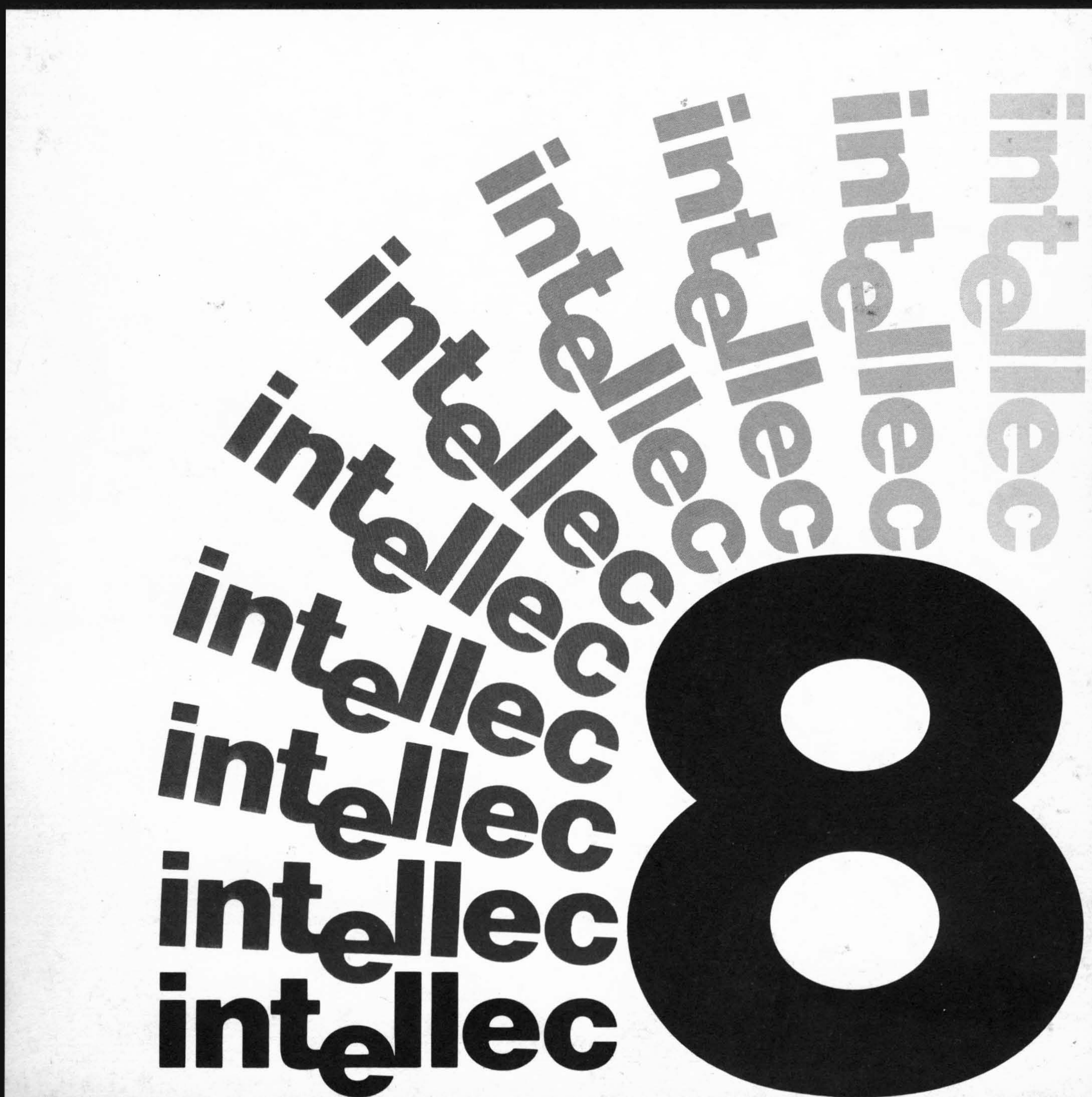
intel

intellec 8

Reference Manual

Rev. 1



intel

intellec 8

Reference Manual

Rev. 1

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

General Description
Specifications
The Scope Of This Manual

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

GENERAL DESCRIPTION

The INTELLEC 8/MOD 8 is a low-cost computer system, designed to simplify the development of microcomputer systems which employ Intel 8008 CPU chip processors.

The INTELLEC 8/MOD 8 uses the 8008-1 as its central processing unit. The 8008-1 is a selected version of the 8008, chosen for its high speed characteristics. The 8008-1 has a basic cycle time of 12.5 microseconds, whereas that of the 8008 is 20 microseconds. The system contains a control console and provides read-write program memory as a substitute for read-only memory. Thus the 8008-1 chip can be accessed via the control console, and programs debugged before being committed to 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 8 (part #imm8-80A) 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, 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 and a PROM Programmer.

The Bare Bones 8 is an INTELLEC 8/MOD 8 without the power supply, display and control console, or cabinet, and is designed for rack-mounting.

Either the INTELLEC 8/MOD 8 or the Bare Bones 8 can be expanded up to a total of 16K bytes of memory, eight input ports, and twenty-four output ports.

The standard software for the INTELLEC 8 includes a resident System Monitor, a Text Editor, and an Assembler. In addition to these INTELLEC 8 resident programs, there are available three development programs, which are designed for operation on **large-scale, general-purpose computers.** These are a macro cross-assembler (MAC/8), a microcomputer simulator (INTERP/8), and a PL/M compiler. PL/M is a high-level language that can shorten program development time significantly.

SPECIFICATIONS

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

1) The imm8-82 Central Processor Module, which operates as the Central Processor for the INTELLEC 8/MOD 8. In this capacity, it performs the following functions.
   a) It controls the execution of program instructions, sending the appropriate control signals to the other modules which make up the INTELLEC 8/MOD 8.
   b) It performs all of the necessary airthmetic, 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. In the imm8-80A Intellec, two 6-28 modules are included for 8K of memory capacity.
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 imm8-60 Input/Output Module, which provides four eight-bit input ports and four eight-bit output ports for system Input/Output operations. *In addition, 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 two of these cards may be used in a system,

giving a total of eight input ports and eight output ports.

5) The imm8-62 Output Module, which provides eight latching output ports for system Output operations. Up to two of these cards may be used in a system, giving a total capability of twenty-four output ports (including the eight output ports provided by the two possible imm8-60 Input/Output Modules).

6) The imm6-76 PROM Programmer Card, which gives the INTELLEC 8/MOD 8 system the capability of programming Intel 1602A or 1702A 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 8 operation. It also provides displays of system status and information.

8) The chassis and power supplies, which serve to hold all of the modules together.

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

THE SCOPE OF THIS MANUAL

This manual provides an understanding of the design concepts and capabilities of the INTELLEC 8/MOD 8 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 8 operating procedures, including software operation, see the INTELLEC 8/MOD 8 Operator's Manual. For a detailed examination of programming at an elementary level, suitable for an engineer with no previous programming experience, see the INTELLEC 8/MOD 8 Programmer's Manual.

INTELLEC 8/MOD 8 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	Forty-eight instructions, including memory-register, register-memory, register-to-register, single register, immediate, and memory arithmetic and logic instructions, conditional and unconditional jump, subroutine handling, input/output, and machine halt instructions.
Arithmetic	8-bit parallel, binary, fixed point, twos complement.
Memory	8,192 eight-bit words, Read/Write; 1,280 eight-bit words, Read-only. (Combination of Read/Write and Read-only memories is expandable to 16,384 words).
Addressing	Direct—up to 16K eight-bit words
Cycle Time	12.5 microseconds
Environment	0°to +55°C.
Power Requirements	5V @ 12 A (max); 6 A (typ) -9V @ 1.8 A (max); 0.5 A (typ) ±12V @ 0.06 A (max); 0.016 A (typ) (More power may be required for expanded INTELLEC 8 systems).
AC Requirement	60 Hz; 115 VAC, 200 Watts
Size	INTELLEC 8: 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 a.

CHAPTER 1 INTELLEC 8/MOD 8 SYSTEM OVERVIEW

Functional Description of Modules
Front Panel Console Operations
Memory Reference Operations
Input/Output Operations
Interrupt Operations
PROM Programming Operations

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

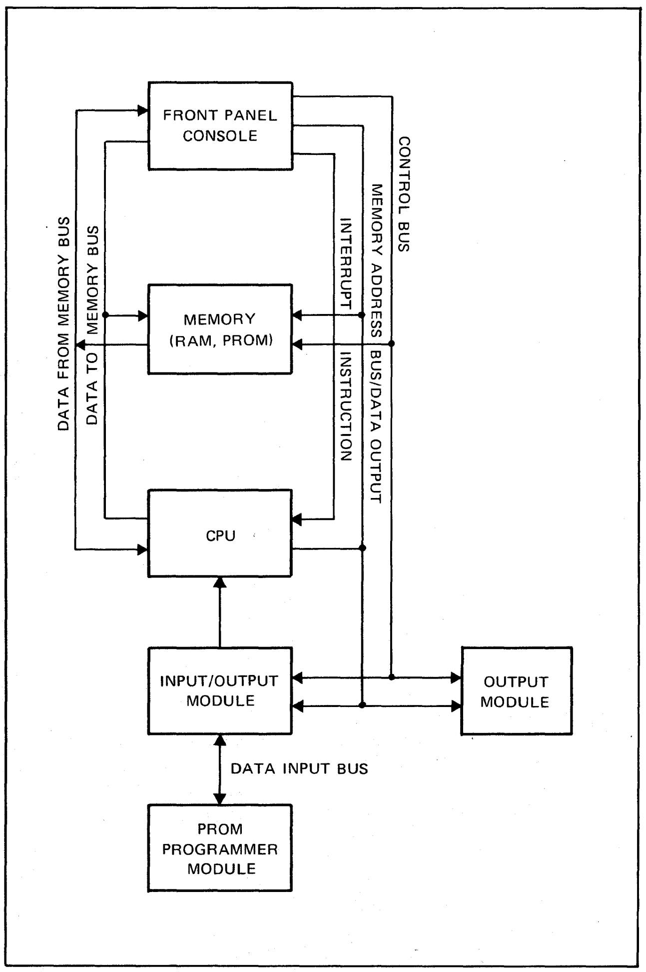


Figure 1-1. A Simplified INTELLEC 8/MOD 8 Block Diagram

FUNCTIONAL DESCRIPTION OF MODULES

Figure 1—1 illustrates the seven functional modules of the INTELLEC 8/MOD 8 system, and shows interconnecting busses. The seven 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. The INTELLEC 8/MOD 8 system includes two of these modules for a capacity of 8K bytes.

3) Input/Output module. Physically there can be one or two Input/Output modules in an INTELLEC 8/MOD 8 system. Each Input/Output module provides four 8-bit input and four 8-bit output ports. communications facility which the INTELLEC 8 uses for teletype interface.

4) Output module. Physically there can be one or two Output modules in an INTELLEC 8/MOD 8 system. Each Output module provides eight individually addressable 8-bit output ports.

5) A Front Panel Display and Control Console, which is most accurately visualized as a peripheral device, placed in parallel with the CPU. The Control Console provides means for manually monitoring and controlling INTELLEC 8/MOD 8 operations.

6) PROM Programmer Module used for programming Intel 1602A or 1702A PROMs via a socket on the front panel.

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

The Memory Address and Data Output Bus carries memory addresses from the console or the CPU to memory, and carries output data from the console or CPU to output ports, and thence to external peripheral devices (e.g., a teletype printer).

The Data to Memory Bus carries data from the console or CPU to the memory.

The Data from Memory Bus carries data from memory to the console and the CPU.

The Data Input Bus carries data from input ports to the CPU but not the console.

The Interrupt Instruction Bus allows the console to transmit a program interrupt to the CPU.

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.

The busses may be visualized as having three way switches that allow information to be routed to/from the CPU or 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 8 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, as described in the section on Memory Reference Operations later in this chapter.

Communications between the INTELLEC 8/MOD 8 and an operator occur via the Front Panel Console, as described in the next section.

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 bus link between the console and memory, data can be read from memory to console, and written from console directly 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 Data from Memory Bus where is displayed on the console.

There is no direct link between CPU registers and the console. In order to examine register contents, (a common program debugging operation), it is necessary to execute an instruction that causes the register contents to be placed on an external bus. Commonly, to examine the contents of a CPU register, a memory reference instruction is executed (see following section).

MEMORY REFERENCE OPERATIONS

This section describes memory reference operations as performed by the INTELLEC 8/MOD 8 system, and is divided into two subsections. The first describes memory input or read operations, and the second describes 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 Data from Memory 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 are to be stored in memory to the memory modules on the Data to Memory 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 Data Buses, 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 8 system, and is divided into three subsections. The first describes Input operations, the second describes Output operations, and the third describes Teletype operations.

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 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.
2) The CPU sends the data which are to be output to the Input/Output and Output modules on the Output Data bus.
3) The Input/Output or Output module 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 in the Input/Output module. **Teletype data enter the Input/Output module, and utilize Input ports 0 and 1 and Output ports 8 and 9.**

Chapter 3 explains how to install the Teletype ASR33.

INTERRUPT OPERATIONS

An Interrupt operation is performed when an external device which requires servicing (e.g. to transmit data via the CPU to memory) 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 to 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 completes the execution of the current instruction and acknowledges the Interrupt signal.
2) The external device sends the 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 the Interrupt switch is depressed, the Front Panel generates an Interrupt signal, and sends the manually entered Interrupt Instruction to the CPU.

In the basic system, only the Control Console initiates interrupts. An interrupt is used to start the processor. The interrupt operation may be extended, however, to the user's peripheral devices, in order to simplify system programming and to increase system throughput.

PROM PROGRAMMING OPERATIONS

The INTELLEC 8/MOD 8 has been designed to offer an easy means of programming Intel 1602A and 1702A 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 to 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 imm8-82 CENTRAL PROCESSOR MODULE

The Function Of A CPU
Functional Organization Of The Central Processor Module
8008-1 Eight-Bit Parallel Central Processor Unit
Peripheral Logic
Utilization

The imm8-82 Central Processor Module is designed to serve as the central processing unit of the INTELLEC 8 /MOD 8 Microcomputer Development System. However, the module's general-purpose architecture permits its use in other eight-bit systems. Inputs and outputs are TTL compatible.

The basic capabilities of the module are obtained through the use of Intel's 8008-1 monolithic CPU. The chip processor provides 48 command instructions, the ability to access over 16K memory bytes directly, 6 working index registers, a seven-level subroutine stack, and interrupt handling capability. The CPU's instruction set permits I/O and register-to-register transfer, arithmetic and logical operations. Four internal status bits permit conditional jumps based on carry (overflow-underflow), sign, zero, and parity (even).

The Central Processor Module contains a crystal controlled clock oscillator which provides a stable timing reference for all circuitry on the board. The use of a selected 8008 (the 8008-1) and an 800 kHz clock permits a basic processor cycle of 12.5 microseconds.

Memory interface and control logic are included on the module. The imm8-82 contains a latched fourteen-bit address bus, an eight-bit input bus for data from memory, and an eight-bit output bus for data to memory. The module generates signals which identify a memory read, a memory write, or an instruction fetch cycle. These are available for the control of external circuitry. A wait request line permits interfacing the processor module with slow memories. If minimum access time exceeds one microsecond, the memory controller can request a temporary pause in the processing cycle, causing the processor to wait for the memory's response to read or a write command.

I/O interface and control are also built into the Central Processor Module. Five digits on the address bus (A_9-A_{13}) are used during I/O operations, to specify one of 32 addressable peripherals. The lower eight addresses (0–7) are reserved for input devices, while the remaining twenty-four (8–31) are used for output. An eight-line data bus for peripheral inputs is included on the module. The output devices share an eight-line output bus with the external memory. Signals generated on the module identify and synchronize I/O operations. These are available for the control of external circuitry.

The imm8-82 is able to process external interrupts. It is equipped with an INTERRUPT request line and with an eight-bit interrupt port. An external device may request service by placing an appropriate instruction code on the interrupt port's lines and activating the INTERRUPT line.

The Central Processor Module is also equipped with a hold request line, which enables external devices to access memory directly. By issuing a wait request, and following the acknowledged wait with a hold, the memory controller can cause the processor to suspend its operation and relinquish control of the main data bus. This allows an external device to command the bus and to effect memory transfers directly.

The imm8-82 is largely self-contained. It requires DC power of:

+5 ± 5% VDC at 2.2 A (max)
-9 ± 5% VDC at 60 mA (max)

All circuitry is mounted on a 6.18" x 8.00" printed circuit board. Signal and power connections enter the module by means of a dual 50-pin double-sided PC edge convertor (0.125" centers). No special installation is necessary.

The following sections furnish a complete description of the imm8-82 Central Processor Module. The first describes a processing system at a fairly elementary level. This material is intended as background 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 might skip this section. The second describes the functional organization of the processor module. In the third we give some detailed information on the 8008-1 CPU. And in the fourth we show how the peripheral logic supports the functions that the 8008-1 must perform. Finally, in the fifth, we give reference information which will be of value to those who are planning to use the module outside the INTELLEC 8/MOD 8 system.

THE FUNCTION OF A CPU (The reader already familiar with basic computer concepts may skip this section.)

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, should you later become confused and decide to refer to it.

The Computer System

The INTELLEC 8/MOD 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 specified as having stand-along capability, meaning that they are generally capable of performing their functions in any system similar to the INTELLEC 8/MOD 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 processing 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/MOD 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 the moment, we shall have to be content with an intuitive understanding of what is meant by the term. Assume for the moment 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 sources, at high rates of speed and in large volumes.

Central processing units operate so rapidly that their responses often seem instantaneous 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 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 write data into memory, and retrieve it at some later point in the program. 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, and so on. An orderly sequence of events like this requires timing, and the CPU therefore contains a clock oscillator which furnishes the reference for all processor actions. The combined fetch and execution of a single instruction is referred to as a *machine cycle*. The portion of a cycle identified with a clearly defined activity is called a *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 a 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 a word of 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 location of memory is a *jump* instruction to another location of memory.

A jump instruction contains the address of the instruction which follows it. Since this 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," the processor replaces the contents of its program counter with the addresses embodied in the jump instruction. 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:

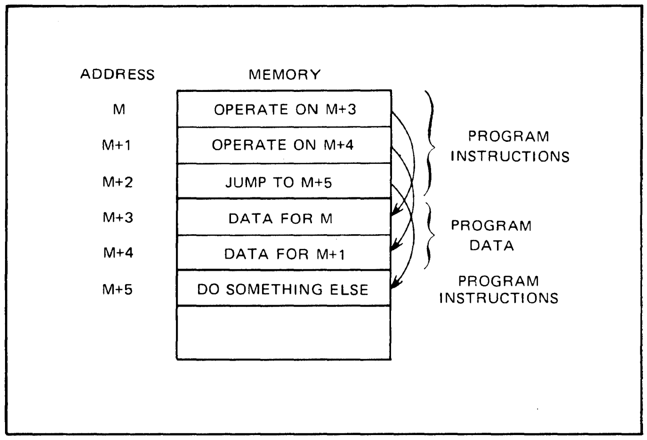


Figure 2-1: Program Jump

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 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 which calculate the square, the sine, or the logarithm of a program variable are good examples of the functions often written as subroutines. Other examples might be programs designed for inputting or outputting data to a particular peripheral device.

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 identity 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 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 stored 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 the 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 virtually unlimited subroutine nesting.

INSTRUCTION REGISTER AND DECODER

Every computer has a *word length* that is characteristic of that machine. An eight-bit parallel processor generally finds it 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 instruction 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 execution of the instructions.

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

MULTIPLE WORD INSTRUCTION

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 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 is 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 minimum 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 request signal 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 machine 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 super adding machine with its keys commanded automatically by the control signals developed in the instruction decoder. This is essentially how the first stored-program digital computer was conceived.

The ALU naturally bears little resemblance to a desk-top adder. The major difference is that the ALU calculates by creating an electrical analogy, rather than by mechanical analogy. Another important difference is that the ALU uses binary techniques—rather than decimal methods—for representing and manipulating numbers.

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

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 register 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 flags. Thus, for example, the program may be designed to jump to a special routine, if the carry bit is set following an addition instruction. The presence of a carry generally indicates an overflow in the accumulator, and sometimes calls for special processing actions.

We have touched very briefly on some of the features of an ALU, in an attempt to explain its function. 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 directly effect 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 INTELLEC 8/MOD 8 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. 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 an 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 the job. No 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.

The interrupt feature in this simple example permits us to increase processing efficiency. 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 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 then 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, however, 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 throughout will be increased by having the device accomplish the transfer directly. The processor must temporarily 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.

FUNCTIONAL ORGANIZATION OF THE CENTRAL PROCESSOR MODULE

The imm8-82 contains the 8008-1 CPU and the logic that supports the chip. In addition to the processor chip, the module contains the following logical blocks:

a) timing generator
b) cycle decoder
c) bus switching logic
d) address latches
e) read/write control
f) wait logic
g) interrupt logic
h) hold logic
i) status latches

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

The 8008-1 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 crystal controlled clock oscillator, a state decoder, logic on the CPU chip itself, and auxiliary timing logic.

The oscillator section generates two non-overlapping 800 kHz clock phases, which drive the processor chip as well as other timing circuitry on the board. Logic contained in the CPU chip derives a symmetrical 400 kHz SYNC signal from the ϕ_2 clock, and this too is made available to the auxiliary timing logic.

The state decoder receives a three-line signal from the processor chip (S_0-S_2), indicating the processor's internal phase. The state decoder produces the following logically exclusive outputs:

$\overline{T1}$, $\overline{T11}$, $\overline{T2}$, $\overline{T3}$, $\overline{T4}$, $\overline{T5}$, $\overline{WAIT}$, and $\overline{STOPPED}$

The auxiliary timing logic receives ϕ_1, ϕ_2, and SYNC. It also receives $\overline{T2}$ and $\overline{T3}$ signals from the state decoder. The auxiliary timing logic uses these inputs to generate:

$\overline{\phi_{12}}$, SYNCA, and T3A

The control signals produces by the state decoder and the timing logic then synchronize and govern all the other internal operations of the Central Processor Module.

The cycle decoder receives the two sub-cycle identification bits that the processor chip boradcasts during the T2 interval. Sub-cycle information is an internal function of the 8008-1, used to indicate which portion of a machine cycle is in progress. There are four possible sub-cycles: instruction fetch (PCI), memory read (PCR), memory write (PCW), or input/output (PCC). This will be explained more fully in the next section, where we describe the processor chip itself. For now, it is sufficient to know that the portions of a machine cycle are so differentiated.

The cycle decoder produces a four-line exclusive output, indicating the kind of sub-cycle in progress. The $\overline{PCW}$ and $\overline{PCC}$ outputs are used by the processor module's control logic. All four signals are available, for controlling external circuitry.

The bus switching logic coordinates the use of the processor chip's main data bus. This function is necessary, as we noted earlier, if we are to prevent conflict among the many devices that ultimately share the main data bus. Bus switching logic consists of the input multiplexer, the input and output gating sections, and the logic that controls these functions.

Control logic for the bus switching section receives signals from the timing generator and from the cycle decoder. Inputs include T3A, $\overline{PCC}$, and $\overline{PCW}$, as well as signals from the interrupt logic and the hold logic. From these, the control logic is able to sense an input or an output operation and can determine which of the external devices should be granted access to the main data bus.

The input multiplexer is a three-way switch which selects one of three eight-line input channels and forwards it to the input gating section. Input signals from the control logic enable the multiplexer to select data from memory, data from the input peripherals, or data from the interrupt bus for input to the processor.

The input gating section receives T3A and $\overline{PCW}$ signals, from the timing generator and the cycle decoder respectively. These allow the gate to forward the multiplexer's output to the processor, at precisely the right moment.

The output gating section controls the output data bus, which is shared by memory and by the output peripherals. The output data bus will normally be enabled continuously. The only time that the module's output bus is inhibited is during direct memory access (DMA) operations. A control signal from the hold logic disables the output gating section when such an operation is in progress.

The address latch consists of two eight-bit latch sections, both of which receive their data inputs from the processor's main data bus. One latch receives the $\overline{\text{T1}}$ timing signal as a strobe, and the other receives $\overline{\text{T2}}$. These latches thus register and hold the address which the processor sends out, during the T1 and T2 intervals of all processor subcycles. The address stored in the latches is presented to memory and to the peripherals continuously during the phase in progress.

The read/write control logic commands a two-state output line. This line signals memory when a write operation is in progress. If no write signal is present, a read occurs. The read/write control uses T3 and $\overline{\text{PCW}}$ to develop its output.

The wait logic monitors the $\overline{\text{WAIT REQUEST}}$ line from the system memory. If the memory is slow to respond to the processor's read or write command, the wait logic causes the processor to idle until the memory can complete the transaction. A $\overline{\text{WAIT}}$ signal is available to external circuitry, during the time that the processor is idling; this serves to acknowledge the wait request. A wait request may be of indefinite length, but the actual wait interval is always an even multiple of the processor's clock period.

The interrupt logic monitors the $\overline{\text{INTERRUPT REQUEST}}$ and the $\overline{\text{HALT INTERRUPT REQUEST}}$ lines from external devices. This section also receives a $\overline{\text{SYNCA}}$ signal from the timing logic. The interrupt section uses these inputs to develop an INTERRUPT signal which is **correctly**

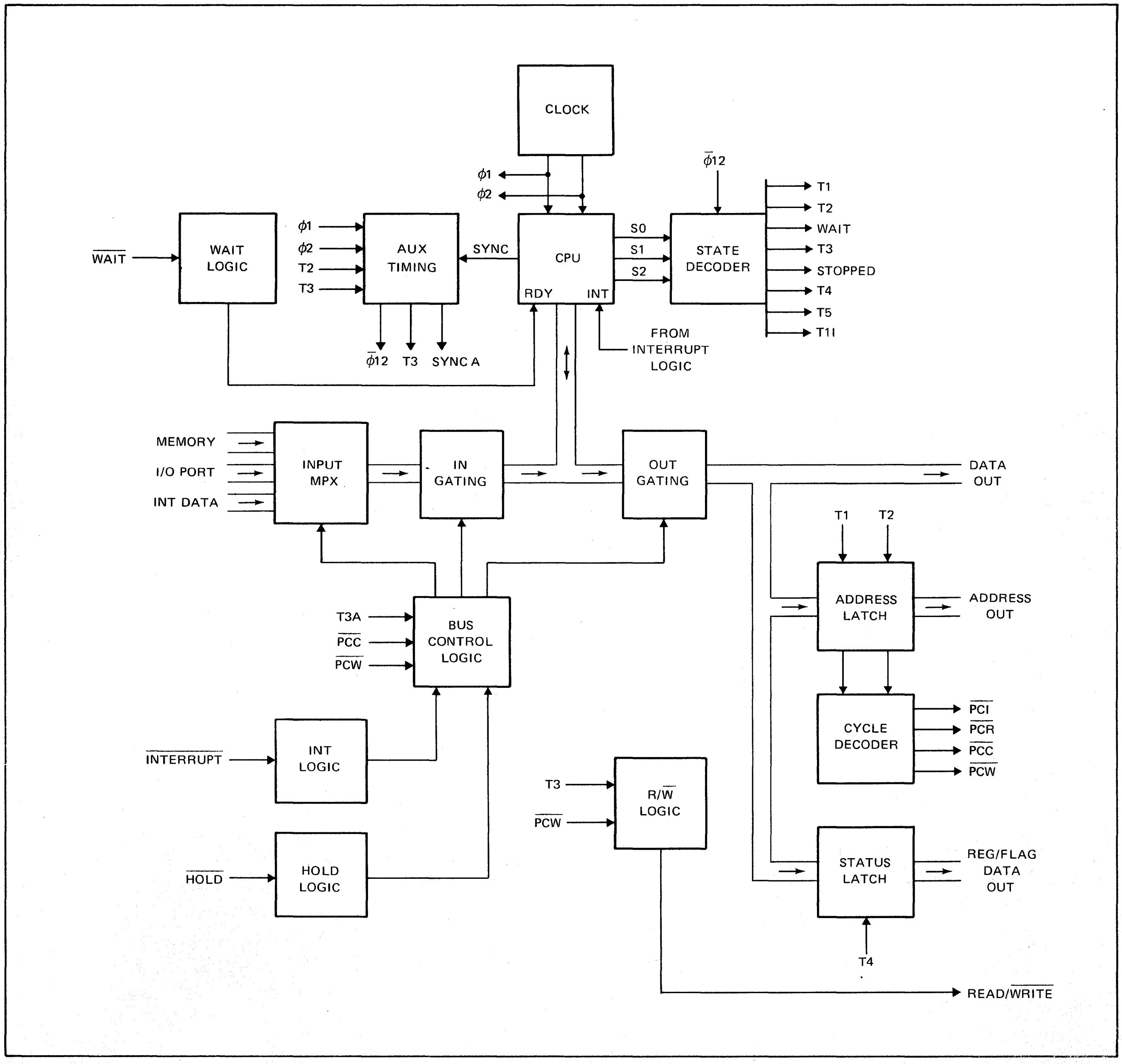


Figure 2–2. CPU Module Functional Block

synchronized with the processor module's ϕ_1 and ϕ_2 clock signals. An INTERRUPT REQUEST LATCH output is available externally, to acknowledge the interrupt request.

The processor module responds to an interrupt by altering the sequence of events that occurs during the next instruction fetch. The processor enters a special alternate phase (T1I), rather than going into the T1 phase as it normally would. As it customarily does, the processor sends out the lower eight bits in its program counter, but the counter itself is not incremented. This is the only difference in the fetch, as far as the processor chip is concerned. The T2 and T3 intervals which follow T1I are identical to those that occur in any other PCI sub-cycle.

However, peripheral logic is equipped to sense the entry into the T1I phase. It responds by sending a control signal to the input multiplexer, causing the multiplexer to select the interrupt instruction port instead of the processor's memory data in port. Thus the eight-bit word jammed into the interrupt port gets interpreted as an instruction by the processor.

Any instruction may be inserted, single or multiple byte. Synchronizing the presentation of successive bytes of a multiple byte instruction, however, requires some additional logic. For this reason, single byte instructions are preferred for interrupts. There are several possibilities.

A HALT instruction may be used to stop the processor upon completion of some task, manually or automatically. Or an output instruction may be used to output the accumulator's contents during a critical phase of the programming. Control and de-bugging are therefore two possible uses of the interrupt feature.

But by far the most convenient instruction for use with interrupts is the RESTART (abbreviated RST). The RST is a one byte call instruction especially intended for use with interruptive processing. The 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, 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 ordinary three byte jump, to a location where such a program is stored.

An important use of the RST instruction is the start-up of the processor chip, which always comes to rest in a HALT state after power is initially applied. The machine is started by means of an interruptive jump to memory location zero (or to some other desired location).

Note that in the INTELLEC 8/MOD 8 system the operator's console is the only device for which interrupt capability is provided.

The hold logic receives a $\overline{\text{HOLD REQUEST}}$ signal from one or more peripheral devices. It also receives $\overline{\text{WAIT}}$ and STOP signals from the timing generator. When a $\overline{\text{HOLD REQUEST}}$ coincides with a $\overline{\text{WAIT}}$ or a $\overline{\text{STOP}}$, the hold logic issues a HOLD ACKNOWLEDGE signal to the peripheral. At the same time the hold logic:

a) floats the module's address bus
b) floats the module's data out bus
c) floats the read/write line to memory
d) floats the $\overline{\text{I/O OUT}}$ output line
e) floats the $\overline{\text{I/O IN}}$ output line
f) disables the output from the state decoder

This action prevents the processor from exerting any influence on memory, either via the data bus or by means of external control signals. The peripheral originating the HOLD REQUEST is therefore free to command the memory until the WAIT REQUEST is retracted. Note that it is the WAIT REQUEST, not the HOLD REQUEST, that maintains the holding state.

The status latch is an eight-bit latch which receives its data input from the processor's output data bus. The strobe input to this banked latch is the T4 signal from the timing section. During the T4 interval, the 8008-1 broadcasts the state of its four status flags (carry, sign, zero, and parity). This information is saved in the latch and made available to external circuitry. It has no effect on the internal operation of the Central Processor Module.

8008-1 EIGHT-BIT PARALLEL CENTRAL PROCESSOR UNIT

A brief description of the 8008-1 CPU chip is essential to a thorough understanding of the imm8-82 Central Processor Module.

Capabilities of the 8008-1

The 8008-1 is a selected version of Intel's 8008 CPU. The device is chosen for its ability to run at high speed, and has a basic cycle time of 12.5 microseconds. By way of contrast, the basic cycle of the standard 8008 is 20 microseconds.

The list of the 8008's capabilities reads much like a description of the imm8-82 Central Processor Module itself. In a very real sense, it is the chip processor that gives the module its "personality." The CPU chip has a repertoire of 48 instructions, with provision for arithmetic and logical operations, register-to-register and register-to-memory transfers, subroutine handling, and I/O transactions. Four status flags permit conditional branching, based on carry, sign, zero, and parity.

The 8008 can access 16,384 memory locations directly, and this inherent ability can be extended further through the use of bank-switching. The chip has six index registers (scratchpad). An eight-level, fourteen-bit stack and program counter permits the nesting of subroutines up to seven levels. Built-in interrupt capability and synchronization provision for slow memories round out the chip's capabilities.

Clock Timing

The 8008-1 is driven by a two-phase TTL oscillator, at a maximum frequency of 800 kHz. All processing activities are referred to the period of this clock. Ten clock periods establish the basic system cycle of 12.5 microseconds.

The two clock phases are labeled ϕ_1 and ϕ_2. External circuitry provides these reference signals. The processor chip contains divide-by-two logic which derives the SYNC signal from ϕ_2.

Observe that a state is defined roughly as one full cycle of the SYNC pulse, and that two cycles of ϕ_1 and ϕ_2 occur during each processor state. The positive-going leading edge of the ϕ_1 clock precedes that of the ϕ_2 clock. For this reason, ϕ_1 is generally used to pre-charge bus and signal lines, preparatory to a data transfer. The ϕ_2 clock is then used as the strobe that pulses the destination register.

The Processor Cycle

As we mentioned before, a machine cycle consists of two parts. The first is the instruction fetch, and the second is the execution. The 8008, however, uses a variable-length machine cycle. The fetch routine is the same for all instructions, but the duration of the execution portion depends upon the kind of instruction that is fetched. Many uses find this variable cycle confusing, and it is therefore worthwhile to spend a little time on this subject.

Every machine cycle consists of one, two, or three sub-cycles. Each sub-cycle, in turn, consists of three, four, or five states. A state is defined as a constant interval, equal to two periods of the clock oscillator. That is, a state is so defined in all but two cases. Exceptions to the rule are the WAIT state and the STOPPED state, already described in earlier sections. A moment's consideration assures us that this is reasonable, since the halt and the wait are by nature indeterminate in length. It is worth noting, however, that even the STOPPED and WAIT states must be synchronized with the clock pulses. Both states occur in even multiples of the integral clock period.

To summarize then, two clock *periods* make a state; three to five *states* make a *sub-cycle*; and one to three *sub-cycles* make a complete *machine cycle*. A full cycle requires anywhere from three to eleven states for its completion (7.5 microseconds to 27.5 microseconds), depending on the kind of instruction involved.

CYCLE ENCODING

Let's concentrate for the moment on the question of sub-cycles. Just one consideration determines how many sub-cycles are required for a given instruction cycle: the number of times that the processor must reference a memory address, or an addressable peripheral device. The 8008, transmits one address during any given sub-cycle. Thus, if an instruction requires two memory references, then the machine cycle requires two sub-cycles. If three such references are necessary, then the machine has three sub-cycles.

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

Consider some examples. The halt instruction (HLT) is an instruction that requires only a single sub-cycle (PCI) for its completion. Once the processor has fetched and decoded this instruction, the only executive action required is to suspend the output from the processor's internal timing section. This is quickly accomplished. The fetch of the instruction and its execution require only four states, and only one reference to memory is necessary.

At the other extreme is the jump instruction (JMP). Execution of the jump requires three sub-cycles (PCI/PCR/PCR). This is true, because the jump is a three-byte instruction. The first byte contains the definitive instruction code. The second and third bytes contain the lower eight bits and the upper six bits of the jump address, respectively. These three bytes are stored in three successive memory locations, and the processor must access memory three times in order to obtain the information that it needs. The first and second sub-cycles require three states each, while the third requires five. The entire machine cycle takes eleven states (27.5 microseconds).

Most instructions fall between the extremes typefied by the halt and the jump instructions. The input (IN) and output (OUT) instructions, for example, require only two sub-cycles: one to fetch the instruction from memory, and one to address the object peripheral (PCI/PCC).

To reiterate information given previously, there are four possible sub-cycles that may occur in a machine cycle. They are identified as the PCI (instruction fetch), the PCR (memory read), the PCW (memory write), and the PCC (input/output). The sub-cycles that occur in any particular machine cycle depend upon the instruction type, with the overriding stipulation that the first sub-cycle is always a PCI.

The processor identifies the sub-cycle in progress, by sending out two CYCLE bits during the second state of every sub-cycle. These may be latched and decoded, and used to develop control signals for external circuitry. The identification bits are carried on the D_6 and D_7 lines of the processor's main data bus. The encoding is shown in Table 2-1.

Cycle Control Coding

CYCLE	D6	D7
PCI	0	0
PCR	0	1
PCC	1	0
PCW	1	1

Table 2-1.

STATE ENCODING

Every sub-cycle within the machine cycle consists of from three to five states. The number of states depends upon the instruction being performed and on the particular sub-cycle within the greater machine cycle.

The processor indicates its internal state by means of three encoded lines which emanate from the chip. These STATE lines (S_0 - S_2) are decoded by external circuitry, to determine which state is in progress. Table 2-2 shows how information on the STATE lines is encoded.

State Control Encoding

STATE	S_0	S_1	S_2
T1	0	1	0
T1I	0	1	1
T2	0	0	1
WAIT	0	0	0
T3	1	0	0
STOPPED	1	1	0
T4	1	1	1
T5	1	0	1

Table 2-2.

Every sub-cycle within the machine cycle passes through at least three states. During the first, the lower eight bits of a memory address are sent out onto the main data bus. In the second, the processor transmits a field consisting of six address bits and two CYCLE control bits on the eight lines of the main data bus. Bus outputs D_0 through D_5 carry the upper six bits of the referenced memory address. D_6 and D_7 carry the cycle control bits, as described in the previous section. External circuitry must capture this address information and present it to the memory, in parallel.

Once the processor has sent an address to memory, there is an opportunity for the memory to request a WAIT. This is done by pulling the processor's READY line low, prior to the trailing edge of the last ϕ_2 clock pulse in T2 (ϕ_{22}). As long as the line remains low, the processor will idle. During this state, the STATE lines boradcast a WAIT condition to the external circuitry.

The WAIT period may be of indefinite duration, but always consists of an even number of integral clock periods. **In order to guarantee an exit from the WAIT state, the processor's READY line must go high at least 350 nanoseconds prior to the trailing edge of ϕ_{22}.** When this condition is fulfilled, the processor proceeds to the T3 state, beginning with the next ϕ_1 clock pulse.

The events that take place during the T3 state depend upon the kind of sub-cycle in progress. In a PCI cycle, the processor interprets the data on its bus as an instruction. During PCR, the bus contents is construed as data. The processor itself outputs data during a PCW sub-cycle. And in a PCC sub-cycle, the processor may either transmit or receive data, depending on the kind of I/O instruction.

After the T3 state, it becomes extremely difficult to generalize. Almost every instruction has a unique sequence of events. If a halt (HLT) instruction is fetched, the processor enters the STOPPED state at the end of T3. While the machine is halted, the STATE lines indicate this condition to the external logic circuitry. An INTERRUPT is required to restart the machine (this is explained in a later section).

The T4 and the T5 states are available, if the execution of a particular instruction requires them. If not, the processor may skip one or both states and proceed directly to T1 of the next sub-cycle. Again, this depends upon the kind of instruction fetched, and on the particular sub-cycle in progress. T4 and T5 are reserved in all cases for internal processor operations. **No external device is ever referenced during T4 or T5.**

The chart contained in Table 2-3 shows the state sequence involved in the execution of each kind of instruction. You should refer to that table, if you have questions on how a specific instruction is executed. The processing activity associated with each state is briefly summarized.

The T1I state is an alternative to the T1 state. It is used only for interrupts. An INTERRUPT request is always acknowledged at the beginning of a machine cycle, to prevent the abort of any instruction that may have been in progress when the request arrived. The 8008-1 responds to an INTERRUPT by entering the T1I state, rather than T1, during the PCI sub-cycle. The program counter is not incremented during T1I, as it normally would be in T1, a provision which permits the interrupted program to be resumed following the INTERRUPT. The processor indicates the T1I condition on its STATE lines, but that is the only other departure from a normal PCI sub-cycle.

The remainder of the INTERRUPT handling is delegated to external logic. It is up to external circuitry to interpret T1I as an acknowledgement of the INTERRUPT request. Upon receipt of this acknowledgement, the external logic is required to disconnect the processor from the memory data in bus. This permits the interrupting device to

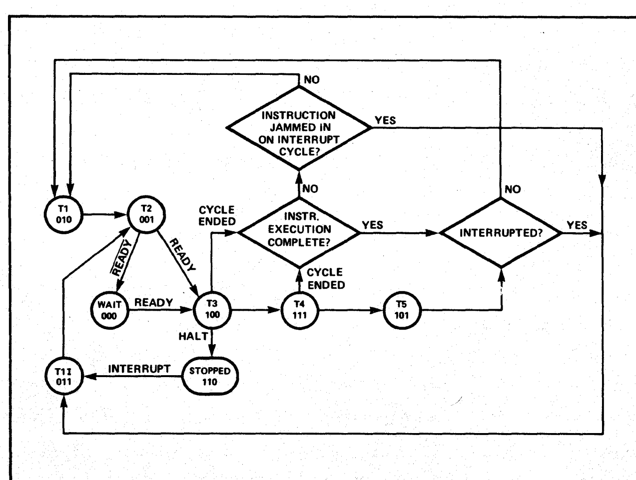


Figure 2-3. CPU State Transition Diagram

"jam" an eight-bit instruction word directly onto the processor's data bus during T3. For multi-byte interrupt instructions, the CPU continues to generate T11 instead of T1.

Architecture of the 8008-1

Internally, the 8008-1 consists of:

a) timing generator
b) program counter/stack
c) instruction register and decoder
d) arithmetic logic unit (ALU)
e) index registers (scratchpad)
f) I/O buffer
g) memory refresh circuitry

Figure 2-4 is a functional block diagram of the 8008-1.

The timing generator accepts the ϕ_1 and ϕ_2 inputs from the external clock oscillator, and uses them to develop the SYNC output. From these signals, the timing logic develops an array of timing signals that coordinate the activities of all other functional blocks, as well as producing the coded STATE outputs to external circuitry. The timing generator consists of the CLOCK GENERATOR, STATE TIMING GENERATOR, MACHINE CYCLE CONTROL, and STATUS SIGNALS blocks shown in Figure 2-4.

The program counter and stack is a dynamic memory array containing 8 fourteen-bit registers, pointer logic, and counter incrementation facilities. It is configured as a revolving pushdown stack, with a wrap-around pointer. This section maintains the memory address of the current program instruction, as well as the return addresses for up to seven nested subroutines. The program counter is incremented automatically during every PCI sub-cycle, and the stack is managed through the use of ten specialized instructions that include conditional jumps and returns. A short-form call instruction (RST) is available for use with interrupts.

The instruction register stores the eight-bit instruction word that is returned to the processor during PCI-T3. This instruction code is presented to the instruction decoder, which also receives inputs from the timing section. The timing signals are combined with the decoder's output, to develop the command signals that control the chip's other circuitry.

The arithmetic logic section contains the accumulator register, the two temporary holding registers (a and b), the adder, and the status bit logic. This section is equipped to perform both arithmetic and boolean logic operations, in parallel, on eight-bit binary quantities. All logical manipulation of data takes place in the ALU, under supervision of the instruction control logic.

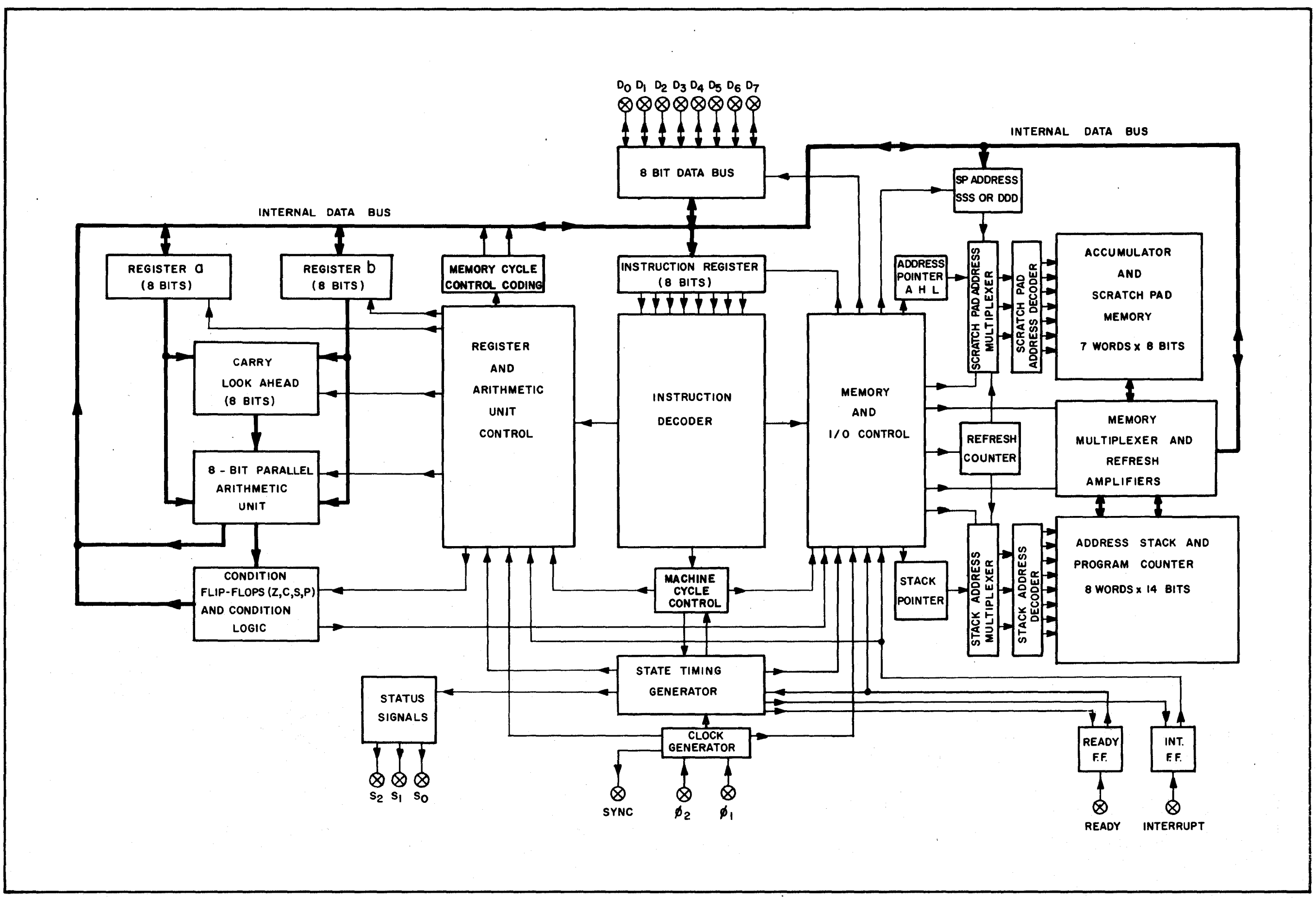


Figure 2-4. 8008-1 CPU Block Diagram

INSTRUCTION CODING $D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	OPERATION	# OF STATES TO EXECUTE INSTRUCTION	SUB-CYCLE ONE (1) T1 (2)	T2	T3	T4 (3)	T5
INDEX REGISTER INSTRUCTIONS									
1 1	D D D	S S S	MOV r_1, r_2	5	PC_L OUT (4)	PC_H OUT	FETCH INSTR.(5) TO IR & REG. b	SSS TO REG. b (6)	REG. b TO DDD
1 1	D D D	1 1 1	MOV r, M	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→ (7)	→
1 1	1 1 1	S S S	MOV M, r	7	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	→
0 0	D D D	1 1 0	MVI r	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	1 1 1	1 1 0	MVI M	9	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	D D D	0 0 0	INR r	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	ADD OP - FLAGS AFFECTED
0 0	D D D	0 0 1	DCR r	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	SUB OP - FLAGS AFFECTED
ACCUMULATOR GROUP INSTRUCTIONS									
1 0	P P P	S S S	ALU OP r	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	ALU OP - FLAGS AFFECTED
1 0	P P P	1 1 1	ALU OP M	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	P P P	1 0 0	ALU OP I	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	0 0 0	0 1 0	RLC	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 0 1	0 1 0	RRC	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 1 0	0 1 0	RAL	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 1 1	0 1 0	RAR	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS									
0 1	X X X	1 0 0	JMP	11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	0 C C	0 0 0	JNC, JNZ, JP, JPO	9 or 11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	1 C C	0 0 0	JC, JZ, JM, JPE	9 or 11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	X X X	1 1 0	CALL	11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	0 C C	0 1 0	CNC, CNZ, CP, CPO	9 or 11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	1 C C	0 1 0	CC, CZ, CM, CPE	9 or 11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	X X X	1 1 1	RET	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	POP STACK	X
0 0	0 C C	0 1 1	RNC, RNZ, RP, RPO	3 or 5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X
0 0	1 C C	0 1 1	RC, RZ, RM, RPE	3 or 5	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X
0 0	A A A	1 0 1	RST	5	PC_L OUT	PC_H OUT	FETCH INSTR. TO REG. b AND PUSH STACK (0→REG. a)	REG. a TO PC_H	REG. b TO PC_L (14)
I/O INSTRUCTIONS									
0 1	0 0 M	M M 1	IN	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	R R M	M M 1	OUT	6	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→	→
MACHINE INSTRUCTIONS									
0 0	0 0 0	0 0 X	HLT	4	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b & HALT (18)		

NOTES:
1. The first sub-cycle is always a PCI (instruction) cycle.
2. Internally, states are defined as T1 through T5. In some cases more than one sub-cycle is required to execute an instruction.
3. Content of the internal data bus at T4 and T5 is available at the data bus. This is designed for testing purposes only.
4. Lower order address bits in the program counter are denoted by PC_L and higher order bits are designated by PC_H.
5. During an instruction fetch the instruction comes from memory to the instruction register and is decoded.
6. Temporary registers are used internally for arithmetic operations and data transfers (Register a and Register b.)
7. These states are skipped.
8. PCR cycle (Memory Read Cycle).
9. "X" denotes an idle state.
10. PCW cycle (Memory Write Cycle).
11. When the JUMP is conditional and the condition fails, states T4 and T5 are skipped and the state counter advances to the next memory cycle.

Table 2-3. State Transition Sequence

SEQUENCE

SUB-CYCLE TWO					SUB-CYCLE THREE					# BYTES
T1	T2	T3	T4 (3)	T5	T1	T2	T3	T4 (3)	T5	
										1
REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X (9)	REG. b TO DDD						1
REG. L OUT (10)	REG. H OUT	REG. b TO OUT								1
PC_L OUT (8)	PC_H OUT	DATA TO REG. b	X	REG. b TO DDD						2
PC_L OUT (8)	PC_H OUT	DATA TO REG. b	→	→	REG. L OUT (10)	REG. H OUT	REG. b TO OUT			2
										1
										1
										1
REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X	ALU OP - FLAGS AFFECTED						1
PC_L OUT (8)	PC_H OUT	DATA TO REG. b	X	ARITH OP - FLAGS AFFECTED						2
										1
										1
										1
										1
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L	3
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L	3
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L	3
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L	3
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a (12)	REG. a TO PC_H	REG. b TO PC_L	3
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→	→	PC_L OUT (8)	PC_H OUT	HIGHER ADD. REG. a (12)	REG. a TO PC_H	REG. b TO PC_L	3
										1
										1
										1
										1
REG. A TO OUT (15)	REG. b TO OUT	DATA TO REG. b	COND ff OUT (16)	REG. b TO REG. A						1
REG. A TO OUT (15)	REG. b TO OUT	X (17)								1
										1

12. When the CALL is conditional and the condition fails, states T4 and T5 are skipped and the state counter advances to the next memory cycle. If the condition is true, the stack is pushed at T4, and the lower and higher order address bytes are loaded into the program counter.
13. When the RETURN condition is true, pop up the stack; otherwise, advance to next memory cycle skipping T4 and T5.
14. Bits D_3 through D_5 are loaded into PC_L and all other bits are set to zero; zeros are loaded into PC_H.
15. PCC cycle (I/O Cycle).
16. The content of the condition flip-flops is available at the data bus: S at D_0, Z at D_1, P at D_2, C at D_3. (D_4 – D_7 all ones)
17. A READY command must be supplied for the OUT operation to be completed. An idle T3 state is used and then the state counter advances to the next memory cycle.
18. When a HALT command occurs, the CPU internally remains in the T3 state until an INTERRUPT is recognized. Externally, the STOPPED state is indicated.

Table 2-3. State Transition Sequence (continued)

The 8008-1 contains six general purpose, eight-bit index registers. These serve as convenient working storage for the intermediate results of the ALU's operations. The registers are designated B, C, D, E, H, and L. The H and L registers are also provided with logic that permits them to serve as pointers during the execution of memory reference instructions. The H register holds the upper six bits, and the L register holds the lower eight. The contents of these registers are sent out as an address in lieu of the program counter, during the PCR and PCW sub-cycles of memory referencing operations.

The I/O buffer controls the flow of data, between the processor's internal data bus and the external data bus. It receives signals from the timing and control sections that permit it to perform the necessary gating functions.

All storage on the 8008-1 is of the dynamic type. That is, it consists of capacitor-like elements which are charged to specific levels in order to store a binary "1" or "0." This is true of the accumulator, the index registers, the stack pointer, and all other similar provisions. Leakage would soon destroy the information stored in these elements, if they were not scanned periodically and "refreshed" as necessary. Active circuits on the chip perform this function. That is the purpose of the refresh provision. The internal memories are scanned automatically during WAIT, T3, and STOPPED phases. Under worst-case conditions, a complete refresh cycle occurs every eighty clock periods.

8008-1 Instruction Set

The instruction set of the 8008-1 consists of 48 instructions, in four logical groups.

Seven index register instructions permit the transfer of data, between individual registers and between registers and memory. Two instructions enable the programmer to increment and decrement the contents of any register (≠A).

Twenty-eight accumulator group instructions permit a variety of arithmetic and logical manipulations. There are twenty-four ALU instructions, divided into three groups of eight. The three groups are: a) those operations that reference index registers, b) those operations that reference memory via the H and L pointer, and c) those operations that reference "immediate" memory locations. ALU operations in each category allow for add and subtract operations (with or without carry/borrow), boolean AND, OR, and EXCLUSIVE-OR operations, and equality tests involving the accumulator. Four shift instructions permit shifting the accumulator left and right, through or around the CARRY bit.

Ten stack control instructions provide for jumps, calls, and returns, both unconditionally and based upon tests of the four status bits (carry, sign, zero, and parity). A special one-word call, the restart (RST), is provided for use with interrupts.

The instruction set of the 8008-1 contains two I/O instructions. IN provides for transferring an eight-bit word to the accumulator, from one of eight input ports implied in the instruction field. OUT causes the contents of the accumulator to be output to one of 24 implicit output addresses.

Also included in the 8008's repertoire are two machine instructions: the no-operation (NOP) and the halt (HLT). The NOP is actually a register-to-register transfer, in which the source and the destination registers specified are the same. The HLT causes the processor to enter the STOPPED state (an INTERRUPT is required to exit from STOPPED state.)

Interrupt

The 8008-1 contains a built-in interrupt facility. An INTERRUPT request is initiated by pulling the processor's INTERRUPT line high.

Transitions on the INTERRUPT line must be synchronized to impulses of the clock used to drive the chip. Specifications state that the INTERRUPT line must not be permitted to change within 200 nanoseconds of the high-to-low transition of the ϕ_1 clock. The most convenient synchronizing impulse thus becomes the low-to-high leading edge of the ϕ_2 clock. Synchronization of the INTERRUPT request in this fashion produces an INTERRUPT signal that precedes the falling edge of ϕ_2 by more than 200 nanoseconds, and one that at the same time allows more than 200 nanoseconds between the ϕ_1 clock's trailing edge and the high-to-low transition fo the INTERRUPT request. The timing of an INTERRUPT is illustrated in Figure 2–5.

A properly synchronized INTERRUPT request is acknowledged at the beginning of a machine cycle. Instead of entering the PCI-T1 state, as usual, the interrupted processor enters an alternative state: PCI-T1I. The only difference between T1I and T1 is that the processor's internal program counter is not incremented during a T1I state. Thus, the program does not advance during an INTERRUPT cycle. This permits the interrupted program to resume its execution following the INTERRUPT. The processor acknowledges the INTERRUPT by placing $\overline{S_0}$-S_1-S_2 on the STATE output lines, during the T1I state.

The interrupt cycle is otherwise indistinguishable from an ordinary PCI subcycle. The processor itself takes no further special action. It is the responsibility of the peripheral logic to see that the desired interrupt instruction is "jammed" onto the processor's data bus at PCI-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 jammed onto the main bus at T3 just like any other fetched instruction. Thus, any of the 48 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 reference sub-cycles (PCR), fully expecting that the proper information will be on the bus at the proper time. Providing such data at the right time will involve some additional peripheral logic. For this reason, one-byte instructions are preferred for use with interrupts.

A special one-byte call is provided for use with interrupts (the ordinary program call takes three bytes). This is the restart instruction (RST). The eight bits of the RST contain a variable three-bit field, which enables the interrupting device to direct 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 addresses may be used to store the first byte of a routine designed to service the requirements of an interrupting device.

Start-Up Of The 8008-1

When power is initially applied to the 8008-1, the processor enters the STOPPED state automatically. The next sixteen clock periods are used to clear all dynamic storage, including accumulator, index registers, and the stack. The processor may then be started.

An INTERRUPT is always required, in order to exit the STOPPED state. The use of the INTERRUPT is described in the preceding section.

Electrical Characteristics and Timing of the 8008-1

The next two pages provide a complete electrical description of the Intel 8008/8008-1, for those who require this information.

PERIPHERAL LOGIC

In this section, we describe the peripheral logic on the imm8-82 Central Processor Module, the logic which supports the activities of the 8008-1 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 a few descriptive examples of module operations, showing how the peripheral logic extends the basic capabilities of the 8008-1 CPU chip.

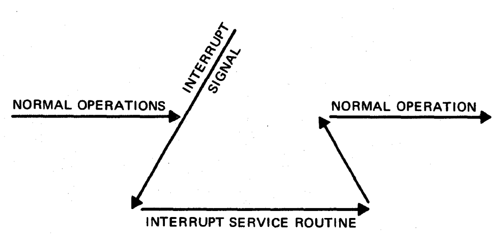


Figure 2-5. Interrupt Timing.

Timing Logic

The timing logic consists of a crystal controlled clock oscillator, the state decoder, and auxiliary timing logic. These provisions are shown on the module schematic, Figure 2-8.

The clock oscillator furnishes two non-overlapping clock phases, at 800 kHz, to the TTL-level inputs of the 8008-1 CPU. The clock outputs are also used by the auxiliary timing logic, to develop other necessary timing signals. The clock oscillator consists of components shown in the upper central portion of the module schematic.

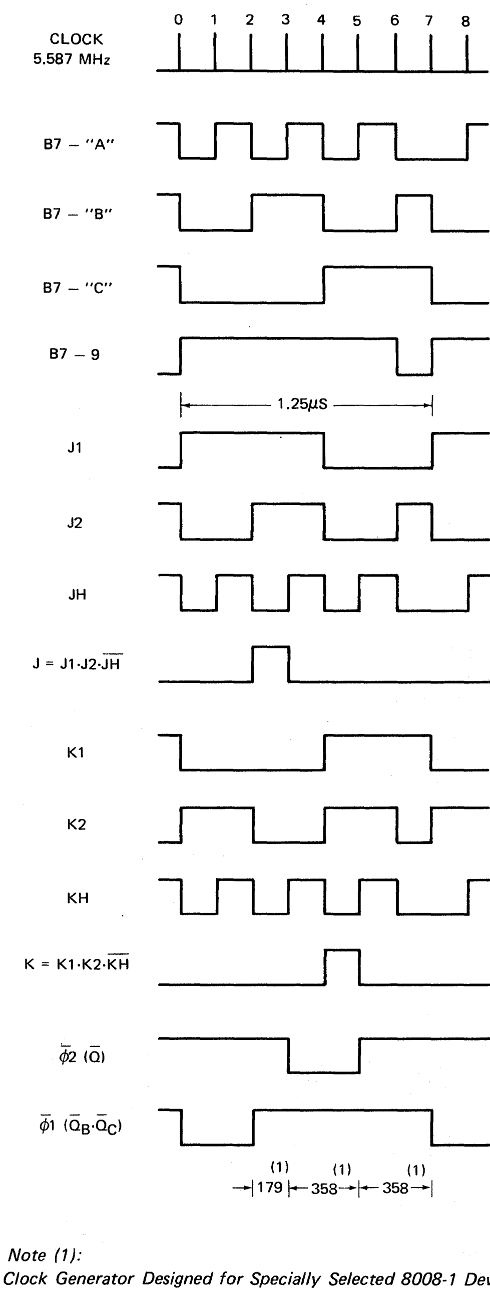


Figure 2-6. CPU Clock Timing

The 5.587 MHz quartz crystal Y1 is the basic frequency reference. A portion of the crystal's signal output is developed across C1 and applied to a 74H04 inverter section. Two cascaded inverters are used here, and each has a feedback resistor connected between its input and its output. Both inverters are thus operating as operational amplifiers, at a reduced gain. Together, they provide the amplification and the phase shift necessary to sustain oscillation in the crystal.

The output of the oscillator is applied to the input of a Fairchild 9316 binary counter, through a NAND-gate used as an inverter. A second NAND-gate section senses the coincidence of Q_B and Q_C outputs, and clamps the PE input (#9) low on the sixth count following reset. This enables the seventh clock pulse to reset the 9316. The output of the counter is used to produce two non-overlapping clocks at 800 kHz.

The $\overline{QB}$ and $\overline{QC}$ outputs of the 9316 are ANDed, to generate the ϕ_1 clock. A different technique is used to create ϕ_2. The QA, QB, and $\overline{QC}$ outputs are applied to the "J" inputs of a 7470 J-K flip-flop. Inputs to the "K" section of the flip-flop are QA, $\overline{QB}$, and QC. Operating as it is, in the steered mode, the flip-flop reacts to the third and fifth clock pulses in each counter cycle. The output of the flip-flop is ϕ_2. The timing relationships within the clock section are diagrammed in Figure 2-6.

The ϕ_1 and the ϕ_2 clock phases are applied to the clock inputs of the CPU chip, which produces a SYNC output derived from ϕ_2. Then SYNC and clock signals are fed to the auxiliary timing logic.

In the auxiliary logic section the $\overline{SYNC}$ is applied to the D input of a 7474 latch section which is also clocked by the low-to-high transition of ϕ_2. This produces the SYNCA signal which stands in a predictable relationship to the ϕ_2 clock (note that the relationship between the trailing edge of ϕ_2 and the leading edge of the processor's SYNC output varies from chip to chip, as shown in Figure 2-7). SYNCA is used to synchronize external interrupt requests, and in the derivation of other timing signals on the module.

$\overline{SYNCA}$ and the ϕ_1 clock are ANDed in a 74H00 NAND-gate section, to obtain the half frequency clock ϕ_{12}. The derivation of this signal is shown in the module timing diagram, Figure 2-7. $\overline{\phi_{12}}$ is an intermediate signal, used in the derivation of other timing pulses.

The $\overline{\phi_{12}}$ clock is applied to the ENABLE input of a 3205 Three-to-Eight Line Converter, used as the module's state decoder. The DATA inputs to the 3205 are the STATE lines emanating from the processor ship. This produces a pulsed, exclusive eight-line output which is used directly to control and time many of the module's activities. Outputs from the state decoder include: $\overline{T1}$, $\overline{T1I}$, $\overline{T2}$, $\overline{WAIT}$, $\overline{T3}$, $\overline{STOPPED}$, and $\overline{T4}$ (a T5 output is available, but not used).

The $\overline{T2}$ output of the state decoder is forwarded to the D input of a 3404 latch section, which is strobed by the negative-going transition of $\overline{\phi_{12}}$. The intermediate timing signal produced at the latch's output is called $\overline{T2L}$. It occupies the interval between the leading edge of the T2 pulse and the leading edge of T3, as shown in Figure 2-7. $\overline{T2L}$ is used solely to derive the T3A timing signal.

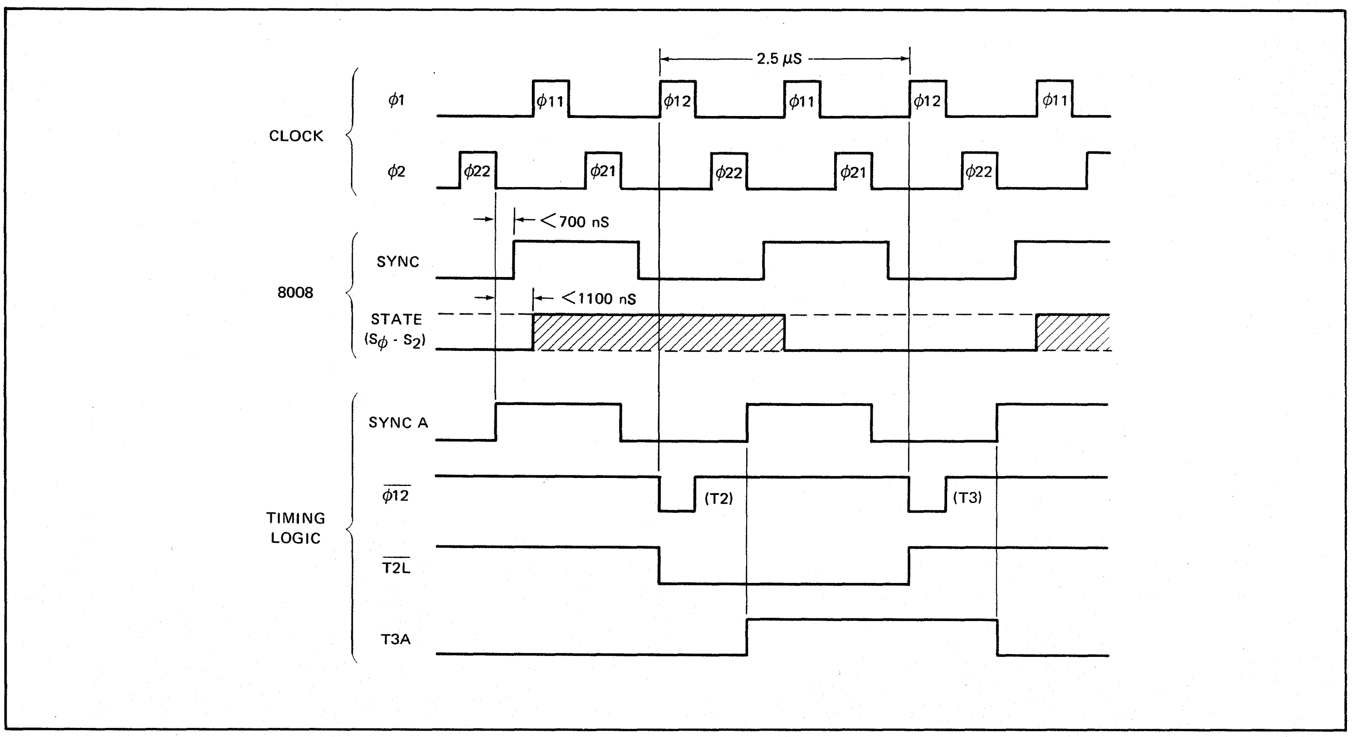


Figure 2–7. CPU Module Timing

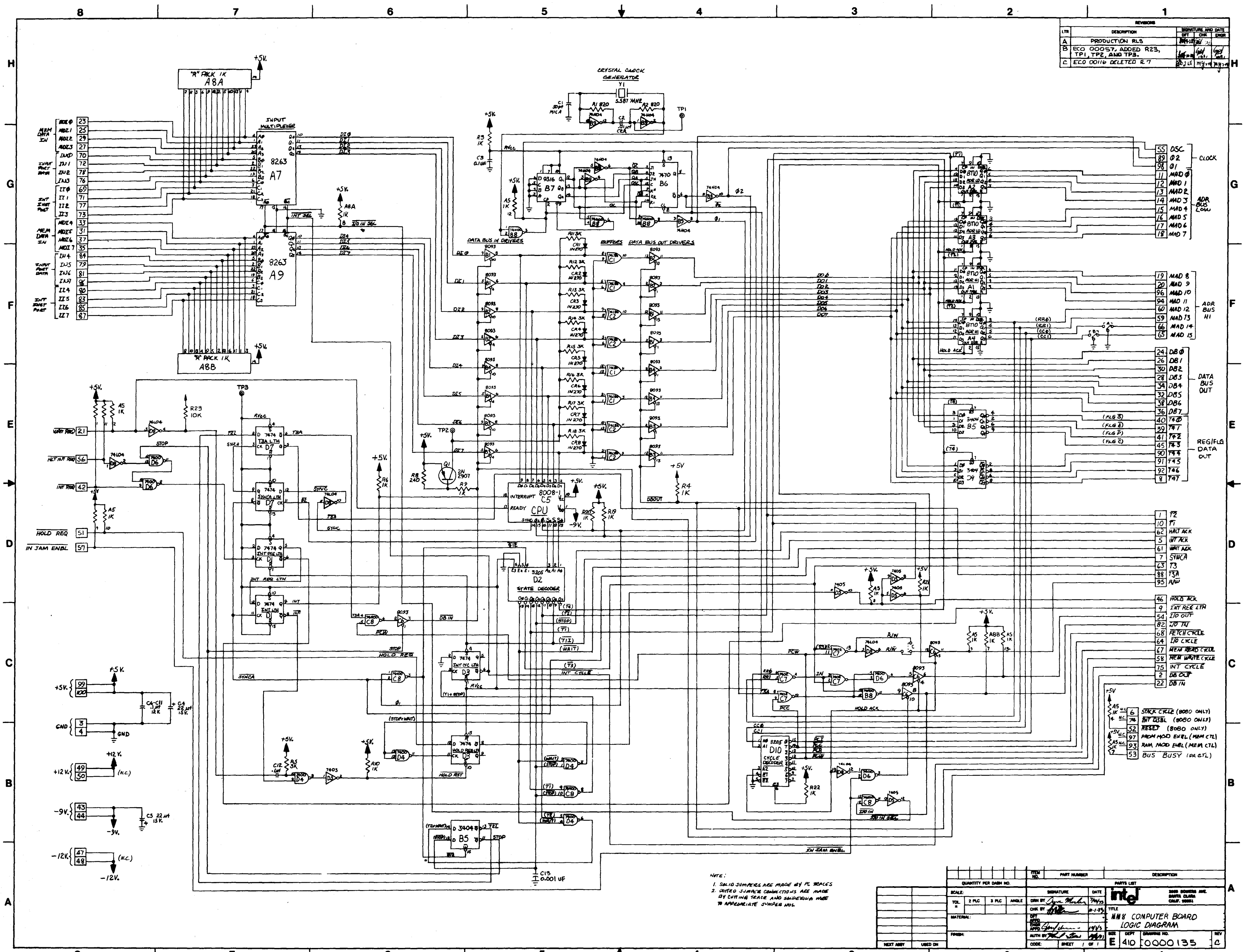


Figure 2–8. Central Processor Module Schematic

A 7474 latch section is used to generate T3A. The clocking input to the latch is the low-to-high transition of the SYNCA signal. The D input is $\overline{\text{T2L}}$. T3A and $\overline{\text{T3A}}$ signals are produced at the latch's Q outputs.

Refer to the module timing diagram, Figure 2-7, and observe that the so-called T3A signal precedes the actual T3 signal, by some 1608 nanoseconds. By using T3A as a gating signal during I/O input operations, we allow ample time for the input device to precharge the processor's data bus prior to the actual transfer of data.

Instruction Fetch (PCI)

An instruction fetch (PCI sub-cycle) is the first part of every machine cycle. The events that take place during an instruction fetch are as follows.

The processor chip transmits the lower eight bits of the referenced location during T1. This byte is sent out on the eight lines of the main data bus and presented to the address latches, A1 through A4. Two of the latches are strobed by the $\overline{\text{T1}}$ output of the state decoder, causing them to register and hold the address byte.

During T2, the processor chip sends out the six high order bits of the referenced address, plus the two CYCLE bits, in similar fashion, The $\overline{\text{T2}}$ output of the state decoder is used to strobe the remaining two address latches, and these elements save this information.

The fourteen low order bits held in the address latches point the location of the instruction that the processor intends to fetch. The two remaining bits indicate that a PCI sub-cycle is in progress. The CYCLE information is applied to the cycle decoder (D10).

The cycle decoder is an Intel 3205 Three-to-Eight Line Converter, used here to provide an exclusive four-line output. Each of the decoder's output lines indicates when one of the four sub-cycles (PCI, PCR, PCW, or PCC) is in progress. This information is available, for the control of external devices. In addition, the $\overline{\text{PCC}}$ and $\overline{\text{PCW}}$ outputs are furnished to circuitry on the PCU module itself, permitting the module's control logic to generate $\overline{\text{I/O IN}}$, $\overline{\text{I/O OUT}}$, and R/$\overline{\text{W}}$ control signals.

Under ordinary conditions, the two 12-to-4 line multiplexers, A7 and A9 select and forward the information on the memory data in bus to the processor module's eight-line input gate section. This tri-state buffer bank is enabled by the T3A timing signal, operating through a 74H00 NAND-gate and an 8093 section used here as a coincidence indicator. The output of the buffer is the $\overline{\text{DB IN}}$ signal, and this enables the gate to forward the information from the multiplexer section to the processor. During T3, the processor reads this bus, and the information on these lines is transferred to the processor's instruction register. This completes the fetch portion of the machine cycle.

Memory Reference Operations (PCR and PCW)

Every operation that the CPU performs is preceded by an instruction fetch sub-cycle (PCI) as just described. In the case of certain instructions, it may be necessary to reference memory one or more additional times in order to 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. During a PCR or a PCW cycle, the addressing, input multiplexing, and bus gating functions are handled in much the same way as for an instruction fetch.

As far as the peripheral logic is concerned there is one important difference. A $\overline{\text{PCR}}$ or a $\overline{\text{PCW}}$ signal will be broadcast by the processor chip during T2. If the CYCLE code indicates a PCR sub-cycle, then external conditions are exactly the same as for an instruction fetch from memory. If a PCW is indicated, two special actions are taken.

First, the cycle decoder activates the $\overline{\text{PCW}}$ line, and this level is applied to pin #5 of C8. The presence of a low inhibits the gate, disabling the $\overline{\text{DB IN}}$ signal and preventing the input devices from affecting data that is going out on the main bus.

And secondly, the $\overline{\text{PCW}}$ output is applied to a 7402 section used as a coincidence indicator. The coincidence of $\overline{\text{PCW}}$ and $\overline{\text{T3}}$ at the inputs to this gate generate a $\overline{\text{WRITE}}$ pulse on the R/$\overline{\text{W}}$ command line. The $\overline{\text{W}}$ signal indicates to the external memory that data on the module's output bus is to be stored in the addressed location.

I/O Operations

All input and output operations require two processor sub-cycles: A PCI to fetch the instruction, and a PCC to execute. The PCI sub-cycle is described in a previous section.

The instruction that the processor fetches from memory contains a five-bit field which specifies one of 32 peripherals. In order to distinguish an input from an output instruction, the lower eight addresses are reserved for input devices, and the upper 24 for outputs.

The address of the object I/O device is sent to the A_9 through A_{13} address latches during T2, to identify the object peripheral. Since only the lower eight addresses are used for input, the A_{12} and A_{13} lines will never be high unless an output operation is in progress. These two lines are therefore applied to the inputs of a 7402 section (C7-4/5/6) which produces a low during input operations and a high during output. The remaining circuitry uses this signal, in conjunction with $\overline{\text{PCC}}$, $\overline{\text{T3}}$, and $\overline{\text{T3A}}$, to produce $\overline{\text{I/O IN}}$ and $\overline{\text{I/O OUT}}$ control signals.

If an input operation is indicated, C7-4 will be high. The coincidence of a $\overline{\text{T3A}}$ signal and a $\overline{\text{PCC}}$ produces a high at C7-10. The outputs of these two gates are applied to the input of a 74H00 NAND-gate section, where they are ANDed to produce the $\overline{\text{I/O IN}}$ control signal. The $\overline{\text{I/O IN}}$ is buffered in an 8093 section and made available at the edge connector for use by the object peripherals.

On the Central Processor Module itself, the $\overline{\text{I/O IN}}$ is routed through a 74H00 section (used as OR) and a 7405 section and applied to the pin #17 inputs of the two input multiplexers, A7 and A9. This causes the multiplexers to select data from the external input ports, and forward this to the input gating section. The input gates are enabled by the presence of T3A, and the data from the addressed input device therefore passes to the processor, via the main data bus.

If an output operation is indicated, on the other hand, C7-4 will be low. This low is ANDed with $\overline{\text{T3}}$ in a 7402 section, to produce a positive-going pulse output at C7-1. This output is applied to one input of a 7400 NAND-gate. The other input to the gate is the output of C7-10 which, as we have seen, is high during the coincidence of $\overline{\text{T3A}}$ and the $\overline{\text{PCC}}$ cycle. As a result of the signals applied, D6-6 goes low during the T3 phase of output, producing the $\overline{\text{I/O OUT}}$ command. This signal is buffered in an 8093 section, and made available for the control of external devices.

Interrupt Cycle

From the point of view of the CPU chip, the interrupt cycle is simply a modified PCI. Externally, the function of the processor chip appears much the same.

Peripheral logic does the bulk of the work during an interrupt. It is the function of the peripheral logic to synchronize the external INTERRUPT REQUEST with the processor module's clocks. The instruction word presented to the module's interrupt port must also be switched onto the main data bus, at the appropriate time.

An incoming $\overline{\text{INTERRUPT REQUEST}}$ is applied to the clock input of a 7474 section, labelled INTERRUPT REQUEST LATCH on the module schematic. The latch stores the request, until such time as the module's logic can acknowledge it.

The resultant high at pin #5 of the INTERRUPT REQUEST LATCH is applied to the D input of the INTERRUPT LATCH itself. The clock input to this latch is the positive-going transition of the $\overline{\text{SYNCA}}$ signal, which coincides with the trailing edge of ϕ_{22}. Thus, the INTERRUPT LATCH registers the INTERRUPT REQUEST in proper synchronization with the module's reference clocks. The Q output of the INTERRUPT LATCH goes directly to the processor chip's INTERRUPT input pin.

As explained previously, the processor chip acknowledges the INTERRUPT by going into an alternate phase (T1I) at the beginning of the next PCI cycle. At this time, the processor chip's STATE lines indicate the T1I state to the state decoder.

The T1I output from the state decoder is used directly to set the INTERRUPT CYCLE LATCH, shown just below the processor chip on the module schematic. The Q output from the 7474 is available to external circuitry, indicating that the processor itself has honored the interrupt request. The output of the INTERRUPT CYCLE LATCH is also directed to the pin #16 control inputs of the A7 and A9 multiplexers, causing them to select and forward the data presented to the module's interrupt instruction port. This data passes through the input gating logic during T3A, onto the main data bus.

The INTERRUPT CYCLE LATCH is reset, immediately following the interrupt cycle, by a signal applied to its clock input. The latch may be reset by either the T1 or the $\overline{\text{T1}}$ or the $\overline{\text{STOPPED}}$ outputs of the state decoder.

Hold Operations

A $\overline{\text{HOLD REQUEST}}$ must always be preceded by a $\overline{\text{WAIT REQUEST}}$ applied to pin #21 of the module. The processor module must be waiting or stopped, before it can acknowledge a $\overline{\text{HOLD REQUEST}}$.

If the state decoder indicates that the processor is in the WAIT or the STOPPED state, a negative-going 400 kHz pulse will be applied to the clock input of the HOLD REQUEST LATCH. The coincidence of a low-to-high transition at the clock input and a HOLD REQUEST at the D input resets this latch.

The resulting high at pin 8 of the 7474 is applied to a 7405 inverter section, and the inverter's output furnishes a HOLD ACKNOWLEDGE to external circuitry. The output of the 7405 is also directed in parallel to the inputs of two more 7405s. The outputs of these inverters perform the following control functions:

a) float the address bus
b) float the module's 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 R/$\overline{\text{W}}$ control line
f) disable the cycle decoder

These actions ensure that the peripheral originating the $\overline{\text{HOLD REQUEST}}$ will have complete control of the memory's busses and control lines.

The $\overline{\text{HOLD REQUEST}}$ may be removed as soon as the module has acknowledged the request. The processor module will continue to hold, until the $\overline{\text{WAIT REQUEST}}$ is removed or until the module receives an $\overline{\text{INTERRUPT REQUEST}}$.

When the clamp on the $\overline{\text{WAIT REQUEST}}$ line is lifted, at the end of the DMA operation, D8-4 goes from high to low. The output of this inverter is coupled through C12 to one input of a 7400 NAND-gate. The gate's output passes through a 7405 inverter to the preset input of the HOLD REQUEST LATCH, setting the latch and terminating the hold.

An $\overline{\text{INTERRUPT REQUEST}}$ can also set the HOLD REQUEST latch. When the INTERRUPT LATCH registers an interrupt, its $\overline{\text{Q}}$ output goes from high to low. The low is coupled to D4-9, causing the hold latch to be set and immediately terminating the $\overline{\text{HOLD ACKNOWLEDGE}}$.

Whenever two or more peripherals in the same system have DMA capability, there is always a chance of conflict. One device may request a hold while the other is already in the process of conducting a transfer. Finding the 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 $\overline{\text{BUS BUSY}}$ state status line. Pin #53 of the module is returned internally to the +5 Volt supply, through a 1K pullup 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.

UTILIZATION

This section provides installation and utilization information for the imm8-82 Central Processor Card application.

Installation Requirements

The installation requirements for the imm8-82 Central Processor Card are given in Table 2-4.

imm8-82 Central Processor Card Installation Requirements

Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned at 0.5 in. centers minimum.
Operating Temperature:	0°C to +55°C
DC Power Requirements:	+5v +5% @ 2.2A max (1.0 A typical) -9v +5% @ 0.06A max (0.03A typical)

Table 2-4.

Signal Requirements

All signal inputs and outputs on the Central Processor Module are TTL compatible. However, the load/drive specifications for certain signals vary from standard. Observe the following specifications:

Input Signals	Maximum Load	Conditions
$\overline{\text{WAIT REQ}}$	5.5 mA @ 0.4 V	V_{CC}=5.25V
$\overline{\text{HLT INT REQ}}$	5.5 mA @ 0.4 V	V_{CC}=5.25V
$\overline{\text{DB IN}}$	5.5 mA @ 0.4 V	V_{CC}=5.25V
All other inputs	5.5 mA @ 0.4 V	V_{CC}=5.25V

Output Signals	Maximum Load	Conditions
$MAD_{0\text{-}11}$	32 mA @ 0.4 V	V_{CC}=4.75V
$MAD_{12\text{-}15}$	30 mA @ 0.4 V	V_{CC}=4.75V
STATE & CYCLE OUTPUTS	6.4 mA @ 0.4 V	V_{CC}=4.75V
All other outputs	8 mA @ 0.4 V	V_{CC}=4.75V

Pin List

This section provides all of the signals to be input and output from the imm8-82 Central Processor Card and the associated pin numbers on the imm8-82 edge connector. Any special requirements will be noted, as will the section number in which the associated signal is dealt with in detail.

Processor Module Output Connector

INPUT SIGNALS			
NAME	**PIN #**	**FUNCTION**	**NOTE**
MD10	23		
MD11	25		
MD12	29		
MD13	27	Memory data	
MD14	33	input (8 bits)	
MD15	31		
MD16	37		
MD17	35		
IN0	70		
IN1	72		
IN2	78		
IN3	76	Input port data	
IN4	84	Input (8 bits)	
IN5	79		
IN6	81		
IN7	86		
II0	69		
II1	71		
II2	77		
II3	73	Interrupt	
II4	80	instruction input	
II5	83	(8 bits)	
II6	85		
II7	87		
$\overline{\text{WAIT REQ}}$	21	Request processor WAIT mode	1
$\overline{\text{HALT INT REQ}}$	56	Request interrupt when processor enters STOPPED state	1
$\overline{\text{INT REQ}}$	42	Requests interrupt	1
$\overline{\text{HOLD REQ}}$	51	Requests HOLD Operation	1
$\overline{\text{INT JAM ENBL}}$	57	Enables INPUT JAM operation	1
POWER SUPPLIES			
+5v	99,100	Vcc	
GND	3,4	Ground	
+12v	49,50	Not used on imm8-82	
-9v	43,44	Vdd	
-12v	47,48	Not used on imm8-82	

Table 2-5.

Processor Module Output Connector

OUTPUT SIGNALS			
NAME	**PIN #**	**FUNCTION**	**NOTE**
OSC	55	Output of clock oscillator	
$\phi 1$	98	Timing signals	
$\phi 2$	89		
MAD0	11	Low-order Memory Address (8 bits)	
MAD1	12		
MAD2	13		
MAD3	14		
MAD4	15		
MAD5	16		
MAD6	17		
MAD7	18		
MAD8	19	High-order Memory Address (6 bits)	
MAD9	20		
MAD10	96		
MAD11	94		
MAD12	60		
MAD13	59		
MAD14	66	Tied to ground on basic imm8-82	2
MAD15	65		
DB0	24	Data Out Bus (8 bit)	
DB1	26		
DB2	30		
DB3	28		
DB4	34		
DB5	32		
DB6	38		
DB7	36		
$\overline{T40}$	40	Status Flag S for input instruction	
$\overline{T41}$	39	Status Flag Z for input instruction	
$\overline{T42}$	41	Status Flag P for input instruction	
$\overline{T43}$	45	Status Flag C for input instruction	
$\overline{T44}$	90	T4 State Data Out	
$\overline{T45}$	91		
$\overline{T46}$	92		
$\overline{T47}$	8		
$\overline{T1}$	10	Machine State Code Signals	
$\overline{T2}$	1		
$\overline{T3}$	63		
$\overline{T3A}$	88		

Table 2-5. (Continued)

Processor Module Output Connector

NAME	PIN #	FUNCTION	NOTE
$\overline{\text{HALT ACK}}$	62	Indicates STOPPED state	
$\overline{\text{INT ACK}}$	5	Indicates T1I state	
$\overline{\text{WAIT ACK}}$	61	Indicates WAIT state	
$\overline{\text{SYNCA}}$	7	Timing signal	
$\text{R}/\overline{\text{W}}$	95	Memory Write strobe	
$\overline{\text{HOLD ACK}}$	46	Indicates HOLD operation	
$\overline{\text{INT REQ LTH}}$	9	Indicates Interrupt received	
$\overline{\text{I/O OUT}}$	54	Indicates I/O Output cycle	
$\overline{\text{I/O IN}}$	82	Indicates I/O Input cycle	
$\overline{\text{FETCH CYCLE}}$	68	Indicates Instruction Fetch cycle	
$\overline{\text{I/O CYCLE}}$	64	Indicates Input/Output cycle	
$\overline{\text{MEM READ CYCLE}}$	67	Indicates Memory Read cycle	
$\overline{\text{MEM WRITE CYCLE}}$	58	Indicates Memory Write cycle	
$\overline{\text{INT CYCLE}}$	75	Indicates Interrupt cycle	
$\overline{\text{DB OUT}}$	2	Enables data output from CPU	
$\overline{\text{DB IN}}$	22	Enables data input to CPU	

There are also three jumpers which may be changed by the user for special applications:

(1) If a Write strobe of the opposite polarity from the standard strobe, $\text{R}/\overline{\text{W}}$ (low when Write), is desired, the jumper marked C in Figure 2-8 may be changed to the position shown in dotted lines. This will give signal $\text{R}/\overline{\text{W}}$ as a Write strobe, high when true.

(2) The edge pins corresponding to MAD14 and MAD15 are tied to ground in the basic system. They can be changed to reflect the cycle control bits CC0-CC1 by changing their respective jumpers.

Table 2-5. (Continued)

CHAPTER 3 THE imm8-60 INPUT/OUTPUT CARD

The imm8-60 Input/Output Card — General Functional Description
The imm8-60 Input/Output Card — Theory of Operation
The imm8-60 Input/Output Card — Utilization

The imm8-60 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 imm8-60 Card has been designed to allow two cards to be used in a system, with each card having a unique address by which it is referenced. The imm8-60 Card includes all logic necessary to support a multi-card implementation.

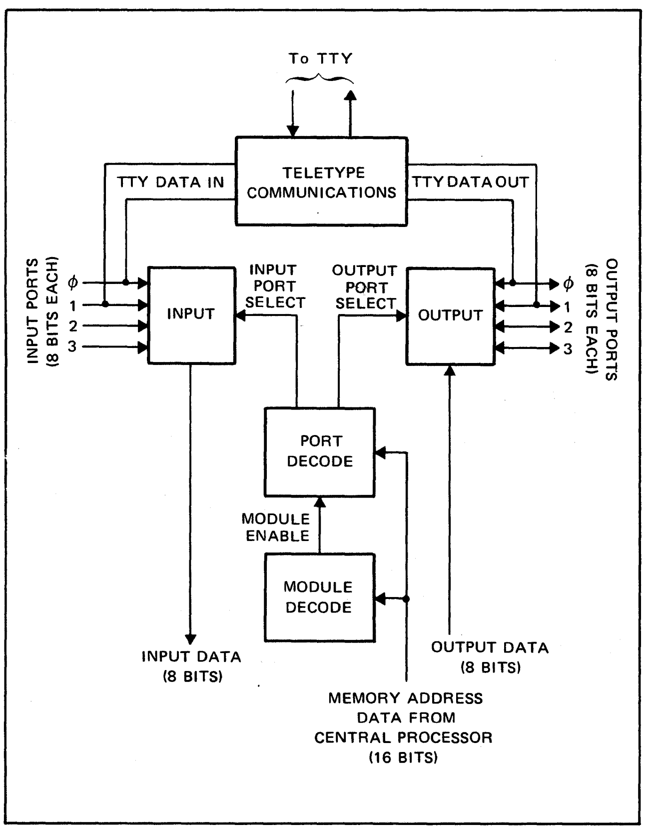


Figure 3–1. I/O Functional Block Diagram

Although the imm8-60 Card has been designed to support the Intel imm8-82 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 imm8-60 Input/Output Card at three levels; the operation of the imm8-60 is described on a basic functional level in the first section; the theory of operation is provided in the second; necessary information to effectively use the imm8-60 card is given in the third. This last section covers such areas as user-available options, signal and installation requirements, etc.

THE imm8-60 INPUT/OUTPUT CARD – GENERAL FUNCTIONAL DESCRIPTION

This section describes the operations of the imm8-60 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.

The Functional Units

In order to describe its operation, the imm8-60 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 eight 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 imm8-60 Card uses one or more of these units in its execution.

A block diagram of the imm8-60 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 imm8-60 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 imm8-82 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.

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 eight ports is being addressed by the I/O Address. 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 imm8-60 Input/Output Card 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 imm8-60 Card executes an output operation in the following steps:

1) The Central Processor sends the I/O Address to the imm8-60 Card, and a Module and Port Select operation is performed, as described earlier in the section on Module and Port Select Operations.

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 usable by the Input block, and sends the data and status signals to the Input block on input ports 0 and 1.

3) When the proper enabling signals are sent to the Input block by a Module and Port Select operation as described earlier, 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

The 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 8 to the imm8-60 Card, and a Module and Port Select operation is performed as described earlier.

2) Teletype output data is sent by the Central Pro-

cessor to the Output block via the Output Data bus.

3) The Teletype data is placed into output port 8 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 8 is sent to the Teletype Control block, which converts it into a form usable 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.

imm8-60 INPUT/OUTPUT CARD – THEORY OF OPERATION

This section describes, in detail, the theory of operation of the imm8-60 Input/Output Card. The circuit-level implementation of the features described in the General Functional Description will be given.

Due to difference between the functional description and the actual implementation of imm8-60 operations, this section's organization differs from that of the last. First the Module Select operation is described. Port Selection operations are discussed in the second and third sections which deal with input and output operations, respectively. The fourth deals with all Teletype communications, which utilize the Teletype communications circuits as an external device, but otherwise are the same as non-Teletype Input/Output operations.

Module Selection

If two imm8-60 Cards are 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 imm8-60 Card edge pins; the module address is complemented by a series of inverting latches and the complemented address is present at additional imm8-60 Card edge pins. The user selects an address for each imm8-60 Card, and implements the address by selecting a set of Address and Complemented Address signals; select signals are externally jumpered to the Module Selection circuits, which combine the incoming signals through a NAND gate to provide the enabling signal which is sent to other circuitry on the card.

Note that there is a wide choice of signals with which to select Address information. RR0 and RR1 (as generated by the Intel imm8-82 Central Processor Card) may be used,

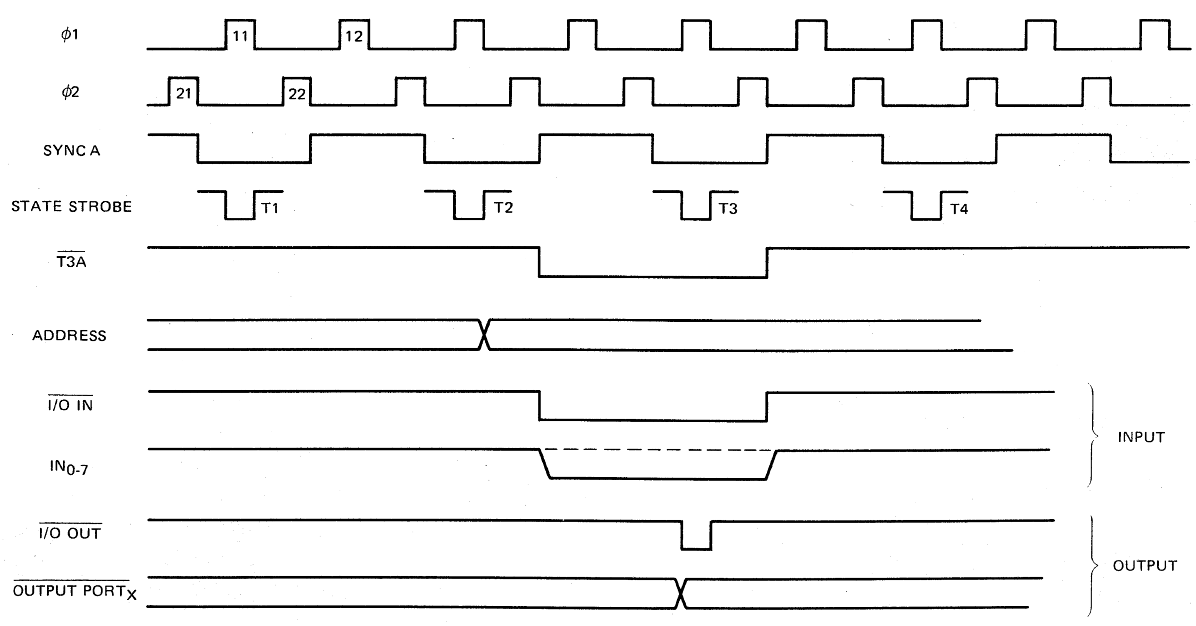


Figure 3–2. I/O Module Timing

or alternatively, signals DA10-DA15 may be used. In either case, the function of the Module Decoding Circuits remains the same. For more details on the options available to the user, see the section on Utilization later in this chapter.

To illustrate use of the Module Decoding Circuits, consider an application in which the imm8-60 Card has been given the arbitrary designation of Module L5. If it is desired to select this module for an operation, the Central Processor would send line DA11 TRUE, DA12 FALSE, DA13 TRUE, DA14 FALSE, and DA15 FALSE (binary 00101). These signals would be complemented by the inverting latches to produce $\overline{\text{DA11}}$ FALSE, $\overline{\text{DA12}}$ TRUE, $\overline{\text{DA13}}$ FALSE, $\overline{\text{DA14}}$ TRUE, and $\overline{\text{DA15}}$ TRUE. The DA11, DA12, DA13, DA14, and DA15 signals would be tied to DS11, DS12, DS13, DS14, and DS15, respectively, causing the enabling signal to go TRUE. Any other combination of signals could have been selected, limited only by the requirement that each card must have a unique address in order to prevent simultaneous addressing of more than one card.

Input Operations

Input operations on the imm8-60 Input/Output Card are handled with the Input Circuits. These are shown on the left in the I/O Module Schematic, Figure 3-3.

The first step in an input operation is the transmission of an I/O Address to the imm8-60 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 as discussed in the last section, 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, along with signal M2S.

When it is enabled by the Module Enable signal, and the I/O IN signal sent by the Central Processor, and signal M2S, 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 imm8-60 Card on lines MAD9 and MAD10.

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. Input timing is shown in Figure 3-2.

Output Operations

Output operations on the imm8-60 Input/Output Card are handled by the Output Circuits, shown on the right in Figure 3-3.

An Output operation begins with the transmission of an I/O Address to the imm8-60 Card from the Central Processor. This I/O Address contains Module and Port Selection information which is used to determine which input or output port is to be used for a particular operation.

The Module Selection information is processed by the Module Select Circuits as discussed earlier, and cause the Module Enable signal to be produced. This signal is led to the Output Decoder chip, where it is used, along with signal M2S, as an enabling signal to that chip.

The Central Processor then sends the data which are to be output to the imm8-60 Card, lines MAD0 - MAD7. Along with the output data is sent the I/O OUT signal, which is led to the Output Decoder and is used as a third enabling signal.

When the Output Decoder is enabled by the three enabling signals Module Enable, M2S, and 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 imm8-60 Card on lines MAD9 and MAD10.

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.

Teletype Communications

Teletype communications can be handled directly by the imm8-60 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 3-3.

Teletype Communications on the imm8-60 Card are handled through Input Ports 0 and 1 and Output Ports 8 and 9. Input Port 0 handles Teletype data which are to be input to the Central Processor; Input Port 1 handles Teletype status information. Output Port 8 holds the data which are output from the Central Processor to the Teletype, and Output Port 9 holds the control data used to control Teletype communications. All Teletype input and output operations are handled by the imm8-60 Card as normal 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 imm8-60 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 imm8-60 Card. It also translates the eight-bit data output by the imm8-60 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. The clock is provided on the imm8-60 Card by a crystal clock generator which provides a 4.9562 MHz signal. This signal is used to clock a series of two cynchronous counters, each of which provides a "divide-by-sixteen" function,

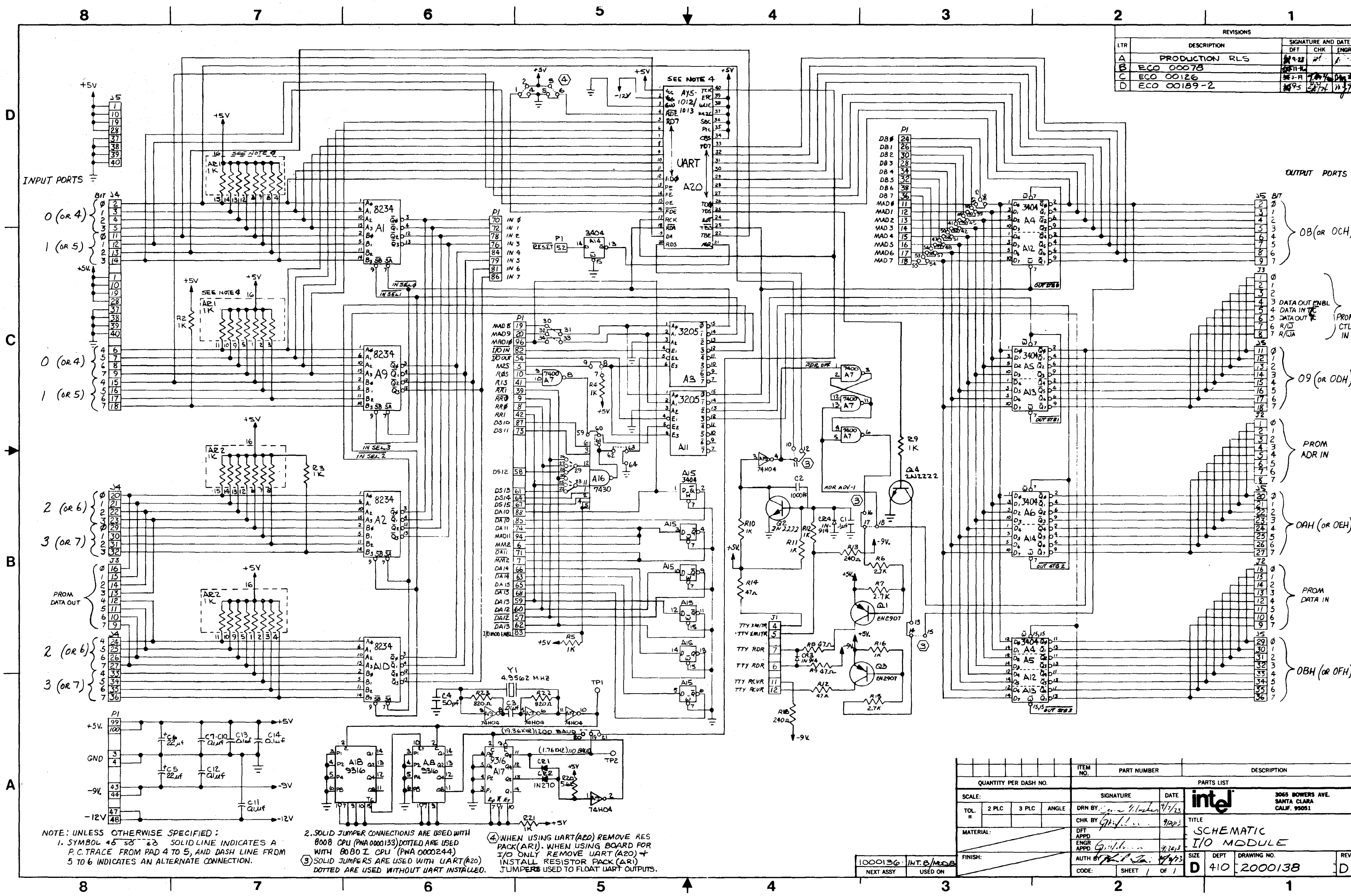


Figure 3–3. I/O Module Schematic Diagram

thus producing a 19.36KHz signal. This signal can be used directly, providing a 1200 baud transmission rate 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" capability, and will provide a 1.76KHz signal which, when used as the UART clock, will provide a 100 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 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 imm8-60 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 in the section on Output Operations 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 8 via an output operation. The data which are to be sent to the Teletype are latched into Output Port 8 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. 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 imm8-60 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 imm8-60 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.

imm8-60 INPUT/OUTPUT CARD – UTILIZATION

This section describes the options available to the user of the imm8-60 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 imm8-60 Card, including the choice of Module Address, the choice of which lines to use as address input lines, which signals to use as the Module Enable signals, which lines to use as the Data Output lines,

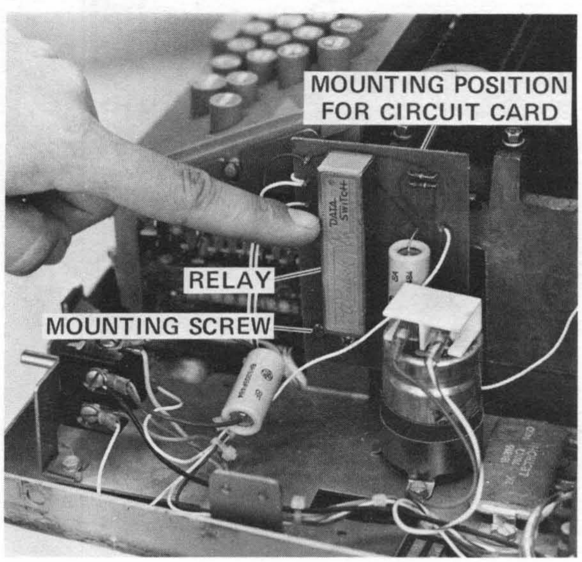


Figure 3–4. Relay Circuit (Alternate)

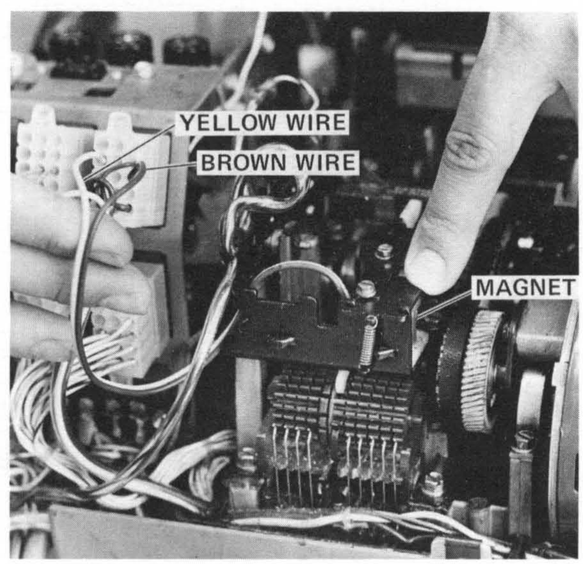


Figure 3–5. Distributor Trip Magnet

and even whether or not to use the Teletype Communications Circuits. Each of these will be discussed in the next section.

User-Available Options

The user has the choice of Module Coding, with the RR0 and RR1 lines for use as the Module Select lines, discussed earlier. When the imm8-60 Input/Output Card is used as a peripheral device with Intel's imm8-82 Central Processor Card, the RR0 and RR1 lines are used, and the standard module coding is:

Module Number	Module Select	Pins Jumped	
I/O 0	Input: $\overline{MM2}$	5-7, 10-8, 41-39	P1
	Output: $\overline{MM2}$,RR0, $\overline{RR1}$		
I/O 1	Input: MM2	5-6, 10-8, 41-39	
	Output: MM2,RR0, $\overline{RR1}$		

If the DA lines were used as the Module Select lines, similar arrangements would be used.

When the RR lines are used as the Module Select lines, the enable line which is generated by the DA lines should be tied to ground, in order to permanently enable it. Similarly, the output generated by the RR lines should be tied to the enable signal generated by the DA lines if the DA lines are to be used. In addition, signal M2S may be permanently enabled, if it is desired. Each of these options is enabled by the positioning of on-card jumpers, as shown in the Schematic Diagram, Figure 3-3.

The second choice which is available to the user is that of the lines to be used as Port Enable lines. This choice is also determined by the positioning of on-card jumpers. If lines MAD9 and MAD10 are to be used, as would be the case if the imm8-82 Central Processor Card were used, the jumpers would be positioned as follows:

31-32, 33-34

Likewise, if it were desired to use lines MAD8 and MAD9, the jumpers would be positioned as follows:

30-31, 32-33

The third option available to the user is the choice of the lines which are to carry the data from the Central Processor to the Output Ports. The user has the choice of using either lines MAD0-7 or lines DB0-7. Again, the choice is implemented by properly positioning jumpers on the card itself. The imm8-82 Central Processor Card uses lines MAD 0-7 for purposes of output.

If it is desired, the imm8-60 Input/Output Card's internal Teletype Communications Circuits may be disabled by removing the UART chip. If this is done, pull-up resistors 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 RDE and FDE to +5V.

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

Output: 14-15 instead of 13-14
Input: 10-11 instead of 11-12

When the Input/Output Module is used for Teletype operations, the user must ensure that no device other than the Teletype is connected to Input Ports 0 and 1 or Output Ports 8 and 9.

The imm8-60 Card has been designed to optionally interface with the Intel imm8-76 PROM Programmer Card. This card uses Input Port 2 for a PROM Data Out port, and

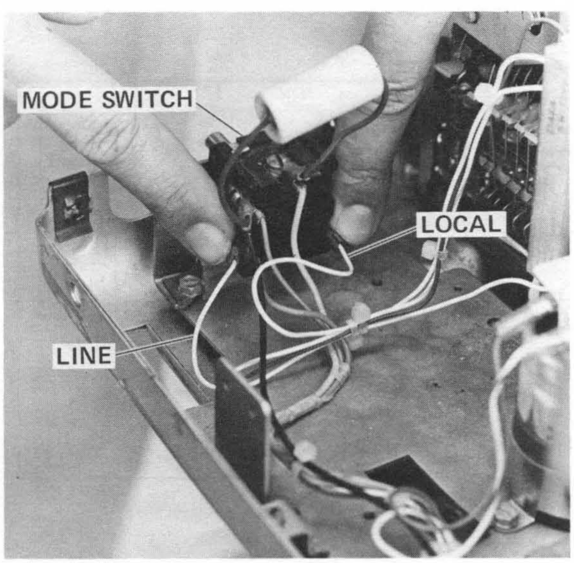


Figure 3–6. Mode Switch

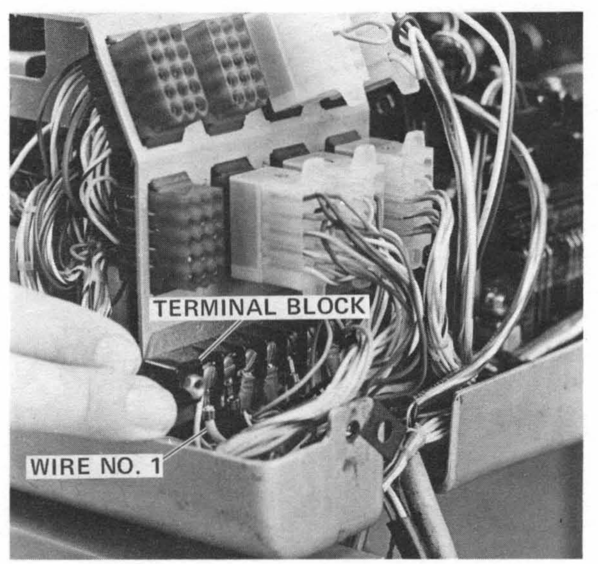


Figure 3–7. Terminal Block

Output Ports 9, 10, 11 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°-55°C
DC Power Requirements:	+5v ± 5%, .820A Max
	-9v ± 5%, .030A Max
	-12v ± 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).
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 Figures 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 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 is added by cutting the brown wire, from the distributor trip magnet, at its connector plug and then splicing it to wire "A." (See Figures 3-5 and 3-9). 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).

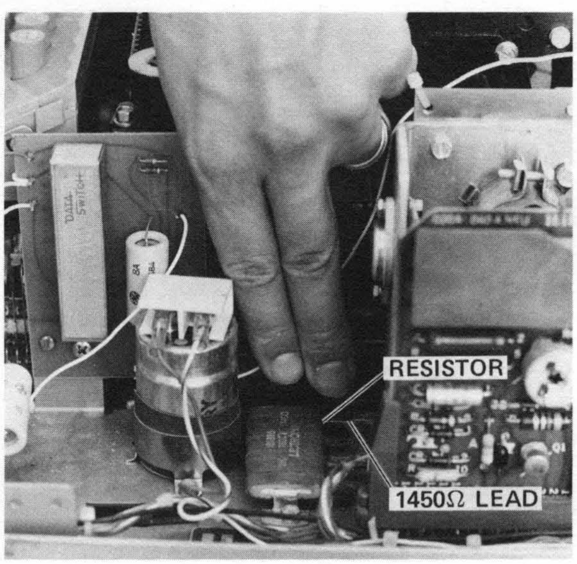


Figure 3–8. Current Source Resistor

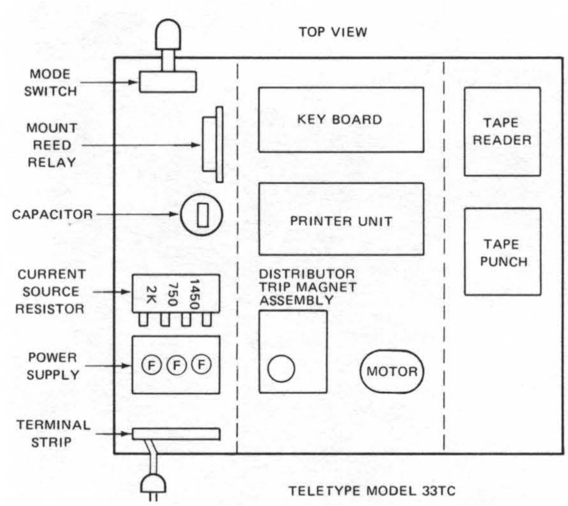


Figure 3–10. Teletype Layout

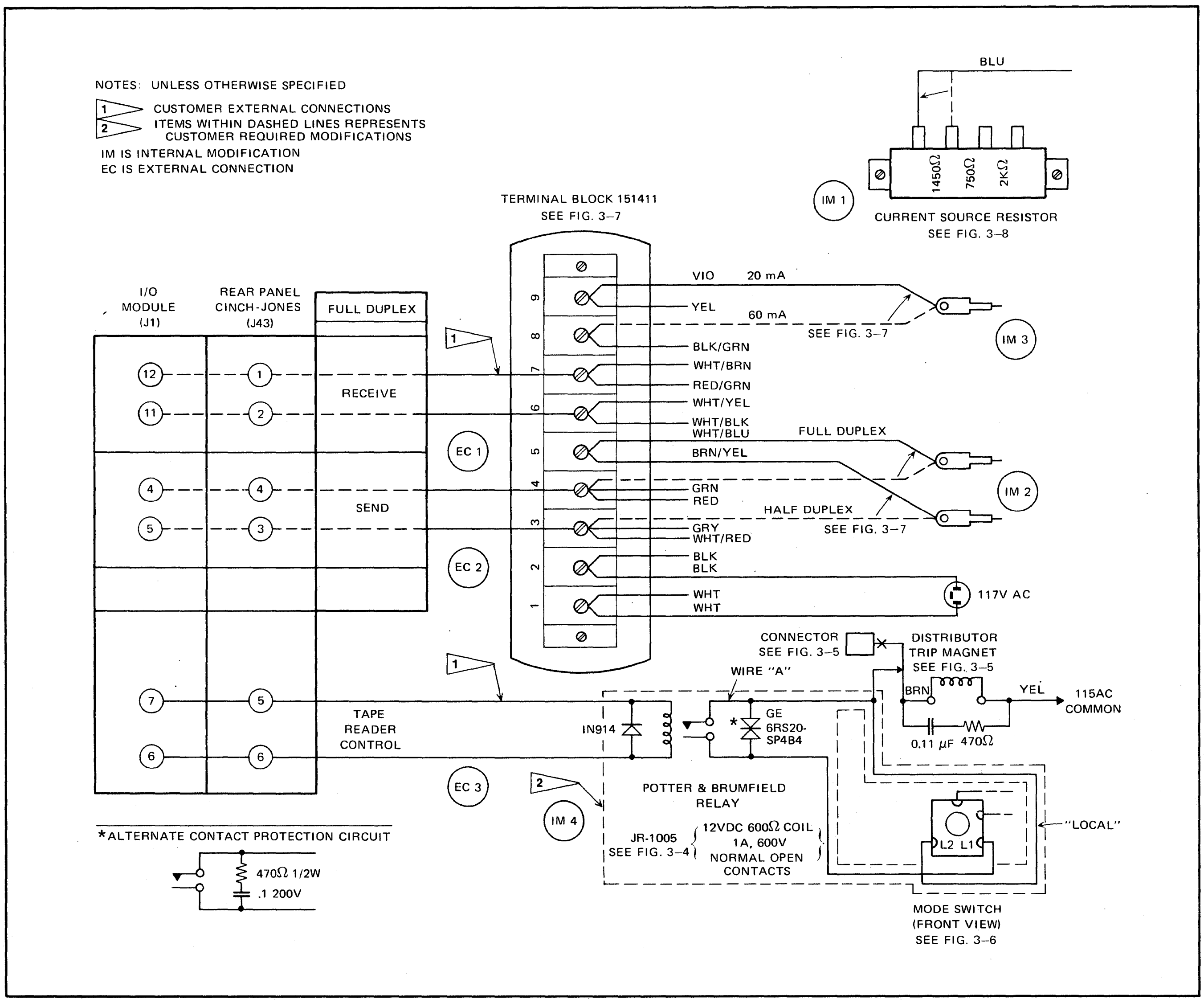


Figure 3–9. TTY Modification

CHAPTER 4 THE imm8-62 OUTPUT CARD

General Functional Description
Detailed Functional Theory
Card Utilization

The imm8-62 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 imm8-82) 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.

A superficial functional description of the imm8-62 Output Card logic is provided in the first section. A more detailed functional theory is given in the second; specific instructions describing the use of the imm8-62 Card are given in the third.

GENERAL FUNCTIONAL DESCRIPTION

The imm8-62 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.

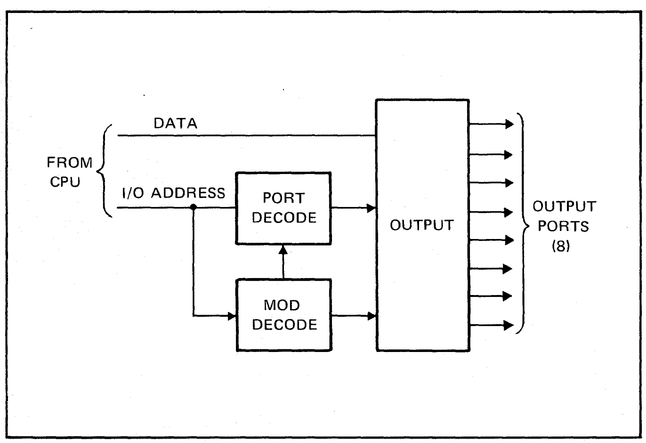


Figure 4–1. Output Module Functional Block Diagram

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.

The Central Processor then sends the data which is to be output to the imm8-62 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 imm8-62 Card. Actual circuit-level implementation of the features described as functional blocks in the previous section are given.

The first section deals with Module Decoding, the second with Port Decoding, and the third describes an actual output operation.

Module Decoding

If it is desired to use more than one imm8-62 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 4-3.

As shown in Figure 4–3, the Module Address information is brought to the imm8-62 Card edge pins and is led to a series of inverting latches. These latches invert the incoming information and supply it, in turn, to another set of card edge pins. The user then selects the proper set of Address and inverted Address signals, and uses external wire jumpers to tie this information to the Module Selection circuits which combine the incoming signals to provide either the MOD SEL signal or the $\overline{\text{OUT MOD SEL}}$ signal. These signals are then used to enable operations of the Output Card.

Note that the user has a wide range of choices regarding which signals to use as the Address information. Signals RR0 and RR1, as generated by the Intel imm8-62 Central Processor Card, may be used, or, as an alternative, signals DA10–DA15 may be used. In either case, the function of the Module Decoding circuits remains the same. (See the section on CPU in Chapter 2 for more details on the options available to the user.)

As an example of the functioning of the Module Decoding Circuits, consider an application in which the imm8-62 Card has been given the arbitrary designation of Module 5. If it was desired to select this module for an operation, the Central Processor would send an address corresponding to the module's designation, such as, perhaps, line DA11 TRUE, DA12 FALSE, DA13 TRUE, DA14 FALSE, DA15 FALSE (binary 00101). These signals would be input to the imm8-62 Card and inverted, producing $\overline{\text{DA11}}$ FALSE, $\overline{\text{DA12}}$ TRUE, $\overline{\text{DA13}}$ FALSE, $\overline{\text{DA14}}$ TRUE, $\overline{\text{DA15}}$ TRUE. The DA10, DA11, $\overline{\text{DA12}}$, DA13, and $\overline{\text{DA14}}$ signals would be tied to DS11, DS12, DS13, DS14, and DS15, respectively, causing the $\overline{\text{OUT MOD SEL}}$ signal to go LOW, enabling operations. Any other combination of signals could have been selected, limited only by the requirement that each card must have a unique address to prevent simultaneous addressing of more than one card.

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 imm8-62 Card. This function is performed by the Port Selection circuits, shown in detail in Figure 4-3.

In order to select one of the eight output ports, three data lines are led to the Port Decoder. When enabled by the MOD SEL or $\overline{\text{OUT MOD SEL}}$ signals, 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 4-3, the Schematic Diagram):

1) The Central Processor sends an I/O Address to the imm8-62 Module on lines MAD9-13.
2) The Module Decoding and Port Decoding circuits decode the incoming I/O Address.
3) The Central Processor sends the data which are to be output to the imm8-62 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 imm8-62 Card are latched into the proper output port by signal $\overline{\text{OUT STB}}$, 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 4-2.

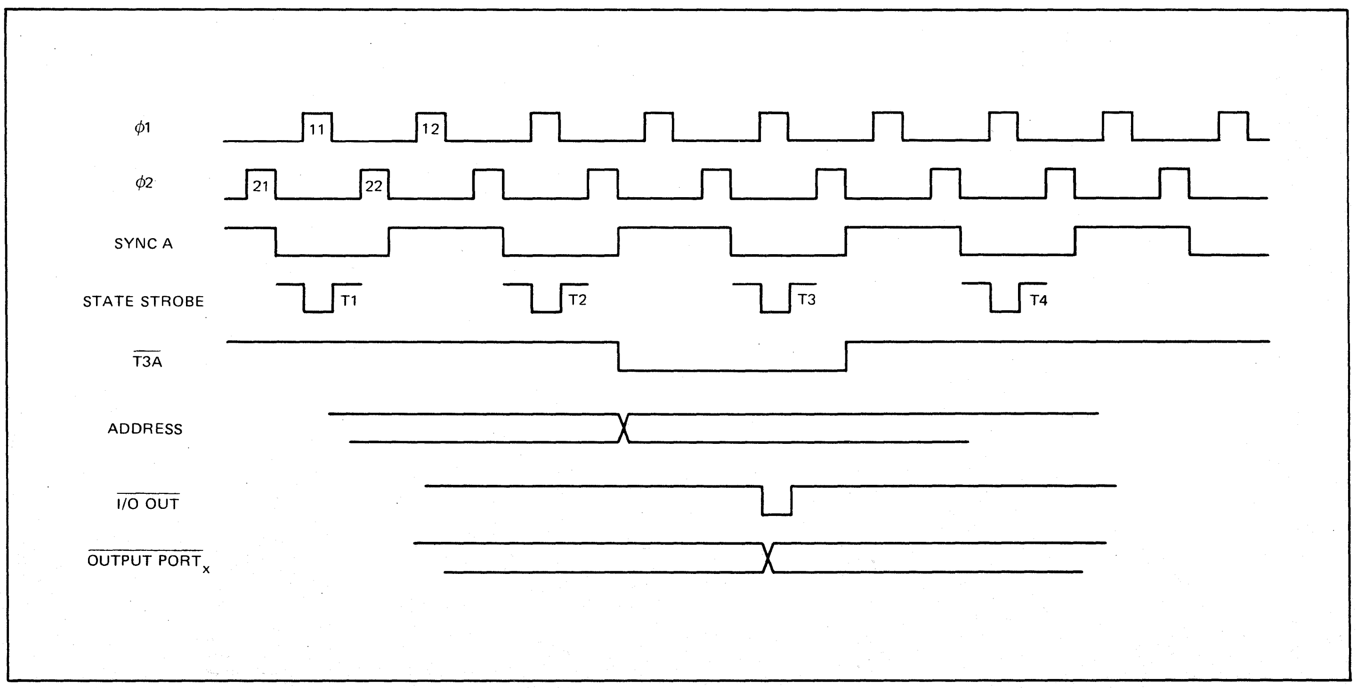


Figure 4—2. Output Module Timing

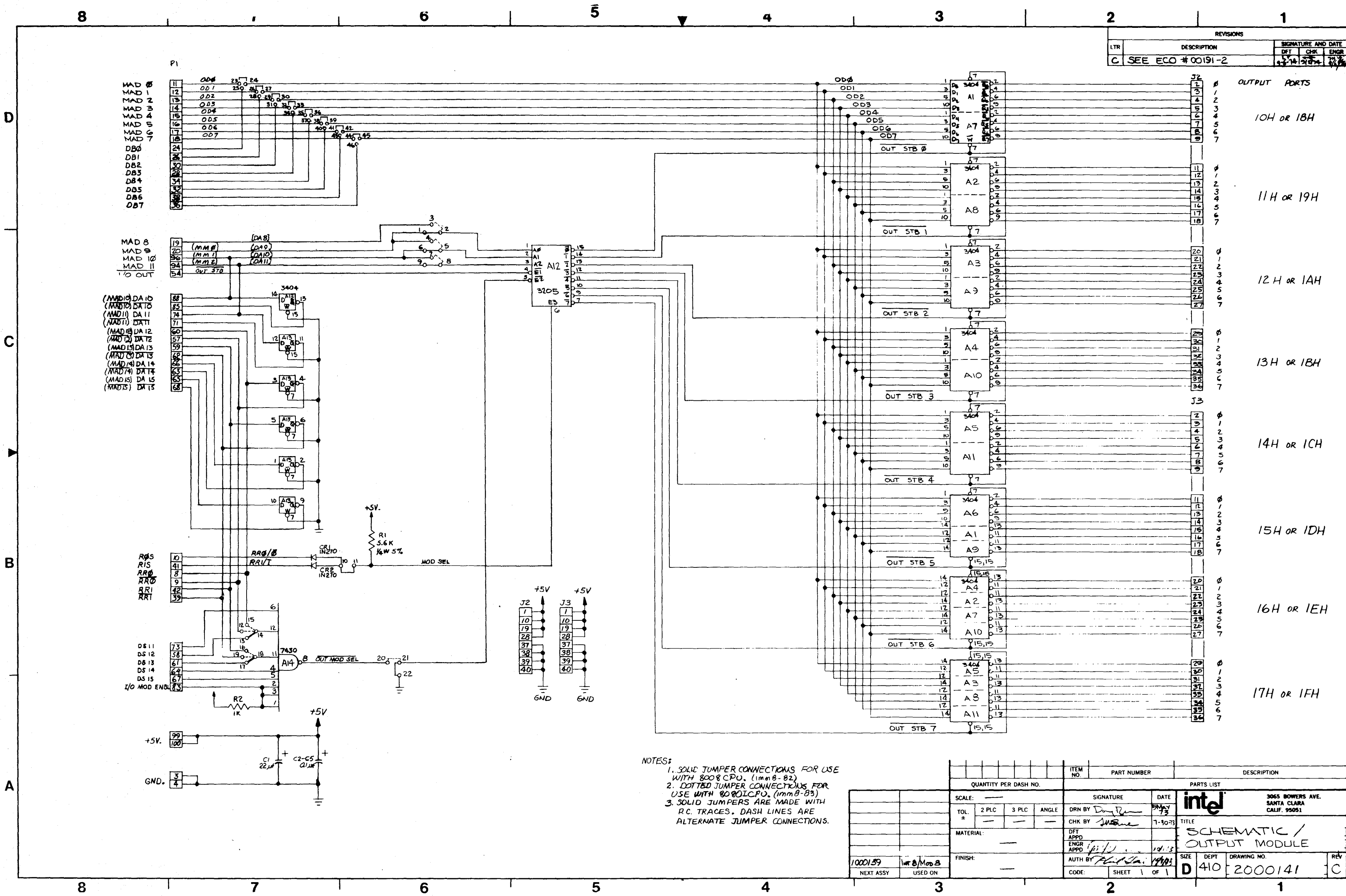


Figure 4–3. Output Module Schematic Diagram

CARD UTILIZATION

There are several options available to the user of the imm8-62 Card. Among these are the choice of the Module Address, the choice of which lines to use as address input lines, which signals to use as Card Enabling signals, and which lines to use as data lines holding the data to be output. This section will cover the options available to the user, and also supplies a complete list of the imm8-62 Card edge pins and their associated signals.

User Options

The user has a wide range of options available on the imm8-62 Output Card. Each option is implemented by means of jumpers, either mounted on the card itself or external, between card edge pins.

The first option is the choice of Module Coding. The user has his choice of either the DA11-DA15 lines or the RR0-RR1 lines for use as the Module Select lines. When the imm8-62 Output Card is used as a peripheral device with the Intel imm8-82 Central Processor Card, the RR0 and RR1 lines are used, and the standard module coding is:

Module Number	Module Select	Pins Jumped
OUT 2	$\overline{\text{RR0}}$, RR1	5-6, 10-9, 41-42
OUT 3	RR0, RR1	5-7, 10-8, 41-42

If the DA lines were used as the Module Select lines, similar arrangements would be used.

When the RR lines are used as the Module Select lines, signal $\overline{\text{OUT MOD SEL}}$ must be tied to GROUND in order to permanently enable it, and, similarly, signal MOD SEL must be tied to +5v. when the DA lines are used.

The second choice available to the user is the choice of the three lines used as Port Select lines. These lines are determined by the positioning of jumpers mounted on the imm8-62 Card itself. If it is desired to use lines MAD9-11, as would be desired when using the imm8-82 Central Processor Card, the jumpers would be positioned as follows:

1-2, 5-6, 8-9

Likewise, if it were desired to use lines MAD8-10, the jumpers would be positioned as follows:

2-3, 4-5, 7-8

The third user option enabled on the imm8-62 Card is the choice of the lines which are to carry the data from the CPU to the Output Card. The user has his choice of lines MAD0-7, or lines DB0-7. If the MAD lines are to be used, the jumpers on the imm8-62 Card would be positioned as follows:

23-24, 26-27, 29-30, 32-33, 35-36, 38-39,
41-42, and 44-45

Similarly, if the DB lines are to be used, the jumpers are positioned as follows:

24-25, 27-28, 30-31, 33-34, 36-37, 39-40,
42-43, and 45-46

Again, if the imm8-62 Output Card is used as peripheral device with the imm8-82 Central Processor Card, lines MAD0-7 would be used.

CHAPTER 5 THE imm6-28 RANDOM ACCESS MEMORY CARD

The imm6-28 Random Access Memory Card — General Functional Description
The imm6-28 Random Access Memory Card — Theory Of Operation
The imm6-28 Random Access Memory Card — Utilization

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 imm8-82 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 imm8-82 Central Processor Card, it can be used in any other system which requires 4K x 8 bits of RAM storage.

This section describes the operation and implementation of the imm6-28 card on three levels: first, the operation of the card is described at a basic functional level. The theory of operation is provided in the second section. Necessary information to effectively use the imm6-28 card is given in the third.

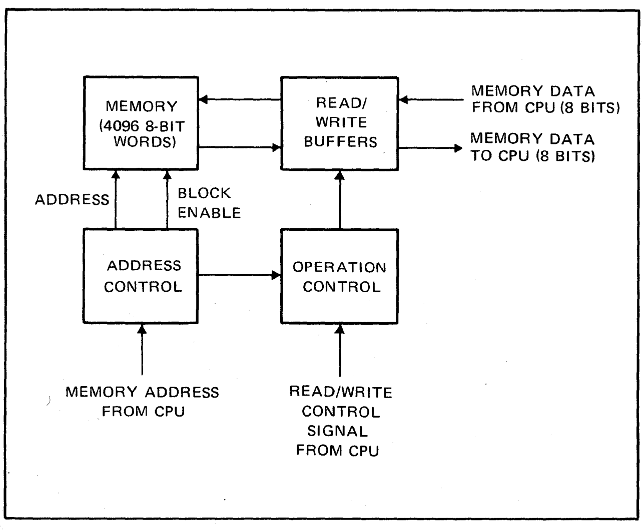


Figure 5–1. RAM Module Functional Block Diagram

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

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 a binary number and which are sent to the imm6-28 card by the Central Processor. Once the Memory Address is received by the imm6-28 card, however, it must be decoded in order 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, as described in the last section.
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, as discussed in the section on Memory Addressing Operations.
3) The Central Processor sends control signals 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 Centrol 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 discussed previously. It is divided into four subsections. The first describes the physical implementation of the imm6-28 memory. The second describes the Address Decoding operation, as this operation is common to both Memory Read and Memory Write operations. The third and fourth describe Memory Read and Memory Write operations, respectively.

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 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 $\overline{\text{MAD12–MAD15}}$, 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 blocks. These two signals are fed to Address 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 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, as described in the last section.

The Central Processor also sends signal $\overline{\text{Write}}$/Read to the imm6-28 card. In its FALSE state, this signal indicates a Write operation, therefore, during a Read operation, it will be TRUE. 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 earlier. 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 data has been completely latched into the Input Latches before attempting to write it into memory. When the delayed Write Enable signal becomes TRUE, the data will be written into the selected memory location.

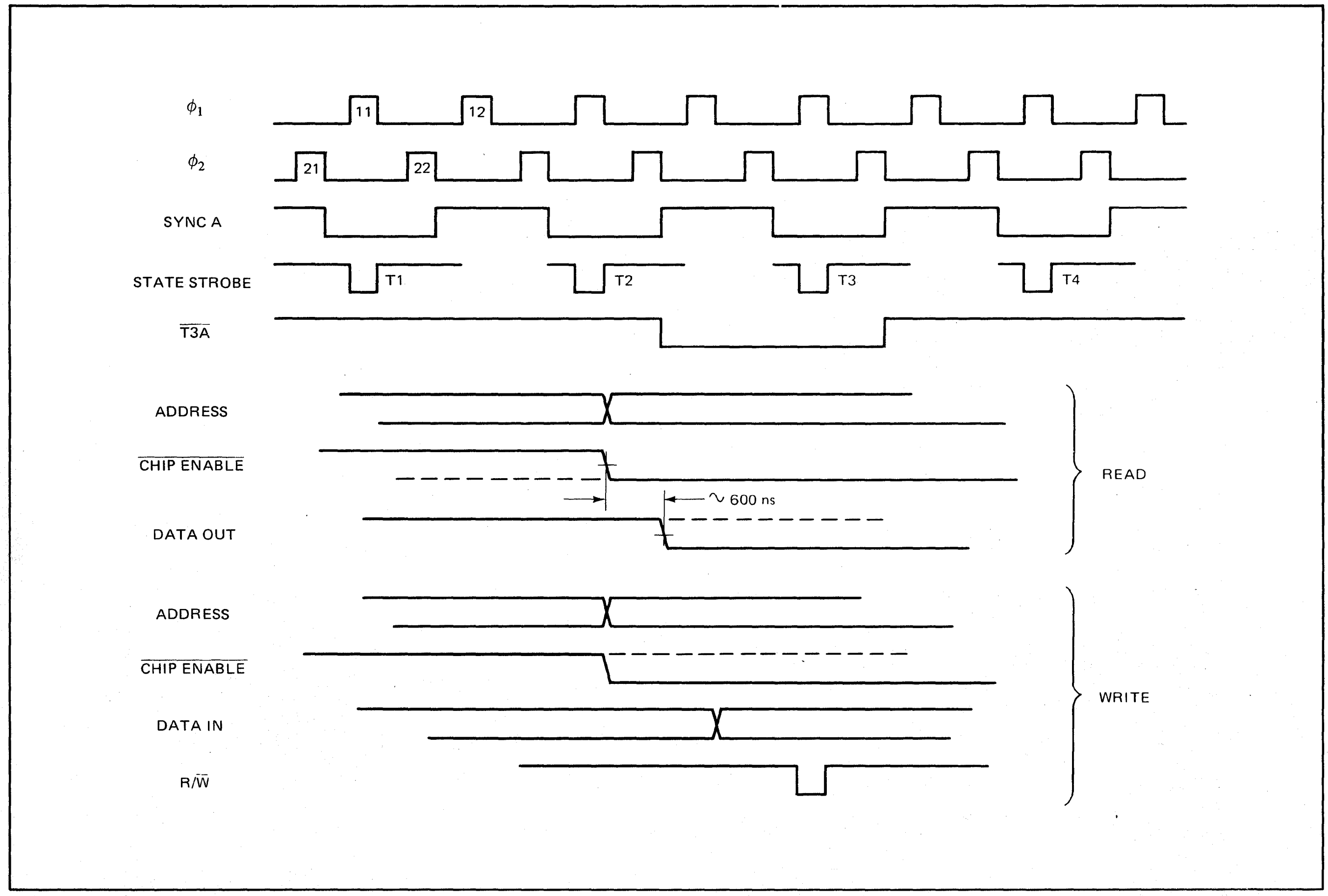


Figure 5-2. RAM Memory Module Timing

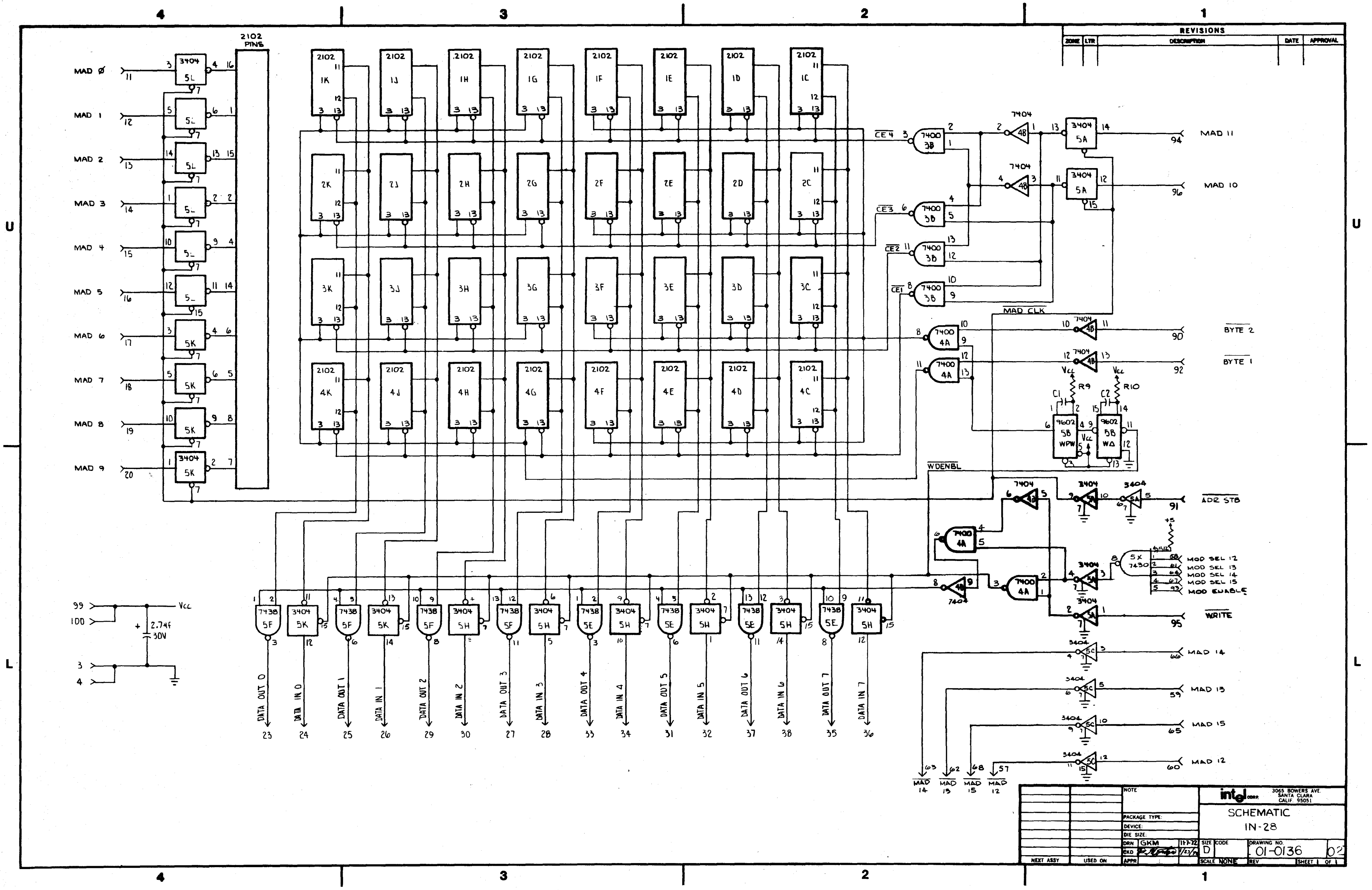


Figure 5-3. RAM Memory Module Schematic Diagram

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 imm 8-82 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, 63-64, 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

Installation Data and Requirements

Connector:	Dual 50-pin, .125 in. centers
Input Voltage:	+5v ± 5% @ 2.5A. Max
Operating Temperature:	0°C - 55°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 1702A 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 imm8-82 Central Processor Card can address up to 16,384 words of memory on four separate imm6-26 cards.

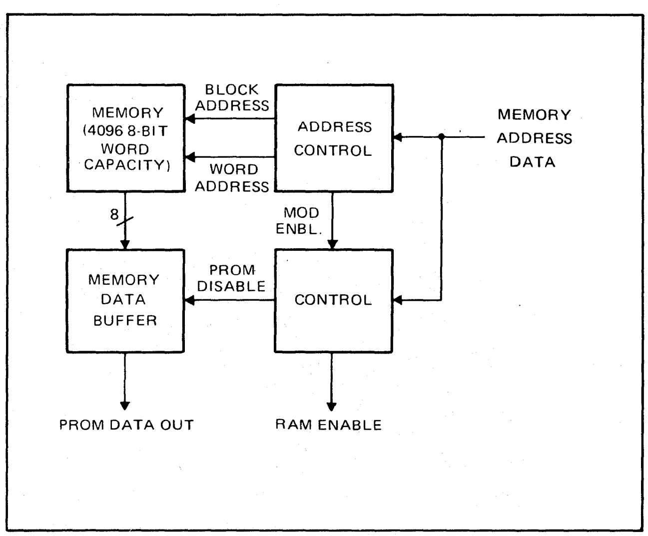


Figure 6-1. PROM Memory Module Functional Block Diagram

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

THE imm6-26 PROGRAMMABLE READ-ONLY 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 two subsections. The first describes the four functional units which enable all of the operations performed by the card; the second describes a Memory Read operation.

The Four Functional Units

In order to describe its operation, the imm6-26 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 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 on 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 described in the last section. It is divided into four subsections. The first describes the physical implementation of the imm6-26 memory. The second describes the Address Decoding operation. The third describes the Memory Read operation, and the fourth describes the Random Access Enable operation.

Physical Memory Implementation

The actual memory of the imm6-26 Card is made up to sixteen Intel 1702A 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.

Since there are sixteen 256 word PROMs on each imm6-26 card, each card has a total capacity of 4,096 words. 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-MAD 15 onto the imm6-26 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 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, as described in the last section.

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 6-2. 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 Data 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 as described earlier, 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. It is divided into four subsections. The first describes the Memory Address Coding for the imm6-26 card. The second de-

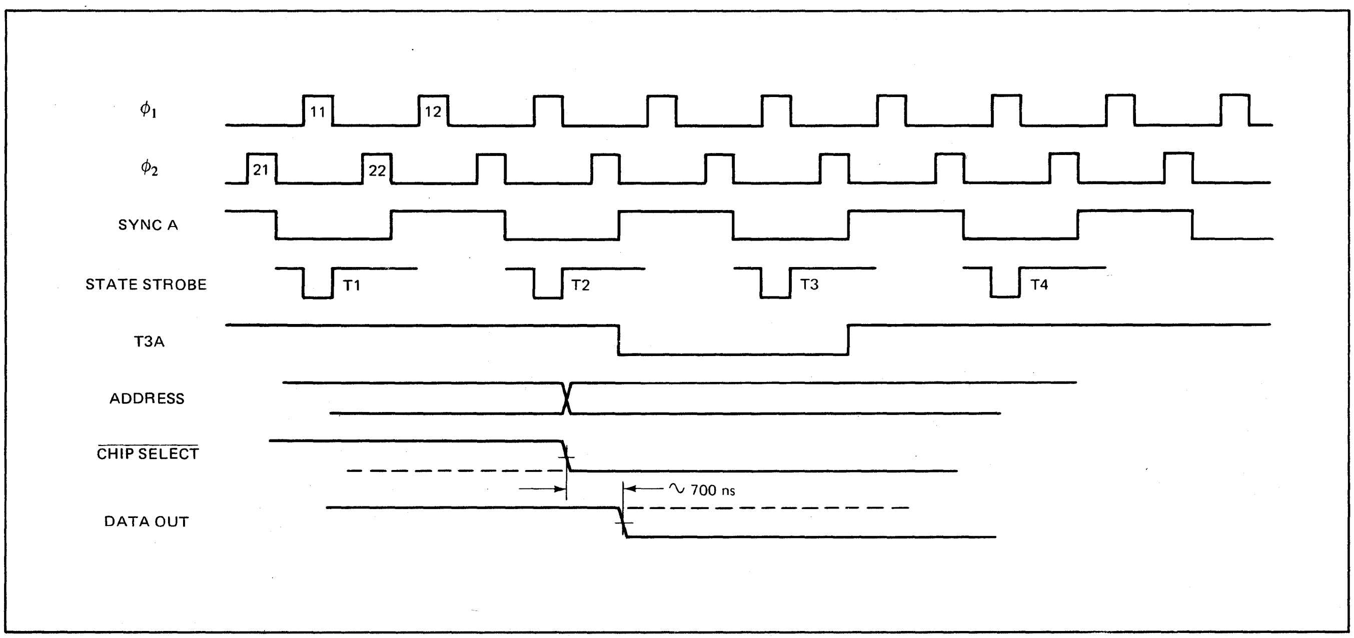


Figure 6-2. PROM Memory Module Timing

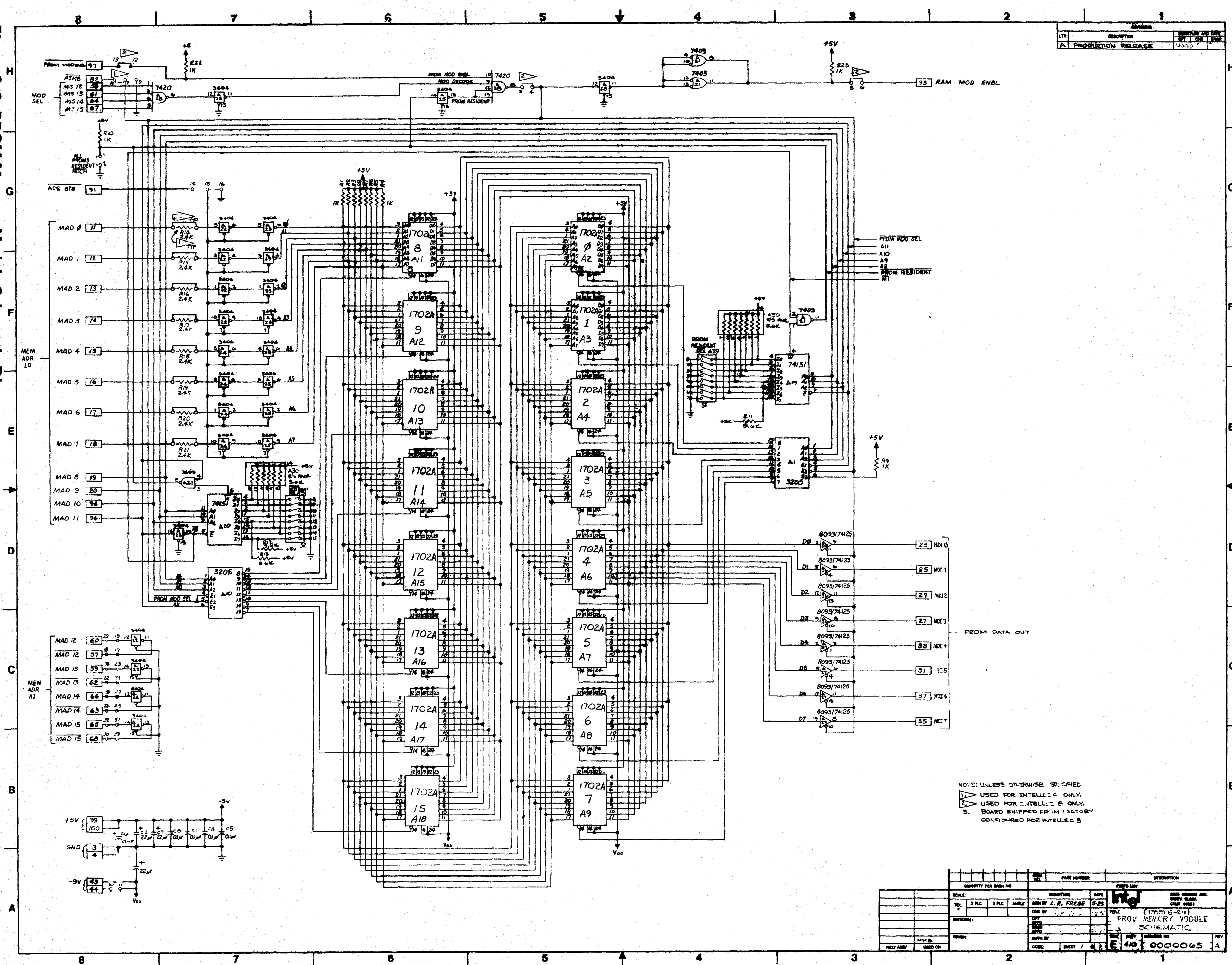


Figure 6-3. PROM Memory Module Schematic Diagram

scribes PROM installation, removal, programming, and erasure. The third given installtion data and requirements.

Memory Address Coding

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

Module No.	Memory Address	Card Select Coding	Jumper Pin Connections
PROM 0	0 - 4095	$\overline{MAD12}$, $\overline{MAD13}$, $\overline{MAD14}$, $\overline{MAD15}$	57-58, 61-62, 63-64, 67-68
PROM 1	4096 -8191	MAD12, $\overline{MAD13}$, $\overline{MAD14}$, $\overline{MAD15}$	58-60, 61-62, 63-64, 67-68
PROM 2	8192 - 12287	$\overline{MAD12}$, MAD13, $\overline{MAD14}$, $\overline{MAD15}$	57-58, 59-61, 63-64, 67-68
PROM 3	12288 -16383	MAD12, MAD13, $\overline{MAD14}$, $\overline{MAD15}$	58-60, 59-61, 63-64, 67-68

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 PROMS RESIDENT patch between points 1 and 2, as shown in Figure 6-3.

The Intel 1702A PROMs used by the imm6-26 card may be programmed by using the imm8-76 PROM Programmer card in conjunction with the Intellec 8 system, or by using an Intel PROM Programmer. They may be erased by exposing them to high intensity short-wave ultraviolet light at a wavelength of 2537 Å. After ten minutes of such exposure, the PROM will be erased to all zeros. No more exposure than is necessary should be used, to avoid damaging the PROM. (See the Intel Memory Design Handbook for more information regarding 1702A 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.

Installation Data and Requirements

Connector:	Dual 50-pin, .125 in. centers
Input Voltage:	+5V ± 5% @ 1.6A (max) -9V ± 5% @ 0.96A (max)
Operating Temperature:	0°C -55°C

CHAPTER 7 THE INTELLEC 8/MOD 8 CONTROL CONSOLE

The Intellec 8 Control Console — Functional Description
The Intellec 8 Front Panel Central Console — Theory Of Operation

The INTELLEC 8/MOD 8 Control Console is designed to provide a user of the INTELLEC 8/MOD 8 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 /MOD 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 8 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 8 Control Console has been designed to support the imm8-82 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 8 Operator's Manual.

THE INTELLEC 8/MOD 8 CONTROL CONSOLE – FUNCTIONAL DESCRIPTION

This section provides a basic, functional overview of INTELLEC 8/MOD 8 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 8008 machine cycle.

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 an input or output operation is performed from the Control Console rather than from 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 operations.
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 section.

Data Display Operations

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

- Status Display
- Cycle Display
- Address Display
- Instruction/Data Display
- Register/Flag 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 INTELLEC 8/MOD 8 at present has no Interrupt Disable capability, so this display is reserved for future expansion. The other seven 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.

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 Stack display is not used by the present INTELLEC 8/MOD 8, and is reserved for future expansion. The other seven 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.

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. On the present INTELLEC 8/MOD 8 System, however, only the first fourteen of these are used.

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 Register/Flag display function provides a visual indication of the contents of the processor Register/Flag latch.

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 processor enters a WAIT state when the console WAIT switch is depressed, giving control of the memory address and control busses to the Control Console.
2) The Mem Access switch on the Control Console is depressed, sending a control signal to the processor.
3) The memory addressed to be accessed is loaded into the Address/Instruction/Data switches on the Control Console.
4) The LOAD switch on the Control Console is depressed, loading the Address/Instruction/Data data into the Address Register.
5) The address held in the Address Register is sent to the memory module on the memory address bus.
6) 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 as discussed in Section 8.1.1.

7) If it is desired to write data into memory, the 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.

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.

A special form of memory access is the Deposit at Halt function. When this function is performed, the Control Console waits until the processor enters a STOPPED state, and then causes the data held in the Address/Instruction/Data switches to be written into the memory location addressed by the contents of the Control Console Address Register.

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 processor enters a WAIT state when the console WAIT switch is depressed, giving control of the Memory Address and Control busses to the Control Console.
2) The I/O Access switch on the Control Console is depressed, sending a control signal to the processor.
3) The I/O Address signifying the I/O device to be used for the manual I/O access operation is loaded into Address/Instruction/Data switches 8-14 on the Control Console.
4) 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.
5) The DEP switch is depressed.
6) The I/O Address and data are sent to the Input/Output and Output modules, which then perform the designated input or output operation.
7) In the case of an Input operation, the data from the selected input port is displayed in the data display light, as discussed earlier.

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 interrupt instruction can be such that processor operation is directed to a routine which will service the device originating the interrupt.

In the case of the Control Console, an interrupt is generally executed in order to start INTELLEC 8/MOD 8 operations, as the CPU requires an interrupt in order to exit from a STOPPED state. 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 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.

A RESET operation is a special case of Interrupt operation, in which a hardwired instruction is presented to the CPU instead of the contents of the Address/Instruction/Data switches. This instruction is a RESTART to zero instruction, causing program execution to begin at memory location 0. A RESET operation also generates a RESET signal which may be used to initialize peripheral devices attached to the INTELLEC 8/MOD 8 System.

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/Instruction/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 1 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 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, as discussed earlier.

Processor Control Operations

The Processor Control operations allow the user to control the operation of the INTELLEC 8/MOD 8 from the Control functions:

1) Sense
2) Search/Wait
3) Deposit
4) Deposit at Halt
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/MOD 8 FRONT PANEL CENTRAL CONSOLE-THEORY OF OPERATION

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

The INTELLEC 8/MOD 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 Control 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

As stated earlier in this chapter, there are five distinct data display operations:

- Status display
- Cycle display
- Address display
- Instruction/Data display
- Register/Flag 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 function are as follows:

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

The Interrupt Disable display is not used in the present Intellec 8 system.

The other seven 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 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 WAIT ACK signal to produce the WAIT STATUS DISP signal, which will go LOW when the processor is in the WAIT state.

3) The HALT status display is lit when the Central Processor is in the STOPPED state. This is accomplished by using the HALT ACK signal to produce the 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 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 let 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.

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 Stack display is not used on the present Intellec 8 system. The other seven displays are produced 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 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 combination 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 the Control Console memory or I/O access operations. 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 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.

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/Data/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-MAD13.

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 reflects the contents of the Processor Register/Flag flip-flops.

Manual Memory Access Operations

Manual Memory Access operations are executed in the following manner, after the WAIT switch is depressed:

1) The Mem Access switch on the front panel is depressed. This causes the Request Multiplexer to generate a 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 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 memroy 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 as described in the last section.

7) If it is desired to write data into memory the 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 DEP REQ signal, which is combined with the SYNCA and MEM ACCESS mode signals to produce the memory write signal R/W. 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 DEP DAEN, produced by combining the DEP REQ and DA ENBL signals. The data will thus be written into the selected memory location.

A special case of memory access, Deposit at Halt, is enabled by the DEP AT HLT switch, which produces the DEP @ HLT MODE signal.

This signal is combined with the RUN signal which indicates that the processor is running. When the processor enters a WAIT or STOPPED state, a deposit cycle will automatically be initiated.

Manual I/O Access Operations

A Manual I/O access operation is performed as follows, after the WAIT switch is depressed:

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 HOLD ACK.

3) The I/O Address signifying the I/O device to be accessed is loaded into A/D/I switches 9-13. 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 Data In lines, gated onto the Data Out lines by signal I/O IN, produced by the I/O ACCESS MODE signal, and will be displayed, as discussed earlier in this chapter.

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 as described in Section 8.2.2, except that 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 first 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 $\overline{\text{INT REQ}}$. 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.

   A Reset operation is executed in the same fashion as an Interrupt operation, except that the Reset switch is depressed, producing signal $\overline{\text{RST CTL SW}}$. This signal causes the Request Multiplexer to issue signal $\overline{\text{RESET}}$, and also produces $\overline{\text{RST REQEN}}$. $\overline{\text{RST REQEN}}$ causes a hard wired RESTART to the instruction at memory location zero to be gated onto the Interrupt Instruction bus.

Sense Operations

A Sense operation is executed in the following manner:

1) The data which is to be input is loaded into the upper 8 Address/Instruction/Data switches.

2) The *Sense* switch is depressed. This causes signal SENSE REQEN to be generated, which causes the switch 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 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.

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 TRUE for approximately 4μs, which enables the processor to execute one cycle of operation, after which it again enters the WAIT mode.

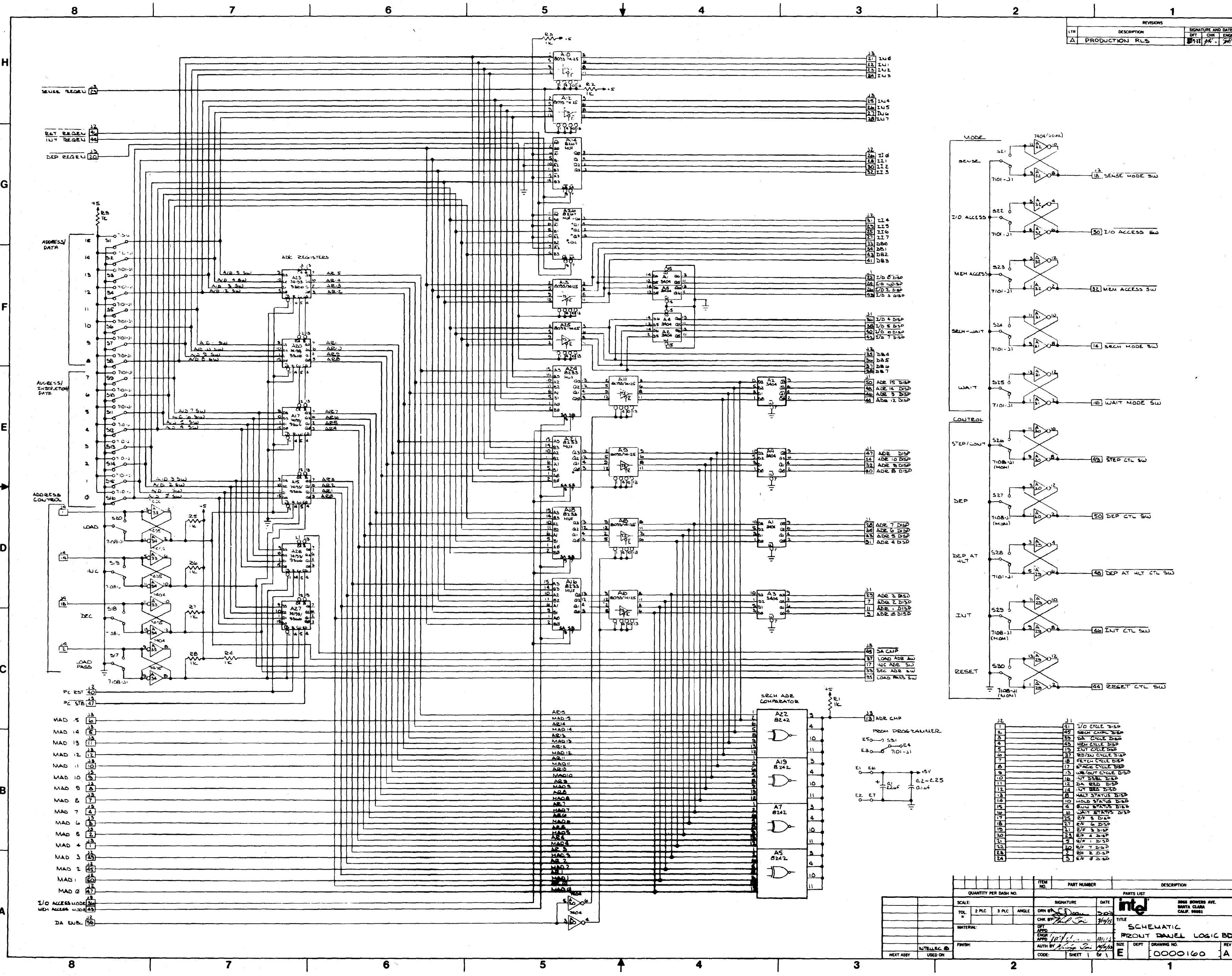


Figure 7-1. Front Panel Logic Schematic Diagram

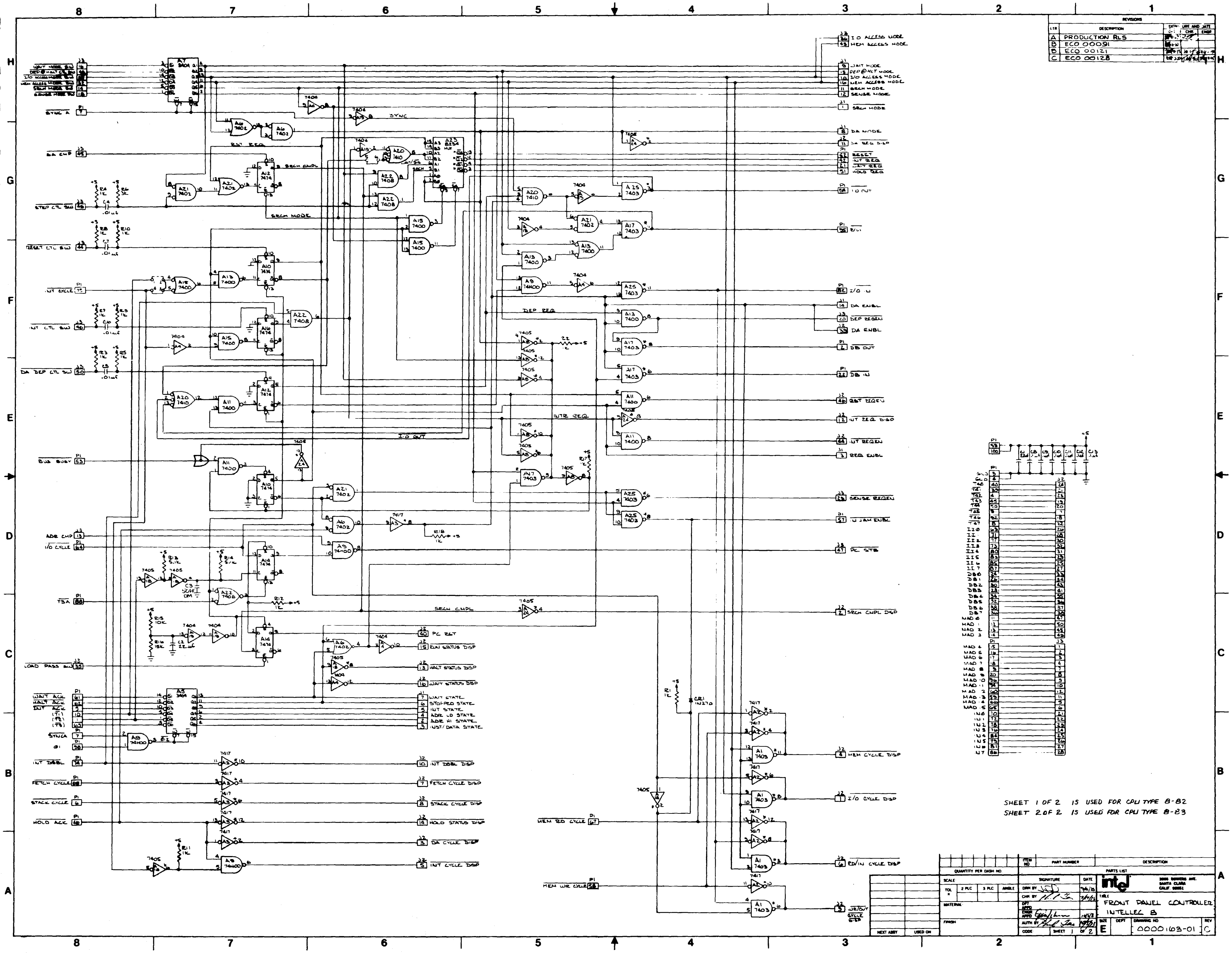


Figure 7-2. Front Panel Controller Schematic Diagram

CHAPTER 8 THE CHASSIS, MOTHER BOARD, AND POWER SUPPLIES

The INTELLEC 8/MOD 8 Chassis, Mother Board, and Power Supplies are designed to provide the housing, interconnection, and power services which bring separate circuit cards together as an INTELLEC 8/MOD 8 system.

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

The INTELLEC 8/MOD 8 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. If greater expansion is planned, an external power supply must be installed to replace the internal supplies.

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 8 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.

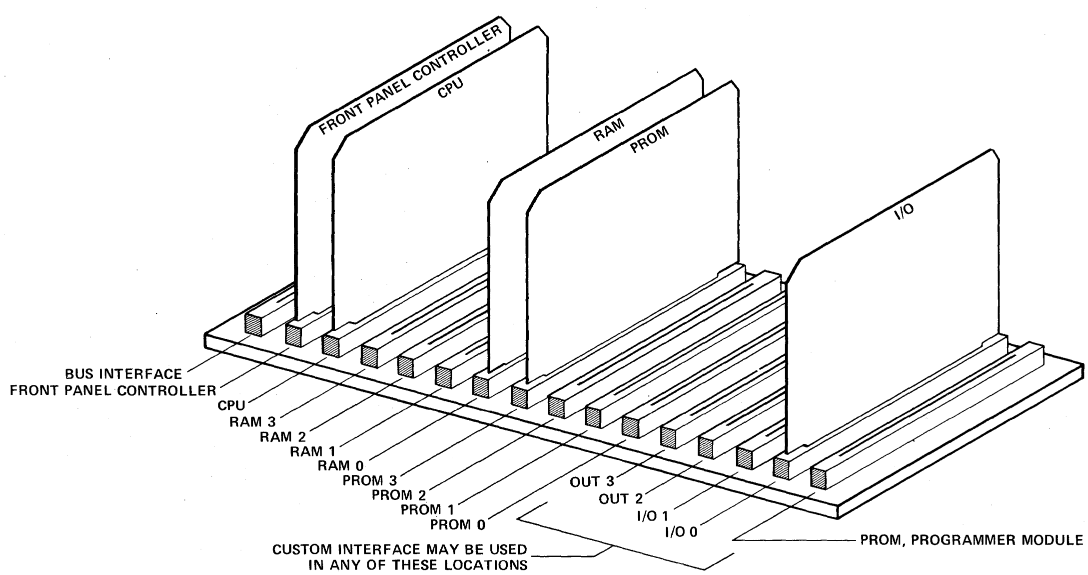


Figure 8-1. INTELLEC 8/MOD 8 Module Assignments

CHAPTER 9 THE imm8-76 PROM PROGRAMMER MODULE

The 1602A And 1702A Programmable Read-Only Memory
Functional Description Of The Module
Interface To The Intellec 8
Theory Of Operation Of The Module
Utilization

The imm8-76 PROM Programmer Module is an optional addition to the INTELLEC 8/MOD 8 system. When used in conjunction with the INTELLEC 8/MOD 8 System Monitor, the Programmer Module permits rapid, automatic loading of Intel 1602A and 1702A Programmable Read Only Memories.

The program to be transferred to a PROM is first stored in the INTELLEC'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 ϕA. It sets up the data to be written into the PROM, at output port ϕB (in Hex). It pulses the power supply the required number of times, at the required duty cycle. And it checks the results 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 8 Operator's Manual.

The imm8-76 is designed for plug-in installation in the INTELLEC 8/MOD 8 mainframe. It makes use of existing connectors and other provisions. No special installation is necessary.

The chapter describes the imm8-76 PROM Programmer. The first subsection contains a brief description of the 1602A/1702A PROM. The second describes the sequence of operations performed by the module during programming. The third gives the detailed theory of operation. The fourth contains utilization information.

THE 1602A AND 1702A PROGRAMMABLE READ ONLY MEMORY

Both the 1602A and the 1702A are programmed by the momentary application of high amplitude pulses on selected pins of the chip. But the 1702A is cleared by a controlled exposure to high intensity ultraviolet. The 1702A may be reloaded as often as desired, making it suitable for use in program development.

Programming of the 1602A or the 1702A requires a carefully controlled sequence of operations. The safety of the chip demands that both the interelement 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 1602A and 1702A. Do not confuse these devices with the 1602 and 1702 versions which preceded them. The 1602 and 1702 PROMs have quite different characteristics, and in particular will not tolerate the power levels used to program the 1602A and 1702A. The imm8-76 is designed to program both the earlier and the later versions, but the utility program contained in the INTELLEC 8/MOD 8 System Monitor is not set up for the programming of 1602 and 1702 PROMs. As a result, any attempt to load 1602 or 1702 memories with the INTELLEC 8/MOD 8 System Monitor will damage the PROM. Such programming is possible, with the proper precautions, but you will have to provide your own software functions. Refer to the INTEL MEMORY DESIGN HANDBOOK for instructions, if you plan to use the imm8-76 for this purpose.

Both the 1602A and 1702A are shipped to the customer in a "cleared" condition; that is, with zeros in all memory locations. An internal zero-state is indicated by a LOW on the output pins of an enabled chip. During programming, ones are loaded selectively into each of the chip's memory locations.

A 1702A 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 1702A, 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 dose (the produce of intensity and the exposure time) is 6 W-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 tissues.
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 1702A, 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 imm8-76 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 switched 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 8

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 8 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's output port #ϕA. The DATA IN lines are connected internally to output port #ϕB. The INTELLEC 8/MOD 8 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 8 System Monitor can check the results of 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 1602A or 1702A PROMs. A LOW applied to pin #30 selects a 2% duty cycle, used when programming 1602 and 1702 devices. In the INTELLEC 8/MOD 8 system, pin #32 of the module is connected to the BIT #7 line of output port #9. Pin #20 is connected to the BIT #6 line of the same output port. The INTELLEC System Monitor controls the Programmer Module by writing into that port.

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 #ϕB. This eight-line signal enters the Programmer Module through a ribbon cable which runs from J2 on the INTELLEC I/O Module ϕ 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 INTELLEC. This data undergoes three successive inversions, between entering and leaving the imm8-76.

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 J2, through a ribbon cable to J3 on the INTELLEC's I/O Module ϕ, and terminates at input port #2. The INTELLEC 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 J2 of the INTELLEC's I/O Module ϕ board. Data originating at output port #ϕB is therefore applied to the eight-line XOR-gate banks, shown on the right in Figure 9-1. The outputs of these gates are 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 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 used to program 1602 and 1702 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.

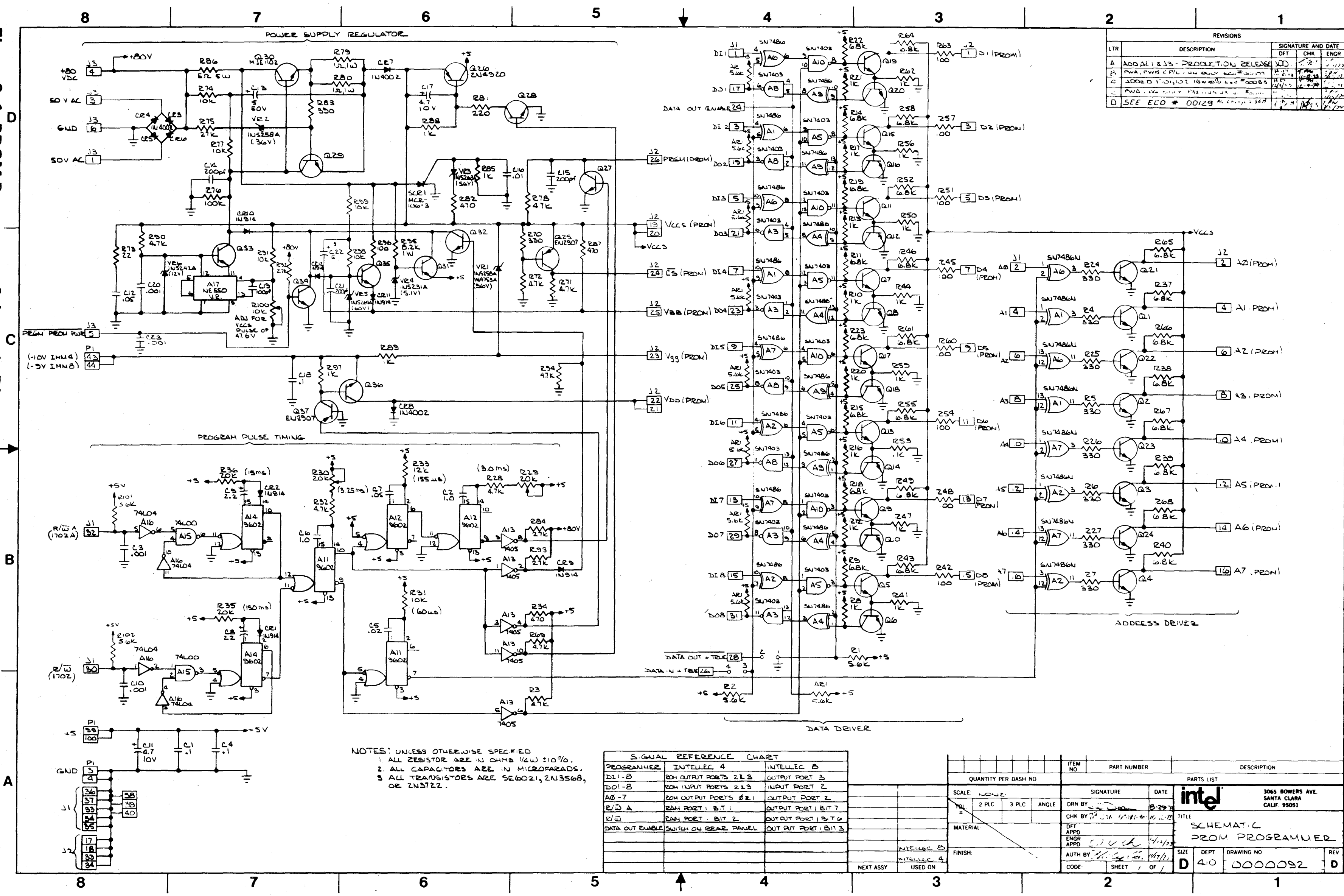
Three and a quarter milliseconds after the beginning of the programming cycle, all signals return to their quiescent levels.

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 9-1. PROM Programmer Schematic Diagram



RECTIFIER AND REGULATOR:

The Programmer Module receives a 50 VAC/60 Hz input, from two 25 Volt transformers which are located on the INTELLEC'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 9-1 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 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 crop, 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

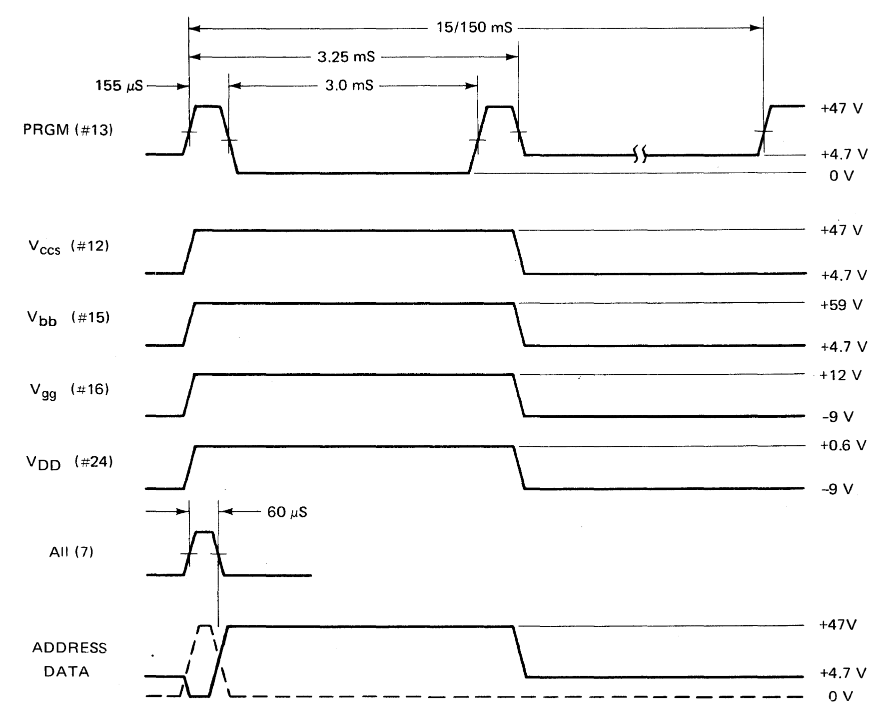


Figure 9-2. PROM Programmer Timing

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. 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 Q26, 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.

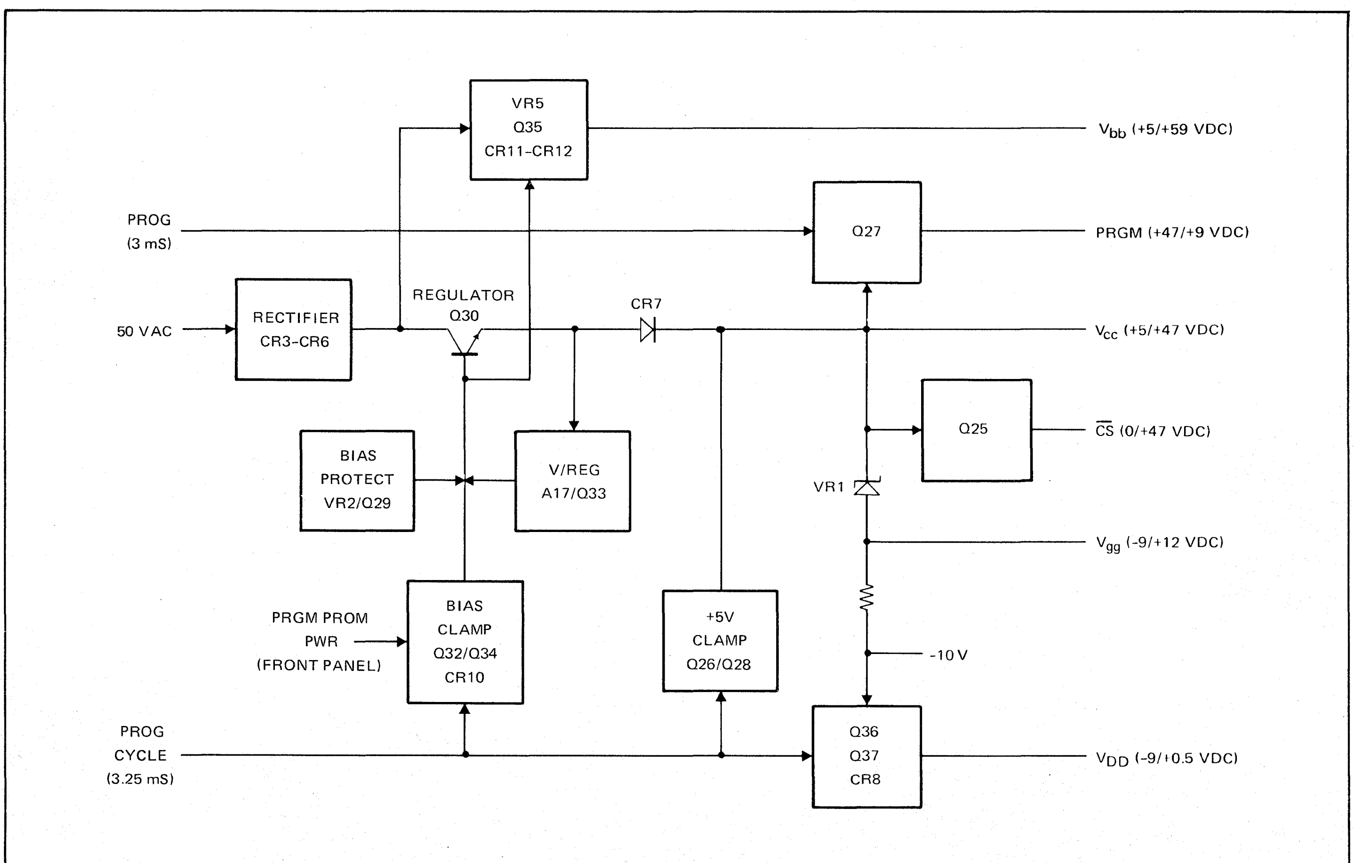


Figure 9-3. Power Supply Functional Block

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 in 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.

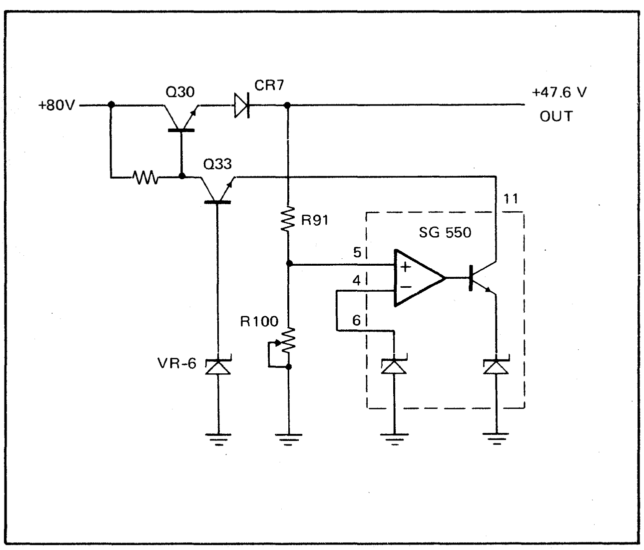


Figure 9-4. Voltage Regulator Loop: Simplified Schematic Equivalent

UTILIZATION

This section describes the utilization of the imm8-76.

Installation

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

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

An umbilical cable, permanently attached to the module, plugs into J34 on the INTELLEC's mother board. This connection supplies AC power and enabling to the Programmer Module.

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

POWER REQUIREMENTS

This module requires power at the following levels:

(a) 50 VAC
(b) +5 ± 5% VDC @ 1.0 A (max)
(c) -9 ± 5% VDC @ 0.2 A (max)

The 50 VAC source shares a fuse with the -9 Volt supply in the INTELLEC. This 0.5 Ampere fuse, F2, is located on the INTELLEC's rear panel.

Pin List

Connector pin allocations on the PROM Programmer Module are given in Tables 9-1, 9-2, and 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		80	
31		81	
32		82	
33		83	
34		84	
35		85	
36		86	
37		87	
38		88	
39		89	
40		90	
41		91	
42		92	
43	-9 VDC	93	
44	-9 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	R/$\overline{W}$A (1702A)
33–40	GROUND

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	GROUND
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33–34	GROUND
35	
36	
37	
38	
39	
40	

Table 9-3

J3 PIN LIST (6-pin Molex Connector)

PIN	SIGNAL FUNCTION		
1	50 VAC (01)		
2			
3	50 VAC (02)		
4	+80 VDC OUT		
5	$\overline{\text{PROGRAM PROM POWER}}$		
6	GROUND		

Table 9-4

CHAPTER 10 INTELLEC 8/MOD 8 SYSTEM UTILIZATION

Intellec 8 Installation
System I/O Interfacing
Intellec 8 System Operating Requirements
External Device Controller Interfacing

This section gives the information necessary to install and operate the INTELLEC 8/MOD 8 system in an application. It is divided into four subsections as follows: the first covers INTELLEC 8/MOD 8 installation; the second describes the requirements for system input/output; the third describes system operation requirements; and the fourth describes the implementation of external device controllers.

INTELLEC 8/MOD 8 INSTALLATION

Installation of the INTELLEC 8/MOD 8 is a very simple matter, 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 8 is almost as simple to install, as it has been designed to mount in any standard 19½-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 8. 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 8 can handle up to eight input ports and twenty-four output ports. These ports are assigned to specific modules as follows:

Module	Ports
I/O 0	INPUT Ports 0-3 OUTPUT Ports 08-0BH
I/O 1	INPUT Ports 4-7 OUTPUT Ports 0C-0FH
OUT 2	Output ports 10-17 H
OUT 3	Output ports 18-1FH

All of the data ports complement data to and from the CPU, and are TTL compatible. Note that the two input ports (0 and 2) and two output ports (08 & 0AH) are not available to the user. The data from the other ports is brought, via flat cables, to the back panel of the INTELLEC, 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 8 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.

A table of the back panel jack pins and their associated signals is provided in Table 10-2.

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 uf capacitor, and the pullup resistor should be 1 Kohm.

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 8 SYSTEM OPERATING REQUIREMENTS

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

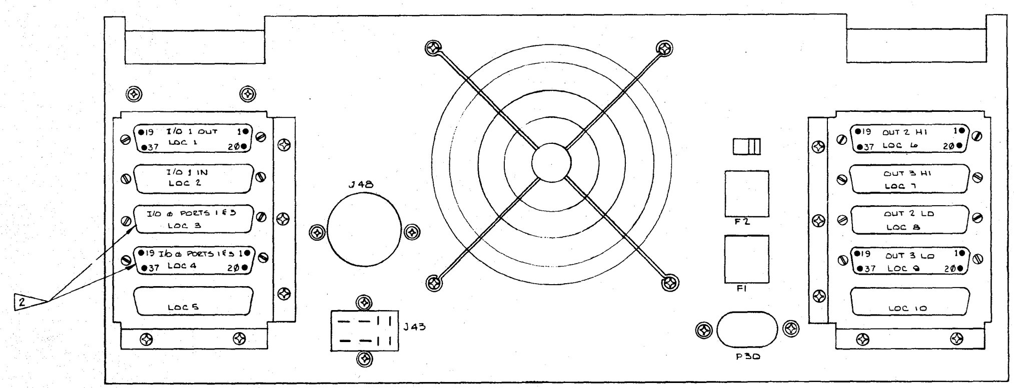
First, *never* operate the INTELLEC 8/MOD 8 with the cover off. If this is done, the proper flow of air will be disrupted, resulting in the burning-out of the internal power supplies.

Second, use extreme care when removing or installing

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

CABLE INTERFACE	
FROM	TO
LOC 1	I/O MOD. 1 J5 (OUT)
LOC 2	I/O MOD. 1 J4 (IN)
LOC 3	TO I/O MOD Ø PORTS 1 & 3 (J4 & J5)
LOC 4	LOC 3
LOC 6	OUTPUT MOD. 2 J3 (HI)
LOC 7	OUTPUT MOD. 3 J3 (HI)
LOC 8	OUTPUT MOD. 2 J2 (LO)
LOC 9	OUTPUT MOD. 3 J2 (LO)
+5V PWR SUP. P47	BACK PANEL J47
AUX OUTLET P48	BACK PANEL J48
TTY CONN P43	BACK PANEL J43
A.C. RECPT. J30	BACK PANEL P30

REVISIONS		
LTR	DESCRIPTION	SIGNATURE AND DATE
A	PROD. RELEASE	
B	SEE ECO# 00171-3 OF 6	



BACK PANEL

NOTE:

1. LOC. 5, 10 AND 51 ARE NOT USED
2. READER CABLE P/N 4000286 OR PUNCH CABLE P/N 4000287 MAYBE INSERTED INTO LOC. 3 OR LOC. 4.

ITEM NO.	PART NUMBER	DESCRIPTION
QUANTITY PER DASH NO.	PARTS LIST	

NEXT ASSY	4000128
USED ON	INTELLEC 8
SCALE	NONE
DRN BY	11/20/73
CHK BY	
ENGR APPD	
AUTH BY	
CODE	SHEET 1 OF 2
intel	3065 BOWERS AVE. SANTA CLARA CALIF. 95051
TITLE	BACK PANEL I/Ø CABLE & WIRE LIST
SIZE	D
DEPT	410
DRAWING NO.	4000240
REV	B

I/O Port Assignments – Module I/O 0

SIGNAL	SYMBOL	COMMENTS	BACK PANEL CONN. PIN # (3)	MODULE PIN # J5
OUTPUT PORT 08, BIT 0	$\overline{OP08}$, 0	UART XMIT DATA 0	(1)	2
	1			3
	2			4
	3			5
	4			6
	5			7
	6			8
OUTPUT PORT 08, BIT 7	$\overline{OP08}$, 7	UART XMIT DATA 7	(1)	9
OUTPUT PORT 09, BIT 0	$\overline{OP09}$, 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 09, BIT 7	$\overline{OP09}$, 7	R/$\overline{W}$A	32	18
OUTPUT PORT 0A, BIT 0	$\overline{OP0A}$, 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 0A, BIT 7	$\overline{OP0A}$, 7	PROM ADR IN 7	(2)	27
OUTPUT PORT 0B, BIT 0	$\overline{OP0B}$, 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 0B, BIT 7	$\overline{OP0B}$, 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. (Continued)

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			
$\overline{\text{IP4}}$	$\overline{\text{OP0C}}$	$\overline{\text{OP10}}$	$\overline{\text{OP14}}$	$\overline{\text{OP18}}$	$\overline{\text{OP1C}}$	0	2	2
						1	3	3
						2	21	4
						3	22	5
						4	4	6
						5	5	7
						6	23	8
$\overline{\text{IP4}}$	$\overline{\text{OP0C}}$	$\overline{\text{OP10}}$	$\overline{\text{OP14}}$	$\overline{\text{OP18}}$	$\overline{\text{OP1C}}$	7	24	9
$\overline{\text{IP5}}$	$\overline{\text{OP0D}}$	$\overline{\text{OP11}}$	$\overline{\text{OP15}}$	$\overline{\text{OP19}}$	$\overline{\text{OP1D}}$	0	6	11
						1	7	12
						2	25	13
						3	26	14
						4	8	15
						5	9	16
						6	27	17
$\overline{\text{IP5}}$	$\overline{\text{OP0D}}$	$\overline{\text{OP11}}$	$\overline{\text{OP15}}$	$\overline{\text{OP19}}$	$\overline{\text{OP1D}}$	7	28	18
$\overline{\text{IP6}}$	$\overline{\text{OP0E}}$	$\overline{\text{OP12}}$	$\overline{\text{OP16}}$	$\overline{\text{OP1A}}$	$\overline{\text{OP1E}}$	0	10	20
						1	11	21
						2	29	22
						3	30	23
						4	12	24
						5	13	25
						6	31	26
$\overline{\text{IP6}}$	$\overline{\text{OP0E}}$	$\overline{\text{OP12}}$	$\overline{\text{OP16}}$	$\overline{\text{OP1A}}$	$\overline{\text{OP1E}}$	7	32	27
$\overline{\text{IP7}}$	$\overline{\text{OP0F}}$	$\overline{\text{OP13}}$	$\overline{\text{OP17}}$	$\overline{\text{OP1B}}$	$\overline{\text{OP1F}}$	0	14	29
						1	15	30
						2	33	31
						3	34	32
						4	16	33
						5	17	34
$\overline{\text{IP7}}$	$\overline{\text{OP0F}}$	$\overline{\text{OP13}}$	$\overline{\text{OP17}}$	$\overline{\text{OP1B}}$	$\overline{\text{OP1F}}$	6	35	35
						7	36	36
						GND	1,18,19,20,37	37-40

Table 10-2

individual circuit cards in the INTELLEC 8/MOD 8, 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/MOD 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 busses, which can relinquish their control of INTELLEC 8/MOD 8 operations to an external device.

In order to make use of this capability, two requirements must be met: a HOLD request must be issued, and the BUS BUSY line must be monitored and controlled.

An external device controller performs a Direct Access operation by requesting a HOLD state of the processor. When the processor acknowledges this request, it goes into a WAIT state, and gives control of the Memory Address and control buses to the device controller. The controller then issues a BUS BUSY signal, which prevents other devices from assuming control of the buses until its operation is complete. A user device controller, therefore, must have the circuitry necessary to generate a HOLD REQ signal and also to check BUS BUSY and, if the bus is not busy, to issue BUS BUSY. See Section 8, the Control Console, for a representative peripheral device controller with these facilities.

APPENDIX A 8008 INSTRUCTION SET

INSTRUCTION SUMMARY

This appendix provides a summary of INTELLEC 8/MOD 8 assembly language instructions. Abbreviations used are as follows:

A	The accumulator (register A)
A_n	Bit n of the accumulator contents, where n may have any value from 0 to 7.
ADDR	Any memory address
Carry	The carry bit
CODE	An operation code
DATA	Any byte of data
DST	Destination register or memory byte
EXP	A constant or mathematical expression
LABEL:	Any instruction label
M	A memory byte
Parity	The parity bit
PC	Program Counter
REGM	Any register or memory byte
sign	The sign bit
SRC	Source register or memory byte
STK	Top stack register
zero	The zero bit
[ ]	An optional field enclosed by brackets
( )	Contents of register or memory byte enclosed by brackets
←	Replace left hand side with right hand side of arrow

A.1 SINGLE REGISTER INSTRUCTIONS

Format:

[LABEL:] CODE REGM

Note: REGM $\neq$ A or M

CODE	DESCRIPTION
INR	(REGM) $\leftarrow$ (REGM) +1 Increment register REGM
DCR	(REGM) $\leftarrow$ (REGM) −1 Decrement register REGM

Condition bits affected: Zero, sign, parity

A.2 MOV INSTRUCTIONS

Format:

[LABEL:] MOV DST,SRC

Note: SRC and DST not both = M

CODE	DESCRIPTION
MOV	(DST) $\leftarrow$ (SRC) Load register DST from register SRC

Condition bits affected: None

A.3 REGISTER OR MEMORY TO ACCUMULATOR INSTRUCTIONS

Format:

[LABEL:] CODE REGM

CODE	DESCRIPTION	
ADD	(A) $\leftarrow$ (A) + (REGM)	Add REGM to accumulator
ADC	(A) $\leftarrow$ (A) + (REGM) + (carry)	Add REGM plus carry bit to accumulator
SUB	(A) $\leftarrow$ (A) − (REGM)	Subtract REGM from accumulator
SBB	(A) $\leftarrow$ (A) − (REGM) − (carry)	Subtract REGM minus carry
ANA	(A) $\leftarrow$ (A) AND (REGM)	AND accumulator with REGM
XRA	(A) $\leftarrow$ (A) XOR (REGM)	Exclusive-OR accumulator with REGM
ORA	(A) $\leftarrow$ (A) OR (REGM)	OR accumulator with REGM
CMP	Condition bits set by (A) − (REGM)	Compare REGM with accumulator

Condition bits affected:

ADD, ADC, SUB, SBB: Carry, sign, zero, parity

ANA, XRA, DRA: Sign, zero, parity. Carry is reset to zero.

CMP; Carry, sign, zero, parity. Zero set if (A) = (REGM)
Carry reset if (A) $<$ (REGM)
Carry set if (A) $\geqslant$ (REGM)

A.4 ROTATE ACCUMULATOR INSTRUCTIONS

Format:

[LABEL:]   CODE

CODE	DESCRIPTION	
RLC	(carry) $\leftarrow A_7$, $A_n + 1$, $\leftarrow A_n$, $A_0 \leftarrow A_7$	Set carry = A_7, rotate accumulator left
RRC	(carry) $\leftarrow A_0$, $A_n \leftarrow A_n + 1$, $A_7 \leftarrow A_0$	Set carry = A_0, rotate accumulator right
RAL	$A_n + 1 \leftarrow A_n$, (carry) $\leftarrow A_7$, $A_0 \leftarrow$ (carry)	Rotate accumulator right through the carry
RAR	$A_n \leftarrow A_n + 1$, (carry) $\leftarrow A_0$, $A_7 \leftarrow$ (carry)	Rotate accumulator left through the carry

Condition bits affected: Carry

A.5 IMMEDIATE INSTRUCTIONS

Format:

[LABEL:]   MVI   REGM, DATA

or

[LABEL:]   CODE   REGM

CODE	DESCRIPTION	
MVI	(REGM) $\leftarrow$DATA	Move immediate DATA into REGM
ADI	(A) $\leftarrow$(A) + DATA	Add immediate data to accumulator
ACI	(A) $\leftarrow$(A) + DATA + (carry)	Add immediate data + carry to accumulator
SUI	(A) $\leftarrow$(A) - DATA	Subtract immediate data from accumulator
SBI	(A) $\leftarrow$(A) – DATA – (carry)	Subtract immediate data and carry from accumulator
ANI	(A) $\leftarrow$(A) AND DATA	AND accumulator with immediate data
XRI	(A) $\leftarrow$(A) XOR DATA	Exclusive-OR accumulator with immediate data
ORI	(A) $\leftarrow$(A) OR DATA	OR accumulator with immediate data
CPI	Condition bits set by (A)–DATA	Compare immediate data with accumulator

Condition bits affected:

MVI: None

ADI, ACI, SUI, SBI: Carry, sign, zero, parity

ANI, XRI, ORI: Zero, sign, parity. Carry is reset to zero.

CPI: Carry, sign, zero, parity.
- Zero set if (A) = DATA
- Carry reset if (A) $<$DATA
- Carry set if (A) $\geqslant$DATA

A.6 JUMP INSTRUCTIONS

Format:

[LABEL]:   CODE   ADDR

CODE	DESCRIPTION	
JMP	(PC) ←ADDR	Jump to location ADDR
JC	If (carry) =1, (PC) ←ADDR If (carry) =0, (PC) ←(PC)+3	Jump to ADDR if carry set
JNC	If (carry) =0, (PC) ←ADDR If (carry) =1, (PC) ←(PC)+3	Jump to ADDR if carry reset
JZ	If (zero) =1, (PC) ←ADDR If (zero) =0, (PC) ←(PC)+3	Jump to ADDR of zero set
JNZ	If (zero) =0, (PC) ←ADDR If (zero) =1, (PC) ←(PC)+3	Jump to ADDR if zero reset
JP	If (sign) =0, (PC) ←ADDR If (sign) =1, (PC) ←(PC)+3	Jump to ADDR if plus
JM	If (sign) =1, (PC) ←ADDR If (sign) =0, (PC) ←(PC)+3	Jump to ADDR if minus
JPE	If (parity) =1, (PC) ←ADDR If (parity) =0, (PC) ←(PC)+3	Jump to ADDR if parity even
JPO	If (parity) =0, (PC) ←ADDR If (parity) =1, (PC) ←(PC)+3	Jump to ADDR if parity odd

Condition bits affected: None

A.7 CALL INSTRUCTIONS

Format:

[LABEL:]   CODE   ADDR

CODE	DESCRIPTION	
CALL	(STK) ←(PC), (PC) ←ADDR	Call subroutine and push return address onto stack
CC	If (carry) =1, (STK) ←(PC), (PC) ←(ADDR) If (carry) =0, (PC) ←(PC)+3	Call subroutine if carry set
CNC	If (carry) =0, (STK) ←(PC), (PC) ←(ADDR) If (carry) =1, (PC) ←(PC)+3	Call subroutine if carry set
CZ	If (zero) =1, (STK) ←(PC), (PC) ←(ADDR) If (zero) =0, (PC) ←(PC)+3	Call subroutine if zero set
CNZ	If (zero) =0, (STK) ←(PC), (PC) ←(ADDR) If (zero) =1, (PC) ←(PC)+3	Call subroutine if zero reset
CP	If (sign) =0, (STK) ←(PC), (PC) ←(ADDR) If (sign) =1, (PC) ←(PC)+3	Call subroutine if sign plus
CM	If (sign) =1, (STK) ←(PC), (PC) ←(ADDR) If (sign) =0, (PC) ←(PC)+3	Call subroutine if sign minus
CPE	If (parity) =1, (STK) ←(PC), (PC) ←(ADDR) If (parity) =0, (PC) ←(PC)+3	Call subroutine if parity even
CPO	If (parity) =0, (STK) ←(PC), (PC) ←(ADDR) If (parity) =1, (PC) ←(PC)+3	Call subroutine if parity odd

Condition bits affected: None

A.8 RETURN INSTRUCTIONS

Format:

[LABEL:] CODE

CODE	DESCRIPTION	
RET	(PC) ←STK	Return from subroutine
RC	If (carry) =1, (PC) ← STK If (carry) =0, (PC) ←(PC)+3	Return if carry set
RNC	If (carry) =0, (PC) ←STK If (carry) =1, (PC) ←(PC)+3	Return if carry reset
RZ	If (zero) =1, (PC) ←STK If (zero) =0, (PC) ←(PC)+3	Return if zero set
RNZ	If (zero) =0, (PC) ←STK If (zero) =1, (PC) ←(PC)+3	Return if zero reset
RM	If (sign) =1, (PC) ←STK If (sign) =0, (PC) ←(PC)+3	Return if minus
RP	If (sign) =0, (PC) ←STK If (sign) =1, (PC) ←(PC)+3	Return if plus
RPE	If (parity) =1, (PC) ←STK If (parity) =0, (PC) ←(PC)+3	Return if parity even
RPO	If (parity) =0, (PC) ←STK If (parity) =1, (PC) ←(PC)+3	Return if parity odd

Condition bits affected: None

A.9 RST INSTRUCTION

Format:

[LABEL:] RST EXP

Note: 0 ⩽ EXP ⩽ 7

CODE	DESCRIPTION	
RST	(STK) ←(PC) (PC) ←00000000EXP000B	Call subroutine at address specified by EXP

Condition bits affected: None

A.10 INPUT/OUTPUT INSTRUCTIONS

Format:

[LABEL:] CODE EXP

Note: For IN, 0 ⩽ EXP ⩽ 7
For OUT, 8 ⩽ EXP ⩽ 31

CODE	DESCRIPTION	
IN	(A) ←input device	Read a byte from device EXP into the accumulator
OUT	output device ←(A)	Send the accumulator contents to device EXP

Condition bits affected: None

A.11 ORG PSEUDO-INSTRUCTION

Format:

ORG EXP

CODE	DESCRIPTION	
ORG	LOCATION COUNTER ← EXP	Set Assembler location counter to EXP

A.12 EQU PSEUDO-INSTRUCTION

Format:

LABEL EQU EXP

CODE	DESCRIPTION	
EQU	LABEL ← EXP	Assign the value EXP to the symbol LABEL

A.13 SET PSEUDO-INSTRUCTION

Format:

LABEL SET EXP

CODE	DESCRIPTION	
SET	LABEL ← EXP	Assign the value EXP to the symbol LABEL, which may have been previously SET.

A.14 END PSEUDO-INSTRUCTION

Format:

END

CODE	DESCRIPTION
END	End the assembly.

A.15 CONDITIONAL ASSEMBLY PSEUDO-INSTRUCTIONS

Format:

```
        IF        EXP
          and
          ENDIF
```

CODE	DESCRIPTION
IF	If EXP =0, ignore assembler statements until ENDIF is reached. Otherwise continue assembling statements.
ENDIF	End range of preceding IF.

A.16 MACRO DEFINITION PSEUDO-INSTRUCTIONS

Format:

```
        NAME      MACRO      LIST
                    and
                   ENDM
```

CODE	DESCRIPTION
MACRO	Define a macro named NAME with parameters LIST.
ENDM	End macro definition.

APPENDIX B ELECTRICAL CHARACTERISTICS OF LOGIC ELEMENTS USED IN THE INTELLEC 8/MOD 8

SINGLE CHIP EIGHT-BIT PARALLEL CENTRAL PROCESSOR UNIT

■ Heart of MCS-8T.M. Microcomputer Set

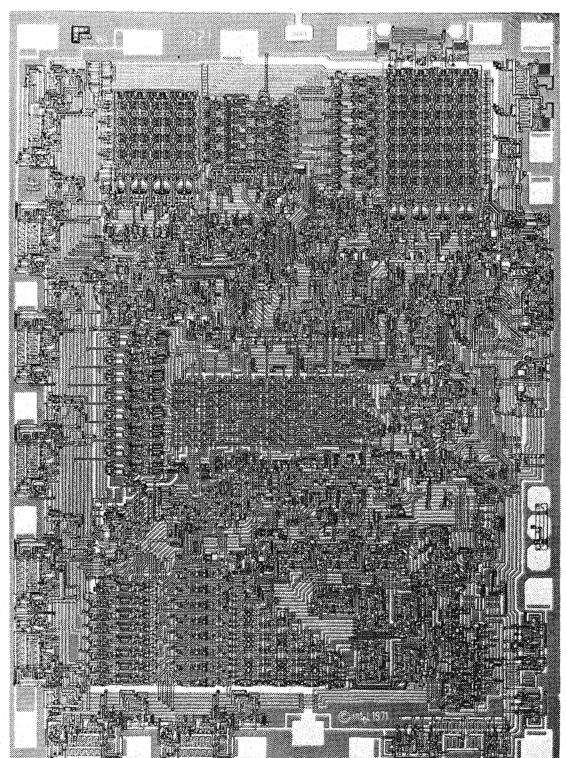
- 8-Bit Parallel CPU on a Single Chip
- 48 Instructions, Data Oriented
- Complete Instruction Decoding and Control Included
- Instruction Cycle Time — 12.5 μs with 8008-1 or 20 μs with 8008
- TTL Compatible (Inputs, Outputs and Clocks)
- Can be used with any type or speed semiconductor memory in any combination
- Directly addresses 16K x 8 bits of memory (RAM, ROM, or S.R.)
- Memory capacity can be indefinitely expanded through bank switching using I/O instructions
- Address stack contains eight 14-bit registers (including program counter) which permit nesting of subroutines up to seven levels
- Contains seven 8-bit registers
- Interrupt Capability
- Packaged in 18-Pin DIP

The 8008 is an 8-bit central processor designed especially to handle large volumes of data. When used with any combination of Intel RAMs, ROMs and shift registers, the 8008 CPU forms the MCS-8 TM micro computer system, a system which can directly address and retrieve as many as 16,000 8-bit bytes stored in the memory devices.

This single chip 8008 CPU fabricated with Intel's standard P-channel silicon-gate MOS process contains an 8-bit accumulator, two 8-bit temporary registers, four flag bits and eight 14-bit address registers. It operates under a powerful set of 48 instructions, has interrupt capability, operates asynchronously or synchronously with external memory, and can execute subroutines nested up to seven levels.

All inputs, including clocks, are TTL compatible. All outputs are low-power TTL signals. Using standard TTL packages, the CPU may be interfaced with ROMs, RAMs, and SRs. A complete functioning computer system may be built with one CPU, one ROM and 20 standard TTL devices.

The 8008 CPU combines with Intel memory devices to provide complete computing and control functions for test systems, data terminals, billing machines, scientific calculators, measuring systems, numeric control systems and process control systems.



8008 Photomicrograph With Pin Designations

(Note: The detailed functional specifications describing the operation of the CPU, the instruction set, and programming and hardware examples are published separately and are available upon request.)

The 8008 is a single chip MOS 8-bit parallel central processor unit for the MCS-8 micro computer system. A micro computer system is formed when the 8008 is interfaced with any type or speed standard semiconductor memory up to 16K x 8-bit words. Examples are Intel's 1101, 1103, and 2102 (RAMs); 1302, 1602A, 1702A, 8316 (ROMs).

The processor communicates over an 8-bit data and address bus (D_0 through D_7) and uses two input leads (READY and INTERRUPT) and four output leads (S_0, S_1, S_2, and Sync) for control. Time multiplexing of the data bus allows control information, 14 bit addresses, and data to be transmitted between the CPU and external memory.

This CPU contains six 8-bit data registers, an 8-bit accumulator, two 8-bit temporary registers, four flag bits (carry, zero, sign, parity), and an 8-bit parallel binary arithmetic unit which implements addition, subtraction, and logical operations. A memory stack containing a 14-bit program counter and seven 14-bit words is used internally to store program and subroutine addresses. The 14-bit address permits the direct addressing of 16K words of memory (any mix of RAM, ROM or S.R.).

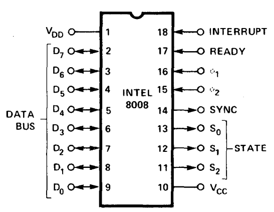
The control portion of the chip contains logic to implement a variety of register transfer, arithmetic control, and logical instructions. Most instructions are coded in one byte (8 bits); data immediate instructions use two bytes; jump instructions utilize three bytes. The 8008 operates with a 500 kHz clock, and executes non-memory referencing instructions in 20 microseconds: the 8008-1 operates with an 800 kHz clock and executes non-memory referencing instructions in 12.5 microseconds.

All inputs (including clocks) are TTL compatible and all outputs are low-power TTL compatible.

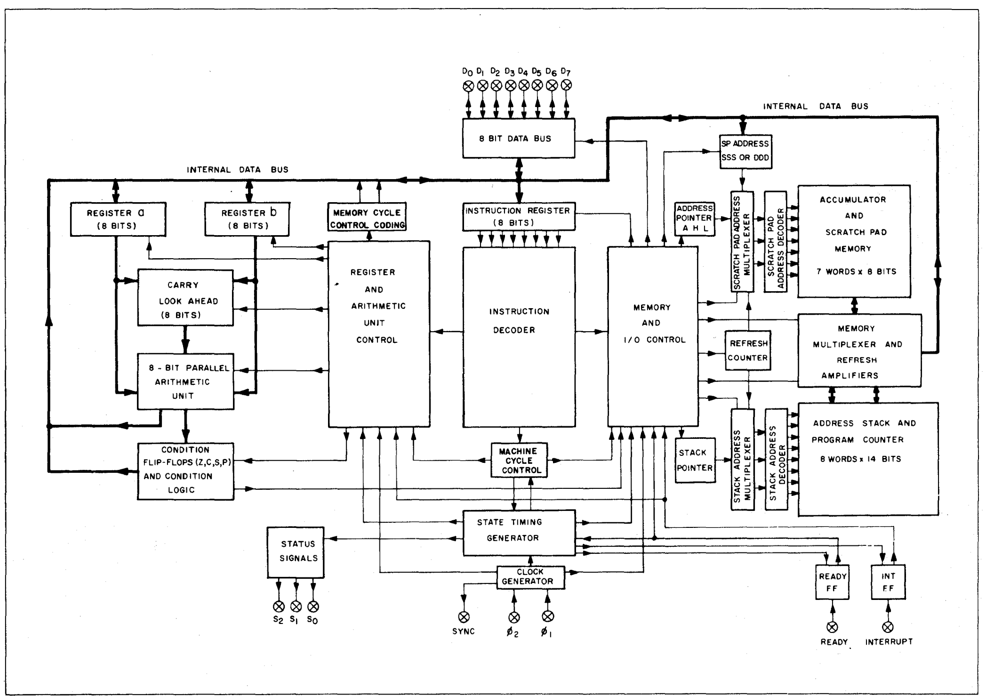
The instruction set of the 8008 consists of 48 instructions including data manipulation, binary arithmetic, and jump to subroutine.

The normal program flow of the 8008 may be interrupted through the use of the INTERRUPT control line. This allows the servicing of slow I/O peripheral devices while also executing the main program.

The READY command line synchronizes the 8008 to the memory cycle allowing any type or speed of semiconductor memory to be used.



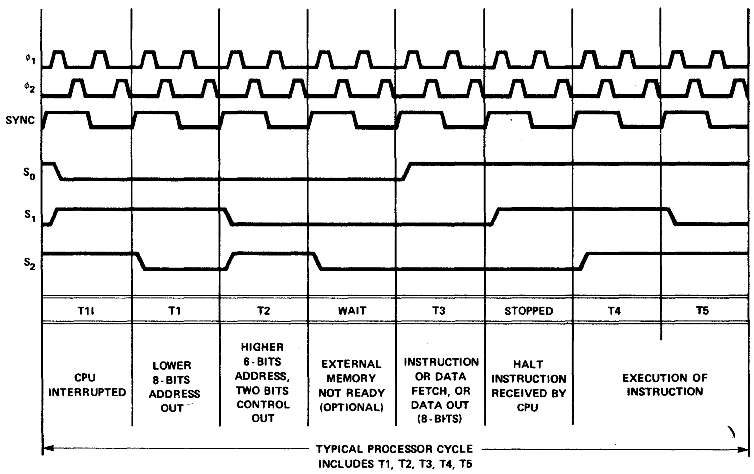
Pin Configuration



8008 Block Diagram

System Timing

Typically, a processor instruction cycle consists of five states, two states in which an address is sent to memory (T1 and T2), one for the instruction or data fetch (T3), and two states for the execution of the instruction (T4 and T5). If the processor is used with slow memories, the READY line synchronizes the processor with the memories. When the memories are not avialable for either sending or receiving data, the processor goes into the WAIT state. The accompanying diagram illustrates the processor activity during a single cycle.



Basic 8008 Instruction Cycle

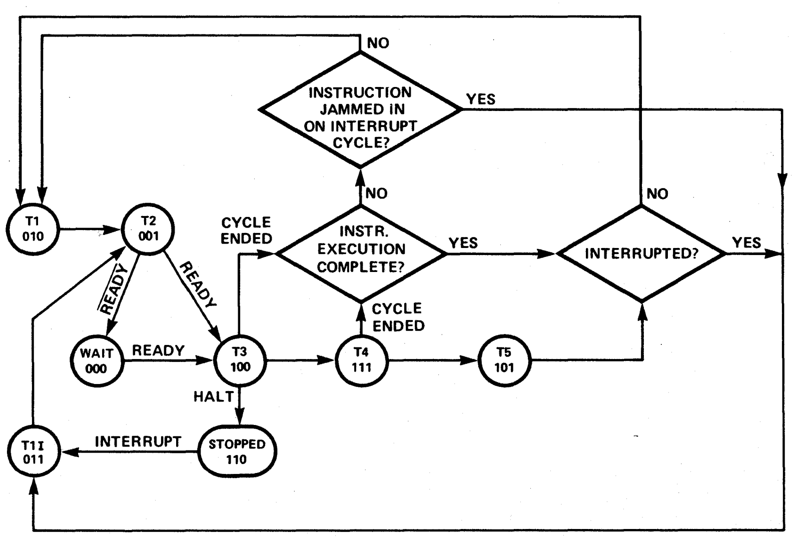
Instructions for the 8008 require one, two, or three machine cycles for complete execution. The first cycle is always an instruction fetch cycle (PCI). The second and third cycles are data reading (PCR), data writing (PCW) or I/O operations (PCC). The processor controls the use of the data bus and determines whether it will be sending or receiving data. State signals S_0, S_1, and S_2, along with SYNC inform the peripheral circuitry of the state of the processor. Many of the multi-cycle instructions for the 8008 do not require the two execution states, T4 and T5. As a result, these states are omitted when they are not needed, and the 8008 operates asynchronously with respect to the cycle length. The state transition diagram for the processor is shown below. Refer to the 8008 manual for the detailed operation of the CPU during each state of each instruction.

State Control Coding

S_0	S_1	S_2	STATE
0	1	0	T1
0	1	1	T1I
0	0	1	T2
0	0	0	WAIT
1	0	0	T3
1	1	0	STOPPED
1	1	1	T4
1	0	1	T5

Cycle Control Coding

D_6	D_7	CYCLE
0	0	PCI
0	1	PCR
1	0	PCC
1	1	PCW



CPU State Transition Diagram

Basic Instruction Set

Data and Instruction Formats

Data in the 8008 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.

		TYPICAL INSTRUCTIONS
One Byte Instructions		
D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	OP CODE	Register to register, memory reference, I/O arithmetic or logical, rotate or return 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 instructions
Three 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	LOW ADDRESS	JUMP or CALL instructions
X X D_5 D_4 D_3 D_2 D_1 D_0	HIGH ADDRESS*	*For the third byte of this instruction, D_6 and D_7 are "don't care" bits.

For the MCS-8 a logic "1" is defined as a high level and a logic "0" is defined as a low level.

Index Register Instructions

The load instructions do not affect the flag flip-flops. The increment and decrement instructions affect all flip-flops except the carry.

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE $D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
(1) MOV r_1, r_2	(5)	1 1	D D D	S S S	Load index register r_1 with the content of index register r_2.
(2) MOV r, M	(8)	1 1	D D D	1 1 1	Load index register r with the content of memory register M.
MOV M, r	(7)	1 1	1 1 1	S S S	Load memory register M with the content of index register r.
(3) MVI r	(8)	0 0	D D D	1 1 0	Load index register r with data B . . . B.
		B B	B B B	B B B	
MVI M	(9)	0 0	1 1 1	1 1 0	Load memory register M with data B . . . B.
		B B	B B B	B B B	
INR r	(5)	0 0	D D D	0 0 0	Increment the content of index register r (r ≠ A).
DCR r	(5)	0 0	D D D	0 0 1	Decrement the content of index register r (r ≠ A).

Accumulator Group Instructions

The result of the ALU instructions affect all of the flag flip-flops. The rotate instructions affect only the carry flip-flop.

ADD r	(5)	1 0	0 0 0	S S S	Add the content of index register r, memory register M, or data
ADD M	(8)	1 0	0 0 0	1 1 1	B . . . B to the accumulator. An overflow (carry) sets the carry
ADI	(8)	0 0	0 0 0	1 0 0	flip-flop.
		B B	B B B	B B B	
ADC r	(5)	1 0	0 0 1	S S S	Add the content of index register r, memory register M, or data
ADC M	(8)	1 0	0 0 1	1 1 1	B . . . B from the accumulator with carry. An overflow (carry)
ACI	(8)	0 0	0 0 1	1 0 0	sets the carry flip-flop.
		B B	B B B	B B B	
SUB r	(5)	1 0	0 1 0	S S S	Subtract the content of index register r, memory register M, or
SUB M	(8)	1 0	0 1 0	1 1 1	data B . . . B from the accumulator. An underflow (borrow)
SUI	(8)	0 0	0 1 0	1 0 0	sets the carry flip-flop.
		B B	B B B	B B B	
SBB r	(5)	1 0	0 1 1	S S S	Subtract the content of index register r, memory register M, or data
SBB M	(8)	1 0	0 1 1	1 1 1	data B . . . B from the accumulator with borrow. An underflow
SBI	(8)	0 0	0 1 1	1 0 0	(borrow) sets the carry flip-flop.
		B B	B B B	B B B	

Basic Instruction Set

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE $D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
ANA r	(5)	1 0	1 0 0	S S S	Compute the logical AND of the content of index register r, memory register M, or data B . . . B with the accumulator.
ANA M	(8)	1 0	1 0 0	1 1 1	
ANI	(8)	0 0 B B	1 0 0 B B B	1 0 0 B B B	
XRA r	(5)	1 0	1 0 1	S S S	Compute the EXCLUSIVE OR of the content of index register r, memory register M, or data B . . . B with the accumulator.
XRA M	(8)	1 0	1 0 1	1 1 1	
XRI	(8)	0 0 B B	1 0 1 B B B	1 0 0 B B B	
ORA r	(5)	1 0	1 1 0	S S S	Compute the INCLUSIVE OR of the content of index register r, memory register m, or data B . . . B with the accumulator.
ORA M	(8)	1 0	1 1 0	1 1 1	
ORI	(8)	0 0 B B	1 1 0 B B B	1 0 0 B B B	
CMP r	(5)	1 0	1 1 1	S S S	Compare the content of index register r, memory register M, or data B . . . B with the accumulator. The content of the accumulator is unchanged.
CMP M	(8)	1 0	1 1 1	1 1 1	
CPI	(8)	0 0 B B	1 1 1 B B B	1 0 0 B B B	
RLC	(5)	0 0	0 0 0	0 1 0	Rotate the content of the accumulator left.
RRC	(5)	0 0	0 0 1	0 1 0	Rotate the content of the accumulator right.
RAL	(5)	0 0	0 1 0	0 1 0	Rotate the content of the accumulator left through the carry.
RAR	(5)	0 0	0 1 1	0 1 0	Rotate the content of the accumulator right through the carry.

Program Counter and Stack Control Instructions

MNEMONIC	MINIMUM STATES REQUIRED	$D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
(4) JMP	(11)	0 1 B_2 B_2 X X	X X X B_2 B_2 B_2 B_3 B_3 B_3	1 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Unconditionally jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$.
(5) JNC, JNZ, JP, JPO	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
JC, JZ JM, JPE	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
CALL	(11)	0 1 B_2 B_2 X X	X X X B_2 B_2 B_2 B_3 B_3 B_3	1 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Unconditionally call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$. Save the current address (up one level in the stack).
CNC, CNZ, CP, CPO	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false, and save the current address (up one level in the stack.) Otherwise, execute the next instruction in sequence.
CC, CZ, CM, CPE	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
RET	(5)	0 0	X X X	1 1 1	Unconditionally return (down one level in the stack).
RNC, RNZ, RP, RPO	(3 or 5)	0 0	0 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
RC, RZ RM, RPE	(3 or 5)	0 0	1 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
RST	(5)	0 0	A A A	1 0 1	Call the subroutine at memory address AAA000 (up one level in the stack).

Input/Output Instructions

MNEMONIC	MINIMUM STATES REQUIRED	$D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
IN	(8)	0 1	0 0 M	M M 1	Read the content of the selected input port (MMM) into the accumulator.
OUT	(6)	0 1	R R M	M M 1	Write the content of the accumulator into the selected output port (RRMMM, RR ≠ 00).

Machine Instruction

MNEMONIC	MINIMUM STATES REQUIRED	$D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
HLT	(4) (4)	0 0 1 1	0 0 0 1 1 1	0 0 X 1 1 1	Enter the STOPPED state and remain there until interrupted.

NOTES:

(1) SSS = Source Index Register, DDD = Destination Index Register — These registers, r_i, are designated A(accumulator—000), B(001), C(010), D(011), E(100), H(101), L(110).

(2) Memory registers are addressed by the contents of registers H & L.

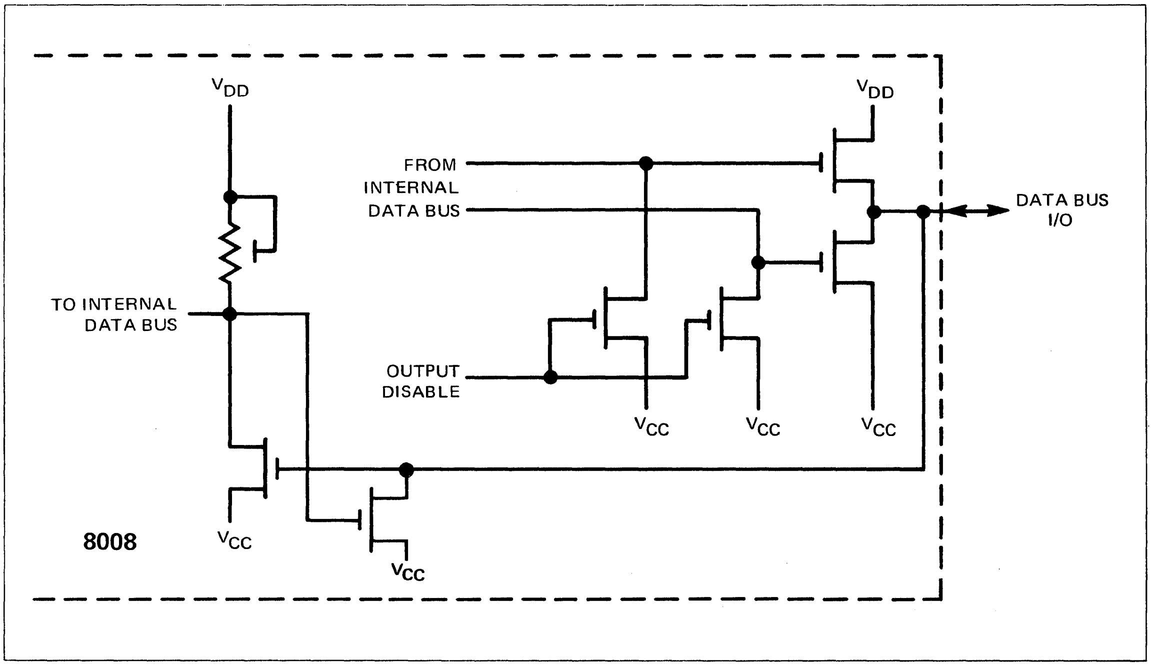
(3) Additional bytes of instruction are designated by BBBBBBBB.

(4) X = "Don't Care".

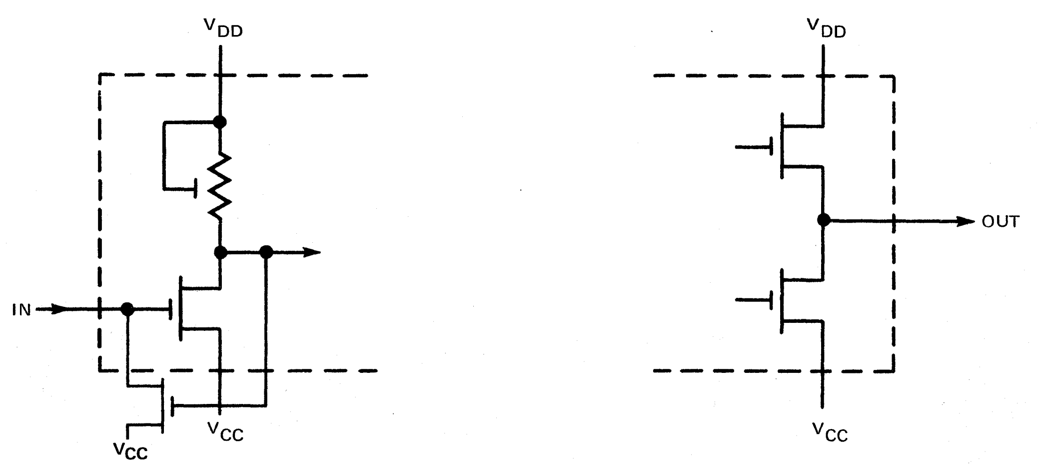
(5) Flag flip-flops are defined by C_4C_3: carry (00-overflow or underflow), zero (01-result is zero), sign (10-MSB of result is "1"), parity (11-parity is even).

ELECTRICAL SPECIFICATION

The following pages provide the electrical characteristics for the 8008. All of the inputs are TTL compatible, but input pull-up resistors are recommended to insure proper V_{IH} levels. All outputs are low-power TTL compatible. The transfer of data to and from the data bus is controlled by the CPU. During both the WAIT and STOPPED states the data bus output buffers are disabled and the data bus is floating.



Data Bus I/O Buffer



Input Buffer
(ϕ_1, ϕ_2, RDY, INT)

Output Buffer
(SYNC, S_0, S_1, S_2)

I/O Circuitry

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias	0°C to +70°C
Storage Temperature	−55°C to +150°C
Input Voltages and Supply Voltage With Respect to V_{CC}	+0.5 to −20V
Power Dissipation	1.0 W @ 25°C

*COMMENT

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

D.C. AND OPERATING CHARACTERISTICS

T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5% unless otherwise specified. Logic "1" is defined as the more positive level (V_{IH}, V_{OH}). Logic "0" is defined as the more negative level (V_{IL}, V_{OL}).

SYMBOL	PARAMETER	LIMITS MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
I_{DD}	AVERAGE SUPPLY CURRENT–OUTPUTS LOADED*		30	60	mA	T_A = 25°C
I_{LI}	INPUT LEAKAGE CURRENT			10	μA	V_{IN} = 0V
V_{IL}	INPUT LOW VOLTAGE (INCLUDING CLOCKS)	V_{DD}		V_{CC}−4.2	V	
V_{IH}	INPUT HIGH VOLTAGE (INCLUDING CLOCKS)	V_{CC}−1.5		V_{CC}+0.3	V	
V_{OL}	OUTPUT LOW VOLTAGE			0.4	V	I_{OL} = 0.44mA, C_L = 200 pF
V_{OH}	OUTPUT HIGH VOLTAGE	V_{CC}−1.5			V	I_{OH} = 0.2mA

*Measurements are made while the 8008 is executing a typical sequence of instructions. The test load is selected such that at V_{OL} = 0.4V, I_{OL} = 0.44mA on each output.

A.C. CHARACTERISTICS

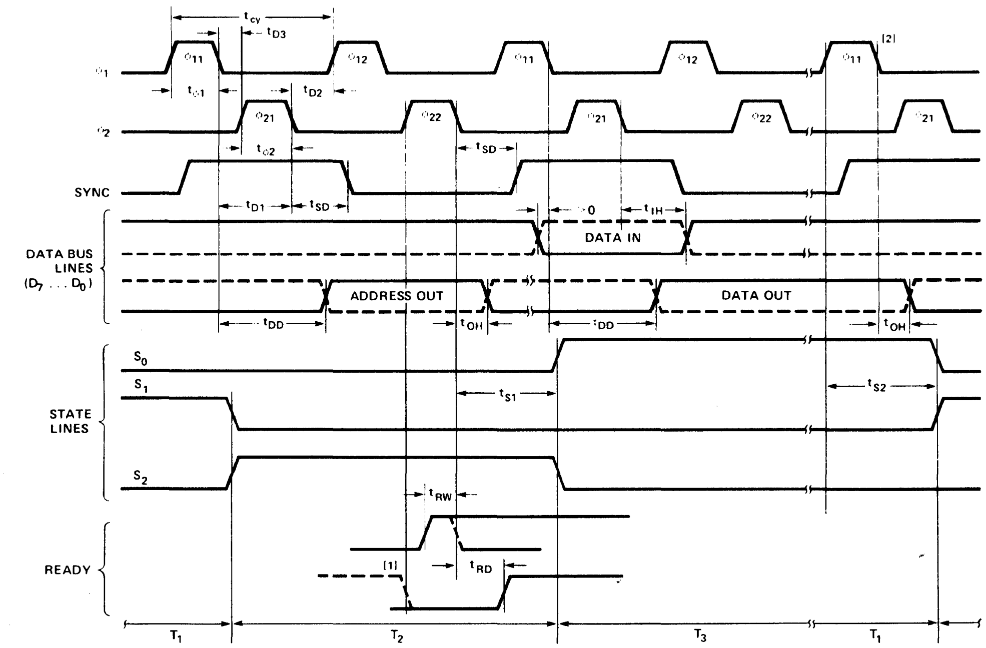
T_A = 0°C to 70°C; V_{CC} = +5V ±5%, V_{DD} = −9V ±5%. All measurements are referenced to 1.5V levels.

SYMBOL	PARAMETER	8008 LIMITS MIN.	8008 LIMITS MAX.	8008-1 LIMITS MIN.	8008-1 LIMITS MAX.	UNIT	TEST CONDITIONS
t_{CY}	CLOCK PERIOD	2	3	1.25	3	μs	t_R, t_F = 50ns
t_R, t_F	CLOCK RISE AND FALL TIMES		50		50	ns	
$t_{\phi 1}$	PULSE WIDTH OF ϕ_1	.70		.35		μs	
$t_{\phi 2}$	PULSE WIDTH OF ϕ_2	.55		.35		μs	
t_{D1}	CLOCK DELAY FROM FALLING EDGE OF ϕ_1 TO FALLING EDGE OF ϕ_2	.90	1.1		1.1	μs	
t_{D2}	CLOCK DELAY FROM ϕ_2 TO ϕ_1	.40		.35		μs	
t_{D3}	CLOCK DELAY FROM ϕ_1 TO ϕ_2	.20		.20		μs	
t_{DD}	DATA OUT DELAY		1.0		1.0	μs	C_L = 100pF
t_{OH}	HOLD TIME FOR DATA BUS OUT	.10		.10		μs	
t_{IH}	HOLD TIME FOR DATA IN	[1]		[1]		μs	
t_{SD}	SYNC OUT DELAY		.70		.70	μs	C_L = 100pF
t_{S1}	STATE OUT DELAY (ALL STATES EXCEPT T1 AND T1I)[2]		1.1		1.1	μs	C_L = 100pF
t_{S2}	STATE OUT DELAY (STATES T1 AND T1I)		1.0		1.0	μs	C_L = 100pF
t_{RW}	PULSE WIDTH OF READY DURING ϕ_{22} TO ENTER T3 STATE	.35		.35		μs	
t_{RD}	READY DELAY TO ENTER WAIT STATE	.20		.20		μs	

[1] t_{IH} MIN ≥ t_{SD}

[2] If the INTERRUPT is not used, all states have the same output delay, t_{S1}.

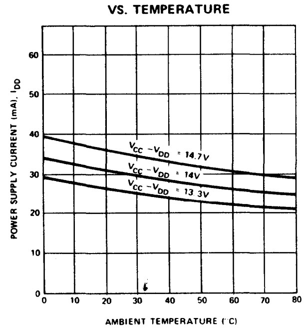
TIMING DIAGRAM



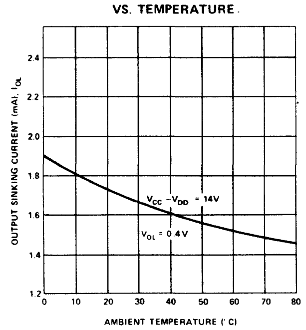
Notes: 1. READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.
2. INTERRUPT line must not change levels within 200ns (max.) of falling edge of ϕ_1.

TYPICAL D. C. CHARACTERISTICS

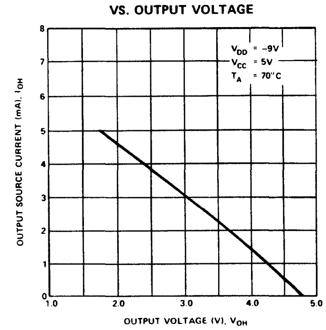
POWER SUPPLY CURRENT VS. TEMPERATURE



OUTPUT SINKING CURRENT VS. TEMPERATURE

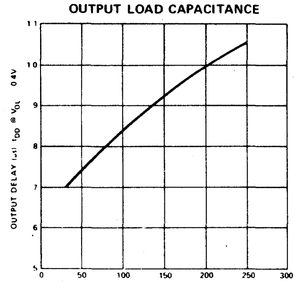


OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



TYPICAL A. C. CHARACTERISTICS

DATA OUT DELAY VS. OUTPUT LOAD CAPACITANCE



CAPACITANCE f = 1MHz; T_A = 25°C; Unmeasured Pins Grounded

SYMBOL	TEST	LIMIT (pF)	
		TYP.	MAX.
C_{IN}	INPUT CAPACITANCE	5	10
C_{DB}	DATA BUS I/O CAPACITANCE	5	10
C_{OUT}	OUTPUT CAPACITANCE	5	10

2048 BIT FULLY DECODED READ ONLY MEMORY

- **Erasable and Field Programmable (1702A--S714)**
- **Field Programmable (1602A--S714)**
- **2 μs Access Time**
- **Compatible with Intel's MCS-8™**

The 1602A/1702A--S714 is ideally suited for use with Intel's MCS-8 Micro Computer Set. It may also be used with systems requiring 2μsec access time.

Absolute Maximum Ratings*

Ambient Temperature Under Bias ...... −0°C to +85°C
Storage Temperature .... −65°C to +125°C
Soldering Temperature of Leads (10 sec) .... +300°C
Power Dissipation .... 2 Watts
Normal 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	16	mA	V_{GG}=V_{CC}, $\overline{CS}$=V_{CC}−2, I_{OL} = 0.0mA, T_A = 25°C	
I_{DD1}	Power Supply Current		35	60	mA	$\overline{CS}$=V_{CC} −2, I_{OL} =0.0mA, T_A = 25°C	Continuous Operation
I_{DD2}	Power Supply Current		32	55	mA	$\overline{CS}$=0.0, I_{OL} =0.0mA, T_A = 25°C	Continuous Operation
I_{DD3}	Power Supply Current		38.5	70	mA	$\overline{CS}$=V_{CC} −2, I_{OL} =0.0mA, T_A = 0°C	Continuous Operation
I_{CF1}	Output Clamp Current		8	19	mA	V_{OUT} = −1.0V, T_A = 0°C	Continuous Operation
I_{CF2}	Output Clamp Current			18	mA	V_{OUT} = −1.0V, T_A = 25°C	Continuous Operation
I_{GG}	Gate Supply Current			10	μA		
V_{IL1}	Input Low Voltage for TTL Interface	−1		0.55	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			mA	V_{OUT} = 0.45V	
I_{OH}	Output Source Current	−350			μA	V_{OUT} = 0.0V	
V_{OL}	Output Low Voltage		−.7	0.45	V	I_{OL} = 1.6mA	
V_{OH}	Output High Voltage	3.5	4.5		V	I_{OH} = −100 μA	

Note 1: In the programming mode, the data inputs 1−8 are pins 4−11 respectively.
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.
Note 3: Typical values are at nominal voltages and T_A = 25°C.

CAUTION: The 1702A is a quartz-lid device. The device environment should not exceed that to which a plastic package would be subjected.

A.C. Characteristics

$T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$, $V_{DD} = -9V \pm 5\%$, $V_{GG} = -9V \pm 5\%$, unless otherwise noted.

SYMBOL	TEST	MINIMUM	TYPICAL	MAXIMUM	UNIT
Freq.	Repetition Rate			0.5	MHz
t_{OH}	Previous read data valid			0	ns
t_{ACC}	Address to output delay		.700	2	μs
t_{DVGG}	Clocked V_{GG} set up	0.5			μs
t_{CS}	Chip select delay			1.1	μs
t_{CO}	Output delay from $\overline{CS}$			0.9	μs
t_{OD}	Output deselect			0.6	μs
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.

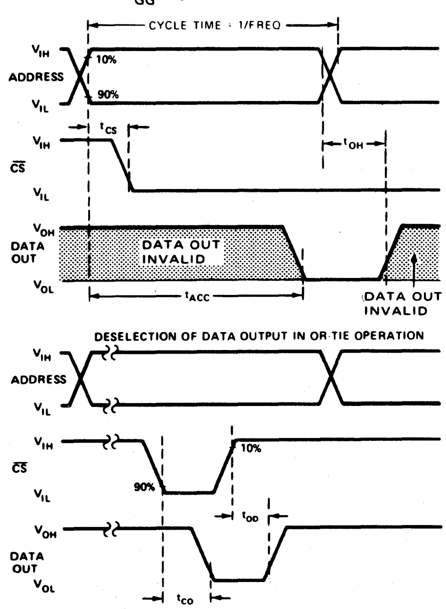
Switching Characteristics

Conditions of Test:

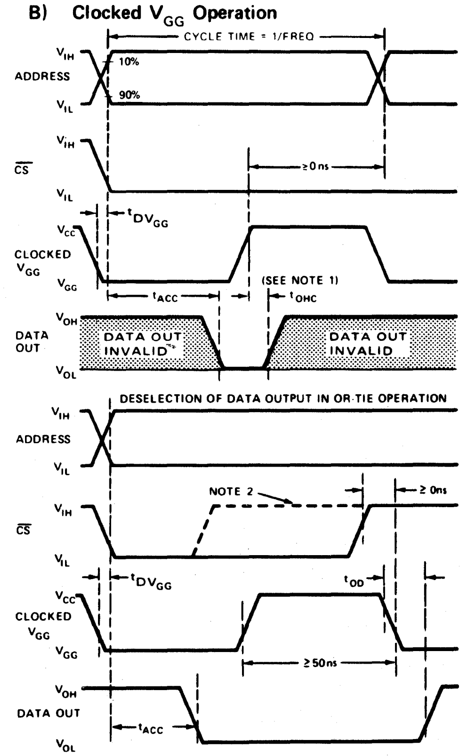
Input pulse amplitudes: 0 to 4V; t_R, $t_F \leq 50$ ns

Output load is 1 TTL gate; measurements made at output of TTL gate ($t_{PD} \leq 15$ 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}.

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

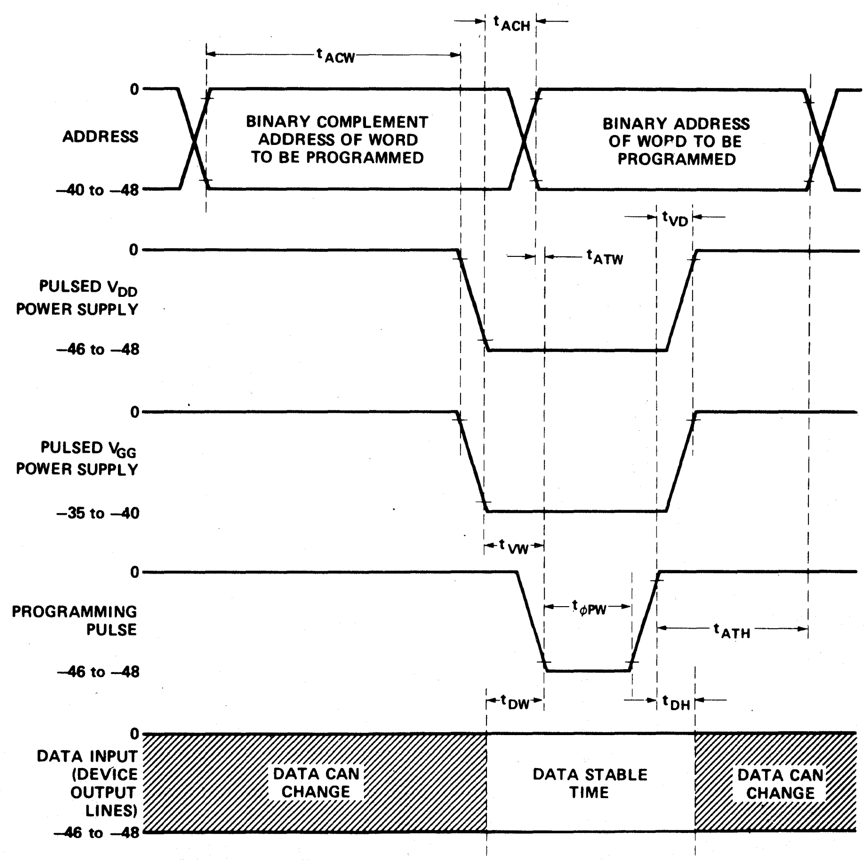
PROGRAM OPERATION

Conditions of Test:

Input pulse rise and fall times $\leq 1\mu sec$

$\overline{CS} = 0V$

PROGRAM WAVEFORMS



Programming Operation

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)

Ordering Information

1. The erasable and field programmable ROM should be ordered as the 1702A/S714.
2. The field programmable ROM should be ordered as the 1602A/S714.

Note: Intel's liability shall be limited to replacing any unit which fails to program as desired.

1024 BIT FULLY DECODED STATIC MOS RANDOM ACCESS MEMORY

- Single +5 Volts Supply Voltage
- Directly TTL Compatible — All Inputs and Output
- Static MOS — No Clocks or Refreshing Required
- Low Power — Typically 150 mW
- Access Time — Typically 500 nsec
- 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 2102 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.

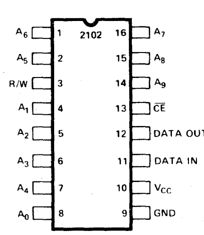
The 2102 is designed for memory applications where high performance, low cost, large bit storage, and simple interfacing are important design objectives.

It is directly TTL compatible in all respects: inputs, output, and a single +5 volt supply. A separate chip enable ($\overline{CE}$) lead allows easy selection of an individual package when outputs are OR-tied.

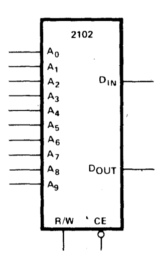
The Intel 2102 is fabricated with N-channel silicon gate technology. This technology allows the design and production of high performance easy to use MOS circuits and provides a higher functional density on a monolithic chip than either conventional MOS technology or P-channel silicon gate technology.

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

PIN CONFIGURATION



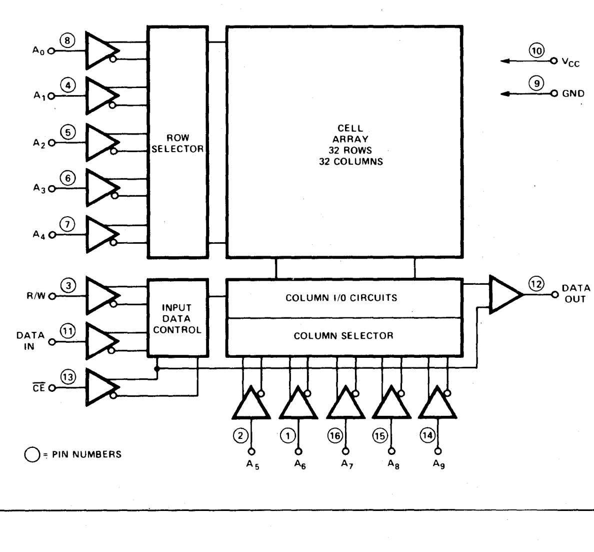
LOGIC SYMBOL



PIN NAMES

D_{IN}	DATA INPUT	$\overline{CE}$	CHIP ENABLE
$A_0 - A_9$	ADDRESS INPUTS	D_{OUT}	DATA OUTPUT
R/W	READ/WRITE INPUT	V_{CC}	POWER (+5V)

BLOCK DIAGRAM



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

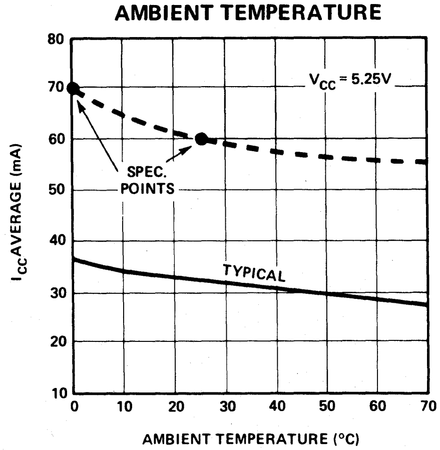
T_A = 0°C to +70°C, V_{CC} = 5V ±5% unless otherwise specified

SYMBOL	PARAMETER	LIMITS MIN.	TYP.(1)	MAX.	UNIT	TEST CONDITIONS
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		Vcc	V	
V_{OL}	OUTPUT "LOW" VOLTAGE			+0.45	V	I_{OL} = 1.9mA
V_{OH}	OUTPUT "HIGH" VOLTAGE	2.2			V	I_{OH} = –100μA

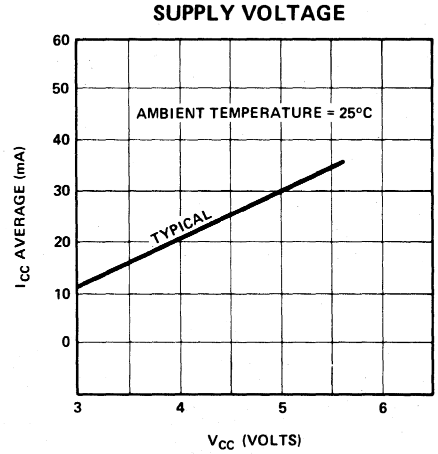
(1) Typical values are for T_A = 25°C and nominal supply voltage.

Typical D.C. Characteristics

POWER SUPPLY CURRENT VS. AMBIENT TEMPERATURE

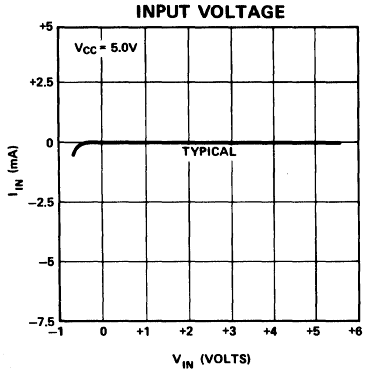


POWER SUPPLY CURRENT VS. SUPPLY VOLTAGE

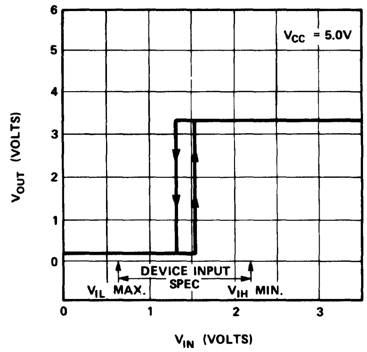


Typical D. C. Characteristics

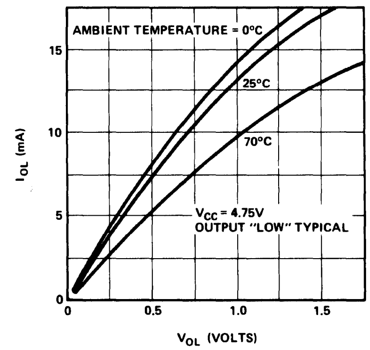
INPUT CURRENT VS. INPUT VOLTAGE



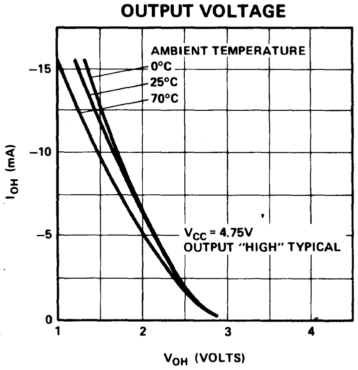
EFFECTIVE INPUT CHARACTERISTIC



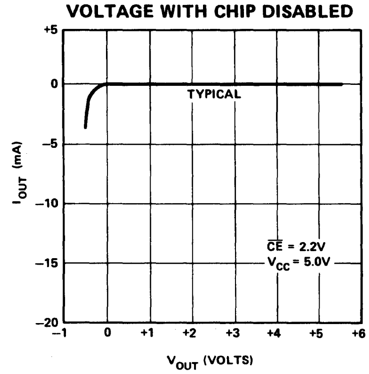
OUTPUT SINK CURRENT VS. OUTPUT VOLTAGE



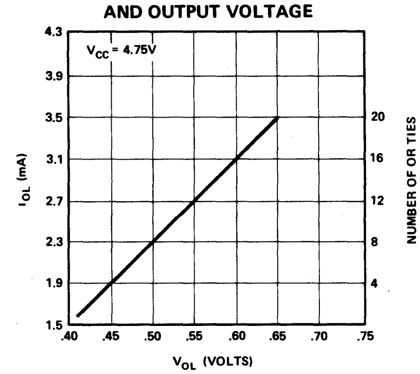
OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT CURRENT VS. OUTPUT VOLTAGE WITH CHIP DISABLED

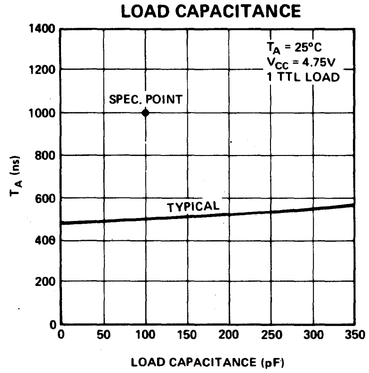


RELATIONSHIP BETWEEN OUTPUT SINK CURRENT, NUMBER OF OR-TIES, AND OUTPUT VOLTAGE

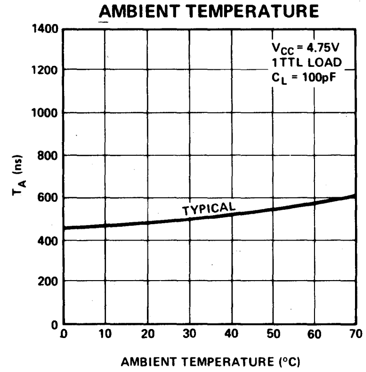


Typical A. C. Characteristics

ACCESS TIME VS. LOAD CAPACITANCE



ACCESS TIME VS. AMBIENT TEMPERATURE



A. C. Characteristics T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

SYMBOL	PARAMETER	LIMITS MIN.	TYP.[1]	MAX.	UNIT
READ CYCLE					
t_{RC}	READ CYCLE	1000			ns
t_A	ACCESS TIME		500	1000	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	1000			ns
t_{AW}	ADDRESS TO WRITE SETUP TIME	200			ns
t_{WP}	WRITE PULSE WIDTH	750			ns
t_{WR}	WRITE RECOVERY TIME	50			ns
t_{DW}	DATA SETUP TIME	800			ns
t_{DH}	DATA HOLD TIME	100			ns
t_{CW}	CHIP ENABLE TO WRITE SETUP TIME	900			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

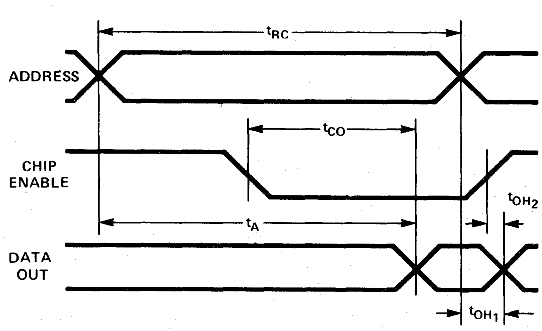
Output Load: 1 TTL Gate and C_L = 100 pF

Capacitance T_A = 25°C, f = 1 MHz

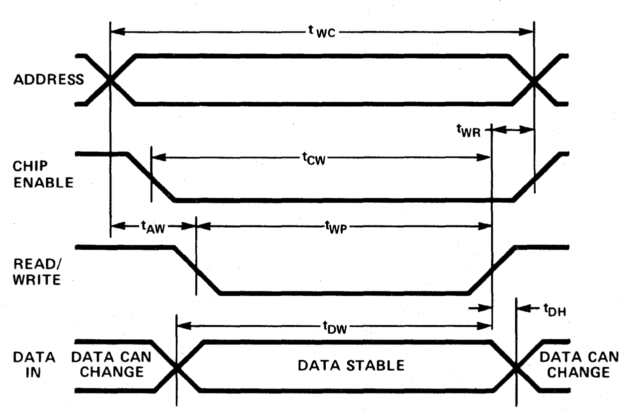
SYMBOL	TEST	LIMITS (pF) TYP.	MAX.
C_{IN}	INPUT CAPACITANCE (ALL INPUT PINS) V_{IN} = 0V	3	5
C_{OUT}	OUTPUT CAPACITANCE V_{OUT} = 0V	7	10

Waveforms

READ CYCLE



WRITE CYCLE



3205 HIGH SPEED 1 OUT OF 8 BINARY DECODER
3404 HIGH SPEED 6-BIT LATCH

- **18 ns max. Delay Over 0° C to 75° C Temperature -- 3205**
- **12 ns max. Data to Output Delay Over 0° C to 75° C Temperature -- 3404**
- **Directly Compatible with DTL and 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.**
- **Simple Expansion -- Enable Inputs.**

3205

The 3205 decoder can be used for expansion of systems which utilize memory components with active low chip select input. When the 3205 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 3205 allow easy memory expansion. For very large memory systems, 3205 decoders can be cascaded such that each decoder can drive 8 other decoders for arbitrary memory expansions.

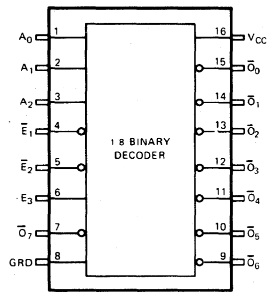
3404

The Intel 3404 contains six high speed latches organized as independent 4-bit and 2-bit latches. They are designed for use as memory data registers, address registers, or other storage elements. The latches act as high speed inverters when the "Write" input is "low".

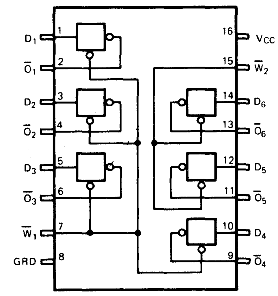
The Intel 3404 is packaged in a standard 16-pin dual-in-line package; and its performance is specified over the temperature range of 0° C to +75° C, ambient. The use of Schottky barrier diode clamped transistors to obtain fast switching speeds results in higher performance than equivalent devices made with a gold diffusion process.

PIN CONFIGURATION

3205



3404



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%

3205, 3404

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

3205 ONLY

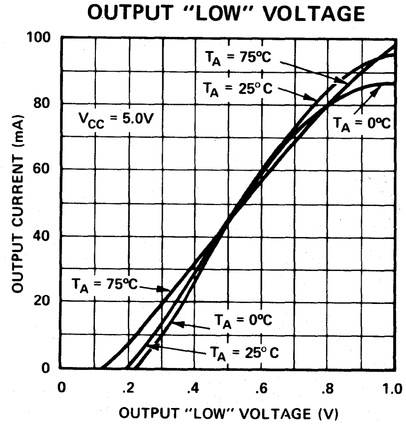
I_{CC}	POWER SUPPLY CURRENT		70	mA	V_{CC} = 5.25V

3404 ONLY

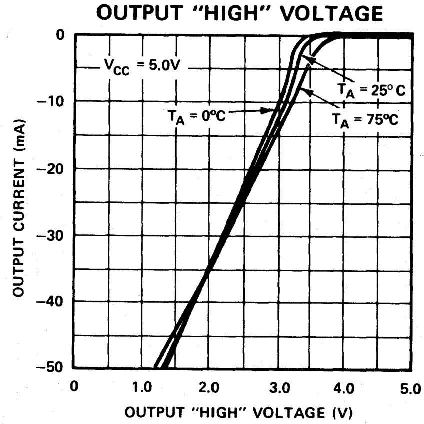
I_{CC}	POWER SUPPLY CURRENT		75	mA	V_{CC}=5.25V
I_{FW1}	WRITE ENABLE LOAD CURRENT PIN 7		−1.00	mA	V_{CC}=5.25V, V_W=0.45V
I_{FW2}	WRITE ENABLE LOAD CURRENT PIN 15		−0.50	mA	V_{CC}=5.25V, V_W=0.45V
I_{RW}	WRITE ENABLE LEAKAGE CURRENT		10	μA	V_R=5.25V

Typical Characteristics

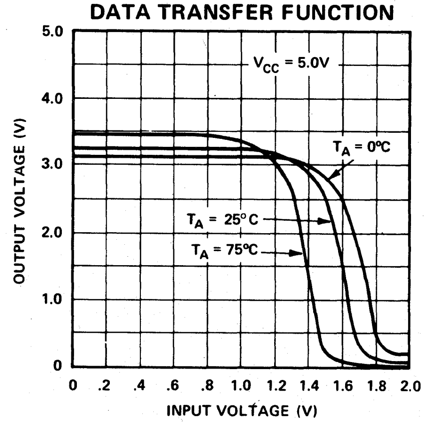
OUTPUT CURRENT VS. OUTPUT "LOW" VOLTAGE



OUTPUT CURRENT VS. OUTPUT "HIGH" VOLTAGE



DATA TRANSFER FUNCTION



3205-HIGH SPEED 1 OUT OF 8 BINARY DECODER
Switching Characteristics

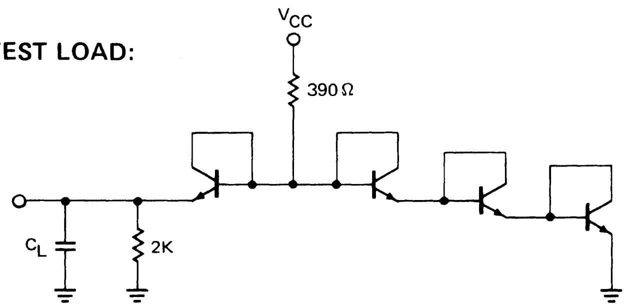
CONDITIONS OF TEST:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec between 1V and 2V

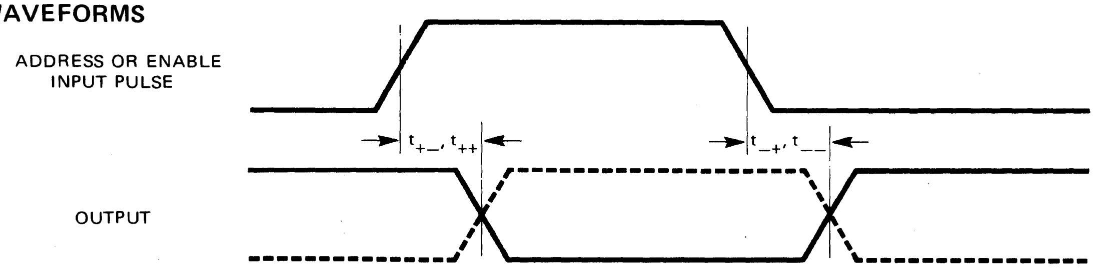
Measurements are made at 1.5V

TEST LOAD:



All Transistors 2N2369 or Equivalent. C_L = 30 pF

TEST WAVEFORMS



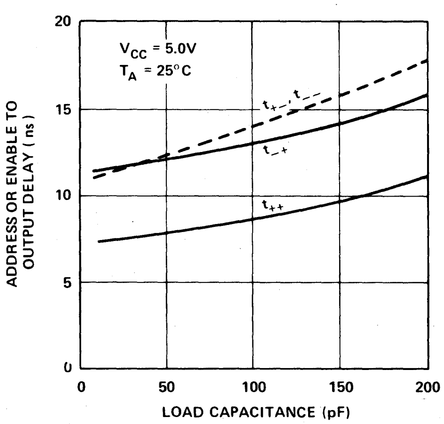
A.C. Characteristics T_A = 0°C to +75°C, V_{CC} = 5.0V ±5% unless otherwise specified.

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	P3205	4(typ.)	pF	f = 1 MHz, V_{CC} = 0V
		C3205	5(typ.)	pF	V_{BIAS} = 2.0V, T_A = 25°C

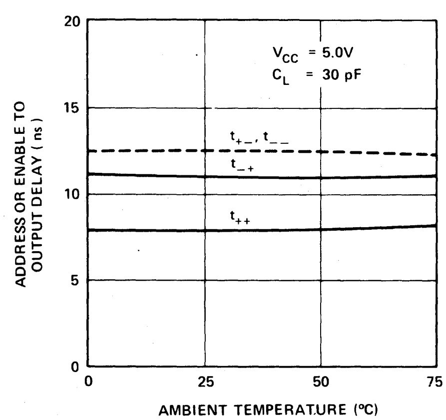
1. This parameter is periodically sampled and is not 100% tested.

Typical Characteristics

ADDRESS OR ENABLE TO OUTPUT DELAY VS. LOAD CAPACITANCE



ADDRESS OR ENABLE TO OUTPUT DELAY VS. AMBIENT TEMPERATURE



3404 - 6-BIT LATCH
Switching Characteristics

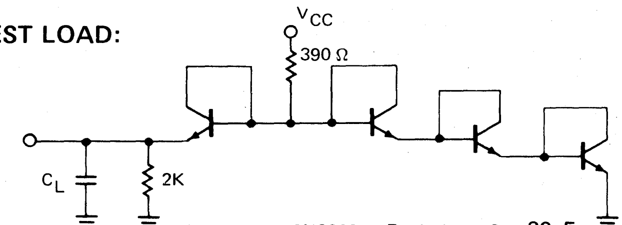
CONDITIONS OF TEST:

Input pulse amplitudes: 2.5V

Input rise and fall times: 5 nsec between 1V and 2V

Measurements are made at 1.5V

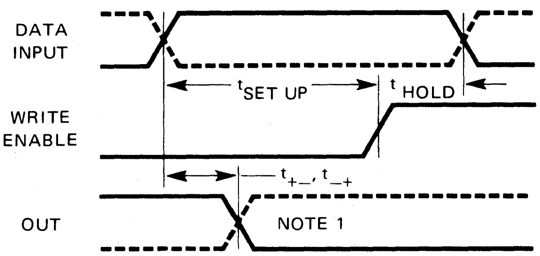
TEST LOAD:



All Transistors 2N2369 or Equivalent. $C_L = 30pF$

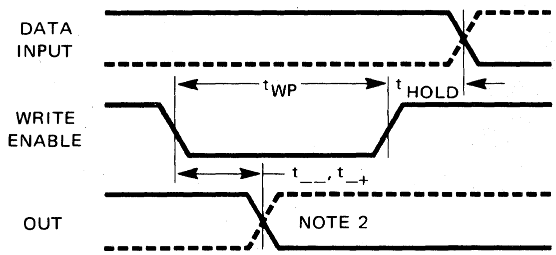
TEST WAVEFORMS

MEASUREMENT FOR DATA DELAY



NOTE 1: Output Data is valid after t_{+-}, t_{-+}

MEASUREMENT FOR WRITE ENABLE DELAY



NOTE 2: Output Data is valid after t_{--}, t_{-+}

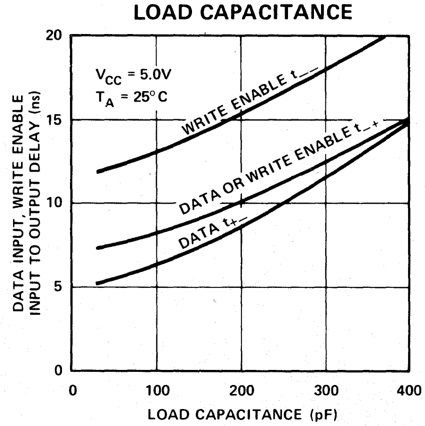
A.C. Characteristics $T_A = 0°C$ to $+75°C$, $V_{CC} = 5.0V \pm 5\%$; unless otherwise specified.

SYMBOL	PARAMETER		LIMITS MIN.	TYP.	MAX.	UNIT	TEST CONDITIONS
t_{+-}, t_{-+}	DATA TO OUTPUT DELAY				12	ns	
t_{--}, t_{-+}	WRITE ENABLE TO OUTPUT DELAY				17	ns	
$t_{SET\ UP}$	TIME DATA MUST BE PRESENT BEFORE RISING EDGE OF WRITE ENABLE		12			ns	
t_{HOLD}	TIME DATA MUST REMAIN AFTER RISING EDGE OF WRITE ENABLE		8			ns	
t_{WP}	WRITE ENABLE PULSE WIDTH		15			ns	
C_{IND}(3)	DATA INPUT CAPACITANCE	P3404		4		pF	f = 1 MHz, V_{CC} = 0V
		C3404		5		pF	V_{BIAS} = 2.0V, T_A = 25°C
C_{INW}(3)	WRITE ENABLE CAPACITANCE	P3404		7		pF	f = 1 MHz, V_{CC} = 0V
		C3404		8		pF	V_{BIAS} = 2.0V, T_A = 25°C

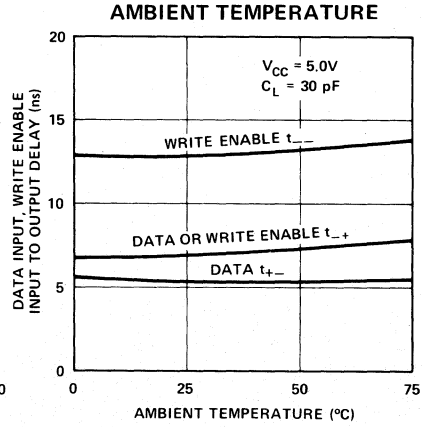
NOTE 3: This parameter is periodically sampled and is not 100% tested.

Typical Characteristics

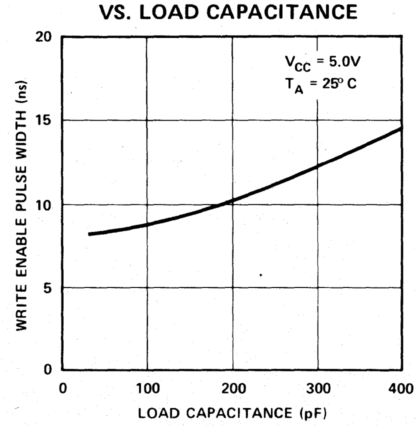
DATA INPUT, WRITE ENABLE TO OUTPUT DELAY VS. LOAD CAPACITANCE



DATA INPUT, WRITE ENABLE TO OUTPUT DELAY VS. AMBIENT TEMPERATURE



WRITE ENABLE PULSE WIDTH VS. LOAD CAPACITANCE



APPENDIX C INSTRUCTION MACHINE CODES

In order to help the programmer examine memory when debugging programs, this appendix provides the assembly language instruction represented by each of the 256 possible instruction code bytes.

Where an instruction occupies two bytes (immediate instruction) or three bytes (jump instruction), only the first (code) byte is given.

DEC	OCTAL	HEX	MNEMONIC	COMMENT
0	000	00	HLT	
1	001	01	—	
2	002	02	RLC	
3	003	03	RNC	
4	004	04	ADI EXP	
5	005	05	RST EXP	
6	006	06	MVI A, EXP	
7	007	07	RET	
8	010	08	INR B	
9	011	09	DCR B	
10	012	0A	RRC	
11	013	0B	RNZ	
12	014	0C	ACI EXP	
13	015	0D	RST EXP	EXP =1
14	016	0E	MVI B, EXP	
15	017	0F	—	
16	020	10	INR C	
17	021	11	DCR C	
18	022	12	RAL	
19	023	13	RP	
20	024	14	SUI EXP	
21	025	15	RST EXP	EXP =2
22	026	16	MVI C, EXP	
23	027	17	—	
24	030	18	INR D	
25	031	19	DCR D	
26	032	1A	RAR	
27	033	1B	RPO	
28	034	1C	SBI EXP	
29	035	1D	RST EXP	EXP =3
30	036	1E	MVI D, EXP	
31	037	1F	—	

DEC	OCTAL	HEX	MNEMONIC	COMMENT
32	040	20	INR E	
33	041	21	DCR E	
34	042	22	—	
35	043	23	RC	
36	044	24	ANI EXP	
37	045	25	RST EXP	EXP =4
38	046	26	MVI E, EXP	
39	047	27	—	
40	050	28	INR H	
41	051	29	DCR H	
42	052	2A	—	
43	053	2B	RZ	
44	054	2C	XRI EXP	
45	055	2D	RST EXP	EXP =5
46	056	2E	MVI H, EXP	
47	057	2F	—	
48	060	30	INR L	
49	061	31	DCR L	
50	062	32	—	
51	063	33	RM	
52	064	34	ORI EXP	
53	065	35	RST EXP	EXP =6
54	066	36	MVI L, EXP	
55	067	37	—	
56	070	38	—	
57	071	39	—	
58	072	3A	—	
59	073	3B	RPE	
60	074	3C	CPI EXP	
61	075	3D	RST EXP	EXP =7
62	076	3E	MVI M, EXP	
63	077	3F	—	
64	100	40	JNC EXP	
65	101	41	IN EXP	EXP =0
66	102	42	CNC EXP	
67	103	43	IN EXP	EXP =1
68	104	44	JMP EXP	
69	105	45	IN EXP	EXP =2
70	106	46	CALL EXP	
71	107	47	IN EXP	EXP =3
72	110	48	JNZ EXP	
73	111	49	IN EXP	EXP =4
74	112	4A	CNZ EXP	
75	113	4B	IN EXP	EXP =5
76	114	4C	—	
77	115	4D	IN EXP	EXP =6
78	116	4E	—	
79	117	4F	IN EXP	EXP =7
80	120	50	JP EXP	
81	121	51	OUT EXP	EXP =8
82	122	52	CP EXP	
83	123	53	OUT EXP	EXP =9
84	124	54	—	
85	125	55	OUT EXP	EXP =10
86	126	56	—	
87	127	57	OUT EXP	EXP =11

DEC	OCTAL	HEX	MNEMONIC	COMMENT
88	130	58	JPO EXP	
89	131	59	OUT EXP	EXP =12
90	132	5A	CPO EXP	
91	133	5B	OUT EXP	EXP =13
92	134	5C	—	
93	135	5D	OUT EXP	EXP =14
94	136	5E	—	
95	137	5F	OUT EXP	EXP =15
96	140	60	JC EXP	
97	141	61	OUT EXP	EXP =16
98	142	62	CC EXP	
99	143	63	OUT EXP	EXP =17
100	144	64	—	
101	145	65	OUT EXP	EXP =18
102	146	66	—	
103	147	67	OUT EXP	EXP =19
104	150	68	JZ EXP	
105	151	69	OUT EXP	EXP =20
106	152	6A	CZ EXP	
107	153	6B	OUT EXP	EXP =21
108	154	6C	—	
109	155	6D	OUT EXP	EXP =22
110	156	6E	—	
111	157	6F	OUT EXP	EXP =23
112	160	70	JM EXP	
113	161	71	OUT EXP	EXP =24
114	162	72	CM EXP	
115	163	73	OUT EXP	EXP =25
116	164	74	—	
117	165	75	OUT EXP	EXP =26
118	166	76	—	
119	167	77	OUT EXP	EXP =27
120	170	78	JPE EXP	
121	171	79	OUT EXP	EXP =28
122	172	7A	CPE EXP	
123	173	7B	OUT EXP	EXP =29
124	174	7C	—	
125	175	7D	OUT EXP	EXP =30
126	176	7E	—	
127	177	7F	OUT EXP	EXP =31
128	200	80	ADD A	
129	201	81	ADD B	
130	202	82	ADD C	
131	203	83	ADD D	
132	204	84	ADD E	
133	205	85	ADD H	
134	206	86	ADD L	
135	207	87	ADD M	
136	210	88	ADC A	
137	211	89	ADC B	
138	212	8A	ADC D	
139	213	8B	ADC E	
140	214	8C	ADC E	
141	215	8D	ADC H	
142	216	8E	ADC L	
143	217	8F	ADC M	

DEC	OCTAL	HEX	MNEMONIC	COMMENT
144	220	90	SUB A	
145	221	91	SUB B	
146	222	92	SUB C	
147	223	93	SUB D	
148	224	94	SUB E	
149	225	95	SUB H	
150	226	96	SUB L	
151	227	97	SUB M	
152	230	98	SBB A	
153	231	99	SBB B	
154	232	9A	SBB C	
155	233	9B	SBB D	
156	234	9C	SBB E	
157	235	9D	SBB H	
158	236	9E	SBB L	
159	237	9F	SBB M	
160	240	A0	ANA A	
161	241	A1	ANA B	
162	242	A2	ANA C	
163	243	A3	ANA D	
164	244	A4	ANA E	
165	245	A5	ANA H	
166	246	A6	ANA L	
167	247	A7	ANA M	
168	250	A8	XRA A	
169	251	A9	XRA B	
170	252	AA	XRA C	
171	253	AB	XRA D	
172	254	AC	XRA E	
173	255	AD	XRA H	
174	256	AE	XRA L	
175	257	AF	XRA M	
176	260	B0	ORA A	
177	261	B1	ORA A	
178	262	B2	ORA C	
179	263	B3	ORA D	
180	264	B4	ORA E	
181	265	B5	ORA H	
182	266	B6	ORA L	
183	267	B7	ORA M	
184	270	B8	CMP A	
185	271	B9	CMP B	
186	272	BA	CMP C	
187	273	BB	CMP D	
188	274	BC	CMP E	
189	275	BD	CMP H	
190	276	BE	CMP L	
191	277	BF	CMP M	
192	300	C0	NOP	
193	301	C1	MOV A,B	
194	302	C2	MOV A,C	
195	303	C3	MOV A,D	
196	304	C4	MOV A,E	
197	305	C5	MOV A,H	
198	306	C6	MOV A,L	
199	307	C7	MOV A,M	

DEC	OCTAL	HEX	MNEMONIC	COMMENT
200	310	C8	MOV B,A	
201	311	C9	MOV B,B	
202	312	CA	MOV B,C	
203	313	CB	MOV B,D	
204	314	CC	MOV B,E	
205	315	CD	MOV B,H	
206	316	CE	MOV B,L	
207	317	CF	MOV B,M	
208	320	D0	MOV C,A	
209	321	D1	MOV C,B	
210	322	D2	MOV C,C	
211	323	D3	MOV C,D	
212	324	D4	MOV C,E	
213	325	D5	MOV C,H	
214	326	D6	MOV C,L	
215	327	D7	MOV C,M	
216	330	D8	MOV D,A	
217	331	D9	MOV D,B	
218	332	DA	MOV D,C	
219	333	DB	MOV D,D	
220	334	DC	MOV D,E	
221	335	DD	MOV D,H	
222	336	DE	MOV D,L	
223	337	DF	MOV D,M	
224	340	E0	MOV E,A	
225	341	E1	MOV E,B	
226	342	E2	MOV E,C	
227	343	E3	MOV E,D	
228	344	E4	MOV E,E	
229	345	E5	MOV E,H	
230	346	E6	MOV E,L	
231	347	E7	MOV E,M	
232	350	E8	MOV H,A	
233	351	E9	MOV H,B	
234	352	EA	MOV H,C	
235	353	EB	MOV H,D	
236	354	EC	MOV H,E	
237	355	ED	MOV H,H	
238	356	EE	MOV H,L	
239	357	EF	MOV H,M	
240	360	F0	MOV L,A	
241	361	F2	MOV L,B	
242	362	F2	MOV L,C	
243	363	F3	MOV L,D	
244	364	F4	MOV L,E	
245	365	F5	MOV L,H	
246	366	F6	MOV L,L	
247	367	F7	MOV L,M	
248	370	F8	MOV M,A	
249	371	F9	MOV M,B	
250	372	FA	MOV M,C	
251	373	FB	MOV M,D	
252	374	FC	MOV M,E	
253	375	FD	MOV M,H	
254	376	FE	MOV M,L	
255	377	FF	—	

APPENDIX D INSTRUCTION EXECUTION TIMES

The number of machine cycles needed to complete each INTELLEC 8/MOD 8 instruction is given in this appendix. The time required to complete an INTELLEC 8/MOD 8 machine cycle is 12.5 microseconds.

INSTRUCTION	CYCLES	
ACI	2	
ADD	1	; 2 cycles if memory is referenced
ADC	1	; 2 cycles if memory is referenced
ADI	2	
ANA	1	; 2 cycles if memory is referenced
ANI	2	
All CALL instructions	3	
CMP	1	; 2 cycles if memory is referenced
CPI	2	
DCR	1	
HLT	1	
IN	2	
INR	1	
All JUMP instructions	3	
MOV	1	; 2 cycles if memory is referenced
MVI	2	; 3 cycles if memory is referenced
ORA	1	; 2 cycles if memory is referenced
ORI	2	
OUT	2	
RAL	1	
RAR	1	
All RETURN instructions	1	
RLC	1	
RRC	1	
RST	1	
SBB	1	; 2 cycles if memory is referenced
SBI	2	
SUB	1	; 2 cycles if memory is referenced
SUI	1	
XRA	1	; 2 cycles if memory is referenced
XRI	2	

APPENDIX E 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 F 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®

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