708
Tel: (714) 839-3341

Mac-I
20121 Ventura Blvd., Suite 240E
Woodland Hills 91364
Tel: (213) 347-5900

Earle Associates, Inc.
4805 Mercury Street
Suite L
San Diego 92111
Tel: (714) 278-5441

COLORADO
Intel Corp.
6000 East Evans Ave.
Bldg. 1, Suite 260
Denver 80222
Tel: (303) 758-8086
TWX: 910-931-2289

Mountaintek
25921 Fern Gulch Road
P.O. Box 1355
Evergreen 80439
Tel: (303) 674-5255

CONNECTICUT
Computer Marketing Association
P.O. Box 172
Fair Haven Station
New Haven 06513
Tel: (203) 789-1013

Intel Corp.
Peacock Alley
1 Padanaram Road, Suite 146
Danbury 06810
Tel: (203) 792-8366
TWX: 710-456-1199

FLORIDA
Pen-Tech Associates, Inc.
201 S.E. 15th Terrace
Deerfield Beach 33441
Tel: (305) 421-4989

Pen-Tech Associates, Inc.
111 So. Maitland Ave.
Maitland 32751
Tel: (305) 645-3444

Intel Corp.
1001 N.W. 62nd Street, Suite 406
Ft. Lauderdale 33309
Tel: (305) 771-0600
TWX: 510-956-9407

Intel Corp.
5151 Adanson Street, Suite 105
Orlando 32804
Tel: (305) 628-2393
TWX: 810-853-9219

GEORGIA
Pen-Tech Associates, Inc.
Suite 305 C
2101 Powers Ferry Roads
Atlanta 30339
Tel: (404) 955-0293

ILLINOIS
Intel Corp.*
900 Jorie Boulevard
Suite 220
Oakbrook 60521
Tel: (312) 325-9510
TWX: 910-651-5881

Dytek-Central, Inc.
121 So. Wilke Road
Suite 103
Arlington Heights 60005
Tel: (312) 394-3380
TWX: 910-687-2267

INDIANA
Electro Reps Inc.
941 E. 86th Street
Indianapolis 46240
Tel: (317) 255-4147
TWX: 810-341-3217

Electro Reps Inc.
3402 N. Anthony Blvd.
Ft. Wayne 46802
Tel: (219) 482-2388

IOWA
Technical Representatives, Inc.
St. Andrews Building
1930 St. Andrews Drive N.E.
Cedar Rapids 52402
Tel: (319) 393-5510

KANSAS
Technical Representatives, Inc.
8245 Nieman Road, Suite #100
Lenexa 66214
Tel: (913) 888-0212, 3, & 4
TWX: 910-749-6412

MARYLAND
Glen White Associates
57 West Timonium Road
Timonium 21093
Tel: (301) 252-7742

Intel Corp.*
57 West Timonium Road
Suite 307
Timonium 21093
Tel: (301) 252-7742
TWX: 710-232-1807

Mesa Inc.
11900 Parklawn Drive
Rockville 20852
Tel: Wash. (301) 881-8430
Balto. (301) 792-0021

MASSACHUSETTS
Intel Corp.*
187 Billerica Road, Suite 14A
Chelmsford 01824
Tel: (617) 256-6567
TWX: 710-343-6333

Computer Marketing Associates
257 Crescent Street
Waltham 02154
Tel: (617) 894-7000

MICHIGAN
Intel Corp.
26500 Northwestern Hwy.
Suite 401
Southfield 48075
Tel: (313) 353-0920
TWX: 910-420-1212
TELEX: 2 31143

Lowry & Associates, Inc.
135 W. North Street
Suite 4
Brighton 48116
Tel: (313) 227-7067

MINNESOTA
Intel Corp.
8200 Normandale Avenue
Suite 422
Bloomington 55437
Tel: (612) 835-6722
TWX: 910-576-2867

MISSOURI
Technical Representatives, Inc.
Trade Center Bldg.
320 Brookes Drive, Suite 104
Hazelwood 63042
Tel: (314) 731-5200
TWX: 910-762-0618

NEW JERSEY
Intel Corp.
1 Metroplaza Office Bldg.
505 Thornall St.
Edison 08817
Tel: (201) 985-9100
TWX: 710-480-6238

NEW MEXICO
BFA Corporation
312 West Parker Road
Las Cruces 88001
Tel: (505) 523-0601
TWX: 910-983-0543

BFA Corporation
3705 Westerfield, N.E.
Albuquerque 87111
Tel: (505) 292-1212
TWX: 910-989-1157

NEW YORK
Intel Corp.*
350 Vanderbilt Motor Pkwy.
Suite 402
Hauppauge 11787
Tel: (516) 231-3300
TWX: 510-227-6236

NEW YORK (cont.)
Intel Corp.
474 Thurston Road
Rochester 14619
Tel: (716) 328-7340
TWX: 510-253-3841

T-Squared
4054 Newcourt Avenue
Syracuse 13206
Tel: (315) 463-8592
TWX: 710-541-0554

T-Squared
642 Kreag Road
P.O. Box W
Pittsford 14534
Tel: (716) 248-5005
TELEX: 97-8289

Intel Corp.
85 Market Street
Poughkeepsie 12601
Tel: (914) 473-2303
TWX: 510-248-0060

Measurement Technology, Inc.
159 Northern Boulevard
Great Neck 11021
Tel: (516) 482-3500

NORTH CAROLINA
Pen-Tech Associates, Inc.
P.O. Box 5382
Highpoint 27262
Tel: (919) 883-9125

Glen White Associates
3700 Computer Drive
Suite 330
Raleigh 27609
Tel: (919) 787-7016

OHIO
Intel Corp.*
8312 North Main Street
Dayton 45415
Tel: (513) 890-5350
TWX: 810-450-2528

Intel Corp.*
Chagrin-Brainard Bldg.
28001 Chagrin Blvd.
Cleveland 44122
Tel: (216) 464-2736

Lowry & Associates, Inc.
42 East Rahn Road
Suite 100
Dayton 45429
Tel: (513) 435-4795

Lowry & Associates, Inc.
24200 Chagrin Blvd.
Suite 146
Cleveland 44122
Tel: (216) 464-8113

OREGON
ES/Chase Company
4095 SW 144th St.
Beaverton 97223
Tel: (503) 641-4111

PENNSYLVANIA
Intel Corp.*
275 Commerce Dr.
200 Office Center
Suite 212
Fort Washington 19034
Tel: (215) 542-9444
TWX: 510-661-2077

Q.E.D. Electronics
300 N. York Road
Hatboro 19040
Tel: (215) 674-9600

PENNSYLVANIA (cont.)
Lowry & Associates, Inc.
Three Parkway Center
Suite 201
Pittsburgh 15220
Tel: (412) 922-5110

TENNESSEE
Glen White Associates
Rt. #12, Norwood S/D
Jonesboro 37659
Tel: (615) 477-8850

Glen White Associates
2523 Howard Road
Germantown 38138
Tel: (901) 754-0483

Glen White Associates
6446 Ridge Lake Road
Hixon 37343
Tel: (615) 842-7799

TEXAS
Intel Corp.
6776 S.W. Freeway
Suite 550
Houston 77074
Tel: (713) 784-3400

Mycrosystems Marketing Inc.
13777 N. Central Expressway
Suite 405
Dallas 75243
Tel: (214) 238-7157
TWX: 910-867-4763

Mycrosystems Marketing Inc.
6610 Harwin Avenue, Suite 125
Houston 77036
Tel: (713) 783-2900

Intel Corp.*
2925 L.B.J. Freeway
Suite 100
Dallas 75234
Tel: (214) 241-9521
TWX: 910-860-5487

UTAH
Mountaintek
3788 Brockbank Drive
Salt Lake City 84117
Tel: (801) 266-9617

VIRGINIA
Glen White Associates
P.O. Box 1104
Lynchburg 24505
Tel: (804) 384-6920

Glen White Associates
Rt. #1, Box 322
Colonial Beach 22443
Tel: (804) 224-4871

WASHINGTON
E.S./Chase Co.
P.O. Box 80903
Seattle 98108
Tel: (206) 762-4824
TWX: 910-444-2298

WISCONSIN
Intel Corp.
4369 S. Howell Ave.
Milwaukee 53207
Tel: (414) 747-0789

CANADA
Intel Corp.
70 Chamberlain Ave.
Ottawa, Ontario K1S 1V9
Tel: (613) 232-8576
TELEX: 053-4419

Multilek, Inc.*
15 Grenfell Crescent
Ottawa, Ontario K2G 0G3
Tel: (613) 226-2365
TELEX: 053-4585

EUROPEAN MARKETING OFFICES

BELGIUM
Intel International*
Rue du Moulin à Papier
51-Boite 1
B-1160 Brussels
Tel: (02) 660 30 10
TELEX: 24814

FRANCE
Intel Corporation, S.A.R.L.*
5 Place de la Balance
Silic 223
94528 Rungis Cedex
Tel: (01) 687 22 21
TELEX: 270475

SCANDINAVIA
Intel Scandinavia A/S*
Lyngbyvej 32 2nd Floor
DK-2100 Copenhagen East
Denmark
Tel: (01) 18 20 00
TELEX: 19567

Intel Sweden AB*
Box 20092
S-16120 Bromma
Sweden
Tel: (08) 98 53 90
TELEX: 12261

ENGLAND
Intel Corporation (U.K.) Ltd.*
Broadfield House
4 Between Towns Road
Cowley, Oxford OX4 3NB
Tel: (0865) 77 14 31
TELEX: 837203

Intel Corporation (U.K.) Ltd.
46-50 Beam Street
Nantwich, Cheshire CW5 5LJ
Tel: (0270) 62 65 60
TELEX: 36620

GERMANY
Intel Semiconductor GmbH*
Seidlstrasse 27
8000 Muenchen 2
Tel: (089) 55 81 41
TELEX: 523 177

Intel Semiconductor GmbH
Abraham Lincoln Strasse 30
6200 Wiesbaden 1
Tel: (06121) 74855
TELEX: 04186183

Intel Semiconductor GmbH
Ernsthaldenstrasse 17
D-7000 Stuttgart 80
Tel: (0711) 7351506
TELEX: 7255346

Intel Vertriebsburo
Hindenburger Strasse 28/29
3000 Hannover
Tel: (0511) 852051
TELEX: 0923625

ORIENT MARKETING OFFICES

JAPAN
Intel Japan Corporation*
Flower Hill-Shinmachi East Bldg.
1-23-9, Shinmachi, Setagaya-ku
Tokyo 154
Tel: (03) 426-9261
TELEX: 781-28426

TAIWAN
Taiwan Automation Co.*
2nd Floor, 224
Nanking East Road
Section 3
Taipei
Tel: (02) 7710940-3
TELEX: 11942 TAIAUTO

HONG KONG
China Electronics
Sea Bird House, 9th Floor
22-28 Wyndham Street
Hong Kong

SINGAPORE
General Engineering Associates
37, Hill Street
Singapore 6

KOREA
Koram Digital
Sam Yung Bldg. #303
71-2 Bukchang - Dong Chung-Ku
Seoul 100

Leewood International, Inc.
C.P.O. Box 4046
112-25, Sokong-Dong
Chung-Ku, Seoul

INTERNATIONAL DISTRIBUTORS

ARGENTINA
S.I.E.S.A.
Av. Pte. Rogue Saenz Pena 1142 9B
1035 Buenos Aires
Tel: 35-6784

AUSTRALIA
A.J.F. Systems & Components PTY. LTD.
44 Prospect Rd.
Prospect 5082
South Australia 17005
Tel: 269-1244
TELEX: 82635

A.J.F. Systems & Components PTY. LTD.
29 Devlin St.
Ryde, N.S.W. 2112
Tel: (02) 807-6878
TELEG: 24906

A.J.F. Systems & Components PTY. LTD.
310 Queen St. Melbourne,
Victoria 3000
Tel: (03) 679-702
TELEX: 30270

Warburton-Franki (Sydney) Pty. Ltd.
199 Parramatta Road
Auburn, N.S.W. 2114
Tel: 648-1711, 648-1381
TELEX: WARFRAN AA 22265

Warburton-Franki Industries (Melbourne) Pty. Ltd.
220 Park Street
South Melbourne, Victoria 3205
Tel: 699-4999
TELEX: WARFRAN AA 31370

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 Brussels
Tel: (02) 660 00 12
TELEX: 25441

DENMARK
Lyngso Komponent A/S
Ostmarken 4
2860 Soborg
Tel: (01) 67 00 77
TELEX: 22990

Scandinavian Semiconductor Supply A/S
Nannasgade 18
DK-2200 Copenhagen N
Tel: (01) 83 50 90
TELEX: 19037

FINLAND
Oy Fintronic AB
Loennrotinkatu 35D
SF 00180
Helsinki 18
Tel: (90) 664 451
TELEX: 12426

FRANCE
Celdis
53, Rue Charles Frerot
94250 Gentilly
Tel: 581 00 20-591 04 69
TELEX: 200 485 F

Metrologie
La Tour d'Asnières
4, Avenue Laurent Cely
92606-Asnieres
Tel: 791 44 44
TELEX: 611 448 F

Tekelec Airtronic*
Cite des Bruyeres
Rue Carle Vernet
92310 Sevres
Tel: (1) 027 75 35
TELEX: 250997

GERMANY
Alfred Neye Enatechnik GmbH
Schillerstrasse 14
D-2085 Quickborn-Hamburg
Tel: (04106) 6121
TELEX: 02-13590

Electronic 2000 Vertriebs GmbH
Neumarkter Strasse 75
D-8000 Muenchen 80
Tel: (089) 434061
TELEX: 522561

GERMANY (cont.)
Jermyn GmbH
Postfach 1146
D-6277 Kamberg
Tel: (06434) 6005
TELEX: 484426

INDIA
Electronics International
128 Mahatma Gandhi Road
Secunderabad
Tel: 73720
TELEX: 015-363
CABLE: GUNTICO

ISRAEL
Eastronics Ltd.*
11 Rozanis Street
P.O. Box 39300
Tel-Aviv
Tel: 475151
TELEX: 33638

ITALY
Eledra 3S S.P.A.*
Viale Elvezia, 18
20154 Milan,
Tel: (02) 3493041
TELEX: 39332

Eledra 3S S.P.A.*
Via Paolo Gaidano, 141 D
10137 Torino
TEL: (011) 30 97 097 - 30 97 114

Eledra 3S S.P.A.*
Via Giuseppe Valmarana, 63
00139 Rome, Italy
Tel: (06) 81 27 290 - 81 27 324
TELEX: 63051

JAPAN
Pan Electron
No. 1 Higashikata-Machi
Midori-Ku, Yokohama 226
Tel: (045) 471-8811
TELEX: 781-4773

Ryoyo Electric Corp.
Konwa Bldg.
1-12-22, Tsukiji, 1-Chome
Chuo-Ku, Tokyo 104
Tel: (03) 543-7711

JAPAN (cont.)
Nippon Micro Computer Co. Ltd.
Mutsumi Bldg. 4-5-21 Kojimachi
Chiyoda-ku, Tokyo 102
Tel: (03) 230-0041

NETHERLANDS
C.N. Rood BV
Cort Vender
Lindenstraat, 13
Postbus 42
Rijswijk 2H2100
Tel: 070-996360
TELEX: 31238

Inelco Nederland
AFD Elektronic
Joan Muyskenweg 22
NL-1006 Amsterdam
Tel: (020) 934824
TELEX: 14622

NEW ZEALAND
W. K. McLean Ltd.
103-5 Felton Matthew Avenue
Glenn Innes, Auckland
Tel: 587-037
TELEX: NZ2763

NORWAY
Nordisk Elektronik (Norge) A/S
Mustads Vei 1
N-Oslo 2
Tel: (02) 55 38 93
TELEX: 16963

PORTUGAL
Ditram
Componentes E Electronica LDA
Av. Miguel Bombarda, 133
Lisboa 1
Tel: 119 45 313

SOUTH AFRICA
Electronic Building Elements
P.O. Box 4609
Pretoria
Tel: 78 92 21
TELEX: 30181

SPAIN
Interface*
Ronda San Pedro 22
Barcelona 10
Tel: 301 78 51
TELEX: 51508 IFCE E

SWEDEN
Nordisk Electronik AB
Fack
S-10380 Stockholm 7
Tel: (08) 248340
TELEX: 10547

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

UNITED KINGDOM
Rapid Recall, Ltd.
11-15 Betterton Street
Drury Lane
London WC2H 9BS
Tel: (01) 379-6741
TELEX: 28752

G.E.C. Semiconductors Ltd.
East Lane
Wembley HA9 7PP
Middlesex
Tel: (01) 904-9303
TELEX: 923429

Jermyn Industries
Vestry Estate
Sevenoaks, Kent
Tel: (0732) 50144
TELEX: 95142

Sintrom Electronics Ltd.*
Arkwright Road
Reading, Berkshire RG20LS
Tel: (0734) 85464
TELEX: 847395

***Field Application Location**

***Note New Telephone Number*

intel®

3065 Bowers Avenue
Santa Clara, California 95051
Tel: (408) 987-8080**
TWX: 910-338-0026
TELEX: 34-6372

U.S. AND CANADIAN DISTRIBUTORS

U.S. AND CANADIAN DISTRIBUTORS

ALABAMA

†*Hamilton/Avnet Electronics
805 Oser Drive NW
Huntsville 35805
Tel: (205) 533-1170

Pioneer
1207 Putman Drive NW
Huntsville 35805
Tel: (205) 837-9300

ARIZONA

†*Hamilton/Avnet Electronics
8155 North 21st Street
Phoenix 85021
Tel: (602) 275-7851

Liberty/Arizona
8155 N. 24th Avenue
Phoenix 85021
Tel: (602) 257-1272
TELEX: 910-951-4282

CALIFORNIA

†*Avnet Electronics
350 McCormick Avenue
Costa Mesa 92626
Tel: (714) 754-6111
Tel: (213) 558-2345

†*Hamilton/Avnet Electronics
575 E. Middlefield Road
Mountain View 94040
Tel: (415) 961-8600

†*Hamilton/Avnet Electronics
8917 Complex Drive
San Diego 92123
Tel: (714) 279-2421

†*Hamilton Electro Sales
10912 W. Washington Boulevard
Culver City 90230
Tel: (213) 558-2121

†Cramer/San Francisco
720 Palomar Avenue
Sunnyvale 94086
Tel: (408) 739-3011

Cramer/Los Angeles
17201 Daimler Street
Irvine 92714
Tel: (714) 979-3000

†Liberty Electronics
124 Maryland Street
El Segundo 90245
Tel: (213) 322-8100
Tel: (714) 638-7601
TWX: 910-348-7140

†Liberty/San Diego
8284 Mercury Court
San Diego 92111
Tel: (714) 565-9171
TELEX: 910-335-1590

†Elmar Electronics
2288 Charleston Road
Mountain View 94040
Tel: (415) 961-3611
TELEX: 910-379-6437

COLORADO

†Elmar/Denver
6777 E. 50th Avenue
Commerce City 80022
Tel: (303) 287-9611
TWX: 910-936-0770

†*Hamilton/Avnet Electronics
5921 No. Broadway
Denver 80216
Tel: (303) 545-1212

CONNECTICUT

†Cramer/Connecticut
35 Dodge Avenue
North Haven 06473
Tel: (203) 239-5641

†*Hamilton/Avnet Electronics
643 Danbury Road
Georgetown 06829
Tel: (203) 762-0361

Harvey Electronics
112 Main Street
Norwalk 06851
Tel: (203) 853-1515

FLORIDA

Cramer/E.W. Hollywood
4035 No. 29th Avenue
Hollywood 33020
Tel: (305) 921-7878

†*Hamilton/Avnet Electronics
6800 Northwest 20th Ave.
Ft. Lauderdale 33309
Tel: (305) 971-2900

Cramer/EW Orlando
345 No. Graham Ave.
Orlando 32814
Tel: (305) 894-1511

Pioneer
6220 S. Orange Blossom Trail
Suite 412
Orlando 32809
Tel: (305) 859-3600

GEORGIA

†Cramer
6456 Warren Drive
Norcross 30071
Tel: (404) 448-9050

**Note New Telephone Number

GEORGIA (cont.)

†*Hamilton/Avnet Electronics
6700 I 85, Access Road, #11
Norcross 30071
Tel: (404) 448-0800

ILLINOIS

†Cramer/Chicago
1911 So. Busse Rd.
Mt. Prospect 60056
Tel: (312) 593-8230

†*Hamilton/Avnet Electronics
3901 No. 25th Ave.
Schiller Park 60176
Tel: (312) 678-6310

Pioneer/Chicago
1551 Carmen Drive
Elk Grove Village 60006
Tel: (312) 437-9680

INDIANA

†Pioneer/Indiana
6408 Castleplace Drive
Indianapolis 46250
Tel: (317) 849-7300

Sheridan Sales
8790 Purdue Road
Indianapolis 46268
Tel: (317) 297-3146

KANSAS

†*Hamilton/Avnet Electronics
9219 Quivira Road
Overland Park 66215
Tel: (913) 888-8900

MARYLAND

†Cramer/EW Washington
16021 Industrial Drive
Gaithersburg 20760
Tel: (301) 948-0110

†*Hamilton Avnet
7235 Standard Drive
Hanover 21076
Tel: (301) 796-5000

†Pioneer/Washington
9100 Gaither Road
Gaithersburg 20760
Tel: (301) 948-0710
TWX: 710-828-0545

MASSACHUSETTS

†Cramer Electronics Inc.
85 Wells Avenue
Newton 02159
Tel: (617) 969-7700

†*Hamilton/Avnet Electronics
100 E. Commerce Way
Woburn 01801
Tel: (617) 933-8000

MICHIGAN

†Sheridan Sales Co.
24543 Indoplex Circle
Farmington Hills 48024
Tel: (313) 477-3800

†Pioneer/Michigan
13485 Stamford
Livonia 48150
Tel: (313) 525-1800

†*Hamilton/Avnet Electronics
32487 Schoolcraft Road
Livonia 48150
Tel: (313) 522-4700
TWX: 810-242-8775

MINNESOTA

†Industrial Components
5280 West 74th Street
Minneapolis 55435
Tel: (612) 831-2666

†Cramer/Bonn
7275 Bush Lake Road
Edina 55435
Tel: (612) 835-7811

†*Hamilton/Avnet Electronics
7683 Washington Avenue So.
Edina 55435
Tel: (612) 941-3801

MISSOURI

†*Hamilton/Avnet Electronics
396 Brookes Lane
Hazelwood 63042
Tel: (314) 731-1144

Sheridan Sales
110 S. Hwy. 67, Suite 10
Florissant 63031
Tel: (314) 837-5200

NEW JERSEY

Cramer/Pennsylvania, Inc.
12 Springdale Road
Cherry Hill Industrial Center
Cherry Hill 08034
Tel: (609) 424-5993
TWX: 710-896-0908

†*Hamilton/Avnet Electronics
218 Little Falls Road
Cedar Grove 07009
Tel: (201) 239-0800
TWX: 710-994-5787

Cramer/New Jersey
1 Cardinal Drive
Little Falls 07424
Tel: (201) 785-4300

NEW JERSEY (cont.)

†Harvey Electronics
389 Passaic Avenue
Fairfield 07006
Tel: (201) 227-1262

†*Hamilton/Avnet Electronics
113 Gaither Drive
East Gate Industrial Park
Mt. Laurel 08057
Tel: (609) 234-2133
TWX: 710-897-1405

NEW MEXICO

†*Hamilton/Avnet Electronics
2524 Baylor Drive, S.E.
Albuquerque 87119
Tel: (505) 765-1500

NEW YORK

†Cramer/Rochester
3000 Winton Road South
Rochester 14623
Tel: (716) 275-0300

†*Hamilton/Avnet Electronics
167 Clay Road
Rochester 14623
Tel: (716) 442-7820

Cramer/Syracuse
6716 Joy Road
East Syracuse 13057
Tel: (315) 437-6671

†*Hamilton/Avnet Electronics
6500 Joy Road
E. Syracuse 13057
Tel: (315) 437-2641

†Cramer/Long Island
129 Oser Avenue
Hauppauge, L.I. 11787
Tel: (516) 231-5600
TWX: 510-227-9863

†*Hamilton/Avnet Electronics
70 State Street
Westbury, L.I. 11590
Tel: (516) 333-5800
TWX: 510-222-8237

†Harvey Electronics
60 Crossways Park West
Woodbury 11797
Tel: (516) 921-8700

NORTH CAROLINA

†Cramer Electronics
938 Burke Street
Winston-Salem 27102
Tel: (919) 725-8711

Pioneer/Carolina
2906 Baltic Avenue
Greensboro 27406
Tel: (919) 273-4441
TWX: 510-925-1114

Hamilton/Avnet Electronics, Inc.
2803 Industrial Drive
Raleigh 27609
Tel: (919) 829-8030

OHIO

†Sheridan Sales Co.
2501 Neff Road
Dayton 45414
Tel: (513) 223-3332

†Cramer/Cleveland
5835 Harper Road
Cleveland 44139
Tel: (216) 248-8400

†*Hamilton/Avnet Electronics
954 Senate Drive
Dayton 45459
Tel: (513) 433-0610
TWX: 810-450-2531

†Pioneer/Dayton
1900 Troy Street
Dayton 45404
Tel: (513) 236-9900

†Sheridan Sales Co.
10 Knollcrest Drive
Cincinnati 45222
Tel: (513) 761-5432
TWX: 810-461-2670

†Pioneer/Cleveland
4800 E. 131st Street
Cleveland 44105
Tel: (216) 587-3600

†*Hamilton/Avnet Electronics
761 Beta Drive, Suite E
Cleveland 44143
Tel: (216) 461-1400

†Sheridan Sales Co.
23224 Commerce Park Road
Beachwood 44122
Tel: (216) 831-0130

OKLAHOMA

†Components Specialties, Inc.
7920 E. 40th Street
Tulsa 74145
Tel: (918) 664-2820

OREGON

†Almac/Stroum Electronics
4475 S.W. Scholls Ferry Rd.
Portland 97225
Tel: (503) 292-3534

PENNSYLVANIA

†Sheridan Sales Co.
1717 Penn Avenue, Suite 5009
Pittsburgh 15221
Tel: (412) 244-1640

Pioneer/Pittsburgh
560 Alpha Drive
Pittsburgh 15238
Tel: (412) 782-2300

PENNSYLVANIA (cont.)

Pioneer/Delaware
141 Gibraltar Road
Horsham 19044
Tel: (215) 674-4000
TWX: 510-665-6778

TENNESSEE

Sheridan Sales Co.
6900 Office Park Circle
Knoxville 37919
Tel: (615) 588-5386

TEXAS

Component Specialties Inc.
8330 Burnett Road, Suite 101
Austin 78758
Tel: (512) 459-3308

†Cramer Electronics
13740 Midway Road
Dallas 75240
Tel: (214) 661-9300

†*Hamilton/Avnet Electronics
4445 Sigma Road
Dallas 75240
Tel: (214) 661-8661

†*Hamilton/Avnet Electronics
3939 Ann Arbor
Houston 77063
Tel: (713) 780-1771

†Component Specialties, Inc.
10907 Shady Trail, Suite 101
Dallas 75220
Tel: (214) 357-6511

†Component Specialties, Inc.
8585 Commerce Park Drive, Suite 590
Houston 77036
Tel: (713) 771-7237

UTAH

†*Hamilton/Avnet Electronics
1585 West 2100 South
Salt Lake City, 84119
Tel: (801) 972-2800

WASHINGTON

†*Hamilton/Avnet Electronics
13407 Northrup Way
Bellevue 98005
Tel: (206) 746-8750

†Almac/Stroum Electronics
5811 Sixth Ave. South
Seattle 98108
Tel: (206) 763-2300

†Liberty Electronics
1750 132nd Avenue NE
Bellevue 98005
Tel: (206) 763-8200

WISCONSIN

†*Hamilton/Avnet
2975 Moorland Road
New Berlin 53151
Tel: (414) 784-4510

CANADA

ALBERTA

†L. A. Varah Ltd.
4742 14th Street N.E.
Calgary T2E 6LT
Tel: (403) 276-8818
Telex: 13 825 89 77

BRITISH COLUMBIA

†L.A. Varah Ltd.
2077 Alberta Street
Vancouver V5Y 1C4
Tel: (604) 873-3211
TWX: 610-929-1068
Telex: 04 53167

ONTARIO

L.A. Varah, Ltd.
505 Kenora Avenue
Hamilton L8E-3P2
Tel: (416) 561-9311
TELEX: 061-8349

†*Hamilton/Avnet Electronics
3688 Nashua Drive, Unit GH
Mississauga L4V IM5
Tel: (416) 677-7432
TWX: 610-492-8867

†*Hamilton/Avnet Electronics
1735 Courtwood Cresc.
Ottawa K2C 3J2
Tel: (613) 226-1700
TWX: 610 562-1906

†Zentronics
141 Catherine Street
Ottawa, Ontario K2P 1C3
Tel: (613) 238-6411

†Zentronics
99 Norfinch Dr.
Downsview, Ontario M3N 1W8
Tel: (416) 635-2822
Telex: 02-021694

QUEBEC

†*Hamilton/Avnet Electronics
2670 Paulus
St. Laurent H4S 1G2
Tel: (514) 331-6443
TWX: 610-421-3731

†Microcomputer System Technical Demonstrator Centers
*Microcomputer System Spares Order Point

INTEL CORPORATION, 3065 Bowers Avenue, Santa Clara, California 95051 (408) **987-8080**

Printed in U.S.A./T90/0778/50K CP