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

8008
8-BIT PARALLEL CENTRAL PROCESSOR UNIT

MCS-8™
MICRO COMPUTER SET

APRIL 1972

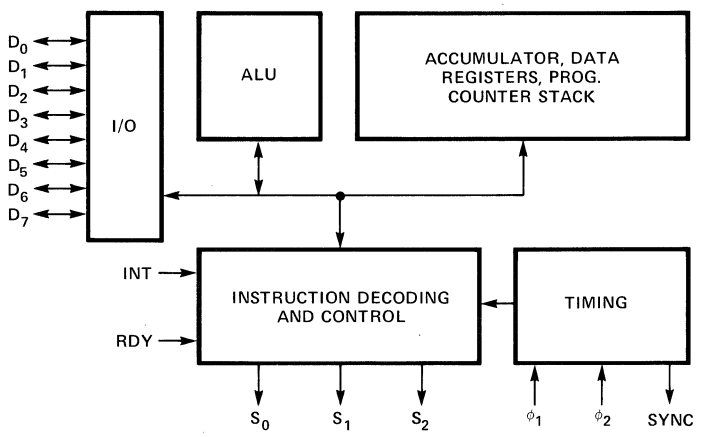
8008
8 Bit Parallel Central Processor Unit

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.

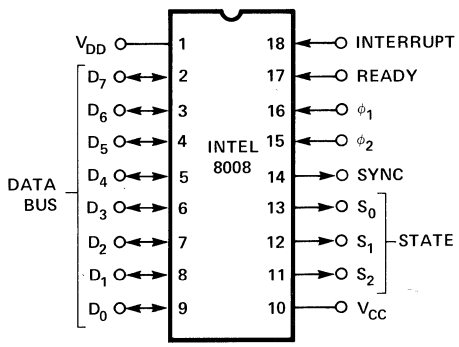
Features

- 8-Bit Parallel CPU on a single chip
- 48 instructions, data oriented
- Complete instruction decoding & control included
- TTL compatible (inputs, outputs & 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 data registers
- Interrupt capability
- Packaged in 18 pin DIP

BLOCK DIAGRAM

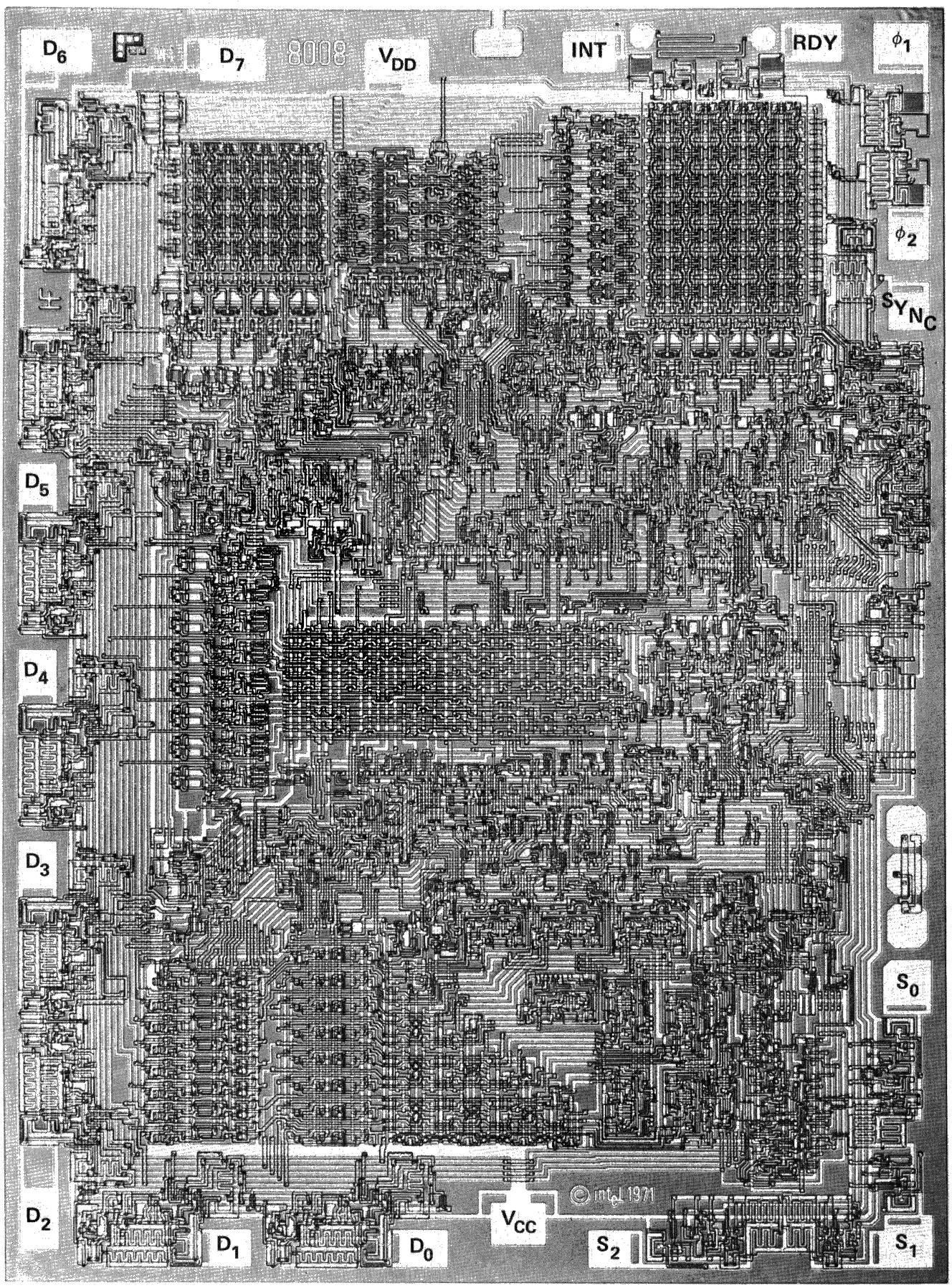


PIN CONFIGURATION



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 (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 (READ 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.

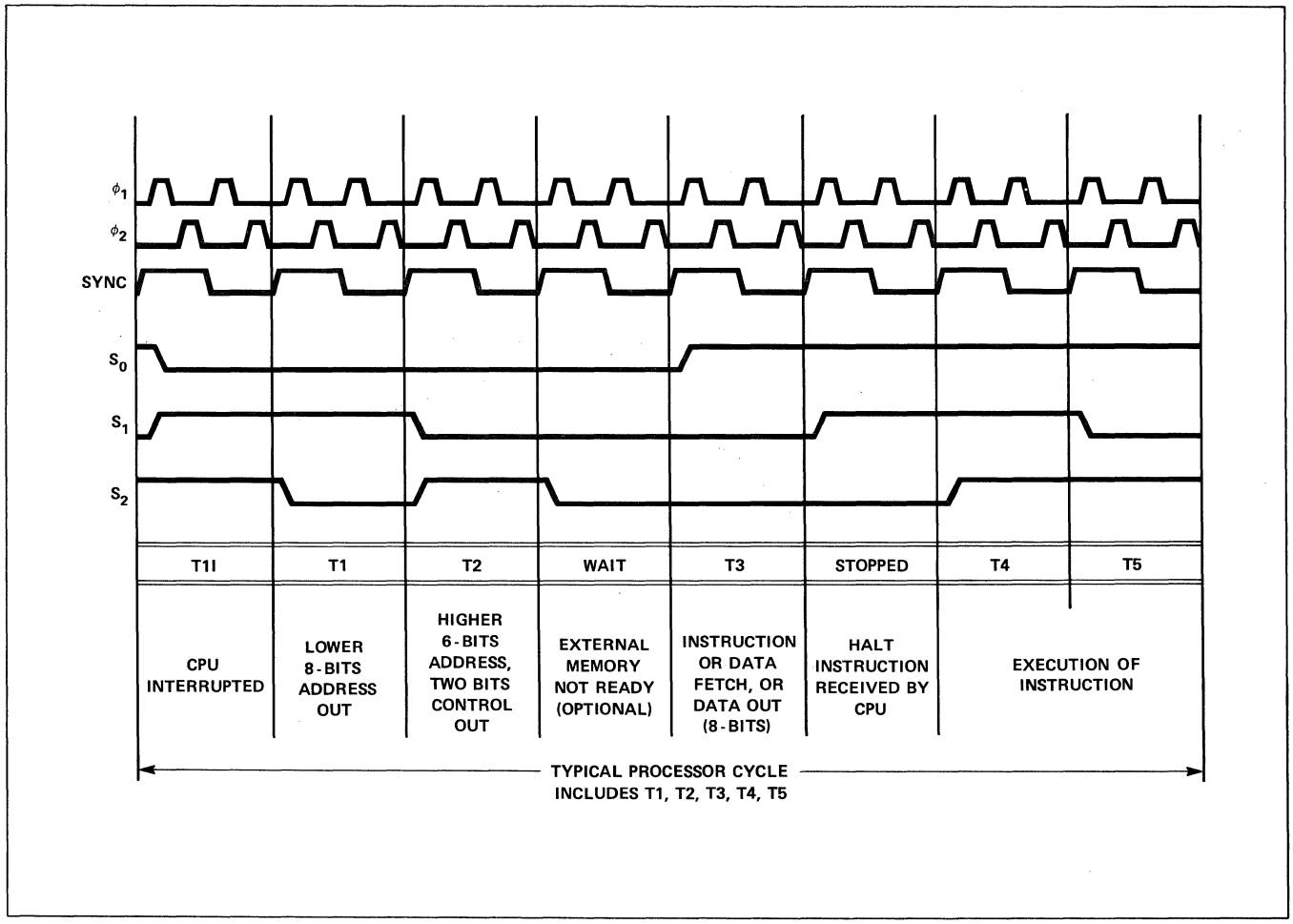


Figure 1. Basic 8008 Instruction Cycle

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.

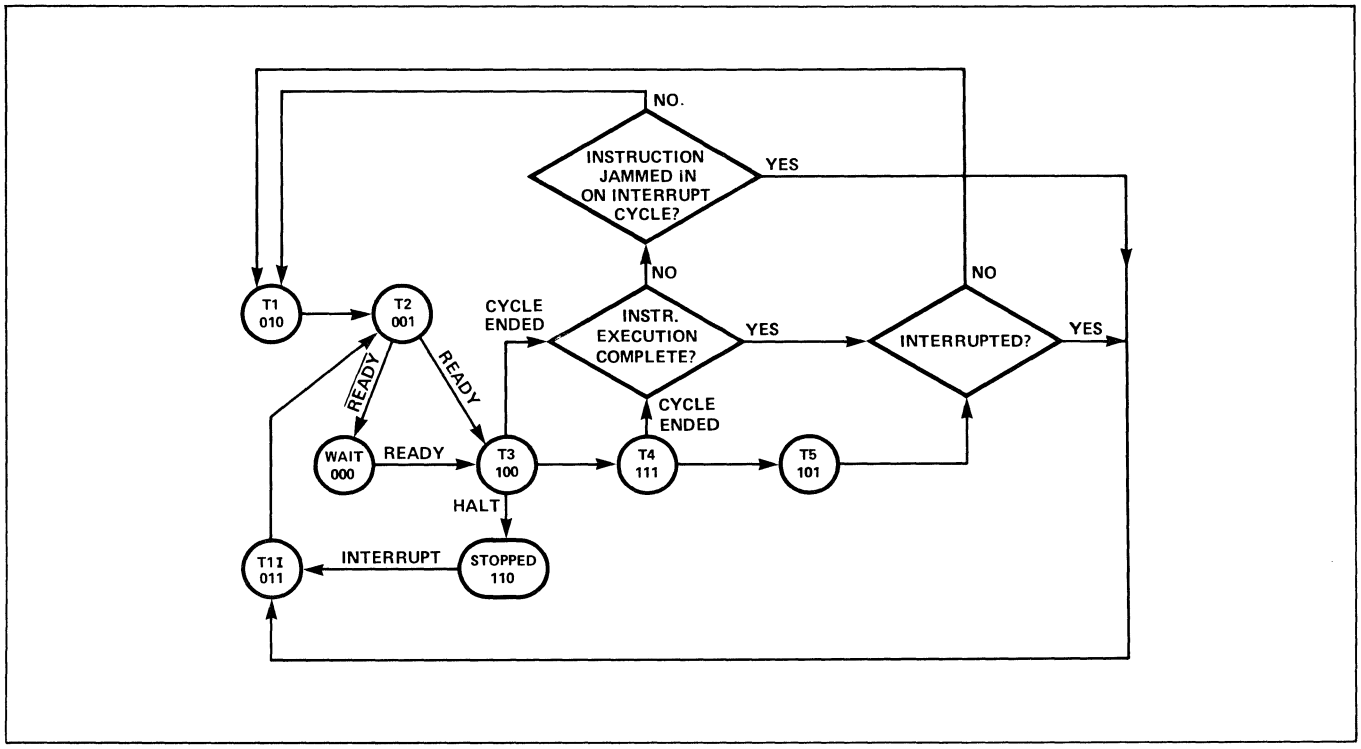


Figure 2. CPU State Transition Diagram

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.

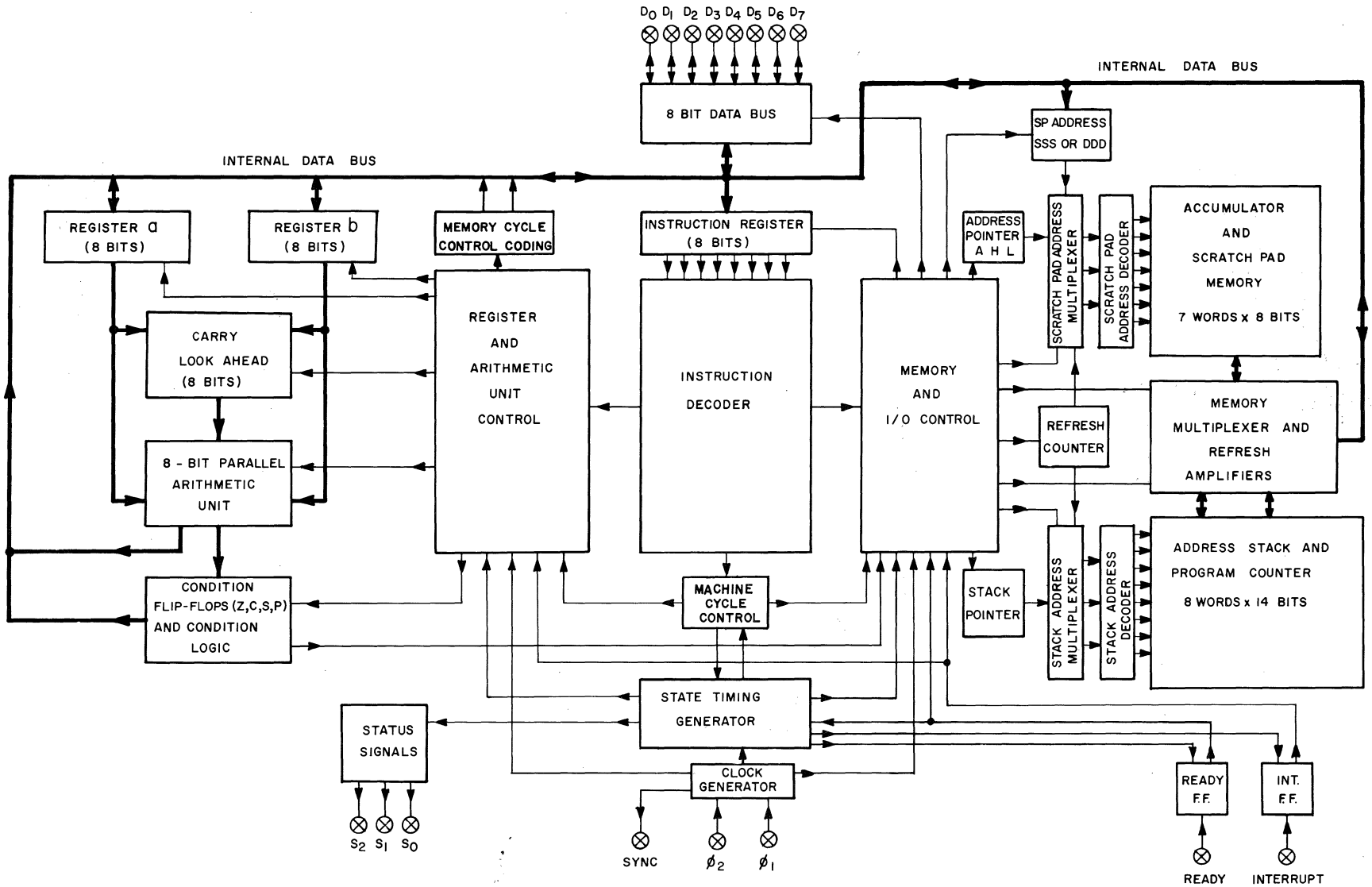


Figure 3. 8008 Block Diagram

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. The instruction formats then depend on the particular operation executed.

FORMAT — **TYPICAL INSTRUCTIONS**

One Byte Instructions

OP CODE

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Register to register, memory reference, I/O arithmetic or logical, rotate or return instructions

Two Byte Instructions

OP CODE								OPERAND							
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Immediate mode instructions

Three Byte Instructions

OP CODE								LOW ADDRESS								HIGH ADDRESS*								
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	X	X	D_5	D_4	D_3	D_2	D_1	D_0	JUMP or CALL instructions

*For the third byte of this instruction, D_6 and D_7 are "don't care" bits.

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) Lrl	(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.

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.
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 to the accumulator with carry.
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.
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.
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	

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	1 0 0	1 0 0	
		B B	B B B	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	1 0 1	1 0 0	
		B B	B B B	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	1 1 0	1 0 0	
		B B	B B B	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	1 1 1	1 0 0	
		B B	B B B	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	X X X	1 0 0	Unconditionally jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
(5) JFc	(9 or 11)	0 1	0 C_4 C_3	0 0 0	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop c is false. Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
JTc	(9 or 11)	0 1	1 C_4 C_3	0 0 0	Jump to memory address $B_3 \ldots B_3B_2 \ldots B_2$ if the condition flip-flop c is true. Otherwise, execute the next instruction in sequence.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CAL	(11)	0 1	X X X	1 1 0	Unconditionally call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$. Save the current address (up one level in the stack).
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CFc	(9 or 11)	0 1	0 C_4 C_3	0 1 0	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ 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.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
CTc	(9 or 11)	0 1	1 C_4 C_3	0 1 0	Call the subroutine at memory address $B_3 \ldots B_3B_2 \ldots B_2$ 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.
		B_2 B_2	B_2 B_2 B_2	B_2 B_2 B_2	
		X X	B_3 B_3 B_3	B_3 B_3 B_3	
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.
RES	(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_4C_3: carry (00), zero (01), sign (10), parity (11).

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.)

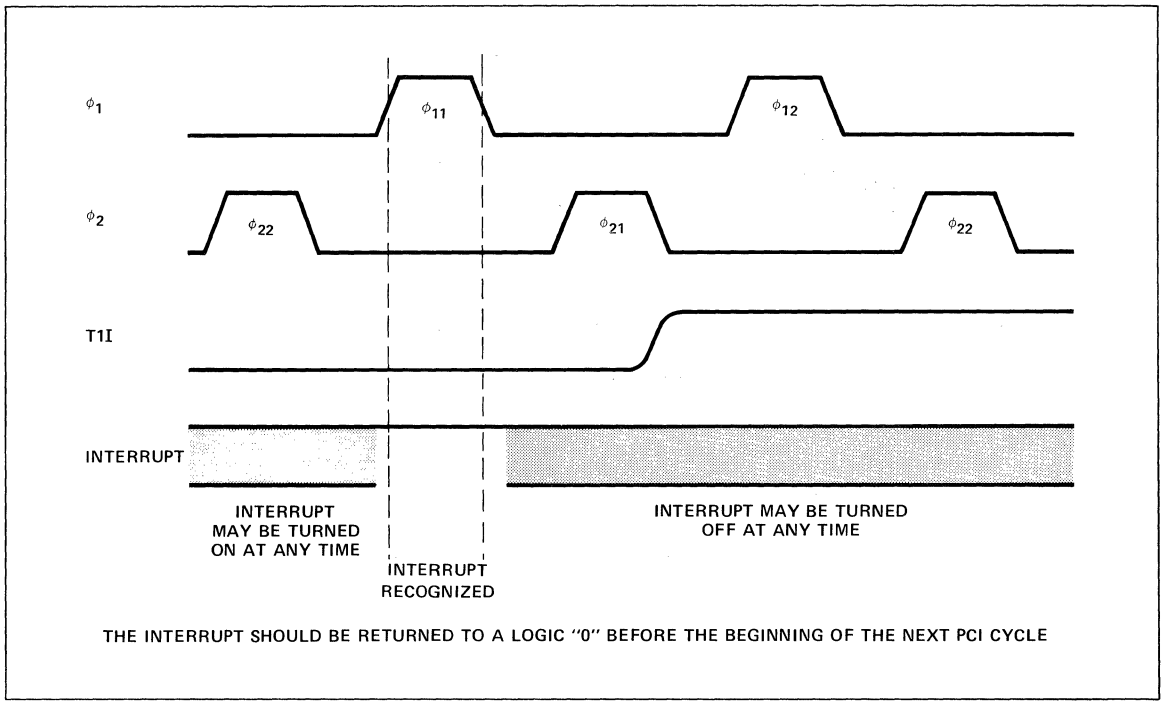


Figure 4. Recognition of Interrupt

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.

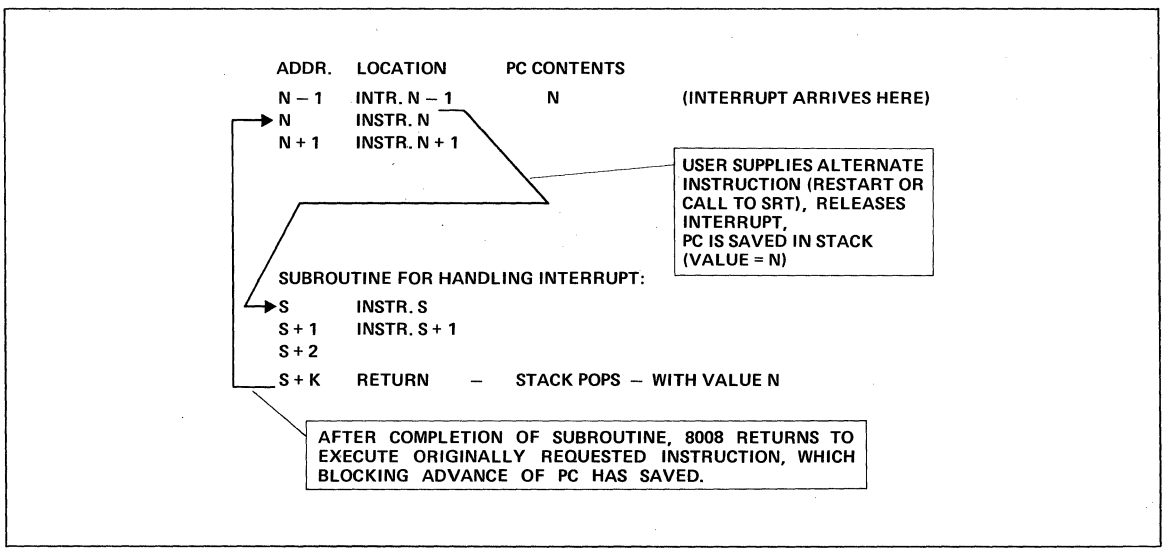


Figure 5. 8008 Interrupt

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. 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.

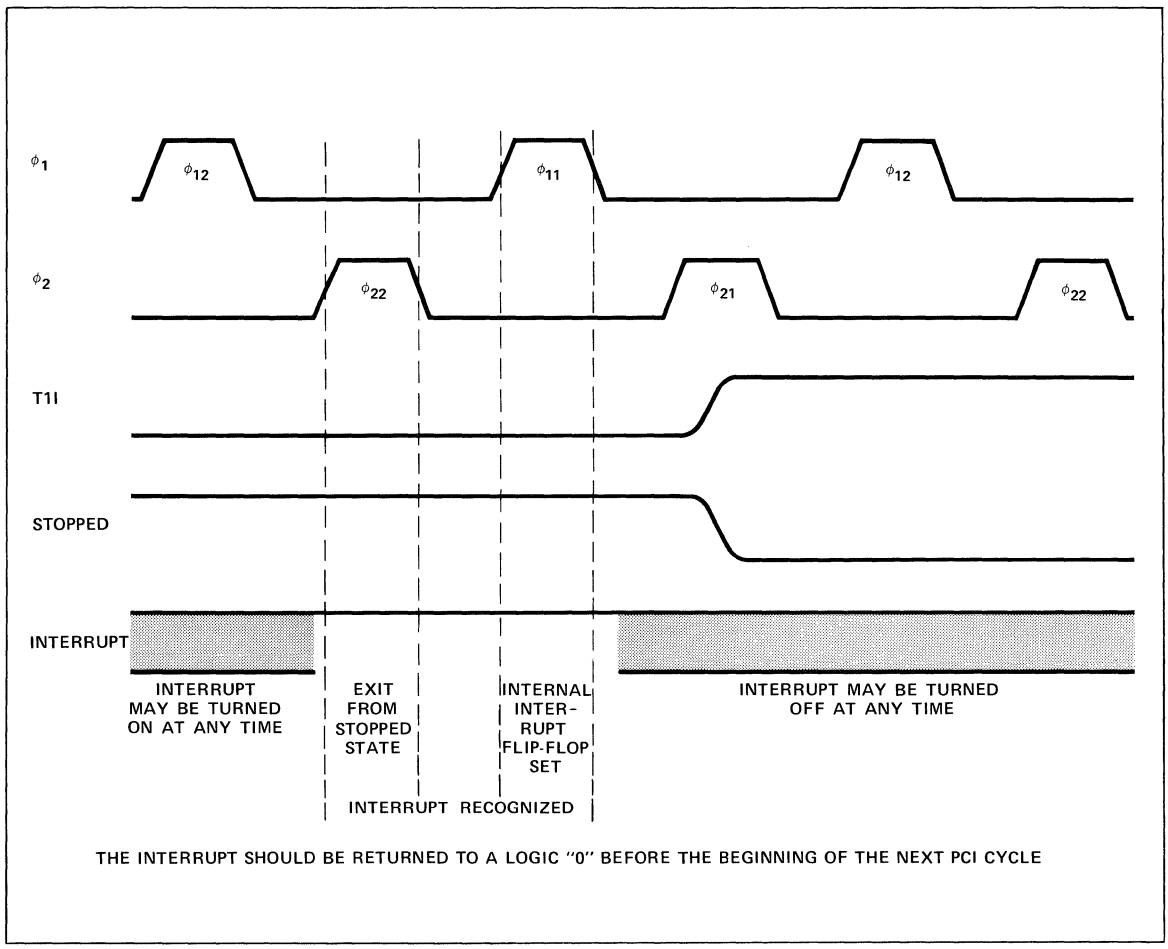


Figure 6. Start-up of the 8008– Interrupt Required

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.

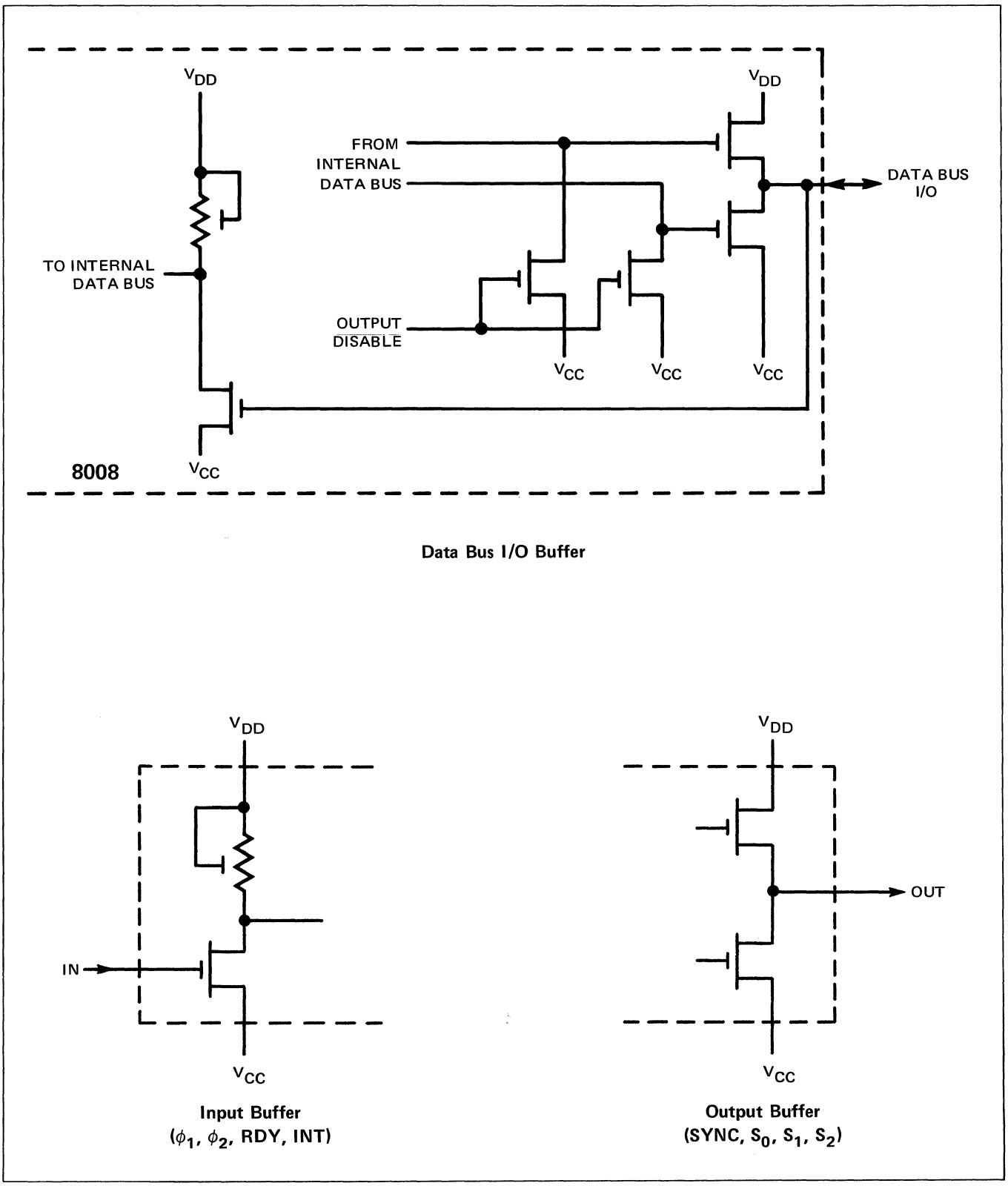


Figure 7. I/O Circuitry

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_{SS}	+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.

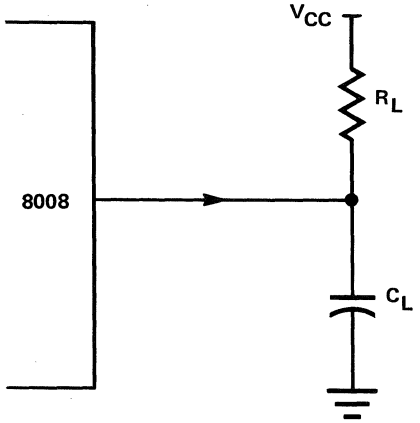
D. C. and OPERATING CHARACTERISTICS

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			UNIT	TEST CONDITIONS
		MIN.	TYP.	MAX.		
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. The test load is shown below.

TEST LOAD

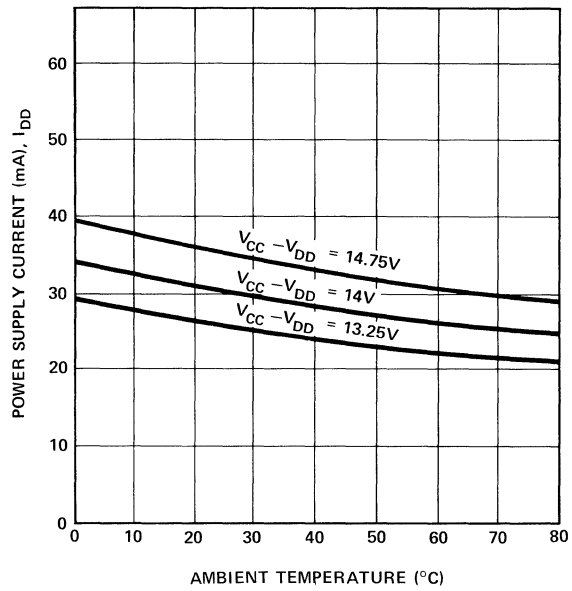


TEST CONDITIONS

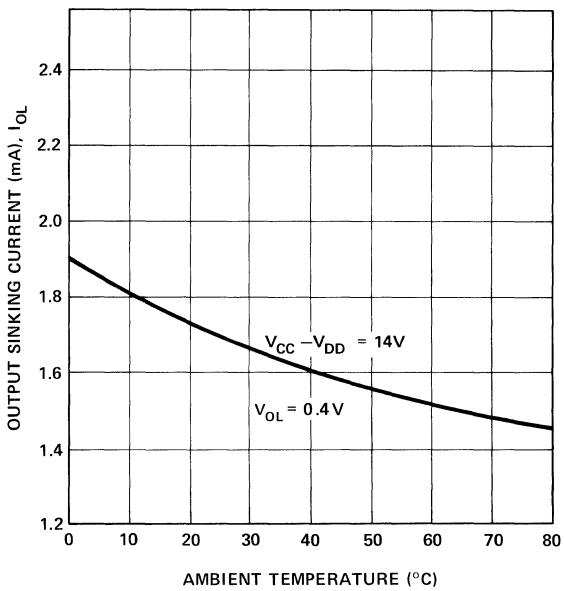
V_{CC} = 5.25V
R_L = 11kΩ
C_L = 100pF

TYPICAL D.C. CHARACTERISTICS

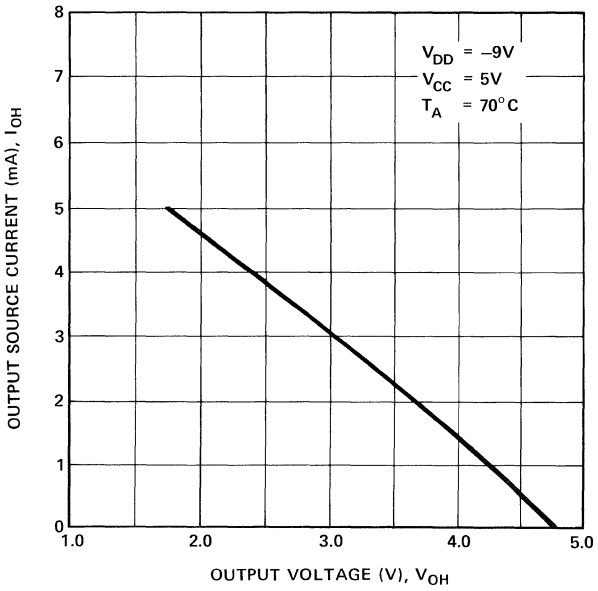
POWER SUPPLY CURRENT VS. TEMPERATURE



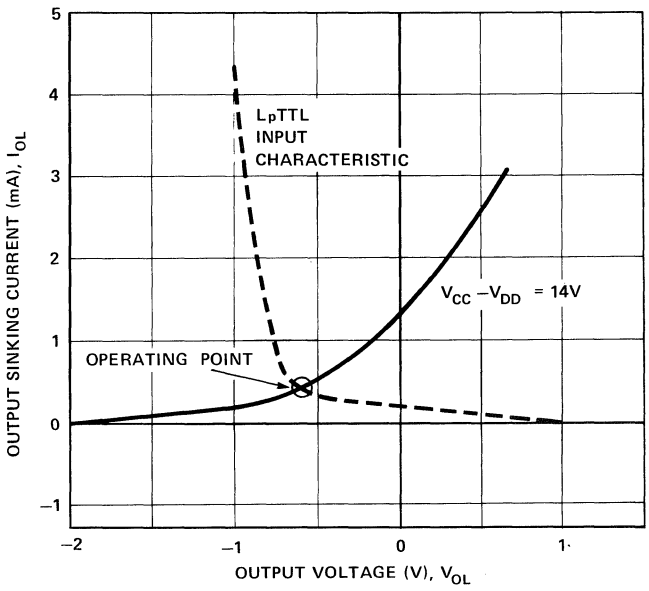
OUTPUT SINKING CURRENT VS. TEMPERATURE



OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



OUTPUT SINKING CURRENT VS. OUTPUT VOLTAGE



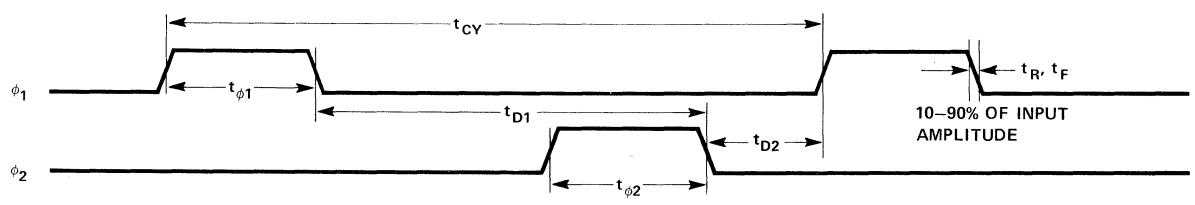
A. C. CHARACTERISTICS

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

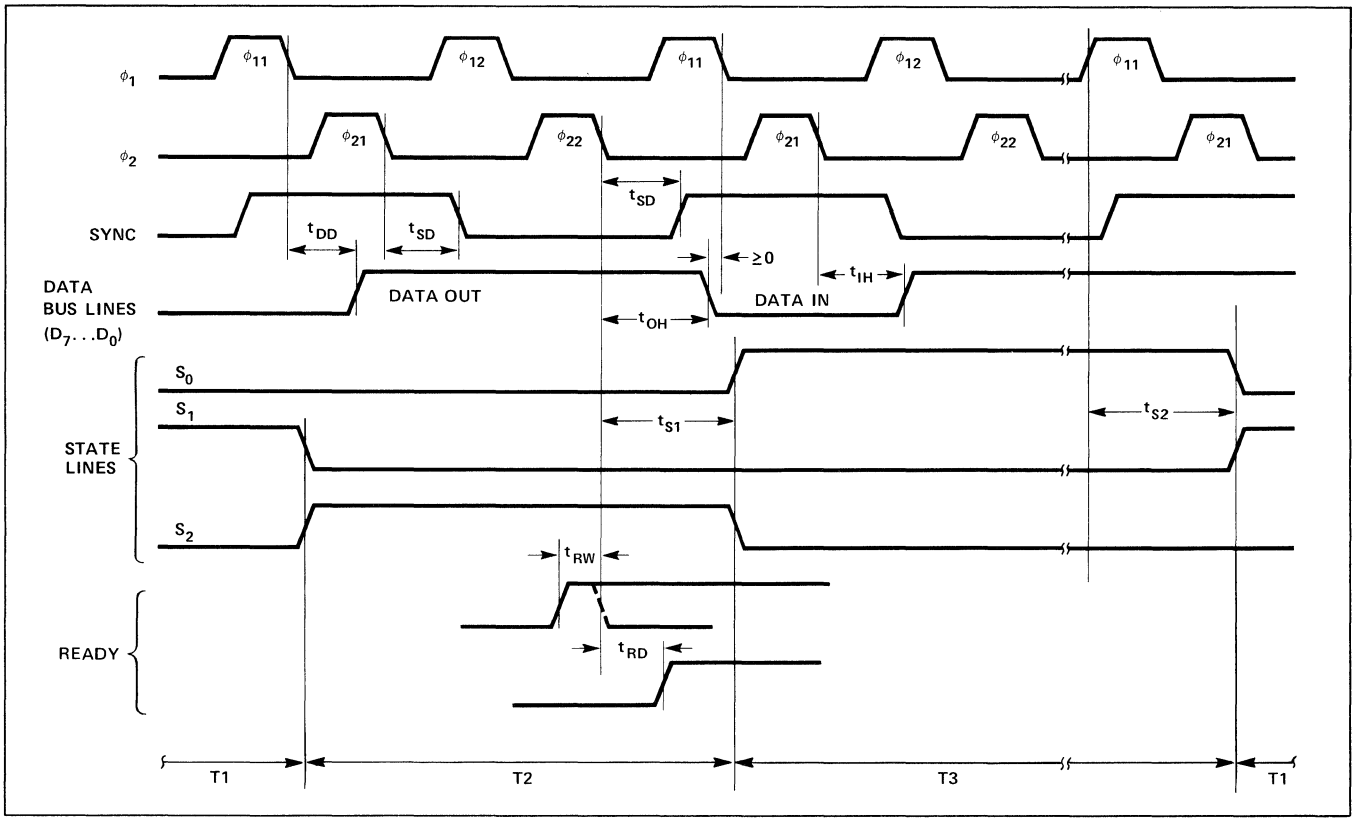
SYMBOL	PARAMETER	LIMITS		UNIT	TEST CONDITIONS
		MIN.	MAX.		
t_{CY}	CLOCK PERIOD	2	3	μs	t_R, t_F = 50ns
t_R, t_F	CLOCK RISE AND FALL TIMES		50	ns	
$t_{\phi 1}$	PULSE WIDTH OF ϕ_1	.70	1.0	μs	
$t_{\phi 2}$	PULSE WIDTH OF ϕ_2	.55	.70	μs	
t_{D1}	CLOCK DELAY FROM FALLING EDGE OF ϕ_1 TO FALLING EDGE OF ϕ_2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

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

CLOCK WAVEFORM



TIMING DIAGRAM

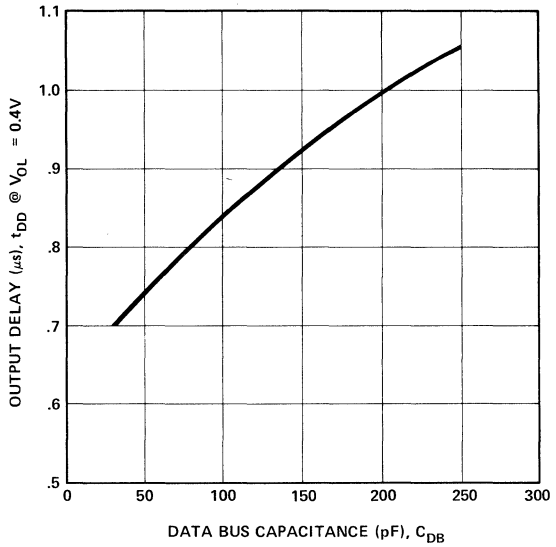


CAPACITANCE f = 1MHz; T_A = 25°C; Unmeasured Pins Grounded

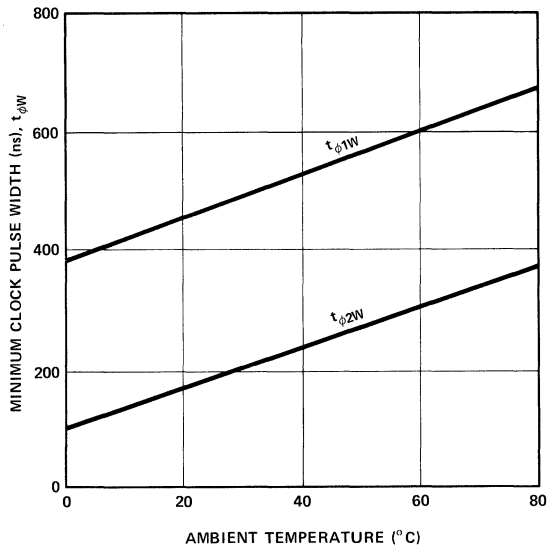
SYMBOL	TEST	LIMIT (pF)	
		TYP.	MAX.
C_{IN}	INPUT CAPACITANCE	5	10
C_{DB}	DATA BUS I/O CAPACITANCE	5	10
C_{OUT}	OUTPUT CAPACITANCE	5	10

TYPICAL A.C. CHARACTERISTICS

DATA OUT DELAY VS. OUTPUT LOAD CAPACITANCE



MINIMUM CLOCK PULSE WIDTH VS. TEMPERATURE



VII MCS-8 PROTOTYPE SYSTEM

The 8008 is designed to operate with standard semiconductor memories. TTL interface circuitry to memories is required to latch the time multiplexed addresses and gate data or instructions onto the data bus. As an aid to program development, Intel has developed a complete prototyping system, the SIM8-01. The preliminary schematic of this board is shown on pages 20 and 21.

The SIM8-01 is a single board which contains all of the circuitry required to interface the 8008 with standard ROMs and RAMs. It contains a 1K x 8 static RAM memory and has sockets for up to 2K x 8 ROM memory. During program development, Intel's 1702 electrically programmable and erasable ROM may be used. The 8008 can directly address up to 16K x 8 of memory. All control lines, address bits, and data lines are provided for memory expansion. Both the INTERRUPT and READY control lines are also made available.

The 8008 can directly reference 32 different I/O ports. Six ports are included on the SIM8-01, two input ports and four output ports with latches.

In addition, a 500 kHz system clock and TTY interface circuits are on the board.

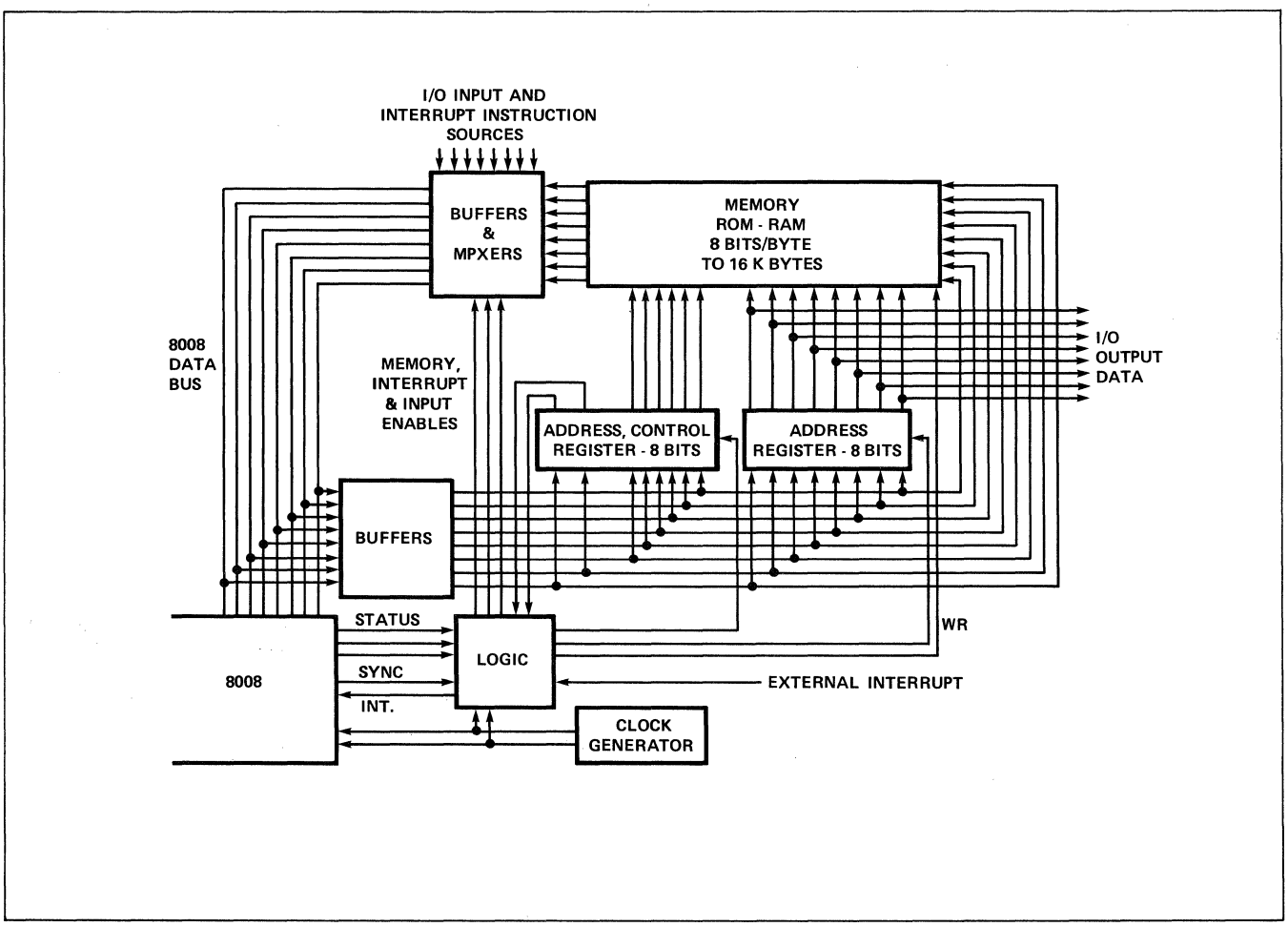


Figure 8. MCS-8 Basic System

SIM8-01 SPECIFICATIONS

Card Dimensions:
- 11.5 inches high
- 8.0 inches deep

System Components Included on Board:
- 8008
- 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

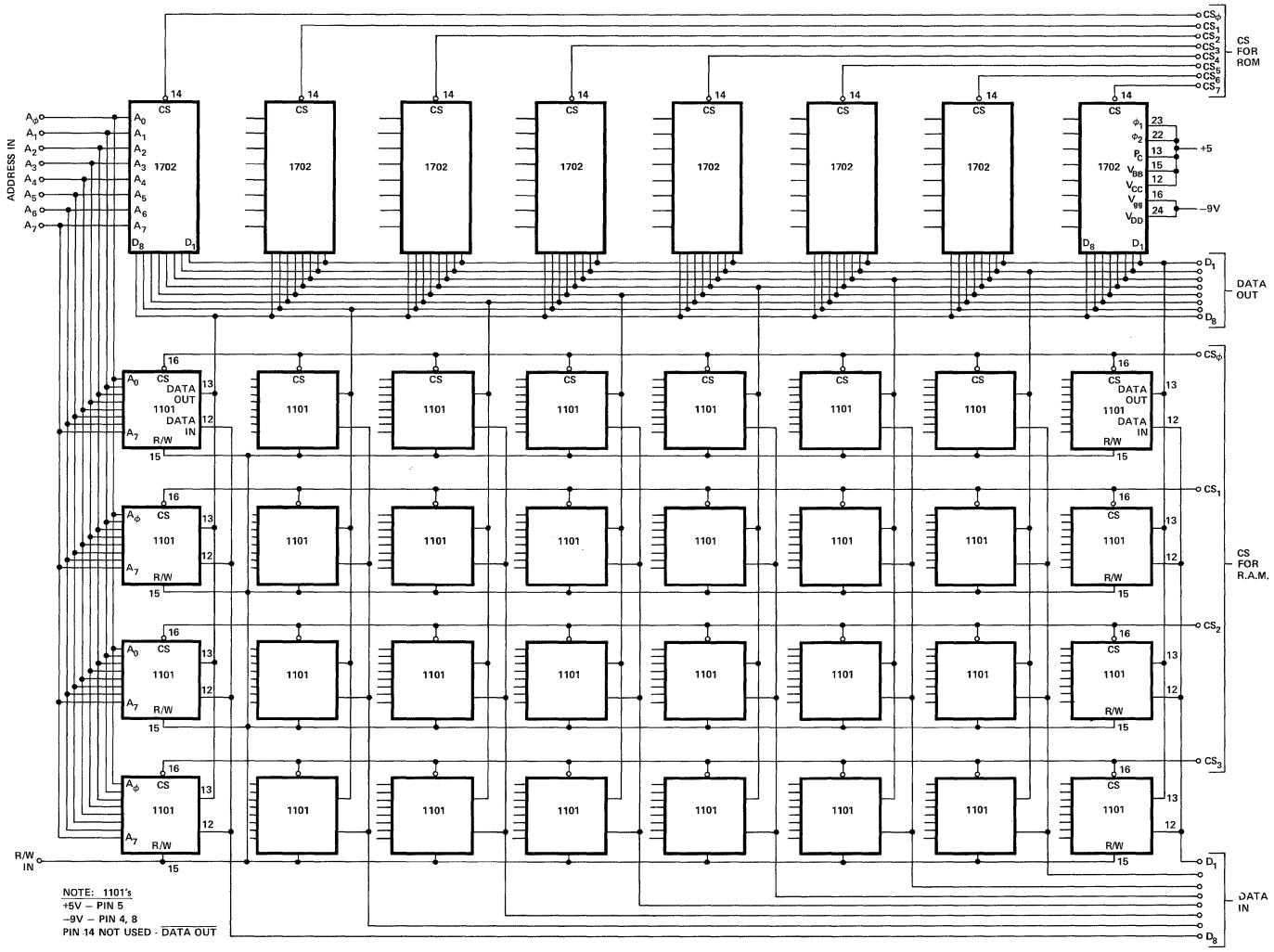
Maximum Memory Configuration:
- 1K x 8 RAM
- 2K x 8 PROM
- All control lines are provided for memory expansion

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:
  No ROMs
  I_{CC} = 1.8 amps
  I_{DD} = .6 amps

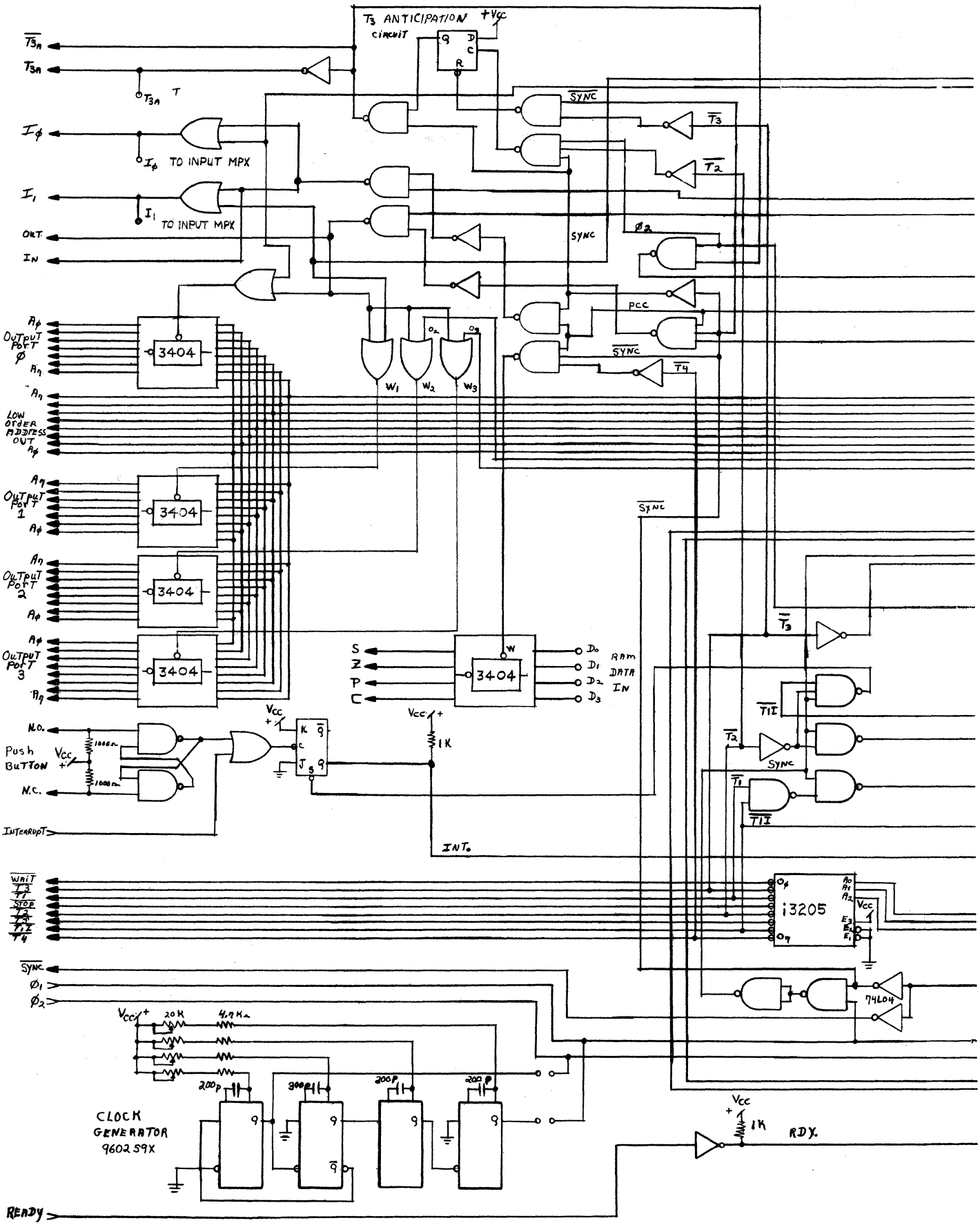
  Eight ROMs
  I_{CC} = 2.2 amps
  I_{DD} = 1 amp

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



The memory system shown above or an equivalent will be used for the SIM 8-01 micro computer.

Figure 9. MCS-8 Memory System


T3 ANTICIPATION CIRCUIT
+Vcc
SYNC
T3
T2
T4
T1I
I TO INPUT MPX
OUT
IN
PCC
W1
W2
W3
3404
OUTPUT PORT Ø
OUTPUT PORT 1
OUTPUT PORT 2
OUTPUT PORT 3
LOW ORDER ADDRESS OUT
RAM DATA IN
Push BUTTON
N.O.
N.C.
INTERRUPT
INT.
1K
i3205
74L04
CLOCK GENERATOR 9602 59X
20K
4.7K
200P
300P
RDY.
READY
WAIT
STOP

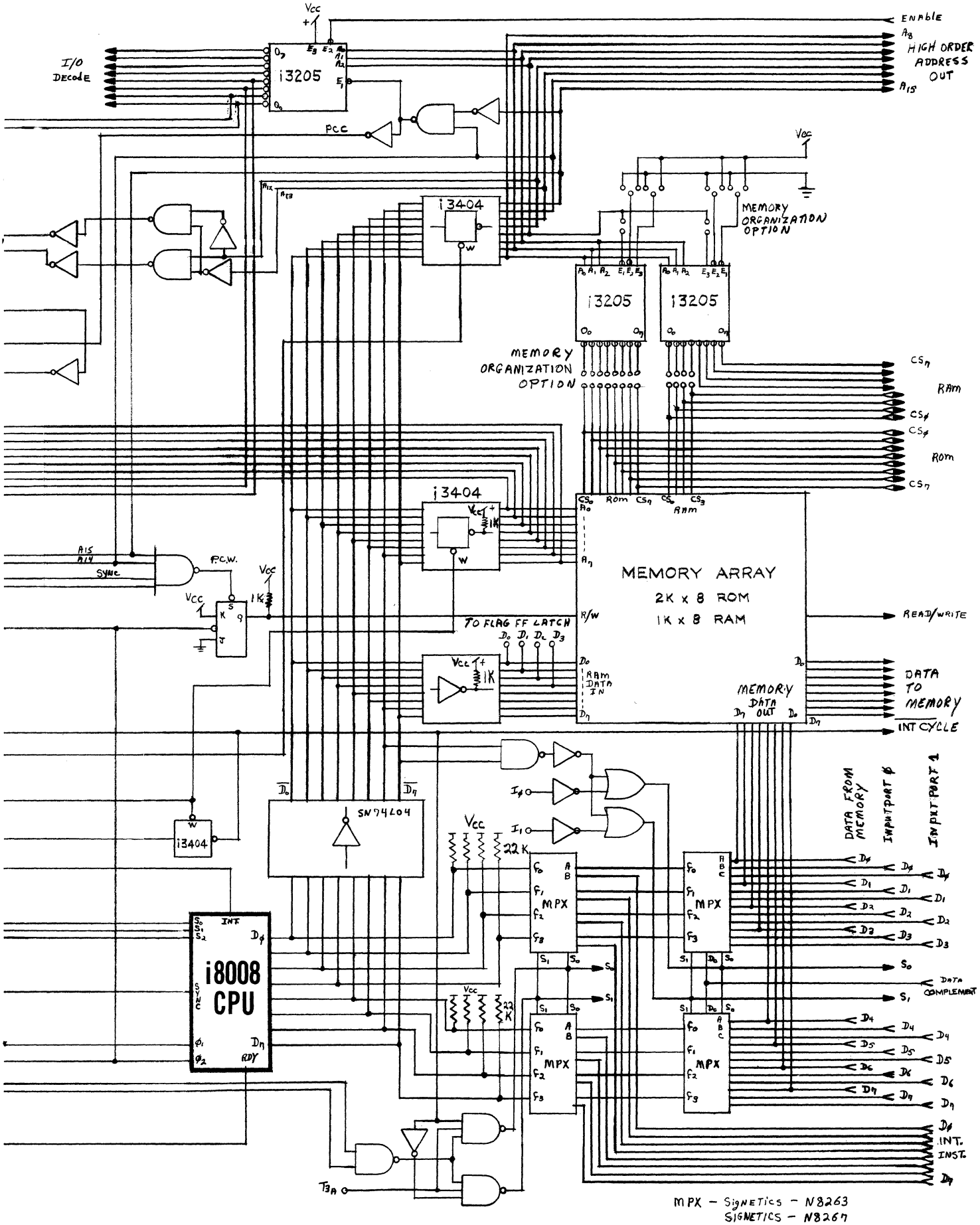


Figure 10. Preliminary SIM8-01 Schematic

VIII. MCS-8 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 two special printed circuit boards produced by Intel 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) ROM programming
2) Format checking
3) Error checking
4) Program listing

For specifications of the Intel ROMs, refer to the 1601/1701, 1602/1702 data sheet.

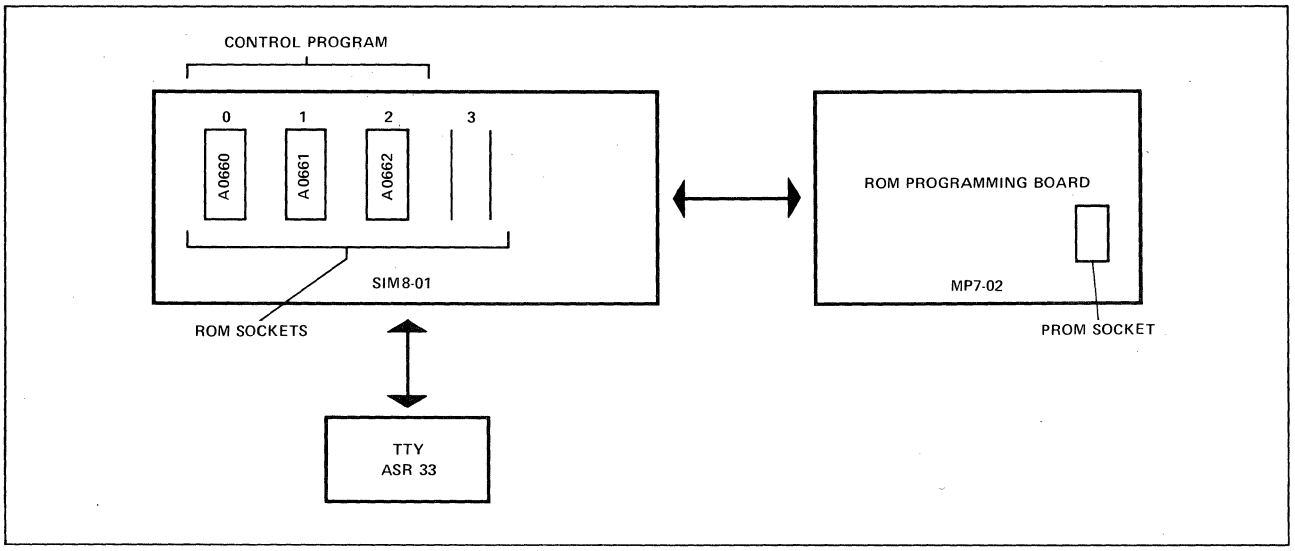


Figure 11. MCS-8 PROM Programming System

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 ROMs (1601/1701 or 1602/1702) for the microprogram control. The total system is controlled by the 8008 CPU.

2) The control programs (Intel tape numbers A0660, A0661, A0662)
   These three 1602 ROMs contain the microprograms which control the programming, format and error checking, and listing functions.

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

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 (−40V − P on tape) will leave a "1" and a high data input level (ground-N on tape) will allow programming of "0" (see table below). All eight bits of one word are programmed simultaneously by setting the desired bit information patterns on the data input terminals.

When the Data Input for the Program Mode is:	Then the Data Output during the Read Mode is:
$V_{ILIP} = \sim -40$ V pulsed	Logic Level High = V_{OH} = 'P' on tape
$V_{IHP} = \sim 0$ V	Logic Level Low = V_{OL} = 'N' on tape

WORD \ ADDRESS	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
|	|	|	|	|	|	|	|	|
|	|	|	|	|	|	|	|	|
255	1	1	1	1	1	1	1	1

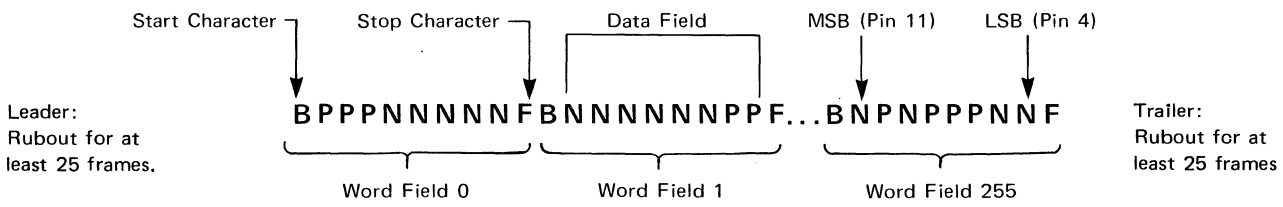
Address Logic Level During Read Mode: Logic 0 = V_{IL} (~.3V) Logic 1 = V_{IH} (~3V)

Address Logic Level During Program Mode*: Logic 0 = V_{IL2P} (~−40V) Logic 1 = V_{IHP} (~0V)

*The Logic Levels for the address inputs are inverted from the Logic Levels for the data inputs during the Program Mode.

TAPE FORMAT

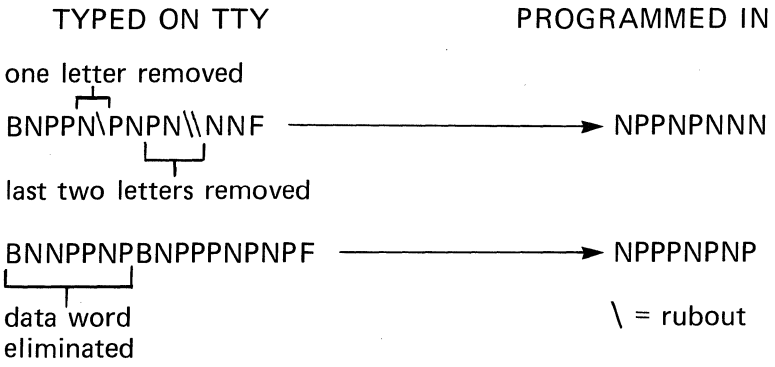
The tape reader used with a model 33 ASR teletype accepts 1" wide paper tape using 7 or 8 bit ASCII code. For a tape to correctly program a 1601/1701 or 1602/1702, it must follow exactly the format rules below:



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 − refer to the table shown above) 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 rubout will eliminate the previous character entered. If a B is entered in place of a rubout, the entire word is eliminated. **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:

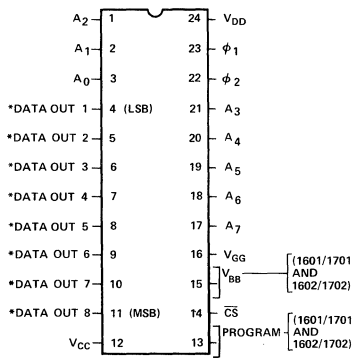


If any character other than P or N is entered, a format error is indicated. If the stop character is entered before the error is noticed, the entire word field, including the B and F, must be rubbed out. **Within the word field, a P results in a high level output, and N results in a low level output.** 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 LLD (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

PROGRAMMING

Two different modes of operation are available*

1) A complete program tape consisting of 256 data words in sequential order may be used.

   a) **To program the complete ROM** place the ROM to be programmed into the socket on the MP7-02 board, reset the system, and then type.

S	(start command)
ØØØ	(initial address - address 0 of the ROM)
255	(final address - address 255 of the ROM)

   Start the tape (data words may also be entered in sequence manually). The ROM will be programmed in all 256 locations.

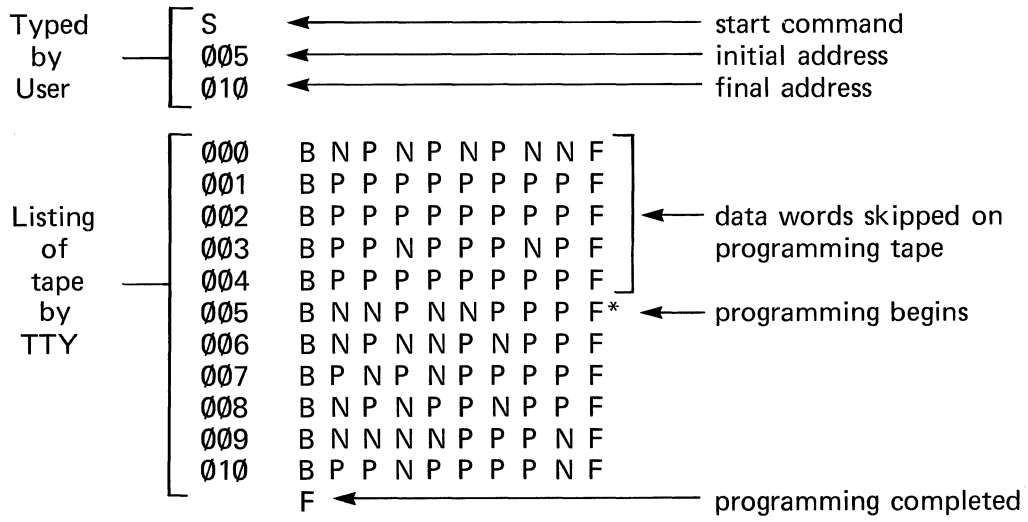
   b) **To skip a section of the ROM** (skipping the same number of data words on the tape) and then program a section of the ROM while still keeping the addresses in sequential form on the complete tape, type,

S	(start command)
XXX	(initial address)
YYY	(final address)

   Start the tape. The ROM will only be programmed in the specified locations.

*Operate the TTY in the "line" position. Depress the RETURN key (RET) after each manual data word entry. When using the tape reader to enter the program data, switch the reader to "start" after the final address is entered.

EXAMPLE 1:



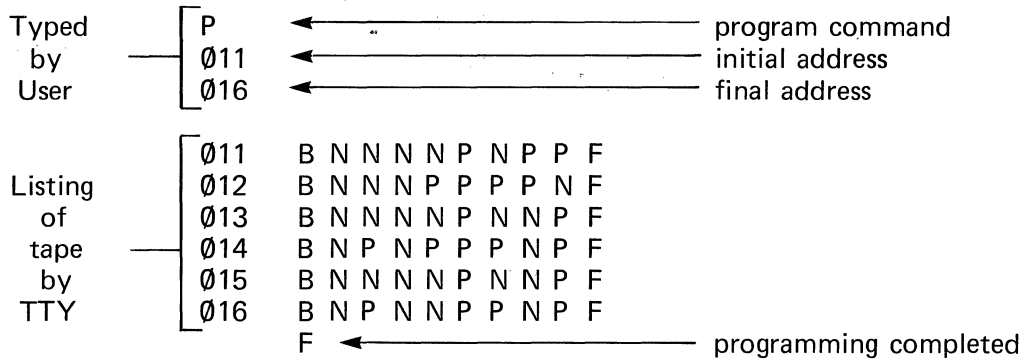
In this example the first five data words on the tape are skipped and the next six data words are programmed in the ROM locations ØØ5 to Ø1Ø. The "*" indicates the first instruction programmed, and the "F" at end of the listing indicates the completion of the programming.

2) **To program any portion of the ROM** without skipping a section of the tape, use a short tape (data words in sequence) and type

P	(program command)
XXX	(initial address)
YYY	(final address)

Start the tape (data words may also be entered manually).

EXAMPLE 2:

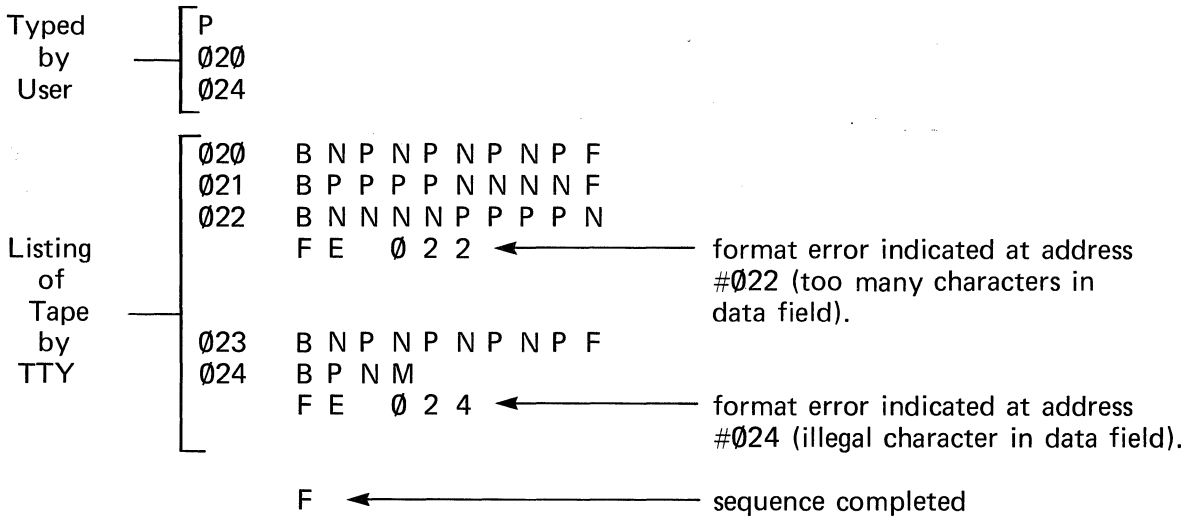


This example shows that the first six instruction words in sequence on a tape are read and directly programmed into ROM locations Ø11 through Ø16 without skipping.

FORMAT CHECKING

When the programmer detects the first format error (data words enter either on tape or manually), it will stop programming the ROM, and it will print out the address where the format error occurred. If a tape is being used, the programming system will continue to list the content of the tape and will print out the address of each subsequent format error.

EXAMPLE 3:

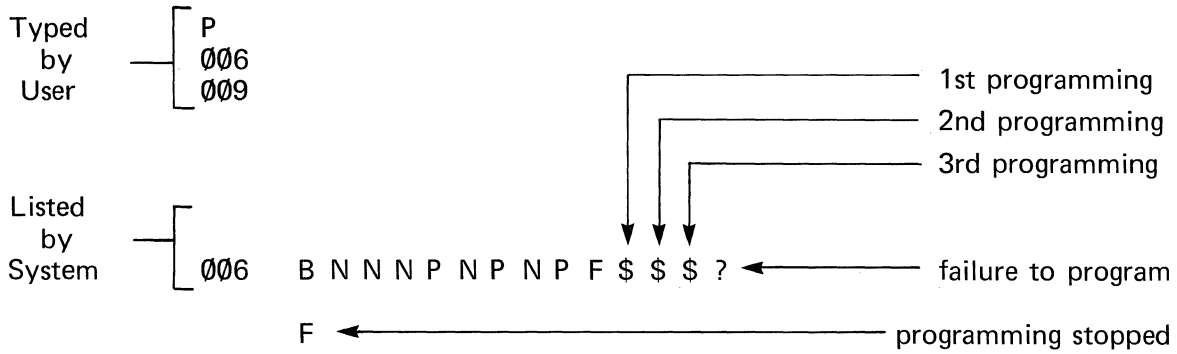


If data words are being entered manually and a format error is encountered, the system must be reset before continuing the programming.

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 ROM's will be identified.

EXAMPLE 4:



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

After the programming is complete, 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, type

L	(list command)
XXX	(initial address)
YYY	(final address)

EXAMPLE 5:

Note that this is a listing of the ROM programmed in Example 1. The listing feature may also be used to verify that a 1701 or 1702 is completely erased.

1701, 1702 ERASING PROCEDURE:

The 1701 and 1702 may be erased by exposure to a high intensity ultraviolet light source. A 1701 or 1702 placed 1 to 1.5 inches from a light source with an ultraviolet wavelength of 2536A at an intensity of 10 mW/cm^2 (i.e. a dosage of 6W sec/cm^2) will be erased in 3 minutes. An example of a light source which is capable of producing the required ultraviolet wavelength and intensity is the Model R51 manufactured by Ultraviolet Products (San Gabriel, California). Any inexpensive light source that meets the spec may be used.

B. MP7-02 Programming System

The MP7-02 easily interfaces with the SIM8-01. All address and data lines are completely TTL compatible. The MP7-02 requires +5 VDC @ 0.8 amps, –9 VDC @ 0.2 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.

The MP7-02 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 MP7-02 card.

The schematic and board layout for the MP7-02 are shown on the following pages.

Programmer Board Specifications

Dimensions:
- 8.4 inches high
- 9.5 inches deep

Power Requirement:
- V_{CC} = +5 @ 0.8 amps
- TTL GRD = 0V
- *V_{DD} = –9V @ 0.2 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.

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

Features:
- Inputs and outputs TTL compatible
- Board sold complete with transformers, capacitor and connector
- Directly interfaces with SIM8-01 Boards (Interconnection available on request)

C. 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

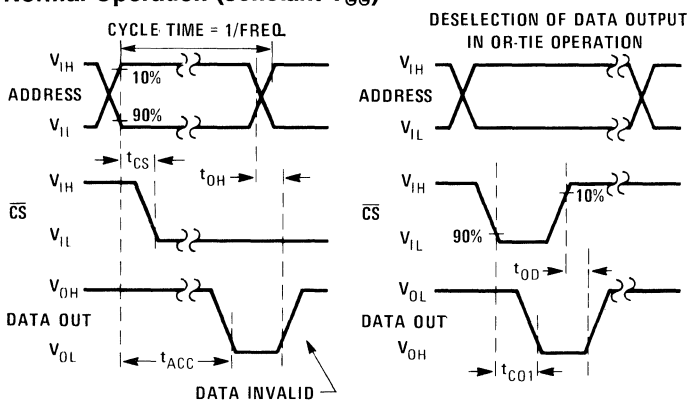
(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, V_{GG} = −9V ±5% unless otherwise noted)

Test	1601/1701 1602/1702 Typ.	1601/1701 1602/1702 Max.	1301 Typ.	1301 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		5	10	mA
Average power supply current @ T_A = 25°C	1601	30	45	mA
	1301	28	40	mA

A.C. Characteristics for dynamic operation

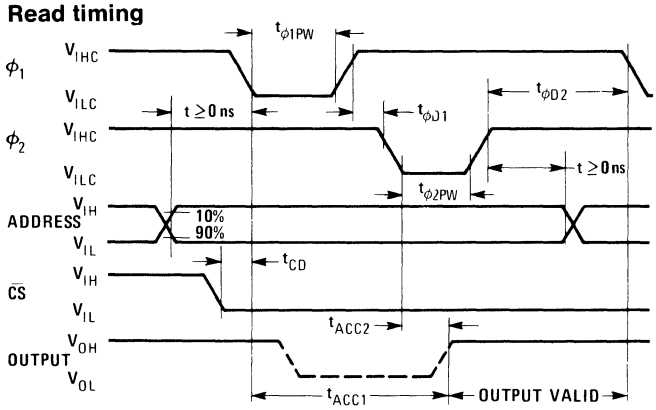
(T_A = 0°C to 70°C, V_{CC} = +5V ±5%, V_{DD} = −9V ±5%, unless otherwise noted)

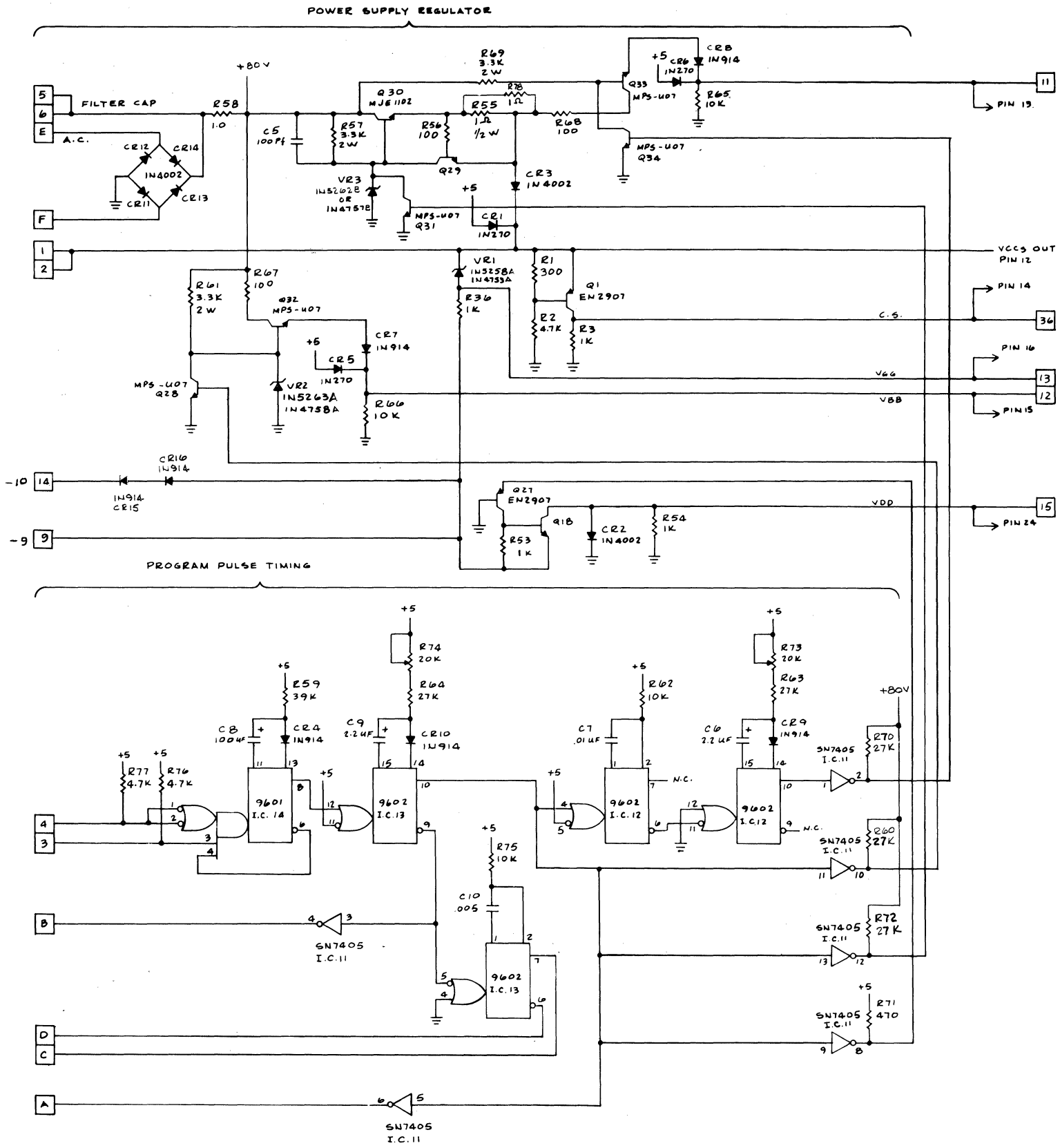
Test	1601/1701 Min.	1601/1701 Typ.	1601/1701 Max.	1301 Min.	1301 Typ.	1301 Max.	Unit
Ø1 Clock pulse width	0.260		2	.260		2	μs
Ø2 Clock pulse width	0.140		2	.140		2	μs
Ø2 delay from Ø1	0.150		2	.150		2	μs
Ø1 delay from Ø2	.05		2	.05		2	μs
Address to output access		450	650		450	650	ns

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)

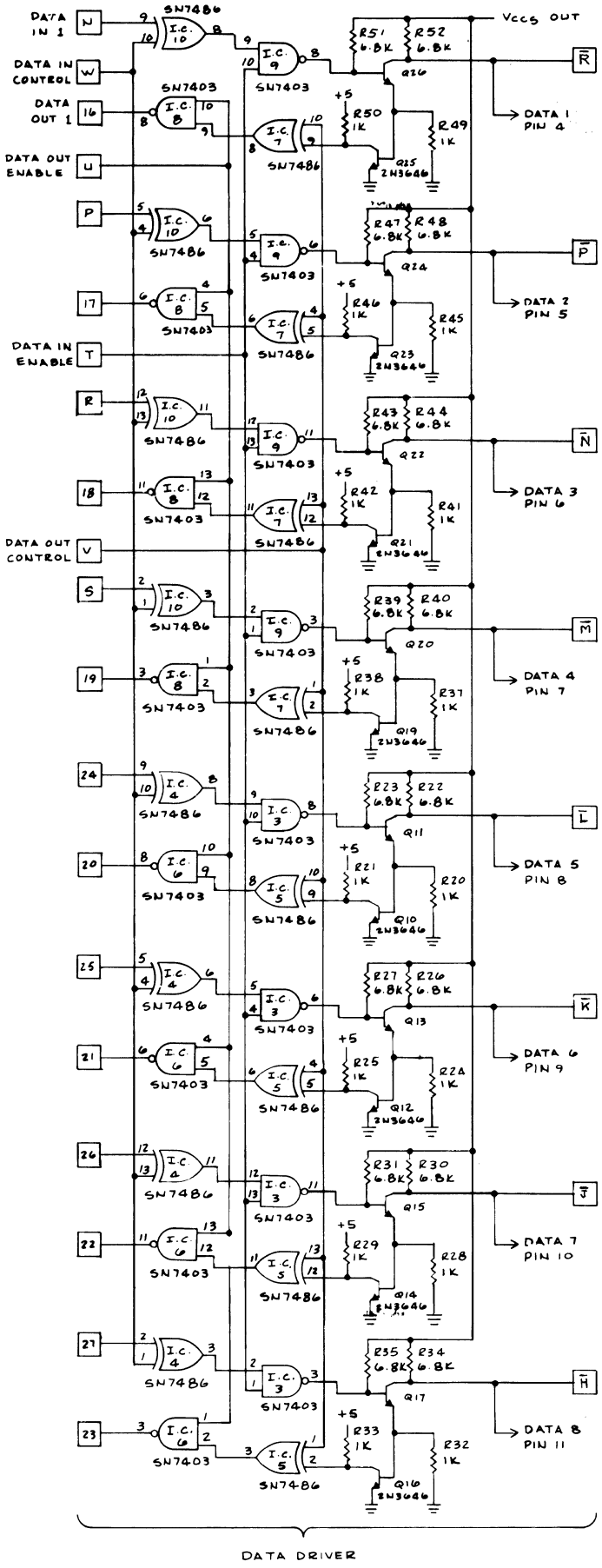
Read timing

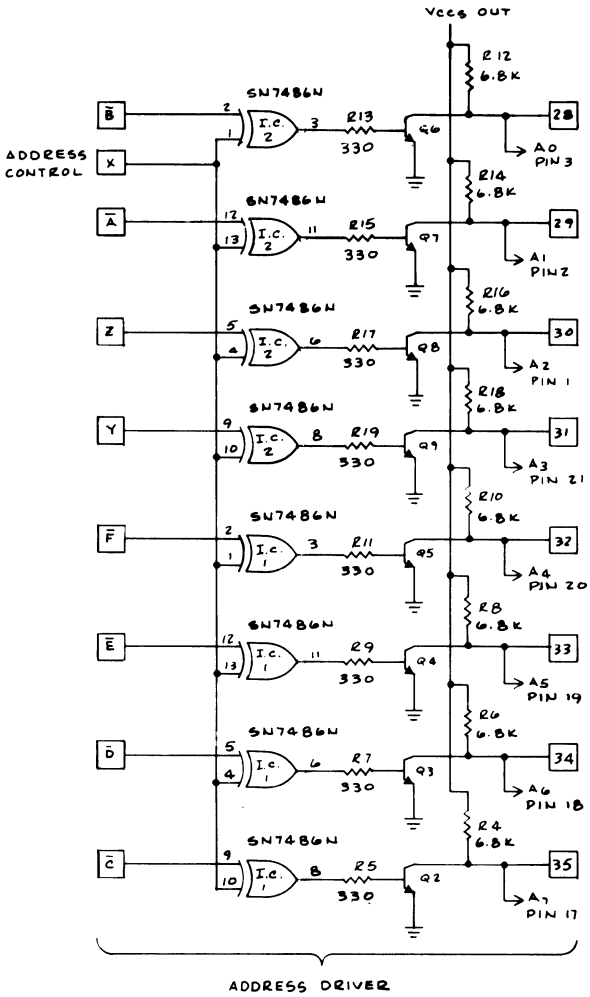




NOTES: Unless Otherwise Specified

1. Resistors are rated in ohms ¼w, 10%.
2. Transistors are SE 6021.
3. Pin numbers are specified for the solder lug connector.





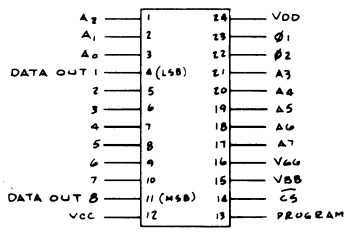


Figure 12. MP7-02 PROM Programmer Board Schematic

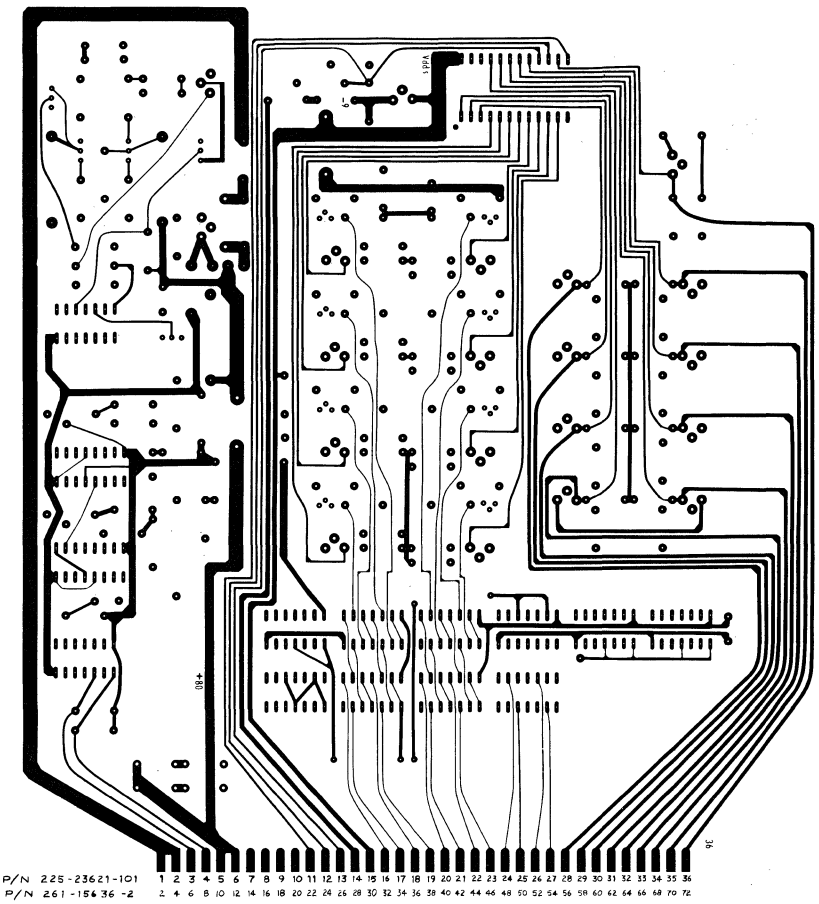


Figure 13. Circuit Side of MP7-02 Card

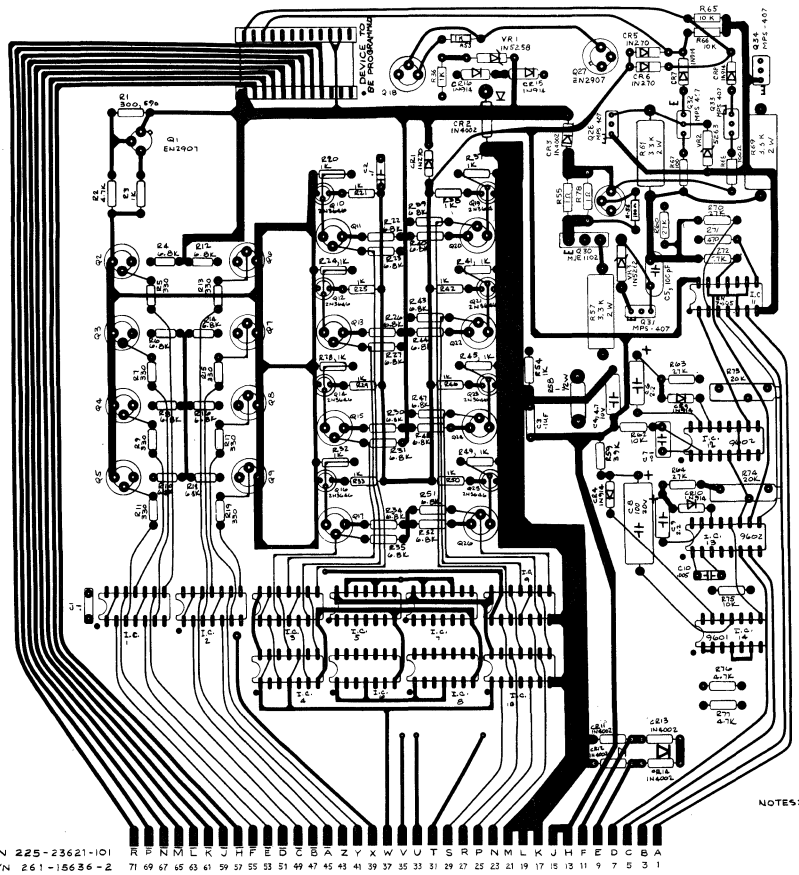


Figure 14. Component Side of MP7-02 Card

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: 00 carry 01 zero 10 sign 11 parity
M	Memory location indicated by the contents of registers H and L
( )	Contents of location or register
$\wedge$	Logical product
$\not\vee$	Exclusive "or"
V	Inclusive "or"
A_m	Bit m of the A-register
STACK	Instruction counter (P) pushdown register
P	Program Counter
←	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)←(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)←(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 $<B_2>$	110	$(r) \leftarrow <B_2>$ Load byte two of the instruction into register r.
LMI (three cycles – PCI/PCR/PCW)	00	111 $<B_2>$	110	$(M) \leftarrow <B_2>$ 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)←(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.

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)←(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)←(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)←(A)–(r) Subtract the content of register r from the content of register A and place the result into register A.

ACCUMULATOR GROUP INSTRUCTIONS - Cont'd.

SBr	10	011	SSS	(A)←(A)−(r)−(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)←(A)∧(r) Place the logical product of the register A and register r into register A.
XRr	10	101	SSS	(A)←(A)∀(r) Place the "exclusive - or" of the content of register A and register r into register A.
ORr	10	110	SSS	(A)←(A)V(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.

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)←(A)+(M) ADD
ACM	10	001	111	(A)←(A)+(M)+(carry) ADD with carry
SUM	10	010	111	(A)←(A)−(M) SUBTRACT
SBM	10	011	111	(A)←(A)−(M)−(borrow) SUBTRACT with borrow
NDM	10	100	111	(A)←(A)∧(M) Logical AND
XRM	10	101	111	(A)←(A)∀(M) Exclusive OR
ORM	10	110	111	(A)←(A)V(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 <B_2>	100	(A)←(A)+<B_2> ADD
ACI	00	001 <B_2>	100	(A)←(A)+<B_2>+(carry) ADD with carry
SUI	00	010 <B_2>	100	(A)←(A)−<B_2> SUBTRACT
SBI	00	011 <B_2>	100	(A)←(A)−<B_2>−(borrow) SUBTRACT with borrow
NDI	00	100 <B_2>	100	(A)←(A)∧<B_2> Logical AND
XRI	00	101 <B_2>	100	(A)←(A)∀<B_2> Exclusive OR
ORI	00	110 <B_2>	100	(A)←(A)V <B_2> Inclusive OR
CPI	00	111 <B_2>	100	(A)− <B_2> 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 $<B_2>$ $<B_3>$	100	(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	$0C_4C_3$ $<B_2>$ $<B_3>$	000	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	$1C_4C_3$ $<B_2>$ $<B_3>$	000	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 $<B_2>$ $<B_3>$	110	(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	$0C_4C_3$ $<B_2>$ $<B_3>$	010	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	$1C_4C_3$ $<B_2>$ $<B_3>$	010	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.

RET	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 ≠ 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.

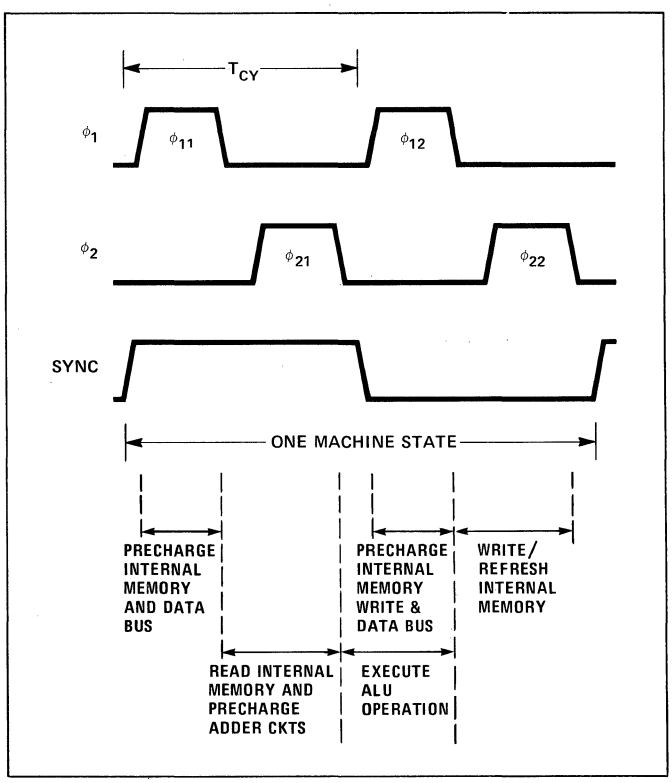


Figure 15. Internal Timing Activity Within Any State

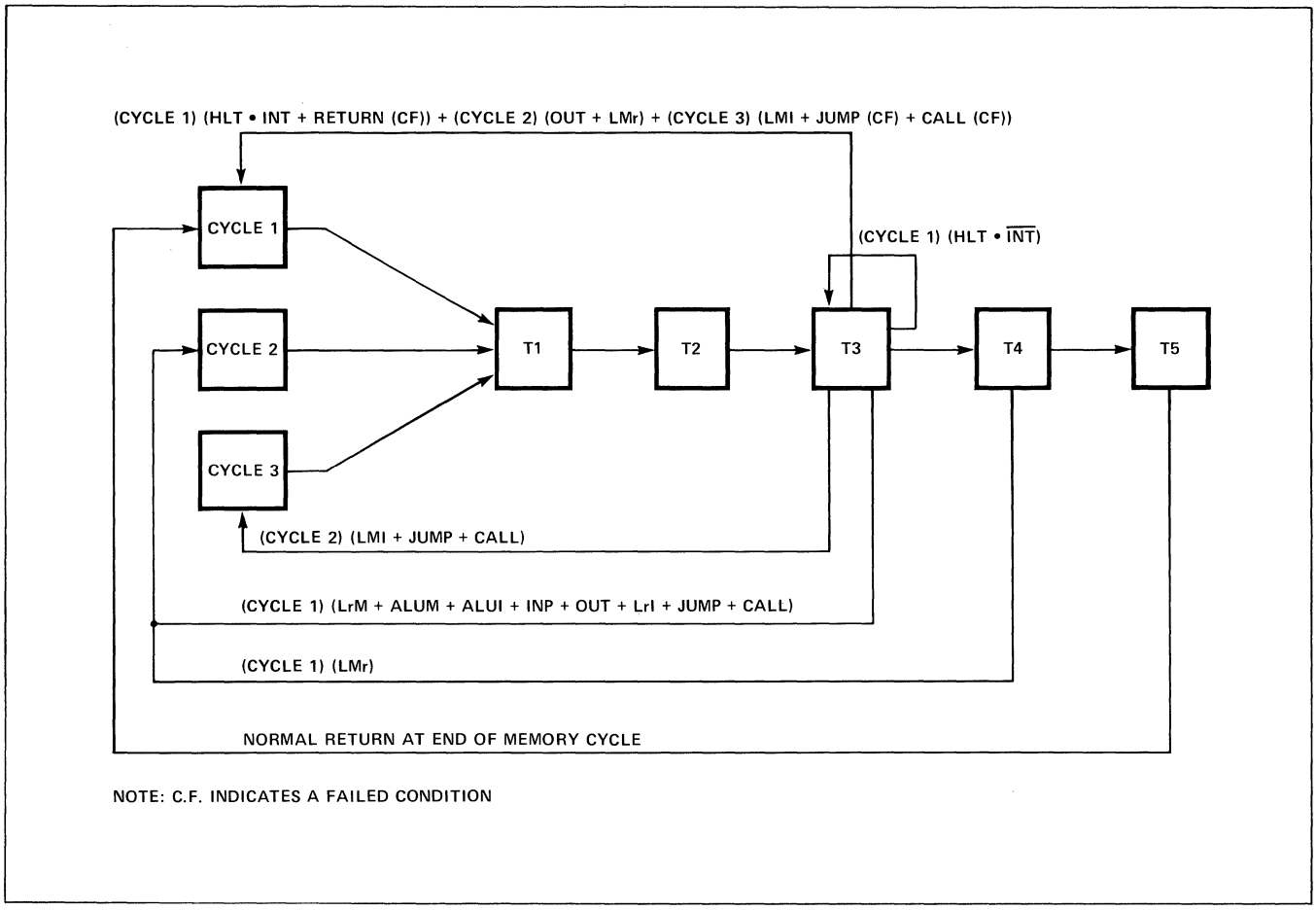


Figure 16. Transition State Diagram (Internal)

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 DD
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	Lrl	(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 1	INr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ADD OP - FLAGS AFFECTED
0 0	D D D	0 0 0	DCr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	SUB OP - FLAGS AFFECTED

ACCUMULATOR GROUP INSTRUCTIONS

1 0	P P P	S S S	ALU OP r	(5)	PC_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 STOCK CONTROL INSTRUCTIONS

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

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

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_L OUT (8)	PC_H OUT	DATA TO REG. b	X	REG. b TO DDD					
PC_L OUT (10)	PC_H OUT	DATA TO REG. b	→		REG. L OUT	REG. H OUT	REG. b TO OUT		

REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X	ALU OP - FLAGS AFFECTED					
PC_L OUT (8)	PC_H OUT	DATA TO REG. b	X	ARITH OP - FLAGS AFFECTED					

PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	HIGHER ADD. REG. a (12)	REG. a TO PC_H	REG. b TO PC_L
PC_L OUT (8)	PC_H OUT	LOWER ADD. TO REG. b	→		PC_L OUT	PC_H OUT	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.

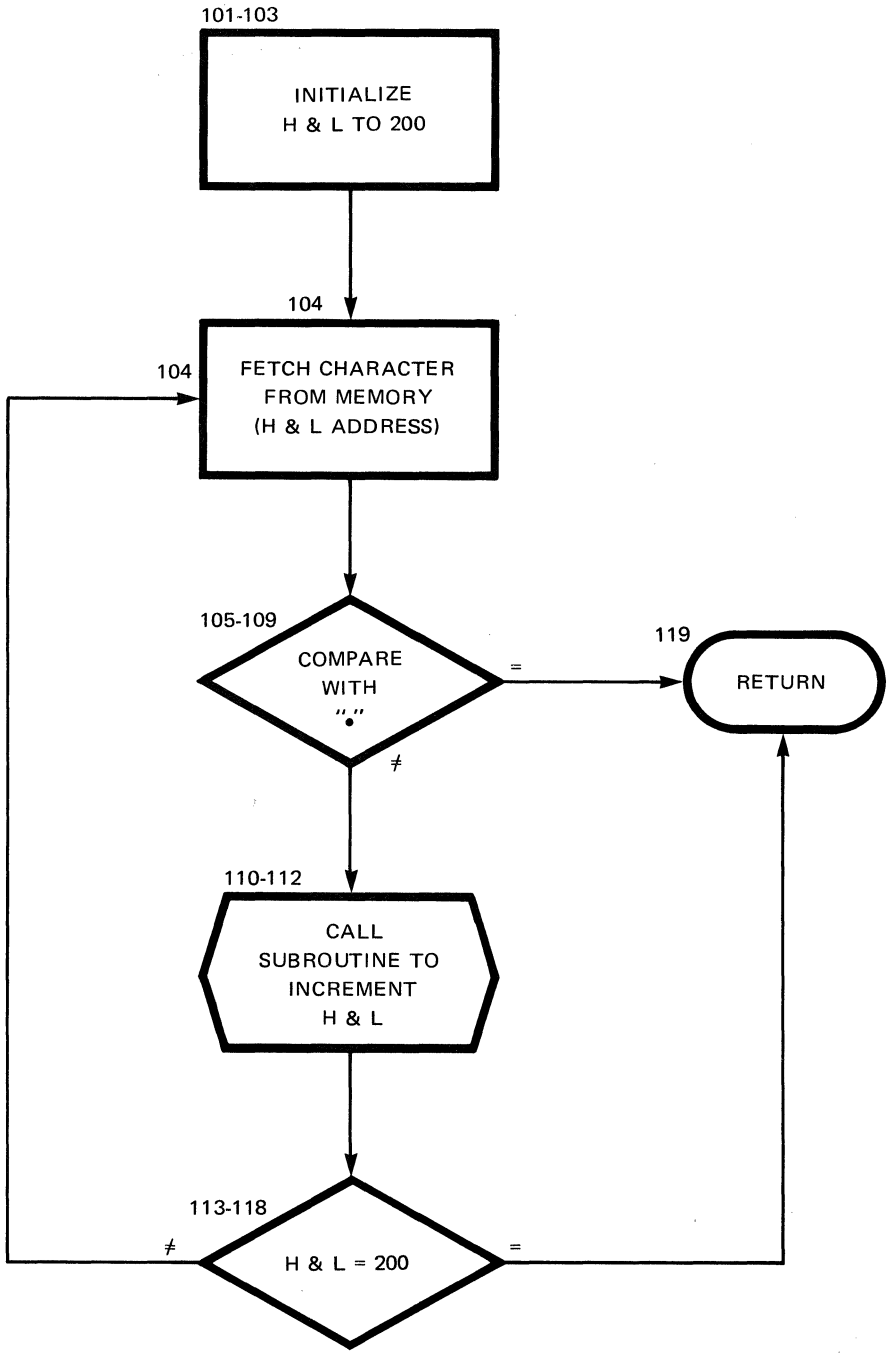


Figure 17. Subroutine to Search For Period

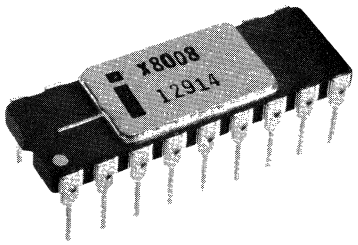
APPENDIX III

PROGRAMMING EXAMPLE

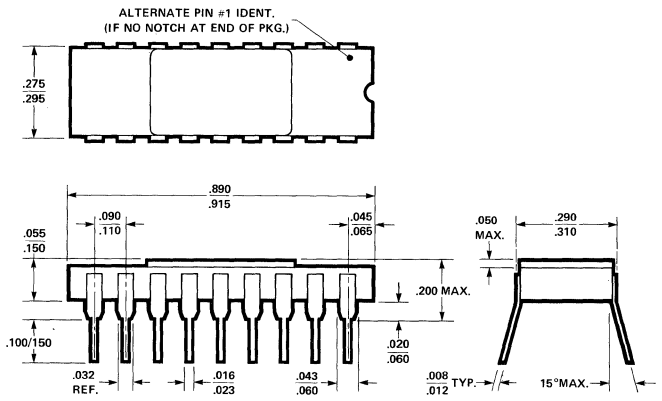
Sample Program To Search A String Of Characters
In Memory Locations 200-220 For A Period (.)

MNEMONIC	OPERAND	EXPLANATION	BYTES	LOCATION	ROM CODE	COMMENT
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	(•)
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	

Packaging Information



CERAMIC PACKAGE OUTLINE



Ordering Information

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

(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). Contact Intel regarding availability.

(3) **Memory Expansion**

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

RAM 1101	ROM 1702/1602	SR 1404
1103	1301	2401

Complete RAM memory systems are available from Intel's Memory System Group. Specify

in-14	in-16	in-20
8K x 8	4K x 8	1K x 8

(4) **MP7-02 ROM Programmer**

This is the programmer board for 1601/1701 or 1602/1702. The three 1602 control ROMs used with the SIM8-01 for an automatic programming system are specified by pattern numbers A0660, A0661, A0662. Contact Intel regarding availability.

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) 925-3144
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) Lrl	(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.

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.
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 to the accumulator with carry.
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.
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.
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

(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 $B_3 \ldots B_3B_2 \ldots B_2$.
(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 $B_3 \ldots B_3B_2 \ldots B_2$ 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 $B_3 \ldots B_3B_2 \ldots B_2$ 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 $B_3 \ldots B_3B_2 \ldots B_2$. 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 $B_3 \ldots B_3B_2 \ldots B_2$ 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 $B_3 \ldots B_3B_2 \ldots B_2$ 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.
RES	(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, 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), zero (01), sign (10), parity (11).

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

Printed in U.S.A.