intel® MCS-8™ User's Manual

April 1975

98-017B

INTEL SUPPORT MAKES SYSTEM BUILDING EASY.

The **MCS-8™** parallel 8-bit microcomputer set is designed for efficient handling of large volumes of data. It has interrupt capability, operates synchronously or asynchronously with external memory, and executes subroutines nested up to seven levels. The 8008 CPU, heart of the MCS-8, replaces 125 TTL packs. With it you can easily address up to 16k 8-bit words of ROM, RAM or shift registers. Using bank switching techniques, you can extend its memory indefinitely.

The **PL/M™ High Level Language** is an easy-to-learn, systems oriented language derived from IBM's PL/I by Intel for programming the MCS-8 and MCS-80 microcomputers. It gives the microcomputer programmer the same high level language advantages currently available in mini and large computers. By actual tests, PL/M programming and debugging requires less than 10% of the time needed for assembly language. The PL/M compiler is written in Fortran IV for time-share, and needs little or no alteration for most general purpose computers.

Intellec® 8 Development Systems provide flexible, inexpensive, and simplified methods for OEM product development. They use RAM for program storage instead of ROM, making program loading and modification easier. The Intellec features are:

- Display and Control Console
- Standard software package
- Expandable memory
- Expandable I/O
- TTY interface
- PROM programming capability

The Intellec control panel is used for system monitoring and debugging. These features and the many standard Intellec modules add up to faster turn around and reduced costs for your product development.

And, There's More . . . . .
Intel's **Microcomputer Systems Group** continues to develop new design aids that make microcomputer system-building easier. They will provide assistance in every phase of your program development.

For additional information:

Microcomputer Systems Group
INTEL Corporation
3065 Bowers Avenue
Santa Clara, California 95051
Phone (408) 246-7501

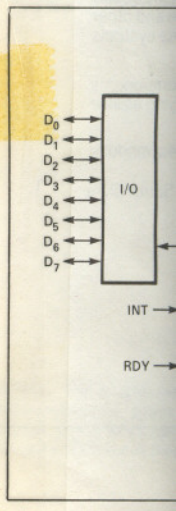
intel® delivers.

8

The 8008 is a co
having capacities
uses two leads fo
parallel arithmeti
instruction decoc

Featu

- **8-Bi Sing**
- **48 In Orie**
- **Com Deco Inclu**
- **Instr 12.5 with**
- **TTL Outp**
- **Can or sp mem**



8008

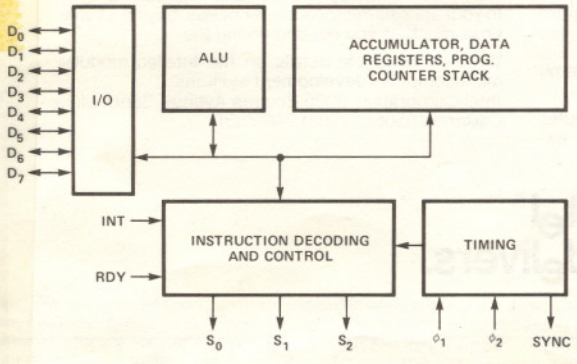
8 Bit Parallel Central Processor Unit

The 8008 is a complete computer system central processor unit which may be interfaced with memories having capacities up to 16K bytes. The processor communicates over an 8-bit data and address bus and uses two leads for internal control and four leads for external control. The CPU contains an 8-bit parallel arithmetic unit, a dynamic RAM (seven 8-bit data registers and an 8x14 stack), and complete instruction decoding and control logic. Clock Driver for 8008 is 8201

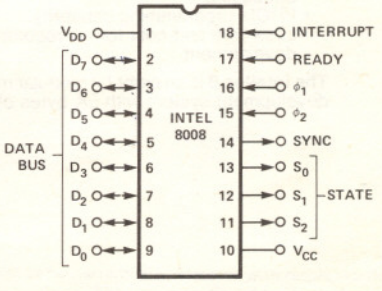
Features

- **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**

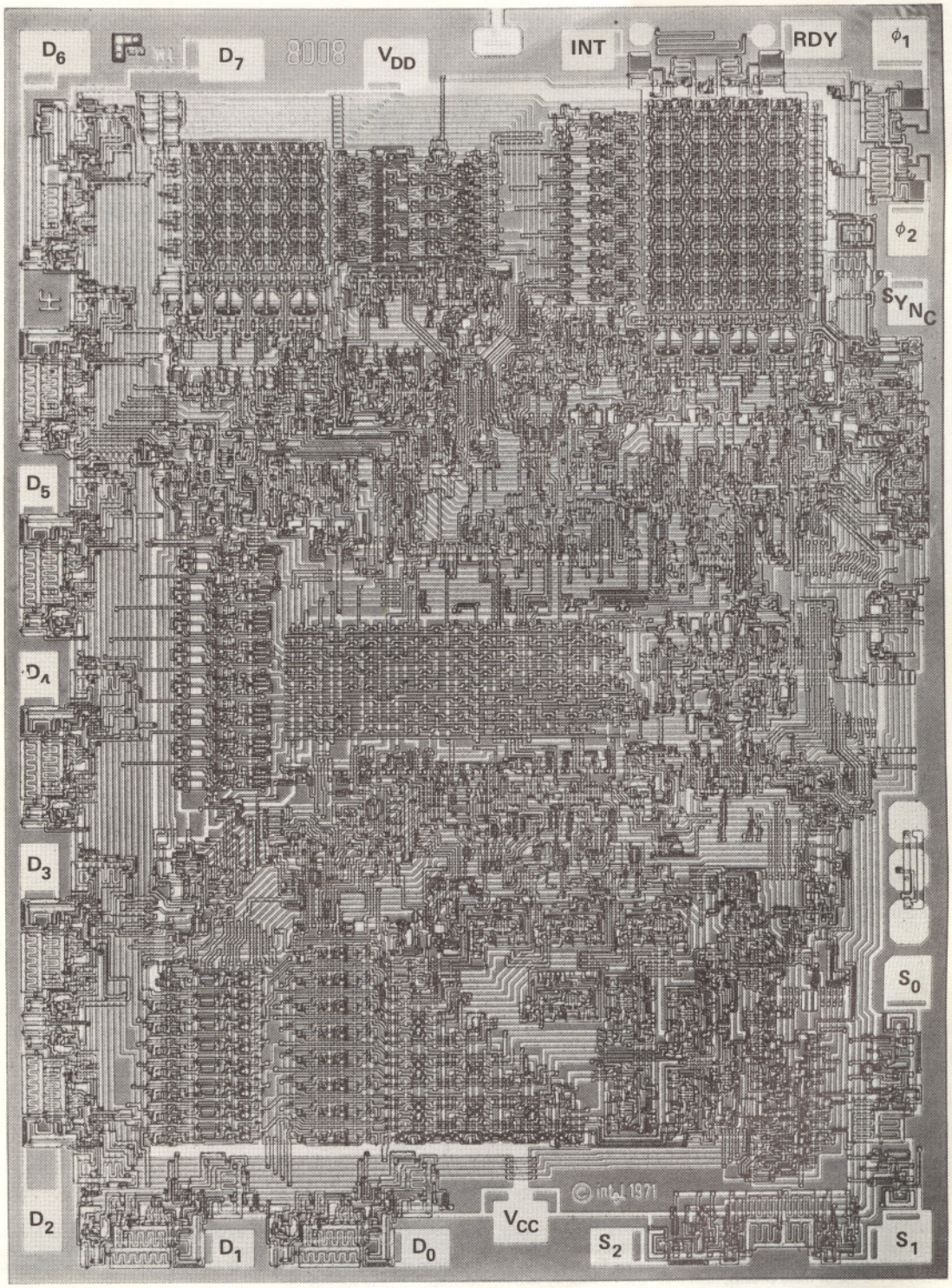
BLOCK DIAGRAM (BASIC)



PIN CONFIGURATION



CONTENTS



8008 Photomicrograph With Pin Designations

I. INTRODU

The 8008 is a si
system. A micro
standard semico
1302, 1602A, 1

The processor c
(READY and IN
of the data bus
CPU and extern

This CPU conta
bits, and an 8-b
operations. A m
to store progra
of memory (any

The control por
control, and log
structions use tw
8008 CPU execu
8008-1, executes
800kHz clock.

All inputs (inclu

The instruction
metic, and jump

The normal prog
control line. Thi
program.

The "READY"
of semiconducto

STATE and SYN

I. INTRODUCTION

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 8-bit words. Examples are INTEL's 1101, 1103, 2102 (RAMs), 1302, 1602A, 1702A (ROMs), 1404, 2405 (Shift Registers).

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, 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. Operating with a 500kHz clock, the 8008 CPU executes non-memory referencing instructions in 20 microseconds. A selected device, the 8008-1, executes non-memory referencing instructions in 12.5 microseconds when operating from an 800kHz clock.

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.

STATE and SYNC outputs indicate the state of the processor at any time in the instruction cycle.

II. PROCESSOR TIMING

The 8008 is a complete central processing unit intended for use in any arithmetic, control, or decision-making system. The internal organization is centered around an 8-bit internal data bus. All communication within the processor and with external components occurs on this bus in the form of 8-bit bytes of address, instruction or data. (Refer to the accompanying block diagram for the relationship of all of the internal elements of the processor to each other and to the data bus.) For the MCS-8™ a logic "1" is defined as a high level and a logic "0" is defined as a low level.

A. State Control Coding

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. A table of the binary state codes and the designated state names is shown below.

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

B. Timing

Typically, a machine 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 available for either sending or receiving data, the processor goes into the WAIT state. The accompanying diagram illustrates the processor activity during a single cycle.

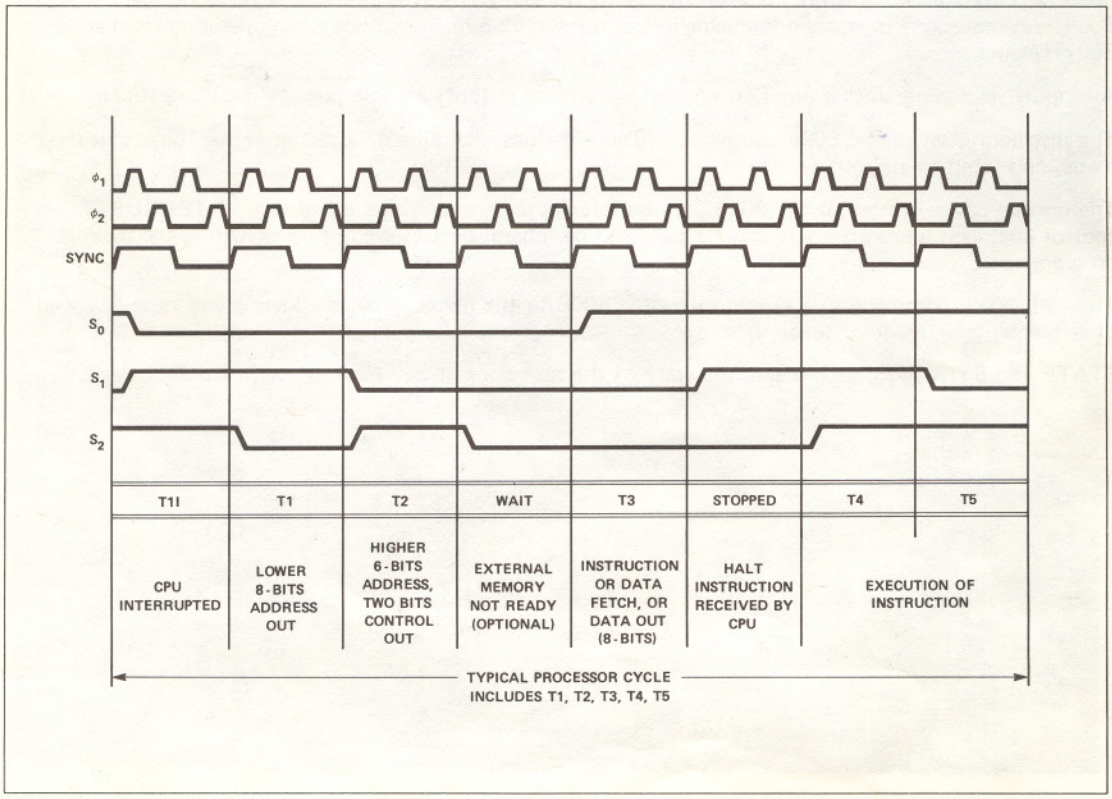


Figure 1. Basic 8008 Instruction Cycle

The receipt of an INTERRUPT is acknowledged by the T1I. When the processor has been interrupted, this state replaces T1. A READY is acknowledged by T3. The STOPPED state acknowledges the receipt of a HALT instruction.

Many of the instructions for the 8008 are multi-cycle and 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 external state transition is shown below. Note that the WAIT state and the STOPPED may be indefinite in length (each of these states will be 2n clock periods). The use of READY and INTERRUPT with regard to these states will be explained later.

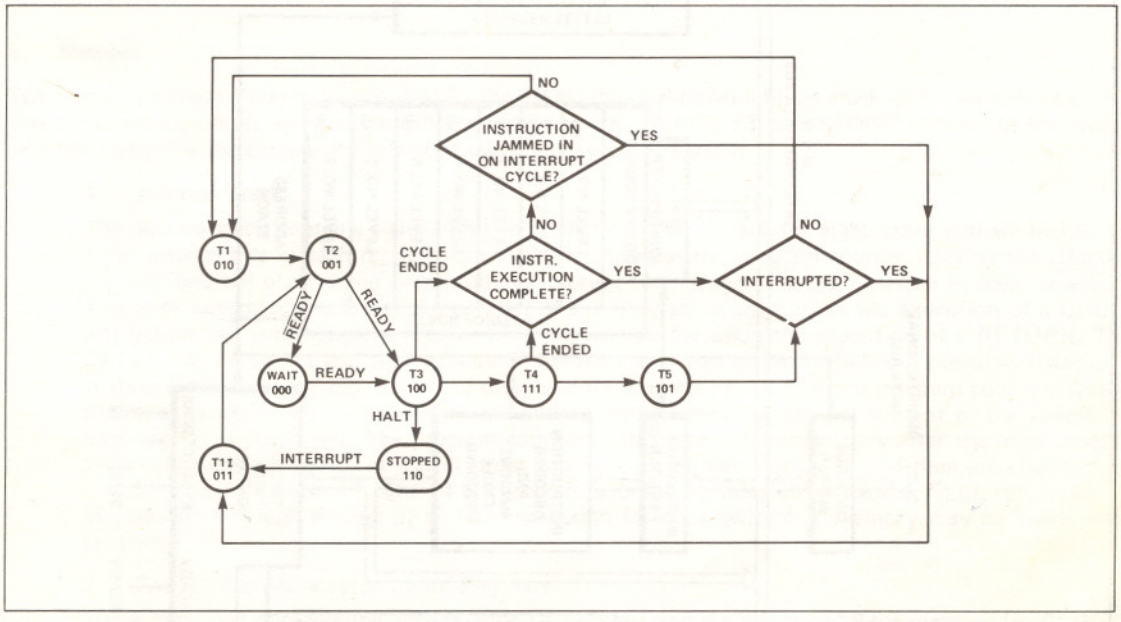


Figure 2. CPU State Transition Diagram

C. Cycle Control Coding

As previously noted, 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 for data reading (PCR), data writing (PCW), or I/O operations (PCC).

The cycle types are coded with two bits, D_6 and D_7, and are only present on the data bus during T2.

D_6	D_7	CYCLE	FUNCTION
0	0	PCI	Designates the address is for a memory read (first byte of instruction).
0	1	PCR	Designates the address is for a memory read data (additional bytes of instruction or data).
1	0	PCC	Designates the data as a command I/O operation.
1	1	PCW	Designates the address is for a memory write data.

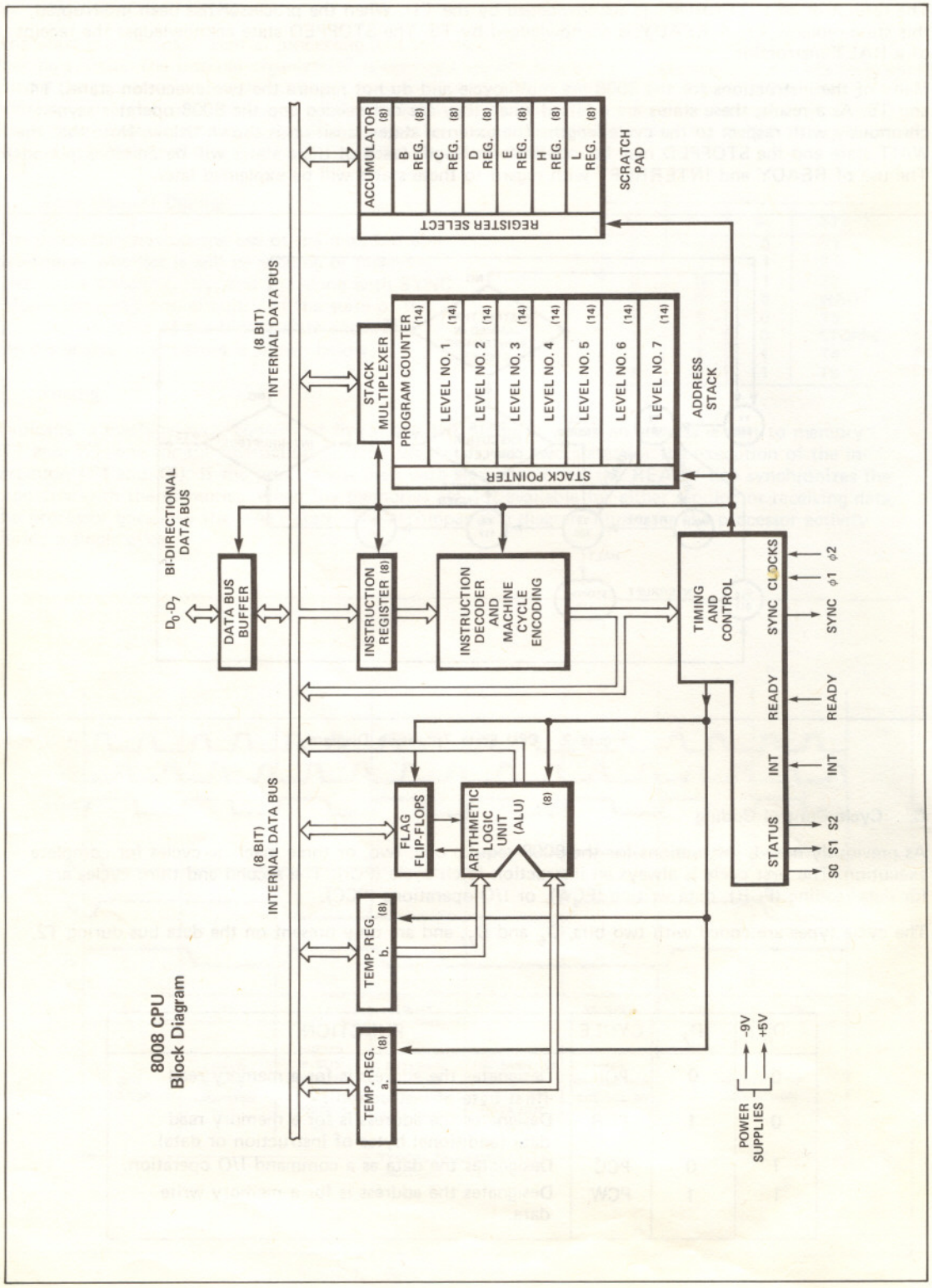


Figure 3. 8008 Block Diagram

III. BASIC FUNCTIONAL BLOCKS

The four basic functional blocks of this Intel processor are the instruction register, memory, arithmetic-logic unit, and I/O buffers. They communicate with each other over the internal 8-bit data bus.

A. Instruction Register and Control

The instruction register is the heart of all processor control. Instructions are fetched from memory, stored in the instruction register, and decoded for control of both the memories and the ALU. Since instruction executions do not all require the same number of states, the instruction decoder also controls the state transitions.

B. Memory

Two separate dynamic memories are used in the 8008, the pushdown address stack and a scratch pad. These internal memories are automatically refreshed by each WAIT, T3, and STOPPED state. In the worst case the memories are completely refreshed every eighty clock periods.

1. Address Stack

The address stack contains eight 14-bit registers providing storage for eight lower and six higher order address bits in each register. One register is used as the program counter (storing the effective address) and the other seven permit address storage for nesting of subroutines up to seven levels. The stack automatically stores the content of the program counter upon the execution of a CALL instruction and automatically restores the program counter upon the execution of a RETURN. The CALLs may be nested and the registers of the stack are used as last in/first out pushdown stack. A three-bit address pointer is used to designate the present location of the program counter. When the capacity of the stack is exceeded the address pointer recycles and the content of the lowest level register is destroyed. The program counter is incremented immediately after the lower order address bits are sent out. The higher order address bits are sent out at T2 and then incremented if a carry resulted from T1. The 14-bit program counter provides direct addressing of 16K bytes of memory. Through the use of an I/O instruction for bank switching, memory may be indefinitely expanded.

2. Scratch Pad Memory or Index Registers

The scratch pad contains the accumulator (A register) and six additional 8-bit registers (B, C, D, E, H, L). All arithmetic operations use the accumulator as one of the operands. All registers are independent and may be used for temporary storage. In the case of instructions which require operations with a register in external memory, scratch pad registers H & L provide indirect addressing capability; register L contains the eight lower order bits of address and register H contains the six higher order bits of address (in this case bit 6 and bit 7 are "don't cares").

C. Arithmetic/Logic Unit (ALU)

All arithmetic and logical operations (ADD, ADD with carry, SUBTRACT, SUBTRACT with borrow, AND, EXCLUSIVE OR, OR, COMPARE, INCREMENT, DECREMENT) are carried out in the 8-bit parallel arithmetic unit which includes carry-look-ahead logic. Two temporary resisters, register "a" and register "b", are used to store the accumulator and operand for ALU operations. In addition, they are used for temporary address and data storage during intra-processor transfers. Four control bits, carry flip-flop (c) , zero flip-flop (z) , sign flip-flop (s) , and parity flip-flop (p) , are set as the result of each arithmetic and logical operation. These bits provide conditional branching capability through CALL, JUMP, or RETURN on condition instructions. In addition, the carry bit provides the ability to do multiple precision binary arithmetic.

D. I/O Buffer

This buffer is the only link between the processor and the rest of the system. Each of the eight buffers is bi-directional and is under control of the instruction register and state timing. Each of the buffers is low power TTL compatible on the output and TTL compatible on the input.

Figure 3. 8008 Block Diagram

IV. BASIC INSTRUCTION SET

A. 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 B B	D D D B B B	1 1 0 B B B	Load index register r with data B . . . B.
MVI M	(9)	0 0 B B	1 1 1 B B B	1 1 0 B B B	Load memory register M with data 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 B . . . B to the accumulator. An overflow (carry) sets the carry flip-flop.
ADD M	(8)	1 0	0 0 0	1 1 1	
ADI	(8)	0 0 B B	0 0 0 B B B	1 0 0 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 B . . . B from the accumulator with carry. An overflow (carry) sets the carry flip-flop.
ADC M	(8)	1 0	0 0 1	1 1 1	
ACI	(8)	0 0 B B	0 0 1 B B B	1 0 0 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 data B . . . B from the accumulator. An underflow (borrow) sets the carry flip-flop.
SUB M	(8)	1 0	0 1 0	1 1 1	
SUI	(8)	0 0 B B	0 1 0 B B B	1 0 0 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 B . . . B from the accumulator with borrow. An underflow (borrow) sets the carry flip-flop.
SBB M	(8)	1 0	0 1 1	1 1 1	
SBI	(8)	0 0 B B	0 1 1 B B B	1 0 0 B B B	

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

(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 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 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 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 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 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 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

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

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 $C_4 C_3$: carry (00-overflow or underflow), zero (01-result is zero), sign (10-MSB of result is "1"), parity (11-parity is even).

B. Detailed 8008 Instruction Set

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

SINGL

Format:

[L

Note: REGM

CODE
INR
DCR

Condition bits

MOV IN

Format:

[L

Note: SRC and

CODE
MOV

Condition bits a

REGIST

Format:

[LA

CODE
ADD
ADC
SUB
SBB
ANA
XRA
ORA
CMP

Condition bits af

ADD, ADC, SUB

ANA, XRA, DR

CMP; Carry, sig

SINGLE REGISTER INSTRUCTIONS

Format:

[LABEL:] CODE REGM

Note: REGM ≠ A or M

CODE	DESCRIPTION
INR	(REGM) ← (REGM) +1 Increment register REGM
DCR	(REGM) ← (REGM) –1 Decrement register REGM

Condition bits affected: Zero, sign, parity

MOV INSTRUCTIONS

Format:

[LABEL:] MOV DST,SRC

Note: SRC and DST not both = M

CODE	DESCRIPTION
MOV	(DST) ← (SRC) Load register DST from register SRC

Condition bits affected: None

REGISTER OR MEMORY TO ACCUMULATOR INSTRUCTIONS

Format:

[LABEL:] CODE REGM

CODE	DESCRIPTION	
ADD	(A) ← (A) + (REGM)	Add REGM to accumulator
ADC	(A) ← (A) + (REGM) + (carry)	Add REGM plus carry bit to accumulator
SUB	(A) ← (A) – (REGM)	Subtract REGM from accumulator
SBB	(A) ← (A) – (REGM) – (carry)	Subtract REGM minus carry
ANA	(A) ← (A) AND (REGM)	AND accumulator with REGM
XRA	(A) ← (A) XOR (REGM)	Exclusive-OR accumulator with REGM
ORA	(A) ← (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) ≥ (REGM)

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

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

JUMP INS

Format:

[LAB

CODE
JMP
JC
JNC
JZ
JNZ
JP
JM
JPE
JPO

Condition bits

CALL INST

Format:

[LABE

CODE
CALL
CC
CNC
CZ
CNZ
CP
CM
CPE
CPO

Condition bits a

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

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

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

RST INSTRUCTION

Format:

[LABEL:] RST EXP

Note: $0 \leqslant EXP \leqslant 7$

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

Condition bits affected: None

INPUT/OUTPUT INSTRUCTIONS

Format:

[LABEL:] CODE EXP

Note: For IN, $0 \leqslant EXP \leqslant 7$
For OUT, $8 \leqslant EXP \leqslant 31$

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

C. Instruction

In order to
language instructio

Where an ins
byte is given.

DEC
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

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 C	
139	213	8B	ADC D	
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 B	
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	

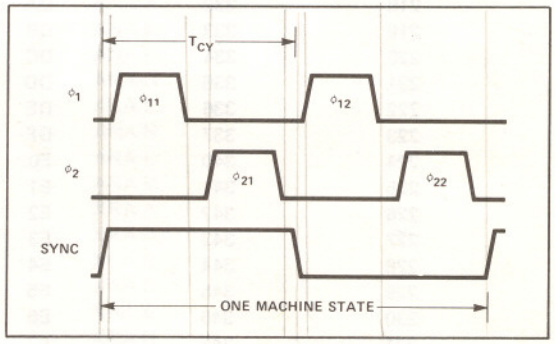
DEC	OCTAL	HEX	MNEMONIC	COMMENT
200	310	C8	MOV B,A	
201	311	C9	MOV B,B	
202	312	CA	MOV B,C	
203	313	CB	MOV B,D	
204	314	CC	MOV B,E	
205	315	CD	MOV B,H	
206	316	CE	MOV B,L	
207	317	CF	MOV B,M	
208	320	D0	MOV C,A	
209	321	D1	MOV C,B	
210	322	D2	MOV C,C	
211	323	D3	MOV C,D	
212	324	D4	MOV C,E	
213	325	D5	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	—	

D. Internal Processor Operation

Internally the processor operates through five different states:

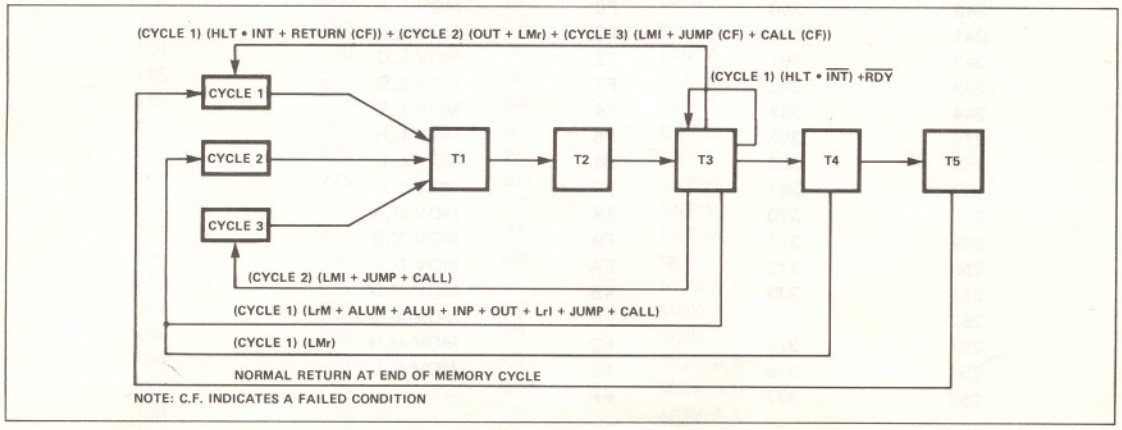
	Internal State	Typical Function
T1	NORMAL	Send out lower eight bits of address and increment program counter.
	INTERRUPT	Send out lower eight bits of address and suppress incrementing of program counter and acknowledge interrupt.
T2		Send out six higher order bits of address and two control bits, D_6 and D_7. Increment program counter if there has been a carry from T1.
T3	WAIT	Wait for READY signal to come true. Refresh internal dynamic memories while waiting.
	NORMAL	Fetch and decode instruction; fetch data from memory; output data to memory. Refresh internal memories.
	STOPPED	Remain stopped until INTERRUPT occurs. Refresh internal memories.
T4 and T5		Execute instruction and appropriately transfer data within processor. Content of data bus transfer is available at I/O bus for convenience in testing. Some cycles do not require these states. In those cases, the states are skipped and the processor goes directly to T1.

The 8008 is driven by two non-overlapping clocks. Two clock periods are required for each state of the processor. ϕ_1 is generally used to precharge all data lines and memories and ϕ_2 controls all data transfers within the processor. A SYNC signal (divide by two of ϕ_2) is sent out by the 8008. This signal distinguishes between the two clock periods of each state.



Processor Clocks

The figure below shows state transitions relative to the internal operation of the processor. As noted in the previous table, the processor skips unnecessary execution steps during any cycle. The state counter within the 8008 operates is a five bit feedback shift register with the feedback path controlled by the instruction being executed. When the processor is either waiting or stopped, it is internally cycling through the T3 state. This state is the only time in the cycle when the internal dynamic memories can be refreshed.



Transition State Diagram (Internal)

The following pages show the processor activity during each state of the execution of each instruction.

STATE TRANSITION SEQUENCE

INDEX REGISTER INSTRUCTIONS

INSTRUCTION CODING D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES TO EXECUTE	# OF MACHINE CYCLES IN INSTRUCTION	SUB-CYCLE ONE (1) T1/2	SUB-CYCLE ONE (1) T2	SUB-CYCLE ONE (1) T3	SUB-CYCLE ONE (1) T4 (3)	SUB-CYCLE ONE (1) T5	SUB-CYCLE TWO T1	SUB-CYCLE TWO T2	SUB-CYCLE TWO T3	SUB-CYCLE TWO T4 (3)	SUB-CYCLE TWO T5	SUB-CYCLE THREE T1	SUB-CYCLE THREE T2	SUB-CYCLE THREE T3	SUB-CYCLE THREE T4 (3)	SUB-CYCLE THREE T5	# BYTES
11	DDD	SSS	MOV r1 r2	5	1	P_{CL} OUT (4)	P_{CH} OUT	FETCH INSTR./SSS TO REG. b TO IR & REG. b	SSS TO REG. b (6)	REG. b TO DDD											1
11	DDD	111	MOV r M	8	2	P_{CL} OUT	P_{CH} OUT	FETCH INSTR. TO IR & REG. b	(7)		REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X (9)	REG. b TO DDD						1
11	111	SSS	MOV M r	7	2	P_{CL} OUT	P_{CH} OUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b		REG. L OUT (10)	REG. H OUT	REG. b TO OUT								1
00	DDD	110	MVI r	8	2	P_{CL} OUT	P_{CH} OUT	FETCH INSTR. TO IR & REG. b			P_{CL} OUT (8)	P_{CH} OUT	DATA TO REG. b	X	REG. b TO DDD						2
00	111	110	MVI M	9	3	P_{CL} OUT	P_{CH} OUT	FETCH INSTR. TO IR & REG. b			P_{CL} OUT (8)	P_{CH} OUT	DATA TO REG. b			REG. L OUT (10)	REG. H OUT	REG. b TO OUT			2
00	DDD	000	INR r	5	1	P_{CL} OUT	P_{CH} OUT	FETCH INSTR.	X	ADD OF - FLAGS											

STATE TRANSITION SEQUENCE

INDEX REGISTER INSTRUCTIONS

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	SUB-CYCLE TWO T1	T2	T3	T4 (3)	T5	SUB-CYCLE THREE T1	T2	T3	T4 (3)	T5	# BYTES
1 1	D D D	S S S	MOV r1, r2	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 1	D D D	1 1 1	MOV r, M	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	(7) →		REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X (9)	REG. b TO DDD						1
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	→	REG. L OUT (10)	REG. H OUT	REG. b TO OUT								1
0 0	D D D	1 1 0	MVI r	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		PC_L OUT (8)	PC_H OUT	DATA TO REG. b	X	REG. b TO DDD						2
0 0	1 1 1	1 1 0	MVI M	9	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		PC_L OUT (8)	PC_H OUT	DATA TO REG. b	→		REG. L OUT (10)	REG. H OUT	REG. b TO OUT			2
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											1
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											1

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
1 0	P P P	1 1 1	ALU OP M	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X	ALU OP - FLAGS AFFECTED						1
0 0	P P P	1 0 0	ALU OP I	8	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		PC_L OUT (8)	PC_H OUT	DATA TO REG. b	X	ARITH OP - FLAGS AFFECTED						2
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											1
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											1
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											1
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											1

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	→		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
0 1	U C C	0 0 0	JNC, JNZ, JP, JPO	9 or 11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		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
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	→		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
0 1	X X X	1 1 0	CALL	11	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		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
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	→		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
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	→		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
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											1
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											1
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											1
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)											1

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	→		REG. A TO OUT (15)	REG. b TO OUT	DATA TO REG. b	COND ↑↑ OUT (16)	REG. b TO REG. A						1
0 1	R R M	M M 1	OUT	6	PC_L OUT	PC_H OUT	FETCH INSTR. TO IR & REG. b	→		REG. A (15) TO OUT	REG. b TO OUT	X (17)								1

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)													1

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.
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 1. State Transition Sequence.

V. PROCESSOR CONTROL SIGNALS

A. Interrupt Signal (INT)

1) INTERRUPT REQUEST

If the interrupt line is enabled (Logic "1"), the CPU recognizes an interrupt request at the next instruction fetch (PCI) cycle by outputting $S_0\,S_1\,S_2$ = 011 at T1I time. The lower and higher order address bytes of the program counter are sent out, but the program counter is not advanced. A successive instruction fetch cycle can be used to insert an arbitrary instruction into the instruction register in the CPU. (If a multi-cycle or multi-byte instruction is inserted, an interrupt need only be inserted for the first cycle.)

When the processor is interrupted, the system INTERRUPT signal must be synchronized with the leading edge of the ϕ_1 or ϕ_2 clock. To assure proper operation of the system, the interrupt line to the CPU must not be allowed to change within 200ns of the falling edge of ϕ_1.

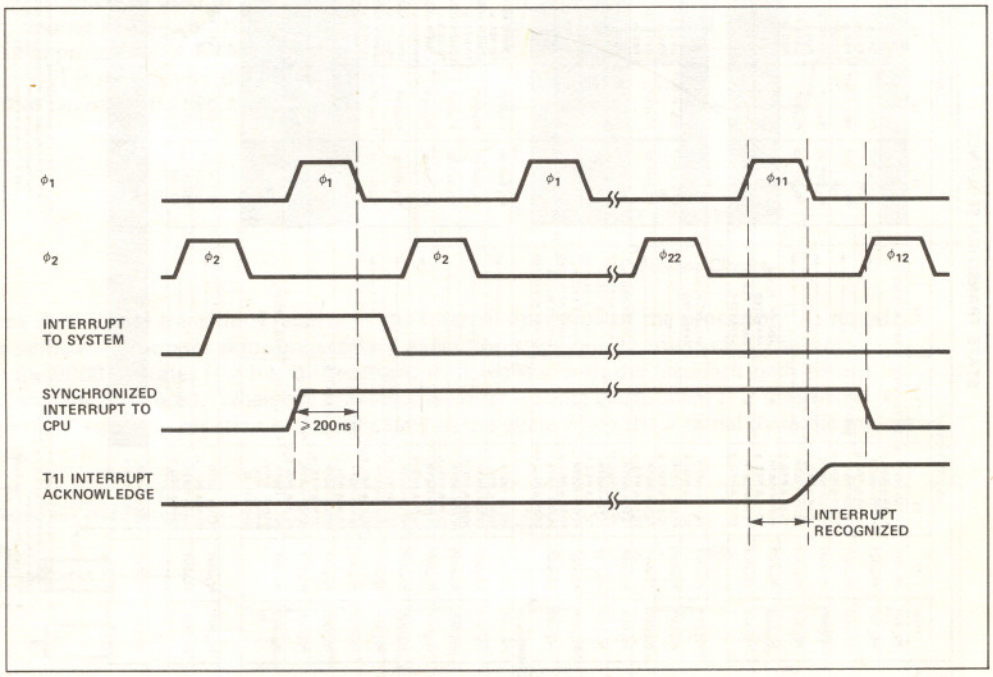


Figure 4. Recognition of Interrupt

If a HALT is inserted, the CPU enters a STOPPED state; if a NOP is inserted, the CPU continues; if a "JUMP to 0" is inserted, the processor executes program from location 0, etc. The RESTART instruction is particularly useful for handling interrupt routines since it is a one byte call.

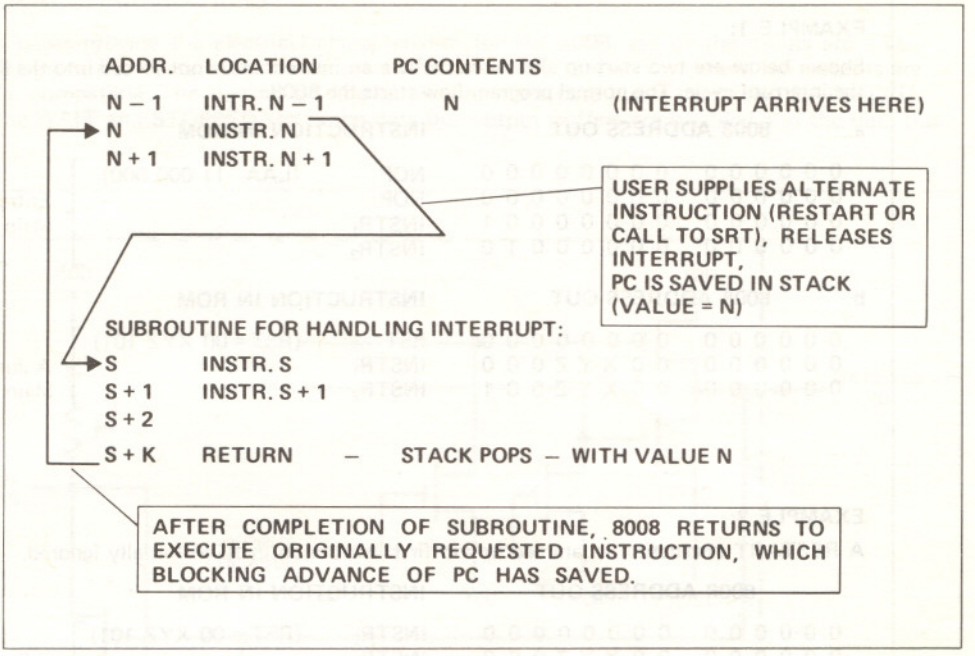


Figure 5. 8008 Interrupt

2) START-UP OF THE 8008

When power (V_{DD}) and clocks (ϕ_1, ϕ_2) are first turned on, a flip-flop internal to the 8008 is set by sensing the rise of V_{DD}. This internal signal forces a HALT (00000000) into the instruction register and the 8008 is then in the STOPPED state. The following sixteen clock periods after entering the STOPPED state are required to clear (logic "0") memories (accumulator, scratch pad, program counter, and stack). During this time the interrupt line has been at logic "0". Any time after the memories are cleared, the 8008 is ready for normal operation.

To reset the flip-flop and also escape from the stopped state, the interrupt line must go to a logic "1"; It should be returned to logic "0" by decoding the state T1I at some time later than ϕ_{11}. Note that whenever the 8008 is in a T1I state, the program counter is not incremented. As a result, the same address is sent out on two successive cycles.

Three possible sequences for starting the 8008 are shown on the following page. The RESTART instruction is effectively a one cycle call instruction, and it is convenient to use this instruction to call an initiation subroutine. Note that it is not necessary to start the 8008 with a RESTART instruction.

The selection of initiation technique to use depends on the sophistication of the system using the 8008. If the interrupt feature is used only for the start-up of the 8008 use the ROM directly, no additional external logic associated with instructions from source other than the ROM program need be considered. If the interrupt feature is used to jam instructions into the 8008, it would then be consistent to use it to jam the initial instruction.

The timing for the interrupt with the start-up timing is shown on an accompanying sheet. The jamming of an instruction and the suppression of the program counter update are handled the same for all interrupts.

EXAMPLE 1:

Shown below are two start-up alternatives where an instruction is not forced into the 8008 during the interrupt cycle. The normal program flow starts the 8008.

a.

8008 ADDRESS OUT		INSTRUCTION IN ROM		
0 0 0 0 0 0	0 0 0 0 0 0 0 0	NOP	(LAA 11 000 000)	Entry Directly To Main Program
0 0 0 0 0 0	0 0 0 0 0 0 0 0	NOP		
0 0 0 0 0 0	0 0 0 0 0 0 0 1	$INSTR_1$		
0 0 0 0 0 0	0 0 0 0 0 0 1 0	$INSTR_2$		

b.

8008 ADDRESS OUT		INSTRUCTION IN ROM		
0 0 0 0 0 0	0 0 0 0 0 0 0 0	RST	(RST = 00 XYZ 101)	A Jump To The Main Program
0 0 0 0 0 0	0 0 X Y Z 0 0 0	$INSTR_1$		
0 0 0 0 0 0	0 0 X Y Z 0 0 1	$INSTR_2$		
	.	.		
	.	.		
	.	.		

EXAMPLE 2:

A RESTART instruction is jammed in and first instruction in ROM initially ignored.

8008 ADDRESS OUT		INSTRUCTION IN ROM		
0 0 0 0 0 0	0 0 0 0 0 0 0 0	$INSTR_1$	(RST = 00 XYZ 101)	Start-up Routine
0 0 0 0 0 0	0 0 X Y Z 0 0 0	$INSTR_a$		
0 0 0 0 0 0	0 0 X Y Z 0 0 1	$INSTR_b$		
	.	.		
	.	.		
	.	.		
0 0 0 0 0 0	0 0 n n n n n n	RETURN		
0 0 0 0 0 0	0 0 0 0 0 0 0 0	$INSTR_1$	($INSTR_1$ executed now)	Main Program
0 0 0 0 0 0	0 0 0 0 0 0 0 1	$INSTR_2$		
	.	.		
	.	.		
	.	.		

Note that during the interrupt cycle the flow of the instruction to the 8008 either from ROM or another source must be controlled by hardware external to 8008.

START-UP OF THE 8008

B. Ready (RDY)

The 8008 is designed to operate with any type or speed of semiconductor memory. This flexibility is provided by the READY command line. A high-speed memory will always be ready with data (tie READY line to V_{CC}) almost immediately after the second byte of the address has been sent out. As a result the 8008 will never be required to wait for the memory. On the other hand, with slow ROMs, RAMs or shift registers, the data will not be immediately available; the 8008 must wait until the READY command indicates that the valid memory data is available. As a result any type or any combination of memory types may be used. The READY command line synchronizes the 8008 to the memory cycle. **When a program is being developed, the READY signal provides a means of stepping through the program, one cycle at a time.**

VI. ELECTRICAL SPECIFICATIONS

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.

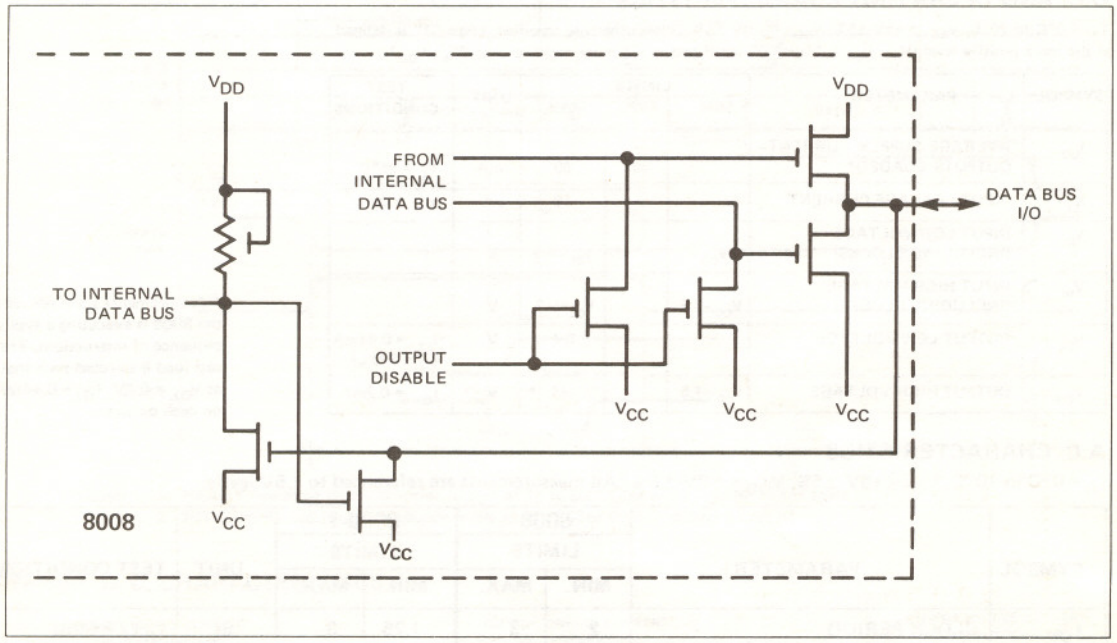


Figure 6. Data Bus I/O Buffer

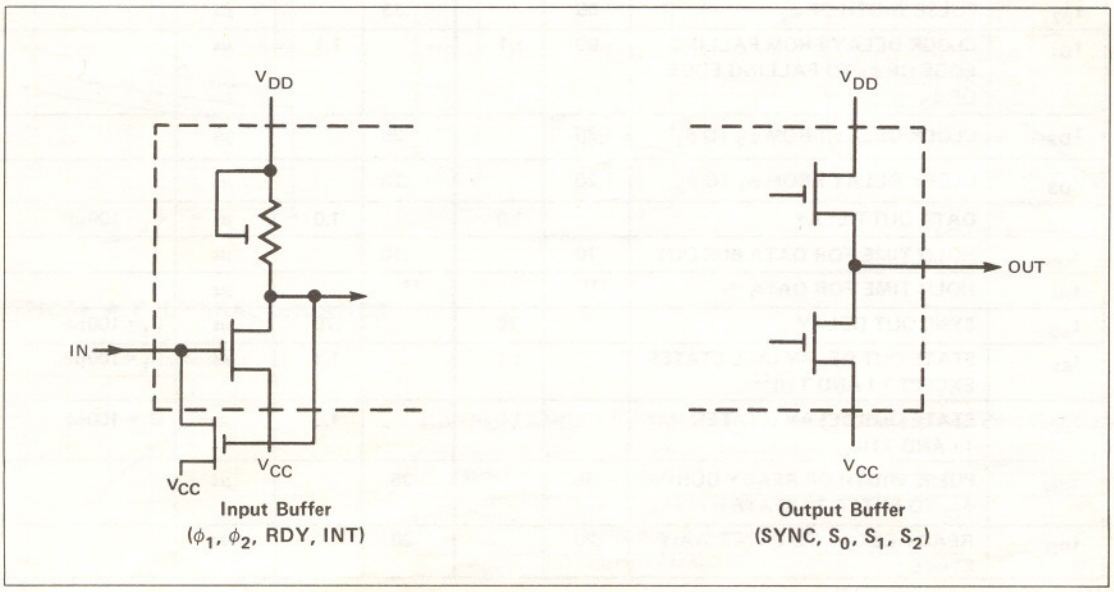


Figure 7. I/O Circuitry

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.	LIMITS TYP.	LIMITS 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 DIAG

DATA B
LIN
(D_7 ...

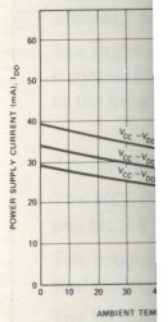
STA
LINE

READ

Notes:

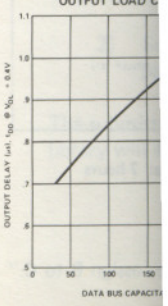
TYPICAL D.C.

POWER SUPP
VS. TEMP

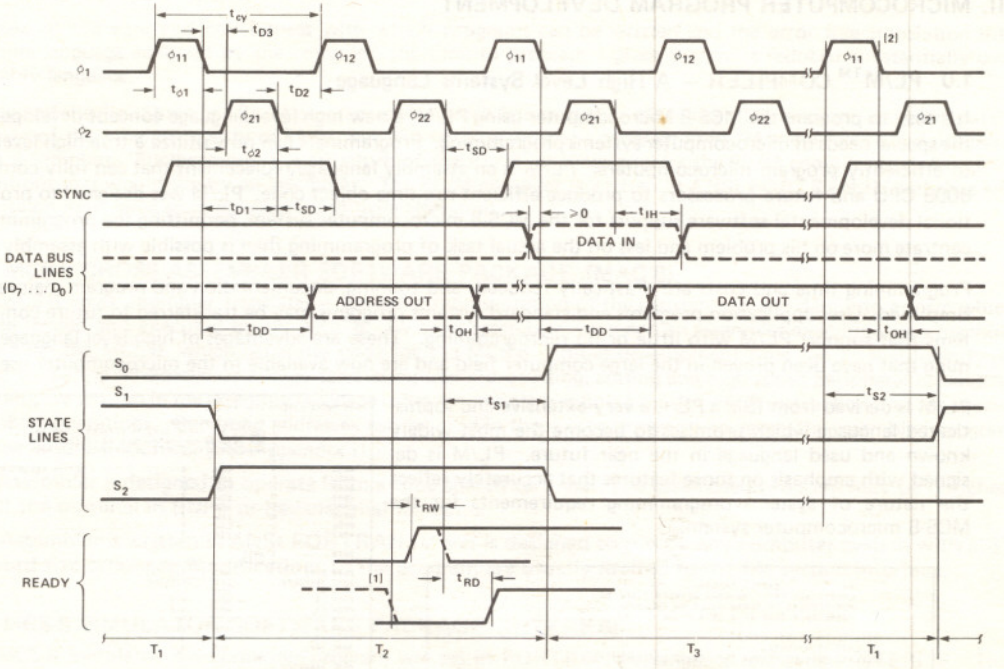


TYPICAL A.C.

DATA OUT D
OUTPUT LOAD C



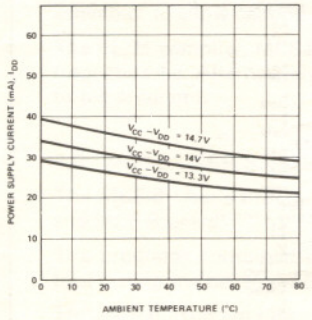
TIMING DIAGRAM



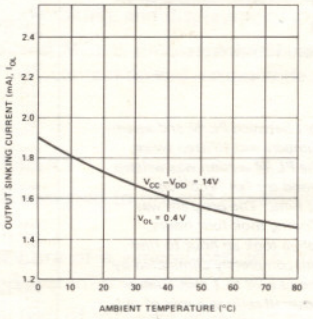
Notes: 1. READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.
2. INTERRUPT line must not change levels within 200 ns (max.) of falling edge of ϕ_1.

TYPICAL D.C. CHARACTERISTICS

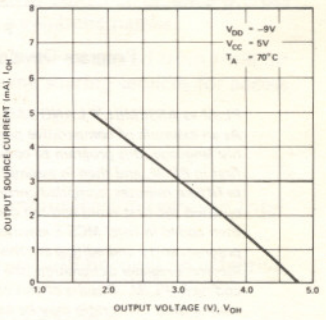
POWER SUPPLY CURRENT VS. TEMPERATURE



OUTPUT SINKING CURRENT VS. TEMPERATURE

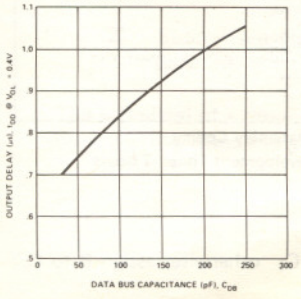


OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



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

VII. MICROCOMPUTER PROGRAM DEVELOPMENT

1.0 PL/M™ COMPILER — A High Level Systems Language

It's easy to program the MCS-8 Microcomputer using PL/M, a new high level language concept developed to meet the special needs of microcomputer systems programming. Programmers can now utilize a true high level language to efficiently program microcomputers. PL/M is an assembly language replacement that can fully command the 8008 CPU and future processors to produce efficient run-time object code. PL/M was designed to provide additional developmental software support for the MCS-8 microcomputer system, permitting the programmer to concentrate more on his problem and less on the actual task of programming than is possible with assembly language.

Programming time and costs are drastically reduced, and training, documentation and program maintenance are simplified. User application programs and standard systems programs may be transferred to future computer systems that support PL/M with little or no reprogramming. These are advantages of high-level language programming that have been proven in the large computer field and are now available to the microcomputer user.

PL/M is derived from IBM's PL/I, a very extensive and sophisticated language which promises to become the most widely known and used language in the near future. PL/M is designed with emphasis on those features that accurately reflect the nature of systems programming requirements for the MCS-8 microcomputer system.

```
/*          SAMPLE PROGRAM
  LOCATE ALL PRIME NUMBERS BETWEEN 1 AND 50,
  PUT RESULTS IN TRUTH TABLE AS FOLLOWS:
  PRIME(I) = TRUE IF I IS A PRIME      */

DECLARE PRIME(50) BYTE;
DECLARE (I,K) BYTE;
DECLARE TRUE LITERALLY '1', FALSE LITERALLY '0';

  PRIME(1) = TRUE; /* 1 IS A PRIME */

  DO I = 2 TO 50;
    PRIME(I) = FALSE; /* INITIALIZE TABLE TO FALSE */
    K = 2;
    DO WHILE I MOD K <> 0; /* LOOP UNTIL TEST FOR PRIME FAILS */
      K = K + 1;
    END;
    IF K = I THEN
      DO; /* FOUND A PRIME */
        PRIME(I) = TRUE;
      END;
  END;

  EOF /* END OF PROGRAM */
```

PL/M Coding
Program Development Time: 15 minutes

PL/M vs ASSEMBLY LANGUAGE
As an example of comparative programming effort between PL/M and assembly language, this program to computer prime numbers was written twice, first in PL/M, and then in assembly language. The PL/M version was written in fifteen minutes, compiled correctly on the second try (an "end" was omitted the first time) and ran correctly the first time. The program was then coded in Intel MCS-8 assembly language. Coding took four hours, program entry and editing another two hours, debug took an hour to find incorrect register designation, the kind of problem completely eliminated by coding in PL/M. Results of this one short test shows a 28 to 1 reduction in coding time. This ratio may be somewhat high, overall ratio in a mix of programs is more on the order of 10 to 1.

```
MCS8 MACRO ASSEMBLER:  PAGE 1
                       /* SAMPLE PROGRAM
                       * LOCATE ALL PRIME NUMBERS BETWEEN 1 AND 50.
                       * PUT RESULTS IN TRUTH TABLE AS FOLLOWS:
                       * PRIME(I) = TRUE IF I IS A PRIME,      */
                       ;
0000                   A       EQU     0               ; DEFINE REGISTERS
0001                   B       EQU     1
0002                   C       EQU     2
0003                   D       EQU     3
0004                   E       EQU     4
0005                   H       EQU     5
0006                   L       EQU     6
0007                   M       EQU     7
                       ;
0000                   START:
                       ; PRIME(1) = TRUE; /* 1 IS A PRIME */
0000 0601                      MVI     A,1
0002 366F2E00                  LXI     H,PRIME
0006 86                        ADD     L
0007 C8                        MOV     A,B
0008 C5                        MOV     H,A
0009 0E00                      ACI     0
000B F1                        MOV     B,L
000C E8                        MOV     A,H
000D 3E01                      MVI     M,1
                       ; DO I = 2 TO 50;
000F 36222E00                  LXI     H,I
0013 3E02                      MVI     M,2
0015                   LOOP1:
0015 0632                      MVI     A,50
0017 36222E00                  LXI     H,I
001B BF                        CMP     M
001C 6A6E2E00                  JC      DONE
                       ;   PRIME(I) = FALSE; /* INITIALIZE TABLE TO FALSE */
0021 366F2E00                  LXI     H,PRIME
0025 86                        ADD     L
0026 C8                        MOV     A,B
0027 C5                        MOV     H,A
0028 0E00                      ACI     0
002A F1                        MOV     B,L
002B E8                        MOV     A,H
002C 3E00                      MVI     M,0
                       ;   K = 2;
002E 36232E00                  LXI     H,K
0032 3E02                      MVI     M,2
                       ;   DO WHILE I MOD K <> 0; /* LOOP UNTIL TEST FOR PRIME FAILS */
0034                   LOOP1A:
0034 36222E00                  LXI     H,I
0038 C7                        MOV     M,A

MCS8 MACRO ASSEMBLER:  PAGE 2
0039 36232E00                  LXI     H,K
003D CF                        MOV     M,B
003E                   LOOP2:
003E 91                        SUB     B
003F 4843C00                   JNC     LOOP2
0043 81                        ADD     B
0044 3C00                      CPI     0
0046 6847C00                   JZ      LOOP3
                       ;      K = K + 1;
0049 CF                        MOV     M,B
004A 38                        INR     B
004B F9                        MOV     B,M
                       ;   END;
004C 443300                    JMP     LOOP1
004F                   LOOP3:
                       ;   IF K = I THEN
                       ;      DO; /* FOUND A PRIME */
004F 36232E00                  LXI     H,K
0053 C7                        MOV     M,A
0054 31                        DCR     L
0055 BF                        CMP     M
0056 4E6400                    JNZ     LOOP4
                         *;    PRIME(I) = TRUE;
0059 C7                        MOV     M,A
005A 316F2E00                  LXI     H,PRIME
005E B6                        ADD     L
005F C8                        MOV     A,B
0060 C5                        MOV     H,A
0061 0E00                      ACI     0
0063 F1                        MOV     B,L
0064 E8                        MOV     A,H
0065 3E01                      MVI     M,1
                       ;      END;
0067                   LOOP4:
0067 36222E00                  LXI     H,I
006B CF                        MOV     M,B
006C 38                        INR     B
006D F9                        MOV     B,M
                       ;   END;
006E 447500                    JMP     LOOP0
0071                   DONE:
                       ; EOF /* END OF PROGRAM */
0071 02                        HLT
                       ;
                       ; DECLARE PRIME(50) BYTE;
                       ; DECLARE (I,K) BYTE;
                       ; DECLARE TRUE LITERALLY '1', FALSE LITERALLY '0';
                       ;
0072                   PRIME:  DS      51
0091                   I:      DS      1
0092                   K:      DS      1
                               END
```

Assembly Coding
Program Development Time: 7 hours

PL/M Is An Efficient Language

Tests on sample programs indicate that a PL/M program can be written in less than 10% of the time it takes to write the same program in assembly language with little efficiency loss. The main reason for this savings in time is the fact that PL/M allows the programmer to define his problem in terms natural to him, not in the computer's terms. Consider the following sample program which selects the largest of two numbers. In PL/M, the programmer might write:
If A > B, then C = A; else C = B;
Meaning: "If variable A is greater than variable B, then assign A to variable C; otherwise, assign B to C."

A corresponding program in assembly language is twelve separate machine instructions, and conveys little of original intent of the program.

Because of the ease and conciseness with which programs can be written and the error free translation into machine language achieved by the compiler, the time to program a given system is reduced substantially over assembly language.

Debug and checkout time of a PL/M™ program is also much less than that of an assembly language program, partly because of the inherent clarity of PL/M, but also because writing a program in PL/M encourages good programming techniques. Furthermore, the structure of the PL/M language enables the PL/M compiler to detect error conditions that would slip by an assembler. The PL/M compiler is written in ANSI FORTRAN IV and thus will execute on most large scale machines with little alteration.

2.0 MCS-8™ CROSS ASSEMBLER SOFTWARE PACKAGE (MAC 8)

The MCS-8 cross assembler translates a symbolic representation of the instructions and data into a form which can be loaded and executed by the MCS-8. By cross assembler, we mean an assembler executing on a machine other than the MCS-8, which generates code for the MCS-8. Initial development time can be significantly reduced by taking advantage of a large scale computer's processing, editing and high speed peripheral capability. Programs are written in the assembly language using mnemonic symbols both for 8008 instruction and for special assembler operations. Symbolic addresses can be used in the source program; however, the assembled program will use absolute address. (See Appendix II.)

The Assembler is designed to operate from a time shared terminal. The assembled program may be punched out at the terminal in BNPF or hexadecimal format.

The Assembler is written in ANSI FORTRAN IV and is designed to run on any computer system with a 32 bit word size or longer. Modifications to the program are usually limited to the file system interface.

3.0 MCS-8 SIMULATOR SOFTWARE PACKAGE (INTERP 8)

The MCS-8 Simulator is a computer program written in FORTRAN IV language and called INTERP/8. This program provides a software simulation of the Intel 8008 CPU, along with execution monitoring commands to aid program development for the MCS-8.

INTERP/8 accepts machine code produced by the 8008 Assembler, along with execution commands from a time sharing terminal, card reader, or disk file. The execution commands allow manipulation of the simulated MCS-8 memory and the 8008 CPU registers. In addition, operand and instruction breakpoints may be set to stop execution at crucial points in the program. Tracing features are also available which allow the CPU operation to be monitored. INTERP/8 also accepts symbol tables from either the PL/M compiler or MCS-8 cross assembler to allow debugging, tracing and braking, and displaying of program using symbolic names.

The PL/M compiler, MCS-8 assembler, and MCS-8 simulator software packages may be procured from Intel on magnetic tape. Alternatively, designers may contact several nation-wide computer time sharing services for access to the programs.

4.0 MCS-8 USER'S PROGRAM LIBRARY

The MCS-8 User's Program Library is designed to provide a creative path of communication between Intel and MCS-8 users.

The programs contained in the library are of a general nature so that the library will become an effective tool during the development of MCS-8 system software.

The MCS-8 Program Library is a collection of software generated by both Intel and MCS-8 users; it is packaged in a three-ring binder with updates provided to all members on a quarterly basis.

To become a member of the MCS-8 User's Program Library simply:

1) Make payment of $100 for annual membership fee.

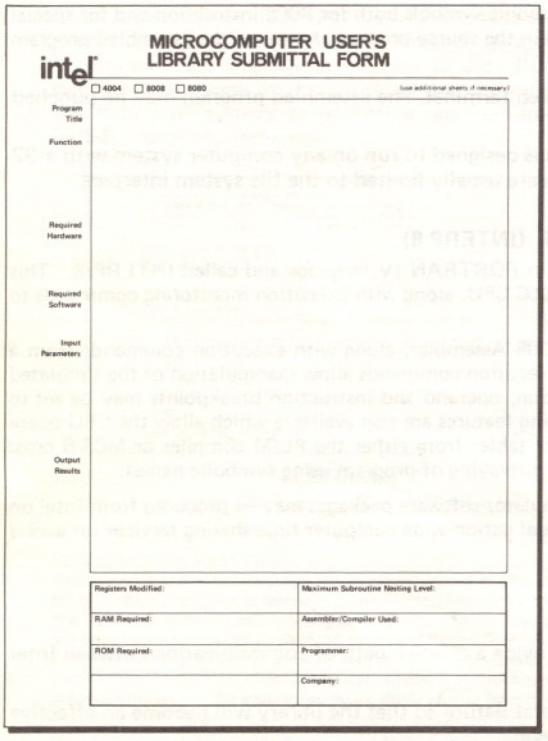
OR

2) Submit a program of general nature (non-proprietary) to Intel via the submittal form shown on page 30.

The submittal of a program is the preferred method of membership qualification so that the content of the library will continue to grow and all members will benefit from mutual contribution.

Examples of programs in the MCS-8 User's Program Library:

- Binary Search Subroutine
- Floating Point Arithmetic Package
- Floating Point I/O Conversion Package
- Save/Restore CPU State on Interrupt
- Cross Assembler for HP2100
- BCD To/From Binary Conversion
- 8-Bit Multiply
- 8-Bit Divide
- 16-Bit Multiply
- 16-Bit Divide
- RAM Test Program
- Binary Load/Dump Routine
- Signed 16-Bit Multiply
- Octal Memory Dump Routine
- PROM Programming Routine
- Morse Code Generator



intel

MICROCOMPUTER USER'S LIBRARY SUBMITTAL FORM

☐ 4004 ☐ 8008 ☐ 8080 (use additional sheets if necessary)

Program Title

Function

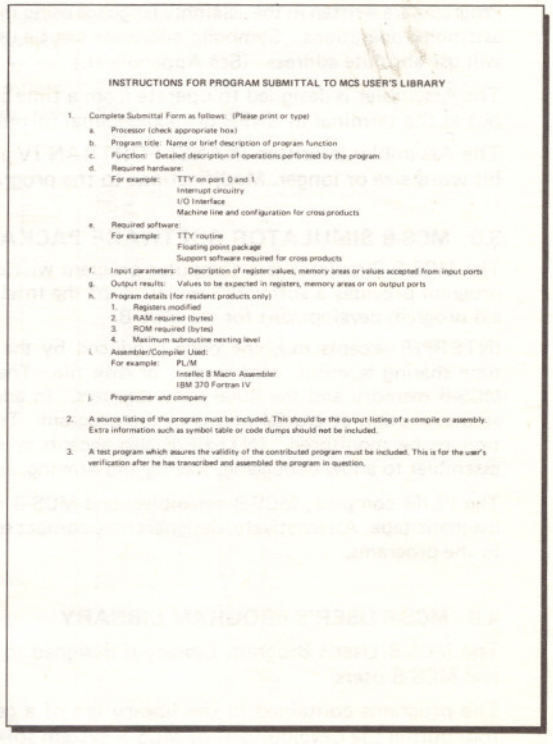
Required Hardware

Required Software

Input Parameters

Output Results

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



INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

1. Complete Submittal Form as follows: (Please print or type)
   a. Processor (check appropriate box)
   b. Program title: Name or brief description of program function
   c. Function: Detailed description of operations performed by the program
   d. Required hardware:
      For example: TTY on port 0 and 1
      Interrupt circuitry
      I/O Interface
      Machine line and configuration for cross products
   e. Required software:
      For example: TTY routine
      Floating point package
      Support software required for cross products
   f. Input parameters: Description of register values, memory areas or values accepted from input ports
   g. Output results: Values to be expected in registers, memory areas or on output ports
   h. Program details (for resident products only)
      1. Registers modified
      2. RAM required (bytes)
      3. ROM required (bytes)
      4. Maximum subroutine nesting level
   i. Assembler/Compiler Used:
      For example: PL/M
      Intellec 8 Macro Assembler
      IBM 370 Fortran IV
   j. Programmer and company
2. A source listing of the program must be included. This should be the output listing of a compile or assembly. Extra information such as symbol table or code dumps should not be included.
3. A test program which assures the validity of the contributed program must be included. This is for the user's verification after he has transcribed and assembled the program in question.

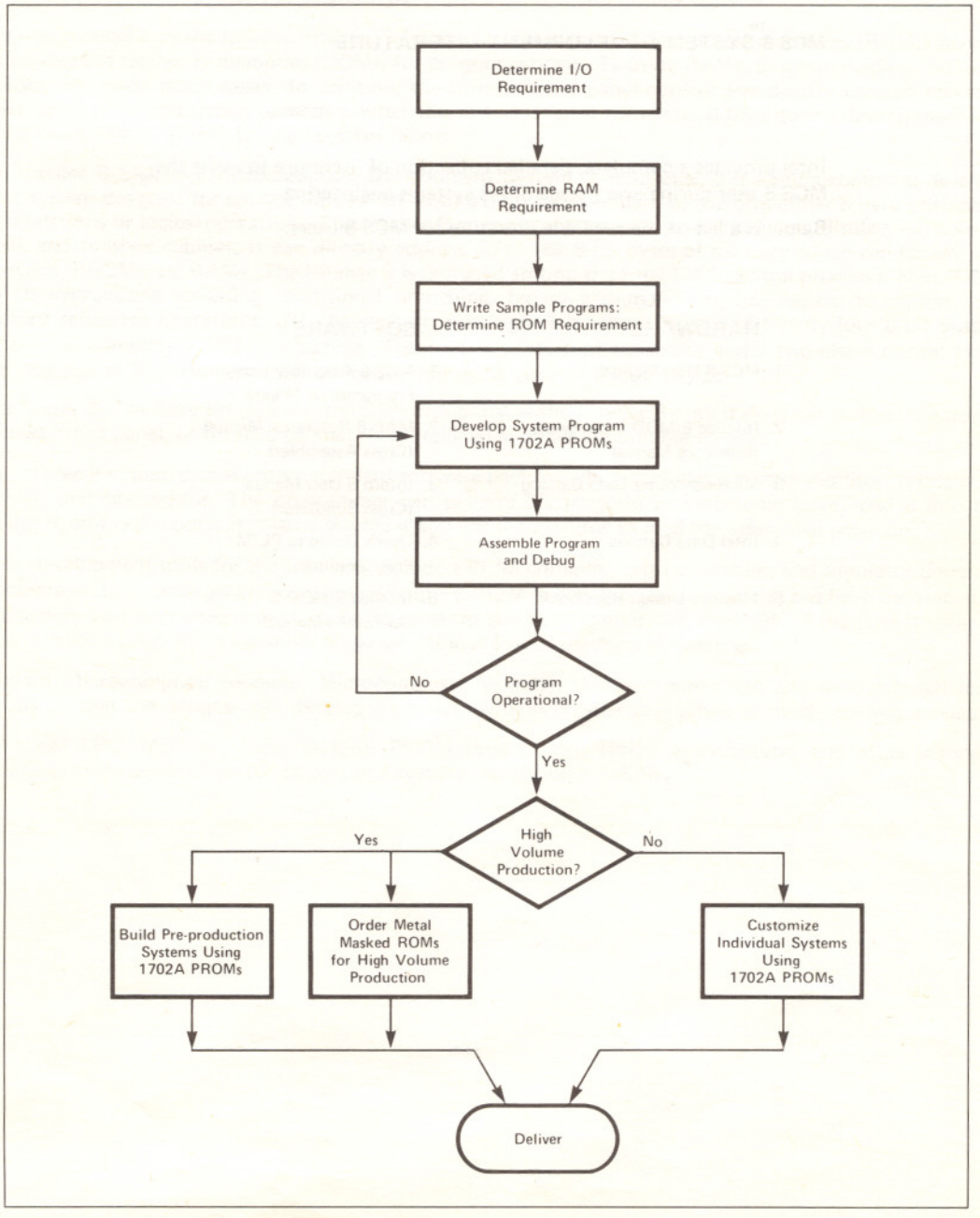
NOTE: Tear-out copies of Submittal Form are available in back of manual.

B. Develo

The flowchar
system can b
system hardw
(100 or more

B. Development of a Microcomputer System

The flowchart shows the steps required for the development of a microcomputer system. The INTELLEC® 8/MOD 8™ system can be used throughout the complete cycle for program assembly, PROM programming, and prototype system hardware. Ultimately, custom systems using 1702A PROMs may be delivered. For high volume applications (100 or more identical systems) lower cost metal masked ROMs may be used.



MCS-8™ SYSTEM DEVELOPMENT LITERATURE

Intel provides a complete, detailed collection of literature to assist the MCS-8 user during every stage of his systems engineering.

Below is a list of the available literature for MCS-8 users:

HARDWARE

1. MCS-8 User Manual
2. Intellec® 8/MOD 8 Hardware Reference Manual
3. Microcomputer Data Catalog
4. Intel Data Catalog
5. Memory Design Handbook

SOFTWARE

1. MCS-8 Assembly Language Programming Manual
2. MAC 8 Reference Manual (Cross Assembler)
3. Interp 8 User Manual (Cross Simulator)
4. User's Guide to PL/M Programming
5. Intellec 8/MOD 8 Operator's Manual

VIII. INTELLEC® 8/MOD 8/BARE BONES 8 AND MICROCOMPUTER MODULES

The widespread usage of low-cost microcomputer systems is made possible by Intel's development and volume production for MCS-8™ microcomputer sets. To make it easier to use these sets, Intel offers complete 8-bit modular microcomputer development systems called Intellec 8.

The Intellec modular microcomputers provide a flexible, inexpensive, and simplified method for developing OEM systems. They are self-contained, expandable systems complete with central processor, memory, I/O, crystal clock, power supplies, standard software, and a control and display console.

The major benefit of the Intellec modular microcomputers is that random access memories (RAMs) may be used instead of read-only-memories (ROMs) for program storage. By using RAMs, program loading and modification are made much easier. In addition, the Intellec front panel control and display console makes it easier to monitor and debug programs. What this means is faster turn-around time during development, enabling you to arrive at that finished system sooner.

The Intellec 8 Eight-Bit Microcomputer Development System. The Intellec 8 is a microcomputer development system designed for applications which require 8-bit bytes of data to perform either binary arithmetic manipulations or logical operations. The Intellec 8 comes complete with power supplies, display and control panel, and finished cabinet. It can directly address up to 16k 8-bit bytes of memory which can be any mix of ROMs, PROMs, or RAMs. The Intellec 8 is designed around the Intel 8008 central processor chip. There are 48 instructions including conditional branching, binary arithmetic, logical, register-to-register, and memory reference operations. I/O channels provide eight 8-bit input ports and twenty-four 8-bit output ports — all completely TTL compatible. The unit has interrupt capability and a two-phase crystal clock that operates at 800kHz providing an instruction cycle time of about 12.5μs.

Bare Bones 8. The Bare Bones 8 has the same capability as the Intellec 8 only it does not include the power supplies, front panel, or finished cabinet. It is designed as a rack-mountable version.

The Intellec 8 system comes with a standard software package which includes a system monitor, resident assembler, and text editor. The programmer can prepare his program in mnemonic form, load it into the Intellec 8, edit and modify it, then assemble it and use the monitor to load the assembled program.

Other development tools for the Intellec8 include a PL/M™ compiler, cross assembler, and simulator designed to operate on large scale general purpose computers. PL/M, a new high-level language, has been developed as an assembly language replacement. A PL/M program can be written in less than 10% of the time it takes to write that same program in assembly language without loss of machine efficiency.

Standard Microcomputer Modules. Microcomputer Modules, standard cards that can be purchased individually so that the designer can develop his system with as little or as much as he needs, are also available.

Additional CPU, Memory, Input/Output, PROM Programmer, Universal Prototype, and other standard modules provide developmental support and systems expansion capability.

Intellec® 8/MOD 8 and Bare Bones 8/MOD 8

MCS-8™ MICROCOMPUTER DEVELOPMENT SYSTEMS

- **Intellec 8 (imm8-80A): Complete Microcomputer Development System**
  - **Central Processor Module**
  - **RAM Memory Modules (8192 x 8)**
  - **Input/Output Module (TTL compatible)**
  - **PROM Memory Module (4k x 8 capacity; 1k Resident System Monitor included)**
  - **PROM Programmer Module**
  - **Control Console and Display**
  - **Power Supplies and Cabinet**
- **Bare Bones 8: MCS-8 System without power supplies, cabinet, or control console**
- **Standard Software**
  - **Resident System Monitor**
  - **Assembler Text Editor** — **Requires 8k of RAM**
- **10k bytes of Memory (expandable to 16,384 bytes — Intellec 8)**
- **5k bytes of Memory (expandable to 16,384 bytes — Bare Bones 8)**
- **Direct Access to Memory and I/O**
- **Four 8-bit input ports (expandable to eight)**
- **Four 8-bit output ports (expandable to twenty-four)**
- **Universal Asynchronous Transmitter Receiver for serial communications interface**
- **Real time interrupt capability**
- **Crystal controlled master system clock**

The Intellec 8 is a complete microcomputer development system for MCS-8 microcomputer systems. Its modular design allows the development of any size MCS-8 system, and it has built-in features to make this task easier than it has ever been before.

The basic Intellec 8 (imm8-80A) consists of six microcomputer modules (CPU, 2-RAM, PROM, I/O and PROM programmer), power supplies, and console and displays in a small compact package. The heart of the system is the imm8-82 Central Processor Module. It is built around Intel's 8008-1, an 8-bit CPU on a chip. It contains all necessary interface to control up to 16k of memory, eight 8-bit input ports, twenty-four 8-bit output ports, and to respond to real time interrupts.

The Intellec 8 has 10k bytes of memory in its basic configuration and may be expanded up to a maximum of 16,384 bytes of memory. Of the basic 10k bytes of memory, 8192 bytes are random access read/write memory located on the imm6-28 RAM Memory Modules and are addressed as the lower 8k of memory. This memory may be used for both data storage and program storage. The remaining 2k bytes of memory are located on the imm6-26 PROM Memory Module and addressed as the upper 2k bytes of the 16k memory. This portion of memory is a system monitor in eight 1702A PROMs. Eight additional sockets are available on the imm6-26 for monitor or program expansion. Control for the PROM Programmer Module (imm6-76) is included with the monitor for system control.

PROM memory modules and RAM memory modules may be used in any combination to make up the 16k of directly addressable memory. Facilities are built into these modules so that any combination of RAM and ROM or PROM may be mixed in 256 byte increments.

Input and output in the Intellec 8 is provided by the imm8-60 I/O module. It contains four 8-bit input ports, and four 8-bit output ports. In addition it contains a universal asynchronous transmitter/receiver chip as well as a teletype driver, receiver, and reader control. Bit serial communication using only the teletype drivers, receivers, and the I/O port, is also possible with this module.

The universal asynchronous transmitter receiver chip may operate at either 110 baud for standard teletype interface or 1200 baud for communication with a high speed CRT terminal. Additional I/O modules, imm8-60, and output modules, imm8-62, can expand the I/O capability of the Intellec 8 to eight input ports and twenty-four output ports, all TTL compatible.

An interrupt line and an 8-bit interrupt instruction port is built into the imm8-82 Central Processor Module. When an interrupt occurs, the processor executes the instruction which is present at the interrupt instruction port. In the Intellec 8, both the interrupt line and the interrupt instruction port are connected to the console. The processor may be interrupted by depressing the switch labeled INT, and the interrupt instruction is entered in the ADDRESS/INSTRUCTION/DATA switches.

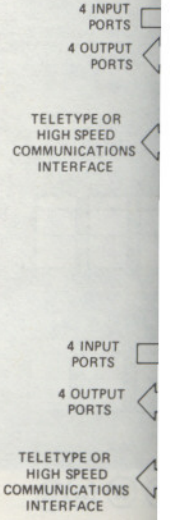
Additional module locations are available in the Intellec 8 so the user may develop his own custom interface using the imm6-70 Universal Prototype Module. All necessary control signals, data, and address buses are present at the connectors of the unused module locations for this expansion. When memory, I/O, and custom interfaces are added to the Intellec 8, care should be taken not to exceed the built-in power supply capability.

Every Intellec 8 comes with three basic pieces of software: the systems monitor, a resident program located in the upper 2k bytes of memory, a symbolic assembler and a text editor. The resident systems monitor allows the operator to punch and load tapes, display and alter memory, and execute programs.

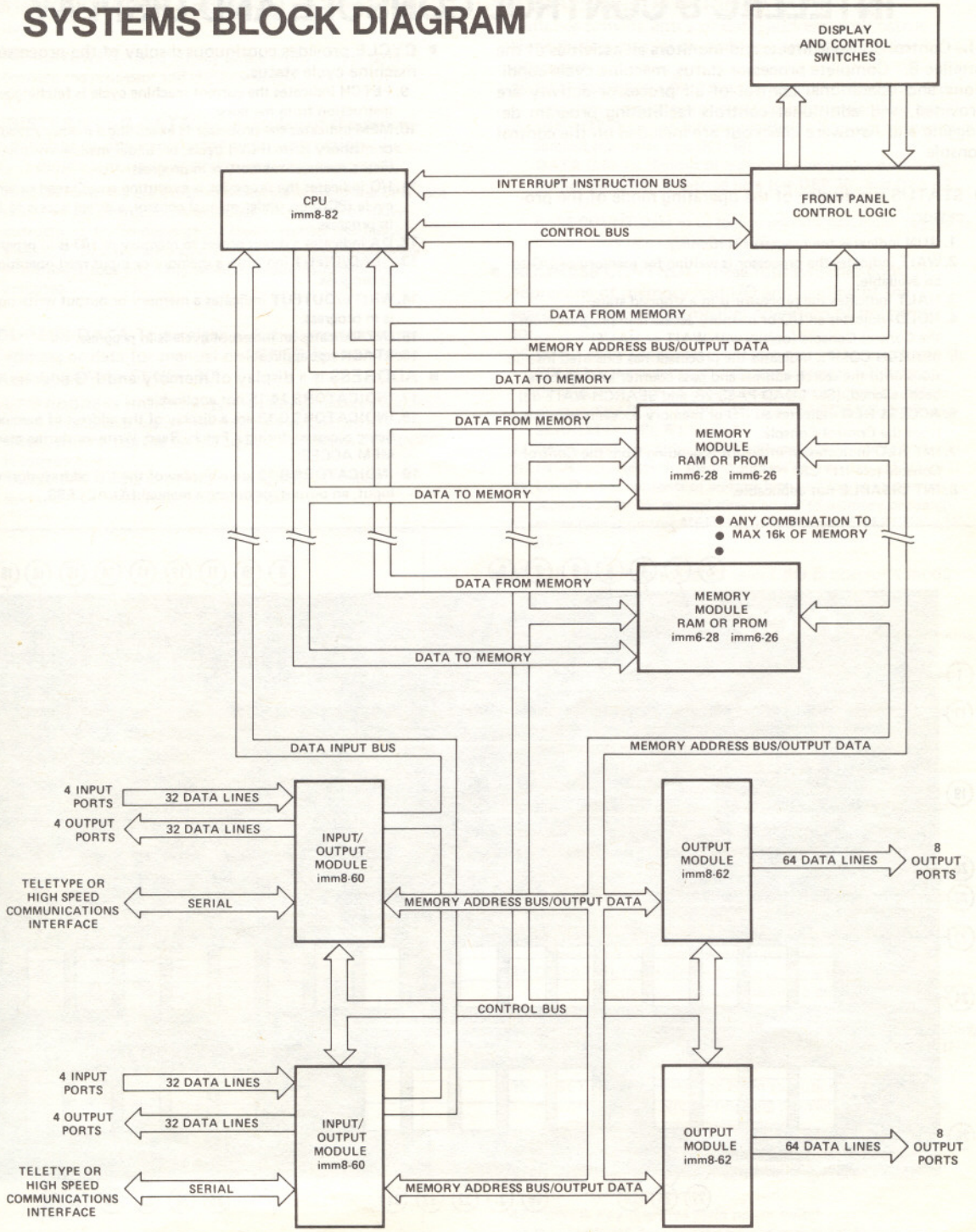
With the PROM Programmer Module, 1702A PROMs may be programmed and verified under control of the system monitor.

The text editor is a paper tape editor to allow the operator to edit his source code before assembly. The assembler takes this source tape and translates it into object code to run on the Intellec 8 or any MCS-8 system.

The Intellec 8 microcomputer development system is also available in a Bare Bones 8 version. In this version the power supply, chassis, console, and display are removed leaving the user a compact rack mountable chassis to imbed in his own system.



SYSTEMS BLOCK DIAGRAM

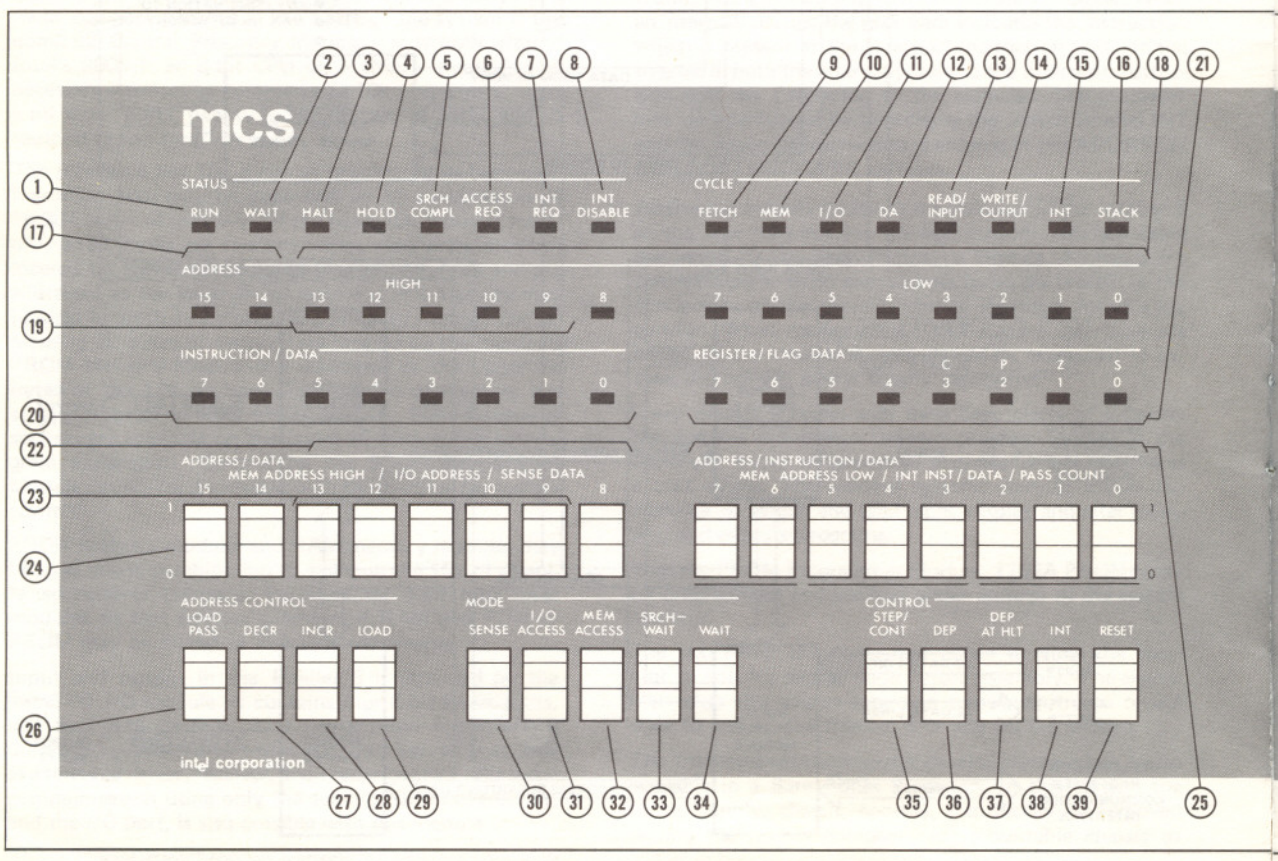


INTELLEC 8 CONTROL CONSOLE AND DISPLAY

The Control Console directs and monitors all activities of the Intellec 8. Complete processor status, machine cycle conditions and operational control of all processor activity are provided, and additional controls facilitating program debugging and hardware checkout are included on the control console.

- **STATUS** is a display of the operating mode of the processor.
  1. **RUN** indicates the processor is running.
  2. **WAIT** indicates the processor is waiting for memory or I/O to be available.
  3. **HALT** indicates the processor is in a stopped state.
  4. **HOLD** indicates an I/O or memory access is in progress from the Control Console (occurs with WAIT or HALT).
  5. **SEARCH COMPL** indicates the processor has executed instructions until the search address and pass counter settings have been reached. (See LOAD PASS 26, and SEARCH-WAIT 33)
  6. **ACCESS REQ** indicates an I/O or memory access is pending from the Control Console.
  7. **INT REQ** indicates an interrupt is pending from the Control Console (see INT 38).
  8. **INT DISABLE** not applicable.
- **CYCLE** provides continuous display of the processor's machine cycle status.
  9. **FETCH** indicates the current machine cycle is fetching an instruction from memory.
  10. **MEM** indicates the processor is executing a memory read (PCR) or memory write (PCW) cycle, or, under manual control, a direct access to memory is in progress.
  11. **I/O** indicates the processor is executing an I/O read or write cycle (PCC) or, under manual control, a direct access to I/O is in progress.
  12. **DA** indicates a direct access to memory or I/O is in progress.
  13. **READ/INPUT** indicates a memory or input read operation is in progress.
  14. **WRITE/OUTPUT** indicates a memory or output write operation is in progress.
  15. **INT** indicates an interrupt cycle is in progress.
  16. **STACK** not applicable.
- **ADDRESS** is a display of memory and I/O address.
  17. **INDICATORS** 14-15 not applicable.
  18. **INDICATORS** 0-13 are a display of the address of memory being accessed during a Fetch, Read, Write, or during manual MEM ACCESS.
  19. **INDICATORS** 9-13 are a display of the I/O address during an input, an output, or during a manual I/O ACCESS.

- **INSTRUCTION/DA** or data.
  20. **INDICATORS** 0-7 ar between the processo
- **REGISTER/FLAG D** sor data bus during e play is dependent up
  21. **INDICATORS** 0-7 ar bus when the instruct instructions, the cont Flags C, P, Z, and S ar in the lower four bits, executed.
- **ADDRESS/DATA** Th of address or data for the processor (see SE
  22. **MEM ADDRESS HIGI** for direct access or sea
  23. **I/O ADDRESS** The fi is entered here.
  24. **SENSE DATA** Data to operation is entered h



- **INSTRUCTION/DATA** is a display of the instruction or data.

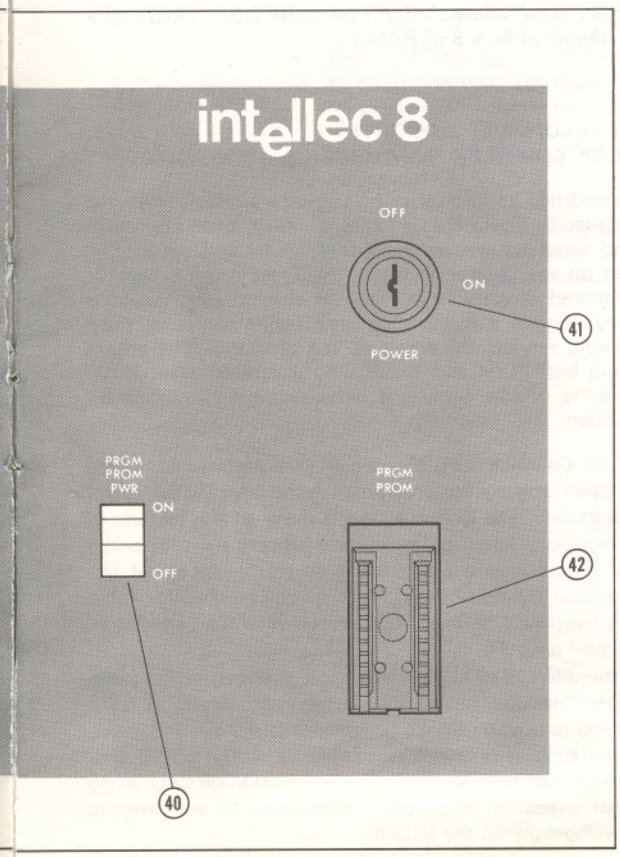
  20. **INDICATORS** 0-7 are a display of the instruction or data between the processor and memory or I/O.

- **REGISTER/FLAG DATA** is the display of the processor data bus during executions of an instruction (display is dependent upon instruction being executed).

  21. **INDICATORS** 0-7 are a display of the contents of the CPU data bus when the instruction is executed. In the case of move instructions, the contents of the source register is displayed. Flags C, P, Z, and S are a special case. The flag status appears in the lower four bits, only when an input instruction is executed.

- **ADDRESS/DATA** These eight switches provide entry of address or data for manual or SENSE operation of the processor (see SENSE 30).

  22. **MEM ADDRESS HIGH** The upper six bits of memory address for direct access or search operations are entered here.
  23. **I/O ADDRESS** The five bit I/O address for manual I/O ACCESS is entered here.
  24. **SENSE DATA** Data to be input during a SENSE mode operation is entered here (see SENSE 30).



- **ADDRESS/INSTRUCTION DATA** These eight switches provide entry of data, address, and instructions during manual or interrupt operation of the processor.

  25. **MEM ADDRESS LOW** The lower eight bits of memory address for direct access or search mode operation are entered here.
  INT INST During an interrupt cycle the interrupt instruction is fetched from here (see INT 38).
  DATA Data for deposit to memory or an output port during manual operation is entered here (see DEP 36 , and DEP AT HLT 37).
  PASS COUNT Data to be loaded into the pass count register is entered here (see LOAD PASS 26.).

- **ADDRESS CONTROL** These four switches control addressing of memory and I/O and loading of the search address during manual operation of the processor.

  26. **LOAD PASS** Loads pass count into pass count register (PASS COUNT is the number of times the processor will iterate through the search address during a search operation before indicating SEARCH COMPLETE (see SEARCH-WAIT 33 and SEARCH COMPL 5)
  27. **DECR** decrements the loaded address by one (see LOAD 29).
  28. **INCR** increments the loaded address by one (see LOAD 29).
  29. **LOAD** loads contents of address high and low into memory access register for manual direct access to memory or search mode operation (see MEM ACCESS 32 , and SEARCH-WAIT 33).

- **MODE** These five switches select the processor's mode of operation.

  30. **SENSE** causes the processor to input data from the SENSE DATA switches during execution of an input instruction instead of the addressed input port (see SENSE DATA 24).
  31. **I/O ACCESS** provides access to any input port and control of any output port when the processor is in a WAIT mode.
  32. **MEM ACCESS** allows access to and control of any location in memory when the processor is in the WAIT mode.
  33. **SEARCH-WAIT** provides for execution of a program to a specific location, where the processor enters a wait mode and displays current system conditions.
  34. **WAIT** causes the processor to go into a manual WAIT mode.

- **CONTROL** These five switches provide operator control of the processor.

  35. **STEP/CONT** provides single step execution of a program while the processor is in a WAIT mode or continuation of a program from the SEARCH COMPLETE condition.
  36. **DEP** deposits an 8-bit word to memory or output during a memory or I/O access operation (see DATA 25).
  37. **DEP AT HLT** deposits an 8-bit word to a selected memory location or output automatically during a programmed HALT (see DATA 25).
  38. **INT** causes the processor to execute an interrupt cycle, fetching the interrupt instruction from the INT INST switches (see INT INST 25).
  39. **RESET** causes processor to begin execution of program at memory location zero by resetting program counter to zero. All other registers remain unchanged.

- **POWER and PROM PROGRAMMING**

  40. **PRGM PROM PWR** Power switch for high voltage used with PROM programmer.
  41. **POWER** Key operated main power switch
  42. **PRGM PROM** Zero insertion force socket for 1602A or 1702A PROM to be programmed

SYSTEMS SOFTWARE

The Intellec 8 and Bare Bones 8 Microcomputer Development Systems come with three pieces of software: Resident System Monitor, Text Editor and Symbolic Assembler. The Text Editor and Assembler are supplied on paper tape and are loaded with the System Monitor.

SYSTEM MONITOR

- Loads and punches paper tape
- Displays and alters contents of memory
- Fills memory with constants
- Executes programs in memory
- Moves blocks of data in memory
- Programs 1602A or 1702A PROMs

The System Monitor is contained in eight 1702A PROMs and is assigned to the upper 2k words of memory, leaving the lower 14k of memory for program and data storage. This executive software allows the operator to load and punch BNPF or hexadecimal format tapes, display and alter memory, load constants to memory, move blocks of RAM memory, and execute user programs.

The System Monitor is extended by the control software for the imm6-76 programmer module, which gives the monitor the ability to program 1602A to 1702A PROMs as well as being able to load memory from already programmed PROMs for duplication and verify the contents of PROMs against master tapes.

TEXT EDITOR

- Edits symbolic data from paper tape with data from operator's terminal
- Edited output is available via paper tape
- Appends text to editor input buffer
- Moves pointer to any desired location
- Finds and inserts or substitutes strings
- Deletes lines selectively

The Text Editor allows the operator to edit his source code, making corrections and additions. He may append code, delete code, locate strings, insert strings, substitute strings and output edited code via paper tape. The text editor runs on a minimum Intellec 8 system with teletype I/O. (Requires a minimum of 8k x 8 of RAM.)

ASSEMBLER

- Standard symbolic assembler
- Input via prepunched paper tape
- Output in 8008 object code

The Symbolic Assembler is a multiple pass type. During Pass 1 the assembler reads the source code from the paper tape and generates a symbol table for later use. During Pass 2 the assembler generates the assembly listing. Also at this time, any detectable errors such as undefined jumps or missing symbols are indicated by a diagnostic printout on the teletype. Pass 3 may now be run. It generates object code, and punches it on paper tape. [**Requires a minimum of 8k x 8 of RAM.**]

DEVELOPMENT SUPPORT: PL/M™ COMPILER, ASSEMBLER and SIMULATOR

In addition to the standard software available with the Intellec 8, Intel offers a PL/M compiler, cross assembler, and simulator written in FORTRAN IV and designed to run on any large scale computer. These routines may be procured directly from Intel, or alternatively, designers may contact a number of nation-wide computer time-sharing services for access to the programs. The output from both PL/M and the MCS-8™ Assembler may be run directly on the Intellec 8 Microcomputer Development System.

PL/M Compiler: PL/M is a high level procedure-oriented systems language for programming the Intel MCS-8 microcomputer. The language retains many of the features of a high-level language, without sacrificing the efficiencies of assembly language. A significant advantage of this language is that PL/M programs can be compiled for either the Intel 8008 or Intel 8080 processors without altering the original program.

Assembler: The MCS-8 Assembler generates object codes from symbolic assembly language instructions. It is designed to operate from a timeshared terminal.

Simulator: The MCS-8 Simulator, called INTERP/8, provides a software simulation of the Intel 8008 CPU, along with execution monitoring commands to aid program development for the MCS-8.

Word Size:

Memory Size:

Instruction Set:

Machine Cycle Time:
System Clock:
I/O Channels:

Interrupt:
Direct Access to Memor
Memory Cycle Time:
Operating Temperature
DC Power Supplies:
(standard Intellec 8)

DC Power Requirement

AC Power Requirement
(standard Intellec 8)

Physical Size:

BUS INTERF
FRONT PANEL CON

SYSTEMS SPECIFICATIONS

Word Size:	Data: 8 bits Instruction: 8, 16, or 24 bits
Memory Size:	10k bytes Intellec 8/6k bytes Bare Bones expandable to 16k bytes
Instruction Set:	48, including: conditional branching, binary arithmetic, logical, register-to-register and memory reference operations
Machine Cycle Time:	12.5 μs
System Clock:	Crystal controlled at 800 kHz ±0.01%
I/O Channels:	4 expandable to 8 input ports; 4 expandable to 24 output ports — TTL Compatible
Interrupt:	Single Level
Direct Access to Memory:	Standard via control console
Memory Cycle Time:	1 μs
Operating Temperature:	0°C to 55°C
DC Power Supplies: (standard Intellec 8)	V_{CC} = 5V, I_{CC} = 12A* V_{DD} = −9V, I_{DD} = 1.8A* V_{GG} = −12V, I_{GG} = 0.06A
DC Power Requirement:	V_{CC} = 5V±5%, I_{CC} = 11A max.,6A typ. V_{DD} = −9±5%, I_{DD} = 1A max., 0.5A typ. V_{GG} = −12V±5%, I_{GG} = 0.03A max., 0.016A typ.
AC Power Requirement: (standard Intellec 8)	50-60 Hz, 115 VAC, 200 Watts *Larger power supplies may be required for expanded systems.
Physical Size:	Intellec 8: 7" x 17 1/8" x 12 1/4" (table top only) Bare Bones 8: 6 3/4" x 17" x 12" (suitable for mounting in standard RETMA 7" x 19" panel space)
Weight:	30 lb.
Standard Software:	System Monitor Resident Assembler Text Editor
Support Software:	PL/M™ Compiler, Cross Assembler, Simulator — written in FORTRAN IV

STANDARD SYSTEMS and OPTIONAL MODULES

Intellec 8 (imm8-80A) Standard System includes the following Modules and Accessories:

- Central Processor Module
- Input/Output Module
- PROM Memory Module
- RAM Memory Modules (Two)
- Chassis with Mother Board
- Power Supplies
- Control and Display Panel
- Finished Cabinet
- Standard Software:
  - System Monitor
  - Resident Assembler
  - Text Editor
- PROM Programming Module

Bare Bones 8 (imm8-81) Standard System includes the following Modules:

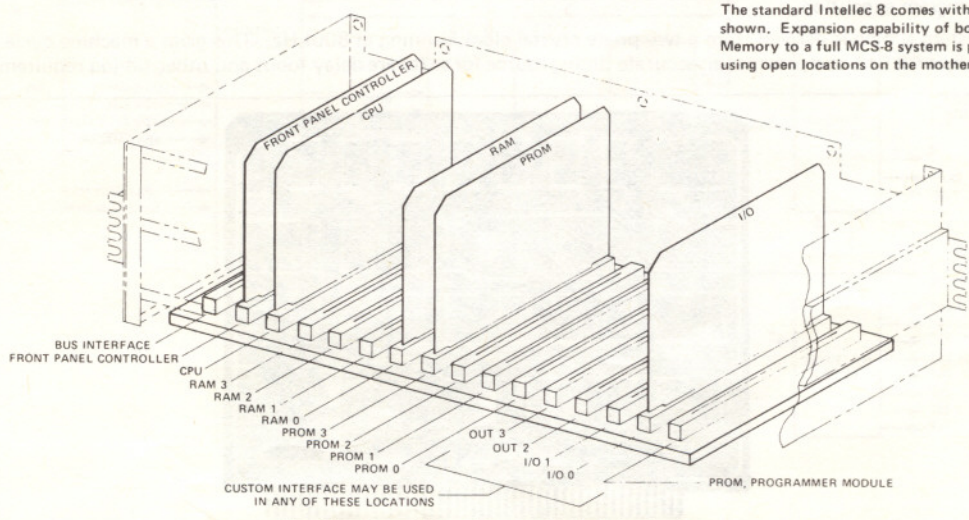
- Central Processor Module
- Input/Output Module
- PROM Memory Module
- RAM Memory Module
- Chassis (rack mountable with Mother Board)
- Standard Software:
  - System Monitor
  - Resident Assembler *
  - Text Editor *

*Requires a minimum of 8k of RAM

Optional Modules available for the Intellec 8 and Bare Bones 8:

- Additional I/O or Output Modules
- Additional RAM Memory Modules
- Universal Prototype Module
- Module Extender
- Rack mounting kit for Intellec 8

The standard Intellec 8 comes with the modules shown. Expansion capability of both I/O and Memory to a full MCS-8 system is provided by using open locations on the motherboard.



Intellec 8 and Bare Bones 8 Module Assignments

imm 8-82 CENTRAL PROCESSOR MODULE

- **Complete Central Processor Module with system clocks, interface and control for memory, I/O ports, and real time interrupt**
- **The heart of this module is Intel's 8008-1 processor on a chip — p-channel silicon gate MOS**
- **48 instructions, data oriented**
- **Accumulator and six working registers**
- **Direct addressing of up to 16,384 bytes of memory. (PROM, ROM, or RAM)**
- **Directly addresses eight input ports and twenty-four output ports**
- **Subroutine nesting to seven levels**
- **Real time interrupt capability**
- **Direct memory access capability**
- **Interface to memory, I/O and interrupt ports through separate TTL buses**
- **Two phase crystal clock — 800kHz**
- **12.5μs instruction cycle**

The imm8-82 Central Processor Module is a complete 8-bit parallel central processor unit. It contains complete control for interface to memory and I/O. This is the main module in Intel's Intellec™ 8 systems.

The imm8-82 is built around Intel's 8008-1 CPU on a chip. It executes 48 instructions including conditional branching, register to register transfers, arithmetic, logical and I/O instructions. Six 8-bit registers and an 8-bit accumulator are provided. Subroutines may be nested to seven levels. Real time interrupt capability is provided and the processor may directly address up to 16,384 bytes of memory.

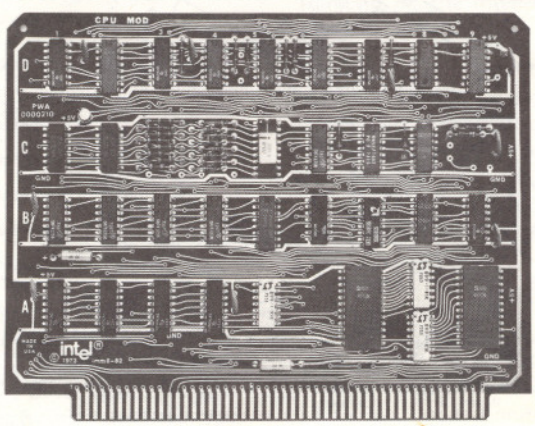
The imm8-82 has a fourteen bit TTL compatible memory address bus, an 8-bit data output bus and an 8-bit memory data input bus. Memory read and write signals and the wait request signal provide interface at TTL levels to any type of memory (including PROM, ROM, and RAM). Asynchronous interface to slower speed memories (access $> 1\mu s$) is provided by the wait request signal. This causes the processor to wait for memory response to a read or write command.

The Central Processor Module directly addresses up to eight 8-bit input ports and twenty-four 8-bit output ports. The 5-bit I/O address is contained in the upper byte of the memory address bus. Addresses 0 through 7 are defined as input ports, and 8 through 31 as output ports. Control signals, I/O cycle, I/O in and I/O out, define the I/O cycle and its function. An 8-bit data output bus and an 8-bit data input bus, both TTL compatible, provide data channels in and out of the processor module.

Real time interrupt capability and direct memory access capability complete the list of functional features for the imm8-82. During an interrupt, the Central Processor Module responds to the instruction presented at the 8-bit interrupt instruction port. Unless the main program flow is altered by the interrupt instruction, the execution will continue where it left off before processing the interrupt. Eight bits of data including sign, carry, zero and parity flags are latched on a separate bus during the execution portion of most instructions.

The direct memory access capability allows an alternate source to access memory or I/O while temporarily suspending processor operation. At the end of this alternative access to memory, the processor may return to normal program execution.

All system timing is derived from a two phase crystal clock running at 800kHz. This gives a machine cycle time of 12.5μs ± 0.01% and provides an accurate timing source for software delay loops and other timing requirements.

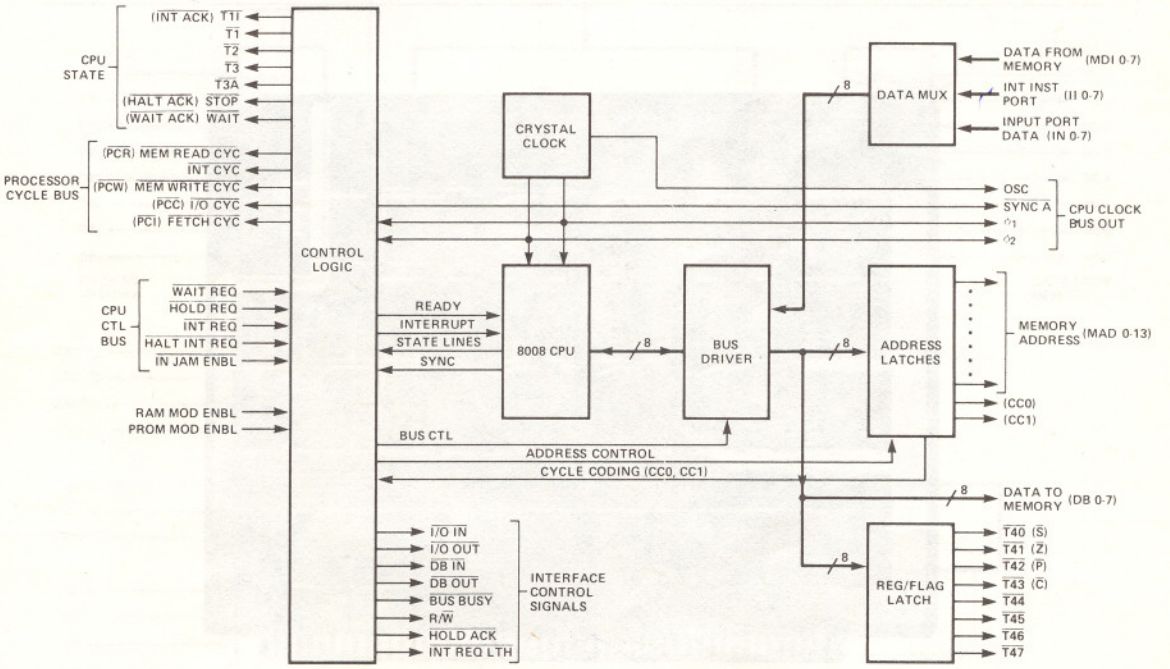


Central Processor Module

Central Processor Module Specifications

Word Size:	Instruction: 8, 16, or 24 bits Data: 8 bits
Central Processor:	8008-1 CPU, 8 bit accumulator, six 8-bit registers, subroutine nesting to seven levels, interrupt capability, asynchronous operation with memory
Instruction Set:	48 including conditional branching, binary arithmetic, logical operations, register-to-register transfers, and I/O
Memory Addressing:	Any combination of PROM, ROM and RAM up to 16,384 bytes
Memory Interface:	Address: 14 bits TTL latching bus Data: 8-bit TTL bus to and from memory
I/O Addressing:	Input: Eight 8-bit input ports Output: twenty-four 8-bit latching output ports
I/O Interface:	8-bit TTL compatible buses to and from CPU. 8-bit TTL latched bus with execution data including flags (sign, parity, zero, and carry information)
System Clock:	Crystal controlled, 800kHz ± 0.01% Processor cycle time: 12.5μs
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum
Operating Temp:	0°C to 70°C
DC Power Requirements:	V_{CC} = +5V ± 5%, I_{CC} = 2.2A max, 1.0A typical V_{DD} = −9V ± 5%, I_{DD} = 0.06A max., 0.03A typical
Support Software:	PL/M Compiler Cross Assembler Simulator Written in FORTRAN IV

imm8-82 Block Diagram

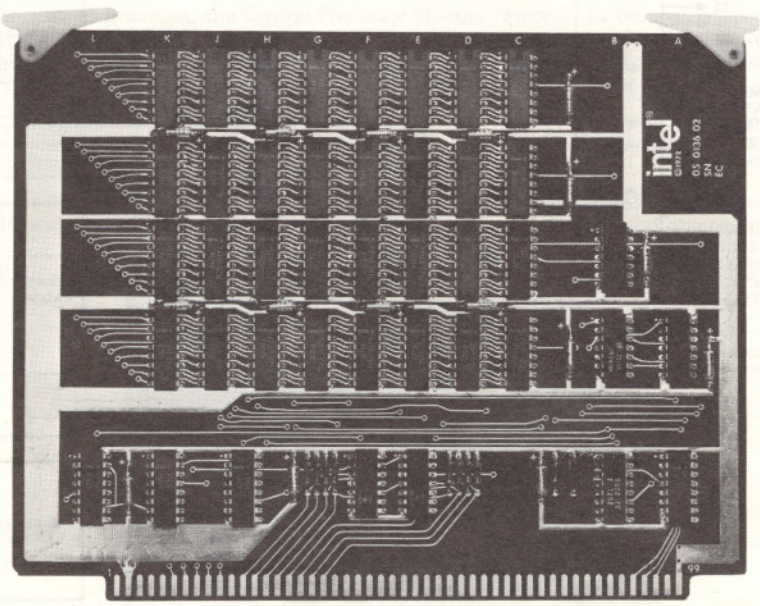


imm6-28 RAM MEMORY MODULE

- 4096 8-bit bytes per module
- Static memory, no clocks required
- Interfaces with the imm8-82 8-bit Central Processor Module
- Single +5V power supply
- Low power requirements
- For use in expansion of Intellec 8 systems to 16k bytes of memory
- Built-in decoding of module select for expansion to 65k bytes of memory

The imm6-28 RAM Memory Module is a standard 4k x 8 memory module designed for use with the Intellec 8 Microcomputer Development System. This module contains address and data buffers, read/write timing circuits and is implemented with Intel's 2102 1k x 1 static RAM. Although the basic memory module is 4096 x 8, configurations as small as 1024 x 8 are also available.

The imm6-28 RAM Memory Module is used with the MCS-8™ Micro Processor in configurations of up to 16k bytes of memory (4 modules). The imm8-82 Central Processor Module directly interfaces with the imm6-28 RAM Memory Module with all module select decoding done directly on the connector. This allows an imm6-28 to be moved to any location within the 16k of memory without making any changes in the module. This built-in decoding allows additional expansion of memory by bank switching.

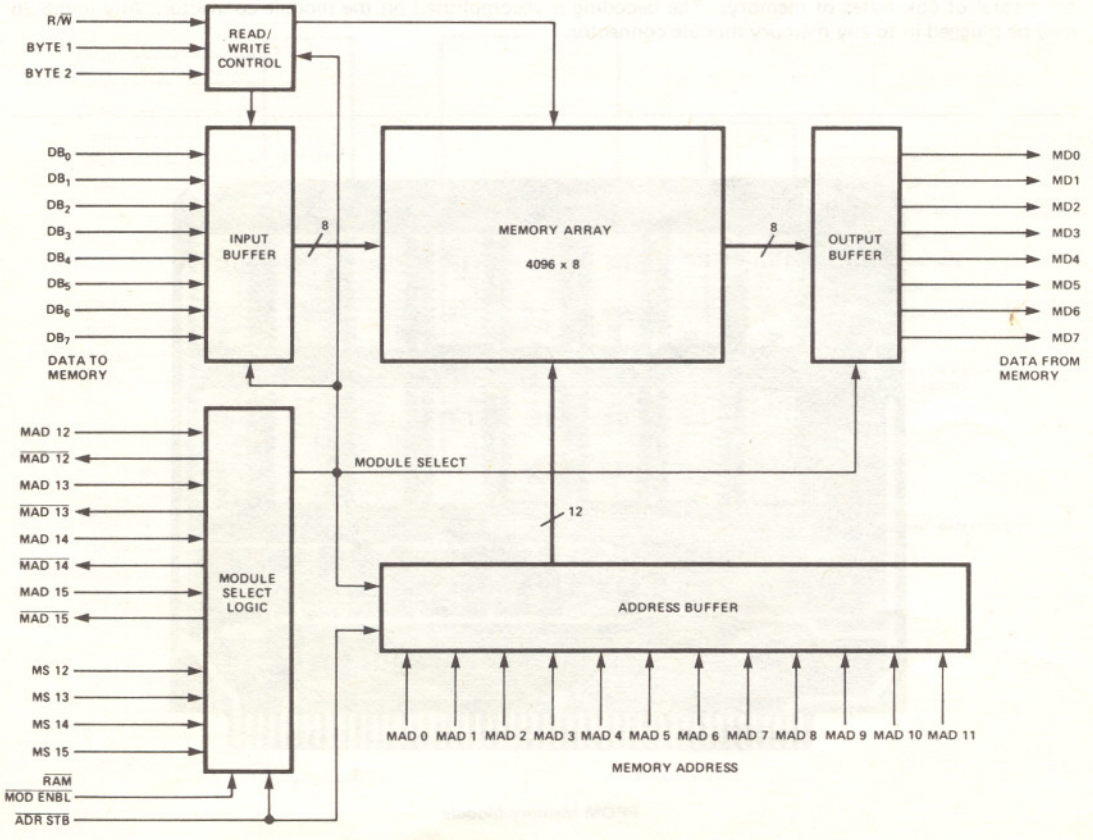


RAM Memory Module

RAM Memory Module Specifications

Memory Size:	4k bytes
Word Size:	8 bits
Memory Expansion:	To 65k bytes (16 modules)
Cycle Time:	1μs
Interface:	TTL compatible inputs; open collector outputs (positive true logic)
Capacity:	4096 bytes
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.
Operating Temperature:	0°C to 70°C
DC Power Requirement:	V_{CC} = +5V ± 5%, I_{CC} = 2.5A max., 1.25A typical

imm6-28 Block Diagram



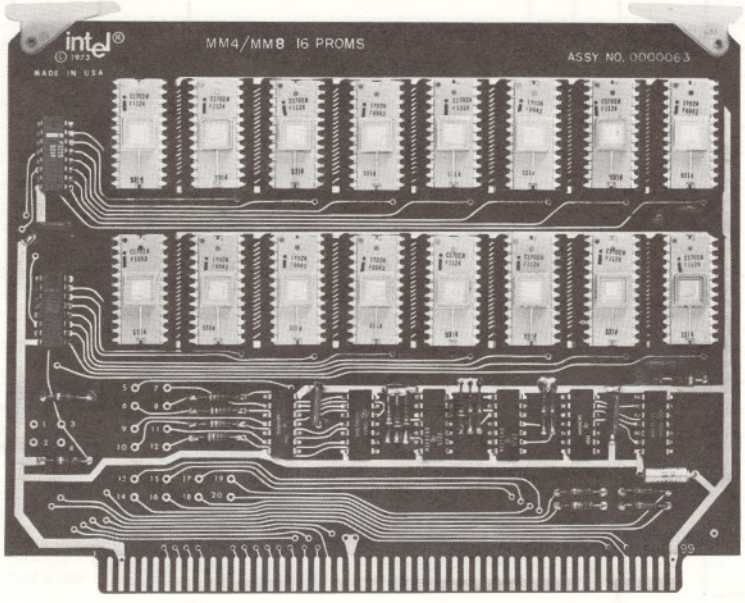
imm6-26 PROM MEMORY MODULE

- **Provides sockets for up to sixteen PROMs (4096 x 8)**
- **Static memory, no clocks required**
- **Interfaces with imm8-82 8-bit Central Processor Module**
- **Accepts Intel 1602A or 1702A PROMs or 1302 ROMs**
- **Logic to allow any mix of PROM in 256 byte (8-bits) increments with RAM to 16k when used with the imm8-82 8-bit Central Processor Module**
- **Built in decoding of module select for expansion to 65k of memory**

The imm6-26 PROM Memory Module may be used with the imm8-82 8-bit Central Processor Module for non-volatile program storage. Each PROM Memory Module has sockets for from one to sixteen of Intel's 1602A or 1702A PROMs. In addition, the 1302 mask programmed ROM may be used in place of the PROMs in OEM applications.

The PROM Memory Module is used for program storage and look-up-tables with the MCS-8™ 8-bit Micro Processor. It interfaces directly with the imm8-82 Central Processor Module and may be used with the imm6-28 RAM Memory Module in any combination to 16k bytes. Special control logic on the imm6-28 module allows any mix of PROM and RAM in a system in 256 byte increments.

For memories larger than 4k bytes, decoding on the module allows addressing of up to sixteen imm6-28 modules for a total of 65k bytes of memory. The decoding is accomplished on the module connector. Any imm6-26 may be plugged in to any memory module connector.



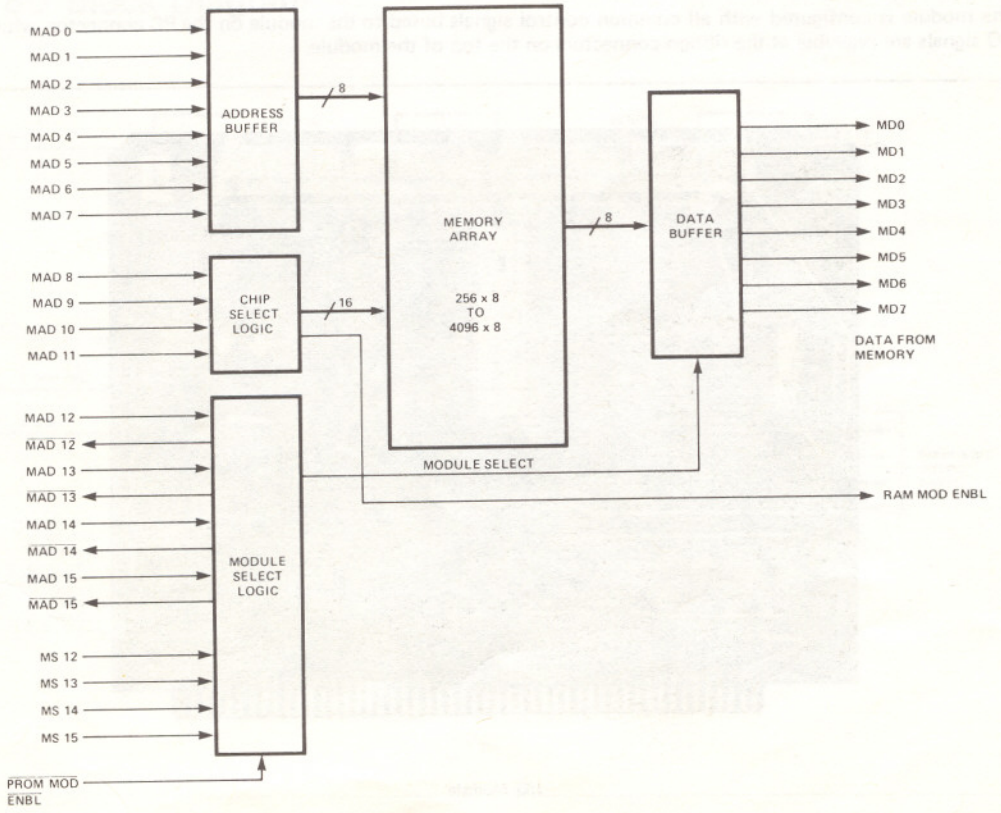
PROM Memory Module

PROM Memory Module Specifications

Memory Size:	4k bytes
Word Length:	8 bits
Memory Expansion:	To 65k bytes (16 modules)
Interface:	TTL compatible inputs; open collector outputs (positive true logic)
Capacity:	256 to 4096 bytes in 256 byte increments
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.
Operating Temperature:	0°C to 70°C
DC Power Requirement:	V_{CC} = +5V ±5% I_{CC} = 1.6A max., 1.1A typical(1) V_{DD} = −9V ±5% I_{DD} = 1.6A max., 1.0A typical(1)

(1)Board loaded with all 16 PROMs.

imm6-26 Block Diagram



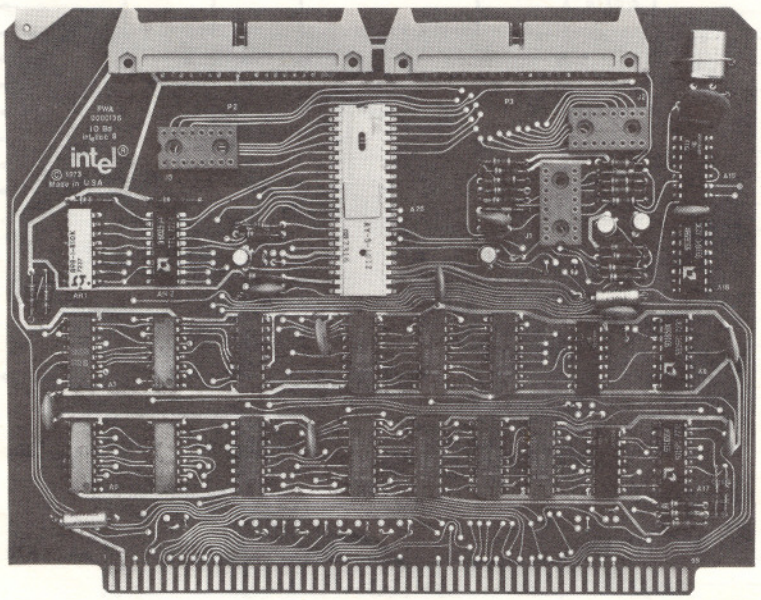
imm8-60 INPUT/OUTPUT MODULE

- Four 8-bit input ports and four 8-bit latching output ports
- TTL compatible
- Interfaces directly with imm8-82 Central Processor Module
- Teletype asynchronous transmitter/receiver and controls on board
- Transmission rates of 110 or 1200 baud
- Crystal clock for asynchronous transmitter/receiver
- Capable of high speed serial communications to 9600 baud

The imm8-60 I/O Module provides four 8-bit TTL compatible input ports and four 8-bit TTL compatible latching output ports. It interfaces directly with the imm8-82 Central Processor Module. Built-in decoding on the board provides for expansion of I/O to the maximum with the addition of one imm8-60 and two imm8-62 Output Modules (eight input ports and twenty four output ports).

For more efficient use of the imm8-82 Central Processor, an asynchronous transmitter receiver is included in the module. This frees the processor of time-consuming bit manipulation during bit serial data transmission. The transmitter receiver operates at either 110 or 1200 baud and by alteration of the basic clock frequency, data rates to 9600 baud may be obtained. The module contains drivers and receivers for connection to a teletype. These may be used with the asynchronous transmitter receiver or directly with I/O ports for bit serial transmission and reception of teletype data.

The module is configured with all common control signals bused to the module on the PC connector, while all I/O signals are available at the ribbon connectors on the top of the module.

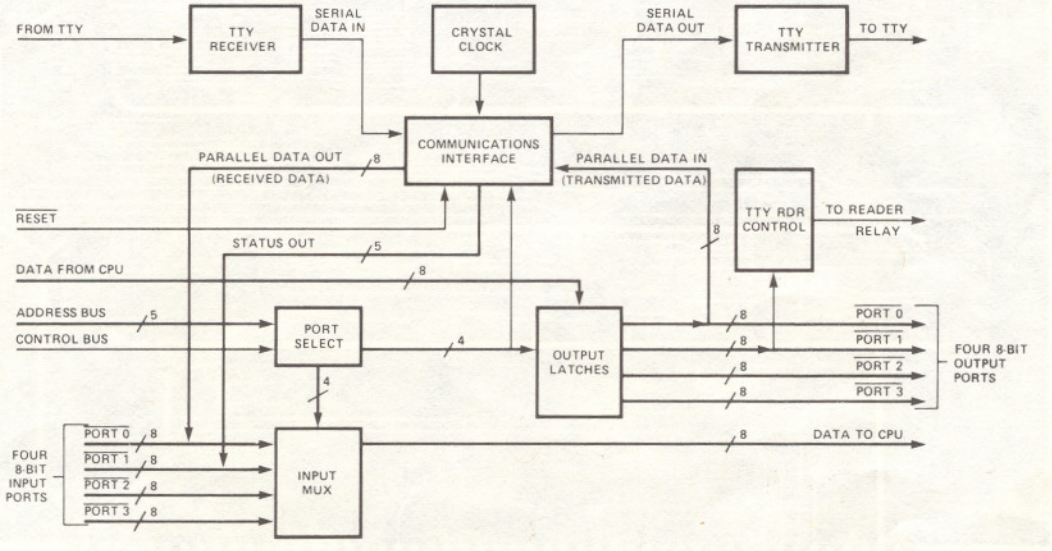


I/O Module

I/O Module Specifications

Word Size:	8 bits
Capacity:	Four 8-bit input ports, four 8-bit output ports
I/O Interface:	Input ports: TTL compatible (complement Data In) Output ports: TTL compatible (complement Data Out) Communications Interface: Direct: TTL compatible input and output TTY: 20mA TTY interface with discrete transmitter and receiver TTY RDR Control: Discrete relay interface
Serial Communication Rate:	Crystal controlled to 110 or 1200 baud
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC Ribbon Type P/N 3417 from 3M
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.
Operating Temperature:	0°C to 70°C
DC Power Requirement:	V_{CC} = +5V ± 5%, I_{CC} = 0.820A max., 0.478A Typical V_{DD} = −9V ± 5%, I_{DD} = 0.080A max., 0.050 Typical V_{GG} = −12V ± 5%, I_{GG} = 0.030A max., 0.016A Typical

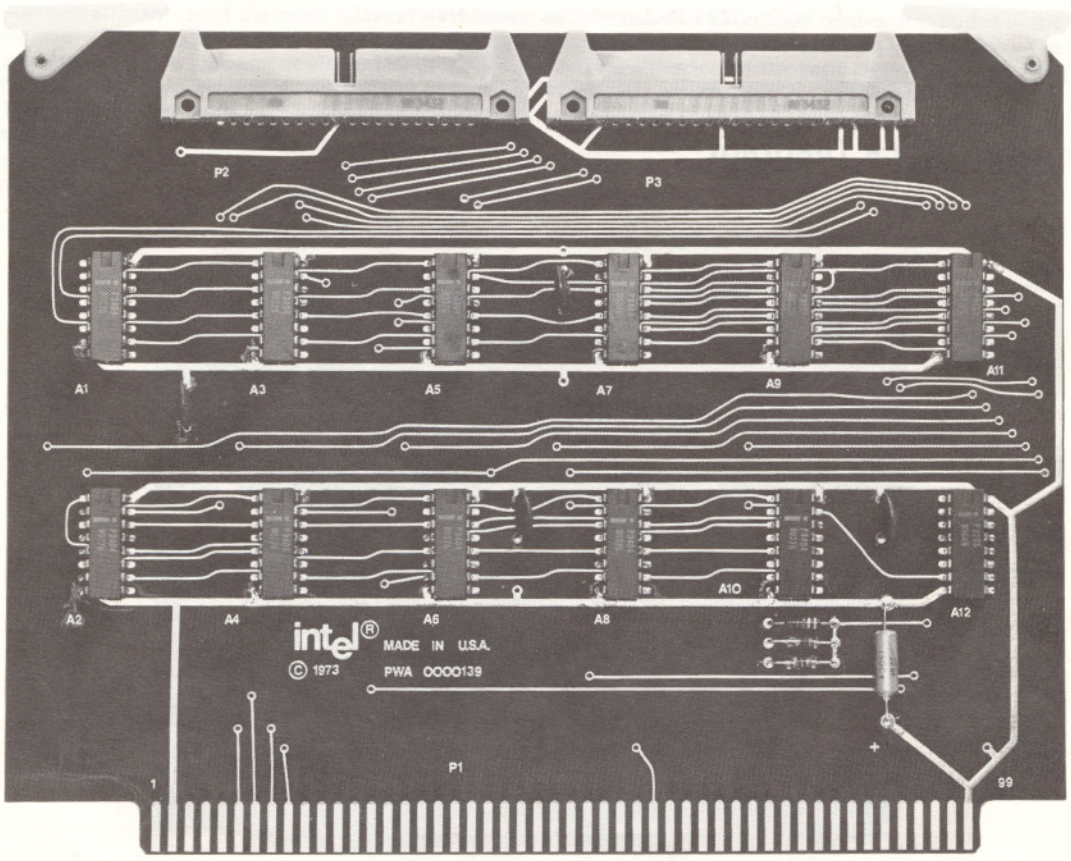
imm 8-60 Block Diagram



imm8-62 OUTPUT MODULE

- Eight 8-bit Latching Output Ports
- Interfaces Directly with imm8-82 CPU Module
- Decoding for Expansion to Full Output Complement
- TTL Compatible

The imm8-62 Output Module provides eight 8-bit latching output ports for direct interface with the imm8-82 CPU Module. Each port is individually addressable, and all outputs are TTL compatible. The module address includes decoding for expansion to a full complement of 24 output ports. This may be accomplished by using two imm8-60 I/O Modules and two imm8-62 Output Modules. All output signals are available through a ribbon connector at the top of the module.



Output Module

intel®

Output Modul

Word Size:
Capacity:
Interface:
Connector:

Board Dimensions:
Operating Temperat
DC Power Requirem

imm8-62 Bloc

DATA FROM CPU

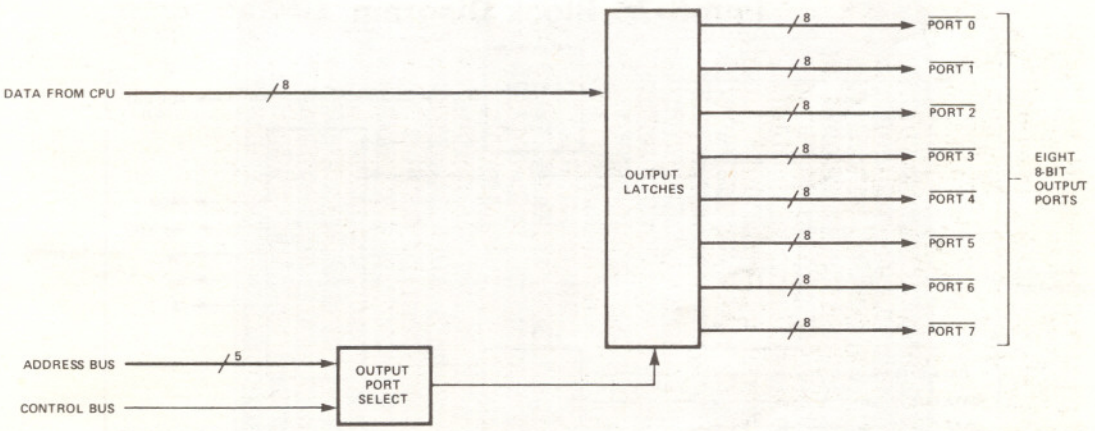
ADDRESS BUS

CONTROL BUS

Output Module Specifications

Word Size:	8-bits
Capacity:	Eight 8-bit latching output ports
Interface:	TTL compatible (complement Data Out)
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC Ribbon Type P/N 3417 from 3M
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.
Operating Temperature:	0°C to 70°C
DC Power Requirement:	V_{CC} = +5V ± 5%, I_{CC} = 0.840A max., 0.420A typical

imm8-62 Block Diagram



imm6-76 PROM PROGRAMMER MODULE

- High speed programming of Intel's 1702A or 1602A PROM
- All necessary timing and level shifting included
- Direct interface with Intel's Intellec 8 Microcomputer Development System
- Complete software necessary for use included with Intellec® 8 system monitor

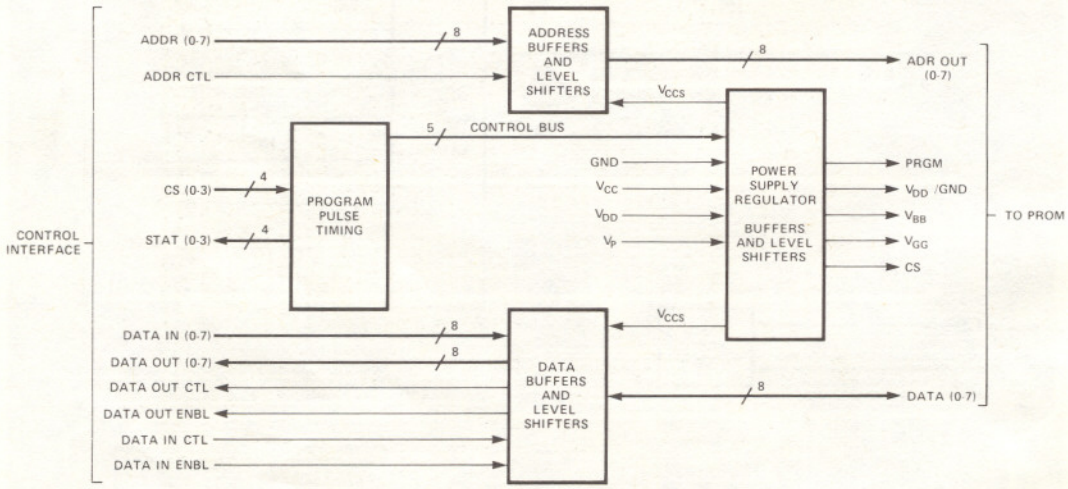
The imm6-76 PROM Programmer Module provides all necessary hardware and software to add PROM programming capability to the Intellec 8 microcomputer development system.

The module has been designed to slip into the Intellec 8 and provides all connections to the zero insertion force socket on the front panel. All required timing and level shifting is accomplished on the module utilizing the high voltage power supply already located in the Intellec 8.

Software to control programmer operation is included as part of the Intellec 8 system monitor. This software is specifically written for the Intellec 8 and allows both programming and verification of 1602A and 1702A PROMs. In addition, the contents of any PROM may be listed or unloaded into memory for duplication.

The imm6-76 may also be used as a stand alone PROM programmer with toggle switches or with another computer providing data address and control signals.

imm6-76 Block Diagram



PROM Programmer Module Specifications

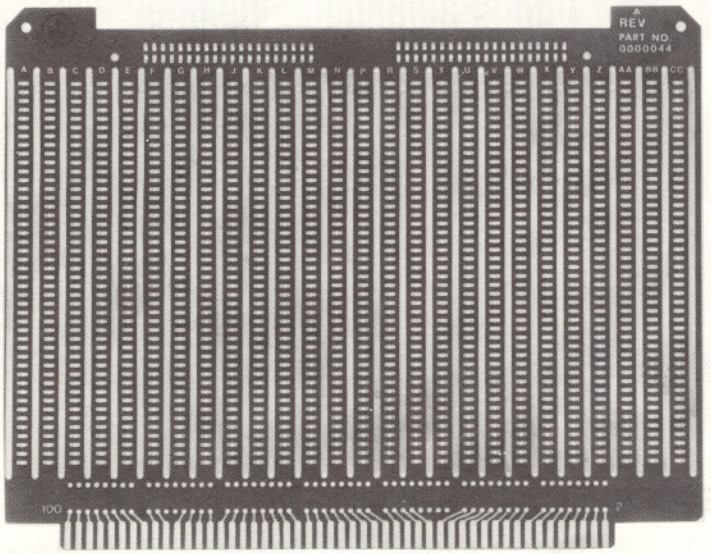
System Interface:	All inputs and outputs are TTL compatible and available at the ribbon connector at the top of the module. Control for either "True" or "False" data is provided. Direct interface to Intellec 8.
Control Software:	Included in the Intellec 8 executive monitor.
Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC Ribbon Type P/N 3417 from 3M
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers min.
Operating Temperature:	0°C to +55°C
DC Power Requirements:	V_{CC} = +5V ± 5%, I_{CC} = 0.8A max., 0.5A typical V_{DD} = −9V ± 5%, I_{DD} = 0.1A max., 0.08A typical V_P = +50V, I_P = 1.0A max.

imm6-70 UNIVERSAL PROTOTYPE MODULE

- Provides breadboard capability for developing custom interfaces
- Standard size of all microcomputer modules
- Will accept two 3M 40 pin ribbon connectors on top of module for direct I/O connections
- Will accept standard wirewrap sockets with 0.1 in. x 0.3 in. or 0.1 in. x 0.6 in lead spacing
- Capacity for 60 16-pin or 14-pin sockets or 24 24-pin sockets
- All power is bused on board. Pins on PC connector and pins to individual sockets are uncommitted for maximum flexibility

The imm6-70 Universal Prototype Module is a standard size microcomputer module with power buses which interface with the Intellec® 8. It provides a standard format for prototyping both customer interface and system control.

The module will accept dual in-line packaged components having pin center-to-center dimensions of 0.100 inch by 0.300 inch or 0.100 inch by 0.600 inch. These parts should be mounted in standard wirewrap sockets.



Universal Prototype Module

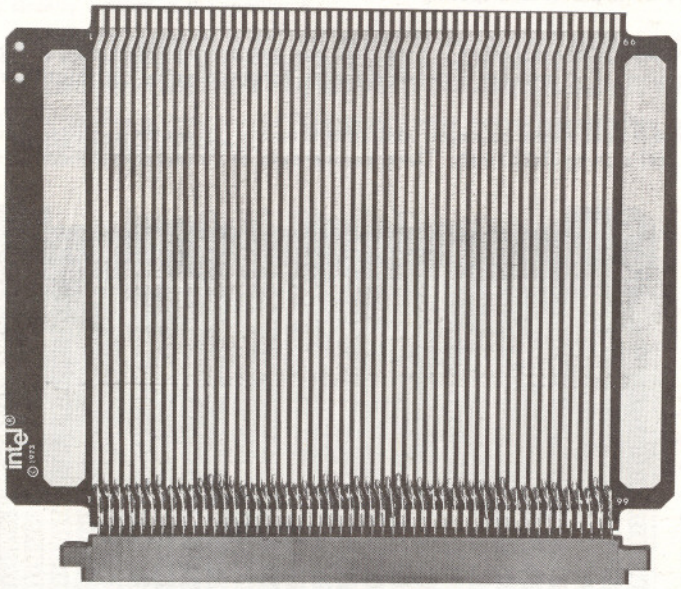
Universal Prototype Module Specifications

Capacity:	60 16-pin or 14-pin sockets or 24 24-pin sockets. Standard wirewrap sockets with pins on 0.100 in. by 0.300 in. centers or 0.100 in. by 0.600 in. centers. Board spacing dependent on components and sockets used.
Connector:	Dual 50-pin on 0.125 in. centers. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC Ribbon Type P/N 3417 from 3M
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.

imm6-72 MODULE EXTENDER

- Allows any module to be extended for ease of debugging, testing, and maintenance
- Standard dual 50-pin configuration for use with all microcomputer modules

The imm6-72 Module Extender is designed to be used with the Intellec® 8 system. It allows the operator to extend any module out of the cage for servicing while maintaining all electrical connections.



Module Extender

Module Extender Specifications

Connector:	Dual 50-pin on 0.125 in. centers. Connectors in rack must be positioned on 0.5 in. centers min. Wirewrap P/N C800100 from SAE P/N VPB01C50E00A1 from CDC Extending connector is mounted on board.
Board Dimensions:	6.18 in. x 8.0 in. x 0.062 in. Board to be on 0.5 in. centers minimum.

imm8-90 Intellec® 8 High Speed Paper Tape Reader

The imm8-90 high speed paper tape reader provides all Intellec 8 Microcomputer Development Systems with a high speed paper tape input that is over twenty times faster than the standard ASR-33 teletype reader. This translates into a significantly faster development cycle due to a marked reduction in the time required for repetitive program loading, assembly, and editing operations.

The Intellec 8 monitor software provides two key capabilities which significantly enhance the systems performance of the imm8-90. A general purpose paper tape reader driver is included in the Intellec 8 Monitor. It enables all systems software to utilize the high speed reader features and is callable by user written application programs. The monitor also provides dynamic I/O reconfiguration permitting instantaneous reassignment of physical devices to logical devices.

All reader/Intellec 8 interface cables are included with the imm8-90. A fanfold tape guide (and associated installation instructions) is included to provide fanfold punch capability for the ASR-33 teletype.

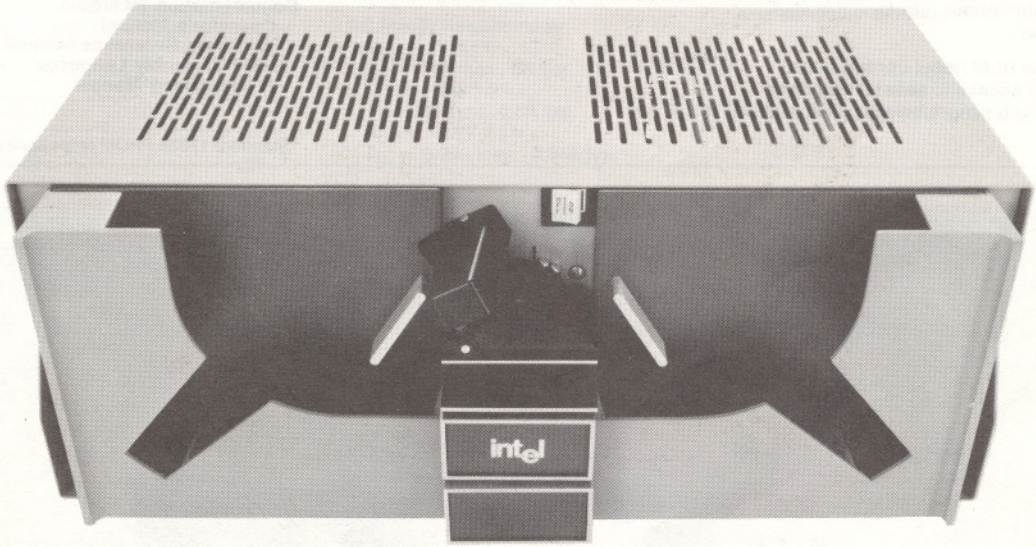
Directly compatible with all Intellec® 8 Microcomputer Development Systems

20 times faster than standard ASR-33 teletype reader

Loads any 8K Intellec® 8 program memory in less than 90 seconds

Data transfer at asynchronous rates in excess of 200 characters per second

Rack mountable or stand-alone



SPECIFICATIONS

TAPE MOVEMENT

Tape Reading Speed:
0 to 200 characters per second asynchronous

Tape Stopping:
Stops "On Character"

TAPE CHARACTERISTICS

Tapes must be prepared to ANSI X 3.18 or EMCA 10 Standards for base materials and perforations.

Reads tape of any material with thickness between 0.0027" and 0.0045" with transmissivity less than or equal to 57% (oiled buff paper tape).

Tape loading: in line

Tape width: 1 inch

PHYSICAL CHARACTERISTICS

Height	7.75 in. (19.69 cm)
Width	19.25 in. (48.90 cm)
Depth	11.62 in. (29.52 cm)
Weight	13 lbs. (5.9 kg.)

ELECTRICAL CHARACTERISTICS

AC Power Requirement:
3 wire input with center conductor (earth ground) tied to chassis. 100, 115, or 127 VAC, single phase at 3.0 amps or 220 or 240 VAC and 1.5 amps; 47 to 63 Hz.

ENVIRONMENTAL CHARACTERISTICS

Temperature
Operating: 0 to 55°C (free air)
Non-operating: -55°C to +85°C

Humidity
Operating: Up to 90% relative humidity without condensation
Storage: All conditions without condensation of water or frost

EQUIPMENT SUPPLIED

Paper Tape Reader
Reader Cable
Reader Flat Cable
Fanfold Tape Guide
Fanfold Paper Tape
Hardware Manual
Installation and Operations Guide
Fanfold Guide Installation Instructions

NOTE: Version 2 software must be used when operating with Intellec® 8/Mod 8 Microcomputer Development System.

ORDERING INFORMATION	Part Number	Description
	imm8-90	High Speed Paper Tape Reader

intel®

Microcomputer Systems

imm8-88 Conversion Kit

The imm8-88 conversion kit provides an upgrade path for Intellec® 8/MOD 8 microcomputer development systems. It includes all the hardware and software products required to fully support 8080 CPU based microcomputer system development.

With the imm8-88 conversion kit installed in an Intellec 8/MOD 8, complete 8080 CPU hardware and software development capability is provided. The 8080 CPU module (imm8-83) has 78 basic instructions, six 8-bit working registers, an 8-bit accumulator, an 8-bit bidirectional data bus, and a 16-bit memory address bus which enables direct access to 64k words of memory. Control circuits for memory and input/output interface are also provided.

The new front panel control board enables access to several functions including a programmable 8-bit output port display, stack and interrupt disable indication, and full 16-bit memory address display which are not implemented on the Intellec 8/MOD 8 front panel.

Enhanced monitor capabilities include provisions to set software breakpoints and the ability to interrogate and modify CPU registers.

The conversion kit is installed by simply plugging in the three new hardware modules in the appropriate Intellec 8/MOD 8 chassis connectors and installing the new system monitor. The system can be quickly reconfigured to support 8008 CPU chip development by replacing the original boards and system monitor.

Upgrades Intellec 8/MOD 8 microcomputer development system to support complete MCS-80 microcomputer set development.

Provides all Intellec 8/MOD 80 basic and expansion capabilities.

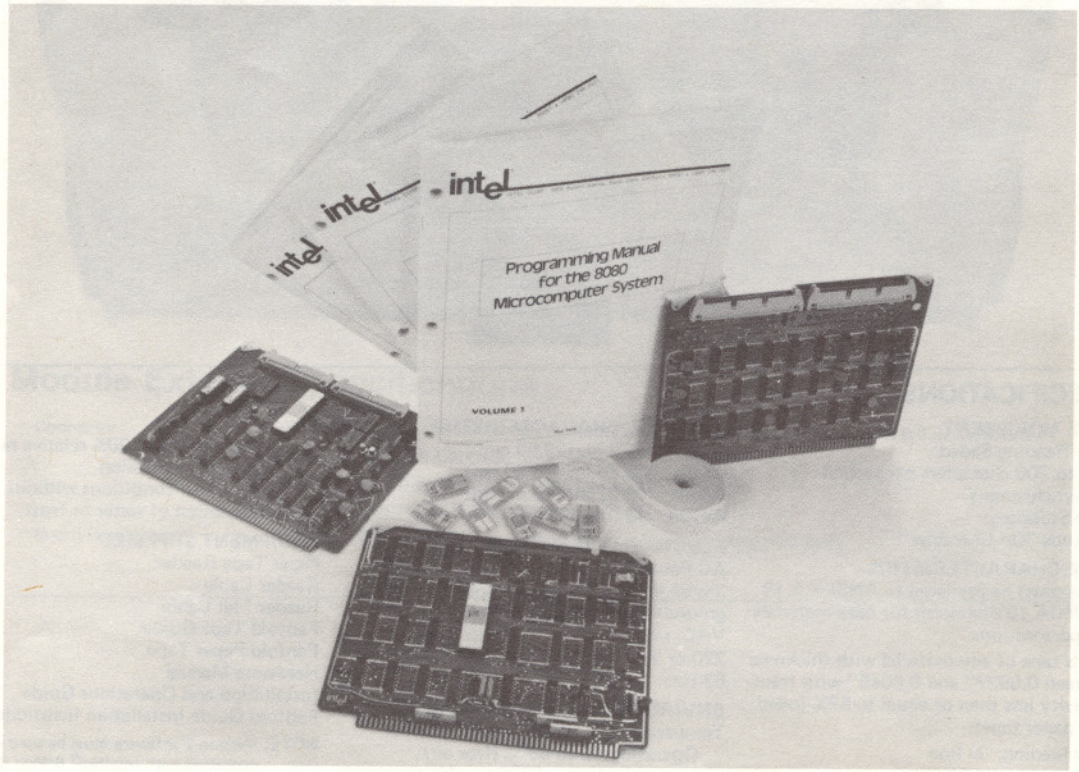
Compatible with Intellec 8/MOD 80 RAM and PROM* memory modules.

Hardware Products include:
- **imm8-83 CPU Module**
- **imm8-61 I/O Module**
- **Front Panel Control Module**

Software Products include:
- **PROM Resident System Monitor**
- **RAM Resident Macro Assembler**
- **RAM Resident Text Editor**

Documentation includes:
- **Operator's Manual**
- **Hardware Reference Manual**
- **8080 Assembly Language Programming Manual**

*Standard 1702A PROM chips must be used.



SPECIFICAT

WORD SIZE
Data: 8 bits
Instruction: 8, 16

MEMORY SIZE
10k bytes expan
within system ch
external user de

INSTRUCTION S
78, including cor
binary arithmetic
register and inpu
ence with four ad

MACHINE CYCL
2.5 μs

SYSTEM CLOCK
Crystal controlle

I/O CHANNELS
Maximum Input/
available with I/O

imm8-61
imm8-63 (with one imm8-61)

INTERRUPT
Standard via cor
signed multiple
bility available.

imm8-88 Conversion Kit

SPECIFICATIONS

WORD SIZE
Data: 8 bits
Instruction: 8, 16, or 24 bits

MEMORY SIZE
10k bytes expandable to 16k bytes within system chassis, 64k bytes in external user designed enclosures.

INSTRUCTION SET:
78, including conditional branching, binary arithmetic, logical, register-to-register and input/output memory reference with four addressing modes.

MACHINE CYCLE TIME
2.5 μs

SYSTEM CLOCK
Crystal controlled at 2 MHz $\pm$ 0.01%

I/O CHANNELS
Maximum Input/Output configuration available with I/O or Output Modules.

	Input Ports	Output Ports
imm8-61	16	16
imm8-63 (with one imm8-61)	4	28

INTERRUPT
Standard via control console. User designed multiple level interrupt capability available.

DIRECT MEMORY ACCESS
Standard via control console. User designed DMA capability available.

MEMORY ACCESS TIME
1 μs with standard memory modules. Faster access time available with user designed memory systems.

PHYSICAL CHARACTERISTICS
Intellec®8/MOD 80: 7" x 17⅛" x 12¼" (table top only)
Bare Bones 80: 6¾" x 17" x 21" (suitable for mounting in standard RETMA 7" x 19" panel space)
Weight: 30 lb. (13.61 kg)

ELECTRICAL CHARACTERISTICS
DC Power Supplies:
V_{CC} = 5V, I_{CC} = 12A*
V_{DD} = −9V, I_{DD} = 1.8A*
V_{GG} = +12V, I_{GG} = 0.06A
DC Power Requirement:
V_{CC} = 5V ±5%,
I_{CC} = 11A max., 6A typ.
V_{DD} = −9V ±5%,
I_{DD} = 1A max., 0.5A typ.
V_{CC} = +12V ±5%,
I_{GG} = 0.06A max., 0.04A typ.

AC Power Requirement:
50-60 Hz, 115/230 VAC, 200 Watts
*Larger power supplies may be required for expanded systems.

ENVIRONMENTAL CHARACTERISTICS
Operating Temperature: 0°C to 55°C

OPTIONAL MODULES
Available for the Intellec 8/MOD80 and Bare Bones 80:
imm8-61 I/O Module
imm8-63 Output Module
imm6-28 RAM Memory Module
imm6-70: Universal Prototype Module
imm6-72: Module Extender
imm6-36: Drawer Slides and Extenders for Rack Mounting

EQUIPMENT SUPPLIED:
Central Processor Module
Input/Output Module
Front Panel Control Module
PROM Resident System Monitor
RAM Resident Macro-Assembler
RAM Resident Text Editor
Complete Hardware and Software Documentation including schematics and assembly drawings.

INSTRUCTIONS FOR PROGRAM SUBMITTAL TO MCS USER'S LIBRARY

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

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

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

Ordering In

1. The 8008 (CPU) i
   ordered as C8008

2. PL/M Compiler So
   Programs for the M
   language and comp
   written in FORTR
   or directly from In

3. MCS-8 Cross Asse
   This software prog
   into machine instr
   tions by the 8008,
   and is available via

Packaging

U.S. SALES AND MARK
U.S. MARKETING HEADQU
3065 Bowers Avenue
Santa Clara, California 9505
Tel: (408) 246-7501
TWX: 910-338-0026
TELEX: 34-6372

U.S. REGIONAL SALES MAN
WESTE
William T. O'Brien
1651 E. 4th St.
Suite 228
Santa Ana, California 92701
Tel: (714) 835-9642
TWX: 910-595-1114

EUROPEAN MARKETI
EUROPEAN MARKETING H
BELGIUM
Tom Lawrence
Intel Office
216 Avenue Louise
Brussels B1050
Tel: 649-20-03
TELEX: 24814

ORIENT MARKETING
ORIENT MARKETING HEAD
JAPAN
Y. Magami
Intel Japan Corporation
Kasahara Bldg.
1-6-10, Uchikanda
Chiyoda-ku
Tokyo 101
Tel: 03-354-8251
TELEX: 781-28426

Ordering Information

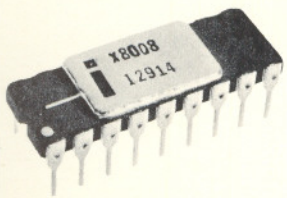
1. The 8008 (CPU) is available in ceramic only and should be ordered as C8008 or C8008-1.

2. PL/M™ Compiler Software Package
   Programs for the MCS-8 may now be developed in a high level language and compiled to 8008 machine code. This program is written in FORTRAN IV and is available via time sharing service or directly from Intel.

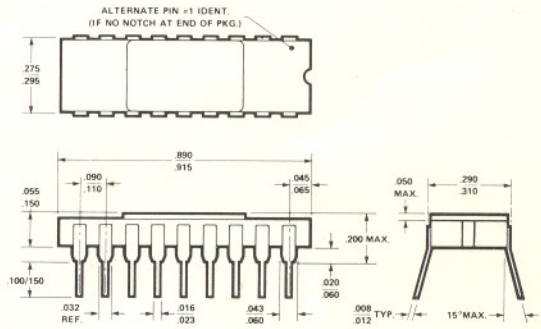
3. MCS-8™ Cross Assembler and Simulator Software Package
   This software program converts a list of instruction mnemonics into machine instructions and simulates the execution of instructions by the 8008. This program is written in FORTRAN IV and is available via time sharing service or directly from Intel.

4. Intellec® 8/MOD8
   The Intellec 8, Bare Bones 8, and microcomputer modules must be specified individually by product code.
   - imm8-80A Intellec 8 (complete table top system)
   - imm8-81 Bare Bones 8 (complete rack mountable system)
   - imm8-82 Central Processor – includes 8008-1 CPU crystal clock and interface logic
   - imm6-26 PROM Memory – includes sockets for sixteen 1702A PROMs
   - imm6-28 RAM Memory – 4k x 8 static memory
   - imm8-60 Input/Output – 4 input and 4 output ports
   - imm6-76 1702A PROM programmer and control software
   - imm6-70 Universal prototype module
   - imm6-72 Module extender
   - imm8-90 Intellec 8 High Speed Paper Tape Reader
   - imm8-88 Conversion Kit

Packaging Information



CERAMIC PACKAGE OUTLINE



U.S. SALES AND MARKETING OFFICES

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

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

U.S. REGIONAL SALES MANAGERS' OFFICES

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

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

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

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

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

EUROPEAN MARKETING OFFICES

EUROPEAN MARKETING HEADQUARTERS

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

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

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

EUROPEAN MARKETING OFFICES

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

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

ORIENT MARKETING OFFICES

ORIENT MARKETING HEADQUARTERS

JAPAN
Y. Magami
Intel Japan Corporation
Kasahara Bldg.
1-6-10, Uchikanda
Chiyoda-ku
Tokyo 101
Tel: 03-354-8251
TELEX: 781-28426

ORIENT DISTRIBUTORS

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

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

Printed in U.S.A./MCS-348a-0475/25K