APPLICATIONS MANUAL

BULLETIN 80007

1974 EDITION

MF8008

CENTRAL PROCESSING UNIT
MOD 8 - MODULAR µPROCESSOR
MONITOR 8 - SOFTWARE
OPERATING SYSTEM

MICRO SYSTEMS
a Northern Electric company

SYSTEM PROTOTYPING MADE EASY WITH THE MF8008

MICRO SYSTEMS
a Northern Electric company

MOD 8

Modular microprocessor – complete hardware kits for all your system requirements

CPU

MF8008 – 8 bit central processor

RAM

MF1101–256x1 bit static operation
MF2102–2048x1 bit static operation
—and the "by 4" versions:
MF2111–18 lead package, common I/O with bus control
MF2112–16 lead package, common I/O
MF2113–22 lead package, separate I/O

ROM

MF1702A–256x8 bit field eraseable and reprogrammable
MF1602–256x8 bit electrically programmable
MF1302–256x8 bit mask programmable
MF8316–2048x8 bit mask programmable
MF8317–4096x4 bit mask programmable

MONITOR 8

A complete software operating system for the MOD 8 modular μprocessor

For additional information please contact any of the Microsystems offices listed below:

IN CANADA

Montreal, Quebec: 105 Laurentian Blvd., Tel. (514) 747-5880, TWX 610-421-3634
Ottawa, Ontario: 75 Moodie Drive, Tel. (613) 828-9191, TWX 610-562-1910, Telex 053-4580

IN THE UNITED STATES

Palo Alto, California: 450 San Antonio Road, Tel. (415) 493-0848, TWX 910-373-1281
Schaumburg, Illinois: 1635 West Wise Road, Tel. (312) 894-7660, TWX 910-291-3548
Huntingdon Valley, Pennsylvania: 1 Fairway Plaza, Tel. (215) 947-5641/2, TWX 510-665-5303
Santa Ana, California: 1712 Newport Circle, Tel. (714) 979-6522, TWX 910-595-1780

IN EUROPE

United Kingdom and Scandinavia:
Hayes, Middlesex, England: Ventura House, 72-74 Station Road Tel. 01-573-6582/4, Telex 934421
Germany:
7000 Stuttgart 1(W), West Germany: Gustav-Sieglestrasse, 96, Tel. (0711) 653116/26, Telex 722290

MICRO SYSTEMS
a Northern Electric company

MF8008 APPLICATIONS MANUAL

Description	Page	Description	Page	
1.0 **Introduction**	A–2	4.0 **Basic Instruction Set**	A–6	COMPONENT DESCRIPTION
2.0 **Processor Timing**	A–2	4.1 Data Instruction Formats	A–6	
2.1 State Control Coding	A–2	4.2 Summary of Processor Instructions	A–6	
2.2 Timing	A–2	4.3 Complete Functional Definition	A–9	
2.3 Cycle Control Coding	A–3	4.4 Internal Processor Operation	A–14	
3.0 **Basic Functional Blocks**	A–4	5.0 **Processor Control Signals**	A–17	
3.1 Instruction Register and Control	A–4	5.1 Interrupt Signal(INT)	A–17	
3.2 Memory	A–4	5.2 Start-up of the MF8008	A–18	
3.3 Arithmetic Unit(ALU)	A–5	6.0 **Electrical Specification**	A–20	
3.4 I/O Buffer	A–5			

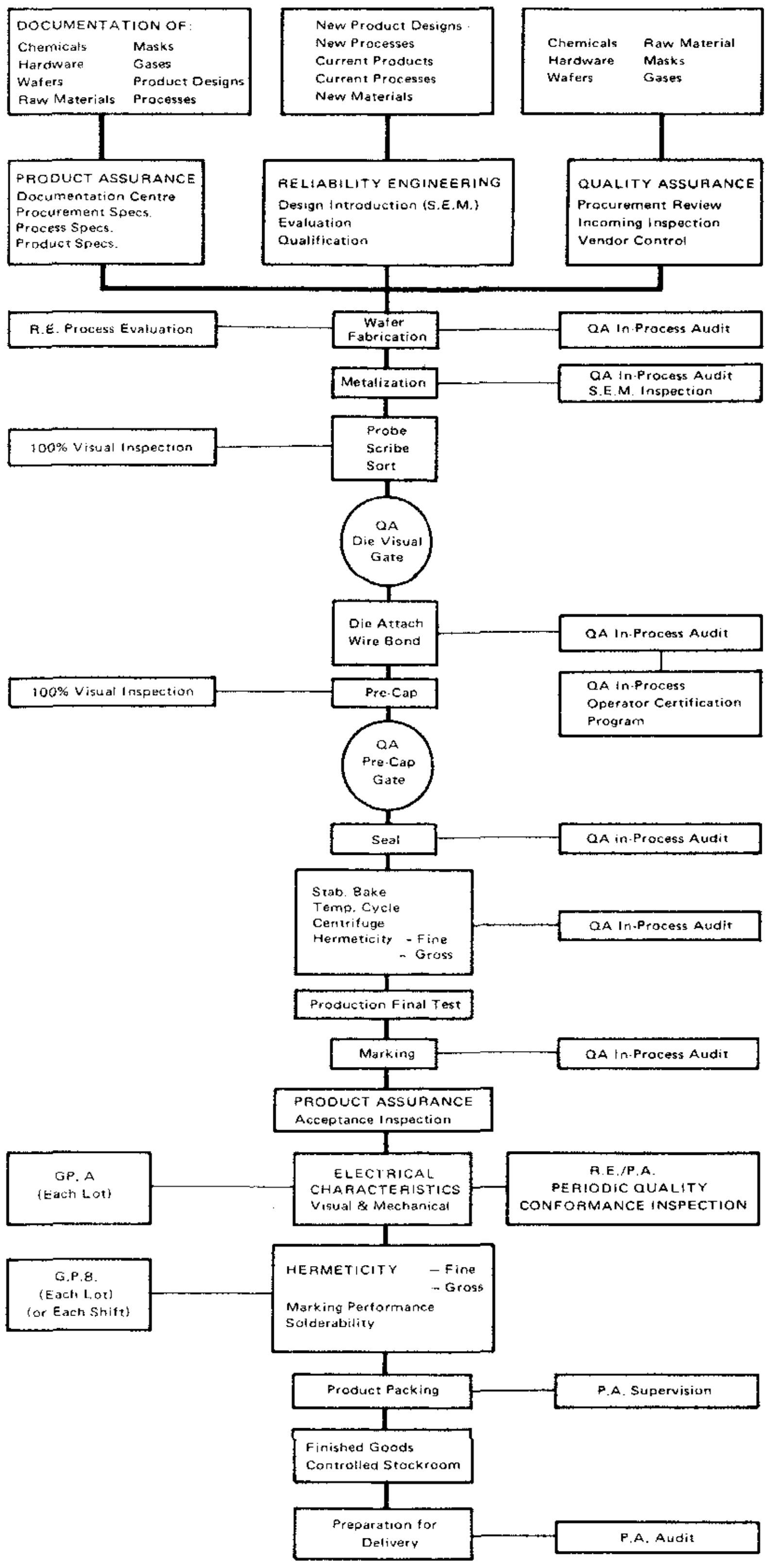
1.0 **MOD8, MF8008 Based Modular Microcomputer**	B–2	2.1 MOD8 CPU Board	B–3	MODULAR MICROCOMPUTER
1.1 MOD8 CPU Board	B–2	2.2 Restart & TTY I/O Board	B–4	
1.2 MOD8 Restart, TTY I/O Board	B–2	2.3 Control/Buffer Board	B–4	
1.3 MOD8 Control/Buffer Board	B–2	2.4 ROM Board	B–4	
1.4 MOD8 ROM Board	B–2	2.5 RAM Board	B–5	
1.5 MOD8 RAM Board	B–2	2.6 Input Board	B–5	
1.6 MOD8 Input Board	B–2	2.7 Output Board	B–5	
1.7 MOD8 Output Board	B–2	2.8 MOD8 Schematic & PCB Layout Diagrams	B–5	
2.0 **MOD8 Detailed Board Description**	B–3			

1.0 **Monitor 8 User's Guide**	C–2	1.11 LBF(Load BNPF Format)	C–3	SOFTWARE USER'S GUIDE
1.1 System Start-up(MOD8)	C–2	1.12 DBF(Dump BNPF Format)	C–3	
1.2 System Start-up(SIM8-01)	C–2	1.13 EDT(Enter Edit Mode)	C–3	
1.3 Addressing	C–2	1.14 XQT(Initiate Program Execution)	C–3	
1.4 Monitor 8 Command Summary	C–2	1.15 CYP(Copy Routine)	C–4	
1.5 LOC(Set Current Location Pointer)	C–2	1.16 TRN(Translate Routine)	C–4	
1.6 DLP(Display Current Location Pointer)	C–2	1.17 SBP(Set Break-point)	C–4	
1.7 Symbolic Program Input	C–2	1.18 CBP(Clear Break-point)	C–4	
1.8 DPS(Dump Symbolic)	C–3	1.19 PRG(Program pROM)	C–4	
1.9 LDL(Load Octal)	C–3	1.20 Control A	C–4	
1.10 DPO(Dump Octal)	C–3	1.21 Rubout	C–4	

1.0 **Monitor 8 Software Listings**	D–2	SOFTWARE LISTING

MICROSYSTEMS INTERNATIONAL LIMITED, BOX 3529 STATION C, OTTAWA, CANADA, K1Y 4J1

Microsystems' Standard Product Processing and 100% Screening



Microsystems' Q. & R. system is accepted and approved by the Canadian Department of National Defense to DND–1015. It is also a "Certified Manufacturer" to MIL–M–38510A and maintains a "Certified" (Lab certification No. F11155) qualification laboratory to MIL–STD–883/38510A.

Microsystems' Q. & R. policy is that there can be no compromise regarding the quality and reliability of the products of the company, and to ensure this, the Q. & R. department has been established as a major branch of the company, headed by the Director of Quality & Reliability reporting directly to the President and whose authority within the organizational structure is superceded only by the President's. The Director of Quality and Reliability has the authority and responsibility to implement and maintain the total Q. & R. system, thus ensuring that all products delivered meet or exceed customer requirements.

Three major departments exist within the Quality & Reliability organization:

Quality Assurance
Product Assurance
Reliability Engineering

Each department is responsible to the Director of Quality & Reliability, and although maintained as separate operational organizations, they function as a single entity to provide a total Quality and Reliability system.

Microsystems has developed, within the Q. & R. department, a number of unique state-of-the-art methods and techniques for Quality Control, many of which surpass the capabilities of other microcircuit manufacturers. Three-on-line terminals linked to the company's main computer are utilized daily by Quality and Reliability to provide rapid statistical data feed-back manufacturing operations. Two Scanning Electron Microscopes are operated and maintained within the department for product development, failure analysis and in-process control. Many of the sophisticated methods utilized were developed within the organization.

At Microsystems International the Quality & Reliability responsibility encompasses the total scope of product manufacturing, from raw materials through to delivery.

MICRO SYSTEMS
a Northern Electric company

MF8008 APPLICATIONS MANUAL

COMPONENT DESCRIPTION

1.0 **Introduction** A–2
2.0 **Processor Timing** A–2
2.1 State Control Coding A–2
2.2 Timing A–2
2.3 Cycle Control Coding A–3
3.0 **Basic Functional Blocks** A–4
3.1 Instruction Register and Control A–4
3.2 Memory A–4
3.3 Arithmetic Unit(ALU) A–5
3.4 I/O Buffer A–5
4.0 **Basic Instruction Set** A–6
4.1 Data Instruction Formats A–6
4.2 Summary of Processor Instructions A–6
4.3 Complete Functional Definition A–9
4.4 Internal Processor Operation A–14
5.0 **Processor Control Signals** A–17
5.1 Interrupt Signal(INT) A–17
5.2 Start-up of the MF8008 .... A–18
6.0 **Electrical Specification** A–20

MICROSYSTEMS INTERNATIONAL LIMITED, BOX 3529 STATION C, OTTAWA, CANADA, K1Y 4J1

SECTION A
MF8008 COMPONENT DESCRIPTION

1.0 INTRODUCTION

The MF8008 is a single chip MOS 8-Bit central processor unit for micro computer systems. A micro computer system is formed when the MF8008 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 (D0 through D7) and uses two input leads (READY and INTERRUPT) and four output leads(S0, 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, an 8-Bit accumulator, two 8-Bit temporary registers, four flag bits, and an 8-Bit parallel binary arithmetic unit which implements addition, subtraction, and logical operations. A memory stack containing a 14-Bit program counter and seven 14-Bit words is used internally to store program and subroutine addresses. The 14-Bit address permits the direct addressing of 16K words of memory (any mix of RAM, ROM).

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 MF8008 CPU executes non-memory referencing instructions in 20 microseconds. A selected device, the MF8008-1, 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 MF8008 consists of 48 instructions including data manipulation, binary arithmetic and jump to subroutine.

The normal program flow of the MF8008 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 MF8008 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.

2.0 PROCESSOR TIMING

The MF8008 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.

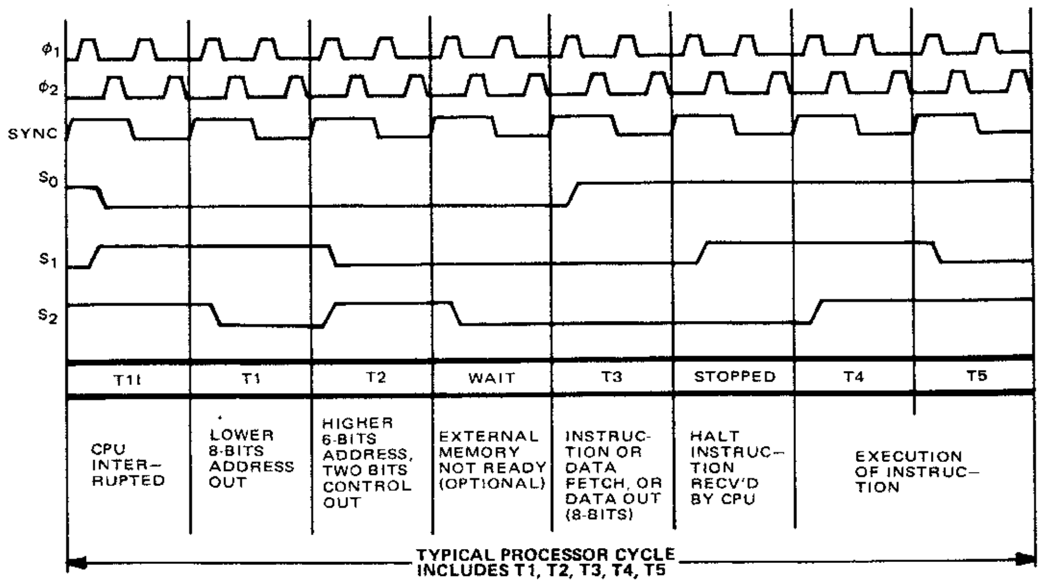
2.1 STATE CONTROL CODING

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

2.2 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.


φ1
φ2
SYNC
S0
S1
S2
T1I
T1
T2
WAIT
T3
STOPPED
T4
T5
CPU INTER– RUPTED
LOWER 8-BITS ADDRESS OUT
HIGHER 6-BITS ADDRESS, TWO BITS CONTROL OUT
EXTERNAL MEMORY NOT READY (OPTIONAL)
INSTRUC- TION OR DATA FETCH, OR DATA OUT (8-BITS)
HALT INSTRUC– TION RECV'D BY CPU
EXECUTION OF INSTRUC– TION
TYPICAL PROCESSOR CYCLE INCLUDES T1, T2, T3, T4, T5

ONE CYCLE OF PROCESSOR TIMING

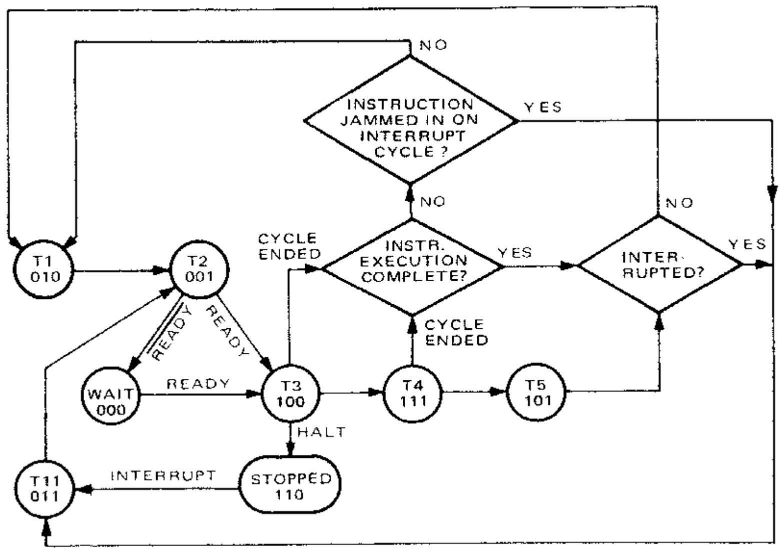
The receipt of an INTERRUPT is acknowledged by T1I. When the processor has been interrupted, this state replaces T1. A READY is acknowledged by T3. The stopped state acknowledges the receipt of a HALT instruction

Many of the instructions for the MF8008 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 MF8008 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.

2.3 CYCLE CONTROL CODING

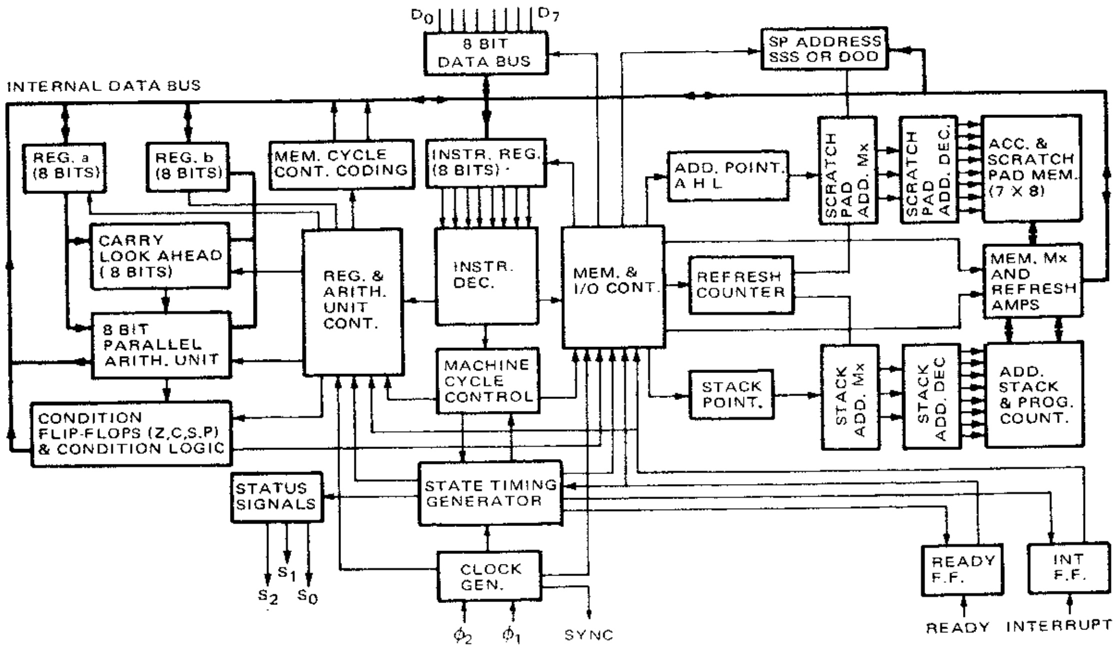
As previously noted, instructions for the MF8008 require one, two, or three machine cycles for complete execution. The first cycle is always an instruction fetch cycle (PCI). The second and third cycles are for data reading (PCR), data writing (PCW), or I/O Operations (PCC).

The cycle types are coded with two bits, D6 and D7 which are present on the data bus during T2.



EXTERNAL STATE TRANSITIONS

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 is a command I/0 operation.
1	1	PCW	Designates the address is for a memory write data.



SYSTEM BLOCK DIAGRAM

3.0 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.

3.1 INSTRUCTION REGISTER AND CONTROL

The instruction register is the centre 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.

3.2 MEMORY

Two separate dynamic memories are used in the MF8008, 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.

3.2.1. 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 addresss pointer recycles and the content of the lowest level register is destroyed. The program counter is incremented immediately after the lower order address bits are sent out. The higher order address bits are sent out at T2 and then incremented if a carry resulted from T1. The 14-bit program counter provides direct addressing of 16K bytes of memory. Through the use of an I/O instruction for bank switching, memory may be indefinitely expanded.

3.2.2. 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 and 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").

3.3 ARITHMETIC 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 intraprocessor 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.

3.4 I/O BUFFER

This buffer is the only link between the processor and the rest of the system. Each of the eight buffers is bidirectional 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.

4.0 BASIC INSTRUCTION SET

The Following section presents the basic instruction set of the MF8008.

4.1 DATA AND INSTRUCTION FORMATS

Data in the MF8008 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	DESCRIPTION	TYPICAL INSTRUCTIONS
D7 D6 D5 D4 D3 D2 D1 D0	Op Code	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	Immediate Mode Instructions
D7 D6 D5 D4 D3 D2 D1 D0	Operand	
Three Byte Instructions		
D7 D6 D5 D4 D3 D2 D1 D0	Op Code	Jump or call instruction
D7 D6 D5 D4 D3 D2 D1 D0	Low Address	
X X D5 D4 D3 D2 D1 D0	*High Address*[1]	

Note 1:
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.

4.2 SUMMARY OF PROCESSOR INSTRUCTIONS

4.2.1 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 D7 D6 D5 D4 D3 D2 D1 D0	DESCRIPTION OF OPERATION
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 B B B B B B B B	Load index register R with data B....B.
LMI	(9)	0 0 1 1 1 1 1 0 B B B B B B B B	Load memory register M with data B....B.
INR	(5)	0 0 D D D 0 0 0	Increment the content of index register R (R $\neq$ A).
DCR	(5)	0 0 D D D 0 0 1	Decrement the content of index register R (R $\neq$ A).

4.2.2 Accumulator Group Instructions

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

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	DESCRIPTION OF OPERATION
ADR	(5)	1 0 0 0 0 S S S	Add the content of index register R, memory register M, or data B....B to the accumulator. An overflow (carry) sets the carry flip-flop.
ADM	(8)	1 0 0 0 0 1 1 1	
ADI	(8)	0 0 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	
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.

MNEMONIC	MINIMUM STATES REQUIRED	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	DESCRIPTION OF OPERATION
JMP	(11)	0 1 X X X 1 0 0 B2 B2 B2 B2 B2 B2 B2 B2 X X B3 B3 B3 B3 B3 B3	Unconditionally jump to memory address B3...B3 B2...B2.
JFC	(9 or 11)	0 1 0 C4 C3 0 0 0 B2 B2 B2 B2 B2 B2 B2 B2	Jump to memory address B3...B3 B2...B2 if the condition flip-flop C is false. Otherwise, execute the next instruction in sequence.
JTC	(9 or 11)	0 1 1 C4 C3 0 0 0 B2 B2 B2 B2 B2 B2 B2 B2 X X B3 B3 B3 B3 B3 B3	Jump to memory address B3...B3 B2...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 B2 B2 B2 B2 B2 B2 B2 B2 X X B3 B3 B3 B3 B3 B3	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 C4 C3 0 1 0 B2 B2 B2 B2 B2 B2 B2 B2 X X B3 B3 B3 B3 B3 B3	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 C4 C3 0 1 0 B2 B2 B2 B2 B2 B2 B2 B2 X X B3 B3 B3 B3 B3 B3	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 C4 C3 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 C4 C3 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.)

4.2.3 Input/Output Instructions

INP	(8)	0 1 0 0 M M M 1	Read the content of the selected input port (MMM) into the accumulator.
OUT	(6)	0 1 R R M M M 1	Write the content of the accumulator into the selected output port (RRMMM, RR $\neq$ 00).

4.2.4 Machine Instructions

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 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 BBBBBBBB.
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).

4.3 COMPLETE FUNCTIONAL DEFINITION

The following pages present a detailed description of the complete MF8008 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
C4 C3	Flag flip-flop codes — Condition for true 00 Carry — Overflow, underflow 01 Zero — Result is zero 10 Sign — MSB of result is "1" 11 Parity — Parity of result is even
M	Memory location indicated by the contents of registers H and L
( )	Contents of location or register
$\wedge$	Logical product
$\forall$	Exclusive "OR"
V	Inclusive "OR"
AM	Bit M of the A-register
STACK	Instruction counter (P) pushdown register
P	Program counter
$\leftarrow$	Is transferred to
XXX	A "DON'T CARE"
SSS	Source Register for data
DDD	Destination register for data Register No. (SSS or DDD) — Register Name 000 — A 001 — B 010 — C 011 — D 100 — E 101 — H 110 — L

4.3.1 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) ← (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) ← (M) load register R with the content of the memory location addressed by the contents of registers H and L. (DDD ≠ 111 – HALT instr.)
LMR (Two Cycles – PCI/PCW)	11	111	SSS	(M) ← (R) load the memory location addressed by the contents of registers H and L with the content of register R. (SSS ≠ 111 – HALT instr.)

LOAD DATA IMMEDIATE – TWO BYTES

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

LRI (Two Cycles – PCI/PCR)	00	DDD	110	(R) ← < B2 > load byte two of the instruction into register R.
LMI (Three Cycles – PCI/PCR/PCW)	00 <B2>	111	110	(M) ← < 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) ← (R) + 1. The content of register R is incremented by one. All of the condition flip-flops except carry are affected by the result. Note that DDD ≠ 000 (halt instr.) and DDD ≠ 111 (content of memory cannot be incremented).

DECREMENT INDEX REGISTER – ONE BYTE

DCR (One Cycle – PCI)	00	DDD	001	(R) ← (R) – 1. The content of register R is decremented by one. All of the condition flip-flops except carry are affected by the result. Note that DDD ≠ 000 (HALT instr.) and DDD ≠ 111 (content of memory can not be decremented).

4.3.2 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) ← (A) + (R) Add the content of register R to the content of register A and place the result into register A.
ACR	10	001	SSS	(A) ← (A) + (R) + (Carry) Add the content of register R and the contents of the carry flip-flop to the contents of the A register and place the result into register A.
SUR	10	010	SSS	(A) ← (A) – (R) Subtract the content of register R from the content of register A and place the result into register A. Two's complement subtraction is used.
SBR	10	011	SSS	(A) ← (A) – (R) – (Borrow) Subtract the content of register R and the content of the carry flip-flop from the content of register A and place the result into register A.

NDR	10	100	SSS	(A) ← (A) ∧ (R) Place the logical product of the register A and register R into register A.
XRR	10	101	SSS	(A) ← (A) ⊻ (R) Place the "exclusive – or" of the content of register A and register R into register A.
ORR	10	110	SSS	(A) ← (A) V (R) Place "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)←(A)+(M) ADD
ACM	10 001 111	(A)←(A)+(M)+(CARRY) ADD WITH CARRY
SUM	10 010 111	(A)←(A)–(M) SUBTRACT
SBM	10 011 111	(A)←(A)–(M)–(BORROW) SUBTRACT WITH BORROW
NDM	10 100 111	(A)←(A)∧(M) LOGICAL AND
XRM	10 101 111	(A)←(A)⊻(M) EXCLUSIVE OR
ORM	10 110 111	(A)←(A)∨(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)←(A)+<B2>
ACI	00 001 100 <B2>	(A)←(A)+<B2>+(CARRY)
SUI	00 010 100 <B2>	(A)←(A)–<B2> SUBTRACT
SBI	00 011 100 <B2>	(A)←(A)–<B2>–(BORROW) SUBTRACT WITH BORROW
NDI	00 100 100 <B2>	(A)←(A)∧<B2> LOGICAL AND
XRI	00 101 100 <B2>	(A)←(A)⊻<B2> EXCLUSIVE OR
ORI	00 110 100 <B2>	(A)←(A)∨<B2> INCLUSIVE OR
CPI	00 111 100 <B2>	(A)–<B2> 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, AO ← 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.

4.3.3 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 (Jump Unconditionally)	01	XXX 100 <B2> <B3>	(P) ← <B3><B2> jump unconditionally to the instruction located in memory location addressed by byte two and byte three.
JFC (Jump if Condition False)	01	0C4C3 000 <B2> <B3>	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.
JTC (Jump if Condition True)	01	1C4C3 000 <B2> <B3>	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.

CALL INSTRUCTIONS – Three Bytes
(Three Cycles – PCI/PCR/PCR)

Subroutines may be called and nested up to seven levels.

CAL (Call Subrouting Unconditionally)	01	XXX 110 <B2> <B3>	(Stack) ← (P), (P) ←<B3><B2>. Shift the content P to the pushdown stack. Jump unconditionally to the instruction located in memory location addressed by byte two and byte three.

CFC (Call Subroutine if Condition False)	01	0C4C3010 <B2> <B3>	If (C) = 0, (stack) ← (P), (P) ← <B3><B2>. Otherwise, (P) = (P) + 3. If the content of flip-flop C is zero, then shift contents of P to the pushdown stack and jump to the instruction located in memory location <B3> <B2>; otherwise, execute the next instruction in sequence.
CTC (Call Subroutine if Condition True)	01	1C4C3 010 <B2> <B3>	If (C) = 1, (stack) ← (P), (P) ←<B3><B2>. Otherwise,(P)=(P)+3. If the content of flip-flop C is one, then shift contents of P to the pushdown stack and jump to the instruction located in memory location <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) ← (Stack). Return to the instruction in the memory location addressed by the last value shifted into the pushdown stack. The stack pops up one level.
RFC (Return Condition False)	00	0C4C3 011	If (C) = 0, (P) ← (Stack); otherwise, (P) = (P) + 1. If the content of flip-flop C is zero, then return to the instruction in the memory location addressed by the last value inserted in the pushdown stack. The stack pops up one level. Otherwise, execute the next instruction in sequence.
RTC (Return Condition True)	00	1C4C3 011	If (C) = 1, (P) ← (Stack); otherwise, (P) = (P) + 1. If the content of flip-flop C is one, then return to the instruction in the memory location addressed by the last value inserted in the pushdown stack. The stack pops up one level. Otherwise, execute the next instruction in sequence.

RESTART INSTRUCTION – One Byte
(One Cycle – PCI)

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

RST	00	AAA 101	(Stack) ← (P), (P) ← (000000 00AAA000) Shift the contents of P to the pushdown stack. The content, AAA, of the instruction register is shifted into bits 3 through 5 to 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.

4.3.4 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 the 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)

4.3.5. Machine Instruction

HALT INSTRUCTION – One Byte
(One Cycle – PCI)

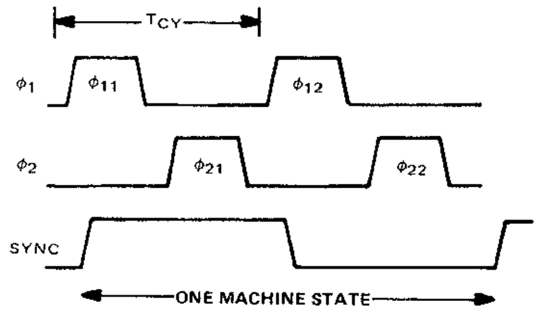
HLT	00 11	000 or 111	00X 111	On receipt of the HALT instruction, the activity of the processor is immediately suspended in the stopped state. The content of all registers and memory is unchanged. The P—counter has been updated and the internal dynamic memories continue to be refreshed.

4.4 INTERNAL PROCESSOR OPERATION

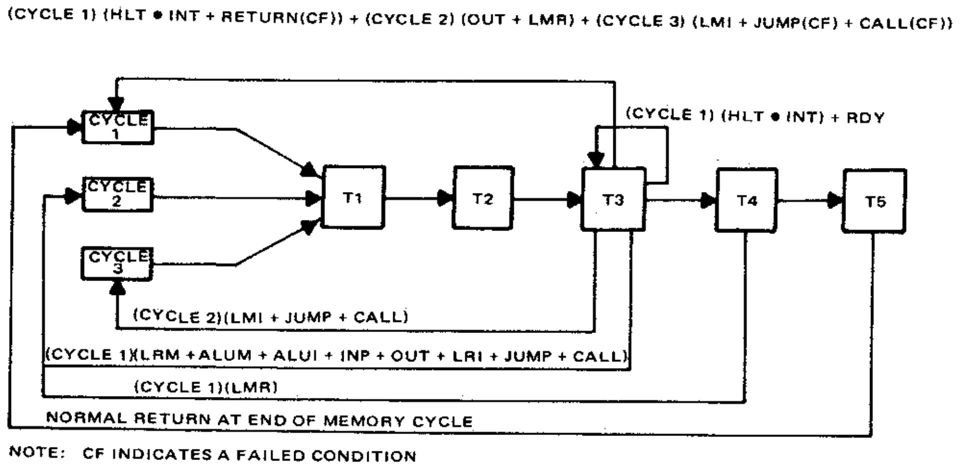
Internally the processor operates through five different states:

INTERNAL STATE		TYPICAL FUNCTION
T1	Normal	Send out lower eight bits of address and increment program counter.
	Interrupt	Send out lower eight bits of address and suppress incrementing of program counter and acknowledge interrupt.
T2		Send out six higher order bits of address and two control bits, D6 and D7. Increment program counter if there has been a carry from T1.
T3	Wait	Wait for ready signal to come true. Refresh internal dynamic memories while waiting.
	Normal	Fetch and decode instruction; fetch data from memory; output data to memory. Refresh internal memories.
	Stopped	Remain stopped until interrupt occurs. Refresh internal memories.
T4 and T5		Execute instruction and appropriately transfer data within processor. Content of data bus transfer is available at I/0 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 MF8008 is driven by two non-overlapping clocks. Two clock periods are required for each state of the processor. Ø1 is generally used to precharge all data transfers within the processor. A sync signal (divide by two of Ø2) is sent out by the MF8008. 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 MF8008 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.


(CYCLE 1) (HLT • INT + RETURN(CF)) + (CYCLE 2) (OUT + LMR) + (CYCLE 3) (LMI + JUMP(CF) + CALL(CF))
(CYCLE 1) (HLT • INT) + RDY
CYCLE 1
CYCLE 2
CYCLE 3
T1
T2
T3
T4
T5
(CYCLE 2)(LMI + JUMP + CALL)
(CYCLE 1)(LRM + ALUM + ALUI + INP + OUT + LRI + JUMP + CALL)
(CYCLE 1)(LMR)
NORMAL RETURN AT END OF MEMORY CYCLE
NOTE: CF INDICATES A FAILED CONDITION

STATE TRANSITIONS

INTERNAL PROCESSOR OPERATION

INSTRUCTION CODING			OPERATION	# OF STATES TO EXECUTE INSTRUCTION	MEMORY CYCLE ONE (1)					MEMORY CYCLE TWO					MEMORY CYCLE THREE				
D_7D_6	$D_5D_4D_3$	$D_2D_1D_0$			T1(2)	T2	T3	T4(3)	T5	T1	T2	T3	T4	T5	T1	T2	T3	T4	T5
INDEX REGISTER INSTRUCTIONS																			
1 1	D D D	S S S	Lr_1r_2	(5)	PC_LOUT (4)	PC_HOUT	FETCH INSTR.(5) TO IR & REG. b	SSS TO REG. b (6)	REG. b TO DDD										
1 1	D D D	1 1 1	LrM	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→ (7)	→	REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X (9)	REG. b TO DDD					
1 1	1 1 1	S S S	LMr	(7)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	→	REG. L OUT (10)	REG. H OUT	REG. b TO OUT							
0 0	D D D	1 1 0	LrI	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	DATA TO REG. b	X	REG. b TO DDD					
0 0	1 1 1	1 1 0	LMI	(9)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	DATA TO REG. b	→	→	REG. L OUT(10)	REG. H OUT	REG. b TO OUT		
0 0	D D D	0 0 0	INr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ADD OP - FLAGS AFFECTED										
0 0	D D D	0 0 1	DCr	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	SUB OP - FLAGS AFFECTED										
ACCUMULATOR GROUP INSTRUCTIONS																			
1 0	P P P	S S S	ALU OP r	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	SSS TO REG. b	ALU OP - FLAGS AFFECTED										
1 0	P P P	1 1 1	ALU OP M	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	REG. L OUT (8)	REG. H OUT	DATA TO REG. b	X	ALU OP - FLAGS AFFECTED					
0 0	P P P	1 0 0	ALU OP I	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	DATA TO REG. b	X	ARITH OP - FLAGS AFFECTED					
0 0	0 0 0	0 1 0	RLC	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
0 0	0 0 1	0 1 0	RRC	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
0 0	0 1 0	0 1 0	RAL	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
0 0	0 1 1	0 1 0	RAR	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	X	ROTATE REG. A CARRY AFFECTED										
PROGRAM COUNTER AND STACK CONTROL INSTRUCTIONS																			
0 1	X X X	1 0 0	JMP	(11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L
0 1	0 C C	0 0 0	JFc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L
0 1	1 C C	0 0 0	JTc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a (11)	REG. a TO PC_H	REG. b TO PC_L
0 1	X X X	1 1 0	CAL	(11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a	REG. a TO PC_H	REG. b TO PC_L
0 1	0 C C	0 1 0	CFc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a (12)	REG. a TO PC_H	REG. b TO PC_L
0 1	1 C C	0 1 0	CTc	(9 or 11)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	PC_LOUT (8)	PC_HOUT	LOWER ADD. TO REG. b	→	→	PC_LOUT(8)	PC_HOUT	HIGHER ADD. REG. a (12)	REG. a TO PC_H	REG. b TO PC_L
0 0	X X X	1 1 1	RET	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK	X										
0 0	0 C C	0 1 1	RFc	(3 or 5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X										
0 0	1 C C	0 1 1	RTc	(3 or 5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	POP STACK (13)	X										
0 0	A A A	1 0 1	RST	(5)	PC_LOUT	PC_HOUT	FETCH INSTR. TO REG. b AND PUSH STACK (0→REG. a)	REG. a TO PC_H	REG. b TO PC_L (14)										
I/O INSTRUCTIONS																			
0 1	0 0 M	M M 1	INP	(8)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	REG. A TO OUT (15)	REG. b TO OUT	DATA TO REG. b	COND FF OUT (16)	REG. b TO REG. A					
0 1	R R M	M M 1	OUT	(6)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b	→	→	REG. A TO OUT (15)	REG. b TO OUT	X (17)							
MACHINE INSTRUCTIONS																			
0 0	0 0 0	0 0 X	HLT	(4)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b & HALT (18)												
1 1	1 1 1	1 1 1	HLT	(4)	PC_LOUT	PC_HOUT	FETCH INSTR. TO IR & REG. b & HALT (18)												

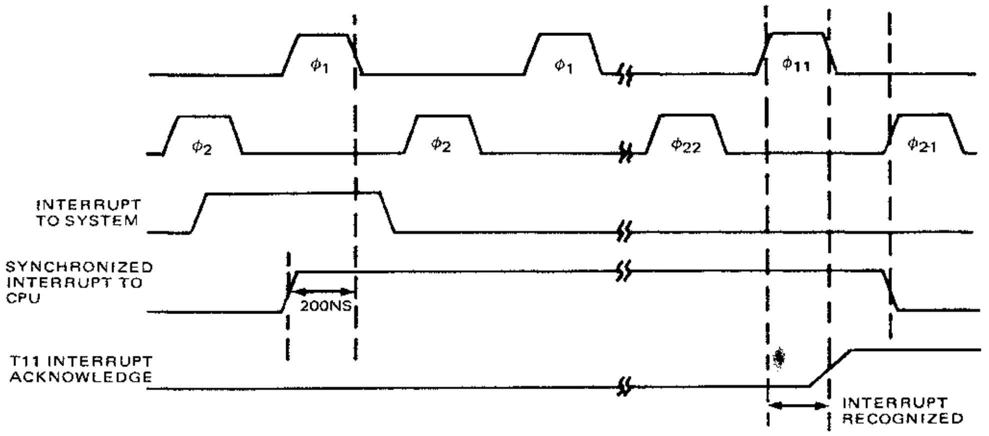
NOTES:
1. The first memory cycle is always a PCI (instruction) cycle.
2. Internally, states are defined as T1 through T5. In some cases more than one memory cycle is required to execute an instruction.
3. Content of the internal data bus at T4 and T5 is available at the data bus. This is designed for testing purposes only.
4. Lower order address bits in the program counter are denoted by PC_L and higher order bits are designated by PC_H.
5. During an instruction fetch the instruction comes from memory to the instruction register and is decoded.
6. Temporary registers are used internally for arithmetic operations and data transfers (Register a and Register b.)
7. These states are skipped.
8. PCR cycle (Memory Read Cycle).
9. "X" denotes an idle state.
10. PCW cycle (Memory Write Cycle).
11. When the JUMP is conditional and the condition fails, states T4 and T5 are skipped and the state counter advances to the next memory cycle.
12. When the CALL is conditional and the condition fails, states T4 and T5 are skipped and the state counter advances to the next memory cycle. If the condition is true, the stack is pushed at T4, and the lower and higher order address bytes are loaded into the program counter.
13. When the RETURN condition is true, pop up the stack; otherwise, advance to next memory cycle skipping T4 and T5.
14. Bits D_3 through D_5 are loaded into PC_L and all other bits are set to zero; zeros are loaded into PC_H.
15. PCC cycle (I/O Cycle).
16. The content of the condition flip-flops is available at the data bus: S at D_0, Z at D_1, P at D_2, C at D_3.
17. A READY command must be supplied for the OUT operation to be completed. An idle T3 state is used and then the state counter advances to the next memory cycle.
18. When a HALT command occurs, the CPU internally remains in the T3 state until an INTERRUPT is recognized. Externally, the STOPPED state is indicated.

5.0 PROCESSOR CONTROL SIGNALS

5.1 INTERRUPT SIGNAL (INT)

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 S0S1S2 = 011 at T1I time. The lower and higher order address bytes of the program counter are sent out, but the program counter is not advanced. A successive instruction fetch cycle can be used to insert an arbitrary instruction into the instruction register in the CPU. (If a multi-cycle or multi-byte instruction is inserted, an interrupt need only be inserted for the first cycle.)

When the processor is interrupted, the system interrupt signal must be synchronized with the leading edge of the Ø1 or Ø2 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 Ø1. An example of a synchronizing circuit is shown on the schematic for the CPU board.



INTERRUPT TIMING

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.

ADDR.	LOCATION	PC CONTENTS	
N − 1	INTR. N − 1	N	(INTERRUPT ARRIVES HERE)
N	INSTR. N		
N + 1	INSTR. N + 1		

USER SUPPLIES ALTERNATE INSTRUCTION(RESTART OR CALL TO SRT), RELEASES INTERRUPT, PC IS SAVED IN STACK (VALUE = N)

SUBROUTINE FOR HANDLING INTERRUPT:

S	INSTR. S					
S + 1	INSTR. S + 1					
S + 2						
S + K	RETURN	–	STACK POPS	–	WITH VALUE N	

AFTER COMPLETION OF SUBROUTINE, MF8008 RETURNS TO EXECUTE ORIGINALLY REQUESTED INSTRUCTION, WHICH BLOCKING ADVANCE OF PC HAS SAVED.

5.2 START-UP OF THE MF8008

When power (VDD) and clocks (Ø1, Ø2) are turned on, a flip-flop internal to the MF8008 is set by sensing the rise of VDD. This internal signal forces a HALT (00000000) into the instruction register and the MF8008 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 is at logic "0". Any time after the memories are cleared, the MF8008 is ready for normal operation.

To reset the flip-flop and also escape from the stopped state, the interrupt line must go to a logic "1"; it should be returned to logic "0" by decoding the state T1I at some time later than Ø11. Note that whenever the MF8008 is in a T1I state, the program counter is not incremented. As a result, the same address is sent out on two successive cycles.

Three possible sequences for starting the MF8008 are shown in the following examples. 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 MF8008 with a restart instruction.

The selection of initiation technique to use depends on the sophistication of the system using the MF8008. If the interrupt feature is used only for the start-up of the MF8008 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 MF8008, 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 in the timing diagram. 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 MF8008 during the interrupt cycle. The normal program flow starts the 8008.

MF8008 Address out														**Instruction in ROM**		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP	(LAA 11 000 000)	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	NOP		Entry Directly to Main Program
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	0	1	0	INSTR2		

MF8008 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	INSTR1		
0	0	0	0	0	0	0	0	X	Y	Z	0	0	1	INSTR2		
						.								.		
						.								.		
						.								.		

Example 2:

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

MF8008 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	INSTRA		
0	0	0	0	0	0	0	0	X	Y	Z	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)	
0	0	0	0	0	0	0	0	0	0	0	0	0	1	INSTR2		Main Program
						.								.		
						.								.		
						.								.		

Note that during the interrupt cycle the flow of the instruction to the MF8008 either from ROM or another source must be controlled by hardware external to MF8008.

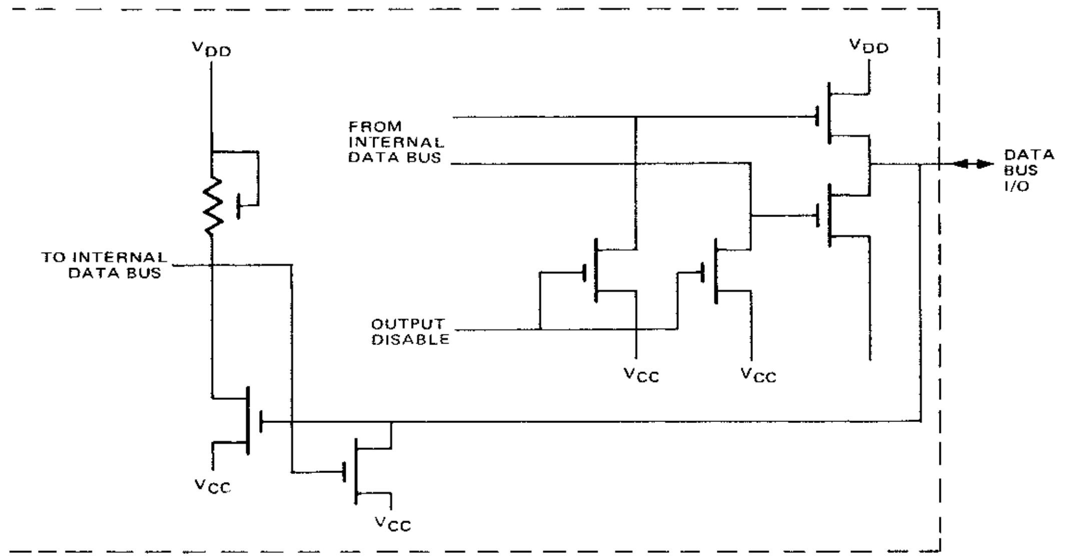
5.2.1 Ready (RDY)

The MF8008 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 MF8008 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 MF8008 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 MF8008 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.

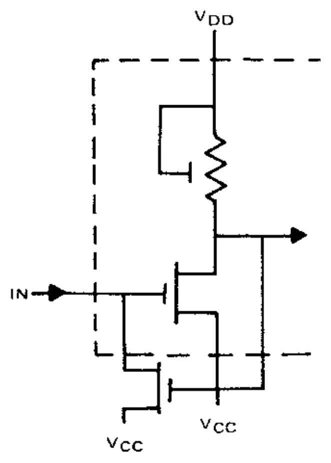
6.0 ELECTRICAL SPECIFICATION

The following data provides the electrical characteristics for the MF8008. All of the inputs are TTL compatible but input pull-up resistors are recommended to insure proper VIH 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.

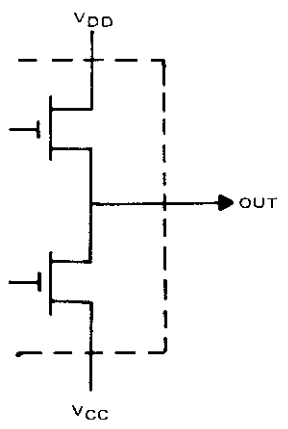
THE MF8008 DATA BUS ARRANGEMENT



INPUT AND OUTPUT BUFFERING



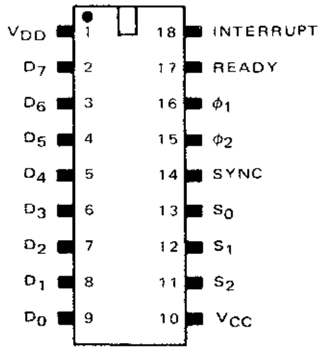
INPUT BUFFER
(ϕ_1, ϕ_2, RDY, INT)



OUTPUT BUFFER
(SYNC, S_0, S_1, S_2)

PIN CONFIGURATION

(Top View)



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}	+0.3V to −20V
Power Dissipation	1.0 W @ +25°C

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

AC CHARACTERISTICS

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

PARAMETER	SYMBOL	CONDITIONS	MF8008 MIN.	MF8008 MAX.	MF8008-1 MIN.	MF8008-1 MAX.	UNIT
Clock period	t_{CY}	$t_R t_F$ = 50ns	2	3	1.25	3	μs
Clock rise and fall times	t_R, t_F	–	–	50	–	50	ns
Pulse width of ϕ 1	$t\phi_1$	–	.70	–	.35	–	μs
Pulse width of ϕ_2	$t\phi_2$	–	.55	–	.35	–	μs
Clock delay from falling edge of ϕ_1 to falling edge of ϕ_2	t_{D1}	–	.90	1.1	–	1.1	μs
Clock delay from ϕ_2 to ϕ_1	t_{D2}	–	.40	–	.35	–	μs
Clock delay from ϕ_1 to ϕ_2	t_{D3}	–	.20	–	.20	–	μs
Data out delay	t_{DD}	C_L= 100pF	–	1.0	–	1.0	μs
Hold time for data out	t_{OH}	–	.10	–	.10	–	μs
Hold time for data in (1)	t_{IH}	–	–	–	–	–	
Sync out delay	t_{SD}	C_L = 100pF	–	.70	–	.70	μs
State out delay (all states except T1 and T11)(2)	t_{S1}	C_L = 100pF	–	1.1	–	1.1	μs
State and delay (states T1 and T11)	t_{S2}	C_L = 100pF	–	1.0	–	1.0	μs
Pulse width of ready during ϕ_{22} to enter T3 state	t_{RW}	–	.35	–	.35	–	μs
Ready delay to enter wait state	t_{RD}	–	.20	–	.20	–	μs

NOTES: 1. $t_{IH}MIN \geqslant t_{SD}$ 2. If the INTERRUPT is not used, all states have the same output delay, t_{S1}.

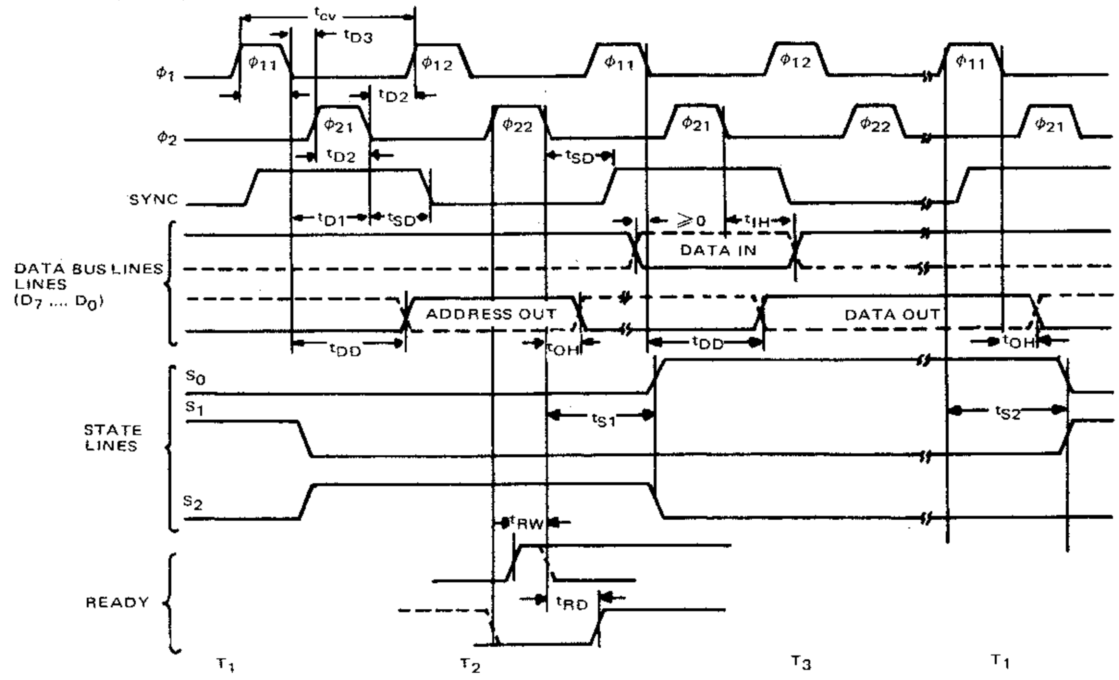
DC AND OPERATING CHARACTERISTICS

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}).

PARAMETER	SYMBOL	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Average supply current – *outputs loaded*(1)	I_{DD}	T_A =+25°C	–	30	60	mA
Input leakage current	I_{LI}	V_{IN} = 0V	–	–	10	μA
Input low voltage (Including clocks)	V_{IL}	–	V_{DD}	–	V_{CC}−4.2	V
Input high voltage (Including clocks)	V_{IH}	–	V_{CC}−1.5	–	V_{CC} +0.3	V
Output low voltage	V_{OL}	I_{OL} =0.44mA C_L =200 pF	–	–	0.4	V
Output high voltage	V_{OH}	I_{OH} =0.2mA	V_{CC}−1.5	–	–	V

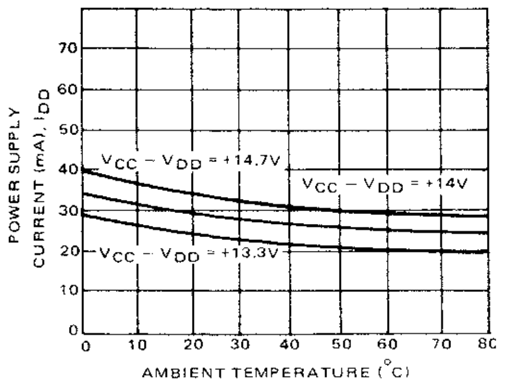
NOTE 1. Measurements are made while the MF8008 is executing a typical sequence of instructions. Test load: at V_{OL} = 0.4V, I_{OL} = 0.44mA each O/P.

TIMING DIAGRAM

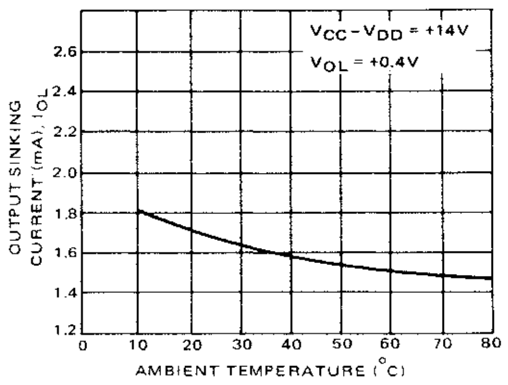


TYPICAL CHARACTERISTICS

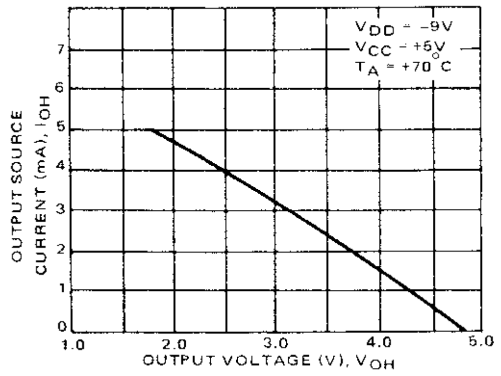
POWER SUPPLY CURRENT VS. TEMPERATURE



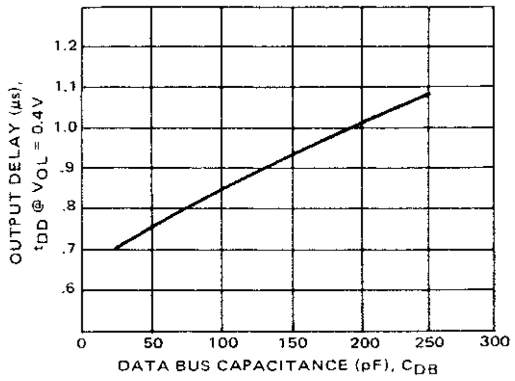
OUTPUT SINKING CURRENT VS. TEMPERATURE



OUTPUT SOURCE CURRENT VS. OUTPUT VOLTAGE



DATA OUTPUT DELAY VS. OUTPUT LOAD CAPACITANCE



CAPACITANCE f = 1MHz; $T_A = 20°C$; Unmeasured Pins Grounded.

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

MICRO SYSTEMS
a Northern Electric company

MF8008 APPLICATIONS MANUAL

1.0 MOD8, MF8008 Based Modular Microcomputer B–2
1.1 MOD8 CPU Board B–2
1.2 MOD8 Restart, TTY I/O Board B–2
1.3 MOD8 Control/Buffer Board B–2
1.4 MOD8 ROM Board B–2
1.5 MOD8 RAM Board B–2
1.6 MOD8 Input Board B–2
1.7 MOD8 Output Board B–2
2.0 MOD8 Detailed Board Description B–3
2.1 MOD8 CPU Board B–3
2.2 Restart & TTY I/O Board B–4
2.3 Control/Buffer Board B–4
2.4 ROM Board B–4
2.5 RAM Board B–5
2.6 Input Board B–5
2.7 Output Board B–5
2.8 MOD8 Schematic & PCB Layout Diagrams B–5

MICROSYSTEMS INTERNATIONAL LIMITED, BOX 3529 STATION C, OTTAWA, CANADA, K1Y 4J1

SECTION B
MODULAR MICROCOMPUTER

1.0 MOD8, AN MF8008 BASED MODULAR MICROCOMPUTER

The MOD8 is a modular MF8008 based prototyping system. This hardware in conjunction with the MONITOR8 software described in Section C allows the user to develop his own hardware system to cater to his particular requirements. The basic MOD8 configuration consists of seven 4 1/2" X 6" printed circuit boards with the following functions:

1.1 MOD8 CPU BOARD (MOD 8–1)

This board contains the MF8008 CPU, clock generators, state decoding and bus switching control logic.

1.2 MOD8 RESTART, TTY I/O BOARD (MOD 8–2)

This board contains teletype I/O, reader control and system restart logic.

1.3 MOD8 CONTROL/BUFFER BOARD (MOD 8–3)

This board contains 8-Bit bi-directional bus switches and address latches.

1.4 MOD8 ROM BOARD (MOD 8–4)

This board contains 2K X 8 of MF1702 pROM or MF1302 mask programmable ROM.

1.5 MOD8 RAM BOARD (MOD 8–5)

This board contains 2K X 8 of MF2102 RAM.

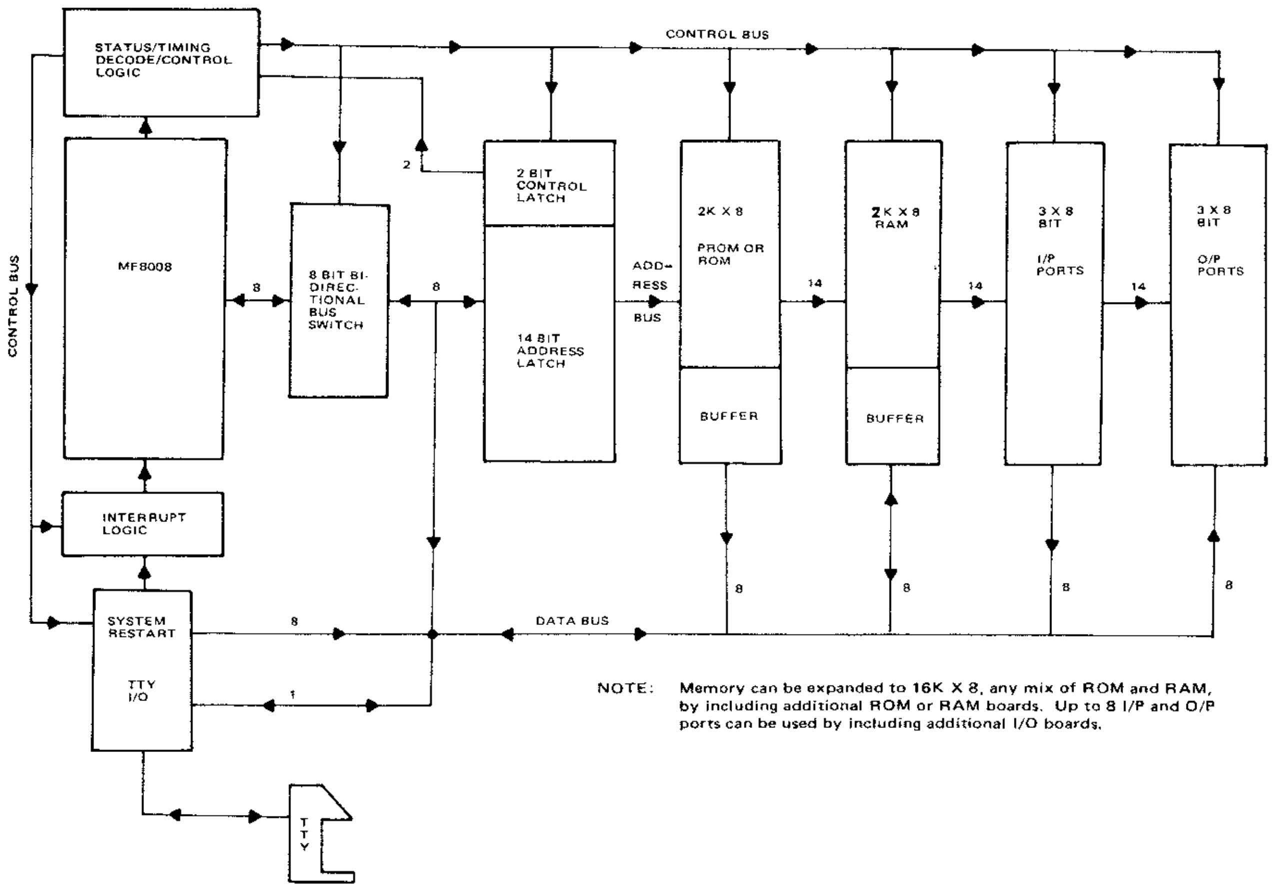
1.6 MOD8 INPUT BOARD (MOD 8–6)

This board contains three 8-Bit input channels.

1.7 MOD8 OUTPUT BOARD (MOD 8–7)

This board contains three 8-Bit output channels.

The MOD8 system decodes the time division multiplexed signals from the MF8008 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 MOD8 system organization is shown below. The MOD8 system is designed to run with 500 KHz symmetric non-overlapping clocks.



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 O/P ports can be used by including additional I/O boards.

MOD8 MODULAR MICROCOMPUTER

2.0 MOD 8 DETAILED BOARD DESCRIPTIONS.

2.1 CPU BOARD (MOD8–1)

The MF8008 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 Ø1 pulse and 15A the Ø2 pulse which are then fed to the MF8008. The 8 data lines of the MF8008 are provided with eight 22KΩ pull-up resistors and are only loaded with one LPTTL input each. The MF8008 READY line normally is tied to a logic 1 via a 10KΩ resistor. This means that the MF8008 will not pause at T3. The READY 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 READY line to single step through the program. The SYNC signal is buffered out from the MF8008 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 MD6150 or 3205 one out of eight decoder (6). This decoder only presents a low power input load to the MF8008. The outputs of the decoder are normally high, only going low for the selected output. $\overline{\text{T1}}$ signal from the decoder is fed via 11A to 11B where it is combined with $\overline{\text{S}}$.Ø2. Thus the output of 11B goes low during Ø2 of the second half of the T1 cycle. This signal $\overline{\text{ADLL}}$ 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. ADLL is also used to clock 10B which samples the decoder $\overline{\text{T1I}}$ output. $\overline{\text{T1}}$ and $\overline{\text{T1I}}$ are combined in 11A and will both produce an ADLL sample pulse. T1I is generated in place of T1 when the processor has just been interrupted. When a $\overline{\text{T1I}}$ state has been generated 10B will hold a low on Q and a high on $\overline{\text{Q}}$ until it is restrobed at the start of the next instruction. 4D, 12C and 4A generate S.Ø2, S, $\overline{\text{S}}$, $\overline{\text{S}}$.Ø2 where S=SYNC. During T2 $\overline{\text{ADHL}}$ 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. $\overline{\text{ADHL}}$ is also used to clock 10A which will output a high on Q until $\overline{\text{S}}$ 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, T3A, is a phase advanced signal synthesized ahead of T3 and is used to generate control signals determining the data flow on the 8-bit data bus. Interrupt signals can origniate 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 10KΩ pull-up resistor and INTB is grounded. 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 is transmitted to the Q output of 14, which provides the interrupt to pin 18 of the MF8008. When the interrupt has been acknowledged and a $\overline{\text{T1I}}$ state generated, the high level from 10B is combined with T2 and $\overline{\text{S}}$.Ø2 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 $\overline{\text{WRITE}}$ 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 $\overline{\text{Q}}$ 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.Ø2 of T3 which will set 13A with a low output on $\overline{\text{Q}}$ (i.e. the RAM is in the write mode) until 13A is cleared again by the clock at the end of $\overline{\text{S}}$.Ø2.

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 $\overline{\text{ALBE}}$ goes low during T3A. Similarly OUT is low during S.Ø2 of T3A during an output operation. $\overline{\text{ALBE}}$ is used to transmit data from the low order 8-Bit address latches back onto the data bus. $\overline{\text{OUT}}$ 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 MF8008 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 $\overline{\text{(A12 + A13)}}$ via 2C, when the PCC operation specifically is an input operation.

During $\overline{\text{S}}$.Ø2 in T4 of a PCC input operation the condition flip-flops S, Z, P and C are available at the MF8008 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 $\overline{\text{IBS}}$ 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.

2.2 RESTART AND TTY I/0 BOARD (MOD8–2)

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 $\overline{\text{IBS}}$ 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 MF8008 on receipt of a TTY interrupt. This has the effect of releasing the MF8008 from the stopped state which it enters in the MONITOR8 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 $\overline{\text{IBS}}$ now goes low, as a result of the interrupt generated by pressing the push button reset, the upper diode array forces 00000101 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 MF8008 on receipt of a push button reset interrupt. This has the effect of restarting execution at the beginning of the MONITOR8 software.

TTY INT enable is normally held high by a 10KΩ pull-up resistor enabling 7D to pass signals from the TTY input. In the idle mode the TTY provides closed contacts between 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 transistions from low to high. This causes a TTY interrupt to be generated on the CPU board. After the TTY input routine has been entered $\overline{\text{INP}}$ 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 $\overline{\text{OUT}}$, 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 $\overline{\text{1Q}}$ output of latch 8. The $\overline{\text{1Q}}$ 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 $\overline{\text{3Q}}$ of 8. This output drives the reader control solenoid, allowing the paper tape reader to be started and stopped under program control.

2.3 CONTROL BUFFER BOARD (MOD8–3)

Packages 1, 2, 10, 11, 5, 6, 7 provide a controlled bidirectional bus switch which communicates between the low power MF8008 8-Bit data lines (8-BO 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 10KΩ pull-up resistor. The BSE line is brought out to allow isolation of the MF8008 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 MF8008 data bus to the system data bus. When DOE is low data flows from the system data bus to the MF8008. The bus switch only presents one LPTTL load on each of the MF8008 data lines. The eight 10KΩ resistors on the system data bus provide pull-up loads for the open collecter 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 $\overline{\text{ADLL}}$. 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 $\overline{\text{ADHL}}$. 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 $\overline{\text{ALBE}}$ 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.

2.4 ROM BOARD (MOD8–4)

Packages 13, 12, 10, 9, 7, 6, 5 and 4 provide 2K x 8 of MF1702 pROM or MF1302 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 $\overline{\text{CS}}$ 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 $\overline{\text{E1}}$, $\overline{\text{E2}}$, and 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.

2.5 RAM BOARD (MOD8–5)

Packages 19, 17, 14, 12, 9, 6, 3 and 1 provide 1K x 8 of MF2102 RAM similarly 20, 18, 15, 13, 10, 7, 4 and 2 provide a further 1K x 8 of RAM. The $\overline{\text{WRITE}}$ control line dictates whether the RAM is in the read or write mode. The RAM is in the write mode when $\overline{\text{WRITE}}$ 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.

2.6 INPUT BOARD (MOD8–6)

This board provides 3 8-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 $\overline{\text{INP}}$ 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.

2.7 OUTPUT BOARD (MOD8–7)

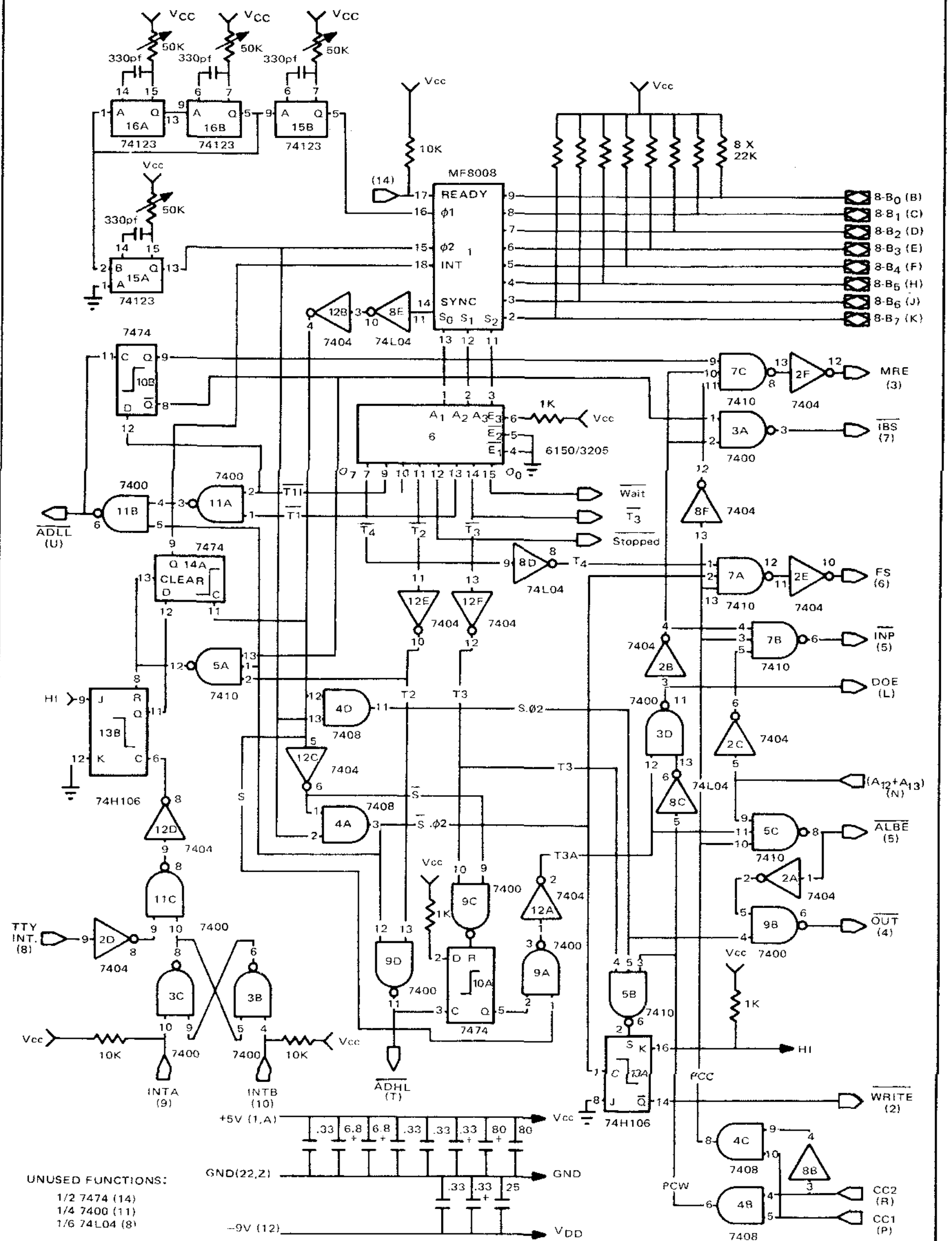
This provides 3 8-Bit output channels. DBO–DB7, the 8-Bit system data bus is buffered in via open collecter buffers with 10KΩ pull-up resistors on their outputs. Up to 24 8-Bit output ports can be accomodated by the system. They are selected by a one out of eight decode performed on A9 - A11. For an output function A12, A13 can furthermore have 3 states 10, 11 and 01. These additonal 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 $\overline{\text{OUT}}$ 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.

2.8 MOD 8 SCHEMATICS AND PCB LAYOUTS

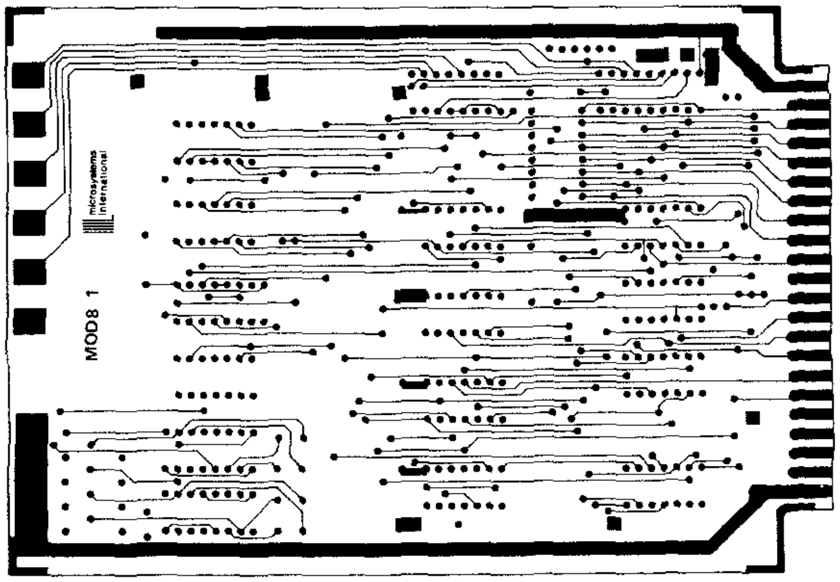
The following diagrams depict the circuit schematic, component side PCB layout, underside PCB layout and component placement. A recommended supplier of the MOD 8-1, MOD 8-2, MOD 8-3, MOD 8-4, MOD 8-5, MOD 8-6 and MOD 8-7 unpopulated PCB's is:

Space Circuits Ltd.
156 Roger Street,
Waterloo, Ontario.
Canada.

The assembled tested boards may be purchased from Microsystems International Ltd. A PCB back plane to interconnect the modules is also available with or without pROM programming capability.

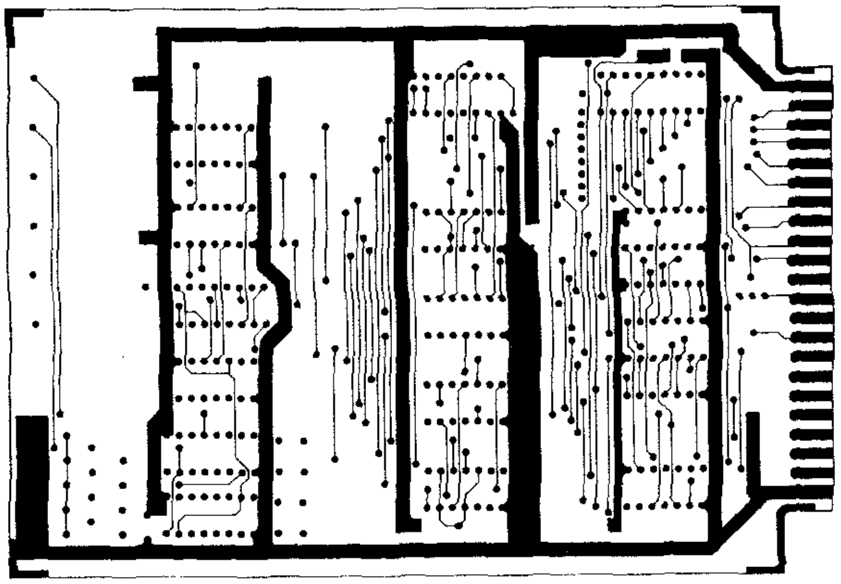


MOD8–1(CPU) CIRCUIT SCHEMATIC



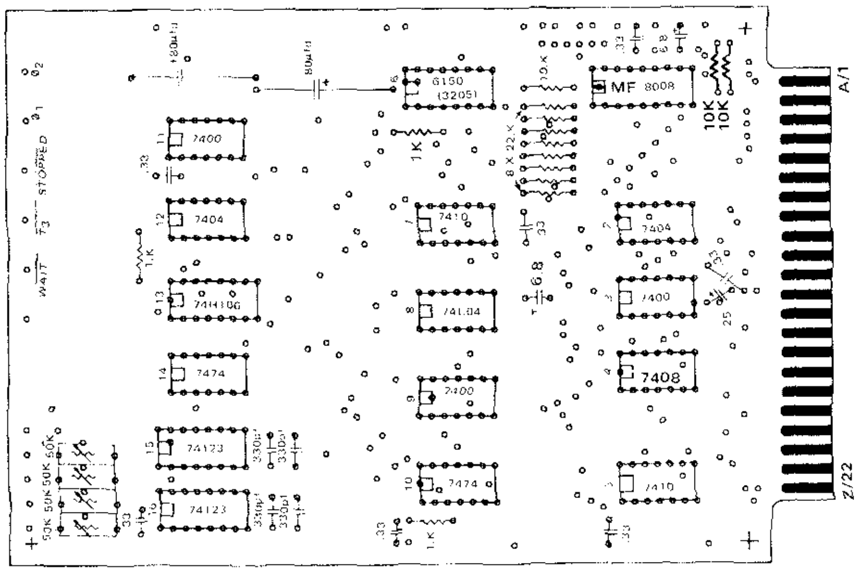
A. +5V
B. 8 B0
C. 8 B1
D. 8 B2
E. 8 B3
F. 8 B4
H. 8 B5
J. 8 B6
K. 8 B7
L. DOE
M.
N. A12+A13
P. CC1
R. $\overline{\text{CC2}}$
S. $\overline{\text{ALBE}}$
T. $\overline{\text{ADHL}}$
U. $\overline{\text{ADLL}}$
V.
W.
X.
Y.
Z. OV

(TOP)



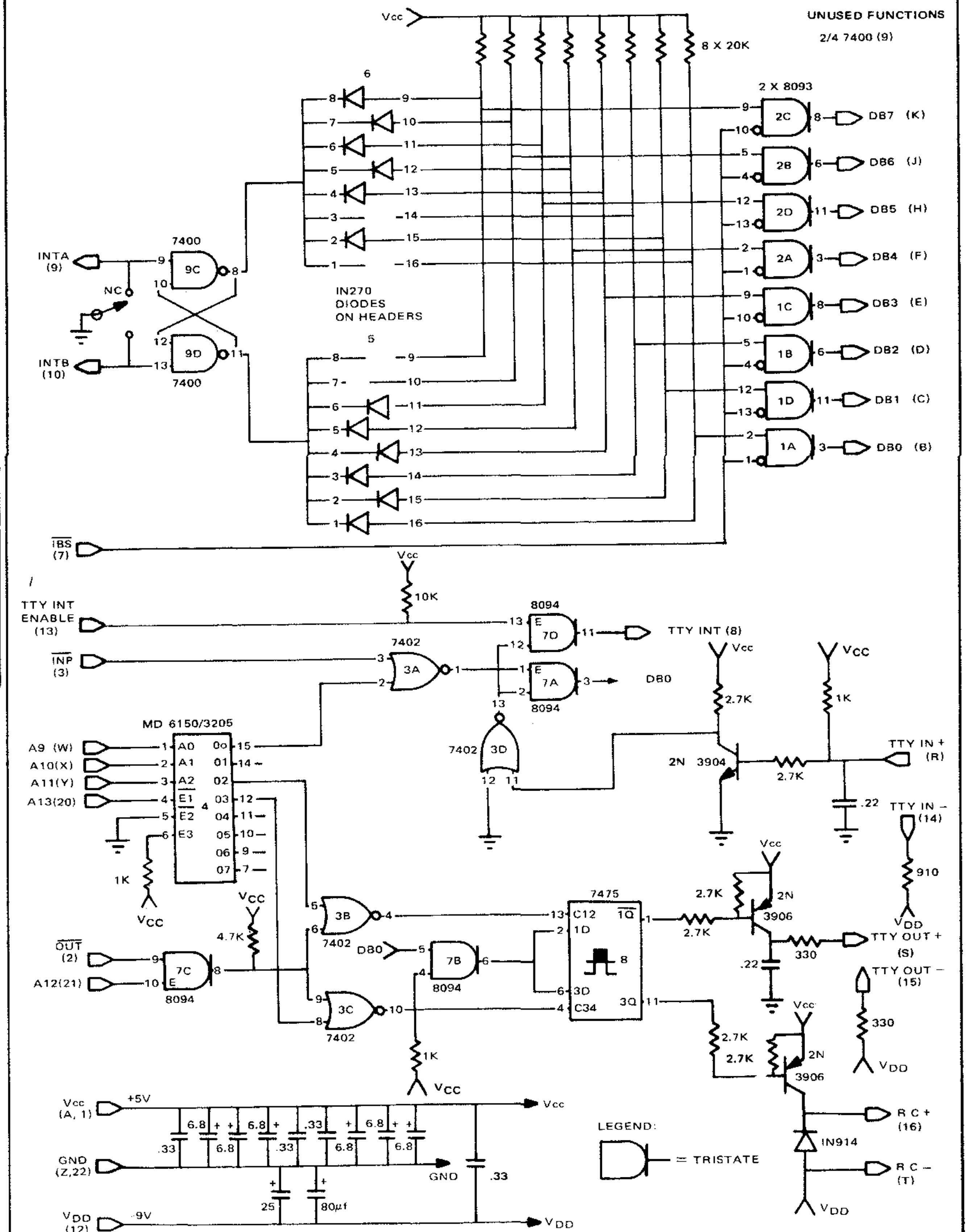
1. +5V
2. $\overline{\text{WRITE}}$
3. $\overline{\text{MRE}}$
4. $\overline{\text{OUT}}$
5. $\overline{\text{INP}}$
6. $\overline{\text{FS}}$
7. $\overline{\text{IBS}}$
8. TTY INT
9. INTA
10. INTB
11.
12. −9V
13.
14. READY
15.
16.
17.
18.
19.
20.
21.
22. OV

(BOTTOM)

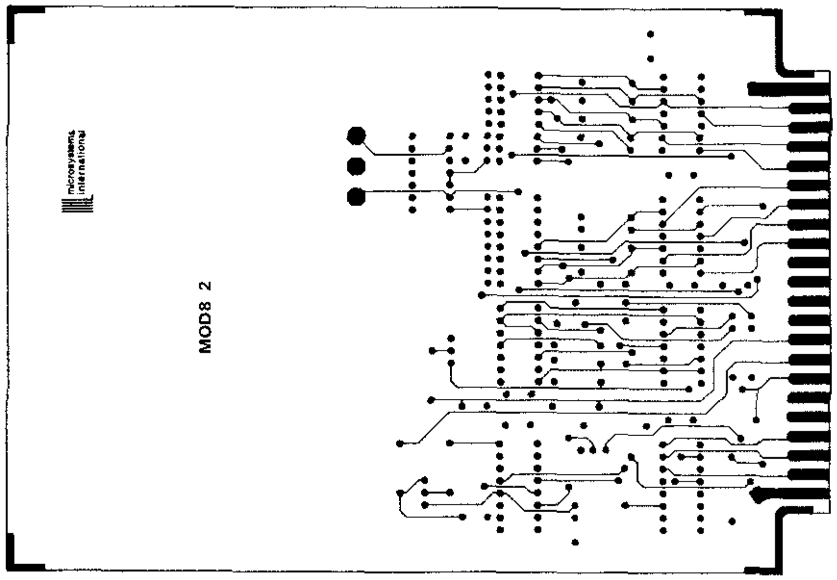


(COMPONENTS LAYOUT)

MOD8–1 PRINTED CIRCUIT CARD

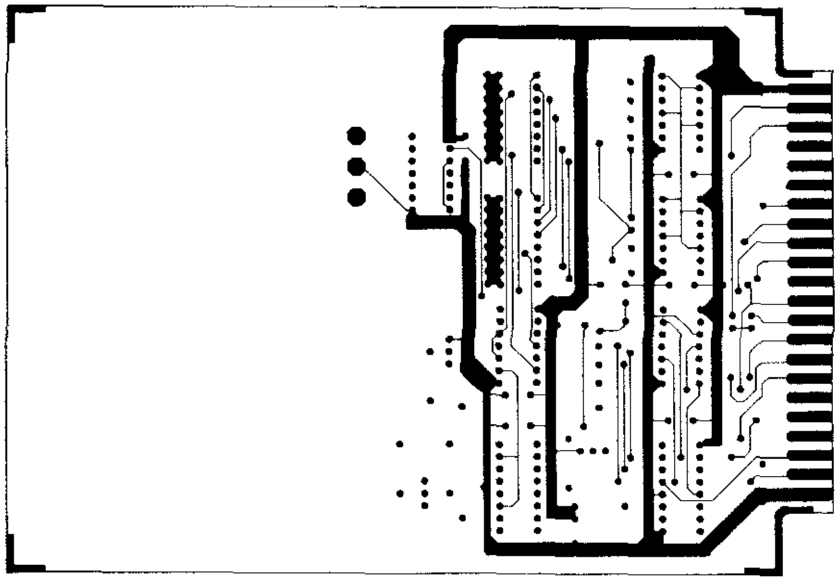


MOD8–2 (RESTART TTY I/O) CIRCUIT SCHEMATIC



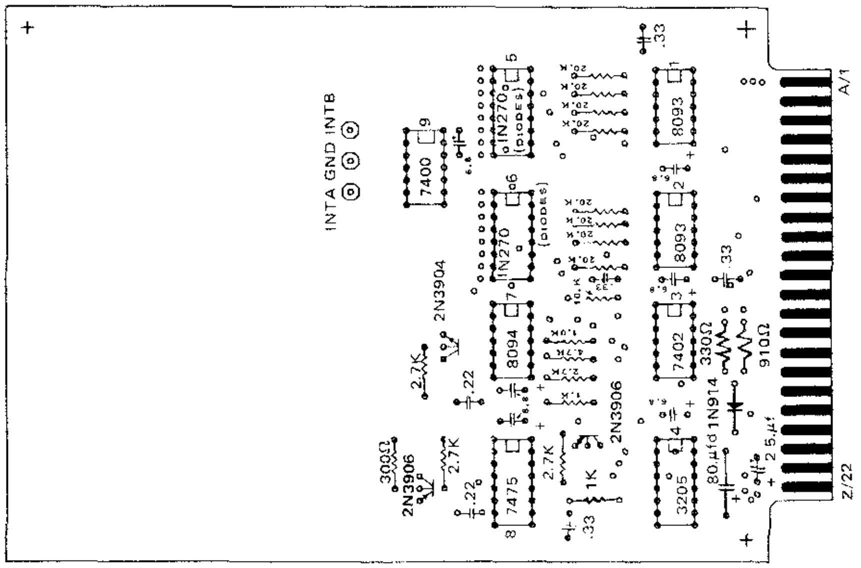
A. +5V
B. DBO
C. DB1
D. DB2
E. BD3
F. DB4
H. DB5
J. DB6
K. DB7
L.
M.
N.
P.
R. TTY IN(+)
S. TTY OUT(+)
T. RC(–)
U.
V.
W. A9
X. A10
Y. A11
Z. OV

(TOP)



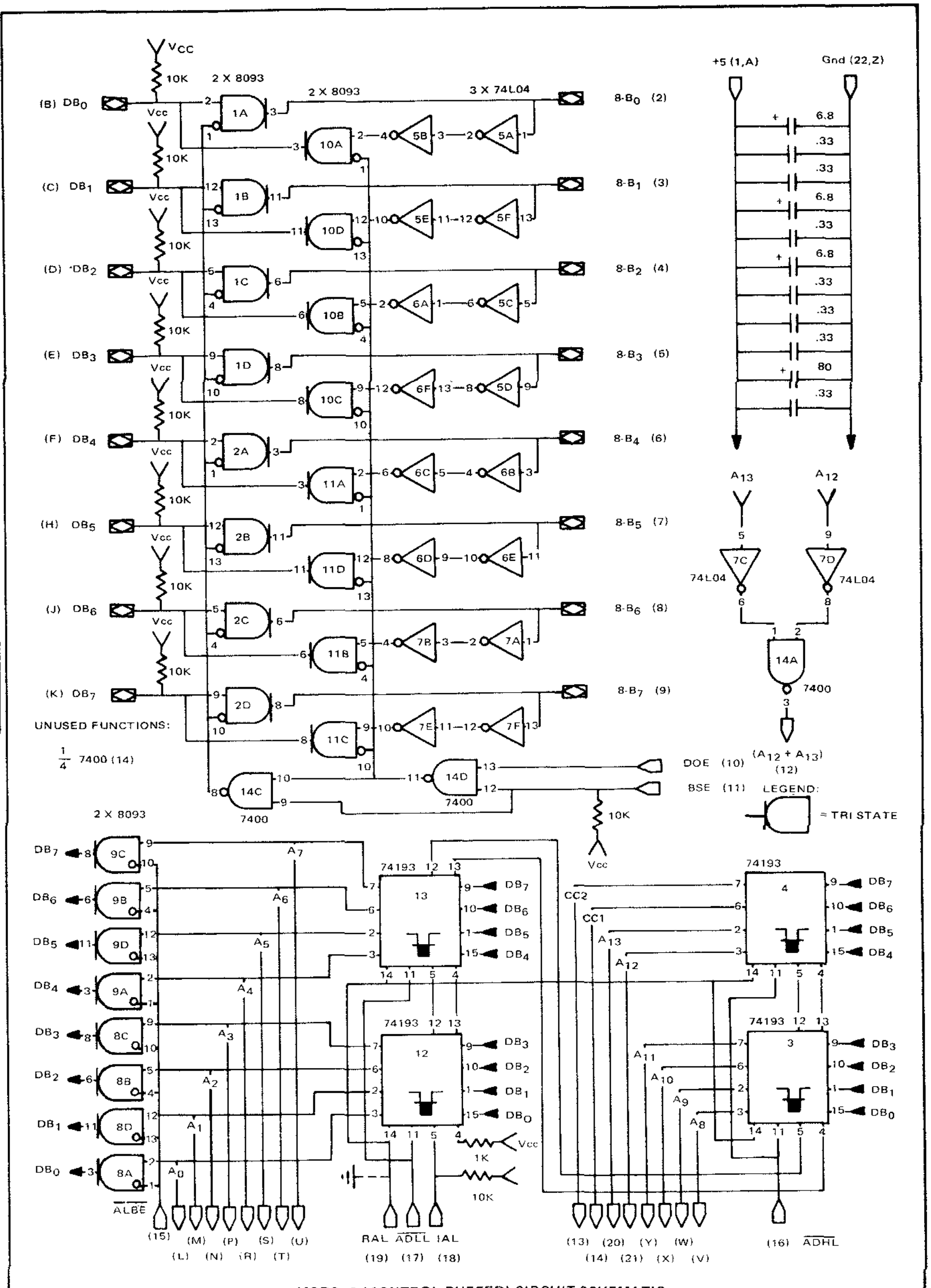
1. $\overline{+5V}$
2. $\overline{\text{OUT}}$
3. $\overline{\text{INP}}$
4.
5.
6.
7. $\overline{\text{IBS}}$
8. TTY INT
9. INTA
10. INTB
11.
12. –9V
13. TTY INT ENABLE
14. TTY IN(–)
15. TTY OUT(–)
16. RC(+)
17.
18.
19.
20. A13
21. A12
22. OV

(BOTTOM)

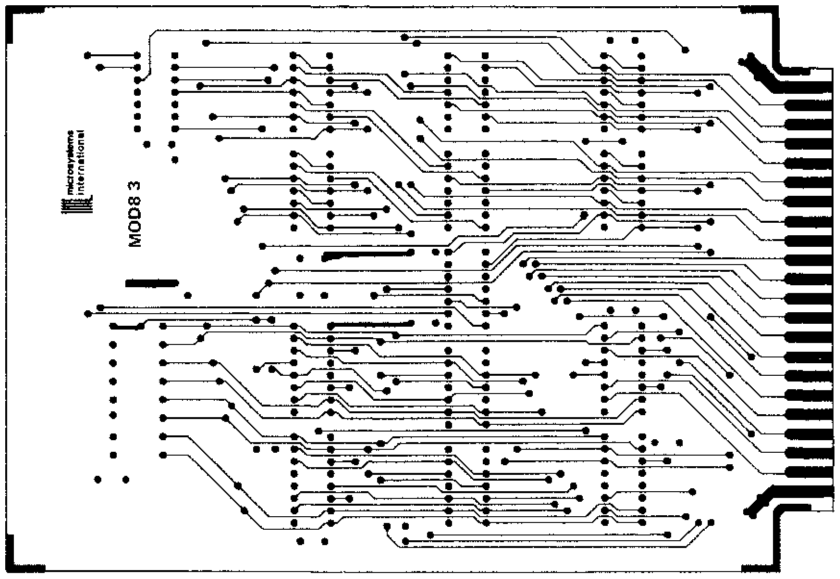


(COMPONENTS LAYOUT)

MOD8–2 PRINTED CIRCUIT CARD

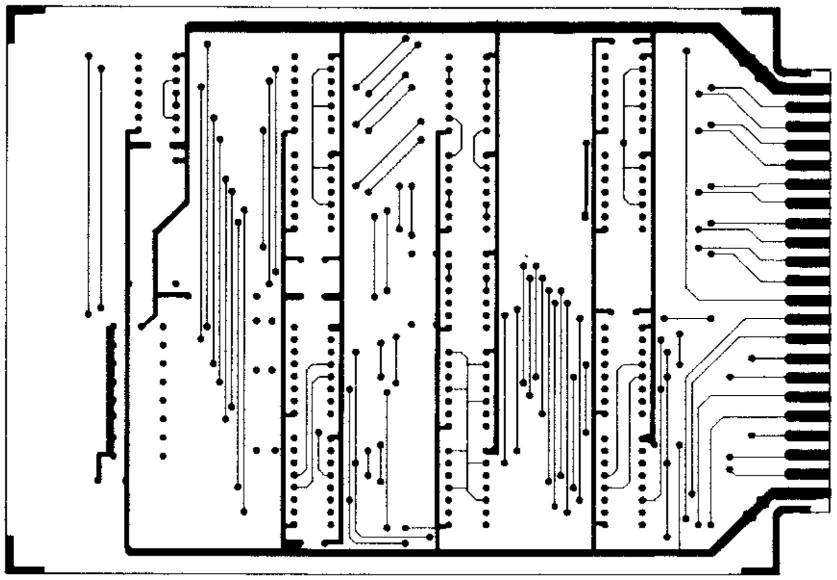


MOD8–3 (CONTROL BUFFER) CIRCUIT SCHEMATIC



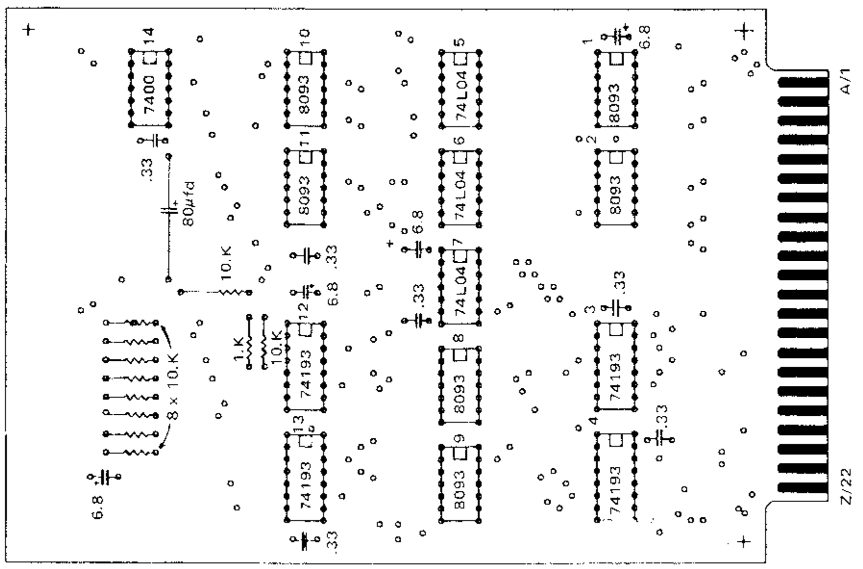
A. +5V
B. DB0
C. DB1
D. DB2
E. DB3
F. DB4
H. DB5
J. DB6
K. DB7
L. A0
M. A1
N. A2
P. A3
R. A4
S. A5
T. A6
U. A7
V. A8
W. A9
X. A10
Y. A11
Z. OV

(TOP)



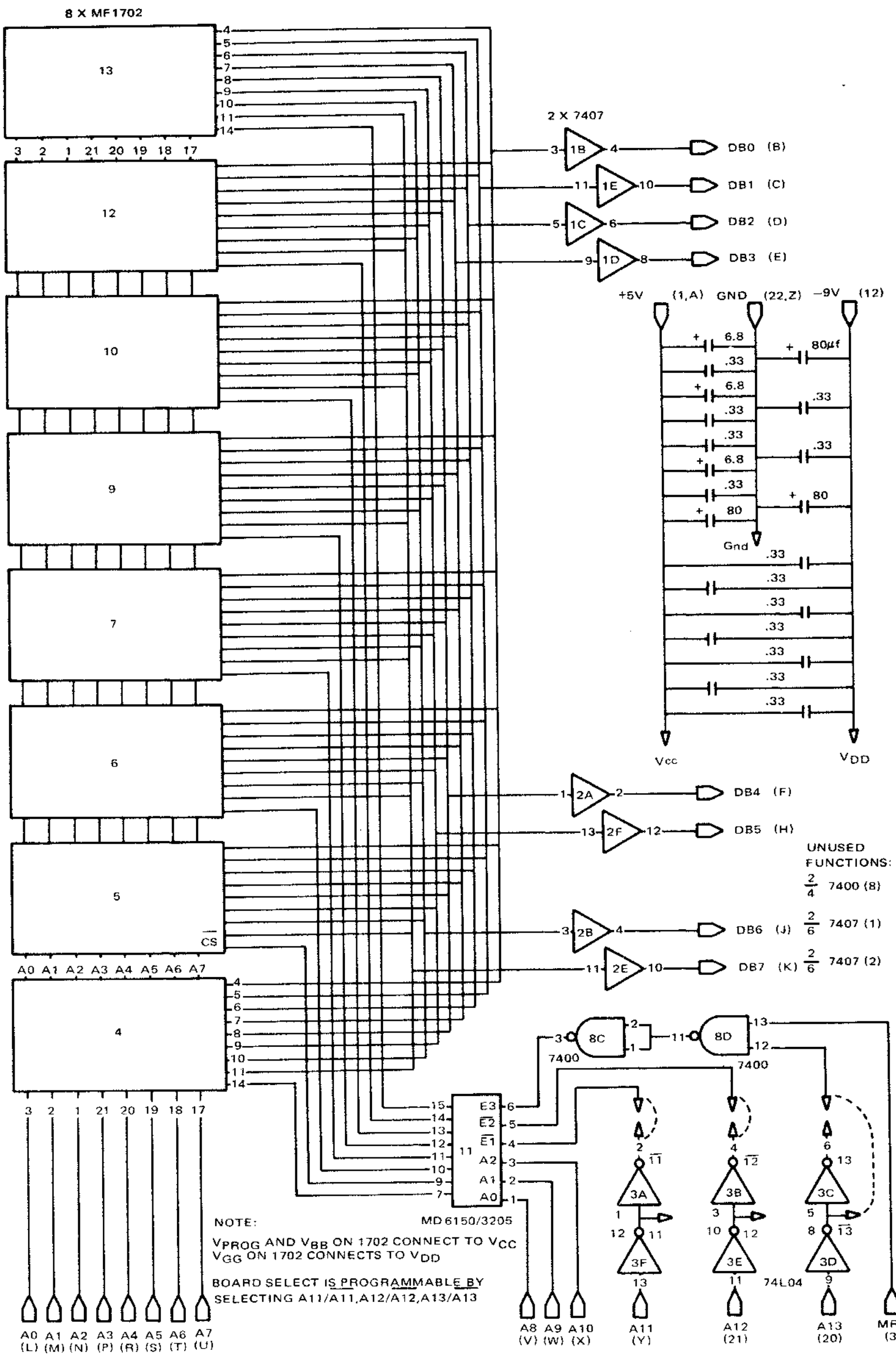
1. +5V
2. 8 B0
3. 8 B1
4. 8 B2
5. 8 B3
6. 8 B4
7. 8 B5
8. 8 B6
9. 8 B7
10. DOE
11. BSE
12. A12+A13
13. CC2
14. $\overline{\text{CC1}}$
15. $\overline{\text{ALBE}}$
16. $\overline{\text{ADHL}}$
17. ADLL
18. IAL
19. RAL
20. A13
21. A12
22. OV

(BOTTOM)



(COMPONENTS LAYOUT)

MOD8–3 PRINTED CIRCUIT CARD

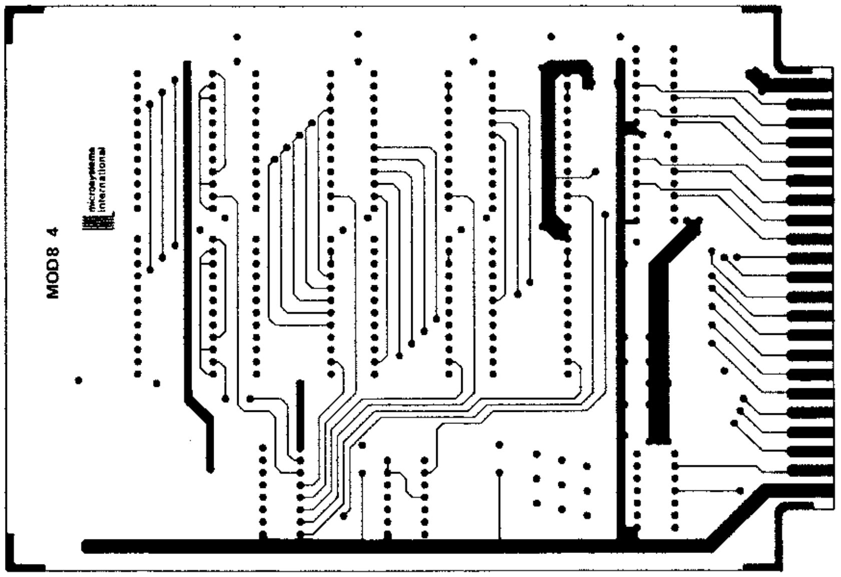


NOTE:

V_{PROG} AND V_{BB} ON 1702 CONNECT TO V_{CC}
V_{GG} ON 1702 CONNECTS TO V_{DD}

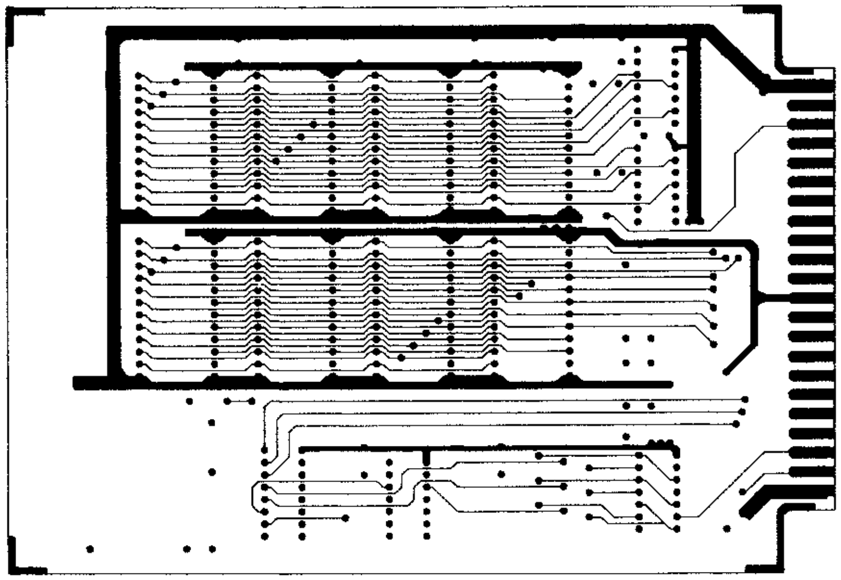
BOARD SELECT IS PROGRAMMABLE BY SELECTING A11/$\overline{A11}$,A12/$\overline{A12}$,A13/$\overline{A13}$

MOD8–4 (ROM) CIRCUIT SCHEMATIC



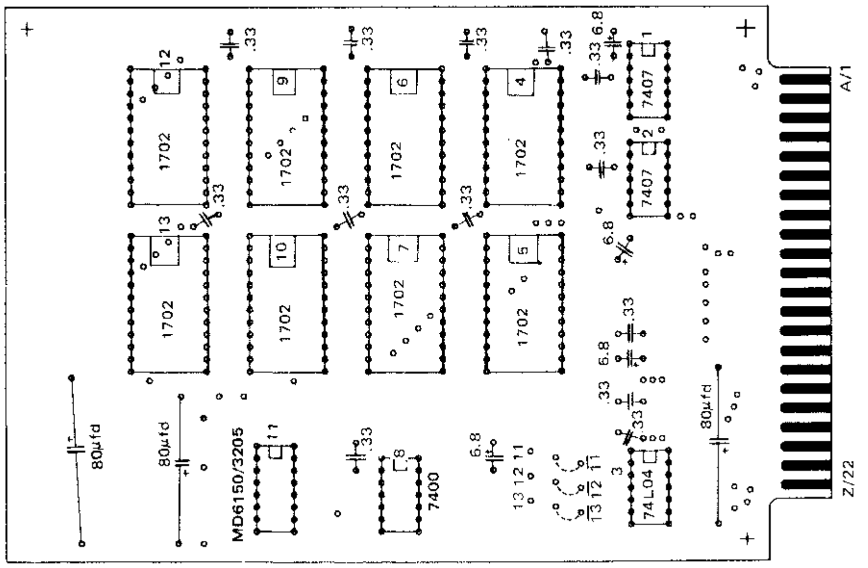
A. +5V
B. DB0
C. DB1
D. DB2
E. DB3
F. DB4
H. DB5
J. DB6
K. DB7
L. A0
M. A1
N. A2
P. A3
R. A4
S. A5
T. A6
U. A7
V. A8
W. A9
X. A10
Y. A11
Z. OV

(TOP)



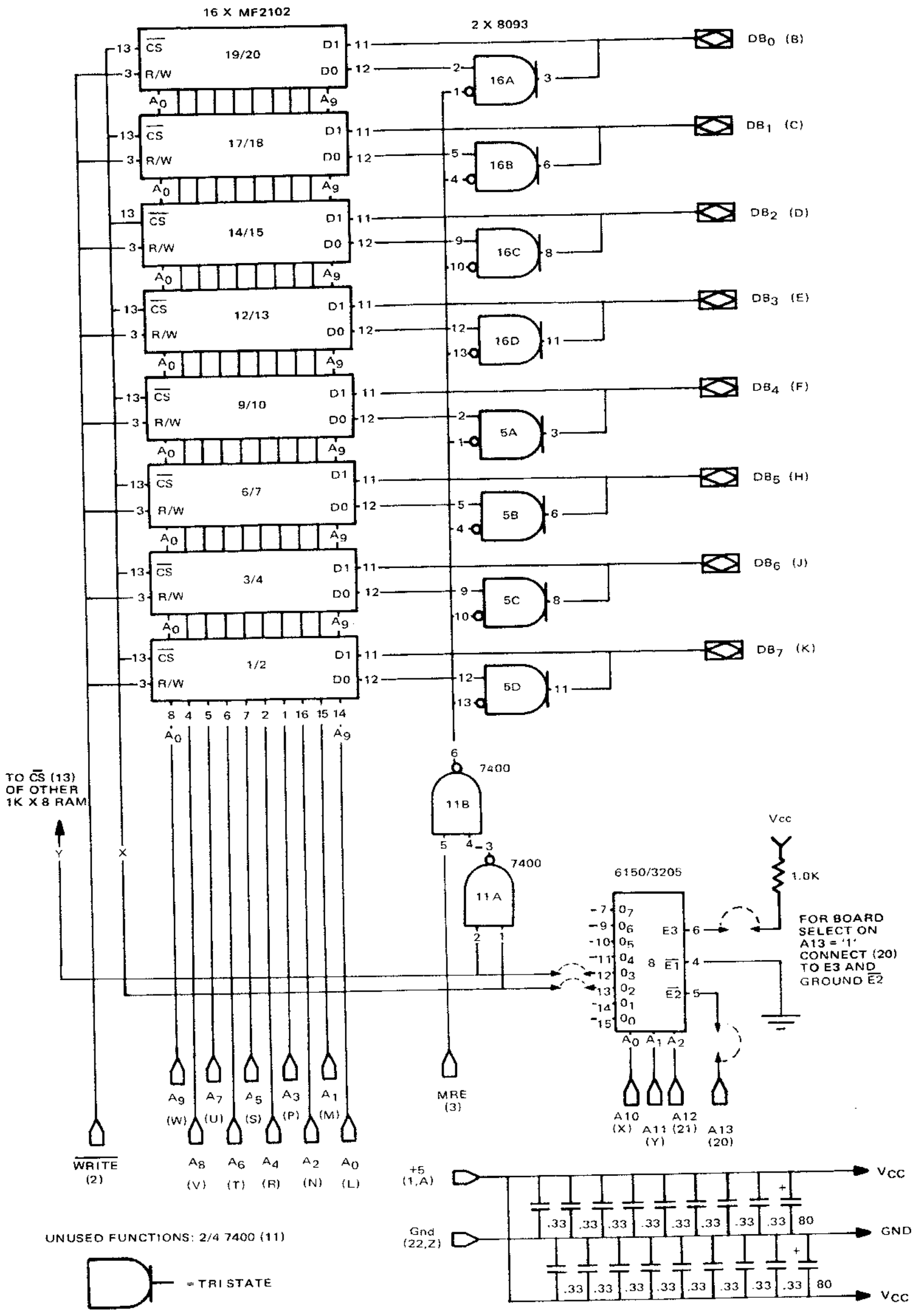
1. +5V
2.
3. MRE
4.
5.
6.
7.
8.
9.
10.
11.
12. –9V
13.
14.
15.
16.
17.
18.
19.
20. A13
21. A12
22. OV

(BOTTOM)

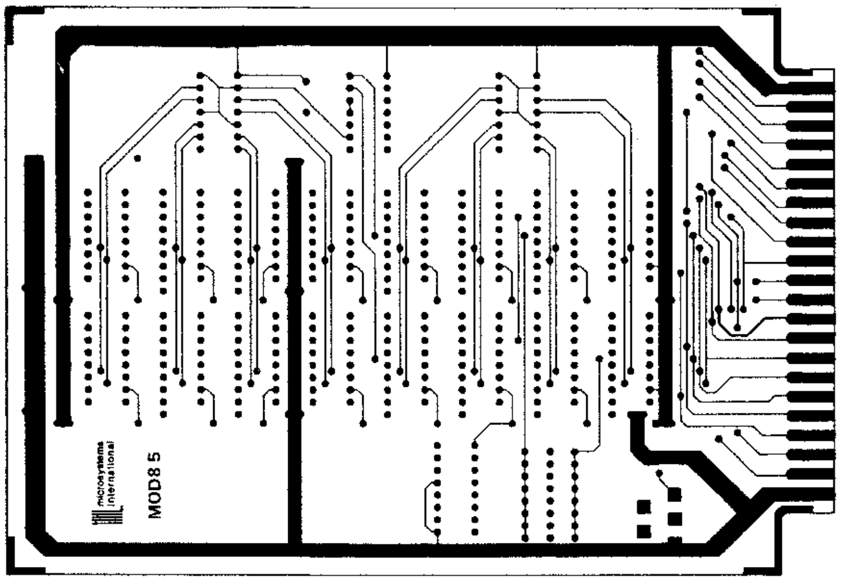


(COMPONENTS LAYOUT)

MOD8–4 PRINTED CIRCUIT CARD

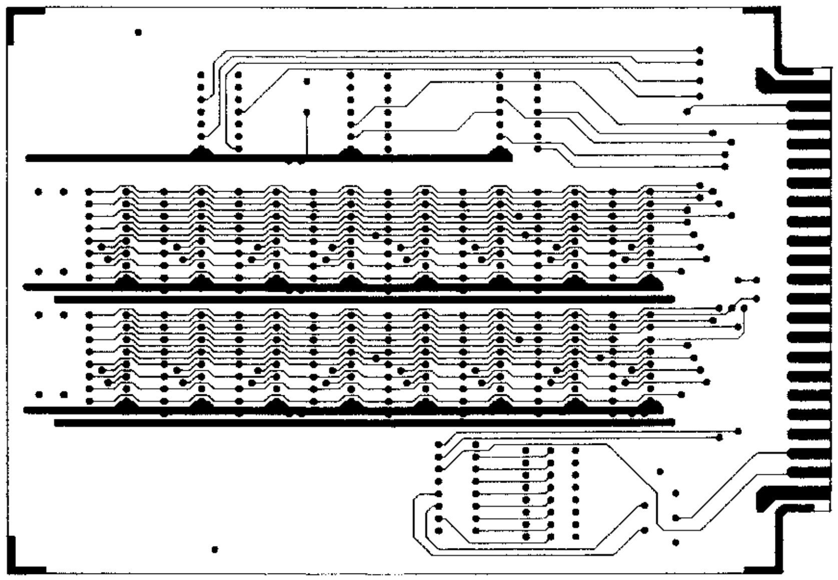


MOD8–5 (RAM) CIRCUIT SCHEMATIC



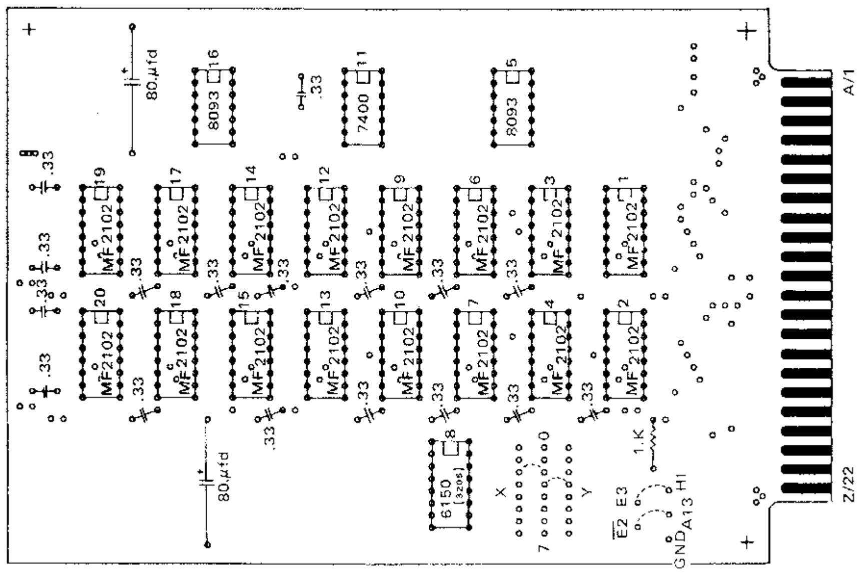
A. +5V
B. DB0
C. DB1
D. DB2
E. DB3
F. DB4
H. DB5
J. DB6
K. DB7
L. A0
M. A1
N. A2
P. A3
R. A4
S. A5
T. A6
U. A7
V. A8
W. A9
X. A10
Y. A11
Z. 0V

(TOP)



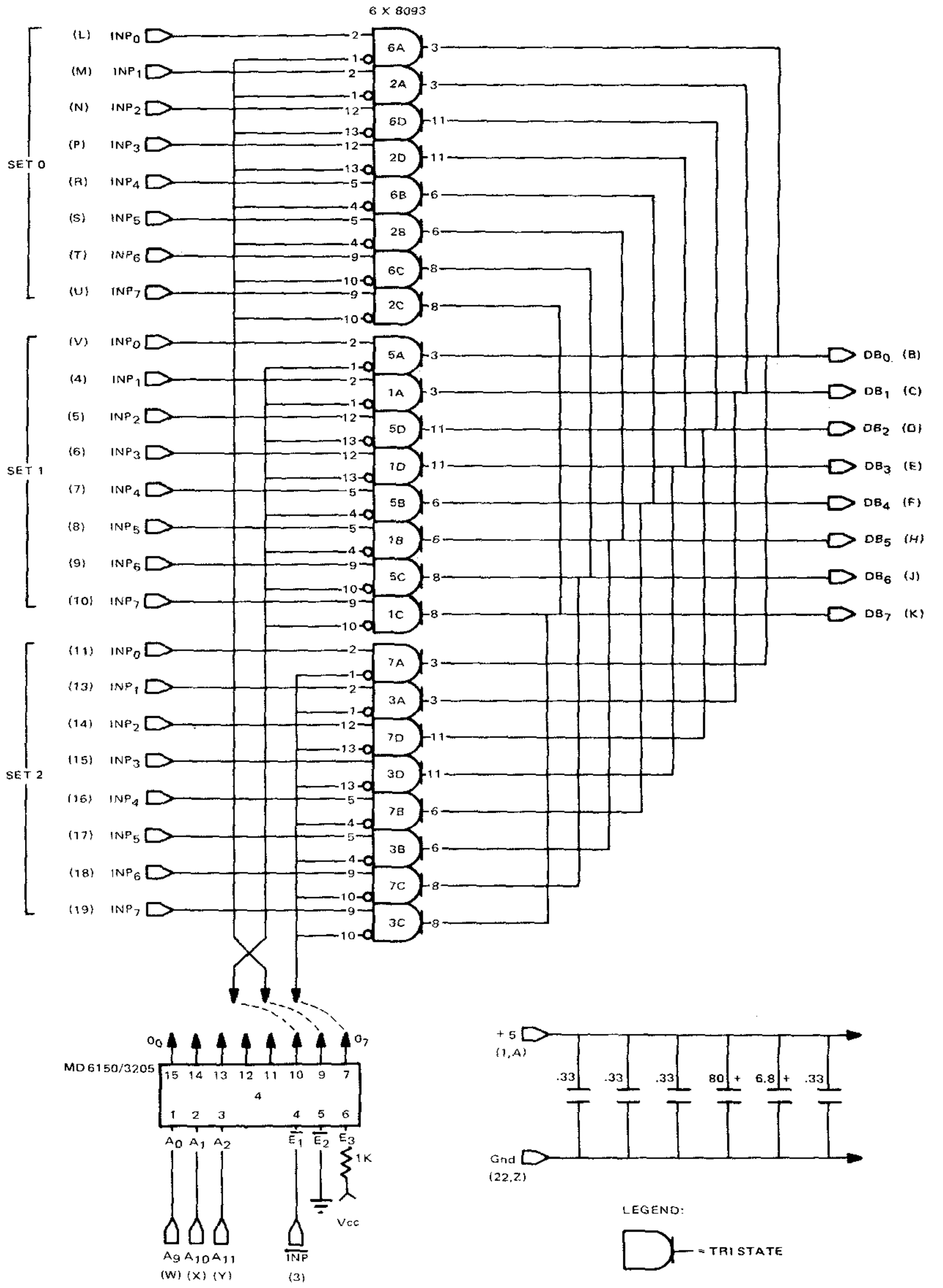
1. +5V
2. $\overline{\text{WRITE}}$
3. MRE
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.
16.
17.
18.
19.
20. A13
21. A12
22. 0V

(BOTTOM)

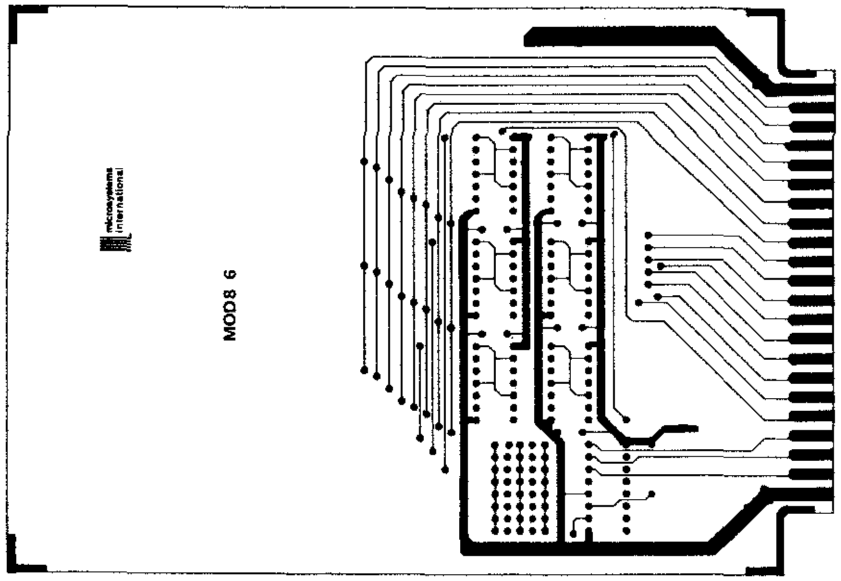


(COMPONENTS LAYOUT)

MOD8–5 PRINTED CIRCUIT CARD

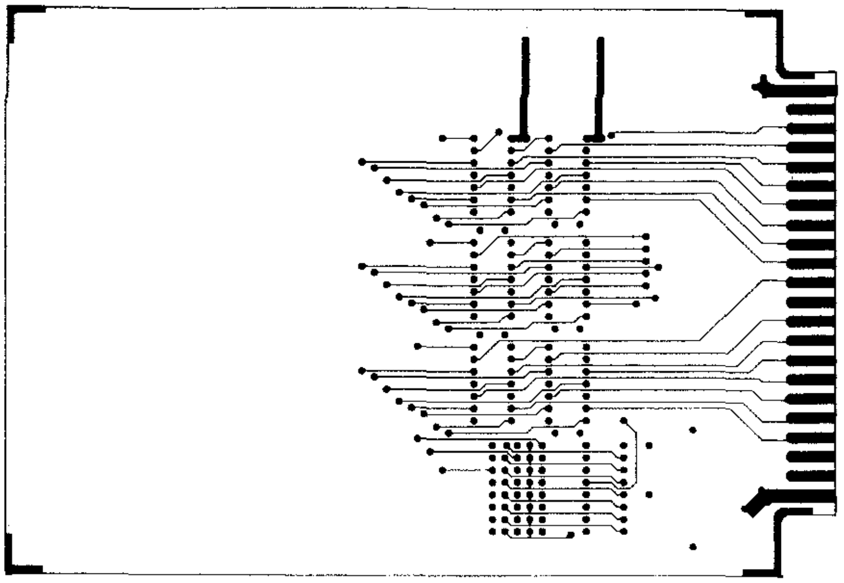


MOD8–6 (INPUT) CIRCUIT SCHEMATIC



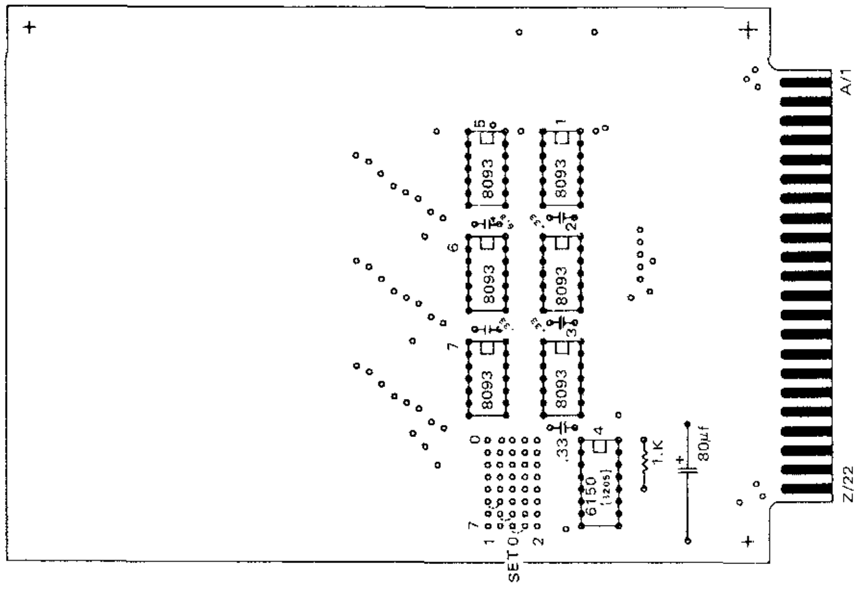
A. +5V
B. DB0
C. DB1
D. DB2
E. DB3
F. DB4
H. DB5
J. DB6
K. DB7
L. INP0 ┐
M. INP1
N. INP2
P. INP3
R. INP4 Set 0
S. INP5
T. INP6
U. INP7 ┘
V. INP0 — Set 1
W. A9
X. A10
Y. A11
Z. 0V

(TOP)



1. +5V
2.
3. $\overline{\text{INP}}$
4. INP1 ┐
5. INP2
6. INP3
7. INP4 Set 1
8. INP5
9. INP6
10. INP7 ┘
11. INP0 ┐
12.
13. INP1
14. INP2
15. INP3
16. INP4 Set 2
17. INP5
18. INP6
19. INP7 ┘
20. A13
21. A12
22. 0V

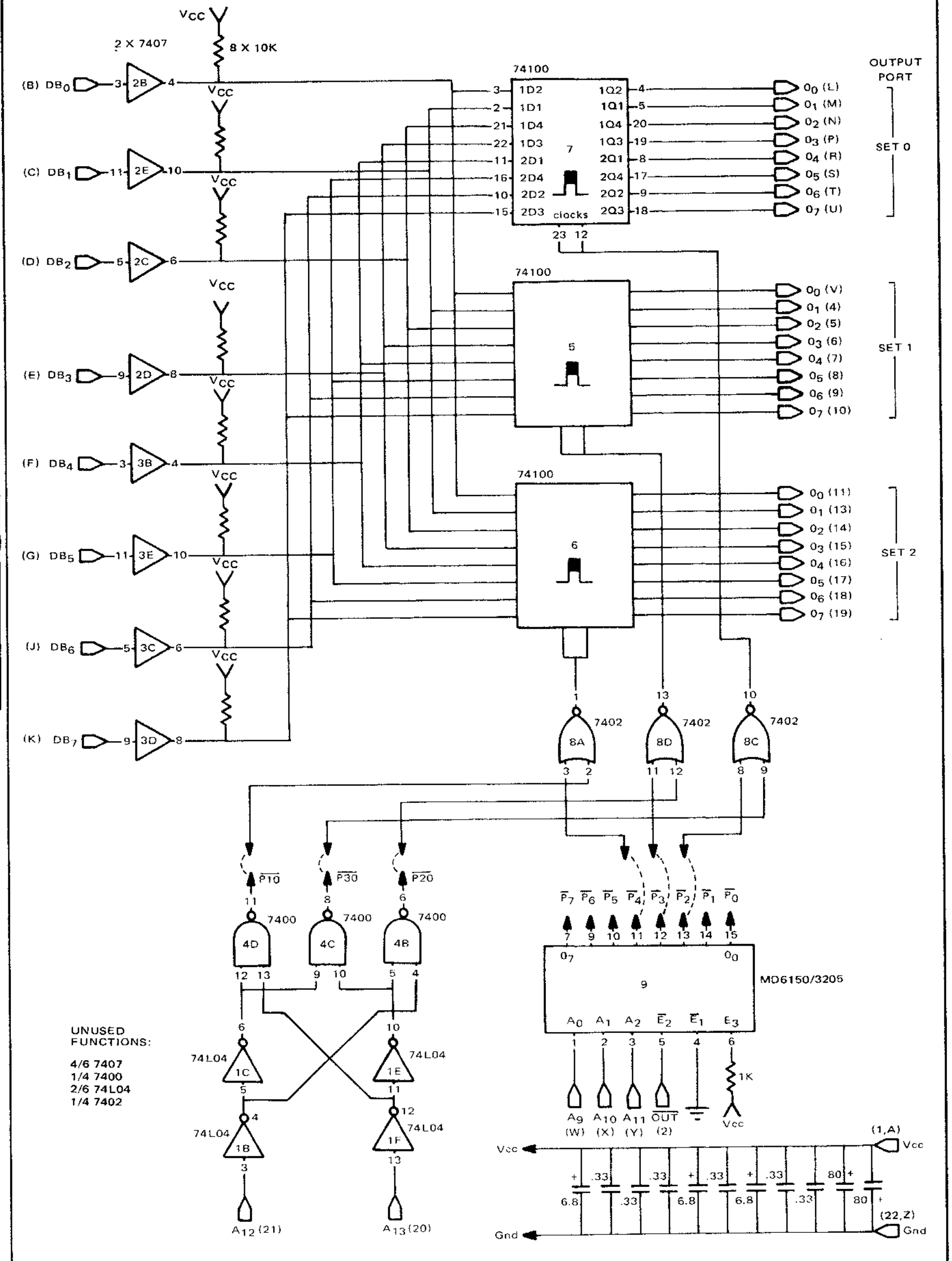
(BOTTOM)



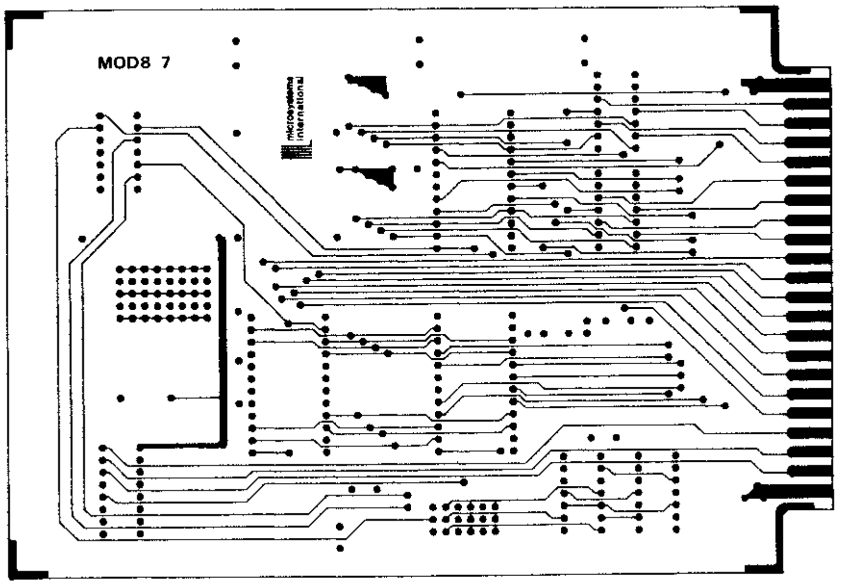
(COMPONENTS LAYOUT)

MODULAR MICROCOMPUTER

MOD8–6 PRINTED CIRCUIT CARD



MOD8–7 (OUTPUT) CIRCUIT SCHEMATIC

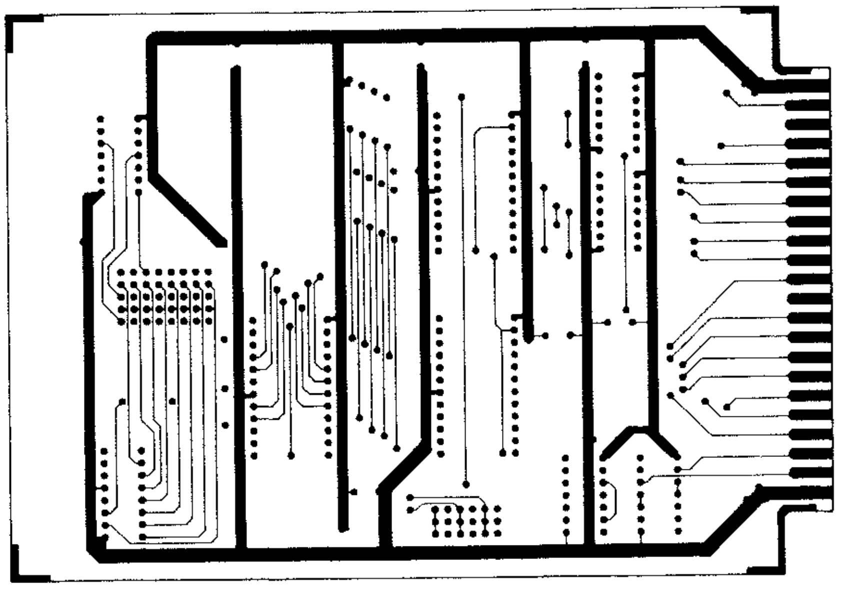


A. +5V
B. DB0
C. DB1
D. DB2
E. DB3
F. DB4
H. DB5
J. DB6
K. DB7
L. 00
M. 01
N. 02
P. 03
R. 04
S. 05
T. 06
U. 07
V. 00
W. A9
X. A10
Y. A11
Z. OV

Set 0

Set 1

(TOP)

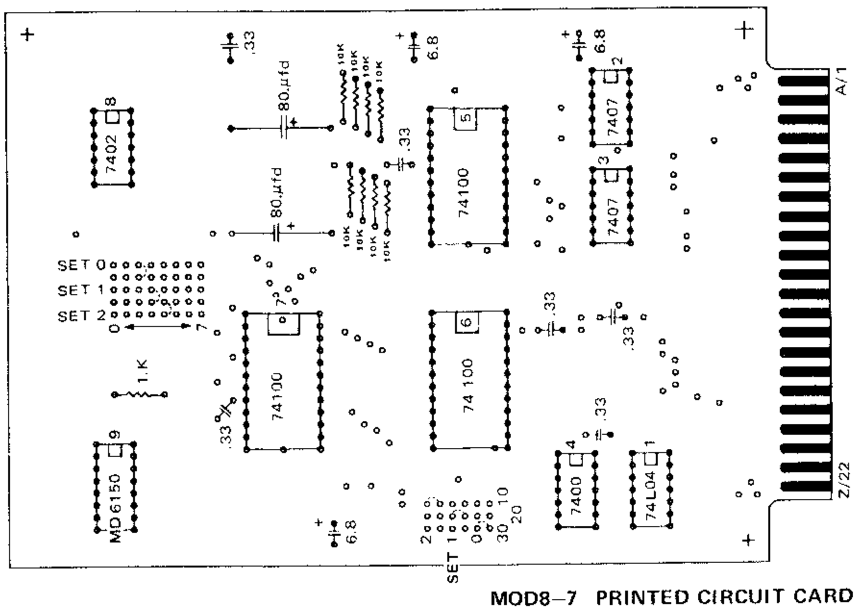


1. +5V
2. $\overline{\text{OUT}}$
3.
4. 01
5. 02
6. 03
7. 04
8. 05
9. 06
10. 07
11. 00
12.
13. 01
14. 02
15. 03
16. 04
17. 05
18. 06
19. 07
20. A13
21. A12
22. OV

Set 1

Set 2

BOTTOM)



(COMPONENTS LAYOUT)

MOD8—7 PRINTED CIRCUIT CARD

MICRO SYSTEMS
a Northern Electric company

MF8008 APPLICATIONS MANUAL

1.0	**Monitor 8 User's Guide**	C–2	1.11	LBF(Load BNPF Format)	C–3
1.1	System Start-up(MOD8)	C–2	1.12	DBF(Dump BNPF Format)	C–3
1.2	System Start-up(SIM8-01)	C–2	1.13	EDT(Enter Edit Mode)	C–3
1.3	Addressing	C–2	1.14	XQT(Initiate Program Execution)	C–3
1.4	Monitor 8 Command Summary	C–2	1.15	CYP(Copy Routine)	C–4
1.5	LOC(Set Current Location Pointer)	C–2	1.16	TRN(Translate Routine)	C–4
1.6	DLP(Display Current Location Pointer)	C–2	1.17	SBP(Set Break-point)	C–4
1.7	Symbolic Program Input	C–2	1.18	CBP(Clear Break-point)	C–4
1.8	DPS(Dump Symbolic)	C–3	1.19	PRG(Program pROM)	C–4
1.9	LDL(Load Octal)	C–3	1.20	Control A	C–4
1.10	DPO(Dump Octal)	C–3	1.21	Rubout	C–4

SOFTWARE USER'S GUIDE

MICROSYSTEMS INTERNATIONAL LIMITED, BOX 3529 STATION C, OTTAWA, CANADA, K1Y 4J1

SECTION C
SOFWARE USER'S GUIDE

1.0 MONITOR 8 USERS GUIDE

The monitor 8 software allows symbolic loading and dumping of MF8008 programs, and also offers utility editing and manipulation facilities.

1.1 SYSTEM START-UP (MOD 8 HARDWARE CONFIGURATION)

1.1.1 Ensure power off to programmer (if one is included), TTY set to local.

1.1.2 Apply CPU power.

1.1.3 Push Reset button.

1.1.4 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.

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

1.2.1 Ensure power off to programmer, TTY set to local.

1.2.2 Apply CPU power with reset switches set to 005.

1.2.3 Push reset button.

1.2.4 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, (e.g., Reset button pushed with switches set to 005 ------TTY response.)

1.2.5 Set reset switches to 300.

1.3 ADDRESSING

The memory in the MF8008 system is organized into banks. Each bank is 0 to 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, value 00-77 possible

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 SIMO8 and MOD8 systems.

N_2 N_1 N_0 = Byte location within bank, value 000 to 377 possible.

1.4 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)

1.5 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 (t_0). The user must then specify a six digit address (see adressing). 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.

1.6 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.

1.7 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 MF8008 as a halt command. The following commands will be interpreted by the monitor as HALT command bit combinations.

Mnemonic	Resultant Octal	MF8008 Interpretation
HLT	000	HLT
INA	000	HLT
DCA	001	HLT
LMM	377	HLT

1.8 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. Then 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 ? ? ? .

1.9 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.

1.10 DPO (DUMP OCTAL)

The dump routine will list 8 three digit octal values per line. Each line is started by the current address followed by a /. The user must specify the starting and ending address. When a DPO is typed the machine will respond with a CRLF *. The first valid octal digit (0-7) typed will be interpreted as the beginning of the "initial address". (see Adressing). After N$_0$ has been entered the machine will respond with a space. Next the ending address must be typed. After both addresses are entered the machine does a CRLF and stops, allowing time to prepare the paper tape punch. Pushing any key will start the Dump. It will continue until it has typed the final location and then return to the controller.

If a dump of the pROM station is required the address is specified by a 1 or 3 in the modifier bits of the address (see Addressing).

1.11 LBF (LOAD BNPF FORMAT)

The BNPF load routine is similar to the octal load routine in its initiation. The initial and final addresses are entered and any key will initiate the load. A B signifies the start of a data field and F signifies the end. All enclosed characters must be either P's (1) or N's (0). If a format error occurs, the present memory location is displayed followed by a ? and control returns to the monitor.

1.12 DBF (DUMP BNPF FORMAT)

If a DBF command is run the machine responds with a CRLF and waits for a starting and ending address. These must have the same format as in an octal dump. After the final address is entered a CRLF is typed and the machine halts. Typing any character will start the dump. Each memory location is listed sequentially, five bytes to a line. The dump or BNPF dump routines as described in this manual will list any portion of memory, including the pROM programming station (if one is attached to the system).

1.13 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.

1.14 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.

1.15 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.

1.16 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.

1.17 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 L and H 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.

1.18 CBP (CLEAR BREAK-POINT)

The CBP command will restore the data at the present break-point location.

1.19 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.

To accomodate MF1602A/1702A devices it is necessary to change the programming duty cycle from 2% to 20%. After receiving the initial and final address to be programmed the programming routine will respond with CRLF "%". The user must then type an "A" or "N" to determine the timing loop, Typing A will give a program pulse duty cycle of 20% for MF1702A types and an "N" gives approximately 2% duty cycle for MF1702 devices. There is no check for validity of the constants entered and under no condition should standard devices be programmed with excessive duty cycles.

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 device is hit with a single program pulse and the data is again checked against the desired data. When the data is finally read as being valid, after B program pulses, the device is hit with an additional 4 x B program pulses.

It should be noted that attempting to program a standard device using 1602A/1702A timing (20% duty cycle) will destroy the device. Note also that the unprogrammed state for an A series device is all lows whereas for standard devices it is all highs.

1.20 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.

1.21 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.

MICRO SYSTEMS
a Northern Electric company

MF8008 APPLICATIONS MANUAL

1.0 Monitor 8 Software Listings D–2

SOFTWARE

SECTION D
SOFTWARE LISTING

1.0 MONITOR SOFTWARE LISTINGS

This section contains a complete listing of the MONITOR 8 software. In addition it 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 Limited at a nominal surcharge over the normal component price, to cover the cost of programming the ROMs.

```
                   RESET INDEX
RESET NØ.          FUNCTIØN
RST 000  CØLD START,GENERAL RESTART
RST 010  GØ TØ RØM 7 (FØR USER)
RST 020  ØUTPUT AN ASCII CHARACTER
RST 030  INPUT AN ASCII CHARACTER
RST 040  TEST FØR RUBØUT
RST 050  SEARCH FØR CHARACTER IN 'E' REGISTER
RST 060  BREAKPØINT EXECUTE
RST 070  TIMING LØØP


                   SUBRØUTINE INDEX
         (START ADDRESSES ØF MANY ØF THE RØUTINES USED HERE,
         WHICH MAY BE USABLE IN ØTHER SØFTWARE)

START ADDRESS      FUNCTIØN
000013   ØUTPUT CARRIAGE RETURN AND LINE FEED
000177   TEST FØR ØCTAL CHARACTER
000205   3 DIGIT ØCTAL INPUT (CØMPRESSED TØ 1 BYTE)
000253   3 DIGIT ØCTAL ØUTPUT (USED TØ DISPLAY 1 BYTE)
000311   ADDRESS INCREMENT (USES CLP-LØC 013377,013376)
000326   ADDRESS DECREMENT
000344   ADDRESS CØMPARE (CLP,CLP-1)
000362   CØMPARE AND INCREMENT (USED TØ TEST FØR END ØF RØUTINE)
001000   ØCTAL DUMP (DPØ)
001023   FETCH DATA FRØM LØCATIØN ADDRESSED BY CLP
001047   DISPLAY DATA AT CLP
001055   DISPLAY BLANK, CLP (ADDRESS)
001073   ØUTPUT CR/LF, CLP
001111   PUT DATA INTØ CLP
001120   ØCTAL INPUT (LDØ)
001200   INPUT AN ADDRESS (2 BYTES)
001236   ØCTAL EDITØR (EDT)
001336   INDIRECT JUMP
001353   CLEAR BREAKPØINT (CBP)
002000   PRØM PRØGRAMMING RØUTINE (PRG)
002110   SET UP CLP (LØC)
002115   DUMP IN BNPF FØRMAT (DBF)
002201   LØAD IN BNPF FØRMAT (LBF)
002257   BANK TØ BANK TRANSLATE (TRN)
002347   SET BREAKPØINT (SBP)
003000   CØNTRØLLER RØUTINE
003131   GENERAL ERRØR RØUTINE
003150   TABLE SEARCH
003244   BREAKPØINT EXECUTE
005063   REGISTER DECØDE
005313   PRINT 3 ASCII BYTES
000000/ 006  LAI 001    (RST 000)     CØLD START
000002/ 125  ØUT 012    IDLE TTY
000003/ 250  XRA
000004/ 127  ØUT 013    IDLE PTR
000005/ 104  JMP 003000 GØ TØ CØNTRØLLER
000010/ 104  JMP 007000 (RST 010)     USERS RØUTINE
000013/ 016  LBI 215    (CR)          CR/LF RØUTINE
000015/ 025  RST 020
000016/ 016  LBI 212    (LF)
*N
```

```
000020/ 026  LCI 375     (RST 020)      Ø/P ØNE CHARACTER
000022/ 036  LDI 177    SET UP TIMING
000024/ 075  RST 070    1ST BIT IS LØNGER
000025/ 104  JMP 000140 CØNTINUED ELSEWHERE
000030/ 006  LAI 001    (RST 030)      I/P CHARACTER
000032/ 127  ØUT 013    ENABLE PTR
000033/ 036  LDI 302    SET UP TIMING
000035/ 104  JMP 000075 CØNTINUED
000040/ 006  LAI 177    (RST 040)      RUBØUT TEST
000042/ 271  CPB
000043/ 013  RFZ        NØT RUBØUT SØ RETURN
000044/ 016  LBI 337    Ø/P ARRØW
000046/ 025  RST 020
000047/ 007  RET        FLAG SET TØ IGNØRE INPUT
000050/ 035  RST 030    (RST 050)      SEARCH FØR CHAR
000051/ 301  LAB        FETCH I/P      IN REG E
000052/ 274  CPE        CØMPARE
000053/ 053  RTZ        GØT CHAR
000054/ 104  JMP 000050 TRY NEXT ØNE
000057/ 377  HLT        UNUSED BYTE
000060/ 104  JMP 003244 (RST 060)      XQT BRKPT
000063/ 301  LAB                       I/P (CØNT)
000064/ 074  CPI 001    CNTRL A I/P
000066/ 013  RFZ        NØ- GØ AHEAD
000067/ 005  RST 000    YES- PANIC AND RESTART
000070/ 030  IND        (RST 070)      TIMING LØØP
000071/ 110  JFZ 000070 LØØPING
000074/ 007  RET        DØNE
000075/ 377  HLT        WAIT FØR I/P   I/P(CØNT)
000076/ 075  RST 070    TIME 1ST BIT
000077/ 250  XRA        CLEAR A REG
000100/ 127  ØUT 013    IDLE PTR FØR NØW
000101/ 125  ØUT 012    START Ø/P
000102/ 026  LCI 370    SET UP 1 BIT DELAY
000104/ 036  LDI 171
000106/ 075  RST 070    WAIT FØR IT
000107/ 101  INP 000    GET BIT
000110/ 054  XRI 377    CØMPLEMENT I/P
000112/ 125  ØUT 012    ECHØ TØ Ø/P
000113/ 032  RAR        RØTATE INTØ B
000114/ 301  LAB        WITH PREVIØUS
000115/ 032  RAR        BITS
000116/ 310  LBA
000117/ 020  INC        BUMP CØUNTER
000120/ 110  JFZ 000104 LØØP FØR MØRE BITS
000123/ 301  LAB        GØT 8 BITS NØW
000124/ 044  NDI 177    IGNØRE PARITY (MSB)
000126/ 310  LBA
000127/ 036  LDI 171    Ø/P STØP
000131/ 075  RST 070    AND Ø/P IDLE STATE
000132/ 006  LAI 001
000134/ 125  ØUT 012
000135/ 104  JMP 000063 TØ BE CØNTINUED
000140/ 020  INC                       Ø/P (CØNT)
000141/ 110  JFZ 000022 KEEP TIMING
000144/ 250  XRA        CLEAR A
000145/ 125  ØUT 012    START Ø/P
000146/ 026  LCI 370    SET UP TIMING
000150/ 036  LDI 171
000152/ 075  RST 070    WAIT FØR NEXT BIT
000153/ 301  LAB        FETCH BIT FRØM B
000154/ 125  ØUT 012    AND ØUTPUT BIT
000155/ 032  RAR        NØW SET UP THE NEXT
000156/ 310  LBA        BIT, STØRE IT IN B
000157/ 006  LAI 000
000161/ 032  RAR
000162/ 201  ADB
000163/ 310  LBA
000164/ 020  INC        BUMP CØUNT
000165/ 110  JFZ 000150 MØRE TØ Ø/P, SØ LØØP
000170/ 036  LDI 171    DØNE
000172/ 075  RST 070    Ø/P STØP AND IDLE BITS
000173/ 006  LAI 001
000175/ 125  ØUT 012
```

```
000176/ 007  RET         GØØDBYE
000177/ 035  RST 030     FETCH CHAR    TEST FØR ØCTAL
000200/ 044  NDI 370     MASK 3 BITS
000202/ 074  CPI 060     IS IT 06X? (ZF TELLS ALL)
000204/ 007  RET         GØ AWAY
000205/ 106  CAL 000177  GET DIGIT     ØCTAL I/P
000210/ 110  JFZ 000205  NØT ØCTAL,TRY AGAIN
000213/ 301  LAB         PUT DIGIT IN A
000214/ 012  RRC         RØTATE
000215/ 012  RRC         RØTATE
000216/ 370  LMA         STASH IT
000217/ 035  RST 030     FETCH DIGIT
000220/ 045  RST 040     TEST FØR RUBØUT
000221/ 150  JTZ 000205  TRY AGAIN
000224/ 307  LAM         FETCH LAST DIGIT
000225/ 044  NDI 300     MASK UNUSED BITS
000227/ 370  LMA         STØRE IT
000230/ 006  LAI 007     PUT IN 3 MØRE BITS
000232/ 241  NDB         RØTATE INTØ PØSITIØN
000233/ 002  RLC
000234/ 002  RLC
000235/ 002  RLC
000236/ 207  ADM         ADD IN THE ØLD DATA
000237/ 370  LMA         STØRE IT
000240/ 035  RST 030     FETCH DIGIT   ENTRY FØR
000241/ 045  RST 040     TEST          CØRRECTIØN DØNE
000242/ 150  JTZ 000217  RUBØUT
000245/ 006  LAI 007     MASK ALL BUT 3 BITS
000247/ 241  NDB
000250/ 207  ADM         ADD THE PREVIØUS BITS
000251/ 370  LMA         STASH DATA
000252/ 007  RET         DØNE
000253/ 016  LBI 240     (BLANK)       ØCTAL Ø/P
000255/ 025  RST 020                   3 DIGITS
000256/ 307  LAM         FETCH BYTE
000257/ 002  RLC         MØVE BITS 7 AND 8
000260/ 002  RLC         TØ PØS 1 AND 2
000261/ 044  NDI 003     MASK THE REST
000263/ 004  ADI 260     CØNVERT TØ ASCII
000265/ 310  LBA         SET UP FØR Ø/P
000266/ 025  RST 020     Ø/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 LØCATIØNS
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  Ø/P CLP CØNTENTS
001011/ 106  CAL 000362  INCR AND CØMPARE CLP
001014/ 041  DCE         INCR LINE CØUNT
001015/ 150  JTZ 001001  NEW LINE,PRINT ADR
001020/ 104  JMP 001006  SAME LINE, JUST LØØP
001023/ 066  LLI 377     SET TØ CLP    GET DATA (FRØM CLP)
001025/ 056  LHI 013                   EXTENDED ADDRESS
001027/ 307  LAM         MS IN 'A'     PUTS DATA INTØ 013370
001030/ 061  DCL
001031/ 367  LLM         LS IN L
001032/ 350  LHA         NØW PUT MS IN H
001033/ 002  RLC         RØTATE TØ TEST BIT 8
001034/ 307  LAM         FETCH LS
001035/ 003  RFC         RETURN IF NØT EXTENDED MEMØRY
001036/ 056  LHI 013     GET DATA FRØM I/P PØRT
001040/ 306  LAL         SET TEMP STØRE LØCATIØN
001041/ 066  LLI 370
001043/ 121  ØUT 010     Ø/P LS
001044/ 103  INP 001     GET DATA
001045/ 370  LMA         PUT INTØ MEMØRY
001046/ 007  RET         GØ!
001047/ 106  CAL 001023                GET DATA FRØM CLP
001052/ 104  JMP 000253                AND PRINT IT
001055/ 066  LLI 377     SET TØ CLP    Ø/ ' HHHLLL'
001057/ 056  LHI 013
001061/ 016  LBI 240     Ø/P BLANK
001063/ 025  RST 020
001064/ 106  CAL 000256  Ø/P MS BYTE
001067/ 061  DCL         Ø/P LS BYTE
001070/ 104  JMP 000256  AND RETURN TØ CALL WHEN DØNE
001073/ 066  LLI 377     SET TØ CLP    PRINT CR/LF HHHLLL
001075/ 056  LHI 013
001077/ 106  CAL 000013  Ø/P CR/LF
001102/ 106  CAL 001064  Ø/P ADR
001105/ 016  LBI 257     Ø/P SLASH
001107/ 025  RST 020
001110/ 007  RET         DØNE
001111/ 106  CAL 001023  SET TØ CLP    GET CLP PUT DATA THERE
001114/ 104  JMP 000205  FETCH DATA
001117/ 300  LAA         NØP NØT USED
001120/ 106  CAL 000013  Ø/P CR/LF     ØCTAL INPUT (LDØ)
001123/ 046  LEI 057     SEARCH FØR 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 FØR ANØTHER SLASH
001135/ 106  CAL 001111 NØ- PUT DATA AT CLP
001140/ 106  CAL 000362 CØMPARE AND INCR CLP
001143/ 104  JMP 001126 LØØP
001146/ 066  LLI 373    SET UP 'L'    CØPY (CPY)
001150/ 106  CAL 001205 INPUT NEW START ØF BLØCK
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 TØ NEW ADR
001164/ 372  LMC        STØRE DATA
001165/ 106  CAL 000362 INCR FRØM ADR
001170/ 066  LLI 372
001172/ 106  CAL 000313 INCR TØ ADR
001175/ 104  JMP 001153 LØØP
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  JTZ 003014 YES-THEN RETURN
001256/ 074  CPI 052    TEST FØR '*'
001260/ 066  LLI 377    SET L TØ CLP
001262/ 150  JTZ 002110 GØ TØ LØC RØUTINE
001265/ 074  CPI 100    TEST FØR '@'
001267/ 150  JTZ 003320 GØ TØ XQT
001272/ 074  CPI 136    TEST FØR 'UP ARRØW'
001274/ 150  JTZ 000326 THEN DECR CLP
001277/ 074  CPI 040    TEST FØR BLANK
001301/ 150  JTZ 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 LØCATIØN
001376/ 373  LMD
001377/ 007  RET         GO AWAY
002000/ 016  LBI 045     TYPE  %     PRØGRAMMER (PRG)
002002/ 025  RST 020
002003/ 035  RST 030     INPUT TIMING CØNSTANT
002004/ 002  RLC         GIVING A= 005 (1602A/1702A)
002005/ 002  RLC                N= 071 (1602/1702)
002006/ 340  LEA         SAVE AWAY FØR LATER
002007/ 106  CAL 001023 GET DATA AND SET H AND L
002012/ 306  LAL         GET PRØM ADDRESS
002013/ 121  ØUT 010     AND ØUTPUT TØ PRØGRAMMER
002014/ 103  INP 001     GET RØM DATA
002015/ 277  CPM         AND IF NØT EQUAL
002016/ 112  CFZ 002027 GØ PRØGRAM
002021/ 106  CAL 000362 INCREMENT ADDRESS PØINTER
002024/ 104  JMP 002007 AND GØ BACK TØ TEST NEXT BYTE
002027/ 016  LBI 001     START WITH 1 TRY   PRØG. SEQUENCE
002031/ 106  CAL 002051 GØ PRØGRAM IT
002034/ 301  LAB         DATA IS NØW READ AS CØRRECT
002035/ 002  RLC         SØ ØVERKILL 4 TIMES
002036/ 002  RLC
002037/ 310  LBA         B STILL CØUNTS TRIES
002040/ 106  CAL 002051 GØ ØVERKILL
002043/ 011  DCB         UNTIL B
002044/ 110  JFZ 002040 EQUALS ZERØ
002047/ 025  RST 020     AND ØUTPUT  NULL CHARACTER
002050/ 007  RET         INDICATING END ØF BYTE
002051/ 307  LAM         GET DATA
002052/ 054  XRI 377     CØMPLEMENT IT
002054/ 123  ØUT 011     AND PUT IT IN THE BUFFER
002055/ 006  LAI 004     SET UP FØR THE 15MSEC
002057/ 127  ØUT 013     PULSE GENERATØR
002060/ 250  XRA         HIT IT ØNCE
002061/ 127  ØUT 013     GIVING ØNE  3.0 MSEC PULSE
002062/ 324  LCE         E STILL HAS TIMING CØNSTANT
002063/ 036  LDI 325     INNER TIME-ØUT
002065/ 075  RST 070     LØØP
002066/ 021  DCC         TØTAL LØØP TIME IS
002067/ 110  JFZ 002063 15 MSEC ØR 160 MSEC
002072/ 103  INP 001     HØWS THE DATA LØØK?
002073/ 277  CPM
002074/ 053  RTZ         IF THE SAME RETURN
002075/ 010  INB         B CØUNTS UNSUCCESSFUL TRIES
002076/ 110  JFZ 002051 IF NØT 377 TRIES TRY AGAIN
002101/ 106  CAL 001055 PRINT CURRENT LØCATIØN PØINTER
002104/ 016  LBI 277     AND THEN A  ?
002106/ 025  RST 020     AND GIVE UP BY DØING
002107/ 005  RST 000     A CØMPLETE RSTART
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 LØCATIØN
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
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 SØ CHANGE TØ 'P'
002150/ 025  RST 020     Ø/P WHATEVER IT IS
002151/ 041  DCE         ØNE MØRE BIT DØNE
002152/ 110  JFZ 002136 LØØP IF MØRE
```

```
002155/ 016  LBI 306    DØNE 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        ØNE MØRE BYTE Ø/P
002166/ 011  DCB
002167/ 371  LMB
002170/ 110  JFZ 002122 MØRE ØN THIS LINE
002173/ 106  CAL 000013 NEW LINE (CR/LF)
002176/ 104  JMP 002116 KEEP GØING
002201/ 300  LAA        NØP            BNPF LØAD (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 SØME RAM
002214/ 035  RST 030    FETCH CHARACTER
002215/ 301  LAB        INTØ 'A'
002216/ 074  CPI 116    IS IT 'N'
002220/ 150  JTZ 002232 YES- STASH IT
002223/ 054  XRI 377    NØ -CØMPLEMENT
002225/ 074  CPI 257    IS IT 'P'
002227/ 110  JFZ 002101 NØ-ERRØR
002232/ 032  RAR        YES- PUT BIT IN CARRY
002233/ 307  LAM        GET PREVIØUS 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,LØØP
002242/ 035  RST 030    YES-ØNE MØRE CHECK
002243/ 301  LAB
002244/ 074  CPI 106    LAST CHARACTER MUST BE AN 'F'
002246/ 110  JFZ 002101 NØ-PANIC
002251/ 106  CAL 000362 YES-INCR CLP,CHECK IF DØNE
002254/ 104  JMP 002202 LØØP 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 ARRØW
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 LØØP FØR MØRE
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) ØF CØURSE
002343/ 374  LME        REPLACE MS BYTE
002344/ 104  JMP 002273 NØW- WE ARE REALLY DØNE
002347/ 106  CAL 001353 CLEAR ØLD      SET BREAKPØINT (SBP)
002352/ 106  CAL 001200 FETCH ADR ØF NEW BRKPT
002355/ 106  CAL 001023 SET UP H,L TØ CLP
002360/ 326  LCL        SAVE H,L
002361/ 335  LDH
002362/ 076  LMI 065    SET RST 060 INTØ LØCATIØN
002364/ 056  LHI 013    SAVE THE ØLD 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 GØ HØME TØ MØMMY
003000/ 106  CAL 000013 CR/LF         MØNITØR AND CØNTRØLLER
003003/ 046  LEI 010    SET UP LØØP
003005/ 016  LBI 255    CHARACTER IS '-'
003007/ 025  RST 020    GØ PRINT '--------'
003010/ 041  DCE        CØUNT
003011/ 110  JFZ 003007 LØØP
003014/ 146  CAL 000013 NEW LINE
003017/ 146  CAL 003067 FETCH INPUT
003022/ 100  JFC 003017 LØØP IF NØT 'A'-'Z'
003025/ 146  CAL 003100 NØW GET TWØ MØRE CHARACTERS
003030/ 146  CAL 003150 FIND IT IN THE TABLE
003033/ 150  JTZ 006000 NØT FØUND!! GØ TØ LDS
003036/ 066  LLI 373                  EXEC RØUTINE
003040/ 056  LHI 013
003042/ 370  LMA        STØRE ADDRESS (MS)
003043/ 061  DCL        FRØM 5 BYTE TABLE
003044/ 371  LMB        STØRE ADDRESS (LS)
003045/ 044  NDI 200    FRØM 5 BYTE TABLE
003047/ 112  CFZ 003137 GØ 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 SØ GØ
003064/ 144  JMP 003014 AND CØNTINUE WHEN DØNE
003067/ 035  RST 030    GET CHAR       CHAR TEST
003070/ 006  LAI 100    TEST FØR  LT 'A'
003072/ 271  CPB
003073/ 301  LAB
003074/ 003  RFC        PASS IF GT ØR EQ 'A'
003075/ 074  CPI 133    TEST IF GT 'Z'
003077/ 047  RET        GØ, CARRY TELLS ALL
003100/ 046  LEI 002                  SYM INPUT
003102/ 066  LLI 350    CHAR IN 013350
003104/ 056  LHI 013
003106/ 370  LMA        STØRE
003107/ 146  CAL 003067 TEST NEXT CHAR
003112/ 100  JFC 003131 NØT 'A'-'Z' !! ERRØR!
003115/ 060  INL        SET UP NEXT ØNE
003116/ 041  DCE        CØUNT
003117/ 110  JFZ 003106 NØT DØNE, LØØP
003122/ 370  LMA        DØNE- STØRE LAST CHAR
003123/ 340  LEA        NØW SET UP REG, 3 GØES 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 DØNE
003131/ 016  LBI 277                  ERRØRS CØME HERE
003133/ 025  RST 020    PRINT '?'
003134/ 144  JMP 003014 GØ GET ANØTHER INPUT
003137/ 106  CAL 001200 CR/LF AND '*' INITIAL AND FINAL ADR
003142/ 061  DCL        GØT FIRST ADR
003143/ 016  LBI 240    PRINT BLANK
003145/ 104  JMP 001207 GET FINAL ADR,GØ BACK HØME
003150/ 016  LBI 022                  SEARCH TABLE
003152/ 066  LLI 021    5 BYTE TABLE
003154/ 056  LHI 004
003156/ 307  LAM        NØW GET 1ST CHAR
003157/ 060  INL        READY FØR NEXT CHAR
003160/ 272  CPC        CØMPARE TABLE AND I/P
003161/ 110  JFZ 003204 JMP IF NØT EQUAL
003164/ 307  LAM        GET 2ND
003165/ 060  INL        READY FØR 3RD
003166/ 273  CPD        CØMPARE
003167/ 110  JFZ 003205 JMP IF NØT THE SAME
003172/ 307  LAM        NØW FØR THE 3RD
003173/ 060  INL        AND PREPARE FØR DATA
003174/ 274  CPE        CØMPARE AS BEFØRE
003175/ 110  JFZ 003206 AND JUMP IF NØT NICE
003200/ 317  LBM        GET 'GØ TØ' ADDRESS
```

```
003201/ 060  INL
003202/ 307  LAM        2 BYTES ØF IT
003203/ 007  RET        AND RETURN
003204/ 060  INL                        LØØK AT NEXT SYMBØL
003205/ 060  INL                        IN THE TABLE
003206/ 060  INL
003207/ 060  INL
003210/ 011  DCB        CØUNT ØUR TRYS
003211/ 053  RTZ        END ØF TABLE
003212/ 104  JMP 003156 MØRE TØ CHECK
003215/ 056  LHI 004                    3BYTE TABLE SEARCH
003217/ 021  DCC        'C' IS CØUNTER
003220/ 053  RTZ        RETURN WHEN DØNE TABLE
003221/ 307  LAM        NØW LØØK AT THE FIRST ENTRY
003222/ 060  INL        CØMPARE WITH DATA
003223/ 273  CPD        JMP IF NØT LIKED
003224/ 110  JFZ 003237 2ND ENTRY AS ABØVE
003227/ 307  LAM
003230/ 274  CPE
003231/ 110  JFZ 003237 AND JUMP, MAYBE
003234/ 060  INL        FETCH DATA FRØM TABLE
003235/ 307  LAM
003236/ 007  RET        RETURN TØ LDS RØUTINE
003237/ 060  INL        NEXT ENTRY
003240/ 060  INL        EØØP AND TRY AGAIN
003241/ 104  JMP 003217 LØØP AND TRY AGAIN
003244/ 345  LEH                        BRKPT EXECUTE
003245/ 336  LDL        SAVE L,H  LØSING D,E
003246/ 066  LLI 364    SAVE REGISTERS A-E
003250/ 056  LHI 013    IN RAM (LØC 013364 TØ 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    NØW DISPLAY CARRY FLAG
003265/ 022  RAL        RØTATE IN CARRY AND CØNVERT TØ
003266/ 340  LEA        ASCII
003267/ 016  LBI 240    Ø/P BLANK
003271/ 025  RST 020
003272/ 314  LBE        Ø/P CARRY FLAG
003273/ 025  RST 020
003274/ 046  LEI 005    SET UP CØUNT TØ PRINT REGISTERS
003276/ 066  LLI 360    START ØF SAVED REGISTERS
003300/ 106  CAL 000253 PRINT BYTE AS ØCTAL
003303/ 060  INL        NEXT REGISTER
003304/ 041  DCE        CØUNT
003305/ 110  JFZ 003300 LØØP TILL DØNE
003310/ 061  DCL        GØ BACK ØNE REG
003311/ 106  CAL 001027 AND GET DATA AT H,L LØCATIØN
003314/ 106  CAL 000253 AND PRINT IT
003317/ 005  RST 000    NØW WERE DØNE, GØ HØME
003320/ 066  LLI 373                    XQT RØUTINE
003322/ 106  CAL 003143 LØAD ADDRESS
003325/ 104  JMP 003052 EXEC WILL SEND US THERE
```

RØM NUMBER 3 CØNTINUED WITH DPS RØUTINES
RØM 4 CØNTAINS THE SYMBØL TABLES,
AS FØLLØWS:

1. 5 BYTE TABLE

THE 5 BYTE TABLE ØCCUPIES PØSITIØNS 004021 TØ 004157 INCLUSIVE AND CØNTAINS ALL MØNITØR CØMMANDS PLUS THE MACHINE CØMMANDS HLT, INP,ØUT,RST, AND THE SPECIAL SYMBØL ???, INDICATIING A NØ FIND CØNDITIØN ØN ØUTPUT. THE INPUT RØUTINE DØES NØT USE THIS SYMBØL. THE FØRMAT IS THUS:

```
ASCII    X    Q    T   (DATA FIELD)
ØCTAL   130  121  124  320   003
```

ADDRESS 021 022 023 024   025

WHEN A FIND IS MADE DURING A SEARCH, THE DATA FIELD IS MØVED TØ REGISTERS A AND B, AND AN INDIRECT JUMP MADE TØ THAT ADDRESS, IF THE MS HALF ØF THE ADDRESS IF A 2XX, THE EXEC WILL LØØK FØR TWØ ADDRESSES BEFØRE GØING TØ THE RØUTINE.
DURING A SYMBØLIC DUMP, THE LAST 5 SYMBØLS ARE USED FØR THE APPRØPRIATE MACHINE CØMMANDS, AND ARE STØRED AS ØUTPUT.

ADR	SYMB	LS	MS (ADDRESS ØF RØUTINE)
004021/	XQT	320	003
004026/	EDT	236	001
004033/	LDØ	120	201
004040/	LBF	201	202
004045/	DPØ	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/	LØC	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 CØNTAINS TWØ BYTES ØF ASCII CØDE AND ØNE DATA BYTE, WHICH IS A MASKED PØRTIØN ØF THE INSTRUCTIØN. THE FØRMAT IS:

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

THE TABLE ØCCUPIES LØCATIØNS 004156 TØ 004273,AND IS USED IN TWØ WAYS. THE LDS RØUTINE CØMPARES THE TWØ ASCII CHARACTERS TØ THE INPUT CHARACTERS, AND RETURNS THE DATA IN THE A REGISTER IF A FIND IS MADE.
FØR THE DPS RØUTINE, THE PARTIAL WØRD (DATA) IS TESTED, AND THE H AND L REGISTERS ARE USED TØ RETRIEVE THE ASCII AS NEEDED.

3 BYTE TABLE:

LØCATIØN	ASCII	DATA	
004156	LC	002	
004161	RC	012	
004164	AL	022	
004167	AR	032	
004172	JMP	104	(JMP)
004176	CAL	106	(CAL)
004202	RET	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 ØCCUPIES PØSITIØNS 004274 TØ 004377, AND IS USED BY THE DPS RØUTINE.
THE FØRMAT IS:

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

THE MASK CHARACTER IS USED TØ MASK (AND) DØN'T CARE BITS IN THE INPUT BYTE, THE REMAINING BITS ARE CØMPARED TØ THE DATA IN THE NEXT FIELD TØ DECØDE AN INSTRUCTIØN. IF A FIND IS MADE THE ADDRESS IS USED FØR AN INDIRECT JUMP (TØ 005AAA). THE LAST ENTRY IS AN UNCØNDITIØNAL FIND WHICH ØUTPUTS THE ERRØR SYMBØL ???.
THE DATA FIELD CØLUMN IS USED FØR VARIØUS PURPØSES BY THE CALLED RØUTINES.

4 BYTE TABLE

LØCATIØN	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

SYMBØLIC RØUTINES

NØTE: THESE RØUTINES CØVER PART ØF RØM 3,4 AND ALL ØF RØMS 5,6

```
003330/ 106  CAL 005352 GET 3 BYTES   DPS ØUTPUT
003333/ 106  CAL 005104 LØAD THEM INTØ REGISTERS
003336/ 106  CAL 005313 ØUTPUT THEM
003341/ 347  LEM        LØAD E WITH DATA
003342/ 106  CAL 006320 DECØDE LENGTH
003345/ 012  RRC        1 BYTE INSTR?
003346/ 100  JFC 005363 YES-GØ TØ 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 SØ
003364/ 106  CAL 000253 Ø/P IMMEDIATE DATA
003367/ 104  JMP 005363 AND GØ TØ LINE CHECK
003372/ 327  LCM        YES ITS 3 BYTE! GET LS ADR
003373/ 106  CAL 000362 INC CLP
003376/ 312  LBC        MØVE ADR TØ B
003377/ 300  LAA        NØP (UNUSED BYTE)
004000/ 106  CAL 001023 GET DATA (MS ADR BYTE)
004003/ 327  LCM        SAVE IN C
004004/ 106  CAL 005104 LØAD 3 BYTES
004007/ 060  INL        SET UP DATA PØINTERS
004010/ 060  INL
004011/ 106  CAL 001061 ØUTPUT THIS ADRESS
004014/ 104  JMP 005363 AND GØ ØN TØ LINE CHECK
005000/ 016  LBI 012                SYMBØLIC 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 DESTINATIØN 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        FETCH 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 00XXXXX0
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 00XXX000 (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        FETCH J,C,ØR R FRØM TABLE
005221/ 303  LAD        RESTØRE BINARY
005222/ 044  NDI 307    MASK AND CHECK IF UNCØNDITIØNAL
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 CØNDITIØN
005235/ 303  LAD        IN 3 BYTE TABLE
005236/ 044  NDI 070    MASK ALL BUT CØNDITIØN
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        FETCH 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  JTZ 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
005355/ 327  LCM        GET 3RD CHAR FRØM TABLE
005356/ 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  JTZ 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  SYMBØLIC 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 INSTRUCTIØN
006010/ 303  LAD          LØØK AT 2ND CHAR
006011/ 146  CAL 006345 ENCØDE AS DETINATIØN 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  JTZ 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  JTS 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 LØØP
006105/ 106  CAL 000311 INCR CLP       3BYTE (MUST WANT AN ADR)
006110/ 106  CAL 001023 GET MØRE DATA
006113/ 106  CAL 001212 AND STØRE TWØ BYTES (CLP,CLP-1)
006116/ 104  JMP 006101 GØ BACK TØ LØØP
006121/ 377  HLT          UNUSED HALT(!)
006122/ 074  CPI 122                        TEST FØR 1ST CHAR ='R'
006124/ 110  JFZ 006146 NØ- KEEP LØØKING
006127/ 026  LCI 005      IS IT A RØTATE?
006131/ 066  LLI 156
006133/ 106  CAL 003215 SEARCH 3 BYTE TABLE
006136/ 066  LLI 202
006140/ 150  JTZ 006234 IF NØ FIND,TEST FØR RETURN
006143/ 144  JMP 006046 GØ FINISH UP
006146/ 314  LBE                            ACC GRØUP
006147/ 343  LED          PUT CHARACTERS AWAY
006150/ 332  LDC
006151/ 026  LCI 013      SET UP TABLE SEARCH
006153/ 066  LLI 236      (ACC GRØUP,IN(R),DC(R))
006155/ 106  CAL 003215 SEARCH TABLE
006160/ 150  JTZ 006214 NØ FIND, KEEP LØØKING
006163/ 320  LCA          GET 1ST CHAR
006164/ 044  NDI 200      CHECK FØR IMMEDIATE INSTR
006166/ 302  LAC          RESTØRE CHAR
006167/ 341  LEB
006170/ 110  JFZ 006020 GØ AWAY IF IMMEDIATE INSTR
006173/ 301  LAB          TEST THE 3RD CHAR
006174/ 106  CAL 006345 ENCØDE AS A REGISTER
006177/ 002  RLC
006200/ 002  RLC
006201/ 002  RLC
006202/ 202  ADC          ADD TØ PARTIAL WØRD
006203/ 104  JMP 006046 FINISH UP
006206/ 106  CAL 001023 6 BYTES NØT USED (!)
006211/ 104  JMP 003143 (!!!)
006214/ 323  LCD                             TRANSFER GRØUP (JMP,CAL,RET)
```

```
006215/ 334  LDE          MUSICAL REGISTERS
006216/ 341  LEB
006217/ 066  LLI 172      START ØF 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 (!) MUST BE AN ERRØR
006234/ 060  INL          IS IT UNCØNDITIØNAL?
006235/ 026  LCI 002
006237/ 146  CAL 003215 THEN GØ TØ FINISH
006242/ 110  JFZ 006046 GET PART WØRD
006245/ 061  DCL          AS MUCH AS WE CAN
006246/ 307  LAM          BLANK ØUT SØME
006247/ 044  NDI 303
006251/ 310  LBA          AND LØØK UP THE CØNDITIØN
006252/ 066  LLI 206
006254/ 026  LCI 011
006256/ 146  CAL 003215
006261/ 150  JTZ 003131 NØ-FIND ERRØR(!)
006264/ 201  ADB          ADD IN CØNDITIØN BITS
006265/ 144  JMP 006046 FINISH IT
006270/ 343  LED                   INP/ØUT/RST
006271/ 016  LBI 240      ENTER AS MØNITØR RØUTINE
006273/ 025  RST 020      PRINT A BLANK
006274/ 146  CAL 000205 INPUT THE ØCTAL ARGUMENT
006277/ 300  LAA          NØP (AGAIN -REALLY TØM !!)
006300/ 304  LAE          GET 2ND CHAR
006301/ 074  CPI 123      IS IT AN 'S'
006303/ 307  LAM          GET THE ØCTAL 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/ØUT-RØTATE ARGUMENT
006312/ 046  LEI 101      AND PUT THE REST INTØ E
006314/ 204  ADE          ADD THE TWØ PARTS TØGETHER
006315/ 144  JMP 006046 AND FINISH (GØØDBYE CRUEL WØRLD)
006320/ 250  XRA                   INSTRUCTIØN 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-BEGØNE
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 GRØUP)
006340/ 010  INB          NØW SET UP B
006341/ 010  INB
006342/ 010  INB          CØME HERE IF 2 BYTE
006343/ 301  LAB          SØ NØW A IS 001 ØR 003
006344/ 007  RET          GØ HØME AND TELL ABØUT IT
006345/ 066  LLI 370               REGISTER DECØDE
006347/ 056  LHI 006      LØØK AT TABLE
006351/ 277  CPM          TEST
006352/ 110  JFZ 006361 NØ FIND -LØØP
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 NØT ZERØ GØ LØØP
006365/ 104  JMP 003131 NØT IN TABLE- ITS AN ERRøR FØLKS
```

REGISTER LØØK UP TABLE

LØCATIØN	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	MEMØRY(M)	7	115

SOFTWARE LISTING

microsystems' memory products

ARE AVAILABLE FROM THESE OUTLETS:

U.S.A.

microsystems sales offices

CA., Palo Alto, (415)493-0848
CA., Santa Ana, (714)979-6522
IL., Schaumburg, (312)894-7660
PA., Huntingdon Valley, (215)947-5641/2
MA., Waltham(617)890-2255
NJ., Morris Plains(201)539-8050

sales representatives

AL., Huntsville, Rep Inc. (205)881-9270
AZ., Scottsdale, Barnhill Five (602)947-5745
CA., Los Alamitos, J.S. Heaton (213)598-6676
CA., Los Altos, W.W. Posey (415)948-7771
CA., San Diego, J.S. Heaton (714)452-1456
CO., Littleton, Simpson (303)794-8381
FL., Orlando, G.F. Bohman (305)855-0274
IL., Chicago, L-Tec (312)286-1500
IN., Fort Wayne, Waid & Assoc. (219)422-4344
MD., Towson, L.H. Kolman (301)752-8756
MA., Waltham, Rantek (617)890-5110
MI., Milford, Tom Mulligan (313)363-5575
MN., Minneapolis, Comstrand (612)560-5300
NJ., Tenafly, ABC (201)568-2354
NY., Cicero, Adv. Comp. (315)699-2651
NY., Tarrytown, ABC (914)631-2650
NY., Williston Park, ABC (516)747-6610
OH., Cincinnati, Ron Makin (513)232-5588
OR., Beaverton, Arneson (503)643-5754
PA., Huntingdon Valley, ABC (215)947-5641
TX., Dallas, J. Clay (214)350-1281
UT., Salt Lake City, Simpson (801)486-3731
WA., Seattle, Arneson (206)762-7664

stocking distributors

AZ., Phoenix, Moltronics (602)272-7951
AZ., Phoenix, Sterling (602)258-4531
CA., Riverside, Electronic Supply (714)683-7300
CA., San Diego, Moltronics (714)278-5020
CA., San Diego, Sterling (714)565-2441
CA., San Carlos, Sterling (415)592-2353
CA., Southgate, Moltronics (714)521-7412
CA., Sunnyvale, Bell Industries (408)734-8570
CA., Sun Valley, Sterling (213)767-5030
CT., Westport, Hi-Rel (203)226-4731
FL., Clearwater, Diplomat (813)443-4514
IL., Elk Grove Village, Diplomat (312)595-1000
IL., Elmhurst, Semispecs (312)279-1000
IL., Skokie, Bell Industries (312)965-7500
IN., Indianapolis, Semispecs (317)243-8271
MD., Baltimore, Arrow (301)247-5200
MD., Gaithersburg, Pioneer (301)948-0710
MA., Boston, Demambro (617)787-1200
MA., Dedham, Gerber (617)329-2400
MA., Watertown, Sterling (617)926-9720
MI., Farmington, Diplomat (313)477-3200
MI., Farmington, Semispecs (313)478-2700
MN., Bloomington, Arrow (612)888-5522
MN., Minneapolis, Diplomat (612)788-8601
MN., Minneapolis, Semispecs (612)854-8841
MO., Hazelwood, Semispecs (314)731-2400
MO., Kansas City, Semispecs (816)452-3900

stocking distributors

MO., St-Louis, Diplomat (314)634-8550
NJ., Haddonfield, Mid-Atlantic (609)428-8288
NJ., Kenilworth, Newark (201)272-8410
NJ., Saddlebrook, Arrow (201)797-5800
NM., Albuquerque, Sterling (505)345-6602
NY., Farmingdale, Arrow (516)694-6800
NY., Freeport, Milgray (516)546-6000
NY., New York, C. Tennant (212)679-1300
NY., Rochester, Simcona (716)328-3230
NY., Woodbury, C. Tennant (516)364-9070
OH., Dayton, Diplomat (513)228-1080
OH., Dayton, Semispecs (513)278-9455
OH., Solon, Repco (216)248-8900
PA., Horsham, Pioneer (215)674-5710
PA., Pittsburgh, Semispecs (412)781-8120
TX., Dallas, KA Electronic Sales (214)634-7870
TX., Dallas, Semispecs (214)358-5211
TX., Dallas, Sterling (214)357-9131
TX., Houston, Sterling (713)623-6600
UT., Salt Lake City, Diplomat (801)486-7227
WA., Seattle, Sterling (206)762-9102
WI., Wauwatosa, Semispecs (414)257-1330

CANADA

microsystems sales offices

QUE., Montreal, (514)747-5551
ONT., Ottawa, (613)828-9191
ONT., Toronto, (416)741-3667

stocking distributors

BC., Vancouver, R.A.E. (604)687-2621
ONT., Malton, Semispecs (416)678-1444
ONT., Rexdale, Future (416)677-7820
ONT., Toronto, Zentronics (416)789-5811
QUE., Montreal, Future (514)735-5775

EUROPE

microsystems sales offices

W. GERMANY, Stuttgart, (0711)65 31 16/26
U.K., London, (01)573-6582/4

stocking distributors

A., Vienna, Elbatex 0222-83-02-16
CH., Wettingen, Elbatex 004056-26-56-41
D., Berlin, Jastron 030/3121203
D., Frankfurt, Sommer 0611-55-02-89
D., Heilbronn, Elbatex 071-31-87126
D., Karlsruhe, ISA-Elektronik 0721/43975
E., Madrid, I.E.P. 274-1007
F., Paris, R.T.F. 722-70-40
GB., East Grinstead, Barlec, E. Grinstead 24383
GB., Egham, Athena, Egham 6161
GB., Harlow, Distronic, Harlow 32947
GB., Portsmouth, Phoenix, Cosham 73441
GB., Yiewsley, Semispecs, W. Drayton 46415
GB., Glasgow, Spenco 36311/3
I., Milano, Silverstar 02-4996
N., Amsterdam, Tekelec 020-92-87-66
S., Stockholm, Stahlberg & Nilsson 847-29-80

JAPAN

Tokyo., Mitsubishi 210-2121

MICROSYSTEMS INTERNATIONAL LIMITED, BOX 3529, STA C, OTTAWA, CANADA K1Y 4J1

MF8008

SALES OFFICES:

Ottawa, Ontario: Tel. (613)828-9191, TWX 610-562-1910
Telex 053-4580
Montreal, Quebec: Tel. (514) 747-5880, TWX 610-421-3634
Palo Alto, Cal.: Tel. (415)493-0848, TWX 910-373-1281
Santa Ana, Cal.: Tel. (714) 979-6522, TWX 910-595-1780
Schaumburg, Illinois: Tel. (312) 894-7660, TWX 910-291-3548
Huntingdon Valley, Pa.: Tel. (215) WI7-5641/2, TWX 510-665-5303
London, England: Tel. (01)573-6582/4, Telex 934421
Stuttgart, Germany: Tel. (0711) 65 31 16/26, Telex 722290