

DEPT 4093

MF8008

central processing unit

applications manual



C O N T E N T S

MF8008	COMPONENT DESCRIPTION	P. 1
MOD 8	MODULAR MICRO COMPUTER	P. 31
MONITOR 8	SOFTWARE USERS GUIDE	P. 67
MONITOR 8	SOFTWARE LISTING	P. 75

THIS APPLICATIONS MANUAL IS PUBLISHED AS A GUIDE FOR DESIGNERS, CIRCUIT DIAGRAMS AND SOFTWARE LISTINGS SHOWN ILLUSTRATE TYPICAL APPLICATIONS ONLY, AND NO RESPONSIBILITY WILL BE ASSUMED FOR ANY CONSEQUENCE OF THEIR USE. THIS IS A PRELIMINARY RELEASE OF THIS MANUAL. WHERE THE DESIGNATION 8008 IS USED THE MICROSYSTEMS INTERNATIONAL LTD. PRODUCT MF8008 IS INTENDED.

1. INTRODUCTION

THE 8008 IS A SINGLE CHIP MOS 8-BIT CENTRAL PROCESSOR UNIT FOR MICRO COMPUTER SYSTEMS. 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 THE MF1101, MF1103, MF2102 (RAMS), MF1302, MF1602, MF1702 (ROMS).

THE PROCESSOR COMMUNICATES OVER AN 8-BIT DATA AND ADDRESS BUS (DO THROUGH D7) AND USES TWO INPUT LEADS (READY AND INTERRUPT) AND FOUR OUTPUT LEADS (SO, S1, S2 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, AND 8-BIT ACCUMLATOR, 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).

THE CONTROL PORTION OF THE CHIP CONTAINS LOGIC TO IMPLEMENT A VARIETY OF REGISTER TRANSFER, ARITHMETIC CONTROL, AND LOGICAL INSTRUCTIONS. MOST INSTRUCTIONS ARE CODED IN ONE BYTE (8 BITS); DATA IMMEDIATE INSTRUCTIONS USE TWO BYTES; JUMP INSTRUCTIONS UTILIZE THREE BYTES. OPERATING WITH A 500 KHZ CLOCK, THE 8008 CPU EXECUTES NON-MEMORY REFERENCING INSTRUCTIONS IN 20 MICROSECONDS. A SELECTED DEVICE, THE 8008-1, EXECUTES NON-MEMORY REFERENCING INSTRUCTIONS IN 12.5 MICROSECONDS WHEN OPERATING FROM AN 800 KHZ CLOCK.

ALL INPUTS (INCLUDING CLOCKS) ARE TTL COMPATIBLE AND ALL OUTPUTS ARE LOW-POWER TTL COMPATIBLE.

THE INSTRUCTION SET OF THE 8008 CONSISTS OF 48 INSTRUCTIONS INCLUDING DATA MANIPULATION, BINARY ARITHMETIC, AND JUMP TO SUBROUTINE.

THE NORMAL PROGRAM FLOW OF THE 8008 MAY BE INTERRUPTED THROUGH THE USE OF THE INTERRUPT CONTROL LINE. THIS ALLOWS THE SERVICING OF SLOW I/O PERIPHERAL DEVICES WHILE ALSO EXECUTING THE MAIN PROGRAM.

THE READY COMMAND LINE SYNCHRONIZES THE 8008 TO THE MEMORY CYCLE ALLOWING ANY TYPE OR SPEED OF SEMICONDUCTOR MEMORY TO BE USED.

STATE AND SYNC OUTPUTS INDICATE THE STATE OF THE PROCESSOR AT ANY TIME IN THE INSTRUCTION CYCLE.

2. 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.) 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 S0, S1 AND S2, ALONG WITH SYNC INFORM THE PERIPHERAL CIRCUITRY OF THE STATE OF THE PROCESSOR. THE BINARY STATE CODES AND THE DESIGNATED STATE NAMES ARE AS SHOWN.

S0	S1	S2	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. FIGURE 1 ILLUSTRATES THE PROCESSOR ACTIVITY DURING A SINGLE CYCLE.

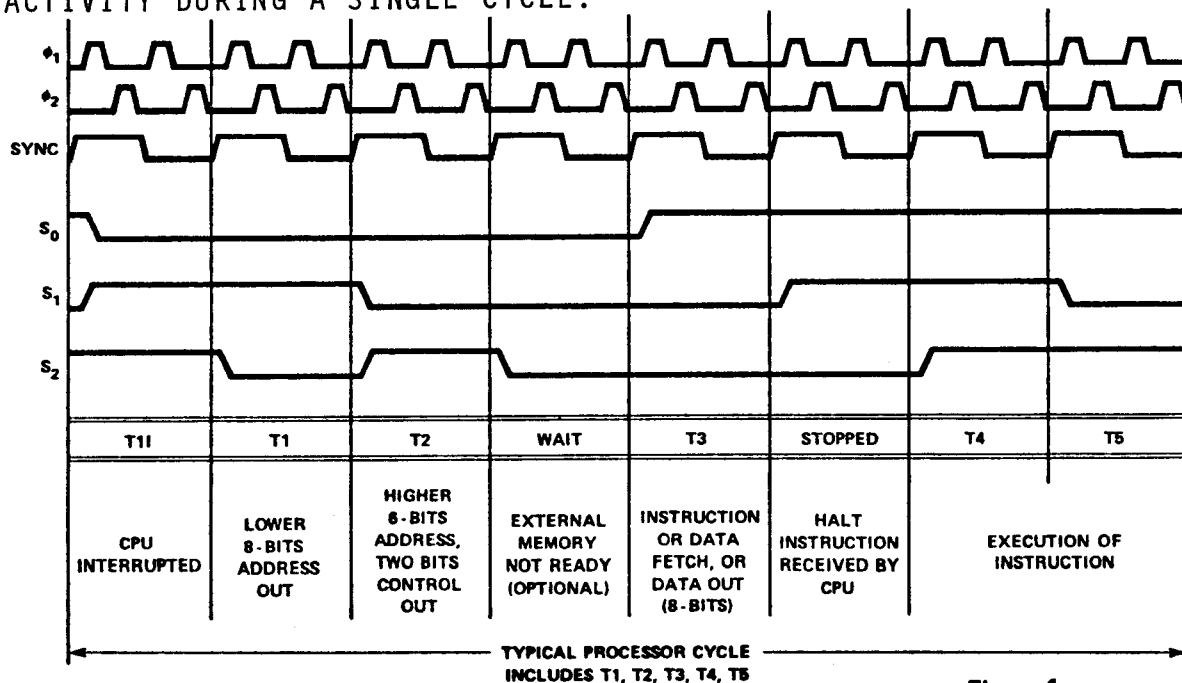
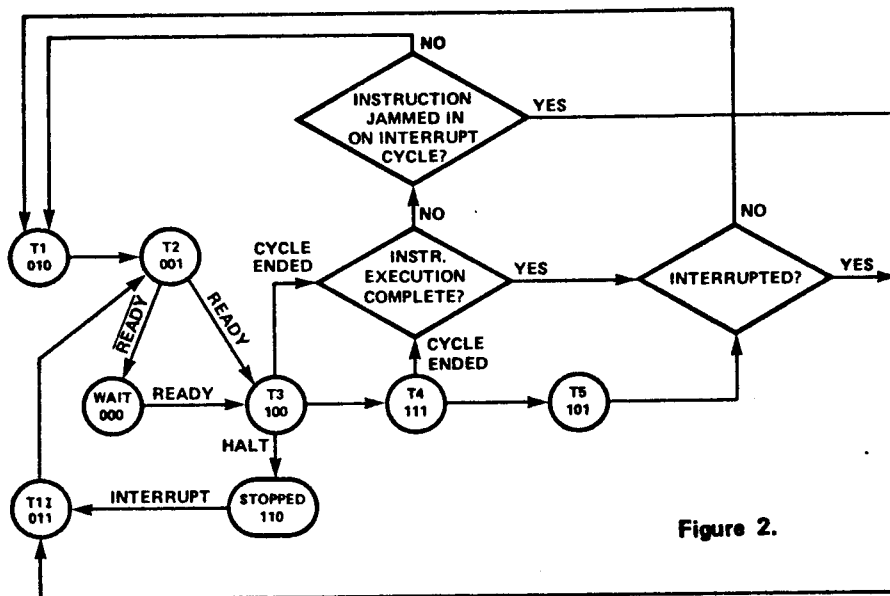


Figure 1.

THE RECEIPT OF AN INTERRUPT IS ACKNOWLEDGED BY THE T11. 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 AND STOPPED STATES 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.



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, D6 AND D7 WHICH ARE PRESENT ON THE DATA BUS DURING T2.

D6	D7	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.

3. BASIC FUNCTIONAL BLOCKS

THE FOUR BASIC FUNCTIONAL BLOCKS OF THE 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 A 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 CARY 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 REGISTERS, 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.

4. BASIC INSTRUCTION SET

THE FOLLOWING SECTION PRESENTS THE BASIC INSTRUCTION SET OF THE 8008.

A. DATA AND INSTRUCTION FORMATS

DATA IN THE 8008 IS STORED IN THE FORM OF 8-BIT BINARY INTEGERS. ALL DATA TRANSFERS TO THE SYSTEM DATA BUS WILL BE IN THE SAME FORMAT.

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

DATA WORD

THE PROGRAM INSTRUCTIONS MAY BE ONE, TWO, OR THREE BYTES IN LENGTH. MULTIPLE BYTE INSTRUCTIONS MUST BE STORED IN SUCCESSIVE WORDS IN PROGRAM MEMORY. THE INSTRUCTION FORMATS THEN DEPEND ON THE PARTICULAR OPERATION EXECUTED.

ONE BYTE INSTRUCTIONS

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

OP CODE

TYPICAL INSTRUCTIONS

REGISTER TO REGISTER, MEMORY REFERENCE, I/O ARITHMETIC OR LOGICAL, ROTATE OR RETURN INSTRUCTIONS

TWO BYTE INSTRUCTIONS

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

OP CODE

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

OPERAND

IMMEDIATE MODE INSTRUCTIONS

THREE BYTE INSTRUCTIONS

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

OP CODE

D7	D6	D5	D4	D3	D2	D1	D0
----	----	----	----	----	----	----	----

LOW ADDRESS JUMP OR CALL INSTRUCTION

X	X	D5	D4	D3	D2	D1	D0
---	---	----	----	----	----	----	----

HIGH ADDRESS*

*FOR THE THIRD BYTE OF THIS INSTRUCTION, D6 AND D7 ARE "DON'T CARE" BITS.

A LOGIC "1" IS DEFINED AS A HIGH LEVEL AND A LOGIC "0" IS DEFINED AS A LOW LEVEL.

B. SUMMARY OF PROCESSOR INSTRUCTIONS

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			DESCRIPTION OF OPERATION
		D7D6	D5D4D3	D2D1D0	
LR1R2	(5)	1 1	D D D	S S S	LOAD INDEX REGISTER R1 WITH THE CONTENT OF INDEX REGISTER R2.
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
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#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. 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 TO 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 D7D6 D5D4D3 D2D1D0	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 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.

JMP	(11)	0 1 X X X 1 0 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	UNCONDITIONALLY JUMP TO MEMORY ADDRESS B3...B3B2...B2.
JFC	(9 OR 11)	0 1 0 C4C3 0 0 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	JUMP TO MEMORY ADDRESS B3...B3B2...B2 IF THE CONDITION FLIP FLOP C IS FALSE. OTHERWISE, EXECUTE THE NEXT INSTRUCTION IN SEQUENCE.
JTC	(9 OR 11)	0 1 1 C4C3 0 0 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	JUMP TO MEMORY ADDRESS B3...B3B2...B2 IF THE CONDITION FLIP FLOP C IS TRUE. OTHERWISE, EXECUTE THE NEXT INSTRUCTION IN SEQUENCE.
CAL	(11)	0 1 X X X 1 1 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	UNCONDITIONALLY CALL THE SUBROUTINE AT MEMORY ADDRESS B3...B3B2...B2. SAVE THE CURRENT ADDRESS (UP ONE LEVEL IN THE STACK.)
CFC	(9 OR 11)	0 1 0 C4C3 0 1 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	CALL THE SUBROUTINE AS MEMORY ADDRESS B3...B3B2...B2 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 1 C4C3 0 1 0 B2B2 B2B2B2 B2B2B2 X X B3B3B3 B3B3B3	CALL THE SUBROUTINE AT MEMORY ADDRESS B3...B3B2...B2 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 C4C3 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 C4C3 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 (RRMM, 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:

SSS = SOURCE INDEX REGISTER

DDD = DESTINATION INDEX REGISTER

THESE REGISTERS, R1, ARE DESIGNATED
 A (ACCUMULATOR -- 000),
 B (001), C(010), D(011), E(100), H(101)
 L (110).

MEMORY REGISTERS ARE ADDRESSED BY THE CONTENTS OF REGISTERS H&L.

ADDITIONAL BYTES ON INSTRUCTION ARE DESIGNATED BY BBBB BBBB.

X = "DON'T CARE".

FLAG FLIP-FLOPS ARE DEFINED BY C4C3: CARRY (00-OVERFLOW OR UNDERFLOW), ZERO (01-RESULT IS ZERO), SIGN (10-MSB OF RESULT IS "1").

PARITY (11-PARITY IS EVEN).

C. COMPLETE FUNCTIONAL DEFINITION

THE FOLLOWING PAGES PRESENT A DETAILED DESCRIPTION OF THE COMPLETE 8008 INSTRUCTION SET.

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
C4C3	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
^	LOGICAL PRODUCT
✕	EXCLUSIVE "OR"
V	INCLUSIVE "OR"
AM	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 # REGISTER NAME
	(SSS OR DDD)
	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.

LR1R2 (ONE CYCLE - PCI)	11	DDD	SSS	(R1) \leftarrow (R2) LOAD REGISTER R1 WITH THE CONTENT OF R2. THE CONTENT OF R2 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 \neq 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 \neq 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	(R) \leftarrow <B2> LOAD BYTE TWO OF THE INSTRUCTION INTO REGISTER R.
LMI (THREE CYCLES - PCI/PCR/PCW)	00	111	110	(M) \leftarrow <B2> 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 \neq 000 (HALT INSTR.) AND DDD \neq 111 (CONTENT OF MEMORY CANNOT 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 \neq 000 (HALT INSTR.) AND DDD \neq 111 (CONTENT OF MEMORY CANNOT 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	$(A) \leftarrow (A) + (R) + (\text{CARRY})$ 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.
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	$(A) \leftarrow (A) \vee (R)$ 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 ($A=R$) IS INDICATED BY THE ZERO FLIP-FLOP SET TO "1". LESS THAN ($A < R$) IS INDICATED BY THE CARRY FLIP-FLOP, SET TO "1".

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	$(A) \leftarrow (A) \vee (M)$ 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	00 000 100 <B2>	$(A) \leftarrow (A) + \langle B2 \rangle$
ACI	00 001 100 <B2>	$(A) \leftarrow (A) + \langle B2 \rangle + (\text{CARRY})$
SUI	00 010 100 <B2>	$(A) \leftarrow (A) - \langle B2 \rangle$ SUBTRACT
SBI	00 011 100 <B2>	$(A) \leftarrow (A) - \langle B2 \rangle - (\text{BORROW})$ SUBTRACT WITH BORROW
NDI	00 100 100 <B2>	$(A) \leftarrow (A) \wedge \langle B2 \rangle$ LOGICAL AND
XRI	00 101 100 <B2>	$(A) \leftarrow (A) \vee \langle B2 \rangle$ EXCLUSIVE OR
ORI	00 110 100 <B2>	$(A) \leftarrow (A) \vee \langle B2 \rangle$ INCLUSIVE OR
CPI	00 111 100 <B2>	$(A) - \langle B2 \rangle$ 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	AM+1←AM, A0←A7, (CARRY)←A7 ROTATE THE CONTENT OF REGISTER A LEFT ONE BIT. ROTATE A7 INTO A0 AND INTO THE CARRY FLIP-FLOP.
RRC	00	001	010	AM←AM+1, A7←A0, (CARRY)←A0 ROTATE THE CONTENT OF REGISTER A RIGHT ONE BIT. ROTATE A0 INTO A7 AND INTO THE CARRY FLIP-FLOP.
RAL	00	010	010	AM+1←AM, A0←(CARRY), (CARRY)←A7 ROTATE THE CONTENT OF REGISTER A LEFT ONE BIT. ROTATE THE CONTENT OF THE CARRY FLIP-FLOP INTO A0. ROTATE A7 INTO THE CARRY FLIP-FLOP.
RAR	00	011	010	AM←AM+1, A7←(CARRY), (CARRY)←A0 ROTATE THE CONTENT OF REGISTER A RIGHT ONE BIT. ROTATE THE CONTENT OF THE CARRY FLIP-FLOP INTO A7. ROTATE A0 INTO THE CARRY FLIP-FLOP.

PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS

JUMP INSTRUCTIONS - THREE BYTES

(THREE CYCLES - PCI/PCR/PCR)

NORMAL FLOW OF THE PROGRAM MAY BE ALTERED TO AN ADDRESS SPECIFIED BY BYTES TWO AND THREE OF AN INSTRUCTION.

JMP	01	XXX	100	(P)←<B3><B2> JUMP UNCONDITIONALLY TO THE INSTRUCTION LOCATED IN MEMORY LOCATION ADDRESSED BY BYTE TWO AND BYTE THREE.
(JUMP UNCONDITIONALLY)		<B2>	<B3>	
JFC	01	0C4C3	000	IF (C) = 0, (P)←<B3><B2>. OTHERWISE (P) = (P)+3. IF THE CONTENT OF FLIP-FLOP C IS ZERO, THEN JUMP TO THE INSTRUCTION LOCATED IN MEMORY LOCATION <B3><B2>; OTHERWISE, EXECUTE THE NEXT INSTRUCTION IN SEQUENCE.
(JUMP IF CONDITION FALSE)		<B2>	<B3>	
JTC	01	1C4C3	000	IF (C) = 1, (P)←<B3><B2>. OTHERWISE, (P) = (P)+3. IF THE CONTENT OF FLIP-FLOP C IS ONE, THEN JUMP TO THE INSTRUCTION LOCATED IN MEMORY LOCATION <B3><B2>; OTHERWISE, EXECUTE THE NEXT INSTRUCTION IN SEQUENCE.
(JUMP IF CONDITION TRUE)		<B2>	<B3>	

CALL INSTRUCTIONS - THREE BYTES

(THREE CYCLES - PCI/PCR/PCR)

SUBROUTINES MAY BE CALLED AND NESTED UP TO SEVEN LEVELS.

CAL 01 XXX 110 (STACK) \leftarrow (P), (P) \leftarrow <B3><B2>. SHIFT THE CONTENT P
 (CALL SUBROUTINE TO THE PUSHDOWN STACK. JUMP UNCONDITIONALLY TO
 UNCONDITIONALLY) THE INSTRUCTION LOCATED IN MEMORY LOCATION
 <B2> ADDRESSED BY BYTE TWO AND BYTE THREE.
 <B3>

CFC 01 0C4C3 010 IF (C) = 0, (STACK) \leftarrow (P), (P) \leftarrow <B3><B2>. OTHERWISE,
 (CALL SUBROUTINE (P) = (P)+3. IF THE CONTENT OF FLIP-FLOP C IS
 IF CONDITION FALSE) ZERO, THEN SHIFT CONTENTS OF P TO THE PUSHDOWN
 <B2> STACK AND JUMP TO THE INSTRUCTION LOCATED IN
 <B3> MEMORY LOCATION<B3><B2>; OTHERWISE, EXECUTE THE
 NEXT INSTRUCTION IN SEQUENCE.

CTC 01 1C4C3 010 IF (C) = 1, (STACK) \leftarrow (P), (P) \leftarrow <B3><B2>. OTHERWISE,
 (CALL SUBROUTINE (P) = (P)+3. IF THE CONTENT OF FLIP-FLOP C IS
 IF CONDITION TRUE) ONE, THEN SHIFT CONTENTS OF P TO THE PUSHDOWN
 <B2> STACK AND JUMP TO THE INSTRUCTION LOCATED IN
 <B3> MEMORY LOCATION<B3><B2>; OTHERWISE, EXECUTE THE
 NEXT INSTRUCTION IN SEQUENCE.

IN THE ABOVE JUMP AND CALL INSTRUCTIONS <B2> CONTAINS THE LEAST SIGNIFICANT HALF OF THE ADDRESS AND <B3> CONTAINS THE MOST SIGNIFICANT HALF OF THE ADDRESS. NOTE THAT D6 AND D7 OF <B3> 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) \leftarrow (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 00 0C4C3 011 IF (C) = 0, (P) \leftarrow (STACK); OTHERWISE, (P) = (P)+1.
 (RETURN CONDITION IF THE CONTENT OF FLIP-FLOP C IS ZERO, THEN RETURN
 FALSE) 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 00 1C4C3 011 IF (C) = 1, (P) (STACK); OTHERWISE, (P) = (P)+1.
 (RETURN CONDITION IF THE CONTENT OF FLIP-FLOP C IS ONE, THEN RETURN
 TRUE) 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-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 D0, D1, D2, D3 RESPECTIVELY AT T4 OF 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

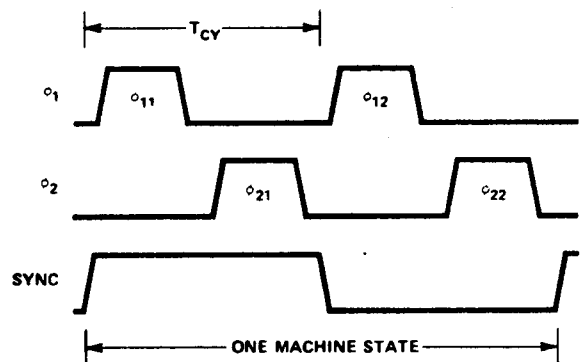
ON RECEIPT OF THE HALT INSTRUCTION, THE ACTIVITY OF THE PROCESSOR IS IMMEDIATELY SUSPENDED IN THE STOPPED STATE. THE CONTENT OF ALL REGISTERS AND MEMORY IS UNCHANGED. THE P-COUNTER HAS BEEN UPDATED AND THE INTERNAL DYNAMIC MEMORIES CONTINUE TO BE REFRESHED.

D. INTERNAL PROCESSOR OPERATION

INTERNALLY THE PROCESSOR OPERATES THROUGH FIVE DIFFERENT STATES:

	INTERNAL STATE	TYPICAL FUNCTION
T1	NORMAL INTERRUPT	SEND OUT LOWER EIGHT BITS OF ADDRESS AND INCREMENT PROGRAM COUNTER. SEND OUT LOWER EIGHT BITS OF ADDRESS AND SUPPRESS INCREMENTING OF PROGRAM COUNTER AND ACKNOWLEDGE INTERRUPT.
T2		SEND OUT SIX HIGHER ORDER BITS OF ADDRESS AND TWO CONTROL BITS, D6 AND D7. INCREMENT PROGRAM COUNTER IF THERE HAS BEEN A CARRY FROM T1.
T3	WAIT NORMAL STOPPED	WAIT FOR READY SIGNAL TO COME TRUE. REFRESH INTERNAL DYNAMIC MEMORIES WHILE WAITING. FETCH AND DECODE INSTRUCTION; FETCH DATA FROM MEMORY; OUTPUT DATA TO MEMORY. REFRESH INTERNAL MEMORIES. REMAIN STOPPED UNTIL INTERRUPT OCCURS. REFRESH INTERNAL MEMORIES.
T4 AND T5		EXECUTE INSTRUCTION AND APPROPRIATELY TRANSFER DATA WITHIN PROCESSOR. CONTENT OF DATA BUS TRANSFER IS AVAILABLE AT I/O BUS FOR CONVENIENCE IN TESTING. SOME CYCLES DO NOT REQUIRE THESE STATES. IN THOSE CASES, THE STATES ARE SKIPPED AND THE PROCESSOR GOES DIRECTLY TO T1.

THE 8008 IS DRIVEN BY TWO NON-OVERLAPPING CLOCKS. TWO CLOCK PERIODS ARE REQUIRED FOR EACH STATE OF THE PROCESSOR. ϕ_1 IS GENERALLY USED TO PRECHARGE ALL DATA LINES AND MEMORIES AND ϕ_2 CONTROLS ALL DATA TRANSFERS WITHIN THE PROCESSOR. A SYNC SIGNAL (DIVIDE BY TWO OF ϕ_2) IS SENT OUT BY THE 8008. THIS SIGNAL DISTINGUISHES BETWEEN THE TWO CLOCK PERIODS OF EACH STATE.



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 AS 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.

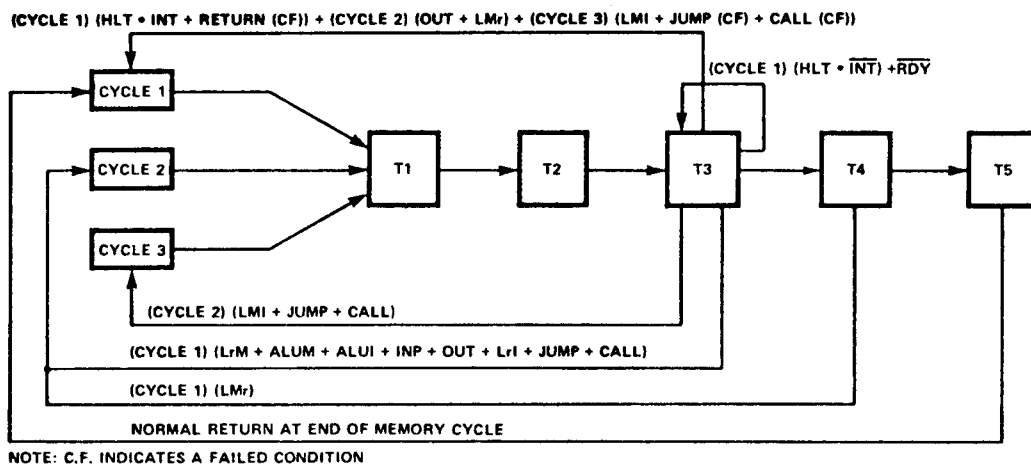


Figure 4.

5. 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 $SOSIS2 = 011$ AT T11 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 CPU BOARD.

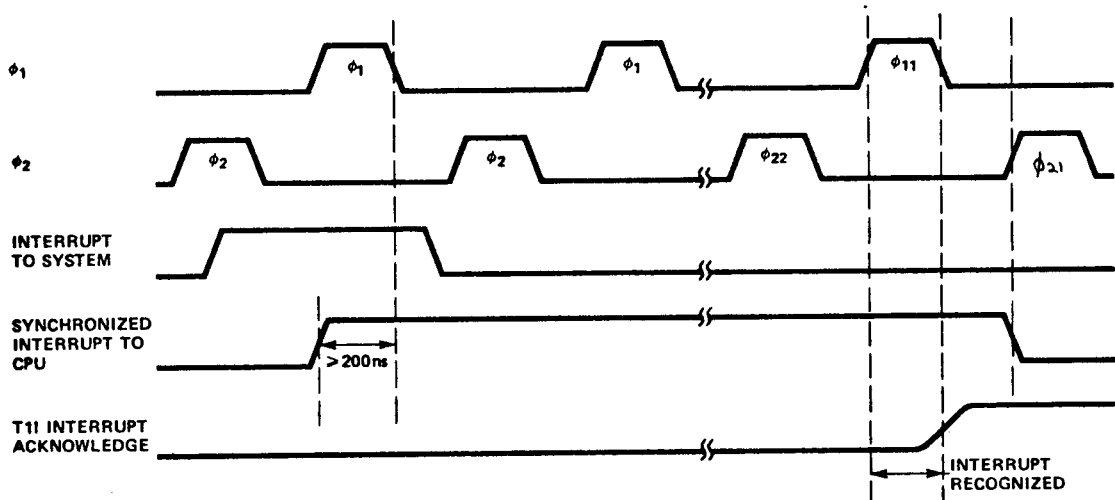
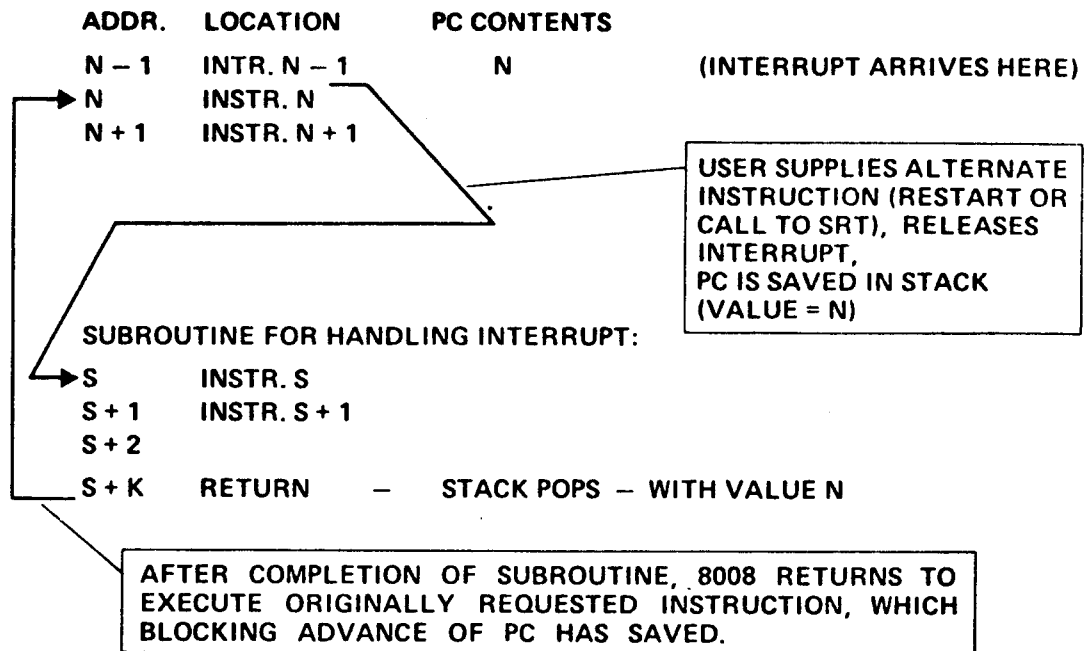


Figure 5.

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.



2) START-UP OF THE 8008

WHEN POWER (VDD) AND CLOCKS ($\phi 1$, $\phi 2$) ARE TURNED ON, A FLIP-FLOP INTERNAL TO THE 8008 IS SET BY SENSING THE RISE OF VDD. 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 T11 AT SOME TIME LATER THAN $\phi 11$. NOTE THAT WHENEVER THE 8008 IS IN A T11 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 1 0 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 0	INSTR ₁	
	0 0 0 0 0 0 0 0 X Y Z 0 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 0	INSTR _a	
	0 0 0 0 0 0 0 0 X Y Z 0 0 0 1	INSTR _b	
	⋮	⋮	
	0 0 0 0 0 0 0 0 n n n n n n	RETURN	} Main Program
	0 0 0 0 0 0 0 0 0 0 0 0 0 0	INSTR ₁ (INSTR ₁ executed now)	
	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.

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	INSTR1	
	0 0 0 0 0 0 0 0 0 0 0 1 0 0	INSTR2	
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 0	INSTR1	
	0 0 0 0 0 0 0 0 X Y Z 0 0 0 1	INSTR2	
	.	.	
	.	.	
	.	.	

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	INSTR1 (RST=00 XYZ 101)	START-UP ROUTINE
	0 0 0 0 0 0 0 0 X Y Z 0 0 0 0	INSTRA	
	0 0 0 0 0 0 0 0 X Y Z 0 0 0 1	INSTRB	
	.	.	
	.	.	
	.	.	
	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	INSTR1 (INSTR1 EXECUTED NOW)	MAIN PROGRAM
	0 0 0 0 0 0 0 0 0 0 0 0 0 1	INSTR2	
	.	.	
	.	.	
	.	.	

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.

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 Vcc) 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.

6. 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.

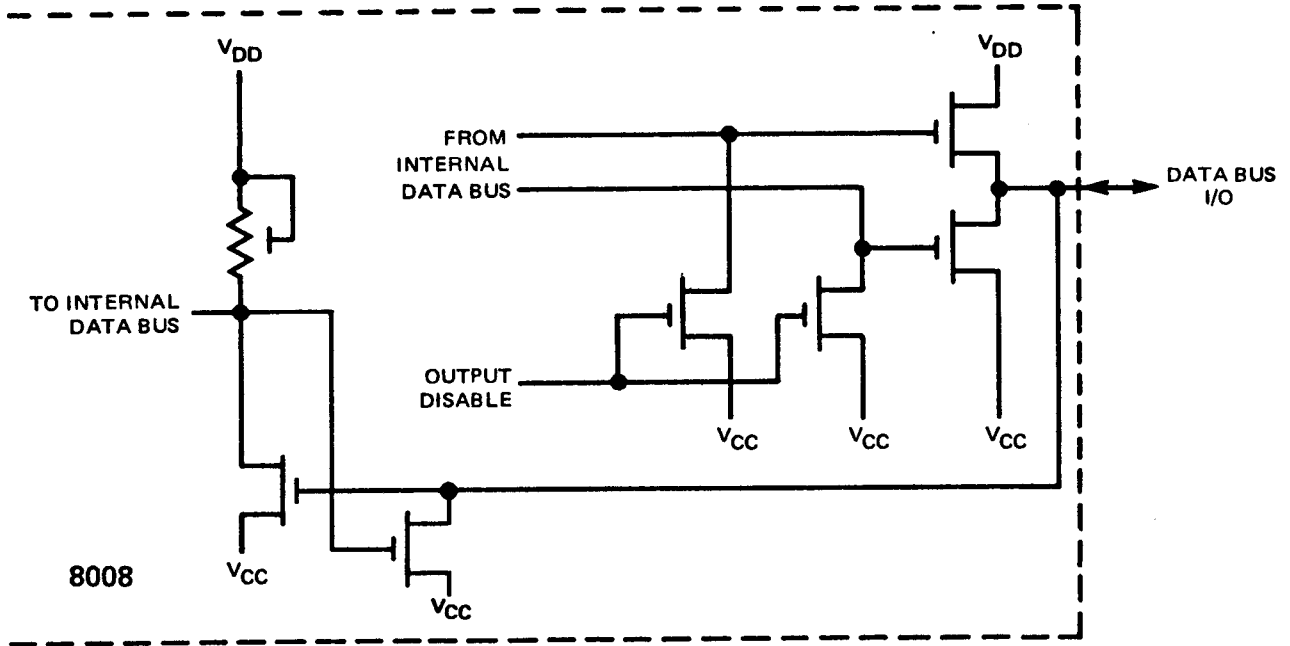


Figure 6.

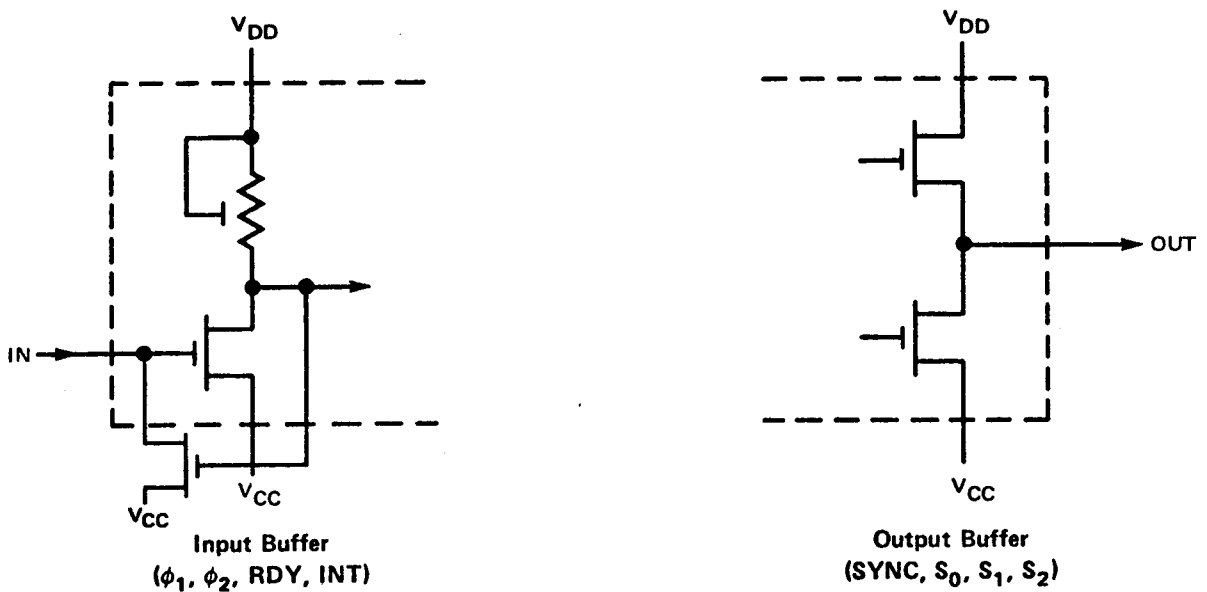


Figure 7.

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			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

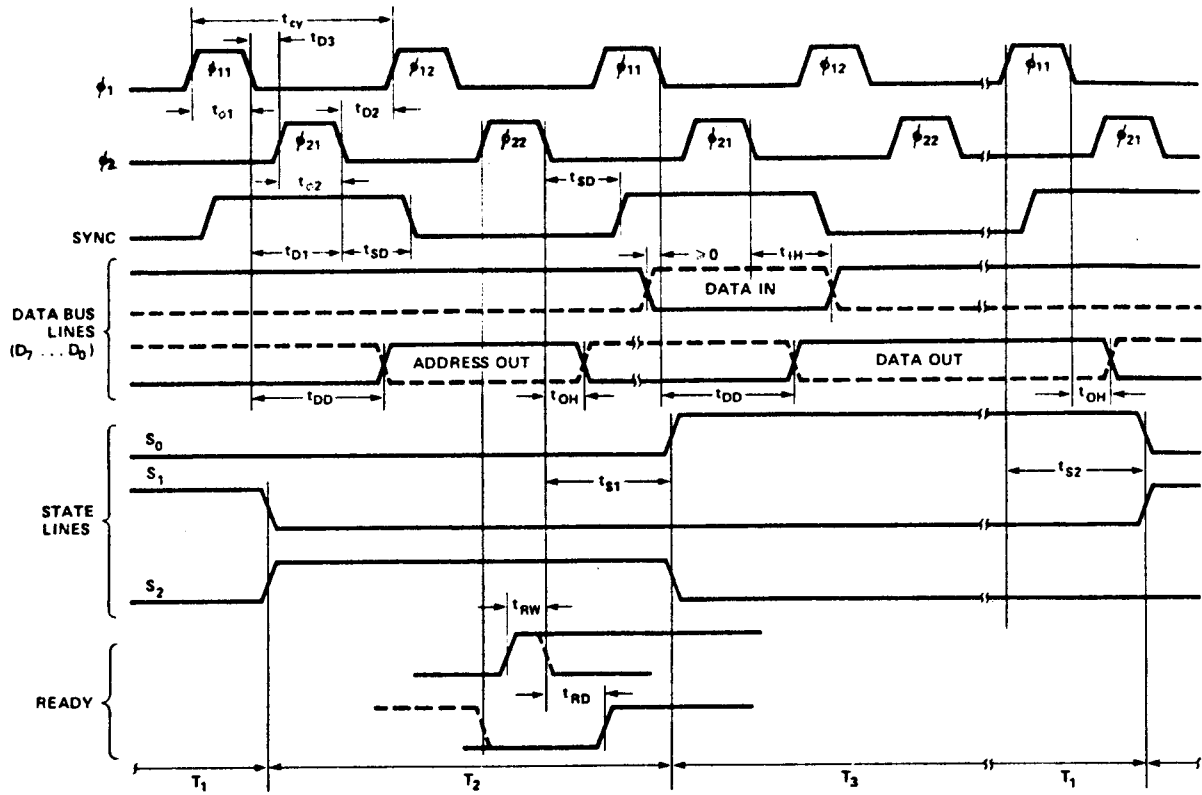
*Measurements are made while the 8008 is executing a typical sequence of instructions. The test load is selected such that at V_{OL} = 0.4V, I_{OL} = 0.44mA on each output.

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.

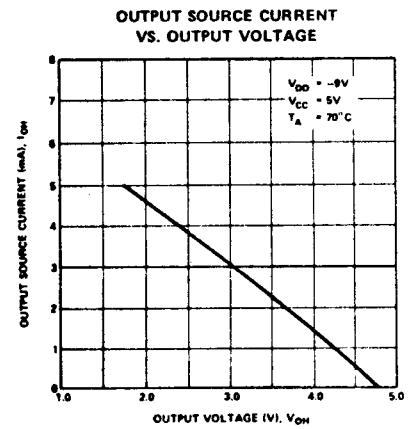
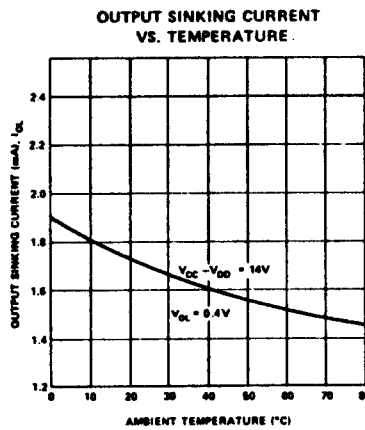
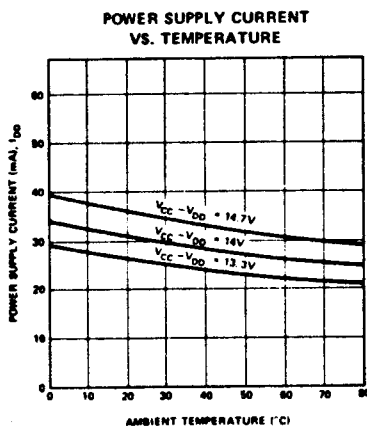
SYMBOL	PARAMETER	8008		8008-1		UNIT	TEST CONDITIONS
		LIMITS		LIMITS			
		MIN.	MAX.	MIN.	MAX.		
t _{CY}	CLOCK PERIOD	2	3	1.25	3	μs	t _R , t _F = 50ns
t _R , t _F	CLOCK RISE AND FALL TIMES		50		50	ns	
t _{φ1}	PULSE WIDTH OF φ ₁	.70		.35		μs	
t _{φ2}	PULSE WIDTH OF φ ₂	.55		.35		μs	
t _{D1}	CLOCK DELAY FROM FALLING EDGE OF φ ₁ TO FALLING EDGE OF φ ₂	.90	1.1		1.1	μs	
t _{D2}	CLOCK DELAY FROM φ ₂ TO φ ₁	.40		.35		μs	
t _{D3}	CLOCK DELAY FROM φ ₁ TO φ ₂	.20		.20		μs	
t _{DD}	DATA OUT DELAY		1.0		1.0	μs	C _L = 100pF
t _{OH}	HOLD TIME FOR DATA BUS OUT	.10		.10		μs	
t _{IH}	HOLD TIME FOR DATA IN					μs	
t _{SD}	SYNC OUT DELAY		.70		.70	μs	C _L = 100pF
t _{S1}	STATE OUT DELAY (ALL STATES EXCEPT T1 AND T11)		1.1		1.1	μs	C _L = 100pF
t _{S2}	STATE OUT DELAY (STATES T1 AND T11)		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	

TIMING DIAGRAM

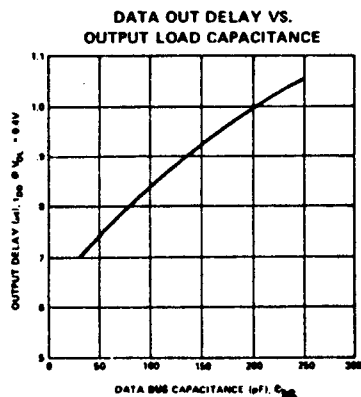


READY line must be at "0" prior to ϕ_{22} of T_2 to guarantee entry into the WAIT state.

TYPICAL D. C. CHARACTERISTICS



TYPICAL A. C. CHARACTERISTICS



CAPACITANCE $f = 1MHz; T_A = 25^\circ C; \text{Unmeasured Pins Grounded}$

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

7. MOD 8, AN 8008 BASED MODULAR MICRO COMPUTER.

THE MOD 8 IS A MODULAR 8008 BASED PROTOTYPING SYSTEM. THIS HARDWARE IN CONJUNCTION WITH THE MONITOR 8 SOFTWARE DESCRIBED IN SECTION 9 ALLOWS THE USER TO DEVELOP HIS OWN HARDWARE SYSTEM TO CATER TO HIS PARTICULAR REQUIREMENTS. THE BASIC MOD 8 CONFIGURATION CONSISTS OF SEVEN 4-1/2" x 6" PRINTED CIRCUIT BOARDS WITH THE FOLLOWING FUNCTIONS.

- 7.1 MOD 8 CPU BOARD. THIS BOARD CONTAINS THE 8008 CPU, CLOCK GENERATORS, STATE DECODING AND BUS SWITCHING CONTROL LOGIC.
- 7.2 MOD 8 RESTART, TTY I/O BOARD. THIS BOARD CONTAINS TELETYPE I/O, READER CONTROL AND SYSTEM RESTART LOGIC.
- 7.3 MOD 8 CONTROL/BUFFER BOARD. THIS BOARD CONTAINS 8 BIT BI-DIRECTIONAL BUS SWITCHES AND ADDRESS LATCHES.
- 7.4 MOD 8 ROM BOARD. THIS BOARD CONTAINS 2K x 8 OF 1702 pROM OR 1302 MASK PROGRAMMABLE ROM.
- 7.5 MOD 8 RAM BOARD. THIS BOARD CONTAINS 2K x 8 OF 2102 RAM.
- 7.6 MOD 8 INPUT BOARD. THIS BOARD CONTAINS THREE 8 BIT INPUT CHANNELS.
- 7.7 MOD 8 OUTPUT BOARD. THIS BOARD CONTAINS THREE 8 BIT OUTPUT CHANNELS.

THE MOD 8 SYSTEM DECODES THE TIME DIVISION MULTIPLEXED SIGNALS FROM THE 8008 TO PROVIDE A 14 BIT PARALLEL ADDRESS BUS AND AN 8 BIT BI-DIRECTIONAL DATA BUS AT NORMAL TTL LEVELS. THIS MODULARLY EXPANDABLE BUS ORGANIZED STRUCTURE ALLOWS THE MEMORY CAPACITY AND TYPE AND THE NUMBER OF I/O PORTS TO BE TAILORED TO ANY PARTICULAR SYSTEM REQUIREMENT. THE OVERALL MOD 8 SYSTEM ORGANIZATION IS SHOWN IN FIGURE 8.

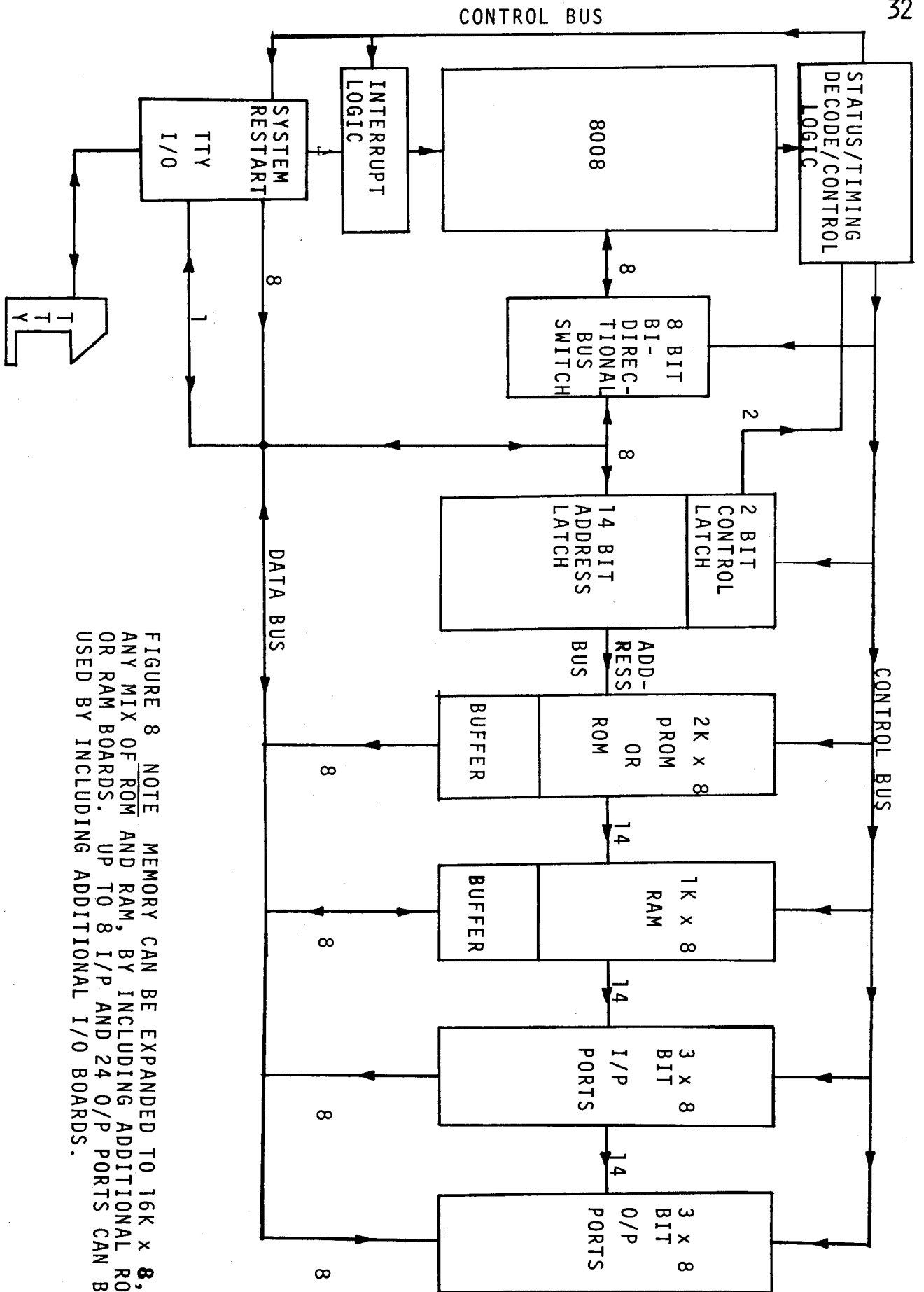


FIGURE 8 NOTE MEMORY CAN BE EXPANDED TO 16K x 8, ANY MIX OF ROM AND RAM, BY INCLUDING ADDITIONAL ROM OR RAM BOARDS. UP TO 8 I/P AND 24 O/P PORTS CAN BE USED BY INCLUDING ADDITIONAL I/O BOARDS.

8. MOD 8 DETAILED BOARD DESCRIPTIONS.

8.1 MOD 8 CPU BOARD

THE 8008 PROCESSOR (1) ON THIS BOARD REQUIRES 2 PHASE NON-OVERLAPPING CLOCKS. THESE CLOCKS ARE GENERATED USING 2 x 74123 DUAL MONOSTABLES. 16A AND 16B ARE CASCADED, WITH Q OF 16B CONNECTED BACK TO A OF 16A. THIS FORMS AN OSCILLATOR WITH A PERIOD EQUAL TO THE SUM OF THE DELAYS OF 16A AND 16B. THE OUTPUT OF 16B IS CONNECTED TO THE NEGATIVE EDGE TRIGGER POINT OF 15B AND THE POSITIVE EDGE TRIGGER POINT OF 15A. 15B PRODUCES THE ϕ_1 PULSE AND 15A THE ϕ_2 PULSE WHICH ARE THEN FED TO THE 8008. THE 8 DATA LINES OF THE 8008 ARE PROVIDED WITH EIGHT 22K Ω PULL-UP RESISTORS AND ARE ONLY LOADED WITH ONE LPTTL INPUT EACH. THE 8008 READY LINE NORMALLY IS TIED TO A LOGIC 1 VIA A 10K Ω RESISTOR. THIS MEANS THAT THE 8008 WILL NOT PAUSE AT T3. THE WAIT LINE IS HOWEVER BROUGHT OUT TO PIN 14 OF THE BOARD TO ALLOW THE USER TO RUN THE SYSTEM WITH SLOW MEMORY OR TO USE THE WAIT LINE TO SINGLE STEP THROUGH THE PROGRAM. THE SYNC SIGNAL IS BUFFERED OUT FROM THE 8008 VIA ONE LOW POWER INVERTER (8E) AND ONE MEDIUM POWER INVERTER 12B. THE THREE STATE LINES S0, S1, AND S2 ARE FED DIRECTLY TO A 6150 OR 3205 ONE OUT OF EIGHT DECODER (6). THIS DECODER ONLY PRESENTS A LOW POWER INPUT LOAD TO THE 8008. THE OUTPUTS OF THE DECODER ARE NORMALLY HIGH, ONLY GOING LOW FOR THE SELECTED OUTPUT. \overline{TT} SIGNAL FROM THE DECODER IS FED VIA 11A TO 11B WHERE IT IS COMBINED WITH $\overline{S.02}$. THUS THE OUTPUT OF 11B GOES LOW DURING ϕ_2 OF THE SECOND HALF OF THE T1 CYCLE. THIS SIGNAL \overline{ADLL} IS USED TO GATE THE LOWER 8 BITS OF THE ADDRESS, WHICH ARE VALID DURING T1, INTO THE ADDRESS LATCHES ON THE CONTROL BUFFER BOARD. \overline{ADLL} IS ALSO USED TO CLOCK 10B WHICH SAMPLES THE DECODER \overline{TTI} OUTPUT. \overline{TT} AND \overline{TTI} ARE COMBINED IN 11A AND WILL BOTH PRODUCE AN \overline{ADLL} SAMPLE PULSE. \overline{TTI} IS GENERATED IN PLACE OF \overline{TT} WHEN THE PROCESSOR HAS JUST BEEN INTERRUPTED. WHEN A \overline{TTI} STATE HAS BEEN GENERATED 10B WILL HOLD A LOW ON Q AND A HIGH ON \overline{Q} UNTIL IT IS RESTROBED AT THE START OF THE NEXT INSTRUCTION. 4D, 12C AND 4A GENERATE $\overline{S.02}$, S, \overline{S} AND $\overline{S.02}$ WHERE S = SYNC. DURING T2 \overline{ADHL} IS GENERATED BY 9D. THIS SIGNAL IS USED TO GATE THE UPPER 6 BITS OF THE ADDRESS PLUS CONTROL BITS CC1 AND CC2 INTO THE LATCHES ON THE CONTROL BUFFER BOARD. \overline{ADHL} IS ALSO USED TO CLOCK 10A WHICH WILL OUTPUT A HIGH ON Q UNTIL \overline{S} DURING T3 WHEN IT IS RESET VIA 9C. Q FROM 10A IS COMBINED WITH S IN 9A, THE OUTPUT OF WHICH IS THEN INVERTED IN 12A. THE RESULTANT SIGNAL, $\overline{T T3A}$, IS A PHASE ADVANCED SIGNAL SYNTHESIZED AHEAD OF $\overline{TT3}$ AND IS USED TO GENERATE CONTROL SIGNALS DETERMINING THE DATA FLOW ON THE 8 BIT DATA BUS. INTERRUPT SIGNALS CAN ORIGINATE FROM THE TTY INT SIGNAL VIA 2D OR FROM THE SYSTEM RESET PUSH BUTTON VIA 3C AND 3B. WITH THE TTY IN THE IDLE STATE, THE SIGNAL AT PIN 9 OF 11C IS A STEADY HIGH STATE. WHEN THE PUSH BUTTON RESET IS NOT DEPRESSED INTA IS HELD LOW AND HENCE THE OUTPUT OF 11C IS LOW AND THE INPUT OF 13B HIGH. WHEN THE PUSH BUTTON RESET IS PRESSED HOWEVER, INTA IS PULLED HIGH BY THE 10K PULL-UP RESISTOR AND INTB IS GROUNDDED. AS A RESULT 3B OUTPUTS A HIGH AND 3C A LOW WHICH PRODUCES A LOW TO HIGH TRANSITION AT THE OUTPUT OF 11C AND A HIGH TO LOW TRANSITION AT THE INPUT OF 13B. THIS CAUSES 13B TO SAMPLE AND HOLD A HIGH AT IT'S Q OUTPUT. WHEN S NEXT GOES HIGH THE HIGH STATE FROM 13B IS TRANSMITTED TO THE Q OUTPUT OF 14, WHICH PROVIDES THE INTERRUPT TO PIN 18 OF THE 8008. WHEN THE INTERRUPT HAS BEEN ACKNOWLEDGED AND A \overline{TTI} STATE GENERATED, THE HIGH LEVEL FROM 10B IS COMBINED WITH $\overline{T2}$ AND $\overline{S.02}$ IN 5A WHICH OUTPUTS A LOW CLEARING THE INTERRUPT REQUEST FROM 13B AND 14.

TWO CONTROL BITS CC1 AND CC2 FROM THE LATCH ON THE CONTROL BUFFER BOARD ARE DECODED BY 4C AND 4B. 4C OUTPUTS A HIGH FOR A PCC CYCLE (INPUT/OUTPUT OPERATION) AND 4B OUTPUTS A HIGH FOR A PCW CYCLE (WRITE CYCLE). THE WRITE CONTROL SIGNAL IS GENERATED AS FOLLOWS.

DURING A NON PCW CYCLE 5B ALWAYS OUTPUTS A HIGH AND HAS NO EFFECT ON 13A WHICH WILL BE SET WITH A HIGH OUTPUT ON ITS \bar{Q} TERMINAL (i.e. THE RAM IS IN THE READ MODE). WHEN THE CYCLE IS A PCW CYCLE, HOWEVER, 5B WILL OUTPUT A LOW DURING S.02 OF T3 WHICH WILL SET 13A WITH A LOW OUTPUT ON \bar{Q} (i.e. THE RAM IS IN THE WRITE MODE) UNTIL 13A IS CLEARED AGAIN BY THE CLOCK AT THE END OF S.02.

DURING A PCC CYCLE THE OUTPUT OF 4C IS HIGH AND THIS SIGNAL GOES TO 5C. IN ADDITION 5C RECEIVES T3A AND (A12 + A13). THE (A12 + A13) SIGNAL COMES FROM THE CONTROL BUFFER BOARD AND IS HIGH IF THE PCC OPERATION IS AN OUTPUT OPERATION. THUS FOR AN OUTPUT OPERATION ALBE GOES LOW DURING T3A. SIMILARLY OUT IS LOW DURING S.02 OF T3A DURING AN OUTPUT OPERATION. ALBE IS USED TO TRANSMIT DATA FROM THE LOW ORDER 8 BIT ADDRESS LATCHES BACK ONTO THE DATA BUS. OUT IS USED TO STROBE THE OUTPUT BOARD DECODER AND HENCE THE OUTPUT LATCHES.

DOE IS SUPPLIED BY 3D. DOE IS HIGH EXCEPT DURING T3A OF A NON PCW INSTRUCTION. (i.e. DURING T3A WHEN NO WRITE TO RAM IS BEING PERFORMED). DOE WHEN IT IS HIGH ALLOWS THE 8008 DATA OUTPUT TO CONTROL THE 8 BIT DATA BUS. DOE IS INVERTED AND SUPPLIES A HIGH TO 7B DURING T3A WHEN NO WRITE TO MEMORY OPERATION IS BEING PERFORMED. 7B IS IN ADDITION FED WITH A HIGH FROM 4C DURING A PCC OPERATION, AND WITH A HIGH FROM (A12 + A13) VIA 2C, WHEN THE PCC OPERATION SPECIFICALLY IS AN INPUT OPERATION.

DURING S.02 IN T4 OF A PCC INPUT OPERATION THE CONDITION FLIP-FLOPS S, Z, P AND C ARE AVAILABLE AT THE 8008 DATA OUTPUTS D0, D1, D2 AND D3 RESPECTIVELY. IN THIS SPECIFIC TIME SLOT FS BECOMES HIGH VIA 7A AND 2E. THIS LINE CAN BE USED TO SAMPLE THE CONDITION FLIP-FLOPS, BUT IS NOT USED IN THE BASIC MOD8 CONFIGURATION. DURING THE PRESENCE OF AN ACKNOWLEDGED INTERRUPT, DURING T3A OF A NON PCW INSTRUCTION IBS WILL GO LOW, GIVING CONTROL OF THE 8 BIT DATA BUS TO THE RESTART TTY I/O BOARD. DURING T3A OF A NON PCC/NON PCW (i.e. PCI OR PCR CYCLE) AND NOT DURING AN ACKNOWLEDGED INTERRUPT 7C WILL RECEIVE ALL HIGH INPUTS, CAUSING MRE TO GO HIGH VIA 2F. WHILST MRE IS HIGH THE MEMORY HAS CONTROL OF THE 8 BIT DATA BUS.

8.2 RESTART AND TTY I/O BOARD

THE PUSH BUTTON RESET ON THIS BOARD NORMALLY HOLDS INTA LOW AND 9C OUTPUTS A HIGH AND 9D A LOW. IF THE TTY GENERATES AN INTERRUPT IBS GOES LOW AND THE LOWER DIODE ARRAY FORCES 11000000 ONTO THE 8 BIT DATA BUS VIA 2C, 2B, 2D, 2A, 1C, 1B, 1D AND 1A. THIS IS AN LAA (i.e. NOP) INSTRUCTION JAMMED INTO THE 8008 ON RECEIPT OF A TTY INTERRUPT. THIS HAS THE EFFECT OF RELEASING THE 8008 FROM THE STOPPED STATE WHICH IT ENTERS IN THE MONITOR 8 SOFTWARE WHEN WAITING FOR TTY INPUT TO COMMENCE. IF THE PUSH BUTTON RESET IS PRESSED INTA GOES HIGH AND INTB IS TAKEN LOW. THIS FORCES 9D TO OUTPUT A HIGH AND 9C TO OUTPUT A LOW. WHEN IBS NOW GOES LOW, AS A RESULT OF THE INTERRUPT GENERATED BY PRESSING THE PUSH BUTTON RESET, THE UPPER DIODE ARRAY FORCES 0000101 ONTO THE 8 BIT DATA BUS VIA 2C, 2B, 2D, 2A, 1C, 1B, 1D AND 1A. THIS IS AN RST (RESTART) AT ADDRESS ZERO INSTRUCTION JAMMED INTO THE 8008 ON RECEIPT OF A PUSH BUTTON

RESET INTERRUPT. THIS HAS THE EFFECT OF RESTARTING EXECUTION AT THE BEGINNING OF THE MONITOR 8 SOFTWARE.

TTY INT ENABLE IS NORMALLY HELD HIGH BY A 10K Ω PULL-UP RESISTOR ENABLING 7D TO PASS SIGNALS FROM THE TTY INPUT. IN THE IDLE MODE THE TTY PROVIDES CLOSED CONTACTS BETWEEN TTY IN+ AND TTY IN- HENCE THE TTY INPUT BUFFER PROVIDES A HIGH TO 3D WHICH IN TURN PROVIDES A LOW TO THE OTHER INPUT OF 7D. THE OUTPUT OF 7D IS NORMALLY LOW AND THIS IS THE TTY INT SIGNAL FED TO THE CPU BOARD. WHEN A START BIT IS RECEIVED FROM THE TTY, THE TTY INPUT BUFFER OUTPUT GOES LOW AND THE TTY INT SIGNAL NOW TRANSITIONS FROM LOW TO HIGH. THIS CAUSES A TTY INTERRUPT TO BE GENERATED ON THE CPU BOARD. AFTER THE TTY INPUT ROUTINE HAS BEEN ENTERED INP IS PULLED LOW AT THE CENTRE OF EACH OF THE INCOMING DATA BITS FROM THE TTY. THE SOFTWARE ALSO SELECTS OUTPUT 0 OF THE ONE OUT OF EIGHT DECODER 4. THUS AT THE SAMPLE TIME FOR EACH OF THE INCOMING BITS THE OUTPUT OF 3A GOES HIGH TRANSMITTING THE INCOMING INFORMATION ONTO DBO OF THE 8 BIT DATA BUS VIA 7A.

TO TRANSMIT INFORMATION OUT TO THE TELETYPE, THE SOFTWARE PROVIDES A HIGH ON A12 AND A LOW ON OUT, CAUSING THE NORMALLY HIGH OUTPUT OF 7C TO GO LOW. THE SOFTWARE IN ADDITION SELECTS OUTPUT 2 OF THE ONE OUT OF EIGHT DECODER 4 TO GO LOW FOR A TTY OUTPUT OPERATION. THUS, THE OUTPUT OF 3B WILL GO FROM ITS NORMALLY LOW STATE TO A TEMPORARILY HIGH STATE. THIS GATES THE OUTPUT INFORMATION FROM DBO ON THE 8 BIT DATA BUS VIA 7B TO THE TQ OUTPUT OF LATCH 8. THE TQ OUTPUT OF 8 IN TURN DRIVES THE TTY OUTPUT BUFFER. SIMILARLY WHEN THE SOFTWARE SELECTS OUTPUT 3 OF 4 TO GO LOW DBO IS LATCHED TO APPEAR ON 3Q OF 8. THIS OUTPUT DRIVES THE READER CONTROL SOLENOID, ALLOWING THE PAPER TAPE READER TO BE STARTED AND STOPPED UNDER PROGRAM CONTROL.

8.3 CONTROL BUFFER BOARD

PACKAGES 1, 2, 10, 11, 5, 6, 7 PROVIDE A CONTROLLED BI-DIRECTIONAL BUS SWITCH WHICH COMMUNICATES BETWEEN THE LOW POWER 8008 EIGHT BIT DATA LINES (8-B0 THRU 8-B7) AND THE 8 BIT SYSTEM BUS (DBO THRU DB7). THE BSE INPUT TO 14D AND 14C IS NORMALLY HELD HIGH VIA A 10K Ω PULL-UP RESISTOR. THE BSE LINE IS BROUGHT OUT TO ALLOW ISOLATION OF THE 8008 FROM THE 8 BIT SYSTEM DATA BUS BY DISABLING THE BUS SWITCH. THIS FEATURE IS INCLUDED TO ALLOW THE IMPLEMENTATION OF A DMA FACILITY IF REQUIRED. NORMALLY, THE DIRECTION OF DATA FLOW IS DETERMINED BY THE DOE CONTROL SIGNAL. WHEN DOE IS HIGH DATA FLOWS FROM THE 8008 DATA BUS TO THE SYSTEM DATA BUS. WHEN DOE IS LOW DATA FLOWS FROM THE SYSTEM DATA BUS TO THE 8008. THE BUS SWITCH ONLY PRESENTS ON LPTTL LOAD ON EACH OF THE 8008 DATA LINES. THE EIGHT 10K Ω RESISTORS ON THE SYSTEM DATA BUS PROVIDE PULL-UP LOADS FOR THE OPEN COLLECTOR BUFFERS ON THE ROM BOARDS.

7C, 7D AND 14A PROVIDE A DECODE FUNCTION. IF A12 OR A13 OR BOTH ARE HIGH, THE OUTPUT OF 14A IS HIGH. PACKAGES 12 AND 13 SERVE TO LATCH THE LOW ORDER ADDRESS BITS A0-A7 DURING T1 TIME. THE LATCHES ARE STROBED BY ADLL. SIMILARLY 3 AND 4 SERVE TO LATCH THE HIGH ORDER ADDRESS BITS A8-A13 AND CONTROL BITS CC1 AND CC2 DURING T2 TIME. THESE LATCHES ARE STROBED BY ADHL. IT WILL BE NOTED THAT 74193 UP/DOWN COUNTERS HAVE BEEN USED IN PLACE OF ORDINARY LATCHES THIS IS TO ALLOW RESETTING (RAL) AND INCREMENTING (IAL) OF THE ADDRESS LATCHES UNDER EXTERNAL CONTROL (e.g. DURING A DMA OPERATION).

FOR NORMAL OPERATION RAL AND IAL ARE RENDERED INOPERATIVE BY TYING THEM LOW AND HIGH RESPECTIVELY. PACKAGES 8 AND 9 ALLOW A0-A7 LATCHES TO OUTPUT LATCHED INFORMATION BACK ONTO THE 8 BIT SYSTEM DATA BUS. THIS TRANSFER IS CARRIED OUT WHEN \overline{ALBE} IS LOW AND IS USED FOR OUTPUT OPERATIONS, WHERE THE CONTENTS OF REGISTER A (i.e. THE DATA TO BE OUTPUT) WILL HAVE BEEN ENTERED INTO LATCHES 12 AND 13 DURING T1 TIME OF THE SECOND MEMORY CYCLE.

8.4 ROM BOARD

PACKAGES 13, 12, 10, 9, 7, 6, 5 AND 4 PROVIDE 2K x 8 OF 1702 PROM OR 1302 MASK PROGRAMMABLE ROM. ALL EIGHT ROM'S ARE ADDRESSED BY A0-A7. A8-A10 ARE DECODED BY A ONE OUT EIGHT DECODER (11). THE DECODER OUTPUTS ARE USED TO SELECT EACH OF THE 8 ROMS VIA THEIR \overline{CS} CONTROL LINES. THE DATA FROM THE SELECTED ROM IS BUFFERED ONTO THE 8 BIT SYSTEM DATA BUS BY 1 AND 2 WHICH ARE OPEN COLLECTOR BUFFERS. THE ROM BOARD SELECT IS DETERMINED USING $\overline{E1}$, $\overline{E2}$, AND $\overline{E3}$, THE SELECT LINES ON 11. A11, A12 AND A13 ARE BUFFERED THROUGH CASCADED LOW POWER INVERTERS ALLOWING ACCESS TO BOTH THE ADDRESSES AND THEIR COMPLEMENTS. A13 OR $\overline{A13}$ (WHICH EVER IS CHOSEN) IS COMBINED WITH THE MRE CONTROL SIGNAL IN 8D. WHEN THE ROM BOARD IS NOT SELECTED 1 AND 2 ARE OPEN CIRCUIT, ALLOWING CONTROL OF THE 8 BIT SYSTEM DATA BUS BY OTHER SOURCES. THE BOARD SELECT OPTION SHOWN ON THE SCHEMATIC REPLICATES THE SIM08 ROM ADDRESS SPACE.

8.5 RAM BOARD

PACKAGES 19, 17, 14, 12, 9, 6, 3 AND 1 PROVIDE 1K x 8 OF 2102 RAM SIMILARLY 20, 18, 15, 13, 10, 7, 4 AND 2 PROVIDE A FURTHER 1K x 8 OF RAM. THE \overline{WRITE} CONTROL LINE DICTATES WHETHER THE RAM IS IN THE READ OR WRITE MODE. THE RAM IS IN THE WRITE MODE WHEN \overline{WRITE} IS LOW. THE RAM OUTPUTS ARE BUFFERED ONTO THE 8 BIT SYSTEM DATA BUS VIA 16 AND 5. THE SIXTEEN RAMS ARE DIRECTLY ADDRESSED BY A0-A9. THE RAM BOARD SELECT IS PERFORMED BY THE ONE OUT OF EIGHT DECODER (8). THIS DECODER DECODES A10-A12. THE BOARD SELECT CODE FOR EACH 1K x 8 OF RAM IS DETERMINED BY MOVING THE OUTPUT TAP ON 8. THE CONNECTION SHOWN ON THE SCHEMATIC REPLICATES THE SIM08 RAM ADDRESS SPACE. THE DECODED BOARD SELECT SIGNAL IS INVERTED AND COMBINED WITH THE MRE CONTROL LINE IN 11B, WHICH IN TURN CONTROLS THE OUTPUT BUFFERS FROM THE RAM BOARD.

8.6 INPUT BOARD

THIS BOARD PROVIDES 3 EIGHT BIT INPUT PORTS. THE INPUT PORT SELECT IS DECODED IN A ONE OUT OF EIGHT DECODER FED FROM A9-A11. THIS ALLOWS A SYSTEM TOTAL OF EIGHT POSSIBLE 8 BIT INPUT PORTS. THE ONE OUT OF EIGHT DECODER IS CONTROLLED BY THE \overline{INP} CONTROL SIGNAL. WHEN THIS SIGNAL IS LOW, AN INPUT FUNCTION IS BEING PERFORMED. THE SELECTED 8 BIT INPUT PORT IS THEN GATED ONTO THE 8 BIT SYSTEM DATA BUS.

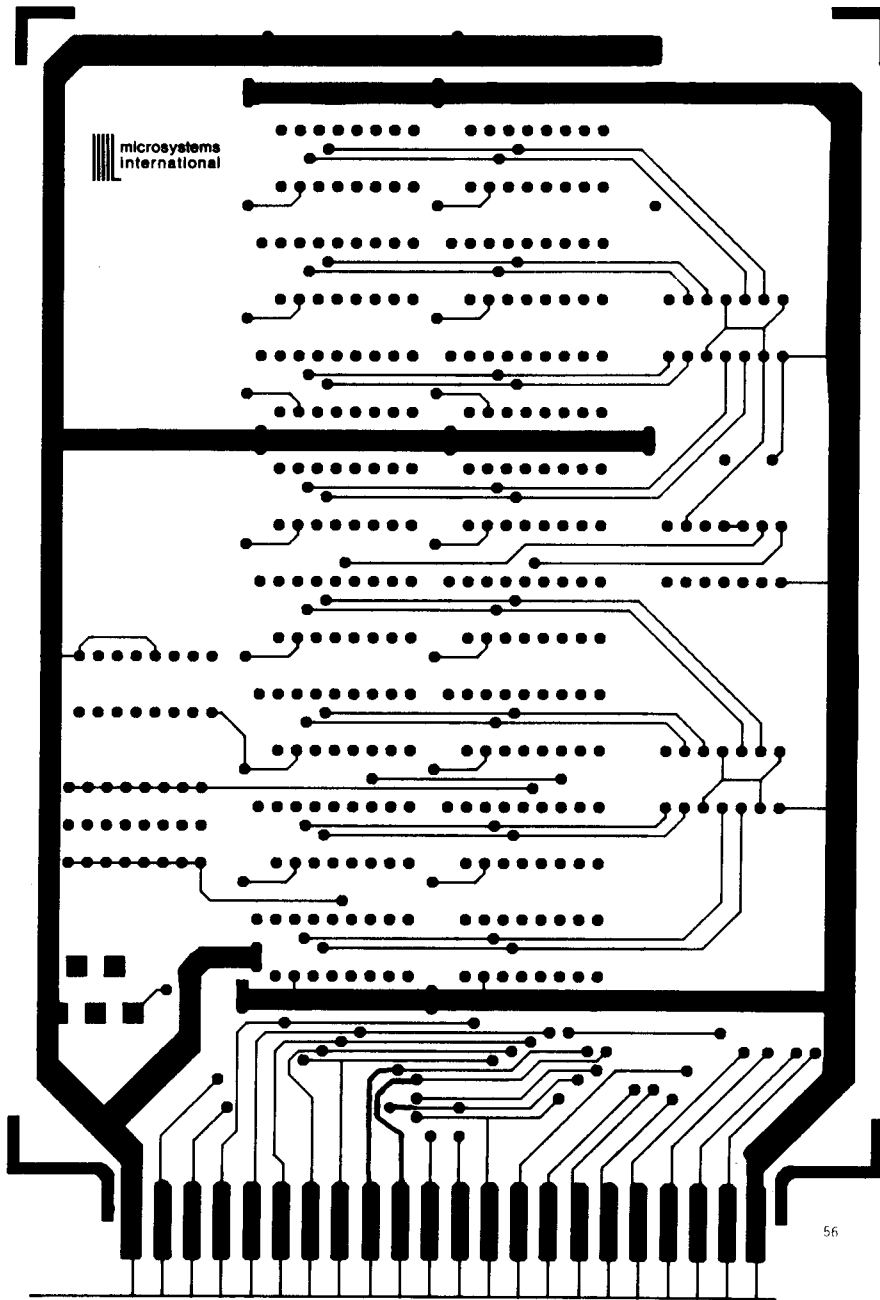
8.7 OUTPUT BOARD

THIS BOARD PROVIDES 3 EIGHT BIT OUTPUT CHANNELS. DBO - DB7, THE 8 BIT SYSTEM DATA BUS IS BUFFERED IN VIA OPEN COLLECTOR BUFFERS WITH $10K\Omega$ PULL-UP RESISTORS ON THEIR OUTPUTS. UP TO 24 EIGHT BIT OUTPUT PORTS CAN BE ACCOMODATED BY THE SYSTEM. THEY ARE SELECTED BY A ONE OUT OF EIGHT DECODE PERFORMED ON A9 - A11. FOR AN OUTPUT FUNCTION A12, A13 CAN FURTHERMORE HAVE 3 STATES 10, 11 AND 01. THESE ADDITIONAL SELECT SIGNALS ARE DECODED AND COMBINED WITH THE ONE OUT OF EIGHT SELECT SIGNALS, TO YIELD A TOTAL OF 24 OUTPUT PORT SELECT COMBINATIONS. THE ONE OUT OF EIGHT DECODER IS ENABLED BY ~~OUT~~ ~~OUT~~ BEING LOW. THIS CORRESPONDS TO AN OUTPUT FUNCTION BEING PERFORMED. WHEN A PARTICULAR OUTPUT PORT IS SELECTED ITS 8 BIT DATA LATCH IS STROBED TO LATCH THE INFORMATION CURRENTLY BEING FED IN FROM THE 8 BIT SYSTEM DATA BUS.

8.8 MOD8 SCHEMATICS AND PCB LAYOUTS

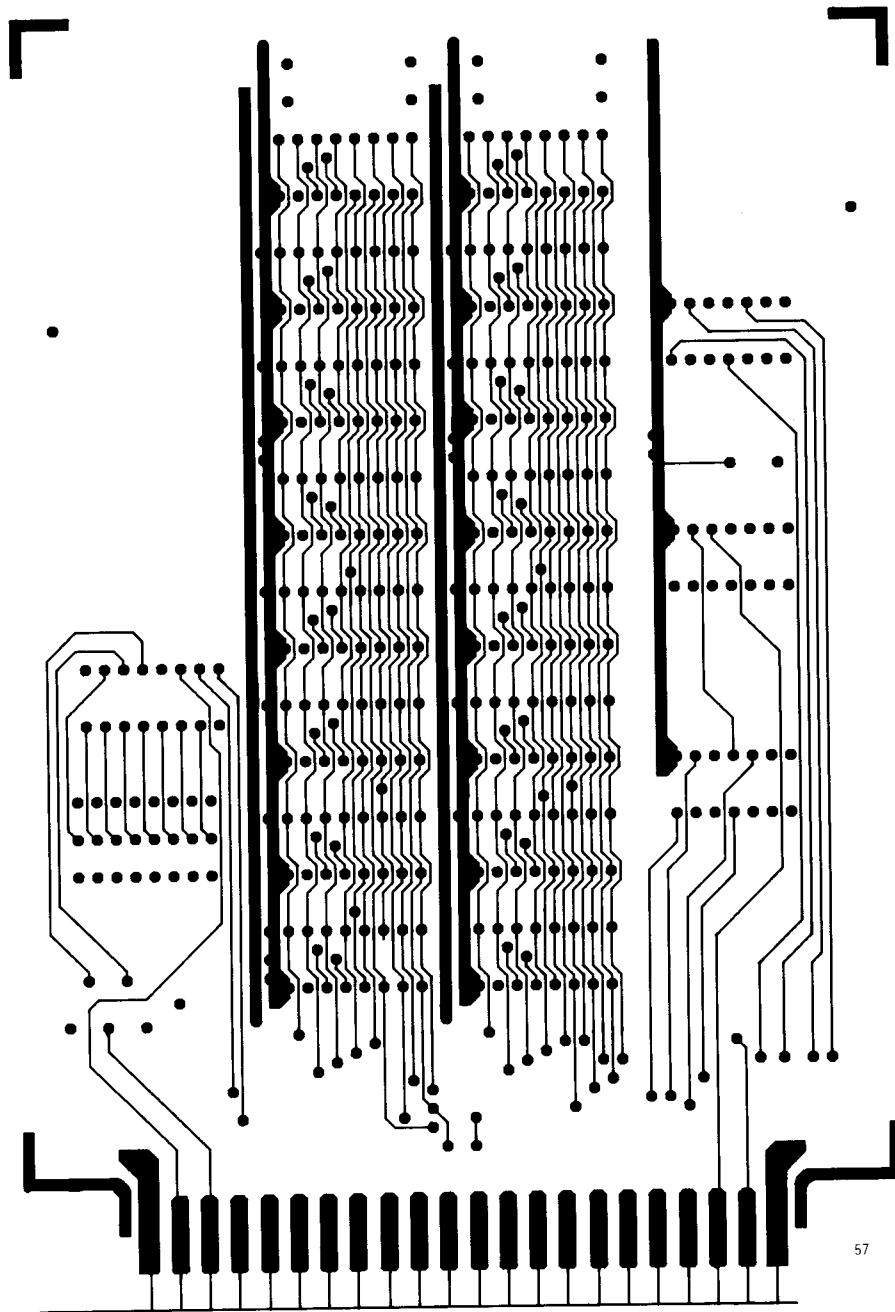
THE FOLLOWING DIAGRAMS DEPICT THE CIRCUIT SCHEMATIC, COMPONENT SIDE PCB LAYOUT, UNDERSIDE PCB LAYOUT AND COMPONENT PLACEMENT. A RECOMMENDED SUPPLIER OF THE MOD8-1, MOD8-2, MOD8-3, MOD8-4, MOD8-5, MOD8-6 AND MOD8-7 UNPOPULATED PCB'S IS:

SPACE CIRCUITS LTD.
156 ROGER ST.
WATERLOO
ONTARIO
CANADA



4

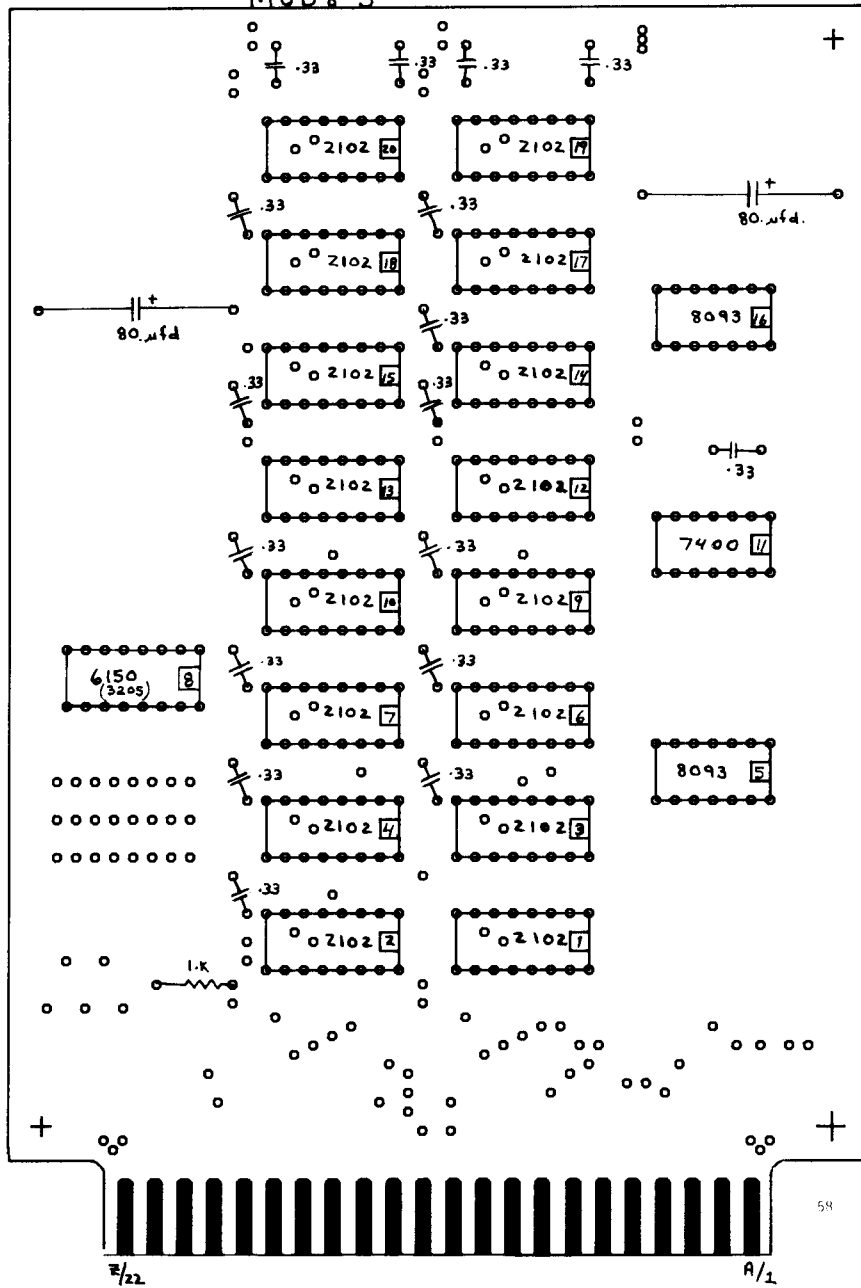
4



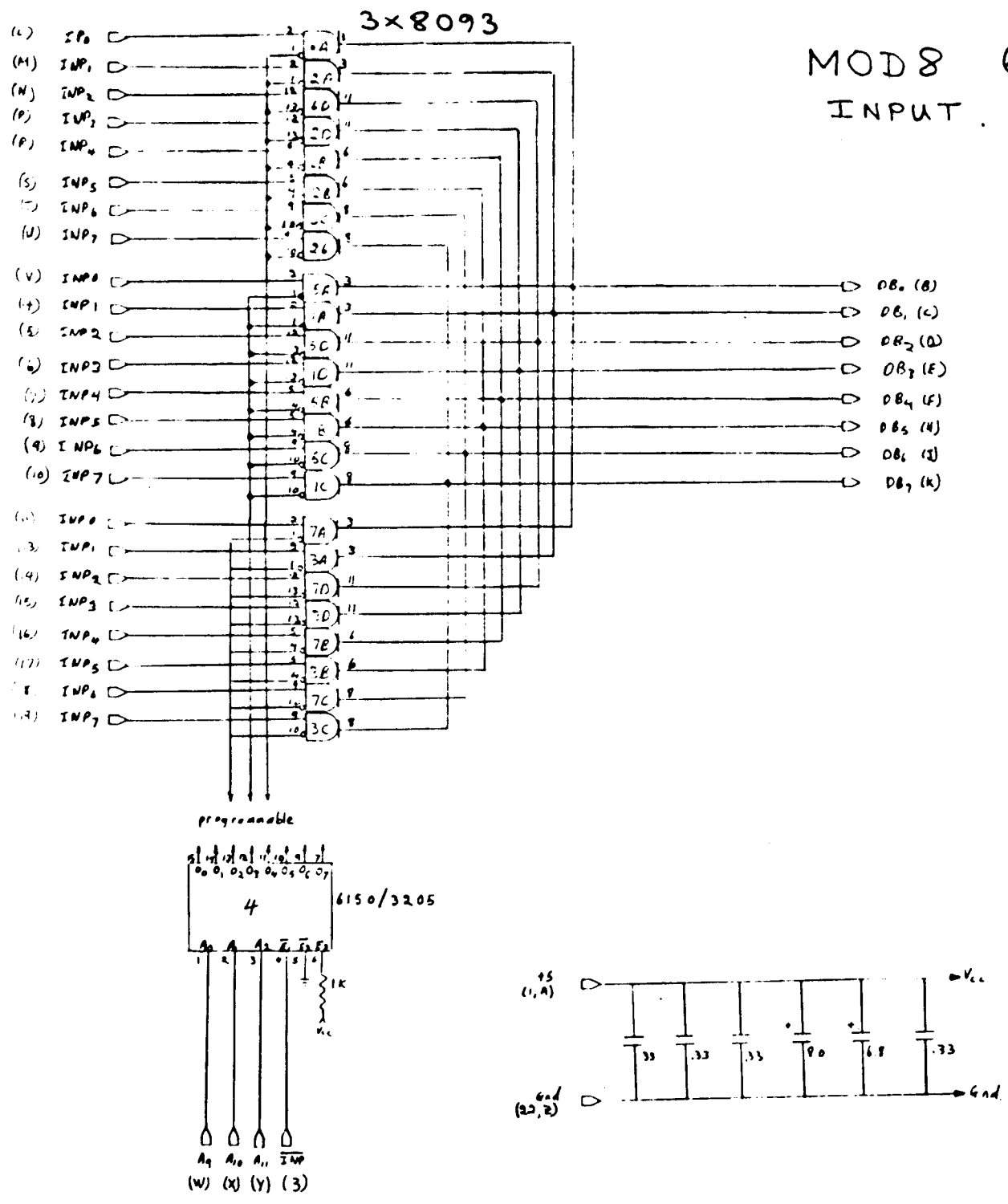
44

44

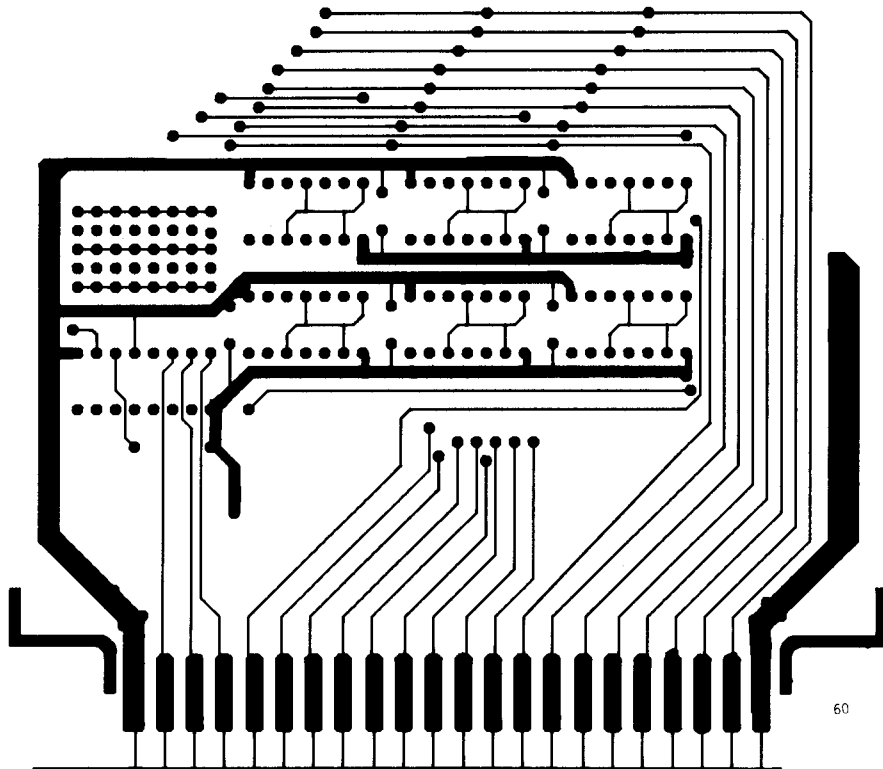
MOD 8 5



MOD 8 6
INPUT.

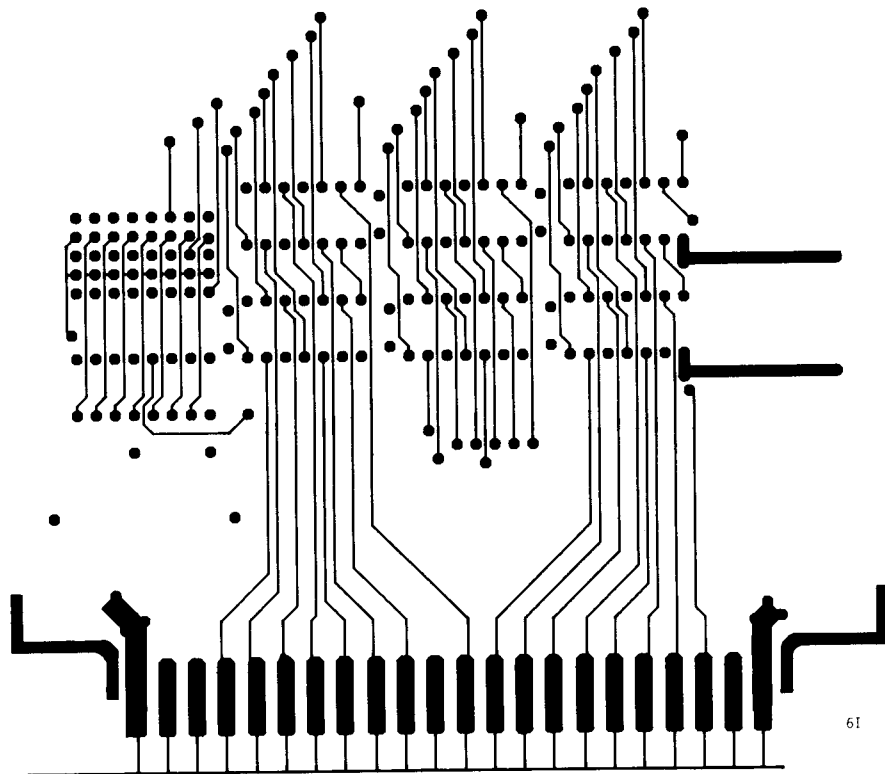


: Tristate

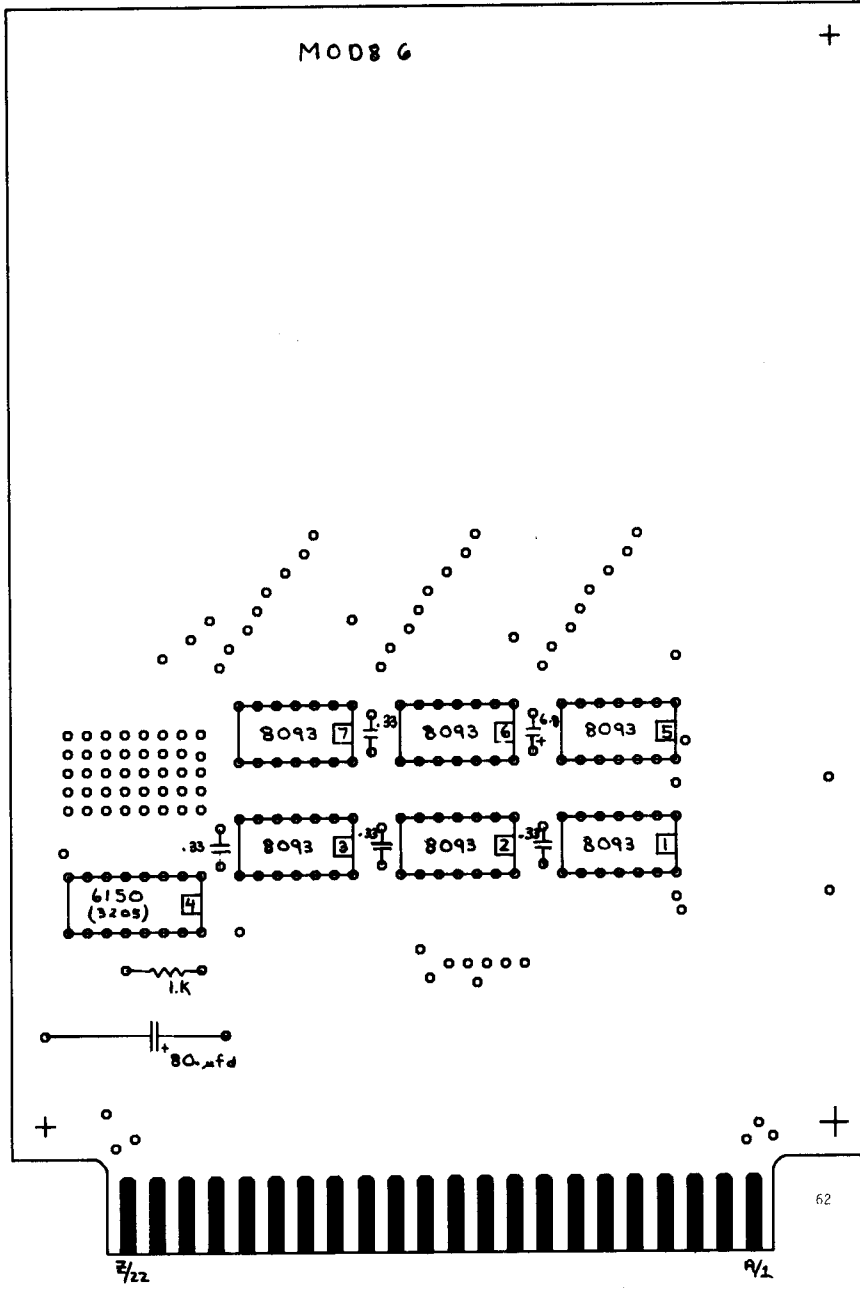




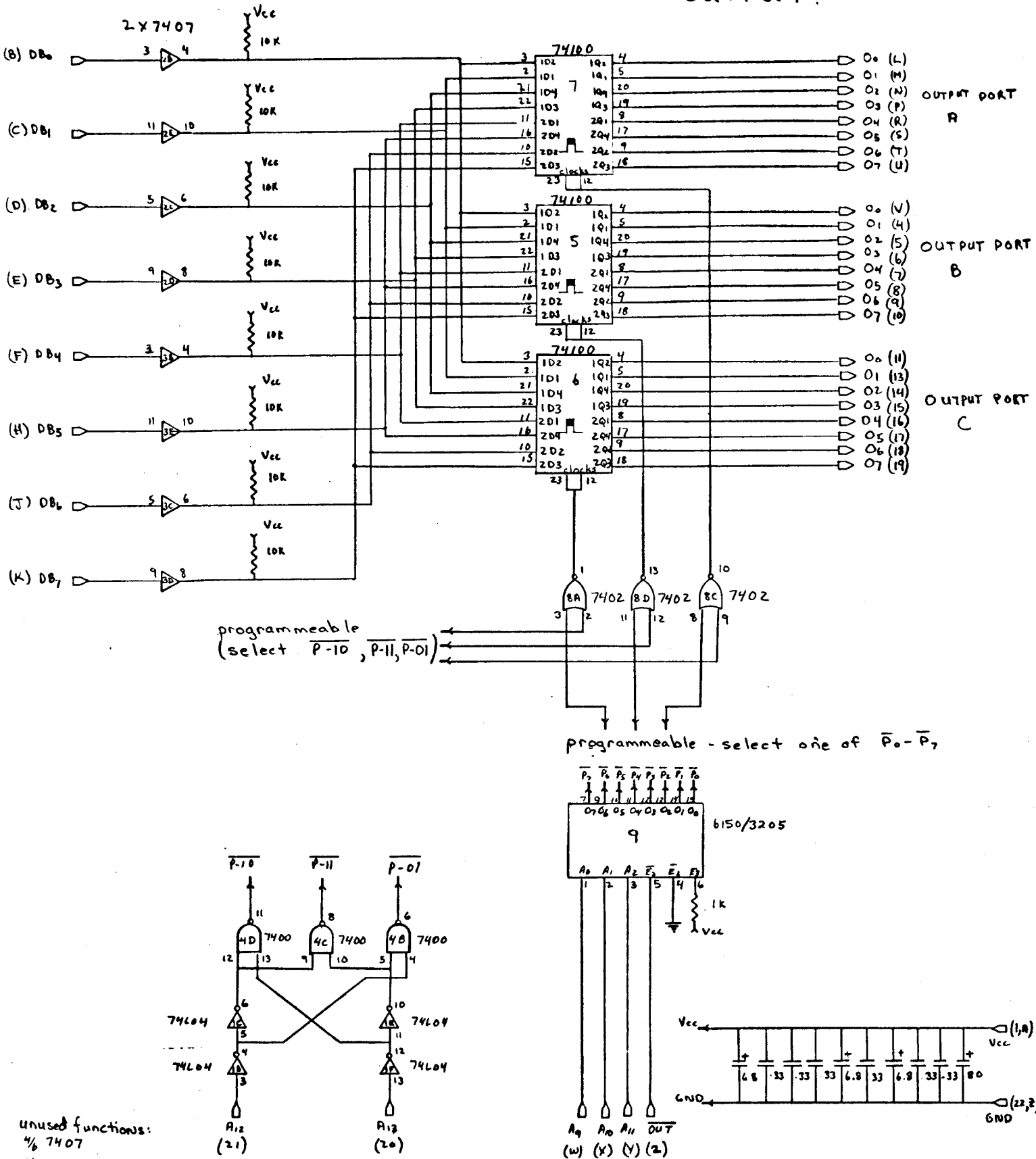
44

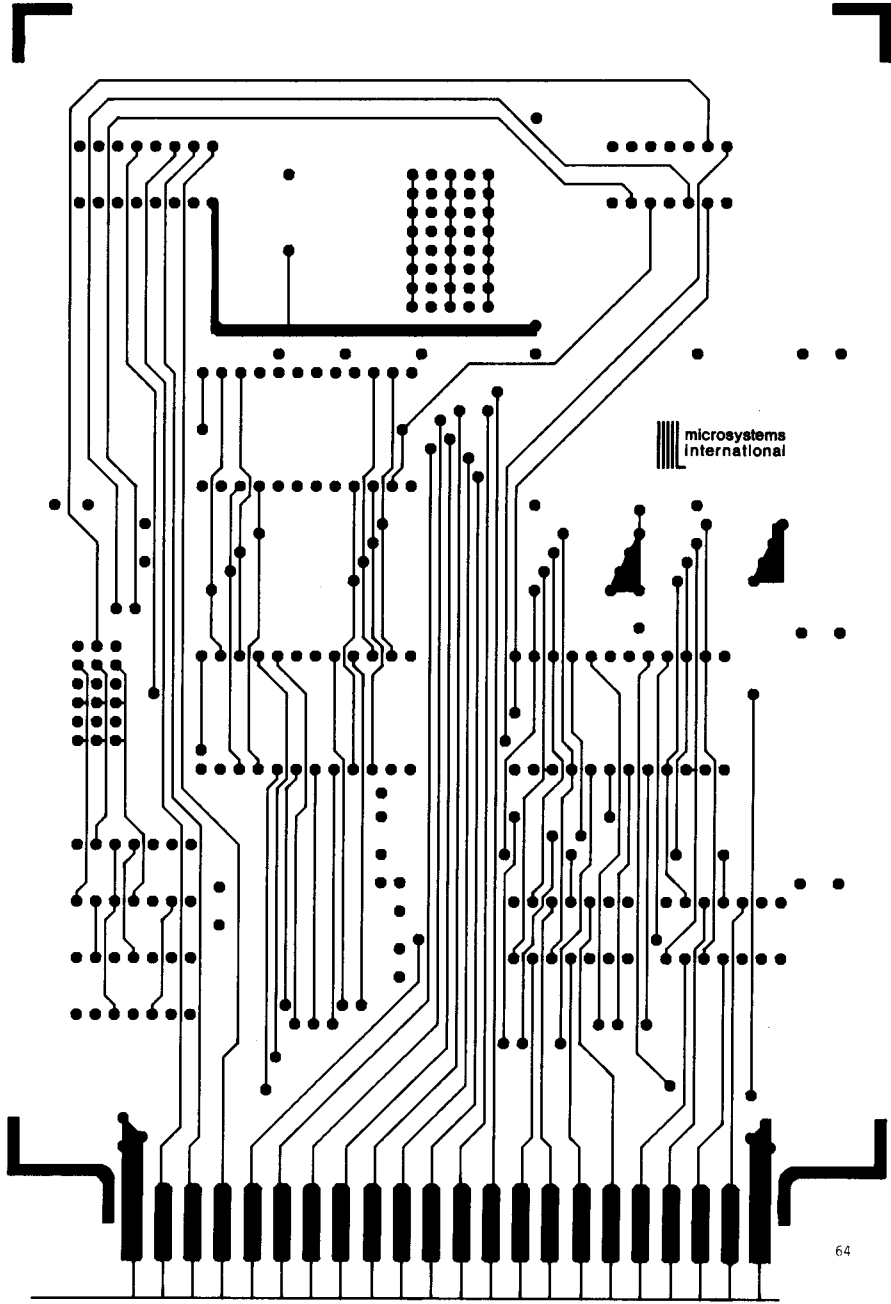


MOD 8 6



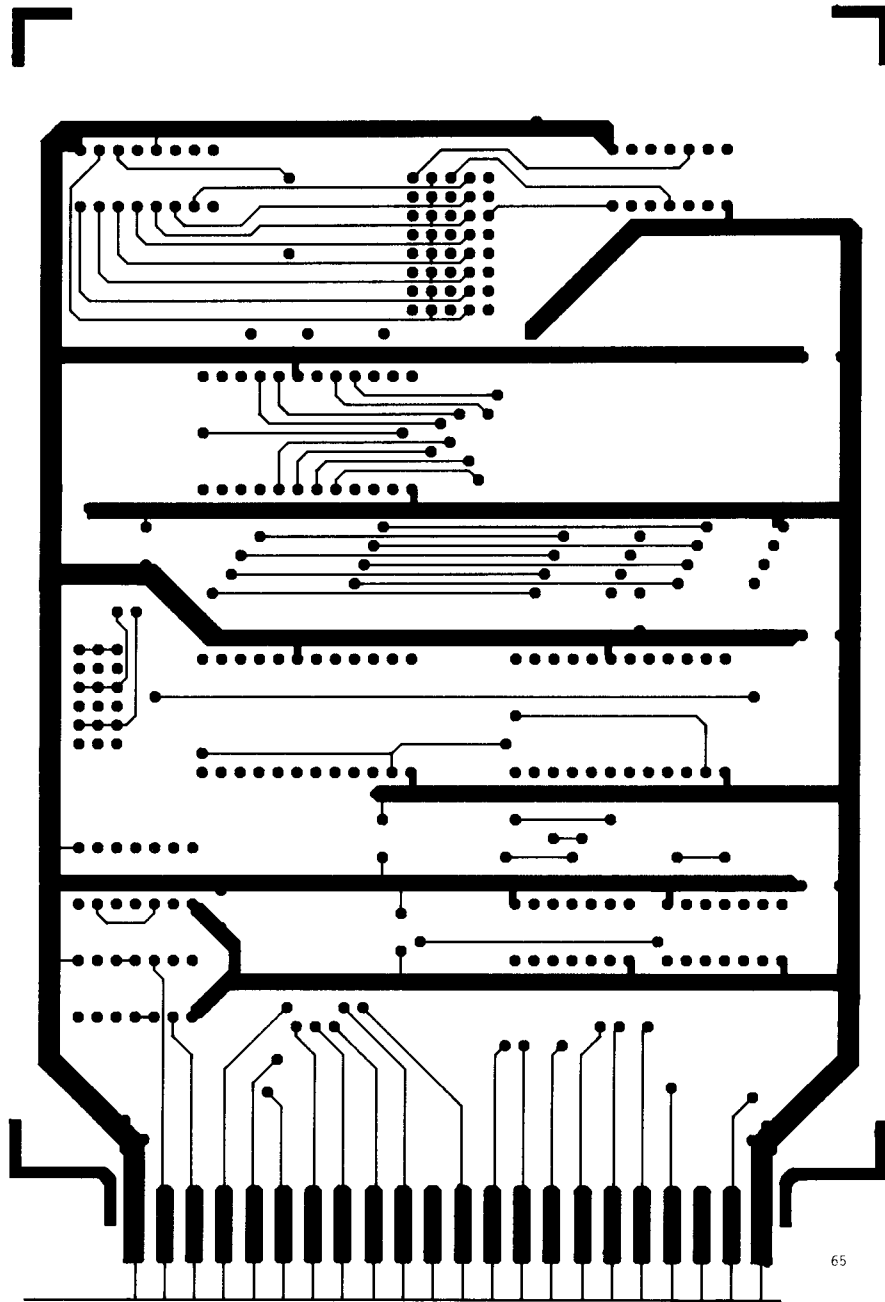
MOD 8 7 OUTPUT.





4

4



9. MONITOR 8 USERS GUIDE

THE MONITOR 8 SOFTWARE ALLOWS SYMBOLIC LOADING AND DUMPING OF 8008 PROGRAMS, AND ALSO OFFERS UTILITY EDITING AND MANIPULATION FACILITIES.

9.1 SYSTEM START-UP (MOD 8 HARDWARE CONFIGURATION)

- 9.1.1 ENSURE POWER OFF TO PROGRAMMER (IF ONE IS INCLUDED), TTY SET TO LOCAL.
- 9.1.2 APPLY CPU POWER.
- 9.1.3 PUSH RESET BUTTON.
- 9.1.4 TURN TTY TO "ON LINE" AND PUSH RESET AGAIN. WHEN TTY IS ON LINE AND A RESET IS EXECUTED THE TTY WILL TYPE A CRLF AND 8 DASHES FOLLOWED BY A CRLF.
(e.g. RESET BUTTON PUSHED
-----TTY RESPONSE.)

9.2 SYSTEM START-UP (SIM8 -01 HARDWARE CONFIGURATION)

- 9.2.1 ENSURE POWER OFF TO PROGRAMMER, TTY SET TO LOCAL.
- 9.2.2 APPLY CPU POWER WITH RESET SWITCHES SET TO 005.
- 9.2.3 PUSH RESET BUTTON.
- 9.2.4 TURN TTY TO "ON LINE" AND PUSH RESET AGAIN. WHEN THE TTY IS ON LINE AND A RESET 005 INSTRUCTION IS EXECUTED, THE TTY WILL TYPE CRLF AND 8 DASHES, FOLLOWED BY A CRLF.
(e.g. RESET BUTTON PUSHED WITH SWITCHES SET TO 005
-----TTY RESPONSE.)
- 9.2.5 SET RESET SWITCHES TO 300.

9.3 ADDRESSING

THE MEMORY IN THE 8008 SYSTEM IS ORGANIZED INTO BANKS. EACH BANK IS 377 OCTAL (256 DECIMAL) BYTES IN LENGTH. WHEN COMMUNICATING WITH MONITOR 8 THE ADDRESSES TAKE THE FOLLOWING FORM:

$$N_5 \ N_4 \ N_3 \ N_2 \ N_1 \ N_0$$

$N_0 - N_5$ ARE OCTAL DIGITS WITH THE FOLLOWING SIGNIFICANCE:

N_5 = SPECIAL MODIFIER VALUE 0-3 POSSIBLE.

$N_5 = 0$ OR 1 MEMORY ACCESSED IS NORMAL ROM OR RAM.

$N_5 = 2$ OR 3 MEMORY ACCESSED IS THE pROM IN THE PROGRAMMING STATION, IF ONE IS ATTACHED TO THE SYSTEM.

$N_4 N_3$ = MEMORY BANK NUMBER

$N_4 N_3 = 00$ TO 07 MEMORY ACCESSED IS ROM IN SIM 08 AND MOD 8 SYSTEMS. $N_4 N_3 = 10$ TO 13 MEMORY ACCESSED IS RAM IN SIM08 AND MOD8 SYSTEMS.

$N_2 N_1 N_0$ = BYTE LOCATION WITHIN BANK, VALUE 000 to 377 POSSIBLE.

9.4 MONITOR 8 COMMAND SUMMARY

THE MONITOR 8 SYSTEM IS NOW READY TO LOAD SYMBOLIC PROGRAM INPUT OR ACCEPT ONE OF THE FOLLOWING UTILITY COMMANDS.

LOC	(SET CURRENT LOCATION POINTER)
DLP	(DISPLAY CURRENT LOCATION POINTER)
DPS	(DUMP SYMBOLIC)
LDO	(LOAD OCTAL)
DPO	(DUMP OCTAL)
LBF	(LOAD BNPF FORMAT)
DBF	(DUMP BNPF FORMAT)
EDT	(ENTER EDIT MODE)
XQT	(INITIATE PROGRAM EXECUTION)
CPY	(COPY ROUTINE)
TRN	(TRANSLATE ROUTINE)
SBP	(SET BREAK-POINT)
CBP	(CLEAR BREAK-POINT)
PRG	(PROGRAM pROM)

9.5 LOC (SET CURRENT LOCATION POINTER)

ALL DATA ENTRY AND MANIPULATION IS DONE AT THE ADDRESS INDICATED BY THE CURRENT LOCATION POINTER (CLP). THE POINTER VALUE IS STORED AND USED BY THE MONITOR SOFTWARE. AFTER EACH MACHINE INSTRUCTION IS ENTERED THE CLP IS UPDATED TO POINT AT THE NEXT AVAILABLE MEMORY LOCATION. THE TWO PSEUDO OPERATORS LOC AND DLP ALLOW THE USER TO PRESET AND DISPLAY THE CURRENT LOCATION POINTER.

WHEN LOC IS TYPED THE MACHINE RESPONDS WITH A SPACE (␣). THE USER MUST THEN SPECIFY A SIX DIGIT ADDRESS (SEE ADDRESSING). AFTER THE LAST ADDRESS DIGIT HAS BEEN ENTERED, THE MACHINE RESPONDS WITH CRLF AND WAITS FOR THE NEXT COMMAND. THE MONITOR SOFTWARE USES RAM ADDRESSES 013350 - 013377 INCLUSIVE, BUT ALL OTHER ADDRESSES ARE AVAILABLE TO THE USER.

9.6 DLP (DISPLAY CURRENT LOCATION POINTER)

IF THE USER WISHES TO DISPLAY THE CLP, HE MAY TYPE IN DLP. THE MACHINE RESPONDS BY TYPING OUT THE CLP AND THEN PERFORMS A CRLF AND WAITS FOR THE NEXT INSTRUCTION.

NOTE: THE CLP IS DESTROYED BY SEVERAL OF THE MONITOR ROUTINES. WHEN THIS IS THE CASE, THE MONITOR WILL PRINT 8 DASHES ON COMPLETION OF THE REQUESTED FUNCTION. IN THESE INSTANCES, THE USER SHOULD RESPECIFY THE CLP USING THE LOC COMMAND BEFORE PROCEEDING.

9.7 SYMBOLIC PROGRAM INPUT

ONCE THE CLP HAS BEEN INITIALIZED, THE USER MAY TYPE IN HIS PROGRAM. AFTER EACH MNEMONIC INSTRUCTION HAS BEEN ENTERED, THE MACHINE WILL RESPOND WITH A CRLF OR, IF THE INSTRUCTION REQUIRES AN ARGUMENT, WITH A SPACE. ALL IMMEDIATE INSTRUCTIONS REQUIRE A 3 DIGIT OCTAL DATA BYTE. ALL JUMP AND CALL COMMANDS REQUIRE A 6 DIGIT SPLIT OCTAL ADDRESS (SEE ADDRESSING). INPUT/OUTPUT AND RESTART INSTRUCTIONS REQUIRE A 3 DIGIT OCTAL NUMBER TO SPECIFY A PORT NUMBER OR RESTART ADDRESS. AFTER THE INSTRUCTION AND THE CORRESPONDING ARGUMENT HAVE BEEN ENTERED, A CRLF WILL BE GENERATED AND THE NEXT INSTRUCTION MAY THEN BE ENTERED. AFTER EACH ENTRY, THE CLP IS AUTOMATICALLY UPDATED TO POINT TO THE NEXT AVAILABLE MEMORY LOCATION.

THERE ARE SEVERAL BIT COMBINATIONS WHICH WILL BE INTERPRETED BY THE 8008 AS A HALT COMMAND. THE FOLLOWING COMMANDS WILL BE INTERPRETED BY THE MONITOR TO GIVE RISE TO HALT COMMAND BIT COMBINATIONS.

<u>MNEMONIC</u>	<u>RESULTANT OCTAL</u>	<u>8008 INTERPRETATION</u>
HLT	000	HLT
INA	000	HLT
DCA	001	HLT
LMM	377	HLT

9.8 DPS (DUMP SYMBOLIC)

A SYMBOLIC LISTING IS GENERATED BY TYPING DPS. THE MACHINE WILL RESPOND WITH A CRLF AND A * (THIS IS THE PROMPTER INDICATING THAT THE MACHINE REQUIRES FURTHER ADDRESS INFORMATION). THE USER MUST NOW TYPE IN THE INITIAL AND FINAL ADDRESS, DEFINING THE BLOCK OF CODE TO BE DUMPED. THESE TWO ADDRESSES MUST BE ENTERED AS A 6 DIGIT SPLIT OCTAL NUMBER (SEE ADDRESSING). WHEN THE INITIAL ADDRESS HAS BEEN ENTERED, THE MACHINE RESPONDS WITH A BLANK AND AWAITS THE FINAL ADDRESS. WHEN THE FINAL ADDRESS HAS BEEN ENTERED THE MACHINE RESPONDS WITH 3 CRLF'S AND COMMENCES LISTING. THE LISTING INCLUDES THE CURRENT MEMORY ADDRESS, THE OCTAL INSTRUCTION AND THE MNEMONIC. FOR A MULTI-BYTE INSTRUCTION THE LISTED ADDRESS IS THAT OF THE FIRST BYTE OF THE INSTRUCTION. ANY DATA FIELDS ASSOCIATED WITH THE INSTRUCTION (IMMEDIATE DATA), ADDRESSES, I/O PORT NUMBERS OR RESTART ADDRESSES) WILL BE PRINTED FOLLOWING THE MNEMONIC. ONE INSTRUCTION IS LISTED PER LINE WITH 62 LINES GENERATED PER PAGE. AN AUTO PAGING FEATURE SEPARATES EACH 11" PAGE BY 3 CRLF'S. INVALID INSTRUCTIONS ARE DISPLAYED AS ???.

9.9 LDO (LOAD OCTAL)

TYPING LDO WILL INITIATE THE OCTAL LOAD ROUTINE. AS IN THE DUMP ROUTINE, THE MACHINE WAITS FOR TWO OCTAL ADDRESSES. IT THEN OUTPUTS A CRLF AND WILL BEGIN READING IN FROM THE KEYBOARD OR TAPE READER. EACH LINE WHICH CONTAINS DATA MUST HAVE A / SYMBOL TO THE LEFT OF THE DATA FIELD. EACH 3 DIGIT OCTAL VALUE WHICH FOLLOWS THE / IS INTERPRETED AS DATA. LEADING ZEROS MUST BE INCLUDED AND EACH VALUE MUST BE SEPARATED BY AT LEAST 1 BLANK. ANY DATA TO THE LEFT OF THE FIRST / IS IGNORED (NOTE THAT THIS IS USUALLY THE ADDRESSES GENERATED BY THE DPO ROUTINE). WHEN THE FINAL ADDRESS SPECIFIED HAS BEEN FILLED, THE ROUTINE RETURNS TO THE MONITOR.

9.13 EDT (ENTER EDIT MODE)

THE EDIT MODE IS ENTERED BY TYPING EDT. THE EDITOR RESPONDS WITH A CRLF AND TYPES THE VALUE OF THE CLP FOLLOWED BY A /. IT IS NOW READY TO ACCEPT ONE OF THE FOLLOWING COMMANDS:

- nnn - WHERE nnn IS A THREE DIGIT OCTAL VALUE TO BE LOADED INTO MEMORY.
- † - DISPLAY MEMORY VALUE
- ↑ - DECREMENT THE CURRENT LOCATION POINTER
- *AAAAAA - REDEFINE THE CURRENT LOCATION POINTER WITH THE VALUE AAAAAA
- @ - EQUIVALENT TO XQT
- R - RETURN TO THE MONITOR

IF DATA IS TO BE LOADED IT MUST IMMEDIATELY FOLLOW THE / SYMBOL. AN INVALID SYMBOL WILL CAUSE A CRLF WITH THE CLP RETYPED. THE nnn VALUE IS ASSEMBLED AS AN 8 BIT WORD AND STORED IN THE MEMORY. ATTEMPTING TO WRITE INTO A ROM ADDRESS WILL NOT BE FLAGGED, YET THE DATA WILL NOT (CANNOT) BE WRITTEN.

IF A BLANK IS ENTERED AFTER THE / THE CURRENT MEMORY LOCATION WILL BE DISPLAYED. TWO OPTIONS ARE THEN AVAILABLE:

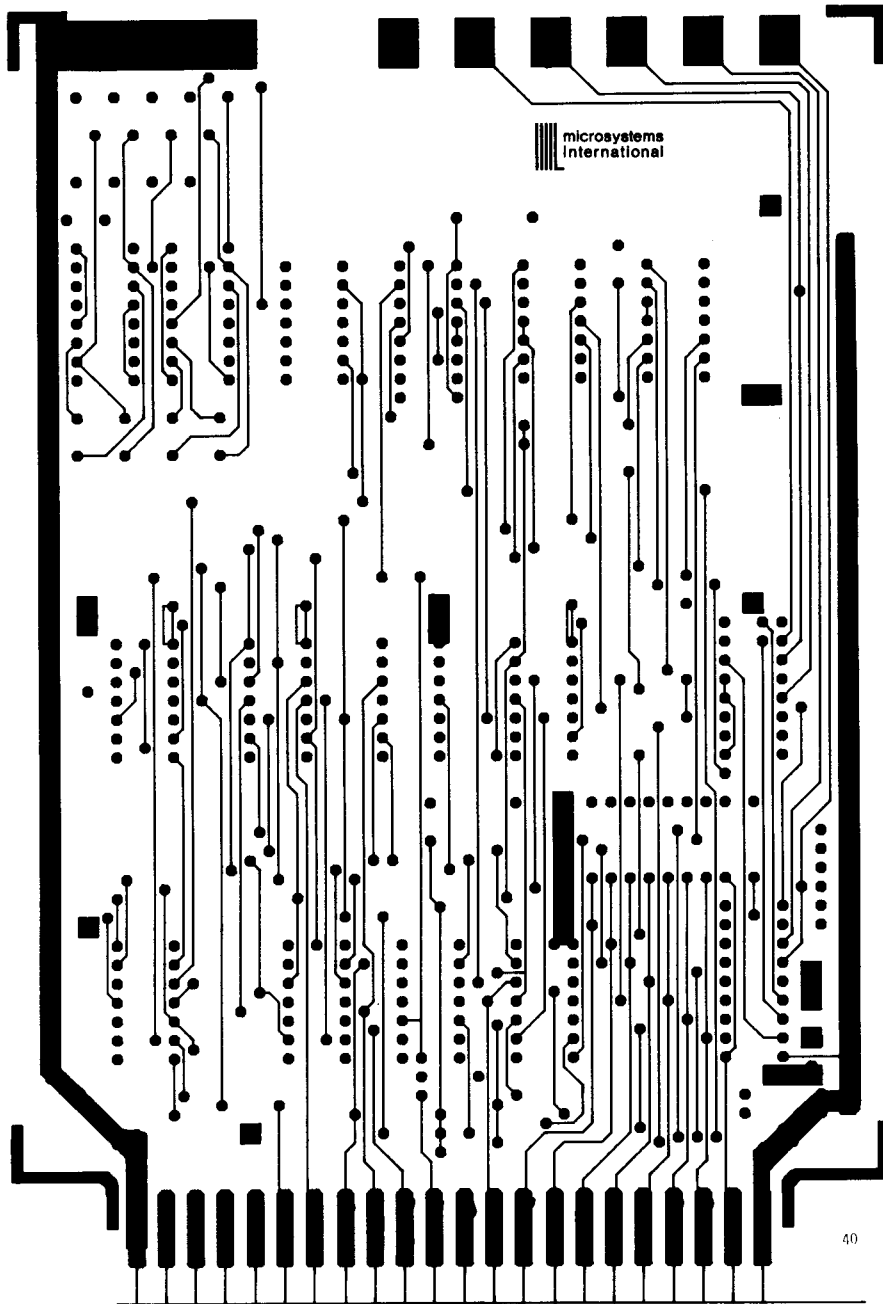
- a) ← nnn REPLACE THE CURRENT VALUE WITH nnn.
- b) ANY OTHER SYMBOL WILL INCREMENT THE CLP.

FOLLOWING THE CLP / THE EDITOR EXAMINES THE FIRST CHARACTER INPUTTED TO DETERMINE THE COMMAND. IF DATA IS TO BE INPUT IMMEDIATELY, IT MUST BE IN THE FIRST THREE LOCATIONS FOLLOWING THE /. IF THE DATA FOLLOWS a← (USED TO REPLACE DISPLAYED DATA) THE INPUT IS RELATIVELY FORMAT FREE. THE FIRST OCTAL DIGIT WILL DEFINE THE REPLACEMENT DATA, ANY OTHER SYMBOLS MAY APPEAR BETWEEN THE ← SYMBOL AND THE DATA. THE SAME IS TRUE OF THE *AAAAAA COMMAND. FOLLOWING THE COMMAND OR DATA THE EDITOR TYPES THE NEW CLP ON THE NEXT LINE AND IS READY TO ACCEPT THE NEXT COMMAND.

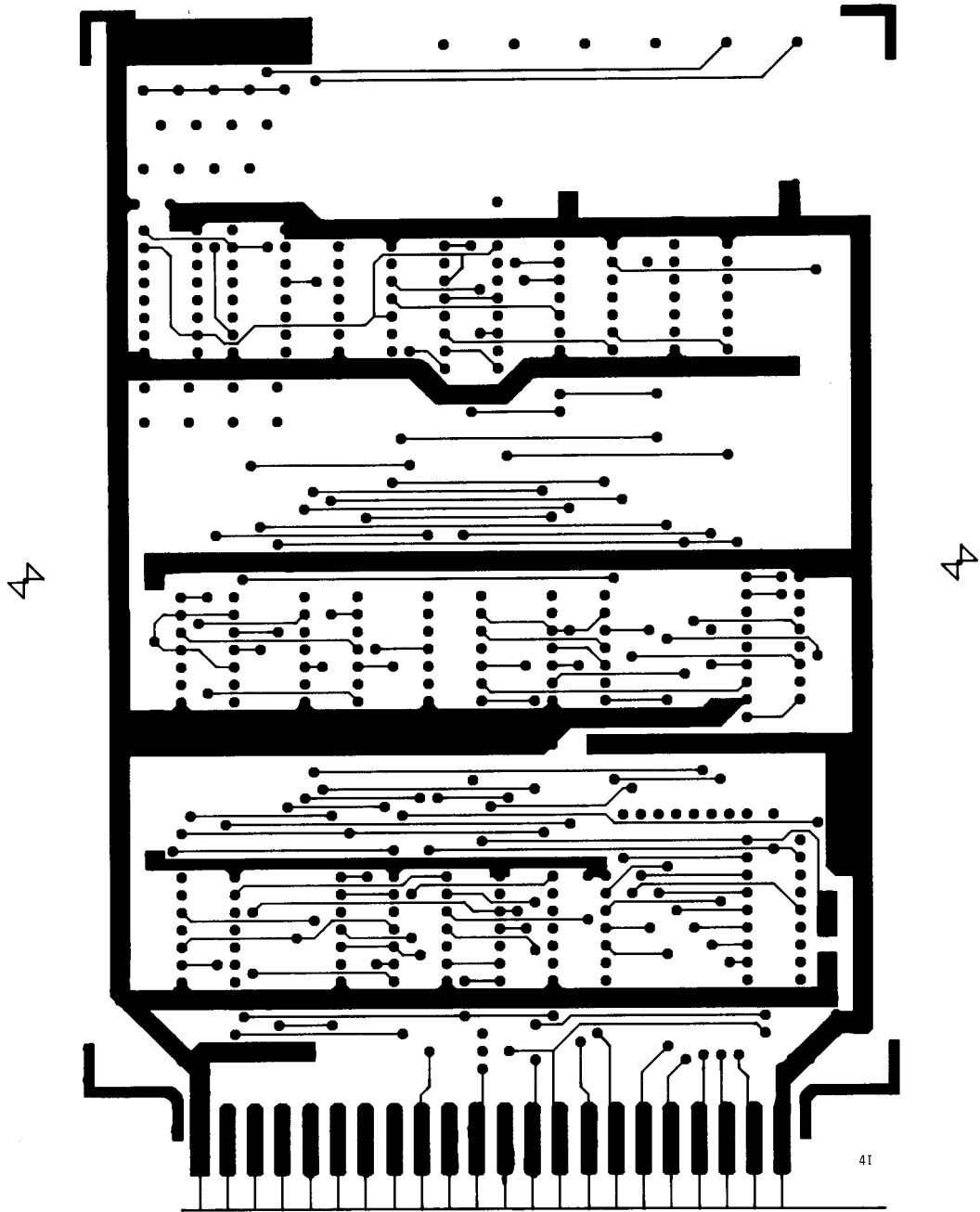
9.14 XQT (INITIATE PROGRAM EXECUTION)

THE XQT COMMAND ALLOWS THE USER TO START THE EXECUTION OF HIS PROGRAM. FOLLOWING THE TYPING OF XQT THE MACHINE WILL RESPOND WITH A SPACE AND WAIT FOR THE STARTING ADDRESS OF THE PROGRAM. THE ENTIRE USER ROUTINE IS TREATED AS A SUBROUTINE WHICH IS CALLED FROM THE MONITOR. THE USER MAY RETURN TO MONITOR BY INCLUDING A RET (RETURN) AT THE END OF HIS ROUTINE.

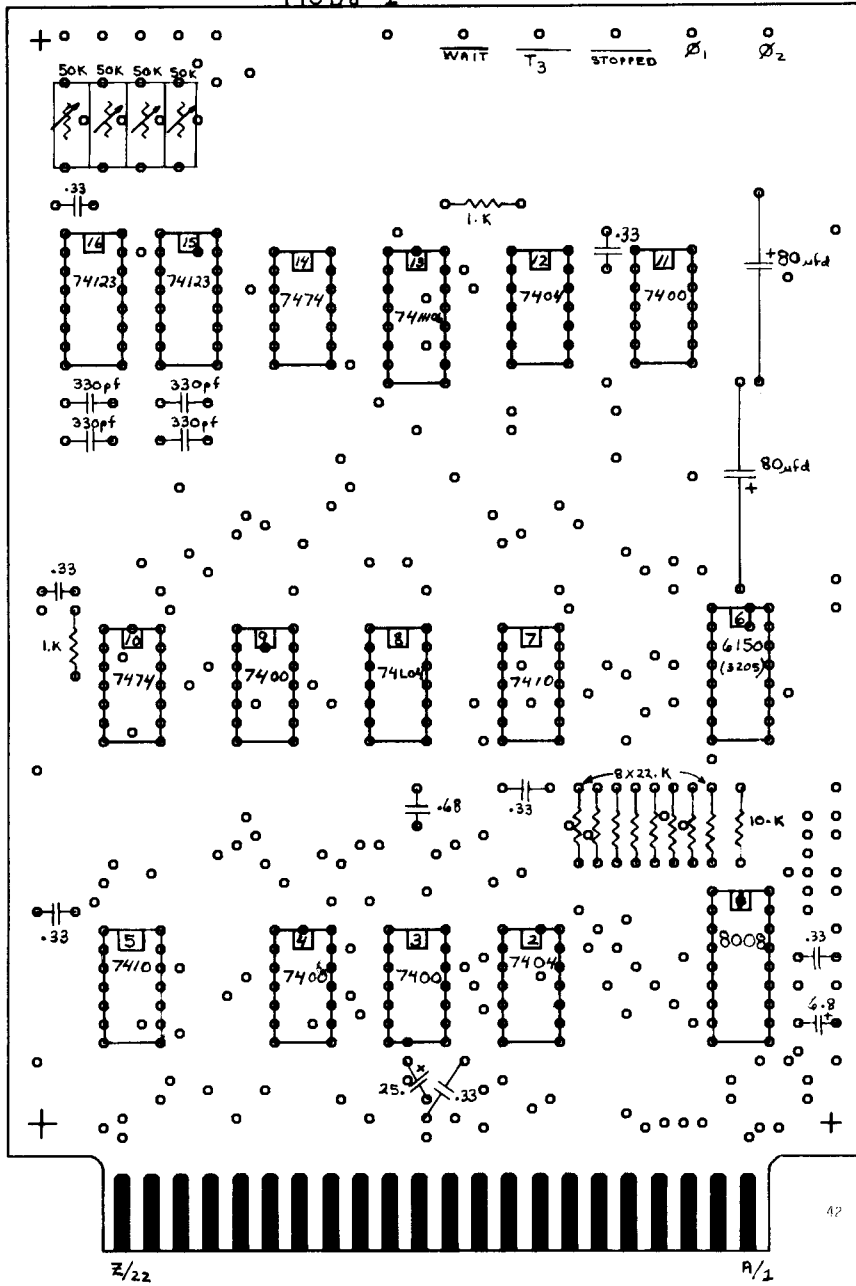
4

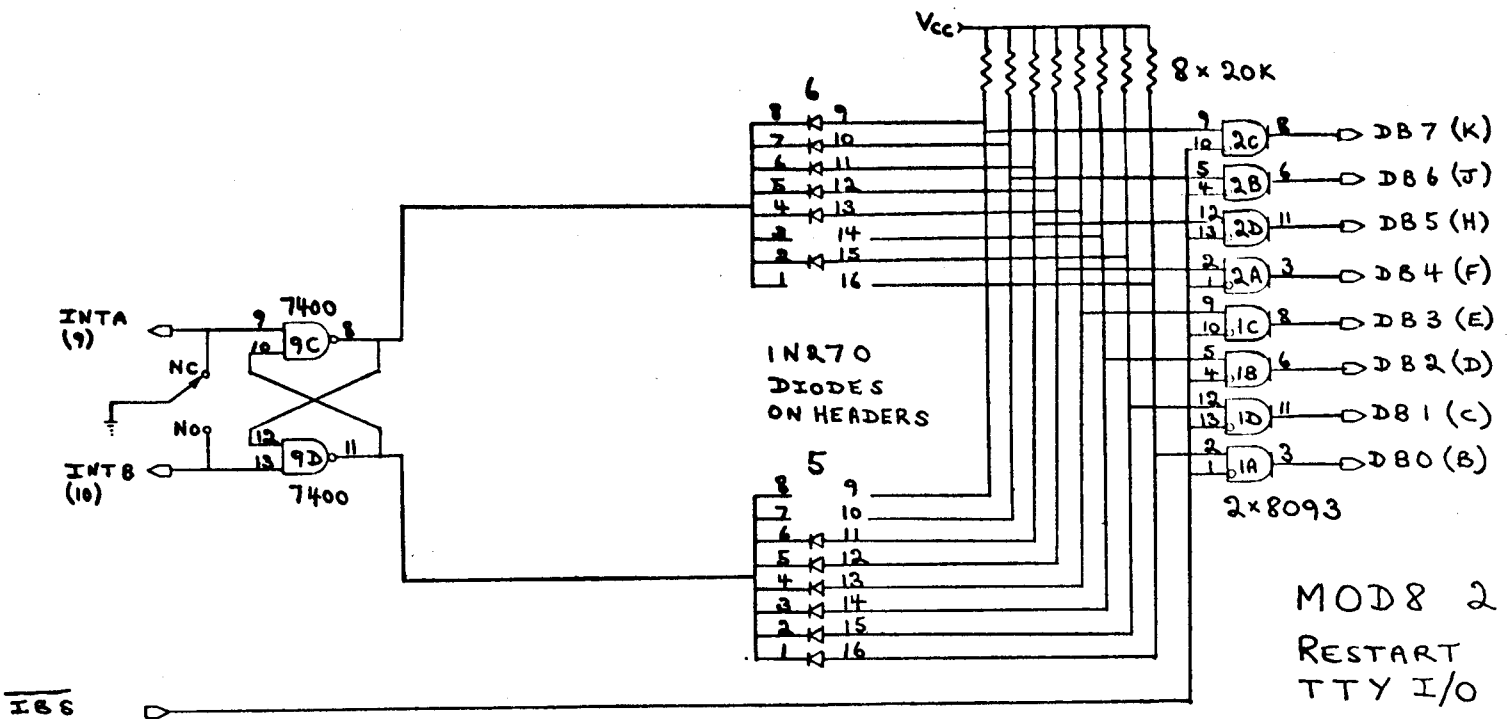


4

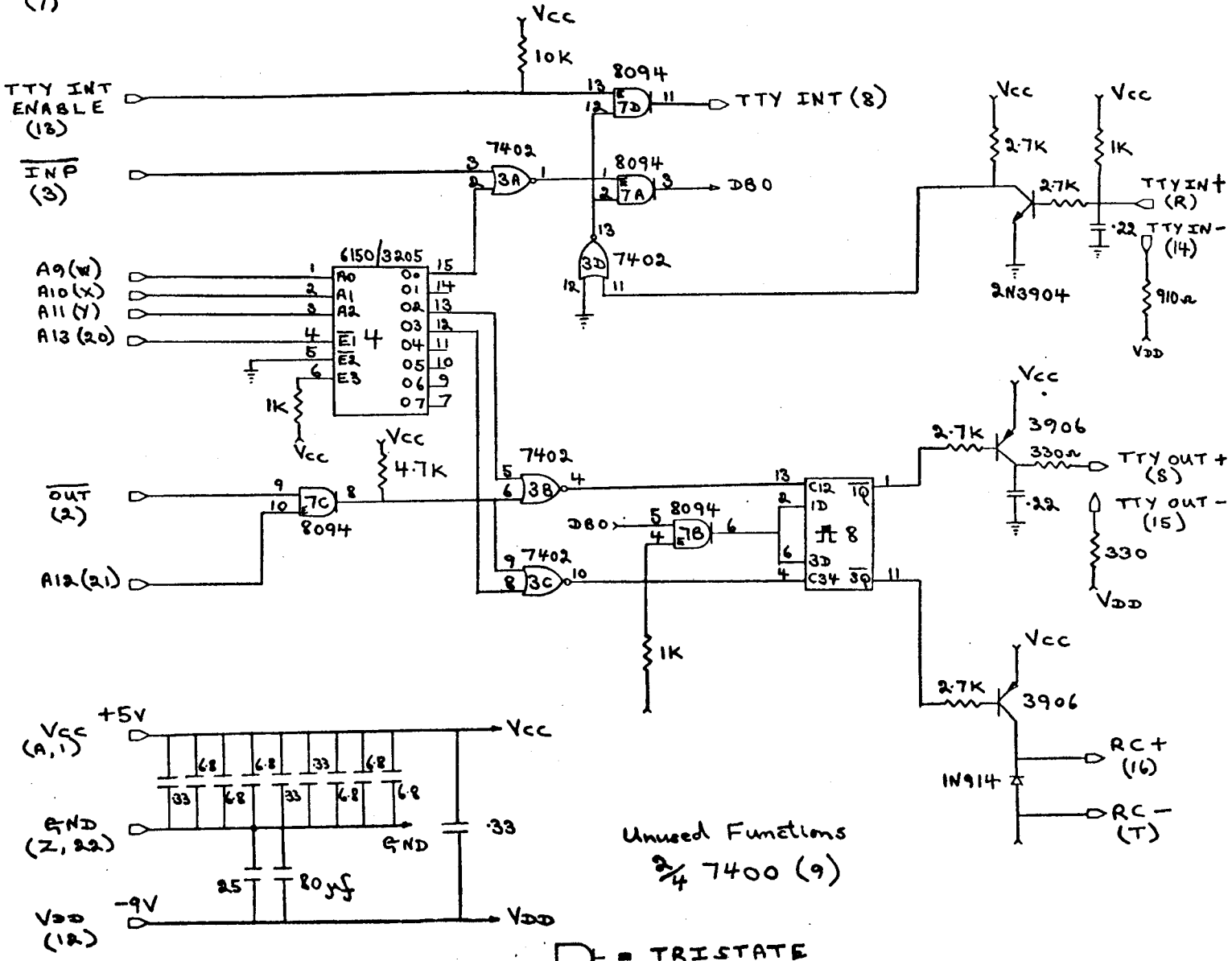


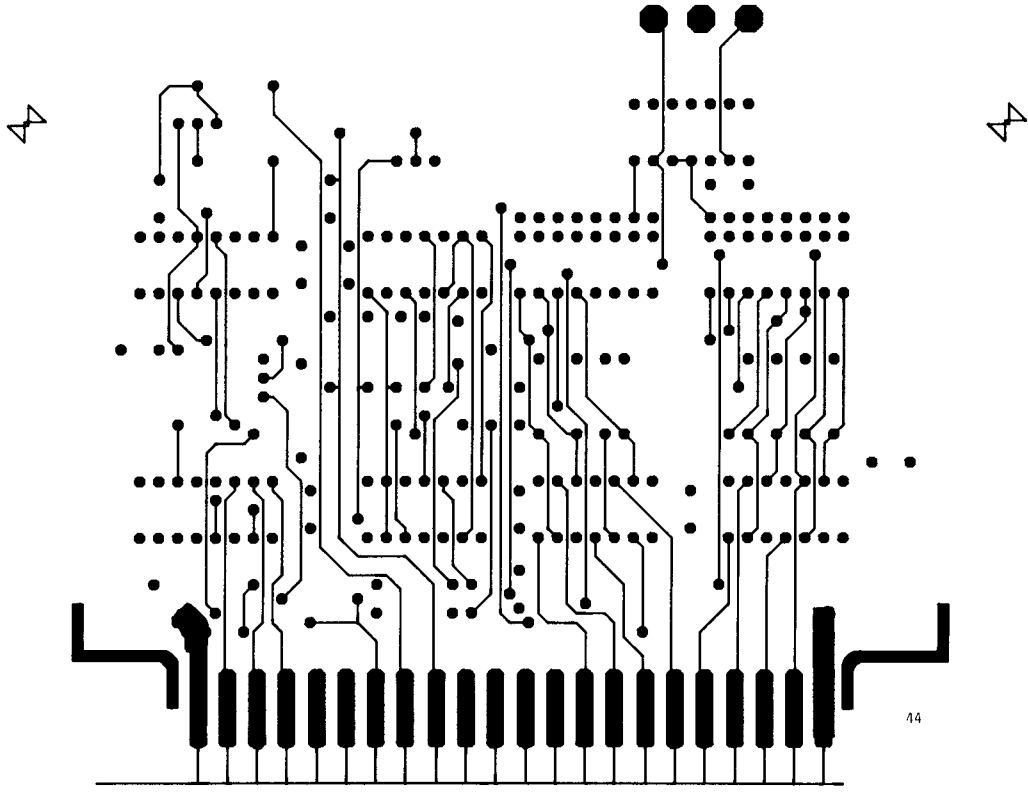
MOD 8 1

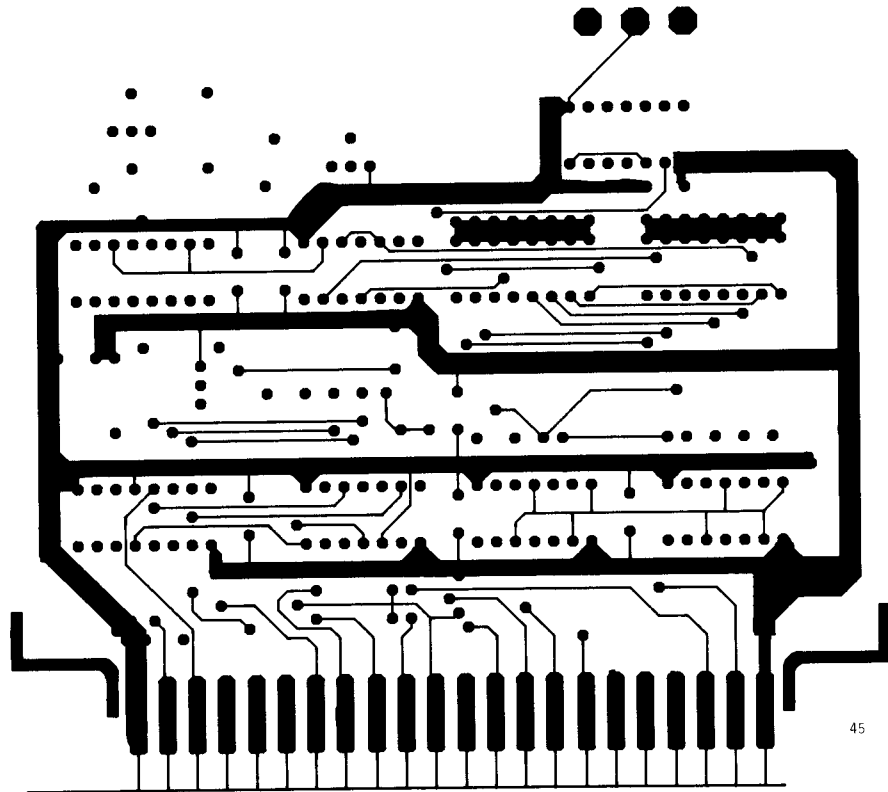




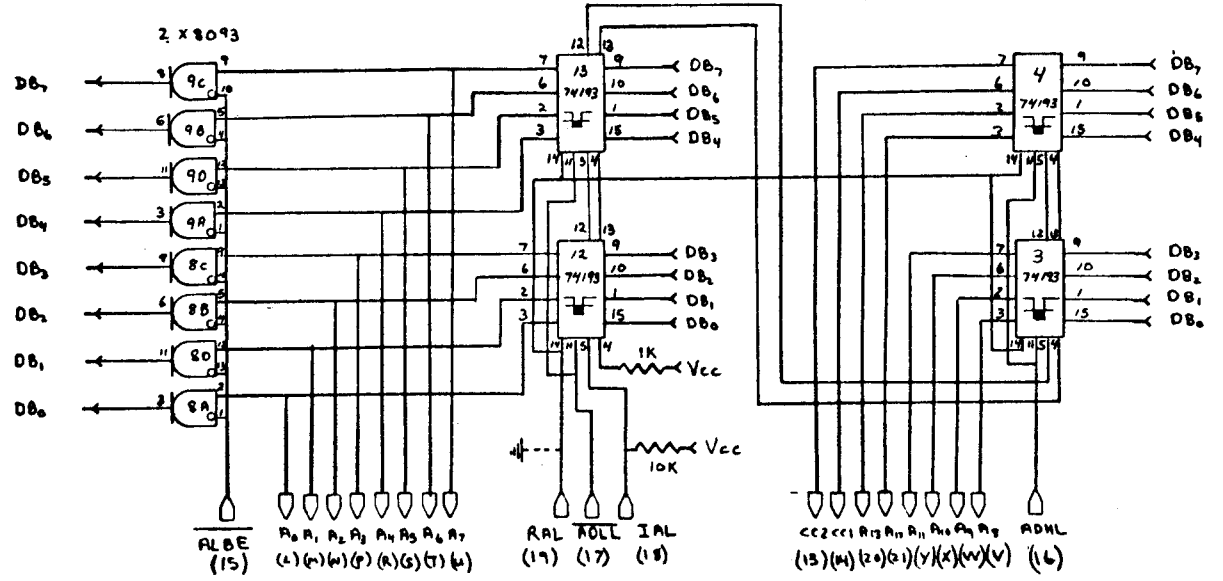
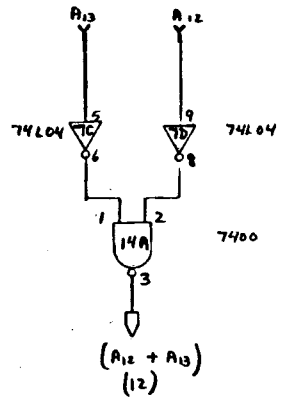
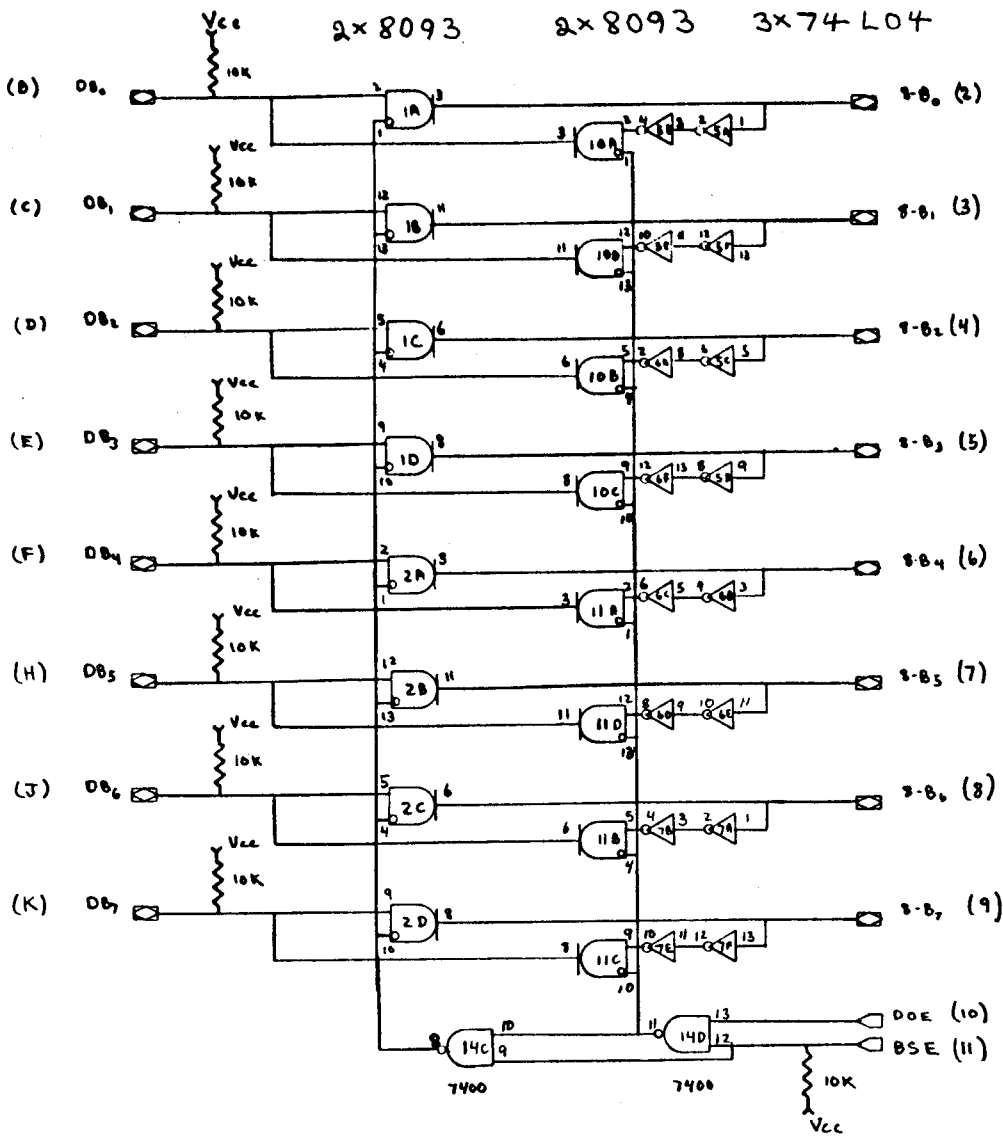
I 8 6
(7)



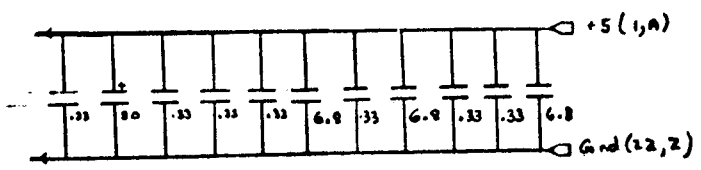




MOD 8 3
CONTROL
BUFFER.

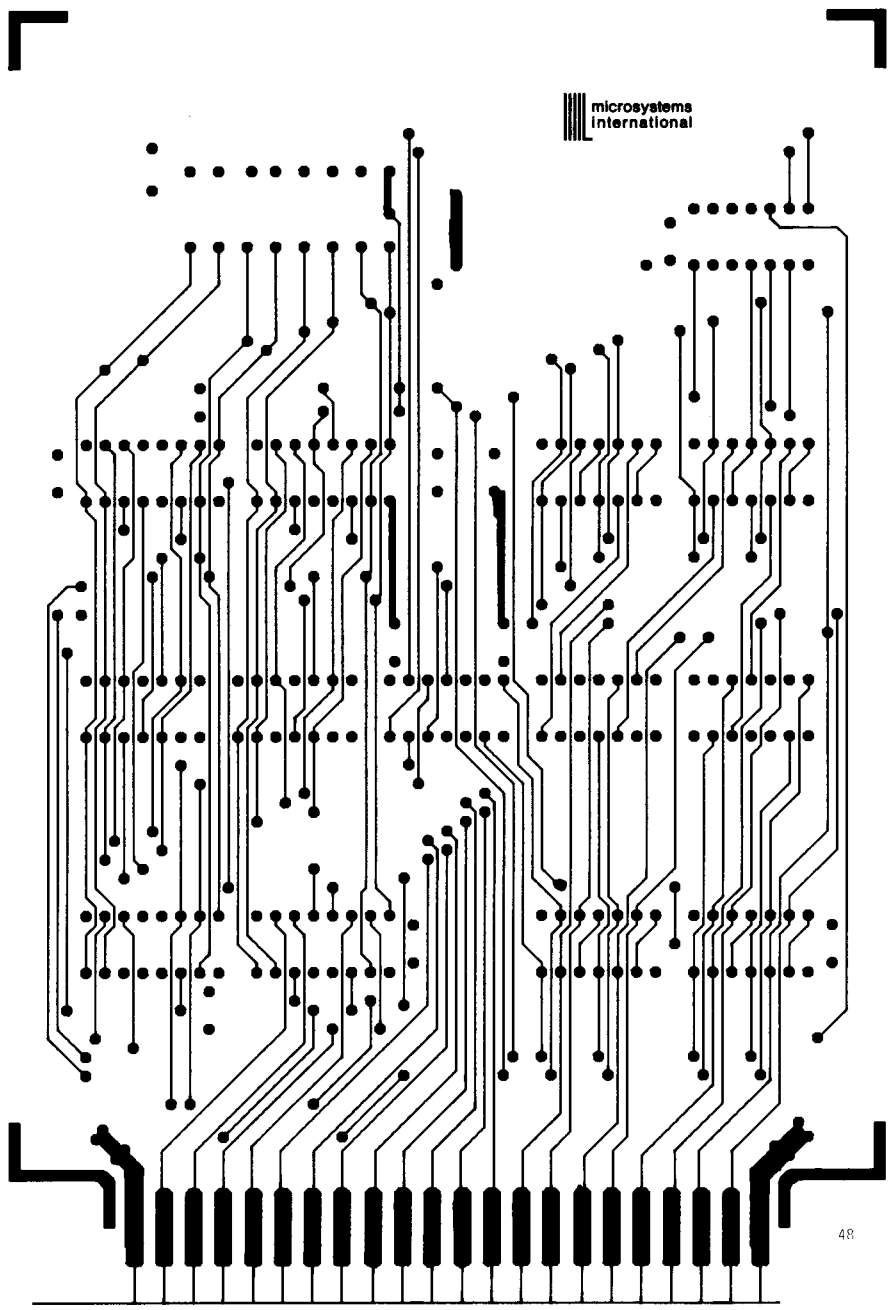


Unused functions:
1/4 7400 (14)



D - TRISTATE

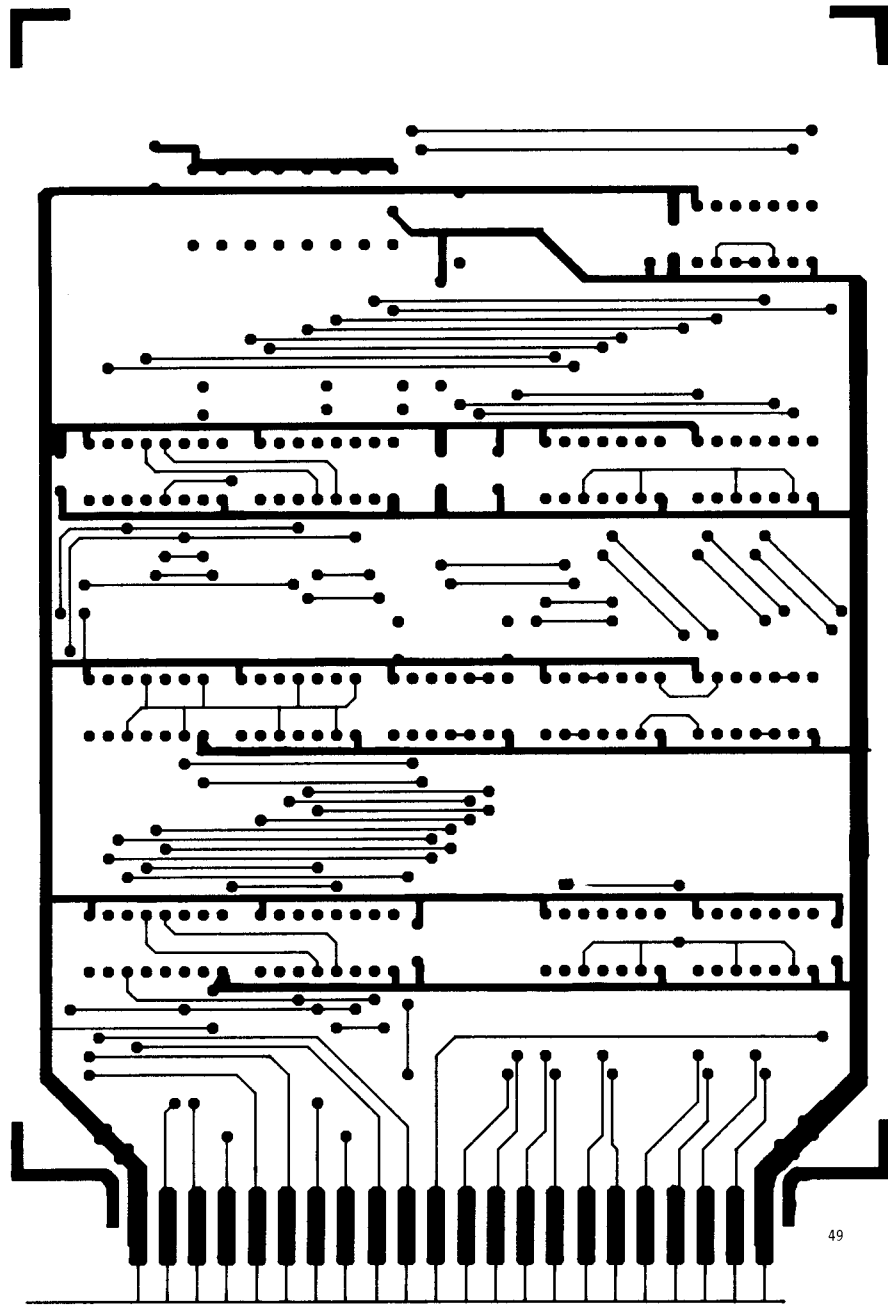
100
Part



4

4

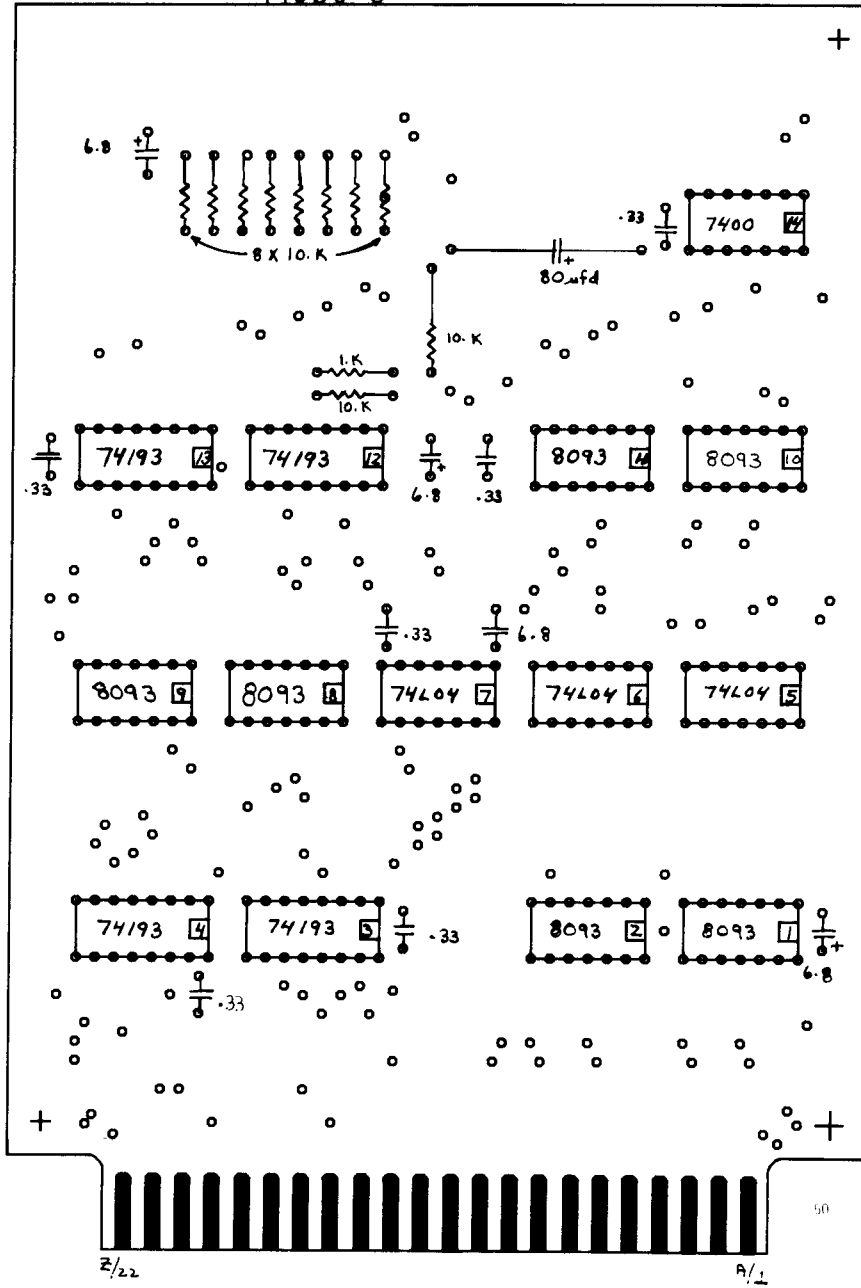
48



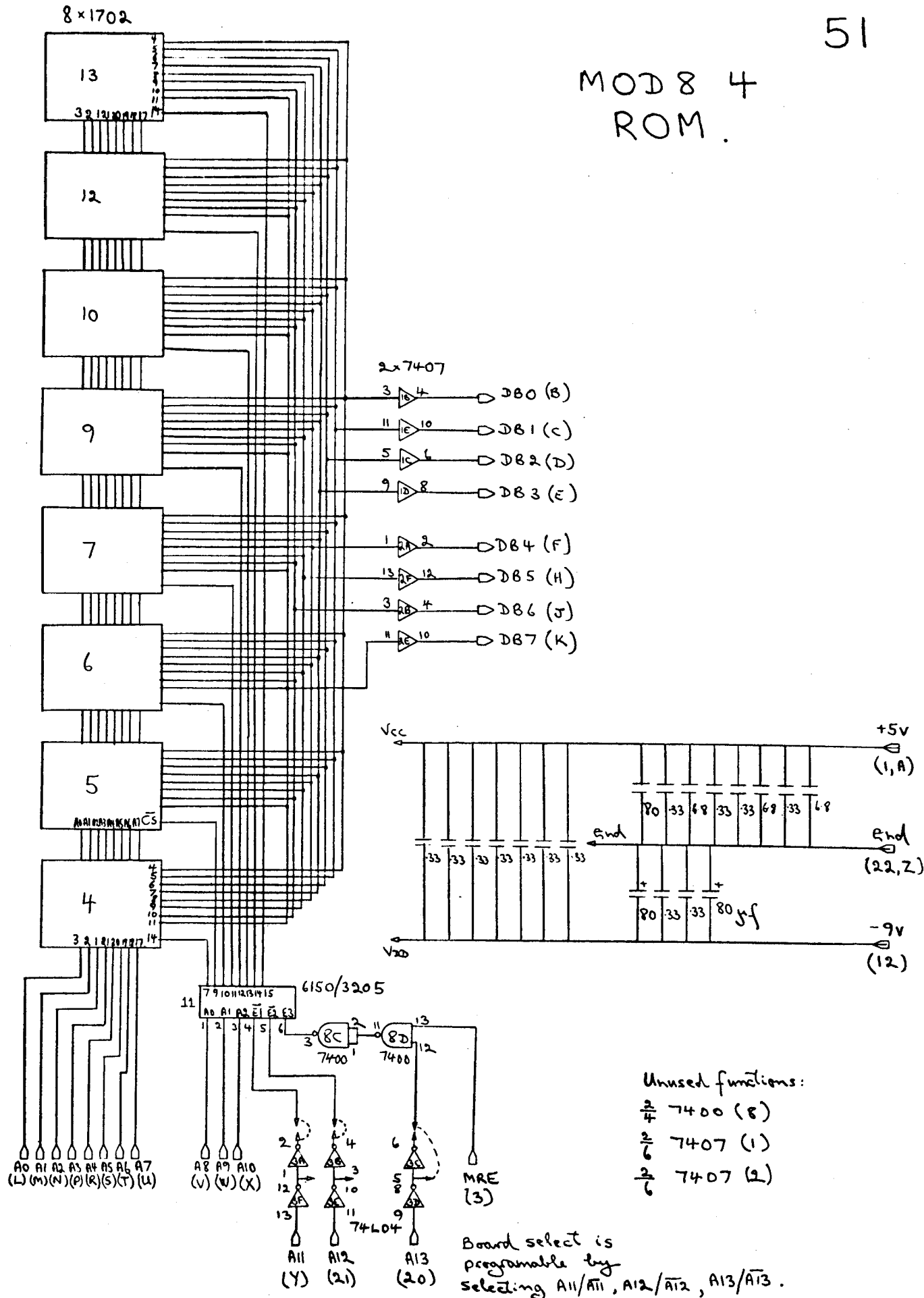
44

44

MODE 3



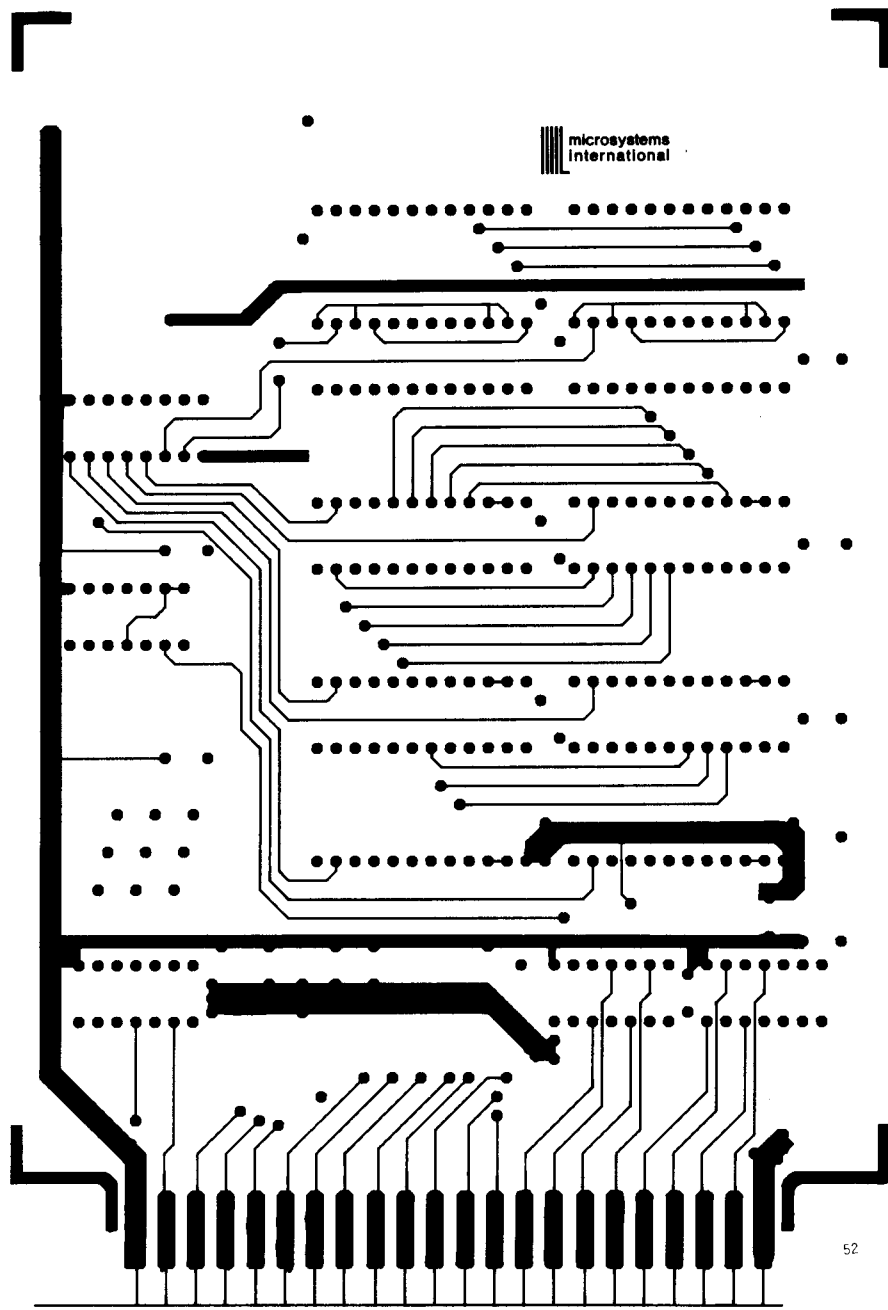
MOD 8 4
ROM.

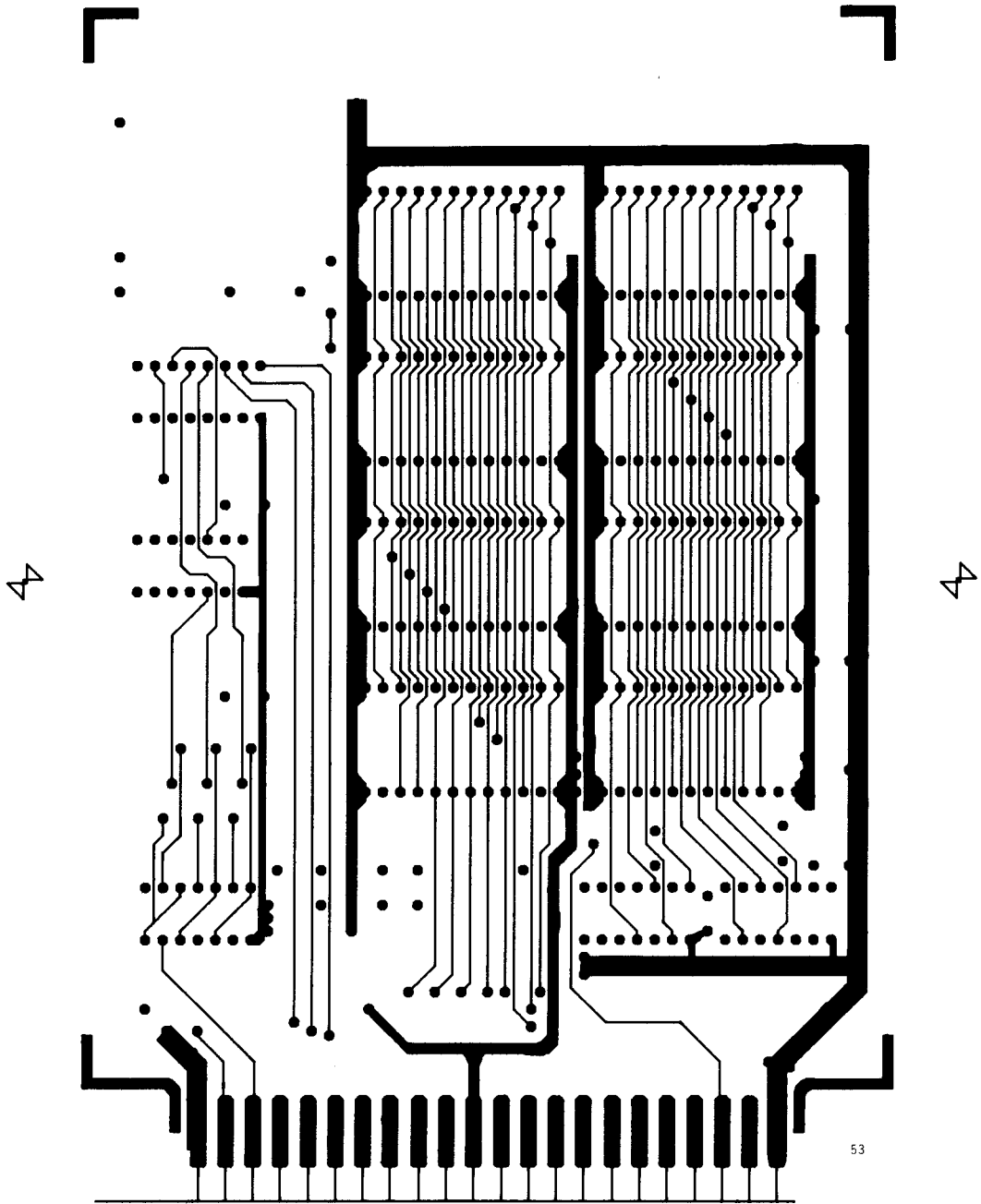


Note ϕ_1, ϕ_2, V_{max} and V_{DD} on 1702 connect to V_{CC} .
 V_{SS} on 1702 connects to V_{DD} .

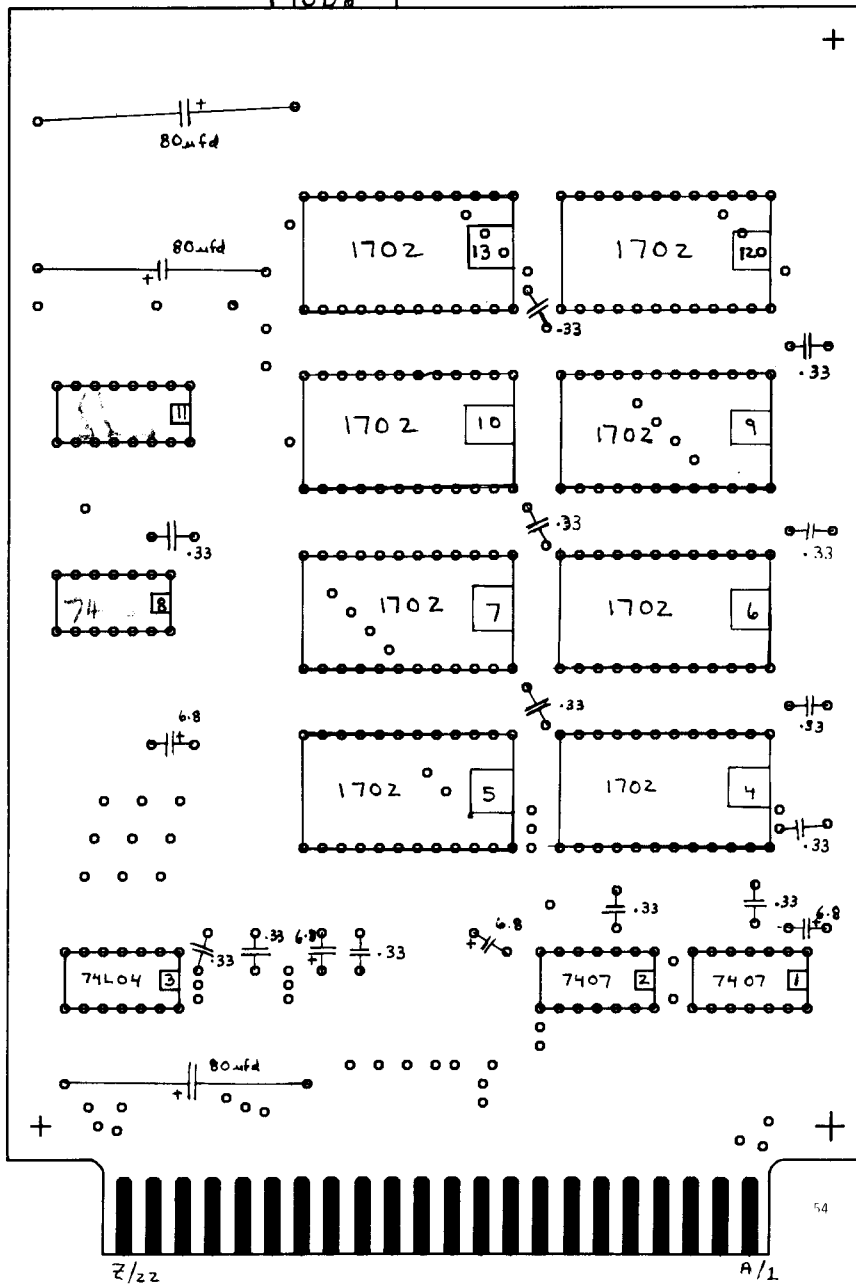
Board select is programmable by selecting $A_{11}/\bar{A}_{11}, A_{12}/\bar{A}_{12}, A_{13}/\bar{A}_{13}$.

0.0 8. 4
1.2 2.2





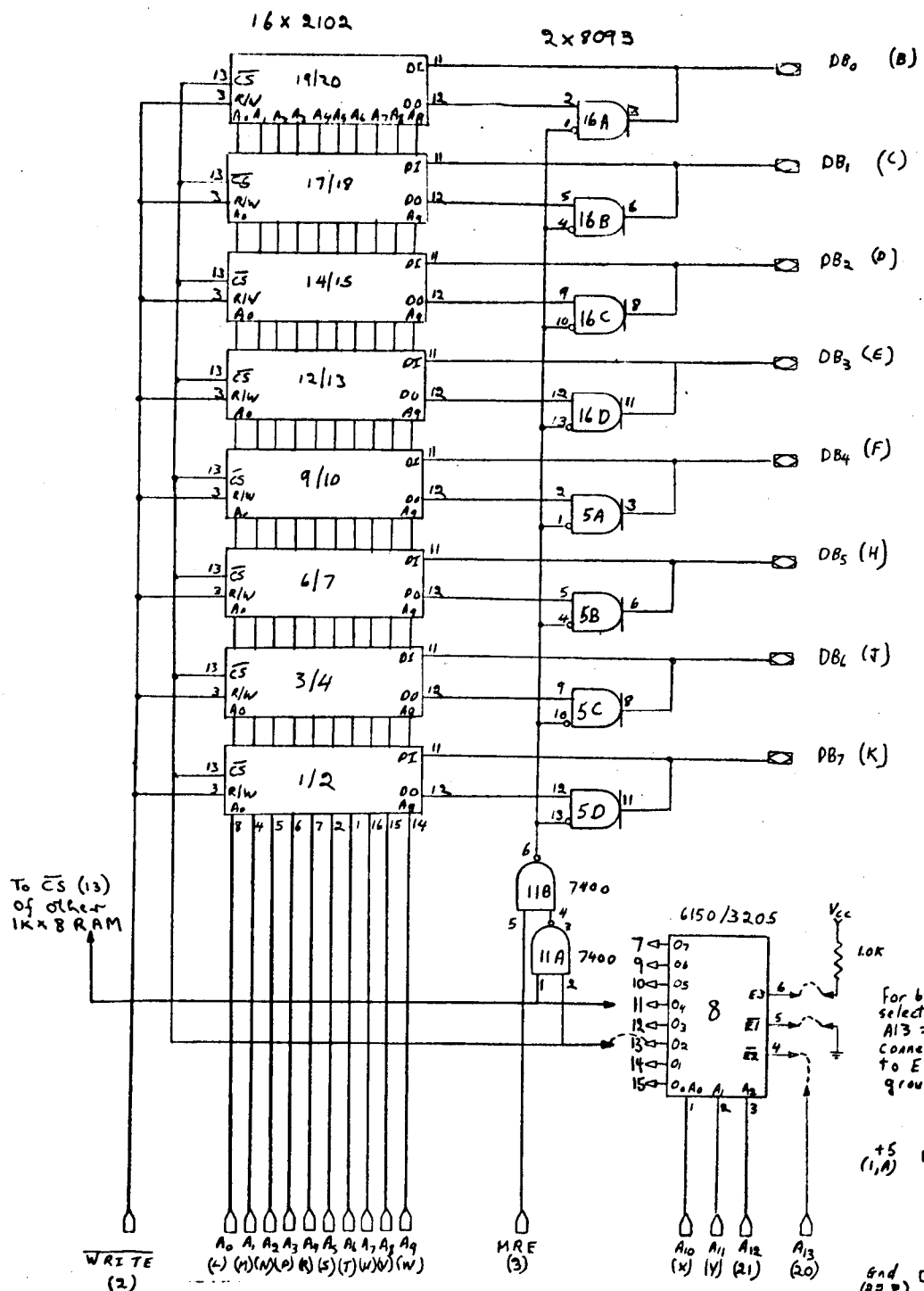
MOD 4



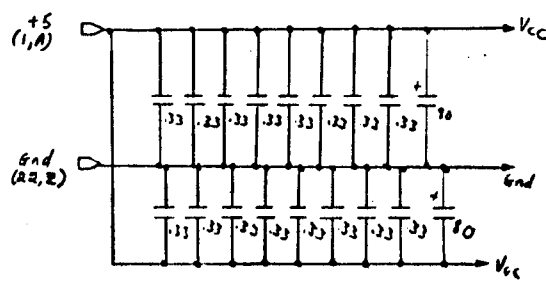
4

4

MOD 8 5 RAM



For board select on A13 = 1, connect (20) to E3 and ground E2



\square = Tri state

9.15 CPY (COPY ROUTINE)

TYPING CPY WILL INITIATE A COPY OF BLOCKS OF MEMORY. LIKE THE DUMP AND LOAD ROUTINES THIS ROUTINE REQUIRES A START ADDRESS AND AN END ADDRESS (DEFINING THE BLOCK TO BE MOVED). IN ADDITION AFTER THE BLOCK END ADDRESS HAS BEEN ENTERED, THE MACHINE WILL RESPOND WITH A CRLF* AND WAIT FOR THE ENTRY OF A THIRD ADDRESS, THE NEW START ADDRESS FOR THE BLOCK TO BE COPIED. AFTER THE THIRD ADDRESS HAS BEEN ENTERED, THE ENTIRE BLOCK SPECIFIED WILL BE COPIED UNCHANGED STARTING AT THE NEW START ADDRESS. WHEN THE COPY HAS BEEN COMPLETED CONTROL RETURNS TO THE MONITOR.

9.16 TRN (TRANSLATE ROUTINE)

TYPING TRN WHEN IN THE MONITOR MODE INITIATES THE TRANSLATE. THIS ROUTINE IS INTENDED FOR USE AFTER A PROGRAM IS RUNNING IN RAM AND IT IS DESIRED TO STORE IT IN PROM WHICH WILL RESIDE IN A DIFFERENT BANK. NO MOVEMENT OF DATA OCCURS, BUT ALL JUMP AND CALL ADDRESSES WHICH ARE INTERNAL TO THE BANK WILL BE CHANGED TO REFLECT THE NEW SPECIFIED BANK. THIS ROUTINE AGAIN REQUIRES A START OF BLOCK AND AN END OF BLOCK ADDRESS, TO DEFINE THE BLOCK TO BE OPERATED ON. AFTER THE SECOND ADDRESS HAS BEEN ENTERED, THE MACHINE RESPONDS WITH A CRLF. THE MACHINE IS NOW WAITING FOR TWO THREE DIGIT OCTAL BANK NUMBERS (POSSIBLE RANGE 000 TO 077). AFTER THE FIRST BANK NUMBER HAS BEEN ENTERED (THE SOURCE BANK NUMBER), THE MACHINE RESPONDS WITH A ← AND WAITS FOR THE SECOND BANK NUMBER (THE DESTINATION BANK NUMBER). AFTER THE SECOND BANK NUMBER HAS BEEN ENTERED, THE MACHINE SEARCHES THE SPECIFIED BLOCK FOR ALL CALL AND JUMP REFERENCES TO THE SOURCE BANK AND CHANGES THESE TO REFER TO THE DESTINATION BANK. WHEN THE CHANGES HAVE BEEN COMPLETED, THE MACHINE RETURNS TO THE MONITOR MODE.

9.17 SBP (SET BREAK-POINT)

BREAK-POINTS ALLOW THE TRACING OF PROGRAM FLOW DURING ITS EXECUTION. IF A RST 060 COMMAND IS ENCOUNTERED DURING PROGRAM EXECUTION THE MONITOR SOFTWARE WILL PRINT OUT THE CONTENTS OF THE CARRY FLAG, A B C H AND L REGISTERS, THE MEMORY CONTENTS ADDRESSED BY THE H AND L REGISTERS AND THEN RETURN TO THE MONITOR SOFTWARE.

THE SBP COMMAND INSERTS A RST 060 COMMAND AT THE ADDRESS SPECIFIED BY THE USER. THE ADDRESS AT WHICH THE BREAK-POINT IS INSERTED AND THE INSTRUCTION ORIGINALLY FOUND THERE IS RETAINED BY THE MONITOR. BEFORE SETTING SUBSEQUENT BREAK-POINTS, THE MONITOR WILL FIRST RESTORE THE DATA AT THE PREVIOUS BREAK-POINT LOCATION.

9.18 CBP (CLEAR BREAK-POINT)

THE CBP COMMAND WILL RESTORE THE DATA AT THE PRESENT BREAK-POINT LOCATION.

9.19 PRG (PROGRAM pROM)

THE MONITOR SOFTWARE ALSO CONTAINS THE FACILITY FOR CONTROLLING A pROM PROGRAMMING STATION IF ONE IS ATTACHED TO THE SYSTEM. THE PROGRAMMING ROUTINE IS ENTERED BY TYPING PRG. THE PROGRAMMING ROUTINE WILL ALLOW PROGRAMMING A pROM WITH DATA PRESENTLY LOCATED IN MEMORY. AN INITIAL AND FINAL ADDRESS MUST BE SPECIFIED. THE ROUTINE WILL PROGRAM THE DATA FROM SPECIFIED LOCATION TO THE CORRESPONDING WORD LOCATION WITHIN THE ROM.

e.g. 010177 LOCATION 177 OF THE pROM

e.g. THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
THE ADDRESS BEING READ WITHIN A BANK AND THE
ADDRESS BEING PROGRAMMED IN THE pROM.

THE PROGRAMMING ROUTINE WILL FIRST CHECK IF THE pROM DATA IS EQUAL TO THE PROGRAM DATA. IF THE BYTE PATTERNS ARE IDENTICAL THE ROUTINE PROCEEDS TO THE NEXT ADDRESS. IF THE LOCATION MUST BE PROGRAMMED, THE pROM IS HIT WITH PROGRAMMING PULSES IN TWO SECOND INTERVALS FOR A MAXIMUM OF 10 CYCLES. AFTER EACH INTERVAL, THE DATA IS RECHECKED. ONCE THE DATA IS READ AS PROGRAMMED, A FINAL 2 SECOND HIT CYCLE IS EXECUTED BEFORE PROCEEDING TO THE NEXT LOCATION. IF THE pROM FAILS TO PROGRAM AFTER 10 TRIES THE CURRENT LOCATION IS PRINTED FOLLOWED BY A ? AND THE ROUTINE RETURNS TO THE MONITOR.

CONTROL A

INCLUDED IN THE TTY INPUT ROUTINE IS A CHECK FOR THE CTRLA KEY. DEPRESSING THE CTRL BUTTON AND A KEY SIMULTANEOUSLY WILL CAUSE THE MACHINE TO IMMEDIATELY RETURN TO THE MONITOR ROUTINE, AND IS EQUIVALENT TO A MONITOR RESTART.

RUBOUT

OCTAL DATA INPUT ROUTINES WILL ACCEPT A RUBOUT COMMAND. EACH TIME THE RUBOUT KEY IS PRESSED A + SYMBOL IS PRINTED AND A CHARACTER IS DELETED. TYPING TWO RUBOUTS WILL DELETE TWO CHARACTERS ETC. THE RUBOUT ROUTINE FOR OCTAL VALUES WILL "BACK SPACE" ONLY TO THE BEGINNING OF THE FIELD. DATA IS REPRESENTED BY 1 FIELD (OR BYTE) WHEREAS ADDRESSES ARE REPRESENTED BY TWO BYTES (FIELDS). THE ROUTINE WILL TYPE A + FOR EACH RUBOUT UNTIL IT REACHES THE BEGINNING OF THE FIELD WHERE IT WILL ACCEPT A RUBOUT BUT WILL NOT TYPE ANY SYMBOL AND WILL NOT CONTINUE TO BACK SPACE.

10. MONITOR 8 SOFTWARE LISTINGS

THE FOLLOWING SECTION CONTAINS A COMPLETE LISTING OF THE MONITOR 8 SOFTWARE. IN ADDITION PAGE 76 CONTAINS A LIST OF THE EIGHT RESET POINTS (RESTART 0-7) AND A LIST OF ENTRY POINTS FOR THE MONITOR 8 SUBROUTINES. TO SAVE THE TIME REQUIRED TO RECODE THIS SOFTWARE, THE COMPLETE SOFTWARE PACKAGE, 7 ROM'S, MAY BE PURCHASED FROM MICROSYSTEMS INTERNATIONAL LTD. AT A NOMINAL SURCHARGE OVER THE NORMAL COMPONENT PRICE, TO COVER THE COST OF PROGRAMMING THE ROM'S.

RESET NO.	RESET INDEX FUNCTION
RST 000	COLD START, GENERAL RESTART
RST 010	GO TO ROM 7 (FOR USER)
RST 020	OUTPUT AN ASCII CHARACTER
RST 030	INPUT AN ASCII CHARACTER
RST 040	TEST FOR RUBOUT
RST 050	SEARCH FOR CHARACTER IN 'E' REGISTER
RST 060	BREAKPOINT EXECUTE
RST 070	TIMING LOOP

SUBROUTINE INDEX
(START ADDRESSES OF MANY OF THE ROUTINES USED HERE,
WHICH MAY BE USABLE IN OTHER SOFTWARE)

START ADDRESS	FUNCTION
000013	OUTPUT CARRIAGE RETURN AND LINE FEED
000177	TEST FOR OCTAL CHARACTER
000205	3 DIGIT OCTAL INPUT (COMPRESSED TO 1 BYTE)
000253	3 DIGIT OCTAL OUTPUT (USED TO DISPLAY 1 BYTE)
000311	ADDRESS INCREMENT (USES CLP-L0C 013377,013376)
000326	ADDRESS DECREMENT
000344	ADDRESS COMPARE (CLP,CLP-1)
000362	COMPARE AND INCREMENT (USED TO TEST FOR END OF ROUTINE)
001000	OCTAL DUMP (DP0)
001023	FETCH DATA FROM LOCATION ADDRESSED BY CLP
001047	DISPLAY DATA AT CLP
001055	DISPLAY BLANK, CLP (ADDRESS)
001073	OUTPUT CR/LF, CLP
001111	PUT DATA INTO CLP
001120	OCTAL INPUT (LD0)
001200	INPUT AN ADDRESS (2 BYTES)
001236	OCTAL EDITOR (EDT)
001336	INDIRECT JUMP
001353	CLEAR BREAKPOINT (CBP)
002000	PROM PROGRAMMING ROUTINE (PRG)
002110	SET UP CLP (L0C)
002115	DUMP IN BNPFF FORMAT (DBF)
002201	LOAD IN BNPFF FORMAT (LBF)
002257	BANK TO BANK TRANSLATE (TRN)
002347	SET BREAKPOINT (SBP)
003000	CONTROLLER ROUTINE
003131	GENERAL ERROR ROUTINE
003150	TABLE SEARCH
003244	BREAKPOINT EXECUTE
005063	REGISTER DEC0DE
005313	PRINT 3 ASCII BYTES

*N

```

000000/ 006 LAI 001 (RST 000) COLD START
000002/ 125 OUT 012 IDLE TTY
000003/ 250 XRA
000004/ 127 OUT 013 IDLE PTR
000005/ 104 JMP 003000 GO TO CONTROLLER
000010/ 104 JMP 007000 (RST 010) USERS ROUTINE
000013/ 016 LBI 215 (CR) CR/LF ROUTINE
000015/ 025 RST 020
000016/ 016 LBI 212 (LF)
000020/ 026 LCI 375 (RST 020) O/P ONE CHARACTER
000022/ 036 LDI 177 SET UP TIMING
000024/ 075 RST 070 1ST BIT IS LONGER
000025/ 104 JMP 000140 CONTINUED ELSEWHERE
000030/ 006 LAI 001 (RST 030) I/P CHARACTER
000032/ 127 OUT 013 ENABLE PTR
000033/ 036 LDI 302 SET UP TIMING
000035/ 104 JMP 000075 CONTINUED
000040/ 006 LAI 177 (RST 040) RUBOUT TEST
000042/ 271 CPB
000043/ 013 RFZ NOT RUBOUT SO RETURN
000044/ 016 LBI 337 O/P ARROW
000046/ 025 RST 020
000047/ 007 RET
000050/ 035 RST 030 FLAG SET TO IGNORE INPUT
000051/ 301 LAB (RST 050) SEARCH FOR CHAR
000052/ 274 CPE FETCH I/P IN REG E
000053/ 053 RTZ COMPARE
000054/ 104 JMP 000050 GOT CHAR TRY NEXT ONE
000057/ 377 HLT UNUSED BYTE
000060/ 104 JMP 003244 (RST 060) XQT BRKPT
000063/ 301 LAB I/P (CONT)
000064/ 074 CPI 001 CNTRL A I/P
000066/ 013 RFZ NO- GO AHEAD
000067/ 005 RST 000 YES- PANIC AND RESTART
000070/ 030 IND (RST 070) TIMING LOOP
000071/ 110 JFZ 000070 LOOPING
000074/ 007 RET DONE
000075/ 377 HLT WAIT FOR I/P I/P(CONT)
000076/ 075 RST 070 TIME 1ST BIT
000077/ 250 XRA CLEAR A REG
000100/ 127 OUT 013 IDLE PTR FOR NOW
000101/ 125 OUT 012 START O/P
000102/ 026 LCI 370 SET UP 1 BIT DELAY
000104/ 036 LDI 171
000106/ 075 RST 070 WAIT FOR IT
000107/ 101 INP 000 GET BIT
000110/ 054 XRI 377 COMPLEMENT I/P
000112/ 125 OUT 012 ECHO TO O/P
000113/ 032 RAR ROTATE INTO B
000114/ 301 LAB WITH PREVIOUS
000115/ 032 RAR BITS
000116/ 310 LBA
000117/ 020 INC BUMP COUNTER
000120/ 110 JFZ 000104 LOOP FOR MORE BITS
000123/ 301 LAB GOT 8 BITS NOW
000124/ 044 NDI 177 IGNORE PARITY (MSB)
000126/ 310 LBA
000127/ 036 LDI 171 O/P STOP
000131/ 075 RST 070 AND O/P IDLE STATE
000132/ 006 LAI 001
000134/ 125 OUT 012
000135/ 104 JMP 000063 TO BE CONTINUED

```

```

000140/ 020 INC                                0/P (CONT)
000141/ 110 JFZ 000022 KEEP TIMING
000144/ 250 XRA                                CLEAR A
000145/ 125 0UT 012                          START 0/P
000146/ 026 LCI 370                          SET UP TIMING
000150/ 036 LDI 171
000152/ 075 RST 070                          WAIT FOR NEXT BIT
000153/ 301 LAB                              FETCH BIT FROM B
000154/ 125 0UT 012                          AND OUTPUT BIT
000155/ 032 RAR                              NOW SET UP THE NEXT
000156/ 310 LBA                              BIT, STORE IT IN B
000157/ 006 LAI 000
000161/ 032 RAR
000162/ 201 ADB
000163/ 310 LBA
000164/ 020 INC                                BUMP COUNT
000165/ 110 JFZ 000150 MORE TO 0/P, SO LOOP
000170/ 036 LDI 171                          DONE
000172/ 075 RST 070                          0/P STOP AND IDLE BITS
000173/ 006 LAI 001
000175/ 125 0UT 012
000176/ 007 RET                                GOODBYE
000177/ 035 RST 030                          FETCH CHAR    TEST FOR OCTAL
000200/ 044 NDI 370                          MASK 3 BITS
000202/ 074 CPI 060                          IS IT 06X? (ZF TELLS ALL)
000204/ 007 RET                                GO AWAY
000205/ 106 CAL 000177 GET DIGIT    OCTAL I/P
000210/ 110 JFZ 000205 NOT OCTAL, TRY AGAIN
000213/ 301 LAB                              PUT DIGIT IN A
000214/ 012 RRC                              ROTATE
000215/ 012 RRC                              ROTATE
000216/ 370 LMA                              STASH IT
000217/ 035 RST 030                          FETCH DIGIT
000220/ 045 RST 040                          TEST FOR RUBOUT
000221/ 150 JIZ 000205 TRY AGAIN
000224/ 307 LAM                              FETCH LAST DIGIT
000225/ 044 NDI 300                          MASK UNUSED BITS
000227/ 370 LMA                              STORE IT
000230/ 006 LAI 007                          PUT IN 3 MORE BITS
000232/ 241 NDB                              ROTATE INTO POSITION
000233/ 002 RLC
000234/ 002 RLC
000235/ 002 RLC
000236/ 207 ADM                              ADD IN THE OLD DATA
000237/ 370 LMA                              STORE IT
000240/ 035 RST 030                          FETCH DIGIT    ENTRY FOR
000241/ 045 RST 040                          TEST          CORRECTION DONE
000242/ 150 JIZ 000217 RUBOUT
000245/ 006 LAI 007                          MASK ALL BUT 3 BITS
000247/ 241 NDB
000250/ 207 ADM                              ADD THE PREVIOUS BITS
000251/ 370 LMA                              STASH DATA
000252/ 007 RET                              DONE
000253/ 016 LBI 240                          (BLANK)      OCTAL 0/P
000255/ 025 RST 020                          3 DIGITS
000256/ 307 LAM                              FETCH BYTE
000257/ 002 RLC                              MOVE BITS 7 AND 8
000260/ 002 RLC                              TO POS 1 AND 2
000261/ 044 NDI 003                          MASK THE REST
000263/ 004 ADI 260                          CONVERT TO ASCII
000265/ 310 LBA                              SET UP FOR 0/P
000266/ 025 RST 020                          0/P

```

000267/	307	LAM	FECTH BYTE	
000270/	012	RRC	SET UP BITS 4,5,6	
000271/	012	RRC		
000272/	012	RRC		
000273/	044	NDI 007	MASK	
000275/	004	ADI 260	CØNVERT	
000277/	310	LBA		
000300/	025	RST 020	Ø/P	
000301/	307	LAM	BITS 1,2,3 THIS TIME	
000302/	044	NDI 007		
000304/	004	ADI 260		
000306/	310	LBA	NØW Ø/P	
000307/	025	RST 020		
000310/	007	RET	ALL DØNE	
000311/	066	LLI 376	SET TØ CLP	ADDRESS INCR
000313/	056	LHI 013		2 BYTES
000315/	317	LBM	FETCH	
000316/	010	INB	INCR LSB	
000317/	371	LMB	STØRE	
000320/	013	RFZ	CARRY	
000321/	060	INL	YES-INCR MSB	
000322/	317	LBM	FETCH	
000323/	010	INB	INCR	
000324/	371	LMB	STØRE	
000325/	007	RET	DØNE	
000326/	066	LLI 376	SET TØ CLP	ADDRESS DECR
000330/	056	LHI 013		2 BYTES
000332/	317	LBM	FETCH	
000333/	011	DCB	DECR	
000334/	371	LMB	STØRE	
000335/	060	INL	PØINT TØ MSB	
000336/	010	INB	WAS LSB	
000337/	013	RFZ	NØ-RETURN	
000340/	317	LBM	YES-DØ MSB	
000341/	011	DCB		
000342/	371	LMB		
000343/	007	RET	DØNE	
000344/	066	LLI 377	SET TØ CLP	ADDRESS CØMP
000346/	056	LHI 013		4 BYTES
000350/	307	LAM	FETCH MSB	
000351/	061	DCL		
000352/	317	LBM	FETCH LSB	
000353/	061	DCL		
000354/	277	CPM	CØMPARE MSB	
000355/	013	RFZ	NØT EQUAL-RETURN	
000356/	301	LAB	PUT LSB INTØ A	
000357/	061	DCL		
000360/	277	CPM	CØMPARE	
000361/	007	RET	GØ AWAY (ZF=1 IF EQUAL)	
000362/	106	CAL 000344	CØMP ADDR	CØMP ADDR AND
000365/	110	JFZ 000311	INCR ADDR	INCR IF NØT =
000370/	005	RST 000	NØW RESTART	
000371/	300	LAA		UNUSED LOCATIONS
000372/	300	LAA		
000373/	300	LAA		
000374/	300	LAA		
000375/	300	LAA		
000376/	300	LAA		
000377/	300	LAA		
001000/	377	HLT	WAIT	ØCTAL DUMP (DPØ)
001001/	046	LEI 010	SET UP CHAR/LINE	
001003/	106	CAL 001073	Ø/P CLP	


```

001006/ 106 CAL 001047 0/P CLP CONTENTS
001011/ 106 CAL 000362 INCR AND COMPARE CLP
001014/ 041 DCE INCR LINE COUNT
001015/ 150 JTZ 001001 NEW LINE, PRINT ADR
001020/ 104 JMP 001006 SAME LINE, JUST LOOP
001023/ 066 LLI 377 SET TO CLP GET DATA (FROM CLP)
001025/ 056 LHI 013 EXTENDED ADDRESS
001027/ 307 LAM MS IN 'A' PUTS DATA INTO 013370
001030/ 061 DCL
001031/ 367 LLM LS IN L
001032/ 350 LHA NOW PUT MS IN H
001033/ 002 RLC ROTATE TO TEST BIT 8
001034/ 307 LAM FETCH LS
001035/ 003 RFC RETURN IF NOT EXTENDED MEMORY
001036/ 056 LHI 013 GET DATA FROM I/P PORT
001040/ 306 LAL SET TEMP STORE LOCATION
001041/ 066 LLI 370
001043/ 121 OUT 010 0/P LS
001044/ 103 INP 001 GET DATA
001045/ 370 LMA PUT INTO MEMORY
001046/ 007 RET GO!
001047/ 106 CAL 001023 GET DATA FROM CLP
001052/ 104 JMP 000253 AND PRINT IT
001055/ 066 LLI 377 SET TO CLP 0/ 'HHHLLL'
001057/ 056 LHI 013
001061/ 016 LBI 240 0/P BLANK
001063/ 025 RST 020
001064/ 106 CAL 000256 0/P MS BYTE
001067/ 061 DCL 0/P LS BYTE
001070/ 104 JMP 000256 AND RETURN TO CALL WHEN DONE
001073/ 066 LLI 377 SET TO CLP PRINT CR/LF HHHLLL
001075/ 056 LHI 013
001077/ 106 CAL 000013 0/P CR/LF
001102/ 106 CAL 001064 0/P ADR
001105/ 016 LBI 257 0/P SLASH
001107/ 025 RST 020
001110/ 007 RET DONE
001111/ 106 CAL 001023 SET TO CLP GET CLP PUT DATA THERE
001114/ 104 JMP 000205 FETCH DATA
001117/ 300 LAA NOP NOT USED
001120/ 106 CAL 000013 0/P CR/LF OCTAL INPUT (LD0)
001123/ 046 LEI 057 SEARCH FOR SLASH (/)
001125/ 055 RST 050
001126/ 035 RST 030 FETCH CHARACTER
001127/ 301 LAB
001130/ 074 CPI 015 IS IT A CR
001132/ 150 JTZ 001123 YES- WAIT FOR ANOTHER SLASH
001135/ 106 CAL 001111 NO- PUT DATA AT CLP
001140/ 106 CAL 000362 COMPARE AND INCR CLP
001143/ 104 JMP 001126 LOOP
001146/ 066 LLI 373 SET UP 'L' COPY (CPY)
001150/ 106 CAL 001205 INPUT NEW START OF BLOCK
001153/ 106 CAL 001023 SET UP H AND L
001156/ 327 LCM FETCH DATA
001157/ 066 LLI 373
001161/ 106 CAL 001025 SET H,L TO NEW ADR
001164/ 372 LMC STORE DATA
001165/ 106 CAL 000362 INCR FROM ADR
001170/ 066 LLI 372
001172/ 106 CAL 000313 INCR TO ADR
001175/ 104 JMP 001153 LOOP
001200/ 066 LLI 377 SET CLP GET ADDRESS (2 BYTES)

```

```

001202/ 106 CAL 000013 Ø/P CR/LF
001205/ 016 LBI 252 Ø/P *
001207/ 025 RST 020
001210/ 056 LHI 013 CLP PAGE
001212/ 106 CAL 000205 GET A BYTE (MS)
001215/ 061 DCL SET L FØR LS
001216/ 035 RST 030 GET NEXT BYTE
001217/ 045 RST 040 RUB-ØUT?
001220/ 110 JFZ 000213 NØ-GET THE NEW BYTE AS BEFØRE
001223/ 060 INL YES-RESTØRE L TØ MS ADR
001224/ 307 LAM FETCH MS BYTE
001225/ 044 NDI 370 MASK 3 BITS
001227/ 370 LMA STØRE
001230/ 106 CAL 000240 GET 3 NEW BITS
001233/ 104 JMP 001215 NØW- THE LS BYTE
001236/ 106 CAL 001073 CR/LF+CLP ØCTAL EDITØR (EDT)
001241/ 106 CAL 001247 PRØCESS LINE (BYTE)
001244/ 104 JMP 001236 LØØP
001247/ 035 RST 030 FETCH I/P
001250/ 301 LAB
001251/ 074 CPI 122 TEST FØR 'R'
001253/ 150 JIZ 003014 YES-THEN RETURN
001256/ 074 CPI 052 TEST FØR '*'
001260/ 066 LLI 377 SET L TØ CLP
001262/ 150 JIZ 002110 GØ TØ LØC RØUTINE
001265/ 074 CPI 100 TEST FØR '@'
001267/ 150 JIZ 003320 GØ TØ XQT
001272/ 074 CPI 136 TEST FØR 'UP ARRØW'
001274/ 150 JIZ 000326 THEN DECR CLP
001277/ 074 CPI 040 TEST FØR BLANK
001301/ 150 JIZ 001321 PRINT THIS BYTE
001304/ 106 CAL 000200 FAILED ALL TESTS, IS I/P ØCTAL?
001307/ 013 RFZ NØ- IGNØRE IT
001310/ 106 CAL 001023 YES-SET H AND L
001313/ 106 CAL 000213 GET 2 MØRE DIGITS AND STØRE THE BYTE
001316/ 104 JMP 000311 INCR CLP AND LØØP
001321/ 106 CAL 001047 FETCH AND PRINT DATA
001324/ 035 RST 030 I/P MØRE
001325/ 301 LAB TØ 'A' REG
001326/ 074 CPI 137 IS IT BACK ARRØW
001330/ 152 CTZ 000205 YES-REPLACE DATA BYTE
001333/ 104 JMP 000311 INCR CLP AND LØØP
001336/ 066 LLI 371 INDIRECT JUMP
001340/ 056 LHI 013 SET H,L TØ UNUSED RAM
001342/ 076 LMI 104 STØRE 'JMP'
001344/ 060 INL
001345/ 371 LMB LS ADR IN 'B'
001346/ 060 INL
001347/ 370 LMA MS ADR IN 'A'
001350/ 104 JMP 013371 GØ JMP IN
001353/ 066 LLI 365 CLEAR BREAKPØINT (CBP)
001355/ 056 LHI 013 3 BYTES FØR BRKPT PØINTERS
001357/ 347 LEM WHAT WAS INSTR
001360/ 060 INL
001361/ 060 INL
001362/ 106 CAL 001027 SET H AND L
001365/ 036 LDI 100 IS L= 100 (NØ BRKPT SET)
001367/ 273 CPD
001370/ 053 RTZ YES-RETURN UNTØUCHED
001371/ 374 LME NØ- CLEAR BRKPT
001372/ 066 LLI 367 REPLACE INSTR
001374/ 056 LHI 013 PUT 100 IN MS ADR LØCATION

```

```

001376/ 373 LMD
001377/ 007 RET          GØ AWAY
002000/ 106 CAL 002050 COMPARE DATA          PRØM PRØGRAMMING
002003/ 110 JFZ 002014 PRØGRAM IF NEEDED      RØUTINE (PRG)
002006/ 106 CAL 000362 INCR ADR
002011/ 104 JMP 002000 LØØP
002014/ 046 LEI 013      ALLØW 10. TRYS
002016/ 106 CAL 002060 GØ PRØGRAM
002021/ 150 JTZ 002037 RØM AND RAM THE SAME, THEN DØNE
002024/ 041 DCE          NØ-TRY AGAIN
002025/ 110 JFZ 002016 IF ALLØVED
002030/ 106 CAL 001055 TØØ MANY TRYS =BAD RØM
002033/ 016 LBI 277      SØ Ø/P-CLP,?
002035/ 025 RST 020
002036/ 005 RST 000      GØ AWAY AND SULK
002037/ 106 CAL 002060 DATA ØK-MAKE SURE!!!
002042/ 016 LBI 002      Ø/P NULL TØ ITY
002044/ 025 RST 020      KEEP EVERYØNE AWAKE
002045/ 104 JMP 002006 DØ NEXT BYTE
002050/ 106 CAL 001023 SET UP H,L
002053/ 306 LAL          TELL RØM LS BYTE
002054/ 121 ØUT 010
002055/ 103 INP 001      DATA FRØM RØM
002056/ 277 CPM          IS RAM DIFFERENT?
002057/ 007 RET          GØ BACK AND THINK (ZF WILL TELL)
002060/ 307 LAM          FETCH DATA
002061/ 054 XRI 377      CØPLEMENT
002063/ 123 ØUT 011      Ø/P TØ PRØM
002064/ 006 LAI 002      ENABLE PRØGRAMMER
002066/ 127 ØUT 013
002067/ 026 LCI 000      TIME FØR ABØUT
002071/ 036 LDI 177      2 SECØNDS
002073/ 075 RST 070      WITH NESTED LØØPS
002074/ 020 INC
002075/ 110 JFZ 002071
002100/ 250 XRA          TURN ØFF PRØGRAMMER
002101/ 127 ØUT 013
002102/ 036 LDI 000      WAIT FØR THINGS TØ SETTLE
002104/ 075 RST 070
002105/ 103 INP 001      GET RØM DATA
002106/ 277 CPM          DID IT PRØGRAM?
002107/ 007 RET          GØ THINK
002110/ 066 LLI 377      ADR ØF CLP      SET CLP (LØC)
002112/ 104 JMP 003143 I/P ADR, RET HØME
002115/ 000 HLT          WAIT          BNPF DUMP (DBF)
002116/ 066 LLI 371      SCRATCH LØCATION
002120/ 076 LMI 005      Ø/P 5 BYTES PER LINE
002122/ 016 LBI 240      NØW, Ø/P A BLANK
002124/ 025 RST 020
002125/ 106 CAL 001023 GET DATA
002130/ 360 LLA          SAVE IT IN L
002131/ 016 LBI 302      Ø/P 'B'
002133/ 025 RST 020
002134/ 046 LEI 010      8 BITS PER BYTE (I THINK)
002136/ 306 LAL          RØTATE DATA IN 'L'
002137/ 002 RLC          PUT NEXT BIT IN CARRY
002140/ 360 LLA
002141/ 016 LBI 316      SET 'B' TØ 'N'
002143/ 100 JFC 002150 IF BIT IS 0, JUMP
002146/ 016 LBI 320      BIT= 1 CHANGE TØ 'P'
002150/ 025 RST 020      Ø/P WHATEVER IT IS
002151/ 041 DCE          ØNE MØRE BIT DØNE

```

```

002152/ 110 JFZ 002136 LOOP IF MORE
002155/ 016 LBI 306 DONE BYTE,Ø/P 'F'
002157/ 025 RST 020
002160/ 106 CAL 000362 INCR,CØMP CLP
002163/ 066 LLI 371 SET UP 'L' AGAIN
002165/ 317 LBM ONE MORE BYTE Ø/P
002166/ 011 DCB
002167/ 371 LMB
002170/ 110 JFZ 002122 MORE ØN THIS LINE
002173/ 106 CAL 000013 NEW LINE (CR/LF)
002176/ 104 JMP 002116 KEEP GØING
002201/ 300 LAA NØP BNPf LOAD (LBF)
002202/ 046 LEI 102 WAIT FØR A 'B'
002204/ 055 RST 050
002205/ 046 LEI 370 NØW 8 BITS EXPECTED
002207/ 106 CAL 001023 SET H,L
002212/ 076 LMI 000 CLEAR SOME RAM
002214/ 035 RST 030 FETCH CHARACTER
002215/ 301 LAB INTØ 'A'
002216/ 074 CPI 116 IS IT 'N'
002220/ 150 JIZ 002232 YES- STASH IT
002223/ 054 XRI 377 NØ -CØPLEMENT
002225/ 074 CPI 257 IS IT 'P'
002227/ 110 JFZ 002030 NØ-ERRØR
002232/ 032 RAR YES- PUT BIT IN CARRY
002233/ 307 LAM GET PREVIOUS BITS
002234/ 022 RAL RØTATE IN NEW BIT
002235/ 370 LMA STASH IT
002236/ 040 INE CØUNT YØUR BITS
002237/ 110 JFZ 002214 NØT DØNE,LØOP
002242/ 035 RST 030 YES-ØNE MORE CHECK
002243/ 301 LAB
002244/ 074 CPI 106 LAST CHARACTER MUST BE AN 'F'
002246/ 110 JFZ 002030 NØ-PANIC
002251/ 106 CAL 000362 YES-INCR CLP,CHECK IF DØNE
002254/ 104 JMP 002202 LOOP IF YØU GET HERE
002257/ 066 LLI 373 BANK TØ BANK TRANSLATE(TRN)
002261/ 106 CAL 000205 FETCH ØLD BANK NØ.
002264/ 016 LBI 337 Ø/P BACK ARROW
002266/ 025 RST 020
002267/ 061 DCL FETCH NEW BANK NØ.
002270/ 106 CAL 000205
002273/ 106 CAL 001023 GET DATA (INSTR)
002276/ 347 LEM
002277/ 106 CAL 006320 IS IT 1,2 ØR 3 BYTE INSTR
002302/ 340 LEA 'A' HAS PØINTER (0=1BYTE)
002303/ 106 CAL 000362 INCR CLP (1=2BYTE)
002306/ 304 LAE RØTATE PØINTER (3=3BYTE)
002307/ 012 RRC
002310/ 140 JTC 002302 LOOP FØR MORE
002313/ 074 CPI 140 WAS IT A 3 BYTE INSTR (JMP ØR CAL)
002315/ 110 JFZ 002273 NØ-GØ TØ NEXT BYTE
002320/ 106 CAL 001023 YES-SET UP H,L
002323/ 061 DCL TØ LAST BYTE ØF JMP ØR CAL
002324/ 307 LAM FETCH BYTE
002325/ 056 LHI 013 WAS IT ØUR MAGIC NØ.?
002327/ 066 LLI 373
002331/ 277 CPM
002332/ 110 JFZ 002273 NØ-GØ AWAY
002335/ 061 DCL YES-GET THE NEW ØNE
002336/ 347 LEM
002337/ 106 CAL 001023 SET UP H,L

```

```

002342/ 061 DCL      (LS-1) OF COURSE
002343/ 374 LME      REPLACE MS BYTE
002344/ 104 JMP 002273 NOW- WE ARE REALLY DONE
002347/ 106 CAL 001353 CLEAR OLD      SET BREAKPOINT (SBP)
002352/ 106 CAL 001200 FETCH ADR OF NEW BRKPT
002355/ 106 CAL 001023 SET UP H,L TO CLP
002360/ 326 LCL      SAVE H,L
002361/ 335 LDH
002362/ 076 LMI 065   SET RST 060 INTO LOCATION
002364/ 056 LHI 013   SAVE THE OLD INSTR
002366/ 066 LLI 365
002370/ 370 LMA      IT WAS LEFT IN 'A'
002371/ 060 INL
002372/ 372 LMC      'L' (LS ADR)
002373/ 060 INL
002374/ 373 LMD      'H' (MS ADR)
002375/ 104 JMP 003000 GO HOME TO MOMMY
003000/ 106 CAL 000013 CR/LF      MONITOR AND CONTROLLER
003003/ 046 LEI 010   SET UP LOOP
003005/ 016 LBI 255   CHARACTER IS '-'
003007/ 025 RST 020   GO PRINT '-----'
003010/ 041 DCE      COUNT
003011/ 110 JFZ 003007 LOOP
003014/ 146 CAL 000013 NEW LINE
003017/ 146 CAL 003067 FETCH INPUT
003022/ 100 JFC 003017 LOOP IF NOT 'A'-'Z'
003025/ 146 CAL 003100 NOW GET TWO MORE CHARACTERS
003030/ 146 CAL 003150 FIND IT IN THE TABLE
003033/ 150 JTZ 006000 NOT FOUND!! GO TO LDS
003036/ 066 LLI 373   EXEC ROUTINE
003040/ 056 LHI 013
003042/ 370 LMA      STORE ADDRESS (MS)
003043/ 061 DCL      FROM 5 BYTE TABLE
003044/ 371 LMB      STORE ADDRESS (LS)
003045/ 044 NDI 200   FROM 5 BYTE TABLE
003047/ 112 CFZ 003137 GO FETCH INITIAL AND FINAL ADDRESS
003052/ 112 CFZ 000013 IF MS=1XXXXXXX, START WITH CR/LF
003055/ 066 LLI 371   JMP IN 371
003057/ 076 LMI 104   YES IT IS AN INDIRECT JMP
003061/ 106 CAL 013371 SO GO
003064/ 144 JMP 003014 AND CONTINUE WHEN DONE
003067/ 035 RST 030   GET CHAR      CHAR TEST
003070/ 006 LAI 100   TEST FOR LI 'A'
003072/ 271 CPB
003073/ 301 LAB
003074/ 003 RFC      PASS IF GT OR EQ 'A'
003075/ 074 CPI 133   TEST IF GT 'Z'
003077/ 047 RET      GO, CARRY TELLS ALL
003100/ 046 LEI 002   SYM INPUT
003102/ 066 LLI 350   CHAR IN 013350
003104/ 056 LHI 013
003106/ 370 LMA      STORE
003107/ 146 CAL 003067 TEST NEXT CHAR
003112/ 100 JFC 003131 NOT 'A'-'Z' !! ERROR!
003115/ 060 INL      SET UP NEXT ONE
003116/ 041 DCE      COUNT
003117/ 110 JFZ 003106 NOT DONE, LOOP
003122/ 370 LMA      DONE- STORE LAST CHAR
003123/ 340 LEA      NOW SET UP REG, 3 GOES IN 'E'
003124/ 061 DCL
003125/ 337 LDM      2 IN 'D'
003126/ 061 DCL

```

```

003127/ 327 LCM          AND 1 IN 'C'
003130/ 007 RET          ALL DONE
003131/ 016 LBI 277          ERRORS COME HERE
003133/ 025 RST 020      PRINT '?'
003134/ 144 JMP 003014 G0 GET ANOTHER INPUT
003137/ 106 CAL 001200 CR/LF AND '*' INITIAL AND FINAL ADR
003142/ 061 DCL          GOT FIRST ADR
003143/ 016 LBI 240      PRINT BLANK
003145/ 104 JMP 001207 GET FINAL ADR,G0 BACK HOME
003150/ 016 LBI 022          SEARCH TABLE
003152/ 066 LLI 021      5 BYTE TABLE
003154/ 056 LHI 004
003156/ 307 LAM          NOW GET 1ST CHAR
003157/ 060 INL          READY FOR NEXT CHAR
003160/ 272 CPC          COMPARE TABLE AND I/P
003161/ 110 JFZ 003204 JMP IF NOT EQUAL
003164/ 307 LAM          GET 2ND
003165/ 060 INL          READY FOR 3RD
003166/ 273 CPD          COMPARE
003167/ 110 JFZ 003205 JMP IF NOT THE SAME
003172/ 307 LAM          NOW FOR THE 3RD
003173/ 060 INL          AND PREPARE FOR DATA
003174/ 274 CPE          COMPARE AS BEFORE
003175/ 110 JFZ 003206 AND JUMP IF NOT NICE
003200/ 317 LBM          GET 'G0 T0' ADDRESS
003201/ 060 INL
003202/ 307 LAM          2 BYTES OF IT
003203/ 007 RET          AND RETURN
003204/ 060 INL          LOOK AT NEXT SYMBOL
003205/ 060 INL          IN THE TABLE
003206/ 060 INL
003207/ 060 INL
003210/ 011 DCB          COUNT OUR TRYS
003211/ 053 RTZ          END OF TABLE
003212/ 104 JMP 003156 MORE TO CHECK
003215/ 056 LHI 004          3BYTE TABLE SEARCH
003217/ 021 DCC          'C' IS COUNTER
003220/ 053 RTZ          RETURN WHEN DONE TABLE
003221/ 307 LAM          NOW LOOK AT THE FIRST ENTRY
003222/ 060 INL          COMPARE WITH DATA
003223/ 273 CPD          JMP IF NOT LIKED
003224/ 110 JFZ 003237 2ND ENTRY AS ABOVE
003227/ 307 LAM
003230/ 274 CPE
003231/ 110 JFZ 003237 AND JUMP, MAYBE
003234/ 060 INL          FETCH DATA FROM TABLE
003235/ 307 LAM
003236/ 007 RET          RETURN TO LDS ROUTINE
003237/ 060 INL          NEXT ENTRY
003240/ 060 INL          E00P AND TRY AGAIN
003241/ 104 JMP 003217 LOOP AND TRY AGAIN
003244/ 345 LEH          BRKPT EXECUTE
003245/ 336 LDL          SAVE L,H LOSING D,E
003246/ 066 LLI 364      SAVE REGISTERS A-E
003250/ 056 LHI 013      IN RAM (LOC 013364 TO 013360)
003252/ 374 LME
003253/ 061 DCL
003254/ 373 LMD
003255/ 061 DCL
003256/ 372 LMC
003257/ 061 DCL
003260/ 371 LMB

```

```

003261/ 061 DCL
003262/ 370 LMA
003263/ 006 LAI 030 NOW DISPLAY CARRY FLAG
003265/ 022 RAL ROTATE IN CARRY AND CONVERT TO
003266/ 340 LEA ASCII
003267/ 016 LBI 240 O/P BLANK
003271/ 025 RST 020
003272/ 314 LBE O/P CARRY FLAG
003273/ 025 RST 020
003274/ 046 LEI 005 SET UP COUNT TO PRINT REGISTERS
003276/ 066 LLI 360 START OF SAVED REGISTERS
003300/ 106 CAL 000253 PRINT BYTE AS OCTAL
003303/ 060 INL NEXT REGISTER
003304/ 041 DCE COUNT
003305/ 110 JFZ 003300 LOOP TILL DONE
003310/ 061 DCL GO BACK ONE REG
003311/ 106 CAL 001027 AND GET DATA AT H,L LOCATION
003314/ 106 CAL 000253 AND PRINT IT
003317/ 005 RST 000 NOW WERE DONE, GO HOME
003320/ 066 LLI 373 XQT ROUTINE
003322/ 106 CAL 003143 LOAD ADDRESS
003325/ 104 JMP 003052 EXEC WILL SEND US THERE

```

R0M NUMBER 3 CONTINUED WITH DPS ROUTINES

R0M 4 CONTAINS THE SYMBOL TABLES,
AS FOLLOWS:

1. 5 BYTE TABLE

THE 5 BYTE TABLE OCCUPIES POSITIONS 004021 TO 004157 INCLUSIVE AND CONTAINS ALL MONITOR COMMANDS PLUS THE MACHINE COMMANDS HLT, INP, OUT, RST, AND THE SPECIAL SYMBOL ???, INDICATING A NO FIND CONDITION ON OUTPUT. THE INPUT ROUTINE DOES NOT USE THIS SYMBOL. THE FORMAT IS THUS:

ASCII	X	Q	T	(DATA FIELD)
OCTAL	130	121	124	320 003
ADDRESS	021	022	023	024 025

WHEN A FIND IS MADE DURING A SEARCH, THE DATA FIELD IS MOVED TO REGISTERS A AND B, AND AN INDIRECT JUMP MADE TO THAT ADDRESS, IF THE MS HALF OF THE ADDRESS IS A 2XX, THE EXEC WILL LOOK FOR TWO ADDRESSES BEFORE GOING TO THE ROUTINE.

DURING A SYMBOLIC DUMP, THE LAST 5 SYMBOLS ARE USED FOR THE APPROPRIATE MACHINE COMMANDS, AND ARE STORED AS OUTPUT.

ADR	SYMB	LS	MS	(ADDRESS OF ROUTINE)
004021/	XQT	320	003	
004026/	EDT	236	001	
004033/	LD0	120	201	
004040/	LBF	201	202	
004045/	DP0	000	201	
004052/	DBF	115	202	
004057/	DPS	000	205	
004064/	CPY	146	201	
004071/	TRN	257	202	
004076/	SBP	347	002	
004103/	CBP	353	001	
004110/	PRG	000	202	
004115/	L0C	110	002	

004122/ DLP 055 001
 004127/ HLT 046 006
 004134/ RST 270 006
 004141/ INP 270 006
 004146/ ØUT 270 006
 004153/ ???

2. 3 BYTE TABLE

THIS TABLE CONTAINS TWO BYTES OF ASCII CODE AND ONE DATA BYTE, WHICH IS A MASKED PORTION OF THE INSTRUCTION. THE FORMAT IS:

ASCII	N	D	(DATA)
ØCTAL	116	104	244
LØCATION	252	253	254

THE TABLE OCCUPIES LØCATIONS 004156 TO 004273, AND IS USED IN TWO WAYS. THE LDS RØUTINE COMPARES THE TWO ASCII CHARACTERS TO THE INPUT CHARACTERS, AND RETURNS THE DATA IN THE A REGISTER IF A FIND IS MADE. FOR THE DPS RØUTINE, THE PARTIAL WORD (DATA) IS TESTED, AND THE H AND L REGISTERS ARE USED TO RETRIEVE THE ASCII AS NEEDED.

3 BYTE TABLE:

LØCATION	ASCII	DATA	
004156	LC	002	
004161	RC	012	
004164	AL	022	
004167	AR	032	
004172	MP	104	(JMP)
004176	AL	106	(CAL)
004203	ET	007	(RET)
004206	TC	040	
004211	FC	000	
004214	TZ	050	
004217	FZ	010	
004222	TS	060	
004225	FS	020	
004230	TP	070	
004233	FP	030	
004236	AD	204	
004241	AC	214	
004244	SU	224	
004247	SB	234	
004252	ND	244	
004255	XR	254	
004260	ØR	264	
004263	CP	274	
004266	IN	000	
004271	DC	001	

3. 4 BYTE TABLE

THE 4 BYTE TABLE OCCUPIES POSITIONS 004274 TO 004377, AND IS USED BY THE DPS RØUTINE.

THE FORMAT IS:

	MASK	DATA	ADDRESS	DATA FIELD
	361	101	161	144
LØC	310	311	312	313

THE MASK CHARACTER IS USED TO MASK (AND) DON'T CARE BITS IN THE INPUT BYTE, THE REMAINING BITS ARE COMPARED TO THE DATA IN THE

NEXT FIELD TO DECODE AN INSTRUCTION. IF A FIND IS MADE THE ADDRESS IS USED FOR AN INDIRECT JUMP (TO 005AAA). THE LAST ENTRY IS AN UNCONDITIONAL FIND WHICH OUTPUTS THE ERROR SYMBOL ???.

THE DATA FIELD COLUMN IS USED FOR VARIOUS PURPOSES BY THE CALLED ROUTINES.

4 BYTE TABLE

LOCATION	MASK	DATA	ADDRESS	DATA FIELD
004274	377	377	155	132
004300	376	000	155	132
004304	376	070	155	156
004310	361	101	161	144
004314	347	002	251	037
004320	307	006	262	352
004324	307	005	161	137
004330	307	004	125	111
004334	307	001	142	273
004340	307	000	142	270
004344	303	003	215	202
004350	303	102	215	176
004354	301	101	161	151
004360	303	100	215	172
004364	300	300	272	000
004370	300	200	120	000
004374	000	000	155	156

SYMBOLIC ROUTINES

NOTE: THESE ROUTINES COVER PART OF ROM 3,4 AND ALL OF ROMS 5,6

003330/	106	CAL	005352	GET 3 BYTES	DPS OUTPUT
003333/	106	CAL	005104	LOAD THEM INTO	REGISTERS
003336/	106	CAL	005313	OUTPUT THEM	
003341/	347	LEM		LOAD E WITH	DATA
003342/	106	CAL	006320	DECODE LENGTH	
003345/	012	RRC		1 BYTE INSTR?	
003346/	100	JFC	005363	YES-GO TO	LINE CHECK
003351/	340	LEA		SAVE LENGTH	BITS
003352/	106	CAL	000362	INCR ADR	
003355/	106	CAL	001023	GET DATA	
003360/	041	DCE		3 BYTES	MAYBE?
003361/	160	JTS	003372	SIGN FLAG =1	IF S0
003364/	106	CAL	000253	O/P IMMEDIATE	DATA
003367/	104	JMP	005363	AND GO TO	LINE CHECK
003372/	327	LCM		YES ITS 3	BYTE! GET LS ADR
003373/	106	CAL	000362	INC CLP	
003376/	312	LBC		MOVE ADR TO	B
003377/	300	LAA		NOP (UNUSED	BYTE)
004000/	106	CAL	001023	GET DATA (MS	ADR BYTE)
004003/	327	LCM		SAVE IN C	
004004/	106	CAL	005104	LOAD 3 BYTES	
004007/	060	INL		SET UP DATA	POINTERS
004010/	060	INL			
004011/	106	CAL	001061	OUTPUT THIS	ADDRESS
004014/	104	JMP	005363	AND GO ON TO	LINE CHECK
005000/	016	LBI	012		SYMBOLIC DUMP (DPS)
005002/	025	RST	020	PRINT 3 LF'S	
005003/	025	RST	020		
005004/	025	RST	020		
005005/	046	LEI	076	SET UP LINES/PAGE	

```

005007/ 066 LLI 353 AND STØRE NUMBER AT 013353
005011/ 056 LHI 013
005013/ 374 LME
005014/ 106 CAL 001073 GET CLP AND PRINT IT
005017/ 106 CAL 001047 GET DATA AND PRINT IT
005022/ 347 LEM SAVE DATA IN 'E'
005023/ 337 LDM AND IN 'D'
005024/ 106 CAL 005063 ASSUME BITS 3-5 ARE A DESTINATION REG.
005027/ 061 DCL STØRE IT IN 013351,013352
005030/ 370 LMA
005031/ 066 LLI 274 SET UP START ØF 4 BYTE TABLE
005033/ 056 LHI 004
005035/ 303 LAD GET MASK FRØM TABLE
005036/ 247 NDM AND MASK DØNT CARE BITS
005037/ 060 INL NØW CHECK THE REST
005040/ 277 CPM WITH THE TABLE
005041/ 110 JFZ 005055 JUMP IF NØ FIND
005044/ 006 LAI 005 LØAD MS BYTE ØF ADR
005046/ 060 INL
005047/ 317 LBM LØAD LS BYTE ØF ADR
005050/ 060 INL
005051/ 327 LCM LØAD C WITH DATA FRØM TABLE
005052/ 104 JMP 001336 AND DØ AN INDIRECT JUMP TØ RØUTINE
005055/ 060 INL (NØ FIND) INCR L TØ
005056/ 060 INL NEXT TABLE ENTRY
005057/ 060 INL
005060/ 104 JMP 005035 GØ LØØP
005063/ 303 LAD GET DATA REGISTER DECØDE
005064/ 012 RRC
005065/ 012 RRC LØØK AT BITS 3-5
005066/ 012 RRC
005067/ 044 NDI 007 MASK THE REST
005071/ 004 ADI 370 AND ADD START ØF TABLE
005073/ 360 LLA TABLE ADR TØ 'L'DR
005074/ 056 LHI 006 MS ADR ØF TABLE
005076/ 307 LAM GET REGISTER
005077/ 066 LLI 352
005101/ 104 JMP 005114 DØNE
005104/ 066 LLI 352 SET DP 3 BYTE LØAD
005106/ 056 LHI 013
005110/ 372 LMC SAVE C
005111/ 061 DCL
005112/ 371 LMB SAVE B
005113/ 061 DCL L=350 NØW
005114/ 056 LHI 013 ENTRY FØR 1 BYTE LØAD
005116/ 370 LMA AND SAVE A
005117/ 007 RET GØ AWAY SØMWHERE
005120/ 303 LAD ACC GRØUP RØUTINE
005121/ 106 CAL 005067 DECØDE SØURCE REG
005124/ 320 LCA AND PUT IN C
005125/ 303 LAD ENTRY FØR IMMEDIATE
005126/ 044 NDI 070 MASK ØUT SØURCE PART
005130/ 004 ADI 204 (SPECIALLY FØR 'I' INSTR)
005132/ 066 LLI 240 START ØF ACC IN 3 BYTE TABLE
005134/ 106 CAL 005336 GØ FIND DATA IN TABLE
005137/ 104 JMP 003333 GØ PRINT IT
005142/ 106 CAL 005063 INX,DCX RØUTINE
005145/ 362 LLC SET UP ADR FØR 3 BYTE TABLE
005146/ 320 LCA SAVE 'A' FØR NØW
005147/ 303 LAD GET BINARY DATA
005150/ 044 NDI 001 MASK ALL BUT LS BIT
005152/ 104 JMP 005134 SEACH TABLE, GØ HØME

```

```

005155/ 362 LLC          GET ADR FØR 5 BYTE TABLE
005156/ 104 JMP 003330 GØ TØ ØUTPUT
005161/ 362 LLC          TABLE ADR      INP/ØUT/RST
005162/ 343 LED
005163/ 106 CAL 005305 Ø/P SYMBØL
005166/ 307 LAM          FEICH DATA
005167/ 044 NDI 300     CHECK BITS 6-7
005171/ 307 LAM          AND RESTØRE DATA
005172/ 150 JTZ 005210 JMP IF 00XXXXXX (RST)
005175/ 044 NDI 076     MASK TØ 00XXXXXØ
005177/ 012 RRC          SET UP I/Ø PØRT NØ.
005200/ 066 LLI 352     PUT THE NUMBER AWAY
005202/ 056 LHI 013     FØR NØW
005204/ 370 LMA
005205/ 104 JMP 003364 GØ TØ ØUTPUT
005210/ 044 NDI 070     MASK DATA TØ 00XXXXØØ (RST NØ.)
005212/ 104 JMP 005200 GØ ØUTPUT IT
005215/ 362 LLC          JMP/CAL/RET GRØUP
005216/ 056 LHI 004     SET UP FØR TABLE
005220/ 347 LEM          FEICH J,C,ØR R FRØM TABLE
005221/ 303 LAD          RESTØRE BINARY
005222/ 044 NDI 307     MASK AND CHECK IF UNØONDITIONAL
005224/ 060 INL          TRANSFER
005225/ 060 INL
005226/ 060 INL
005227/ 277 CPM
005230/ 150 JTZ 003330 YES-GØ TØ ØUTPUT
005233/ 066 LLI 210     NØ-LØØK UP ØONDITION
005235/ 303 LAD          IN 3 BYTE TABLE
005236/ 044 NDI 070     MASK ALL BUT ØONDITION
005240/ 106 CAL 005336 GØ SEARCH
005243/ 321 LCB          CHAR 3
005244/ 310 LBA          CHAR 2
005245/ 304 LAE          CHAR 1
005246/ 104 JMP 003333 GØ ØUTPUT
005251/ 303 LAD          GET DATA      RØTATE GRØUP
005252/ 242 NDC          MASK AS PER TABLE
005253/ 066 LLI 160     RØT IN 3 BYTE TABLE
005255/ 046 LEI 122     LØAD E WITH 'R' AND PRETEND
005257/ 104 JMP 005240 ITS A TRANSFER
005262/ 362 LLC          SET UP ADR      LØAD IMMEDIATE
005263/ 056 LHI 013
005265/ 076 LMI 111     LØAD 3RD CHAR AS 'I' (SØURCE REG.)
005267/ 104 JMP 005276 AND TREAT AS ØRDINARY LØAD
005272/ 303 LAD          GET DATA      LØAD (REG TØ REG)
005273/ 106 CAL 005067 GET SØURCE REG
005276/ 061 DCL
005277/ 061 DCL
005300/ 076 LMI 114     LØAD 'L' AS 1ST CHAR
005302/ 104 JMP 003336 GØ TØ ØUTPUT
005305/ 106 CAL 005352 3 BYTE TRANSFER
005310/ 106 CAL 005104 3 BYTE LØAD
005313/ 016 LBI 240     PRINT 3 BYTES (RØUTINE)
005315/ 025 RST 020     Ø/P TWØ BLANKS
005316/ 025 RST 020
005317/ 066 LLI 350     ADR ØF FIRST CHAR
005321/ 056 LHI 013
005323/ 317 LBM          FEICH IT
005324/ 025 RST 020     PRINT
005325/ 060 INL          NEXT CHAR
005326/ 317 LBM
005327/ 025 RST 020     PRINT

```

```

005330/ 060 INL          ØNCE MØRE NØW
005331/ 317 LBM
005332/ 025 RST 020
005333/ 104 JMP 001023 GØ GET MØRE DATA
005336/ 056 LHI 004          3 BYTE TABLE SEARCH
005340/ 277 CPM          CØMPARE
005341/ 150 JIZ 005356 EXIT IF FØUND
005344/ 060 INL          NEXT ENTRY
005345/ 060 INL
005346/ 060 INL
005347/ 104 JMP 005340 LØØP
005352/ 056 LHI 004          3 BYTE TRANFER
005354/ 061 DCL
005357/ 317 LBM          2ND CHAR
005360/ 061 DCL
005361/ 307 LAM          1ST CHAR
005362/ 007 RET          GØ!
005363/ 106 CAL 000362 INCR CLP          LINE CHECK
005366/ 066 LLI 353
005370/ 347 LEM          FETCH LINE CØUNT
005371/ 041 DCE          UPDATE IT
005372/ 150 JIZ 005000 END ØF PAGE Ø/P 3 LF'S
005375/ 104 JMP 005013 ØK GØ ØN TØ NEXT LINE
006000/ 302 LAC          GET 1ST CHAR SYMBOLIC LØAD (LDS)
006001/ 074 CPI 114      IS IT AN 'L'
006003/ 110 JFZ 006122 NØ-TEST FØR 'R'
006006/ 016 LBI 306     PARTIAL WØRD LØAD INSTRUCTION
006010/ 303 LAD          LØØK AT 2ND CHAR
006011/ 146 CAL 006345 ENCØDE AS DETINATION REG
006014/ 002 RLC
006015/ 002 RLC
006016/ 002 RLC
006017/ 201 ADB          STASH WITH PARTIAL WØRD
006020/ 310 LBA          IN 'B'
006021/ 006 LAI 111     IS 3RD CHAR AN 'I' ?
006023/ 274 CPE
006024/ 150 JIZ 006043 YES- GØ TØ IMMEDIATE RØUTINE
006027/ 304 LAE          NØ-ENCØDE SØURCE REGISTER AS ABØVE
006030/ 146 CAL 006345
006033/ 320 LCA
006034/ 301 LAB          GET DUMMY WØRD
006035/ 044 NDI 370     DISCARD BITS 0-3 (A=3X6)
006037/ 202 ADC          AND PUT IN THE REAL ØNE
006040/ 144 JMP 006046 NØW GØ CLEAN UP
006043/ 006 LAI 077     IMMEDIATE LØAD
006045/ 241 NDB          MASK TØ 00XXXXXX
006046/ 340 LEA          A HAS INSTR FINISH RØUTINE
006047/ 146 CAL 001023 GET CLP
006052/ 374 LME          PUT INSTR THERE
006053/ 300 LAA          NØP (NØT USED)
006054/ 146 CAL 006320 DECØDE LENGTH
006057/ 146 CAL 000311 INCR CLP
006062/ 012 RRC          CHECK LENGTH BITS
006063/ 100 JFC 003014 LEAVE US WHEN NØ MØRE BITS IN CARRY
006066/ 340 LEA          NØT DØNE-SAVE THE BITS
006067/ 016 LBI 240     PRINT A BLANK
006071/ 025 RST 020
006072/ 041 DCE          IS IT A 3BYTE INSTR?
006073/ 160 JIS 006105 SIGN FLAG TELLS ALL (SF=1 FØR 3 BYTE INSTR)
006076/ 106 CAL 001111 GET DATA AND INPUT

```

```

006101/ 250 XRA          CLEAR A
006102/ 104 JMP 006057 AND LOOP
006105/ 106 CAL 000311 INCR CLP          3BYTE (MUST WANT AN ADR)
006110/ 106 CAL 001023 GET MORE DATA
006113/ 106 CAL 001212 AND STORE TWO BYTES (CLP,CLP-1)
006116/ 104 JMP 006101 GO BACK TO LOOP
006121/ 377 HLT          UNUSED HALT(1)
006122/ 074 CPI 122          TEST FOR 1ST CHAR = 'R'
006124/ 110 JFZ 006146 NO- KEEP LOOKING
006127/ 026 LCI 005          IS IT A ROTATE?
006131/ 066 LLI 156
006133/ 106 CAL 003215 SEARCH 3 BYTE TABLE
006136/ 066 LLI 202
006140/ 150 JTZ 006234 IF NO FIND, TEST FOR RETURN
006143/ 144 JMP 006046 GO FINISH UP
006146/ 314 LBE          ACC GROUP
006147/ 343 LED          PUT CHARACTERS AWAY
006150/ 332 LDC
006151/ 026 LCI 013        SET UP TABLE SEARCH
006153/ 066 LLI 236        (ACC GROUP, IN(R), DC(R))
006155/ 106 CAL 003215 SEARCH TABLE
006160/ 150 JTZ 006214 NO FIND, KEEP LOOKING
006163/ 320 LCA          GET 1ST CHAR
006164/ 044 NDI 200        CHECK FOR IMMEDIATE INSTR
006166/ 302 LAC          RESTORE CHAR
006167/ 341 LEB
006170/ 110 JFZ 006020 GO AWAY IF IMMEDIATE INSTR
006173/ 301 LAB          TEST THE 3RD CHAR
006174/ 106 CAL 006345 ENCODE AS A REGISTER
006177/ 002 RLC
006200/ 002 RLC
006201/ 002 RLC
006202/ 202 ADC          ADD TO PARTIAL WORD
006203/ 104 JMP 006046 FINISH UP
006206/ 106 CAL 001023 6 BYTES NOT USED (1)
006211/ 104 JMP 003143 (111)
006214/ 323 LCD          TRANSFER GROUP (JMP,CAL,RET)
006215/ 334 LDE          MUSICAL REGISTERS
006216/ 341 LEB
006217/ 066 LLI 172        START OF TABLE (JMP)
006221/ 302 LAC
006222/ 277 CPM          TRY 1ST CHAR
006223/ 150 JTZ 006234 JUMP IF FIND
006226/ 066 LLI 176        TRY 'CAL'
006230/ 277 CPM
006231/ 110 JFZ 003131 NO (1) MUST BE AN ERROR
006234/ 060 INL          IS IT UNCONDITIONAL?
006235/ 026 LCI 002
006237/ 146 CAL 003215 THEN GO TO FINISH
006242/ 110 JFZ 006046 GET PART WORD
006245/ 061 DCL          AS MUCH AS WE CAN
006246/ 307 LAM          BLANK OUT SOME
006247/ 044 NDI 303
006251/ 310 LBA          AND LOOK UP THE CONDITION
006252/ 066 LLI 206
006254/ 026 LCI 011
006256/ 146 CAL 003215
006261/ 150 JTZ 003131 NO-FIND ERROR(!)
006264/ 201 ADB          ADD IN CONDITION BITS
006265/ 144 JMP 006046 FINISH IT
006270/ 343 LED          INP/OUT/RST
006271/ 016 LBI 240        ENTER AS MONITOR ROUTINE

```

```

006273/ 025 RST 020 PRINT A BLANK
006274/ 146 CAL 000205 INPUT THE OCTAL ARGUMENT
006277/ 300 LAA NOP (AGAIN -REALLY TOM !!)
006300/ 304 LAE GET 2ND CHAR
006301/ 074 CPI 123 IS IT AN 'S'
006303/ 307 LAM GET THE OCTAL ARGUMENT
006304/ 046 LEI 005 ASSUME IT'S RST
006306/ 150 JTZ 006314 AND SKIP AHEAD IF IT IS
006311/ 002 RLC MUST BE INP/OUT-ROTATE ARGUMENT
006320/ 250 XRA INSTRUCTION LENGTH TEST
006321/ 310 LBA CLEAR REGISTERS
006322/ 304 LAE GET DATA
006323/ 044 NDI 305
006325/ 074 CPI 004 IS IT IMMEDIATE?
006327/ 150 JTZ 006342 YES-BEGONE
006332/ 044 NDI 301
006334/ 074 CPI 100 IS IT A TRANSFER?
006336/ 301 LAB CLEAR A
006337/ 013 RFZ PASS IF 3 BYTE (JMP,CAL GROUP)
006340/ 010 INB NOW SET UP B
006341/ 010 INB
006342/ 010 INB COME HERE IF 2 BYTE
006343/ 301 LAB SO NOW A IS 001 OR 003
006344/ 007 RET GO HOME AND TELL ABOUT IT
006345/ 066 LLI 370 REGISTER DECODE
006347/ 056 LHI 006 LOOK AT TABLE
006351/ 277 CPM TEST
006352/ 110 JFZ 006361 NO FIND -LOOP
006355/ 306 LAL A FIND! GET THE ADDRESS
006356/ 044 NDI 007 MASK 00000XXX
006360/ 007 RET AND RETURN WITH A NUMBER
006361/ 060 INL NEXT VALUE
006362/ 110 JFZ 006351 NOT ZERO GO LOOP
006365/ 104 JMP 003131 NOT IN TABLE- ITS AN ERROR FOLKS

```

REGISTER LOOK UP TABLE

LOCATION	REGISTER	BINARY	ASCII
006370	A	0	101
006371	B	1	102
006372	C	2	103
006373	D	3	104
006374	E	4	105
006375	H	5	110
006376	L	6	114
006377	MEMORY(M)	7	115

1110 LINES TALLIED AT EOF

*
TIMEOUT