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

8008

8-BIT PARALLEL CENTRAL PROCESSOR UNIT

MCS-8 T.M.

MICRO COMPUTER SET

USERS MANUAL

INCLUDES the 8008-1, SIM8-01, MP7-03, and the Bootstrap Loader

NOV. 1972

REV. 2

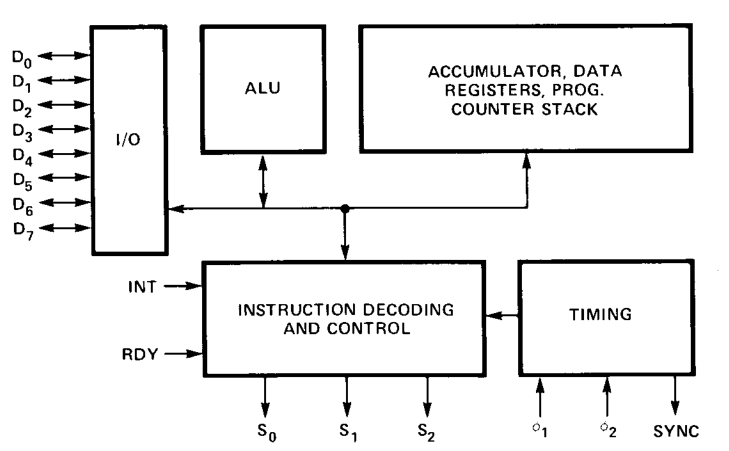
8008
8 Bit Parallel Central Processor Unit

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.

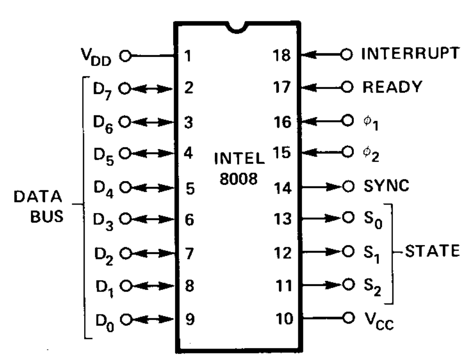
Features

- 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

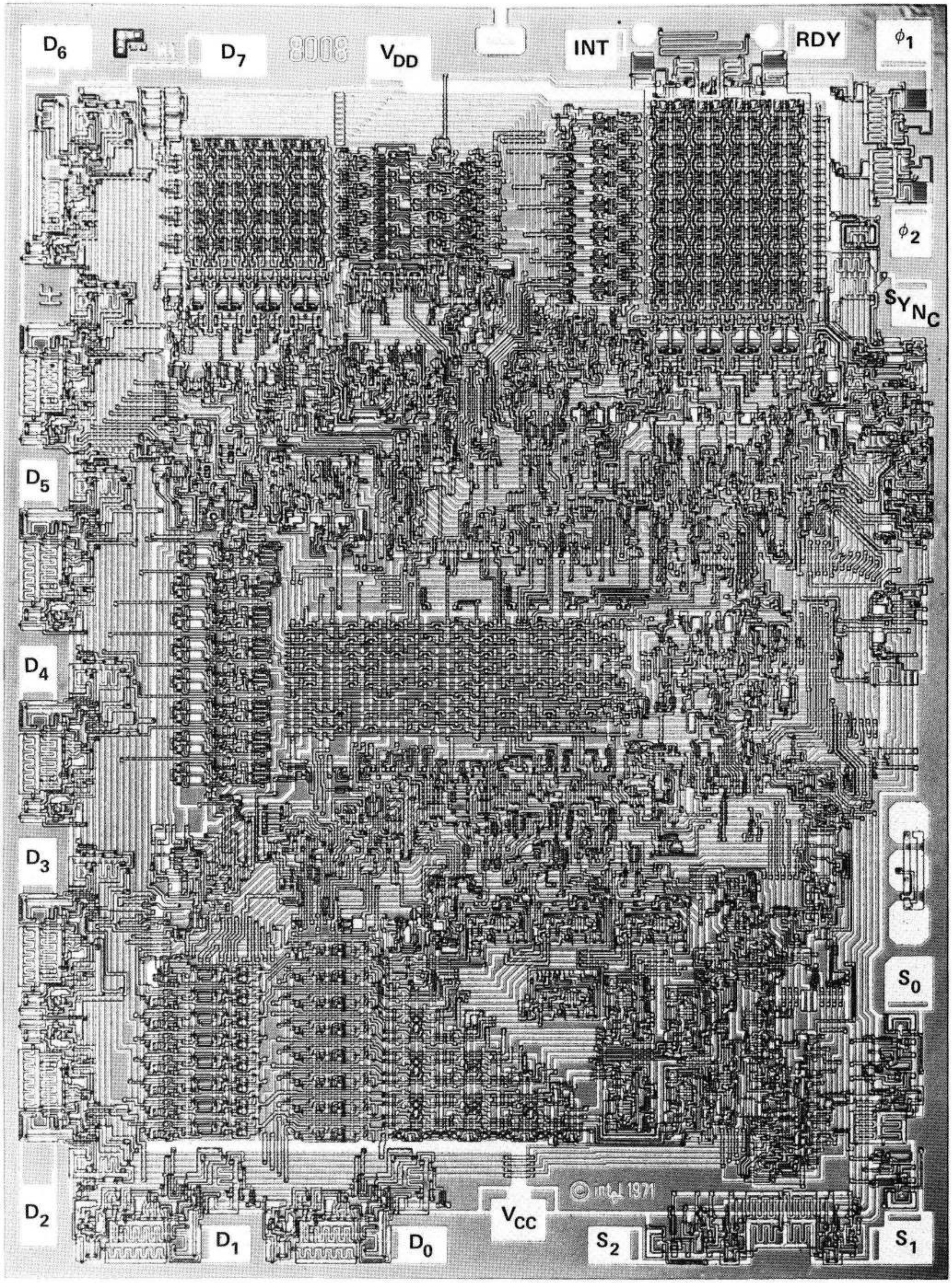


PIN CONFIGURATION



CONTENTS

APPENDIX



8008 Photomicrograph With Pin Designations

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), 1301, 1601, 1701 (ROMs), 1404, 2401 (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. Presently operating with a 500 kHz clock, the CPU executes non-memory referencing instructions in 20 microseconds.

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

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

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

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

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.

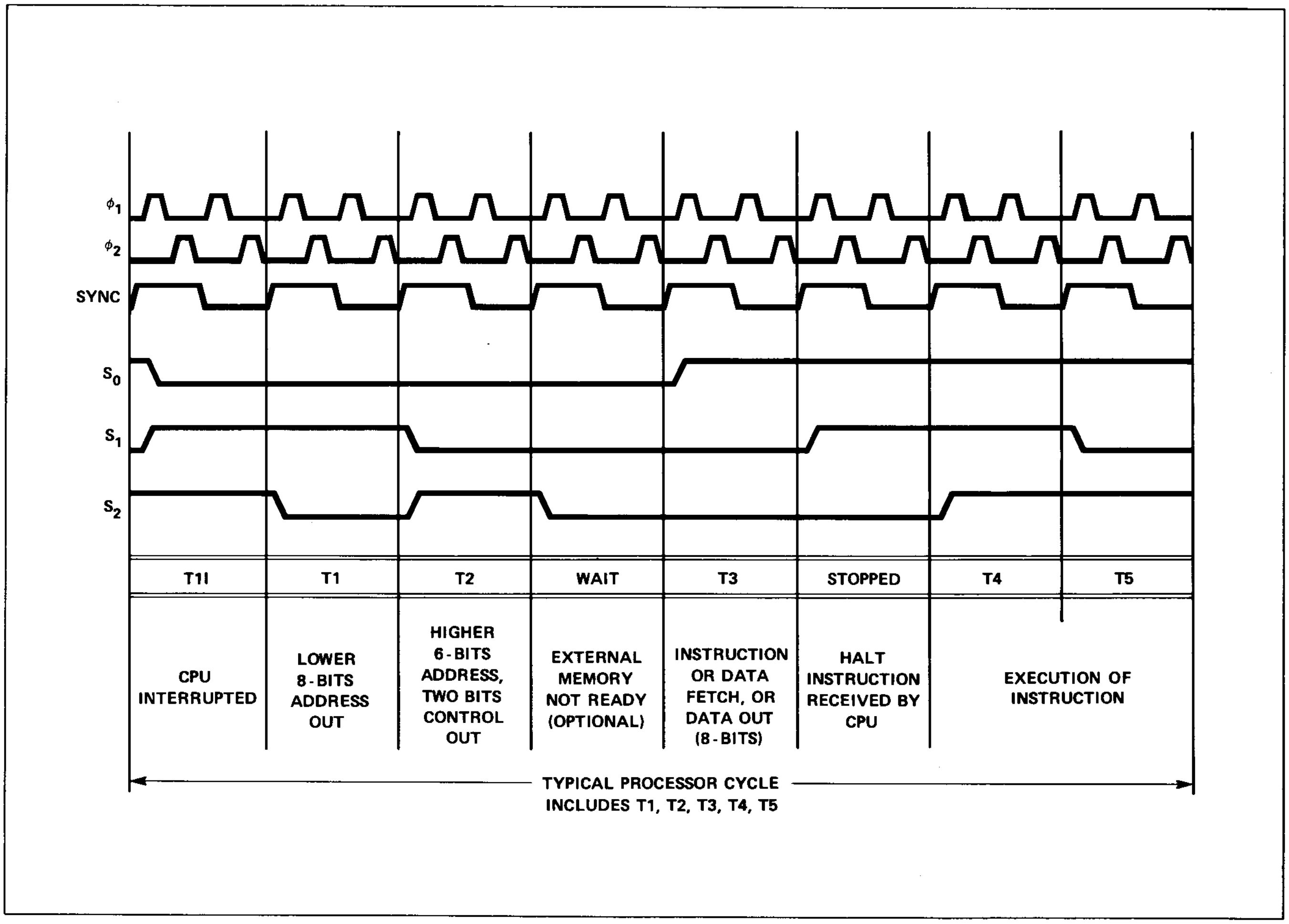


Figure 1. Basic 8008 Instruction Cycle

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.

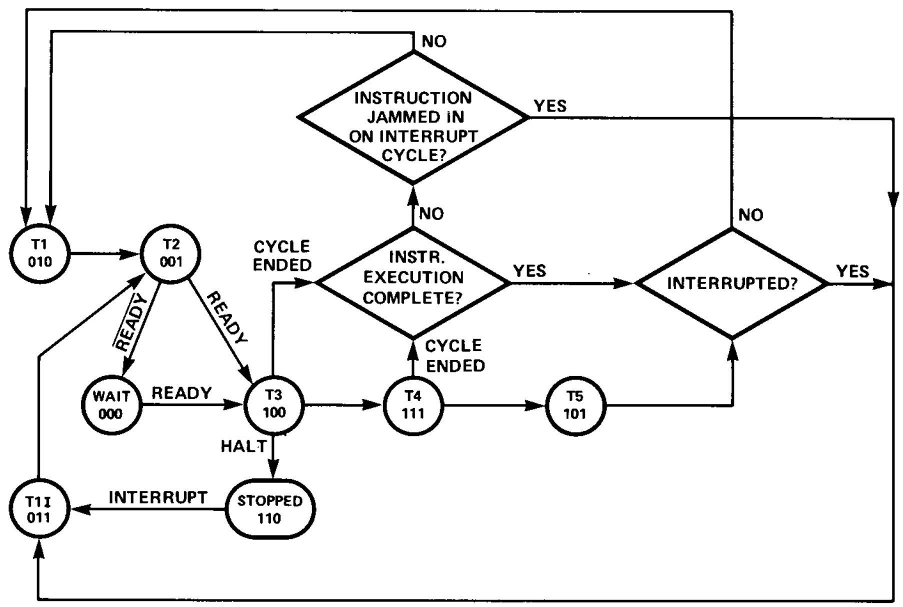


Figure 2. CPU State Transition Diagram

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.

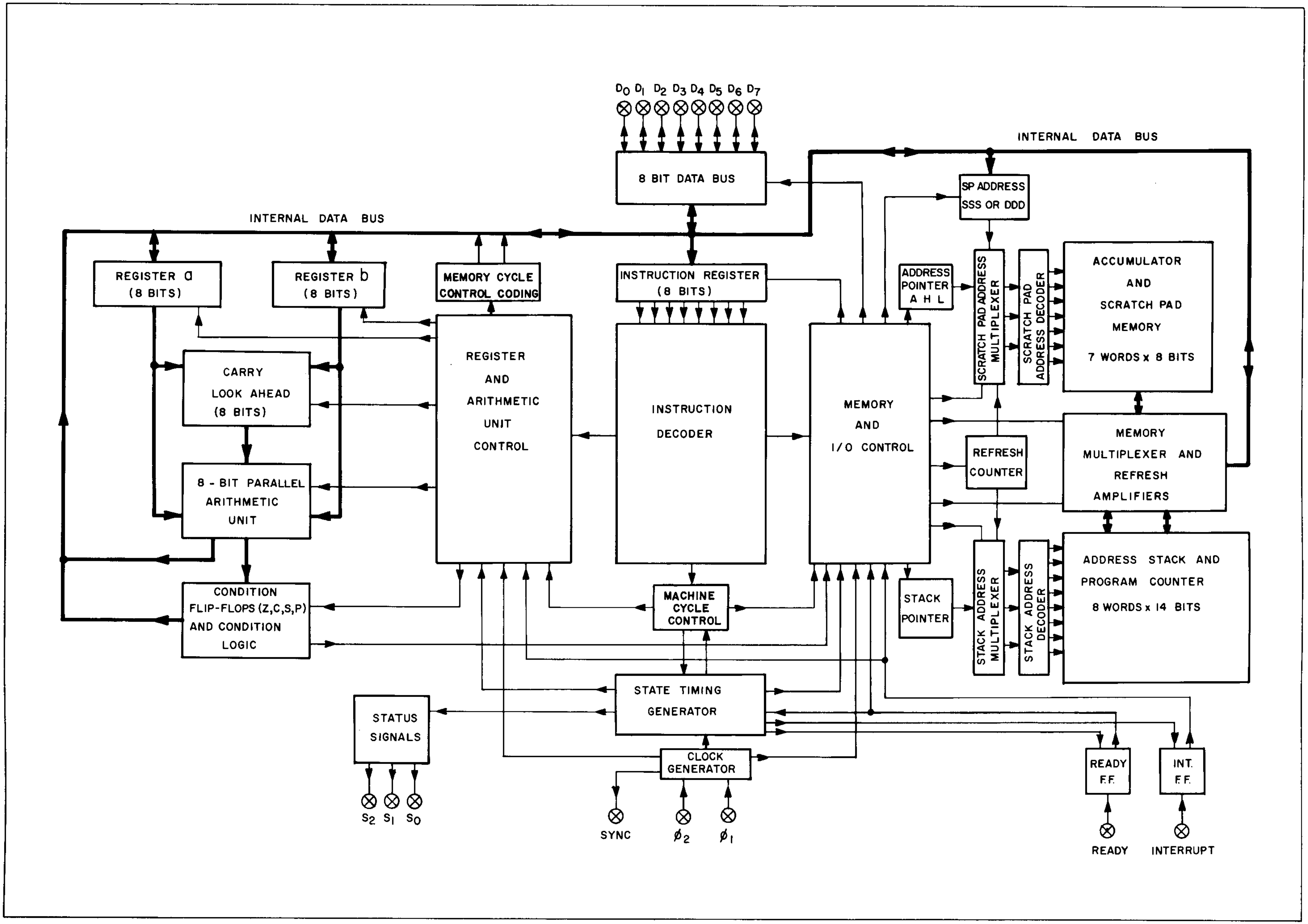


Figure 3. 8008 Block Diagram

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.

IV. BASIC INSTRUCTION SET

The following section presents the basic instruction set of the 8008. For a detailed description of the execution of each instruction, refer to Appendix I.

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) Lr_1r_2	(5)	1 1	D D D	S S S	Load index register r_1 with the content of index register r_2.
(2) LrM	(8)	1 1	D D D	1 1 1	Load index register r with the content of memory register M.
LMr	(7)	1 1	1 1 1	S S S	Load memory register M with the content of index register r.
(3) LrI	(8)	0 0	D D D	1 1 0	Load index register r with data B . . . B.
		B B	B B B	B B B	
LMI	(9)	0 0	1 1 1	1 1 0	Load memory register M with data B . . . B.
		B B	B B B	B B B	
INr	(5)	0 0	D D D	0 0 0	Increment the content of index register r ($r \neq A$).
DCr	(5)	0 0	D D D	0 0 1	Decrement the content of index register r ($r \neq A$).

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.

ADr	(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.
ADM	(8)	1 0	0 0 0	1 1 1	
ADI	(8)	0 0	0 0 0	1 0 0	
		B B	B B B	B B B	
ACr	(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.
ACM	(8)	1 0	0 0 1	1 1 1	
ACI	(8)	0 0	0 0 1	1 0 0	
		B B	B B B	B B B	
SUr	(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.
SUM	(8)	1 0	0 1 0	1 1 1	
SUI	(8)	0 0	0 1 0	1 0 0	
		B B	B B B	B B B	
SBr	(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.
SBM	(8)	1 0	0 1 1	1 1 1	
SBI	(8)	0 0	0 1 1	1 0 0	
		B B	B B B	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
NDr	(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.
NDM	(8)	1 0	1 0 0	1 1 1	
NDI	(8)	0 0 B B	1 0 0 B B B	1 0 0 B B B	
XRr	(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.
XRM	(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	
ORr	(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.
ORM	(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	
CPr	(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.
CPM	(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) JFc	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
JTc	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 0 0 B_2 B_2 B_2 B_3 B_3 B_3	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
CAL	(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).
CFc	(9 or 11)	0 1 B_2 B_2 X X	0 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false, and save the current address (up one level in the stack.) Otherwise, execute the next instruction in sequence.
CTc	(9 or 11)	0 1 B_2 B_2 X X	1 C_4 C_3 B_2 B_2 B_2 B_3 B_3 B_3	0 1 0 B_2 B_2 B_2 B_3 B_3 B_3	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
RET	(5)	0 0	X X X	1 1 1	Unconditionally return (down one level in the stack).
RFc	(3 or 5)	0 0	0 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
RTc	(3 or 5)	0 0	1 C_4 C_3	0 1 1	Return (down one level in the stack) if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
RST	(5)	0 0	A A A	1 0 1	Call the subroutine at memory address AAA000 (up one level in the stack).

Input/Output Instructions

INP	(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)	0 0	0 0 0	0 0 X	Enter the STOPPED state and remain there until interrupted.
HLT	(4)	1 1	1 1 1	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).

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. An example of a synchronizing circuit is shown on the schematic for the SIM8-01 (Section VII). This is a new circuit recently added to the SIM8-01 board.

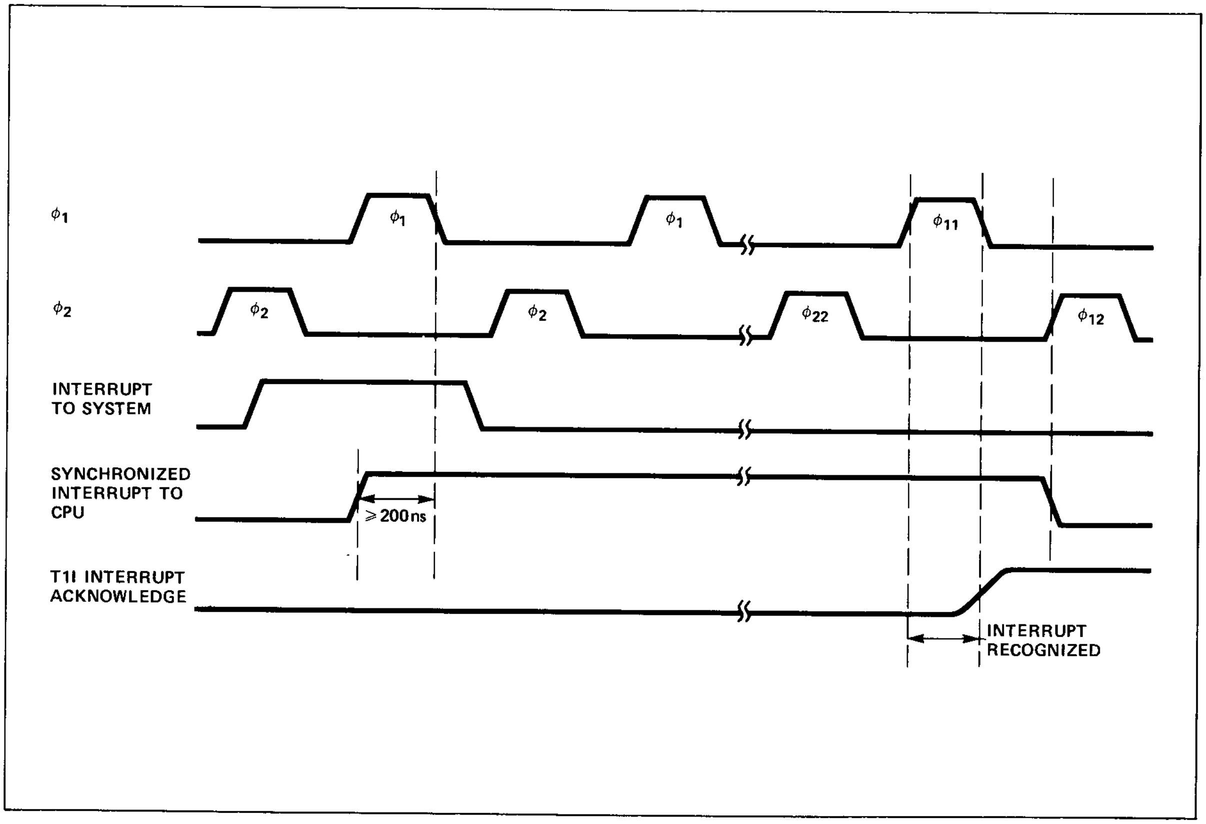


Figure 4. Recognition of Interrupt

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.

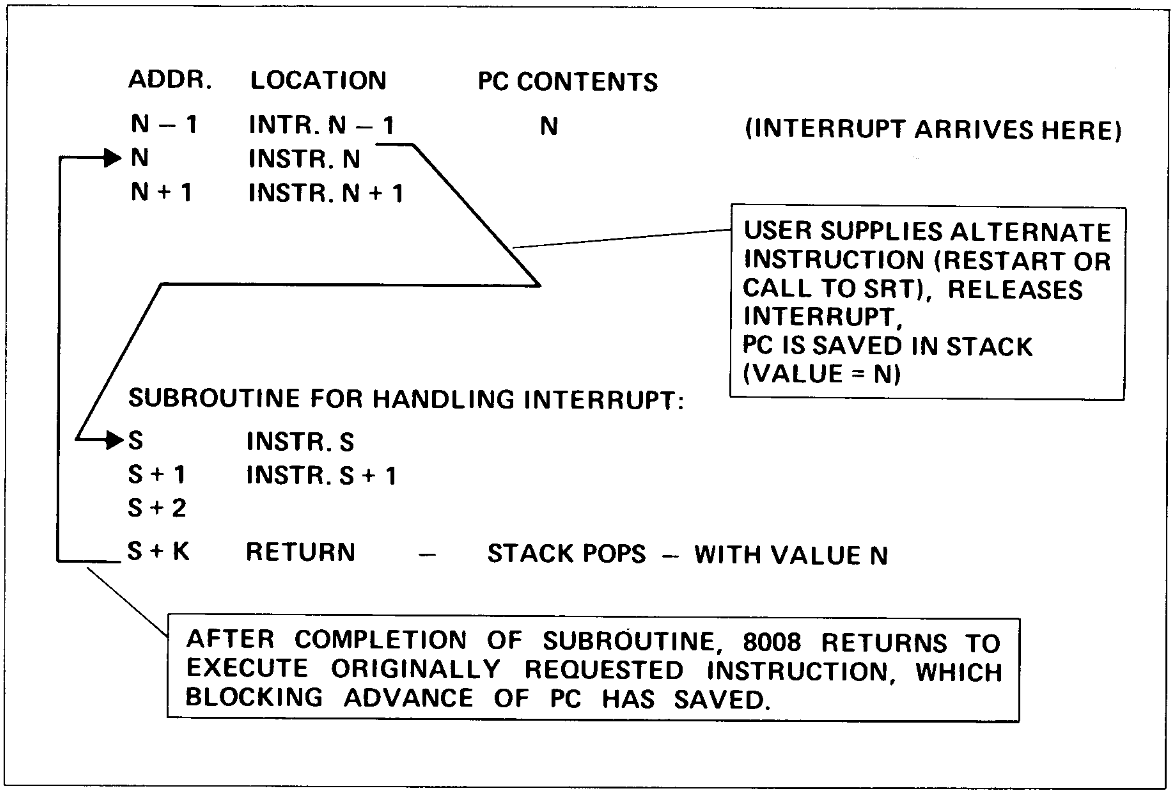


Figure 5. 8008 Interrupt

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 SPECIFICATION

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

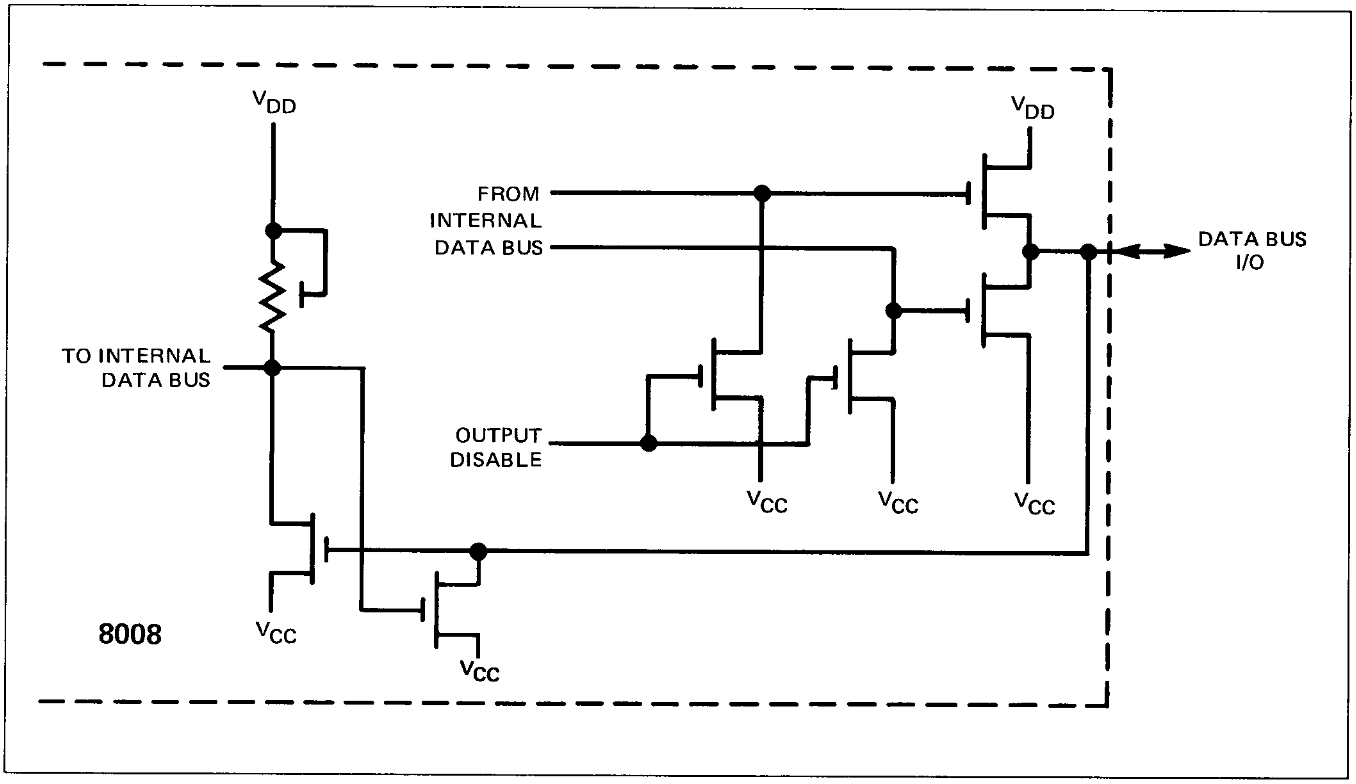
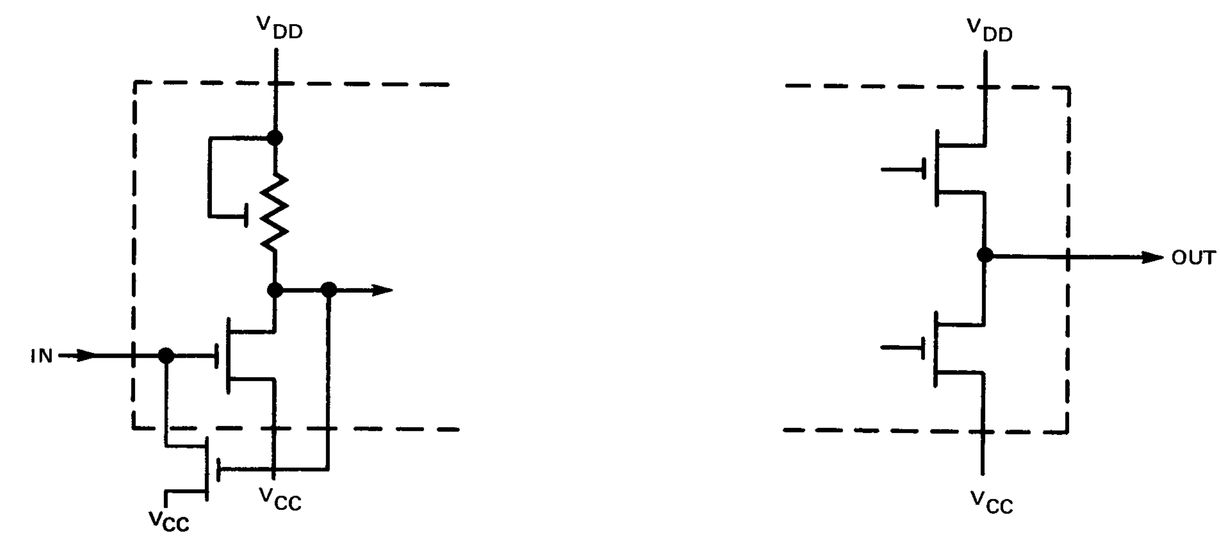


Figure 6. Data Bus I/O Buffer



Input Buffer
(ϕ_1, ϕ_2, RDY, INT)

Output Buffer
(SYNC, S_0, S_1, S_2)

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

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

A.C. CHARACTERISTICS

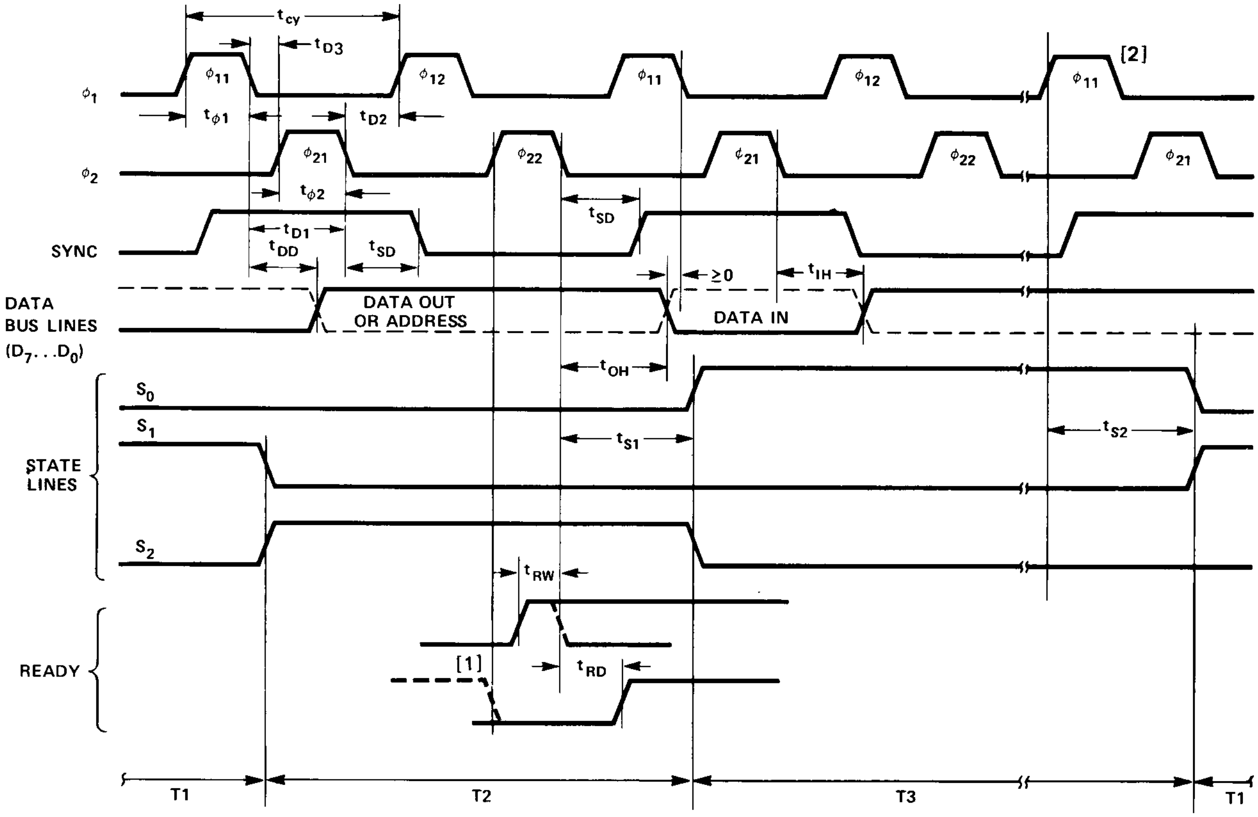
T_A = 0°C to 70°C; V_{CC} = +5V ±5%, V_{DD} = −9V ±5%. All measurements are referenced to 1.5V levels.

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

[1] t_{IH} MIN ≥ t_{SD}

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

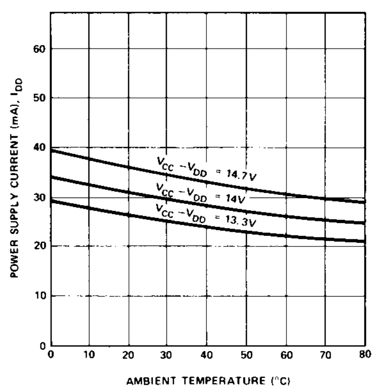
TIMING DIAGRAM

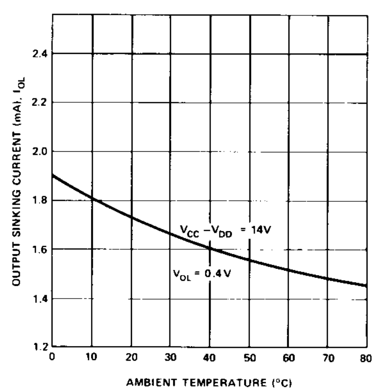


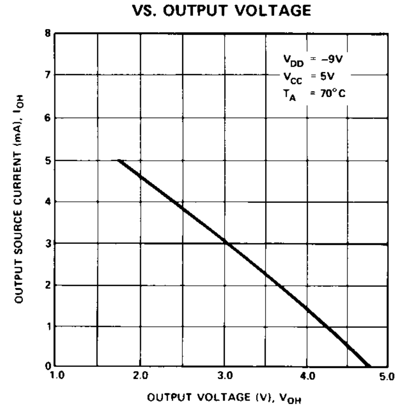
[1] READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.

[2] INTERRUPT line must not change levels within 200ns (max.) of the falling edge of ϕ_1.

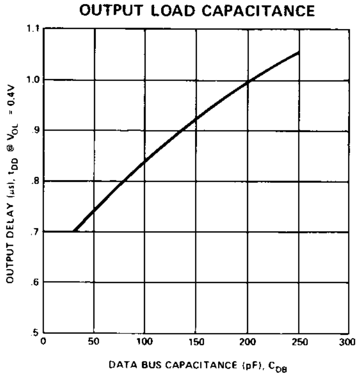
TYPICAL D. C. CHARACTERISTICS







TYPICAL A. C. CHARACTERISTICS



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 THE SIM8-01 — AN MCS-8$^{T.M.}$ MICRO COMPUTER

During the development phase of systems using the 8008, Intel's single chip 8-bit parallel central processor unit, both hardware and software must be designed. Since many systems will require similar memory and I/O interface to the 8008, Intel has developed a prototyping system, the SIM8-01. Through the use of this system and Intel's programmable and erasable ROMs (1702), MCS-8 systems can be completely developed and checked-out before committing to mask programmed ROMs (1301).

The SIM8-01 is a complete byte-oriented computing system including the processor (8008), 1K x 8 memory (1101), six I/O ports (two in and four out), and a two-phase clock generator. Sockets are provided for 2K x 8 of ROM or PROM memory for the system microprogram. The SIM8-01 may be used with either the 8008 or 8008-1. To operate at clock frequencies greater than 500kHz, former SIM8-01 boards must be modified as detailed in the schematic and the following system description. Note that all Intel-developed 8008 programs interface with TTY and require system operation at 500kHz. Currently, the SIM8-01 is supplied with the 8008-1 CPU and the system clock preset to 500kHz.

The following block diagram shows the basic configuration of the SIM8-01. All interface logic for the 8008 to operate with standard ROM and RAM memory is included on the board. The following pages present the SIM8-01 schematic and detailed system description.

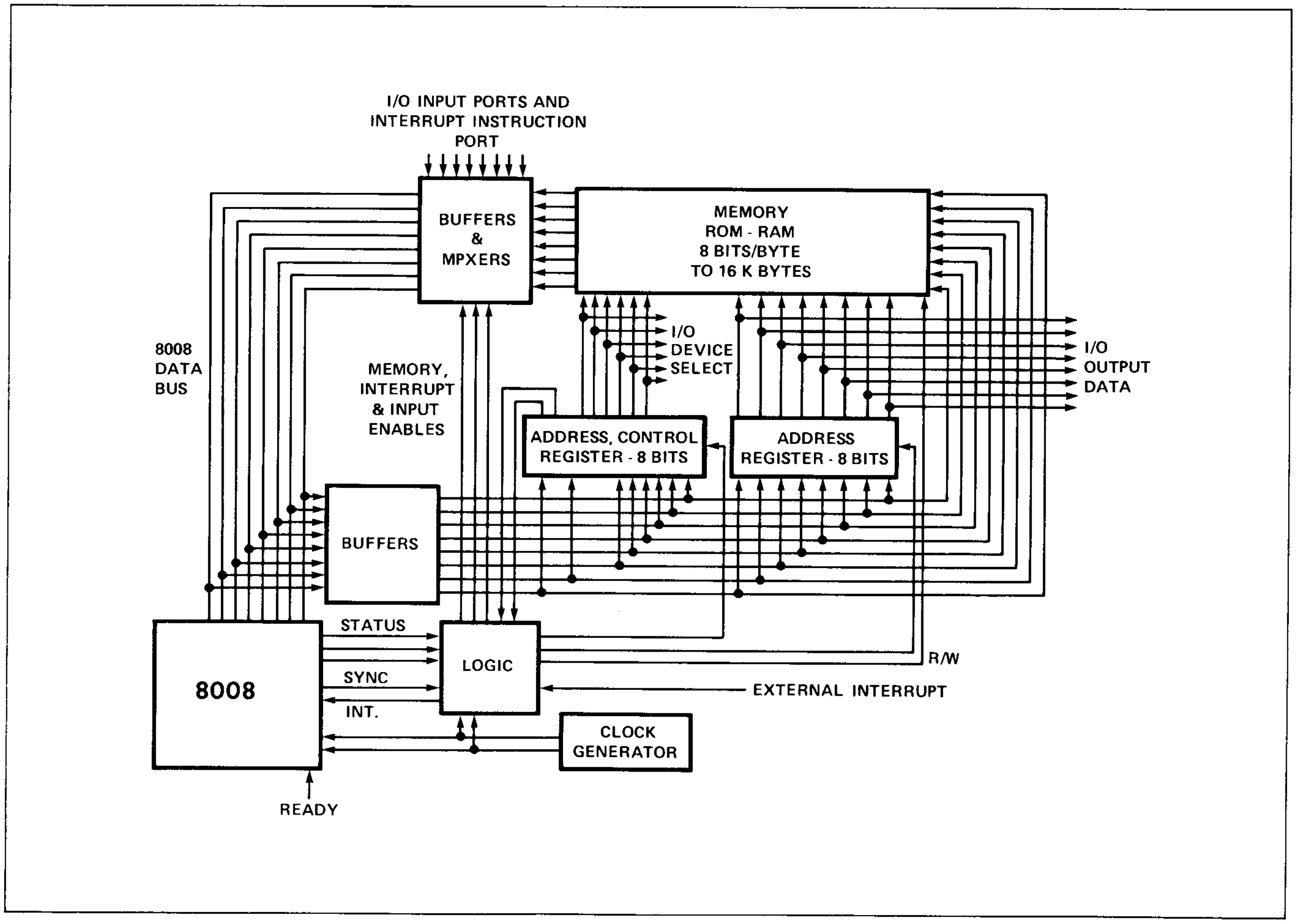


Figure 8. MCS-8 Basic System

SIM8-01 SPECIFICATIONS

Card Dimensions:

- 11.5 inches high
- 9.5 inches deep

System Components Included on Board:

- 8008-1
- Complete TTL interface to memory
- 1K x 8 RAM memory
- Sockets for 2K x 8 PROM memory
- TTY interface ckts.
- Two input and four output ports
- Two phase clock generator

Maximum Memory Configuration:

- 1K x 8 RAM
- 2K x 8 PROM
- All control lines are provided for memory expansion

Operating Speed

- 2 μs clock period
- 20 μs typical instruction cycle

D.C. Power Requirement:

- Voltage:

  V_{CC} = 5V ±5%

  TTL GRD = 0V

  V_{DD} = −9V ±5%

- Current:

  Eight ROMs

		Typical	Maximum
I_{CC}	=	2.5 amps	4.0 amps.
I_{DD}	=	1.0 amps	1.5 amps.

Connector:

- Wire wrap type Amphenol 86 pin connector P/N 261-10043-2

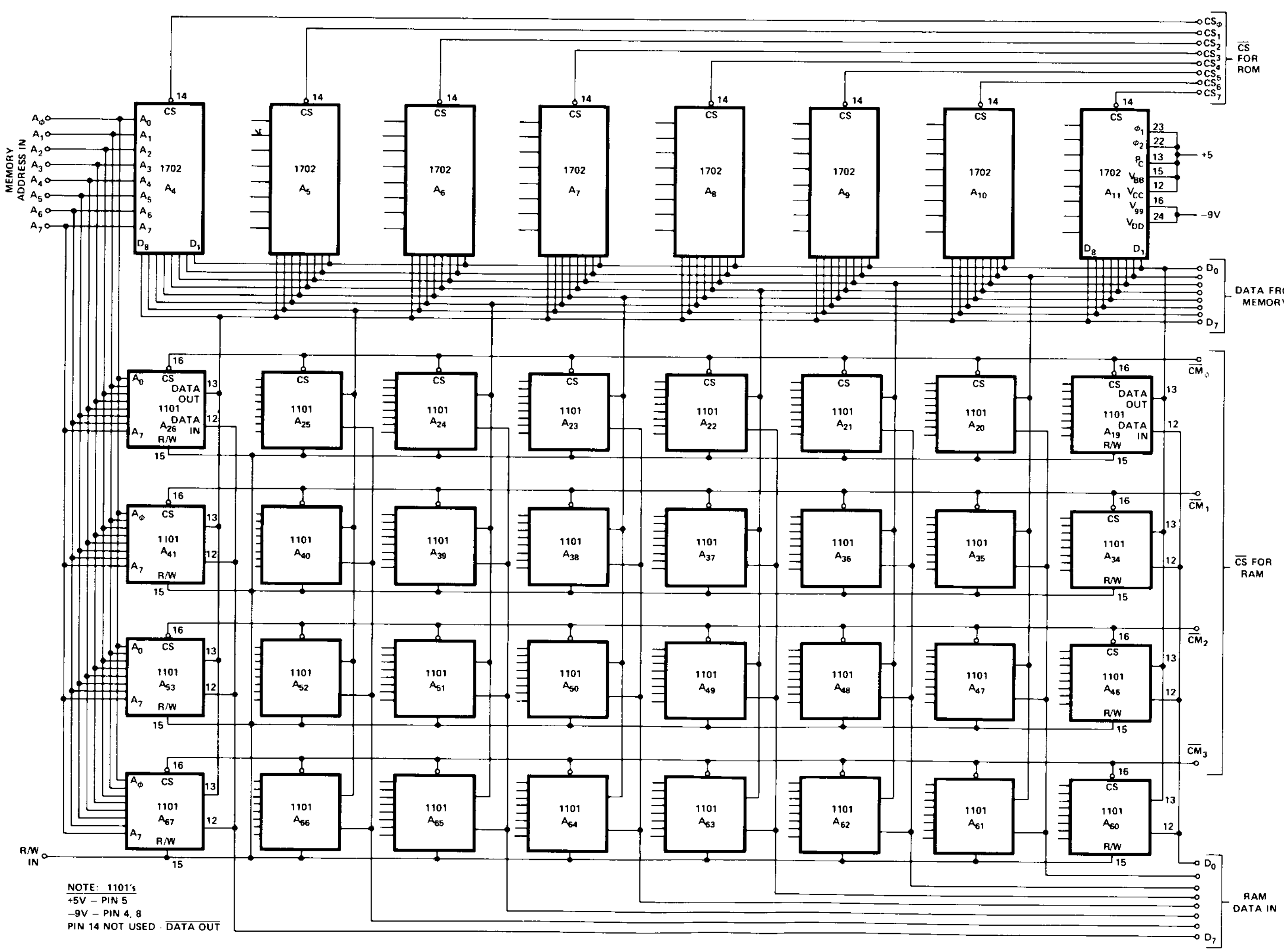


Figure 9. MCS-8 Memory System

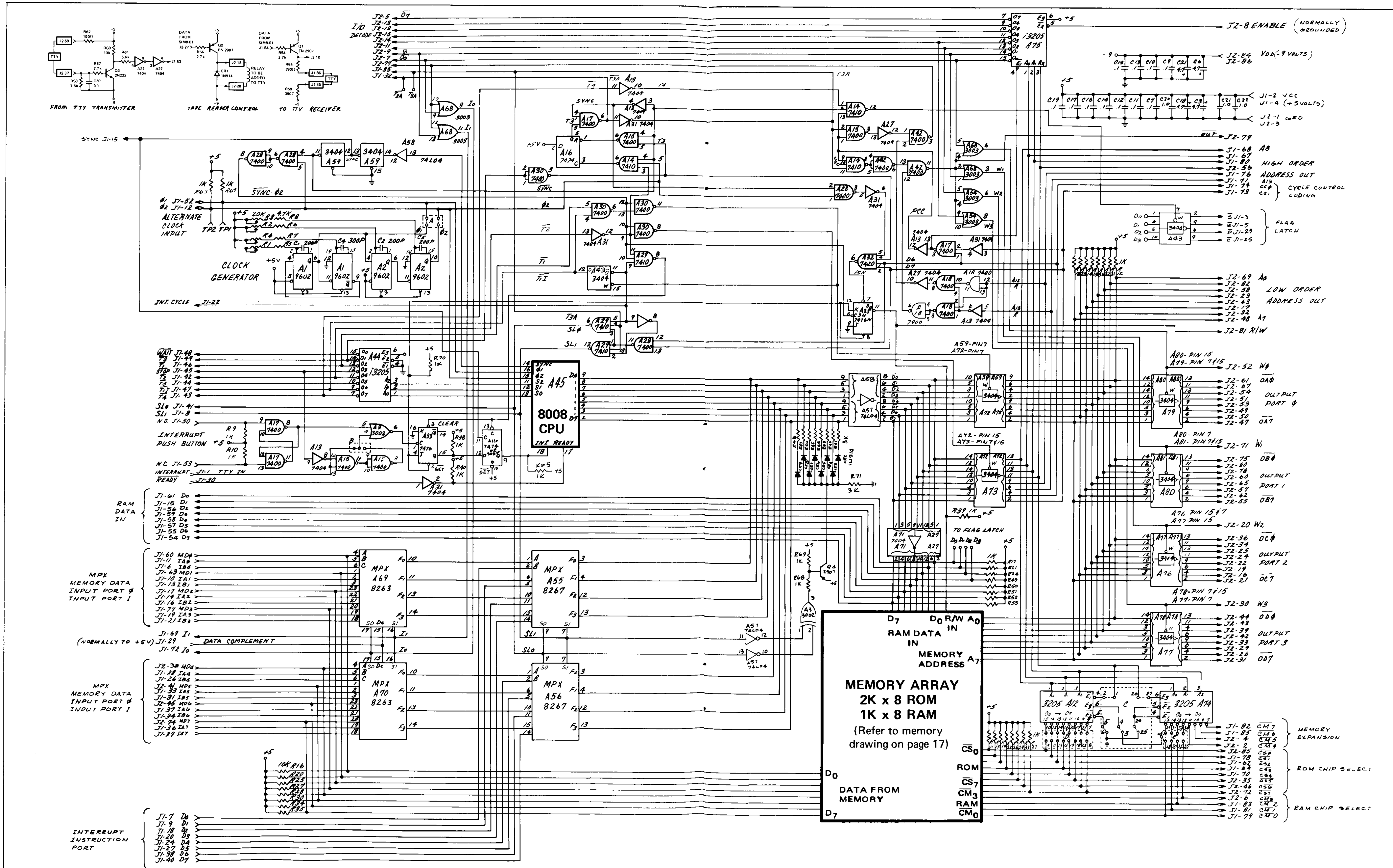


Figure 10. Complete SIM8-01 Schematic

SYSTEM DESCRIPTION

The 8008 processor communicates over an 8-bit data bus (D_0 through D_7) and uses two input lines (READY and INTERRUPT) and four output lines (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, memory, and I/O. All inputs, outputs, and control lines for the SIM8-01 are positive-logic TTL compatible.

Two Phase Clock Generator

The basic system timing for the SIM8-01 is provided by two non-overlapping clock phases generated by 9602 single shot multivibrators (A_1, A_2). The clocks are factory adjusted as shown in the timing diagram below. Note that this is the maximum specified operating frequency of the 8008. An option is provided on the board for using external clocks. If the jumper wires in box A are removed, external clocks may be connected at pins J1-52 and J1-12 (Normally these pins are the output of the clock generators on the board). The clock generator may be adjusted for operation up to 800kHz when using the 8008-1.

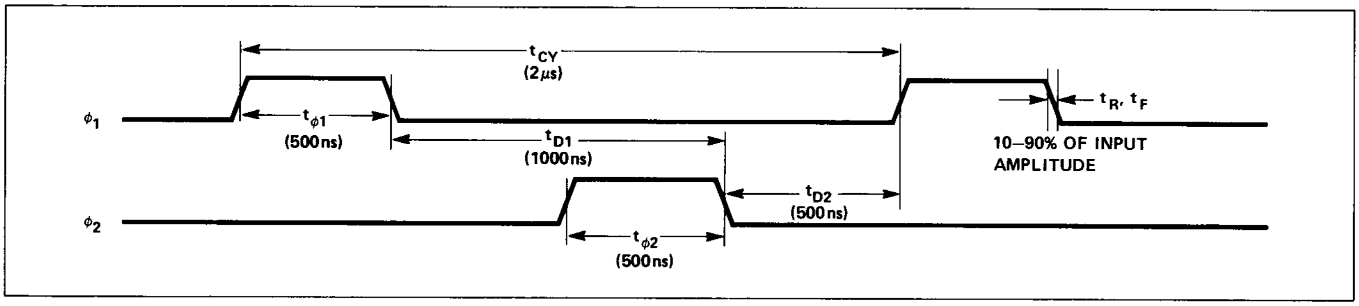


Figure 11. SIM8-01 Timing Diagram

Memory Organization

The SIM8-01 has capacity for 2K x 8 of ROM or PROM and 1K x 8 of RAM. The memory can easily be expanded to 16K x 8 using the address and chip select control lines provided. Further memory expansion may be accomplished by dedicating an output port to the control of memory bank switching.

In an MCS-8 system, it is possible to use any combination of memory elements. The SIM8-01 is shipped from the factory with the ROM memory designated from address 0 → 2047, RAM memory from 2048 → 3071, and memory expansion for all addresses 3072 and above. Jumper wires provided on the board (boxes C, D, E) allow complete flexibility of the memory organization. They may be rearranged to meet any requirement. the Intel 3205 data sheet provides a complete description of the one of eight decoder used in this system. the 3205 truth table is shown below.

ADDRESS			ENABLE			OUTPUTS							
A_0	A_1	A_2	E_1	E_2	E_3	0	1	2	3	4	5	6	7
L	L	L	L	L	H	L	H	H	H	H	H	H	H
H	L	L	L	L	H	H	L	H	H	H	H	H	H
L	H	L	L	L	H	H	H	L	H	H	H	H	H
H	H	L	L	L	H	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
H	L	H	L	L	H	H	H	H	H	H	L	H	H
L	H	H	L	L	H	H	H	H	H	H	H	L	H
H	H	H	L	L	H	H	H	H	H	H	H	H	L
X	X	X	L	L	L	H	H	H	H	H	H	H	H
X	X	X	H	L	L	H	H	H	H	H	H	H	H
X	X	X	L	H	L	H	H	H	H	H	H	H	H
X	X	X	H	H	L	H	H	H	H	H	H	H	H
X	X	X	H	L	H	H	H	H	H	H	H	H	H
X	X	X	L	H	H	H	H	H	H	H	H	H	H
X	X	X	H	H	H	H	H	H	H	H	H	H	H

CONTROL LINES

- **Interrupt**

The interrupt control line is directly available as an input to the board. For manual control, a normally open push-button switch may be connected to terminals J1-50 and J1-53. The interrupt may be inserted under system control on pin J1-1. An external flip-flop (A33) latches the interrupt and is reset by T11 when the CPU recognizes the interrupt. Instructions inserted under interrupt control may be set up automatically or by toggle switches at the interrupt input port as shown on the schematic. Use the interrupt line and interrupt input port to start up the 8008.

Note that the interrupt line has two different connections to the input to the board (box B). The path from J1-1 directly to pin 4 of package A3 is the normal interrupt path (**the board is shipped from the factory with this connection**). If the connection from pin 8 of package A15 to pin 4 of package A3 is made instead, the processor will recognize an interrupt only when it is in the STOPPED state. This is used to recognize the "start character" when entering data from TTY.

- **Ready**

The ready line on the 8008 provides the flexibility for operation with any type of semiconductor memory. On the SIM8-01 board, the ready line is buffered; and at the connector (J1-30), the READY line is active low. During program development, the READY line may be used to step the system through a program.

NORMAL OPERATION OF SYSTEM

The 8008 CPU exercises control over the entire system using its state lines (S_0, S_1, S_2) and two control bits (CC0, CC1) which are sent onto the data bus with the address. The state lines are decoded by a 3205 (A44) and gated with appropriate clock and SYNC signals. The two control bits form part of the control for the multiplexers to the data bus (A55, A56), the memory read/write line (A33) and the I/O line (A17).

In normal operation, the lower order address is sent out of the CPU at state T1, stored in 3404 latches (A59, A72) and provided to all memories. The high order address is sent out at a state T2 and stored in 3404 latches (A72, A73). These lines are decoded as the chip selects to the memory. The two highest order bits (CC0, CC1) are decoded for control.

To guarantee that instructions and data are available to the CPU at the proper time, the T3 state is anticipated by setting a D-type flip-flop (A16) at the end of each T2 state. This line controls the multiplexing of data to the 8008. This flip-flop is reset at the end of each T3 state. In addition, switched pull-up resistors are used on the data-bus to minimize data bus loading and increase bus response. **The use of switched resistors on the data bus is mandatory when using the 8008-1. SIM8-01 boards built prior to October, 1972 must be modified in order to operate with the 8008-1 at clock frequencies greater than 500kHz.**

Normally, the 8008 executes instructions and has no interaction with the rest of the system during states T4 and T5. In the case of the INP instruction, the content of the flag flip-flops internal to the 8008 is sent out at state T4 and stored in a 3404 latch (A43).

Instructions and data are multiplexed onto the 8008 data bus through four multiplexers (A55, A56, A69, A70). **In normal operation, line J1-29 should be at +5V in order for "true" data to reach the 8008 data bus.**

System I/O

The SIM8-01 communicates with other systems or peripherals through two input ports and four output ports. All control and I/O selection decoding lines are provided for expansion to the full complement of eight input ports and twenty-four output ports. To expand the number of input ports, break the trace at the output of Device A68, pin 11, and generate input port decoding external to the SIM8-01. Control the input multiplexer through pin J1-69. The output ports latch data and remain unchanged until referenced again under software control. **Note that all output ports complement data.** When power is first applied to the board, the output ports should be cleared under software control to guarantee a known output state. **To enable the I/O device decoder, pin J2-8 should be at ground.**

Teletype Interface

The 8008 is designed to operate with all types of terminal devices. A typical example of peripheral interface is the teletype (ASR-33). The SIM8-01 contains the three simple transistor TTY interface circuits shown on the following page. One transistor is used for receiving serial data from the teletype, one for transmitting data back to the teletype, and the third for tape reader control.

The teletype must be operating in the full duplex mode. Refer to your teletype operating manual for making connections within the TTY itself. Many models include a nine terminal barrier strip in the rear of the machine. It is at this point where the connections are made for full duplex operation. The interconnections to the SIM8-01 for transmit and receive are made at this same point. (See Figure 12)

A micro computer bulletin describing the interconnection of Intel micro computers and the ASR-33 teletype is available on request.

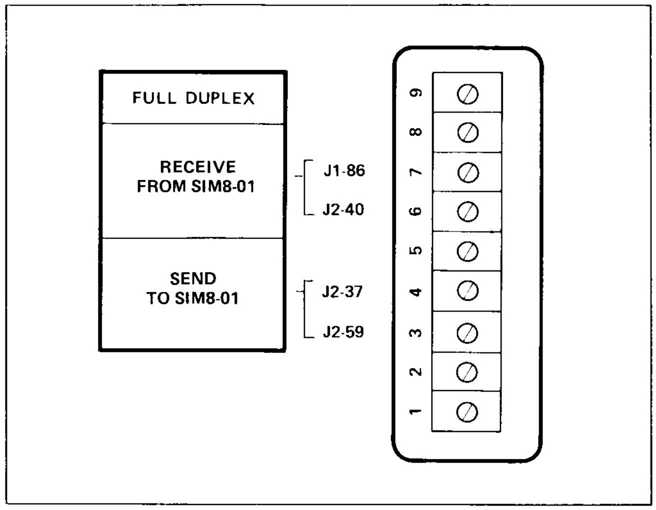


Figure 12. Teletype Terminal Strip

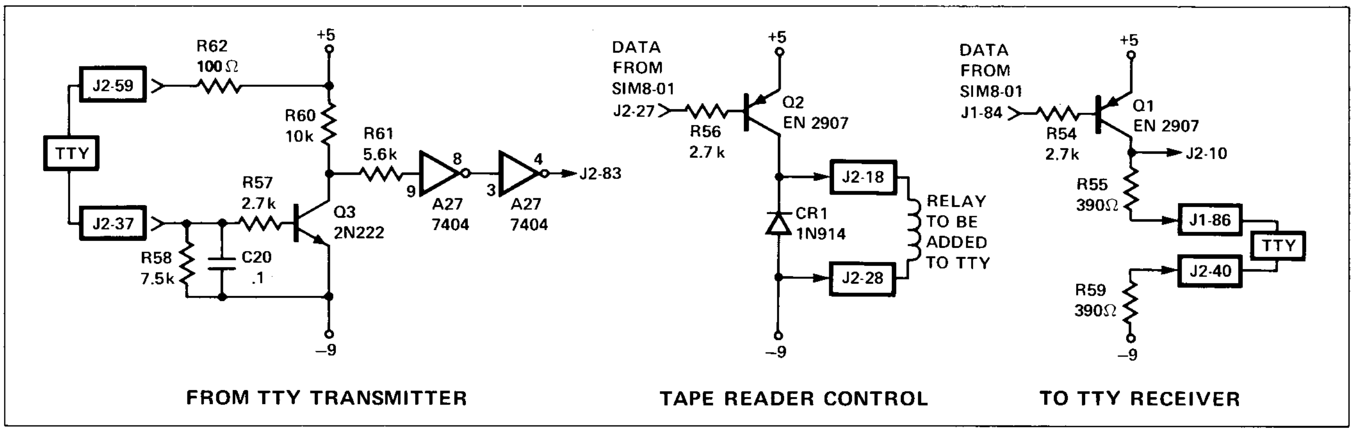


Figure 13. SIM8-01 Teletype Interface Circuitry

To use the teletype tape reader with the SIM8-01, the machine must contain a reader power pack. The contacts of a 10V dc relay must be connected in series with the TTY automatic reader (refer to TTY manual) and the coil is connected to the SIM8-01 tape reader control as shown.

For all Intel developed TTY programs for the SIM8-01, the following I/O port assignments have been made:

1. DATA IN -- INPUT PORT 0, BIT 0 (J2-83 connected to J1-11)
2. DATA OUT -- OUTPUT PORT 2, BIT 0 (J1-84 connected to J2-36)
3. READER CONTROL -- OUTPUT PORT 3, BIT 0 (J2-27 connected to J2-44)

Note that the SIM8-01 clock generator must remain set at 500kHz. All Intel developed TTY programs are synchronized to operate with the SIM8-01 at 500kHz.

In order to sense the start character, data in is also sensed at the interrupt input (J2-83 connected to J1-1) and the interrupt jumper (box B) must be between pin 8 of A15 and pin 4 of A3. It requires approximately 110ms for the teletype to transmit or receive eight serial data bits plus three control bits. The first and last bits are idling bits, the second is the start bit, and the following eight bits are data. Each bit stays 9.09ms. While waiting for data to be transmitted, the 8008 is in the STOPPED state; when the start character is received, the processor is interrupted and forced to call the TTY processing routine. Under software control, the processor can determine the duration of each bit and strobe the character at the proper time.

A listing of a teletype control program is shown in Appendix III.

SIM8-01 MICRO COMPUTER BOARD PIN DESCRIPTION

Pin No.	Connector	Symbol	Description
2,4	J1	+5V	+5VDC POWER SUPPLY
84 & 86	J2	-9V	-9VDC POWER SUPPLY
1,3	J2	GND	GROUND
60	J1	MD_0	DATA FROM MEMORY Ø BIT Ø
63	J1	MD_1	DATA FROM MEMORY 1 BIT 1
17	J1	MD_2	DATA FROM MEMORY 2 BIT 2
77	J1	MD_3	DATA FROM MEMORY 3 BIT 3
38	J2	MD_4	DATA FROM MEMORY 4 BIT 4
41	J2	MD_5	DATA FROM MEMORY 5 BIT 5
45	J2	MD_6	DATA FROM MEMORY 6 BIT 6
74	J2	MD_7	DATA FROM MEMORY 7 BIT 7
11	J1	IA_0	DATA INPUT PORT Ø BIT Ø
10	J1	IA_1	DATA INPUT PORT Ø BIT 1
14	J1	IA_2	DATA INPUT PORT Ø BIT 2
19	J1	IA_3	DATA INPUT PORT Ø BIT 3
28	J1	IA_4	DATA INPUT PORT Ø BIT 4
33	J1	IA_5	DATA INPUT PORT Ø BIT 5
37	J1	IA_6	DATA INPUT PORT Ø BIT 6
36	J1	IA_7	DATA INPUT PORT Ø BIT 7
6	J1	IB_0	DATA INPUT PORT 1 BIT Ø
13	J1	IB_1	DATA INPUT PORT 1 BIT 1
16	J1	IB_2	DATA INPUT PORT 1 BIT 2
21	J1	IB_3	DATA INPUT PORT 1 BIT 3
26	J1	IB_4	DATA INPUT PORT 1 BIT 4
31	J1	IB_5	DATA INPUT PORT 1 BIT 5
34	J1	IB_6	DATA INPUT PORT 1 BIT 6
39	J1	IB_7	DATA INPUT PORT 1 BIT 7
61	J2	$\overline{OA}_{\emptyset}$	OUTPUT PORT Ø BIT Ø
67	J2	$\overline{OA}_1$	OUTPUT PORT Ø BIT 1
54	J2	$\overline{OA}_2$	OUTPUT PORT Ø BIT 2
51	J2	$\overline{OA}_3$	OUTPUT PORT Ø BIT 3
53	J2	$\overline{OA}_4$	OUTPUT PORT Ø BIT 4
49	J2	$\overline{OA}_5$	OUTPUT PORT Ø BIT 5
50	J2	$\overline{OA}_6$	OUTPUT PORT Ø BIT 6
47	J2	$\overline{OA}_7$	OUTPUT PORT Ø BIT 7
75	J2	$\overline{OB}_{\emptyset}$	OUTPUT PORT 1 BIT Ø
80	J2	$\overline{OB}_1$	OUTPUT PORT 1 BIT 1
78	J2	$\overline{OB}_2$	OUTPUT PORT 1 BIT 2
60	J2	$\overline{OB}_3$	OUTPUT PORT 1 BIT 3
65	J2	$\overline{OB}_4$	OUTPUT PORT 1 BIT 4
57	J2	$\overline{OB}_5$	OUTPUT PORT 1 BIT 5
62	J2	$\overline{OB}_6$	OUTPUT PORT 1 BIT 6
55	J2	$\overline{OB}_7$	OUTPUT PORT 1 BIT 7
36	J2	$\overline{OC}_{\emptyset}$	OUTPUT PORT 2 BIT Ø
34	J2	$\overline{OC}_1$	OUTPUT PORT 2 BIT 1
25	J2	$\overline{OC}_2$	OUTPUT PORT 2 BIT 2
24	J2	$\overline{OC}_3$	OUTPUT PORT 2 BIT 3
22	J2	$\overline{OC}_4$	OUTPUT PORT 2 BIT 4
19	J2	$\overline{OC}_5$	OUTPUT PORT 2 BIT 5
16	J2	$\overline{OC}_6$	OUTPUT PORT 2 BIT 6
21	J2	$\overline{OC}_7$	OUTPUT PORT 2 BIT 7
44	J2	$\overline{OD}_{\emptyset}$	OUTPUT PORT 3 BIT Ø
43	J2	$\overline{OD}_1$	OUTPUT PORT 3 BIT 1
39	J2	$\overline{OD}_2$	OUTPUT PORT 3 BIT 2
42	J2	$\overline{OD}_3$	OUTPUT PORT 3 BIT 3
33	J2	$\overline{OD}_4$	OUTPUT PORT 3 BIT 4
29	J2	$\overline{OD}_5$	OUTPUT PORT 3 BIT 5
26	J2	$\overline{OD}_6$	OUTPUT PORT 3 BIT 6
31	J2	$\overline{OD}_7$	OUTPUT PORT 3 BIT 7
69	J2	$A_{\emptyset}$	LOW ORDER ADDRESS OUT
82	J2	A_1	LOW ORDER ADDRESS OUT
58	J2	A_2	LOW ORDER ADDRESS OUT
23	J2	A_3	LOW ORDER ADDRESS OUT
63	J2	A_4	LOW ORDER ADDRESS OUT
17	J2	A_5	LOW ORDER ADDRESS OUT
32	J2	A_6	LOW ORDER ADDRESS OUT
48	J2	A_7	LOW ORDER ADDRESS OUT
68	J1	A_8	HIGH ORDER ADDRESS OUT
67	J1	A_9	HIGH ORDER ADDRESS OUT
80	J1	A_{10}	HIGH ORDER ADDRESS OUT
56	J2	A_{11}	HIGH ORDER ADDRESS OUT
76	J1	A_{12}	HIGH ORDER ADDRESS OUT
71	J1	A_{13}	HIGH ORDER ADDRESS OUT
74	J1	$CC_{\emptyset}$	CYCLE CONTROL CODING
73	J1	CC_1	CYCLE CONTROL CODING
61	J1	$D_{\emptyset}$	RAM DATA IN $D_{\emptyset}$
15	J1	D_1	RAM DATA IN D_1
56	J1	D_2	RAM DATA IN D_2
59	J1	D_3	RAM DATA IN D_3
58	J1	D_4	RAM DATA IN D_4

Pin No.	Connector	Symbol	Description
57	J1	D_5	RAM DATA IN D_5
55	J1	D_6	RAM DATA IN D_6
54	J1	D_7	RAM DATA IN D_7
48	J1	$\overline{WAIT}$	STATE COUNTER
49	J1	$\overline{T}_3$	STATE COUNTER
46	J1	$\overline{T}_1$	STATE COUNTER
45	J1	$\overline{STOP}$	STATE COUNTER
42	J1	$\overline{T}_2$	STATE COUNTER
44	J1	$\overline{T}_5$	STATE COUNTER
47	J1	$\overline{T_1I}$	STATE COUNTER
43	J1	$\overline{T}_4$	STATE COUNTER
79	J1	$\overline{CM}_{\emptyset}$	RAM CHIP SELECT Ø
81	J1	$\overline{CM}_1$	RAM CHIP SELECT 1
63	J1	$\overline{CM}_2$	RAM CHIP SELECT 2
6	J2	$\overline{CM}_3$	RAM CHIP SELECT 3
2	J2	$\overline{CM}_4$	RAM CHIP SELECT 4
4	J2	$\overline{CM}_5$	RAM CHIP SELECT 5
85	J1	$\overline{CM}_6$	RAM CHIP SELECT 6
82	J1	$\overline{CM}_7$	RAM CHIP SELECT 7
85	J2	$\overline{CS}_{\emptyset}$	ROM CHIP SELECT Ø
78	J1	$\overline{CS}_1$	ROM CHIP SELECT 1
62	J1	$\overline{CS}_2$	ROM CHIP SELECT 2
64	J1	$\overline{CS}_3$	ROM CHIP SELECT 3
70	J1	$\overline{CS}_4$	ROM CHIP SELECT 4
35	J2	$\overline{CS}_5$	ROM CHIP SELECT 5
46	J2	$\overline{CS}_6$	ROM CHIP SELECT 6
72	J2	$\overline{CS}_7$	ROM CHIP SELECT 7
5	J2	$\overline{O}_7$	I/O DECODE OUT O_7
13	J2	$\overline{O}_6$	I/O DECODE OUT O_6
12	J2	$\overline{O}_5$	I/O DECODE OUT O_5
15	J2	$\overline{O}_4$	I/O DECODE OUT O_4
14	J2	$\overline{O}_3$	I/O DECODE OUT O_3
11	J2	$\overline{O}_2$	I/O DECODE OUT O_2
9	J2	$\overline{O}_1$	I/O DECODE OUT O_1
7	J2	$\overline{O}_{\emptyset}$	I/O DECODE OUT $O_{\emptyset}$
3	J1	$\overline{S}$	FLAG FLIP FLOP-Sign
5	J1	$\overline{Z}$	FLAG FLIP FLOP-Zero
23	J1	$\overline{P}$	FLAG FLIP FLOP-Parity
25	J1	$\overline{C}$	FLAG FLIP FLOP-Carry
7	J1	D_0	INTERRUPT INSTRUCTION INPUT Ø
9	J1	D_1	INTERRUPT INSTRUCTION INPUT 1
18	J1	D_2	INTERRUPT INSTRUCTION INPUT 2
20	J1	D_3	INTERRUPT INSTRUCTION INPUT 3
24	J1	D_4	INTERRUPT INSTRUCTION INPUT 4
27	J1	D_5	INTERRUPT INSTRUCTION INPUT 5
38	J1	D_6	INTERRUPT INSTRUCTION INPUT 6
40	J1	D_7	INTERRUPT INSTRUCTION INPUT 7
59	J2		FROM TTY TRANSMITTER IN } TTY BUFFER
37	J2		FROM TTY TRANSMITTER OUT } TTY BUFFER
83	J2		DATA FROM TTY TRANSMITTER BUFFER
27	J2		TAPE READER CONTROL IN
18	J2		TAPE READER CONTROL OUT
28	J2		TAPE READER CONTROL (-9VDC)
84	J1		DATA TO TTY RECEIVER BUFFER
10	J2		TO TTY RECEIVER OUT } TTY BUFFER
86	J1		TO TTY RECEIVER OUT } TTY BUFFER
40	J2		TO TTY RECEIVER OUT } TTY BUFFER
81	J2		READ/WRITE
72	J1	IØ	MULTIPLEXER CONTROL LINES N8263
41	J1	SLØ	MULTIPLEXER CONTROL LINES N8267
69	J1	I1	MULTIPLEXER CONTROL LINES N8263
8	J1	SL1	MULTIPLEXER CONTROL LINES N8267
29	J1		DATA COMPLEMENT
52	J1	$\emptyset_1$	$\emptyset_1$ CLOCK (alternate clock)
12	J1	$\emptyset_2$	$\emptyset_2$ CLOCK (alternate clock)
75	J1	SYNC	SYNC OUT
30	J1	$\overline{READY}$	READY IN
1	J1	INTERRUPT	INTERRUPT IN
8	J2	I/O ENABLE	ENABLE OF I/O DEVICE DECODER
79	J2	$\overline{OUT}$	SYSTEM OUTPUT CONTROL
77	J2	$\overline{IN}$	SYSTEM INPUT CONTROL
50	J1	N.O.	PUSH BUTTON SWITCH } INTERRUPT
53	J1	N.C.	PUSH BUTTON SWITCH } INTERRUPT
52	J2	$W_{\emptyset}$	OUTPUT LATCH STROBE PORT Ø
71	J2	W_1	OUTPUT LATCH STROBE PORT 1
20	J2	W_2	OUTPUT LATCH STROBE PORT 2
30	J2	W_3	OUTPUT LATCH STROBE PORT 3
22	J1	INT CYCLE	INTERRUPT CYCLE INDICATOR
32	J1	$\overline{T3}_A$	ANTICIPATED $\overline{T}_3$ OUTPUT
35	J1	$T3_A$	ANTICIPATED T_3 OUTPUT

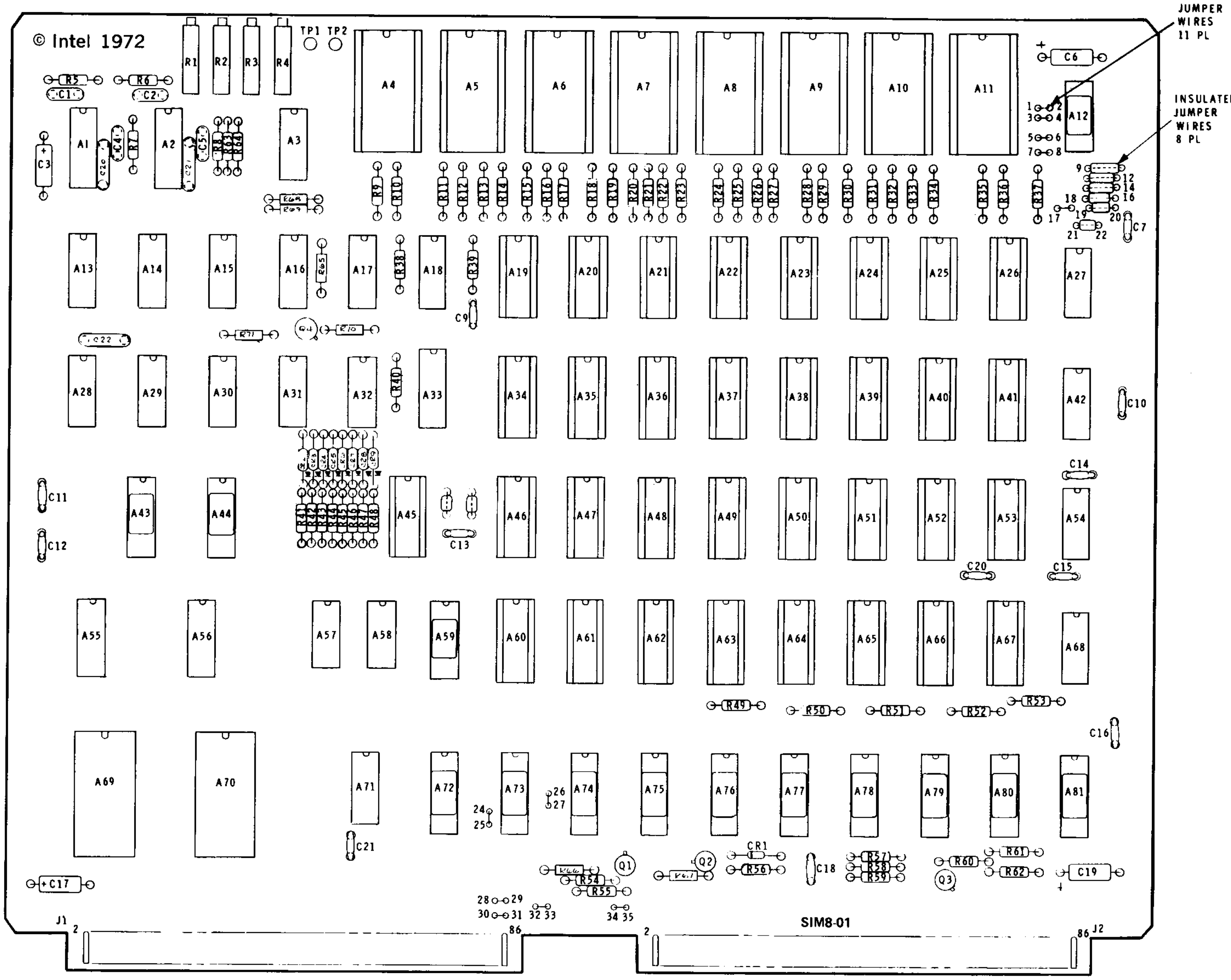


Figure 14. SIM8-01 Assembly Diagram

VIII. PROM PROGRAMMING SYSTEM

A. General System Description and Operating Instructions

Intel has developed a low-cost micro computer programming system for its electrically programmable ROMs. Using Intel's eight bit micro computer system and a standard ASR 33 teletype (TTY), a complete low cost and easy to use ROM programming system may be assembled. The system features the following functions:

1) Memory loading
2) Format checking
3) ROM programming
4) Error checking
5) Program listing

For specifications of the Intel ROMs, refer to the Intel Data Catalog.

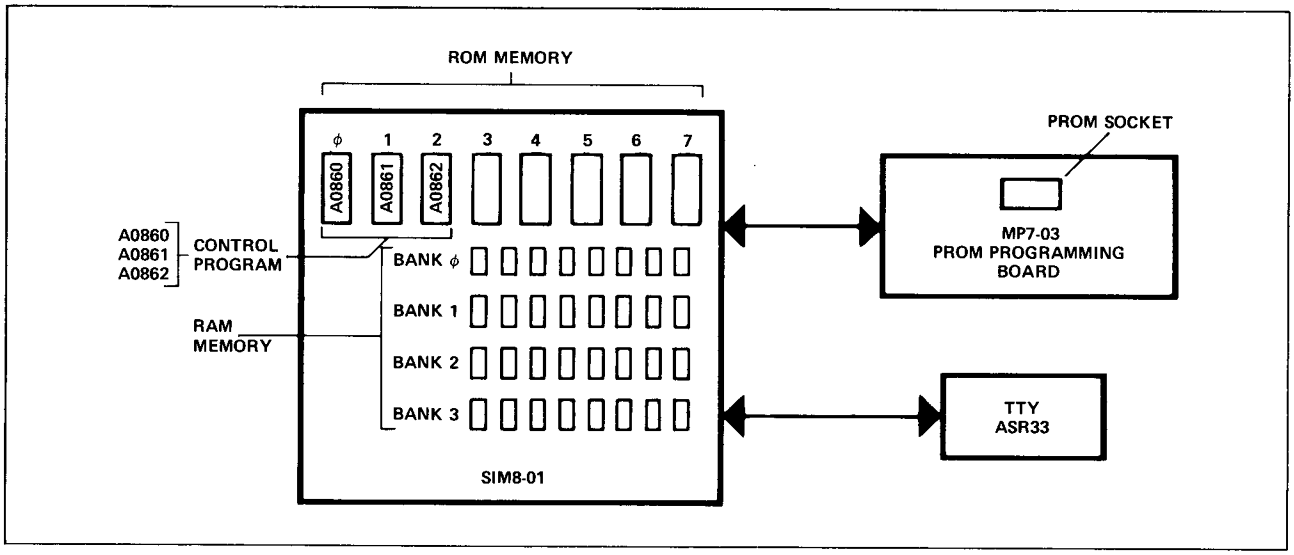


Figure 15. MCS-8 PROM Programming System

This programming system has four basic parts:

1) The micro computer (SIM8-01)
   This is the **MCS-8** prototype board, a complete micro-computer which uses 1702 PROMs for the microprogram control. The total system is controlled by the 8008 CPU.

2) The control program (A0860, A0861, A0862)
   These control ROMs contain the microprograms which control the bootstrap loading, programming, format and error checking, and listing functions. For high speed programming of Intel's new 1702A PROM (three minutes) use control PROM A0863 in place of A0862.

3) The programmer (MP7-03)
   This is the programmer board which contains all of the timing and level shifting required to program the Intel ROMs. This is the successor of the MP7-02.

4) ASR 33 (Automatic Send Receive) Teletype
   This provides both the keyboard and paper tape I/O devices for the programming system.

In addition, a short-wave ultraviolet light is required if the erasable and reprogrammable 1701's or 1702's are used.

This system has two modes of operation:

1) Automatic – A paper tape is used in conjunction with the tape reader on the teletype. The tape contains the program for the ROM.

2) Manual – The keyboard of the TTY is used to enter the data content of the word to be programmed.

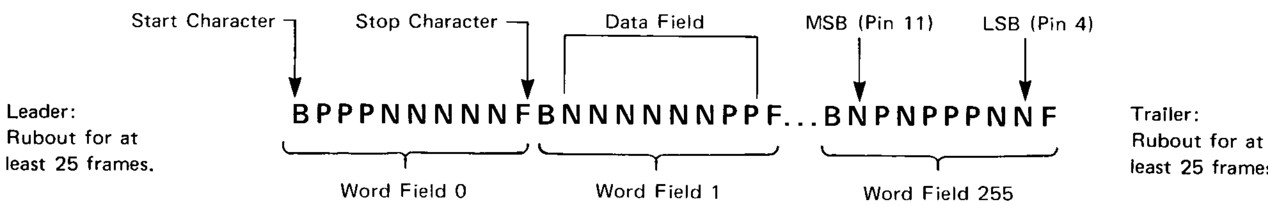
PROGRAMMING THE 1601/1701 AND 1602/1702

Information is introduced by selectively programming "1"s (output high) and "0"s (output low) into the proper bit locations. Note that these ROMs are defined in terms of positive logic.

Word address selection is done by the same decoding circuitry used in the READ mode. The eight output terminals are used as data inputs to determine the information pattern in the eight bits of each word. A low data input level (ground – P on tape) will leave a "1" and a high data input level (+48V – N on tape) will allow programming of "0". All eight bits of one word are programmed simultaneously by setting the desired bit information patterns on the data input terminals.

TAPE FORMAT

The tape reader used with a model 33 ASR teletype accepts 1" wide paper tape using 7 or 8 bit ASCII code. For a tape to correctly program a 1601/1701 or 1602/1702, it must follow exactly the format rules below:



The format requirements are as follows:

1) There must be exactly 256 word fields in consecutive sequence, starting with word field 0 (all address lines low) to program an entire ROM. If a short tape is needed to program only a portion of the ROM, the same format requirements apply.

2) Each word field must consist of ten consecutive characters, the first of which must be the start character B. Following that start character, there must be exactly eight data characters (P's or N's) and ending with the stop character F. NO OTHER CHARACTERS ARE ALLOWED ANYWHERE IN A WORD FIELD. If an error is made while preparing a tape and the stop character "F" has not been typed, a typed "B" will eliminate the previous characters entered. **This is a feature not available on Intel's 7600 programmer; the format shown in the 1601/1701 data sheet must be used when preparing tapes for other programming systems.** An example of this error correcting feature is shown below:

TYPED ON TTY — PROGRAMMED IN ROM

BNNPPNPBNPPPNPNPF ——→ NPPPNPNP

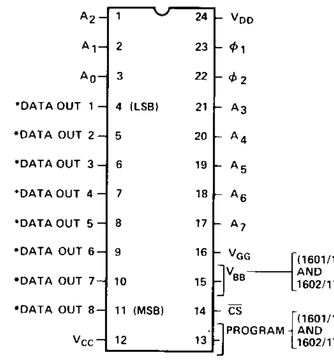
(BNNPPNP: data word eliminated)

If any character other than P or N is entered, a format error is indicated. If the stop character is entered before the error is noticed, the entire word field, including the B and F, must be rubbed out. **Within the word field, a P results in a high level output, and N results in a low level output.** The first data character corresponds to the desired output for data bit 8 (pin 11), the second for data bit 7 (pin 10), etc.

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

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

PROM PIN CONFIGURATION



*THIS PIN IS THE DATA INPUT LEAD FOR THE 1601/1701 AND 1602/1702 DURING PROGRAMMING

IMPORTANT

It should be noted that the PROM's are described in the data sheet with respect to positive logic (high level = p-logic 1). The MCS-8 system is also defined in terms of positive logic. Consider the instruction code for LHD (one of the 48 instructions for the MCS-8).

1 1 1 0 1 0 1 1

When entering this code to the programmer it should be typed,

B P P P N P N P P F

This is the code that will be put into the 1301, Intel's mask programmed ROM, when the final system is defined.

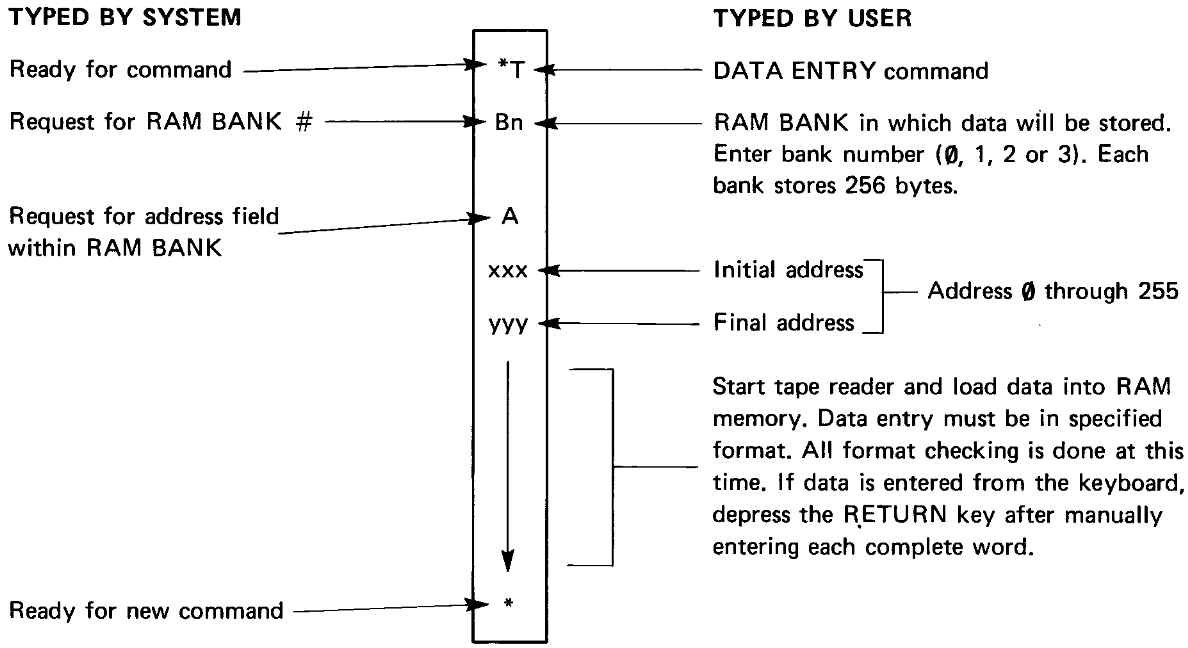
OPERATING THE PROGRAMMER

The SIM8-01 is used as the micro computer controller for the programming. The control program performs the function of a bootstrap loader of data from the TTY into the RAM memory. It then presents data and addresses to the PROM to be programmed and controls the programming pulse. The following steps must be followed when programming a PROM:

1) Place control ROMs in SIM8-01
2) Turn on system power
3) Turn on TTY to "line" position
4) Reset system with an INTERRUPT (Instr. RST = 00 000 101)
5) Change instruction at interrupt port to a NO OP
6) Start system with an INTERRUPT (Instr NO OP = 11 000 000)
7) Load data from TTY into micro computer memory
8) Insert PROM into MP7-03
9) Program PROM
10) Remove PROM from MP7-03. To prevent programming of unwanted bits, never turn power on or off while the PROM is in the MP7-03.

LOADING DATA TO THE MICRO COMPUTER (THE BOOTSTRAP LOADER)

The programming system operates in an interactive mode with the user. After resetting and starting the system with an INTERRUPT [steps 4), 5), 6)], a "*" will appear on the TTY. This is the signal that the system is ready for a command. To load a data tape, the following sequence must be followed:



This RAM bank may be edited by re-entering blocks of data prior to programming a PROM. More than one RAM bank may be loaded in preparation for programming several different PROMs or to permit the merging of blocks of data from different banks into a single PROM. (See the explanation of the CONTINUE command in section IX.)

FORMAT CHECKING

When the system detects the first format error (data words entered either on tape or manually), it will stop loading data and it will print out the address where the format error occurred.

At this time, an "R" may be typed and the data can be RE-ENTERED manually. This is shown below.

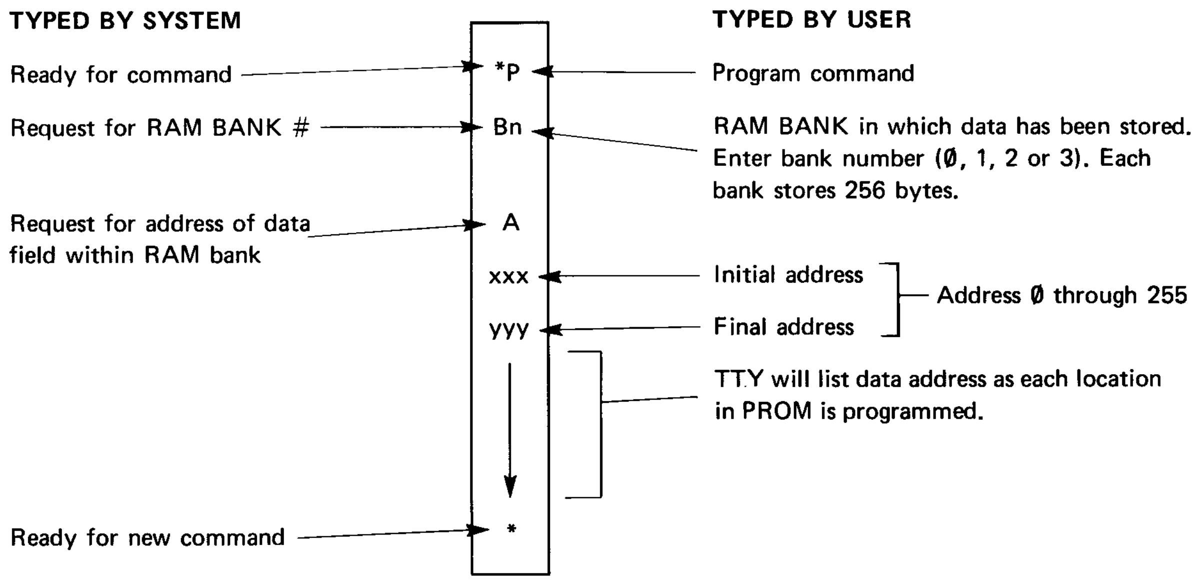
EXAMPLE 1:

Listing by TTY

```
Ø2Ø   B N P N P N P N P F
Ø21   B P P P P N N N N F
Ø22   B N N N N P P P P N    F E
      Ø 2 2   <-- format error indicated at address
                  #022 (too many characters in
                  data field).
      R       <-- RE-ENTER command
      B N N N N P P P P F   <-- Stop tape reader and manually
                                RE-ENTER the data word
Ø23   B N P N P N P N P F   <-- Start the tape reader and continue
Ø24   B P N M    F E
      Ø 2 4   <-- Format error indicated at address
                  #024 (illegal character in data field).
      R       <-- RE-ENTER command
      B P N P N P N P N F   <-- RE-ENTER data
          .
          .
          .   <-- Continue to completion of data
          .       entry.
      *       <-- Ready for new command
```

PROGRAMMING

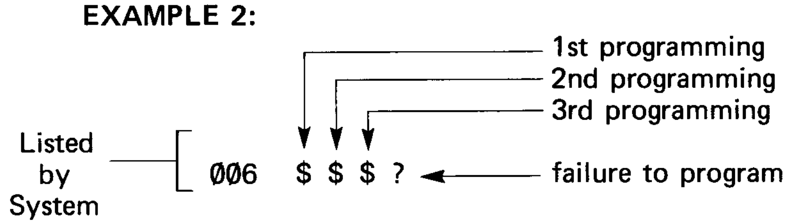
After data has been entered, the PROM may be programmed. Data from a designated address field in a designated RAM bank is programmed into corresponding addresses in the PROM. A complete PROM or any portion of a PROM may be programmed in the following manner:



ERROR CHECKING

After each location in ROM is programmed, the content of the location is read and compared against the programming data. In the event that the programming is not correct, the ROM location will be programmed again. The MCS-8 programming system allows each location of the ROM to be reprogrammed up to four times. A "$" will be printed for each reprogramming. If a location in ROM will not accept a data word after the fourth time, the system will stop programming and a "?" will be printed. This feature of the system guarantees that the programmed ROM will be correct, and incompletely erased or defective ROMs will be identified.

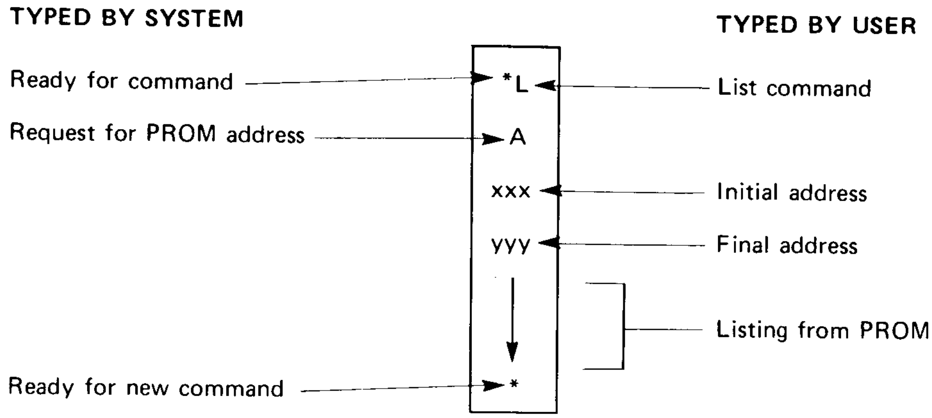
EXAMPLE 2:



If a location in the ROM will not program, a new ROM must be inserted in the programmer. The system must be reset before continuing. (If erasable ROMs are being used, the "faulty" ROM should be erased and reprogrammed).

PROGRAM LISTING

Before or after the programming is finished, the complete content of the ROM, or any portion may be listed on the teletype. A duplicated programming tape may also be made using the teletype tape punch. To list the ROM:



The listing feature may also be used to verify that a 1701 or 1702 is completely erased.

EXAMPLE 3:

```
Ready for command --> *T <------------------------- DATA ENTRY
                      BØ
                      A                              Specification of RAM
                      ØØØ                            memory address
                      Ø1Ø

                      ØØØ   B N P N P N P N N F
                      ØØ1   B P P P P P P P P F
                      ØØ2   B P P P P P P P P F
                      ØØ3   B P P N P P P N P F
                      ØØ4   B P P P P P P P P F      Loading of data listing of
                      ØØ5   B N N P N N P P P F      tape and verifying correct
                      ØØ6   B N P N N P N P P F      format
                      ØØ7   B P N P N P P P P F
                      ØØ8   B N P N P P N P P F
                      ØØ9   B N N N N P P P N F
                      Ø1Ø   B P P N P P P P N F
Ready for command --> *P <------------------------- PROGRAM
                      BØ
                      A                              Specification of PROM
                      ØØ5                            locations to be programmed
                      ØØ8

                      ØØ5
                      ØØ6                            Programming of PROM and
                      ØØ7                            verifying correct transfer of
                      ØØ8                            data
Ready for command --> *L <------------------------- LIST
                      A
                      ØØØ                            Address specification
                      Ø1Ø

                      ØØØ   B P P P P P P P P F
                      ØØ1   B P P P P P P P P F
                      ØØ2   B P P P P P P P P F
                      ØØ3   B P P P P P P P P F
                      ØØ4   B P P P P P P P P F
                      ØØ5   B N N P N N P P P F      Listing of PROM
                      ØØ6   B N P N N P N P P F
                      ØØ7   B P N P N P P P P F
                      ØØ8   B N P N P P N P P F
                      ØØ9   B P P P P P P P P F
                      Ø1Ø   B P P P P P P P P F

Ready for command --> *
```

1701, 1702 ERASING PROCEDURE

The 1701 and 1702 may be erased by exposure to high intensity short-wave ultraviolet light at a wavelength of 2537 Å. The recommended integrated dose (i.e., UV intensity x exposure time) is 6W-sec/cm^2. Example of ultraviolet sources which can erase the 1701 or 1702 in 10 to 20 minutes is the Model S-52 and Model UVS-11 short-wave ultraviolet lamps manufactured by Ultra-Violet Products, Inc. (San Gabriel, California). **The lamps should be used without short-wave filters, and the 1701 or 1702 to be erased should be placed about one inch away from the lamp tubes.**

B. MP7-03 Programming System

The MP7-03 is the PROM programming board which easily interfaces with the SIM8-01. All address and data lines are completely TTL compatible. The MP7-03 requires +5VDC @ 0.8 amps, –9 VDC @ 0.1 amps, and 50 Vrms @ 1 amp. Two Stancor P8180 (or equivalent) filament transformers (25.2 Vrms @ 1 amp) with their secondaries connected in series provide the 50 Vrms.

This programmer board is the successor of the MP7-02. The MP7-03 enables programming of Intel's new 1702A, a pin-for-pin replacement for the 1702.

When the MP7-03 is used under SIM8-01 control with control ROM A0862 replaced by A0863, the 1702A may be programmed an order of magnitude faster than the 1702, less than three minutes.

> **IMPORTANT:**
> Only use the A0863 control PROM when programming the new 1702A. Never use it when programming the 1702. The programming duty cycle is too high for the 1702 and it may be permanently damaged.

The MP7-03 features three data control options:

1) Data-in switch (Normal-Complement). If this switch is in the complement position, data into the PROM is complemented.

2) Data-out switch (Normal-Complement). If this switch is in the complement position, data read from the PROM is complemented.

3) Data-out switch (Enable-Disable). If this switch is in the enable position, data may be read from the PROM. In the disable position, the output line may float up to a high level (logic "1"). As a result, the input ports on the prototype system may be used for other functions without removing the MP7-03 card.

MP7-03 Programmer Board Specifications

Features:

- High speed programming of Intel's new 1702A (three minutes)
- Inputs and outputs TTL compatible
- Board sold complete with transformers, capacitor and connector
- Directly interfaces with SIM8-01 Board

Connector:

a. Solder lug type/Amphenol 72 pin connector P/N 225-23621-101
b. Wire wrap type - Amphenol 72 pin connector P/N 261-15636

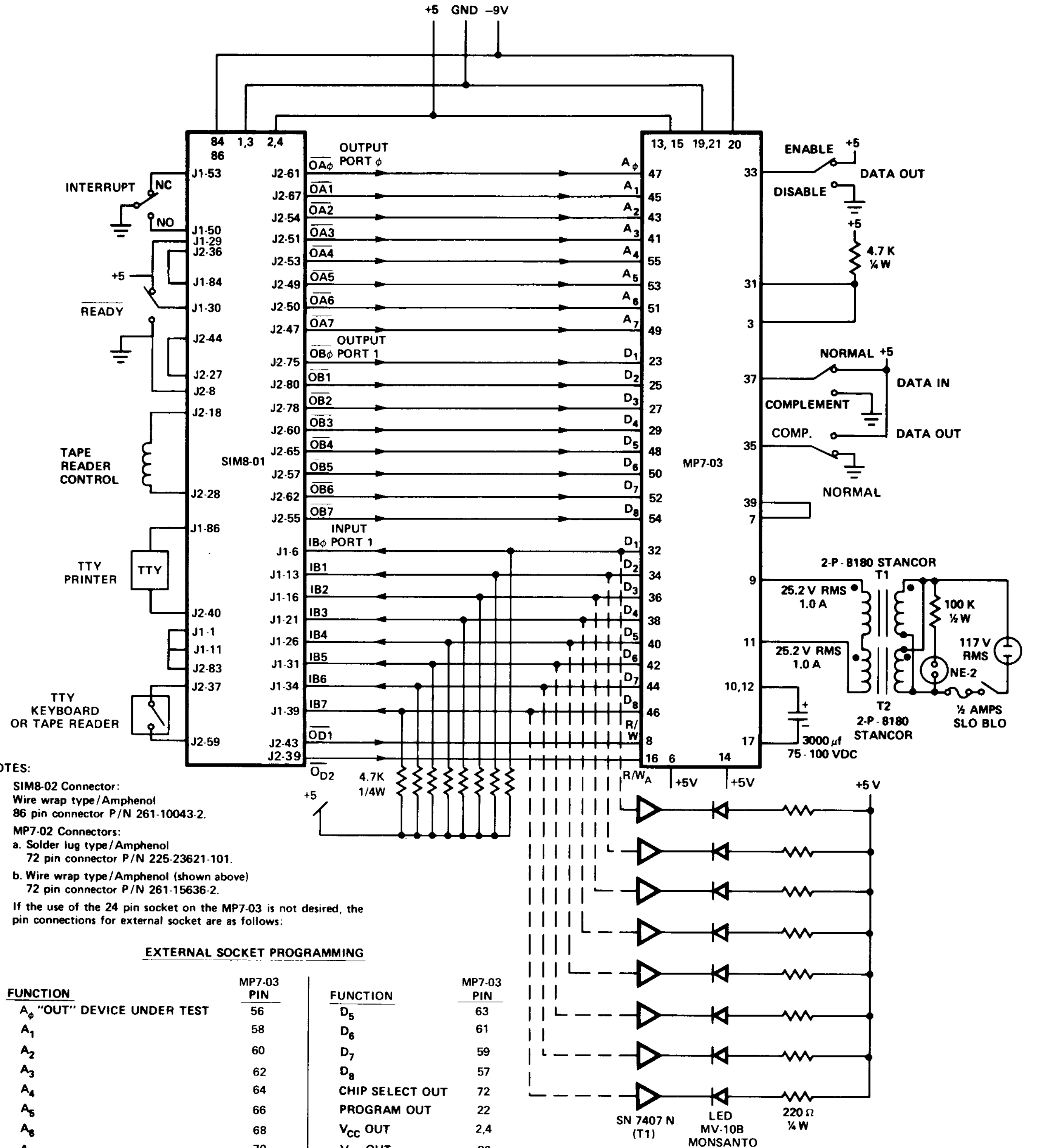
Dimensions:
8.4 inches high
9.5 inches deep

Power Requirement:
V_{CC} = +5 @ 0.8 amps
TTL GRD = 0V
*V_{DD} = –9V @ 0.1 amps
V_P = 50Vrms @ 1 amp

*This board may be used with a –10V supply because a pair of diodes (i.e. 1N914 or equivalent) are located on the board in series with the supply. Select the appropriate pin for either –9V or –10V operation.

A micro computer bulletin which describes the modification of the MP7-02 for programming the 1602A/1702A is available on request. These modifications include complete failsafe circuitry (now on MP7-03) to protect the PROMs and the 50V power supply.

C. Programming System Interconnection



NOTES:

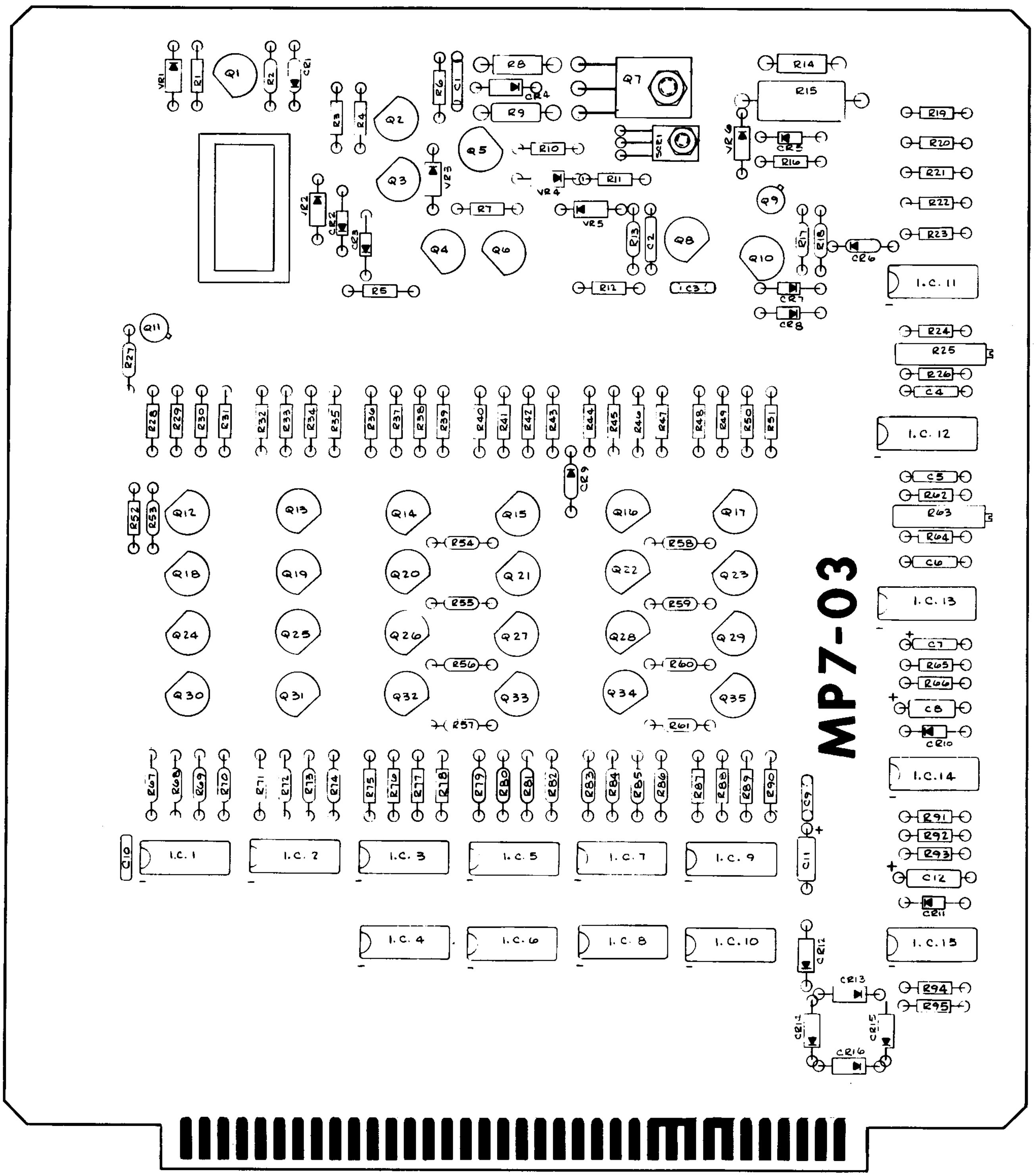
1. SIM8-02 Connector:
   Wire wrap type/Amphenol
   86 pin connector P/N 261-10043-2.
2. MP7-02 Connectors:
   a. Solder lug type/Amphenol
      72 pin connector P/N 225-23621-101.
   b. Wire wrap type/Amphenol (shown above)
      72 pin connector P/N 261-15636-2.
3. If the use of the 24 pin socket on the MP7-03 is not desired, the pin connections for external socket are as follows:

EXTERNAL SOCKET PROGRAMMING

FUNCTION	MP7-03 PIN	FUNCTION	MP7-03 PIN
A_ϕ "OUT" DEVICE UNDER TEST	56	D_5	63
A_1	58	D_6	61
A_2	60	D_7	59
A_3	62	D_8	57
A_4	64	CHIP SELECT OUT	72
A_5	66	PROGRAM OUT	22
A_6	68	V_{CC} OUT	2,4
A_7	70	V_{GG} OUT	26
D_1 "OUT" DEVICE UNDER TEST	71	V_{BB} OUT	24
D_2	69	V_{DD} OUT	30
D_3	67	$\phi 1, \phi 2$ OUT	2
D_4	65		

The complete interconnection between the SIM8-01 and the MP7-03 is provided by the MCB8-10 system interface and control module.

Figure 16. MP7-03/Sim8-01 PROM Programming System



Solder Connector P/N 225-23621-101 R̄ P̄ N̄ M̄ L̄ K̄ J̄ H̄ F̄ Ē D̄ C̄ B̄ Ā Z Y X W V U T S R P N M L K J H F E D C B A

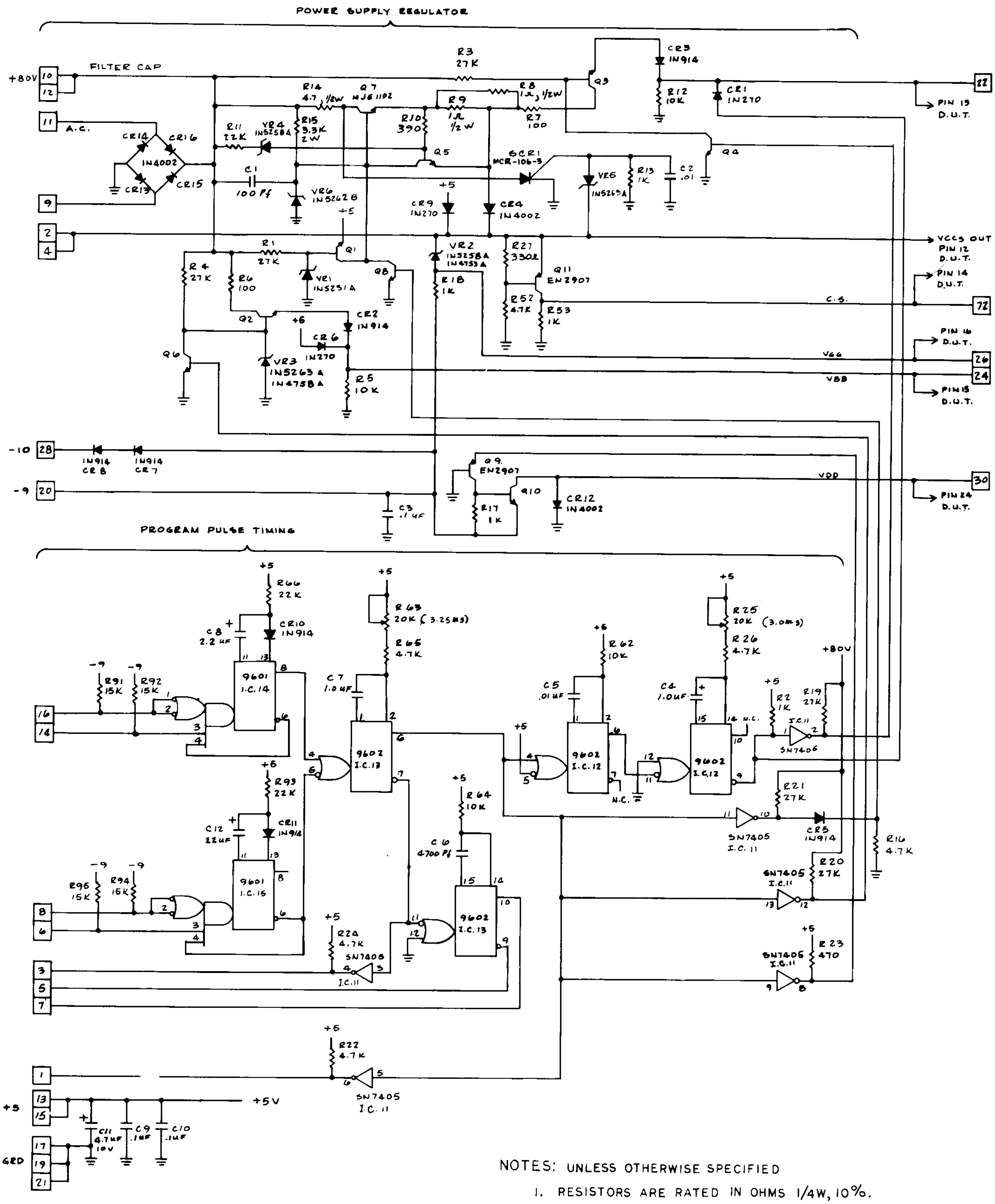
Wirewrap Connector P/N 261-15636-2 71 69 67 65 63 61 59 57 55 53 51 49 47 45 43 41 39 37 35 33 31 29 27 25 23 21 19 17 15 13 11 9 7 5 3 1

Figure 17a. Component Side of MP7-03 Card

Solder Connector P/N 225-23621-101 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 32 33 34 35 36

Wirewrap Connector P/N 261-15636-2 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72

Figure 17b. Pin Definition – Reverse Side of MP7-03 Card



NOTES: UNLESS OTHERWISE SPECIFIED

1. RESISTORS ARE RATED IN OHMS 1/4W, 10%.
2. TRANSISTORS ARE SE6021, OR 2N3658 OR 2N3722.

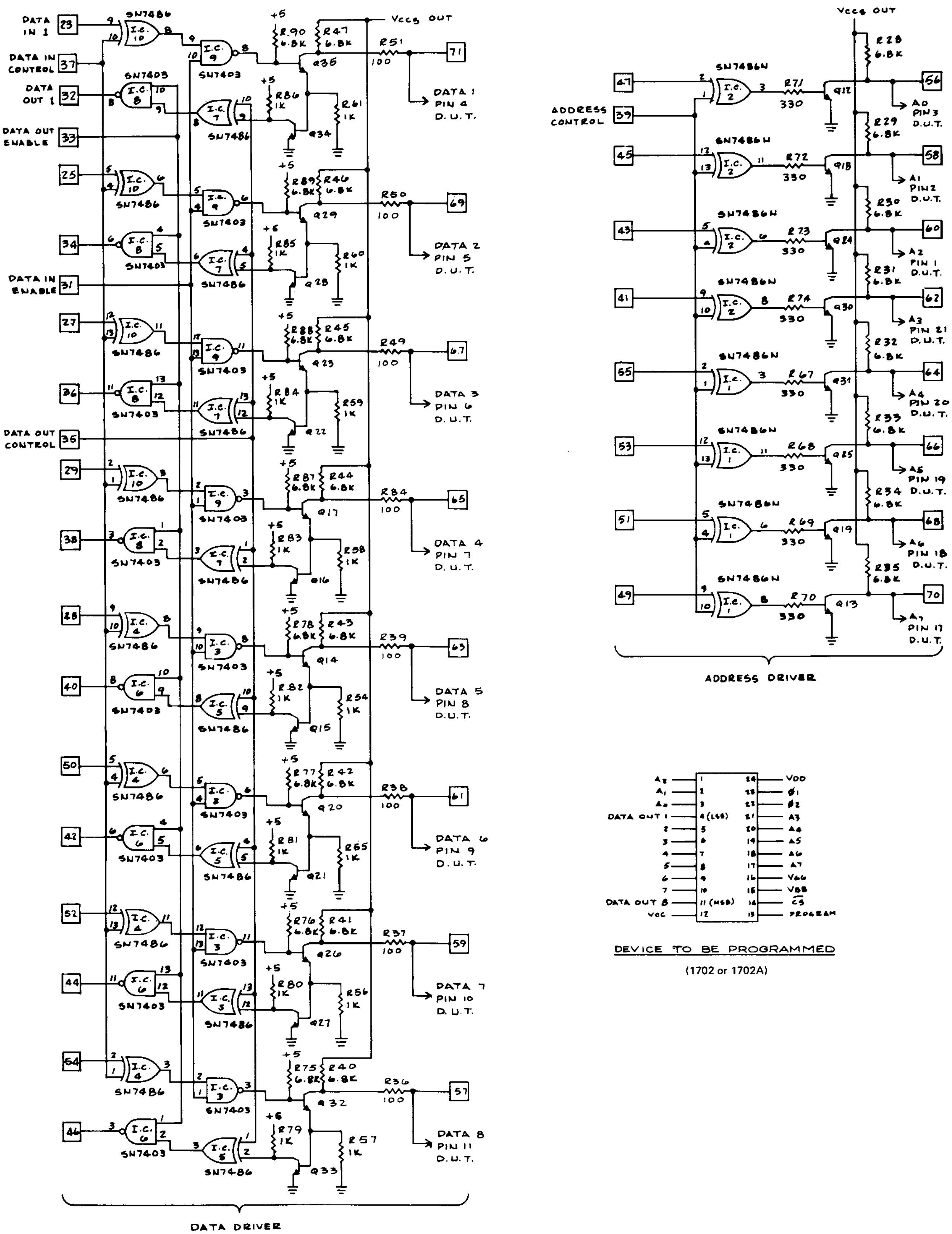


Figure 18. MP7-03 PROM Programmer Board Schematic

IX. PROGRAM ASSEMBLY AND EXECUTION

A. 8008 Assembler Software Package

The 8008 Assembler generates object programs from symbolic assembly language instructions. 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 addresses.

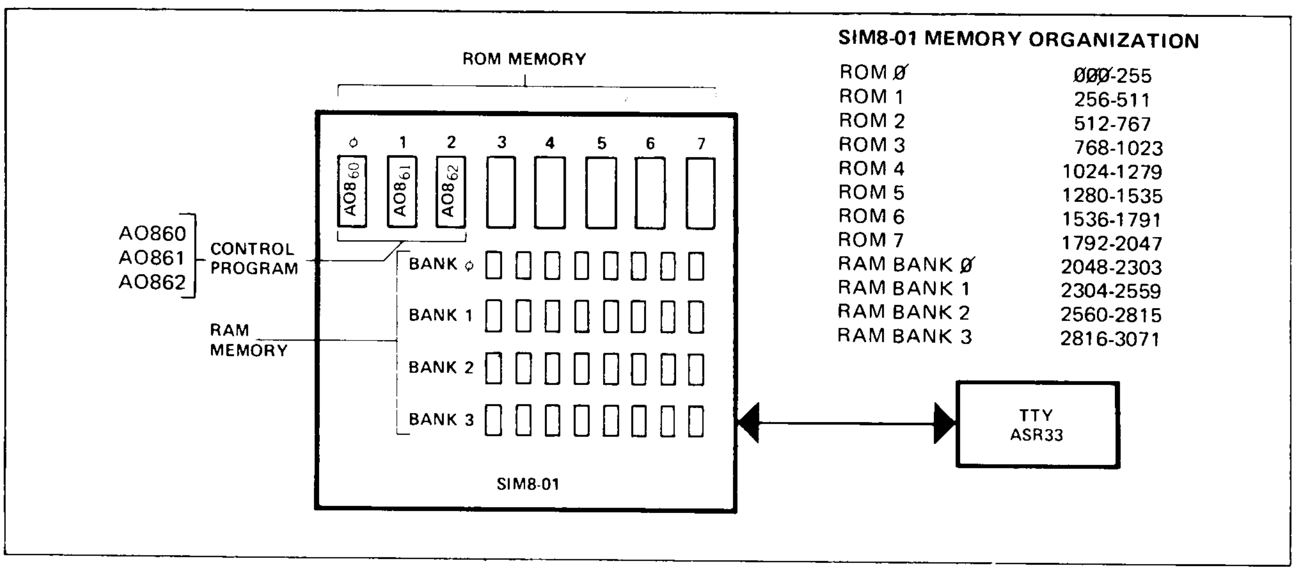
The Assembler is designed to operate from a time shared terminal with input by paper tape or directly from the terminal keyboard. The assembled program is punched out at the terminal in 8 level binary paper tape.

The Assembler is written in batch FORTRAN IV and is designed to run on an XDS 940 or on other machines having FORTRAN IV capability and a word length of 24 bits or more. Modifications to the program may be required for machines other than the XDS 940.

This program is currently available from nationwide computer timesharing services for timesharing computer users or as a software package for adaptation to another machine. Assembler software specifications are available on request.

B. Execution of Programs From RAM on SIM8-01 Using Memory Loader Control Programs.

The previous section provided a description of the preparation of tapes and the programming of PROMs for permanently storing the micro computer programs. During the system development, programs may be loaded, stored, and executed directly from RAM memory. This section explains these additional features.



MCS-8 Operating System

The system has three basic parts

1) The micro computer (SIM8-01)
2) The bootstrap memory loader control program (A0860, A0861, A0862)
3) ASR 33 (Automatic Send Receive) Teletype

The control program provides the complete capability for executing programs from RAM. Two additional program commands are required; "C", the CONTINUE command for loading more than one bank of memory, and "E", the program EXECUTION command.

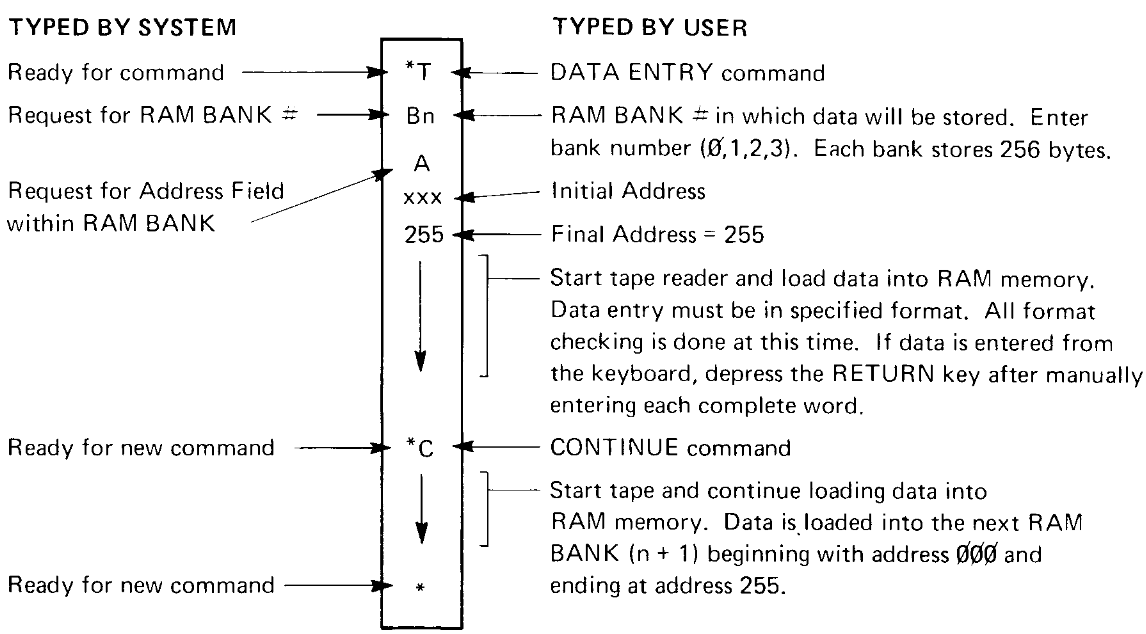
OPERATING THE MICRO COMPUTER SYSTEM

To use the SIM8-01 as the micro computer controller for the bootstrap loading of a program from the TTY into RAM memory and the execution of programs stored in RAM, the following steps must be followed:

1) Place control ROMs in SIM8-01
2) Turn on system power
3) Turn on TTY to "line" position
4) Reset system with an INTERRUPT (Instr. RST = 00 000 101)
5) Change instruction at interrupt port to a NO OP
6) Start system with an INTERRUPT (Instr. NO OP = 11 000 000)
7) Load data from TTY into micro computer RAM memory
8) Execute the program stored in RAM

LOADING OF MULTIPLE RAM BANKS

Through the use of the command "C", (CONTINUE) subsequent RAM banks may be loaded with data without entering a new data entry command and new memory bank and address designations.



Note that the CONTINUE command should only be used when the subsequent RAM will be completely loaded with 256 bytes of data. For partial loading of RAM banks, always use the DATA ENTRY command. The content of a RAM bank may be edited by using the DATA ENTRY command and revising and re-entering sections of the bank. **When a program is being stored in memory, the first instruction of the program should be located at address ØØØ in a RAM bank.** The entire RAM memory with the exception of the last fifteen bytes of RAM bank 3 may be used for program or data storage in conjunction with the bootstrap loader.

PROGRAM EXECUTION

The program which has been loaded into RAM may be executed directly from RAM.

TYPED BY SYSTEM		TYPED BY USER
Ready for command →	*E	← Program EXECUTION command
Request for RAM BANK # →	Bn	← RAM BANK in which the program has been stored. *The first instruction in a program must be at address ØØØ in a RAM bank.*
		Program beginning at address ØØØ of RAM BANK # n will be executed by the MCS-8 system. *To return to the bootstrap control program, the ending statement of the program being executed should be "JMP 462"*
Ready for command →	*	

CAUTION: *When executing a program from a single RAM bank or multiple RAM banks, care must be taken to insure that all JUMP addresses and subroutine CALL addresses are appropriately assigned within the memory storage being used.*

SUMMARY OF SYSTEM COMMANDS

Using Intel's special control ROMs (AO860, AO861, AO862) the following control commands are available:

COMMAND	EXPLANATION
T	**DATA ENTRY** – Enter data from TTY into a RAM bank
C	**CONTINUE** – Continue entering 256 byte blocks of data into subsequent RAM banks
R	**RE-ENTER** – Re-enter a data word where a format error has occurred and continue entering data
E	**EXECUTE** – Execute the program stored in RAM memory
P	**PROGRAM** – Program a PROM using data stored in RAM memory
L	**LIST** – List the content of the PROM on the TTY

Listing of Bootstrap Loader Program
(Intel tape numbers A0860, A0861, A0862, October 2, 1972)

```
BEGIN  LAI 1              SUPPRESS TTY
       OUT 12B            OUTPUT 2
       XRA                CLEAR AC
       OUT 13B            OUTPUT 3 - TAPE READER CONTROL
       HLT
       JMP START
*
*TELETYPE TAPE READER & I/O CONTROL
*
TAPE   LAI 1              TAPE READER ENABLE CODE
       OUT 13B            OUTPUT 3 - ENABLE TAPE READER
TTY    HLT                WAIT FOR TTY START PULSE
       LDI 194            TTY DELAY - 4 MSEC.
ST2    IND
       JFZ ST2
       XRA                TAPE READER DISABLE CODE
       OUT 13B            OUTPUT 3, DISABLE TAPE READER
       OUT 12B            OUTPUT 2, OUTPUT START PULSE
       LEI 248            TTY DATA SAMPLING COUNTER
TTYIN  CAL TTYD1          TTY DELAY - 8.7 MSEC.
       INP 0B             READ TTY DATA INPUT
       XRI 255            COMPLEMENT TTY DATA
       OUT 12B            OUTPUT 2, TTY DATA OUT
       RAR                STORE TTY DATA
       LAB                LOAD TTY DATA TO REG. B
       RAR
       LBA                LOAD AC TO REG. B
       INE                E = E + 1
       JFZ TTYIN          JUMP IF ZERO F/F IS NOT SET
       LAB                LOAD REG. B TO AC
       NDI 127            REMOVE PARITY BIT
       LBA                STORE TTY INPUT DATA
       CAL TTYD1
       LAI 1
       OUT 12B            SUPPRESS TTY
       RET
       LAA                NOP
       LAA
       LAA
       LAA
       LAA
*
*TTY DELAY - 8.7 MSEC.
*
TTYD1  LDI 121            8.7 MSEC. DELAY
ST     IND                D=D+1
       JFZ ST
       RET
*
*BCD TO BINARY CONVERSION
*
BCDBIN LAM                LOAD LSD TO A
       SUI 48             AC=AC-48
       LBA                LOAD A TO B
       DCL                L=L-1
       LAM                LOAD M TO A
       SUI 48             A=A-48
       LEA                LOAD A TO E
BB1    JTZ BB2            IF A=0 JUMP
       LAI 10             AC=10
       ADB                AC=AC+B
       LBA                LOAD AC TO REG. B
       DCE                E=E-1
       JMP BB1
BB2    DCL                L=L-1
       LAM                LOAD M TO A
       SUI 48             A=A-48
       LEA                LOAD A TO E
BB3    JTZ BB4
       LAI 100            AC=100
       ADB                AC=AC+B
       LBA                LOAD AC TO REG. B
       DCE                E=E-1
       JMP BB3
BB4    RET
*
*BINARY TO BCD CONVERSION
*


BINBCD LHI 11
       LLI 241
BNBD   LCI 0              CLEAR REG. C
       LAB
BD1    SUI 100            AC=AC-100
       JTC BD2            JUMP IF AC<100
       INC                C=C+1
       JMP BD1
BD2    LBI 100            LOAD 100 TO REG. B
       ADB                AC=AC+B
       LBA                LOAD AC TO REG. B
       LAI 48             A=A+48
       ADC                A=A+C
       LMA                LOAD A TO MEMORY
       LCI 0              CLEAR REG. C
       LAB                LOAD B TO A
```

```
*TYPE A AND IDENTIFY INITIAL AND FINAL LOCATION
*
ADRESL CAL CRLF
       LBI 301B           LOAD (A)
       CAL TTYOUT         TYPE (A)
AD1    CAL CRLF
       LCI 253            C=253
AD2    CAL TTY            CALL FOR TTY KB INPUT
       INL                L=L+1
       LMB                LOAD TTY KB INPUT TO M
       INC                C=C+1
       JFZ AD2            JUMP IF C IS NOT ZERO
       RET
*
*DATA INPUT ROUTINE
*
DATAIN CAL TAPE           READ TAPE
       LAI 102B           LOAD (B)
       CPB                SEARCH FOR (B)
       JFZ DATAIN         JUMP IF IT IS NOT (B)
DATA1  LHI 11             H=11
       LLI 255            L=255
       LAI 248            DATA BIT COUNTER
       LMA                STORE DATA BIT CONTR
DATA2  CAL TAPE           READ TAPE
       LLI 250            MEMORY LOC. FOR DATA
       LAI 120B           LOAD (P)
       CPB                SEARCH FOR (P)
       JTZ PDATA          IF (P) STORE (1)
       LAI 116B           LOAD (N)
       CPB                SEARCH FOR (N)
       JTZ NDATA          IF (N) STORE (0)
       LAI 102B           LOAD (B)
       CPB                SEARCH FOR (B)
       JTZ DATA1          IF (B) DELETE LAST INSTRCTION
       LAI 177B           LOAD (RO)
       CPB                SEARCH FOR RUBOUT
       JFZ FMEROR         JUMP IF NOT RUBOUT
       CAL RUBOUT         CALL FOR RUBOUT ROUTINE
       JMP DATA2
FMEROR CAL FORMAT         CALL FOR FORMAT ERROR ROUTINE
       JMP DATAEN
PDATA  LAI 1              REPLACE (P) WITH (1)
       RAR                ROTATE RICHT
       LAM
       RAL                ROTATE LEFT
       LMA
       JMP DATA3
NDATA  XRA                CLEAR AC AND CARRY
       LAM
       RAL                ROTATE LEFT
       LMA
DATA3  LLI 255
       LBM                LOAD M TO B
       INB                INC DATA BIT COUNTER
       LMB
       JFZ DATA2          JUMP IF B IS NOT ZERO
FDATA  CAL TAPE           CALL FOR TAPE INPUT
       LAI 106B           LOAD (F)
       CPB                SEARCH FOR (F)
       JTZ DATA4          STORE DATA IF IT IS (F)
       LAI 102B           LOAD (B)
       CPB                SEARCH FOR (B)
       JTZ DATA1          DELETE LAST INSTRUCTION IF IT IS (B)
       LAI 177B           LOAD (RO)
       CPB                SEARCH FOR (RO)
       JFZ FMEROR         JUMP IF IT IS NOT (RO)
       CAL RUBOUT         CALL FOR (RO) ROUTINE
       JMP DATA2
DATA4  XRA                CLEAR AC AND CARRY
DATAEN RET
*
*RUBOUT ROUTINE
*
RUBOUT LAA                NOP
       LAA
       LAA
       LAA
       LAA
       LAA
       LAA
       RET
*
*FORMAT ERROR ROUTINE
*
FORMAT LBI 240B           LOAD (SP)
       CAL TTYOUT         TYPE (SP)
       LBI 306B           LOAD (F)
       CAL TTYOUT         TYPE (F)
       LBI 305B           LOAD (E)
       CAL TTYOUT         TYPE (E)
LISTA  CAL CRLF
PRINTA LLI 253            L=253
       LBM                LOAD MEMORY TO B
       CAL BINBCD         BIN TO BCD CONV
       LEI 253            E=253
```

```
BD3     SUI 10                      AC=AC-10
        JTC BD4                     JUMP IF AC<10
        INC                         C=C+1
        JMP BD3
BD4     LBI 10                      B = 10
        ADB                         AC=AC+B
        LBA                         LOAD AC TO REC B
        LAI 48                      A=A+48
        ADC                         A=A+C
        INL                         L=L+1
        LMA                         LOAD A TO M
        LAI 48                      A=A+48
        ADB                         A=A+B
        INL                         L=L+1
        LMA                         LOAD A TO M
        RET                         RETURN
*
*TTY OUTPUT ROUTINE
*
TTYOUT LCI 253                      C=253
TTYO    CAL TTYD1                   DELAY - 9.012 MSEC.
        INC                         C=C+1
        JFZ TTYO
        XRA
        OUT 12B                     TTY START PULSE
        LCI 248                     REG C=248
TTY1    CAL TTYD1                   TTY DELAY - 9.012 MSEC.
        LAB                         LOAD DATA TO AC
        OUT 12B                     OUTPUT DATA
        RAR                         STORE DATA IN CARRY
        LBA                         LOAD A TO B
        LAI 0                       AC = 0
        RAR                         RESTORE DATA BIT
        ADB                         RESTORE DATA
        LBA                         STORE
        INC                         C=C+1
        JFZ TTY1                    JUMP IF AC IS NOT ZERO
        CAL TTYD1                   TTY DELAY - 9.012 MSEC.
        LAI 1                       A=A+1
        OUT 12B                     SUPPRESS TTY
        RET
*
*CARRIAGE RETURN & LINE FEED
*
CRLF    LBI 215B                    CARRIAGE RETURN - CR
        CAL TTYOUT                  TYPE CR
LF      LBI 212B                    LINE FEED - LF
        CAL TTYOUT                  TYPE LF
        RET
*
*ERROR SIGNAL
*
ERROR   LBI 277B                    (?)
        CAL TTYOUT                  TYPE (?)
        RET
*
*TYPE B AND IDENTIFY RAM BANK
*
ADRESH CAL CRLF
        LBI 302B                    LOAD (B)
        CAL TTYOUT                  TYPE (B)
        CAL TTY                     CALL FOR TTY KB INPUT
        LMB                         STORE INPUT IN MEMORY
        RET
*
START   CAL CRLF
        LBI 252B                    B=252B
        CAL TTYOUT                  TYPE (*)
        CAL TTY                     CALL FOR TTY KB INPUT
        LAI 124B                    LOAD (T) TO AC
        CPB                         AC-B
        JTZ TAPEIN                  JUMP IF AC-B=0
        LAI 105B                    AC=105B,(E)
        CPB                         AC-B
        JTZ EXECUT                  JUMP IF AC-B=0
        LAI 122B                    AC=122B, (R)
        CPB                         AC-B
        JTZ READIN                  JUMP IF AC-B=0
        LAI 103B                    AC=103B, (C)
        CPB                         AC-B
        JTZ CONTIN                  JUMP IF AC-B=0
        LAI 114B                    AC=114B, (L)
        CPB                         AC-B
        JTZ LISTIN                  JUMP IF AC-B=0
        LAI 120B                    AC=120B, (P)
        CPB                         AC-B
        JTZ PROGRM                  JUMP IF AC-B=0
        CAL ERROR                   TYPE (?)
        JMP START
*
*LOAD DATA INPUT TO 1101 RAM
*
TAPEIN CAL ENTERA                   ENTER ADDRESS
READIN CAL DATAIN                   READ TAPE INPUT ROUTI
        RAR                         CHECK FOR FE FLAG
        JTC START                   JUMP IF CARRY=1
        DCL                         L=L-1
        DCL                         L=L-1
FM1     LAM                         LOAD MSD TO AC
        ADI 128                     AC=AC+128
        LBA                         LOAD AC TO B
        CAL TTYOUT                  TYPE BCD LOCATION
        INL                         L=L+1
        INE                         E=E+1
        JFZ FM1                     JUMP IF E IS NOT 0
        LAI 1                       FORMAT ERROR FLAG
        RET
*
*ENTER ADDRESS AND CONVERT THEM INTO BINARY REP.
*
ENTERA LHI 11                       H=11
        LLI 240                     L=240
ENTERH CAL ADRESH                   ENTER BANK NO.
ENTERL CAL ADRESL                   ENTER INITIAL ADDRESS
        CAL AD1                     ENTER FINAL ADDRESS
        CAL CRLF
        LLI 246                     L=246
        CAL BCDBIN                  FINAL ADRES-BINARY
        LCB                         LOAD B TO C
        DCL                         L=L-1
        CAL BCDBIN                  INITIAL ADRES-BINARY
        DCL                         L=L-1
        LAM                         AC=M
        SUI 48                      AC=AC-48
        ADI 8                       AC=AC+8
        LLI 252                     L=252
        LMA                         STORE BANK NO IN M
        INL                         L=L+1=253
        LMB                         STORE INITIAL ADRES IN M
        INL                         L=L+1=254
        LMC                         STORE FINAL ADRES IN M
        RET
*
*SET ADDRESS TO 1101 RAM
*
SETMA   LHI 11                      H=11
        LLI 252                     L=252
        LDM                         BANK NO TO D
        INL                         L=L+1=253
        LAM                         INIT ADR TO E
        OUT 10B                     WRITE ADDRESS TO OUT 0
        LLA                         LOAD AC TO L
        LHD                         D TO H = BANK NO
        RET
*
*ADDRESS CHECKINC
*
ACHECK LHI 11                       H=11
        LLI 254                     L=254
        LAM                         LOAD FINAL ADRES. TO AC
        DCL                         L=L-1=253
        CPM                         COMPARE:AF-AI
        JTZ CHECK                   JUMP IF AF-AI=0
        LCM                         LOAD AI TO AC
        INC                         AI=AI+1
        LMC                         LOAD AI TO MEMORY
CHECK   RET
*
*PROGRAM BEGINS
*

LISTIN LHI 11                       H=11
        LLI 240                     L=240
        CAL ENTERL                  ENTER INITIAL & FINAL ADR.
LISTER CAL CRLF
        LLI 251                     L=251
        LAI 252                     NO. OF INSTR. PER LINE
        LMA                         LOAD AC TO MEMORY
LIST1   CAL PRINTA                  PRINT ADDRESS
        LBI 240B                    LOAD [SP]
        CAL TTYOUT                  PRINT [SP]
        LBI 302B                    LOAD [B]
        CAL TTYOUT                  PRINT [B]
        LLI 253                     L=253
        LAM                         LOAD AI TO AC
        OUT 10B                     OUTPUT AI TO OUT 0
        LEI 248                     READ DELAY/DATA BIT CONTR
        INP 1B                      READ INPUT FROM 1702
LIST2   RAL                         L=249
        LLI 249                     SAVE INPUT DATA
        LMA
        JTC PRINTP                  PRINT [P] IF CARRY=1
        LBI 316B                    LOAD [N]
        CAL TTYOUT                  PRINT [N]
        JMP LIST3
PRINTP LBI 320B                     LOAD [P]
        CAL TTYOUT                  PRINT [P]
LIST3   LAM                         LOAD DATA TO AC
        INE                         E=E+1
        JFZ LIST2                   JUMP IF E IS NOT 0
        LBI 306B                    LOAD [F]
        CAL TTYOUT                  PRINT [F]
```

```
        LLI 250                   L=250
        LCM                       LOAD MEMORY TO C
        CAL SETMA                 SET MEMORY ADDRESS
        LAC
        OUT 11B
        LMA                       LOAD DATA TO MEMORY
        CAL ACHECK                COMPARE AF AND AI
        JTZ START                 JUMP IF A=0
        JMP READIN                READ INPUT DATA
EXECUT  LHI 11                    H=11
        LLI 240                   L=240
BANK0   EQU 4000B                 BANK 0 LOCATION
BANK1   EQU 4400B                 BANK 1 LOCATION
BANK2   EQU 5000B                 BANK 2 LOCATION
BANK3   EQU 5400B                 BANK 3 LOCATION
        CAL ADRESH                ENTER BANK NO
        CAL CRLF
        LAM                       LOAD MEMORY TO AC
        SUI 48                    AC=AC-48
        ADI 8                     AC=AC+8
        LHA                       LOAD AC TO H
        LAI 8                     AC=8
        CPH                       AC=AC-H
        JTZ BANK0                 JUMP,IF AC=0
        LAI 9
        CPH
        JTZ BANK1
        LAI 10
        CPH
        JTZ BANK2
        LAI 11
        CPH
        JTZ BANK3
        CAL ERROR
        JMP START
CONTIN  LHI 11
        LLI 252
        LDM
        IND                       D=D+1
        LMD                       BANK=BANK+1
        INL                       L=L+1
        XRA                       CLEAR AC
        LMA                       INITIAL ADRES=0
        INL
        LAI 255
        LMA                       FINAL ADRES=255
        JMP READIN
*
*PROM LISTING ROUTINE
*
        LBI 240B                  LOAD [SP]
        CAL TTYOUT                PRINT [SP]
        CAL ACHECK                AF - AI
        JTZ START
        LLI 251                   LOAD LINE CONTR. TO AC
        LCM                       LOAD MEMORY TO C
        INC                       C=C+1
        LMC
        JTZ LISTER                JUMP IF LINE CONTR.=4
        JMP LIST1
*
*PROM PROGRAMMER
*
PROGRM  CAL ENTERA                ENTER MEMORY ADDRESS
PG1     LLI 255                   REPROGRAM CONTR.
        LAI 253                   AC=253
        LMA                       LOAD AC TO MEMORY
        CAL LISTA                 PRINT ADDRESS
PG2     CAL SETMA                 SET ADDRESS TO 1702
        LAI 255                   COMPLEMENT INPUT DATA
        XRM                       LOAD DATA TO AC
        OUT 11B                   WRITE DATA TO OUT 1
        LAI 2                     AC=2, DELAY
        OUT 13B                   PROGRAM PULSE ENABLE
        LCI 254                   DELAY = 5 SEC.
PG3     LEI 0                     E=0
PG4     CAL TTYD1                 DELAY - 8.7 MSEC.
        INE                       E=E+1
        JFZ PG4                   JUMP IF E IS NOT 0
        INC                       C=C+1
        JFZ PG3                   JUMP IF D IS NOT 0
        LAI 0                     AC=0
        OUT 13B                   DISABLE PROGRAM PULSE
        INP 1B                    READ DATA FROM 1702
        CPM                       COMPARE DATA
        JTZ PC5                   JUMP IF COMPARED
        LBI 244B                  LOAD [$]
        CAL TTYOUT                PRIN[$]
        LHI 11
        LLI 255
        LBM
        INB
        LMB                       LOAD B TO MEMORY
        JFZ PG2
        CAL ERROR                 PRINT [?]
        JMP START
PG5     CAL ACHECK
        JTZ START
        JMP PG1                   CONTINUE PROG. NEXT INSTR.
        END
```

APPENDIX I

FUNCTIONAL DEFINITION

Symbols	Meaning
<B2>	Second byte of the instruction
<B3>	Third byte of the instruction
r	One of the scratch pad register references: A, B, C, D, E, H, L
c	One of the following flag flip-flop references: C, Z, S, P
C_4C_3	Flag flip-flop codes — Condition for True 00 carry — Overflow, underflow 01 zero — Result is zero 10 sign — MSB of result is "1" 11 parity — Parity of result is even
M	Memory location indicated by the contents of registers H and L
( )	Contents of location or register
$\wedge$	Logical product
$\forall$	Exclusive "or"
V	Inclusive "or"
A_m	Bit m of the A-register
STACK	Instruction counter (P) pushdown register
P	Program Counter
$\leftarrow$	Is transferred to
XXX	A "don't care"
SSS	Source register for data
DDD	Destination register for data Register # (SSS or DDD) — Register Name 000 — A 001 — B 010 — C 011 — D 100 — E 101 — H 110 — L

INDEX REGISTER INSTRUCTIONS

LOAD DATA TO INDEX REGISTERS – One Byte

Data may be loaded into or moved between any of the index registers, or memory registers.

Lr_1r_2 (one cycle – PCI)	11 DDD SSS	$(r_1) \leftarrow (r_2)$ Load register r_1 with the content of r_2. The content of r_2 remains unchanged. If SSS=DDD, the instruction is a NOP (no operation).
LrM (two cycles – PCI/PCR)	11 DDD 111	$(r) \leftarrow (M)$ Load register r with the content of the memory location addressed by the contents of registers H and L. (DDD≠111 – HALT instr.)
LMr (two cycles – PCI/PCW)	11 111 SSS	$(M) \leftarrow (r)$ Load the memory location addressed by the contents of registers H and L with the content of register r. (SSS≠111 – HALT instr.)

LOAD DATA IMMEDIATE – Two Bytes

A byte of data immediately following the instruction may be loaded into the processor or into the memory

LrI (two cycles – PCI/PCR)	00 DDD 110 $<B_2>$	$(r) \leftarrow <B_2>$ Load byte two of the instruction into register r.
LMI (three cycles – PCI/PCR/PCW)	00 111 110 $<B_2>$	$(M) \leftarrow <B_2>$ Load byte two of the instruction into the memory location addressed by the contents of registers H and L.

INCREMENT INDEX REGISTER – One Byte

INr (one cycle – PCI)	00 DDD 000	$(r) \leftarrow (r)+1$. The content of register r is incremented by one. All of the condition flip-flops except carry are affected by the result. Note that DDD≠000 (HALT instr.) and DDD≠111 (content of memory may not be incremented).

DECREMENT INDEX REGISTER – One Byte

DCr (one cycle – PCI)	00 DDD 001	$(r) \leftarrow (r)-1$. The content of register r is decremented by one. All of the condition flip-flops except carry are affected by the result. Note that DDD≠000 (HALT instr.) and DDD≠111 (content of memory may not be decremented).

ACCUMULATOR GROUP INSTRUCTIONS

Operations are performed and the status flip-flops, C, Z, S, P, are set based on the result of the operation. Logical operations (NDr, XRr, ORr) set the carry flip-flop to zero. Rotate operations affect only the carry flip-flop. Two's complement subtraction is used.

ALU INDEX REGISTER INSTRUCTIONS – One Byte

(one cycle – PCI)

Index Register operations are carried out between the accumulator and the content of one of the index registers (SSS=000 thru SSS=110). The previous content of register SSS is unchanged by the operation.

ADr	10 000 SSS	$(A) \leftarrow (A)+(r)$ Add the content of register r to the content of register A and place the result into register A.
ACr	10 001 SSS	$(A) \leftarrow (A)+(r)+(carry)$ Add the content of register r and the contents of the carry flip-flop to the content of the A register and place the result into Register A.
SUr	10 010 SSS	$(A) \leftarrow (A)-(r)$ Subtract the content of register r from the content of register A and place the result into register A. Two's complement subtraction is used.

ACCUMULATOR GROUP INSTRUCTIONS - Cont'd.

SBr	10	011	SSS	$(A) \leftarrow (A) - (r) - (\text{borrow})$ Subtract the content of register r and the content of the carry flip-flop from the content of register A and place the result into register A.
NDr	10	100	SSS	$(A) \leftarrow (A) \wedge (r)$ Place the logical product of the register A and register r into register A.
XRr	10	101	SSS	$(A) \leftarrow (A) \forall (r)$ Place the "exclusive - or" of the content of register A and register r into register A.
ORr	10	110	SSS	$(A) \leftarrow (A) \vee (r)$ Place the "inclusive - or" of the content of register A and register r into register A.
CPr	10	111	SSS	$(A) - (r)$ Compare the content of register A with the content of register r. The content of register A remains unchanged. The flag flip-flops are set by the result of the subtraction. Equality or inequality is indicated by the zero flip-flop. Less than or greater than is indicated by the carry flip-flop.

ALU OPERATIONS WITH MEMORY – One Byte
(two cycles – PCI/PCR)
Arithmetic and logical operations are carried out between the accumulator and the byte of data addressed by the contents of registers H and L.

ADM	10	000	111	$(A) \leftarrow (A) + (M)$ ADD
ACM	10	001	111	$(A) \leftarrow (A) + (M) + (\text{carry})$ ADD with carry
SUM	10	010	111	$(A) \leftarrow (A) - (M)$ SUBTRACT
SBM	10	011	111	$(A) \leftarrow (A) - (M) - (\text{borrow})$ SUBTRACT with borrow
NDM	10	100	111	$(A) \leftarrow (A) \wedge (M)$ Logical AND
XRM	10	101	111	$(A) \leftarrow (A) \forall (M)$ Exclusive OR
ORM	10	110	111	$(A) \leftarrow (A) \vee (M)$ Inclusive OR
CPM	10	111	111	$(A) - (M)$ COMPARE

ALU IMMEDIATE INSTRUCTIONS – Two Bytes
(two cycles –PCI/PCR)
Arithmetic and logical operations are carried out between the accumulator and the byte of data immediately following the instruction.

ADI	00	000 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) + \langle B_2 \rangle$ ADD
ACI	00	001 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) + \langle B_2 \rangle + (\text{carry})$ ADD with carry
SUI	00	010 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) - \langle B_2 \rangle$ SUBTRACT
SBI	00	011 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) - \langle B_2 \rangle - (\text{borrow})$ SUBTRACT with borrow
NDI	00	100 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) \wedge \langle B_2 \rangle$ Logical AND
XRI	00	101 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) \forall \langle B_2 \rangle$ Exclusive OR
ORI	00	110 $\langle B_2 \rangle$	100	$(A) \leftarrow (A) \vee \langle B_2 \rangle$ Inclusive OR
CPI	00	111 $\langle B_2 \rangle$	100	$(A) - \langle B_2 \rangle$ COMPARE

ROTATE INSTRUCTIONS — One Byte

(one cycle — PCI)

The accumulator content (register A) may be rotated either right or left, around the carry bit or through the carry bit. Only the carry flip-flop is affected by these instructions; the other flags are unchanged.

RLC	00 000 010	$A_{m+1} \leftarrow A_m$, $A_0 \leftarrow A_7$, (carry) $\leftarrow A_7$ Rotate the content of register A left one bit. Rotate A_7 into A_0 and into the carry flip-flop.
RRC	00 001 010	$A_m \leftarrow A_{m+1}$, $A_7 \leftarrow A_0$, (carry) $\leftarrow A_0$ Rotate the content of register A right one bit. Rotate A_0 into A_7 and into the carry flip-flop.
RAL	00 010 010	$A_{m+1} \leftarrow A_m$, $A_0 \leftarrow$ (carry), (carry) $\leftarrow A_7$ Rotate the content of Register A left one bit. Rotate the content of the carry flip-flop into A_0. Rotate A_7 into the carry flip-flop.
RAR	00 011 010	$A_m \leftarrow A_{m+1}$, $A_7 \leftarrow$ (carry), (carry) $\leftarrow A_0$ Rotate the content of register A right one bit. Rotate the content of the carry flip-flop into A_7. Rotate A_0 into the carry flip-flop.

PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

JUMP INSTRUCTIONS — Three Bytes

(three cycles — PCI/PCR/PCR)

Normal flow of the microprogram may be altered by jumping to an address specified by bytes two and three of an instruction.

JMP (Jump Unconditionally)	01 XXX 100 $<B_2>$ $<B_3>$	$(P) \leftarrow <B_3><B_2>$ Jump unconditionally to the instruction located in memory location addressed by byte two and byte three.
JFc (Jump if Condition False)	01 $0C_4C_3$ 000 $<B_2>$ $<B_3>$	If (c) = 0, $(P) \leftarrow <B_3><B_2>$. Otherwise, (P) = (P)+3. If the content of flip-flop c is zero, then jump to the instruction located in memory location $<B_3><B_2>$; otherwise, execute the next instruction in sequence.
JTc (Jump if Condition True)	01 $1C_4C_3$ 000 $<B_2>$ $<B_3>$	If (c) = 1, $(P) \leftarrow <B_3><B_2>$. Otherwise, (P) = (P)+3. If the content of flip-flop c is one, then jump to the instruction located in memory location $<B_3><B_2>$; otherwise, execute the next instruction in sequence.

CALL INSTRUCTIONS — Three Bytes

(three cycles — PCI/PCR/PCR)

Subroutines may be called and nested up to seven levels.

CAL (Call subroutine Unconditionally)	01 XXX 110 $<B_2>$ $<B_3>$	(Stack) $\leftarrow$ (P), $(P) \leftarrow <B_3><B_2>$. Shift the content of P to the pushdown stack. Jump unconditionally to the instruction located in memory location addressed by byte two and byte three.
CFc (Call subroutine if Condition False)	01 $0C_4C_3$ 010 $<B_2>$ $<B_3>$	If (c) = 0, (Stack) $\leftarrow$ (P), $(P) \leftarrow <B_3><B_2>$. Otherwise, (P) = (P)+3. If the content of flip-flop c is zero, then shift contents of P to the pushdown stack and jump to the instruction located in memory location $<B_3><B_2>$; otherwise, execute the next instruction in sequence.
CTc (Call subroutine if Condition True)	01 $1C_4C_3$ 010 $<B_2>$ $<B_3>$	If (c) = 1, (Stack) $\leftarrow$ (P), $(P) \leftarrow <B_3><B_2>$. Otherwise, (P) = (P)+3. If the content of flip-flop c is one, then shift contents of P to the pushdown stack and jump to the instruction located in memory location $<B_3><B_2>$; otherwise, execute the next instruction in sequence.

In the above JUMP and CALL instructions $<B_2>$ contains the least significant half of the address and $<B_3>$ contains the most significant half of the address. Note that D_6 and D_7 of $<B_3>$ are "don't care" bits since the CPU uses fourteen bits of address.

RETURN INSTRUCTIONS — One Byte
(one cycle — PCI)

A return instruction may be used to exit from a subroutine; the stack is popped-up one level at a time.

RET	00 XXX 111	(P)←(Stack). Return to the instruction in the memory location addressed by the last value shifted into the pushdown stack. The stack pops up one level.
RFc (Return Condition False)	00 $0C_4C_3$ 011	If (c) = 0, (P)←(Stack); otherwise, (P) = (P)+1. If the content of flip-flop c is zero, then return to the instruction in the memory location addressed by the last value inserted in the pushdown stack. The stack pops up one level. Otherwise, execute the next instruction in sequence.
RTc (Return Condition True)	00 $1C_4C_3$ 011	If (c) = 1, (P)←(Stack); otherwise, (P) = (P)+1. If the content of flip-flop c is one, then return to the instruction in the memory location addressed by the last value inserted in the pushdown stack. The stack pops up one level. Otherwise, execute the next instruction in sequence.

RESTART INSTRUCTION — One Byte
(one cycle — PCI)

The restart instruction acts as a one byte call on eight specified locations of page 0, the first 256 instruction words.

RST	00 AAA 101	(Stack)←(P),(P)←(000000 00AAA000) Shift the contents of P to the pushdown stack. The content, AAA, of the instruction register is shifted into bits 3 through 5 of the P-counter. All other bits of the P-counter are set to zero. As a one-word "call", eight eight-byte subroutines may be accessed in the lower 64 words of memory.

INPUT/OUTPUT INSTRUCTIONS

One Byte
(two cycles — PCI/PCC)

Eight input devices may be referenced by the input instruction

INP	01 00M MM1	(A)←(input data lines). The content of register A is made available to external equipment at state T1 of the PCC cycle. The content of the instruction register is made available to external equipment at state T2 of the PCC cycle. New data for the accumulator is loaded at T3 of the PCC cycle. MMM denotes input device number. The content of the condition flip-flops, S,Z,P,C, is output on D_0, D_1, D_2, D_3 respectively at T4 on the PCC cycle.

Twenty-four output devices may be referenced by the output instruction.

OUT	01 RRM MM1	(Output data lines)←(A). The content of register A is made available to external equipment at state T1 and the content of the instruction register is made available to external equipment at state T2 of the PCC cycle. RRMMM denotes output device number (RR $\neq$ 00).

MACHINE INSTRUCTION

HALT INSTRUCTION — One Byte
(one cycle — PCI)

HLT	00 000 00X or 11 111 111	On receipt of the Halt Instruction, the activity of the processor is immediately suspended in the STOPPED state. The content of all registers and memory is unchanged. The P-counter has been updated and the internal dynamic memories continue to be refreshed.

APPENDIX II

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 surpress incrementing of program counter.
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. 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.

The following timing diagram shows the typical activity during each state. Note that ϕ_1 is generally used to precharge all data lines and memories and ϕ_2 controls all data transfers within the processor.

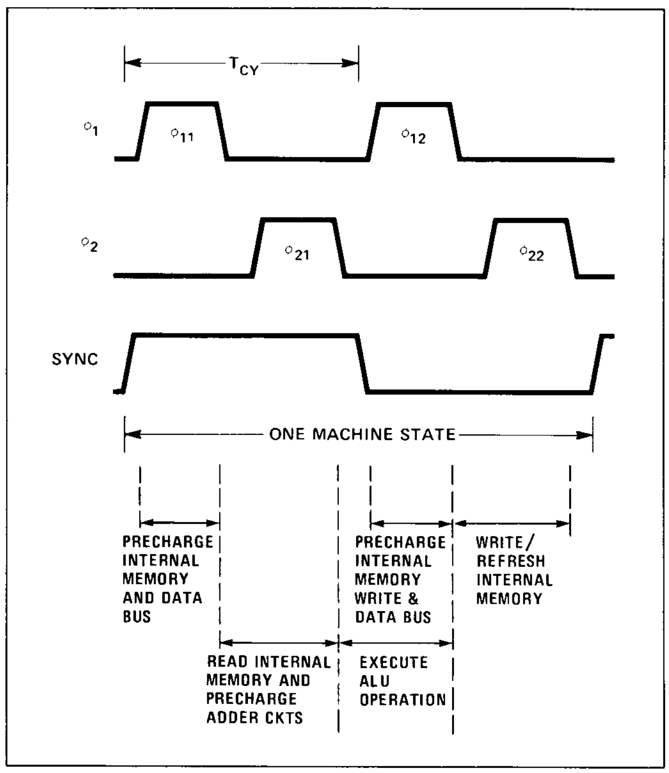


Figure 19. Internal Timing Activity Within Any State

INDEX REGISTER INSTRUCTIONS

INSTRUCTION CODING D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES TO EXECUTE INSTRUCTION	MEMORY CYCLE ONE (1) T1 (2)	T2	T3	T4 (3)	T5
1 1	D D D	S S S	Lr_1r_2	(5)	PC_LOUT (4)	PC_HOUT	FETCH INSTR.(5) TO IR & REG. b	SSS TO REG. b (6)	REG. b TO DDD
1 1	D D D	1 1 1	LrM	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→ (7)	→
1 1	1 1 1	S S S	LMr	(7)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	→
0 0	D D D	1 1 0	LrI	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	1 1 1	1 1 0	LMI	(9)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	D D D	0 0 0	INr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ADD OP - FLAGS AFFECTED
0 0	D D D	0 0 1	DCr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	SUB OP - FLAGS AFFECTED

ACCUMULATOR GROUP INSTRUCTIONS

D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES	T1	T2	T3	T4	T5
1 0	P P P	S S S	ALU OP r	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	ALU OP - FLAGS AFFECTED
1 0	P P P	1 1 1	ALU OP M	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	P P P	1 0 0	ALU OP I	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	0 0 0	0 1 0	RLC	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 0 1	0 1 0	RRC	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 1 0	0 1 0	RAL	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED
0 0	0 1 1	0 1 0	RAR	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED

PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES	T1	T2	T3	T4	T5
0 1	X X X	1 0 0	JMP	(11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	0 C C	0 0 0	JFc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	1 C C	0 0 0	JTc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	X X X	1 1 0	CAL	(11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	0 C C	0 1 0	CFc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	1 C C	0 1 0	CTc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 0	X X X	1 1 1	RET	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK	X
0 0	0 C C	0 1 1	RFc	(3 or 5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X
0 0	1 C C	0 1 1	RTc	(3 or 5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X
0 0	A A A	1 0 1	RST	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO REG. b AND PUSH STACK (0→REG. a)	REG. a TO PC_H	REG. b TO PC_L (14)

I/O INSTRUCTIONS

D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES	T1	T2	T3	T4	T5
0 1	0 0 M	M M 1	INP	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→
0 1	R R M	M M 1	OUT	(6)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→

MACHINE INSTRUCTIONS

D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$	OPERATION	# OF STATES	T1	T2	T3	T4	T5
0 0	0 0 0	0 0 X	HLT	(4)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b & HALT (18)		
1 1	1 1 1	1 1 1	HLT	(4)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b & HALT (18)		

NOTES:

1. The first memory cycle is always a PCI (instruction) cycle.
2. Internally, states are defined as T1 through T5. In some cases more than one memory 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.

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

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.
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.

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.

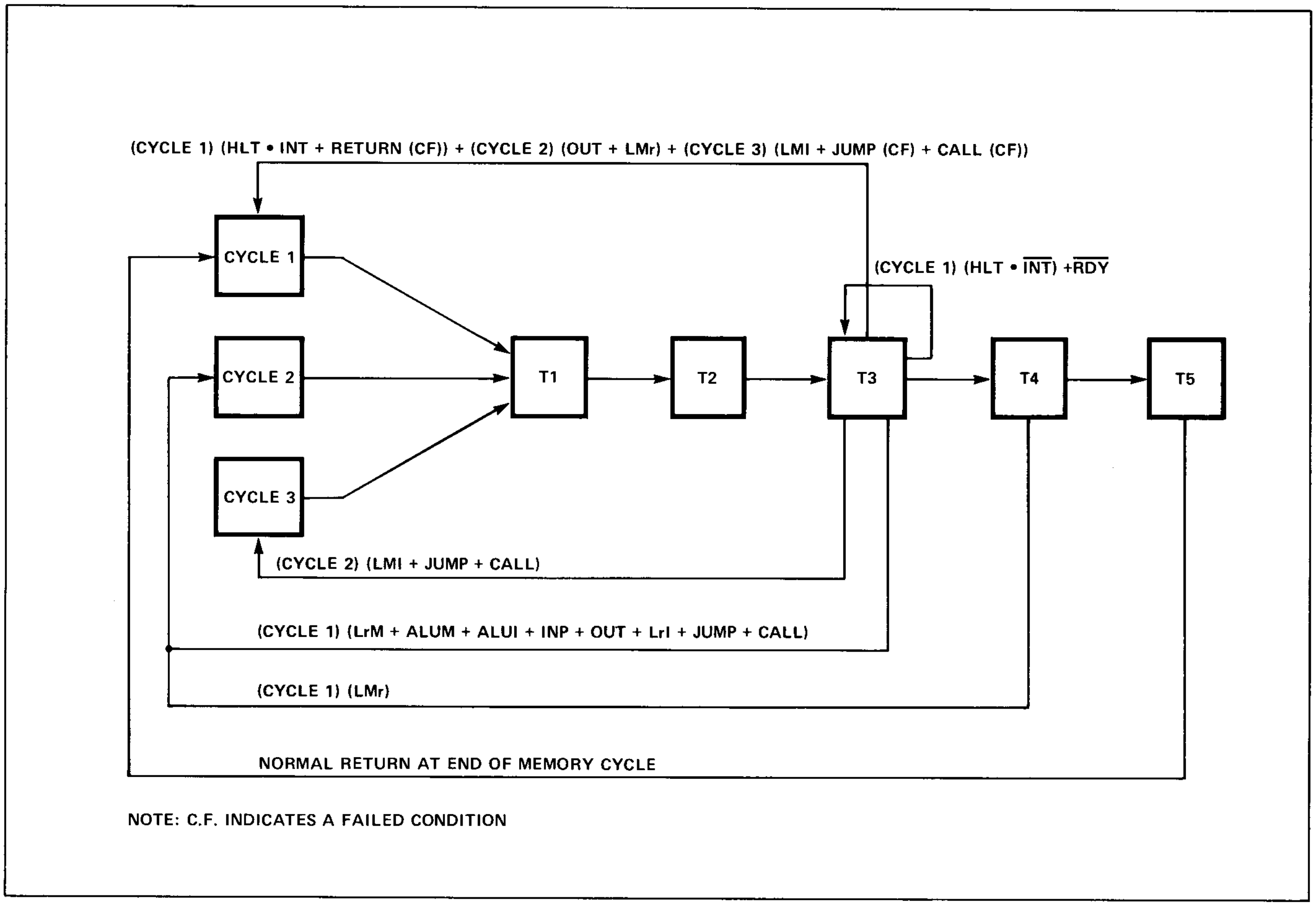


Figure 20. Transition State Diagram (Internal)

APPENDIX III

PROGRAMMING EXAMPLES

A. Sample Program to Search A String Of Characters In Memory Locations 200-219 For A Period (.)

MNEMONIC	OPERAND	EXPLANATION	BYTES	LOCATION	ROM CODE	COMMENT
Start: LLI	200	Load L with 200	2	100 101	00110110 11001000	(200)
LHI	0	Load H with 0	2	102 103	00101110 00000000	(0)
Loop: LAM		Fetch Character from Memory	1	104	11000111	ASC II
CPI	"."	Compare it with period	2	105 106	00111100 00101110	ASC II (•)
JTZ	Found	If equal go to return	3	107 108 109	01101000 01110111 00000000	(119)
CAL	INCR	Call increment H&L subroutine	3	110 111 112	01000110 00111100 00000000	(60)
LAL		Load L to A	1	113	11000110	
CPI	220	Compare it with 220	2	114 115	00111100 11011100	(220)
JFZ	Loop	If unequal go to loop	3	116 117 118	01001000 01101000 00000000	(104)
Found: RET		Return	1	119	00000111	
INCR: INL		Increment L	1	60	00110000	
RFZ		Return if not zero	1	61	00001011	
INH		Increment H	1	62	00101000	
RET		Return	1	63	00000111	

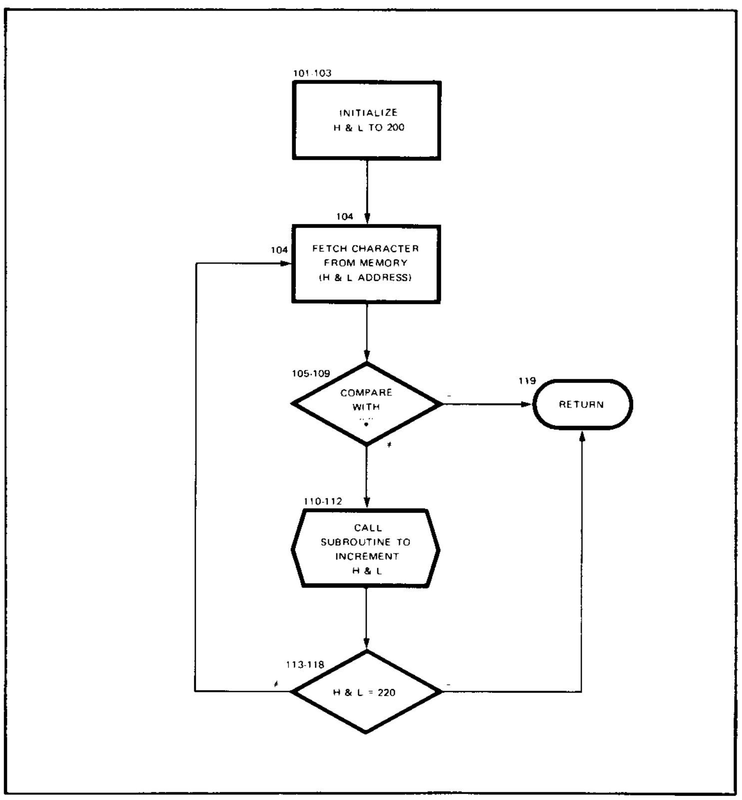


Figure 21. Subroutine to Search For Period

B. Teletype and Tape Reader Control Program (A0800)

```
BEGIN   LAI 1         SUPPRESS TTY
        OUT 12B       OUTPUT 2
        XRA           CLEAR AC
        OUT 13B       OUTPUT 3 - TAPE READER CONTROL
        CAL TAPE      CALL FOR TAPE READER CONT. RT.
        JMP BEGIN
TAPE    LAI 1         TAPE READER ENABLE CODE
        OUT 13B       OUTPUT 3 - ENABLE TAPE READER
        CAL TTYD1     TAPE READER CONTROL DELAY
TTY     HLT           WAIT FOR TTY START PULSE
        CAL TTYD2     TTY DELAY - 4.468 MSEC.
        XRA           TAPE READER DISABLE CODE
        OUT 13B       OUTPUT 3, DISABLE TAPE READER
        INP 0B        INPUT 0, READ START PULSE
        LCI 255       COMPLEMENT TTY START PULSE
        XRC           EXCLUSIVE-OR REG. C
        OUT 12B       OUTPUT 2, OUTPUT START PULSE
        LEI 248       TTY DATA SAMPLING COUNTER
TTYIN   CAL TTYD1     TTY DELAY - 9.012 MSEC.
        INP 0B        READ TTY DATA INPUT
        LCI 255       COMPLEMENT TTY DATA
        XRC
        OUT 12B       OUTPUT 2, TTY DATA OUT
        RAR           STORE TTY DATA
        LAB           LOAD TTY DATA TO REG. B
        RAR
        LBA           LOAD AC TO REG. B
        INE           E = E + 1
        JFZ TTYIN     JUMP IF ZERO F/F IS NOT SET
        LAB           LOAD REG. B TO AC
        OUT 11B       OUTPUT 1, TTY CHARACTER
        SUI 128       REMOVE PARITY BIT
        LBA           STORE TTY INPUT DATA
        CAL TTYD1
        LAI 1
        OUT 12B       SUPPRESS TTY
        RET
TTYD1   LDI 115       9.012 MSEC. DELAY
ST      IND           D = D + 1
        JFZ ST
        RET
TTYD2   LDI 186       4.468 MSEC. DELAY

ST2     IND           D = D + 1
        JFZ ST2
        RET
        END
```

C. Memory Chip Select Decodes and Output Test Program (A0801)

```
BEGIN   LAI 15                 LOAD 15 TO AC
        OUT 10B                WRITE TO OUTPUT 0
        OUT 11B
        OUT 12B
        OUT 13B
        OUT 14B
        OUT 15B
        OUT 16B
        OUT 17B
        CAL DELAY              DELAY 16.436 MSEC.
        CAL DELAY
        CAL DELAY
        CAL DELAY
        XRA                    CLEAR AC
        OUT 10B
        OUT 11B
        OUT 12B
        OUT 13B
        OUT 14B
        OUT 15B
        OUT 16B
        OUT 17B
        LCI 240                LOAD 240 TO REG. C
        LLI 252B               LOAD 252B(OCTAL) TO REG. C
        LHI 0                  LOAD 0 TO REG. H
CSTEST  LAH                    LOAD H TO AC
        OUT 10B
        LAL                    LOAD L TO AC
        OUT 11B
        XRA                    CLEAR AC
        LMA                    WRITE AC TO MEMORY
        CAL DELAY
        CAL DELAY
        INH                    H = H + 1
        INC                    C = C +1
        JFZ CSTEST
        JMP BEGIN
DELAY   LDI 0                  LOAD 0 TO REG. D
D1      IND                    D = D + 1
        JFZ D1
        RET
        END
```

D. RAM Test Program (A0802)

```
BEGIN   LAI 0             LOAD 0 TO AC
        OUT 10B           WRITE TO OUTPUT 0
        OUT 11B           WRITE TO OUTPUT 1
        OUT 12B           WRITE TO OUTPUT 2
        OUT 13B           WRITE TO OUTPUT 3
        LBI 8             LOAD 8 TO REC. B
        LCI 0             LOAD 0 TO REC. C
        LHI 8             LOAD 8 TO REC H
        LLI 0             LOAD 0 TO REC. L
LM1     XRA               CLEAR AC
LM2     LMA               LOAD AC TO MEMORY
        INL               L = L + 1
        CPL               AC - L
        JFZ LM2           JUMP IF AC IS NOT ZERO
        INH               H = H + 1
        LAI 12            LOAD 12 TO AC
        CPH               AC-H
        JFZ LM1           JUMP IF AC IS NOT ZERO
        LHI 8
REPT4   LAB               LOAD REC. B TO AC
        OUT 10B
REPT3   LLC               LOAD REC. C TO L
        LAC               LOAD REC. C TO AC
        OUT 13B
        LAI 255           LOAD 255 TO AC
        LMA               LOAD AC TO MEMORY
        CPM               AC-M
        JFZ ERROR         JUMP IF AC IS NOT ZERO
REPT2   LAH               LOAD REC. H TO AC
        OUT 10B
REPT5   XRA               CLEAR AC
        INL               L = L + 1
        CPL               AC - L
        JTZ REPT1         JUMP IF AC=0
        LAL               LOAD REC. L TO AC
        OUT 11B
        XRA               CLEAR AC
        CPM               AC-M
        JFZ ERROR         JUMP IF AC IS NOT ZERO
        JMP REPT5
REPT1   INH               H = H + 1
        LAI 12
        CPH
        JTZ CONT
        XRA
        CPM
        JFZ ERROR
        JMP REPT2
CONT    LHB               LOAD REC. B TO H
        XRA
        INC               C = C + 1
        CPC               AC - C
        JFZ REPT3
        INB               B = B + 1
        LHB               LOAD REC. B TO H
        LAI 12
        CPB               AC-B
        JFZ REPT4
        JMP BEGIN
ERROR   LAI 240           LOAD 240 TO AC
        ADB               AC=AC+B
        OUT 10B
        LAL               LOAD REC. L TO AC
        OUT 11B
        LAM               LOAD MEMORY TO AC
        OUT 12B
        LAC               LOAD REC. C TO AC
        OUT 13B
        HLT
        END
```

APPENDIX IV
INTEL DEVICE SPECIFICATIONS

A. 1101A/1101A1 256-Bit Fully-Decoded Random Access Memory

- **Access time below 750 ns typically, 1.0 μsec maximum — 1101A1; 1.5 μsec maximum — 1101A: over temperature**
- **Low power dissipation—typically less than 1.5 mW/bit during access**
- **Low power standby mode**
- **Directly DTL and TTL compatible**
- **OR-Tie capability**
- **Simple memory expansion—chip-select input lead**
- **Fully decoded—on-chip address decode and sense**
- **Inputs protected against static charge**
- **Ceramic and plastic package**
- **Silicon gate MOS technology**

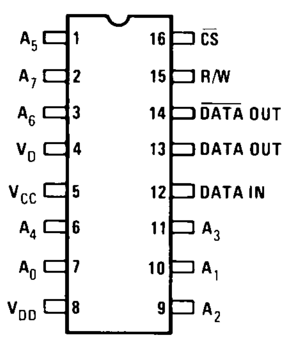
The Intel 1101A Series are 256 word by 1 bit random access memory elements using normally off P-channel MOS devices integrated on a monolithic array. Each unit uses fully dc stable (static) circuitry and therefore requires no clocks to operate.

The 1101A, an improved version of the 1101, requires only two power supplies (+5V and −9V) for operation, and is a direct pin-for-pin replacement for the 1101.

The 1101A Series is designed primarily for small buffer storage applications where high performance, low cost, and ease of interfacing with other standard logic circuits are important design objectives. The unit will directly interface with standard bipolar integrated logic circuits (TTL, DTL, etc.). The data output buffers are capable of driving TTL loads directly. A separate chip select (CS) lead allows easy selection of an individual package when outputs are OR-tied.

For applications requiring a faster access time we recommend the 1101A1 which is a selection from the 1101A and has a guaranteed maximum access time of 1.0 μsec.

Applications. Scratch pad memories, buffer storage, data terminals, minicomputers, calculators, data multiplexers, automatic test equipment, sequential memories, table look up, program memories, buffers for line printers, card readers and pulse height analyzers.



D.C. Characteristics

(T_A = 0°C to +70°C, V_{CC} = +5V ±5%, V_D = −9V ±5%, V_{DD} = −9V ±5% unless otherwise specified)

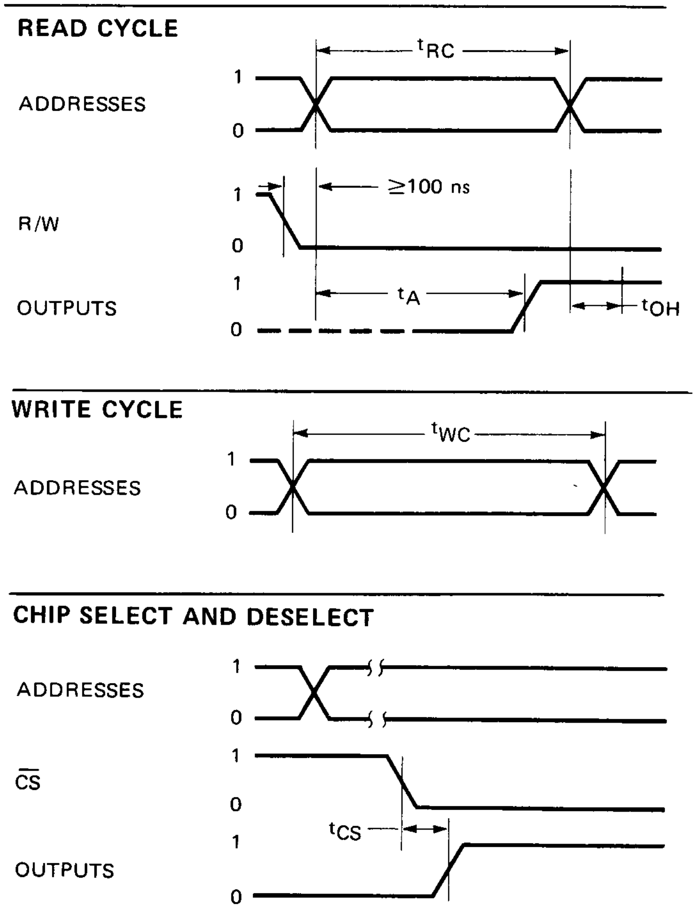
Test	Min.	Typ.	Max.	Unit
Input load current (All Input Pins)		1.0	500	nA
Power supply current, V_{DD} @ 25°C		13	19	mA
Power supply current, V_D @ 25°C			18	mA
Input "low" voltage	−10		V_{CC} −4.5	V
Input "high" voltage	V_{CC} −2		V_{CC} +0.3	V
Output "low" voltage (I_{OL} = 2 mA)			+.45	V
Output "high" voltage	+3.5			V

A.C. Characteristics

(T_A = 0 to 70°C, V_{CC} = +5V ±5%, V_D = −9V ±5%, V_{DD} = −9 ±5%)

Test		Min.	Typ.	Max.	Unit
Write cycle (t_{WC})		0.8			μsec
Read cycle access time (t_A)	1101A			1.5	μsec
	1101A1			1.0	μsec
Access Time Through Chip Select Input (t_{CS})				0.3	μsec

Timing diagram



B. 1601/1701, 1602/1702 Electrically Programmable ROMs

The Intel 1601, 1602, 1701, and 1702 is a 256 word by 8 bit electrically programmable ROM ideally suited for uses where fast turnaround and pattern experimentation are important such as in prototype or in one of a kind systems. The 1601, 1602, 1701, and 1702 is factory reprogrammable which allows Intel to perform a complete programming and functional test on each bit position before delivery.

The four devices 1601, 1602, 1701, and 1702 use identical chips. The 1601 and 1701 is operable in both the static and dynamic mode while the 1602 and 1702 is operable in the static mode only. Also, the 1701 and 1702 has the unique feature of being completely erasable and field reprogrammable. This is accomplished by a quartz lid that allows high intensity ultraviolet light to erase the 1701 and 1702. A new pattern can then be written into the device. This procedure can be repeated as many times as required.

The 1301 is a direct replacement part which is programmed by a metal mask and is ideal for large volume and lower cost production runs of systems initially using the 1601/1701 or the static only 1602/1702.

The dynamic mode of the 1601/1701 and 1301 refers to the decoding circuitry and not to the memory cell. Dynamic operation offers higher speed and lower power dissipation than the static operation.

The 1601, 1602, 1701, and 1702 is fabricated with silicon gate technology. This low threshold technology allows the design and production of higher performance MOS circuits and provides a higher functional density on a monolithic chip than conventional MOS technologies.

D.C. Characteristics for static operation

(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5%)

Test	Min.	Max.	Unit
Standby power supply current @ 25°C		10	μA
Power supply current @ 25°C under continuous operation		46	mA
Input load current		1	μA
Input "high" voltage	V_{CC} −2	V_{CC} +.3	V
Input "low" voltage	V_{CC} −10	V_{CC} −4.2	V
Output "low" voltage @ I_{OL} = 1.6 mA		0.45	V
Output "high" voltage @ I_{OH} = −100 μA	3.5		V

A.C. Characteristics for static operation

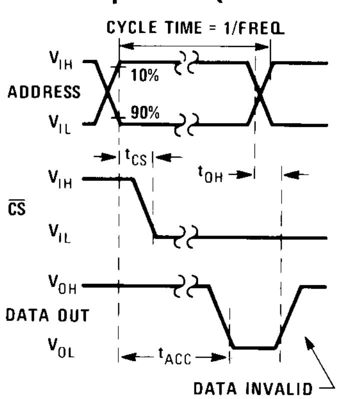
(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5% unless otherwise noted)

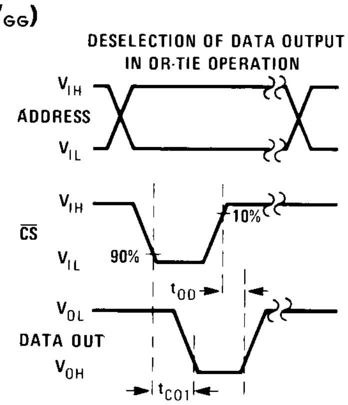
	1601/1701 1602/1702		1301		
Test	Typ.	Max.	Typ.	Max.	Unit
Repetition Rate		1		1	MHz
Address to output delay	.700	1	.550	1	μs

Switching Characteristics for Static Operation

Conditions of:
Input pulse amplitudes: 0 to 4V
Output load is 1 TTL gate

Normal Operation (constant V_{GG})





D.C. Characteristics for dynamic operation

(T_A = 0°C to 70°C, V_{CC} = V_{GG} = +5V ±5%, V_{DD} = −9V ±5%, unless otherwise noted)

Test		Typ.	Max.	Unit
Unselected average power supply current at 25°C	(φ = CS = +5V)	5	10	mA
Average power supply current @ T_A = 25°C	1601/1701	30	45	mA
	1301	28	40	mA

A.C. Characteristics for dynamic operation

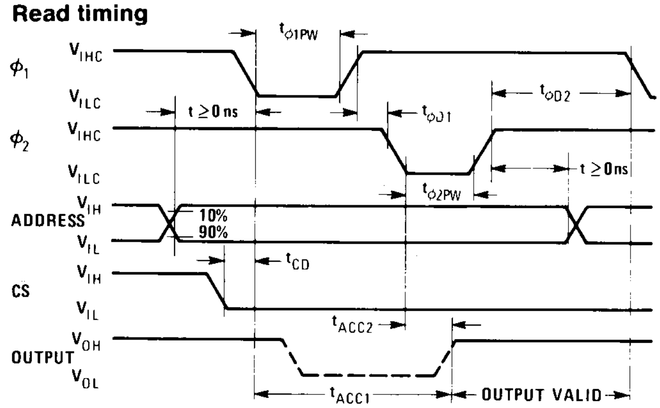
(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, unless otherwise noted)

	1601/1701			1301			
Test	Min.	Typ.	Max.	Min.	Typ.	Max.	Unit
Ø1 Clock pulse width	0.260		2	.260		2	μs
Ø2 Clock pulse width	0.140		2	.140		2	μs
Ø2 delay from Ø1	0.150		2	.150		2	μs
Ø1 delay from Ø2	.05		2	.05		2	μs
Address to output access		450	650		450	650	ns

Switching Characteristics for Dynamic Operation

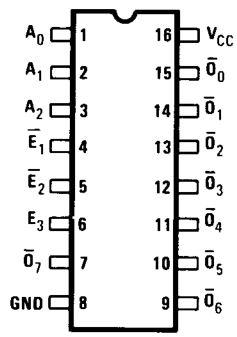
Condition of test:
Input pulse amplitude: 0 to 4V
Input rise and fall times ≤ 50 nsec
Output load is 1 TTL gate; measurements made at output of TTL gate (tpd ≤ 15 nsec)

Read timing



C. 3205 High-Speed 1-Out-Of-8 Binary Decoder

- **Easy memory expansion — 3 enable inputs**
- **Fast — 18 nsec max. delay**
- **TTL/DTL compatible**
- **Low input load currents — 0.25 mA max.**
- **High fan out — 10 mA min. sink current**
- **16 pin ceramic or plastic package**



The 3205 is a high-speed 1-of-8 binary decoder designed for use with fast bipolar memory components such as the Intel 3101A and 3102.

Applications. General purpose high-speed 1-of-8 decode, bipolar memory expansion, chip-select decoder.

A.C. Characteristics

(T_A = 0°C to 75°C, V_{CC} = 5.0V ±5%)

Test	Max.
Address or enable to output delay	18 nsec
Input capacitance	4 pF (typical)

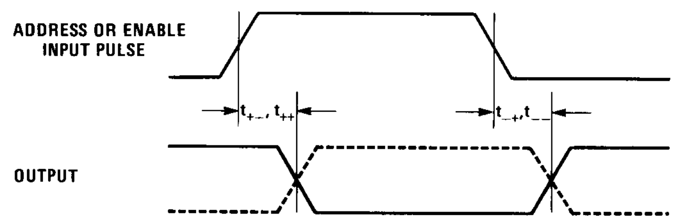
D.C. Characteristics

(T_A = 0°C to 75°C, V_{CC} = 5.0V ±5%)

Test	Min.	Max.	Unit
Input load current		−0.25	mA
Input Leakage current		10	μA
Output "low" voltage (I_{OL}=10 mA)		0.45	V
Output "high" voltage	2.4		V
Input "low" voltage		0.85	V
Input "high" voltage	2.0		V
Power supply current		75	mA

Timing diagram

Condition of test:
Input pulse amplitudes: 2.5V
Input rise and fall times: 5 nsec between 1V and 2V
Measurements are made at 1.5V

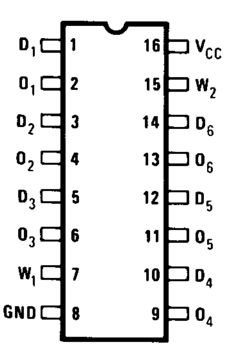


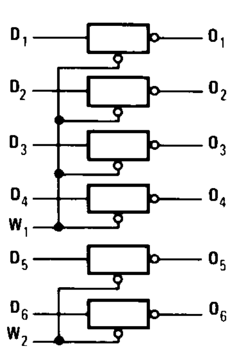
D. 3404 High-Speed 6-Bit Latch

- **Memory register applications**
- **High speed — 12 nsec max. input to output delay**
- **Easy to use — independent 4-bit and 2-bit latches**
- **High fan out — 10 mA min. output sink current**
- **High fan in — 0.25 mA max. input load current**
- **TTL/DTL compatible**
- **Standard 16-pin DIP**

The 3404 is a high-speed 6-bit latch built using Schottky bipolar technology. The circuitry consists of a 4-bit latch and an independent 2-bit latch on the same chip. The latches may be used as very high-speed inverters by applying a continuous "low" at the "write" inputs.

Applications. Memory data register, address register, inverter, high fan out buffers, "pipelined" processors.





A.C. Characteristics

(T_A = 0 to 75°C, V_{CC} = 5.0V ±5%)

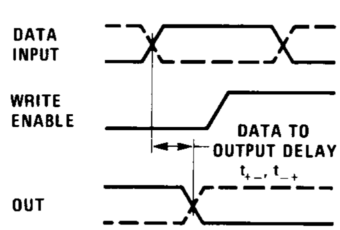
Test	Min.	Max.	Unit
Data to output delay		12	nsec.
Write enable to output delay		17	nsec.
Write enable pulse width (t_{WP})	15		

D.C. Characteristics

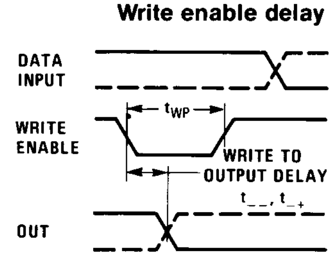
(T_A = 0°C to 75°C, V_{CC} = 5.0V ±5%)

Test	Min.	Max.	Unit
Data input load current		−0.25	mA
Input leakage current		10	μA
Output "low" voltage (I_{OL}=10 mA)		0.45	V
Output "high" voltage	2.4		V

Data delay



Write enable delay



Ordering Information

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

(2) **SIM8-01 Prototyping System**

This MCS-8 system for program development provides complete interface between the CPU and ROMs and RAMs. 1702 electrically programmable and erasable ROMs may be used for the program development. Each board contains one 8008 CPU, 1K x 8 RAM, and sockets for up to eight 1702's (2K x 8 PROM). This system should be ordered as SIM8-01 (the number of PROMs should also be specified).

(3) **Memory Expansion**

Additional memory for the 8008 may be developed from individual memory components. Specify

RAM	ROM	SR
RAM 1101	ROM 1702	SR 1404
1103	1301	2401
2102		

(4) **MP7-03 ROM Programmer**

This is the programmer board for 1702 or 1702A. The 1702 control ROMs used with the SIM8-01 for an automatic programming system are specified by pattern numbers A0860, A0861, A0862. Pattern A0863 should also be specified for programming the 1702A.

(5) **MCB8-10 System Interface and Control Module**

The MCB8-10 is a complete chassis which provides the interconnection between the SIM8-01 and MP7-03. In addition, the MCB8-10 provides the 50 Vrms power supply for PROM programming, complete output display, and single step control capability for program development.

(6) **Bootstrap Loader**

The same control ROM set used with the PROM programming system is used for the bootstrap loading of programs into RAM and execution of programs from RAM. Specify 1702 PROMs programmed to tapes A0860, A0861, A0862.

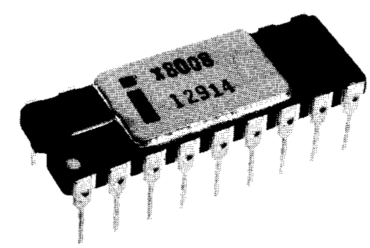
(7) **MCS-8 Fortran Assembler and Fortran Simulator**

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. Contact Intel for all the details.

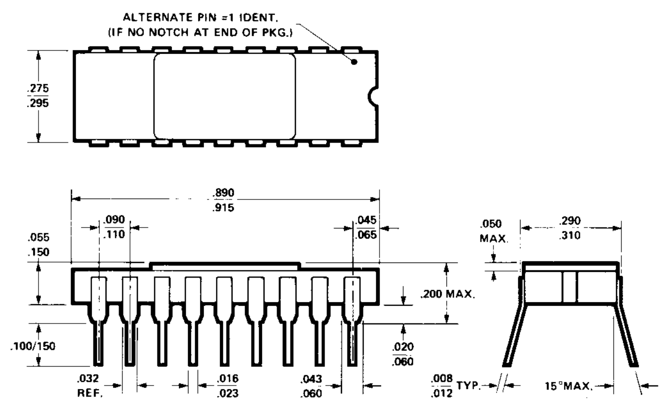
(8) **SIM8 Hardware Assembler**

Eight PROMs containing the assembly program plug into the SIM8-01 prototyping board permitting assembly of all MCS-8 software. To order, specify A0840 through A0847. The complete hardware assembler description and users' guide is available on request.

Packaging Information



CERAMIC PACKAGE OUTLINE



U. S. REGIONAL SALES OFFICES

Western	Tustin, California 92680	17401 Irving Blvd., Suite K (714) 838-1126, TWX 910-595-1114
Central	Bloomington, Minnesota 55437	800 Southgate Office Plaza, 500 West 78th St. (612) 835-6722
Eastern	Lexington, Mass. 02173	594 Marrett Road, Suite 27 (617) 861-1136, Telex: 923493

ORIENT SALES OFFICE

Japan Tokyo 160

INTEL JAPAN CORP.
Han-Ei 2nd Bldg.
1-1, Shinjuku,
Shinjuku-Ku
Tokyo 160, Japan
03-354-8251
Telex: 28426

EUROPEAN SALES OFFICE

Belgium Bruxelles

INTEL CORP.
216 Avenue Louise
Bruxelles 1050, Belgium
Phone: 492003
Telex: 21060

MCS-8 INSTRUCTION SET

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) Lr_1r_2	(5)	1 1	D D D	S S S	Load index register r_1 with the content of index register r_2.
(2) LrM	(8)	1 1	D D D	1 1 1	Load index register r with the content of memory register M.
LMr	(7)	1 1	1 1 1	S S S	Load memory register M with the content of index register r.
(3) LrI	(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.
LMI	(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	(5)	0 0	D D D	0 0 0	Increment the content of index register r (r ≠ A).
DCr	(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.

MNEMONIC	MINIMUM STATES REQUIRED	$D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
ADr	(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.
ADM	(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	
ACr	(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.
ACM	(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	
SUr	(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.
SUM	(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	
SBr	(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.
SBM	(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	
NDr	(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.
NDM	(8)	1 0	1 0 0	1 1 1	
NDI	(8)	0 0 B B	1 0 0 B B B	1 0 0 B B B	
XRr	(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.
XRM	(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	
ORr	(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.
ORM	(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	
CPr	(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.
CPM	(8)	1 0	1 1 1	1 1 1	
CPI	(8)	0 0 B B	1 1 1 B B B	1 0 0 B B B	
RLC	(5)	0 0	0 0 0	0 1 0	Rotate the content of the accumulator left.
RRC	(5)	0 0	0 0 1	0 1 0	Rotate the content of the accumulator right.
RAL	(5)	0 0	0 1 0	0 1 0	Rotate the content of the accumulator left through the carry.
RAR	(5)	0 0	0 1 1	0 1 0	Rotate the content of the accumulator right through the carry.

Program Counter and Stack Control Instructions

MNEMONIC	MINIMUM STATES REQUIRED	$D_7 D_6$	$D_5 D_4 D_3$	$D_2 D_1 D_0$	DESCRIPTION OF OPERATION
(4) JMP	(11)	0 1 $B_2 B_2$ X X	X X X $B_2 B_2 B_2$ $B_3 B_3 B_3$	1 0 0 $B_2 B_2 B_2$ $B_3 B_3 B_3$	Unconditionally jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$.
(5) JFc	(9 or 11)	0 1 $B_2 B_2$ X X	0 $C_4 C_3$ $B_2 B_2 B_2$ $B_3 B_3 B_3$	0 0 0 $B_2 B_2 B_2$ $B_3 B_3 B_3$	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
JTc	(9 or 11)	0 1 $B_2 B_2$ X X	1 $C_4 C_3$ $B_2 B_2 B_2$ $B_3 B_3 B_3$	0 0 0 $B_2 B_2 B_2$ $B_3 B_3 B_3$	Jump to memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
CAL	(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).
CFc	(9 or 11)	0 1 $B_2 B_2$ X X	0 $C_4 C_3$ $B_2 B_2 B_2$ $B_3 B_3 B_3$	0 1 0 $B_2 B_2 B_2$ $B_3 B_3 B_3$	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is false, and save the current address (up one level in the stack.) Otherwise, execute the next instruction in sequence.
CTc	(9 or 11)	0 1 $B_2 B_2$ X X	1 $C_4 C_3$ $B_2 B_2 B_2$ $B_3 B_3 B_3$	0 1 0 $B_2 B_2 B_2$ $B_3 B_3 B_3$	Call the subroutine at memory address $B_3 \ldots B_3 B_2 \ldots B_2$ if the condition flip-flop c is true, and save the current address (up one level in the stack). Otherwise, execute the next instruction in sequence.
RET	(5)	0 0	X X X	1 1 1	Unconditionally return (down one level in the stack).
RFc	(3 or 5)	0 0	0 $C_4 C_3$	0 1 1	Return (down one level in the stack) if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
RTc	(3 or 5)	0 0	1 $C_4 C_3$	0 1 1	Return (down one level in the stack) if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
RST	(5)	0 0	A A A	1 0 1	Call the subroutine at memory address AAA000 (up one level in the stack).

Input/Output Instructions

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

Machine Instruction

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

NOTES:

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

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

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

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

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

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

Printed in U.S.A.