intel®

intellec® 8/ Mod 80 Microcomputer Development System

Reference Manual

© INTEL CORP. 1974

intel®

intellec® 8/ Mod 80 Microcomputer Development System

Reference Manual

CAUTION
Do not operate the INTELLEC 8 with the cover removed. The resulting diversion of cooling air may cause overheating and damage to the internal power supplies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

GENERAL DESCRIPTION

The INTELLEC® 8/MOD 80 system (imm8-84A) is a low-cost computer system, designed to simplify the development of microcomputer systems which employ INTEL 8080 microprocessors.

The INTELLEC 8/MOD 80 system uses the 8080 as its central processing unit. The 8080 has a basic cycle time of 2.0 microseconds. The system contains a control console and provides read-write program memory as a substitute for read-only memory. Thus the 8080 chip can be accessed via the control console, and programs can be debugged before being enabled in read-only memory. Turn around time from initial system concept to finished product is shortened, and systems development costs are thus reduced.

The INTELLEC 8/MOD 80 system has its own power supply, cabinet, display and control panel, 8192 bytes (8K) of Random Access Memory, a Programmable Read-Only Memory Module with 4K capacity, a PROM Programmer Module, and an Input/Output Module which contains four 8-bit input ports and four 8-bit output ports as well as provision for serial communications interface.

The Bare Bones 80 is an INTELLEC 8/MOD 80 system without the power supply, display and control console, or cabinet, and is designed for 4K of RAM memory, rack-mounting.

Both the INTELLEC 8/MOD 80 system and the Bare Bones 80 can be expanded up to 16K bytes of memory; in addition, the I/O capability can be expanded to support sixteen input ports and sixteen output ports, or four input ports and twenty-eight output ports.

The standard software for the INTELLEC 8/MOD 80 system includes a resident System Monitor, a Text Editor, and an Assembler. In addition to these INTELLEC 8/MOD 80 resident programs, there are three development programs available, which are designed for operation on LARGE-SCALE HOST COMPUTERS. These are a macro cross-assembler, a microcomputer simulator (INTERP/80), and a PL/M™ compiler. PL/M is a high-level language that can shorten program development time significantly.

SPECIFICATIONS

The INTELLEC 8/MOD 80 system is made up of separate modules, each of which performs a different task in making up a complete system. These modules are:

1) The imm8-83 Central Processor Module, which operates as the Central Processor for the INTELLEC 8/MOD 80 system. In this capacity, it performs the following fuctions:
   a) It controls the execution of program instructions, sending the appropriate control signals to the other modules which make up the INTELLEC 8/MOD 80 system.
   b) It performs all of the necessary arithmetic, logical, and data manipulation operations necessary for program operation.
   c) It controls overall system timing.
2) The imm6-28 Random Access Memory Module, which provides 4,096 8-bit words of Read/Write memory for system use. As many as four cards can be used in a system, for a memory capacity of 16K.
3) The imm6-26 Programmable Read-only Memory Module, which provides up to 4,096 words of Read-only memory in increments of 256 words, and which may be operated in parallel with the system Random Access Memory. Again, more than one card may be used, giving a total Read-only memory capacity of 16K words.
4) The imm8-61 Input/Output Module, which provides four eight-bit input ports and four eight-bit output ports for system Input/Output operations. *Two of the input ports and two of the output ports may be used with integral Teletype communications circuits to provide Teletype I/O.* Up to four of these cards may be used in a system, giving a total of sixteen input ports and sixteen output ports.
5) The imm8-63 Output Module, which provides eight latching output ports for system Output operations.

Up to three of these cards may be used in a system, giving a total capability of twenty-four output ports.

6) The imm6-76 PROM Programmer Card, which gives the INTELLEC® 8/MOD 80 system the capability of programming INTEL 8702A Programmable Read-Only Memory chips.

7) The Front Panel Controller and Display Console, which provides a means of controlling program execution, program debugging, and INTELLEC 8/MOD 80 operation. It also provides displays of system status and information.

8) The chassis and power supplies.

A summary of the specifications for the INTELLEC 8/MOD 80 system is given in Table i-1. Specific information relating to setting-up and operating the INTELLEC 8/MOD 80 system is contained in Chapter 10 of this manual, and in the INTELLEC 8/MOD 80 Operator's Manual.

THE SCOPE OF THIS MANUAL

This manual provides an understanding of the design concepts and capabilities of the INTELLEC 8/MOD 80 system as a whole and its individual modules, and in addition provides detailed theory of operation and implementation information for each module.

For a detailed description of INTELLEC 8/MOD 80 operating procedures, including software operation, see the INTELLEC 8/MOD 80 Operator's Manual. For a detailed examination of programming at an elementary level, suitable for an engineer with no previous programming experience, see the 8080 Assembly Language Programmer's Manual.

INTELLEC 8/MOD 80 Specifications

SPECIFICATIONS	
Word Length	8 bits
Registers	Seven 8-bit general purpose registers, two of which are used to hold Memory Addresses during Memory Reference operations, and one used as the accumulator.
Instruction Set	Seventy-eight instructions, including Memory-index register, index-register-memory, register-to-register, single register, immediate, and memory arithmetic and logic instructions, as well as conditional and unconditional branch instructions, input/output, and machine instructions.
Arithmetic	8-bit parallel, binary, fixed point, two's complement.
Memory	8192 8-bit words, Read/Write; 4096 8-bit words, Read-only. (Expandable to 16,384 words.)
Addressing	Direct – up to 16K bytes. (up to 64K using external enclosures)
Cycle Time	2.0 microseconds
Environment	0° - 55° C.
Power Requirements	5V @ 12 A (max); 6 A (typ); -9V @ 1.8 A (max); 0.5 A (typ); -12V @ 0.03 A (max); 0.016 A (typ); (More power may be required for expanded INTELLEC 8/MOD 80 systems.)
AC Requirement	60 Hz; 115 VAC, 200 Watts
Size	INTELLEC 8/MOD 80: 7" x 17 1/8" x 1/4" Bare Bones 8: 6 3/4" x 17" x 12" (suitable for standard RETMA 7" x 19" panel space).
Weight	30 lb.

Table i-1.

CHAPTER 1 THE INTELLEC® 8/MOD 80 SYSTEM OVERVIEW

The INTELLEC 8/MOD 80 microcomputer development system consists of six independent functional modules and a power supply, housed in a single chassis and enclosure. This section describes the interrelationship of the INTELLEC 8/MOD 80 functional modules, and shows the part played by each module during typical operations.

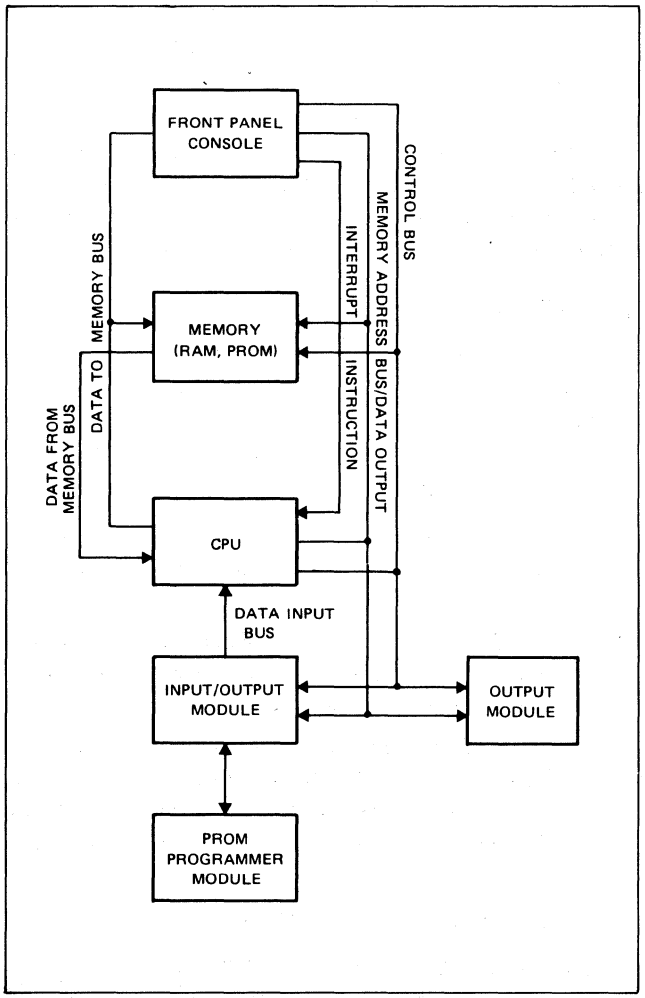


Figure 1-1. A Simplified INTELLEC 8/MOD 80 Block Diagram.

FUNCTIONAL DESCRIPTION OF MODULES

Figure 1-1 illustrates the six functional modules of the INTELLEC 8/MOD 80 system, and shows interconnecting busses. The six functional modules are:

1) A Central Processing Unit (CPU) which performs arithmetic, logical and data manipulation operations.

2) Memory module, which can be Programmable Read-Only (PROM), Random Access (RAM), or a combination of the two. Though Figure 1-1 illustrates memory as a single module, it can be physically implemented as one or more modules, depending on the amount of memory included in a system. The memory module provides data and program storage; a standard system includes two 4K RAM modules and one 4K PROM module.

3) Input/Output module. Physically there can be up to four Input/Output modules in an INTELLEC 8/MOD 80 system. Each Input/Output module provides four individually addressable 8-bit output ports. A serial communications facility, which the INTELLEC 8/MOD 80 system uses for teletype interface, is included in each module.

4) Output module. Physically there can be up to three Output modules in an INTELLEC 8/MOD 80 system. Each Output module provides eight individually addressable 8-bit output ports.

5) A Front Panel Display and Control Console. The Console provides a means for manually monitoring and controlling INTELLEC 8/MOD 80 operations.

6) PROM Programmer Module. This module provides a timing and level shifting circuitry for programming INTEL's 8702A PROMs.

The functional units of the INTELLEC 8/MOD 80 system are interconnected by the following busses:

Bus (a), the Memory Address bus, carries memory

addresses from the console or the CPU to the Memory, Input/Output and Output Modules.

Bus (b) the Output Data bus, carries data from the console or CPU to the Memory, Input/Output and Output Modules.

Bus (c), the data from Memory bus, carries data from memory to the CPU.

Bus (d), the Data Input bus, carries data from input ports to the CPU.

Bus (e), the Interrupt Instruction bus, allows the console to transmit a program interrupt to the CPU.

Bus (f), the Control bus, is used to control instruction execution. Since the console is connected to the control bus, instruction execution can be controlled from the console.

Since the console operates in parallel to the CPU, it contains a considerable amount of parallel logic, including its own data and address registers; thus there are certain states in which the CPU remains in control and the console temporarily suspends operations, and there are other states in which the console completely takes over machine operations.

Conceptually, the CPU module provides the INTELLEC® 8/MOD 80 system with its "computer" capabilities. This module performs arithmetic, logical and data manipulation operations as directed by a stored program.

A stored program is a sequence of numbers (eight binary digits per number) which encode a sequence of individual CPU operations. (Frequently an instruction code is written as two hexadecimal digits rather than eight binary digits). The sequence of individual instructions that constitute a program are stored in the Memory module. If the memory module includes Random Access Memory (RAM), it can also be used to store temporary data that may be generated in the course of executing a program.

Almost all computer applications require information to be transferred between the CPU module and external devices. Such transfers take place via the Input/Output and Output modules.

Communications between the INTELLEC 8/MOD 80 system and an operator occur via the Front Panel Console and teletype.

FRONT PANEL CONSOLE OPERATIONS

Consider how console operations must be performed, given the hardware organization illustrated in Figure 1-1.

Since the console has its own address and data registers, and since there is a bi-directional bus link (through the CPU) between the console and memory, data can be read from memory to console, and written from console to memory.

Although there is no direct path for data from input ports to the console, performing an input access operation from the console causes the input data to be sent through the CPU and onto bus (c), where it is displayed on the console.

There is no direct link between CPU registers and the console. The system monitor has a register interrogation capability.

MEMORY REFERENCE OPERATIONS

This section describes memory reference operations as performed by the INTELLEC 8/MOD 80 system, and is divided into two subsections. Memory input or read operations, and memory output or write operations.

Memory Read Operations

A Memory Read operation is performed in order to obtain data from a certain location in the system memory, and to bring that data to the CPU. It is performed via the following steps:

1) The CPU sends a Memory Address to the Memory modules on the Memory Address bus.

2) The Memory modules send the data contained in the selected memory location to the CPU on the Memory Data Input bus.

The Front Panel can perform a manual Memory Read operation by 'taking over' the Memory Data buses, and by sending a manually entered Memory Address, rather than a CPU-generated Address, to the memory modules.

Memory Write Operations

A Memory Write operation is performed in order to send data from the CPU to a certain selected location in memory. It is performed in the following steps:

1) The CPU sends a Memory Address to the memory modules on the Memory Address bus.

2) The CPU sends the data which is to be stored in memory to the memory modules on the Memory Output Data bus.

3) The CPU sends a control signal to the memory modules which causes the data to be written into the selected memory location.

The Front Panel can perform a manual memory write operation by taking over the Memory Address and Memory Output Data busses, and by sending manually entered Memory Address and Memory Data to the memory module.

INPUT/OUTPUT OPERATIONS

This section describes Input and Output operations as performed by the INTELLEC 8/MOD 80 system, and is divided into three subsections.

Input Operations

An Input operation is performed in order to obtain data from some external device and to bring it into the CPU, where it can be processed. It is performed via the following steps:

1) The CPU sends an I/O Address, which specifies which device is to be used for the Input operation, to the Input/Output modules on the Memory Address bus.
2) The Input/Output module responds by sending the data which is present on the selected Input port back to the CPU on the Data Input bus.

An Input operation can also be performed manually by giving the Front Panel control over the Memory Address bus. It then sends a manually entered I/O Address and an I/O read command to the Input/Output module.

Output Operations

An Output operation is performed in order to send data from the CPU to an external device. It is performed via the following steps:

1) The CPU sends an I/O address, which specifies the device to be used for the Output operation, to the Input/Output and Output modules on the Memory Address bus. At the same time, the CPU sends the data which is to be output to the Input/Output modules on the Output Data bus.
2) The CPU sends an I/O write command to the modules.
3) The Input/Output module latches the data and sends the data which the CPU has supplied to the selected output device.

An Output operation may also be manually executed by giving control of the Memory Address/Data Output bus to the Front Panel. The Front Panel sends a manually entered I/O Address and manually entered data to the Input/ Output and Output modules.

Teletype Operations

Teletype operations are performed in exactly the same fashion as normal, non-teletype Input and Output operations, with the exception that the external device used in the case of Teletype operations is an integral Teletype communications circuit (UART) in the Input/Output module. **Teletype data enters the Input/Output module via input ports 0 and 1; data being sent to the Teletype proceeds through output ports 0 and 1 on the I/O module.** Chapter 3 explains how to install the Teletype ASR33.

INTERRUPT OPERATIONS

An Interrupt operation is performed when an external device which requires servicing sends an Interrupt signal to the CPU. This causes the CPU to interrupt its normal operating sequence, perform the operations required by the external device, and then return to the point at which it was interrupted and resume normal operations. An Interrupt operation is performed in the following steps:

1) The external device sends an Interrupt signal to the CPU. The CPU stops its normal operation and acknowledges the interrupt request.
2) The external device sends an Interrupt Instruction to the CPU.
3) The CPU executes the Interrupt Instruction exactly as if it were a normal instruction.

Usually, the Interrupt Instruction will be a RESTART instruction. A RESTART instruction causes the CPU to branch to a certain location in memory, where an interrupt service routine can be stored.

An Interrupt operation can be performed manually from the Control Console. In order to accomplish this, the Interrupt Instruction is manually entered into the Front Panel. When an Interrupt switch is depressed, the Front Panel will generate an Interrupt signal, and will send the manually entered Interrupt Instruction to the CPU.

In the basic system, only the Control Console initiates interrupts. The ability to interrupt may be extended, however, to the user's peripheral devices, in order to simplify system programming and to increase system throughput. Some modifications to the system, however, are necessary.

PROM PROGRAMMING OPERATIONS

The INTELLEC 8/MOD 80 has been designed to offer an easy means of programming INTEL 8702A Programmable Read-Only memory chips. This is done with the use of the PROM Programming module, and is accomplished by performing three successive Output operations:

1) Send the address within the PROM which is to be programmed
2) Send the data which is to be written into the selected address
3) Send a control word which is used by the PROM Programmer module to initiate programming

The PROM Programmer is used as the external device for each of these Output operations. When it receives the control word, it causes the data specified to be written into the PROM address selected.

CHAPTER 2 THE imm8-83 CENTRAL PROCESSOR MODULE

The imm8-33 Central Processor Module is designed specifically to serve as the central processing unit (CPU) of the INTELLEC® 8/MOD 80 Microcomputer Development System. Its general purpose architecture permits the CPU module to perform similar functions in any eight-bit computer system. Thus the imm8-33, like the other INTELLEC® modules, can be furnished independently on an OEM basis. All inputs and outputs are TTL-compatible, to simplify the external interface.

The basic capabilities of the module are obtained through the use of Intel's 8080 microprocessor. This processor contains an eight-bit accumulator, six eight-bit index registers, and an eight-bit parallel arithmetic and logic unit (ALU). Sixteen latched address lines enable the 8080 to address 65,536 bytes of external memory. As many as 256 eight-bit input ports and 256 eight-bit output ports may also be addressed directly. A sixteen-bit program counter and a sixteen-bit stack pointer permit flexible handling of subroutines. Logic for the processing of holds and interrupts is built into the CPU.

The 8080's internal control logic recognizes and executes 78 different instructions. These are encoded numerically, in a binary format consisting of one, two, or three eight-bit bytes. Instruction categories include:

(a) register-register transfers

(b) register-memory transfers

(c) arithmetic operations, including add and subtract, with and without carry or borrow

(d) Boolean logic operations, including AND, OR, XOR

(e) decimal arithmetic

(f) input/output (I/O)

(g) stack control

(h) interrupt control

(i) register operate

(j) branch control

Five internal status flags enable conditional jumps, calls and returns, based on carry (overflow-underflow), sign, zero, parity, and auxiliary carry.

The Central Processor Module contains a crystal-controlled oscillator and clock generator. These provide a stable timing reference for all circuitry on the board. The use of a 2 MHz clock permits a basic machine cycle of two microseconds, for those instructions that do not reference memory during their execution.

Memory interface and control logic are included on the board. The imm8-83 contains a fully buffered sixteen-line address bus, which communicates with the memory's decoding logic. An eight-line data input bus and a buffered eight-line output bus provide for the actual data transfers. Logic on the board monitors the status signals from the 8080 CPU, and generates a READ/$\overline{\text{WRITE}}$ (R/$\overline{\text{W}}$) command for the control of external memory.

I/O interface and control are also built into the Central Processor Module. I/O peripherals share eight of the module's sixteen address lines with memory, permitting the processor to identify one of 256 input or 256 output devices during execution of an I/O instruction. A separate eight-line input bus provides communication with the input peripherals, while output devices share the module's eight-line data output bus with memory. Control signals generated on the module are available at the edge connector pins, to identify and synchronize input and output operations.

A latched eight-bit output port is included on the imm8-83. It is program addressable (FF_{16}), and is intended primarily for convenience in console de-bugging.

The imm8-83 is equipped with an asynchronous $\overline{\text{INTERRUPT REQUEST}}$ line and with an eight-bit interrupt port, enabling it to process external interrupts. A peripheral device may request service by placing an appropriate binary code on the interrupt port's lines and simultaneously activating the $\overline{\text{INTERRUPT REQUEST}}$ line. By doing so, the interrupting device causes the processor to

execute the instruction whose code appears at the port. Any of the single byte instructions in the processor's repertoire may be used during an interrupt. The restart (RST) instruction, a one-byte call, is particularly useful for interruptive processing. A restart causes the processor to jump to one of eight dedicated memory locations, where service routines may be stored. Return to the interrupted program is accomplished by an ordinary subroutine return (RET), or by one of the conditional return instructions.

The Central Processor Module is also equipped with a $\overline{\text{HOLD REQUEST}}$ line, which enables external devices to conduct direct memory access (DMA) transfers. During an acknowledged $\overline{\text{HOLD REQUEST}}$, the processor suspends its normal activity. The module's address bus and control lines (R/$\overline{\text{W}}$, $\overline{\text{I/O IN}}$, and $\overline{\text{I/O OUT}}$) are disabled, relinquishing control to the active peripheral. The memory input data bus is multiplexed on to the output data bus to facilitate write or output operations. This allows the external device to command the busses and to effect memory transfers directly.

A $\overline{\text{RESET}}$ input permits restarting the program from memory location zero. Any INTERRUPT or HOLD in progress will automatically be terminated by the $\overline{\text{RESET}}$. The program counter is returned to "zero". The accumulator, status flags, and index registers are not cleared. The HL and DE registers may be exchanged.

As a stand-alone CPU, the imm8-83 is almost entirely self contained. It requires only DC power, at levels of:

+12±5%VDC @ 0.06 Amperes
+5±5%VDC @ 1.5 Amperes
-9±5%VDC @ 0.1 Amperes

All circuitry is mounted on a 6.18" x 8.00" printed circuit board, and signal and power connections enter the module through a dual 50-pin double-sided PC edge connector (0.125" centers). No special installation will be necessary.

The imm8-83 may also be used as a plug-in substitute for the imm8-82, to update existing INTELLEC® 8/MOD 80 systems. Installation of the Central Processor Module is straightforward, and the CPU module itself requires no changes. Minor modifications are necessary, however, in the case of other modules.

Although the imm8-83's edge connector pins correspond nominally to those of the imm8-82, it has not been possible to maintain a strict and complete logical overlap in the address and control lines. The imm8-60 I/O Module, the imm8-62 Output Module, and the Front Panel Controller will therefore have to be modified slightly.

Intel provides a kit so simplify the conversion of existing INTELLEC 8 systems. This contains the imm8-83 module, an imm8-61 module, a new front panel controller, and all MOD 80 software. It reduces the conversion to a simple plug-in swap. Those who elect to modify the modules they presently have will find the instructions for doing so in the sections of this book that pertain to those modules.

The following subsections furnish a complete description of the imm8-83 Central Processor Module. The first describes a generalized processing system, at a fairly elementary level, to provide background information for those who are relatively unfamiliar with processors and with the language used to describe them. Users who feel competent to discuss processors at an advanced level should skip this introductory section. The second section describes the functional organization of the processor module. Detailed information on the 8080 CPU is given in the third section. In the fourth section we show how the peripheral logic supports the functions that the 8080 performs. Finally, in the fifth section, we give reference information which will be of value to those planning to use the module outside the INTELLEC 8/MOD 80 system.

THE FUNCTION OF A CPU

This section is intended for those who are unfamiliar with basic computer concepts. It provides background information and definitions which may be useful in later sections of this chapter. Those already familiar with computers may skip this material, at their option. It is organized to permit quick reference.

The Computer System

The INTELLEC® 8 is a modular computer system. This means that the processing functions, the memory functions, and the input/output functions are built into separate plug-in cards which are then combined to form a system. Because the functions of each of the modules are fairly well-defined, individual plug-ins enjoy a certain degree of independence. They are advertised as having stand-alone capability, meaning that they are generally capable of performing their functions in any system similar to the INTELLEC® 8. The modular organization of this reference manual intentionally reflects the modularity of the system it describes.

You must keep in mind, however, that modularity confers a very limited degree of independence. None of these modules can do anything useful outside a system. As a result, the discussion of any individual module must refer continually to the activities of other modules in the same system. It is therefore very important to know something about the functions that each component in a system must perform, before discussing the processor module in detail.

A digital computer consists of:

(a) A central processsing unit (CPU)
(b) A memory
(c) Input and output provisions (I/O)

This applies, in essence, to all such computers. It applies to the INTELLEC 8.

Memory and I/O are relatively simple functions and are fairly easy to rationalize. The memory serves primarily as a place to store **instructions** the coded pieces of data that direct the activities of the CPU. A group of logically related instructions stored in memory is referred to as a **program**. The CPU extracts these instructions singly, in a logically determinate sequence, and uses them to initiate processing actions. If the program structure is coherent and logical, processing produces intelligible and useful results.

Processing is a complex activity, and one which requires a lot of explanation. For now, we shall have to be content with an intuitive understanding of what is meant by the term. Assume for the moment that the machine somehow manipulates data arithmetically to produce the desired result. We shall describe the process later, in detail.

Program instructions are a form of input. The computer can generate an output entirely on the basis of instructions and data stored in its memory by the programmer. In most cases, however, it is desirable to have input provisions which augment the program as a source of data. This is not difficult to understand. One of the most useful features of the computer is its speed, its ability to react quickly to changes in its data environment or to process large volumes of data. In one case, the machine must have access to information much more rapidly than a human operator can supply it. In the other, it requires access to a data bank which can easily exceed its memory capacity. Both problems can be solved partially by providing the machine with one or more **input ports**. The machine can address these ports and read the data contained there, in a manner very similar to that used to read from its memory. The addition of input ports enables the computer to receive information from external machinery, at high rates of speed and in large volumes.

Central processing units operate so rapidly that their responses often seem instanteneous to human operators, but processing usually requires several stages. Many individual instructions can intervene between the input of data and the output of results. Consider the simple addition of two numbers presented to two different input ports. The machine must read the number at one port first. It stores the value obtained in a temporary location, while it reads the number at the second port. Then the number in temporary storage is added to the first, to obtain the desired result. More complex functions than this can generate many stages of intermediate results, all requiring temporary storage at some time during the execution of the program. Thus a secondary function of the memory becomes apparent, the storage of intermediate data. In the course of a processing task, the CPU may store data temporarily in some memory location from which it can later be retrieved. The processor will generally write into a portion of the memory not occupied by program instructions, although the machine can "program itself" under certain exceptional circumstances. Reading and writing in memory are accomplished by means of program instructions known as **memory referencing** instructions, so called because they specify or imply a memory address as an integral part of the instruction. Memory referencing operations will be explained more fully when we describe the CPU itself.

One or more **output ports** permit the computer to communicate the results of its processing to the outside world. The output may go to a display, for use by human operators, or it may go directly to other machines whose responses are controlled by the processor. The output ports are necessary in either event, if the processor is to perform any useful function. Output ports are addressable, in much the same manner as inputs. The input and output ports together permit the processor to interact with the outside world.

The central processor unifies the system. It controls the functions performed by the other components. The CPU must be able to fetch instructions from memory and execute them, and it must be able to reference memory and I/O ports as necessary in the execution of instructions. It must also be able to recognize and respond to external control signals, including INTERRUPT, HOLD, and WAIT requests. These apparently straightforward requirements imply a certain complexity in the way that the CPU operates. Some of the features that enable a processor to perform these functions are described below.

The Architecture of a CPU

TIMING

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

PROGRAM COUNTER

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

Each of the locations in memory is numbered, to distinguish it from all other locations in memory. The

number which identifies a memory location is called its **address**

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

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

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

Program jumps are a convenience for programmers, and the description of their use can become complicated. However, a basic use of the jump can be illustrated here: that where the programmer must interleave program steps with data upon which the processor is directed to operate:

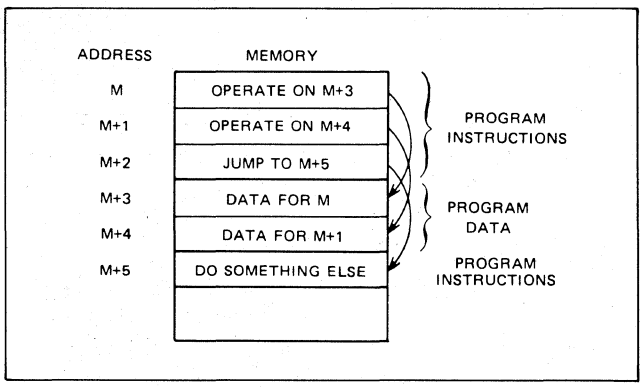


Figure 2-1. Program Jump.

If the jump at location M + 2 were omitted, the processor would continue to operate on the assumption that the program structure was sequential. It would attempt to fetch and execute the data in location M + 3 and M + 4 as though those locations contained instructions. The program would most probably produce results quite contrary to those that the programmer expected.

THE STACK

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

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

To understand the value of subroutines, consider the case where it is necessary to output five characters to a line printer, in the course of a 200 step segment of the main program. Suppose that the program which outputs the character is the same, regardless of the actual idetity of the character; in other words that it is possible to write a generalized program which can output any character that the main program supplies. And assume further that 20 steps are required for such an operation. We then have two possible ways of coding this problem.

One possibility is to write the 20 output steps into the main program, each time we desire to output a character. The total length of the program will be 200 plus 5x20, or 300 steps in all. The other possibility is to write the 20 step output program as a subroutine, and cause the main program to jump to the address of the subroutine (call the subroutine) whenever it is necessary to output a character. In this case, the 20 step program need be stored only once. The total number of instructions in memory will be 200+20, or 220.

Observe that the subroutine in this example will still be executed five times. The processor will still have to perform 300 operations, regardless of how we choose to code this problem. The subroutine structure, however, is preferred. For one thing, it conserves the programmer's time, since he need only code the output routine once. For another, it conserves memory space, for the actual output instructions occupy only 20 memory locations, rather than 100. These are significant advantages.

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

The last instruction in any subroutine is a **return**. Such an instruction need specify no address. When the processor fetches a return instruction, it simply replaces the current contents of the program counter with the address on the top of the stack. This causes the processor to resume

execution of the calling program at the point immediately following the original call.

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

Processors have different ways of maintaining stacks. Some have facilities for the storage of return addresses built into the processor itself. Other processors use a reserved area of memory as the stack and simply maintain a **pointer** register which contains the address of the most recent stack entry. The integral stack is usually more efficient, since fewer steps are involved in the execution of a call or a return. The external stack, on the other hand, allows virturally unlimited subroutine nesting. It also permits saving the contents of the other CPU registers, and so provides for greater flexibility in the handling of subroutines.

INSTRUCTION REGISTER AND DECODER

Every computer has a **word length** that is characteristic of that machine. In most eight-bit systems, it is most efficient to deal with eight-bit binary fields, and the memory associated with such a processor is therefore organized to store eight bits in each addressable memory location. Data and instructions are stored in memory as eight-bit binary numbers, or as numbers that are integral multiples of eight bits: 16 bits, 24 bits, and so on.

This characteristic eight bit field is sometimes referred to as a **byte**.

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

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

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

MULTIPLE WORD INSTRUCTIONS

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

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

MEMORY SYNCHRONIZATION

As previously stated, the activities of the processor are referred to a master clock oscillator. The clock period determines the timing of all processing activity.

The speed of the processing cycle, however, is limited by the memory's **access time**. Once the processor has sent a fetch address to memory, it cannot proceed until the memory has had time to respond. Many memories are capable of responding much faster than the processing cycle requires. A few, however, cannot supply the addressed byte within the minmum time established by the processor's clock.

Therefore, many processors contain a synchronization provision, which permits the memory to request a **wait** phase. When the memory receives a fetch address, it places a low level on the processor's READY line, causing the CPU to idle temporarily. After the memory has had time to respond, it frees the processor's READY line, and the instruction cycle proceeds.

ARITHMETIC LOGIC UNIT

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

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

The fundamental operational unit in the ALU is the accumulator. This is the basic register in which binary quantities are represented symbolically. Different machines use slightly different approaches, but in general the accumulator is both a source and a destination register. A typical instruction will direct the ALU to add the contents of some other register to the contents of the accumulator, and to store the result in the accumulator itself.

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

The adder is a minimum provision, but a comprehensive one as well. Using only the basic adder, a capable programmer can write routines which will subtract, multiply and divide, giving the machine complete arithmetic capabilities. In practice, however, most ALUs provide other built-in functions, including hardware subtraction, Boolean logic operations, and shift capabilities.

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

We have touched here very briefly on some of the features of an ALU, in an attempt to explain their provisions. However, most of the ALU's operations are really outside the province of the logic designer. He never sees their results directly. It is the programmer who is chiefly concerned with the capabilities of the ALU, since they affect directly his ability to construct programs that produce the desired results. Readers who require a more detailed explanation of the arithmetic logic unit are referred to a good programming text, such as the 8080 Assembly Language Programmer's Manual.

INTERRUPTS

Interrupt provisions are included on many central processors, as a method of improving the processor's efficiency. To understand the mechanism of an interrupt, consider the hypothetical situation where two separate processors are working simultaneously on two separate jobs. One processor is working steadily at a low priority job. The other is working at infrequent intervals on a high priority assignment. The processor assigned to the high priority task is chronically underemployed, and we may readily improve the efficiency of this configuration, as follows.

We use a single processor, but one which is equipped to sense an external request for service; in other words, to recognize and interrupt. We set this processor to work on the low priority job, with the provision that it jump to a routine designed to service the high priority channel whenever it receives an interrupt. The processor resumes the low priority task when it is finished handling the interrupt. Note that this is, in principle, quite similar to a subroutine call, except that the jump is initiated externally rather than by the program.

This is quite acceptable, if the low priority task does not consume 100% of the processor's time; that is, if the processor is not required to run at top speed continuously in order to meet the requirements of that job. In most cases this is not a problem, since real-time systems are generally designed with a considerable safety margin in mind. The average load on a properly designed system is well below its peak capacity, to allow for statistically infrequent bursts of activity, and to allow for some inevitable down time.

The interrupt feature in this simple example permits us to increase processing efficieny up to 100%. More complex interrupt structures are possible, in which several interrupting devices share the same processor but have different priority levels. Interruptive processing is an important feature, that enables us to maximize the utilization of a processor's inherent capacity.

HOLD

Another important feature that improves the throughput of a processor is the **hold**. The hold provision enables **direct memory access** operation (DMA).

In ordinary input and output operations, the processor itself supervises the entire transfer. Information to be placed in memory is transferred from the input to the processor, and the from the processor to the designated memory location. In similar fashion, information that goes from memory to output goes by way of the processor.

Some peripheral devices, hoever, are capable of transferring information to and from memory much faster than the processor itself can accomplish the transfer. If any appreciable quantity of data must be transferred to or from such a device, then system throughput can be increased substantially by having the device accomplish the transfer directly. The processor must temporatily suspend its operation during such a transfer, to prevent conflicts that would arise if processor and peripheral attempted to access memory simultaneously. It is for this reason that a hold provision is included on some processors. By placing a hold request, the peripheral with data to transfer can cause the

processor to pause until the DMA is complete. A theoretical improvement in I/O efficiency of up to 100% may be gained by the judicious use of DMA.

FUNCTIONAL ORGANIZATION OF THE CENTRAL PROCESSOR MODULE

The Intel 8080 Eight-Bit Parallel Central Processor Unit is the major functional element on the imm8-83 Central Processor Module. All the other logic on the module supports the functions which the 8080 CPU performs. This leads to a natural and convenient distinction, between the "processor" and its "peripheral logic."

There are a number of reasons for relegating certain functions to support logic, rather than incorporating them into the processor chip itself. The buffering of address and data lines, for example, is a high power function, and high power functions are fundamentally incompatible with small package sizes. Large, hot-running components not only increase the size of the package, they increase its susceptibility to failure. The 8080 is basically a miniature divice, and for this reason, the buffering functions are referred to external circuitry.

Much the same argument applies to multiplexing functions. These too would logically necessitate enlargement of the package, to enable the device to dissipate the additional power. Moreover, functions of this kind imply an expanded number of input and output pins, and this also is inconsistent with small package size. External logic is therefore required for multiplexing.

Still other functions are not amenable to integration. The clock reference oscillator is a prime example. It is not yet possible to fabricate a stable frequency reference using monolithic techniques, so that the clock function too must be provided by peripheral logic.

And finally, some functions are too specialized to be included on the chip directly. One example is the programmed display port (output port FF_{16}) which is built into the imm8-83.

Another would be signal functions such as $\overline{\text{I/O OUT}}$ on the imm8-83, which are dictated by the particular application rather than by the processing function. Signals of this kind are derived by logical conditioning of the 8080's basic outputs. Though the number of functions is often modest, incorporating them into a general-purpose device such as the 8080 would tend to limit the range of applications which the CPU could serve. Such functions are therefore omitted from the chip and are left to the discretion of individual designers to provide.

A number of considerations thus prevent us from realizing a true "computer on a chip," even though the vast majority of the complex functions performed by a classical "computer" are in fact incorporated into the 8080. Memory, input/output, and control functions such as those described above are omitted for practical reasons; in spite of the fact that their inclusion is technically feasible. The omission works to the advantage of the systems designer, who is thereby freed to specify the speed and capacity of his memory, the number of input and output ports in his system, and the number and nature of control functions to be performed by his central processor unit.

The consequence is, however, that the central processor function is essentially a modular activity, rather than a true chip function; that the bulk of central processing activity can be delegated to an all-purpose chip, but that peripheral logic will almost always be required to round out the chip's capabilities. This is the case in the INTELLEC 8/MOD 80 system.

The imm8-83 therefore consists of the 8080 CPU and the logic that supports the functions of the processor. In addition to the 8080 CPU, the module contains the following functional blocks:

(a) timing generator
(b) address buffer
(c) data buffer
(d) input multiplexer
(e) status latches
(f) command logic
(g) wait logic
(h) hold logic
(i) interrupt logic
(j) reset generator
(k) output port

The functional relationship between these blocks is shown in Figure 2-2.

The 8080 CPU exercises complete control over the rest of the logic on the module, according to the instructions it receives from memory.

The timing generator consists of a clock oscillator, a counter section, level shifting circuitry, and gating logic. The crystal-controlled oscillator delivers a symmetircal 32 MHz signal to the input of the counter section, which in turn uses this input to derive two non-overlapping 2 MHz clock phases, designated ϕ_1 and ϕ_2. These are applied to the level shifter and used to drive the 8080 CPU. Logic circuitry within the CPU generates a SYNC pulse each time the processor begins a sub-cycle. From the ϕ_1, ϕ_2, and SYNC outputs, the gating logic develops CLKA, CLKB, and SYNCA signals. Signals produced in the timing section then govern all the other activities of the Central Processor Module.

The address buffer receives its low power input from the 8080's sixteen-line address bus. A sixteen-line high power output is forwarded to the memory and to the I/O peripherals. Note that latching and timing are controlled internally by the CPU, and that the buffer's output merely follows the processor's address lines. Data on the address

bus specifies the destination of data concurrently on the module's main data bus. Either a memory location or an I/O peripheral may be specified. The address buffer also receives a HOLD ACK signal from the hold logic section, whenever the module acknowledges an outside $\overline{\text{HOLD REQUEST}}$. During the time that a hold is in prograss, the address buffer's output is disabled. Disabling the buffer's output enables the requested peripheral to command the address bus directly during the DMA transfers.

The function of the data buffer is similar to that of the address buffer. This section receives an eight-line low power input from the 8080's main data bus, and forwards a high power eight-line output to memory and to the output peripherals. All data transferred out of the processor goes by way of this output bus.

Note however, that somewhat different provisions are made for disabling the data buffer during hold operations. Refer to Figure 2-2. Unlike the address buffer, the data buffer receives an explicit enabling signal ($\overline{\text{DB OUT}}$) from the hold logic section. As shown in the diagram, the peripheral requesting a hold can override the hold logic by commanding the $\overline{\text{DB OUT}}$ line directly. This becomes necessary in those cases where the requesting peripheral has to communicate with memory via the imm8-83's main data bus. The data buffer must be enabled during the time that data is being transferred from memory to the requesting device, but disabled during the time that data is being transferred from the requested device to memory. Control of $\overline{\text{DB OUT}}$ is accorded the peripheral requesting a hold, to provide for bilateral data transfers.

The input multiplexer is a three-way switch which selects and forwards one of three eight-line input channels to the processor. Input signals from the processor, the status latches, and the command logic enable the multiplexer to select data from memory, data from the

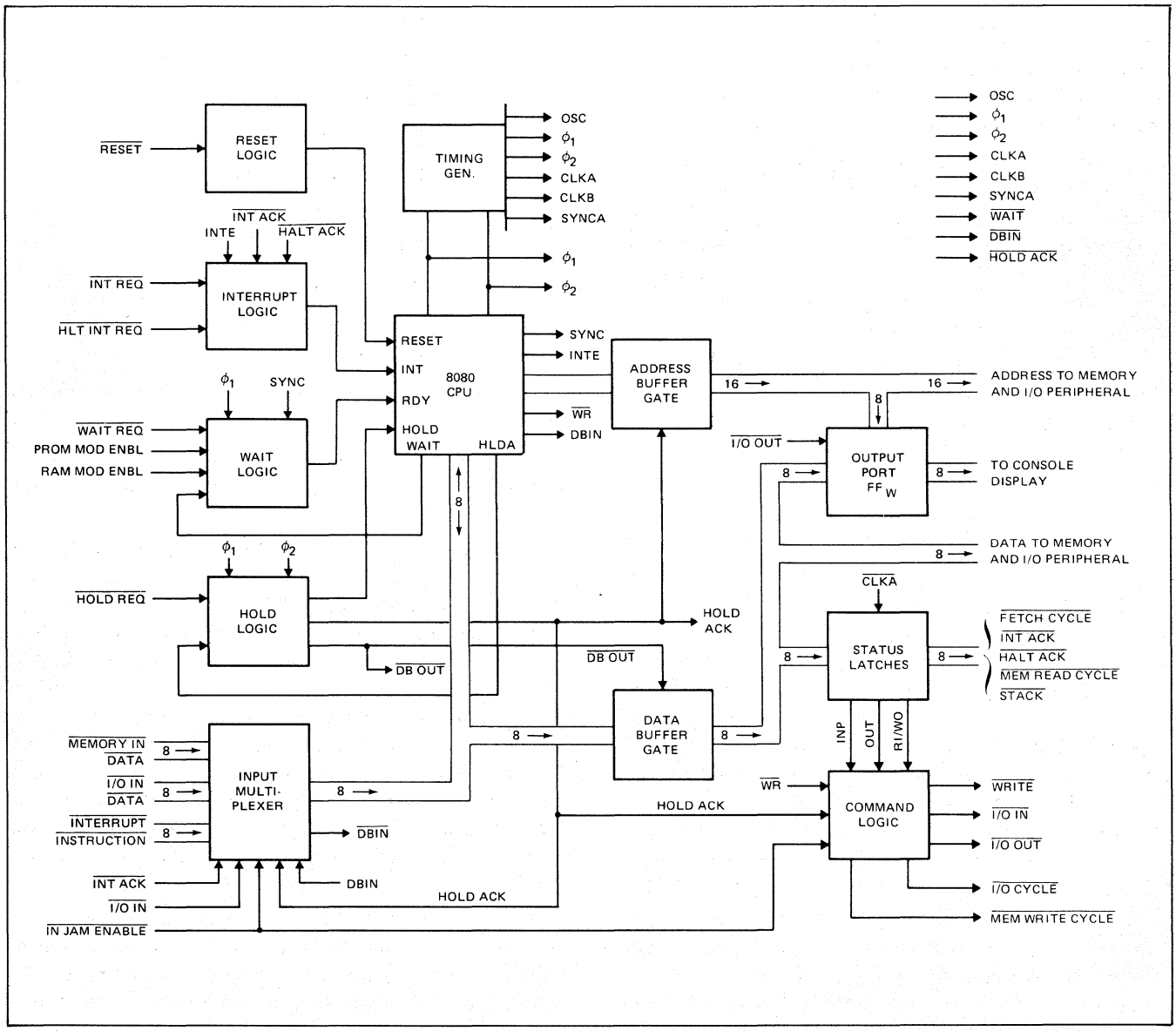


Figure 2-2. CPU Module Functional Block.

input peripherals, or data from the interrupt bus for input to the processor.

The 8080's instruction cycle is composed of one or more machine cycles. The number and kind of macnine cycles in a given instruction cycle depends upon the instruction that the processor happens to have fetched from memory. There are nine possible kinds of machine cycles:

(a) FETCH
(b) MEMORY READ
(c) MEMORY WRITE
(d) STACK READ
(e) STACK WRITE
(f) INPUT
(g) OUTPUT
(h) INTERRUPT
(i) HALT

A description of machine cycles is deferred until Section 3.3, where we discuss the 8080 CPU. Without getting too involved in a description of the processor's activities, however, we may observe that each machine cycle calls for a slightly different response on the part of the peripheral logic. To aid in developing the proper control functions, the CPU outputs status information at the beginning of every machine cycle. Status latches are provided to capture this data, for use by the command logic.

The status latch section receives an eight-line data input from the module's data buffer and a $\overline{\text{CLKA}}$ strobing pulse from the timing generator. These inputs enable the latches to record the eight status information bits that are published on the processor's main data bus at the beginning of every machine cycle. Status information helps coordinate the activities of peripheral logic, so that its responses are appropriately keyed to the internal activities of the processor.

The command logic obtains it principal inputs from the status latches and from the 8080 CPU. Other inputs to this section are the HOLD ACK from the hold logic section and the $\overline{\text{IN JAM ENBL}}$ from the INTELLEC 8's Front Panel Controller. Using these, the command logic generates a $\overline{\text{WRITE}}$ command for the control of external memory, as well as $\overline{\text{I/O IN}}$ and $\overline{\text{I/O OUT}}$ signals for the control of I/O peripherals. $\overline{\text{I/O CYCLE}}$ and $\overline{\text{MEM WRITE CYCLE}}$ outputs are available to the INTELLEC 8's console status display. These, together with the $\overline{\text{FETCH CYCLE}}$ and the $\overline{\text{MEMORY READ CYCLE}}$ outputs from the status latches, enable the console logic to identify the machine cycle in progress.

Wait logic montors the $\overline{\text{WAIT REQUEST}}$ line from the system memory. If the memory is slow to respond to the processor's read or write commands, the wait logic causes the processor to idle unitl the memory can complete the transaction. A WAIT signal is available to external circuitry during the time that the processor is idling, and serves to acknowledge the $\overline{\text{WAIT REQUEST}}$. A $\overline{\text{WAIT REQUEST}}$ may be of indefinite length, but the generated WAIT interval is always an integral multiple of the processor's clock period.

Neither the imm6-28 RAM Memory Module nor the imm6-26 PROM Memory Module used with the INTELLEC 8/MOD 80 can respond fast enough to avoid placing the 8080 CPU in a WAIT state. The RAM Memory Module and the PROM Memory Module have typical access times of 700 nanoseconds and 1200 nanoseconds respectively. The RAM module therefore requires at least one full wait interval during every memory reference. The PROM module requires two. Circuitry in the wait logic section uses the CPU module's ϕ_1 and SYNC timing signals, in conjunction with external RAM MOD ENBL and PROM MOD ENBL signal, to generate an automatic $\overline{\text{WAIT REQUEST}}$ of the desired duration whenever one of these modules is selected. The imm8-83 is designed to respond to a $\overline{\text{PROM MOD ENBL}}$ with an override of the delay introduced for the imm6-28 and 6-26 boards. $\overline{\text{PROM MOD ENBL}}$ may therefore be used to enable memories capable of responding to the 8080 without delay.

The hold logic receives a $\overline{\text{HOLD REQUEST}}$ signal from one or more peripheral devices. It also receives ϕ_1 and ϕ_2 timing signals from the module's timing generator. When a $\overline{\text{HOLD REQUEST}}$ coincides with the rising edge of the ϕ_1 clock pulse, the hold logic forwards a HOLD to the CPU itself. Logic within the 8080 determines when the re-clocked hold request will be acknowledge, to ensure that any processing functions in progress are not disrupted. The processor will acknowledge the HOLD within five clock periods (2.5 microseconds), by sending a HLDA signal to the hold logic section. After a brief delay provided by ϕ_2, the hold logic responds by:

(a) floating the module's address bus
(b) floating the 8080's data bus
(c) floating the $\overline{\text{WRITE}}$ output line to memory
(d) floating the $\overline{\text{I/O IN}}$ output line
(e) floating the $\overline{\text{I/O OUT}}$ output line

This action prevents the processor from exerting any influence on memory, via the data busses or by means of control signals. The peripheral originating the $\overline{\text{HOLD REQUEST}}$ is therefore free to command the memory, until such time as the $\overline{\text{HOLD REQUEST}}$ is retracted.

The interrupt logic monitors the $\overline{\text{INTERRUPT REQUEST}}$ and the $\overline{\text{HALT INTERRUPT REQUEST}}$ lines from external devices. This section also receives $\overline{\text{INT ACK}}$ and $\overline{\text{HALT ACK}}$ signals from the status latch section. The interrupt logic uses these inputs to develop an interrupt signal which is forwarded to the processor's INTERRUPT input. Requests originating at the $\overline{\text{INTERRUPT REQUEST}}$ and the $\overline{\text{HALT INTERRUPT REQUEST}}$ inputs have much the same effect. The only significant difference between the two inputs is that the processor responds to a $\overline{\text{HALT}}$

$\overline{\text{INTERRUPT REQUEST}}$ only when it is stopped. Under those circumstances, an interrupt is required to restart the machine.

The 8080 CPU provides an interrupt enabling signal (INTE) to the interrupt logic, indicating when the processor's INTERRUPT input has been disabled by the program in progress. Instructions in the CPU's repertoire permit the explicit enabling and disabling of this input. From the INTE signal, the interrupt logic develops an $\overline{\text{INT DISABLE}}$ signal which flags the processor's status to peripheral devices. No interrupt requests are recognized unless the program expressly enables the processor's INTERRUPT line. A processor which has been stopped inadvertently while the INTERRUPT input is disabled must be reset or brought up from a cold start, in order to restore it to operation.

The processor module responds to an interrupt by altering the sequence of events that occurs at the end of the last instruction cycle. The processor enters an alternative INTERRUPT machine cycle, rather than the normal FETCH machine cycle. As it customarily does, the processor sends out address and status information at the beginning of the machine cycle, but the program counter is not incremented. An INTA status bit identifies the machine cycle as an INTERRUPT.

These are the only unusual events as far as the processor itself is concerned. In all other respects, the INTERRUPT Machine-cycle resembles an ordinary instruction fetch. Peripheral logic, however, senses the entry into the interrupt mode. The input multiplexer responds by selecting the interrupt instruction port instead of the processor's memory data in port. Thus any eight-bit data word presented to the interrupt port gets interpreted as an instruction by the processor.

Any single-byte instruction may be inserted. There are several possibilities. A halt (HLT) instruction may be used to stop the processor upon completion of some task. an external reset will be necessary for restarting the CPU. Or an output instruction may be used to output the accumulator's contents during a critical phase of the programming. Control and debugging are therefore two possible useds of the interrupt feature.

But by far the most convenient instruction for use with interrupts is the restart (RST). the RST is one byte call instruction especially intended for use with interruptive processing. Its binary instruction field contains three variable digits that permit the programmer to specify a jump to one of eight memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. One of these locations can be used to store the first instruction of a program designed to service the interrupting device. Or it can store the first byte of an ordianry three byte call (CALL), to another location where such a program begins.

An important use of the RST instruction is the start-up of the processor, following the execution of a halt instruction. The machine may be re-started by means of an interruptive jump to memory location ϕ_{10} (or to some other desired location).

Note that in the INTELLEC 8/MOD 80 system the operator's console is the only device for which interrupt capability is provided. Minor modifications, however, could extend the privilege to other peripheral devices.

Reset logic permits an external device to initialize the processor. Logic in this section also senses a power-up sequence, and forces a RESET automatically under those conditions. External application of a 1.5 microsecond pulse (minimum) or the interruption of power to the module restores the processor's program counter to zero. No other circuitry on or around the chip is affected, except for the interrupt request latch which is reset.

The built-in output port receives an eight-line I/O address from the module's sixteen-line address bus (A_{15}–A_8). It also receives an $\overline{\text{I/O OUT}}$ signal from the command logic. These commands cause the latches in the output port to register the contents of the module's data out bus, whenever the decoding logic senses a coincidence of $\overline{\text{I/O OUT}}$ and the hexadecimal address FF_{16}. In the INTELLEC 8/MOD 80 system, the port's output lines communicate with indicators on the console panel. This enables the operator to examine the contents of the processor's accumulator, during test and de-bugging operations.

8080 EIGHT-BIT PARALLEL CENTRAL PROCESSOR UNIT

A brief description of Intel's 8080 CPU is essential to a thorough understanding of the imm8-83 Central Processor Module.

The 8080 is a monolithic LSI central processor, designed for applications that use an eight-bit binary instruction/data format. It is fabircated using N-channel silicon gate technology and is furnished in a 40-pin dual in-line ceramic package. The use of advanced fabrication and layout techniques has produced an exceptionally fast microprocessor. The basic machine cycle of the 8080 is two microseconds, for instructions that do not reference memory during their execution. This compares with a twenty microsecond cycle in the earlier 8008 CPU.

Package geometry and pin configurations are shown in Figure 2-3. All pins, except the clock inputs, are at TTL levels.

A list of the 8080's capabilities reads much like a description of the imm8-83 Central Processor Module itself. In a very real sense, it is the chip processor that determines the character of the module. The 8080 CPU has a repertoire of 78 basic instructions, with provisions for arithmetic and logical operations, register-register and register-memory transfers, subroutine handling, I/O transactions, and decimal arithmetic. Four internal status flags enable the

user to program conditional branches based on carry, sign, zero, and parity.

Using its sixteen latched address lines, the 8080 can access 65,536 (64K) memory locations directly. As many as 256 input devices and up to 256 output devices may be addressed during I/O operations, using either the upper or the lower eight address lines (A_0-A_7 and A_8-A_{15} are redundant for the purpose of I/O instructions). The 8080's inherent addressing capability can be extended further by the use of bank switching, where one of the output ports is used to select among several available blocks of memory.

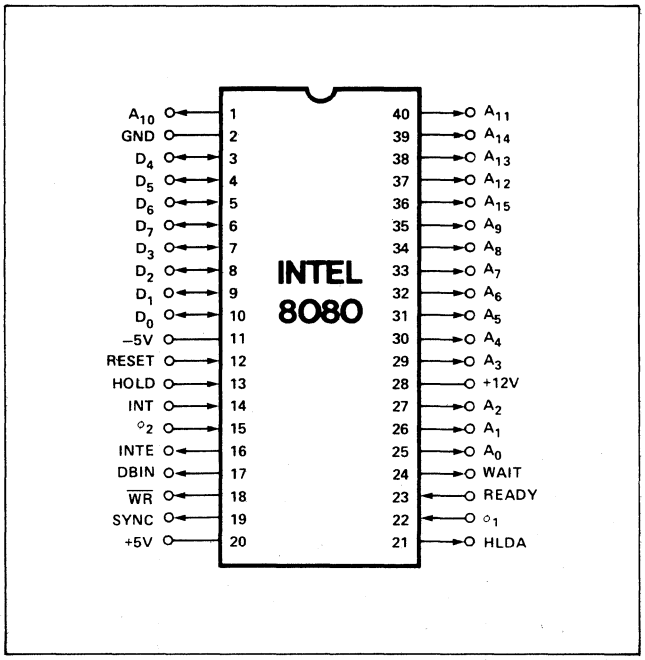


Figure 2-3. 8080 CPU Package Configuration.

The 8080 contains 6 eight-bit index registers (scratchpad). Two of these, the H and the L registers, are designed to double as an address pointer during the execution of memory referencing instructions. A sixteen-bit program counter enables the CPU to address instructions stored in any portion of memory, and a sixteen-bit stack pointer permits the unlimited nesting of subroutines (or multiple-level interrupts). Built-in logic for the processing of holds and interrupts, and a synchronization provision for slow memories, round out the CPU's capabilities.

Architecture of the 8080 CPU

The 8080 CPU consists of the following functional units:

- Register array and address logic
- Arithmetic and logic unit (ALU)
- Instruction register and control section
- Bidirectional, tri-state data bus buffer

Figure 2-4 illustrates the functional blocks within the 8080 CPU.

REGISTERS

The register section consists of a static RAM array organized into six 16-bit registers:

- Program counter (PC)
- Stack pointer (SP)
- Six 8-bit general purpose index registers arranged in pairs, referred to as B,C; D,E; and H,L
- A temporary register pair called W,Z

The program counter maintains the memory address of the current program instruction and is incremented automatically during every instruction fetch. The stack pointer maintains the address of the next available stack location in memory. The stack pointer can be initialized (with a LXI SP instruction) to use any portion of read-write memory as a stack. The stack pointer is decremented when data is "pushed" onto the stack and incremented when data is "popped" off the stack (i.e., the stack grows "downward").

The six general purpose registers can be used either as single registers (8-bit) or as register pairs (16-bit). The temporary register pair, W,Z, are not program addressable and are only used for the internal execution of instructions.

Eight-bit data bytes can be transferred between the internal bus and the register array via the register-select multiplexer. Sixteen-bit transfers can proceed between the register array and the address latch or the incrementer /decrementer circuit. The address latch receives data from any of the three register pairs and drives the 16 address output buffers (A_0-A_{15}), as well as the incrementer/ decrementer circuit. The incrementer/decrementer is a purely combinatorial circuit that receives data from the address latch and sends it to the register array. The 16-bit data can be incremented or decremented or simply transferred without any operation being performed.

ARITHMETIC AND LOGIC UNIT (ALU)

The ALU contains the following registers:

- An 8-bit accumulator (ACC and a carry/link flip-flop (CY)
- An 8-bit temporary accumulator (ACT) and a temporary carry flip-flop (ACT)
- A 5-bit flag register: zero, carry, sign, parity and auxiliary carry
- An 8-bit temporary register (TMP)

Arithmetic, logical and rotate operations are performed in the ALU. The ALU is fed by the temporary register (TMP) and the temporary accumulator (ACT) and carry flip-flop. The result of the operation can be transferred to the internal bus or to the accumulator; the ALU also feeds the flag register.

The temporary register (TMP) receives information from the internal bus and can send all or portions of it to the ALU, the flag register and the internal bus.

The accumulator (ACC) can be loaded from the ALU and the internal bus and can transfer data to the temporary accumulator (ACT) and the internal bus. The contents of the accumulator (ACC) and the auxiliary carry flip-flop can be tested for decimal correction during the execution of the DAA instruction (see Appendix A).

INSTRUCTION REGISTER AND CONTROL

During an instruction fetch, the first byte of an instruction (containing the op code) is transferred from the internal bus to the 8-bit instruction register.

The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, combined with various timing signals, provides the control signals for the memory, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and cycle timing signals.

DATA BUS BUFFERS

This 8-bit bidirectional tri-state buffer is used to isolate the CPU's internal bus from the external data bus (D_0 through D_7). In the output mode, the internal bus content is loaded into an 8-bit latch that, in turn, drives the data bus output buffers. The output buffers are switched off during input or non-transfer operations.

In the input mode, data from the external data bus is transferred to the internal bus. The internal bus is precharged at the beginning of each internal state, except for the transfer state (T3—described later in this chapter).

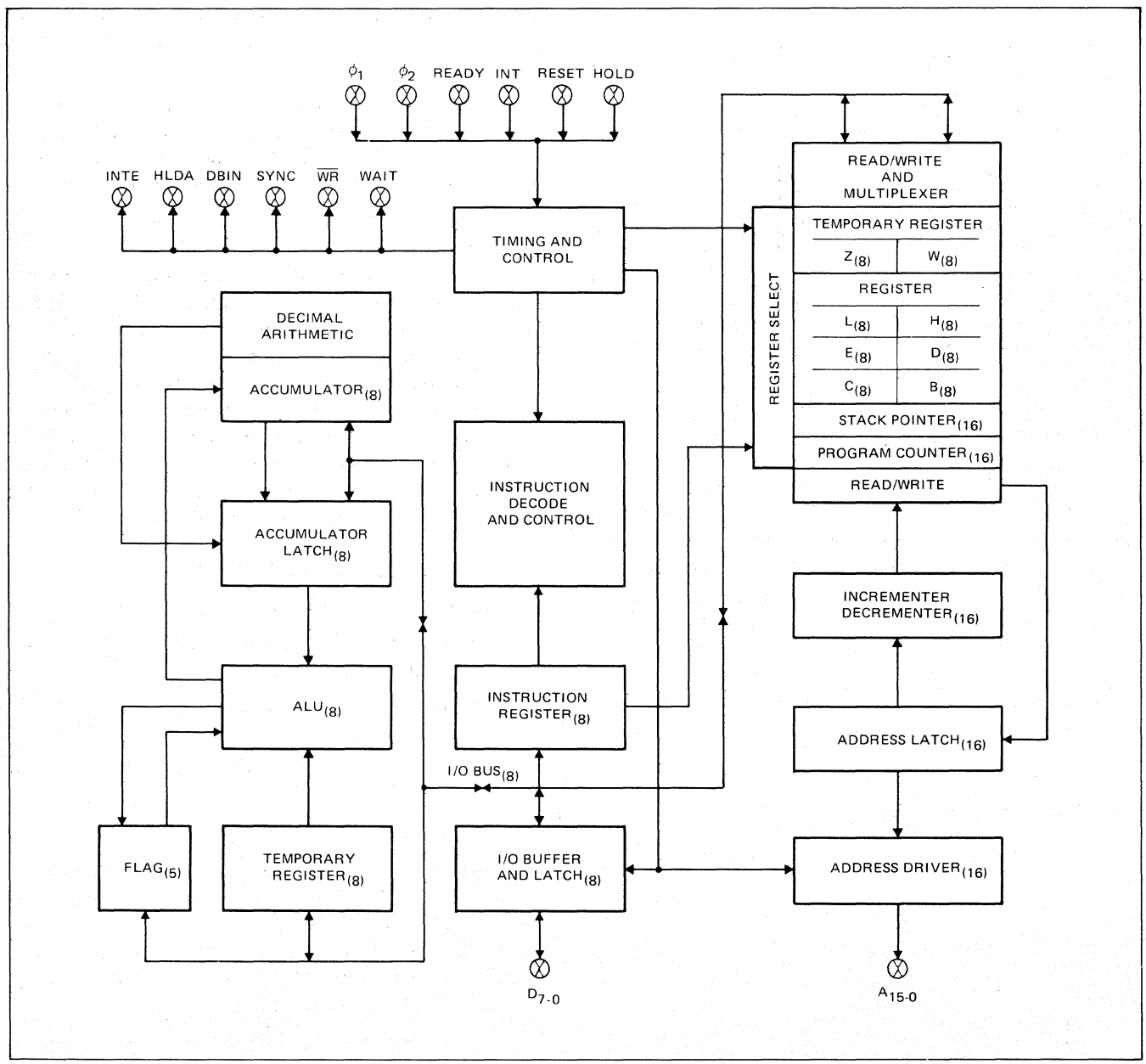


Figure 2-4. 8080 CPU Functional Block Diagram.

The Processor Cycle

The 8080 is driven by a two-phase clock oscillator, at a maximum frequency of 2.08 MHz. All processing activities are referred to the period of this clock. The two non-overlapping clock phases, labeled ϕ_1 and ϕ_2, are furnished by external circuitry. The ϕ_1 clock divides the processing cycle into *states*. A state is the smallest unit of processing activity (480 ns. when the processor is operating at maximum speed) and is defined as the interval between two successive positive-going transitions of the $\phi 1$ clock. Timing logic within the 8080 uses the clock inputs to produce a SYNC pulse, which identifies the first state of every machine cylcle. The SYNC pulse is triggered by the low to high transition of ϕ_2, as shown in Figure 2-5.

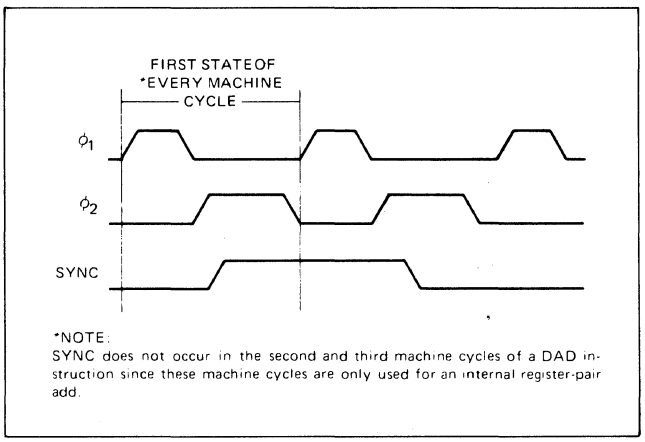


Figure 2-5. ϕ_1, ϕ_2 and Sync Timing.

An instruction cycle consists of two functional parts, the fetch and the execution. Each of these functional parts, in turn, consists of a number of machine cycles. During the fetch, a selected instruction (one, two or three bytes) is extracted from memory and deposited in the CPU's instruction register. During the execution part, the instruction is decoded and translated into specific processing activities. The fetch routine requires one machine cycle for each byte to be fetched. The duration of the executive portion of the instruction cycle depends upon the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to/from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register-pair add.

Every instruction cycle contains one, two, three, four, or five machine cycles. Each machine cycle, in turn, consists of three, four, or five states. A state is defined as a constant interval, equal in length to the period of the clock oscillator which drives the CPU (a phase). That is, a state is so defined in all but three cases. Exceptions to the rule are the WAIT state, the hold (HLDA) state, and the halt (HLTA) state, described later in this chapter. A moment's consideration will show that this is reasonable, since the WAIT, the HLDA, and the HLTA states depend upon external events and are by their nature of indeterminate length. Observe, however, that even these exceptional states must be synchronized with the pulses of the driving clock. Thus the durations of all states, including these, are integral multiples of the clock phase.

To summarize them, each clock *phase* marks a *state*; three to five *states* constitute a *machine cycle*; and one to five *machine cycles* comprise an *instruction cycle*. A full instruction cycle requires anywhere from four to eighteen phases for its completion (2.0 microseconds to 9.0 microseconds), depending on the kind of instruction involved.

MACHINE CYCLE IDENTIFICATION

With the exception of the DAD instruction there is just one consideration that determines how many machine cycles are required in any given instruction cycle: the number of times that the processor must reference a memory address, or an addressable peripheral device, in order to fetch and execute the instruction. Like many processors, the 8080 is so constructed that it can transmit only one address per machine cycle. Thus, if the fetching and execution of an instruction requires two memory references, then the instruction cycle associated with that instruction consists of two machine cycles. If five such references are called for, then the instruction cycle contains five machine cycles.

Every instruction cycle has at least one reference to memory, during which the instruction is fetched. A cycle must always have a fetch, even if the execution of the instruction requires no further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that, there are no fast rules. It depends on the kind of instruction.

Consider some examples. The add-register (ADD r) instruction is an instruction that requires only a single machine cycle (FETCH) for its completion. In this one-byte instruction, the contents of one of the CPU's six index registers is added to the pre-existing contents of the accumulator. Since all the information necessary to execute the command is contained in the eight bits of the instruction code, only one memory reference is necessary: that actually used to fetch the instruction. Three states are used to extract the instruction from memory, and one additional state is used to accomplish the desired addition. The entire instruction cycle thus requires only one machine cycle that consists of four states or four phases of the external clock (2 microseconds).

Suppose now, however, that we wish to add the contents of a specific memory location to the pre-existing contents of the accumulator (ADD M). Althoug this is quite similar in principle to the example just cited, several additional steps will be necessary. An extra machine cycle will be needed, in order to address the desired memory location.

The actual sequence is as follows. First the processor

extracts from memory the one-byte instruction word addressed by its program counter. This takes three states. The eight-bit instruction word obtained during the FETCH machine cycle is deposited in the CPU's instruction register and used to direct activities during the remainder of the instruction cycle. Next, the processor sends out as an address the contents of its H and L registers. The eight-bit data word returned during this MEMORY READ machine cycle is placed in a temporary register inside the 8080 CPU. By now three more clock periods (states) have elapsed. In the seventh and final state, the contents of the temporary register are added to those of the accumulator. Two machine cycles, consisting of seven states in all, complete "ADD M" instruction cycle.

At the opposite extreme is the save H and L registers (SHLD) instruction, which requires five machine cycles. During a "SHLD" instruction cycle, the contents of the processor's H and L index registers are deposited in two sequentially adjacent memory locations; the destination is indicated by two address bytes which are stored in the two memory locations immediately following the operation code byte. The following events occur:

1) A FETCH machine cycle, consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction indicated by its program counter. In the fourth state, the contents of the H and L registers are transferred to temporary registers within the chip, W and Z, respectively. Data previously held in the H and L registers is thus saved, thereby clearing H and L to receive incoming data.

2) A MEMORY READ machine cycle, consisting of three states. During this machine cycle, the byte indicated by the program counter is extracted from memory and placed in the processor's L register.

3) Another MEMORY READ machine cycle, consisting of three states, in which the byte indicated by the processor's program counter is deposited in the H register.

4) A MEMORY WRITE machine cycle, of four states, During the first three states, the contents of the Z register are transferred to the memory location pointed to by the present contents of the H and L registers. The state following the transfer is used to increment the H and L pointers, so that they indicate the next memory location to receive data.

5) A MEMORY WRITE machine cycle, of three states, in which the contents of the W register are transferred to the new memory location pointed to by the H and L registers.

The "SHLD" instruction cycle contains five machine cycles and takes 17 states to execute (8.5 microseconds).

Most instructions fall somewhere between the extremes typified by the "ADD r" and the XHTL instruction which requires 18 states (9.0 microseconds). The input (INP) and the output (OUT), for example, require three machine cycles: a FETCH, to obtain the instruction; a MEMORY READ, to obtain the address of the object peripheral; and an INPUT or an OUTPUT machine cycle, to complete the transfer.

There are nine types of machine cycles that may occur within an instruction cycle; though no one instruction cycle will consist of more than five machine cycles:

(a) FETCH

(b) MEMORY READ

(c) MEMORY WRITE

(d) STACK READ

(e) STACK WRITE

(f) INPUT

(g) OUTPUT

(h) INTERRUPT

(i) HALT

The machine cycles that actually do occur in a particular instruction cycle depend upon the kind of instruction, with the overriding stipulation that the first machine cycle in any instruction cycle is always a FETCH.

The processor identifies the machine cycle in progress, by transmitting an eight-bit status signal during the first state of every machine cycle. Updated status information is published on the 8080's data lines (D_0–D_7), during the SYNC interval. This data may be saved in latches, decoded, and used to develop control signals for external circuitry. Table 2-1 shows how the positive-true status information is distributed on the processor's data bus.

Status signals are provided principally for the control of external circuitry. Simplicity of interface, rather than machine identification, dictates the logical definition of individual status bits. You will therefore observe that certain processor machine cycles are uniquely identified by a single status bit, but that others are not. The M_1 status bit (D_5), for example, unambiguously identifies a FETCH machine cycle. A STACK READ, on the other hand, is indicated by the coincidence of STACK and MEMR signals. Machine cycle identification data can also be valuable in the test and de-bugging phases of system development.

8080 Status Bit Definitions

SYMBOLS	DATA BUS BIT	DEFINITION
HLTA	D_3	Acknowledge signal for HALT instruction.
INTA*	D_0	Acknowledge signal for INTERRUPT request. Signal should be used to gate a restart instruction onto the data bus when DBIN is active.
INP*	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active.
MEMR*	D_7	Designates that the data bus will be used for memory read data.
M_1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the Stack Pointer.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.
*These three status bits can be used to control the flow of data onto the 8080 data bus.		

Table 2-1.

STATE TRANSITION SEQUENCE

Every machine cycle within an instruction cycle consists of three to five active states (referred to as T1, T2, T3, T4, T5 or TW). The actual number of states depends upon the instruction being executed, and on the particular machine cycle within the greater instruction cycle. The state transition diagram in Figure 2-6 shows how the 8080 proceeds from state to state in the course of a machine cycle. The diagram also shows how the READY, HOLD, and INTERRUPT lines are sampled during the machine cycle, and how the conditions on these lines may modify the basic transition sequence. In the present discussion, we are concerned only with the basic sequence and with the READY function. HOLD and INTERRUPT functions will be discussed later.

The 8080 CPU does not indicate its internal state directly, by broadcasting a "state control" output during each state; instead, the 8080 supplies direct control ouputs (INTE, HLDA, DBIN, WR and WAIT) for use by external circuitry.

Recall that the 8080 passes through at least three states in every machine cycle, with each state defined by successive low-to-high transitions of the ϕ_1 clock. Figure 2-7 shows the timing relationships in a typical FETCH machine cycle. Events that occur in each state are referred to transitions of the ϕ_1 and ϕ_2 clock pulses.

The SYNC signal identifies the first state (T1) in every machine cycle. As shown in Figure 2-7, the SYNC signal is related to the leading edge of the ϕ_2 clock. There is a delay between the low-to-high transition of $\phi 2$ and the positive-going edge of the SYNC pulse. There also is a corresponding delay between the next ϕ_2 pulse and the falling edge of the SYNC signal. Status information is displayed on D_0 - D_7 during this same interval. Switching of the status signals is likewise controlled by ϕ_2.

The rising edge of ϕ_2 during the T1 also loads the processor's address lines (A_0–A_{15}). These lines become fully charged and remain charged until the first ϕ_2 pulse after state T3. This gives the processor ample time to read the data returned from memory.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This it does by pulling the processor's READY line low during state T2. As long as the READY line remains low, the processor will idle, giving the memory time to respond to the addressed data request. Refer to Figure 2-7.

The processor responds to a wait request by entering an alternative state (TW) at the end of T2, rather than proceeding directly to the T3 state. Entry into the T_W state is heralded by a WAIT signal from the processor, acknowledging the memory's request. A low-to-high transition on the WAIT line is triggered by the rising edge of the ϕ_1 clock.

A wait period may be of indefinite duration. The processor remains in the waiting condition until its READY line again goes high. The cycle may then proceed, beginning with the rising edge of the next ϕ_1 clock. A WAIT interval will therefore consist of an integral number of T_W states and will always be a multiple of the clock period.

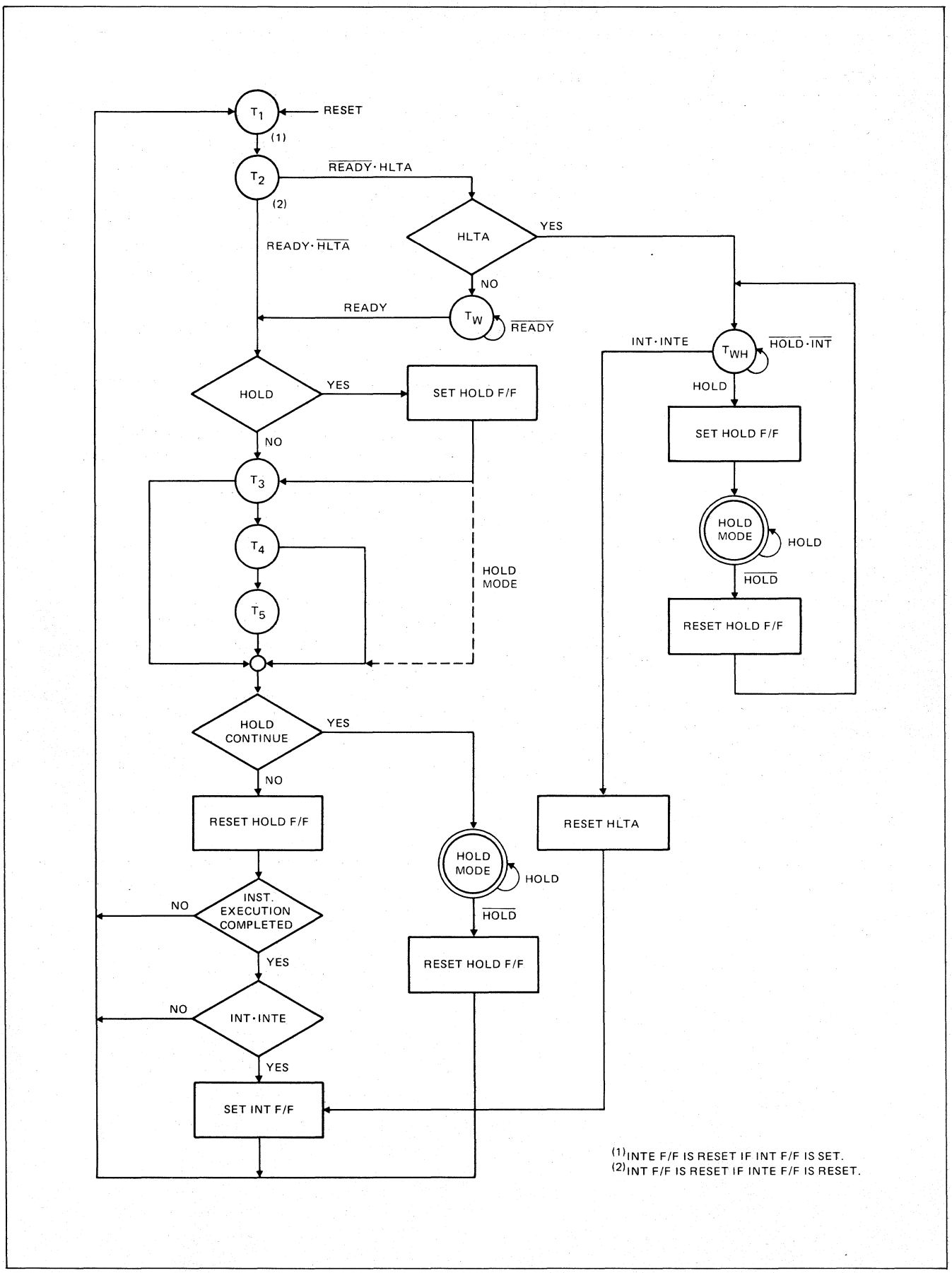


Figure 2-6. CPU State Transition Diagram.

The events that take place during the T3 state are determined by the kind of machine cycle in progress. In a FETCH machine cycle, the processor interprets the data on its main bus as an instruction. During a MEMORY READ or a STACK READ, signals on the same bus are interpreted as a data word. The processor itslef outputs data on this bus during a MEMORY WRITE machine cycle. And during I/O operations, the processor may either transmit or receive data, depending on whether an INPUT or an OUTPUT is involved.

During the input of data to the processor, the 8080 generates a DBIN signal which may be used externally to enable the transfer. Machine cycles in which DBIN is available include: FETCH, MEMORY INPUT, READ, STACK READ, AND INTERRUPT. DBIN is initiated by the rising edge of ϕ_2 during state T2 and terminated by the corresponding edge of ϕ_2 during T3. Any T_W states intervening between T2 and T3 will therefore prolong DBIN by one or more clock periods.

The 8080 CPU generates a $\overline{WR}$ output for the synchronization of external transfers, during those machine cycles in which the processor outputs data. These include MEMORY WRITE, STACK WRITE, and OUTPUT. The negative-going leading edge of $\overline{WR}$ is referred to the rising edge of the first ϕ_1 clock pulse following T2. $\overline{WR}$ remains low until re-triggered by the leading edge of ϕ_1 during state T1 of the next machine cycle. Note that any T_W states intervening between T2 and T3 of the OUTPUT machine cycle will necessarily prolong $\overline{WR}$, in much the same way that DBIN is affected during input operations.

All machine cycles of at least three states: T1, T2, and T3 as just described. If the processor has to wait for a response from the peripheral with which it is communicating, then the machine cycle may also contain one or more T_W states. During the three basic states, data is transferred to or from the processor.

After the T3 state, however, it vecomes difficult to generalize. T4 and T5 states are available, if the execution of a particular instruction requires them. But not all machine cycles make use of these states. It depends upon the kind of instruction being executed, and on the particular machine cycle within the instruction cycle. The processor will terminate any machine cycle as soon as its processing activities are completed, rather than proceeding mechanically through the T4 and T5 states every time. Thus the 8080 may exit a machine cycle following the T3, the T4, or the T5 state and proceed directly to the T1 state of the next machine cycle.

Table 2-2 lists the general activities associated with each state.

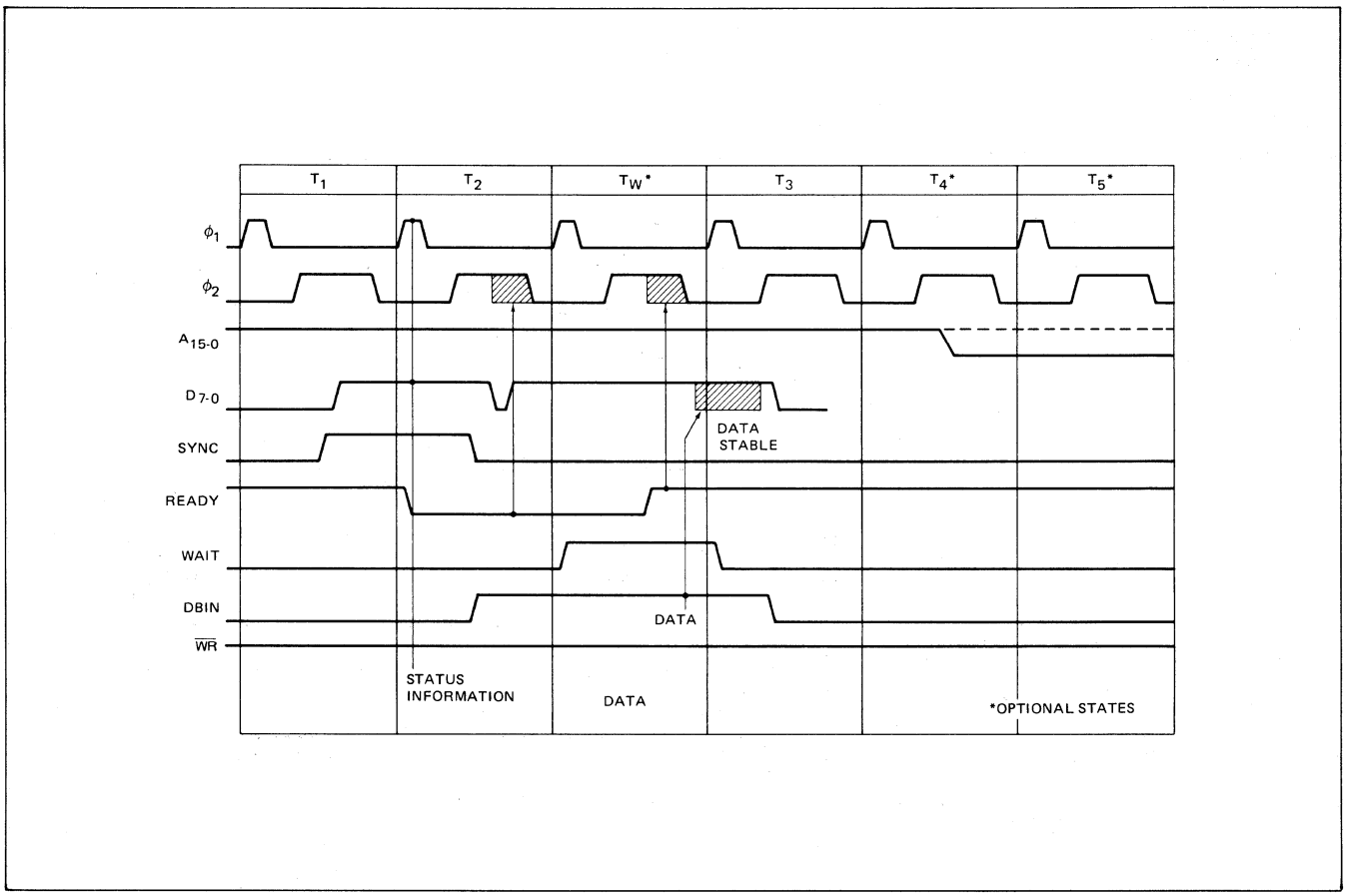


Figure 2-7. Typical Fetch Machine Cycle.

State Definitions

STATE	ASSOCIATED ACTIVITIES
T1	A memory address or I/O device number is placed on the Address Bus (A_{15-0}); status information is placed on Data Bus (D_{7-0}).
T2	The CPU samples the READY and HOLD inputs and checks for halt instruction.
TW (optional)	Processor enters wait state if READY is low or if HALT instruction has been executed.
T3	An instruction byte (FETCH cycle), data byte (MEMORY READ, STACK READ or INPUT cycle), or interrupt instruction (INTERRUPT cycle) is input to the CPU from the Data Bus; or a data byte (MEMORY WRITE, STACK WRITE or OUTPUT cycle) is output onto the data bus.
T4 T5 (optional)	States T4 and T5 are available if the execution of a particular instruction requires them; if not, the CPU may skip one or both of them. T4 and T5 are only used for internal processor operations.

Table 2-2.

Interrupt Sequences

The 8080 has the built-in capacity to handle external interrupt requests. A peripheral device can initiate an interrupt simply by pulling the processor's interrupt (INT) line high.

The interrupt (INT) input is asynchronous, and a request may therefore originate at any time during any instruction cycle. Internal logic re-clocks the external request, so that a proper correspondence with the driving clock is established. As Figure 2-8 shows, an interrupt request (INT) arriving during the time that the interrupt enable line (INTE) is high, acts in coincidence with the ϕ_2 clock to set the internal interrupt latch. *This event takes place during the last state of the instruction cycle in which the request occurs, thus ensuring that any instruction in progress is first completed.*

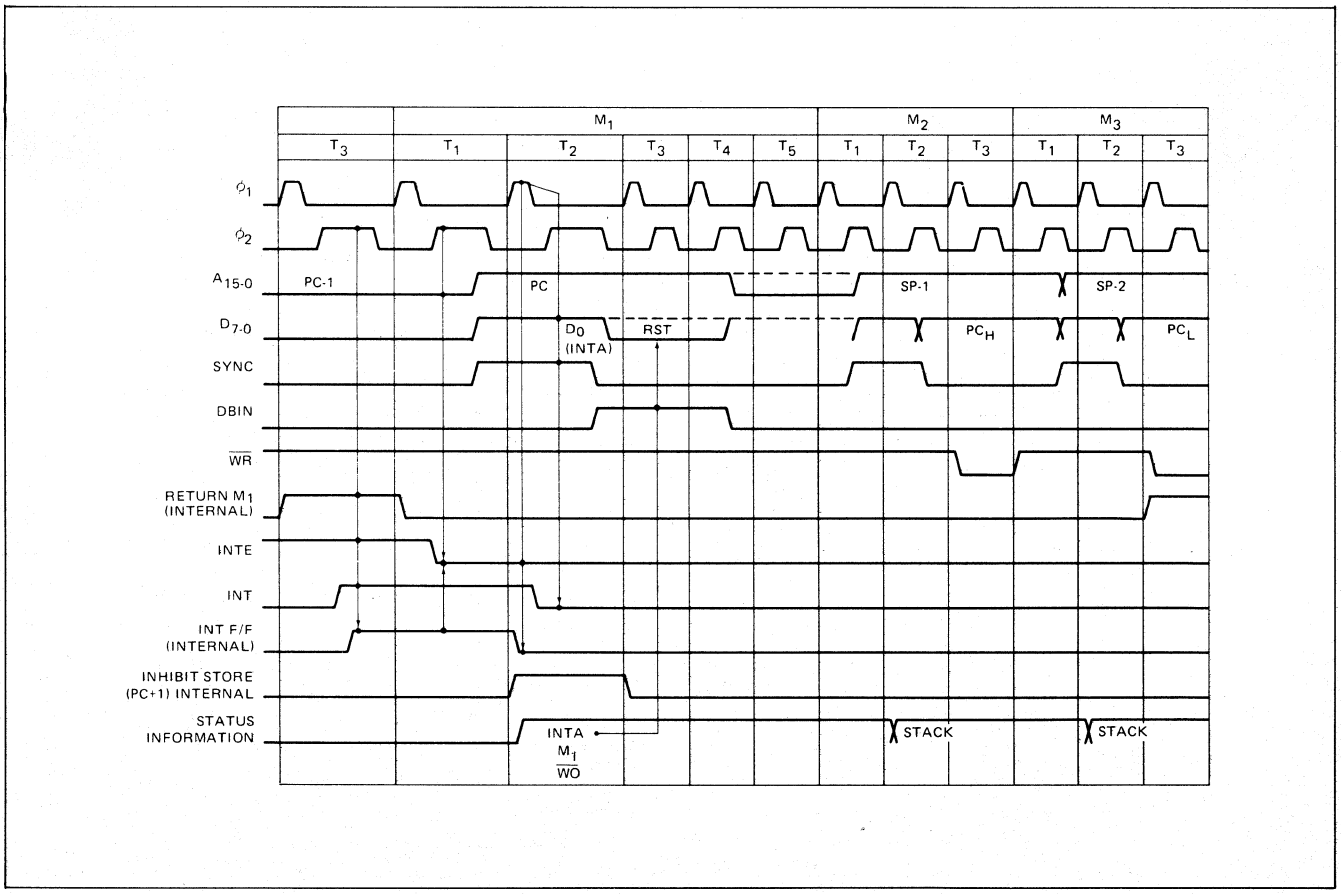


Figure 2-8. INTERRUPT Timing

The INTERRUPT machine cycle which follows the arrival of an enabled interrupt request resembles an ordinary FETCH machine cycle in most respects. The M_1 status bit is published as usual during the SYNC interval. it is accompanied, however, by an INTA status bit (D_0) which acknowledges the external request. The contents of the program counter are latched onto the CPU's address lines during T1, but the counter itself is not incremented during the INTERRUPT machine cycle, as it otherwise would be. In this way, the pre-interrupt status of the program counter is preserved, so that data in the counter may be saved in the stack. This in turn permits an orderly return to the interrupted program after the interrupt request has been processed.

The interrupt cycle is otherwise indistinguishable from an ordinary FETCH machine cycle. The processor itself takes no further special action. It is the responsibility of the peripheral logic to see that an eight-bit interrupt instruction is "jammed" onto the processor's data bus at T3. In a typical system, this means that the data in bus from memory must be temporarily disconnected from the processor's main data bus, so that the interrupting device can command the main bus without interference.

The processor will treat the code placed on the main bus at T3 just like any other fetched instruction. Thus, any of the processor instructions may be inserted during an interrupt. If the code is the first byte of a multiple word instruction, however, a special problem is encountered. The processor will perform succeeding MEMORY READ machine cycles, fully expecting that the proper information will be on its bus at the proper time. But the program counter will advance after the first byte. Because the program counter advances a return to the interrupted, instruction is not possible. For this reason, one-byte instructions are preferable in most systems for use with interrupts.

The 8080's instruction set provides a special one-byte call which facilitates the processing of interrupts (the ordinary program call takes thee bytes). This is the restart instruction (RST). A variable three-bit field embedded in the eight-bit field of the RST enables the interrupting device to direct a jump to one of eight fixed memory locations. The decimal addresses of these dedicated locations are: 0, 8, 16, 24, 32, 40, 48, and 56. Any of these addresses may be used to store the first instruction(s) of a routine designed to service the requirements of an interrupting device.

Hold Sequences

The 8080 CPU contains provisions which enable direct memory access (DMA) operations. By applying a HOLD to the appropriate control pin on the processor, an external device can cause the CPU to suspend its normal operations and relinquish control of the address and data buses. The processor responds to a request of this kind by floating its address and data outputs, so that these exhibit a high impedance to other devices sharing the buses. At the same time, the processor acknowledges the HOLD by placing a high on its HLDA output pin. During an acknowledged HOLD, the address and data buses are under control of the peripheral which originated the request, enabling it to conduct memory transfers without processor intervention.

Unlike the interrupt, the HOLD input must be synchronized with the driving clock. A HOLD signal should coincide with the ϕ_1 clock pulse, and external re-clocking

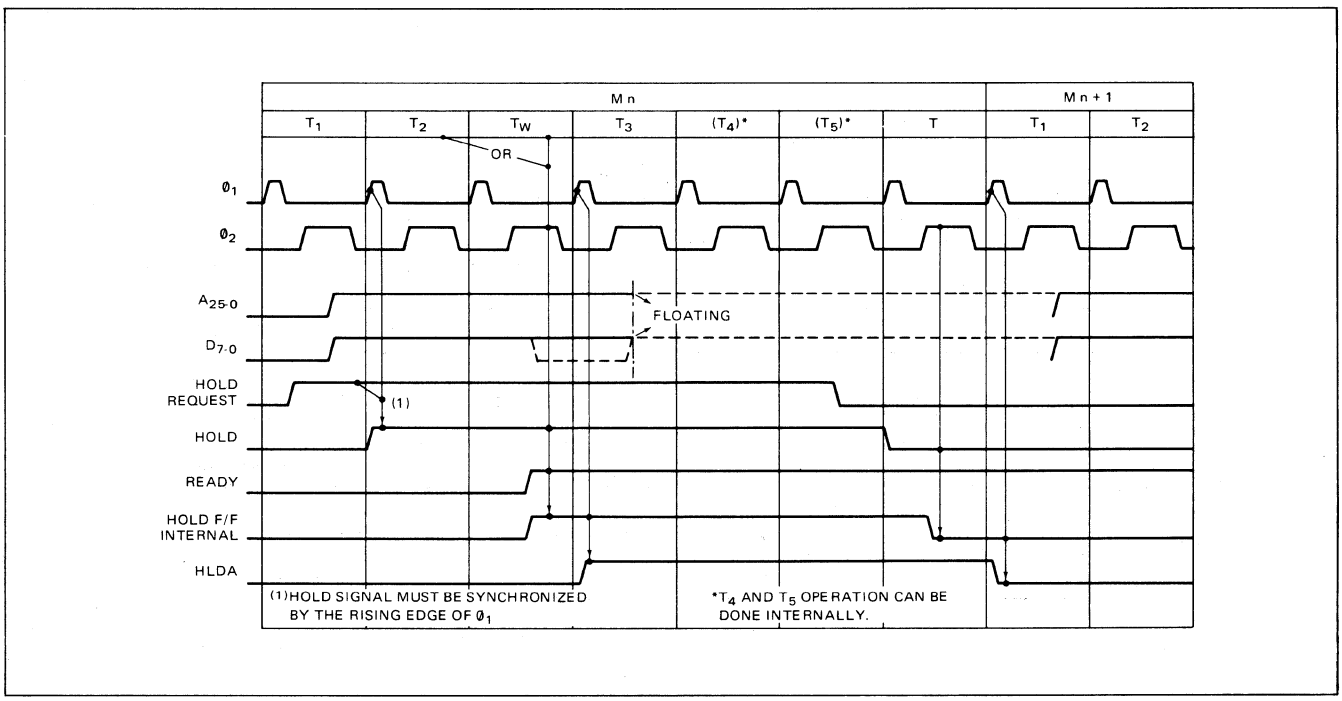


Figure 2-9. Hold Operation (Read Mode).

logic must therefore be provided. In a typical system, an asynchronous HOLD REQUEST will be registered by the rising edge of the ϕ_1 clock impulse, and the resulting synchronized output will drive the CPU's HOLD line. A coincidence of the READY, the HOLD, and the ϕ_2 clock sets the internal hold latch. Setting the latch enables the subsequent rising edge of the ϕ_1 clock pulse to trigger the HLDA output.

Acknowledgement of the HOLD REQUEST precedes slightly the actual floating of the processor's address and data lines. The processor acknowledges a HOLD at the beginning of T3, if a read or an input machine cycle is in progress (see Figure 2-9). Otherwise, acknowledgement is deferred until the beginning of T4 (see Figure 2-10). In both cases, however, the HLDA goes high within a brief delay of the rising edge of the selected ϕ_1 clock pulse. Address and data lines are floated within a brief delay after the rising edge of the next ϕ_2 clock pulse.

To all outward appearances, the preocessor has suspended its operations once the address and data busses are floated. Internally, however, certain functions may continue. If a HOLD REQUEST is acknowledged at T3, and if the processor is in the middle of a machine cycle which requires four or more states to complete, the CPU proceeds through T4 and T5 before coming to a rest. Not until the end of the machine cycle is reached will processing activities be completely stalled. Internal processing is thus permitted to overlap the external DMA transfer, improving both the efficiency and the speed of the entire system.

The processor exits the holding state through a sequence similar to that by which it entered. A HOLD REQUEST is terminated asynchronously, when the external device has completed its data transfer. Re-clocking logic registers this change at the beginning of the next state (rising edge of ϕ_1). The internal hold latch is reset by the subsequent leading edge of the ϕ_2 clock pulse, and the HLDA output returns to a low level following the leading edge of the next ϕ_1. Normal processing resumes with the machine cycle in progress, or with T1 of the next machine cycle, depending on whether the HOLD REQUEST is brief or extended.

Halt Sequences

When a halt instruction (HLT) is executed, the CPU enters the halt state (T_{WH}) after state T2 of the next machine cycle, as shown in Figure 2-11. There are only three ways in which the 8080 can exit the halt state:

- A high on the RESET line will always reset the 8080 to state T1; RESET also clears the program counter and sets the instruction register to zero.
- A HOLD input will cause the 8080 to enter the hold state, as previously described. When the HOLD line goes low, the 8080 re-enters the halt state on the rising edge of the next ϕ_1 clock pulse.
- An interrupt (i.e., INT goes high while INTE is enabled) will cause the 8080 to exit the halt state and enter state T1 on the rising edge of the next ϕ_1 clock pulse. NOTE: The interrupt enable (INTE) flag must be set when the halt state is entered; otherwise, the 8080 will only be able to exit via a RESET signal.

Start-Up of the 8080 CPU

When power is applied initially to the 8080, the processor begins working immediately. The contents of its

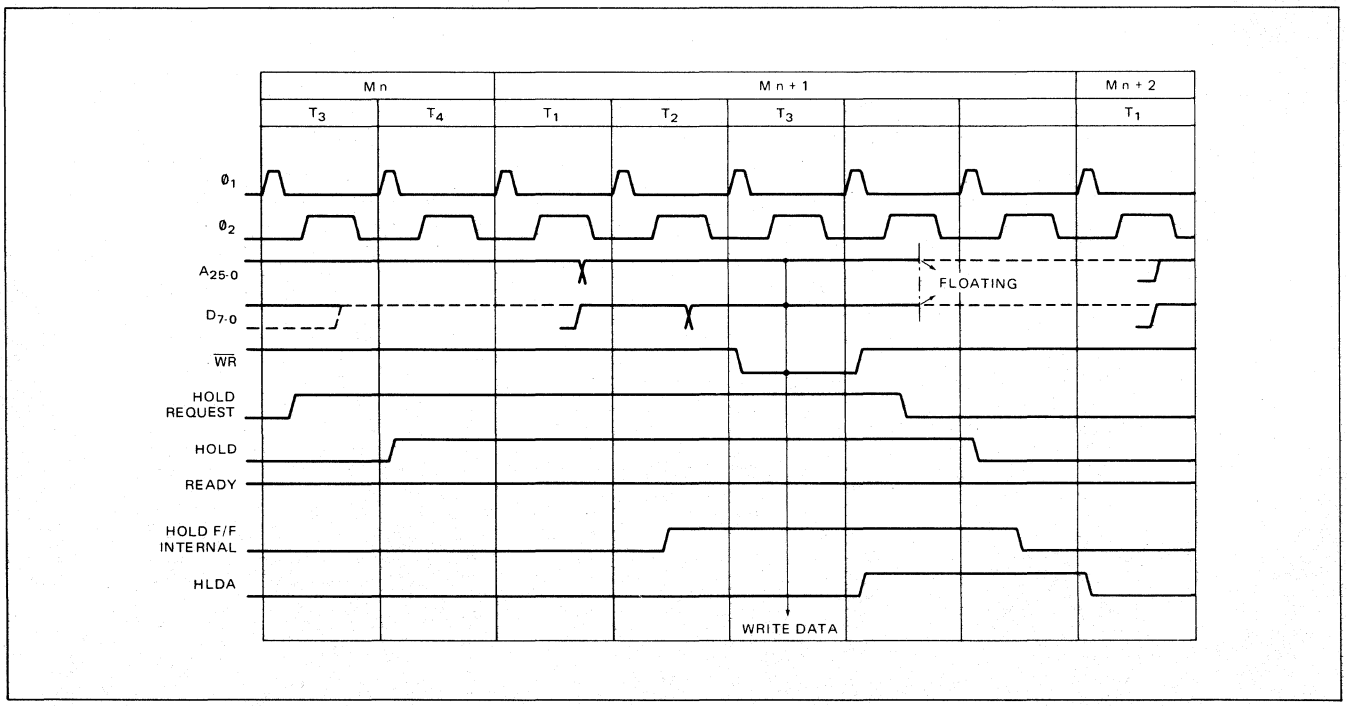


Figure 2-10. Hold Operation (Write Mode).

program counter, stack pointer, and the other working registers are naturally subject to random factors and cannot be specified. For this reason, it will be desirable in many situations to begin the power-up sequence with an automatic forced RESET.

An external RESET signal of 1.5 microseconds' duration (minimum) restores the processor's internal program counter and instruction regester to zero. Program execution thus begins with memory location zero, following a RESET. Note, however, that the RESET has no effect on status flags, or on any of the processor's working registers (accumulator, indices, or stack pointer). The contents of these registers remain indeterminate, until initialized explicitly by the program.

Peripheral Logic

In this section, we describe the peripheral logic on the imm8-83 Central Processor Module, the logic which supports the activities of the 8080 CPU. We begin by explaining the timing logic, since all the operations of the module are ultimately referred to signals generated in that section. Then we give descriptive examples of module operations, showing how the peripheral logic extends the basic capabilities of the 8080 processor.

Timing Logic

The timing logic consists of a crystal-controlled clock oscillator, a counter, level shifting provisions, and miscellaneous counting and gating circuits. These are shown on the module schematic, Figure 2-16.

The clock oscillator furnishes a 32 MHz signal to the input of the counting section, which uses it to develop the ϕ_1 and ϕ_2 clock signals used to generate the remaining timing outputs. The clock oscillator consists of components shown in the upper central portion of the module schematic.

A 32 MHz quartz crystal, operating in the series-resonant mode, is the basic frequency reference. The crystal acts as a bandpass filter at the desired frequency. It thus permits a portion of the signal developed across the capacative divider in the transistor's collector circuit to reach the emitter, in proper phase to sustain oscillation. The output from the oscillator stage is coupled to a second stage, biased to operated as an overdriven amplifier, and the shaped output of the second is used to drive the synchronous counter chain.

Four 74S114 high-speed J-K flip-flops constitute the clock counter. This is a synchronous configuration, with the steering function obtained through the use of external coincidence gates. A slight variation on conventional practice produces a fourth stage output which is "displaced" with respect to the outputs of the first three stages, by one full period of the driving clock. In all other respects, however, the counter resembles the familiar modulo-16 synchronous counters in common usage. Idealized output waveforms are shown in Figure 2-12.

The 2 MHz output of the fourth counting stage becomes the ϕ_2 clock signal. Coincidence in the outputs of the third and fourth stages generates the ϕ_1 clock. As Figure 2-12 shows, this produces two non-overlapping clock signals, with characteristic pulse widths of 125 and 250 nanoseconds and separation intervals of approximately 32 and 94 nanoseconds.

The ϕ_1 and ϕ_2 clock phases are applied to the inputs of an MH0026 level shifter and used to drive the 8080's clock inputs. Timing logic on the processor produces a SYNC output, derived from ϕ_2. Then SYNC and clock signals are fed to the gating logic.

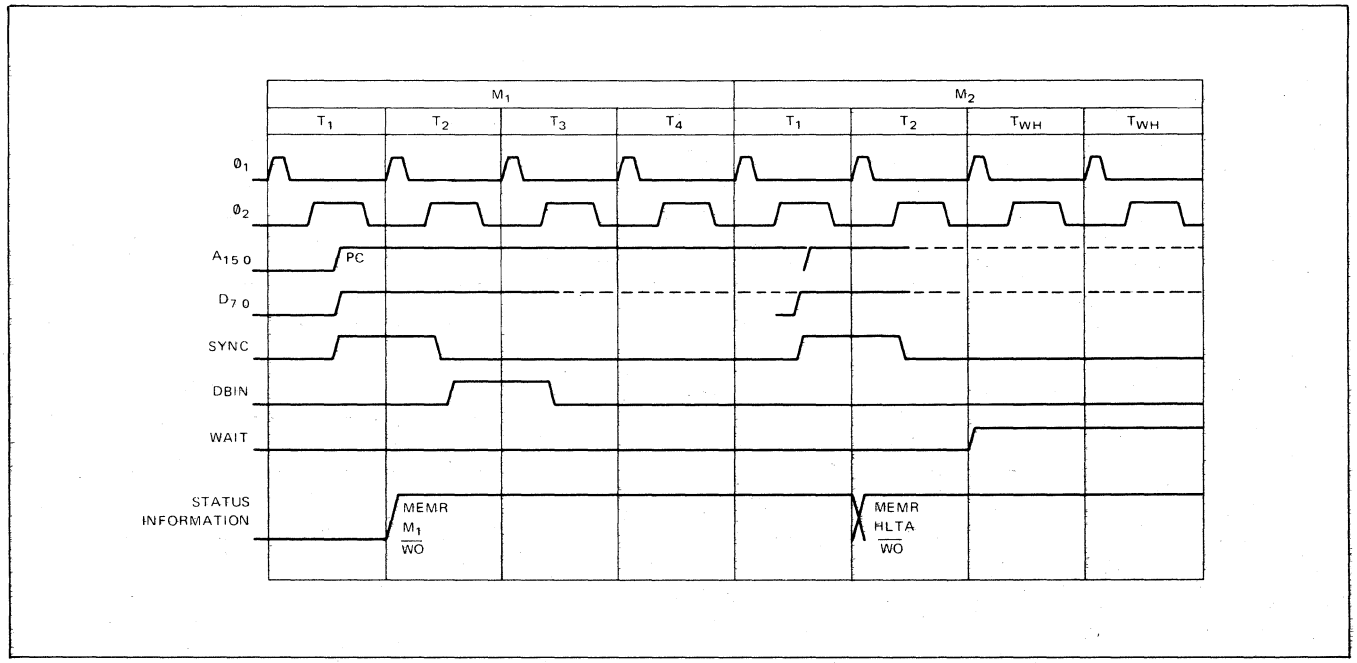


Figure 2-11. Halt Timing.

In the gating section the SYNC and the ϕ_1 clock are combined in a 74H00 NAND-gate section. The coincidence of these two signals produces the $\overline{\text{CLKA}}$ output. $\overline{\text{CLKA}}$ is used as a strobe on the module, to register the status information sent out at the beginning of each processor machine cycle. After passing through cascaded buffer sections, the SYNC signal becomes CLKB. CLKB is available at the PC edge connector, for use by the INTELLEC 8's Front Panel Controller. A 7493 binary counter in the gating logic derives a one-eighth submultiple of the ϕ_1 clock pulse, and this too is made available to logic on the INTELLEC 8's Front Panel Controller. This pulse, known as the SYNCA, is used to synchronize service requests originating at the Console and Display Panel. Figure 2-13 shows the timing of these signals.

Instruction Fetch

Refer to the schematic for the Central Processor Module, Figure 2-16. An instruction fetch machine cycle (FETCH) is the first part of every instruction cycle. The events that take place during an instruction fetch are as follows.

During the T1 state, the processor transmits the contents of its internal program counter to memory, via the 8080's sixteen address lines. Assuming that no holds are in progress, the address data passes through the sixteen enabled tri-state buffers of the address buffer section and is presented to external memory. Data placed on the address bus remains stable until the T4 processing state.

Status information is also broadcast during the T1 interval, on the processor's eight data lines. Again assuming the absence of a hold, data on this bus passes through the eight parallel tri-state buffers of the data buffer section and is forwarded to the status latches. Eight Intel 3404 Hex Inverting Latch sections are used to register the status information. The strobing input to these latches is the $\overline{\text{CLKA}}$ signal from the timing section. Status information reflecting the machine cycle in progress is thus recorded at the beginning of every processor machine cycle. During a FETCH machine cycle, the following status bits are produced: M_1, MEMR, and RI/$\overline{\text{WO}}$.

Status information is cleared from the processor's data bus during the T2 state, in preparation for the data to be returned form memory. Such data must be present and stable at least 20 nanoseconds prior to the end of the T2 state. Neither the PROM Memory Module nor the RAM Memory Module used with the INTELLEC 8/MOD 80 is capable of responding that fast, and the automatic wait logic comes into play accordingly.

Refer to the module schematic, Figure 2-16. The logic used to generate the appropriate wait request consists of two 7474 latch sections and two 7400 NAND-gate sections, shown on sheet 2.

Consider first the case where the PROM module is selected. The PROM MOD ENABLE line (pin #97) will be high, and the RAM MOD ENABLE line (pin #93) will be low. At the beginning of a machine cycle, the SYNC pulse from the timing generator is gated through the 8-9-10 section of the quad NAND-gate A18 to reset both of the 7474 latch sections. The $\overline{\text{Q}}$ outputs of both sections will thus be high, following the rising edge of the T1-ϕ_2 clock pulse. The high at A1-8 is applied to A27-2, as shown on sheet 1, re-clocked by ϕ_1, in a 7474 latch section (A5), and applied as a low to the CPU's READY pin. This indicates a wait request to the 8080, and the CPU responds by entering the T_W state instead of proceeding directly to the T3 state.

Referring back to sheet 2, observe that the D input of the upper latch section is connected through a pull-up to

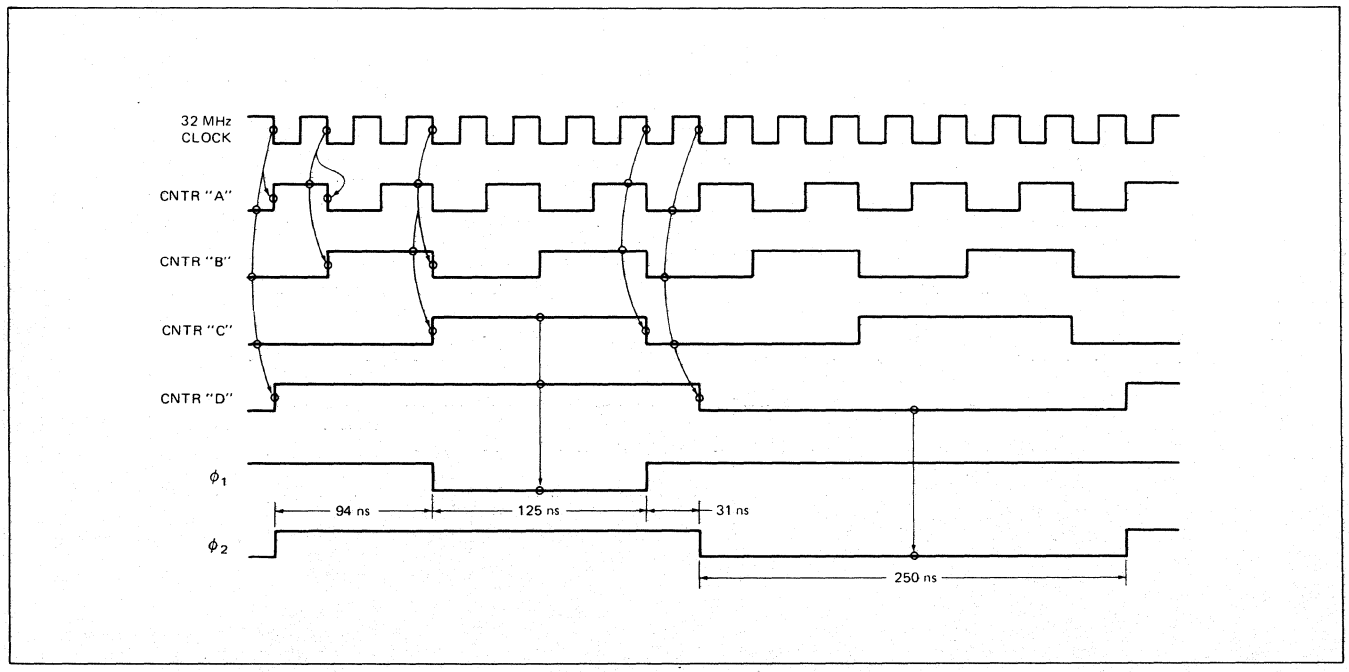


Figure 2-12. Oscillator-Counter Timing.

V_{CC}. An inverted ϕ_2 clock is directed to the clock input of this latch, which is set accordingly by the trailing edge of ϕ_2. Detailed timing is shown in Figure 2-14.

The resulting low at AI-6 is forwarded to A18-1, which applies a high to the D input of the lower latch section. With its D input now high, the latch is set by the trailing edge of the next $\overline{\phi_2}$ pulse. The latch's $\overline{Q}$ output goes low, is re-clocked by ϕ_1 as shown on sheet 1, and is presented as a READY indication to the processor, with the result that the WAIT state is terminated with the next ϕ_1 clock pulse. By referring to the timing diagram in Figure 2-14, we can see that two clock periods have elapsed between the processor's exit from T2 and its entry into T3. This additional one microsecond interval gives the PROM Memory Module sufficient time to respond to the address from the Central Processor Module, and the machine cycle may proceed.

Consider next the sequence of events that takes place when the RAM module is selected. In this case, both the RAM MOD ENABLE and the PROM MOD ENABLE lines shown in Figure 2-16 will be high. The SYNC pulse will again reset the two 7474 latch sections shown on sheet 2. And again, the $\overline{Q}$ output of the lower latch will be re-clocked by ϕ_1 and forwarded to the processor as a low signal level, indicating a wait request. Timing of this sequence is shown in Figure 2-15.

In the present case, however, the high on the RAM MOD ENABLE line causes the A18:1-2-3 NAND-gate section to forward a high to the D input of the lower 7474 latch section. The trailing edge of the ϕ_2 pulse occurring during T2 will set this latch, and the wait request to the processor will be terminated by the next ϕ_1 clock pulse. With reference to the timing diagram, we can see that the resulting WAIT interval is only half that generated when the PROM module was selected. The RAM module is inherently faster than the PROM module, and the 0.5 microsecond waiting period enables it to respond to the imm8-83's addressed request.

The Central Processor Module also contains synchronization provisions for other kinds of memory. Two possibilities are envisioned: those where the memory's access time is even greater than that specified for the PROM Memory Module. Where the imm8-83 is used in conjunction with other memories, selection logic will clamp the PROM MOD ENABLE line low. With this condition prevailing, the A18:8-9-10 NAND-gate section will be inhibited, and the SYNC pulse from the timing logic will be unable to reset the two 7474 wait latches as previously described. A1-8 will be high continuously, and no wait request will reach the processor by that route. In this case, it is the responsibility of the selected memory module or controller to generate a $\overline{\text{WAIT REQUEST}}$ as necessary.

Memories with access times under 500 nanoseconds need no synchronization but those whose access times exceed this figure will require it. A wait is initiated from outside the CPU module by clamping the $\overline{\text{WAIT REQUEST}}$ line low. Figure 2-16 (sheet 1) shows the wait logic. $\overline{\text{WAIT REQUEST}}$ enter the module at pin #21. They are inverted, gated through the 1-2-3 section of A-27, and applied to the D input (pin #2) of A5. A5 is the same 7474 latch section used to re-clock internally generated wait requests. It receives a ϕ_1 pulse at its clock input and produces a

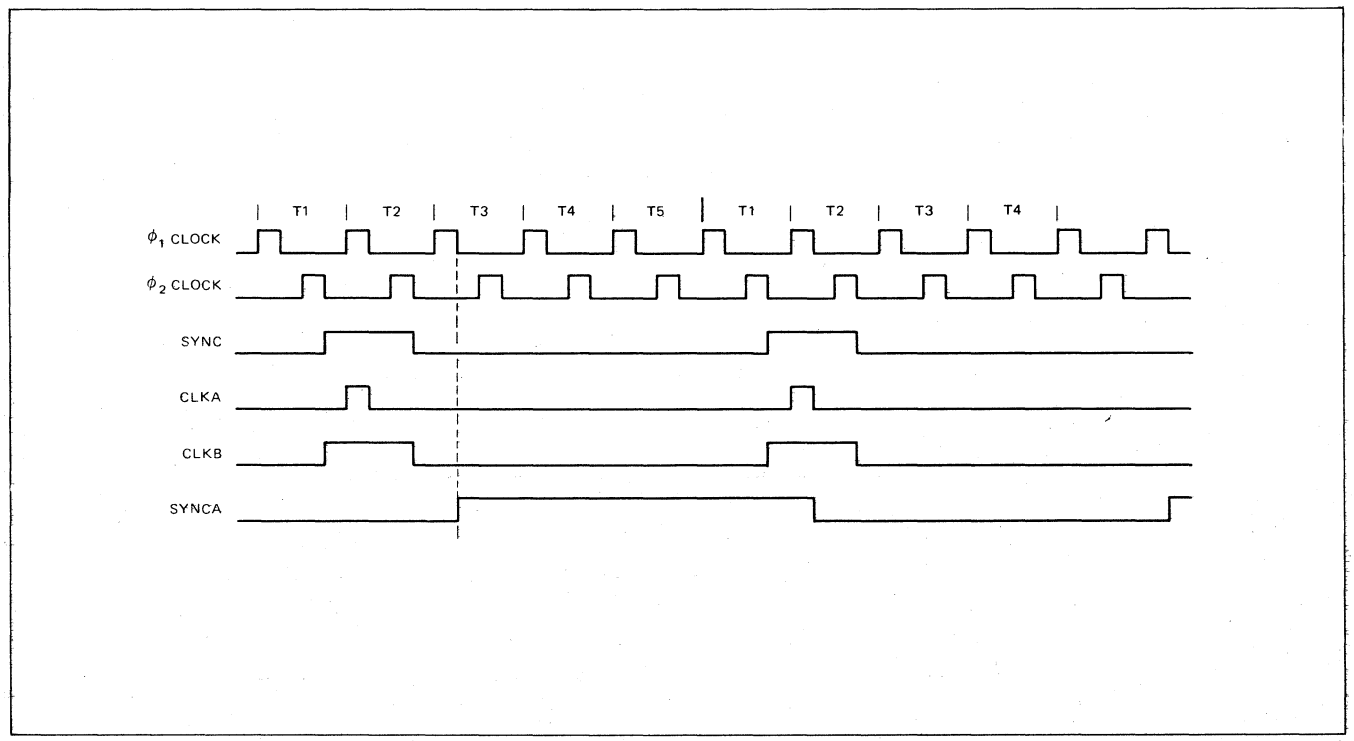


Figure 2-13. Timing Generator

synchronized wait request at its output. The low which is applied to the processor's READY input causes the processor to idle, until such time as the $\overline{\text{WAIT REQUEST}}$ is removed. Note that the re-clocking mechanism requires that a request be terminated by the beginning of the T_W phase, in order to guarantee an exit from the T_W state at the beginning of the next clock phase.

When synchronization has been achieved, the FETCH machine cycle proceeds. It becomes the responsibility of the multiplexing logic to select the memory's reply and forward it to the processor. The INP and the INTA status bits from the status latch section, and the $\overline{\text{DBIN}}$ from the 8080 COU, enable the multiplexing logic to perform this function.

Refer to Figure 2-16. The multiplexing logic is shown at the left of sheet 1, and consists of two pairs of cascaded 8-to-4 line multiplexers type 74S257 and 74S157. These are labelled A22, A23, A30 and A31 on the schematic. Inputs to A30 and A31 consist of the eight-line data in bus from memory, and the eight-line data in bus from the input peripherals. The multiplexer discriminates between these two eight-line inputs on the basis of a control signal furnished to pin #1 of both units. A high level here causes the multiplexers to select data from memory, while a low causes them to select data from the other input devices.

The selected eight-line output of the first multiplexing stage is forwarded to inputs on a similar second stage. The other eight-line input to the second stage comes from the module's interrupt instruction port. Like the first stage, the second selects one of the two inputs, on the basis of the signal at the pin #1 inputs of the two multiplexers. A high selects the output of the first multiplexing stage, while a low selects the interrupt instruction port.

The output of the second multiplexing stage is connected directly to the processor's eight-line data bus (D_0-D_7). Observe, however, that the second stage of the multiplexer requires an explicit enabling input. In the absence of a low on pin #15 of the two units, the outputs of the multiplexers are in a floating, high-impedance state. The ability to entirely disable all inputs to the main bus enables the bus to be used for bilateral exchanges of data. During output, the input bus is disabled, to prevent the conflicts that could arise if the processor and one or more input devices were competing simultaneously for the use of the bus.

To understand the input gating mechanism during a FETCH machine cycle, refer to Figure 2-16 (sheet 2). The first stage of the input multiplexer is controlled by the INP status latch. During machine cycles in which memory is referenced, the output of this latch is high. And assuming that no $\overline{\text{IN JAM ENABLE}}$ is present (this external function occurs only during the artificial input mode, "sense"), high is therefore produced at the output of A32-8. This high causes the first stages of the multiplexer to select and forward data from memory to the input of the second multiplexer stage.

The second stage of the multiplexer is controlled by the INTA status latch, shown on sheet 1 of the module schematic. This status bit is low only during an INTERRUPT sub-cycle. During a FETCH, the INTA latch forwards a high to the control input of the second

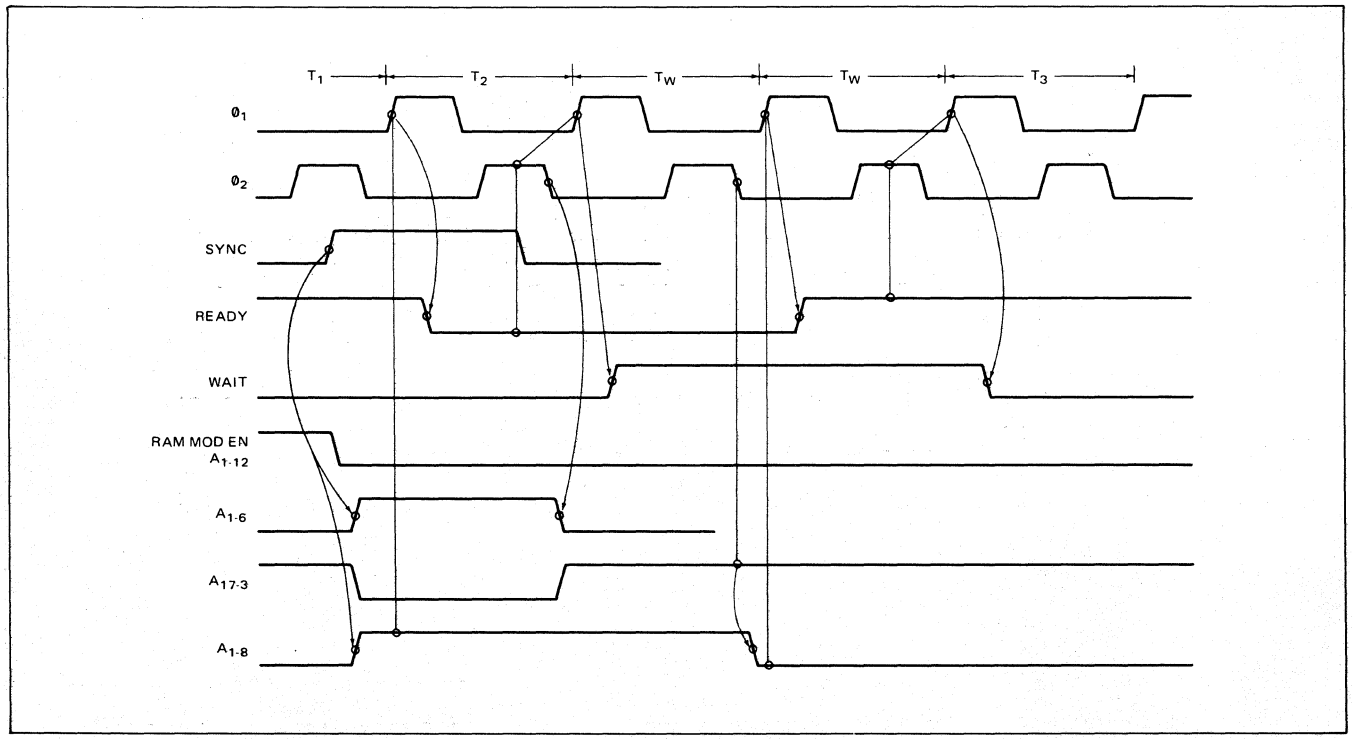


Figure 2-14. PROM Memory Synchronization Timing.

multiplexer stage, causing it to select and forward the data from the first stage to the processor.

When the processor is ready to receive data, it generates a $\overline{\text{DBIN}}$ output signal. The rising edge of this signal coincides with the rising edge of the T2-ϕ_2 clock, and $\overline{\text{DBIN}}$ remains active until reset by the leading edge of the T3-ϕ_2 clock. This signal is applied to a 7402 NOR-gate of the multiplexer. Thus the previously addressed instruction is finally gated through from memory, and stored in the processor's instruction register.

Some instructions will cause the processor to enter FETCH-T4 and FETCH-T5 states. But the activity of the peripheral logic is completed with the return of the instruction from memory. Where T4 and T5 states are used, they are reserved solely for internal processor functions.

The fetched instruction may be executed immediately, completing both the FETCH machine cycle and the instruction cycle. Or it may cause the processor to execute one or more additional memory references. These are described in the next section.

Memory Reference Operations (Memory Read and Memory Write)

Every operation that the CPU performs is preceded by a FETCH machine cycle such as that just described. In the case of certain instructions, it may be necessary to reference memory one or more additional times in order to completely execute the command.

Instructions that reference memory in the course of their execution do so in a manner very similar to that used to fetch instructions. One major difference is that the address of the referenced location may be furnished by the processor's internal H and L pointer registers, rather than by the program counter. But during MEMORY READ and MEMORY WRITE machine cycles, the addressing, multiplexing and gating functions are handled in much the same way as they are for an instruction fetch.

As far as the peripheral logic is concerned, there is one important difference. The processor generates a DBIN signal during those machine cycles in which it intends to input data from memory. During those machine cylces in which the processor outputs data to memory, it generates a $\overline{\text{WR}}$ output signal to cue the transfer.

A MEMORY READ machine cycle is accompanied by a DBIN signal. The peripheral logic thus handles it in exactly the same way as a FETCH. A MEMORY WRITE machine cycle is slightly different. The absence of DBIN inhibits the input multiplexer, and logic on the imm8-83 conditions the processor's $\overline{\text{WR}}$ output, to produce a $\overline{\text{WRITE}}$ command for the control of memory.

Refer to sheet 2 of the module schematic. As shown, the processor's $\overline{\text{WR}}$ output is coupled through an inverter section to pin #4 of A2. Here it is ANDed with the negative output of the OUT status latch, to produce a signal which is low only during the $\overline{\text{WR}}$ portion of a MEMORY WRITE machine cycle. This output is buffered in a tri-state section, to become the $\overline{\text{WRITE}}$ output to memory.

In the course of a MEMORY WRITE machine cycle, address and status information are transmitted just as they are in a MEMORY READ. During the T2 state, however, the processor places the contents of its accumulator on the main data bus where it is forwarded to the memory's data inputs. The $\overline{\text{WRITE}}$ output synchronizes the transfer, completing the write machine cycle.

I/O Operations

All input and output operations require three machine cycles: a FETCH to obtain the instruction, a

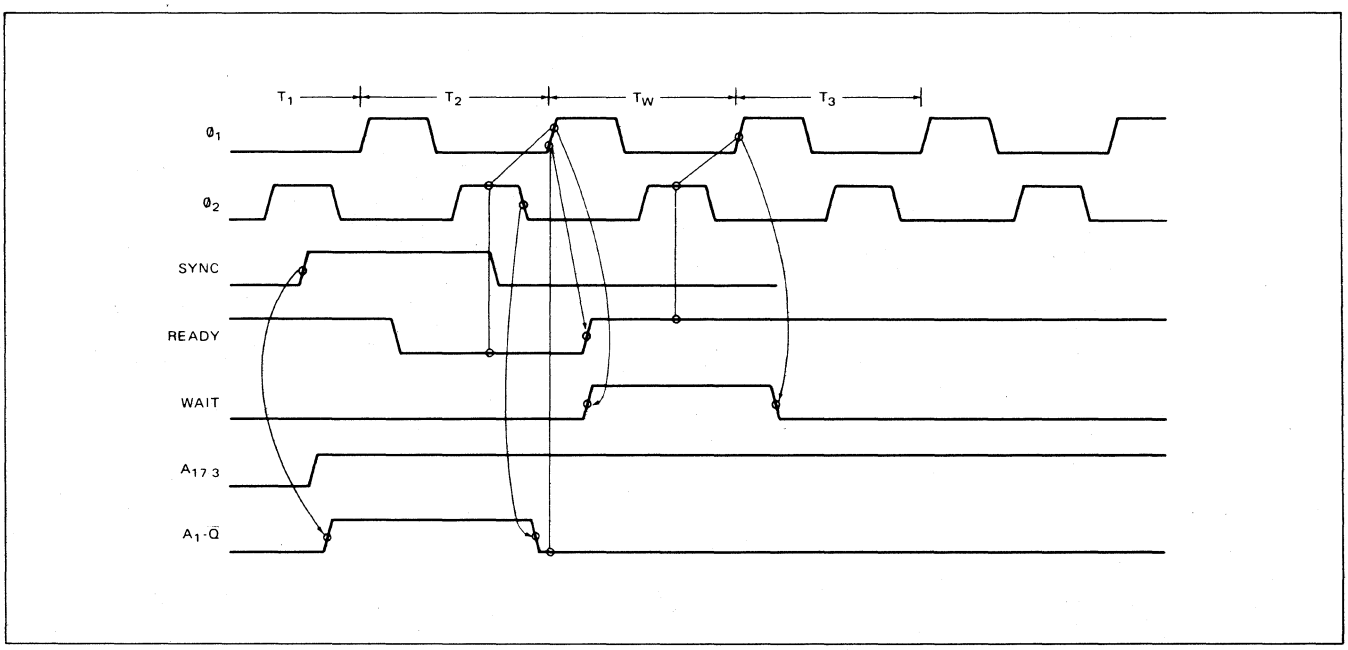


Figure 2-15. RAM Memory Synchronization Timing.

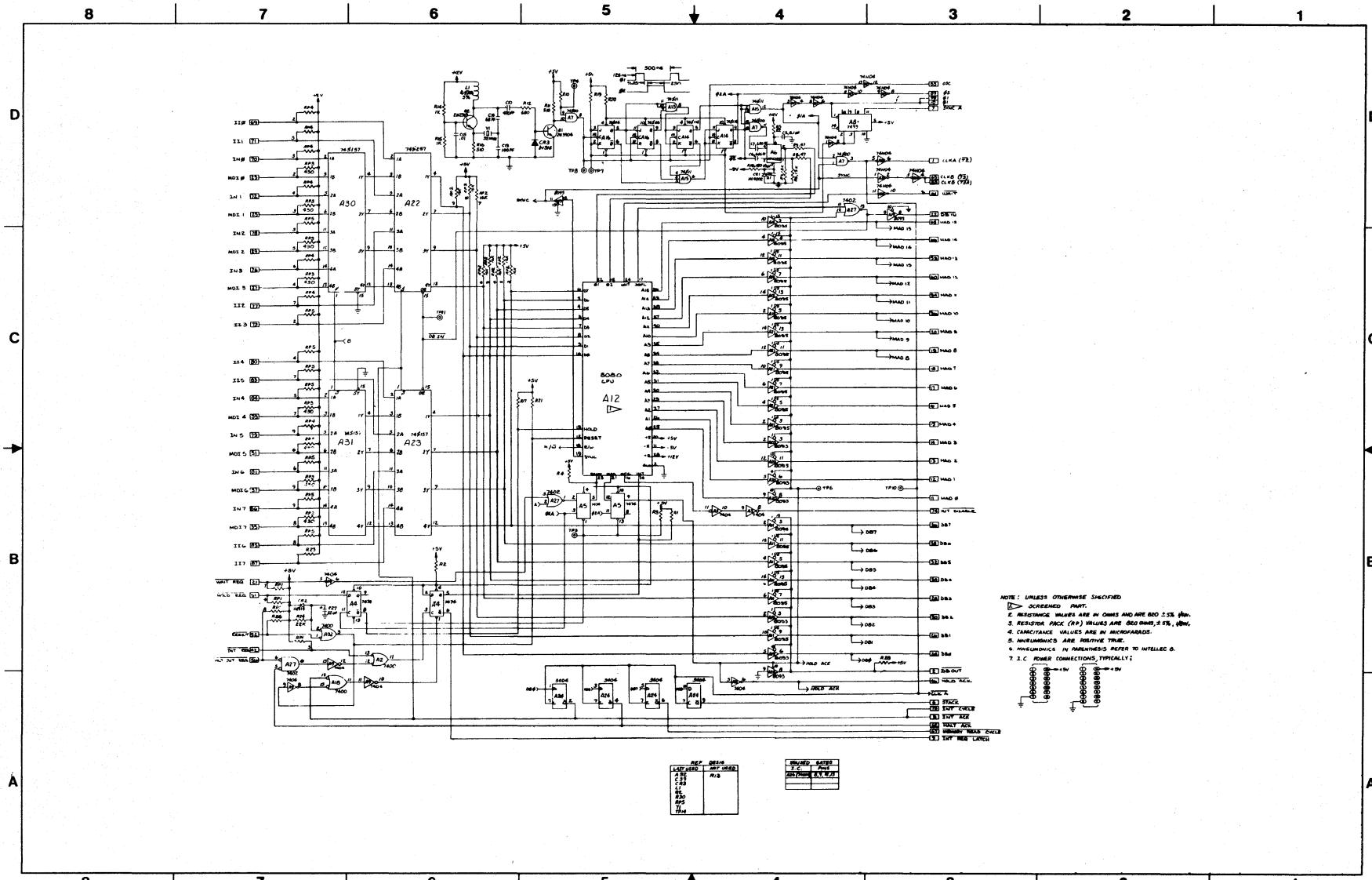


Figure 2-16. imm8-83 Central Processor Module Schematic Diagram (Sheet 1).

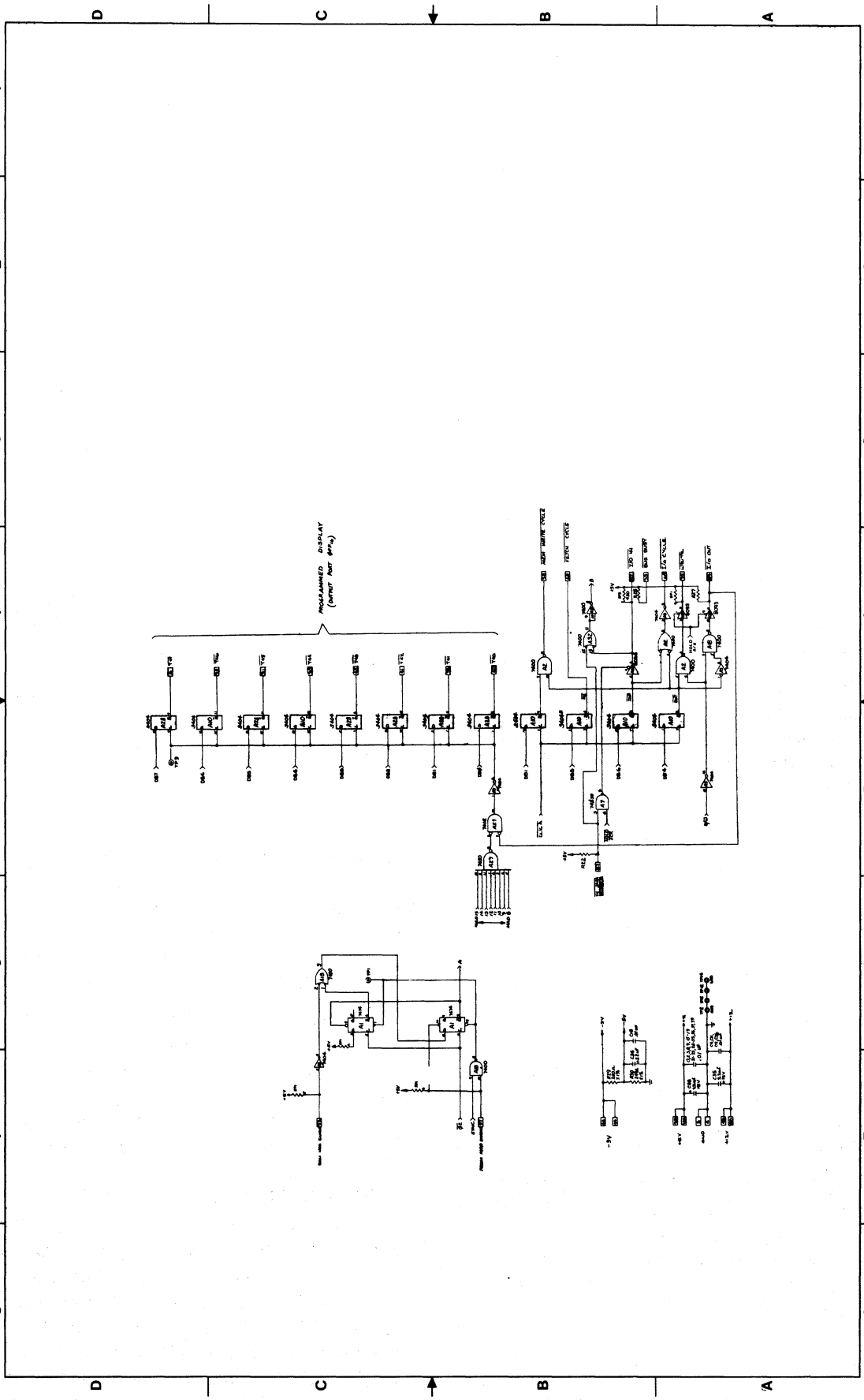


Figure 2-16. imm8-83 Central Processor Module Schematic Diagram (Sheet 2).

MEMORY READ to obtain the eight-bit address of the peripheral device involved, and an INPUT or an OUTPUT to execute the transfer. FETCH and MEMORY READ have already been described. Execution is described below.

The first byte that the processor fetches from memory indicates the kind of transfer to be conduced. The second byte, store immediately following the first, contains the eight-bit address of the object peripheral. Thus, one of 256 input devices or 256 output devices may be designated in such a transfer. Having fetched these two bytes from memory, the processor proceeds as follows.

The third machine cycle is designated an INPUT or an OUTPUT, depending upon the instruction byte originally fetched. An INPUT machine cycle is identified by an INP status bit, published during T1 as usual. An OUTPUT machine cycle, on the other hand, will be identfied by an OUT status bit. In both cases, the address of the designated peripheral is also sent out during T1. Address data on lines A_0-A_7 is repeated on lines A_8-A_{15}.

In an input operation, status information is cleared from the main data bus during the T2 state. Address data remains stable, however, throughout the remainder of the machine cycle. The output of the INP status latch is buffered in a tri-state section and made available the module's edge connector ($\overline{\text{I/O IN}}$), to enable the transfer externally. Details are shown on the module schematic, Figure 2-16 (sheet 2).

Observe also that the $\overline{\text{I/O IN}}$ signal is applied to A32-13. The resulting high at the gate's output is inverted subsequently and routed to the control input of the input multiplexer's first stage. You will recall that a low at this point causes the multiplexer to select and forward data on the peripheral input bus to the second stage multiplexer. The second stage, enabled by a $\overline{\text{DBIN}}$ signal from the processor, in turn forwards the input data to the processor's main bus. There it is picked up and stored in the 8080's accumulator register.

In an output transfer, status information is again cleared from the processor's main data bus during the T2 state. It is replaced, however, with the eight-bit data word stored in the processor's accumulator. Just as in the case of input, the address lines remain stable throughout the machine cycle. Inputs to the processor's main data bus are inhibited by the absence of DBIN, and the CPU generates a $\overline{\text{WR}}$ output signal to implement the data transfer.

Referring again to sheet 2 of Figure 2-16, observe that $\overline{\text{WR}}$ is inverted and applied to A18-4. There it is ANDed with the inverted output of the OUT status latch, producing an $\overline{\text{I/O OUT}}$ signal which synchronizes the external transfer. Output data from the processor thus passes to its addressed destination, completing the output cycle.

Interrupt Cycle

From the point of view of the 8080 CPU, the interrupt cycle is simply a modified FETCH machine cycle. Externally, the operation of the CPU appears much the same. The sequence is as follows:

An incoming $\overline{\text{INTERRUPT REQUEST}}$ enters the Central Processor Module asynchronously at pin #42, as shown in Figure 2-16 (sheet 1). It is gated through the 11-12-13 section of a 7400 NAND-gate (A2), and applied to the clock input of a 7474 latch section (A4). Here the request is stored, until the processor can properly acknowledge it. The high at the latch's Q output is forwarded directly to the 8080's INTERRUPT input.

After completing the machine cycle in progress, the processor acknowledges the interrupt. This it does by entering an alternative INTERRUPT machine cycle, rather than proceeding directly to the next instruction fetch. As we explained, the processor transmits an address during the INTERRUPT-T1 state, but the internal program counter is not incremented. As a result, the logic sequence of the interrupted program is maintained. When the interrupt has been processed, the main program may therefore be resumed with no loss of continuity.

The processor publishes an INTA status bit during T1, identifying the machine cycle in progress as an INTERRUPT machine cycle. This bit is saved in the INTA status latch, and presented at the module's edge connector, as an external acknowledgement of the $\overline{\text{INTERRUPT REQUEST}}$. At approximately the same time, the 8080 CPU disables its INTERRUPT input. This is an internal processor function, but the resulting low at the processor's INTE output is buffered and made available at the edge connector, to indicate that the INTERRUPT facility has been disabled.

The output of the INTA status latch is forwarded to the control pin of the input multiplexer's second stage, causing the multiplexer to select and forward data at the interrupt instruction port to the processor's main data bus. Thus the processor interprets data at the interrupt port as an instruction, and executes it accordingly. The Central Processor Module returns to its normal mode of operation as soon as the INTERRUPT cycle is completed.

Note that the INTA status bit is used within the module to reset the interrupt request latch, removing the request from the processor as soon as it has been acknowledged. By terminating the request promptly, the module's interrupt logic ensures that a spurious second INTERRUPT cycle is not generated inadvertently.

Hold Operations

The peripheral device requesting a hold applies a HOLD REQUEST at pin #51 of the Central Processor Module. The request is forwarded directly to the D input of a 7474 latch section where it is re-clocked and transmitted to the HOLD input of the processor. As explained, the 8080 CPU responds to such a request during the next T3 (or T4) state, by floating its address and data busses and by transmitting a HLDA signal which acknowledges the $\overline{\text{HOLD REQUEST}}$.

If you refer back to Figures 2-9 and 2-10, however, you will observe that there is a brief delay between the rising edge of the HLDA signal and the actual floating of the busses. This makes it advisable to re-clock the HLDA, before using it to acknowledge the request externally. To achieve this purpose, HLDA is applied to the D input of a 7474 latch section (A5) and re-clocked by the ϕ_2 timing pulse. The output of the latch, which now coincides with the processor's internal activity, is buffered and made available at pin #2 at the $\overline{\text{HOLD ACK}}$ signal.

Refer now to the module schematic, Figure 2-16 (sheet 1). Note that the re-clocked output of the hold acknowledge flip-flop is also used by circuitry on the module to perform the following functions:

a) float the address bus

b) float the data output bus

c) float the $\overline{\text{I/O IN}}$ control line

d) float the $\overline{\text{I/O OUT}}$ control line

e) float the $\overline{\text{WRITE}}$ control line

f) enable the second stage of the input multiplexer

g) float the $\overline{\text{DBOUT}}$ control line

These functions may be verified by tracing out the distribution of the HOLD ACK signal, from its origin at A5-9 to the various points shown on the schematic diagram (Figure 2-16, sheet 1 and sheet 2). They assure the requesting peripheral complete control of the memory's busses and control lines, until such time as the external $\overline{\text{HOLD REQUEST}}$ is removed.

Note that the input multiplexer has to be enabled explicitly by the HOLD ACK signal. This is necessary, since the memory's data output lines have no other way of communicating with the input of the requesting peripheral. Note too, that the module's data output lines are inhibited by their common enabling line through the 8-9 section of A25. A25 is placed in a high impedence state with passive pull up allowing the requesting peripheral to override the hold logic when receiving data from memory. By commanding the module's $\overline{\text{DB OUT}}$ line directly, the DMA device can establish continuity between its input lines and the output lines from memory.

Whenever two or more peripherals in the same system have DMA capability, there is always a chance of conflect. One device may request a hold while the other is already in the process of conducting a transfer. Finding the $\overline{\text{HOLD ACKNOWLEDGE}}$ line enabled, the requesting device is liable to proceed with its intention to transfer data. It will come into direct conflict with the first device.

To prevent this possibility, the processor module maintains a $\overline{\text{BUS BUSY}}$ status line. Pin #53 of the module is returned internally to the +5 Volt supply, through a 1K pull-up resistor. It becomes the logical responsibility of a device controller to monitor this line before requesting a hold. If the line is high, the operation may proceed. If not, it must wait. Any controller requesting a hold must clamp the $\overline{\text{BUS BUSY}}$ line, in order to protect its prior right of access. (Must also have a daisy chain between peripherals to establish tie breaking priorities).

Reset

The reset logic is shown on sheet 1 of the module schematic, Figure 2-16.

An external $\overline{\text{RESET}}$ is applied to pin #52 of the Central Processor Module. It passes through the NOR-gate A32: 1-2-3 and is forwarded to the CPU's RESET input. The processor's internal program counter and instruction register are zeroed, as explained on page 24.

However, there is also provision in the reset logic for the generation of an automatic RESET whenever the module is brought up from a power-down condition. Capacitor C24 charges to the level of V_{CC} through a 22K resistance, R23. Under normal operating conditions, the capacitor is fully charged. Whenever power to the module is interrupted, however, the capacitor discharges rapidly through the diode CR2. Thus, when power ultimately returns, the charge on the capacitor must be restored exponentially through R23. During this time, a low is applied to pin 2 of A32, and the output of the gate generates a RESET of the CPU. In this way, proper initialization of the processor is assured.

Programmed Display

Logic for the programmed display port is shown on sheet 2 of the Central Processor Module schematic, Figure 2-16.

As shown, a type 7430 NAND-gate is used to indicate coincidence whenever the address FF_{16} is presented on the module's address bus. The output of the address gate is combined with the $\overline{\text{I/O OUT}}$ signal in a second gate (A27), inverted, and used to drive the common strobe inputs of the 3404 inverting latches shown in the upper right portion of the drawing. The coincidence of the address FF_{16} and the $\overline{\text{I/O OUT}}$ signals accordingly causes these latches to record the data on the module's data out bus. Programs may write data into this port, for display on the INTELLEC 8's Console.

UTILIZATION

This section provides information on utilization of the imm8-83, for using the module outside the INTELLEC 8/MOD 80 system.

Installation

In installing the Central Processor Module, the user must take account of:

a) environmental extremes

b) mounting

c) electrical connections

d) power requirements

e) signal requirements

ENVIRONMENT

Temperature extremes can cause instability, or result in permanent damage to the circuits on the module. Ambient temperature must therefore be maintained within the limits of 0° to 70° Centigrade. Exercise caution in locating the module, giving particular attention to radiant and conducive sources of heat. Remember that the module itself, when installed, will contribute some heat to the environment. Maintain an adequate clearance, to permit the convective dissipation of heat from the elements on the card.

Relative humidity is not critical to the module's operation.

MOUNTING

Avoid locating the module near vibrating machinery. Exposure to prolonged or violent vibration may cause fatigue or impact failure of connections on the board, resulting in abnormally high noise levles or outright failure of the assembly.

Dimensions of the module are 6.18 x 8.00 inches. Be sure to allow enough additional clearance to ensure adequate cooling.

The module is designed to plug directly into a standard 100-pin, doub'e-sided PC edge connector. The connector will serve as a mounting, as well as an electrical junction, if the environment is not too severe. Card guide slots are desirable, for the additional protection they afford. Should vibration be a problem, however, or should the assembly be used in a portable equipment application, an additional retaining bracket will have to be provided. When mounting the board, remember that it is desirable to orient the assembly vertically wherever possible. This optimizes convective cooling of the components on the module.

ELECTRICAL CONNECTIONS

The basic power and control connections to the CPU Module are made through a standard 100-pin, double-sided PC edge connector (0.125" contact centers). CDC #VPB 01C50E00A1 is one suitable type. Pin allocations on the connector are given in Table 2-4.

POWER REQUIREMENTS

The Central Processor Module requires DC power, at the following levels:

Supply Volts	Tolerance	Typ Load	Max Load
+12 VDC	±5%	0.04 Amps	0.06 Amps
+5 VDC	±5%	1.00 Amps	1.50 Amps
-9 VDC	±5%	0.10 Amps	0.15 Amps

Refer to the pin list for power connections.

SIGNAL REQUIREMENTS

All data and control functions on the module are at TTL levels. Electrical characteristics of the inputs and outputs are given in Table 2-3 for the various types of IC devices.

Signal discriptions and connector pin allocations are given in Table 2-4.

Pin List

The following section describes connector pin allocations on the Central Processor Module. The pins and their designated signal functions are listed in Table 2-4.

CPU Module: D.C. Signal Characteristics

Parameter		Device Type				Unit
		74xx	74Sxx 74Hxx	8093	8095	
I_{OH}	High-level output current	0.400	0.500	5.2	5.2	μA
I_{OL}	Low-level output current	16	20	16	32	mA
I_{IH}	High-level input current	0.040	0.050	0.040	0.040	μA
I_{IL}	Low-level input current	1.6	2.0	1.6	1.6	mA

Table 2-3

CPU Module Output Connector

Pin #	Name	Signal Function
1	CLKA	T2 Synchronization
2	$\overline{\text{DB OUT}}$	Output Data Enabling
3	GND	Supply Common
4	GND	Supply Common
5	$\overline{\text{INT ACK}}$	Interrupt Cycle Status
6	$\overline{\text{STACK}}$	Stack Reference Cycle Status
7	$\overline{\text{SYNCA}}$	Ø1 Modulo -8: F/P Logic
8	$\overline{\text{T47}}$	Programmed Display Bit 7
9	$\overline{\text{INT REQ LATCH}}$	Interrupt Requested
10	Ø1 ($\overline{\text{T1}}$)*	Ø1 Processor Clock Out (T1)
11	MADØ	Address Bit Ø
12	MAD1	Address Bit 1
13	MAD2	Address Bit 2
14	MAD3	Address Bit 3
15	MAD4	Address Bit 4
16	MAD5	Address Bit 5
17	MAD6	Address Bit 6
18	MAD7	Address Bit 7
19	MAD8	Address Bit 8
20	MAD9	Address Bit 9
21	$\overline{\text{WAIT REQUEST}}$	Ready Flag from Memory
22	$\overline{\text{DB IN}}$	Input Data Enabling
23	MDIØ	Memory Input Data Bit Ø
24	DBØ	Output Data Bit Ø
25	MDI1	Memory Input Data Bit 1
26	DB1	Output Data Bit 1
27	MDI3	Memory Input Data Bit 3
28	DB3	Output Data Bit 3
29	MDI2	Memory Input Data Bit 2
30	DB2	Output Data Bit 2
31	MDI5	Memory Input Data Bit 5
32	DB5	Output Data Bit 5
33	MDI4	Memory Input Data Bit 4
34	DB4	Output Data Bit 4
35	MDI7	Memory Input Data Bit 7
36	DB7	Output Data Bit 7
37	MDI6	Memory Input Data Bit 6

Pin #	Name	Signal Function
38	DB6	Output Data Bit 6
39	$\overline{\text{T41}}$	Programmed Display Bit 1
40	$\overline{\text{T4Ø}}$	Programmed Display Bit Ø
41	$\overline{\text{T42}}$	Programmed Display Bit 2
42	$\overline{\text{INTERRUPT REQ}}$	Initiate External Interrupt
43	-9 VDC	V_{BB} Source Power
44	-9 VDC	V_{BB} Source Power
45	$\overline{\text{T43}}$	Programmed Display Bit 3
46	HOLD ACK	Acknowledge Hold Request
47	-12 VDC	
48	-12 VDC	
49	+12 VDC	V_{DD} Source Power
50	+12 VDC	V_{DD} Source Power
51	$\overline{\text{HOLD REQ}}$	Initiate External Hold
52	$\overline{\text{RESET}}$	Initiate External Reset
53	$\overline{\text{BUS BUSY}}$	DMA In Progress Signal
54	$\overline{\text{I/O OUT}}$	I/O Output Strobe
55	OSC	32 MHz Oscillator Output
56	$\overline{\text{HLT INT REQ}}$	Processor Restart Interrupt
57	$\overline{\text{IN JAM ENABLE}}$	Disable I/O IN Strobe
58	$\overline{\text{MEM WRITE CYCLE}}$	Memory Write Cycle Status
59	MAD13	Address Bit 13
60	MAD12	Address Bit 12
61	$\overline{\text{WAIT}}$	Wait Request Acknowledge
62	$\overline{\text{HLT ACK}}$	Halt Cycle Status
63	CLKB (T3)*	Processor Cycle SYNC
64	$\overline{\text{I/O CYCLE}}$	I/O Cycle Status
65	MAD15	Address Bit 15
66	MAD14	Address Bit 14
67	$\overline{\text{MEM READ CYCLE}}$	Memory Read Cycle Status
68	$\overline{\text{FETCH CYCLE}}$	Instruction Fetch Cycle Status
69	IIØ	Interrupt Instruction Bit Ø
70	INØ	Peripheral Input Bit Ø
71	II1	Interrupt Instruction Bit 1
72	IN1	Peripheral Input Bit 1

Table 2-4

*imm8-82 function

CPU Module Output Connector

Pin #	Name	Signal Function
73	II3	Interrupt Instruction Bit 3
74	INT DISABLE	Interrupt Disabled Flag
75	INT CYCLE	Interrupt Cycle Status
76	IN3	Peripheral Input Bit 3
77	II2	Interrupt Instruction Bit 2
78	IN2	Peripheral Input Bit 2
79	IN5	Peripheral Input Bit 5
80	II4	Interrupt Instruction Bit 4
81	IN6	Peripheral Input 6
82	$\overline{\text{I/O IN}}$	I/O Input Strobe
83	II5	Interrupt Instruction Bit 5
84	IN4	Peripheral Input Bit 4
85	II6	Interrupt Instruction Bit 6
86	IN7	Peripheral Input Bit 7
87	II7	Interrupt Instruction Bit 7
88	CLKB (T3A)*	Processor Sub-Cycle SYNC
89	Ø2	Ø2 Processor Clock Out
90	$\overline{\text{T44}}$	Programmed Display Bit 4
91	$\overline{\text{T45}}$	Programmed Display Bit 5
92	$\overline{\text{T46}}$	Programmed Display Bit 6
93	RAM MOD ENABLE	RAM Memory SYNC Select
94	MAD11	Address Bit 11
95	$\overline{\text{WRITE}}$	Memory Write Strobe
96	MAD1Ø	Address Bit 1Ø
97	PROM MOD ENABLE	PROM Memory SYNC Select
98	Ø1	Ø1 Processor Clock Out
99	+5 VDC	V_{CC} Source Power
100	+5 VDC	V_{CC} Source Power

Table 2-4 (cont'd.)

*imm8-82 function

CHAPTER 3 THE imm8-61 INPUT/OUTPUT CARD

The imm8-61 Input/Output Card has been designed to provide the user with an input/output facility containing four individually addressable input ports, two of which provide built-in Teletype interfacing and control and four individually addressable output ports, again with two of the ports providing Teletype interfacing. The need for separate external Teletype controllers is thereby eliminated, as is the need to design input and output facilities.

The imm8-61 Card has been designed to allow four cards to be used in an INTELLEC 8/MOD 80 system, with each card having a unique address by which it is referenced. The imm8-61 Card includes all logic necessary to support a multi-card implementation. Though each imm8-61 module has only four input ports and four output ports, the combination of two sets of jumpers and four useable card positions allows implementation of ports 0-63 (out of a possible 256) with four imm8-61 modules.

Although the imm8-61 Card has been designed to support the Intel imm8-83 Central Processor Card, it may be used in any application which can use its easily implemented input/output sub-system, its integral Teletype communications facilities, its great flexibility, and its low cost.

This section describes the operation and implementation of the imm8-61 Input/Output Card at three levels; first, the operation of the imm8-61 is described on a basic functional level; second, theory of operation is provided; third, necessary information to effectively use the imm8-61 Card is given. This last section covers such areas as user-available options, signal and installation requirements, etc.

THE imm8-61 INPUT/OUTPUT CARD – GENERAL FUNCTIONAL DESCRIPTION

This section describes the operations of the imm8-61 Input/Output Card in general functional terms, and is divided into six subsections. The first subsection describes the five functional units which enable all of the operations performed by the card. The second subsection describes the Module Select and Port Select operations, as these two operations are common to all other operations performed by the card. The third subsection describes a typical input operation, showing the interrelationship of the functional blocks in that operation. The fourth subsection describes an output operation in similar terms, while the fifth and sixth subsections describe, respectively, Teletype input and Teletype output operations.

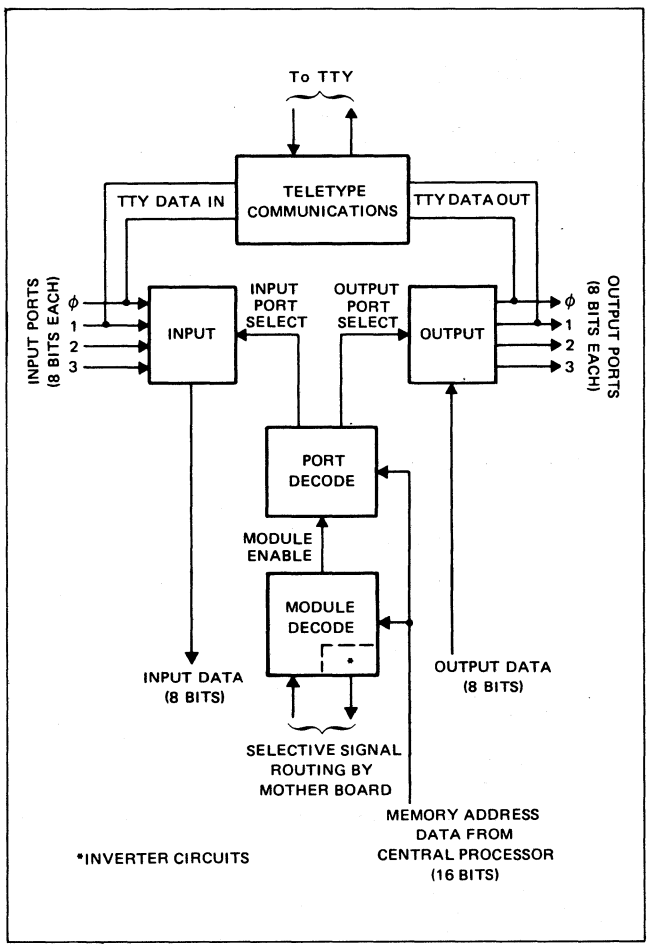


Figure 3-1. I/O Functional Block Diagram.

The Functional Units

In order to describe its operation, the imm8-61 Card can be divided into five functional units:

1) The *Module Decode Block*, which determines which card is to be utilized for an operation when more than one card has been installed in a system.
2) The *Port Decode Block*, which determines which of the 64 possible input and output ports is to be used for an operation.
3) The *Input Block*, which contains the four input ports and their associated logic.
4) The *Output Block*, which contains the four output ports and their associated logic.
5) The *Teletype Control Block*, which receives data from, and transmits data to the Teletype, and which performs the necessary conversion of the data (serial to parallel in the case of Teletype Input, and parallel to serial in the case of Teletype output).

Each operation performed by the imm8-61 Card uses one or more of these units in its execution.

A block diagram of the imm8-61 Input/Output Card, showing the five functional units and their interrelationships, is given in Figure 3-1, and should be referred to when reading the rest of this section.

Module and Port Select Operations

The first operation performed by the imm8-61 Card is always a Module and Port Select operation. A Module and Port Select operation is performed via the following steps:

1) The Central Processor (Intel imm8-83 or equivalent) sends an I/O Address to the Module Select and Port Select Blocks. This I/O Address contains the information necessary to specify which card is to be used for an operation (in a multi-card system), what type of operation is to be performed (Input or Output), and which port is to be used for that operation. Both the complemented and non-complemented levels on the high-order address lines are returned to the mother-board, in turn, selectively returns either the complemented or non-complemented level for each of the high-order address bits (depending on the card position) to the module decoder(s), on lines DS 10, 11, 14 and 15. Thus the position of a module determines which sixteen addresses (of a possible 64) it will respond to. Jumpers on the I/O module's Port Decode Block (jumping address lines 12 and 13) in turn, determine which four of these sixteen addresses are recognized.
2) The selected card is identified by the card's Module Select Block, which generates an enable signal which is transmitted to the rest of the card logic.
3) The Port Decode Block, on the selected card, determines which of the actual eight ports is being addressed by the I/O Address (0-63). It then sends enabling signals to either the Input or the Output block, depending on whether an Input or Output port was addressed.

This sequence of operations takes place before every I/O operation.

Input Operation

An input operation is performed in order to obtain data from an external source and to present it to the Central Processor. The imm8-61 Input/Output performs an input operation in the following steps:

1) The data from the external device is brought into the Input block.
2) When the proper enabling signals are generated by the Module Decode and Port Decode blocks, the data which has been input from the external device to the Input block is sent out to the Central Processor on the Input Data bus.

Output Operation

An output operation is performed in order to receive data which is sent out from the Central Processor and to hold it for use by an external device. The imm8-61 Card executes an output operation in the following steps:

1) The Central Processor sends the I/O Address (0-63) to the imm8-61 Card, and a Module and Port Select operation is performed.
2) The Central Processor sends the data which is to be output to the Output block.
3) The data is placed into the selected output port, under control of enabling signals generated during the Module and Port Select operations.
4) The data is held in the selected output port for use by the external device associated with that port.

Note that data is held in an output port until another output operation is performed using the same output port.

Teletype Input Operation

A Teletype Input operation is performed in order to accept information from an ASR-33 Teletype or Teletype-compatible device, and to send that data to the Central Processor. It is performed in the following steps:

1) Data from the Teletype is sent to the Teletype Control block.
2) The Teletype Control block converts the data to a form useable by the Input block, and sends the data and status signals to the Input block or input ports 0 and 1.
3) When the proper enabling signals are sent to the Input block by a Module and Port Select operation

the Teletype data is sent out to the Central Processor on the Input Data bus.

Note that a Teletype Input operation differs from a non-Teletype Input operation only in that the Teletype Control block acts as a buffer between the Teletype and the Input block.

Teletype Output Operation

A Teletype Output operation is performed in order to send information from the Central Processor to the ASR-33 Teletype or Teletype-compatible device, and is performed in the following steps:

1) The Central Processor sends an I/O Address specifying output port 0 to the imm8-61 Card, and a Module and Port Select operation is performed as described in Module and Port Select Operations.
2) Teletype output data is sent by the Central Processor to the Output block via the Output Data bus.
3) The Teletype data is placed into output port 0 under control of the enabling signals generated by the Module and Port Decode blocks during the Module and Port Select operation.
4) The data in output port 0 is sent to the Teletype Control block, which converts it into a form useable by the Teletype.
5) The Teletype Control block sends the converted data to the Teletype.

Note that an output operation to the Teletype is equivalent to a normal non-Teletype Output operation in which the Teletype Control block is used as the external device.

imm8-61 INPUT/OUTPUT CARD — THEORY OF OPERATION

This section describes, in detail, the theory of operation of the imm8-61 Input/Output Card. The circuit-level implementation of the features described will be given.

Module Selection

If more than one imm8-61 Card is present in a system, provisions must be made for an operation to select one card. This capability is provided by the Module Decoding Circuits.

Module address information is brought to imm8-61 Card edge pins; the module address is complemented by a series of inverting latches and the complemented address is present at additional imm8-61 Card edge pins. The user selects an address for each imm8-61 Card, and implements the address by selecting a set of Address and Complemented Address signals; selected signals are externally jumpered to the Module Selection circuits, which combine the incoming signals through a NAND gate (A16) to provide the enabling signal which is sent to other circuitry on the card.

The high-order six address lines are input through an inverting latch. Both the complemented and non-complemented forms of the address bits are returned to the motherboard. The motherboard, in turn, selectively returns either the complemented or non-complemented form of bits 10, 11, 14 and 15 (on lines DS 10, 11, 14 and 15), depending on the card position. DS 10, 11, 14 and 15 are input to the enabling NAND gate (A16). Address lines 12 and 13 are also input to gate A16, however, these lines are routed through jumpers 20 and 23, respectively, on the I/O module. The jumpers enable either the complemented or non-complemented form of these address lines to gate A16. These jumpers determine which four of the sixteen ports, assigned to this card position, will actually be recognized by the Module Decoding Circuits (see Table 3-1).

If the high-order six address bits specify one of the four port addresses recognized by the I/O module, gate A16 generates the module select enabling signal.

Port Addresses Enabled by I/O Module Jumpers

Card Position	Jumpers			
	20-21 23-24	20-22 23-24	20-21 23-25	20-22 23-25
0	0-3	16-19	32-35	48-51
1	4-7	20-23	36-39	52-55
2	8-11	24-27	40-43	56-59
3	12-15	28-31	44-47	60-63

Table 3-1.

Input Operations

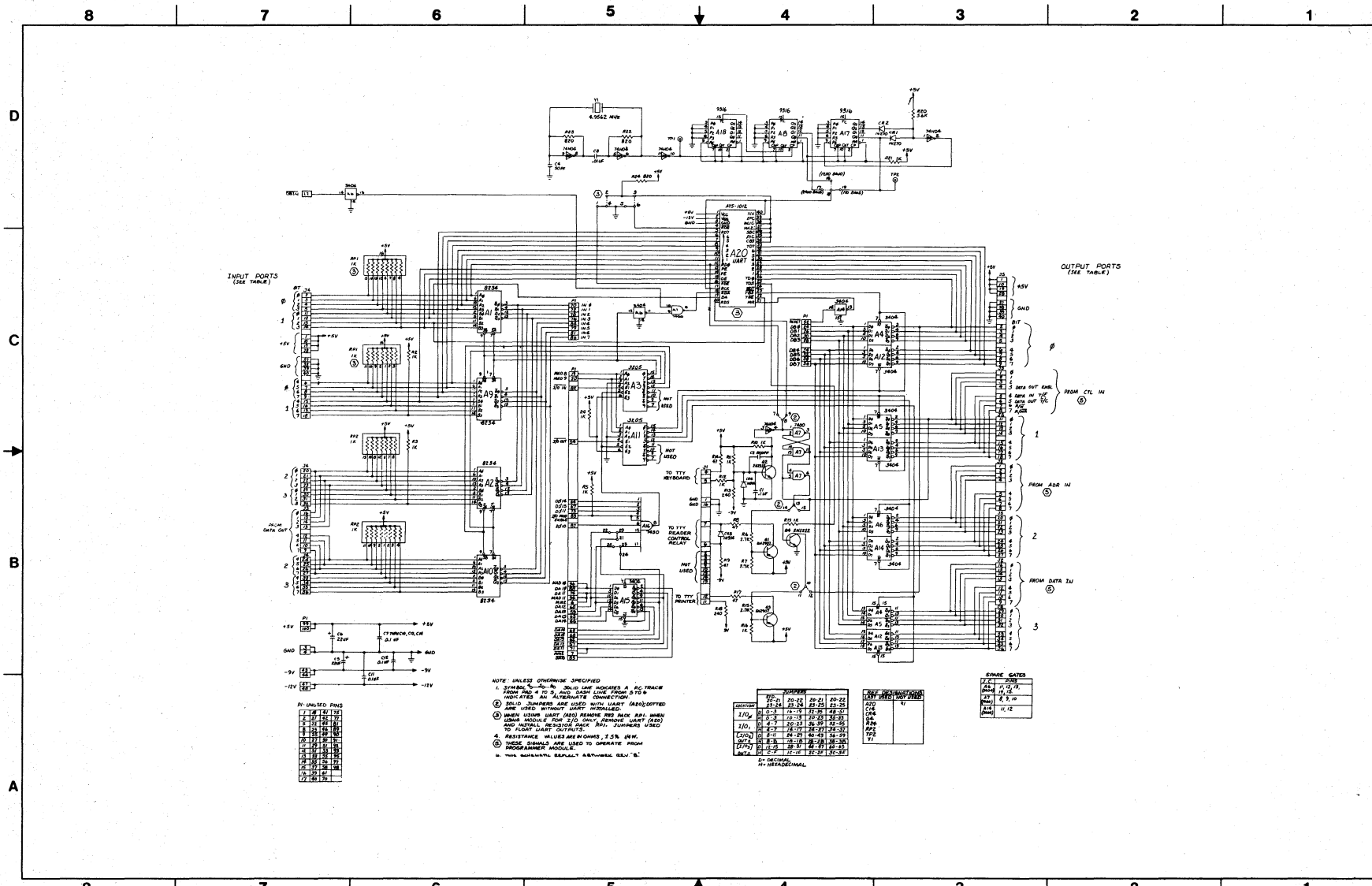
Input operations on the imm8-61 Input/Output Card are handled with the Input Circuits. These are shown on the left in the I/O Module Schematic, Figure 3-2.

The first step in an input operation is the transmission of an I/O Address to the imm8-61 Card from the Central Processor. This I/O Address contains Module and Port Selection information which is necessary to determine which port is to be used for a particular operation.

The Module Selection information is processed by the Module Select Circuits, and causes the Module Enable signal to be produced. This signal is led to the Input Decoder chip, where it is used as an enabling signal.

When it is enabled by the Module Enable signal, and the I/O IN signal sent by the Central Processor, the Input Decoder uses the Port Selection information contained in the I/O Address to produce one of four Port Enable signals. The Port Selection information comes onto the imm8-61 Card on lines MAD8 and MAD9.

The Port Enable signals are led to the four Input Port Multiplexers, and are used to gate one set of input signals through the Input Port Multiplexers onto the Input Data Bus, where the data is available for use by the Central Processor. Timing is shown in Figure 3-3.


INPUT PORTS (SEE TABLE)
OUTPUT PORTS (SEE TABLE)
A20 UART
AY5-1013
4.9662 MHz
TO TTY KEYBOARD
TO TTY READER CONTROL RELAY
TO TTY PRINTER
NOT USED
PROM CTL IN
PROM ADR IN
PROM DATA IN
PROM DATA OUT
NOTE: UNLESS OTHERWISE SPECIFIED
SPARE GATES

Output Operations

Output operations on the imm8-61 Input/Output Card are handled by the Output Circuits, shown on the right in Figure 3-2.

An Output operation begins with the transmission of an I/O Address to the imm8-61 Card from the Central Processor. This I/O Address contains Module and Port Selection information which is used to determine which output port is to be used for a particular operation.

The Module Selection information is processed by the Module Select Circuits and cause the Module Enable signal to be produced. This signal is led to the Outout Decoder chip.

The Central Processor then sends the data which are to be output to the imm8-61 Card on lines DB0-DB7. Along with the output data is sent the $\overline{\text{I/O OUT}}$ signal, which is led to the Output Decoder and is used as a second enabling signal.

When the Output Decoder is enabled by the two enabling signals Module Enable, and $\overline{\text{I/O OUT}}$, it uses the Port Selection information contained in the I/O Address to produce one of four Port Enable signals. The Port Selection comes into the imm8-61 Card on lines MAD8 and MAD9.

The Port Enable signals are used to gate the output data sent by the Central Processor into the proper Output Port Latches. The data is held in the Output Port Latches until another output operation is executed using that output port. Timing is shown in Figure 3-3.

Teletype Communications

Teletype communications can be handled directly by the imm8-61 Input/Output Card, rather than requiring a separate Teletype communications interface and controller. This function is performed by the Teletype Communications Circuits, shown in the upper central section of Figure 3-2.

Teletype Communications on the imm8-61 Card are handled through Input Ports 0 and 1 and Output Ports 0 and 1. Input Port 0 handles Teletype data which are to be input to the Central Processor; Input Port 1 handles Teletype status information. Output Port 0 holds the data which are output from the Central Processor to the Teletype, and Output Port 1 holds the control data used to control Teletype communications. All Teletype input and output operations, with the exception that the on-card Teletype Communications Circuits are used as the input and output device for Teletype operations.

The heart of the Teletype Communications Circuits of the imm8-61 Card is the Universal Asynchronous Transmitter/Receiver chip, or UART. This device receives the serial data word which is sent by the Teletype, and converts it to the eight-bit parallel data format used by the imm8-61 Card. It also translates the eight-bit data output by the imm8-61 Card. It also translates the eight-bit data output by the imm8-61 Card into the serial data word which is used by the Teletype.

The UART requires a clock with a frequency of sixteen times the baud (bits per second) rate at which it is to transmit. This clock is provided on the imm8-61 Card by a crystal clock generator which provides a 4.9562 MHz signal. This signal is used to clock a series of two synchronous counters, each of which provides a "divide-by-sixteen" function, thus producing a 19.36 kHz signal. This signal can be used directly, providing 1200 and 2400 baud transmission rates suitable for Teletype-compatible high-speed terminals, or it may be used to clock another synchronous counter. This third counter is set up to provide a "divide-by-eleven"

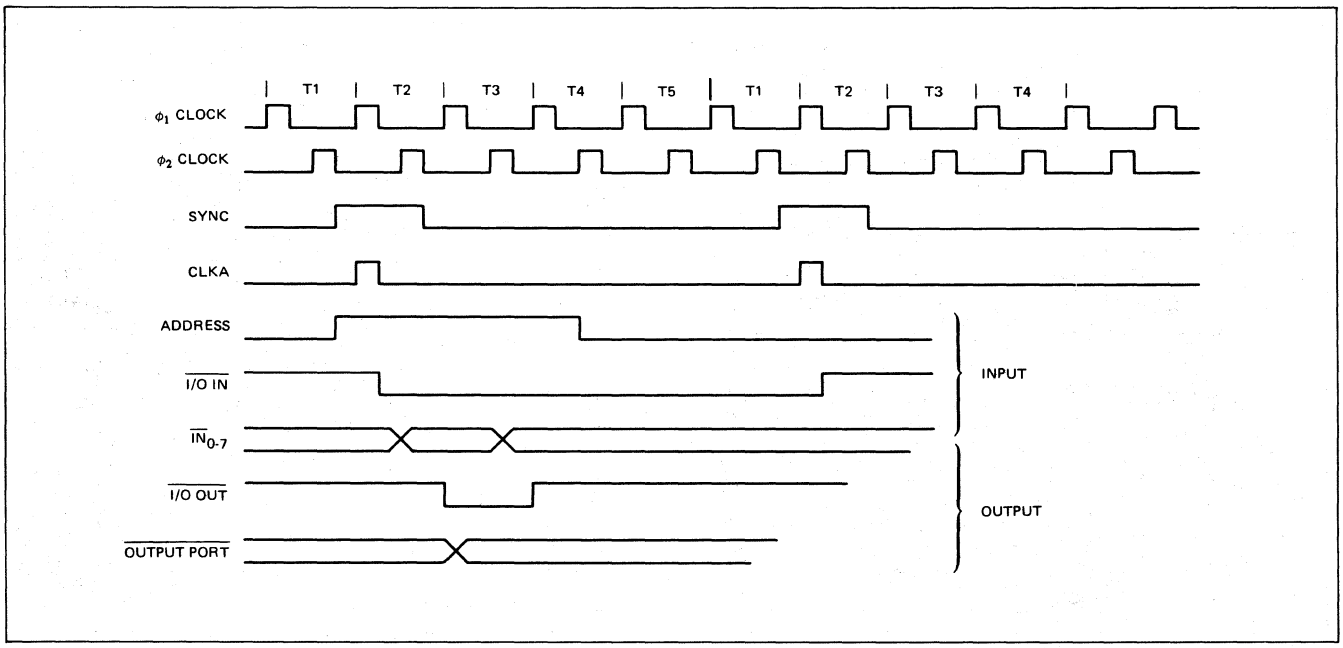


Figure 3-3. I/O Module Timing

capability, and will provide a 1.76 kHz signal which, when used as the UART clock, will provide a 110 baud transmission rate, the standard rate for ASR-33 Teletype communications.

A Teletype input operation begins with the transmission by the Teletype of a data word. This Teletype data is brought onto the imm8-60 Card by way of edge pins as signal TTY XMITR. Since the Teletype information is encoded as variations in current flow, while the UART operates with changes in voltage, the Teletype signal must be converted to a form acceptable to the UART. This is done with transistor Q2 and its associated circuitry. The signal from transistor Q2 is led to the UART Receive Data Input, and the UART converts it into the parallel data used by the imm8-61 and then sends the converted data word to Input Port 0. It also sends status information to Input Port 1. This status information includes Parity Error (PE), Overflow Error (OE), Framing Error (FE), and Data Available (DA). The Central Processor can then execute a normal input operation as described on page 40 in order to obtain the Teletype data.

A Teletype output operation is executed simply by sending the data which are to be output to the Teletype to Output Port 0 via an output operation. The data which are to be sent to the Teletype are latched into Output Port 0 Latch, and sent to the UART. The same enabling signal which was used to latch the data into the Output Port Latch is used to enable transmission by the UART. NOTE: Before a transmission is attempted, Input Port 1 must be interrogated to determine TTY status. The Parallel data will be translated to the serial data format required by the Teletype, and will then be sent to Q3 and Q4, where the necessary conversion from voltage to current coding takes place. The converted signal is then sent to the Teletype as TTY RCVR.

A special feature has been implemented on the imm8-61 Card in order to simplify Teletype paper tape reader operations. Provisions have been made to enable strobing of the paper tape reader one character at a time. This operation is performed when the Central Processor outputs a 1 in the high-order bit of Output Port 1. This signal sets a latch made up of two NAND gates, which in turn produce a signal which is sent to the Teletype paper tape reader as TTY RDR CTL. When a character is read by the Teletype paper tape reader and transmitted to the imm8-61 Card, the signal generated by that transmission, TTY XMITR, resets the latch, causing the TTY RDR CTL signal to fall.

The Teletype Communications Circuits may be reset by a system reset signal. This is done by bringing the signal RESET onto the card, inverting it through an inverting latch, and applying it to the Master Clear input of the UART. This will initialize the UART, and prepare it for further operations.

imm8-61 INPUT/OUTPUT CARD — UTILIZATION

This section describes the options available to the user of the imm8-61 Input/Output Card, and also gives the information necessary to the user for proper installation and operation of the card. There is a wide range of user-available options on the imm8-61 Card, including the choice of usable addresses, the choice of whether or not to use the Teletype Communications Circuits, and the choice of a 110, 1200 or 2400 baud rate for data transmissions.

User-Available Options

By changing a module's card position or by changing jumper connections on an I/O module, the user can choose

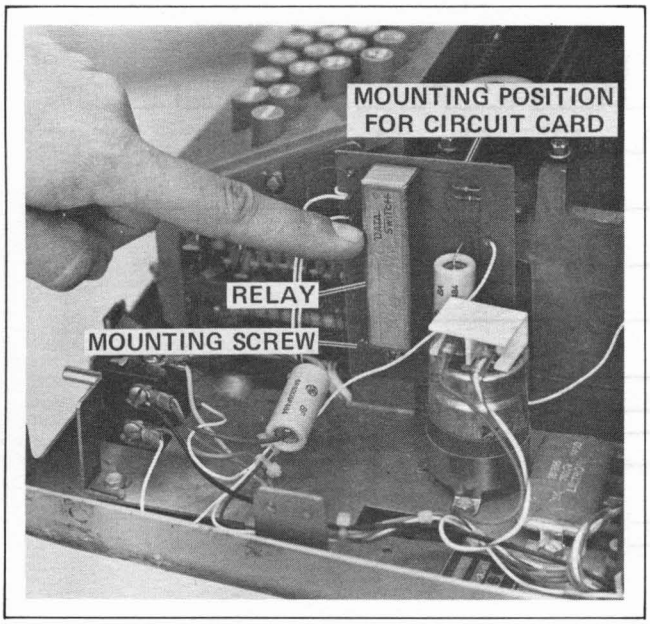


Figure 3-4. Relay Circuit (Alternate)

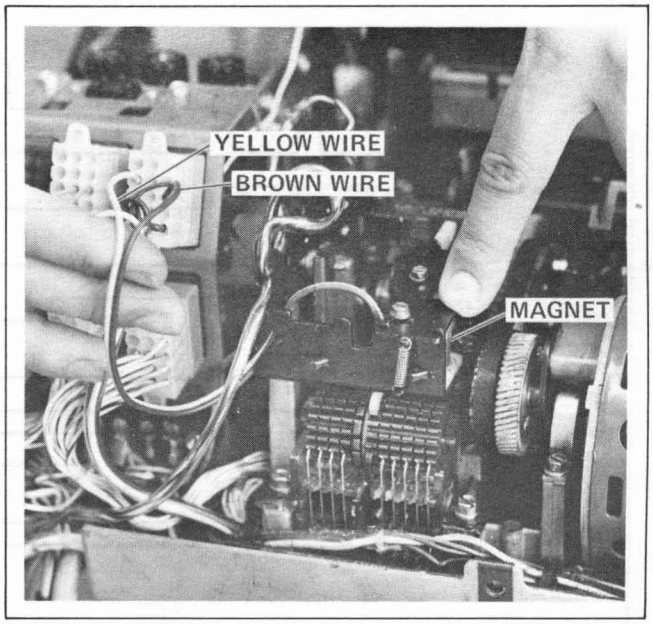


Figure 3-5. Distributor Trip Magnet

the port addresses that a particular I/O module will recognize. Recall that each I/O module has four input ports and four output ports. For any one combination of card position and jumper connections, the module will respond to four addresses (one for each input or output port), but by changing the combinations, the module can be dedicated to respond to any address between 0 and 63. Table 3-1 lists all of the usable combinations.

This option allows a user to develop and debug programs that access up to 64 different input and output device addresses, on an INTELLEC 8/MOD 80 system even though the system actually includes only 16 input and 16 output ports. The option allows lower hardware costs without impeding development.

If it is desired, the imm8-61 Input/Output Card's internal Teletype Communications Circuits may be disabled by removing the UART chip. If this is done, pull-up resistors (resistor pack RP1) must be added to the input data lines on Input Ports 0 and 1. The UART may also be disabled by tying its output enable lines $\overline{RDE}$ and $\overline{FDE}$ to +5v.

Teletype input and output can be accomplished without the use of the UART; that is, on a serial program-controlled basis, by positioning jumpers as follows:

Output: 10-12 instead of 10-11
Input: 7-8 instead of 8-9

When the Input/Output Module is used for the Teletype operations, the user must ensure that no device other than the Teletype is connected to Input Ports 0 and 1 or Output Ports 0 and 1.

The different baud rates can be chosen by positioning jumpers as follows:

110 baud: connect jumpers 18-19; jumper connections 16-18 and 17-18 should be open.

1200 baud: connect jumpers 16-18; jumper connections 18-19 and 17-18 should be open.

2400 baud: connect jumpers 17-18; jumper connections 18-19 and 16-18 should be open.

The imm8-61 Card has been designed to optionally interface with the Intel imm6-76 PROM Programmer Card. This card uses Input Port 2 for a PROM Data Out Port, and Output Ports 1, 2 and 3 as PROM Control In, PROM Address IN, and PROM Data IN, respectively. It is necessary to ensure, if this option is used, that no other device will attempt to use these ports while PROM programming operations are in progress.

Installation Data

Operating Temperature: 0° to $+70^{\circ}$C

DC Power Requirements: +5v ± 5%, .820A Max
-9v ± 5%, .030A Max

Connector: Dual 50-pin, 0.125 in. centers

Teletype Modifications

The ASR-33 Teletype must receive the following internal modifications and external connections.

Internal Modifications

1) The current source resistor value must be changed to 1450 ohms. This is accomplished by moving a single wire (see Figure 3-8).

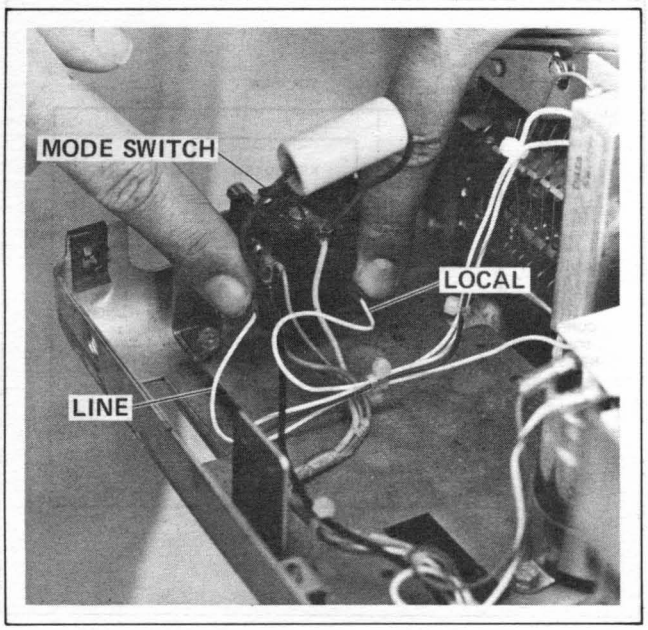


Figure 3-6. Mode Switch

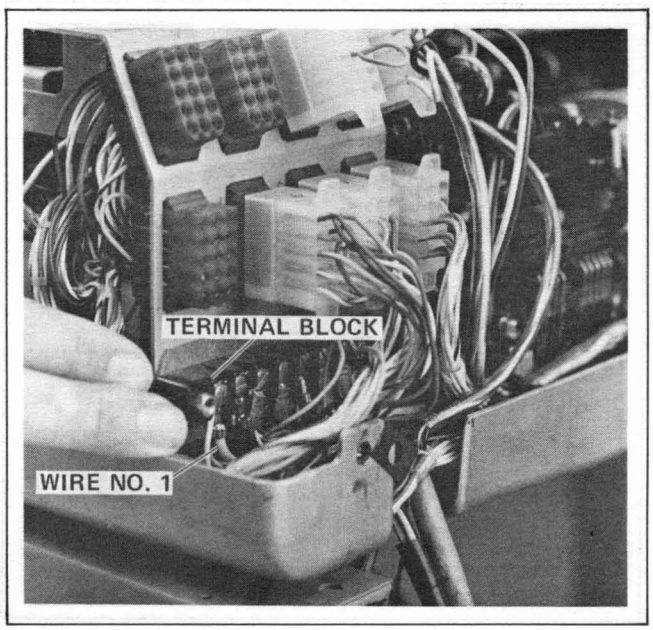


Figure 3-7. Terminal Block

2) A full duplex hook-up must be created internally. This is accomplished by moving two wires on a terminal strip (see Figures 3-7 and 3-9).

3) The receiver current level must be changed from 60mA to 20mA. This is accomplished by moving a single wire (see Figure 3-7 and 3-9).

4) A relay circuit must be introduced into the paper tape reader drive circuit. The circuit consists of a relay, a resistor, a diode, a thyractor and a suitable mounting fixture. This change requires the assembly of a small "vector" board with the relay circuit on it. It may be mounted in the Teletype by using two tapped holes in the base plate (see Figure 3-4). The relay circuit may then be added without alteration of the existing circuit (see Figures 3-5, 3-6, and 3-7). That is, wire "A" (Figure 3-9), to be connected to the brown wire in Figure 3-5, may be spliced into the brown wire near its connector plug. The "line" and "local" wires must then be connected to the mode switch. (See Figures 3-6 and 3-9).

EXTERNAL CONNECTIONS

1) A two-wire receive loop must be created. This is accomplished by the connection of two wires between the Teletype and the SYSTEM in accordance with Figure 3-9.

2) A two-wire send loop similar to the receive loop must be created. (See Figure 3-9).

3) A two-wire tape reader loop connecting the reader control relay to the SYSTEM must be created. (See Figure 3-9).

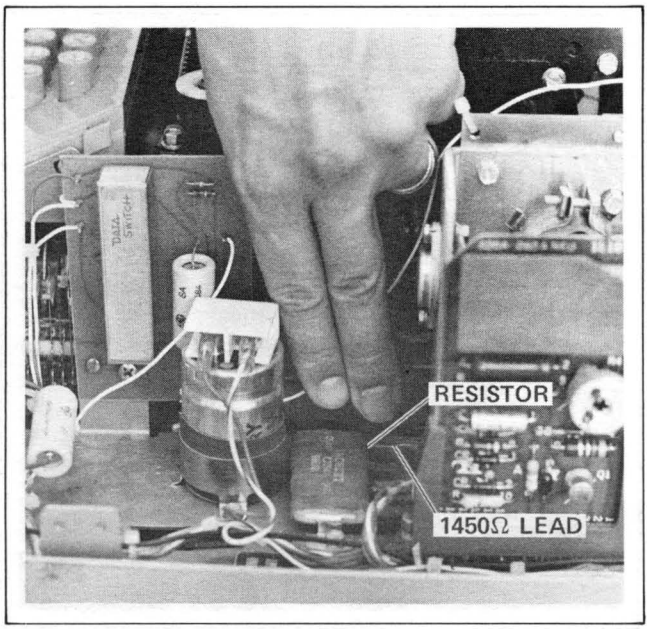


Figure 3-8. Current Source Resistor

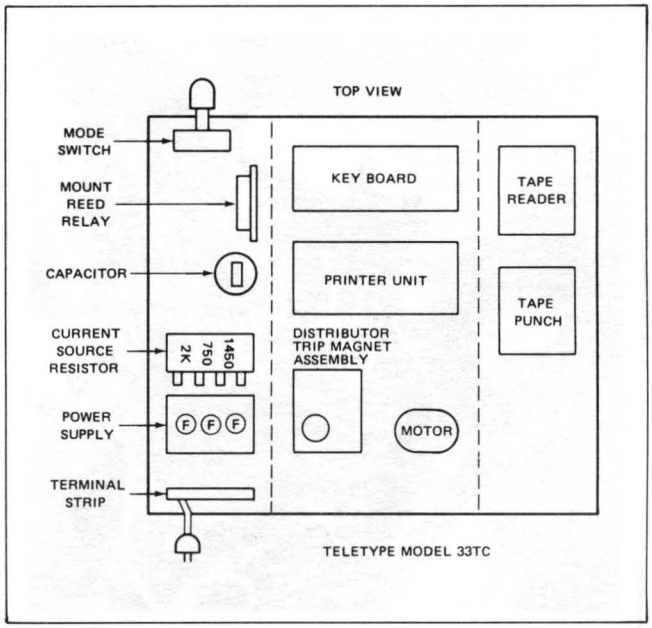


Figure 3-10. Teletype Layout

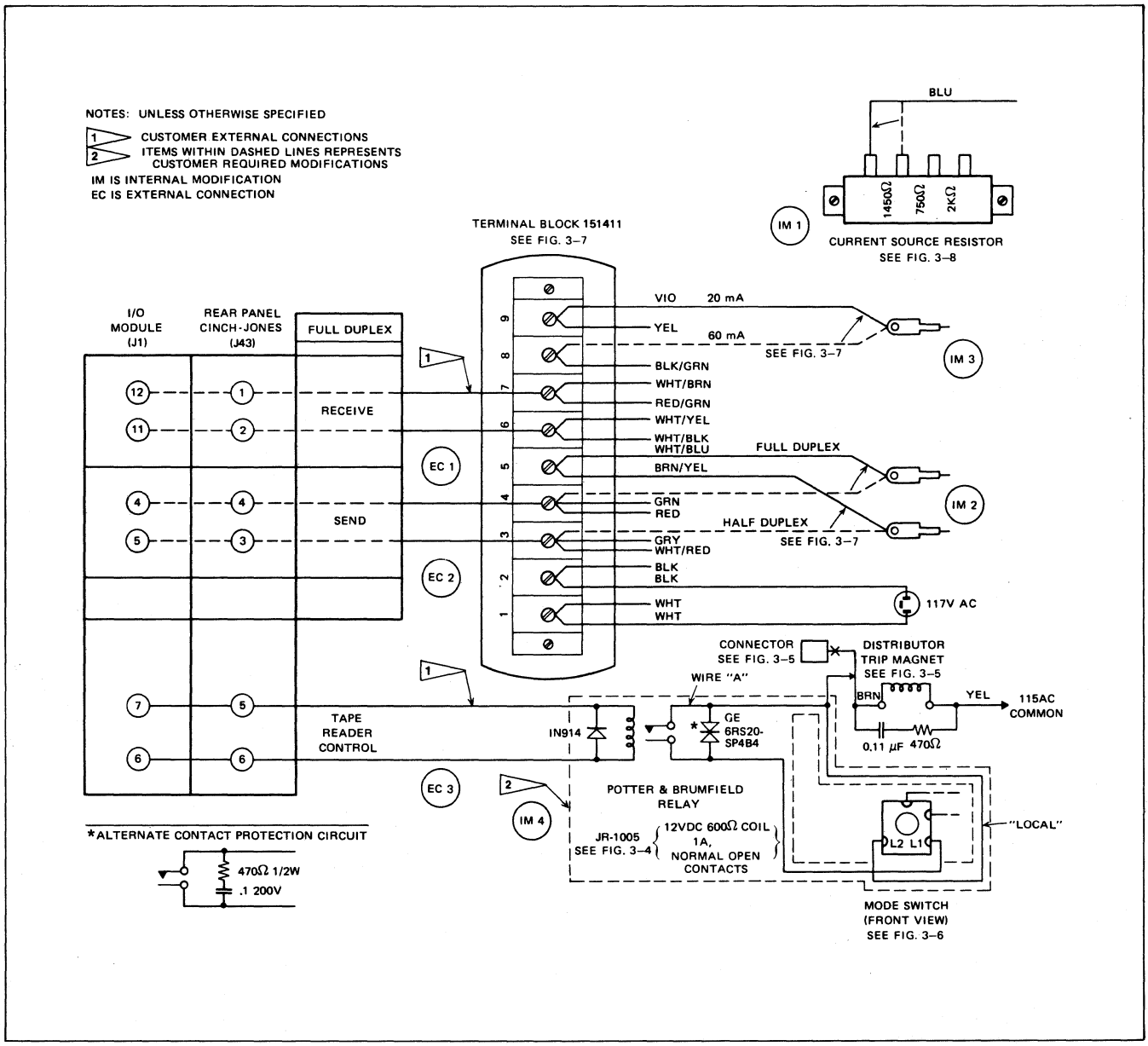


Figure 3-9. TTY Modification

CHAPTER 4 THE imm8-63 OUTPUT CARD

The imm8-63 Output Card contains logic which enables its use as a self-contained output module with eight (8) individually addressable output ports, each of which holds an eight-bit byte of data sent by a Central Processor (such as Intel's imm8-63) for use by an external device. It also contains logic which enables the use of more than one card in any system, with each card individually addressable.

GENERAL FUNCTIONAL DESCRIPTION

The imm8-63 Output Card may be divided into three functional units as shown in Figure 4-1:

- The Module Decode Block
- The Port Decode Block
- The Output Port Block

The Output Port Block contains eight output ports, each of which can communicate with a separate external device. The Port Decode Block determines which of the eight ports is to be used for an operation.

During an output operation, the Central Processor or equivalent device, sends an I/O Address to the Output Card. This information is used by the Module Decode Block to enable output operations (for the particular module being addressed, if there is more than one in the system), and is also used by the Port Decode Block to enable the specific output port which is to be used for output.

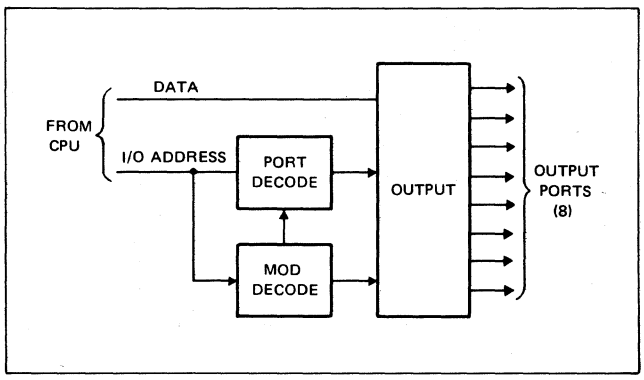


Figure 4-1. Output Module Functional Block Diagram

The Central Processor then sends the data which is to be output to the imm8-63 Card. The data is routed to the Output Port block and is gated into the particular port which was enabled previously by the Port Decode Block. The data are then latched and held for use by the external device associated with that output port.

DETAILED FUNCTIONAL THEORY

This section describes in detail the operation of the imm8-63 Card. Actual circuit-level implementation of the features described as functional blocks in the previous section are given.

Module Decoding

If it is desired to use more than one imm8-63 Output Card in a given system, some provision must be made to enable selection of the particular card which is to be used, out of all of those available. This function is provided by the Module Decoding Circuits, shown in detail in Figure 4-2.

As shown in Figure 4-2, the Module Address information is brought to the imm8-63 Card edge pins and is led to edge pins. The motherboard, in turn, selectively (according to card position) return the proper set of address and inverted address signals to the OUT MOD SEL gate (A14) in the Module Decoding Circuits. In addition, address lines 12 and 13 are routed through jumper connections which provide either an inverted or non-inverted form of the address 12 and 13 signals to the OUT MOD SEL gate (A14). If all the input lines to gate A14 specify that the module is selected, the $\overline{\text{OUT MOD SEL}}$ signal is generated.

Port Decoding

Once the proper module has been selected, as discussed in the previous subsection, an additional selection must be made: that of one of the eight output ports which are on each imm8-63 Card. This function is performed by the Port Selection circuits, shown in detail in Figure 4-2.

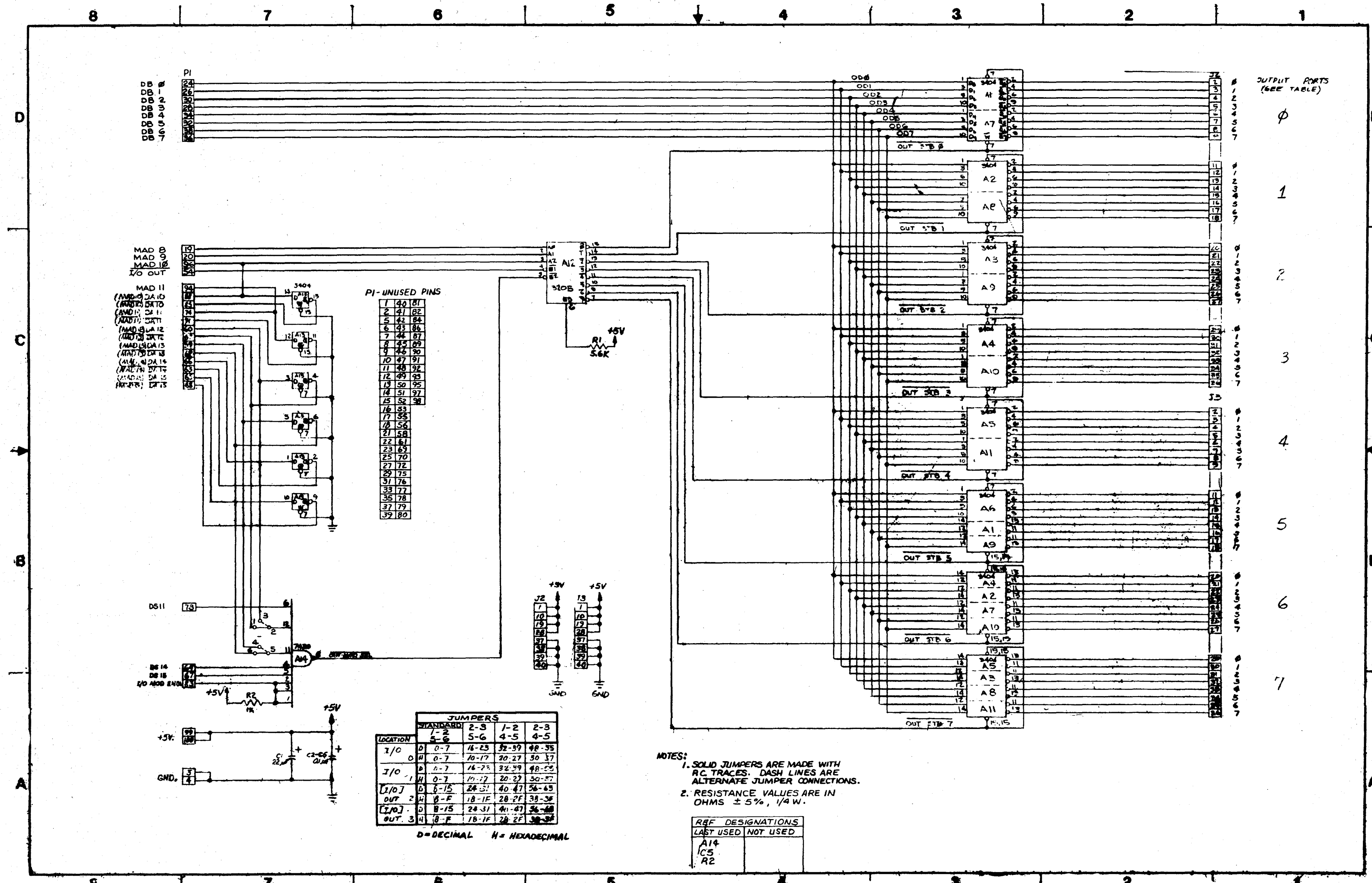


Figure 4-2. Output Module Schematic Diagram

In order to select one of the eight output ports, three data lines are led to the Port Decoder. When enabled by the $\overline{\text{OUT MOD SEL}}$ signal, the Port Decoder will decode the three incoming Port Select signals and will issue an enabling signal to one of the eight output ports.

Output Operations

In a typical output operation, the following steps will be executed (refer to Figure 4-2, the Schematic Diagram):

1) The Central Processor sends an I/O Address to the imm8-63 Module on lines MAD8-15.
2) The Module Decoding and Port Decoding circuits decode the incoming I/O Address, and send enabling signal $\overline{\text{OUT STB}}$ to the proper output port.
3) The Central Processor sends the data which are to be output to the imm8-63 Card, along with an Output enabling signal, $\overline{\text{I/O OUT}}$. $\overline{\text{I/O OUT}}$ activates the internal signal $\overline{\text{OUT STB}}$.
4) The data which have been sent to the imm8-63 Card are latched into the proper output port by signal OUT STB, where they are held for use by external equipment. The data are held until another output operation using the selected port takes place, at which time they are replaced by the new incoming data.

The timing of the output operation is shown in Figure 4-3.

CARD UTILIZATION

The user has the capability of choosing which eight addresses an imm8-63 will respond to; the user can assign any group of addresses between 0 and 63. This section describes how to use the addressing option, and also supplies a complete list of the imm8-63 card edge pins and their associated signals.

User Options

A combination of card positioning and jumper connections (on the Output Modules) determines which eight addresses each module will recognize. Table 4-1 lists the possible combinations of card position and jumper connections, and the address groups associated with each combination.

imm8-63 Addressing Options

Card Position	Jumpers			
	1-2 5-6	2-3 5-6	1-2 4-5	2-3 4-5
0	0-7	16-23	32-39	48-55
1	0-7	16-23	32-39	48-55
2	8-15	24-31	40-47	56-63
3	8-15	24-31	40-47	56-63

Table 4-1

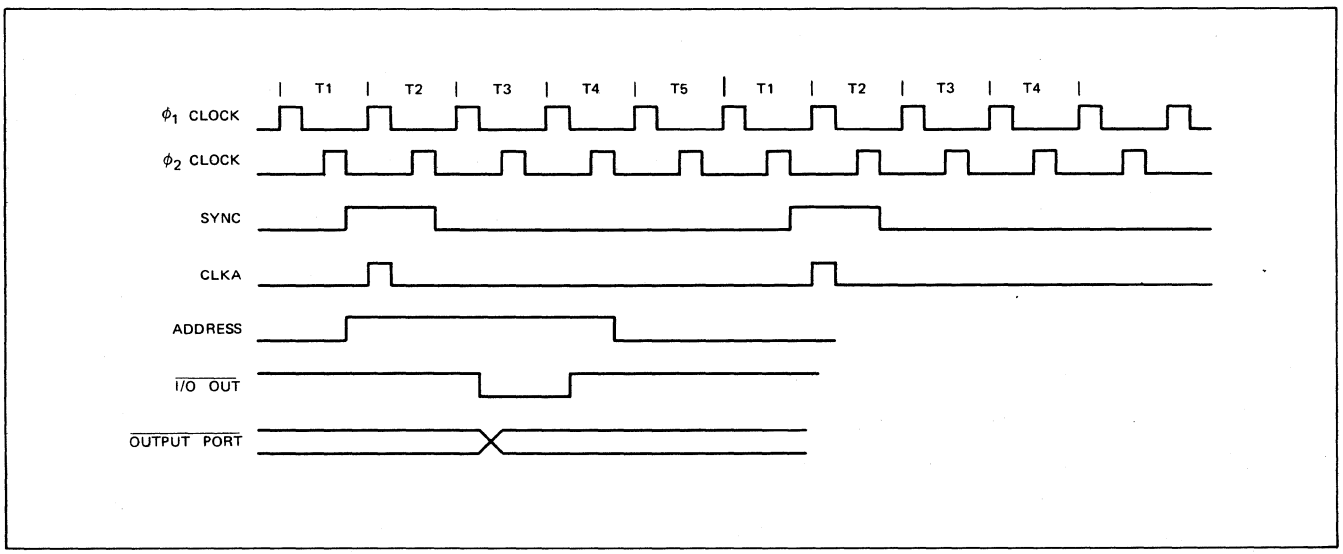


Figure 4-3. Output Module Timing

CHAPTER 5 THE imm6-28 RANDOM ACCESS MEMORY CARD

The imm6-28 Random Access Memory Card has been designed to provide a user with a 4,096 (4K) 8-bit words of random-access memory, which may be used as a computer system's memory device.

More than one imm6-28 card may be included in a system, for example, the imm8-83 Central Processor card can address up to 16,384 words of memory on four separate imm6-28 cards.

Although the imm6-28 Random Access Memory Card has been designed to support the Intel imm8-83 Central Processor Card, it can be used in any other system which requires 4K x 8 bits of RAM storage.

THE imm6-28 RANDOM ACCESS MEMORY CARD–GENERAL FUNCTIONAL DESCRIPTION

This section describes the operation of the imm6-28 Random Access Memory Card in general functional terms, and is divided into four subsections.

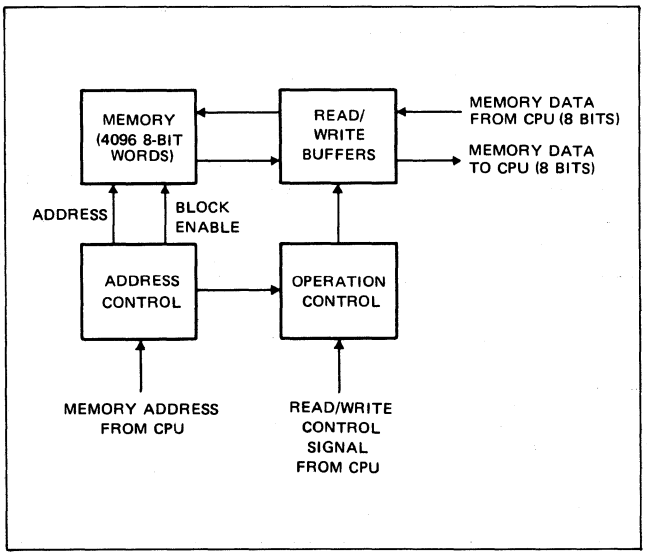


Figure 5-1. RAM Module Functional Block Diagram

The Four Functional Units

In order to describe its operation, the imm6-28 card has been divided into four functional units:

1) The *Address Control Block*, which determines which card is to be used for a memory operation, and which memory location on that card is being addressed.
2) The *Operation Control Block*, which controls the execution of all operations performed by the card.
3) The *Read/Write Buffers*, which buffer the data which is read from or written into memory.
4) The *Memory Block*, which contains the actual memory components.

Each operation performed by the imm6-28 card uses at least one of these functional units.

A block diagram of the imm6-28 card, showing the four functional units and their interrelationship, is given in figure 5-1, and should be referred to when reading the rest of this section.

Memory Addressing Operations

In order to send data to a memory location, or to read data from a location, it is necessary to specify the location which is to be accessed. This function is provided by the memory Address, a group of signals which represent the Central Processor. Once the Memory Address is received to select the correct location for a Memory Read or Write operation.

The Address Control Block performs Memory Address decoding on the imm6-28 card; it receives the Memory Address, and translates it into three types of signals: Module Enabling signals, which enable the selected 4K segment of the memory; Block Enabling signals, which enable one 1024 word block within the larger 4K segment; and Address signals, which access one word within the 1024 word block.

Memory Write Operations

A Memory Write Operation is executed in order to load data into a selected memory word; it is executed in the following steps:

1) The Memory Address for the word which is to be written into is sent to the imm6-28 card by the Central Processor.
2) The Address Control Block receives the Memory Address and generates the signals necessary to access the addressed memory location.
3) The Central Processor sends a data word to the imm6-28 card, where it is received by the Read/Write Buffer. The central Processor also sends control signals to the Operation Control Block which indicate a Memory Write operation.
4) The Operation Control Block generates signals which cause data in the Read/Write Buffer to be written into the selected memory location in the Memory Block.

Memory Read Operations

A Memory Read operation is performed in order to read data from a selected memory location into the Central Processor; it is executed via the following steps:

1) The Memory Address which is to be read is sent to the imm6-28 card by the Central Processor.
2) The Address Control Block receives the Memory Address and generates signals necessary to access the addressed memory location.
3) The Central Processor sends control signals to the to the Operation Control Block which indicate a Memory Read operation.
4) The Operation Control Block generates the control signals necessary to cause the contents of the selected memory location to be sent from the Memory Block to the Read/Write Buffer, whence they are sent on to the Central Processor.

THE imm6-28 RANDOM ACCESS MEMORY CARD—THEORY OF OPERATION

This section describes the theory of operation of the imm6-28 card in detail giving the circuit-level implementation of the features.

Physical Memory Implementation

The actual memory of the imm6-28 card is made up of thirty-two Intel 2102 Random Access Memory chips, each having a capacity of 1024 one bit words. Since the data word used by the imm6-28 card has a total of eight bits, the 2102 memory chips are tied together in blocks of eight, with each of the eight chips in the block handling one of the eight data bits; this results in a basic block of 1024 eight-bit words. Since there are four blocks per card, each imm6-28 card has a capacity of 4096 eight-bit words.

By combining more than one card in a system, memory size can be increased in increments of 4096 words.

Memory Address Decoding

Since more than 4096 words of memory can be addressed by a Central Processor, the imm6-28 card includes address decoding circuits (see Figure 5-2) which allows a Central Processor to select one imm6-28 memory card.

The Memory Address which the Central Processor sends to the imm6-28 cards consists of sixteen bits of information, organized as a sixteen digit binary number, with the

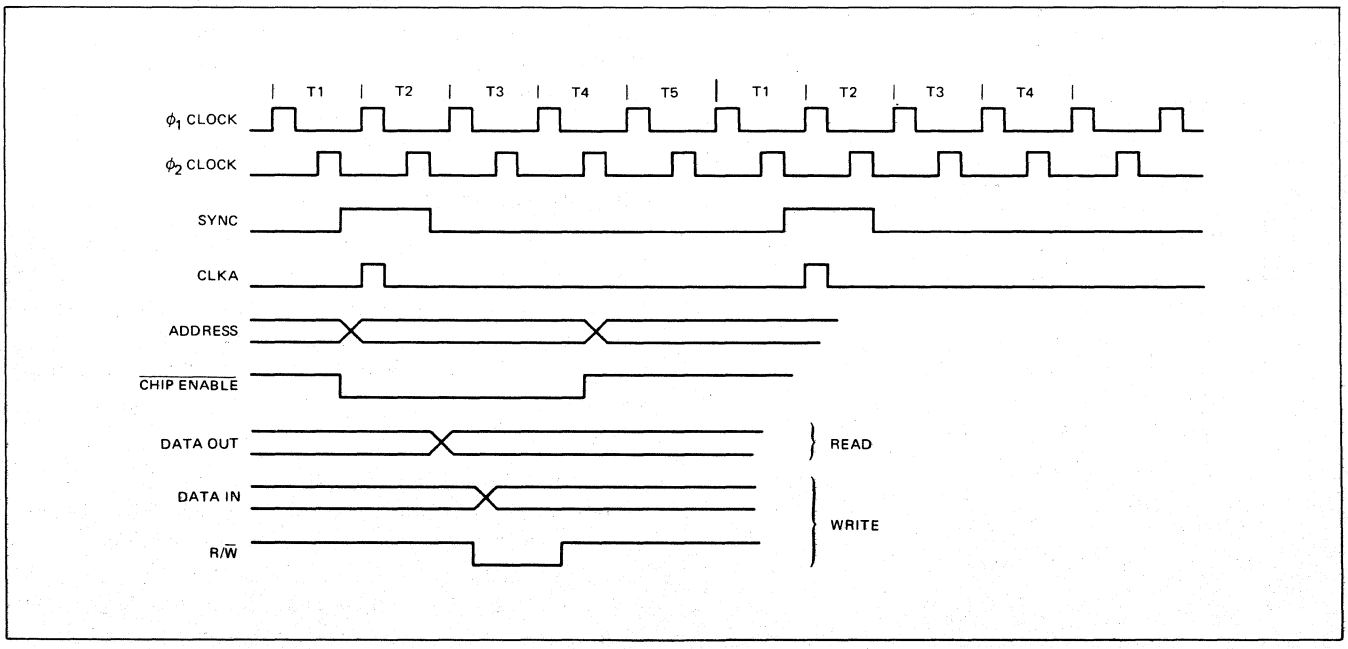


Figure 5-2. RAM Memory Module Timing

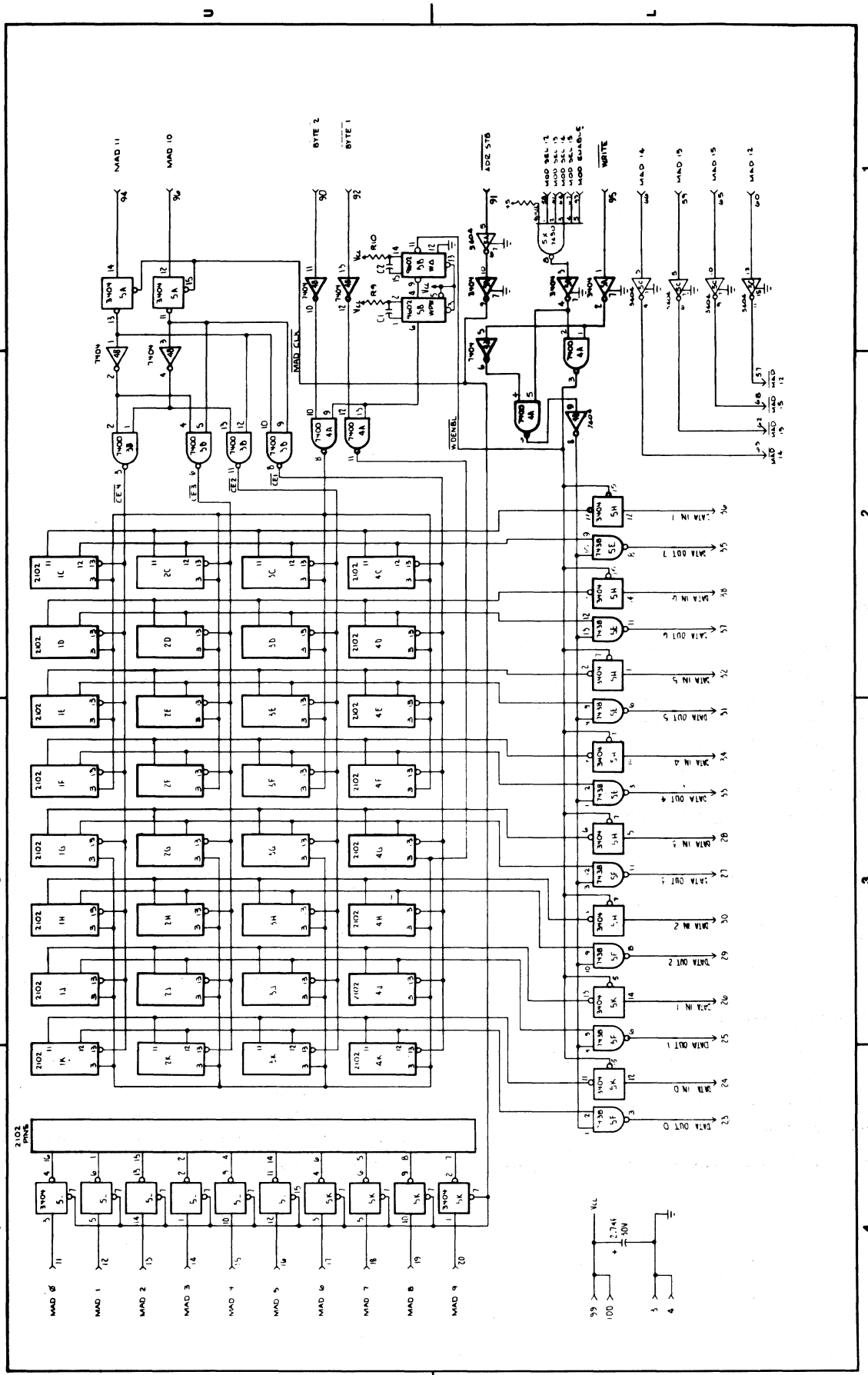


Figure 5-3. RAM Memory Module Schematic Diagram

low order bit on line MAD0 and the highest order bit on line MAD15. The Address Decoding Circuits use this sixteen-bit address as follows:

1) Since the high-order four bits of the Memory Address effectively divide the possible memory locations into sixteen units of 4096 words each, they are used to enable the particular card which is to be used for a given memory operation. This is accomplished by bringing lines MAD12-MAD15 onto the imm6-28 card edge pins, inverting them to form $\overline{\text{MAD12}}$-$\overline{\text{MAD15}}$, and then sending these inverted Memory Address signals out on another set of card edge pins. External jumpers are then used to tie the proper combination of Memory Address and inverted Memory Address signals to the four input lines to the Access Enable Gate, MOD SEL 12-MOD SEL 15. When the proper Memory Address is sent to the imm6-28 card by the Central Processor, the Access Enable Gate will produce a Module Enable signal which is used to enable all memory operations for that card.
2) The next two bits of the Memory Address, MAD10 and MAD11, select one of the four 1024 word Latches which are enabled by the Access Enable Gate's Module Enable signal. The two signals are then latched into the Address Latches by signal $\overline{\text{ADR STB}}$, sent by the Central Processor, and are sent to a group of four NAND gates in both their original and their inverted form. The four NAND gates decode the two Memory Address bits into one of four Chip Enable signals. The Chip Enable signals are used to enable the proper block of eight chips (1024 eight-bit words) out of the four blocks available on each imm6-28 card.
3) The ten low-order bits of the Memory Address, MAD0-MAD9, are tied to Address Latches which are enabled by the Access Enable Gates. They are then sent to all of the individual memory chips, which use them to enable the proper location out of the 1024 available.

Memory Read Operations

A Memory Read operations is initiated by the Central Processor. It sends a sixteen-bit Memory Address to the imm6-28 card, which decodes the address to select one particular memory location.

The Central Processor also sends signal $\overline{\text{Write}}$/Read to the imm6-28 card. In its TRUE state, this signal indicates a Write operation, therefore, during a Read operation, it will be FALSE. Signal $\overline{\text{Write}}$/Read is inverted and applied to a NAND gate along with the Module Enable signal. The NAND gate produces a signal which indicates a Read operation. The Read operation signal is used as the second input to the series of Output Buffer NAND gates, and causes the memory data to be gated through the Output Buffer NAND gates and onto the Data Out lines DATA OUT0-DATA OUT7. Timing is shown in Figure 5-2.

Memory Write Operations

A Memory Write operation is initiated by the Central Processor. It sends a sixteen bit Memory Address to the imm6-28 card, which decodes the address to select one particular memory location for access, as described in Section 6.2.2. When the memory chips receive the Memory Address, they immediately respond by sending the contents of the addressed location to the Output Buffers, which are series of eight NAND gates.

The Central Processor then sends the data which is to be written into memory to the imm6-28 card, where it is led to the Input Latches. The Central Processor also sends out signal $\overline{\text{Write}}$/Read, which indicates a Write operation. This signal is NANDed with the Module Enable signal to produce signal $\overline{\text{WDENBL}}$, which indicates that a Write operation is taking place. This signal causes the data sent by the Central Processor to be latched into the Input Latches.

Signal WDENBL is also used to trigger a pair of one-shot multivibrators. These multivibrators produce a delayed Write Enable signal. The delay is necessary to ensure that the delayed Write Enable signal becomes TRUE, the data will be written into the selected memory location.

THE imm6-28 RANDOM ACCESS MEMORY CARD – UTILIZATION

This section provides the information necessary to efficiently use the imm6-28 card in an application. In particular, the requirements for interfacing with the Intel imm8-83 Central Processor Card are stressed.

Memory Address Coding

In order to enable Memory operations, the imm6-28 card must have an encoded address designation. The proper positioning of the external jumpers for each block of memory is as follows:

Module No.	Memory Addresses	Memory Address Code				Jumpers			
RAM 0	0-4095	$\overline{MAD12}$	$\overline{MAD13}$	$\overline{MAD14}$	$\overline{MAD15}$	57-58,	62-61,	63-63,	67-68
RAM 1	4096-8191	MAD12	$\overline{MAD13}$	$\overline{MAD14}$	$\overline{MAD15}$	58-60,	62-61,	63-64,	67-68
RAM 2	8192-12287	$\overline{MAD12}$	MAD13	$\overline{MAD14}$	$\overline{MAD15}$	57-58,	59-61,	63-64,	67-68
RAM 3	12288-16383	MAD12	MAD13	$\overline{MAD14}$	$\overline{MAD15}$	58-60,	59-61,	63-64,	67-68
RAM 4	16384-20479	$\overline{MAD12}$	$\overline{MAD13}$	MAD14	$\overline{MAD15}$	57-58,	62-61,	64-66,	67-68
RAM 5	20480-24575	MAD12	$\overline{MAD13}$	MAD14	$\overline{MAD15}$	58-60,	62-61,	64-66,	67-68
RAM 6	24576-28671	$\overline{MAD12}$	MAD13	MAD14	$\overline{MAD15}$	57-58,	59-61,	64-66,	67-68
RAM 7	28672-32767	MAD12	MAD13	MAD14	$\overline{MAD15}$	58-60,	59-61,	64-66,	67-68
RAM 8	32768-36863	$\overline{MAD12}$	$\overline{MAD13}$	$\overline{MAD14}$	MAD15	57-58,	62-61,	63-64,	65-67
RAM 9	36864-40959	MAD12	$\overline{MAD13}$	$\overline{MAD14}$	MAD15	58-60,	62-61,	63-64,	65-67
RAM 10	40960-45055	$\overline{MAD12}$	MAD13	$\overline{MAD14}$	MAD15	57-58,	59-61,	63-64,	65-68
RAM 11	45056-49151	MAD12	MAD13	$\overline{MAD14}$	MAD15	58-60,	59-61,	63-64,	65-67
RAM 12	49152-53247	$\overline{MAD12}$	$\overline{MAD13}$	MAD14	MAD15	57-58,	62-61,	64-66,	65-67
RAM 13	53248-57343	MAD12	$\overline{MAD13}$	MAD14	MAD15	58-60,	62-61,	64-66,	65-67
RAM 14	57344-61439	$\overline{MAD12}$	MAD13	MAD14	MAD15	57-68,	59-61,	64-66,	65-67
RAM 15	61440-65535	MAD12	MAD13	MAD14	MAD15	58-60,	59-61,	64-66,	65-67

Installation Data and Requirements

Connector: Dual 50-pin, .125 in. centers
Input Voltage: +5v± 5% @ 2.5A.
Operating Temperature: 0° C-70° C

CHAPTER 6 THE imm6-26 PROGRAMMABLE READ-ONLY MEMORY CARD

The imm6-26 Programmable Read-Only Memory (PROM) Card has been designed to provide a user with 4,096 (4K) words of read-only memory, which may be used as non-volatile program or data storage.

The imm6-26 Card uses Intel 8702A Programmable Read-Only Memory chips as its storage medium. These chips represent a considerable advance in the field of read-only memory, as they can be erased and reprogrammed as the need arises. This capability makes the imm6-26 Card a valuable addition to a system in which the stored data is occasionally subject to change, for example, during the development of mask-programmed read-only memory. The imm6-26 PROM Card can be used to store programs in final stages of correction, before the program is well enough defined to justify the expense of creating masks. Also, the imm6-26 PROM Card can be used instead of read-only memory in pre-production equipment that may have to be shipped before mask-programmed read-only memory is available.

More than one imm6-26 Card may be used in a system. For example, the imm8-83 Central Processor Card can address up to 16,384 words of memory on four separate imm6-26 cards.

The imm6-26 Card may also be used in parallel with an imm6-28 Random Access Memory Card.

Note: When used in conjunction with the imm8-83 the 8702A type used must have an access time of less than 1.5 microsecond.

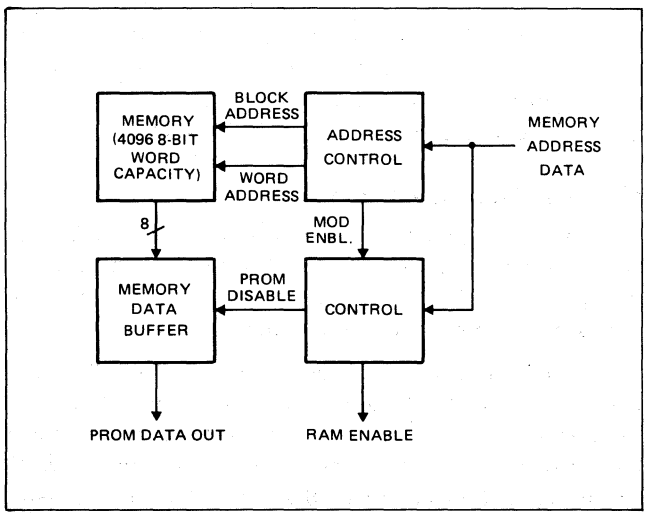


Figure 6-1. PROM Memory Module Functional Block Diagram

THE imm6-26 RANDOM ACCESS MEMORY CARD-GENERAL FUNCTIONAL DESCRIPTION

This section describes the operation of the imm6-26 Programmable Read-Only Memory Card in general functional terms, and is divided into three subsections.

The Four Functional Units

In order to describe its operation, the imm6-26 Card had been divided into four functional units:

1) The Address Control Block, which determines which card is to be used for a memory operation, and which memory location on that card is being addressed.
2) The Operation Control Block, which controls the execution of all operations performed by the card.
3) The Memory Data Buffer, which buffers the data being read from memory.
4) The Memory Block, which contains the actual memory components.

A block diagram of the imm6-26 Card, showing the four functional units and their interrelationship, is given in Figure 6-1, and should be referred to when reading the rest of this section.

Memory Read Operation

In order to obtain data from a memory location, it is necessary to perform a Memory Read operation. This operation can be divided into two phases:

1) The Addressing Phase, in which the desired memory address is sent to the imm6-26 Card, where it is decoded and used to enable the specific memory location which is to be accessed.

2) The Data Phase, where the data is sent out from the imm6-26 Card.

The Addressing Phase is executed in the following steps:

a) The Central Processor sends a Memory Address to the imm6-26 Card Address Control Block.

b) The Address Control Block translates the Memory Address into three types of signals: Module Enabling signals, which enable the selected 4K segment of the memory; Block enabling signals, which enable one 256 word block within the larger 4K segment; and Address signals, which access one word within the 256 word block.

c) The Control Block checks the selected memory address, and determines if it exists on the imm6-26 Card. If it finds that it does not exist, it sends out disabling signals which prevent further operations with the imm6-26 Card. At the same time, it sends out an enabling signal which can be used by an imm6-28 Random Access Memory Card to enable its operation.

The Operation Control Block generates the control signals necessary to cause the contents of the selected memory location to be sent from the Memory Block to the Memory Data Buffers, whence they are sent out to the Central Processor.

THE imm6-26 PROGRAMMABLE READ-ONLY MEMORY CARD-THEORY OF OPERATION

This section describes the theory of operation of the imm6-26 Card in detail, giving the circuit-level implementation of the features.

Physical Memory Implementation

The actual memory of the imm6-26 Card is made up to sixteen Intel 8702A Erasable Programmable Read-Only Memory chips, each having a capacity of 256 eight-bit words. This results in a basic memory block of 256 words. Each 256 word block is a separate unit, and can be changed by removing the existing PROM chip and installing a new PROM, or omitted by removing the existing PROM without replacement. NOTE: Only standard 8702A PROMs can be used with the INTELLEC 8/MOD 80 system; all 8702A PROMs must have access time less than or equal to 1.5 microsecond.

Since there are sixteen 256 word PROMs on each imm6-26 Card, each card has a total capacity of 4,096 words. By combining more than one card in a system, memory size can be increased in increments of 256 words.

Memory Address Decoding

Since more than 4,096 words of memory can be addressed by a Central Processor, the imm6-26 card includes address decoding circuits which allow a Central Processor to select one imm6-26 memory card.

The Memory Address which the Central Processor sends to the imm6-26 card consists of sixteen bits of information, organized as a binary number, with the low order bit on line MAD0 and the high order bit on line MAD15. The Address Decoding circuits use this sixteen-bit address as follows:

1) Since the high order four bits of the Memory Address effectively divide the possible memory locations into sixteen units of 4,096 words each, they are used to enable the particular card which is to be used for a given memory operation. This is accomplished by bringing lines MAD12-MAD15 onto the imm6-26 card edge pins, inverting them to form MAD12-MAD15, and then sending these inverted memory Address signals out on another set of card edge pins. External jumpers are then used to tie the proper combination of Memory Address and inverted Memory Address signals to the four inputs to the Access Enable Gate, MS12-MS15. When the proper Memory Address is sent to the imm6-26 card by the Central Processor, the Access Enable Gate will produce a Module Enable signal which is used to enable memory operations for that card.

2) The next four bits of the Memory Address, MAD8-MAD11, select one of the sixteen 256 word blocks. These two signals are led to two three-to-eight line decoders. Signal MAD11 is then used to enable one of the two decoders, while MAD8-MAD10 are used as inputs to the decoders. The decoders produce Chip Enable signals which are used to enable one of the sixteen 256 word PROM chips on the imm6-26 card.

3) The eight low-order bits of the Memory Address, MAD0-MAD7, are tied to Address Latches which are enabled by the Module Enable Access Enable Gate. They are then sent to all of the available memory chips, which use them to enable the proper location out of the 256 available.

Memory Read Operations

A Memory Read operation is initiated by the Central Processor, which sends a sixteen bit Memory Address to the imm6-26 card. The address decoding circuits decode the address to select one particular memory location.

The Central Processor also sends signal PROM MOD ENBL to the imm6-26 card, enabling operations from that card. This signal is used as an input to the Module Enable Gate along with the Access Enable Gate signal MOD DECODE, as shown in Figure 6-3. When all of the inputs to the Module Enable Gate are TRUE, it generates the PROM

MOD SEL signal, which is sent to the two low-order Address Decoders. It enables the decoders, and the proper chip is enabled. The chip reads the low-order eight bits of the Memory Address, and sends the data contained in the selected memory location to the Memory Data buffers on lines D0-D7. The Memory Buffers are also enabled by the PROM MOD SEL signal, and will gate the data onto the Memory Data Out lines MD10-MD17. Timing is shown in Figure 6-2.

Random Access Enable

Since it may be desired to mix Random Access and Read-Only memories in a system, the imm6-26 card has been designed to determine, for each memory operation, whether or not PROM memory exists for the selected Memory Address. If PROM memory does not exist for that location, the imm6-26 card will generate an enabling signal for Random Access memory which uses the same address. If the two types of memories share common locations, however, the Random Access enabling signal will not be issued, giving the PROM memory priority.

Each PROM location on the imm6-26 card has a corresponding switch which is tied to one input of an eight input multiplexer. In its normal position, this switch, and thus its associated multiplexer input, is tied to +5v. When a PROM is installed on the card, its corresponding switch is depressed, causing the input to the multiplexer to be tied to GROUND. When a memory operation is executed, the four Memory Address lines MAD8-MAD11, which are used by the address decoding circuits to generate chip enable signals, are used as addressing inputs to the multiplexer. If a PROM exists at the addressed location, the multiplexer output will be HIGH. This output is led to the PROM Resident Latch, which produces the PROM RESIDENT signal. This signal is used as an enabling signal to the Module Enable Gate, and thus enables PROM operations when there is a PROM present. Likewise, if there is no PROM present in the addressed location, the output of the multiplexer will be LOW, the PROM RESIDENT signal will be FALSE, the Module Enable Gate output will be FALSE, and imm6-26 operations will be disabled.

When the Module Enable Gate output signal, PROM MOD SEL, is FALSE, signal RAM MOD ENBL is produced by the RAM Module Enable Latch. This signal may be used to enable a Random Access memory device which has the same address as the PROM module.

THE imm6-26 PROGRAMMABLE READ-ONLY MEMORY CARD – UTILIZATION

This section provides the information necessary to efficiently use the imm6-26 card in an application.

Memory Address Coding

In order to enable memory operations, the imm6-26 card must have an encoded address signation. The proper positioning of external jumpers for each block of memory as follows:

Module No.	Memory Addresses	Card Select Coding				Jumper Pin Connections			
PROM 0	0-4095	$\overline{\text{MAD12}}$,	$\overline{\text{MAD13}}$,	$\overline{\text{MAD14}}$,	$\overline{\text{MAD15}}$	57-58,	61-62,	63-64,	67-68
PROM 1	4096-8191	MAD12,	$\overline{\text{MAD13}}$,	$\overline{\text{MAD14}}$,	$\overline{\text{MAD15}}$	58-60,	61-62,	63-64,	67-68
PROM 2	8192-12287	$\overline{\text{MAD12}}$,	MAD13,	$\overline{\text{MAD14}}$,	$\overline{\text{MAD15}}$	57-58,	59-61,	63-64,	67-68
PROM 3	12288-16383	MAD12,	MAD13,	$\overline{\text{MAD14}}$,	$\overline{\text{MAD15}}$	58-60,	59-61,	63-64,	67-68
PROM 4	16384-20479	$\overline{\text{MAD12}}$,	$\overline{\text{MAD13}}$,	MAD14,	$\overline{\text{MAD15}}$	57-58,	61-62,	64-66,	67-68
PROM 5	20480-24575	MAD12,	$\overline{\text{MAD13}}$,	MAD14,	$\overline{\text{MAD15}}$	58-60,	61-62,	64-66,	67-68
PROM 6	24576-28671	$\overline{\text{MAD12}}$,	MAD13,	MAD14,	$\overline{\text{MAD15}}$	57-58,	59-61,	64-66,	67-68
PROM 7	28672-32767	MAD12,	MAD13,	MAD14,	$\overline{\text{MAD15}}$	58-60,	59-61,	64-66,	67-68
PROM 8	32768-36863	$\overline{\text{MAD12}}$,	$\overline{\text{MAD13}}$,	$\overline{\text{MAD14}}$,	MAD15	57-58,	61-62,	63-64,	65-67
PROM 9	36864-40959	MAD12,	$\overline{\text{MAD13}}$,	$\overline{\text{MAD14}}$,	MAD15	58-60,	61-62,	63-64,	65-67
PROM 10	40960-45055	$\overline{\text{MAD12}}$,	MAD13,	$\overline{\text{MAD14}}$,	MAD15	57-58,	59-61,	63-64,	65-67
PROM 11	45056-49151	MAD12,	MAD13,	$\overline{\text{MAD14}}$,	MAD15	58-60,	59-61,	63-64,	65-67
PROM 12	49152-53247	$\overline{\text{MAD12}}$,	$\overline{\text{MAD13}}$,	MAD14,	MAD15	57-58,	61-62,	64-66,	65-67
PROM 13	53248-57343	MAD12,	$\overline{\text{MAD13}}$,	MAD14,	MAD15	58-60,	61-62,	64-66,	65-67
PROM 14	57344-61439	$\overline{\text{MAD12}}$,	MAD13,	MAD14,	MAD15	57-58,	59-61,	64-66,	65-67
PROM 15	61440-65535	MAD12,	MAD13,	MAD14,	MAD15	58-60,	59-61,	64-66,	65-67

PROM Installation, Removal, Programming, and Erasure

In order to provide flexibility in memory assignment, the imm6-26 card can be of any size desired, from 256 words to 4,096 words, in 256 word increments. This flexibility is achieved by enabling installation and removal of the individual PROM chips which make up the imm6-26 card's memory.

When installing PROM chips on the imm6-26 card, the corresponding PROM Resident switch *must* be depressed. If this is not done, the imm6-26 card will not be enabled when that group of memory addresses is accessed. To install a PROM, merely insert it into the socket provided on the imm6-26 card. Likewise, to remove a PROM, merely pull it from the socket. Again, if removing a PROM, ensure that the corresponding switch is disabled. If this is not done, faulty memory operations will ensue. If all of the sixteen PROMs are installed on an imm6-26 card, the PROM *Resident* signal can be permanently enabled by installing the ALL RPOMS RESIDENT patch between points 1 and 2, as shown in Figure 6-3.

The Intel 8702A PROMs used by the imm6-26 card may be programmed by using the imm6-76 PROM Programmer card in conjunction with the Intellec 8 system. They may be erased by exposing them to high intensity short-wave ultraviolet light at a wavelength of 2537Å. After ten minutes of such exposure, the PROM will be erased to all zeroes. No more exposure than is necessary should be used, to avoid damaging the PROM. (See the Intel Memory Design Handbook for more information regarding 8702A PROM programming and erasure). CAUTION: When using an ultraviolet source to erase the PROM, be careful not to expose your skin or eyes to the ultraviolet rays because of the damage which these rays can cause. In addition, short-wavelength ultraviolet light generates considerable amounts of ozone, which is also potentially hazardous.

Note: When used in conjunction with an imm8-83 module the 8702A type used must have an access time of less than 1.5 μsec.

Installation Data and Requirements

Connector: Dual 50-pin, .125 in. centers
Input Voltage: +5v ± 5% @ 1.6A (max)
Operating Temperature: 0° C-70° C

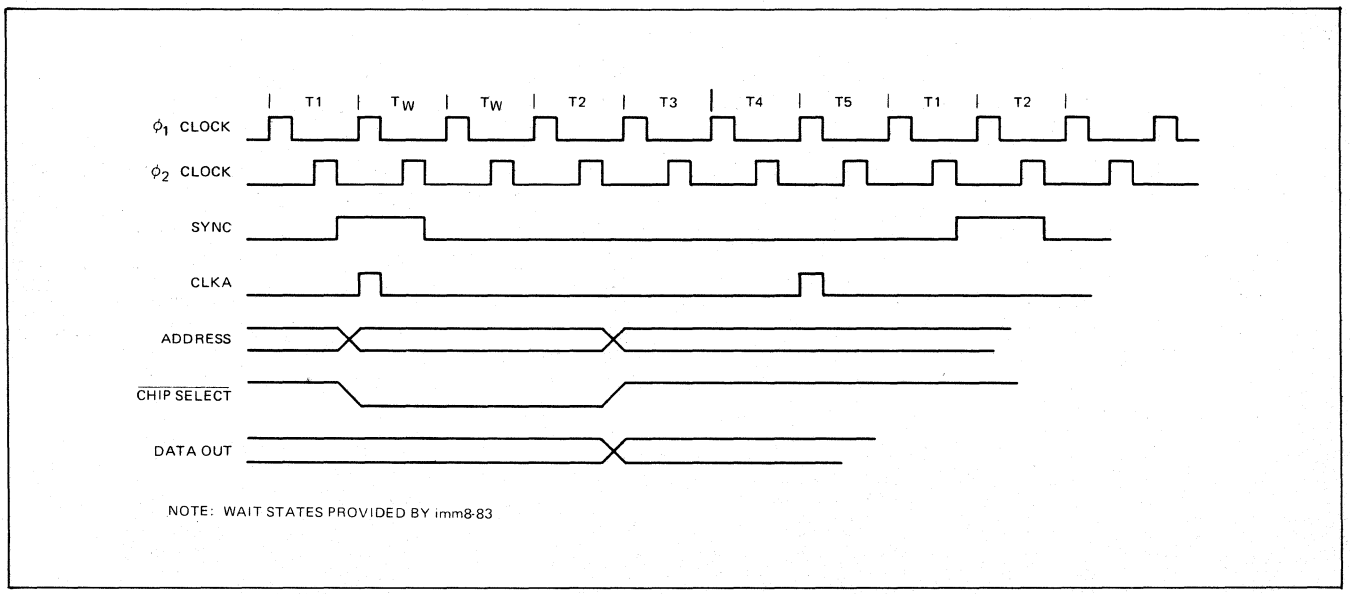


Figure 6-2. PROM Memory Module Timing

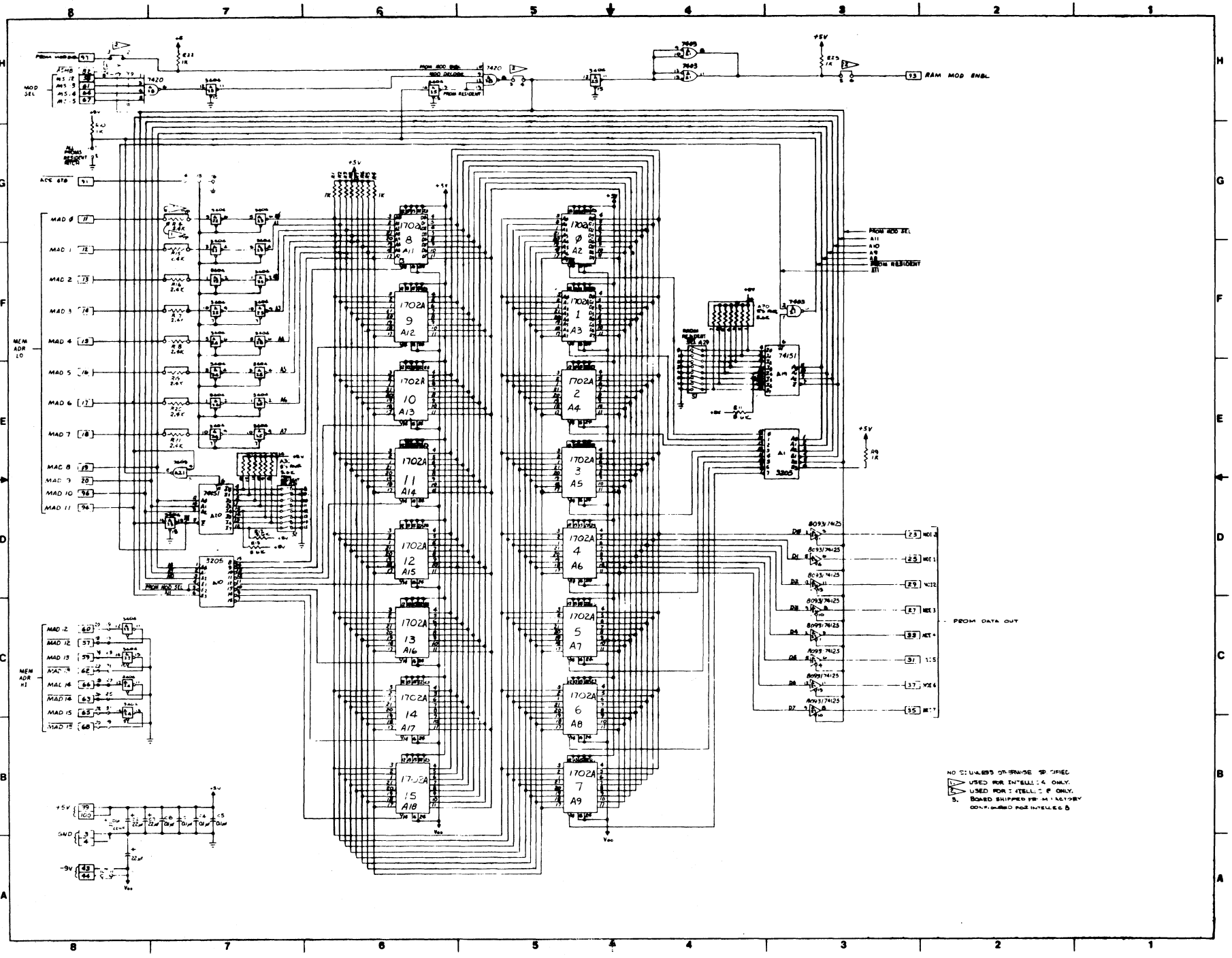


Figure 6-3. PROM Memory Module Schematic Diagram

CHAPTER 7 THE INTELLEC 8/MOD 80 CONTROL CONSOLE

The INTELLEC 8/MOD 80 Control Console is designed to provide a user of the INTELLEC 8/MOD 80 microcomputer development system with an easy to use means of monitoring and controlling machine operation, manually moving data to or from memory or input/output devices, and running or debugging programs. Since the INTELLEC 8 System is specifically designed for microcomputer systems development, the Control Console has several features which are not usually found on "traditional" computer control consoles, e.g., extensive status displays and special debugging aids.

This section describes the operation of the INTELLEC 8/MOD 80 Control Console on two levels: first, on a general functional level, second, on a more detailed theory of operation level.

Since the INTELLEC 8/MOD 80 Control Console has been designed to support the imm8-83 Central Processor Card, many of its operations cannot be described without referring to the operation of that card. It is an absolute necessity, therefore, that Chapter 2 of this manual be read and fully understood before attempting to read this section, as it is in Chapter 2 that many of the basic concepts necessary for a proper understanding of Control Console operation are developed. If a more detailed description of operational procedures using the Control Console is desired, refer to the INTELLEC 8/MOD 80 Operator's Manual.

THE INTELLEC 8/MOD 80 CONTROL CONSOLE — FUNCTIONAL DESCRIPTION

This section provides a basic, functional overview of INTELLEC 8/MOD 80 Control Console operation. The operations performed by the Control Console can be divided into seven groups, as follows:

1) Data display operations, including:

   Memory Data display operations, in which the contents of a selected memory location are displayed;

   I/O Data display operations, in which data used for an input or output operation is displayed;

   Status display operations, which display indications of the operating mode of the Central Processor;

   Cycle display operations, which provide a continuous display of the 8080 machine cycle;

   Programmable display operations, in which the contents of output port FF_{16} are displayed.

2) Manual Memory Access operations, in which data is read from or written into a selected memory location from the Control Console rather than the Central Processor.

3) Manual I/O Access operations, in which input monitoring or output operation is performed from the Control Console rather than the Central Processor.

4) Interrupt operations, in which an interrupt cycle is initiated from the Control Console by the user.

5) Processor Control operations, which allow the user to directly control the operation of the Central Processor.

6) Sense operations, which allow the user to manually enter data during a programmed input operation.

7) Search/Wait operations, which allow a selected instruction to be executed a given number of times, after which the Central Processor enters a WAIT mode.

Each of these operational groups is discussed in a separate subsection of this chapter.

Data Display Operations

The INTELLEC 8/MOD 80 Control Console can perform five distinct data display operations.

- Status Display
- Cycle Display
- Address Display
- Instruction/Data Display
- Programmable Display

The Status display functions provide a visual indication of the Processor's mode of operation. There exist eight status display functions:

- Run
- Wait
- Halt
- Hold
- Search Complete
- Access Request
- Interrupt Request
- Interrupt Disable

The eight functions are performed in the following manner:

1) The RUN status display is lit whenever the Central Processor is not waiting or stopped.
2) The WAIT status display is lit whenever the Processor is in a WAIT state (i.e., waiting for data to be input).
3) The HALT status display is lit whenever the Processor is in a STOPPED state.
4) The HOLD status display is lit whenever the Processor has acknowledged a Hold Request (as for a direct memory or I/O access operation).
5) The SEARCH COMPLETE status display is lit whenever a Search/Wait operation has been completed, and the passcounter has been counted down to zero.
6) The ACCESS REQUEST display is lit whenever a Direct Memory or I/O Access request has been made by depressing the Console Mem Access or I/O Access switches.
7) The INTERRUPT REQUEST display is lit whenever an Interrupt Request has been made via the Control Console Interrupt or Reset switches, and is extinguished when the Processor acknowledges the interrupt request.
8) The INTERRUPT DISABLE display is lit whenever the processor has disabled its interrupt capability.

The cycle display functions provide a visual indication of the Processor machine state. There are eight cycle display functions:

- Fetch
- Memory
- I/O
- DA
- Read/Input
- Write/Output
- Interrupt
- Stack

The eight cycle functions operate as follows:

1) The FETCH cycle display is lit when the processor is executing an Instruction Fetch operation.
2) The MEM cycle display is lit when the processor or the Control Console is executing a Memory Access operation.
3) The I/O cycle display is lit when the processor or the Control Console is executing an I/O Access operation.
4) The DA cycle display is lit when a Memory or I/O Access operation is being performed from the Control Console rather than by the processor.
5) The Read/Input cycle display is lit when either a Memory Read or I/O Input operation is executed.
6) The Write/Output cycle display is lit when either a Memory Write or I/O Output operation is executed.
7) The INT cycle display is lit when a processor Interrupt cycle is in progress.
8) The STACK cycle display is lit when the processor is accessing the stack.

The Address display function provides a visual display of the address data used for a Memory or I/O operation. There are sixteen address display lights, corresponding to the sixteen address lines.

The Address display function is performed by tying the processor memory address lines to the display lights through a series of buffers.

The Instruction/Data display provides a visual indication of the instruction or data fetched from memory or the data which is read from memory or an I/O device. There are eight Instruction/Data display lights, tied to the processor data bus.

The Programmable display function provides an indication of the contents of output port FF_{16}.

Manual Memory Access Operations

A Manual Memory Access operation is performed in order to read or write data to or from memory. It is accomplished via the following steps:

1) The Mem Access switch on the Control Console is depressed, sending a control signal to the processor, which enters the HOLD state.
2) The memory address to be accessed is loaded into the Address/Instruction/Data switches on the Control Console.
3) The LOAD switch on the Control Console is depressed, loading the Address/Instruction/Data data into the Address Register.

4) The address held in the Address Register is sent to the memory module on the memory address bus.

5) The memory module responds by sending the data currently held in the selected memory location to the Control Console, where it is displayed by the Instruction/Data display.

6) If it is desired to write data into memory, the data byte to be written is loaded into the lower eight Address/Instruction/Data switches. Switch DEP is then depressed, sending a control signal to the memory module which causes the switch data to be loaded into the memory address held by the Address Register.

Note: The deposit at halt function is not implemented on the INTELLEC 8/MOD 80.

The address held in the Address Register can be incremented by one, by depressing the INC switch, or decremented by one by depressing the DEC switch.

Manual I/O Access

A Manual I/O Access operation is performed to allow the user to send data to an output device, or read data from an input device, by using the Control Console, rather than the Central Processor. It is executed in the following steps:

1) The I/O Access switch on the Control Console is depressed, sending a control signal to the processor, which enters the HOLD state.

2) The I/O Address signifying the I/O device to be used for the manual I/O access operation is loaded into Address Data switches 8-15 on the Control Console.

3) If an Output operation is to be performed, the data byte which is to be output is loaded into Address/Instruction/Data switches 0-7.

4) The DEP switch is depressed.

5) The I/O Address and data are sent to the Input/Output and Output modules, which then perform the designated input or output operation.

6) In the case of an input operation, the data from the selected input port is displayed in the data display light.

Interrupt Operations

An interrupt operation is performed in order to cause the Central Processor to interrupt its normal sequence of operations and to execute an interrupt instruction. This instruction can be such that processor operation is directed to a routine which will service the device originating the interrupt.

A Control Console interrupt is executed in the following steps:

1) The Interrupt Instruction which is to be executed during the Interrupt operation is loaded into Address/Instruction/Data switches 0-7 on the Control Console.

2) The Interrupt switch is depressed, generating an Interrupt signal which is sent to the Central Processor.

3) The Central Processor disables further interrupts and enters an Interrupt cycle.

4) The Interrupt Instruction loaded into Address/Instruction/Data switches 0-7 is sent to the Central Processor, which executes it as a normal instruction.

Sense Operations

A Sense operation is performed in order to manually input data to the Central Processor while it is running a user program. It is executed in the following steps:

1) The data which is to be input is loaded into the Address/Data 8-15 switches on the Control Console.

2) The SENSE switch is depressed, generating a control signal which is sent to the Central Processor.

3) The control signal causes the CPU to input the data from the switches, rather than from an input device, each time an Input instruction is executed.

Search-Wait Operations

Search-Wait operations are a powerful debugging tool which allows the user to execute a statement in his program a certain specified number of times, from 0 to 256, and then cause the Central Processor to enter a WAIT state, wherein the contents of memory can be examined to ensure proper program operation.

A Search-Wait operation is executed in the following steps:

1) The PASS COUNT, or number of times that an instruction is to be executed, is loaded into Address/Instruction/Data switches 0-7.

2) The LOAD PASS switch is depressed, causing the PASS COUNT to be loaded into the PASS register.

3) The address which is to be monitored is entered into the Address/Instruction/Data switches and the LOAD switch is depressed, loading the address into the Address Register.

4) Each time the referenced instruction address is encountered by the CPU, a control signal is generated. This control signal decrements the Pass Counter Register.

5) When the Pass Counter Register counts down to zero, the processor will be forced into a WAIT state if the Search/Wait switch has been depressed, allowing the user access to the system memory. This also causes the SRCH/COMP light to light.

Processor Control Operations

The Processor Control operations allow the user to control the operation of the INTELLEC 8/MOD 80 from the Control Console There are eight Processor Control functions:

1) Sense
2) Search/Wait
3) Deposit
4) Deposit at Halt (not used in the INTELLEC 8/MOD 80 System)
5) Interrupt
6) Reset
7) Step/Continuous, which allows the user to cause program execution to be performed one machine cycle at a time.
8) Wait, which causes the processor to enter a WAIT state.

The Wait function is executed by depressing the WAIT switch on the Control Console. A control signal is then produced which causes the Central Processor to enter a WAIT state. Normal operations are resumed when the switch is reset to its original position.

The Step/Cont function is dependent on the WAIT function. Single-step operation cannot be performed unless the WAIT mode is entered. Depressing the STEP/CONT switch generates a control signal which causes the CPU to leave the WAIT state and execute one machine cycle. After the cycle has been executed, the WAIT mode is reentered.

THE INTELLEC 8 FRONT PANEL CENTRAL CONSOLE – THEORY OF OPERATION

This section describes the physical implementation of the features described on page 65. Again, it is necessary that Chapter 2 of this manual be understood in order to benefit from this section.

The Intellec 8 Control Console is made up of three modules:

- The Front Panel Logic board, which holds Address Registers, data multiplexers, data buffers, and the Address Comparator.
- The Display board, which holds the circuitry which enables the Light-Emitting Diode displays.
- The Front Panel Controller, which holds the logic necessary to enable the proper performance of Console function.

These three modules work together in order to perform all of the Control Console operations, and so in this section they will be discussed as one unit.

The seven operational groups discussed in this section are:

1) Data Display operations
2) Manual Memory Access operations
3) Manual I/O Access operations
4) Interrupt operations
5) Processor Control Operations
6) Sense Operations
7) Search/Wait operations

Data Display Operations

There are five distinct data display operations:

- Status display
- Cycle display
- Address display
- Instruction/Data display
- Programmable display

All of these display operations utilize Light-Emitting Diodes as their active display element. These diodes are triggered by their input signal going to a LOW level.

The Status display functions are as follows:

- Run
- Wait
- Hold
- Search Complete
- Access Request
- Interrupt Request
- Interrupt Disable

The display functions are executed as follows:

1) The RUN status display is lit when the Central Processor is running: i.e., when it is not in the WAIT or STOPPED state. This is accomplished by combining the two signals $\overline{\text{WAIT ACK}}$, indicating the WAIT state, and $\overline{\text{HALT ACK}}$, indicating a STOPPED state, through a NAND gate. The resulting signal is inverted, producing the $\overline{\text{RUN STATUS DISP}}$ signal which will go LOW when the processor is running.
2) The WAIT status display is lit when the Central Processor is in the WAIT state. This is accomplished by using the $\overline{\text{WAIT ACK}}$ signal to produce the $\overline{\text{WAIT STATUS DISP}}$ signal, which will go LOW when the processor is in the WAIT state. In normal operation, both the RUN and WAIT displays are lit simultaneously. This is because WAIT states occur during all machine cycles, allowing ample time for memory data to be returned to the CPU.
3) The HALT status display is lit when the Central Processor is in the STOPPED state. This is accomplished by using the $\overline{\text{HALT ACK}}$ signal to produce the $\overline{\text{HALT STATUS DISP}}$ signal, which goes LOW when the processor enters the STOPPED state.
4) The HOLD status display is lit when the Central Processor has acknowledged a Hold Request. This is indicated by the presence of signal $\overline{\text{HOLD ACK}}$.

This signal is used to form the HOLD STATUS DISP signal, which goes LOW when a hold request is acknowledged.

5) The Search Complete status display is lit whenever a Search/Wait operation has been completed. This condition is indicated by the presence of signal SRCH CMPL, which is inverted to form SRCH CMPL DISP.

6) The Access Request status display is lit whenever a manual memory or I/O access has been requested from the front panel. The two signals which are produced by such requests are *I/O Access Mode* and *Mem Access Mode*. These two signals are combined by a NOR gate and a NAND gate to produce the ACCESS REQUEST DISP signal.

7) The Interrupt Request status display is lit when an Interrupt Request is made from the Control Console, and extinguished when the request is processed. This is accomplished by using the INT CTL SW signal produced by the Interrupt Request switch, to set a D flip-flop, producing the INTR REQ signal, indicating an interrupt request. This signal is inverted to form INT REQ DISP.

8) The Interrupt Disable status display is lit whenever the CPU disables its interrupt capability. The INT DSBL signal produces INT DSBL DISP.

When the Central Processor acknowledges the interrupt request, it enters an interrupt cycle, indicated by signal INT CYCLE. This signal is used to clear the flip-flop set by the request, thus extinguishing the Interrupt Request display.

The cycle display functions are:

- Fetch
- Memory
- I/O
- DA
- Read/Input
- Write/Output
- Interrupt
- Stack

The displays are as follows:

1) The FETCH display is lit during a processor Instruction Fetch operation. This is indicated by the FETCH CYCLE signal, which is passed through a buffer to produce signal FETCH CYCLE DISP.

2) The Memory Cycle display is lit when either the processor or the Control Console is executing a Memory Access Operation. In the case of the processor, this is indicated by signal MEM RD CYCLE or MEM WR CYCLE. These two signals are separately buffered and tied to a common point as signal MEM CYCLE DISP. This is possible as both signals cannot occur simultaneously. Similarly, the Control Console signal MEM ACCESS MODE cannot occur simultaneously with a processor memory access, so it is combined with DA ENBL, which indicates a memory access in progress, and is then tied to the same point as the two processor memory access signals.

3) The I/O Cycle display is lit when a processor or Control Console I/O Access operation is in progress. The processor indicates this operation with signal I/O CYCLE, which is buffered and tied to a common point with the Console I/O Access Cycle signal, which is produced by combining signals I/O Access Mode and DA ENBL in a fashion similar to that described above for memory access display operations. This produces the I/O CYCLE DISP signal.

4) The DA cycle display is lit during Control Console memory or I/O access operation. A Control Console Access operation is always begun by requesting a HOLD operation. This fact is used to produce the proper signal by buffering the HOLD ACK signal, which indicates a HOLD operation, to produce the DA CYCLE DISP signal.

5) The Read/Input cycle display is lit whenever a Memory Read or I/O Input operation is executed. This is indicated by three signals: I/O IN, produced during a Control Console I/O input operation, MEM RD CYCLE, produced during a Processor memory read operation, and also by the combination of the Memory Access Mode and DA ENBL signals as described in the discussion of the Memory Cycle display. The first two of these three signals are buffered and then tied to a common point along with the third, producing signal RD/IN CYCLE DISP.

6) The Write/Output cycle display is lit when either a memory write or I/O output operation is executed. This is indicated by two signals: MEM WR CYCLE, produced during a memory write operation, and then the combination of I/O IN and I/O CYCLE, which is true only during an I/O OUT cycle. These signals are tied to a common point to produce signal WR/OUT CYCLE DISP.

7) The *Int* cycle display is lit when an interrupt cycle is in progress, which is accomplished by inverting the INT CYCLE signal and combining it through a NAND gate with the HOLD ACK signal which indicates a HOLD operation, thus producing signal INT CYCLE DISP.

8) The Stack cycle display is lit when the stack is being accessed. The STACK CYCLE signal produces STACK CYCLE DISP.

The Address display lights are lit either by the data held in the Control Console Address Register, during a Memory Access operation, or by the data appearing on the Address/Instruction switches, during an I/O Access operation.

The choice of which set of data to use is made at a two-input multiplexer. If neither operation is being performed, the Address display is activated by the data on the Processor Memory Address Lines MAD0-MAD15.

The Instruction/Data display lights are lit by the data appearing on the Processor Data Out lines DB0-DB7 except during a Control Console data deposit operation, when they reflect the contents of the first eight Address/Instruction/Data switches.

The Register/Flag display lights reflect the contents of the Processor Register/Flag flip-flops.

Manual Memory Access Operations

Manual Memory Access operations are executed in the following manner:

1) The Mem Access switch on the front panel is depressed. This causes the Request Multiplexer to generate a $\overline{\text{HOLD REQ}}$ signal, which is sent to the Processor.
2) The Processor responds to the HOLD request by giving control of the memory address and control buses to the Control Console, and issuing signal $\overline{\text{HOLD ACK}}$.
3) The memory address to be accessed is loaded into the Address/Instruction/Data switches on the front panel.
4) The LOAD switch on the front panel is depressed, causing the switch data to be gated into the Address Register, a sixteen-bit up/down counter.
5) The data held by the address register are gated through a multiplexer and fed onto the Memory Address bus, and thence to the memory modules.
6) The memory module responds by sending the data currently held in the addressed memory location back on the Memory Data Input bus. The data is then gated onto the Data Out bus, and is displayed by the Control Console.
7) If it is desired to write data into memory the data byte to be written is loaded into the lower eight Address/Instruction/Data switches, and the DEP switch is depressed. This causes the DEPosit flip-flop to produce the $\overline{\text{DEP REQ}}$ signal, which is combined with the $\overline{\text{SYNCA}}$ and MEM ACCESS mode signals to produce the memory write signal $\overline{\text{R/W}}$. $\overline{\text{R/W}}$ is then used to clear the Deposit flip-flop, producing a pulsed write signal. The data held in the switches is gated onto the Data Out bus at the same time, by signal $\overline{\text{DEP DAEN}}$, produced by combining the DEP REQ and DA ENBL signals. The data will thus be written into the selected memory location.

Manual I/O Access Operations

A Manual I/O access operation is performed as follows:

1) The I/O Access switch on the Control Console is depressed, causing signal HOLD REQ to be generated by the Request Multiplexer and sent to the processor.
2) The processor gives control of the memory address and control buses to the Control Console, and issues signal $\overline{\text{HOLD ACK}}$.
3) The I/O Address signifying the I/O device to be accessed is loaded into A/D switches 8-15. This data is immediately gated onto the Memory Address bus, and sent to the I/O modules. Data which appears on the selected I/O device will be read onto the the Data Out lines by signal I/O IN, produced by the I/O ACCESS MODE signal, and will be displayed.
4) If an I/O Output operation is to be performed, the data to be output is loaded into the first eight A/I/D switches, and switch DEP is depressed. This causes a deposit operation to be performed, except that $\overline{\text{I/O OUT}}$ is produced rather than R/W.

Interrupt Operations

An Interrupt operation is executed as follows:

1) The Interrupt Instruction which is to be executed during the Interrupt Cycle is loaded into the eight Address/Instruction/Data switches on the Control Console.
2) The Interrupt switch is depressed, producing signal $\overline{\text{INT CTL SW}}$, which sets the Interrupt flip-flop. This flip-flop produces signal INT REQ. This signal causes the Request Multiplexer to issue signal $\overline{\text{INT REQ}}$, which is sent to the processor. It is also used to produce signal $\overline{\text{INT REQEN}}$, which causes the data placed in the switches to be gated through a multiplexer and onto the Interrupt Instruction bus.
3) The processor enters an Interrupt Cycle, producing signal $\overline{\text{INT CYCLE}}$, which resets the Interrupt flip-flop.

Sense Operations

A sense operation is executed in the following manner:

1) The data which is to be input is loaded into the 8 Address/Data switches.
2) The *Sense* switch is depressed. This causes signal $\overline{\text{SENSE REQEN}}$ to be generated, which causes the swtich data to be placed on the Input Data bus. It also produces signal $\overline{\text{IN JAM ENBL}}$, which causes

the switch data to be input during an input operation, rather than the normal input source data.

Search/Wait Operations

A Search/Wait operation is performed in the following manner:

1) The pass count is loaded into the lower eight Address/Instruction/Data switches.
2) The LOAD PASS switch is depressed, loading the pass count into the Pass Counter, an eight-bit counter.
3) The address which is to be monitored is loaded into the Address/Instruction/Data switches. The LOAD switch is depressed, loading the switch data into the Address Registers.
4) The contents of the Address Register is compared with the Memory Address buss by the SRCH ADR comparator. Each time they coincide, signal ADR CMP is produced. This signal is used to produce $\overline{\text{PC STB}}$, which is in turn used to count down the Pass Counter by one.
5) When the Pass Counter reaches zero, it produces signal $\overline{\text{SA CMP}}$. This signal is used to set the Search Complete flip-flop. This flip-flop's output causes the Request Multiplexer to issue signal $\overline{\text{WAIT REQ}}$, which causes the processor to enter a WAIT mode.

Processor Control Operations

Most of the processor control operations have been previously discussed. Those which remain are the WAIT and STEP/Continuous functions.

The wait function is executed by depressing the WAIT switch on the Control Console. This produces the WAIT MODE signal, which causes the Request Multiplexer to issue signal $\overline{\text{WAIT REQ}}$, which causes the processor to enter the WAIT mode.

If the WAIT mode is entered, the Step/Continuous function becomes valid. Depressing the STEP/CONT switch causes the $\overline{\text{WAIT REQ}}$ signal to go FALSE for approximately 1 μs., which enables the processor to execute one cycle of operation, after which it again enters the WAIT mode.

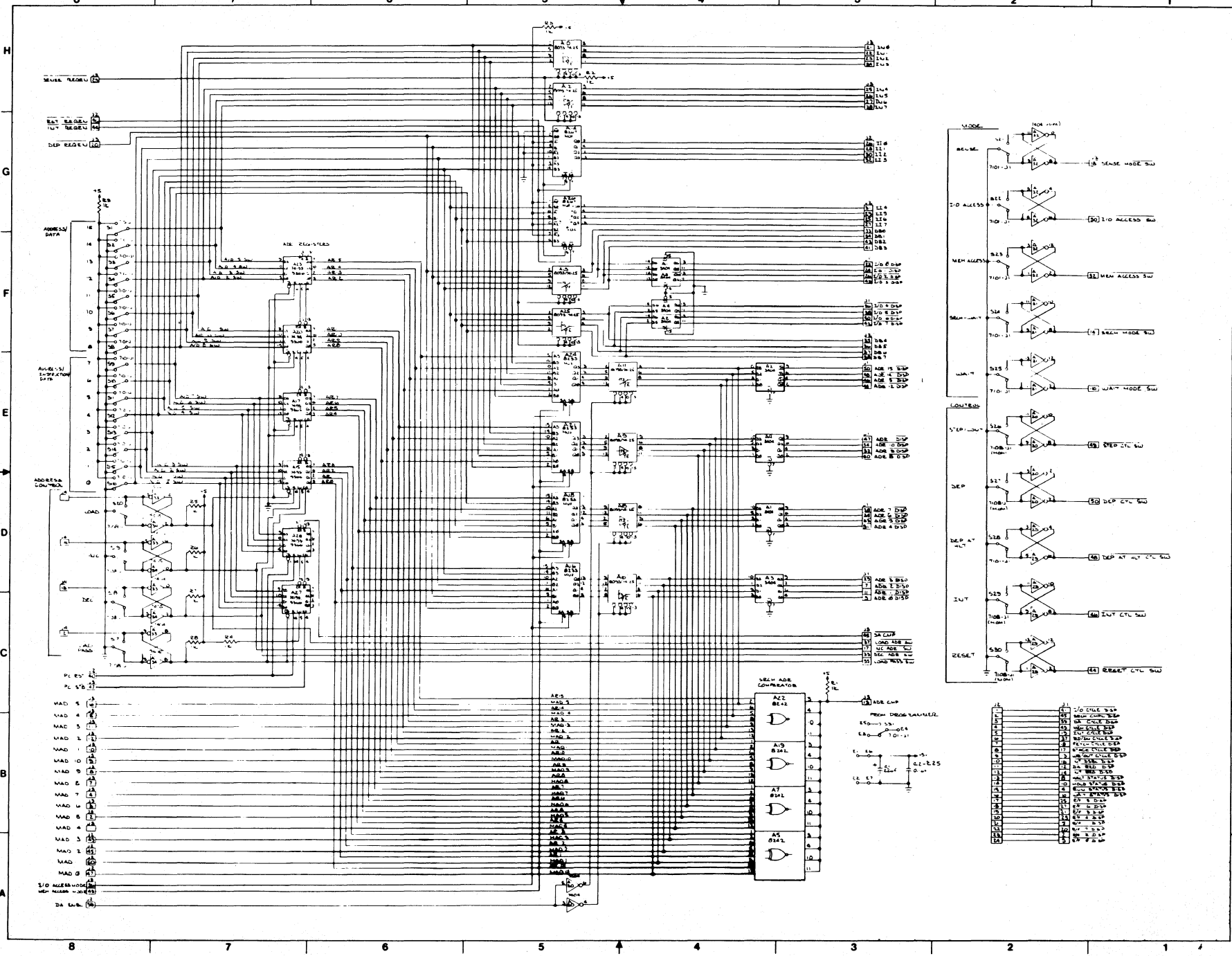


Figure 7-1. Front Panel Logic Schematic Diagram

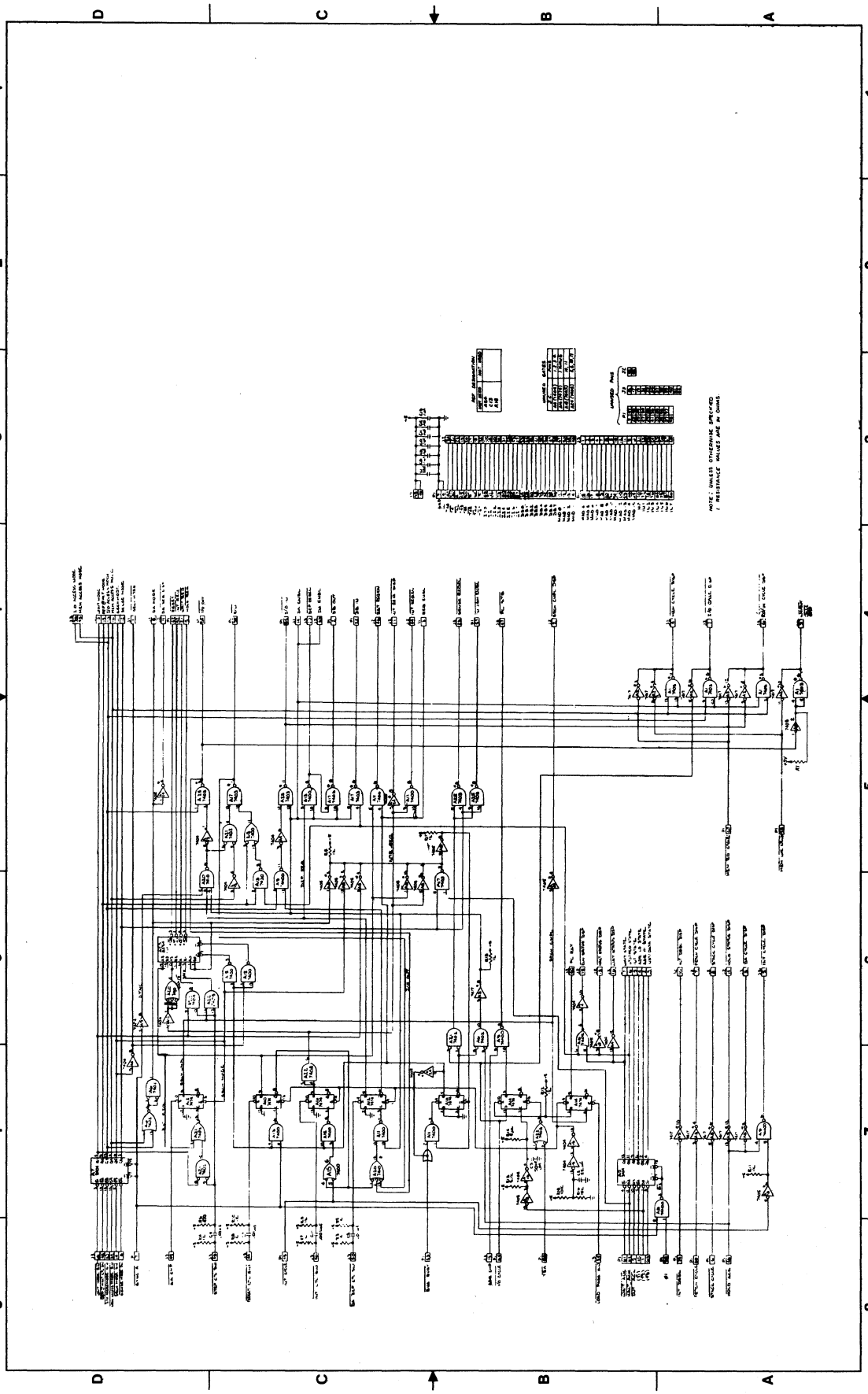


Figure 7-2. Front Panel Controller Schematic Diagram

CHAPTER 8 THE CHASSIS, MOTHER BOARD, AND POWER SUPPLIES

The INTELLEC 8 Chassis, Mother Board, and Power Supplies are designed to provide the housing, interconnection, and power services for the INTELLEC 8/MOD 80 system.

Since these three components of the INTELLEC 8 are, essentially, very simple, they will not be described in detail.

The INTELLEC 8/MOD 80 uses OEM power supplies. One supplies -9V at 1.8 Amperes. A second furnishes +5V at 12 Amperes. And the third supplies ±12V at 60 milliamperes. This is sufficient power to operate the standard INTELLEC 8/MOD 80 with one additional I/O or Output module, and one additional memory module. If greater expansion is planned, maximum and typical current draw should be totaled for all modules and the requirement for an external supply evaluated. System Utilization has more details concerning the use of the external power supply.

The Mother Board is, simply, a printed circuit board which has mounted on it the connectors which hold the various cards which make up the INTELLEC 8/MOD 80 System. The layout of these connectors is such that certain modules must occupy certain locations on the Mother Board. The suggested arrangement is shown in Figure 8-1.

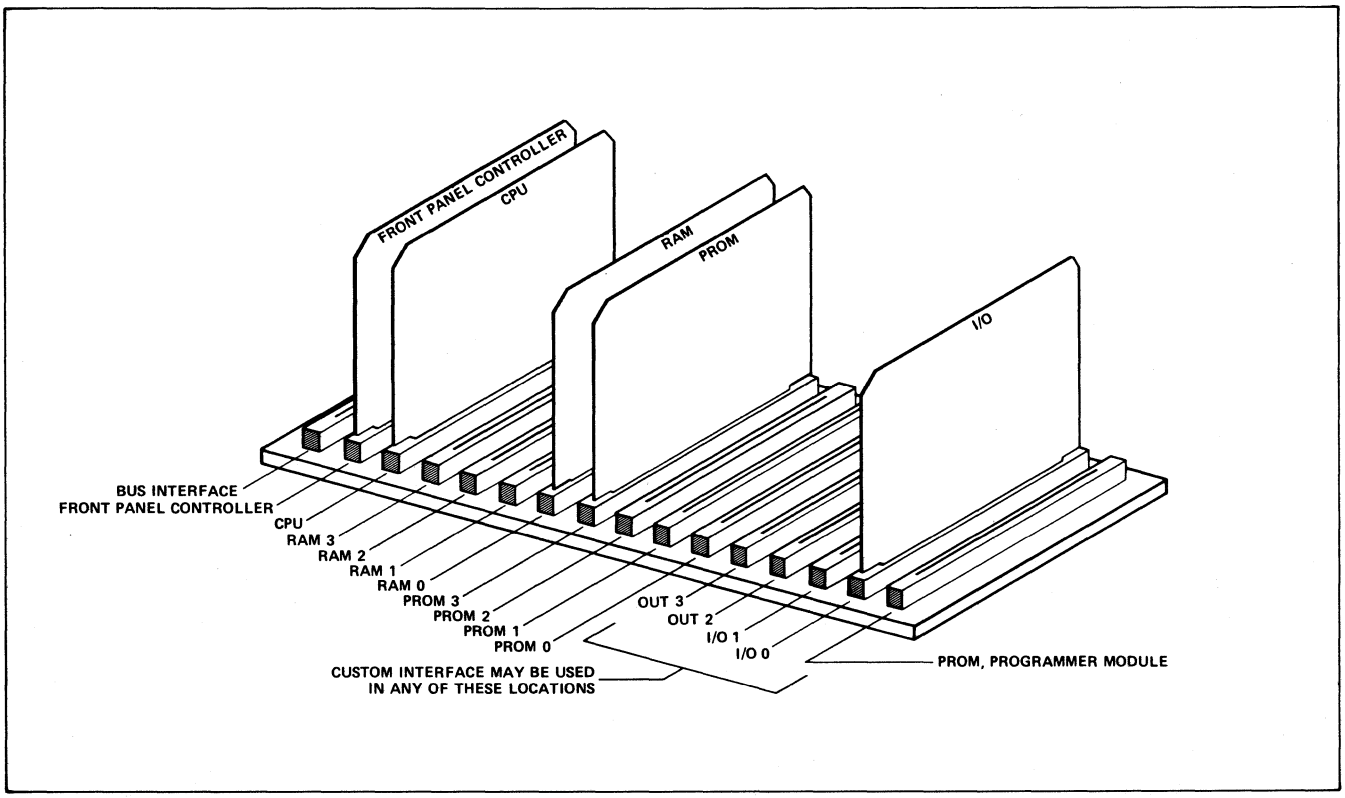


Figure 8-1. INTELLEC 8/MOD 80 Module Assignments.

CHAPTER 9 THE imm6-76 PROM PROGRAMMER MODULE

The imm8-76 PROM Programmer Module is a standard module for the INTELLEC 8/MOD 80 system. When used in conjunction with the INTELLEC 8/MOD 80 System Monitor, the Programmer Module permits rapid, automatic loading of Intel 8702A Programmable Read Only Memories.

The program to be transferred to a PROM is first stored in the INTELLEC 8's program RAM memory. The PROM to be programmed is erased, if necessary, and inserted in the programming socket on the Control and Display Panel. The PRGM PROM PWR switch is turned on, and the console operator types a 'P' followed by parameters which indicate the first and the last RAM addresses to be transferred, as well as the starting address in the PROM.

The software does the rest. It transfers the eight bits of the PROM address to output port 2. It sets up the data to be written into the PROM, at output port 3. It pulses the power supply the required number of times, at the required duty cycle. And it checks the result of its programming by reading the PROM's output through input port 2. If improper programming is indicated, the System Monitor prints an exception notice at the teletype console. This programming cycle is repeated at each of the memory locations bracketed by the initial and the terminal parameters. Complete programming involves the loading of 256 individual locations, a process which requires approximately 2 minutes. The procedure is described fully in the INTELLEC 8/MOD 80 Operator's Manual.

The imm6-76 is designed for plug-in installation in the INTELLEC 8/MOD 80 mainframe. It makes use of existing connectors and other provisions. No special installation is necessary.

THE 8702A PROGRAMMABLE READ ONLY MEMORY

The 8702A is a 256 x 8 bit electrically programmable read-only memory, designed for use in limited quantity OEM production. The 8702A is programmed by the momentary application of high amplitude pulses on selected pins of the chip. The 8702A is cleared by a controlled exposure to high intensity ultraviolet. The 8702A may be reloaded as often as desired, making it suitable for use in program development.

Programming of the 8702A requires a carefully controlled sequence of operations. The safety of the chip demands that both the intermittent voltages and the duty cycle of the programming pulses be maintained within specific limits. This insures against breakdown and overheating. On the other hand, insufficient power levels will lead to programming failures. An accurate balance is necessary. The PROM Programmer Module is designed to provide pulses of the correct level and duration, automatically.

Appendix B of this manual contains full electrical specifications for the Intel 8702A.

The 8702A is shipped to the customer in a "cleared" condition; that is, with zeros in all memory locations. An internal zero-state is indicated by a HIGH on the output pins of an enabled chip. During programming, ones are loaded selectively into each of the chip's memory locations.

A 8702A which has been programmed previously must be erased prior to reloading. Erasure is accomplished by exposing the silicon die to ultraviolet light. The device is made with a transparent quartz lid, to permit such exposure. Conventional room light, flourescent light, and sunlight have no measureable effect on data stored in the 8702A, even after years of exposure. But the device is quickly cleared by a brief exposure to high intensity ultraviolet at a wavelength of 2537 Angstroms. The Model UVS-11 (Ultraviolet Products, Incorporated: San Gabriel, California) is a cheap and effective source for this purpose. Its accompanying filter must first be removed. The recommended integrated does (the product of Intensity and the exposure time) is 6W-sec/cm^2. Ten minutes exposure to the UVS–11, at a distance of 1 inch, will clear the PROM completely. Avoid unnecessary or prolonged exposures, which are potentially damaging to the PROM.

– WARNING –

High intensity ultraviolet can cause serious burns. Ultraviolet radiation can also generate potentially hazardous amounts of ozone. Observe the following precautions, when using the source to erase a PROM:

(1) Never expose skin or eyes to the source directly.
(2) Do not stare fixedly at an object which is under ultraviolet illumination. The light is invisible, but is nevertheless injurious to eye tissue.
(3) Use the source only in a well-ventilated area.

FUNCTIONAL DESCRIPTION OF THE MODULE

An eight-line input, applied to the PROM's addressing lines, specifies the location to be programmed. Data to be written in that location is applied to the chip's eight output lines. Then address lines, data lines, the PRGM pin, and all four power lines (V_{cc}, V_{bb}, V_{gg}, and V_{DD}) are pulsed, to fix the data in location. The procedure requires about 3 milliseconds, and the cycle is repeated 32 times at each of the 256 memory locations. To prevent overheating of the 8702A, the Programmer Module maintains a 20% duty cycle, and it therefore takes approximately 123 seconds to program the entire chip.

To perform the required functions, the imm6-76 contains an address driver bank, a data driver bank, four electronically controlled power supplies, and a control and timing section.

The sequence of events is as follows:

1) Data to be programmed into the PROM is placed on the input lines, in complement (negative-true) form.
2) Address to be programmed is placed on the address lines, in complement (negative-true) form.
3) When the programming cycle begins, the following changes in the static conditions occur:
   a) V_{cc} switches from 5 to 47 Volts.
   b) V_{bb} switches from 5 to 59 Volts.
   c) V_{gg} switches from –9 to 12 Volts.
   d) V_{DD} switches from –9 to 0.6 Volts.
   e) The programming signal (PRGM) goes from 0 to 47 Volts.
   f) Address data changes from 0-5 Volts to 0-47 Volts.
4) 60 microseconds after the cycle begins, the address data is switches from its complement form to its positive-true form.
5) 155 microseconds after the cycle begins, the PRGM signal dips from 47 Volts to approximately 9 Volts.
6) 3 milliseconds later, the PRGM signal returns to 47 Volts.
7) 3.25 milliseconds after the beginning of the cycle, all voltages and signals are switched back to their normal quiescent levels.
8) 15 milliseconds after the beginning of the first cycle, the second cycle begins.

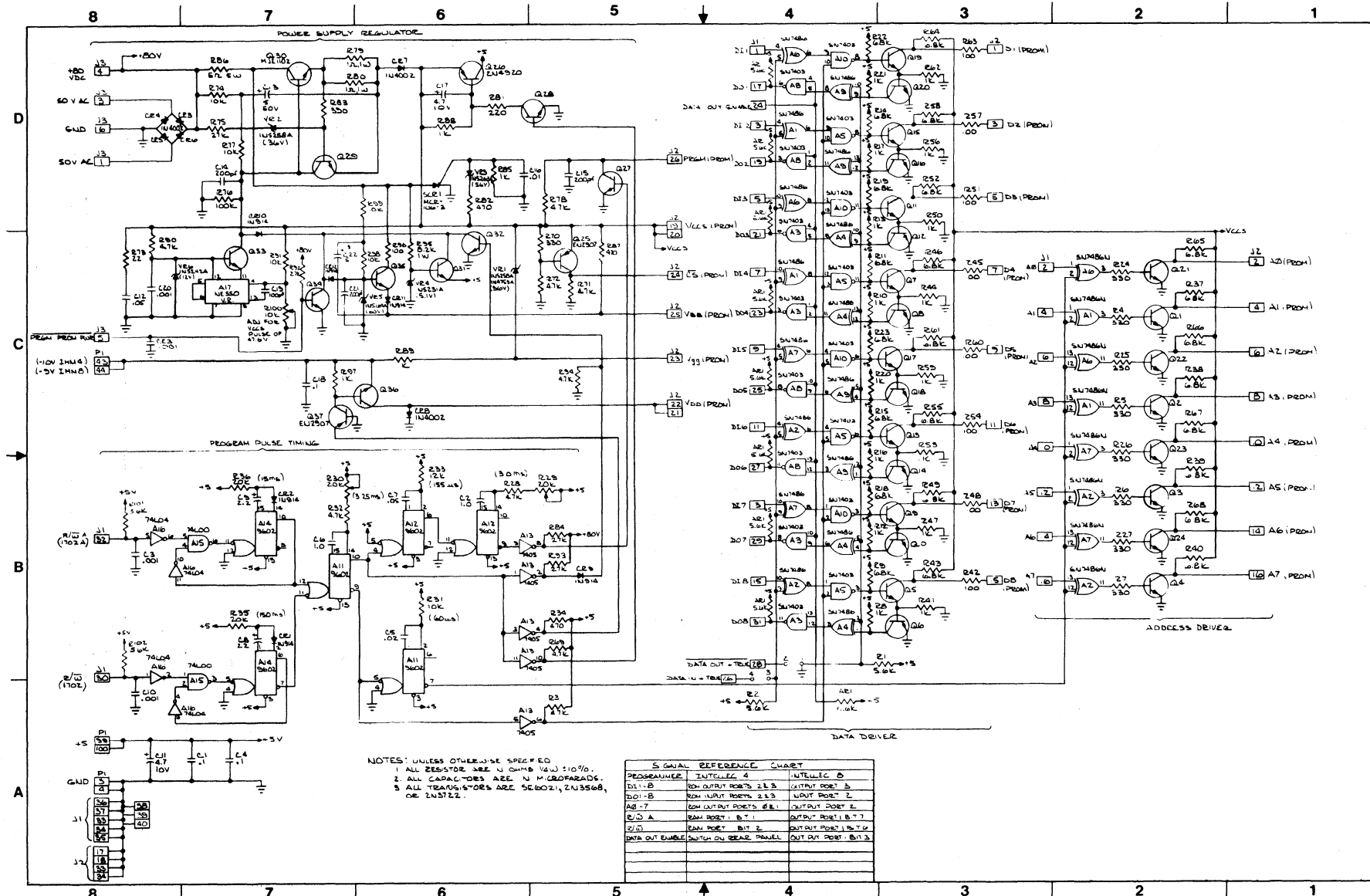
Interface to the INTELLEC 8/MOD 80

Note that the timing relationships above are determined by control circuitry on the PROM Programmer Module itself. The number of pulsed repetitions, however, is determined by the controlling program. The INTELLEC 8/MOD 80 System Monitor contains a timing routine which holds the PROM Programmer enabled for approximately 520 milliseconds, or 35 programming cycles, before stepping to the next memory location.

The ADDRESS IN lines on the Programmer Module are connected to the INTELLEC 8/MOD 80 output port #2. The DATA IN lines are connected internally to output port #3. The INTELLEC 8/MOD 80 System Monitor writes into these ports when a PROM is being programmed.

When the Programmer Module is not actively programming a memory location, the contents of that location are available at the module's DATA OUT pins. These outputs are connected in turn to input port #2, so that the INTELLEC 8/MOD 80 System Monitor can check the results if its programming.

The PROM programmer module also has two negative-true enabling inputs, which initiate the programming cycle. A LOW applied to pin #32 of the module selects a 20% programming duty cycle. This input is used when programming 8702A PROM. A LOW applied to pin #30 selects a 2% duty cycle, used when programming 8702A device. In the INTELLEC 8/MOD 80 system, pin #32 of the module is connected to the BIT #7 line of RAM output port L1. Pin #30 is connected to the BIT #6 line of the same output port. The INTELLEC 8/MOD 80 System Monitor controls the Programmer Module by writing into that port.


POWER SUPPLY REGULATOR
PROGRAM PULSE TIMING
DATA DRIVER
ADDRESS DRIVER
NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL RESISTORS ARE ¼ OHMS ¼W) ±10%.
2. ALL CAPACITORS ARE IN MICROFARADS.
3. ALL TRANSISTORS ARE SE8021, 2N3568, OR 2N3722.
SIGNAL REFERENCE CHART
PROGRAMMER
INTELLEC 4
INTELLEC 8
D1 - 8
ROM OUTPUT PORTS 2 & 3
OUTPUT PORT 3
DO1 - 8
ROM INPUT PORTS 2 & 3
INPUT PORT 2
A0 - 7
ROM OUTPUT PORTS 0 & 1
OUTPUT PORT 2
R/C A
RAM PORT: BIT 1
OUTPUT PORT: BIT 7
R/C
RAM PORT BIT 2
OUTPUT PORT: BIT 6
DATA OUT ENABLE
SWITCH ON REAR PANEL
OUT PUT PORT: BIT 3

THEORY OF OPERATION OF THE MODULE

Refer to Figure 9-1, the PROM Programmer Schematic.

Data Distribution

The data to be programmed into the PROM enter originates at output port #3. This eight-line signal enters the Programmer Module through a ribbon cable which runs from J1 on the INTELLEC 8/MOD 80 motherboard to J1 at the top of the module. Each of the input lines is applied to one input of an XOR-gate. The alternate inputs of these eight gates are returned through a common line to the +5 Volt supply, so that each gate acts as an inverter to the incoming data.

Each of the XOR-gate outputs is directed to one input of a 7403 NAND-gate. The alternate inputs to this bank of gates are driven in common by a signal originating in the control and timing section of the module. At the appropriate time in the cycle, these inputs are permitted to swing HIGH, causing data from the XOR-gate bank to pass through to the bases of eight driver transistors: Q19, Q15, Q11, Q7, Q17, A13, A9, and Q5. The signal at the collectors of these drivers is conducted out of the assembly through a ribbon cable which attaches to J2 at the top of the module. It goes from there to the programming socket on the front panel of the INTELLEC 8/MOD 80. This data undergoes three successive inversions, between entering and leaving the imm8-76. The output will therefore be in complementary form, as required for the programming of the 8702A PROM.

Observe that the bases of the PROM data driver transistors are returned through pull-up resistors to the +5 Volt supply. As a result, these transistors will be conducting whenever the input NAND-gates are inhibited. Under these circumstances, the signal at each of the PROM's data pins will be applied to the base of a transistor, through a divider consisting of a 100-ohm resistor, the DC collector resistance of a driver transistor, and a 1K resistor. Transistors Q20, Q16, Q12, Q8, Q18, Q14, Q10, and Q6 amplify this eight-line signal and forward it to an XOR-gate bank which is used as an eight-line data inverter. The outputs of the XOR-gates are applied to eight NAND-gates which have their alternate inputs tied in common to the +5 Volt supply. These gates are permanently enabled, and also act as data inverters. The output of these gates is in positive-true form. It is routed out of the assembly at J1, through a ribbon cable to J1 on the INTELLEC 8/MOD 80's motherboard, and terminates at input port #2. The INTELLEC 8/MOD 80 System Monitor reads this port, to determine the results of its programming.

Address data enters the module at J1, through a ribbon cable connecting it to J1 of the INTELLEC 8/MOD 80's motherboard. Data originating at output port #3 is therefore applied to the eight-line XOR-gate bank, shown on the right in Figure 9-1. The outputs of these gates are

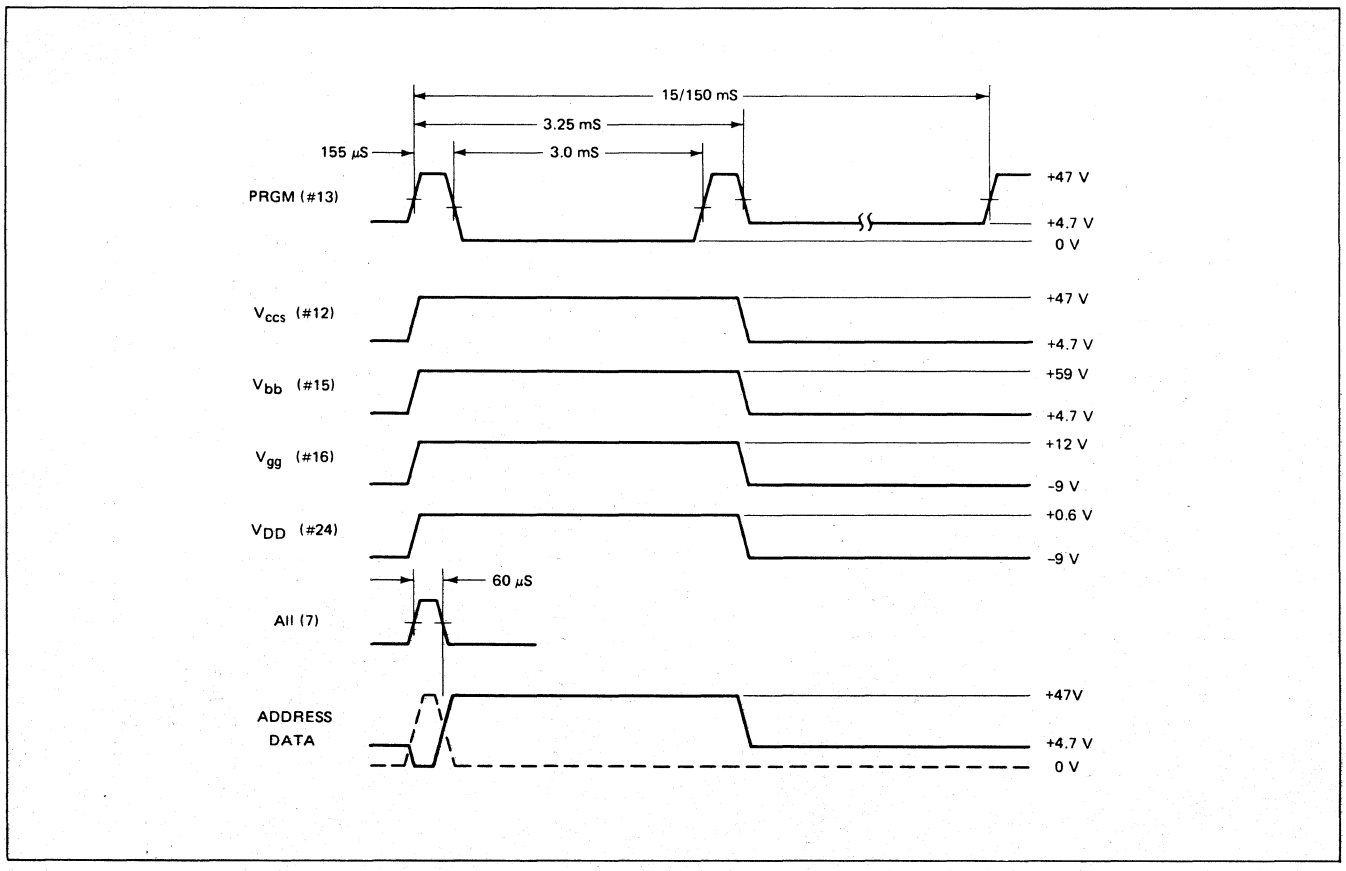


Figure 9-2. PROM Programmer Timing

directed to the bases of eight driver transistors, whose outputs command the PROM address lines. Note that the alternate inputs of the XOR-gates are tied in common to a signal line from the control and timing section. This line swings LOW when the programming cycle begins. It returns to a HIGH condition 60 microseconds later. As a result, the address forwarded to the PROM is in complementary form initially. Sixty microseconds after the programming cycle begins, the address data will switch to its positive-true form, in accordance with the PROM's programming requirements.

Control and Timing

As shown in Figure 9-1, the programming cycle may be initiated by a LOW applied to pin #32 or to pin #30 of the card. The INTELLEC 8/MOD 80 System Monitor enables the pin #32 input, selecting a duty cycle of 20% (3 mS/15 mS). The pin #30 input is set up for the 2% duty cycle to program 8702 devices.

When a LOW is applied to pin #32 of the module, the 15 millisecond input multivibrator re-triggers itself repetitively, until the enabling signal is removed. This provides a series of positive-going excursions with a period of 15 milliseconds, which are used to trigger the 3.25 millisecond program cycle one-shot.

The output of the program cycle one-shot:

1) Complements the address to the PROM.
2) Enables the data drivers.
3) Pulses all four power supplies.
4) Triggers a 155 microsecond cascaded one-shot delay.

Sixty microseconds after the program cycle one-shot fires, the negative-going pulse output at A11-7 subsides, and the address data returns to its positive-true form.

One hundred fifty-five microseconds after the program cycle one-shot fires, A12-9-10-11-12-13-14 fires, causing the power supply to apply a 3 millisecond PRGM pulse to the PROM.

Three and a quarter milliseconds after the beginning of the programming cycle, all signals return to their quiescent levels.

The Programmer Module's control timing is illustrated in Figure 9-2.

Power Supply

The power supply section of the PROM Programmer Module performs the level switching functions required to program PROMs, in response to signals which are generated in the timing and control section of the module. The power supply contains a rectifier section, a voltage regulator section, a regulator control section, and six output switches. The relationship among these is shown in a simplified form, in Figure 9-3.

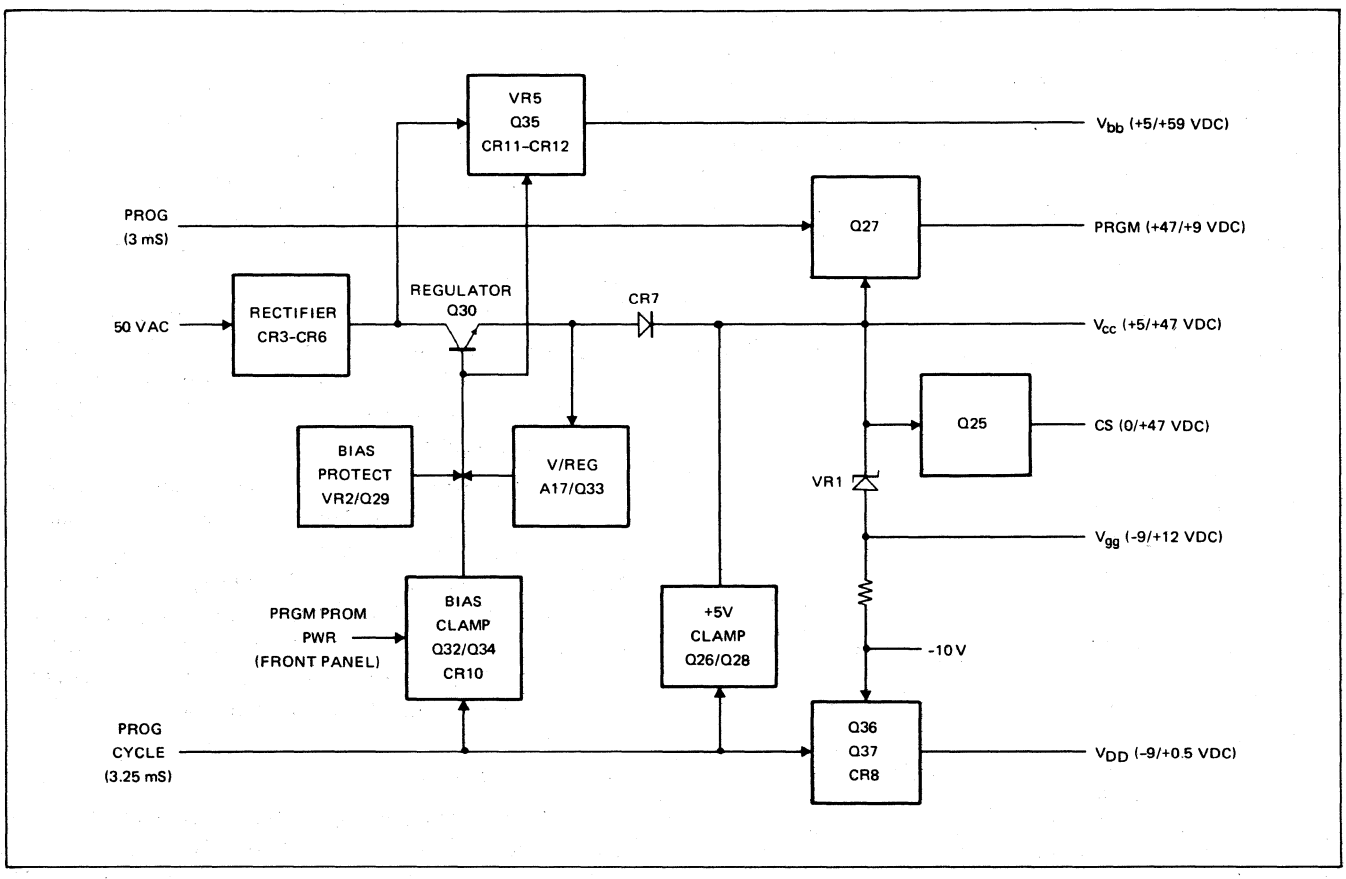


Figure 9-3. Power Supply Functional Block

RECTIFIER AND REGULATOR

The Programmer Module receives a 50 VAC/60 Hz input, from two 25 Volt transformers which are located on the INTELLEC 8/MOD 80's chassis. The secondaries of these transformers are connected so that their outputs are series additive, and the 50 Volt output thus obtained is routed to the Programmer Module through J3. A full-wave bridge consisting of diodes CR3-CR6 rectifies the 50 Volt input to produce a +80 Volt DC output.

The +80VDC output of the rectifier is applied to a series regulator, Q30, shown in the upper left hand corner of Figure 9-1. The output voltage at the emitter of Q30 depends upon the signal at its base. This level is determined in turn by a regulator loop which consists of an integrated voltage regulator (A17), Q33, and Q30 itself.

Figure 9-4 shows a simplified equivalent of the regulator loop. Components within the broken lines are part of the Signetics 550 monolithic voltage regulator.

The loop input is obtained from the regulator's output, through an adjustable resistive divider (R91 and R100). This level is applied to the non-inverting input of an operational amplifier which is incorporated into A17. The output of the amplifier drives a common-emitter stage, also contained within A17, and the inverted output at A17-11 is applied externally to the emitter of Q33. Q33's collector drives the base of the series regulator Q30, completing the negative feedback loop.

In a stabilized configuration such as this, the operational amplifier tends to maintain an output which results in zero error, where the error is the potential difference between the amplifier's inverting and non-inverting inputs. Note that the inverting input is tied to the 550's internal reference (approximately 1.63 Volts). In order to obtain the desired output from the regulator, the resistive divider is adjusted for a zero error when the regulator's output is approximately +47.6 Volts.

Refer to the schematic for the PROM Programmer Module, Figure 9-1. Observe that the series regulator Q30 is

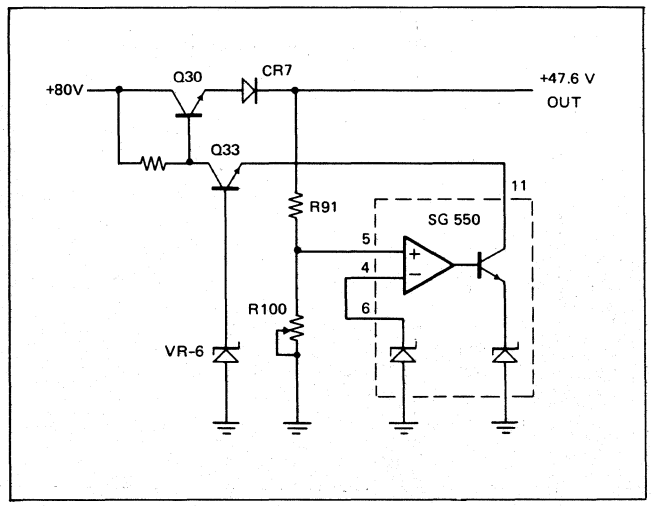


Figure 9-4. Voltage Regulator Loop: Simplified Schematic Equivalent

protected against short-circuit overloads, by a bias protection circuit consisting of Q29 and the Zener diode VR2. Under ordinary operating conditions, Q29 will be off, and the reverse voltage applied to VR2 will be insufficient to cause this diode to conduct. In the event of a short-circuit, however, the voltage drop across Q30 will rise sharply. VR2 will begin conducting when the voltage across Q30 approaches 36 Volts, applying a forward bias to Q29. As a result, the voltage at Q29's collector will drop, clamping the base of Q30 to a relatively low level, and limiting the current output from the supply.

SCR1 is a crowbar switch, used to protect the PROM being programmed from an over-voltage condition in the supply. The normal voltage level on the V_{CCS} line (+47.6 Volts) is insufficient to cause conduction in Zener diode VR3. Should V_{CCS} rise above +56 Volts, however, the diode will conduct, forward biasing the gate of the SCR. SCR1 short-circuits the output of the rectifier, and the over-current condition blows fuse F2, interrupting AC power to the Programmer Module. Capacitor C16 provides an alternate gate current path, to prevent dv/dt triggering of the SCR when power is initially applied.

REGULATOR CONTROL

Refer again to Figure 9-3, the power supply functional block. Note that the bias on Q30 is subject to the condition of a clamp. The clamp circuit consists of Q32, Q34, CR10, and associated components. These are used to switch the regulator output on and off, producing the pulses required for the programming of the PROM.

The base of Q34 is returned to the +80 Volt source, through pull-up resistor R92 (refer to Figure 9-1). Under static conditions, this transistor will conduct through CR10, clamping the base of Q30 to a low value. As a result of the low forward bias, Q30 displays a high impedance, and the output of the regulator will therefore drop to a very low value.

The PRGM PROM PWR switch is located on the Console and Display Panel of the INTELLEC 8/MOD 80. Contacts of the PRGM PROM PWR switch ground the base of Q34 when that switch is turned on. This turns Q34 off, enabling the regulator.

The regulator's output remains clamped, however, by the conduction of Q32. This transistor is commanded by the control and timing section of the Programmer Module. The 3.25 millisecond output of the program cycle one-shot turns Q32 off at the start of the programming cycle. With both Q32 and Q34 disabled, the bias on Q30 rises to the stable level established by the characteristics of the regulator loop. The output of the regulator rises in consequence.

OUTPUT SWITCHES

When no program cycle pulse is present, the regulator's output is at a low level. Diode CR7 is reverse biased, and the output voltage on the V_{CCS} line is determined by the clamp circuit consisting of Q26 and Q28. Under these

conditions, Q26 operates in the reverse beta mode, holding V_{CCS} to approximately +4.7 Volts. When the program cycle begins, the control and timing section applies a negative-going 3.25 millisecond pulse to the base of Q28, turning that transistor off. Q26 now operates in a conventional manner, turned off by the low bias developed across R88. With the clamp removed, the V_{CCS} line is free to follow the rising output of the regulator section. CR7 conducts, and the V_{CCS} line rises to approximately +47 Volts.

Observe that the collectors of both the address drivers and the data drivers are returned to the V_{CCS} line, through their individual load resistors. Thus the normal 0 to 5 Volt logic excursion which prevails under static conditions changes to a 0 to 47 Volt excursion during programming. This is an accord with the electrical requirements of the PROMs.

As V_{CCS} rises, Q25 goes into conduction, causing the level at the CS output to go from 0 Volts to +47 Volts.

Under static conditions, conduction through R89 holds the V_{gg} output to approximately −10 Volts. The 15 Volt drop across VR1 is not sufficient to induce an avalanche in the Zener. During programming, however, V_{CCS} rises to +47 Volts and the diode goes into conduction. As a result, V_{gg} rises to +11 Volts, approximately 36 Volts below the level on the V_{CCS} line.

The V_{DD} output is held to a static level of −10 Volts, by conduction through Q36. When programming begins, a negative-going program cycle signal is applied to the emitter of Q37. The negative-going transition at its collector is coupled to the base of Q36, and Q36 turns off. CR8 conducts, causing V_{DD} to rise to about 0.6 Volts.

Under static conditions, the clamp transistor Q32 is conducting, and Q35 is turned off by the low voltage applied to its base through diode CR12. The V_{bb} output line is tied to V_{CCS} through R87, and the quiescent voltage level at this point is approximately +4.7 Volts. When the program cycle pulse turns Q32 off, CR5 conducts, and the voltage at the base of Q35 rises to the vicinity of +60 Volts. The emitter of Q35 follows this excursion, and CR5 conducts, pulling V_{bb} up to a level of +59 Volts.

The PRGM line is connected to V_{CCS} through R78, and the static level at this output is approximately +4.7 Volts. When V_{CCS} rises to +47 Volts, at the beginning of the programming cycle, the PRGM output follows. One hundred fifty-five microseconds after the start of the cycle, the control and timing section sends a 3 millisecond program pulse to the base of Q27. This positive-going pulse turns the transistor on, and the voltage at its collector falls to approximately +9 Volts. Three milliseconds later, the PRGM output returns to +47 Volts, where it remains until the end of the programming cycle.

UTILIZATION

This section describes the utilization of the imm6-76.

Installation

The PROM Programmer Module is designed for plug-in installation in the INTELLEC 8/MOD 80. No special installation is necessary.

Plug the printed circuit board into J16 on the INTELLEC 8/MOD 80's motherboard. A ribbon cable connects J1 at the top of the module to J1 on the motherboard. A second ribbon cable connects J2 on the module to the programming socket on the front panel of the INTELLEC 8/MOD 80.

An umbilical cable, permanently attached to the module, plugs into J34 on the INTELLEC 8/MOD 80's motherboard. This connection supplies AC power to the Programmer Module.

Refer to the INTELLEC 8/MOD 80 Operator's Manual for instructions on the programming of PROMs using the INTELLEC 8/MOD 80 System Monitor.

Power Requirements

This module requires power at the following levels:

a) 50 VAC
b) +5 ±5% VDC @ 1.0A (max)
c) −10 ±5% VDC @ 0.2A (max)

The 50 VAC source shares a fuse with the −9 Volts supply in the INTELLEC 8/MOD 80. This 0.5 Ampere fuse, F2, is located on the INTELLEC 8/MOD 80's rear panel.

Pin List

Connector pin allocations on the PROM Programmer Module are given in Tables 9-1, 9-2, 9-3, and 9-4.

P1 PIN LIST

PIN	SIGNAL FUNCTION	PIN	SIGNAL FUNCTION
1		51	
2		52	
3	GROUND	53	
4	GROUND	54	
5		55	
6		56	
7		57	
8		58	
9		59	
10		60	
11		61	
12		62	
13		63	
14		64	
15		65	
16		66	
17		67	
18		68	
19		69	
20		70	
21		71	
22		72	
23		73	
24		74	
25		75	
26		76	
27		77	
28		78	
29		79	
30	R/W (1701)	80	
31		81	
32	R/W (1702A)	82	
33		83	
34		84	
35		85	
36		86	
37		87	
38		88	
39		89	
40		90	
41		91	
42		92	
43	-10 VDC	93	
44	-10 VDC	94	
45		95	
46		96	
47		97	
48		98	
49		99	+5 VDC
50		100	+5 VDC

Table 9-1.

J1 PIN LIST

PIN	SIGNAL FUNCTION
1	DATA 0 IN
2	ADDRESS 0 IN
3	DATA 1 IN
4	ADDRESS 1 IN
5	DATA 2 IN
6	ADDRESS 2 IN
7	DATA 3 IN
8	ADDRESS 3 IN
9	DATA 4 IN
10	ADDRESS 4 IN
11	DATA 5 IN
12	ADDRESS 5 IN
13	DATA 6 IN
14	ADDRESS 6 IN
15	DATA 7 IN
16	ADDRESS 7 IN
17	TEST DATA OUT 0
18	
19	TEST DATA OUT 1
20	
21	TEST DATA OUT 2
22	
23	TEST DATA OUT 3
24	
25	TEST DATA OUT 4
26	
27	TEST DATA OUT 5
28	
29	TEST DATA OUT 6
30	
31	TEST DATA OUT 7
32	
33	+5 VDC
34	+5 VDC
35	+5 VDC
36	+5 VDC
37	+5 VDC
38	+5 VDC
39	+5 VDC
40	+5 VDC
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

Table 9-2.

J2 PIN LIST

PIN	SIGNAL FUNCTION
1	PROM DATA OUT 0
2	PROM ADDRESS OUT 0
3	PROM DATA OUT 1
4	PROM ADDRESS OUT 1
5	PROM DATA OUT 2
6	PROM ADDRESS OUT 2
7	PROM DATA OUT 3
8	PROM ADDRESS OUT 3
9	PROM DATA OUT 4
10	PROM ADDRESS OUT 4
11	PROM DATA OUT 5
12	PROM ADDRESS OUT 5
13	PROM DATA OUT 6
14	PROM ADDRESS OUT 6
15	PROM DATA OUT 7
16	PROM ADDRESS OUT 7
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

Table 9-3.

J3 PIN LIST

PIN	SIGNAL FUNCTION
1	50 VAC (01)
2	
3	50 VAC (02)
4	+80 VDC OUT
5	$\overline{\text{PROGRAM PROM POWER}}$
6	GROUND
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	

Table 9-4.

CHAPTER 10 THE INTELLEC 8/MOD 80 SYSTEM UTILIZATION

This section gives the information necessary to install and operate the INTELLEC 8/MOD 80 system in an application. It is divided into four subsections.

INTELLEC 8/MOD 80 INSTALLATION

Installation of the INTELLEC 8/MOD 80 is very simple, as it is delivered in a ready-to-use condition. Simply set it on a convenient surface, plug the 110v supply cord into the nearest 110v AC socket, and connect any desired peripherals, and it is ready to use.

The Bare Bones 80 is almost as simple to install, as it has been designed to mount in any standard 19-1/2 inch RETMA panel.

SYSTEM I/O INTERFACING

This section provides the information necessary to properly interface external input and output equipment to the INTELLEC 8/MOD 80. Since most of the interfacing requirements are supplied by the internal Input/Output and Output cards, interfacing is not a complex task; however, there are certain procedures which must be followed in order to assure the proper operation of any external devices used.

The INTELLEC 8/MOD 80 can support up to 16 input ports and 16 output ports (when four imm8-61 I/O Modules are used) or up to 4 input ports and 28 output ports (when one imm8-61 I/O Module and three imm8-63 Output Modules are used). The ports can be assigned to specific modules as shown (lower left).

Module-Location	Ports
I/O #0	Input ports 0-3; output ports 0-3
I/O #1	Input ports 4-7; output ports 4-7
I/O #2	Input ports 8-11; output ports 8-11
I/O #3	Input ports 12-15; output ports 12-15
OR	OR
* I/O #0	Input ports 0-3; output ports 0-3
OUT #1	Output ports 16-23
OUT #2	Output ports 8-15
OUT #3	Output ports 24-31

*Note that in this configuration none of the output ports respond to addresses 4-7.

All of the data ports complement data to and from the CPU, and are TTL compatible. Note that the two input ports (0 and 1) and two output ports (0 and 1) used for Teletype communications are not available to the user. The data from the other ports is brought, via flat cables, to the back panel of the INTELLEC 8/MOD 80, where it is made available on 37 pin jacks (see Figure 10-1). External devices may connect to these jacks using AMP 205210-1 plugs.

The standard INTELLEC 8/MOD 80 comes equipped with only one Input/Output card, providing four input ports and four output ports. A table of the data signals associated with these ports is given in Table 10-1.

In order to ensure the proper transmission of data through a twisted cable of 12 feet (maximum), the user should provide circuitry which will assist in reducing signal noise. It is suggested that each output line be provided with a filter network and pullup resistors. The filter is made up of a 200 ohm resistor and a .001 μf capacitor, and the pullup resistor should be 1K ohm.

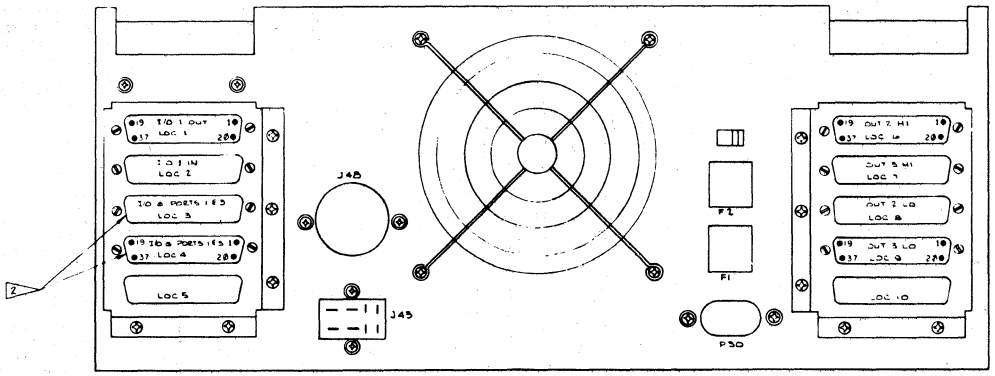
Also, 7404-type drivers are suggested for each input data line. These drivers should, preferably, be open-collector type devices. If input ports 2 or 3 are used, open-collector devices *must* be used, as these ports are shared with the PROM Programmer during programming, transfer and compare PROM operations. The user must disable his input drivers when PROM programming operations are being performed.

INTELLEC 8/MOD 80 SYSTEM OPERATING REQUIREMENTS

In order to ensure proper performance, certain requirements must be met in operating the INTELLEC 8/MOD 80.

Figure 10-1. INTELLEC 8/MOD 80 Rear Panel

CABLE INTERFACE	
FROM	TO
LOC 1	I/O MOD 1 J5 (OUT)
LOC 2	I/O MOD 1 J6 (IN)
LOC 3	W/I/O MOD'S PORTS 1&3 (J4 & J5)
LOC 4	LOC 3
LOC 6	OUTPUT MOD 2 J5 (HI)
LOC 7	OUTPUT MOD 3 J5 (HI)
LOC 8	OUTPUT MOD 2 J2 (LO)
LOC 9	OUTPUT MOD 3 J2 (LO)
+5V PWR SUP P&J1	BACK PANEL J&J
AUX OUTLET P&J8	BACK PANEL J&J8
TTY CONN P&J9	BACK PANEL J&J9
AC RECPT J&J0	BACK PANEL P&J0



BACK PANEL

NOTE

1. LOC. 5, 10 AND 11 ARE NOT USED
2. CABLE P/N 40002866 OR PUNCH CABLE P/N 40002887 MAYBE INTERFACED INTO LOC. 3 OR LOC. 4.

I/O Port Assignments-Module I/O 0

SIGNAL	SYMBOL	COMMENTS	BACK PANEL CONN. PIN # (3)	MODULE PIN # J5
OUTPUT PORT 00, BIT 0	$\overline{OP00}$, 0	UART XMIT DATA 0	(1)	2
	1			3
	2			4
	3			5
	4			6
	5			7
	6			8
OUTPUT PORT 00, BIT 7	$\overline{OP00}$, 7	UART XMIT DATA 7	(1)	9
OUTPUT PORT 01, BIT 0	$\overline{OP01}$, 0	RDR ADV-1	10	11
1	1	PUNCH COMMAND	11	12
2	2	READER COMMAND	29	13
3	3	DATA OUT ENBL	30	14
4	4	DATA IN	12	15
5	5	DATA OUT	13	16
6	6	R/$\overline{W}$	31	17
OUTPUT PORT 01, BIT 7	$\overline{OP01}$, 7	R/$\overline{W}$A	32	18
OUTPUT PORT 02, BIT 0	$\overline{OP02}$, 0	PROM ADR IN 0	(2)	20
1	1	1		21
2	2	2		22
3	3	3		23
4	4	4		24
5	5	5		25
6	6	6		26
OUTPUT PORT 02, BIT 7	$\overline{OP02}$, 7	PROM ADR IN 7	(2)	27
OUTPUT PORT 03, BIT 0	$\overline{OP03}$, 0	PROM DATA IN 0, PUNCH DATA 0	14	29
1	1	1, 1	15	30
2	2	2, 2	33	31
3	3	3, 3	34	32
4	4	4, 4	16	33
5	5	5, 5	17	34
6	6	6, 6	35	35
OUTPUT PORT 03, BIT 7	$\overline{OP03}$, 7	PROM DATA IN 7, PUNCH DATA 7	36	36
GROUND			1, 18, 19, 20, 37	37–40

NOTES:

(1) Dedicated to UART/TTY operations and unavailable to user.

(2) Dedicated to PROM Programming Operation and unavailable to user.

(3) Back Panel Connector Signals appear at both LOC 3 and LOC 4.

Table 10-1.

I/O Port Assignments-Module I/O 0

SIGNAL	SYMBOL	COMMENTS	BACK PANEL CONN. PIN #	MODULE PIN # J4
INPUT PORT 0, BIT 0	$\overline{IP0}$, 0	TTY RCV DATA 0	(1)	2
1	1	1		3
2	2	2		4
3	3	3		5
4	4	4		6
5	5	5		7
6	6	6		8
INPUT PORT 0, BIT 7	$\overline{IP0}$, 7	TTY RCV DATA 7	(1)	9
INPUT PORT 1, BIT 0	$\overline{IP1}$, 0	DATA AVAILABLE	2	11
1	1	OVERRUN ERROR	3	12
2	2	TRANSMIT BUFFER EMPTY	21	13
3	3	FRAMMING ERROR	22	14
4	4	PARITY ERROR (INHIBITED)	4	15
5	5	DATA AVAILABLE (TAPE READER)	5	16
6	6	PUNCH READY	23	17
INPUT PORT 1, BIT 7	$\overline{IP1}$, 7		24	18
INPUT PORT 2, BIT 0	$\overline{IP2}$, 0	PROM DATA OUT (J3-16)	(2)	20
1	1	(15)		21
2	2	(14)		22
3	3	(13)		23
4	4	(12)		24
5	5	(11)		25
6	6	(10)		26
INPUT PORT 2, BIT 7	$\overline{IP2}$, 7	PROM DATA OUT (J3- 9)	(2)	27
INPUT PORT 3, BIT 0	$\overline{IP3}$, 0	READER DATA 0	6	29
1	1	1	7	30
2	2	2	25	31
3	3	3	26	32
4	4	4	8	33
5	5	5	9	34
6	6	6	27	35
INPUT PORT 3, BIT 7	$\overline{IP3}$, 7	READER DATA 7	28	36
GROUND			1, 18, 19, 20, 37	37-40

NOTES:

(1) Dedicated to UART/TTY operations and unavailable to user.

(2) Dedicated to PROM PGMR and unavailable to user.

(3) Back Panel CONNECTOR Signals appear at both LOC 3 and LOC 4.

Table 10-1 (cont.).

I/O Module To Back Panel Interface Chart

SIGNAL/MODULE										CONNECTOR	
			I/O 1		OUT 2		OUT 3				
			IN (J4)	OUT (J5)	OUT_L (J2)	OUT_H (J3)	OUT_L (J2)	OUT_H (J3)	BIT No.	BACK PANEL CONN PIN #	MODULE CONN PIN # (Flat Cable)
			LOC3	LOC1	LOC8	LOC6	LOC9	LOC7			
			$\overline{IP4}$	$\overline{OP05}$	$\overline{OP09}$	$\overline{OP0D}$	$\overline{OP11}$	$\overline{OP1C}$	0	2	2
									1	3	3
									2	21	4
									3	22	5
									4	4	6
									5	5	7
									6	23	8
			$\overline{IP4}$	$\overline{OP05}$	$\overline{OP09}$	$\overline{OP0D}$	$\overline{OP11}$	$\overline{OP1C}$	7	24	9
			$\overline{IP5}$	$\overline{OP06}$	$\overline{OP0A}$	$\overline{OP0E}$	$\overline{OP12}$	$\overline{OP1D}$	0	6	11
									1	7	12
									2	25	13
									3	26	14
									4	8	15
									5	9	16
									6	27	17
			$\overline{IP5}$	$\overline{OP06}$	$\overline{OP0A}$	$\overline{OP0E}$	$\overline{OP12}$	$\overline{OP1D}$	7	28	18
			$\overline{IP6}$	$\overline{OP07}$	$\overline{OP0B}$	$\overline{OP0F}$	$\overline{OP1A}$	$\overline{OP1E}$	0	10	20
									1	11	21
									2	29	22
									3	30	23
									4	12	24
									5	13	25
									6	31	26
			$\overline{IP6}$	$\overline{OP07}$	$\overline{OP0B}$	$\overline{OP0F}$	$\overline{OP1A}$	$\overline{OP1E}$	7	32	27
			$\overline{IP7}$	$\overline{OP08}$	$\overline{OP0C}$	$\overline{OP10}$	$\overline{OP1B}$	$\overline{OP1F}$	0	14	29
									1	15	30
									2	33	31
									3	34	32
									4	16	33
									5	17	34
			$\overline{IP7}$	$\overline{OP08}$	$\overline{OP0C}$	$\overline{OP10}$	$\overline{OP1B}$	$\overline{OP1F}$	6	35	35
									7	36	36
									GND	1,18,19,20,37	37-40

Table 10-2

First, *never* operate the INTELLEC 8/MOD 80 with the cover off. If this is done, the proper flow of air will be disrupted, resulting in the burning-out of the internal power supplies.

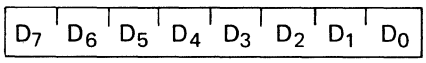
Second, use extreme care when removing or installing individual circuit cards in the INTELLEC 8/MOD 80, especially Input/Output board #1. The PROM Programmer and Teletype connectors to I/O board 0 are very easily damaged, and are located very close to I/O board #1.

EXTERNAL DEVICE CONTROLLER INTERFACING

The INTELLEC 8 may be used with external devices such as disks, etc., which require a Direct Memory Access capability. This is accomplished by the TRI-State capability of the processor memory address and control buses, which can relinquish their control of INTELLEC operations to an external device.

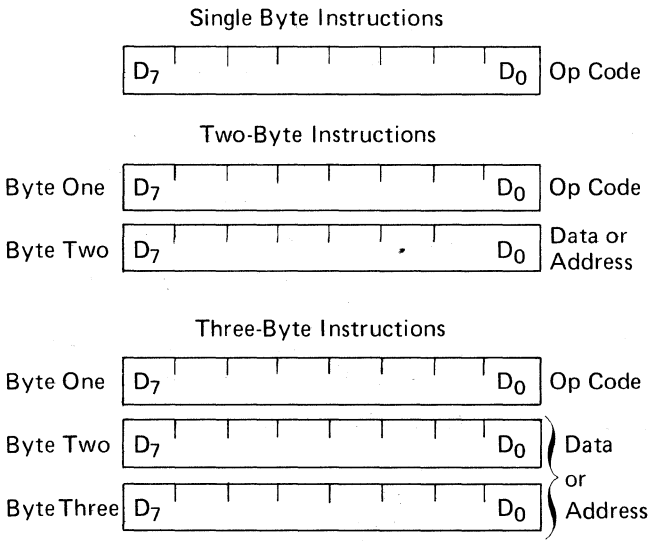
APPENDIX A INSTRUCTION SET SUMMARY

Data in the 8080 is stored in the form of 8-bit binary integers:



DATA WORD

The 8080 program instructions may be one, two or three bytes in length. Multiple byte instructions must be stored in successive memory locations; the address of the first byte is always used as the address of the instruction. The exact instruction format will depend on the particular operation to be executed.



Addressing Modes:

Often the data that is to be operated on is stored in memory. When multi-byte numeric data is used, the data, like instructions, is stored in successive memory locations, with the least significant byte first, followed by increasingly significant bytes. The 8080 has four different modes for addressing data stored in memory or in registers.

- Direct – Bytes 2 and 3 of the instruction contain the exact memory address of the data item (the low-order bits of the address are in byte 2, the high-order bits in style 3).
- Register – The instruction specifies the register or register-pair in which the data is located.
- Register Indirect – The instruction specifies a register-pair which contains the memory address where the data is located (the high-order bits of the address are in the first register of the pair, the low-order bits in the second).
- Immediate – The instruction contains the data itself. This is either an 8-bit quantity or a 16-bit quantity (least significant byte first, most significant byte second).

Unless directed by an interrupt or branch instruction, the execution of instructions proceeds through consecutively increasing memory locations. A branch instruction can specify the address of the next instruction to be executed in one of two ways:

- Direct – The branch instruction contains the address of the next instruction to be executed. (Except for the 'RST' instruction, byte 2 contains the low-order address and byte 3 the high-order address.)
- Register Indirect – The branch instruction indicates a register-pair which contains the address of the next instruction to be executed. (The high-order bits of the address are in the first register of the pair, the low-order bits in the second.)

The RST instruction is a special one-byte call instruction (usually used during interrupt sequences). RST includes a three-bit field; program control is transferred to the instruction whose address is eight times the contents of this three-bit field.

Condition Flags:

There are five condition flags associated with the execution of instructions on the 8080. They are Zero, Sign, Parity, Carry, and Auxiliary Carry, and are each represented by a 1-bit register in the CPU. A flag is "set" by forcing the bit to 1; "reset" by forcing the bit to 0.

Unless indicated otherwise, when an instruction affects a flag, it affects it in the following manner:

Zero: If the result of an instruction has the value 0, this flag is set; otherwise it is reset.

Sign: If the most significant bit of the result of the result of the operation has the value 1, this flag is set; otherwise it is reset.

Parity: If the modulo 2 sum of the bits of the result of the operation is 0, (i.e., if the result has even parity), this flag is set; otherwise it is reset (i.e., if the result has odd parity).

Carry: If the instruction resulted in a carry (from addition or incrementation) or a borrow (from subtraction, decrementation, or comparison) out of the high-order bit, this flag is set; otherwise it is reset.

Auxiliary Carry: If the instruction caused a carry out of bit 3 and into bit 4 of the resulting value, the auxiliary carry is set; otherwise it is reset. This flag is affected by single precision additions: subtractions, increments, decrements, comparisons, and logical operations, but is principally used with additions and increments preceding a DAA (Decimal Adjust Accumulator) instruction.

Symbols and Abbreviations:

The following symbols and abbreviations are used in the subsequent description of the 8080 instructions:

SYMBOL	MEANING
Accumulator	Register A
addr	16-bit address quantity
data	8-bit data quantity
data 16	16-bit data quantity
byte 2	The second byte of the instruction
byte 3	The third byte of the instruction
r, r1, r2	One of the registers A,B,C,D,E,H,L
DDD,SSS	The bit pattern for one of registers A,B, C,D,E,H,L (DDD = destination, SSS = source):

DDD or SSS	REGISTER NAME
000	A
001	B
010	C
011	D
100	E
101	H
110	L

rp — One of the register pairs:

B represents the B,C pair with B as the high-order register and C as the low-order register;

D represents the D,E pair with D as the high-order register and E as the low-order register;

H represents the H,L pair with H as the high-order register and L as the low-order register;

SP represents the 16-bit stack pointer register.

RP — The bit pattern for one of the register pairs B,D,H,SP:

RP	REGISTER PAIR
00	B-C
01	D-E
10	H-L
11	SP

rh — The first (high-order) register of a designated register pair.

rl — The second (low-order) register of a designated register pair.

PC — 16-bit program counter register (PCH and PCL are used to refer to the high-order and low-order 8 bits respectively).

SP — 16-bit stack pointer register (SPH and SPL are used to refer to the high-order and low-order 8 bits respectively).

r_m — Bit m of the register r (bits are number 7 through 0 from left to right).

Z,S,P,CY,CA — The condition flags:

Zero,
Sign,
Parity,
Carry,
and Auxiliary Carry, respectively.

( ) — The contents of the memory location or registers enclosed in the parentheses.

← — "Is transferred to"

∧ — Logical product ("and")

∀ — Exclusive "or"

∨ — Inclusive "or"

+	Addition
-	Two's complement subtraction
⟷	"Is exchanged with"
—	The one's complement
n	The restart number 0 through 7
NNN	The binary representation 000 through 111 for restart number 0 and 7 respectively.

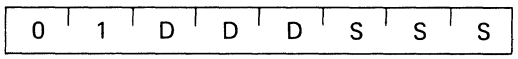
Data Transfer Group:

This group of instructions transfers data to and from registers and memory. Unless otherwise indicated, *condition flags are not affected* by any instructions in this group.

MOV r1, r2 (Move)

(r1) ← (r2)

The content of register r2 is moved to register r1.

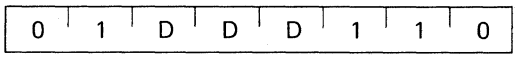


Cycles: 1
States: 5
Addressing: register
Flags: none

MOV r,M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

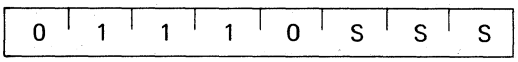


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

MOV M,r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

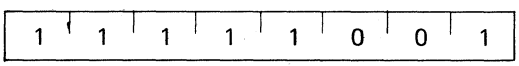


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

SPHL (Move HL to SP)

(SP) ← (H) (L)

The contents of registers H and L (16 bits) are moved to register SP.

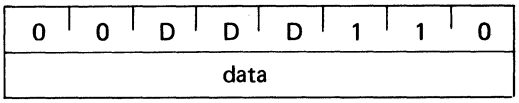


Cycles: 1
States: 5
Addressing: register
Flags: none

MVI r, data

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

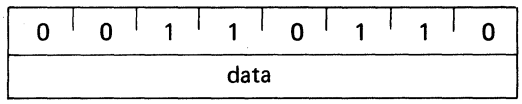


Cycles: 2
States: 7
Addressing: register immed.
Flags: none

MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

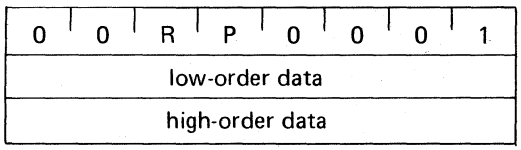


Cycles: 3
States: 10
Addressing: reg./ind. immed.
Flags: none

LXI rp, data 16 (Load register pair immediate)

(rh) ← (byte 3), (rl) ← (byte 2)

Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (r1) of the register pair rp.

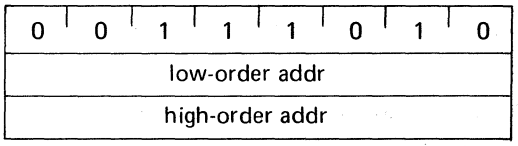


Cycles: 3
States: 10
Addressing: immediate
Flags: none

LDA addr (Load Accumulator direct)

(A) ← ((byte 3) (byte 2))

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.



Cycles: 4
States: 13
Addressing: direct
Flags: none

STA addr (Store Accumulator direct)

((byte 3) (byte 2)) ← (A)

The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles: 4
States: 13
Addressing: direct
Flags: none

LHLD addr (Load H and L direct)

(L) ← ((byte 3) (byte 2))

(H) ← ((byte 3) (byte 2) + 1)

The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

Cycles: 5
States: 16
Addressing: direct
Flags: none

SHLD addr (Store H and L direct)

((byte 3) (byte 2)) ← (L)

((byte 3) (byte 2) +1) ← (H)

The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cycles: 5
States: 16
Addressing: direct
Flags: none

LDAX rp (Load accumulator indirect)

(A) ← ((rp))

The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

0	0	R	P	1	0	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

STAX rp (Store accumulator indirect)

((rp)) ← (A)

The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs rp=B (registers B and C) or rp=D (registers D and E) may be specified.

0	0	R	P	0	0	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: none

XCHG (Exchange H and L with D and E)

(H) ↔ (D)

(L) ↔ (E)

The contents of registers H and L are exchanged with the contents of registers D and E.

1	1	1	0	1	0	1	1

Cycles: 1
States: 4
Addressing: register
Flags: none

Arithmetic Group:

This group of instructions performs arithmetic operations on data in registers and memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, and Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD r (Add)

(A) ← (A) + (r)

The content of register r is added to the content of the accumulator. The result if placed in the accumulator.

1	0	0	0	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADD M (Add from memory)

(A) ← (A) + ((H) (L))

The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.

1	1	0	0	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ADI data (Add immediate)

(A) ← (A) + (byte 2)

The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.

1	1	0	0	0	1	1	0

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ADC r (Add with carry)

(A) ← (A) + (r) + (CY)

The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.

1	0	0	0	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ADC M (Add from memory with carry)

(A) ← (A) + ((H) (L) + (CY))

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

1	0	0	0	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ACI data (Add with carry immediate)

(A) ← (A) + (byte 2) + (CY)

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed the accumulator.

1	1	0	0	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SUB r (Subtract)

(A) ← (A) – (r)

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SUB M (Subtract from memory)

(A) ← (A) – ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SUI data (Subtract immediate)

(A) ← (A) – (byte 2)

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	1	0	1	0	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

SBB r (Subtract with borrow)

(A) ← (A) – (r) – (CY)

The content of register r and the content of the CY flag are both subtracted from the accumulator The result is placed in the accumulator.

1	0	0	1	1	S	S	S

Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

SBB M (Subtract from memory with borrow)

(A) ← (A) – ((H) (L)) – (CY)

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	0	0	1	1	1	1	0

Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

SBI data (Subtract with borrow immediate)

(A) ← (A) – (byte 2) – (CY)

The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

1	1	0	1	1	1	1	0
data							

Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

INC r (Increment)

(r) ← (r) + 1

The content of register r is incremented by one. Note: All condition flags *except CY* are affected.

0	0	D	D	D	1	0	0

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

INC M (Increment memory)

((H) (L)) ← ((H) (L)) + 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags *except CY* are affected.

0	0	1	1	0	1	0	0

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,AC

DCR r (Decrement)

(r) ← (r) –1

The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags *except CY* are affected.

0	0	D	D	D	1	0	1

Cycles: 1
States: 5
Addressing: register
Flags: Z,S,P,AC

DCR M (Decrement memory)

((H) (L)) ← ((H) (L)) – 1

The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags *except CY* are affected.

0	0	1	1	0	1	0	1

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: X,S,P,AC

INX rp (Increment register pair)

(rh) (re) ← (RH) (R1) + 1

The content of the register pair rp is incremented by one. Note: *No condition flags are affected.*

0	0	R	P	0	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

DCX rp (Decrement register pair)

(rh) (r1) ← (rh) (r1) –1

The content of the register pair rp is decremented by one. Note: *No condition flags are affected.*

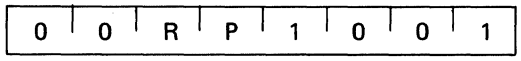
0	0	R	P	1	0	1	1

Cycles: 1
States: 5
Addressing: register
Flags: none

DAD rp (Add register pair to H and L)

(H) (L) ← (H) (L) + (rh) (re)

The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: *Only the CY is affected.* It is set if there is a carry out of the double precision add; otherwise it is reset.



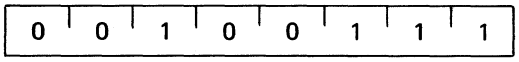
Cycles: 3
States: 10
Addressing: register
Flags: CY

DAA (Decimal Adjust Accumulator)

The eight-bit number in the accumulator is adjusted to form two four-bit binary-coded-decimal digits by the following process:

1. If the value of the least significant 4-bits of the accumulator is greater than 9 *or* if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4-bits of the accumulator is now greater than 9, *or* if the CY flag is set, 6 is added to the most significant 4-bits of the accumulator.

All flags are affected by the additions, if performed, otherwise they are reset.



Cycles: 1
States: 4
Addressing: –
Flags: Z,S,P,CY,AC

Logical Group:

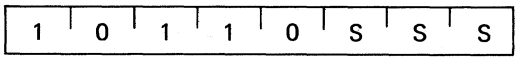
This group of instructions performs logical operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

ANA r (And)

(A) ← (A) ∧ (r)

The content of register r is logically anded with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

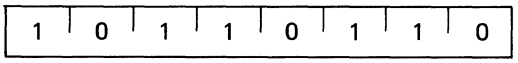


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ANA M (And from memory)

(A) ← (A) ∧ ((H) (L))

The contents of the memory location whose address is contained in the H and L registers is logically anded with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

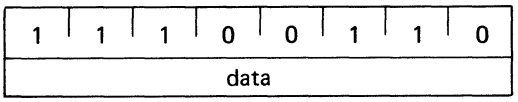


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ANA data (And immediate)

(A) ← (A) ∧ (byte 2)

The content of the second byte of the instruction is logically anded with the contents of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

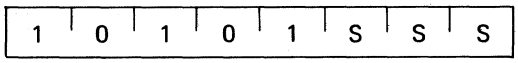


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

XRA r (Exclusive OR)

(A) ← (A) ∀ (r)

The content of register r is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

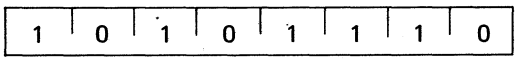


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

XRA M (Exclusive OR)

(A) ← (A) ∀ ((H) (L))

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

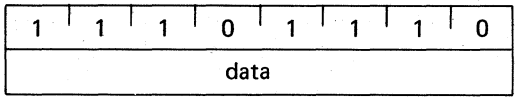


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XRI data (Exclusive or immediate)

(A) ← (A) ∀ (byte 2)

The content of the second byte of the instruction is exclusive-or'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

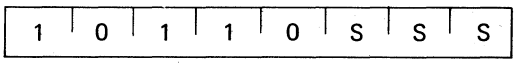


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

ORA r (OR)

(A) ← (A) V (r)

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

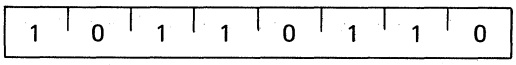


Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

ORA M (OR from memory)

(A) ← (A) V ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*

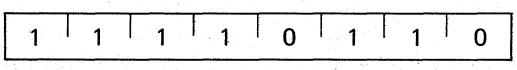


Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

ORI data (OR Immediate)

(A) ← (A) V (byte 2)

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. *The CY and AC flags are cleared.*



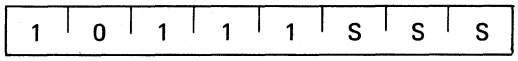
Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

CMP r (COMPARE)

(A) - (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. *The Z flag is set to 1 if (A), = (r). The CY flag is set to 1 if (r) ≠ (A).*

Note: The auxiliary carry is affected.



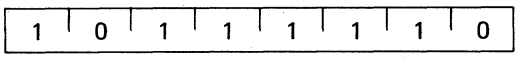
Cycles: 1
States: 4
Addressing: register
Flags: Z,S,P,CY,AC

CMP M (Compare with memory)

(A) - ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = ((H) (L)). ((H) (L)) ≠ A.

Note: *The AC flag is affected.*



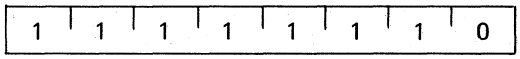
Cycles: 2
States: 7
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

CPI data (Compare immediate)

(A) - (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The accumulator is not changed. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = byte 2). The CY flag is set to 1 if byte 2) ≠ (A).

Note: *The AC flag is affected.*

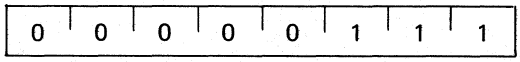


Cycles: 2
States: 7
Addressing: immediate
Flags: Z,S,P,CY,AC

RLC (Rotate left)

$(A_{n+1}) \leftarrow (A_n)$; $(A_0) \leftarrow (A_7)$; $(CY) \leftarrow (A_7)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. *Only the CY flag is affected.*

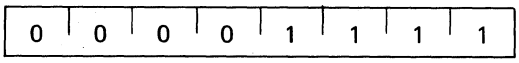


Cycles: 1
States: 1
Flags: CY

RRC (Rotate right)

$(A_n) \leftarrow A_{n-1}$; $(A_7) \leftarrow (A_0)$; $(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. *Only the CY flag is affected.*

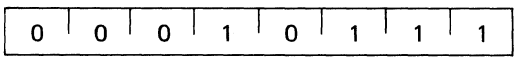


Cycles: 1
States: 4
Flags: CY

RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n)$; $(CY) \leftarrow (A_7)$; $(A_0) \leftarrow (CY)$

The content or the accumulator is rotated left one position through the carry. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. *Only the CY flag is affected.*

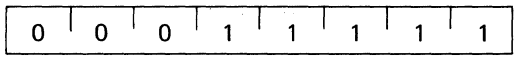


Cycles: 1
States: 4
Flags: CY

RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1})$; $(CY) \leftarrow (A_0)$; $(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is the CY flag. The high-order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the low order bit. *Only the CY flag is affected.*

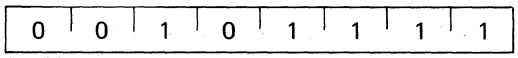


Cycles: 1
States: 4
Flags: CY

CMA (Complement accumulator)

$(A) \leftarrow (\overline{A})$

The contents of the accumulator are complemented (zero bits become 1, one bits become (0). *No flags are affected.*

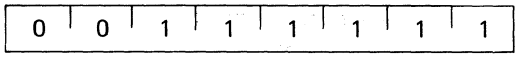


Cycles: 1
States: 4
Flags: none

CMC (Complement carry)

$(CY) \leftarrow (\overline{CY})$

The CY flag is complemented. *No other flags are affected.*

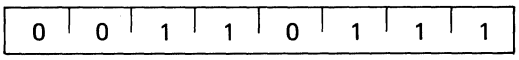


Cycles: 1
States: 4
Flags: CY

STC (Set carry)

$(CY) \leftarrow 1$

The CY flag is complemented. *No other flaps are affected.*



Cycles: 1
States: 4
Flags: CY

Branch Group:

This group of instructions alter normal sequential program flow.

Unless specified otherwise, no condition flags are affected by any instruction in this group.

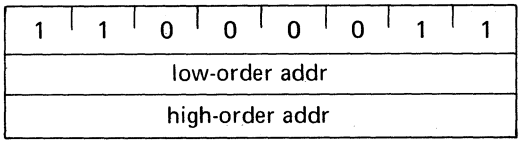
The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
C – carry (CY=1)	000
Z – zero (Z=1)	001
M – minus (5=1)	010
PE – parity even (P=1)	011
NC – no carry (CY=0)	100
NZ – not zero (Z=0)	101
P – plus (5=0)	110
PO – parity odd (P=0)	111

JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.



Cycles: 3
States: 10
Addressing: direct
Flags: none

Jcondition addr (Conditional jump)

If (CCC),
(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

1	1	C	C	C	0	1	0
low-order addr							
high-order addr							

Cycles: 3
States: 10
Addressing: direct
Flags: none

CALL addr (Call)

((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) –2
(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5
States: 17
Addressing: direct
Flags: none

Ccondition addr (Conditional call)

If (CC),
((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← (byte 3) (byte 2)

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	1	0	0
low-order addr							
high-order addr							

Cycles: 3/5
States: 11/17
Addressing: direct
Flags: none

RET (Return)

(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register PS is moved to the high-order eight bits of register PC. The content of register PC is incremented by 2.

1	1	0	0	1	0	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

Rcondition (Conditioned return)

If (CC),
(PCL) ← ((SP))
(PCH) ← ((SP) + 1)
(SP) ← (SP) + 2

If the specified condition is true, the actions specified in the RET instruction (see above are performed; otherwise; control continues sequentially.

1	1	C	C	C	0	0	0

Cycles: 1/3
States: 5/11
Addressing: reg. indirect
Flags: none

RST n (Restart)

((SP) – 1) ← (PCH)
((SP) – 2) ← (PCL)
(SP) ← (SP) – 2
(PC) ← 8* (NNN)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of (NNN).

1	1	N	N	N	1	1	1

Cycles: 3
States: 11
Addressing: direct
Flags: none

PCHL (Jump H and L indirect – move H and L to PC)

(PCH) ← (H)
(PCL) ← (L)

The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.

1	1	1	0	1	0	0	1

Cycles: 1
States: 5
Addressing: register
Flags: none

Stack, I/O, and Machine Control Group:

This group of instructions performs I/O, manipulates the "stack", and alters internal control flags.

Unless otherwise specified, *no condition flags are affected by any instructions in this group.*

PUSH rp (Push)

((SP) – 1) ← (rh)
((SP) – 2) ← (r1)
(SP) ← (SP) – 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2.

Note: Register pair rp = SP may not be specified.

1	1	R	P	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

PUSH PSW (Push processor status word)

((SP) – 1) ← (A)
((SP) – 2)$_0$ ← (CY), ((SP) – 2)$_1$ ← 1
((SP) – 2)$_2$ ← (P), ((SP) – 2)$_3$ ← 0
((SP) – 2)$_4$ ← (AC), ((SP) – 2)$_5$ ← 0
((SP) –2)$_6$ ← (Z), ((SP) –2)$_7$ ← (S)
(SP) ← (SP) – 2

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register PS is decremented by two.

1	1	1	1	0	1	0	1

Cycles: 3
States: 11
Addressing: reg. indirect
Flags: none

POP rp (Pop)

(r1) ← ((SP))
(rh) ← ((SP) + 1)
(SP) ← (SP) + 2

The content of the memory location, whose address specified by the content of register SP, is moved to the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register PS is incremented by 2.

Note: Register pair rp = SP may not be specified.

1	1	R	P	0	0	0	1

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: none

POP PSW (Pop processor status word)

(CY) ← ((SP))$_0$
(P) ← ((SP))$_2$
(AC) ← ((SP))$_4$
(Z) ← ((SP))$_6$
(S) ← ((SP))$_7$
(A) ← ((SP) + 1)
(SP) ← ((SP + 2)

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

1	1	1	1	0	0	0	1

Cycles: 3
States: 10
Addressing: reg. indirect
Flags: Z,S,P,CY,AC

XTHL (Exchange stack top with H and L)

(L) ← ((SP))
(H) ← ((SP) + 1)

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

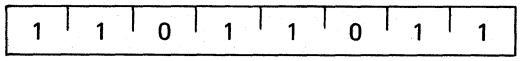
1	1	1	0	0	0	1	1

Cycles: 5
States: 18
Addressing: reg. indirect
Flags: none

IN port (Input)

(A) ← (data)

The data placed on the eight-bit bi-directional data bus by the specified port is moved to register A.



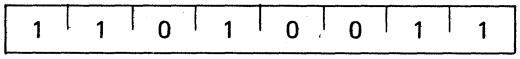
port

Cycles: 3
States: 10
Addressing: direct
Flags: none

OUT port (Output)

(data) ← (A)

The content of register A is placed on the eight-bit bi-directional data bus for transmission to the specified port.

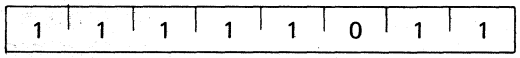


port

Cycles: 3
States: 10
Addressing: direct
Flags: none

EI (Enable interrupt)

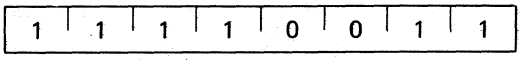
The interrupt system is enabled following the execution of the *next* instruction.



Cycles: 1
States: 4
Flags: none

DI (Disable interrupt)

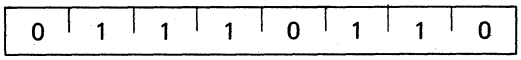
The interrupt system is disabled *immediately* following the execution of the instruction.



Cycles: 1
States: 4
Flags: none

HLT (Halt)

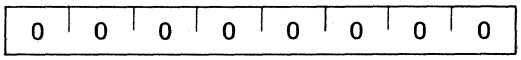
The processor is stopped. The registers and flags are unaffected.



Cycles: 1
States: 4
Flags: none

NOP (No op)

No operation is performed. The registers and flags are unaffected.



Cycles: 1
States: 4
Flags: none

APPENDIX B
ELECTRICAL CHARACTERISTICS OF LOGIC ELEMENTS USED IN THE INTELLEC 8/MOD 80

Silicon Gate MOS **8080**

SINGLE CHIP 8-BIT N-CHANNEL MICROPROCESSOR

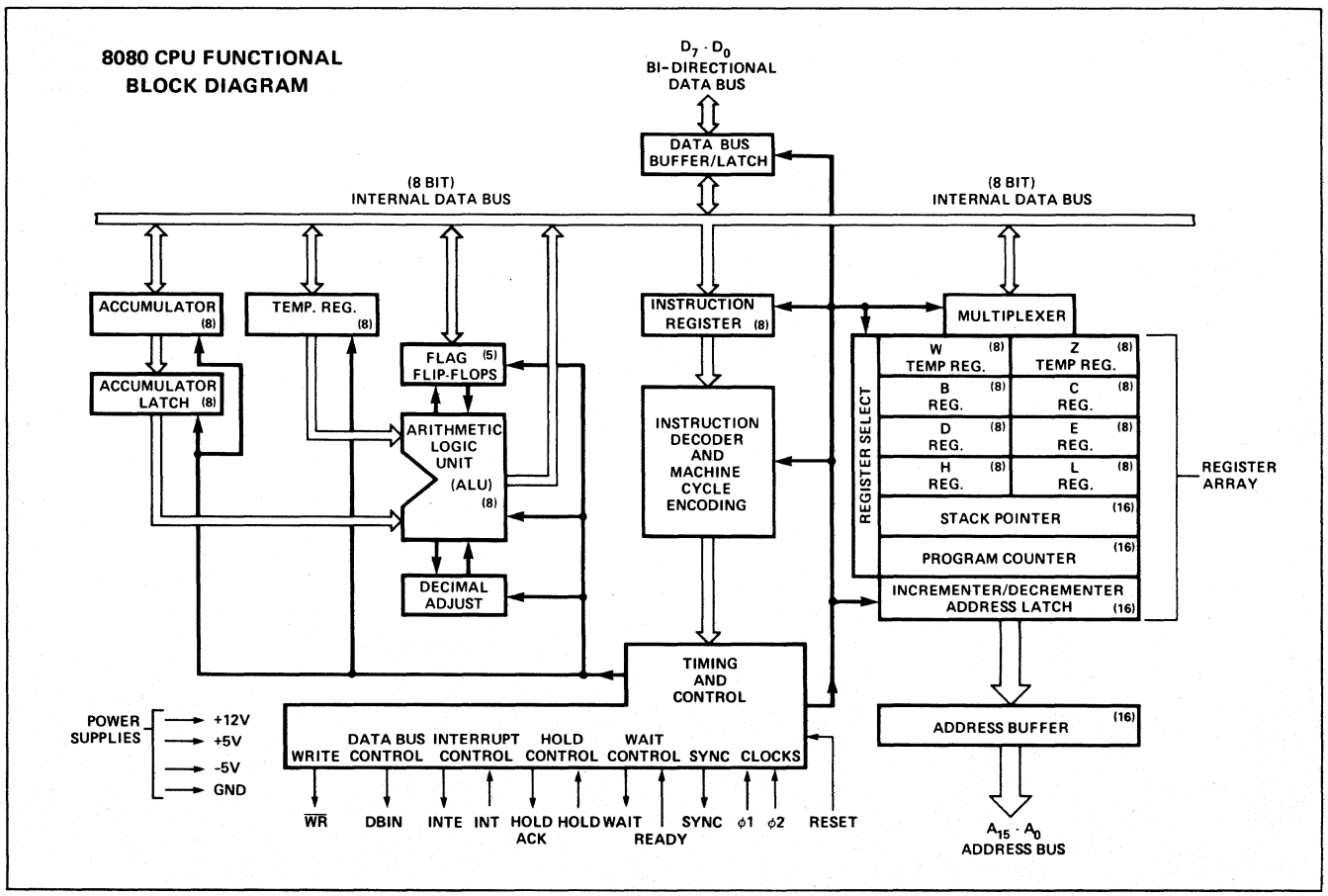
- **2 μs Instruction Cycle**
- **Powerful Problem Solving Instruction Set**
- **Six General Purpose Registers and an Accumulator**
- **Sixteen Bit Program Counter for Directly Addressing up to 64K Bytes of Memory**
- **Sixteen Bit Stack Pointer and Stack Manipulation Instructions for Rapid Switching of the Program Environment**
- **Decimal, Binary and Double Precision Arithmetic**
- **Ability to Provide Priority Vectored Interrupts**
- **512 Directly Addressed I/O Ports**

The Intel 8080 is a complete 8-bit parallel central processing unit (CPU). It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process. This offers the user a high performance solution to control and processing applications.

The 8080 contains six 8-bit general purpose working registers and an accumulator. The six general purpose registers may be addressed individually or in pairs providing both single and double precision operators. Arithmetic and logical instructions set or reset four testable flags. A fifth flag provides decimal arithmetic operation.

The 8080 has an external stack feature wherein any portion of memory may be used as a last in/first out stack to store/retrieve the contents of the accumulator, flags, program counter and all of the six general purpose registers. The sixteen bit stack pointer controls the addressing of this external stack. This stack gives the 8080 the ability to easily handle multiple level priority interrupts by rapidly storing and restoring processor status. It also provides almost unlimited subroutine nesting.

This microcoprocessor has been designed to simplify systems design. Separate 16-line address and 8-line bidirectional data busses are used to facilitate easy interface to memory and I/O. Signals to control the interface to memory and I/O are provided directly by the 8080. Ultimate control of the address and data busses resides with the HOLD signal. It provides the ability to suspend processor operation and force the address and data busses into a high impedance state. This permits OR-tying these busses with other controlling devices for (DMA) direct memory access or multi-processor operation.



©Intel Corp. 1974

8080 FUNCTIONAL PIN DEFINITION

The following describes the function of all of the 8080 I/O pins. Several of the descriptions refer to internal timing periods.[1]

A_{15}-A_0 (output three-state)
ADDRESS BUS; the address bus provides the address to memory (up to 64K 8-bit words) or denotes the I/O device number for up to 256 input and 256 output devices. A_0 is the least significant address bit.

D_7-D_0 (input/output three-state)
DATA BUS; the data bus provides bidirectional communication between the CPU, memory, and I/O devices for instructions and data transfers. D_0 is the least significant bit.

SYNC (output)
SYNCHRONIZING SIGNAL; the SYNC pin provides a signal to indicate the beginning of each machine cycle.

DBIN (output)
DATA BUS IN; the DBIN signal indicates to external circuits that the data bus is in the input mode. This signal should be used to enable the gating of data onto the 8080 data bus from memory or I/O.

READY (input)
READY; the READY signal indicates to the 8080 that valid memory or input data is available on the 8080 data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080 does not receive a READY input, the 8080 will enter a WAIT state for as long as the READY line is low.

WAIT (output)
WAIT; the WAIT signal acknowledges that the CPU is in a WAIT state.

$\overline{WR}$ (output)
WRITE; the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus is stable while the $\overline{WR}$ signal is active ($\overline{WR}$ = 0).

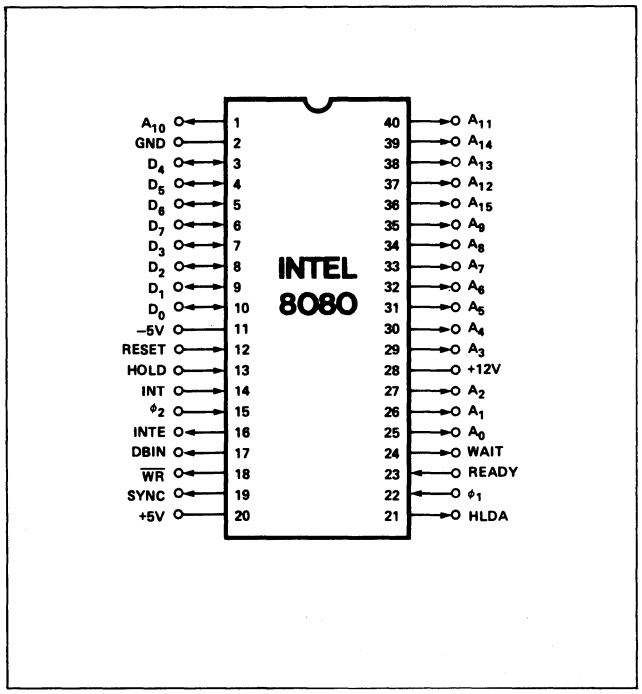
HOLD (input)
HOLD; the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080 address and data bus as soon as the 8080 has completed its use of these buses for the current machine cycle. It is recognized under the following conditions:

- the CPU is in the HALT state.
- the CPU is in the T2 or TW state and the READY signal is active.

As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

HLDA (output)
HOLD ACKNOWLEDGE; the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at:

- T3 for READ memory or input.
- The Clock Period following T3 for WRITE memory or OUTPUT operation.



In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.

INTE (output)
INTERRUPT ENABLE; indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the Enable and Disable Interrupt instructions and inhibits interrupts from being accepted by the CPU when it is reset. It is automatically reset (disabling further interrupts) at time T1 of the instruction fetch cycle (M1) when an interrupt is accepted and is also reset by the RESET signal.

INT (input)
INTERRUPT REQUEST; the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request.

RESET (input)[2]
RESET; while the RESET signal is activated, the content of the program counter is cleared and the instruction register is set to 0. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset. Note that the flags, accumulator, and registers are not cleared.

V_{SS}	Ground Reference.
V_{dd}	+12 ± 5% Volts.
V_{cc}	+5 ± 5% Volts.
V_{bb}	-5 ±5% Volts (substrate bias).
ϕ_1, ϕ_2	2 externally supplied clock phases. (non TTL compatible)

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	0°C to +70° C
Storage Temperature	-65°C to +150°C
All Input or Output Voltages With Respect to V_{BB}	-0.3V to +20V
V_{CC}, V_{DD} and V_{SS} With Respect to V_{BB}	-0.3V to +20V
Power Dissipation	1.5W

*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

T_A = 0°C, to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted.

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	V_{SS}−1		V_{SS}+0.6	V	
V_{IHC}	Clock Input High Voltage	V_{DD}−1		V_{DD}+1	V	
V_{IL}	Input Low Voltage	V_{SS}−1		V_{SS}+0.8	V	
V_{IH}	Input High Voltage	3.3		V_{CC}+1	V	
V_{OL}	Output Low Voltage			0.45	V	I_{OL} = 1.7mA on the Data Bus I_{OL} = .75mA on all other outputs
V_{OH}	Output High Voltage	3.7			V	I_{OH} = 100μA.
$I_{DD\,(AV)}$	Avg. Power Supply Current (V_{DD})		40	67	mA	Operation
$I_{CC\,(AV)}$	Avg. Power Supply Current (V_{CC})		60	75	mA	T_A = 25°C
$I_{BB\,(AV)}$	Avg. Power Supply Current (V_{BB})		.01	1	mA	T_{CY} = .48 μsec
I_{IL}	Input Leakage			±10	μA	$V_{SS} \leqslant V_{IN} \leqslant V_{CC}$
I_{CL}	Clock Leakage			±10	μA	$V_{SS} \leqslant V_{CLOCK} \leqslant V_{DD}$
I_{DL} [3]	Data Bus Leakage in Input Mode			-100	μA	$V_{SS} \leqslant V_{IN} \leqslant V_{CC}$
I_{FL}	Address and Data Bus Leakage During HOLD			+10 -100	μA	$V_{ADDR/DATA} = V_{CC}$ $V_{ADDR/DATA} = V_{SS}$

CAPACITANCE

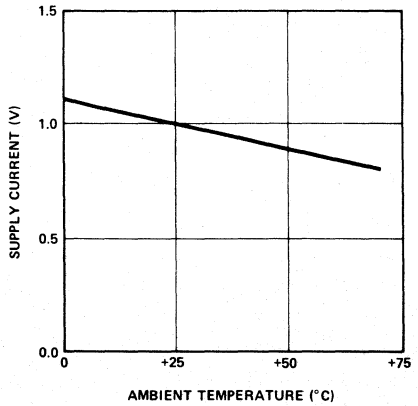
T_A = 25°C $V_{CC} = V_{DD} = V_{SS}$ = 0V, V_{SS} = -5V ±5%

Symbol	Parameter	Typ.	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	10	20	pf	f_c = 1 MHz
C_{IN}	Input Capacitance	5	10	pf	Unmeasured Pins
C_{OUT}	Output Capacitance	10	20	pf	Returned to V_{SS}

NOTES:

1. For definitions the user is directed to the following publications:
   A. Programming Manual for the 8080 Microcomputer System.
   B. 8080 Microcomputer Users Manual.
   C. From CPU to Software.
2. The RESET signal must be active for a minimum of 3 clock cycles.
3. When DBIN is high and $V_{IN} > V_{IH}$ an active pull up of nominally 2kΩ will be switched onto the Data Bus.
4. ΔI supply / ΔT_A = -0.45%/°C.

TYPICAL SUPPLY CURRENT VS. TEMPERATURE, NORMALIZED. [4]



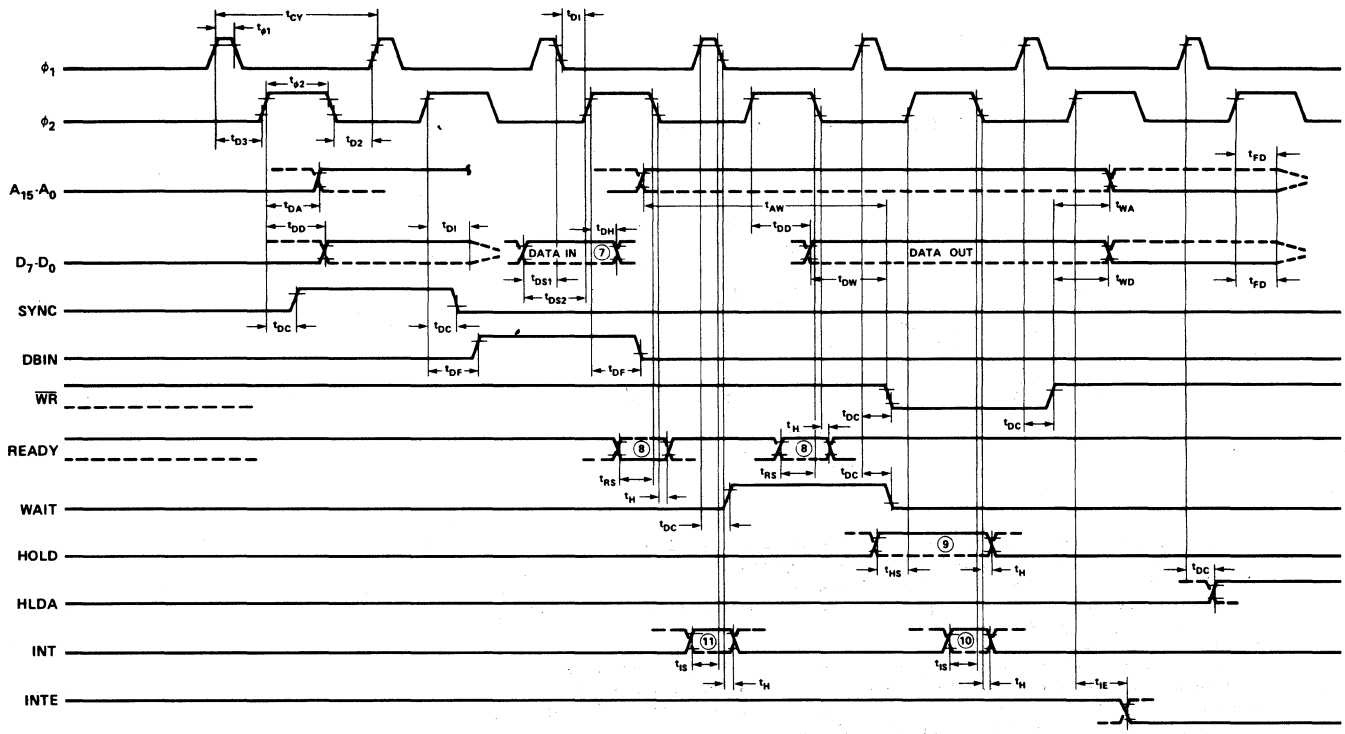
A.C. CHARACTERISTICS

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{CY}[3]	Clock Period	0.48	2.0	μsec	
t_r, t_f	Clock Rise and Fall Time	5	50	nsec	
$t_{\phi 1}$	ϕ_1 Pulse Width	60		nsec	
$t_{\phi 2}$	ϕ_2 Pulse Width	220		nsec	
t_{D1}	Delay ϕ_1 to ϕ_2	0		nsec	
t_{D2}	Delay ϕ_2 to ϕ_1	70		nsec	
t_{D3}	Delay ϕ_1 to ϕ_2 Leading Edges	130		nsec	
t_{DA} [2]	Address Output Delay From ϕ_2		200	nsec	R_L = 4.5kΩ, C_L = 100pf
t_{DD} [2]	Data Output Delay From ϕ_2		220	nsec	R_L = 2.1kΩ, C_L = 100pf
t_{DC} [2]	Signal Output Delay From ϕ_1, or ϕ_2 (SYNC, $\overline{WR}$ WAIT HLDA)		120	nsec	R_L = 4.5kΩ, C_L = 50pf
t_{DF} [2]	DBIN Delay From ϕ_2	25	140	nsec	R_L = 2.1kΩ, C_L = 50pf
t_{DI}[1]	Delay for Input Bus to Enter Input Mode During DBIN		t_{DF}	nsec	
t_{DS1}	Data "Setup Time" During ϕ_1 and DBIN	50		nsec	

TIMING WAVEFORMS [12]

(Note: Timing measurements are made at the following reference voltages: CLOCK "1" = 9.5V, "0" = 1.0V; INPUTS "1" = 3.3V, "0" = 0.8V; OUTPUTS "1" = 2.0V, "0" = 0.8V.)

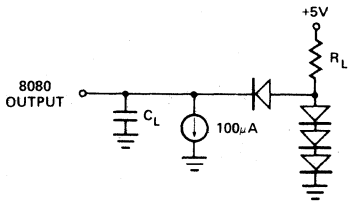


A.C. CHARACTERISTICS (Continued)

T_A = 0°C to 70°C, V_{DD} = +12V ± 5%, V_{CC} = +5V ± 5%, V_{BB} = -5V ± 5%, V_{SS} = 0V, Unless Otherwise Noted

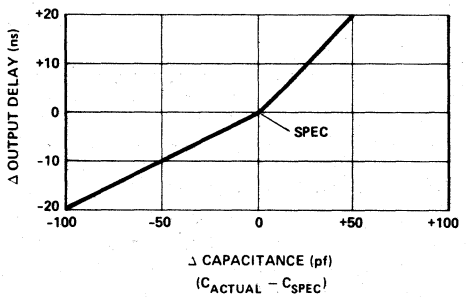
Symbol	Parameter	Min.	Max.	Unit	Test Condition
t_{DS2}	Data "Setup Time" to ϕ_2 During DBIN	150		nsec	
t_{DH} [1]	Data "Hold Time" From ϕ_2 During DBIN	t_{DF}		nsec	
t_{IE} [2]	INTE Output Delay From ϕ_2		200	nsec	R_L = 4.5kΩ, C_L = 50pf
t_{RS}	Ready "Setup Time" During ϕ_2	120		nsec	
t_{HS}	Hold "Setup Time" to ϕ_2	140		nsec	
t_{IS}	INT "Setup Time" During ϕ_2 (During ϕ_1 in Halt Mode)	180		nsec	
t_H	"Hold Time" From ϕ_2 (Ready, INT, Hold)	0		nsec	
t_{FD}	Delay to Float During Hold (Address and DATA BUS)		120	nsec	
t_{WA} [2]	Address Stable From $\overline{WR}$	t_{D3}		nsec	R_L = 4.5kΩ, C_L = 100pf
t_{AW} [2]	Address Stable Prior to $\overline{WR}$	[5]		nsec	R_L = 4.5kΩ, C_L = 100pf
t_{WD} [2]	Output Data Stable From $\overline{WR}$	t_{D3}		nsec	R_L = 2.1kΩ, C_L = 100pf
t_{DW} [2]	Output Data Stable Prior to $\overline{WR}$	[6]		nsec	R_L = 2.1kΩ, C_L = 100pf

NOTES: 1. Data input should be enabled with DBIN status. No bus conflict can then occur and data hold time is assured.
2. Load circuit



3. $t_{CY} = t_{D3} + t_{\phi 2} + t_{D2} + t_{f\phi 2} + t_{r\phi 1} + t_{f\phi 1} \geq 480$ns.

TYPICAL Δ OUTPUT DELAY VS. Δ CAPACITANCE



4. The following are relevant when interfacing the 8080 to devices having V_{IH} = 3.3V:
   a) Maximum output rise time from .8V to 3.3V = 140ns @ C_L = SPEC.
   b) Output delay when measured to 3.0V = SPEC + 60ns @ C_L = SPEC.
   c) If $C_L \neq$ SPEC add .6ns/pf if $C_L > C_{SPEC}$, subtract .3ns/pf (from modified delay) if $C_L < C_{SPEC}$.
5. $t_{AW} = 2\,t_{CY} - t_{D3} - t_{r\phi 2} - 120$nsec.
6. $t_{DW} = t_{CY} - t_{D3} - t_{r\phi 2} - 150$nsec.
7. Data in must be stable for this period during DBIN $^*T_3$. Both t_{DS1} and t_{DS2} must be satisfied.
8. Ready signal must be stable for this period during T_2 or T_W. (Must be externally synchronized.)
9. Hold signal must be stable for this period during T_2 or T_W when entering hold mode, and during T_3, T_4, T_5 and T_{WH} when in hold mode. (Must be externally synchronized.)
10. Interrupt signal must be stable during this period of the last clock cycle of any instruction to be recognized on the following instruction. (External synchronization is not required.)
11. During halt mode only, timing is with respect to ϕ_1 falling edge.
12. This timing diagram shows timing relationships only, it does not represent any specific machine cycle.

INSTRUCTION SET

The accumulator group instructions include ARITHMETIC and LOGICAL OPERATORS with DIRECT, INDIRECT, AND IMMEDIATE addressing modes.

MOVE, LOAD, and STORE instruction groups provide the ability to move either 8 or 16 bits of data between memory, the six working registers and the accumulator using DIRECT, INDIRECT, and IMMEDIATE addressing modes.

The ability to branch to different portions of the program is provided with JUMP, JUMP CONDITIONAL, and COMPUTED JUMPS. Also the ability to CALL to and RETURN from subroutines is provided both conditionally and unconditionally. The RESTART (or single byte call instruction) is useful for interrupt vector operation.

Double precision operators such as STACK manipulation and DOUBLE ADD instructions extend both the arithmetic and interrupt handling capability of the 8080. The ability to INCREMENT and DECREMENT memory, the six general registers and the accumulator is provided as well as EXTENDED INCREMENT and DECREMENT instructions to operate on the register pairs and stack pointer. Further capability is provided by the ability to ROTATE the accumulator LEFT or RIGHT through or around the carry bit.

Input and output may be accomplished using memory addresses as I/O ports or the directly addressed I/O provided for in the 8080 instruction set.

The following special instruction group completes the 8080 instruction set: the NO-OP instruction, HALT to stop processor execution and the DAA instructions provide decimal arithmetic capability. STC allows the carry flag to be directly set, and the CMC instruction allows it to be complemented. CMA complements the contents of the accumulator and XCHG exchanges the contents of two 16-bit register pairs directly.

Data and Instruction Formats

Data in the 8080 is stored in the form of 8-bit binary integers. All data transfers to the system data bus will be in the same format.

$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$

DATA WORD

The program instructions may be one, two, or three bytes in length. Multiple byte instructions must be stored in successive words in program memory. The instruction formats then depend on the particular operation executed.

One Byte Instructions — TYPICAL INSTRUCTIONS

$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OP CODE	Register to register, memory reference, arithmetic or logical, rotate return, PUSH, POP, ENABLE or DISABLE INTERRUPT INSTRUCTIONS

Two Byte Instructions

$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OP CODE	
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OPERAND	Immediate mode or I/O instructions

Three Byte Instructions

$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	OP CODE	JUMP, CALL or DIRECT LOAD AND STORE INSTRUCTIONS
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	LOW ADDRESS OR OPERAND 1	
$D_7\ D_6\ D_5\ D_4\ D_3\ D_2\ D_1\ D_0$	HIGH ADDRESS OR OPERAND 2	

For the 8080 a logic "1" is defined as a high level and a logic "0" is defined as a low level.

INSTRUCTION SET

Summary of Processor Instructions

Mnemonic	Description	Instruction Code[1] D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Clock[2] Cycles
$MOV_{r1,r2}$	Move register to register	0	1	D	D	D	S	S	S	5
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
HLT	Halt	0	1	1	1	0	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
INR r	Increment register	0	0	D	D	D	1	0	0	5
DCR r	Decrement register	0	0	D	D	D	1	0	1	5
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
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
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
ANA r	And register with A	1	0	1	0	0	S	S	S	4
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
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
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
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
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
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
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
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
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
JMP	Jump unconditional	1	1	0	0	0	0	1	1	10
JC	Jump on carry	1	1	0	1	1	0	1	0	10
JNC	Jump on no carry	1	1	0	1	0	0	1	0	10
JZ	Jump on zero	1	1	0	0	1	0	1	0	10
JNZ	Jump on no zero	1	1	0	0	0	0	1	0	10
JP	Jump on positive	1	1	1	1	0	0	1	0	10
JM	Jump on minus	1	1	1	1	1	0	1	0	10
JPE	Jump on parity even	1	1	1	0	1	0	1	0	10
JPO	Jump on parity odd	1	1	1	0	0	0	1	0	10
CALL	Call unconditional	1	1	0	0	1	1	0	1	17
CC	Call on carry	1	1	0	1	1	1	0	0	11/17
CNC	Call on no carry	1	1	0	1	0	1	0	0	11/17
CZ	Call on zero	1	1	0	0	1	1	0	0	11/17
CNZ	Call on no zero	1	1	0	0	0	1	0	0	11/17
CP	Call on positive	1	1	1	1	0	1	0	0	11/17
CM	Call on minus	1	1	1	1	1	1	0	0	11/17
CPE	Call on parity even	1	1	1	0	1	1	0	0	11/17
CPO	Call on parity odd	1	1	1	0	0	1	0	0	11/17
RET	Return	1	1	0	0	1	0	0	1	10
RC	Return on carry	1	1	0	1	1	0	0	0	5/11
RNC	Return on no carry	1	1	0	1	0	0	0	0	5/11
RZ	Return on zero	1	1	0	0	1	0	0	0	5/11
RNZ	Return on no zero	1	1	0	0	0	0	0	0	5/11
RP	Return on positive	1	1	1	1	0	0	0	0	5/11
RM	Return on minus	1	1	1	1	1	0	0	0	5/11
RPE	Return on parity even	1	1	1	0	1	0	0	0	5/11
RPO	Return on parity odd	1	1	1	0	0	0	0	0	5/11
RST	Restart	1	1	A	A	A	1	1	1	11
IN	Input	1	1	0	1	1	0	1	1	10
OUT	Output	1	1	0	1	0	0	1	1	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
PUSH B	Push register Pair B & C on stack	1	1	0	0	0	1	0	1	11
PUSH D	Push register Pair D & E on stack	1	1	0	1	0	1	0	1	11
PUSH H	Push register Pair H & L on stack	1	1	1	0	0	1	0	1	11
PUSH PSW	Push A and Flags on stack	1	1	1	1	0	1	0	1	11
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
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
XCHG	Exchange D & E, H & L Registers	1	1	1	0	1	0	1	1	4
XTHL	Exchange top of stack, H & L	1	1	1	0	0	0	1	1	18
SPHL	H & L to stack pointer	1	1	1	1	1	0	0	1	5
PCHL	H & L to program counter	1	1	1	0	1	0	0	1	5
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
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
INX B	Increment B & C registers	0	0	0	0	0	0	1	1	5
INX D	Increment D & E registers	0	0	0	1	0	0	1	1	5
INX H	Increment H & L registers	0	0	1	0	0	0	1	1	5
INX SP	Increment stack pointer	0	0	1	1	0	0	1	1	5
DCX B	Decrement B & C	0	0	0	0	1	0	1	1	5
DCX D	Decrement D & E	0	0	0	1	1	0	1	1	5
DCX H	Decrement H & L	0	0	1	0	1	0	1	1	5
DCX SP	Decrement stack pointer	0	0	1	1	1	0	1	1	5
CMA	Compliment A	0	0	1	0	1	1	1	1	4
STC	Set carry	0	0	1	1	0	1	1	1	4
CMC	Compliment carry	0	0	1	1	1	1	1	1	4
DAA	Decimal adjust A	0	0	1	0	0	1	1	1	4
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
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

NOTES: 1. DDS or SSS – 000 B – 001 C – 010 D – 011 E – 100 H – 101 L – 110 Memory – 111 A.
2. Two possible cycle times, (5/11) indicate instruction cycles dependent on condition flags.

Silicon Gate MOS **8102-2**

1024 BIT FULLY DECODED STATIC MOS RANDOM ACCESS MEMORY

- **Access Time — 850ns Max.**
- **Single +5 Volts Supply Voltage**
- **Directly TTL Compatible — All Inputs and Output**
- **Static MOS — No Clocks or Refreshing Required**
- **Low Power — Typically 150 mW**
- **Three-State Output — OR-Tie Capability**
- **Simple Memory Expansion — Chip Enable Input**
- **Fully Decoded — On Chip Address Decode**
- **Inputs Protected — All Inputs Have Protection Against Static Charge**
- **Low Cost Packaging — 16 Pin Plastic Dual-In-Line Configuration**

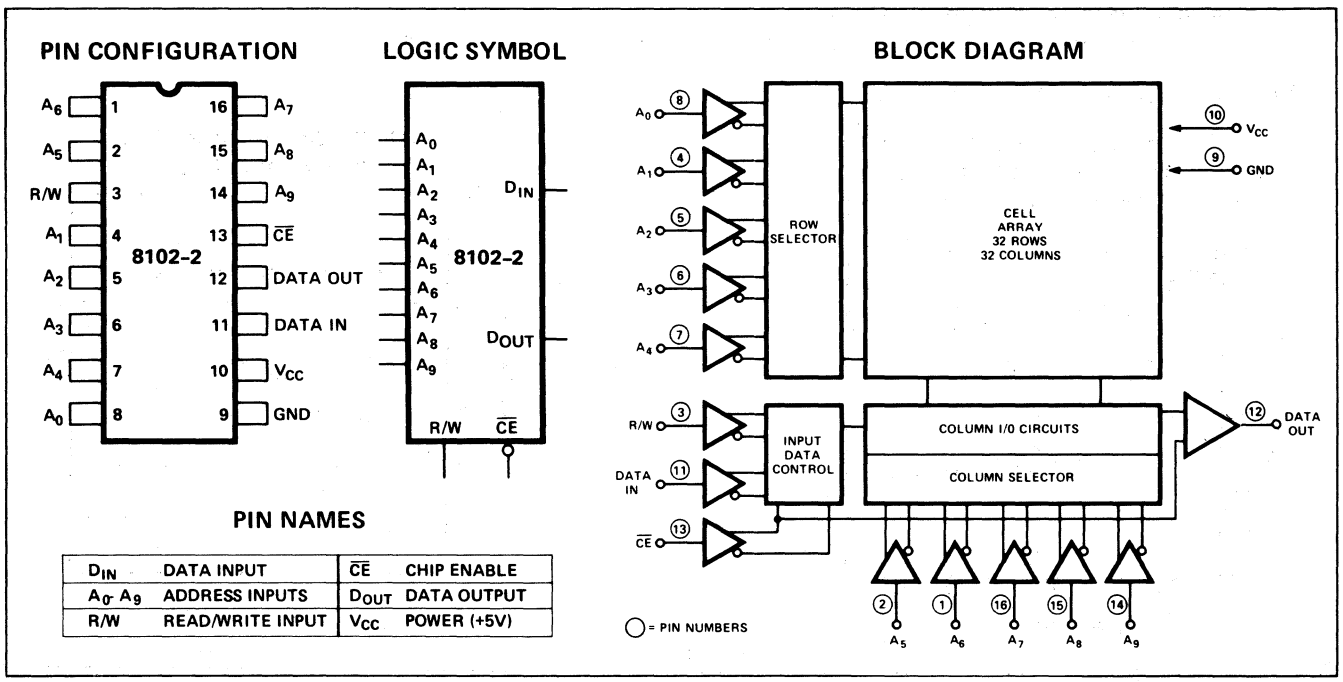
The Intel 8102-2 is a 1024 word by one bit static random access memory element using normally off N-channel MOS devices integrated on a monolithic array. It uses fully DC stable (static) circuitry and therefore requires no clocks or refreshing to operate. The data is read out nondestructively and has the same polarity as the input data.

The 8102-2 is designed for microcomputer 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, output, and a single +5 volt supply. A separate chip enable (CE) lead allows easy selection of an individual package when outputs are OR-tied.

The Intel 8102-2 is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance, easy-to-use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

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



PIN NAMES

D_{IN}	DATA INPUT	$\overline{CE}$	CHIP ENABLE
A_0-A_9	ADDRESS INPUTS	D_{OUT}	DATA OUTPUT
R/W	READ/WRITE INPUT	V_{CC}	POWER (+5V)

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	−0.5V to +7V
Power Dissipation	1 Watt

*COMMENT:

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition 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. AND OPERATING CHARACTERISTICS

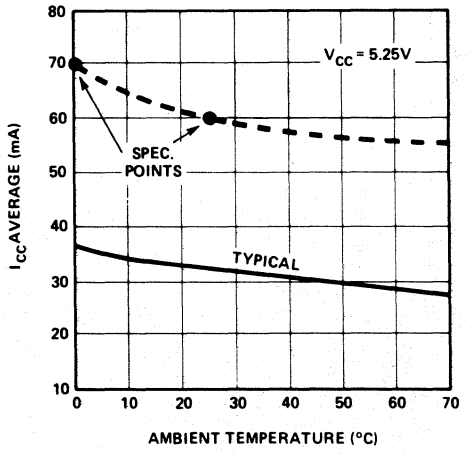
T_A = 0°C to +70°C, V_{CC} = 5V ±5% unless otherwise specified

SYMBOL	PARAMETER	LIMITS			UNIT	TEST CONDITIONS
		MIN.	TYP.(1)	MAX.		
I_{LI}	INPUT LOAD CURRENT (ALL INPUT PINS)			10	μA	V_{IN} = 0 to 5.25V
I_{LOH}	OUTPUT LEAKAGE CURRENT			10	μA	$\overline{CE}$ = 2.2V, V_{OUT} = 4.0V
I_{LOL}	OUTPUT LEAKAGE CURRENT			−100	μA	$\overline{CE}$ = 2.2V, V_{OUT} = 0.45V
I_{CC1}	POWER SUPPLY CURRENT		30	60	mA	ALL INPUTS = 5.25V DATA OUT OPEN T_A = 25°C
I_{CC2}	POWER SUPPLY CURRENT			70	mA	ALL INPUTS = 5.25V DATA OUT OPEN T_A = 0°C
V_{IL}	INPUT "LOW" VOLTAGE	−0.5		+0.65	V	
V_{IH}	INPUT "HIGH" VOLTAGE	2.2		V_{CC}	V	
V_{OL}	OUTPUT "LOW" VOLTAGE			+0.45	V	I_{OL} = 1.9mA
V_{OH}	OUTPUT "HIGH" VOLTAGE	2.2			V	I_{OH} = −100μA

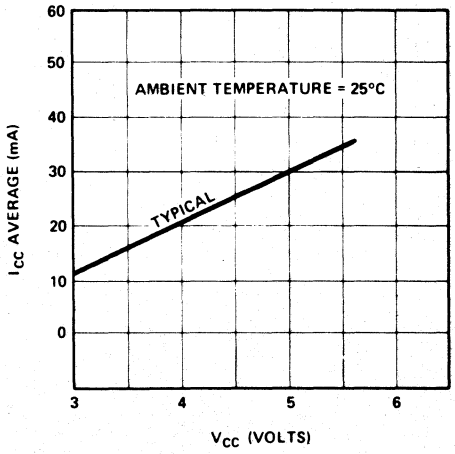
(1) Typical values are for T_A = 25°C and nominal supply voltage.

TYPICAL D.C. CHARACTERISTICS

POWER SUPPLY CURRENT VS. AMBIENT TEMPERATURE



POWER SUPPLY CURRENT VS. SUPPLY VOLTAGE



A.C. CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

SYMBOL	PARAMETER	LIMITS			UNIT
		MIN.	TYP.(1)	MAX.	
READ CYCLE					
t_{RC}	READ CYCLE	850			ns
t_A	ACCESS TIME		500	850	ns
t_{CO}	CHIP ENABLE TO OUTPUT TIME			500	ns
t_{OH1}	PREVIOUS READ DATA VALID WITH RESPECT TO ADDRESS	50			ns
t_{OH2}	PREVIOUS READ DATA VALID WITH RESPECT TO CHIP ENABLE	0			ns
WRITE CYCLE					
t_{WC}	WRITE CYCLE	850			ns
t_{AW}	ADDRESS TO WRITE SETUP TIME	200			ns
t_{WP}	WRITE PULSE WIDTH	600			ns
t_{WR}	WRITE RECOVERY TIME	50			ns
t_{DW}	DATA SETUP TIME	650			ns
t_{DH}	DATA HOLD TIME	100			ns
t_{CW}	CHIP ENABLE TO WRITE SETUP TIME	750			ns

(1) Typical values are for T_A=25°C and nominal supply voltage.

A.C. CONDITIONS OF TEST

Input Pulse Levels: +0.65 Volt to 2.2 Volt

Input Pulse Rise and Fall Times: 20nsec

Timing Measurement Reference Level: 1.5 Volt

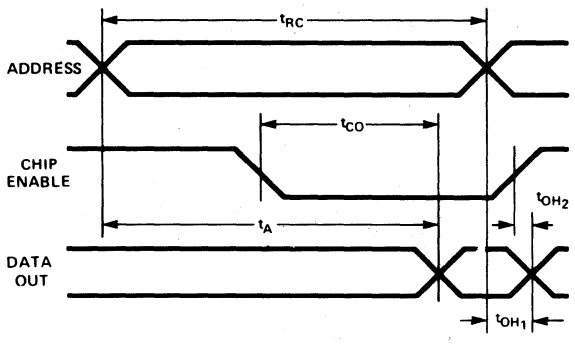
Output Load: 1 TTL Gate and C_L = 100 pF

CAPACITANCE T_A = 25°C, f = 1MHz

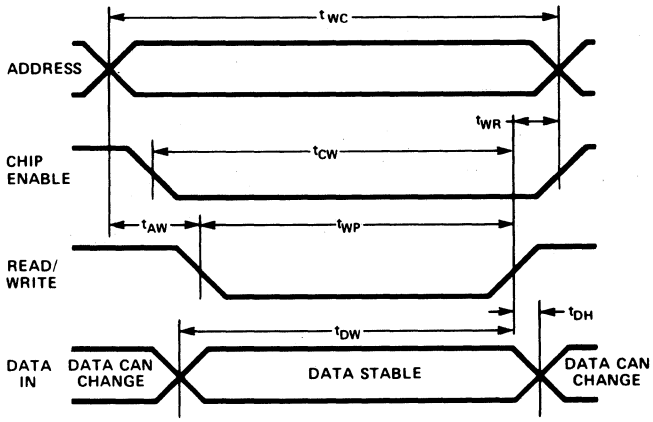
SYMBOL	TEST	LIMITS (pF)	
		TYP.	MAX.
C_{IN}	INPUT CAPACITANCE (ALL INPUT PINS) V_{IN} = 0V	3	5
C_{OUT}	OUTPUT CAPACITANCE V_{OUT} = 0V	7	10

WAVEFORMS

READ CYCLE

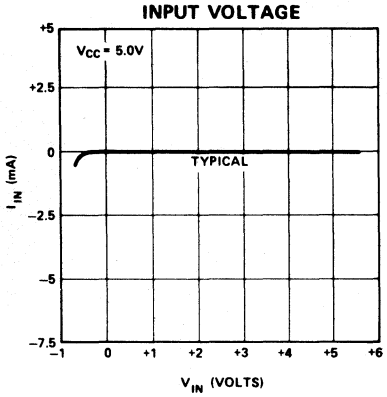


WRITE CYCLE

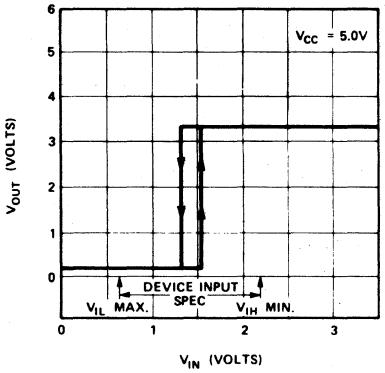


TYPICAL D.C. CHARACTERISTICS

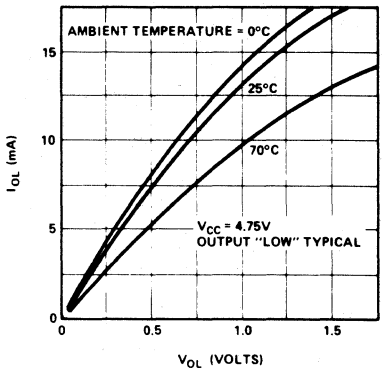
INPUT CURRENT VS. INPUT VOLTAGE



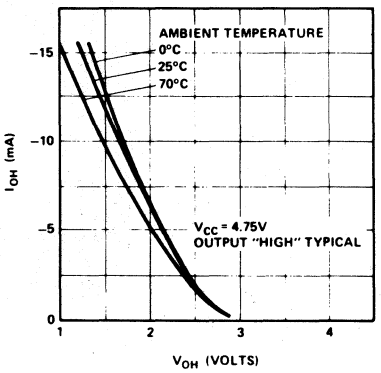
EFFECTIVE INPUT CHARACTERISTIC



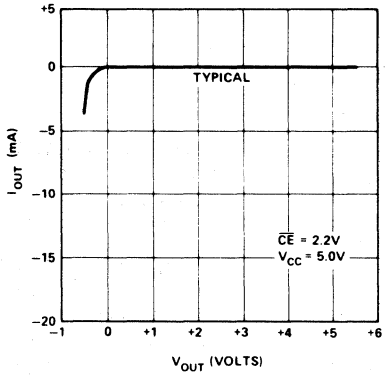
OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



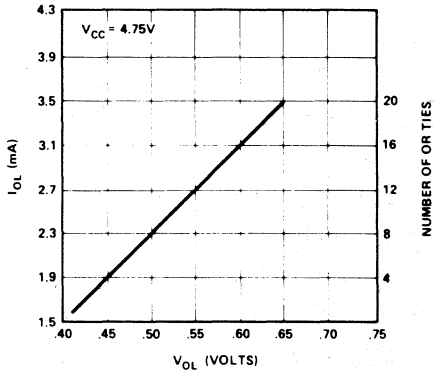
OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT CURRENT VS. OUTPUT VOLTAGE WITH CHIP DISABLED

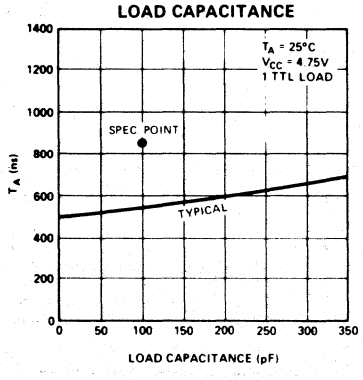


RELATIONSHIP BETWEEN OUTPUT SINK CURRENT, NUMBER OF OR-TIES, AND OUTPUT VOLTAGE

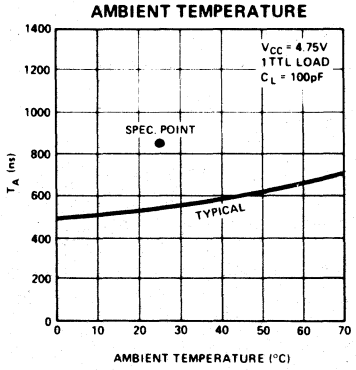


TYPICAL A.C. CHARACTERISTICS

ACCESS TIME VS. LOAD CAPACITANCE



ACCESS TIME VS. AMBIENT TEMPERATURE



Schottky Bipolar 8205

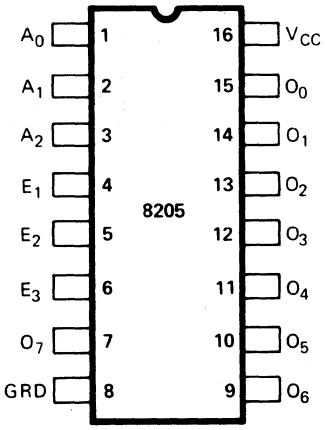
HIGH SPEED 1 OUT OF 8 BINARY DECODER

- **I/O Port or Memory Selector**
- **Simple Expansion — Enable Inputs**
- **High Speed Schottky Bipolar Technology — 18ns Max. Delay**
- **Directly Compatible with TTL Logic Circuits**
- **Low Input Load Current — .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**

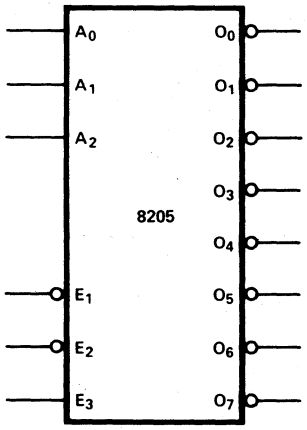
The 8205 decoder can be used for expansion of systems which utilize input ports, output ports, and memory components with active low chip select input. When the 8205 is enabled, one of its eight outputs goes "low", thus a single row of a memory system is selected. The 3 chip enable inputs on the 8205 allow easy system expansion. For very large systems, 8205 decoders can be cascaded such that each decoder can drive eight other decoders for arbitrary memory expansions.

The Intel 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 diffusion process.

PIN CONFIGURATION



LOGIC SYMBOL



PIN NAMES

A_0- A_2	ADDRESS INPUTS
$\overline{E_1}$- $\overline{E_3}$	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

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias:	Ceramic	−65°C to +125°C
	Plastic	−65°C to +75°C
Storage Temperature		−65°C to +160°C
All Output or Supply Voltages		−0.5 to +7 Volts
All Input Voltages		−1.0 to +5.5 Volts
Output Currents		125 mA

*COMMENT

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition 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.

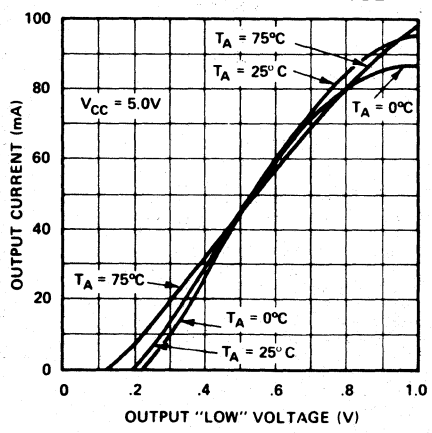
D.C. CHARACTERISTICS T_A = 0°C to +75°C, V_{CC} = 5.0V ±5%

8205

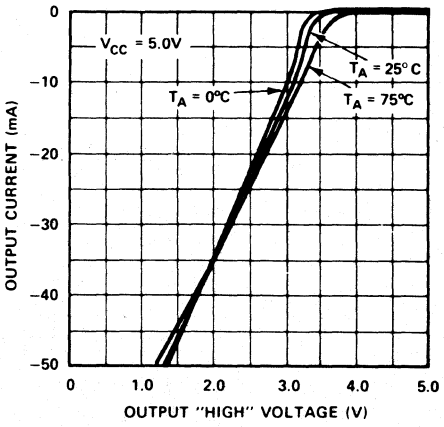
SYMBOL	PARAMETER	LIMIT MIN.	LIMIT MAX.	UNIT	TEST CONDITIONS
I_F	INPUT LOAD CURRENT		−0.25	mA	V_{CC} = 5.25V, V_F = 0.45V
I_R	INPUT LEAKAGE CURRENT		10	μA	V_{CC} = 5.25V, V_R = 5.25V
V_C	INPUT FORWARD CLAMP VOLTAGE		−1.0	V	V_{CC} = 4.75V, I_C = −5.0 mA
V_{OL}	OUTPUT "LOW" VOLTAGE		0.45	V	V_{CC} = 4.75V, I_{OL} = 10.0 mA
V_{OH}	OUTPUT HIGH VOLTAGE	2.4		V	V_{CC} = 4.75V, I_{OH} = −1.5 mA
V_{IL}	INPUT "LOW" VOLTAGE		0.85	V	V_{CC} = 5.0V
V_{IH}	INPUT "HIGH" VOLTAGE	2.0		V	V_{CC} = 5.0V
I_{SC}	OUTPUT HIGH SHORT CIRCUIT CURRENT	−40	−120	mA	V_{CC} = 5.0V, V_{OUT} = 0V
V_{OX}	OUTPUT "LOW" VOLTAGE @ HIGH CURRENT		0.8	V	V_{CC} = 5.0V, I_{OX} = 40 mA
I_{CC}	POWER SUPPLY CURRENT		70	mA	V_{CC} = 5.25V

TYPICAL CHARACTERISTICS

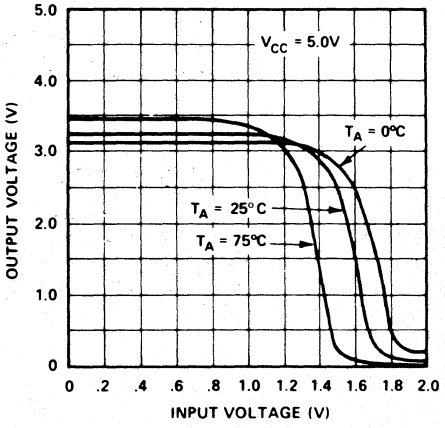
OUTPUT CURRENT VS. OUTPUT "LOW" VOLTAGE



OUTPUT CURRENT VS. OUTPUT "HIGH" VOLTAGE



DATA TRANSFER FUNCTION



8205 SWITCHING CHARACTERISTICS

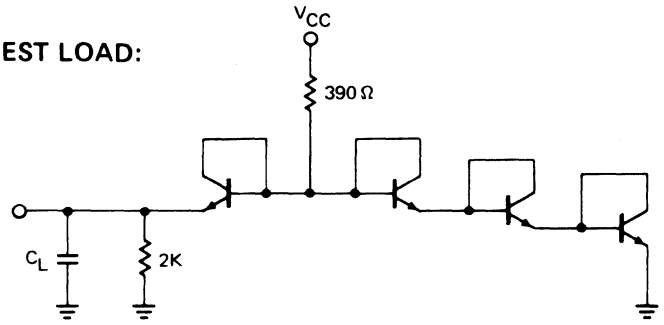
CONDITIONS OF TEST:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec between 1V and 2V

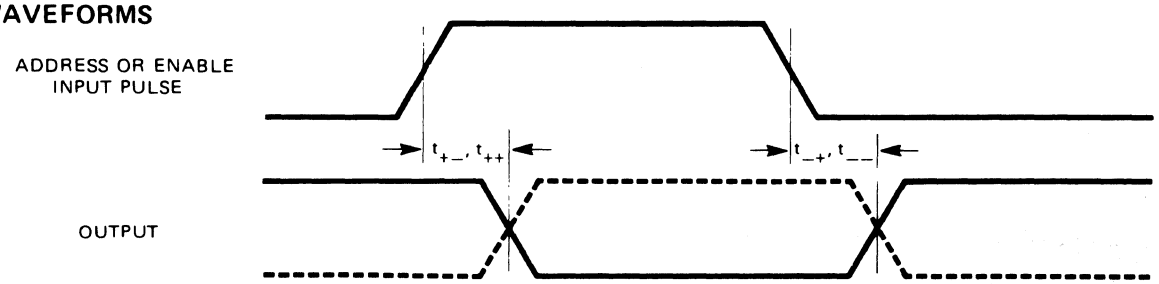
Measurements are made at 1.5V

TEST LOAD:



All Transistors 2N2369 or Equivalent. C_L = 30 pF

TEST WAVEFORMS



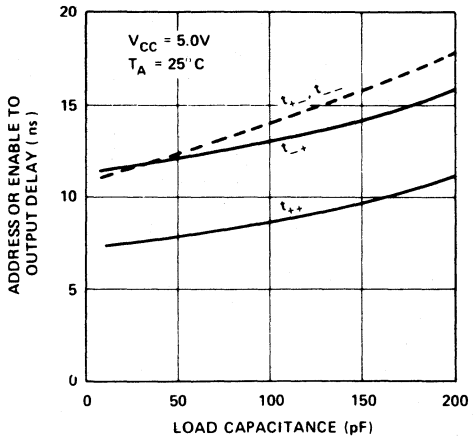
A.C. CHARACTERISTICS T_A = 0°C to +75°C, V_{CC} = 5.0V ±5% unless otherwise specified.

SYMBOL	PARAMETER		MAX. LIMIT	UNIT	TEST CONDITIONS
t_{++}	ADDRESS OR ENABLE TO OUTPUT DELAY		18	ns	
t_{-+}			18	ns	
t_{+-}			18	ns	
t_{--}			18	ns	
C_{IN} (1)	INPUT CAPACITANCE	P8205	4(typ.)	pF	f = 1 MHz, V_{CC} = 0V
		C8205	5(typ.)	pF	V_{BIAS} = 2.0V, T_A = 25°C

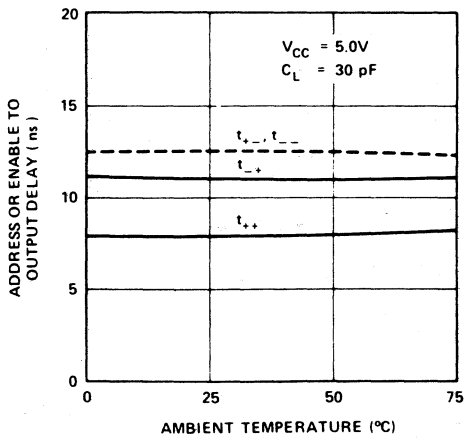
1. This parameter is periodically sampled and is not 100% tested.

TYPICAL CHARACTERISTICS

ADDRESS OR ENABLE TO OUTPUT DELAY VS. LOAD CAPACITANCE



ADDRESS OR ENABLE TO OUTPUT DELAY VS. AMBIENT TEMPERATURE



Silicon Gate MOS **8702A**

2048 BIT ERASABLE AND ELECTRICALLY REPROGRAMMABLE READ ONLY MEMORY

- **Access Time — 1.3 μsec Max.**
- **Fast Programming — 2 Minutes for All 2048 Bits**
- **Fully Decoded, 256 x 8 Organization**
- **Static MOS — No Clocks Required**
- **Inputs and Outputs TTL Compatible**
- **Three-State Output — OR-Tie Capability**
- **Simple Memory Expansion Chip Select Input Lead**

The 8702A is a 256 word by 8 bit electrically programmable ROM ideally suited for microcomputer system development where fast turn-around and pattern experimentation are important. The 8702A undergoes complete programming and functional testing on each bit position prior to shipment, thus insuring 100% programmability.

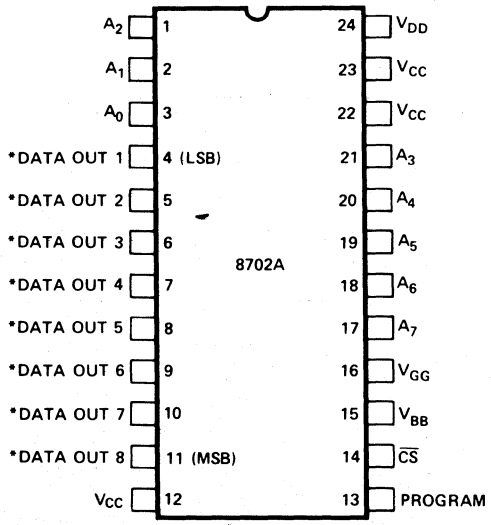
The 8702A is packaged in a 24 pin dual-in line package with a transparent quartz lid. The transparent quartz lid allows the user to expose the chip to ultraviolet light to erase the bit pattern. A new pattern can then be written into the device. This procedure can be repeated as many times as required.

The circuitry of the 8702A is entirely static; no clocks are required.

A pin-for-pin metal mask programmed ROM, the Intel 8302, is ideal for large volume production runs of systems initially using the 8702A.

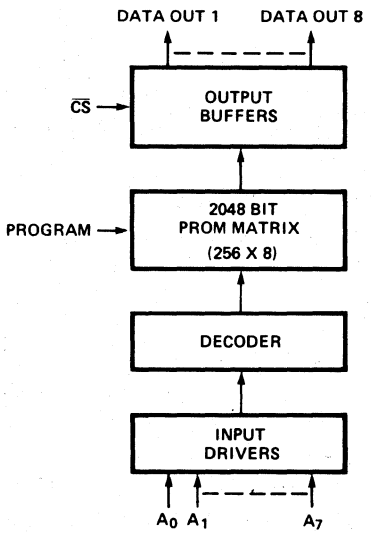
The 8702A is fabricated with silicon gate technology. This low threshold technology allows the design and production of higher performance MOS circuits and provides a higher functional density on a monolithic chip than conventional MOS technologies.

PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD DURING PROGRAMMING.

BLOCK DIAGRAM



PIN NAMES

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

PIN CONNECTIONS

The external lead connections to the 8702A differ, depending on whether the device is being programmed(1) or used in read mode. (See following table.)

MODE \ PIN	12 (V_{CC})	13 (Program)	14 ($\overline{CS}$)	15 (V_{BB})	16 (V_{GG})	22 (V_{CC})	23 (V_{CC})
Read	V_{CC}	V_{CC}	GND	V_{CC}	V_{GG}	V_{CC}	V_{CC}
Programming	GND	Program Pulse	GND	V_{BB}	Pulsed V_{GG} (V_{IL4P})	GND	GND

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias . . . . . 0°C to +70°C
Storage Temperature . . . . . −65°C to +125°C
Soldering Temperature of Leads (10 sec) . . . . . +300°C
Power Dissipation . . . . . 2 Watts
Read Operation: Input Voltages and Supply Voltages with respect to V_{CC} . . . . . +0.5V to −20V
Program Operation: Input Voltages and Supply Voltages with respect to V_{CC} . . . . . −48V

*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 at any other condition 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.

READ OPERATION

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V±5%, V_{DD} = −9V±5%, $V_{GG}^{(2)}$ = −9V±5%, unless otherwise noted.

SYMBOL	TEST	MIN.	TYP.(3)	MAX.	UNIT	CONDITIONS
I_{LI}	Address and Chip Select Input Load Current			10	μA	V_{IN} = 0.0V
I_{LO}	Output Leakage Current			10	μA	V_{OUT} = 0.0V, $\overline{CS}$ = V_{CC} −2
I_{DDO}	Power Supply Current		5	10	mA	V_{GG}=V_{CC}, $\overline{CS}$=V_{CC}−2, I_{OL} = 0.0mA, T_A = 25°C
I_{DD1}	Power Supply Current		35	50	mA	$\overline{CS}$=V_{CC} −2, I_{OL} =0.0mA, T_A = 25°C
I_{DD2}	Power Supply Current		32	46	mA	$\overline{CS}$=0.0, I_{OL} =0.0mA, T_A = 25°C
I_{DD3}	Power Supply Current		38.5	60	mA	$\overline{CS}$=V_{CC} −2, I_{OL} =0.0mA, T_A = 0°C
I_{CF1}	Output Clamp Current		8	14	mA	V_{OUT} = −1.0V, T_A = 0°C
I_{CF2}	Output Clamp Current			13	mA	V_{OUT} = −1.0V, T_A = 25°C
I_{GG}	Gate Supply Current			10	μA	
V_{IL1}	Input Low Voltage for TTL Interface	−1.0		0.65	V	
V_{IL2}	Input Low Voltage for MOS Interface	V_{DD}		V_{CC} −6	V	
V_{IH}	Address and Chip Select Input High Voltage	V_{CC} −2		V_{CC} +0.3	V	
I_{OL}	Output Sink Current	1.6	4		mA	V_{OUT} = 0.45V
V_{OL}	Output Low Voltage		−.7	0.45	V	I_{OL} = 1.6mA
V_{OH}	Output High Voltage	3.5			V	I_{OH} = −200 μA

Continuous Operation

Note 1: In the programming mode, the data inputs 1−8 are pins 4−11 respectively. $\overline{CS}$ = GND.
Note 2: V_{GG} may be clocked to reduce power dissipation. In this mode average I_{DD} increases in proportion to V_{GG} duty cycle. (See p. 5)
Note 3: Typical values are at nominal voltages and T_A = 25°C.

A.C. CHARACTERISTICS

T_A = 0° C to +70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5% unless otherwise noted

SYMBOL	TEST	MINIMUM	TYPICAL	MAXIMUM	UNIT
Freq.	Repetition Rate			1	MHz
t_{OH}	Previous read data valid			100	ns
t_{ACC}	Address to output delay			1.3	µs
t_{DVGG}	Clocked V_{GG} set up	1.0			µs
t_{CS}	Chip select delay			400	ns
t_{CO}	Output delay from $\overline{CS}$			900	ns
t_{OD}	Output deselect			400	ns
t_{OHC}	Data out hold in clocked V_{GG} mode (Note 1)			5	µs

Note 1. The output will remain valid for t_{OHC} as long as clocked V_{GG} is at V_{CC}. An address change may occur as soon as the output is sensed (clocked V_{GG} may still be at V_{CC}). Data becomes invalid for the old address when clocked V_{GG} is returned to V_{GG}.

CAPACITANCE* T_A = 25°C

SYMBOL	TEST	MINIMUM	TYPICAL	MAXIMUM	UNIT	CONDITIONS
C_{IN}	Input Capacitance		8	15	pF	$V_{IN} = V_{CC}$, $\overline{CS} = V_{CC}$, $V_{OUT} = V_{CC}$, $V_{GG} = V_{CC}$ — All unused pins are at A.C. ground
C_{OUT}	Output Capacitance		10	15	pF	
$C_{V_{GG}}$	V_{GG} Capacitance (Clocked V_{GG} Mode)			30	pF	

This parameter is periodically sampled and is not 100% tested.

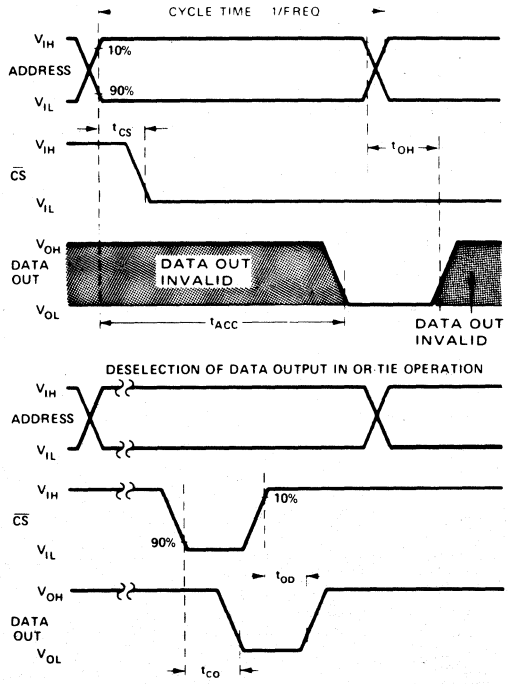
SWITCHING CHARACTERISTICS

Conditions of Test:

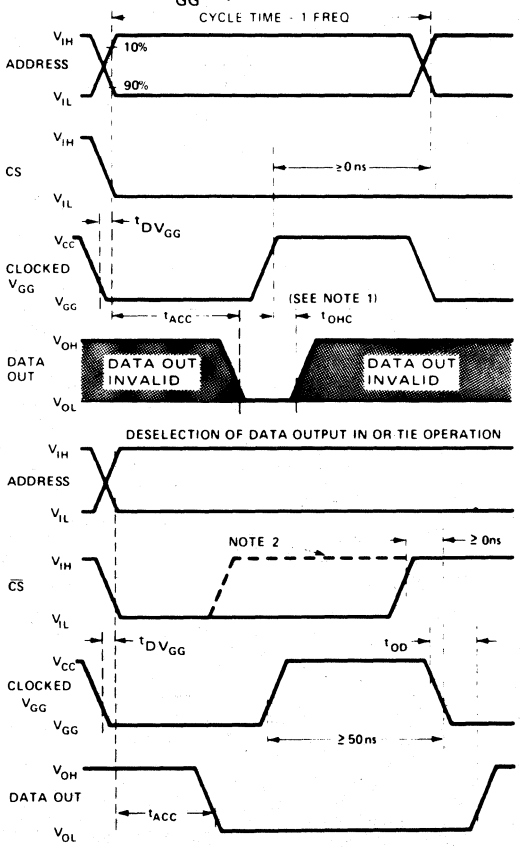
Input pulse amplitudes: 0 to 4V; t_R, t_F ≤50 ns

Output load is 1 TTL gate; measurements made at output of TTL gate (t_{PD} ≤15 ns)

A) Constant V_{GG} Operation



B) Clocked V_{GG} Operation

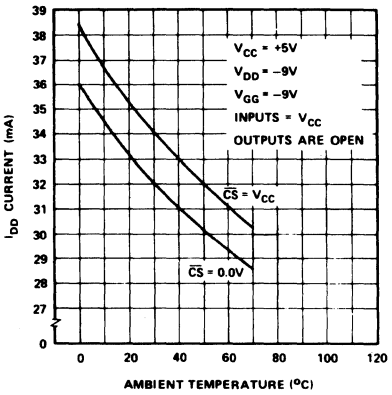


NOTE 1: The output will remain valid for t_{OHC} as long as clocked V_{GG} is at V_{CC}. An address change may occur as soon as the output is sensed (clocked V_{GG} may still be at V_{CC}). Data becomes invalid for the old address when clocked V_{GG} is returned to V_{GG}.

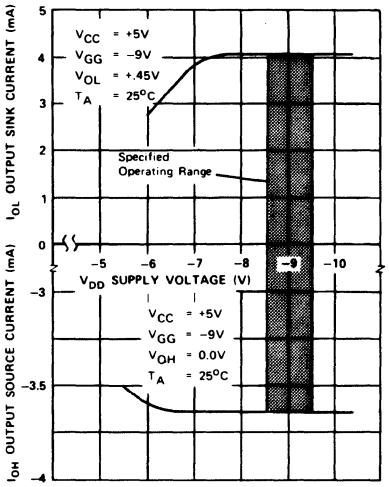
NOTE 2: If $\overline{CS}$ makes a transition from V_{IL} to V_{IH} while clocked V_{GG} is at V_{GG}, then deselection of output occurs at t_{OD} as shown in static operation with constant V_{GG}.

TYPICAL CHARACTERISTICS

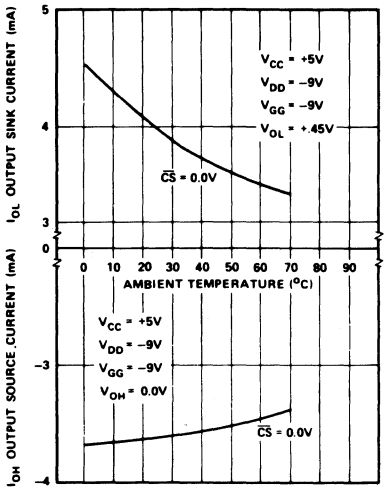
I_{DD} CURRENT VS. TEMPERATURE



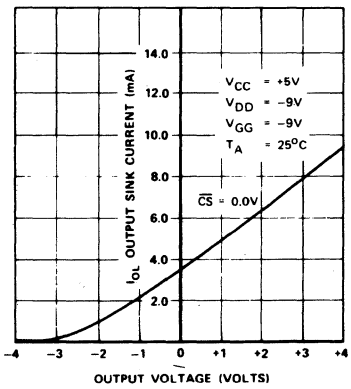
OUTPUT CURRENT VS. V_{DD} SUPPLY VOLTAGE



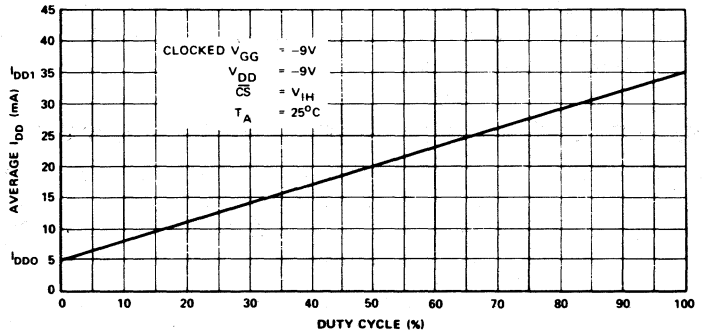
OUTPUT CURRENT VS. TEMPERATURE



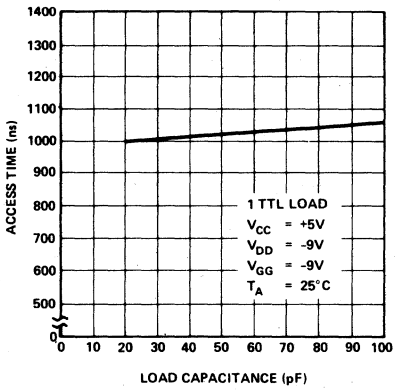
OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



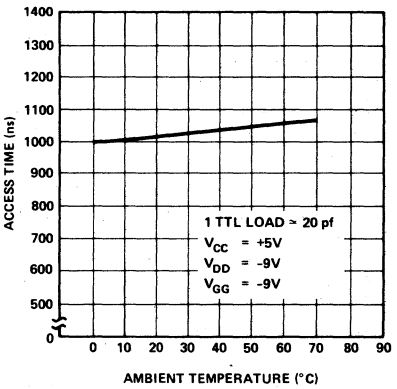
AVERAGE CURRENT VS. DUTY CYCLE FOR CLOCKED V_{GG}



ACCESS TIME VS. LOAD CAPACITANCE



ACCESS TIME VS. TEMPERATURE



PROGRAMMING OPERATION

D.C. AND OPERATING CHARACTERISTICS FOR PROGRAMMING OPERATION

$T_A = 25°C$, $V_{CC} = 0V$, $V_{BB} = +12V \pm 10\%$, $\overline{CS} = 0V$ unless otherwise noted

SYMBOL	TEST	MIN.	TYP.	MAX.	UNIT	CONDITIONS
I_{LI1P}	Address and Data Input Load Current			10	mA	$V_{IN} = -48V$
I_{LI2P}	Program and V_{GG} Load Current			10	mA	$V_{IN} = -48V$
I_{BB}	V_{BB} Supply Load Current		.05		mA	
I_{DDP}(1)	Peak I_{DD} Supply Load Current		200		mA	$V_{DD} = V_{prog} = -48V$ $V_{GG} = -35V$
V_{IHP}	Input High Voltage			0.3	V	
V_{IL1P}	Pulsed Data Input Low Voltage	−46		−48	V	
V_{IL2P}	Address Input Low Voltage	−40		−48	V	
V_{IL3P}	Pulsed Input Low V_{DD} and Program Voltage	−46		−48	V	
V_{IL4P}	Pulsed Input Low V_{GG} Voltage	−35		−40	V	

Note 1: I_{DDP} flows only during V_{DD}, V_{GG} on time. I_{DDP} should not be allowed to exceed 300mA for greater than 100μsec. Average power supply current I_{DDP} is typically 40mA at 20% duty cycle.

A.C. CHARACTERISTICS FOR PROGRAMMING OPERATION

$T_{AMBIENT} = 25°C$, $V_{CC} = 0V$, $V_{BB} = +12V \pm 10\%$, $\overline{CS} = 0V$ unless otherwise noted

SYMBOL	TEST	MIN.	TYP.	MAX.	UNIT	CONDITIONS
	Duty Cycle (V_{DD}, V_{GG})			20	%	
$t_{\phi PW}$	Program Pulse Width			3	ms	$V_{GG} = -35V$, $V_{DD} = V_{prog} = -48V$
t_{DW}	Data Set Up Time	25			μs	
t_{DH}	Data Hold Time	10			μs	
t_{VW}	V_{DD}, V_{GG} Set Up	100			μs	
t_{VD}	V_{DD}, V_{GG} Hold	10		100	μs	
t_{ACW} (2)	Address Complement Set Up	25			μs	
t_{ACH} (2)	Address Complement Hold	25			μs	
t_{ATW}	Address True Set Up	10			μs	
t_{ATH}	Address True Hold	10			μs	

Note 2. All 8 address bits must be in the complement state when pulsed V_{DD} and V_{GG} move to their negative levels. The addresses (0 through 255) must be programmed as shown in the timing diagram for a minimum of 32 times.

SWITCHING CHARACTERISTICS FOR PROGRAMMING OPERATION

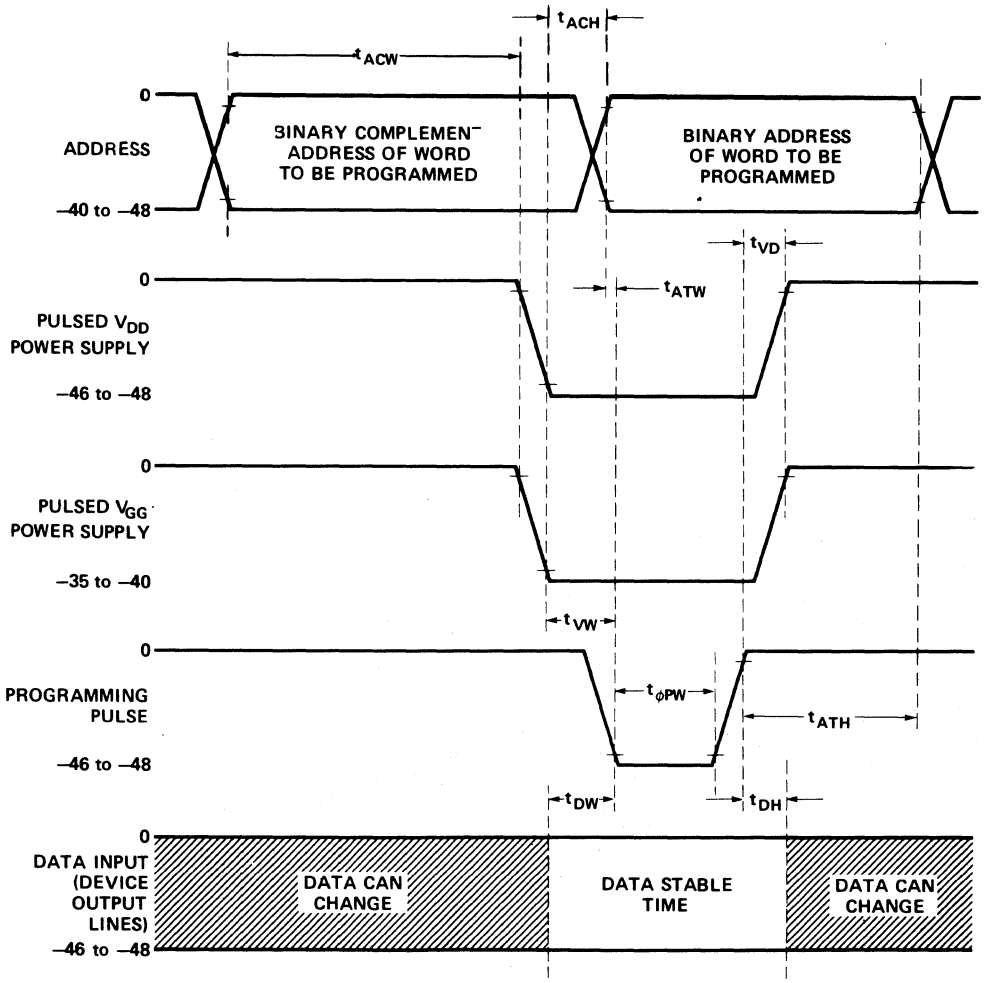
PROGRAM OPERATION

Conditions of Test:

Input pulse rise and fall times $\leq 1\mu$sec

$\overline{CS}$ = 0V

PROGRAM WAVEFORMS



PROGRAMMING OPERATION OF THE 8702A

When the Data Input for the Program Mode is:	Then the Data Output during the Read Mode is:
V_{ILIP} = ~–48V pulsed	Logic 1 = V_{OH} = 'P' on tape
V_{IHP} = ~ 0V	Logic 0 = V_{OL} = 'N' on tape

WORD	ADDRESS							
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|
255	1	1	1	1	1	1	1	1

Address Logic Level During Read Mode: Logic 0 = V_{IL} (~.3V) Logic 1 = V_{IH} (~3V)

Address Logic Level During Program Mode: Logic 0 = V_{IL2P} (~–40V) Logic 1 = V_{IHP} (~0V)

PROGRAMMING INSTRUCTIONS FOR THE 8702A

I. Operation of the 8702A in Program Mode

Initially, all 2048 bits of the ROM are in the "0" state (output low). Information is introduced by selectively programming "1"s (output high) in the proper bit locations.

Word address selection is done by the same decoding circuitry used in the READ mode (see table on page 6 for logic levels). All 8 address bits must be in the binary complement state when pulsed V_{DD} and V_{GG} move to their negative levels. The addresses must be held in their binary complement state for a minimum of 25 μsec after V_{DD} and V_{GG} have moved to their negative levels. The addresses must then make the transition to their true state a minimum of 10 μsec before the program pulse is applied. The addresses should be programmed in the sequence 0 through 255 for a minimum of 32 times. The eight output terminals are used as data inputs to determine the information pattern in the eight bits of each word. A low data input level (-48V) will program a "1" and a high data input level (ground) will leave a "0" (see table on page 6). All eight bits of one word are programmed simultaneously by setting the desired bit information patterns on the data input terminals.

During the programming, V_{GG}, V_{DD} and the Program Pulse are pulsed signals.

II. Programming of the 8702A Using Intel Microcomputers

Intel provides low cost program development systems which may be used to program its electrically programmable ROMs. Note that the programming specifications that apply to the 8702A are identical to those for Intel's 1702A.

A. Intellec 8

The Intellec series of program development systems, the Intellec 8/Mod 8 and Intellec 8/Mod 80, are used as program development tools for the 8008 and 8080 microprocessors respectively. As such, they are equipped with a PROM programmer card and may be used to program Intel's electrically programmable and ultraviolet erasable ROMs.

An ASR-33 teletype terminal is used as the input device. Through use of the Intellec software system monitor, programs to be loaded into PROM may be typed in directly or loaded through the paper tape reader. The system monitor allows the program to be reviewed or altered at will prior to actually programming the PROM. For more complete information on these program development systems, refer to the Intel Microcomputer Catalog or the Intellec Specifications.

B. Users of the SIM8 microcomputer programming systems may also program the 8702A using the MP7-03 programmer card and the appropriate control ROMs:
SIM8 system—Control ROMs A0860, A0861 and A0863.

III. 8702A Erasing Procedure

The 8702A may be erased by exposure to high intensity short-wave ultraviolet light at a wavelength of 2537A. The recommended integrated dose (i.e., UV intensity x exposure time) is 6W-sec/cm^2. Examples of ultraviolet sources which can erase the 8702A in 10 to 20 minutes are the Model UVS-54 and Model S-52 short-wave ultraviolet lamps manufactured by Ultra-Violet Products, Inc. (5114 Walnut Grove Avenue, San Gabriel, California). The lamps should be used without short-wave filters, and the 8702A to be erased should be placed about one inch away from the lamp tubes.

APPENDIX C ASCII TABLE

The INTELLEC 8 uses a seven-bit ASCII code, which is the normal 8 bit ASCII code with the parity (high order) bit always reset.

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
NULL	00
SOM	01
EOA	02
EOM	03
EOT	04
WRU	05
RU	06
BELL	07
FE	08
H. Tab	09
Line Feed	0A
V. Tab	0B
Form	0C
Return	0D
SO	0E
SI	0F
DCO	10
X-On	11
Tape Aux. On	12
X-Off	13
Tape Aux. Off	14
Error	15
Sync	16
LEM	17
S0	18
S1	19
S2	1A
S3	1B
S4	1C
S5	1D
S6	1E
S7	1F

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
ACK	7C
Alt. Mode	7D
Rubout	7F
!	21
"	22
#	23
$	24
%	25
&	26
'	27
(	28
)	29
*	2A
+	2B
,	2C
-	2D
.	2E
/	2F
:	3A
;	3B
<	3C
=	3D
>	3E
?	3F
[	5B
/	5C
]	5D
↑	5E
←	5F
@	40
blank	20
0	30

GRAPHIC OR CONTROL	ASCII (HEXADECIMAL)
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39
A	41
B	42
C	43
D	44
E	45
F	46
G	47
H	48
I	49
J	4A
K	4B
L	4C
M	4D
N	4E
O	4F
P	50
Q	51
R	52
S	53
T	54
U	55
V	56
W	57
X	58
Y	59
Z	5A

APPENDIX D BINARY-DECIMAL-HEXADECIMAL CONVERSION TABLES

HEXADECIMAL ARITHMETIC

ADDITION TABLE

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

MULTIPLICATION TABLE

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	48	5A	69	78	87	96	A5	B4	C3	D2	E1

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 676 950 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125

TABLE OF POWERS OF SIXTEEN$_{10}$

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-12}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

TABLE OF POWERS OF 10_{16}

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	0.28F5 C28F 5C28 F5C3 x 16^{-1}
3E8	3	0.4189 374B C6A7 EF9E x 16^{-2}
2710	4	0.68DB 8BAC 710C B296 x 16^{-3}
1 86A0	5	0.A7C5 AC47 1B47 8423 x 16^{-4}
F 4240	6	0.10C6 F7A0 B5ED 8D37 x 16^{-4}
98 9680	7	0.1AD7 F29A BCAF 4858 x 16^{-5}
5F5 E100	8	0.2AF3 1DC4 6118 73BF x 16^{-6}
3B9A CA00	9	0.44B8 2FA0 9B5A 52CC x 16^{-7}
2 540B E400	10	0.6DF3 7F67 SEF6 EADF x 16^{-8}
17 4876 E800	11	0.AFEB FF0B CB24 AAFF x 16^{-9}
E8 D4A5 1000	12	0.1197 9981 2DEA 1119 x 16^{-9}
918 4E72 A000	13	0.1C25 C268 4976 81C2 x 16^{-10}
5AF3 107A 4000	14	0.2D09 370D 4257 3604 x 16^{-11}
3 8D7E A4C6 8000	15	0.480E BE7B 9D58 566D x 16^{-12}
23 8652 6FC1 0000	16	0.734A CA5F 6226 F0AE x 16^{-13}
163 4578 5D8A 0000	17	0.B877 AA32 36A4 B449 x 16^{-14}
DE0 B6B3 A764 0000	18	0.1272 5DD1 D243 ABA1 x 16^{-14}
8AC7 2304 89E8 0000	19	0.1D83 C94F B6D2 AC35 x 16^{-15}

HEXADECIMAL-DECIMAL INTEGER CONVERSION

The table below provides for direct conversions between hexadecimal integers in the range 0-FFF and decimal integers in the range 0-4095. For conversion of larger integers, the table values may be added to the following figures:

Hexadecimal	Decimal	Hexadecimal	Decimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0331	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0301	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0212	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
700	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
710	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
730	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
740	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
750	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
760	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
780	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
790	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	4761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	3851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2866	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
B80	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
CC0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

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

© 1975 Printed in U.S.A. MCS-307a–0275/1K