66123

22 November 1972

BURROUGHS MICROPROCESSOR MACHINE IMAGE GENERATOR (MIG) USER'S MANUAL

Burroughs Corporation

Federal and Special Systems Group

Paoli, Pa. 19301

66123

22 November 1972

BURROUGHS MICROPROCESSOR MACHINE IMAGE GENERATOR (MIG) USER'S MANUAL

Burroughs Corporation

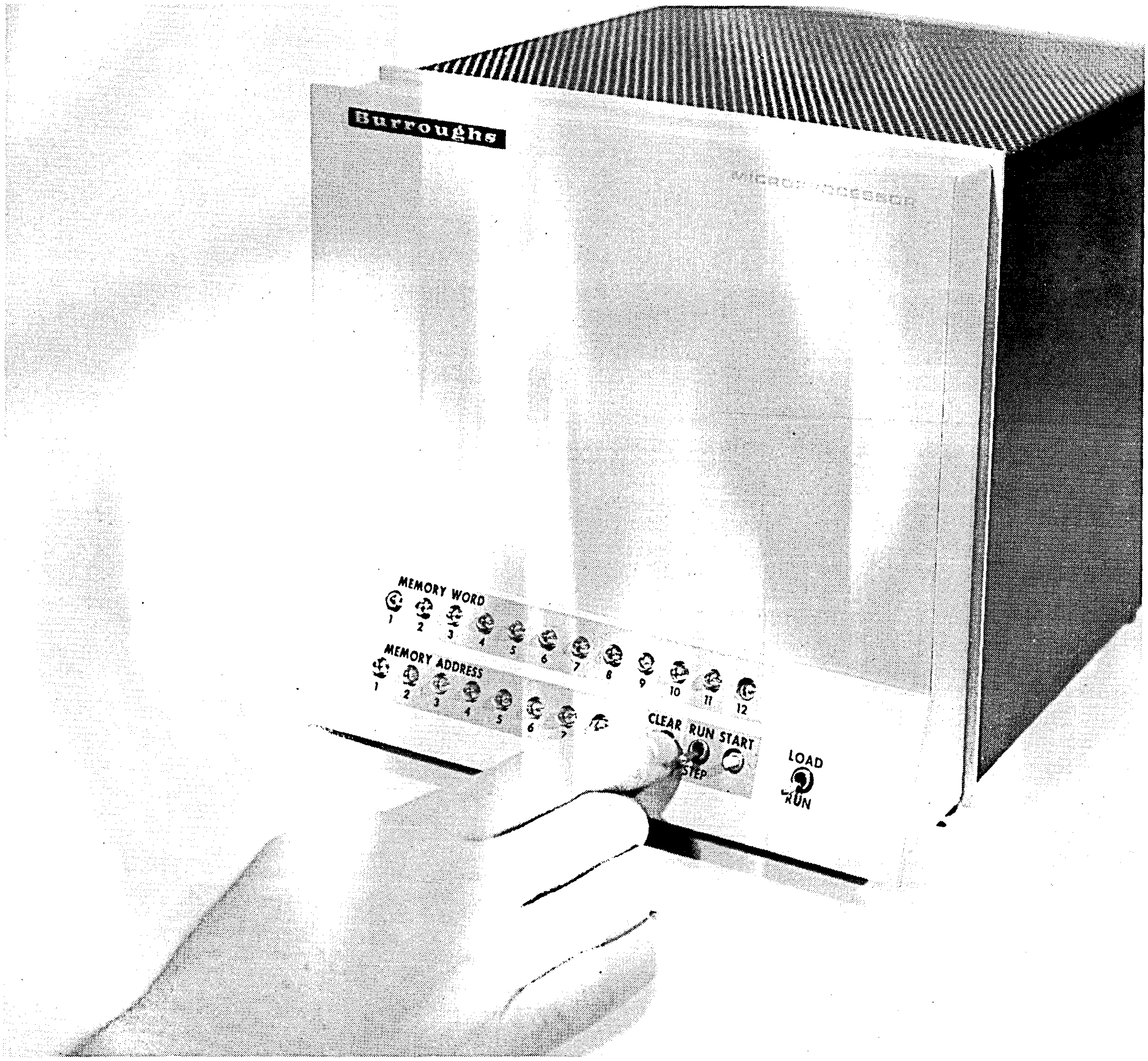
Federal and Special Systems Group

Paoli, Pa. 19301

CONTENTS

CONTENTS (Continued)

ILLUSTRATIONS



Burroughs Microprocessor Machine Image Generator (MIG)

1. INTRODUCTION

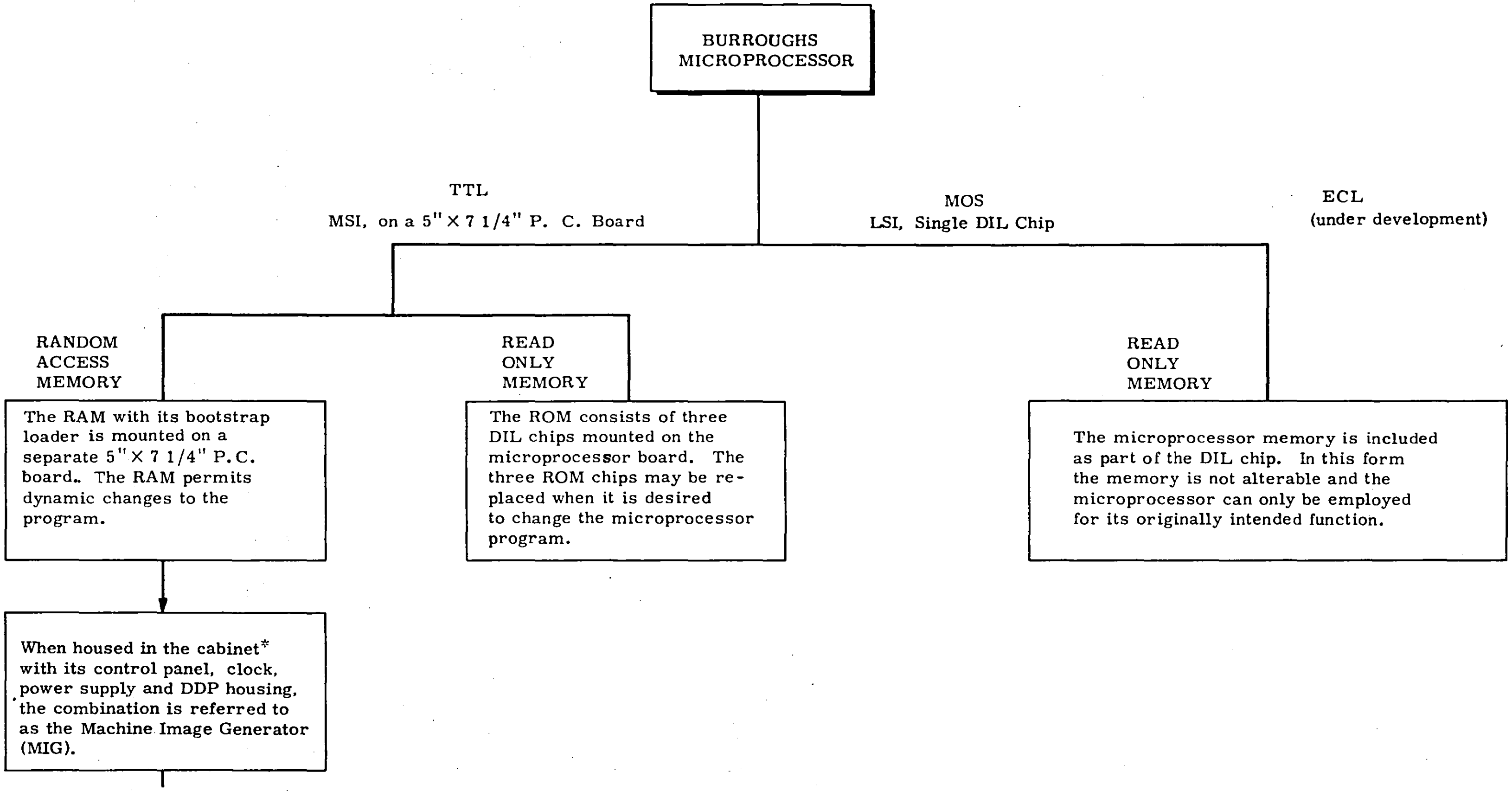
BACKGROUND

The programmable logic controller or microprocessor is a relatively new concept in solid-state control. Its predecessor, the conventional relay bank or solid-state sequencer, must be wired differently for each control problem for the specified sequence of events. The programmable controller, on the other hand, only requires that the new sequence be stored in its memory. The algorithms through which inputs produce desired outputs are implemented in the stored programs. Thus the only wiring is that required to connect inputs and outputs.

Programmable logic controllers perform sequencing operations by (1) scanning inputs such as relay contacts, limit switches, terminal devices, pushbuttons, valves, et., (2) comparing the inputs to the conditions specified in the program; and (3) by sending data, energizing or deenergizing outputs in accordance with the programmed instructions. Current experience suggests that programmable controllers offer a cost/effective problem solution when 50 or more relay functions are to be implemented. In addition, there are many advantages outside of direct costs including:

- Reliability - Controllers built from integrated circuits have inherently higher reliability than a relay.

- Speed - A complete scan of several inputs and the subsequent operation of several outputs can often be performed in less time than it takes to operate a typical 20 millisecond relay.



The RAM with its bootstrap loader is mounted on a separate 5" X 7 1/4" P.C. board.. The RAM permits dynamic changes to the program.

When housed in the cabinet* with its control panel, clock, power supply and DDP housing, the combination is referred to as the Machine Image Generator (MIG).

The ROM consists of three DIL chips mounted on the microprocessor board. The three ROM chips may be replaced when it is desired to change the microprocessor program.

The microprocessor memory is included as part of the DIL chip. In this form the memory is not alterable and the microprocessor can only be employed for its originally intended function.

Figure 1-1. Microprocessor Variations

- Ease of building the control system - The program is entered into a computer memory, eliminating the complex wiring of relays or electronic logic. Programs may be tested and debugged immediately after they are entered, saving days or weeks. Program changes are made by software modification - usually no rewiring is necessary

- Computer monitoring and control - These functions are directly applicable to programmable controllers which have an inherent ease of interfacing with host computers locally or remotely.

- Adaptability - Types of functions are software dependent.

- Expansion - The number of functions is not limited by a "hard wired" system but can be expanded by addition of hardware I/O modules (called Device Dependent ports or DDP's) and software modification.

To exploit these advantages, Burroughs has developed a microprocessor which is basically a miniature version of its successful D-Machine.

The Burroughs D-Machine is a family of digital processors based on advanced system architecture. The D-Machine utilizes a modular building block architecture. Over the past few years the concept of modular building blocks has been extended to give rise to a new and unique concept in the architecture and implementation of data processors. The new concept has been entitled the "Interpreter Based System" or "D-Machine".

The D-Machine incorporates two design concepts: (1) building block structure and (2) "soft machine architecture" through microprogramming. In the D-Machine architecture, the fundamental logic functions have been organized into building blocks, omitting the control logic associated with conventional processor design. These fundamental building blocks thus represent uncommitted logic or hardware which yields maximum flexibility, and which becomes committed to a specific task by control signals originating outside the basic building block. These control signals have two sources: The firmware (the microprogram) and the hardware providing the interface with the external device (device dependent ports).

The Burroughs Microprocessor is a smaller version of the D-Machine and is referred to as the Mini-D. The Mini-D microprocessor is currently available in either of two logic families: (Figure 1-1) i.e., Transistor Transistor Logic (TTL) in Medium Scale Integration (MSI) form or Metal Oxide Semiconductor (MOS) in Large Scale Integration (LSI).

The TTL version is available in two forms: with Random Access Memory (RAM) or Read-Only Memory (ROM). When equipped with RAM the microprocessor consists of two printed circuit cards; the first card is a microprocessor constructed from TTL logic, and the second card contains a bootstrap loader and a random access memory to control the functions of the microprocessor.

If the Mini-D is to be employed in an application where its program is not required to be changed, then a read-only memory (ROM) is the economical choice. Under these conditions only the microprocessor card is necessary; the design of the microprocessor card having been provided with electrical connections for 256 words of read-only memory (ROM). Thus a complete microprocessor on one 7 1/4-inch by 5-inch printed circuit card may be incorporated into a customer's product or system. A more economical processor form is available as a single dual-in-line (DIL) MOS LSI chip. The MOS LSI chip has space for the storage of 256 words of memory. These memory words are not alterable but fixed at the time the chip is manufactured.

The TTL version is designed to have a nominal instruction execution time of 1 μsec., while the MOS version is 10 μsec.

MACHINE IMAGE GENERATOR

Foreseeing the need for a means to develop and debug programs to be employed in the microprocessor, as well as the development of I/O interfaces, the Burroughs Corporation has developed a laboratory instrument known as the Machine Image Generator (MIG). This manual describes the operation and programming of the Microprocessor and MIG.

MANUAL ORGANIZATION

This manual is organized into four sections and four appendices. The introduction provides a brief description of the Burroughs D-Machine series and a detailed description of the Mini-D or Microprocessor including the differences between the various logic forms of the Mini-D. Section 2 describes the physical and functional characteristics of this particular form of the Microprocessor, the "Machine Image Generator". Section 3 contains the operating instructions for the MIG. Section 4 presents complete programming details as well as general software characteristics. The appendices contain material which augment the programming section.

2. THE BURROUGHS MICROPROCESSOR MACHINE IMAGE GENERATOR

Foreseeing the need for a means to develop and debug programs to be employed in the microprocessor as well as the development of I/O interfaces to devices, the Burroughs Corporation has developed a laboratory instrument known as the microprocessor Machine Image Generator (MIG). The MIG consists of the microprocessor card, read-write memory, a control panel, power supply, clock oscillator, and two wire-wrap utility boards; all housed in a 9 1/2-inch by 8 1/2-inch by 11 1/2-inch cabinet as shown in Figure 2-1. For convenience most interconnections are made with ribbon cables and 16-pin DIL plugs and sockets.

In all, the MIG consists of five printed circuit or wire wrap boards and a power supply. In the small raised front portion of the MIG are three printed circuit boards interconnected by flat-ribbon cables. The smallest of the three is the Panel Board (PB) which contains the lamps, lamp drivers and switches used to control the microprocessor. The entire microprocessor is contained on the first printed circuit board (MP-1) and includes three sockets for a 256-word by 12-bit read only memory (ROM).

However, in place of the three ROM chips, three cables carry the eight address bits to, and 12 memory bits to and from the read-write memory which is located on the second printed circuit card (MP-2). Also on this card is a Port Select Unit (PSU) and a 32-word by 12-bit read-only bootstrap loader program. The ROM program loads the read-write memory from teletype while in the LOAD mode.

In the rear section of the cabinet are two wire-wrap utility boards, one of which (UB-1) is partially occupied by the clock oscillator, power connections and a teletype interface. All unpopulated portions of these boards are for the users convenience in constructing DDP's.

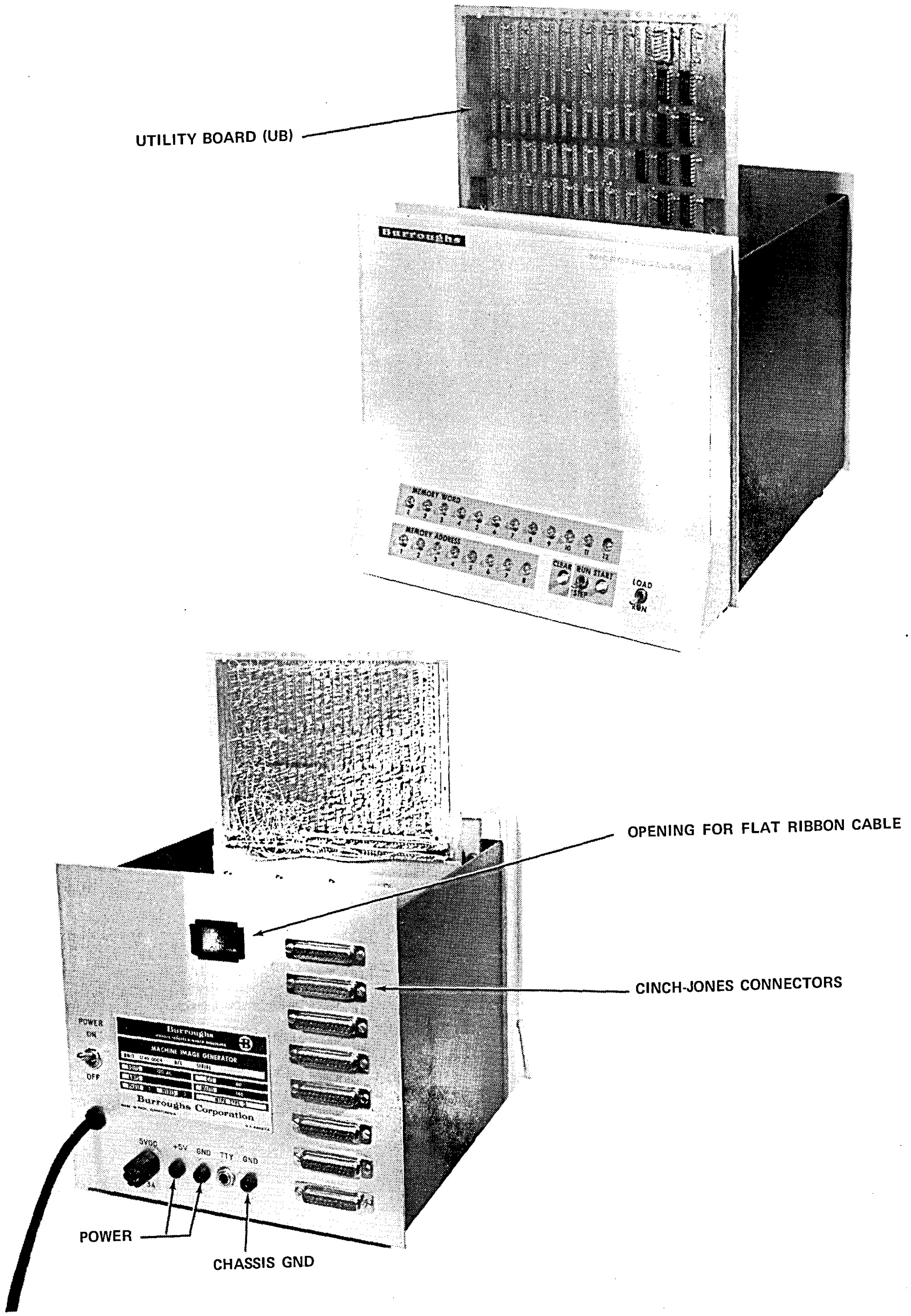


Figure 2-1. Two Views of the MIG

In addition to the three ribbon cables used for memory interconnections there are seven other ribbon cables. One connects the control panel with UB-1 and carries control signals and power. Another cable connects the memory board (MP-2) with UB-1. and carries teletype inputs to the Mini-D from the TTY interface on UB-1. Four cables are used to connect the PSU on MP-2 to the utility boards and can be moved at the users convenience. Finally there is a single flat ribbon cable that connects MP-1, MP-2, and UB-1 and has the same pin configuration as the MOS-LSI version of the Microprocessor.

THE MICROPROCESSOR

The microprocessor is an 8-bit serial machine (serial by bit internally) with a 256-word by 12-bit microprogram memory. Programmatically it appears as a parallel machine for most functional operations. The microprocessor (Figure 2-2) consists of 5 functional parts described in Table 2-1 below.

Table 2-1. Functional Parts

Acronym	Unit	Functions
LU	Logic Unit	Data registers, serial adder
MPM	Microprogram Memory	Microprogram sequences. Some words have literals, others have specific controls created for the microprogrammer.
MCU	Memory Control Unit	Registers for memory addressing
CU	Control Unit	Timing and condition testing, successor selection, instructive decoding
EXI	External Interface	Interface to the external environment.

Also implemented in the MIG is a bootstrap loader - ROM micromemory containing a program that loads the read-write memory from a model 33 Teletype or an equivalent device.

The Microprocessor runs in two modes RUN and LOAD, controlled by a toggle switch on the front panel. In the RUN mode, the 256 X 12 read-write memory determines the sequence of operation. The program in the read-write memory is changed in the LOAD mode. In this mode, with the teletype connected to the phone-plug, the bootstrap program will read characters from the teletype keyboard or paper tape reader. The block diagram (Figure 2-3) shows the relationship of the functional parts of the MIG. Figure 2-4 shows the functional details.

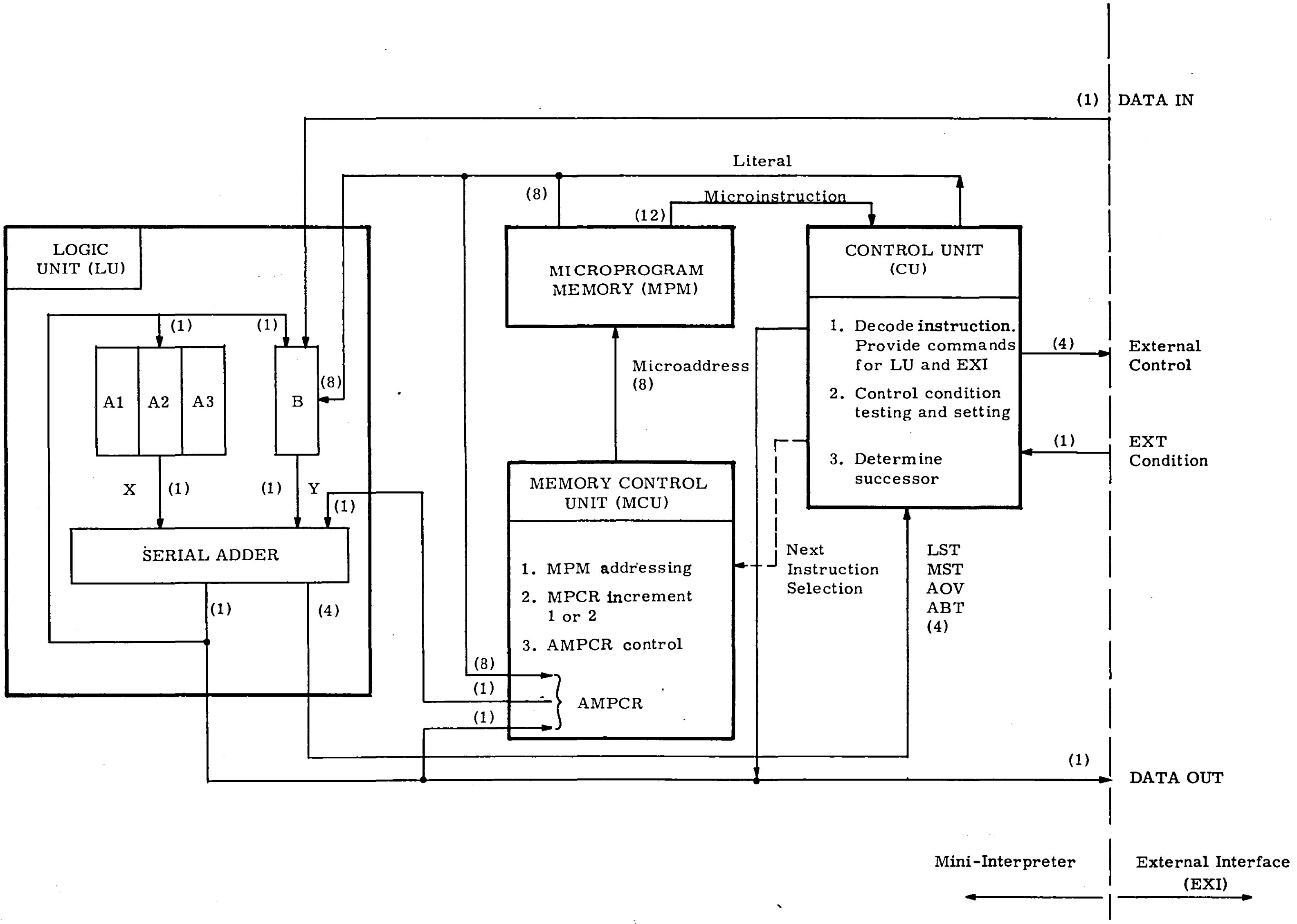


Figure 2-2. Microprocessor Organization

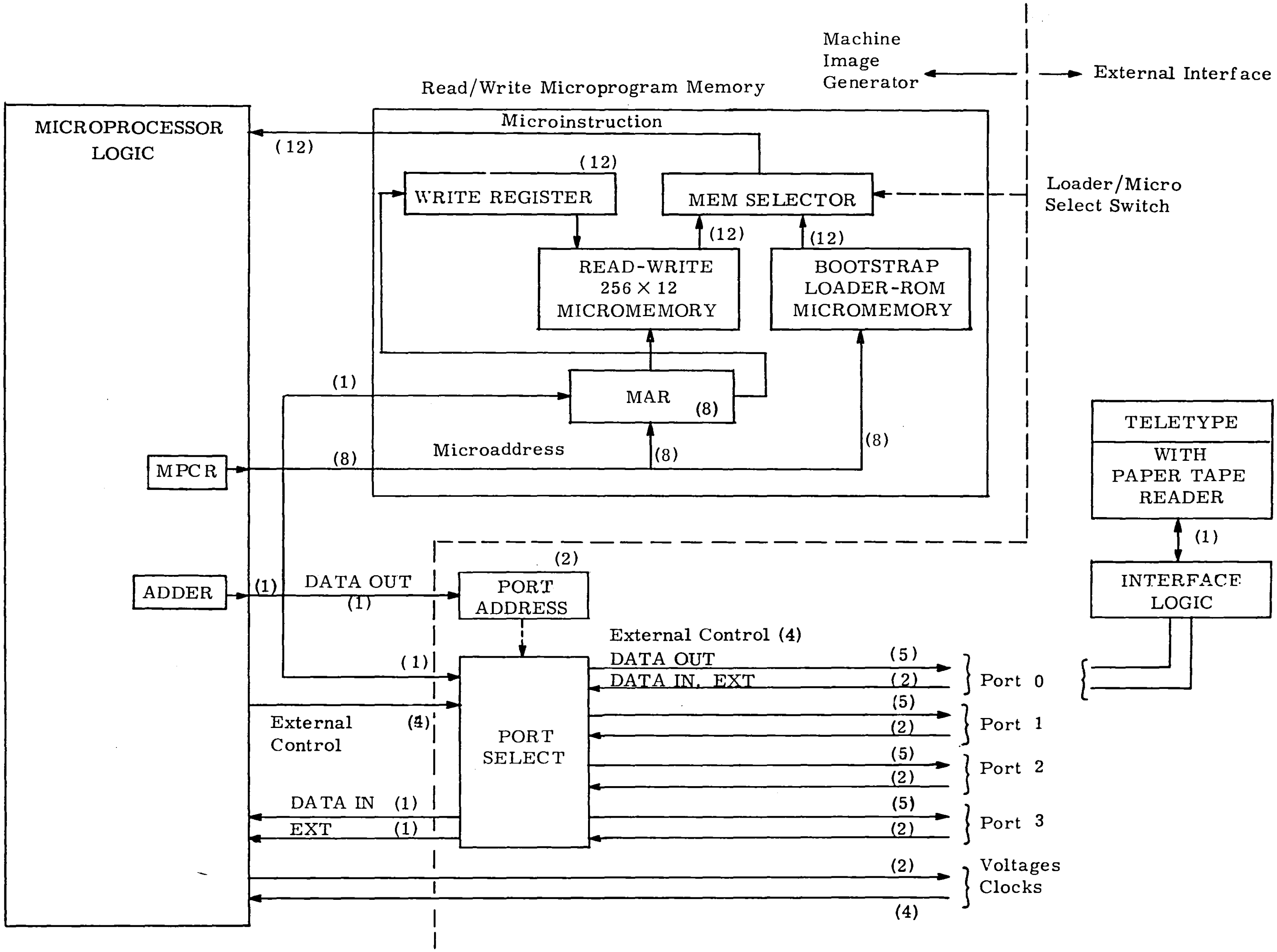


Figure 2-3. MIG Block Diagram

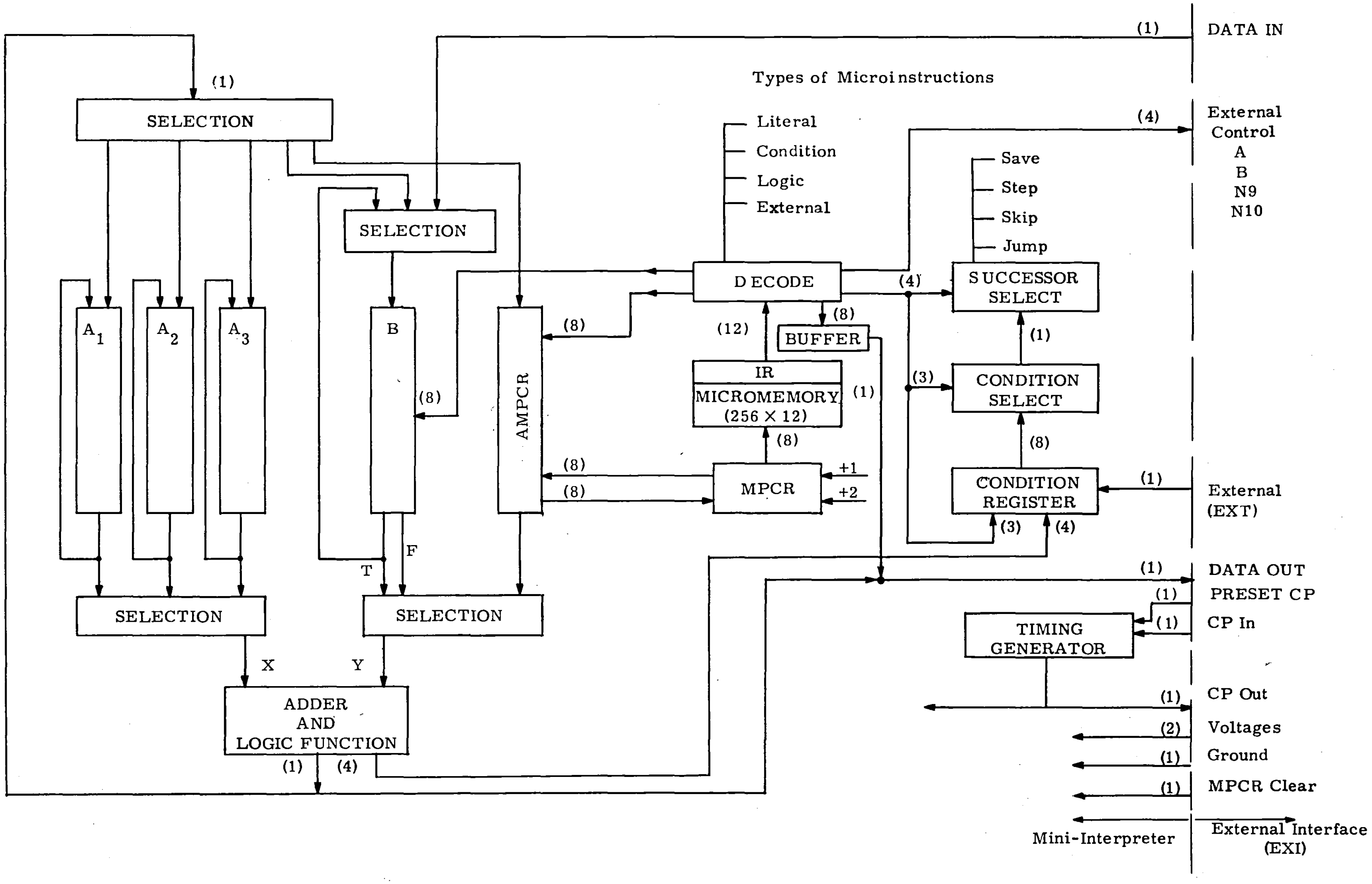


Figure 2-4. Microprocessor Detail Functions

The Logic Unit LU

The logic unit consists of three 8-bit A registers (A1, A2, A3), an 8-bit B register, a serial adder, a carry flip-flop, and selectors. The registers are recirculating static shift registers so that information can be transferred into the adder without changing the input registers. The inputs to the adder are one of the A-registers or zeros, and one of B or AMPCR (alternate microprogram count register). The output from the adder can be to A1, A2, A3, B, AMPCR and external registers (via the DATA out line). The adder also feeds four conditions to the condition registers, "least bit true" (LST), "most bit true" (MST), "overflow" (AOV), and "all bits true" ABT. LST is set if the least significant or first bit out of the adder is a binary 1 and reset if 0. MST is set if the most significant last bit or eighth bit is a 1 and reset if 0. If all bits out of the adder are binary 1, ABT is set and otherwise reset. AOV true indicates that an overflow has taken place in an addition and it is preset or reset in a logic operation.

The Memory Control Unit MU

The memory control unit consists of two 8-bit registers and a selector, the Microprogram Count Register (MPCR) and the Alternate Microprogram Count Register (AMPCR). The MPCR is an 8-bit counter that can be incremented by one or two. The AMPCR is used to store jump addresses for changing the sequence of instructions. The MPCR is used to select the next instruction (successor) from the microprogram memory.

The Microprogram Memory

The Memory contains 256 12-bit words. The memory contains only executable instructions and cannot be changed under program control if a Read Only Memory is used. The 12 bits of an instruction are decoded into four types: literal, condition, logical, and external (DEV). Eight of the 12 bits can be transferred directly into the AMPCR or into the B register.

Control Unit CU

The control unit provides eight testable conditions, condition selection logic, successor determination, instruction decoding logic and timing for the processor. The eight conditions which may be tested are AOV, MST, LST, ABT, (which have already been mentioned), plus 3 local conditions set or reset by the program, LC1, LC2, LC3, and external asynchronous condition EXT. The successor selection is either MPCR+1, MPCR+2, or AMPCR, which are also called STEP, SKIP, and JUMP, respectively. The microprocessor uses an external clock line for timing. During each instruction, eight counts are made. After the eighth count the microprocessor waits for a Memory Cycle Complete (MCC) pulse before starting the next instruction. The CU also provides two outgoing clock pulses. One is an 8-count clock signal Clock Out (CO) synchronous

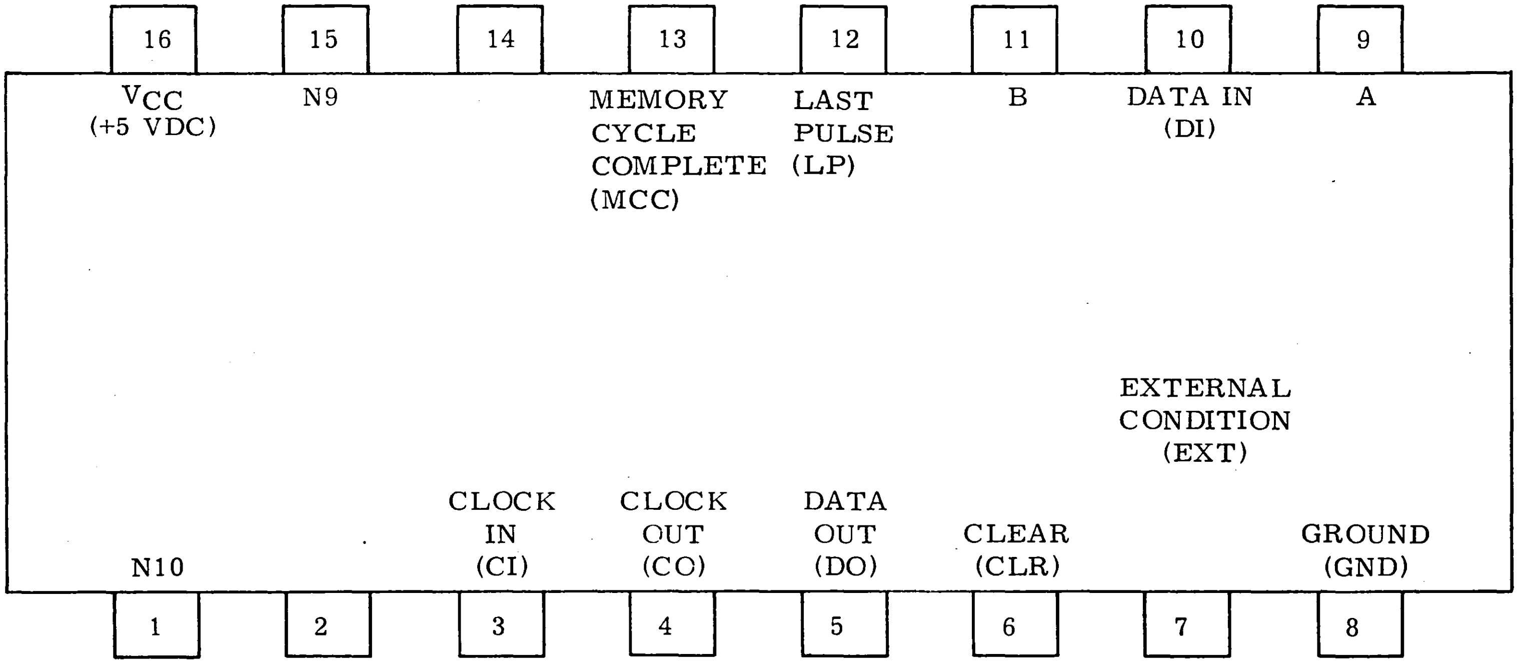


Figure 2-5. 16-Pin Interface and MOS-LSI Dip Chip Pin Configuration

with the instruction clock. The other is a signal the marks the end of each instruction called Last Pulse (LP).

External Interface

This is the interface that connects the microprocessor to the outside world. The connection is synchronized by the CLOCK OUT signal described above. An external asynchronous input level EXT is available to obtain the attention of the microprocessor. The interface is as in Figure 2-5.

Pins 2 and 14 are reserved for voltage connections in the LSI-MOS version. Signals N9, N10, A, and B are external control lines used to aid in the flow of information into and out of the processor. Signals A and B tell the outside world what type of instruction the Mini-D is executing.

A	B	
0	0	No externally significant instruction being executed
0	1	"BEX" instruction (data input requested)
1	0	"OUT." instruction (data output available)
1	1	"DEV" instruction (a memory transfer)

Signals N9 and N10 indicate to the outside wordd, which register, of OUT0, OUT1, OUT2, or OUT3 is specified during a Logic Unit "OUT" instruction. N9 and N1 0 are actually the 9th and 10th bits of the instruction word.

The CLR signal is an input used to clear the MPCR to zero address. LP, as described above marks the end of each instruction. Data is fed into the B register during a BEX-type logic instruction serially by way of the Data-In (DI) line. Output from the Microprocessor come by way of the Data Out line (DO); this line carries the output from the adder during all logic instructions, and a literal during the DEV instruction, otherwise the signal is undetermined and constant. Information is sensed by the microprocessor on the trailing (negative going) edge of the Clock Out pulse and likewise the Data Out (DO) signal should be sensed by the DDP on this edge. Clock In (CI) and Memory Cycle Complete (MCC) are the two timing signals that must be supplied external to the microprocessor itself. Clock In is the high speed clock input connection. MCC is a pulse that "initiates" the instruction cycle.

To provide for teletypewriter input on the Data In (DI) line, the DI signal from the EXI and the TTY signal had to be gated for selection. To accomplish this pin 2 is used as an ungated DI input and pin 10 is the selected or gated DI signal. Therefore to provide the 16-pin EXI as shown in Figure 2-5 it was necessary to compensate by plugging the interface cable from MP-2 into a socket on UB-1 and wiring over to an adjacent socket, pin for pin, except that pin 2 from the interface cable connector connects to pin 10 of the user's interface socket. This two socket arrangement is wired twice on UB-1. Once with the internal clock signals also wired in and once without. In this way the user can disconnect the supplied clock system and use his own merely by moving the cable connection fro m one socket to another (See Figure 2-6).

In order to allow operation of the MIG at speeds compatible with either TTL or MOS MOS versions of the microprocessor there is a "divide by ten" circuit provided as part of the clock circuit on UB-1. The speed selection is made by rotating the connector plug 180° (Figure 2-6).

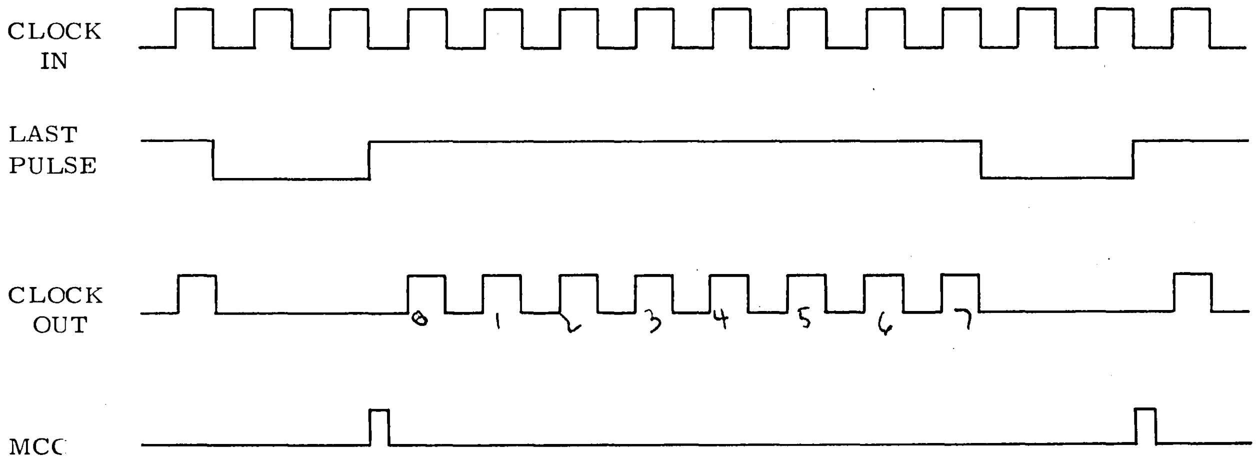


Figure 2-6. Timing in the Mini-D

THE PORT SELECT UNIT

In order to aid the user in constructing DDP's the design of the MIG includes four separate 16-pin interfaces and ports that can be selected by the DEV instruction. Bits 1 and 2 of the literal transmitted "open" a given port and keep it open until another DEV instruction opens a different port.

Common to each of the four ports are pins 1 through 7 which carry these signals: B, N9, $\overline{\text{N9}}$, N10, $\overline{\text{N10}}$, A B, A $\overline{\text{B}}$. Also common to each are the DO signal and the EXT signal on lines 15 and 16 respectively. The signals that are selectively transmitted, and therefore constitute an "open" port are MCC on pin 10, LP on pin 12, CO on pin 13 and D.I. on pin 14.

INSTRUCTION SET

There are four types of instructions; literal, condition, logical and external. Literal instructions bring 8 bits of info into the AMPCR or B-regsiter. Condition instruction test one of eight conditions and change successors accordingly. Logic instructions handle data in the registers and operate on the eight bit strings. External (DEV) instructions are literal instructions but with a difference; these instructions send literals, via DO, to external devices.

Literals Assignment Instruction

Formats:

1 2 3 4 5 6 7 8	9	10	11	12
Literal From MEM to B	1	0	1	1

1 2 3 4 5 6 7 8	9 10 11	12
Literal Jump Address to AMPCR	Not Used 0	0
Literal Address to MPCR and AMPCR	Not Used 1	0

Literal assignment instuctions contain a literal in the first 8 bits that is to be transferred to the register specified by the last 2 bits. In executing a "Literal to B" instruction the input bits are complemented in the process of loading into B. This is not the case when loading into the AMPCR. But there are two options available. If bits 11 and 12 are 0's then the literal is loaded into the AMPCR, but if bit 11 is 1 and 12 is 0 the literal is loaded into the AMPCR and into the MPCR, with the result that the next instruction executed is the one at the address specified by the literal. Bits 9 and 10 are available for memory extension by way of paging memories.

Condition Test Instruction

Format:

1 2 3	4 5	6 7	8 9	10 11 12
Condition	Set	True	False	1 1 1

Command code

False successor

00	Jump
01	Step
10	Skip
11	Save

True successor

00	Jump
01	Step
10	Skip
11	Save

Set operation

00	Set LC1
01	Set LC2
10	Set LC3
11	None

Condition select

000	MST
001	AOV
010	LST
011	ABT
100	LC1
101	LC2
110	LC3
111	EXT

This instruction performs a test on one of eight conditions (specified by bits 1, 2, 3). If the condition is true then the true successor (bits 6, 7) determines the next instruction. If the condition is false, then the false successor (bits 8, 9) determines the next instruction. If the condition is true, then in addition to the true successor selection, the set field (bits 4, 5) is checked to determine if a local condition is to be set.

Condition

The setting and resetting of the local condition is shown in Table 2-2. As indicated, the local condition bits (LC1, LC2, LC3) are reset on testing, and the set operation is used to set a local condition. It should be noted that it is necessary to test a true condition to be able to set a local condition. The external (EXT) condition bit is completely controlled by the external interface and usually the OR of the interrupts from several devices gated by their respective device addresses or it can be used for timing purposes. The four adder conditions (LST, MST, ABT, AOV) indicate the result from the last logic unit instruction. These conditions are not reset by testing and are sustained until execution of another logic unit instruction.

Table 2-2. Set and Reset of Conditions

Condition	Set	Reset
LC1	Set LC1	Reset by testing
LC2	Set LC2	Reset by testing
LC3	Set LC3	Reset by testing
EXT	A level from external devices-controlled by external interface (usually the OR of interrupts from several devices)	Reset by testing
LST	First bit from adder (least significant bit true - bit 8=1	*
MST	Last bit from adder (most significant bit true - bit 1=1	*
ABT	All bits true from adder (bits 1 through 8 are all ones)	*
AOV	Adder overflow true (This is really the carry bit for the serial adder; when eight bits of information have been serially added, it represents the overflow bit.)	*

*Changed only by logic unit instructions.

A literal assignment instruction loading the B register or AMPCR may change the value of an adder input, but this will not change the value of any of these conditions. Several logic unit operations have unusual side effects on these adder conditions, as explained in greater detail in "Logic Unit Instruction."

Successors

The two successors (true - bits 6, 7 and false - bits 8, 9) must be explicitly selected to determine the next instruction to be executed. Uncondition successors must have the same successor selected in both true and false field. The choices for each successor are (Table 2-3):

STEP Step to the next instruction in sequence from MPCR.

SKIP Skip to the second next instruction in sequence from MPCR.

SAVE Step and save current MPCR address +1 in AMPCR.

JUMP Transfer control to AMPCR address.

All other microinstructions have an implicit successor of STEP.

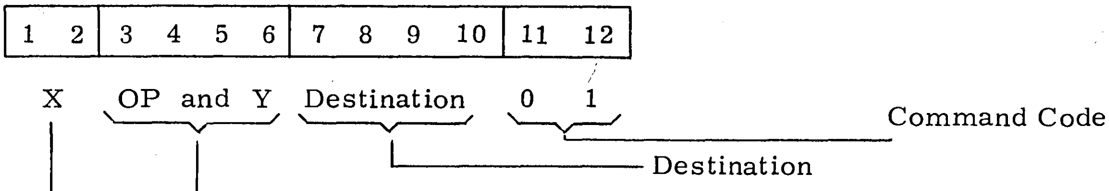
Table 2-3. Microprogram Memory Addressing

Sucessor Command	Next Instruction Address	Next Content of MPCR	Next Content of AMPCR
STEP	MPCR+1	MPCR+1	**
SKIP	MPCR+2	MPCR+2	**
SAVE	MPCR+1	MPCR+1	MPCR+1
JUMP	AMPCR	AMPCR	**

** Not changed by successor specification

Logic Unit Instruction

Format



Operation and Y Select[**]

0000	X+B+1
0001	X+B
0010[*]	X+Z+1
0011[*]	X+Z
0100	X EQV B $(XB \vee \overline{X}\overline{B})$
0101	X XOR B $(X\overline{B} \vee \overline{X}B)$
0110	X - B $(X+\overline{B}+1)$
?111	X-B-1 $(X+\overline{B})$
1000	X NOR B $\overline{(X \vee B)}$
1001	X NAN B $(\overline{XB})$
1010[*]	X NOR Z $(\overline{X \vee Z})$
1011[*]	X NAN Z $(\overline{XZ})$
1100	X OR B $(X \vee B)$
1101	X AND B (XB)
1110	X RIM B $(X \vee \overline{B})$
1111	X NIM B $(X\overline{B})$

0000	B
0001	A1
0010	A2
0011	A3
0100	OUT 0
0101	OUT 1
0110	OUT 2
0111	OUT 3, AMPCR
1000 #	B, BEX
1001 #	A1, BEX
1010 #	A2, BEX
1011 #	A3, BEX
1100 ##	B S
1101 ##	A1, S
1110 ##	A2, S
1111 ##	A3, S

X Select

00	0
01	A1
10	A2
11	A3

[*] Z = AMPCR. When AMPCR is not selected as a destination, the AMPCR will be "zero" (i.e., Z = 0) in all operations as a Y select input.

[**] Y select = B or Z as indicated

[#] "BEX" indicates serial transfer from an external register to B register via D.I. while adder transfers to other specified register (if B, then two inputs are ORed).

[##] "S" indicates a one-bit right shift of the destination register end off, with the MSB being filled by the adder output.

Definition of Logic Unit Instruction

The logic unit instruction specifies the adder inputs, the operation and the destination specifications for the adder. The X select to the input of the adder is either zero or one of the three A registers (specified by bits 1, 2). The operation and Y select to the input of the adder are specified by bits 3, 4, 5, 6 and include both arithmetic and logic operations on both the AMPCR and B register as indicated. The destinations of the adder output as shown are specified by bits 7, 8, 9, 10. The output of the adder can go to B, A1, A2, A3 or AMPCR. The adder output always goes ungated to the external interface, when a logic operation is selected, but if OUT0, OUT1, OUT2 or OUT3 is selected as a destination, then a special 4-bit code is generated on the external control lines (as explained in "External Interface") to enable gating from the adder to a specific external register. Note, if any of the "BEX" destinations are selected, a 2-bit BEX code is sent out on the external control lines enabling an 8-bit serial transfer from the external DATA IN register to the B register to take place in parallel with the adder output into the specified register (i.e., A1, A2, A3, B). If the destination register is "B, BEX", then an OR of the adder output and the external input is performed. Normally, the adder output in this case would be set to transfer zeroes from the adder, thereby allowing a simple external load of the B register. It should be noted that the use of OUT3 will alter the AMPCR and an AMPCR destination is the same as OUT3. As noted by "*", if the AMPCR is not selected as the destination register, then the four operations using AMPCR as a Y select will have "zero" for a Y input. This means operations using AMPCR as a Y select can only be transferred back to AMPCR. Through the use of this feature 0, not 0, X and not X can be transferred to any destination register except the AMPCR.

The destinations with the "S" for SHIFT allow the destinations to be shifted right by one bit, and the most significant bit is supplied by the adder operating on the least significant bit of the X and Y selected operands. It should be noted, that the adder operation is performed on all eight bits of the input operands, and the adder condition bits (LST, MST, ABT, AOV) are set accordingly.

If one wishes to perform a right shift (end off) of one bit on the B destination, then select (X=0, X+Z, B, S) for the instruction.

If one wishes to perform a circular shift of one bit on the B destination, then select (X=0, X+B, B , S) for the instruction. The primary purpose of the shift of the destination is to achieve right and circular shifts on A1, A2, A3 and B, but all other allowed functions are valid into the destination's MSB. It is also interesting to observe, if (X=A1, X+B, A1 , S) instruction is used, that the addition takes place on the bit 8 of both A and B, and the resulting bit is placed into bit 1 (MSB) of A1. Thereafter, bit & (LSB+1) of A1 is added to all bits of B, and the side effects on the adder condition bits result accordingly. The last interesting side effect of a serial implementation of the adder is that the adder overflow (AOV) condition is really the initial and intermediate carry flip-flop for the serial adder. As such, whenever a +1 operation is called for, the initial carry is set. In fact, the initial carry is set whenever bit 6 of the OP-Y select field is zero. However, the initial carry flip-flop is enabled for intermediate carries only on arithmetic functions. For example, on X OR B operation, bit 6 is zero, therefore AOV is set and remains set until a subsequent logic unit operation changes it.

External Instruction

Format

1	2	3	4	5	6	7	8	9	10	11	12
Literal & Device								0	0	1	1

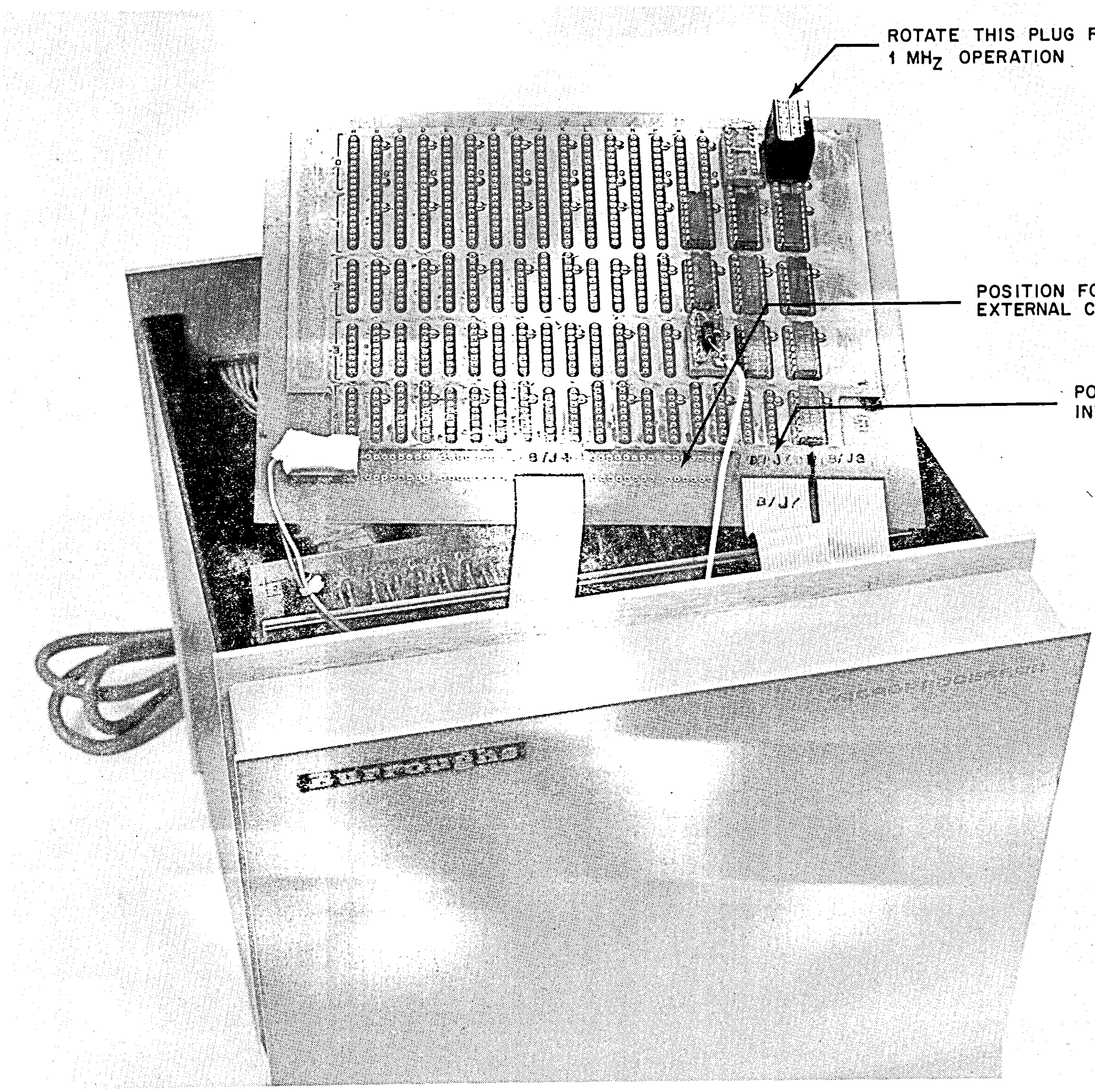
The external instruction, also called a DEV instruction, takes the first eight bits of the word and sends them serially out on the DO line (bit 8 first). This instruction has only the use that the programmer and DDP designer give it with respect to outside devices.

The coding of the function specified by the literal to the device on the outside, and the design of that device's hardware should be done in parallel in order to minimize the hardware expense and maximize program efficiency.

TIMING IN THE MINI-D

Timing in the Mini-D is controlled by a clock external to the Mini-D itself. The instruction time is eight clock pulses at the end of which the Mini-D produces the LP signal and then waits for an MCC pulse to begin the next instruction in the MIG. As shown in Figure 2-7, the MCC is synchronized in the 10th clock pulse. The waiting time between instructions is for memory cycling and instruction decoding.

The user has the option of using the clock supplied with the MIG or supplying his own. The 10-MHz clock supplied makes the basic instruction time 800 ns plus 200 ns for memory cycle and instruction decoding. By reversing an adapter on UB-1, the basic instruction time is increased to 8 μs plus 2 μs allowed for memory cycle and instruction coding. (See Figure 2-6). This is done by selecting either the output of the 10 MHz clock oscillator or the output of a divide by ten counter. The load mode requires that the instruction time plus memory cycle time total 9 μs; due to the real-time bootstrap program that is used to load from the teletype. This 9-μs total instruction time is automatically selected when the load switch is flipped to the load position. The MIG's clock system also provides the MCC pulse used to start the instruction.



ROTATE THIS PLUG FOR 1 MH_Z OPERATION

POSITION FOR EXTERNAL CLOCKING

POSITION FOR INTERNAL CLOCKING

Figure 2-7. Timing in the Mini-D

3. OPERATION AND USE

The Burroughs Microprocessor Machine Image Generator is a laboratory instrument which aids in the design of device dependent I/O ports (DDP's) and for the debugging of microcode. With this in mind, provision has been made for easy connection with other devices and for testing DDP's in the MIG itself. All functions are controlled from the front panel and all connections are made at the rear or through a cutout in the back of the cabinet.

FRONT PANEL

The Front Control Panel has four switches and 20 indicator lights. The two toggle switches are the RUN/STEP switch and the LOAD/RUN switch. Two pushbutton switches are START and CLEAR.

Switch	Function
RUN/STEP	In the run position the microprocessor runs under clock control. In the step position a single instruction is executed every time the START button is pressed.
LOAD/RUN	In the LOAD position the microprocessor is under the control of the bootstrap program and will load the R/W memory from the teletypewriter. In the RUN position, the microprocessor is under program control. In transition to the LOAD mode the machine will halt.

Switch — Function

START — With RUN/STEP in the step position this button is used to cause the execution of a single instruction. If R/S is in the RUN position it causes a resumption of processing after a halt.

CLEAR — In any mode causes the processor to halt and clears MPCR to zero.

NOTE: Since RUN/STEP and START are functionally part of the clock system they function only when the Mini-D runs on its internal clock.

There are two rows of lights that display information to the user. The top row is the bit configuration of the micromemory word to be executed next (i.e. at location specified in MPCR). The bottom row is the address of that word.

THE REAR PANEL

The rear panel of the MIG is designed to provide flexibility in connecting devices to the MIG. Eight Cinch-Jones 25-pin sockets are on the right of the panel (rear view). On the right is the power switch and 110-VAC power cord. At the bottom are (from left to right) a fuse (3A), the fused +5V terminal, a ground connection, a phone jack for teletype connection, and a chassis ground. Top-center is an insulated slot.

The eight Cinch-Jones 25-pin connectors and the insulated slot are for connecting from the MIG to other devices. The slot is to allow easy access with flat ribbon cables with D.I.L. plugs to the U.B's. The Cinch-Jones connectors allow for a more rugged connection.

Power can be drawn from the MIG at the +5V connector, fused at 3A. Two ground connections are also on the rear. One is paired with the +5V for power; the other is a chassis ground terminal. Both connections can be used as a scope ground.

THE UTILITY BOARDS

Two utility boards are supplied with each MIG for use in building DDP's. Burroughs recommends the use of the 14 XA2 Gardner-Denver wire-wrap tool with the 506445 bit and 500350 sleeve and a 26-gauge wire.

About one and three quarters of a board is available for use, including about 80 I.C. chip positions. One board is partially populated with the clock, and power connections. It is suggested that the user build his DDP's on these cards and connect outside devices via the eight Cinch-Jones connectors and/or flat ribbon cables.

POSSIBLE MIG CONFIGURATIONS AND USES

The Burroughs Microprocessor Machine Image Generator is designed to be as flexible as possible in application to a given design problem. The MIG includes a clock oscillator and timing circuit and a power supply, both of which can be used for outside devices. However, after DDP's have been developed it would be desirable to use the microprocessor in the final configuration. Toward this end the MIG has been designed so that the microprocessor can be easily disconnected from everything internal to the MIG but power (since removal of power alters the memory).

LOADING

The Burroughs Microprocessor MIG is equipped with a bootstrap loader program that is contained in a 32-word X 12-bit read only memory. The microprocessor immediately switches to the bootstrap when the RUN/LOAD switch is changed to the LOAD position. The bootstrap program as shown in Table 3-1 is a real-time program designed to load the first four bits after the start space of a teletype or equivalent device. Such a device must have a 110 bits/sec. send rate as per Model 33 teletypewriter.

Loading Procedure

The MIG loads hexadecimal characters one at a time. After every sixth character is presented to the Mini-D, the memory is loaded at the address specified by the first two characters, and the instruction loaded is contained in the next three characters. The sixth character, a period (or other convenient separator), is not loaded but is necessary. The character set used is given in Table 3-2.

In order to load, the TTY must be plugged into the jack in the rear of the MIG marked "TTY" and turned on. Then switch the RUN/LOAD control to the LOAD position and hit CLEAR then START. If the teletype is correctly installed the MIG will loop in address locations 04, 05, and 06. To load from paper tape or the keyboard the MIG must be preset by loading spaces or zeros, or holding BREAK key down until the MIG hangs up in address 1E. By hitting CLEAR and then START the MIG is ready to load. It is important that the first character read is the first character of the first instruction; i.e., no preceeding blanks.

Since no accommodations have been made to ignore carriage return and line feed as characters, this must be handled in a special way. This is done by preceding a carriage return by an unused address and the hit carriage return three times and then L.F. For example, the address / / (HEX FF or 255) could be used. When the last address has been loaded, depress CLEAR, switch to RUN and the MIG is ready to run. To run hit START.

Table 3-1. Microprocessor Bootstrap Loader Program

ADDRESS	NANO-WORD	INSTRUCTION	COMMENT
00	000	0 =: AMPCR	
01	0CD	0 =: A3	% CLEAR REGISTER
02	9FF	IF LC1 THEN SAVE ELSE SAVE	% RESET LC1, START LOOP
03	C4B	4/3B =: B	
04	F2D	A3 OR B =: A3, BEX	% SENSE START BIT
05	005	B + 1 =: A1	
06	3E7	IF AOV THEN SAVE ELSE JUMP	
07	485	A1 + 1 =: A1	% TIME OUT 1/4 BIT-TIME
08	1A7	IF MST ELSE JUMP	
09	C8D	A3 + 1 =: A3	% COUNT TO BIT CENTER
0A	CCB	4/33 =: B	
0B	E65	A3 NAN B =: A1, BEX	% COUNT FOUR BITS
0C	637	IF ABT THEN SET LC1 ELSE SKIP	
0D	079	B =: A2, S	
0E	0C5	0 =: A1	
0F	827	IF LC1 THEN SET LC1 ELSE JUMP	% CHAR. FIRST FOUR BITS
10	20B	4/DF =: B	
11	F01	A3 OR B =: B	
12	7A7	IF ABT ELSE JUMP	
13	020	4/02 =: AMPCR	
14	DCF	IF LC3 THEN SKIP	
15	707	IF ABT THEN SET LC3 JUMP ELSE JUMP	
16	3FB	4/C0 =: B	% COUNT SIX CHARACTERS
17	C4D	A3 + B =: A3	
18	337	IF AOV THEN SET LC3 ELSE SKIP	
19	8D1	A2 =: OUT0	% LOAD 2 CHARACTERS
1A	D8F	IF LC3 THEN JUMP	
1B	0CD	0 =: A3	
1C	8D5	A2 =: OUT1	% LOAD 2 CHAR AND WRITE
1D	7E7	IF ABT THEN SAVE ELSE JUMP	
→ 1E	187	JUMP	
1F	000	0 =: AMPCR	

Table 3-2. MIG Loader Character Set

Binary	Hexadecimal	Character	Bit Assignment
0000	0	0	011 0000
0001	1	1	011 0001
0010	2	2	011 0010
0011	3	3	011 0011
0100	4	4	011 0100
0101	5	5	011 0101
0110	6	6	011 0110
0111	7	7	011 0111
1000	8	8	011 1000
1001	9	9	011 1001
1010	A	J	100 1010
1011	B	K	100 1011
1100	C	L	100 1100
1101	D	M	100 1101
1110	E	N	100 1110
1111	F	/ (slash)	010 1111
	Delimiter	. (period)	010 1110

RUNNING

When running it may be convenient to use a single instruction mode for debugging purposes. With the STEP/RUN switch in the STEP position the START button will initiate a single instruction every time it is depressed. The control lights indicate the instruction that will be executed when the START button is depressed.

4. PROGRAMMING

To facilitate the microprocessor programming, Burroughs has developed an assembler. The source programs are compiled on the B 3500 computer and a listing and a teletype tape are produced as output. The contents of the tape can then be loaded into the microprocessor via the tape reader on the teletype.

The language used to program the MINI-D (MINI-X) has used ALGOL as a model. Unlike ALGOL almost all of the language is composed of reserved words, however, since the system designer must have complete control of all the Interpreter functions. Reserved words have very specific meaning to MINIX and cause specific microinstructions to be developed.

LANGUAGE DESCRIPTION CONVENTIONS

Backus-Naur form (BNF) is used as the metalanguage to define the syntax of MINIX. The following BNF symbols are used:

1. < > Left and Right Broken Brackets are used to bracket the names of syntactic categories.
2. ::= Colon Colon Equal means "is defined as" and separates the name of the syntactic category from its definition.
3. | Bar separates alternative definitions of a syntactic category.
4. { } Left and Right Braces enclose an English language description of a syntactic unit.
5. Juxtaposition of metalanguage symbols, symbols, or reserved words is used to indicate concatenation.

Any character or symbol in a metalanguage formula which is not a metalanguage symbol and is not enclosed within matching braces or broken brackets, denotes itself.

In addition, to express the language syntactically, this manual will use a modified COBOL normal notation. The square brackets, [ ] mean what is contained within is optional. The parentheses, ( ), mean one may pick one of the functions inside.

In the BNF terminology, the basic elements of TRANSLANG are as follows:

Letter ::= | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
R | S | T | U | V | W | X | Y | Z |

This would be written in English something like "An element of the syntactic category of letters is either A or B or C:

Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

< Hex Digit > ::= < Digit > A | B | C | D | E | F

< Symbol > ::= , | ; | + | - | : | = | % | " | (|) | *

< Single Space > ::= {One horizontal blank position}

< Space > ::= < Single Space > | < Space > < Single Space >

Note: "a space is any number of continuous single spaces."

< Assignment Op > ::=

< Character > ::= < Letter > | < Digit > | < Single Space > | < Symbol >

< Comment Character > ::= < Character > | . | # | & | $ | [|] | / | \

< Empty > ::= {The null string of characters}

< Comment > ::= {Any sequence of < Comment characters >
except ; } ;

vecause the semi-colon is the comment delimiter.

SEMANTICS

MINIX uses a character set of 56 characters including < Single space > of which 8 are only used in comments. All letters are upper case.

Space — No space may appear between the letters of a reserved word or within an < Assignment Op >; otherwise, they will be interpreted as two or more eleelements. Spaces are used as a delimiter to separate reserved words, labels, or integers. Spaces may appear between any two basic components without affecting their meaning, where basic components indicate words, symbols, or labels.

Parenthese — The parentheses are treated as spaces. They are used for the convenience of the microprogrammer to make code more readable. (e.g., instruction elements which are irrelevant to the current instruction but are used only to allow shared use of a nanoinstruction by several M-instructions.) Parentheses do not imply precedence.

Comments – In order to include explanatory material at various points in a program, two conventions exist as defined:

1. COMMENT { Any sequence of comment characters except; } ; The comment statement acts the same as a semi-colon and may appear anywhere a semi-colon may occur if within a line of program. As multi-line documentation the semi-colon terminator indicates that the microtranslator should resume processing code. Always follow a comment statement with a semi-colon.

2. % { any sequence of comment characters until the end of line } All comment characters after the % in a line of program are ignored by the microtranslator.

Comments are for documentation purposes only. They appear only in the source file, are significant only in listings and do not affect the machine language generated.

The following printing characters are used for control purposes and should not be used in comments.

< > ? ' ! ←

This control character is equivalent to the end of a card if card input is used to build a source file. It is not part of the character set processed by the microtranslator.

MINIX - TRANSLATOR

The translator for the Mini-D Interpreter is a one for one translation between source code and object code. Within the translator are five (5) classes of source statements expressed in a COBOL/ALGOL type syntax. These are:

1. PSEUDO statements
2. Literal Assignment Statements
3. Conditional Test Statements
4. Logic Unit Statements
5. External Service Statements

PSEUDO INSTRUCTIONS

This class of instructions provides for program identification, starting address assignments, comments, and the termination of the program being assembled.

Syntax

Column 8

```
        PROGRAM    PROGRAM NAME    (20 Characters maximum).

        ADR  4/xx

        END

        COMMENT (comments (any characters except ;) ;
```

Semantics

The statements defines four (4) pseudoinstructions. These instructions emit no microinstructions but are used purely to control the MINIX assembler and provide a few convenience features to the microprogrammer. Specifics on each pseudoinstruction are as follows:

PROGRAM – This instruction designates the name to be carried with the program throughout the assembly process. This must be the first card of the MINIX Translator.

ADR – This instruction provides the hexadecimal address where the programmer wishes the program to start in micromemory. 4/xx indicates that the hexadecimal address follows and xx are two hexadecimal characters. This card should be the second card in the MINIX program deck. If the card is omitted, MINIX will assign a starting address of HEX (00).

COMMENT. The comment card is transferred with the source code and is for notation within program at listing time. It must end with a semi-colon.

END. This instruction terminates the program. A file containing a source program must have a file name of 20 or less alphanumeric characters. Each record on this file contains 72 data characters (+8 for sequence numbers ignored by the microtranslator). One line of source program is written per record.

The first record is the program name. It contains the program internal name for the microprogram. The program internal name should be the saem as the file name. Only the file name has any external significance. A non empty start address becomes a hexadecimal absolute microprogram address.

The body of a program contains one or more lines. Following the body is the end line containing END. Each successive line containing an instruction normally becomes the next microaddress. Addresses increase strictly through a program. A start address less than the assembler's next address in the program sequence causes an error.

MICROINSTRUCTIONS

As mentioned, there are four major types of microinstructions for the microprocessor.

Syntax

```
              / Literal Assignment Instruction \
[Label:]     (  Condition Test Instruction      )     [% Comment]
             (  Logic Unit Instruction          )
              \ External Instruction           /
```

Examples

```
255    =: AMPCR                % LITERAL TYPE

IF AOX THEN SKIP ELSE STEP     % CONDITION TYPE

LOOP:  A1 + B =: A2            % LOGIC TYPE WITH  LABEL

A1 NOR B =: B

127 =: DEV                     % EXTERNAL TYPE
```

Semantics

There is a restriction of one instruction per input record. Each instruction, however, can contain a comment filling out the remainder of the record and any instruction can be labeled. The label notes to the assembler the address of this instruction in microprogram memory and can therefore be used in the literal assignment statement. For loading jump addresses a label must start with a letter which can be followed by any combination of letters or digits. No spaces or symbols may appear in a label. A label used in a program may be chosen freely except for the reserved words.

A label can be as little as one letter and as long as 15 letters and digits. The same label may not be used to locate more than one instruction in the same program.

Literal Assignment Instructions

This class of instructions allows specifications of varied source statements to be translated into 8 bits, which is subsequently transferred to the B register, or AMPCR at execution time in the MINI-D.

Syntax

Literal =: B

Literal =: AMPCR

GOTO Literal

where the literal definition is

$$\left(\begin{array}{l} \begin{matrix}[\text{COMP}]\\ [\ -\ ]\end{matrix} \quad \begin{pmatrix}\text{DECIMAL INTEGER}\\ 4/\text{xx}\end{pmatrix} \\ \qquad\qquad\quad \left(\left(\begin{matrix}[\text{Label}]\\ [*]\end{matrix}\right) \ [\pm\ \text{Integer}]\right) \end{array}\right)$$

That is there are two main options, namely

$$\begin{pmatrix}[\text{Comp}]\\ [-]\end{pmatrix}\begin{pmatrix}\text{Decimal Integer}\\ 4/\text{xx}\end{pmatrix}$$

and

$$\left(\left(\begin{array}{c}\left[\text{Label}\right]\\ \left[\ *\ \right]\end{array}\right)\qquad\left[\pm\ \text{Integer}\right]\right)$$

The first of these says a literal can either be a decimal Integer or a hex address

For example:

Decimal Integer

```
    255 =:  AMPCR
```

Hexedecimal Integer

```
    4/1F =:  B
```

In addition, COMP defines Ones complement be performed and "-" defines Twos complement be performed. For example:

```
COMP 255 =:    AMPCR          % ZERO INTO AMPCR

COMP 10  =:    B              % 1's COMPLEMENT

-10      =:    B              % 2's COMPLEMENT
```

The second of the main options indicates that a literal may be defined as a label alone, asterisk alone, label ± Integer or Asterisk ± Integer.

For example:

```
LOOP      =:   AMPCR          % LABEL

*  =:   B                     % INSTRUCTION ADDRESS TO B

*  +2 =:  B                   % PRESENT ADDRESS +2 TO B
```

Semantics

The Literal Assignment Instruction is used for: (1) loading jump and return addresses into the registers, (2) constants for loop control, (3) loading character codes for comparisons, (4) loading constants for general use.

The Label internal to the Literal Assignment Instruction is a program point or a label defined under the Pseudo Instruction.

The asterisk indicates assigning this program address (where the assembler is) to the 8-bit Literal field.

± Integer is used with the LABEL or * designation which causes the address to be incremented or decremented by the Integer.

The Decimal Integer is a decimal number to be converted and placed in the 8-bit literal field of the instruction. Thus, the integer number is restricted to 0 through 255. All codes that do not fill the entire eight bits will be generated right-justified. These constants are restricted by hardware to being loaded into the AMPCR or the B register. Note: When loading a literal into the B register the value specified is the value loaded.

4/XX represents a hexadecimal specification of the 2 characters to be inserted in the Literal Field of the instruction.

Condition Test Instructions

This class of instructions allows for the programmatic testing of eight (8) conditions, the setting of conditions, and the selection of true or false selectors.

Syntax

```
/Successor      [IF Condition]                                      \
\IF Condition     [THEN]  [SET Op]  True Succ  [ELSE False Succ]    /
```

where

Condition	Successor	Set Op
LST	STEP	SET LC1
MST	SKIP	SET LC2
AOV	SAVE	SET LC3
ABT	JUMP	
EXT		
LC1		
LC2		

Specifies on CONDITION TEST INSTRUCTIONS are as follows:

Successor

This conditional test instruction will set up a conditional test on the MST condition and in the micro code will assign the successor in both the true/false successor positions.

Examples:

```
STEP
SKIP
SAVE
JUMP
```

Successor IF Condition

This conditional test instruction allows the determined successor to be indentified as the true successor with an implied false successor of STEP, based on a conditional test. The syntax is shown below:

$$\left(\begin{array}{l}\text{STEP}\\\text{SKIP}\\\text{SAVE}\\\text{JUMP}\end{array}\right)\ \text{IF}\ \left(\begin{array}{l}\text{LST}\\\text{MST}\\\text{AOV}\\\text{ABT}\\\text{EXT}\\\text{LC1}\\\text{LC2}\\\text{LC3}\end{array}\right)$$

Some examples are:

```
JUMP    IF    LC2                  % JUMP AND RESET LC2

SKIP    IF    ABT                  % IF ALL BITS TRUE, SKIP ELSE STEP
```

IF Condition THEN Successor

This conditional test instruction allows the testing of a condition with the true and false successors specified on the outcome of the test. The syntax is:

$$\text{IF} \quad \begin{pmatrix}\text{LST}\\ \text{MST}\\ \text{AOV}\\ \text{ABT}\\ \text{EXT}\\ \text{LC1}\\ \text{LC2}\\ \text{LC3}\end{pmatrix} \quad \text{THEN} \quad \begin{pmatrix}\text{STEP}\\ \text{SKIP}\\ \text{SAVE}\\ \text{JUMP}\end{pmatrix} \quad \text{ELSE} \quad \begin{pmatrix}\text{STEP}\\ \text{SKIP}\\ \text{SAVE}\\ \text{JUMP}\end{pmatrix}$$

In cases where ELSE False-succ is omitted STEP will be implied and will be inserted in the microinstruction. Some examples are:

```
IF LC2                              % RESET LC2, IMPLIES STEP

IF ABT THEN JUMP ELSE STEP          % CONDITION BRANCH TRUE

IF ABT THEN STEP ELSE JUMP          % CONDITION BRANCH FALSE

IF ABT THEN SKIP                    % IMPLIED ELSE STEP
```

IF Condition THEN Set

This conditional test instruction allows the testing of a condition, the setting of a local condition, and the selection of the true and false successors. The syntax is:

$$\text{IF} \quad \underset{\text{Condition}}{\begin{pmatrix}\text{LST}\\ \text{MST}\\ \text{AOV}\\ \text{ABT}\\ \text{EXT}\\ \text{LC1}\\ \text{LC2}\\ \text{LC3}\end{pmatrix}} \quad \text{THEN} \quad \underset{\substack{\text{Optional}\\ \text{Set Op}}}{\begin{pmatrix}\text{SET LC1}\\ \text{SET LC2}\\ \text{SET LC3}\end{pmatrix}} \; \underset{\substack{\text{True}\\ \text{Successor}}}{\left[\begin{pmatrix}\text{STEP}\\ \text{SKIP}\\ \text{SAVE}\\ \text{JUMP}\end{pmatrix}\right]} \text{ELSE} \; \underset{\substack{\text{False}\\ \text{Successor}}}{\left[\begin{pmatrix}\text{STEP}\\ \text{SKIP}\\ \text{SAVE}\\ \text{JUMP}\end{pmatrix}\right]}$$

Some examples are:

IF ABT THEN SET LC1 STEP ELSE SKIP

IF AOV THEN SET LC2 ELSE STEP

The Condition Test Instructions are used for one or a combination of the following purposes: conditional or unconditional transfer of control, and setting and/or resetting local condition bits. The eight conditions consist of four adder conditions (least bit - LST, most bit -, MSR, overflow - AOV, all bits true - ABT), an external condition (EXT) and three local conditions (LC1, LC2, LC3). Note: setting of a local condition is possible only if a condition test is true.

The condition instruction specifies a true and false successor explicitly or implicitly, indicating the control to be used for the next instruction selection. A successor of the unconditional type results in both successors being identical. Otherwise, one or two successors may appear in the conditional type. The four choices for each successor are:

STEP	Step to the next instruction
SKIP	Skip to the second next instruction
SAVE	Step and save present addresses +1 in AMPCR
JUMP	Transfer control to AMPCR address

Any successor not explicitly stated in STEP by default. All other microinstructions have an implicit successor of STEP. Note the AMPCR normally contains the address of an alternative instruction.

Logic Unit Instructions

This class of statements allows for the performance of adder and logical operations. Within this class are 4 groups: Shifts, Adder 1 OP's, Adder 2 OP's, and Adder 3 OP's.

Syntax

$$\left(\begin{array}{lcl} \begin{pmatrix}\text{X Select}\\ \text{B}\end{pmatrix} & & \text{(Shift)} \\ \begin{pmatrix}\text{Adder}\\ \text{Op 1}\end{pmatrix} & =: & \begin{pmatrix}\text{(Destination}\\ \text{AMPCR}\end{pmatrix} \\ \begin{pmatrix}\text{Adder}\\ \text{Op 2}\end{pmatrix} & =: & \text{AMPCR} \\ \begin{pmatrix}\text{Adder}\\ \text{Op 3}\end{pmatrix} & =: & \text{Destination} \end{array}\right)$$

Specifics on Logic Unit Instructions are as follows:

Shift

This Logic Unit Instruction will allow the selected register to be shifted right (R), or right circular (C). The syntax is:

(Register)	(Shift)
A1	R
A2	C
A3	
B	

Some examples are:

```
A1 R                 % RIGHT SHIFT 1 BIT, 0 FILL

B  C                 % CIRCULAR SHIFT  1 BIT
```

Adder Op 1

This Logic Unit Instruction has a select constant of B, and destinations allowed are all registers and the AMPCR. The syntax is:

$$\begin{pmatrix}\text{(Adder OP1}\end{pmatrix} \quad =: \quad \begin{pmatrix}\text{Destination}\\ \text{AMPCR}\end{pmatrix}$$

$$\begin{array}{lcl} \begin{array}{r} X + B\\ B\\ X + B + 1\\ B + 1\end{array} & \qquad \begin{pmatrix}\text{A1}\\ \text{A2}\\ \text{A3}\\ \text{B}\end{pmatrix} & \qquad \begin{array}{l} {[\text{BEX}]}\\ {[\text{S}]}\\ \\ \\ \end{array} \end{array}$$

X - B	AMPCR
- B	OUT 0
X -- B - 1	OUT1
X NOR B	OUT2
X NIM B	OUT 3
X AND B	
X NOR B	
X EQV B	
X NAN B	
X OR B	
X RIM B	
NOT B	
NOT 0	
0	

where X = (0 or A1 or A2 or A3)

and where

Operation	Definition		
NOR	$\overline{X}v\overline{B}$		
NIM	$X\overline{B}$	-1	NOT 0
AND	XB		
XOR	$X\overline{B}\ v\ \overline{X}\overline{B}$		
EQV	$XB\ v\ \overline{X}\overline{B}$		
NAN	$\overline{XB}$		
OR	XvB		
RIM	$X\ v\ \overline{B}$		

Some examples are:

```
A1 + B =: AMPCR         % LOAD AMPCR
B      =: AMPCR         % LOAD AMPCR
A1 EQV B =:  B          % EQUIVALENCE
NOT B =:  B             % 1's COMPLEMENT B
```

Adder OP 2

This Logic Unit Instruction has a Y select constant of AMPCR, and the destination is restricted to AMPCR. The syntax is:

(Adder OP 2)	=:	AMPCR
X + AMPCR		
AMPCR		
X + AMPCR + 1		
AMPCR + 1		
X NOR AMPCR		
X NAN AMPCR		
NOT AMPCR		

Some examples are:

```
AMPCR + 1 =: AMPCR          % AMPCR CAN ONLY BE
A1 + AMPCR =: AMPCR         % TRANSFERRED INTO AMPCR
```

Adder OP 3

The Logic Unit Instruction has an X designation with no Y selection and a destination of everyone but AMPCR. The syntax is:

(Adder OP 3)	=:	(Destination)	(Option)
X		A1	BEX
X + 1		A2	S
NOT X		A3	
1		B	
		OUT0	
		OUT1	
		OUT2	
		OUT3	

Where X = (A1 or A2 or A3 or 0)

Some examples are:

```
A1 =:    A2 BEX     % ADDER TO A2, EXTERNAL TO B

 0 =:    B BEX      % EXTERNAL TO B

A1 =:    OUT 1      % A1 TO EXTERNAL OUT 1
```

The logic operations include the selection of adder inputs, the adder operation, and the destination specifications for the adder. There are three A registers (A1, A2, A3) which may be used for data storage within an Interpreter. Any one of the A registers may be selected to the Adder in the X select part of the instruction. The B register is the primary interface for external inputs from external data memory or devices.

The destination of adder OP 1 operations can be A1, A2, A3 or B. The 8-bit serial load of B register from external register via "BEX" command is possible on these destinations. Other adder destinations are AMPCR or the external output registers (OUT0, OUT1, OUT2, and OUT3) on the destination, then this register is shifted one place to the right and the least significant bit is lost while the most significant bit is loaded from the adder.

The adder 2 operator can be applied to any X select and the AMPCR. Note if AMPCR is used as an operand, then AMPCR must be selected as the destination. Therefore, AMPCR can not be transferred to anywhere except to the AMPCR. The AMPCR and OUT3 are changed simultaneous when either is specified.

The adder OP3 acts on any X select and has a restriction of the AMPCR not being selected as a destination. "NOT" select implies the 1's complement of content of the X select register. A shift right "R" or circular "C" one bit can be applied to A1, A2, A3 and B registers. A shift right "R" implies a zero fill left most bit (bit 1).

External Instructions

This class of instructions allows for the performance of operation outside the Mini-D Machine and presently is structured as follows:

```
Literal =: DEV

Literal - As defined previously
```

SUMMARY

Figure 4-1 provides a summary of the various types of microinstructions while figure 4-2 is a printout of the translator locations, codes and source statement.

MICROINSTRUCTION

```
          ( LITERAL ASSIGNMENT INSTRUCTION )
[Label:]  ( CONDITION TEST INSTRUCTION     )  [% Comment]
          ( LOGIC UNIT INSTRUCTION         )
          ( EXTERNAL INSTRUCTION           )
```

NOTES :

**	Indicates Zero or More Repetitions
( )	Indicates a Choice
[ ]	Indicates Optionally Present

LITERAL ASSIGNMENT INSTRUCTION

```
( Literal =: B     )
( Literal =: AMPCR )
( GOTO Literal     )
```

```
              Label
------------------------------
Letter  ( Digit  )**
        ( Letter )
```

```
                         Literal
---------------------------------------------------
( [COMP] ) ( (Label) [± Integer] )
( [ - ]  ) (    *                )
           ( Decimal Integer     )
           ( 1/ BINARY           )
           ( 3/ OCTAL            )
           ( 4/ HEX              )
           ( B/ BCL              )
           ( A/ ASCII            )
           ( E/ EBCDIC           )
```

PSEUDO INSTRUCTIONS

```
( ( PROGRAM     Program Name     ) )
( ( [Label:]  INSERT File Name   ) )
(   ADR     Hex Address            )
(   END                            )
(   Label  *  ( Decimal Integer )  )
(             (     4/ HEX      )  )
(   COMMENT   Comment ;            )
```

CONDITIONAL TEST INSTRUCTION

```
( Successor [; If Condition [Then Set Op]]                                                  )
( If Condition [Then ( [Set Op;] True Successor   ) [Else False Successor]]                  )
(                    ( Set Op [; True Successor]  )                                         )
```

Condition	Successors
LST	STEP
MST	SKIP
AOV	SAVE
ABT	JUMP
EXT	Set Op.
LC1	
LC2	SET LC1
LC3	SET LC2
	SET LC3

NUMERIC COMPARE OPERATIONS

Relation	Logic Unit Instruction:	Test True	Test False
$x < y$	$x - y$		AOV
$x \leq y$	$x - y - 1$		AOV
$x \neq y$	x EQV y		ABT
$x = y$	x EQV y	ABT	
$x \geq y$	$x - y$	AOV	
$x > y$	$x - y - 1$	AOV	

AFTER SUCCESSORS

	MPCR	AMPCR
STEP	+1	-
SKIP	+2	-
SAVE	+1	MPCR+1
JUMP	AMPCR	-

LOGIC UNIT INSTRUCTION

```
( ( x Select ) ( Shift )                       )
( ( B        )                                 )
( ( Adder ) =: [ Destination    ]              )
( ( Op 1  )    [ AMPCR[,OUT3]   ]              )
( ( Adder ) =: ( AMPCR[,OUT3] )                )
( ( Op 2  )                                    )
( ( Adder ) =: [Destination]                   )
( ( Op 3  )                                    )
```

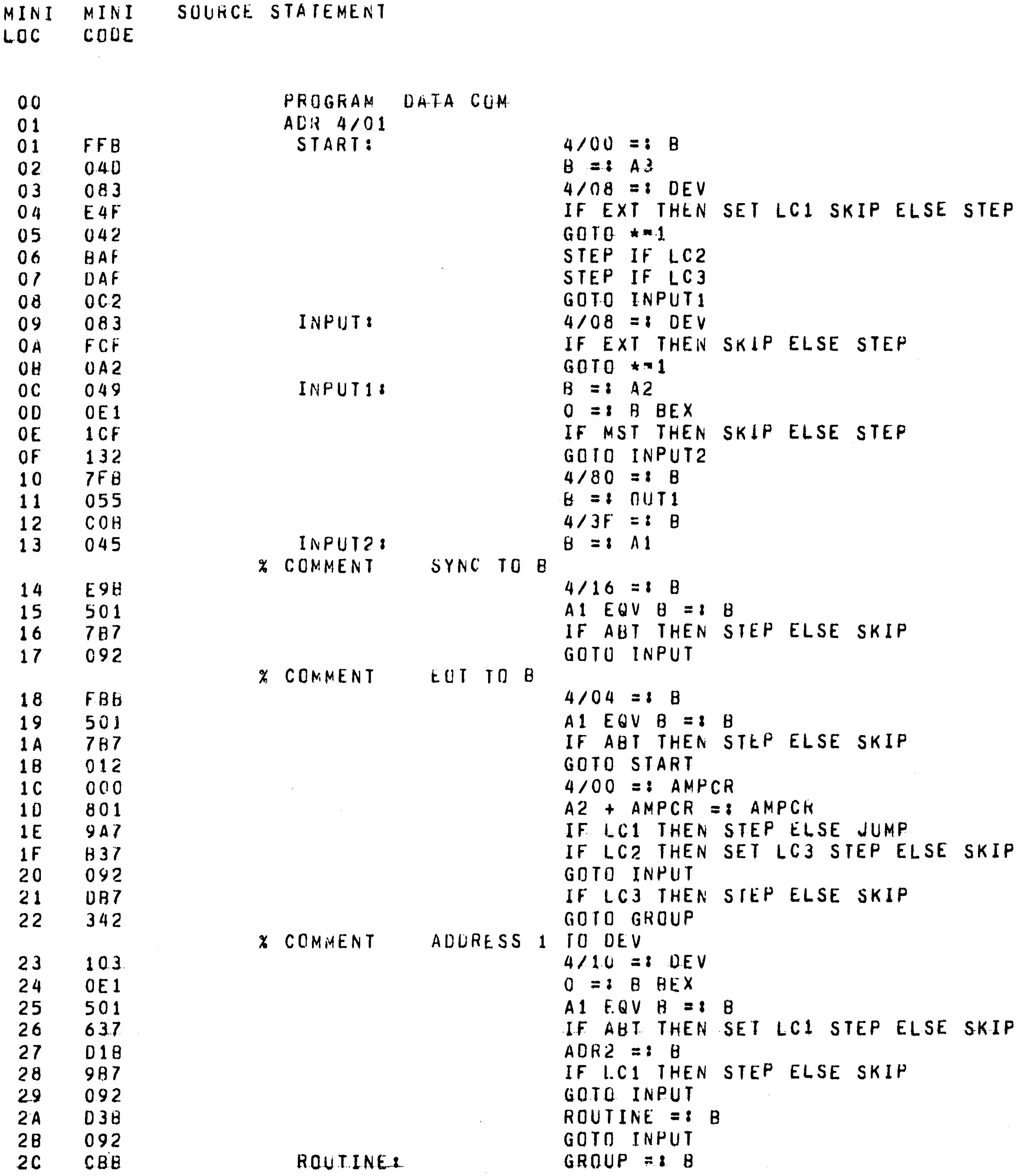
x Select	Shift	Adder Op 1 y = B	Adder Op 2 y = AMPCR	Adder Op 3	Destination
0	R	x + y	x + y	x	A1 ([,BEX])
A1	C	y	y	x + 1	A2 ([S])
A2		x + y + 1	x + y + 1	NOT x	A3
A3		y + 1	y + 1	1	B
		x − y	x NOR y		
		− y	x NAN y		OUT 0
		x − y − 1	NOT y		OUT 1
		− 1	NOT 0		OUT 2
		x NOR y	0		
		x NIM y			
		x AND y			
		x XOR y			
		x EQV y			
		x NAN y			
		x OR y			
		x RIM y			
		NOT y			

Operation	Definitions
NOR	$\overline{x \vee y}$
NIM	$x\,\bar{y}$
AND	$x\,y$
XOR	$x\,\bar{y} \vee \bar{x}\,y$
EQV	$x\,y \vee \bar{x}\,\bar{y}$
NAN	$\overline{xy}$
OR	$x \vee y$
RIM	$x \vee \bar{y}$
− y	$\bar{y} + 1$
x − y − 1	$x + \bar{y}$
− 1	NOT 0

EXTERNAL INSTRUCTION

(Literal =: DEV)

Figure 4-1. Summary of Microinstruction Types



```
MINI  MINI   SOURCE STATEMENT
LOC   CODE


 00                    PROGRAM  DATA COM
 01                    ADR 4/01
 01   FFB               START:                  4/00 =: B
 02   04D                                       B =: A3
 03   083                                       4/08 =: DEV
 04   E4F                                       IF EXT THEN SET LC1 SKIP ELSE STEP
 05   042                                       GOTO *-1
 06   BAF                                       STEP IF LC2
 07   DAF                                       STEP IF LC3
 08   0C2                                       GOTO INPUT1
 09   083               INPUT:                  4/08 =: DEV
 0A   FCF                                       IF EXT THEN SKIP ELSE STEP
 0B   0A2                                       GOTO *-1
 0C   049               INPUT1:                 B =: A2
 0D   0E1                                       0 =: B BEX
 0E   1CF                                       IF MST THEN SKIP ELSE STEP
 0F   132                                       GOTO INPUT2
 10   7FB                                       4/80 =: B
 11   055                                       B =: OUT1
 12   C0B                                       4/3F =: B
 13   045               INPUT2:                 B =: A1
                     % COMMENT    SYNC TO B
 14   E9B                                       4/16 =: B
 15   501                                       A1 EQV B =: B
 16   7B7                                       IF ABT THEN STEP ELSE SKIP
 17   092                                       GOTO INPUT
                     % COMMENT    EOT TO B
 18   FBB                                       4/04 =: B
 19   501                                       A1 EQV B =: B
 1A   7B7                                       IF ABT THEN STEP ELSE SKIP
 1B   012                                       GOTO START
 1C   000                                       4/00 =: AMPCR
 1D   801                                       A2 + AMPCR =: AMPCR
 1E   9A7                                       IF LC1 THEN STEP ELSE JUMP
 1F   B37                                       IF LC2 THEN SET LC3 STEP ELSE SKIP
 20   092                                       GOTO INPUT
 21   DB7                                       IF LC3 THEN STEP ELSE SKIP
 22   342                                       GOTO GROUP
                     % COMMENT    ADDRESS 1 TO DEV
 23   103                                       4/10 =: DEV
 24   0E1                                       0 =: B BEX
 25   501                                       A1 EQV B =: B
 26   637                                       IF ABT THEN SET LC1 STEP ELSE SKIP
 27   D1B                                       ADR2 =: B
 28   9B7                                       IF LC1 THEN STEP ELSE SKIP
 29   092                                       GOTO INPUT
 2A   D3B                                       ROUTINE =: B
 2B   092                                       GOTO INPUT
 2C   CBB               ROUTINE:                GROUP =: B
```

Figure 4-2. Translator Locations Codes and Source Statements

APPENDIX I

SUMMARY OF MICROINSTRUCTION CODES

LITERAL INSTRUCTIONS

1	2	3	4	5	6	7	8	9	10	11	12
Literal From MEM ──► B								1	0	1	1

1	2	3	4	5	6	7	8	9	10	11	12
Literal Jump Address ──► AMPCR								Not Used		0	0
TO Literal								Not Used		1	0

EXTERNAL INSTRUCTION

1	2	3	4	5	6	7	8	9	10	11	12
Literal ──► Dev								0	0	1	1

LOGIC UNIT INSTRUCTION

1	2	3	4	5	6	7	8	9	10	11	12
X Select		Operation and Y Select				Destination Select				0	1

X Select		Operation and Y Select			Destination Select	
00	0	0000	X+B+1		0000	B
01	A1	0001	X+B		0001	A1
10	A2	0010*	X+Z+1		0010	A2
11	A3	0011*	X+Z		0011	A3
		0100	X EQV B	$(\overline{X}\overline{B} \vee XB)$	0100	OUT0, ---
		0101	X XOR B	$(X\overline{B} \vee \overline{X}B)$	0101	OUT1
		0110	X-B	$(X+\overline{B}+1)$	0110	OUT2
		0111	X-B-1	$(X+\overline{B})$	0111	AMPCR, OUT3
		1000	X NOR B	$(\overline{X \vee B})$	1000	B, BEX
		1001	X NAN B	$(\overline{XB})$	1001	A1, BEX
		1010*	X NOR Z	$(\overline{X \vee Z})$	1010	A2, BEX
		1011*	X NAN Z	$(\overline{XZ})$	1011	A3, BEX
		1100	X OR B	$(X \vee B)$	1100	B S
		1101	X AND B	(XB)	1101	A1 S
		1110	X RIM B	$(X \vee \overline{B})$	1110	A2 S
		1111	X NIM B	$(X\overline{B})$	1111	A3 S

*When Z is not selected as destination, Z = 0.

CONDITION TEST INSTRUCTION

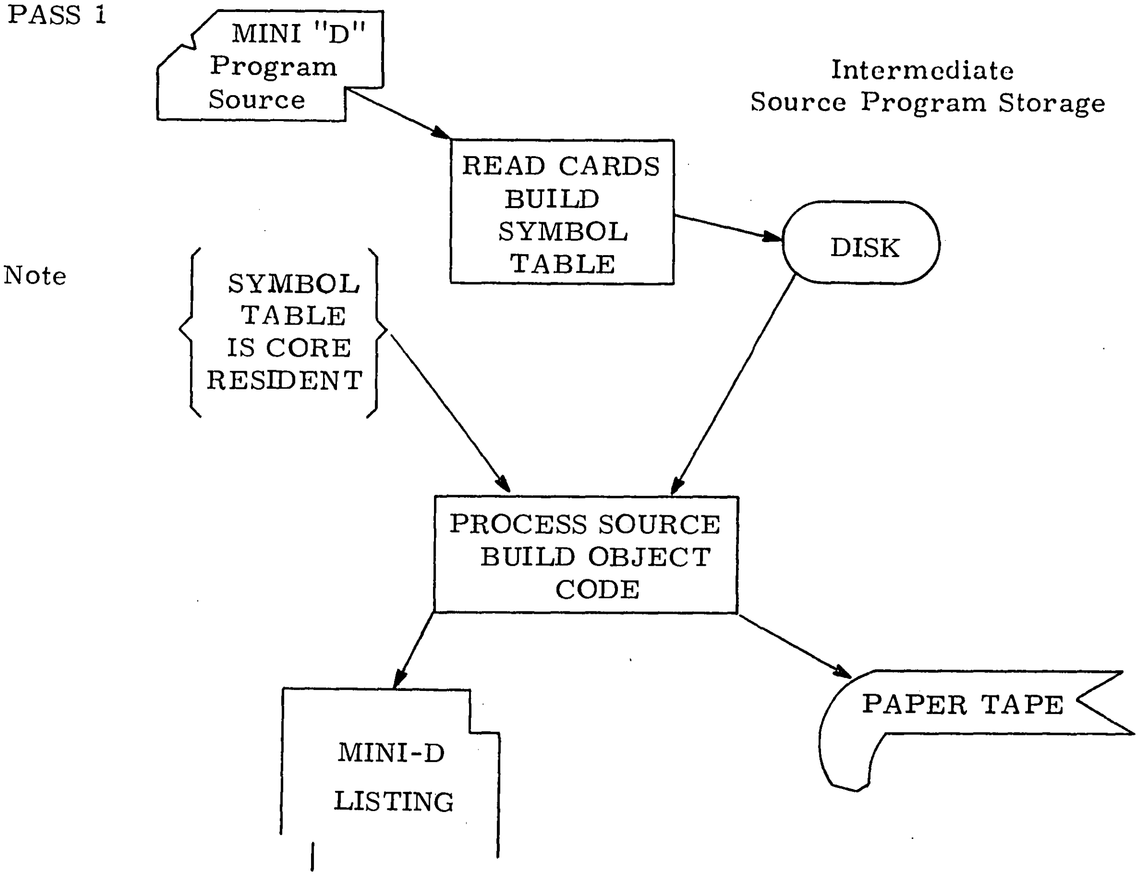
1	2	3	4	5	6	7	8	9	10	11	12
Condition Select			Set Operation		True Successor		False Successor		1	1	1

Condition Select		Set Operation		True Successor		False Successor	
000	MST	00	Set LC1	00	Jump	00	Jump
001	AOV	01	Set LC2	01	Step	01	Step
010	LST	10	Set LC3	10	Skip	10	Skip
011	ABT	11	None	11	Save	11	Save
100	LC1						
101	LC2						
110	LC3						
111	EXT						

APPENDIX II

MINIX TRANSLATOR ORGANIZATION

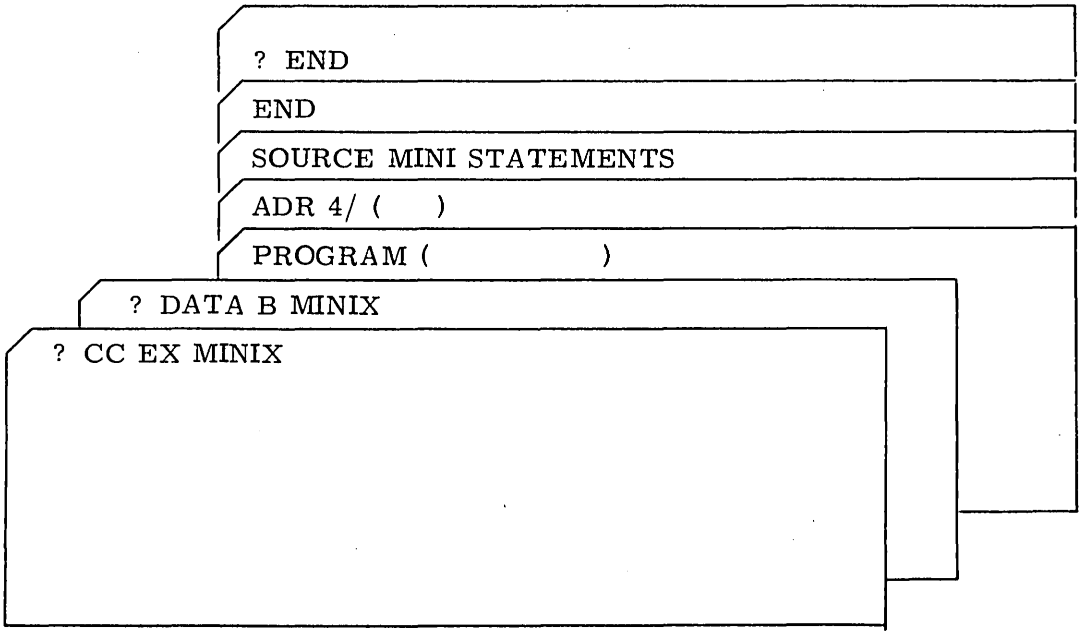
The Minix translator is a two pass sequence where pass one develops the symbol table, and pass two produces the object code for each source statement.


PASS 1
MINI "D" Program Source
READ CARDS BUILD SYMBOL TABLE
Intermediate Source Program Storage
DISK
Note
SYMBOL TABLE IS CORE RESIDENT
PROCESS SOURCE BUILD OBJECT CODE
MINI-D LISTING
PAPER TAPE

APPENDIX III

MINIX EXECUTION

The control card organization required for the B3500 system in executing MINIX for a MINI-D translation is shown below.



```
? END
END
SOURCE MINI STATEMENTS
ADR 4/ (     )
PROGRAM (                 )
? DATA B MINIX
? CC EX MINIX
```

APPENDIX IV

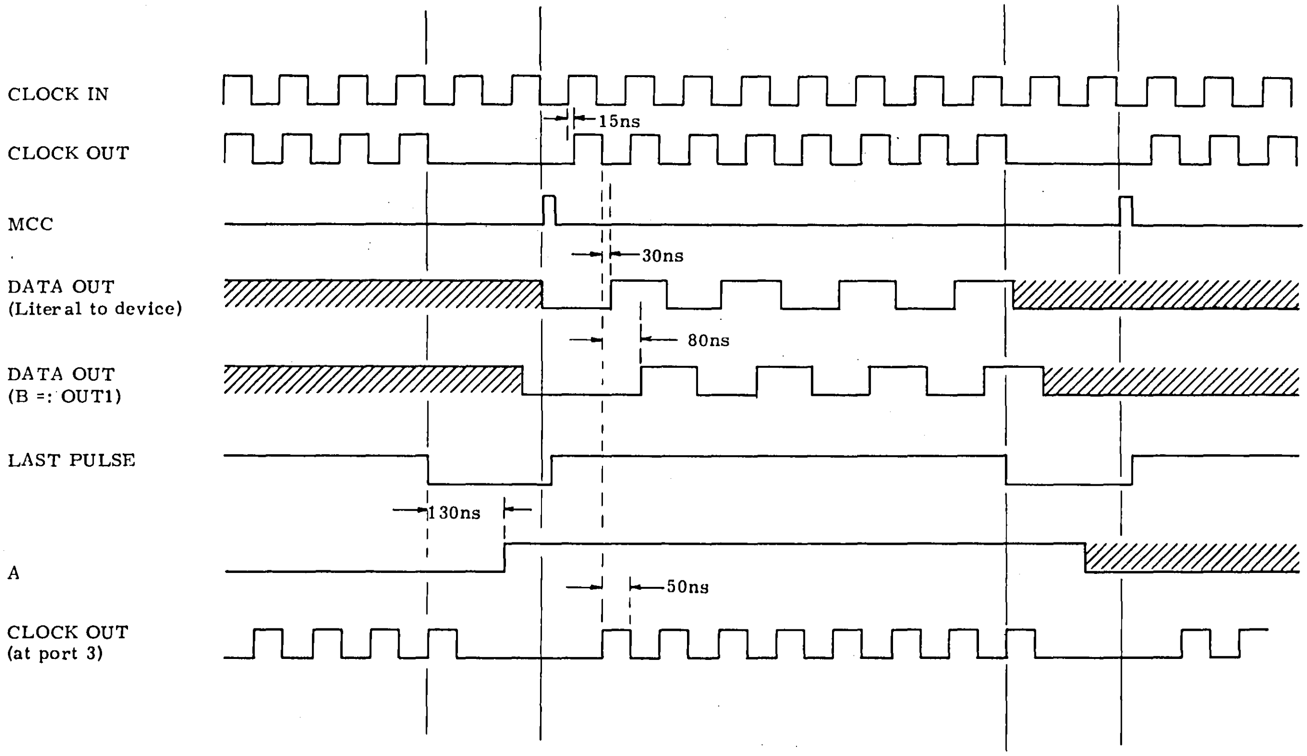
MINIX INDICATOR LIST

IND	Explanation
A	Card must contain less than 50 items = item is a separation between words "A" or special char
B	WORD-ITEMS (Labels) are restricted to 20 characters.
C	Card Interpreted as Blank
D	Label was previously defined.
E	Specification of location must have label.
F	Invalid Constant Delimiters not specified.
G	Symbol table overflowed
H	Invalid Condition Specified (LST MST Etc.)
I	No true successor specified.
J	Required word omitted "ELSE"
K	No DCW DW DR Specification in Hex
L	Logic Unit operations must start with A1, A2, A3, B, -, NOT, 0, AMPCR, or 1.
M	No logic Unit Separator or Select terminate X = :
N	Illogical Sequence B. AMPCR, 1 must be selected
O	Invalid Selection for ADDER 1 OP - B Required
P	Logic & select must be B or AMPCR

Q	Illogical Operator OR, NOR, etc.
R	Logic select off Y must be B Register
S	Invalid select must be B Register
T	Invalid NOT Select Parameter
U	No dest selected
V	Adder op 2 requires AMPCR
W	AMPCR not allowed in ADDER OP 3.
X	No condition specified after SET
Y	No false successor Specified--warning Step implied.
Z	Period delimiter omitted
0	Label not in symbol table
1	Incomplete instruction NO = :

APPENDIX V

MINI-D AND PSU TIME DELAYS



Burroughs Corporation

Federal and Special Systems Group

Paoli, Pennsylvania 19301

Printed in U.S. America