EK-M8207-TM-002

M8207 Microprocessor Technical Manual

digital

EK-M8207-TM-002

M8207 Microprocessor Technical Manual

Prepared by Educational Services
of
Digital Equipment Corporation

1st Edition, May 1979
2nd Edition (Rev), November, 1982

Printed in U.S.A.

This document was set on DIGITAL's DECset-8000 computerized typesetting system.

CONTENTS

CONTENTS (Cont)

FIGURES

TABLES

PREFACE

The *M8207 Technical Manual* describes the hardware portion of the processor operation. The information presented consists of a brief discussion of the purpose and features of the M8207, a technical description of circuit operation which includes key timing functions, and module level diagnostics. Appendix A contains Microdiagnostic Tests, Appendix B contains Integrated Circuit Descriptions, and Appendix C contains the M8207 microprocessor jumper configurations. A Glossary and an Index are also provided.

The following documents provide supplementary information.

- *M8207 Print Set* (D-CS-M8207-0-0)
- *M8203 Technical Manual* (EK-M8203-TM-001)

CHAPTER 1
INTRODUCTION

1.1 PURPOSE

The M8207, a Direct Memory Access (DMA) device, is a UNIBUS compatible, general purpose microprocessor with Read Only Memory (ROM) control storage. Working as a parallel Input/Output (I/O) processor, the M8207 is used primarily to reduce I/O load on the Central Processing Unit (CPU), and therefore, functions more like a data handler than a data processor.

1.2 DESCRIPTION

The M8207 microprocessor is implemented on a single hex multilayer board and will operate in most hex SPC slots of a UNIBUS (DD11-B backplane cannot be used). The only power requirement is +5 Vdc.

The functions performed by the M8207 are determined by the microprogram contained in its instruction memory as specified by the microcode stored in the ROMs.

The architecture is based on a 16-bit instruction set. The M8207 cannot modify its own instruction area.

All communications between the microprocessor and external UNIBUS devices are accomplished through a Multiport Random Access Memory (MP RAM). An 8-bit port (BERG Port) allows the microprocessor to communicate with a line unit or other device without use of the UNIBUS.

M8207 features are outlined below.

- Instructions – Located in 2-K by 16-bit ROM
  (increase to 6K in 2K increments)
  Length = 16-bits
  Time = 180 ns minimum
  except Branch True = 240 ns
  MP RAM Access $\geqslant$ 210 ns
- Programming – ROMs perform an appropriate task
- Data Memory – 4K- by 8-bits
- Scratchpad – 16- by 8-bits
- Data Paths – 8-bits wide
- Non-Processor Request Address – 18-bits
- Interrupt Vector – 2 (XX0 and XX4)
- Control and Status Register defined by the hardware – 1 byte

- Control and Status Registers defined by microcode – 7 bytes
- Arithmetic Logic Unit Functions – 16
- Multiple Transfer per Non-Processor Request
- Program Timer – 1 millisecond
- UNIBUS Transfer – byte or word
- Read Program Counter and Memory Address Register
- Assert AC LO is jumper selectable
- BERG Port is compatible with M8200 and M8204

1.3 M8207 BASIC CONTROLLING ELEMENTS

A simplified block diagram of the M8207 is shown in Figure 1-1. The block diagram identifies the basic controlling elements of the M8207 and shows the functional location of the microprocessor between the UNIBUS and its I/O Port. The M8207 I/O BERG Port interfaces with a line unit through a one foot cable.

The M8207 interfaces with the UNIBUS and performs DMA operations using Non-Processor Requests (NPRs) and interrupts (Bus Requests – BRs). Communication between CPU and the microprocessor's UNIBUS Control and Status Registers (CSRs) is done through Data In (DATI) transactions for reading and Data Out (DATO) or Data Out Byte (DATOB) transactions for writing.

1.4 MAJOR FUNCTIONAL AREAS

The major functional areas of the M8207 Microprocessor are:

Main Memory (MEM) and Memory Address Register (MAR)

The MEM is the M8207 data storage area consisting of a 4096 by 8-bit RAM. This area cannot be used for instruction storage nor can the stored data be executed as an instruction. The MEM cannot be accessed directly by the CPU.

The MAR is set up by the microprocessor to address the MEM.

Branch Register (BRG)

The BRG is an 8-bit register used as a temporary data register for branch determination and for a rotate right operation. It has three modes of operation: load, shift right, and hold. Input data comes from the Buffered Arithmetic Logic Unit (BALU). During a right shift, only the BRGs most significant bit (bit 7) is sourced from BALU bit 0. Output data goes to the B input of the ALU via the Data Multiplexer (DMUX).

Program Counter (PC)

The PC is a 14-bit counter that can be parallel loaded from the ALU for the lower eight bits during a branch instruction, when the branch is true. The upper six bits are loaded from two sources: from the Program Counter Register (PCR – OUTBUS* 13) on branch condition 0, or, on branch conditions 1 through 7, from bits 11 and 12 of the branch instruction for the page bits. The field bits are simply rewritten from the PC. The branch not true and the move instruction increment the PC. The output of the PC controls the address of the Control ROM directly.

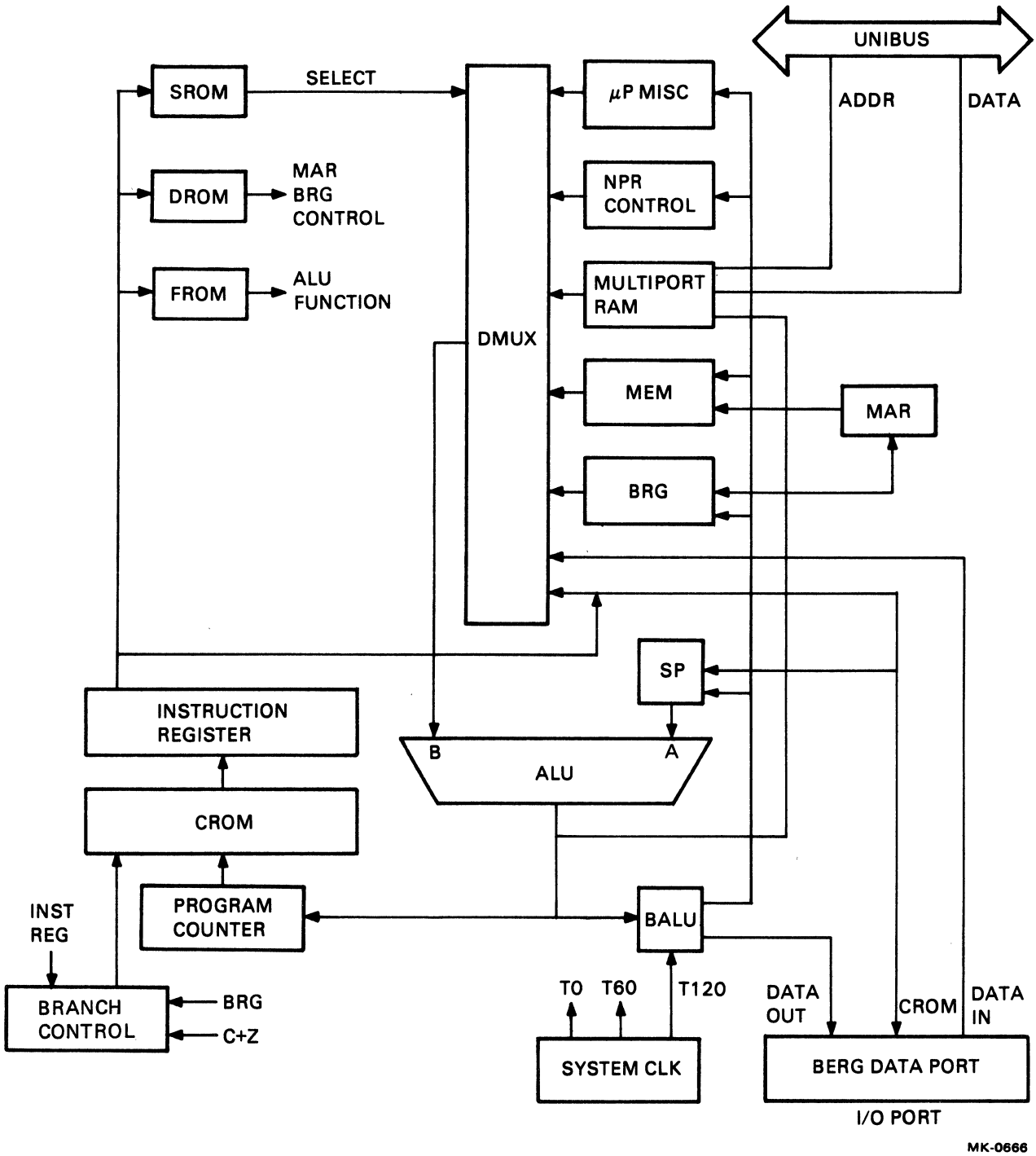


Figure 1-1 Microprocessor Simplified Block Diagram

Control ROM (CROM), Instruction Register (IR), and Maintenance Instruction Register (MIR)
The M8207 uses up to a 6K- by 16-bit array of ROMs for the instruction memory. The CROM can be configured for 2K, 4K, or 6K with the 23XXXF4 PROM. The CROMs are addressed by the PC during execution of the previous instruction. At time state zero, the output of the ROMs are clocked into the IR. The outputs of the ROMs are wire-ORed with the outputs of the MIR. The MIR is used for loading a single microinstruction from the CPU (typically for maintenance purposes). Selection between the MIR and the ROMs is done by CSR 0, bit 9 (ROMI). When bit 9 is set, the MIR is selected. The MIR is loaded every time CSR 6 is loaded.

Data Multiplexer (DMUX) and Source ROM (SROM)
The DMUX is an 8-bit wide, 8-to-1 line multiplexer. Its 8-bit output goes to the B input of the ALU. Input selection for the DMUX is controlled by the SROM which is a 32- by 8-bit ROM. The SROM also determines if a move instruction is to be executed.

Scratchpad Memory (SP) and Addressing Multiplexer
The SP is a 16- by 8-bit read/write memory that is used for temporary storage of data. An operand can be presented to the A input of the ALU only through the SP. Scratchpad addressing is done through a multiplexer. During normal operation, the addressing is controlled by the lower four bits of the CROM. During an output transfer to the destination field OUT or OUT* registers, SP address 0000 is automatically presented to the A input of the ALU.

Arithmetic Logic Unit (ALU) and Function ROM (FROM)
The ALU allows the microprocessor to perform arithmetic and logic operations on its A and B inputs. The FROM, which is a 32- by 8-bit ROM, controls up to 16 functions to be performed by the ALU. The carry output (C bit) of the ALU is connected to a flip-flop to store the carry indication if it occurs during a move, or if it is forced by the FROM.

The AB output (Z bit) of the ALU is connected to a flip-flop to store the indication of equality of the A and B inputs of the ALU if it occurs during a move instruction.

Multiport RAM (MP RAM) and Associated Logic
The Multiport RAM is a RAM that contains all the M8207 status registers except four: NPR Control, Miscellaneous, PC, and MAR registers. The Multiport RAM has two ports (A and B) that can be accessed simultaneously for a read operation; however, only the A port can be written into. The associated logic consists of read/write control and addressing multiplexers which allow access to the Multiport RAM by the microprocessor and the CPU. The Multiport RAM contains the CSRs that are accessible by both the M8207 and the CPU program. It also contains the NPR buffer pointers and NPR data that are accessible only by the M8207.

Microprocessor registers 0 through 7 of OUT/IBUS and OUT*/IBUS*, and CSR bytes 0 through 7 are implemented in the MP RAM.

NOTE

A write to CSR byte 1 sets and clears bits in internal controlling logic and MP RAM. A read from CSR 1 reads the RAM only.

System Clock
The clock provides clock pulses for the microprocessor. It generates a series of three nonoverlapping 60 ns time pulses for a time interval of 180 ns for branch instructions, condition not true, and move instructions that do not access the Multiport RAM. The branch true instruction delays T60 by 60 ns for an instruction time of 240 ns. The move instruction that accesses the Multiport RAM adds a minimum of 30 ns to the instruction time if the Multiport RAM is not being accessed by the processor. The 30 ns is extended for a processor write or an NPR in progress; the system clock is held up until the transfer is complete.

Non-Processor Request (NPR) and Control Logic
The control logic allows the microprocessor to initiate an NPR under microprogram control and assume UNIBUS mastership so that it can transfer data to or from the CPU memory. The not last transfer bit can be used to retain bus mastership for multiple DMA transfers.

Interrupt Control Logic
This logic allows the microprocessor to interrupt the CPU and cause vectoring to either of two locations in the floating vector address space. The vector address is switch selectable and occupies two consecutive vector locations: XX0 and XX4. The M8207 interrupt priority is plug selectable for level 4, 5, 6, or 7; however, it is normally shipped as a level 5 device.

Address Selection Logic
This logic consists of switches to specify the microprocessor device address plus logic to decode this address. Address selection logic also decodes the selected register and the type of UNIBUS transaction requested. Using these factors, the logic generates the appropriate control signals to implement a read or write operation as directed by the UNIBUS control lines.

Destination ROM (DROM) and D Decode
DROM and D Decode are driven by the destination field of the microinstruction, and control the destination location of data in a move instruction.

NPR Control Register (Register 10) and Microprocessor Miscellaneous Register (Register 11)
These registers are used for M8207 general control and status. The bit functions are described in Chapter 2.

Branch Control
The branch control multiplexer is driven directly by the instruction register output bits CROM <10:8>. It is used for determining branch conditions.

1.5 MICROPROCESSOR FEATURES
Features of the M8207 provided in Paragraph 1.2 are compared with features of other microprocessors in Table 1-1.

Table 1-1 Microprocessor Features

Function	M8200	M8207	(M8204) KMC11-A	(M8206)*
Instruction Time	300 ns min except MP RAM 330 ns	180 ns min except Branch true 240 ns & MP RAM 210 ns	300 ns min except MP RAM 330 ns	180 ns min except Branch true 240 ns & MP RAM 210 ns
Instruction Memory	1K × 16	6K × 16	1K × 16	4K × 16
Data Memory	256 × 8	4K × 8	1K × 8	4K × 8
Instructions in ROM	yes	yes	no	no
Instructions in RAM	no	no	yes	yes
Multiple Transfer per NPR	no	yes	yes	yes
Program Timer	1 ms	1 ms	50 μs	50 μs
Load PC	no	no	no	yes
Read PC (Host)	no	yes	no	yes
Read MAR	no	yes	no	yes
Assert AC LO	yes	selectable	yes	selectable
Load MAR HI	no	yes	yes	yes
Latch of IBUS* 12-15 in Maintenance	no	no	no	yes
IBUS* 12-15	no	yes	no	yes
LU Stop CLK	no	yes	no	yes
Power	+5 V @7 A	+5 V @7 A	+5 V @7 A	+5 V @7 A

*M8206 is a CSS special programming module for the M8207.

CHAPTER 2
TECHNICAL DESCRIPTION

2.1 SCOPE

The technical description is addressed to circuit operation on a functional level. Logic block diagrams are included to supplement the text, and reference to the *M8207 Print Set* is made throughout the chapter. The block diagram of Figure 2-1 (Sheet 1) shows the major interconnections between the basic controlling elements of the microprocessor. A map of the Control and Status Registers (CSRs), is provided in Figure 2-1 (Sheets 2 and 3). Detailed descriptions of the area represented in Figure 2-1 constitute the following sections of this chapter.

NOTE
Dx references in the text are based on the signal map rather than the actual sheet numbers of the Print Set.

2.2 TIMING (Figure 2-2)

The basic timing is generated by a 33.33 MHz crystal that can be disabled by removing jumper W1 to accommodate automated circuit board testers. The crystal output provides the clock input to the MSYNC flip-flop (D16 of Print Set) and buffered clock inputs, via a 74S240 inverter, to the WAIT SYNC, SYSTEM CLOCK (SYS CLK), and DELAY flip-flops.

The SYS CLK flip-flop:

- is used to generate the 60 ns timing pulses for the timing chain.
- is only held clear when the WAIT SYNC flip-flop is clear or in a wait state.
- is held set during a master clear.
- clear and K inputs are tied to +3 volts.
- J input comes from the output of the WAIT SYNC flip-flop.
- output is used to drive the clocks of the RUN SYNC, T0, T60, and T120 flip-flops.

The WAIT SYNC flip-flop:

- is used to eliminate timing problems when the wait signal goes away too close to the clocking edge of the system clock.

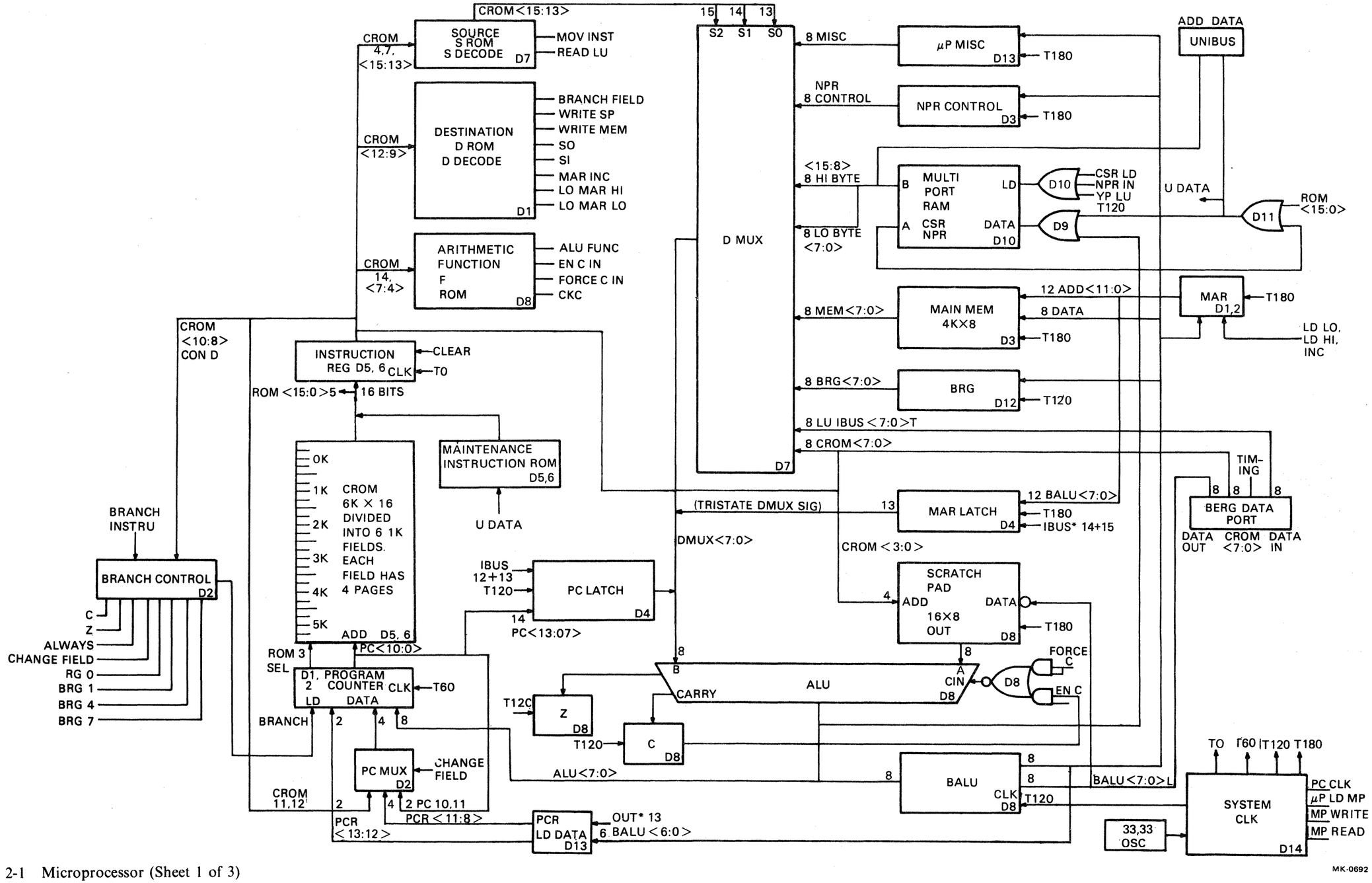


Figure 2-1 Microprocessor (Sheet 1 of 3)
a. Interconnection Between Controlling Elements

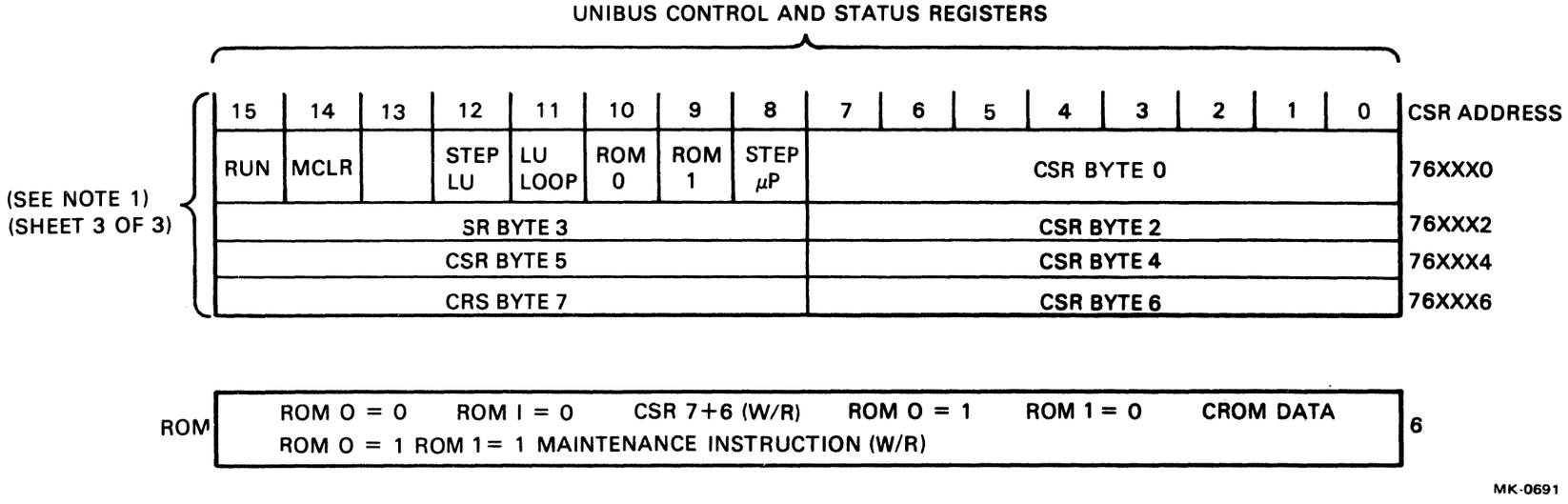
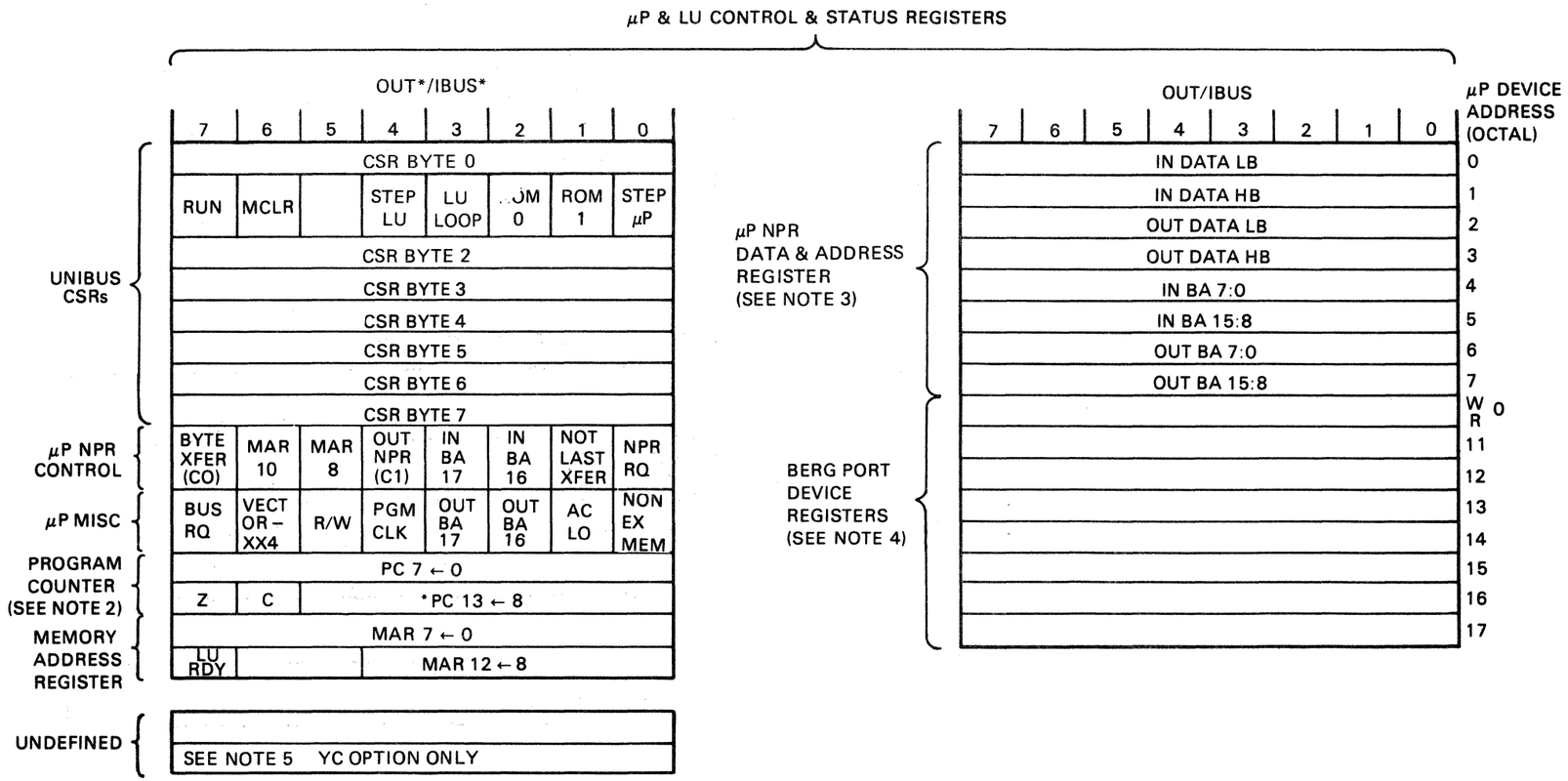


Figure 2-1 Microprocessor (Sheet 2 of 3)
b. UNIBUS Control and Status Registers



NOTES

1. UNIBUS CSRs ARE IN THE MULTIPORT RAM. BYTE 1 HAS A HARDWARE STATUS REGISTER THAT IS ADDRESSABLE ONLY FROM THE UNIBUS. IF THE μP ADDRESSES BYTE 1, IT WILL ONLY WRITE INTO THE RAM. ALL THE OTHER CSR BYTES ARE DEFINED BY THE MICROCODE EXCEPT IN MAINTENANCE MODE.

2. IBUS* 12–15 REFLECT THE PROGRAM COUNTER FOR THE INSTRUCTION BEING EXECUTED AND THE MEMORY ADDRESS REGISTER FOR THE PREVIOUS INSTRUCTION. OUT*13 IS THE PCR WHICH IS A SHADOW REGISTER. THE CONTENTS OF THE PCR WILL BE TRANSFERRED TO THE PC ONLY ON A BRANCH (CHANGE FIELD).

3. NPR DATA AND BA REGISTERS ARE RAMs; HOWEVER, ADDRESS ASSIGNMENTS ARE HARDWIRED AND CANNOT BE REDEFINED.

4. OPTIONAL – THESE REGISTERS ARE ONLY VALID IF EXTERNAL DEVICES ARE ATTACHED TO THE BERG PORT

5. OPTIONAL – YC ONLY FOR 22 BIT NPR ADDRESS

MK-0664

Figure 2-1 Microprocessor (Sheet 3 of 3)
c. Microprocessor and Line Unit Control

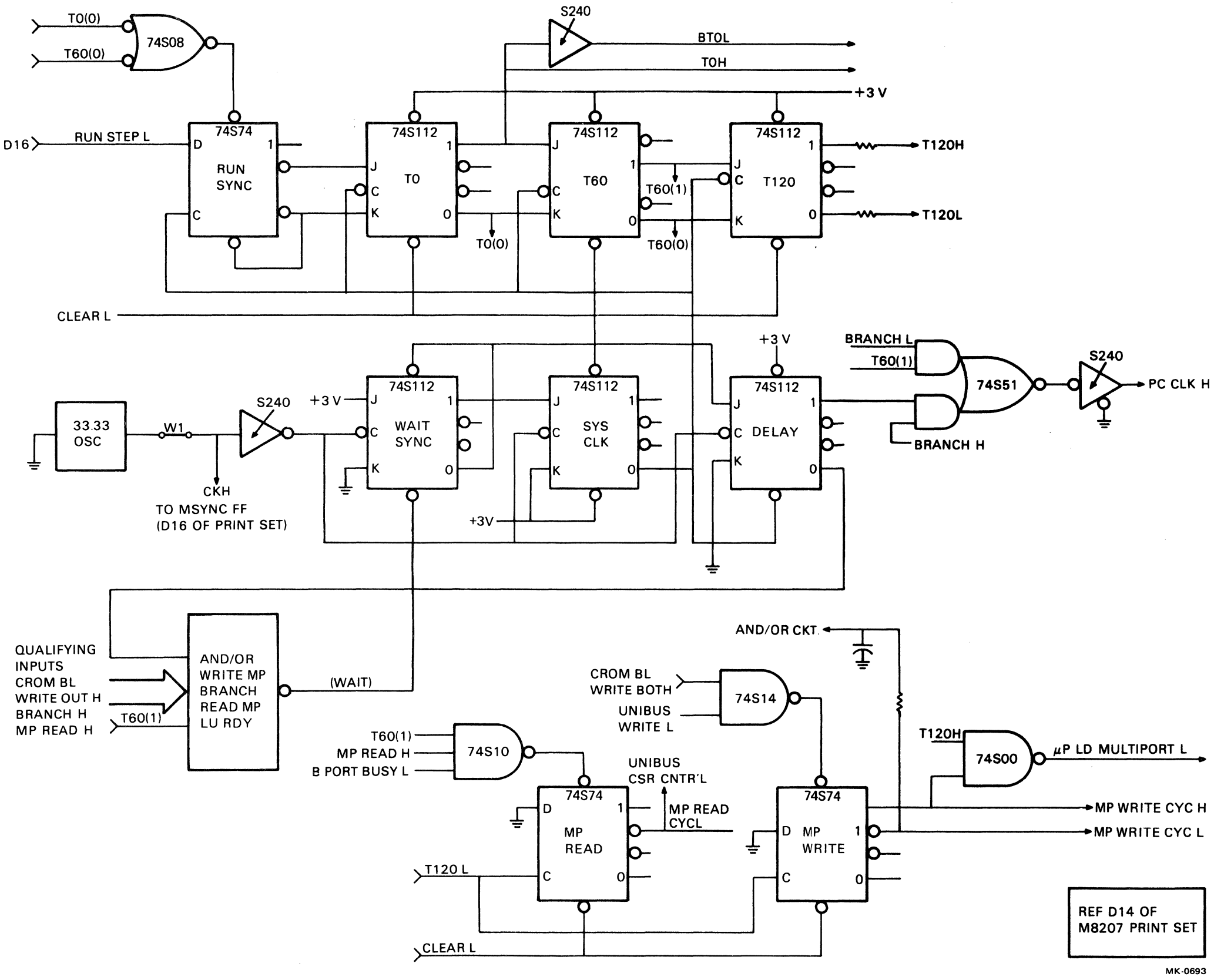


Figure 2-2 System Clock

The RUN SYNC flip-flop:

- is used to sync the transition of run or step to the system clock to eliminate race conditions.
- is a 74S74 flip-flop whose data input is low true when either the run or step function is true.
- is clocked clear to enable the start of the main timing chain.

Main Timing Chain (Figures 2-2 and 2-3)
Both high and low outputs of the RUN SYNC flip-flop go to T0s inputs. Once clocked clear, RUN SYNC is latched clear until it is set by a negative OR on its preset side during time states T0 and T60. Data input to the RUN SYNC flip-flop must then be true low during T120 in order to reset RUN SYNC for the next instruction.

The preset side of the time state flip-flops are tied to +3 volts. This ensures that all time states are not true at the same time. The clear side of all time states are common to the CLEAR L signal from the UNIBUS. The timing chain is cleared and restarted on a master clear or power up.

- T0 – T0 is used to start an instruction. The high output from T0 clocks the Instruction Register (IR) and the line unit, or option, connected to the BERG Connector. Both outputs from T0 are input to T60 so that the next clock will generate the next time state.
- T60 – T60 is clocked on the trailing edge of T0. T0 is always 60 ns long. However, T60 is delayed by a wait during a branch true or move instruction using the Multiport RAM (line unit WAIT L when the line unit is addressed and is not ready). The high output from T60 is used to increment the Program Counter (PC) when branch is not true, and to generate the wait pulse. During this time period, the instruction has been decoded, the arithmetic function has been executed, and execution of the instruction has started. Both outputs from T60 are input to T120.
- T120 – The next clock clocks T120. The leading edge of T120 is used to clock data out of the Arithmetic Logic Unit (ALU). The source and arithmetic functions are complete at this time; normally 120 ns from the beginning of the instruction. The trailing edge of T120 loads all the registers and memories with data from the Buffered ALU (BALU). The exception is the Multiport RAM which is loaded on the leading edge from the ALU.

  During T120, the RUN SYNC flip-flop is allowed to reset for the next instruction. Both the leading and trailing edges of T120 are actually used to load all registers and memories on the board, except the PC. The PC is loaded by the DELAY flip-flop during a branch true, or incremented by the leading edge of T60. Gating for the PC clock is done with a 74S51 (D14 of Print Set).

The wait for the timing chain is defined through a 74S64, which is a 4-2-3-2 input AND-OR gate. The qualifying inputs are defined by the instruction and are qualified at the leading edge of T60. The four qualifying conditions are:

- Line unit not ready to be read
- Branch true
- Read to the Multiport RAM
- Write to the Multiport RAM

The wait output is sync'd with the system clock through the WAIT SYNC flip-flop.

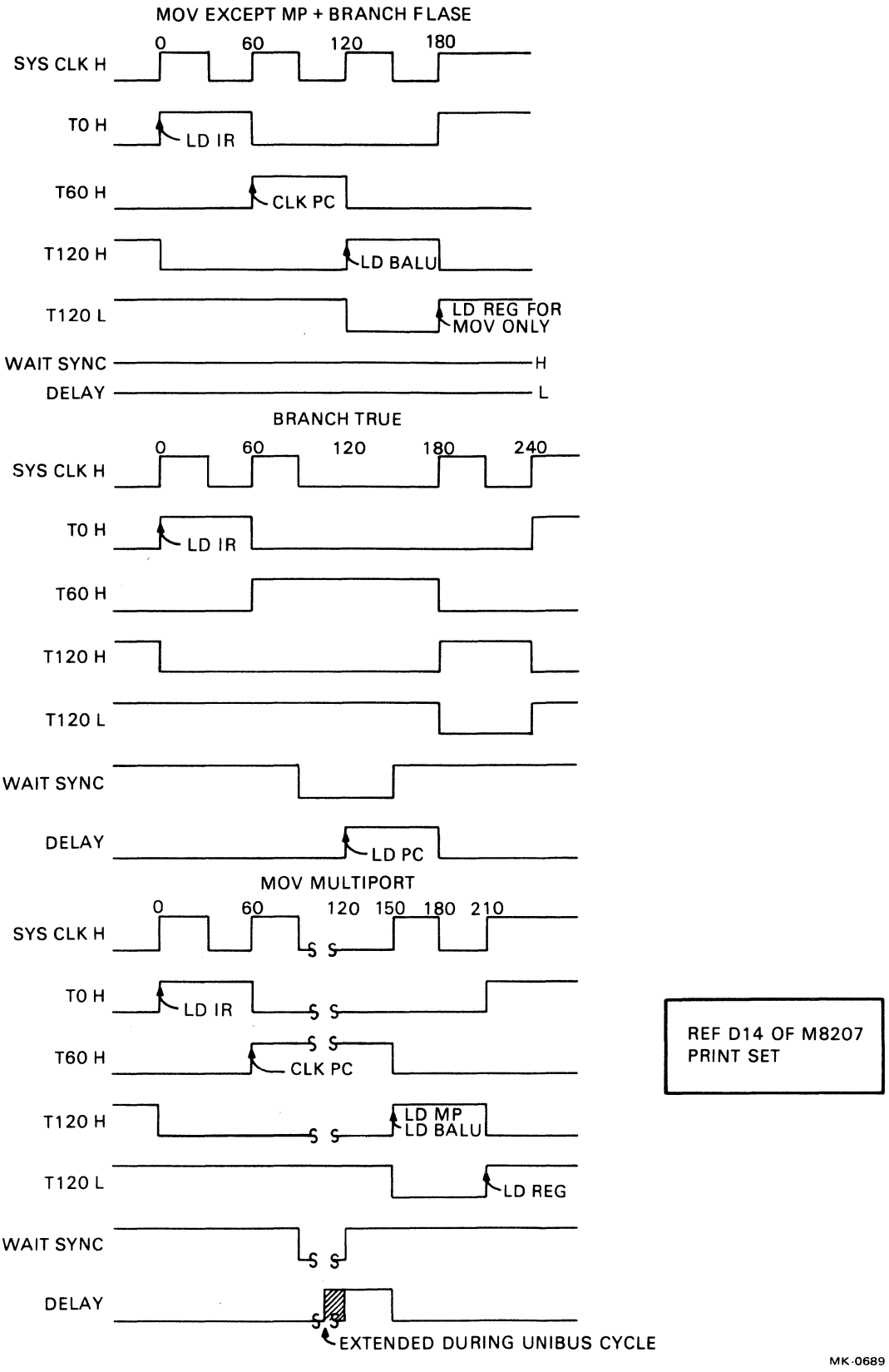


Figure 2-3 Main Timing Chain

2.3 CONTROL ROM (CROM) AND INSTRUCTION REGISTER (IR) (Figure 2-4)

CROM consists of up to six 2K- by 8-bit (82S191) high speed ROMs. The ROMs are broken up into two groups of 2K each, whose address is controlled by the PC. The groups are selected and deselected via three 74S00 gates (D2 of Print Set). These gates are driven by the high order bits of the PC and the maintenance selection bit, CROMIN L, which is CSR 0 bit 9. When bit 9 is set, both sets of ROMs are disabled and the Maintenance Instruction Register (MIR) is selected, allowing an instruction to be loaded in CSR 6.

When loaded, CSR 6 loads bytes 6 and 7 of the Multiport RAM (D9 of Print Set) with data for the microprocessor and loads the 16-bit MIR consisting of two 74S374 octal tristate D flip-flops (D5 and D6 of Print Set). These flip-flops are selected during a maintenance instruction.

CROM and MIR outputs are wire-ORed to the IR; two 74S175s are input by the lower eight bits <7:0>, and a 74S374 octal D flip-flop is input by the upper eight bits <15:8>. The low outputs of the 74S175 are used by the BERG Port for address selection of LU IBUS and OUT registers.

The IR is clocked at the leading edge of T0; at the beginning of an instruction. The outputs of the IR are used to control the microprocessor. They are decoded for the following via respective ROMs.

- Type of instruction (Source ROM - SROM)
- Function (Function ROM - FROM)
- Destination (Destination ROM - DROM)

CROM outputs directed to the Data Multiplexer (DMUX) are used during maintenance to verify the contents of the ROMs.

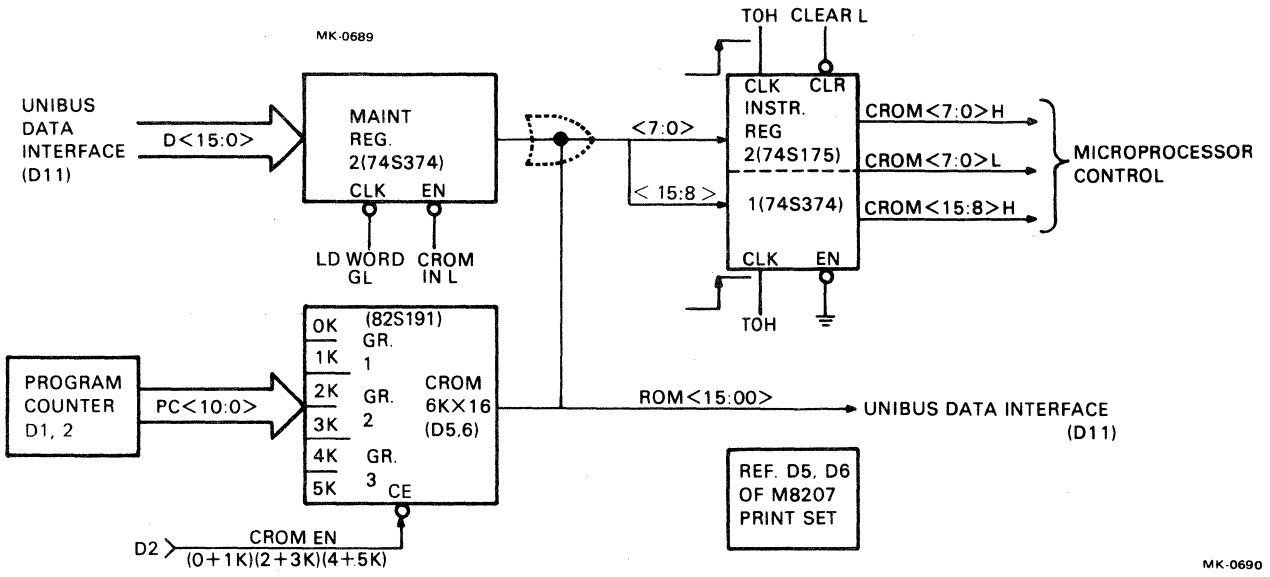


Figure 2-4 Control ROM, Maintenance and Instruction Registers

2.4 SOURCE ROM (SROM) AND DATA MULTIPLEXER (DMUX) (Figure 2-5)

The DMUX consists of a tristate 8-to-1 multiplexer, and four octal tristate registers controlled by the SROM and the Source Decoder. SROM is a 32- by 8-bit (23120A1) ROM driven from instruction register CROM bits 4, 7, 13, 14, and 15, which define the instruction and its source. SROM is always enabled. Output bits 6, 7, and 8 are DMUX select inputs S0, S1, and S2 respectively.

Table 2-1 DMUX Select Input

DMUX Select				Data Selected	Definition
S2	S1	S0			
L	L	L	(0)	CROM<7:0>H	Immediate branch and move instructions
L	L	H	(1)	LU IBUS<7:0>H	Reads all ones if nothing is connected to the BERG Connector. There are 10_8 addresses provided, usually from a MUX on the line unit
L	H	L	(2)	MEM<7:0>H	Location addressed by Memory Address Register
L	H	H	(3)	BRG<7:0>H	Single, 8-bit register used for shifting branch conditions
H	L	L	(4)	B PORT<7:0>H	From Multiport RAM
H	L	H	(5)	B PORT<15:8>H	IBUS and IBUS* registers for CSR and Non-Processor Request registers
H	H	L	(6)	NPR Control Register	(See Paragraph 2.9)
H	H	H	(7)	Microprocessor Miscellaneous Register	Can only be accessed by the microprocessor (See Paragraph 2.9)

Disabling the DMUX and selecting individual octal registers is accomplished through a 74S138 decoder. The decoder monitors the source and instruction register lines. The 74LS374 octal registers are clocked at the leading edge of T120. At this time the data in the registers are read from the PC and Memory Address Register (MAR).

The outputs of the octal registers are wire-ORed with the outputs of the DMUX and provide the B input to the ALU. Additional outputs from the SROM and Source Decoder are used by the timing and branch logic for control of the microprocessor.

2.5 ARITHMETIC LOGIC UNIT (ALU), SCRATCHPAD (SP), AND FUNCTION ROM (FROM) (Figure 2-6)

The ALU consists of two 74S181 chips and performs arithmetic and logical operations on two designated operands; input A is the SP operand (scratchpad has 17 octal locations), and input B from the DMUX is the Main Memory (MEM) or Branch Register (BRG) source operand for move and branch instructions. A 5-bit function bus defines the type of operation to perform on the A and B operands and is gated into the ALU from the FROM. The FROM is a 32- by 8-bit ROM whose address lines, CROM 14, and <7:4> are from the IR.

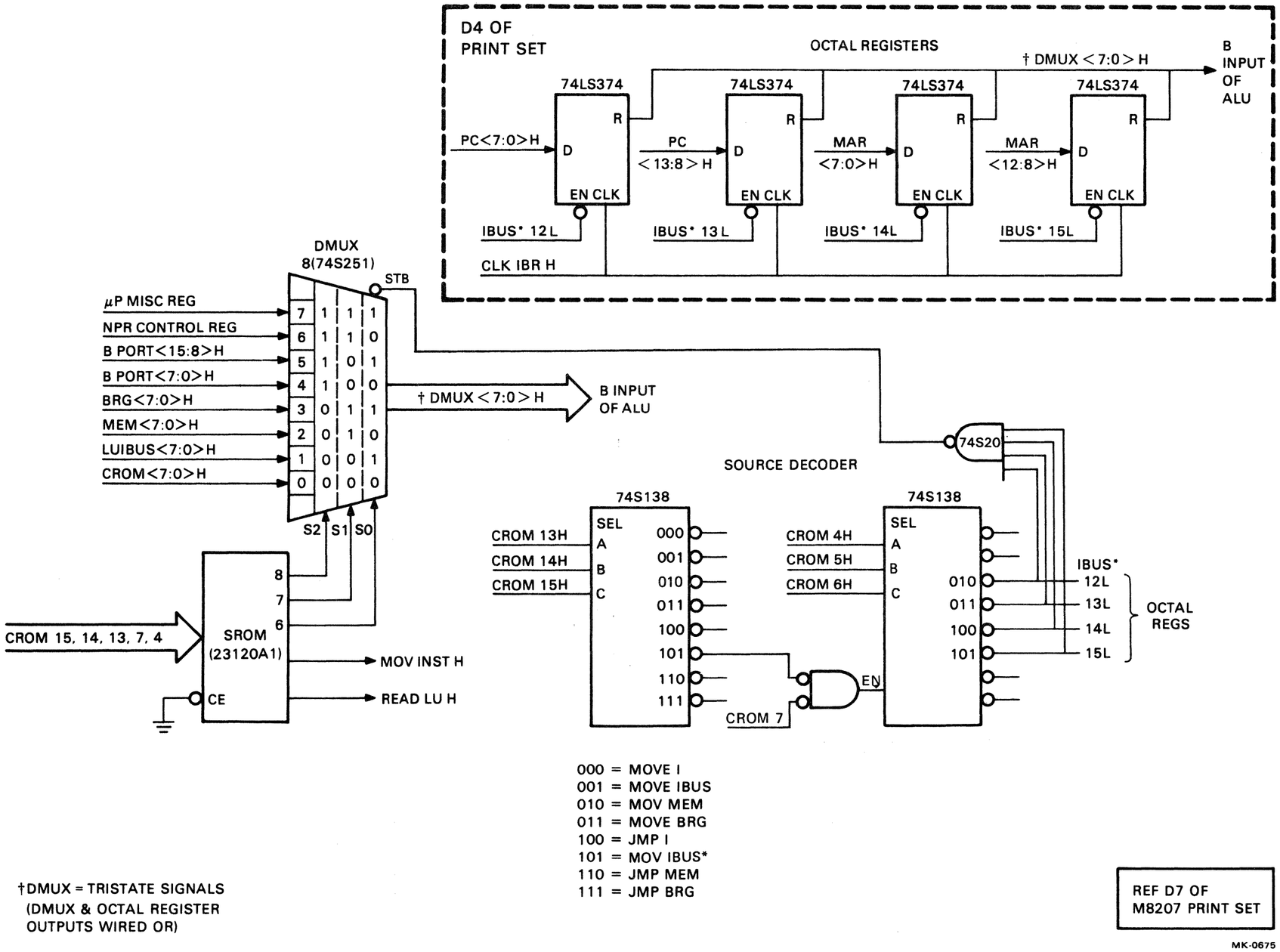


Figure 2-5 Source ROM and DMUX

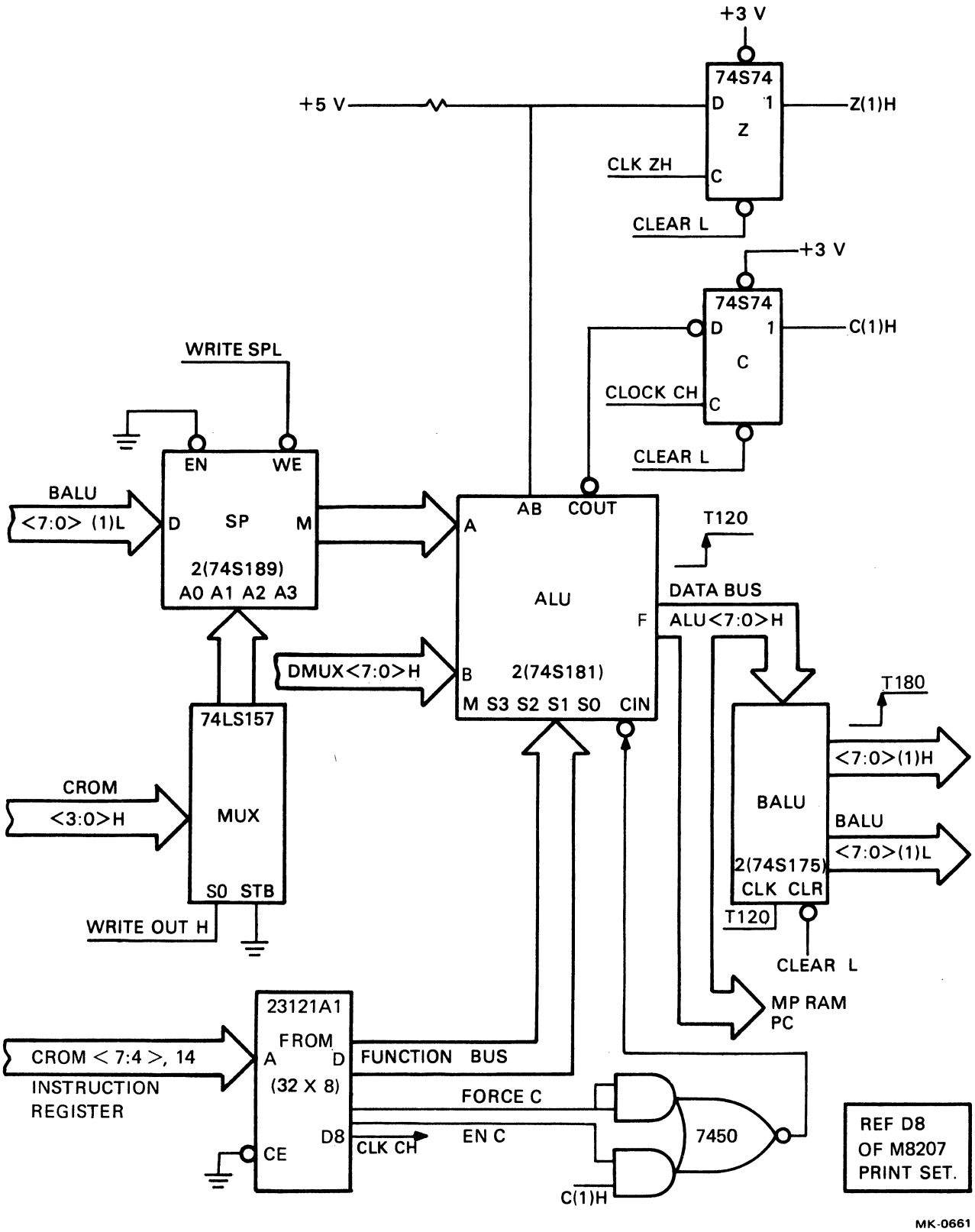


Figure 2-6 ALU, Scratchpad, and Function ROM

ALU functions are not allowed when bits 4 through 7 are used for address or data. The ALU then selects the B inputs only. The results of the operations performed by the ALU are sent to: (1) the BALU, (2) the DMUX for the Multiport RAM, and (3) the lower eight bits of the PC for branch instructions.

The BALU (two 74S175 quad D flip-flops with both high and low outputs) clocks data out of the ALU at T120. The output data from the BALU is clocked at T180 and provides input to the data memory, SP, BRG, Non-Processor Request (NPR) control, miscellaneous register, Program Counter Register (PCR), and the BERG Port. The BALU is clocked directly from the T120 flip-flop, and its clear input is generated by master clear or initialize.

The ALU chips are configured for a ripple carry that is clocked into flip-flop C, which is used in branch operations. The C flipflop is clocked by a 74S157 MUX, which provides the clock only on a move instruction and when the FROM output CK C H is true. This occurs at T120.

The AB output from the ALU is asserted when the ALU output data bus contains all ones. The AB outputs from each chip are wire-ORed and input to the Z flip-flop. The Z flip-flop is clocked at T120 on every move instruction. Both C and Z flip-flops are clocked from the same 74S157 MUX and cleared by master clear or initialize.

The ALU A operand is always generated at the SP (two 74S189 RAMs). The SP is addressed by CROM bits 0 to 3 via a 74LS157 MUX during all instructions except OUT and OUT*, when the SP address is zero.

WRITE OUT H is asserted during an OUT or OUT* operation. The SP is inverted so the low side of the BALU is used for data inputs (the RAM chips are always enabled). The SP is clocked from a 74S158 decoder MUX (D1 of Print Set), which selects the output of the DROM on a move instruction by loading the SP at T180.

2.6 MULTIPORT RAM (Figure 2-7)

The Multiport RAM (82S112 chips) is a 16- by 4-bit random access memory that is the main interface between the processor and microprocessor. All data transferred from NPRs or through CSRs go through the Multiport RAM. Data can only be written through Port A, but can be read from ports A and B simultaneously. The read from ports A and B is disabled during the write time to avoid reading invalid data.

Addressing and data transfer is accomplished through 74LS157 Multiplexers. The microprocessor is able to read and write the registers. The NPR logic reads and writes half of the Multiport RAM and the Central Processing Unit (CPU) reads the other half.

Port A addressing goes through two levels of multiplexing. The first level is for control by the microprocessor or second multiplexer. The second level is for control by the CSR or NPR logic. This is acomplished by two 74LS157 MUXs. Selection control for the first MUX (MP RAM A ADDRS) is via the microprocessor MP WRITE CYCLE H signal (D14 of Print Set). The signal switches the address to the lower three bits of the instruction (CROM <2:0>), and selects the data address to be written from the ALU via the Multiport DMUX.

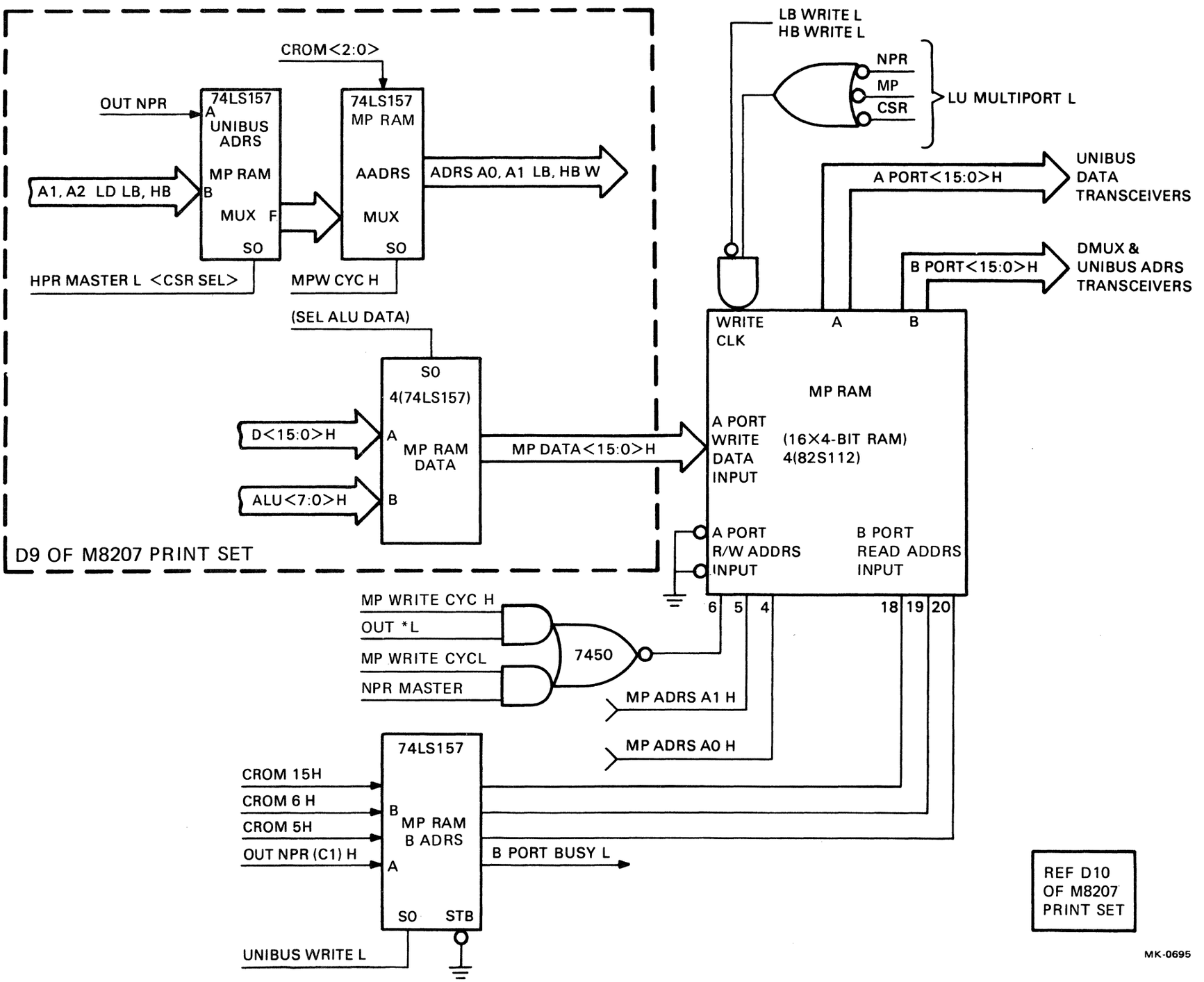


Figure 2-7 Multiport RAM

Control of the A Port address by the second MUX (UNIBUS ADRS MP RAM) occurs when the microprocessor is not attempting to write to the Multiport RAM. The UNIBUS ADRS MP RAM MUX is controlled by the NPR MASTER L signal. When the M8207 is bus master, bytes 0 + 1 or bytes 2 + 3 are always addressed (A input is selected). RAM location is a function of direction. Bytes 0 + 1 are selected for a data transfer in; bytes 2 + 3 are selected for a data transfer out. CSR selection comes directly from the UNIBUS address transceivers. When NPR MASTER L is high, the B input is selected, allowing the UNIBUS address bits to control Multiport RAM address selection. Port A data input is selected by four 74LS157 MUXs. Data is gated from the UNIBUS transceivers except when the microprocessor MP WRITE CYCLE H signal is true, in which case the data comes from the ALU.

Port B address is selected by one 74LS157 MUX. CROM bits 5, 6 and 15 from the IR determine the B address. The exception is when the UNIBUS WRITE L select input to the MUX selects OUT NPR (CI)H which is used for the NPR address, Multiport RAM address bytes 4 + 5 are selected for an NPR read, and address bytes 6 + 7 are selected for an NPR write.

Port A output goes to the UNIBUS data transceivers (D11 of Print Set). Port B output goes to the DMUX (D7 of Print Set) and to the UNIBUS address transceivers (D12 of Print Set). The write signal to the Multiport RAM comes from a three input OR gate under three conditions:

- NPR Load Multiport
- Microprocessor Load Multiport
- CSR Load Multiport

Its output is ANDed with the byte select for the byte to be written; LB WRITE L for bits 7-0; HB WRITE L for bits 15-8.

2.7 DATA MEMORY (Figure 2-8)

The data memory, a 4K- by 8-bit array of 2147-3 chips, can only be accessed by the microprocessor for data storage. The data memory is completely separate from the instruction memory. The memory is addressed from the 12-bit MAR. A move instruction loads the MAR, high or low bytes, and also increments the MAR via CROM bits 11 and 12 of the move instruction, which are input to the Destination ROM. DROM controls both loading and incrementing of the MAR and data memory. The MAR is clocked on every move instruction and cleared low on master clear and initialize. The contents of the MAR can be read through IBUS* registers 14 and 15, with page and field overflow on bits 5 and 6 of IBUS* register 10. IBUS* registers 14 and 15, represent the MAR for the previous instruction (see Figure 2-1, Sheet 3 of 3). Data memory is loaded from the BALU at time 180. The write pulse comes from a 74S158 (D1 of Print Set) on a move instruction when the DROM output is true. Data memory output is sent to the DMUX (D7 of Print Set).

2.8 PROGRAM COUNTER (PC) (Figure 2-9)

The PC is a 14-bit counter that is incremented at time 60 unless the instruction is a true branch. In this case, the PC is parallel loaded 120 ns from the beginning of the instruction. The lower eight bits of the PC are loaded from the ALU; the next four bits are loaded through a 74LS157 MUX, and the upper two bits are loaded from the Program Counter Register (PCR), bits 4 and 5.

- If branch condition zero exists - the MUX selects PCR outputs <13:8>H
- If branch conditions 1 through 7 exist - CROM bits 11 and 12 of the IR are used for the paging bits, and the field bits are reloaded from the PC.

The PCR is OUT* register 13. It contains the upper six bits to be loaded (PCR <13:8>H) into the PC on branch condition 0. This provides extra field addressing with future expansion capability.

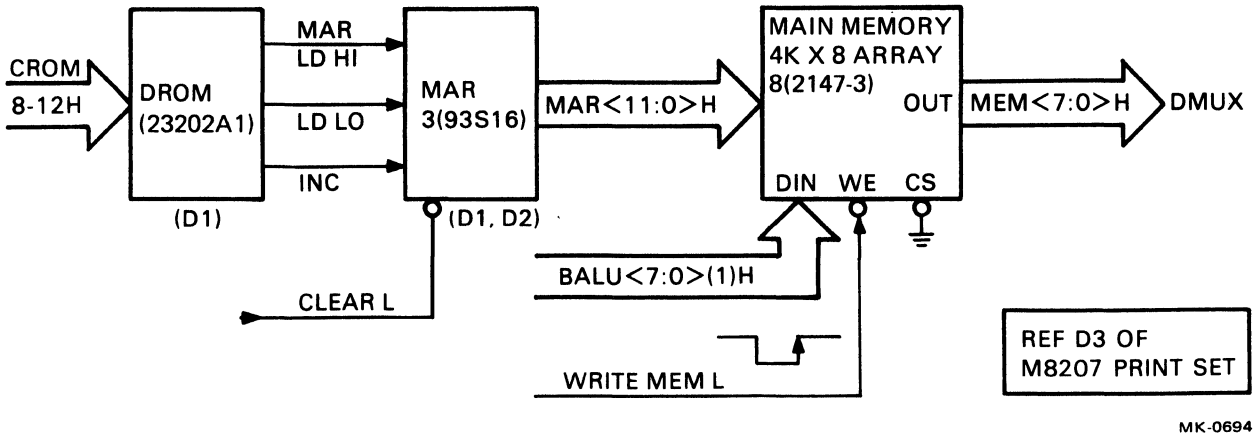


Figure 2-8 Data Memory

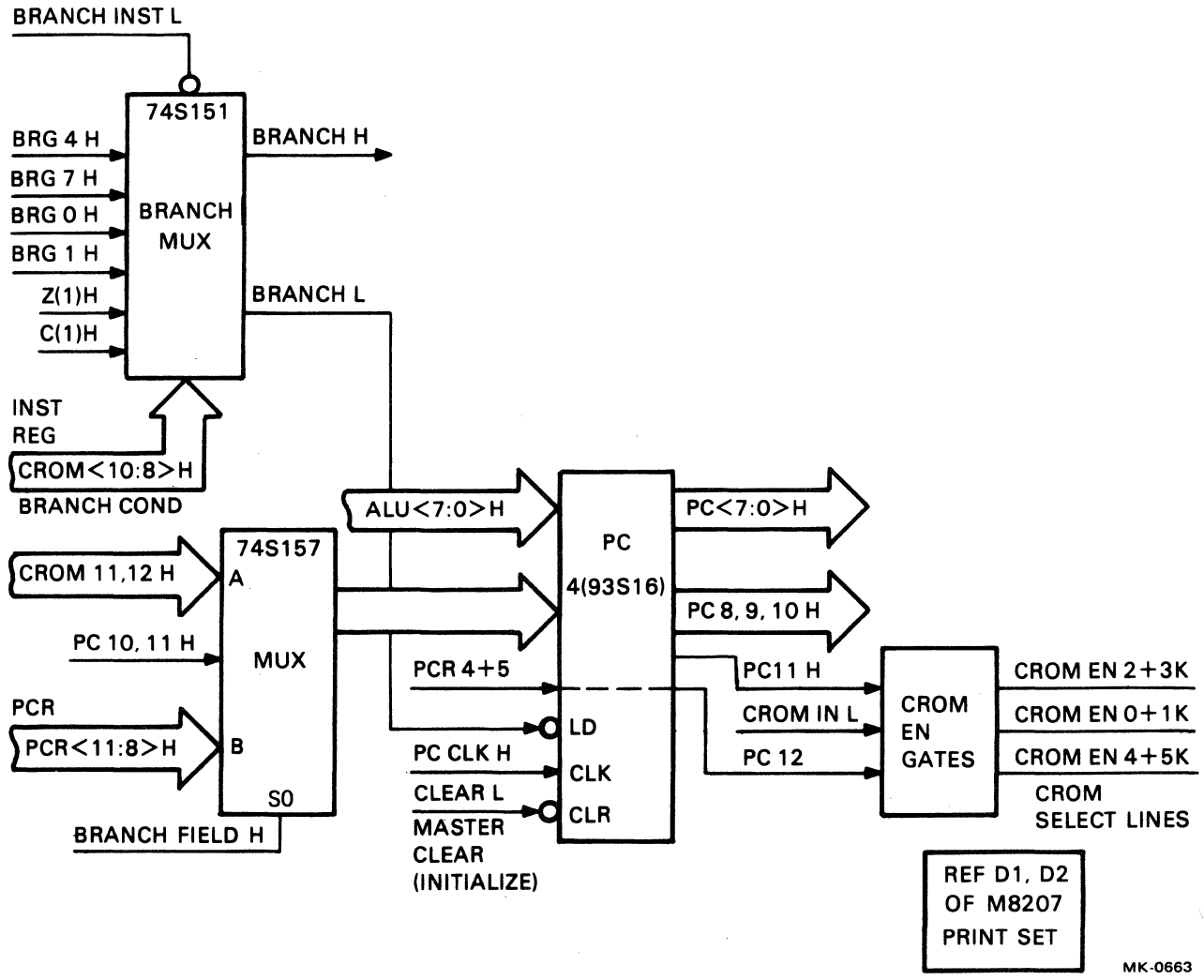


Figure 2-9 Program Counter

The contents of the PC can be read by the program through INBUS* registers 12 and 13 for the instruction that is being executed. The PC is cleared with CLEAR L during a master clear and bus initialization. Branch control is accomplished through an 8-to-1 (74S151) multiplexer driven by CROM bits 8 through 10 from the IR. The branch MUX is only enabled during branch instructions.

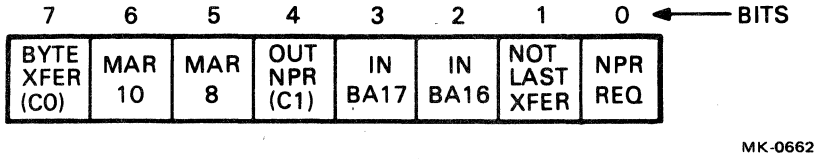
2.9 NPR AND MISCELLANEOUS REGISTERS (Figures 2-10 and 2-11)

The microprocessor has two registers that are used for microprocessor hardware control and status, INBUS*/OUT* 10 and 11. Register 10 controls the NPR circuitry and starts the NPR timing. Register 11 is a status register. The following discussion describes the bits and functions of both registers.

NOTE

Circled numerics are used in the following discussion to associate those areas of text with corresponding areas in Figures 2-10 and 2-11.

2.9.1 Microprocessor NPR Control Register (Register 10)



- *Bit 0* is the NPR bit and can only be written to a 1 by the microprocessor; clearing is done by the hardware. The NPR flip-flop (Fig. 2-11 (1)) is set through a NAND gate, via the flip-flop's preset input when the program writes a 1 to bit 0. This starts the NPR cycle through NPR control chip (Fig. 2-10 (2)), which generates BUS NPR, then waits for the request to be granted via NPG IN. The microprocessor then becomes bus master and the NPR MASTER H signal triggers the START one-shot (Fig. 2-10 (3)), a 200 ns timer. The 200 ns time period allows: (1) the multiplexers to select the address and data from the Multiport RAM, (2) propagation through the Multiport RAM, the bus transceivers, and bus delays, and (3) a 75 ns deskew time for the signals on the bus.

  The trailing edge of the 200 ns pulse from the start timer triggers the second timer CK1 – a 75 ns timer (Fig. 2-10 (4)). The leading edge of CK1 clocks the NPR DONE flip-flop (Fig. 2-10 (5)). The data input is clear at this time. The trailing edge of CK1 clocks the OMSYNC flip-flop (Fig. 2-10 (6)), which generates master sync on the bus and enables CK2 – a 75 ns timer (Fig. 2-10 (7)). CK2 is triggered after slave sync is returned and presets the NPR DONE flip-flop.

  The leading edge of CK2 does nothing; the trailing edge clocks the NPR LOAD MULTIPORT flip-flop (Fig. 2-10 (8)) during a transfer in and retriggers CK1. The leading edge of CK1 now clocks NPR DONE which is set and the data input is true, keeping NPR DONE in a set state. The trailing edge of CK1 clears OMSYN and retriggers CK2 which clears NPR LOAD MULTIPORT and retriggers CK1. CK1 clears NPR DONE which clocks the NPR RQ flip-flop whose input is grounded.

  Clearing the NPR RQ flip-flop clears the NPR control unless this is not the last transfer, in which case NOT LAST TRANSFER signal is set (Figures 2-10, 2-11 (9)). When OMSYN is set, an 18 microsecond timer is started (Fig. 2-10 (10)). If this timer times out before slave sync is returned, the TIME OUT flip-flop is set (Fig.2-10 (11)). This clears the NPR control circuitry as if slave sync had been returned, and presets a flip-flop (Fig. 2-11 (12)) that generates a signal line called NON EX MEM H gated to the DMUX.

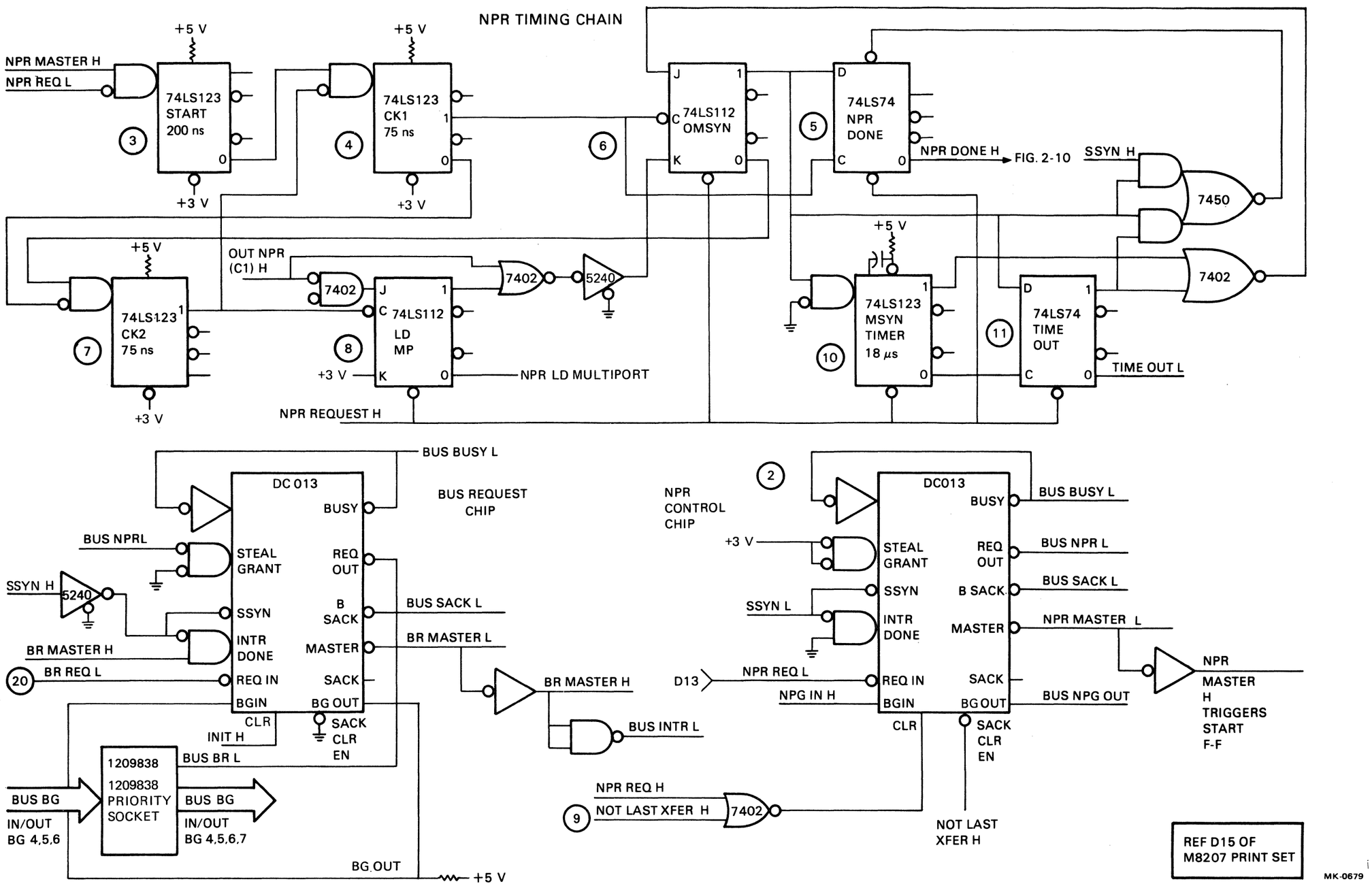


Figure 2-10 Interrupt and NPR Control

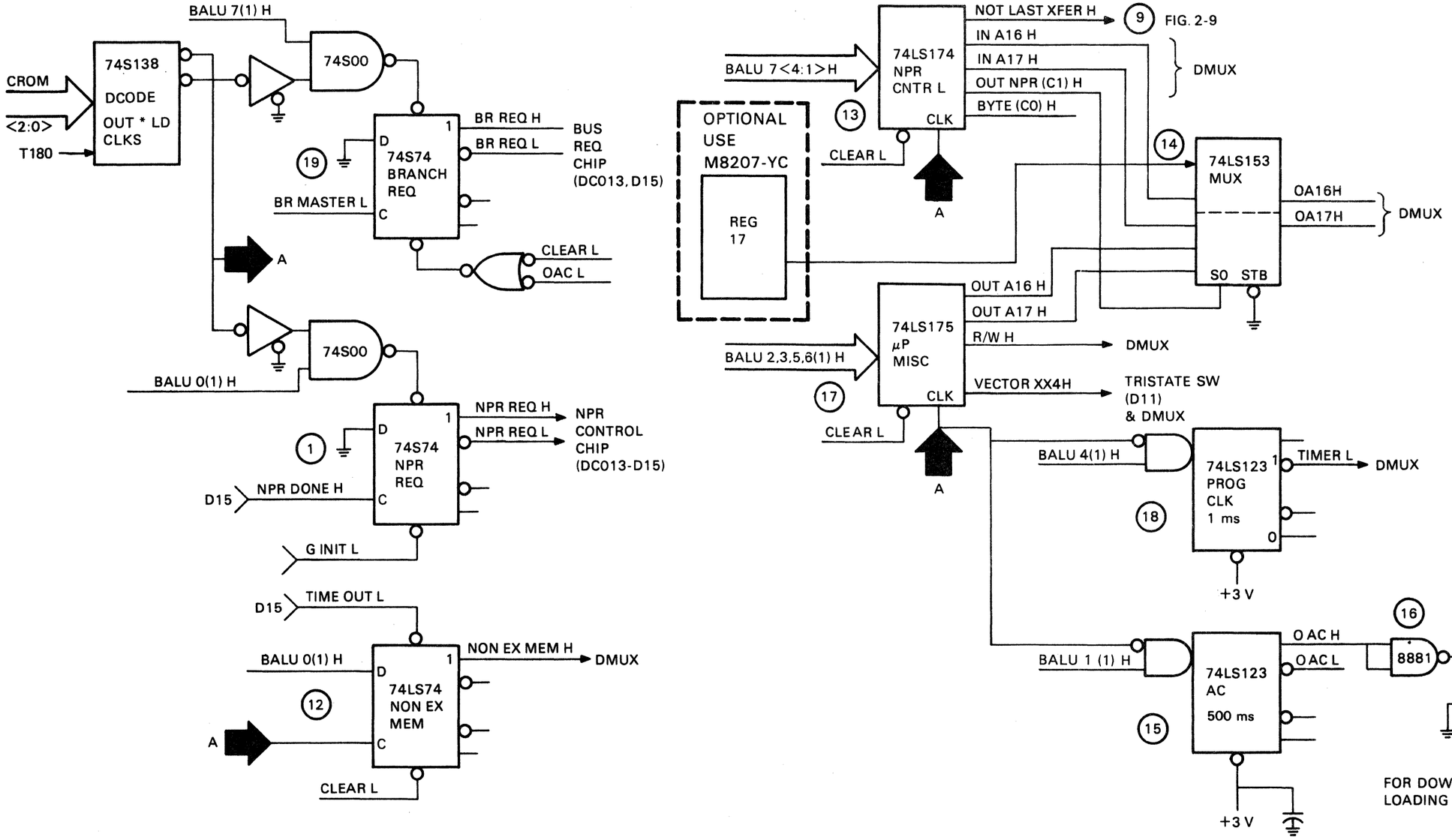


Figure 2-11 Miscellaneous Registers

- *Bit 1* is the not last transfer bit which is part of a hex D flip-flop used for register 10. The output of the flip-flip goes to the NPR circuit and keeps the NPR control chip (DC013) from clearing at the end of an NPR cycle. This maintains BUS BUSY for multiple Direct Memory Access (DMA) transfers. The NOT LAST XFER H signal line is also sent to the DMUX (D7 of Print Set) and should be cleared when the last NPR RQ is set.

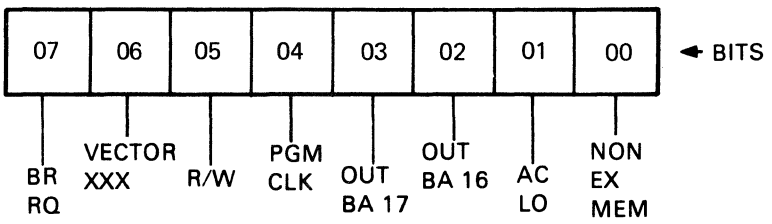
- *Bits 2 and 3* are NPR address bits 16 and 17 respectively. These bits are loaded into the hex D flip-flop used for register 10 via the BALU (Fig.2-11 (13)). Outputs of the hex D flip-flop (IN16 H, IN17 H) are gated into a MUX (Fig. 2-11 (14)) which is controlled by OUT NPR (C1) H. If this control signal is not set, bits 2 and 3 of register 10 are used as the upper two bits of the NPR address. The MUX upper two bit outputs OA16 H and OA17 H are also gated into the DMUX to be read by the microprocessor.

- *Bit 4* is the OUT NPR (C1) H bit of register 10 used to control the direction of an NPR transfer. Bit 4 is used by the multiplexers to select: (1) the registers to be used for the address of an NPR transfer, and (2) the register to be loaded for an IN (read) transfer, or the register to be loaded onto the bus for an OUT (write) transfer. Bit 4 is also loaded into the hex D flip-flop (Fig. 2-11 (13)) and the output, OUT NPR (C1) H controls bits 2 and 3 as described in the previous paragraph. The control line is also gated into the DMUX to be read by the microprocessor.

- *Bits 5 and 6* are read only bits showing MAR bits 8 and 10, which are page and field overflow bits. These bits are duplicated in register 15 and kept in register 10 for compatibility with previous versions of the microprocessor family.

- *Bit 7* is used for byte transfers. This bit uses the hex D flip-flop and can be read through the DMUX by the microprocessor.

2.9.2 Microprocessor Miscellaneous Register 11 (Register 11)



MK-0680

- *Bit 0* is a nonexistent memory error. It uses a 74LS74 flip-flop that can be set and cleared by the microprocessor. When a time out occurs during an NPR transfer, the flip-flop is preset, indicating that slave sync was not returned within 18 microseconds. Bit 0 can be read through the DMUX.

- *Bit 1* is AC LO which controls a one-shot (Fig. 2-11 (15)) that can be used to generate an ac low on the UNIBUS, and a control signal to TP1 (spade lug located near BERG Connector). This control bit should not be used for non-DEC programming unless the AC LO function to the UNIBUS is disabled by removing jumper W3. Field Service representatives should check this function if unusual problems are occurring with AC LO.

  The AC LO feature is used for initializing the CPU and causes a one-half second AC LO pulse on the UNIBUS (if W3 is installed). During this time, the microprocessor protects itself from initialization by the UNIBUS INIT signal.

The output of the one-shot is also gated into an 8881 driver (Fig. 2-11 (16)) connected to TP1. The output of the 8881 is GND when the AC LO bit is asserted and requires a pull-up resistor (typically 1K ohms at 5 volts). The intention is to drive a boot module directly without using AC LO on the bus. The OAC H output is also input to the DMUX as a status bit.

- *Bits 2 and 3* are OUT NPR address bits 16 and 17 respectively. These bits are loaded into a quad D flip-flop (Fig. 2-11 (17)) used for register 11 via the BALU. Outputs OUT A16 H and OUT A17 H are gated into the same MUX as IN A16 H and IN A17 H (see bits 2 and 3 of the NPR Control Register 10). The MUX is used to select the extended IN or OUT address bits for NPR transfer (see bit 4 of register 10). These bits can also be read through the DMUX by the microprocessor.

- *Bit 4* is the program clock which is a 1 ms one-shot (normally the function can change with the "Y" version) (Fig. 2-11 (18)) used by the microprogram for timing. The output of the one-shot, TIMER L, can be read through the DMUX by the microprocessor and does not affect the operation of the microprocessor hardware.

- *Bit 5* has no hardware function, but should be reserved for future versions of the microprocessor. The bit is generated via register 11s quad D flip-flop and the output can be read through the DMUX by the microprocessor.

- *Bits 6 and 7* are used for interrupts. Bit 6 selects the vector to be used for the interrupt. If bit 6 is set, the interrupt vector is XX4. If bit 6 is clear, the interrupt vector is XX0. Bit 7 initiates the interrupt.

  The actual bit is written into register 11s quad D flip-flop (Fig. 2-11 (17)). The flip-flop output (XX4/XX0) goes to the tristate switch (D11 of Print Set) for the interrupt vector address, and to the DMUX to be read by the microprocessor. Bit 7 is BR RQ (Fig. 2-11 (19)) which can only be written to a one by the microprocessor. The flip-flop must be cleared by the hardware. A NAND gate is used to clock the data to the preset input of a 74S74 flip-flop. The data input to the flip-flop is tied to ground and used only to clear the flip-flop on the completion of an interrupt. The BR RQ flip-flop is cleared by the CLEAR L or OAC L signals. The output of the BR RQ flip-flop is used to initiate the interrupt through the DC013 chip (Fig. 2-10 (20)). The DC013 uses the priority plug to ensure the proper level, normally level 5. When the DC013 gives up bus mastership, the BR RQ flip-flop is cleared.

2.10 BERG PORT (Figure 2-12)

The BERG Port is an 8-bit parallel port which allows the microprocessor to communicate with a line unit without using the UNIBUS. The port runs at an instruction speed of 180 ns. Eight addresses are used both for read and for write transactions (using the microprocessor as reference).

- Read Transactions: The address is output 20 ns after the beginning of the instruction at time zero (T0). Buffered T0 goes to the port for syncing the line unit to the microprocessor. Data must be valid for a read, 50 ns after the address is set up, and must remain valid until 20 ns after the leading edge of IBR low. IN BUS has been read (IBR low) starting at T120 and ending at T180.

- Write Transactions: The address is valid 20 ns from T0. The data is valid on the BALU lines 140 ns after T0. The data should be clocked on the trailing edge of OUT BUS WRITE (OBW). Recommended interface chips are shown on the last sheet of the *M8207 Print Set.*

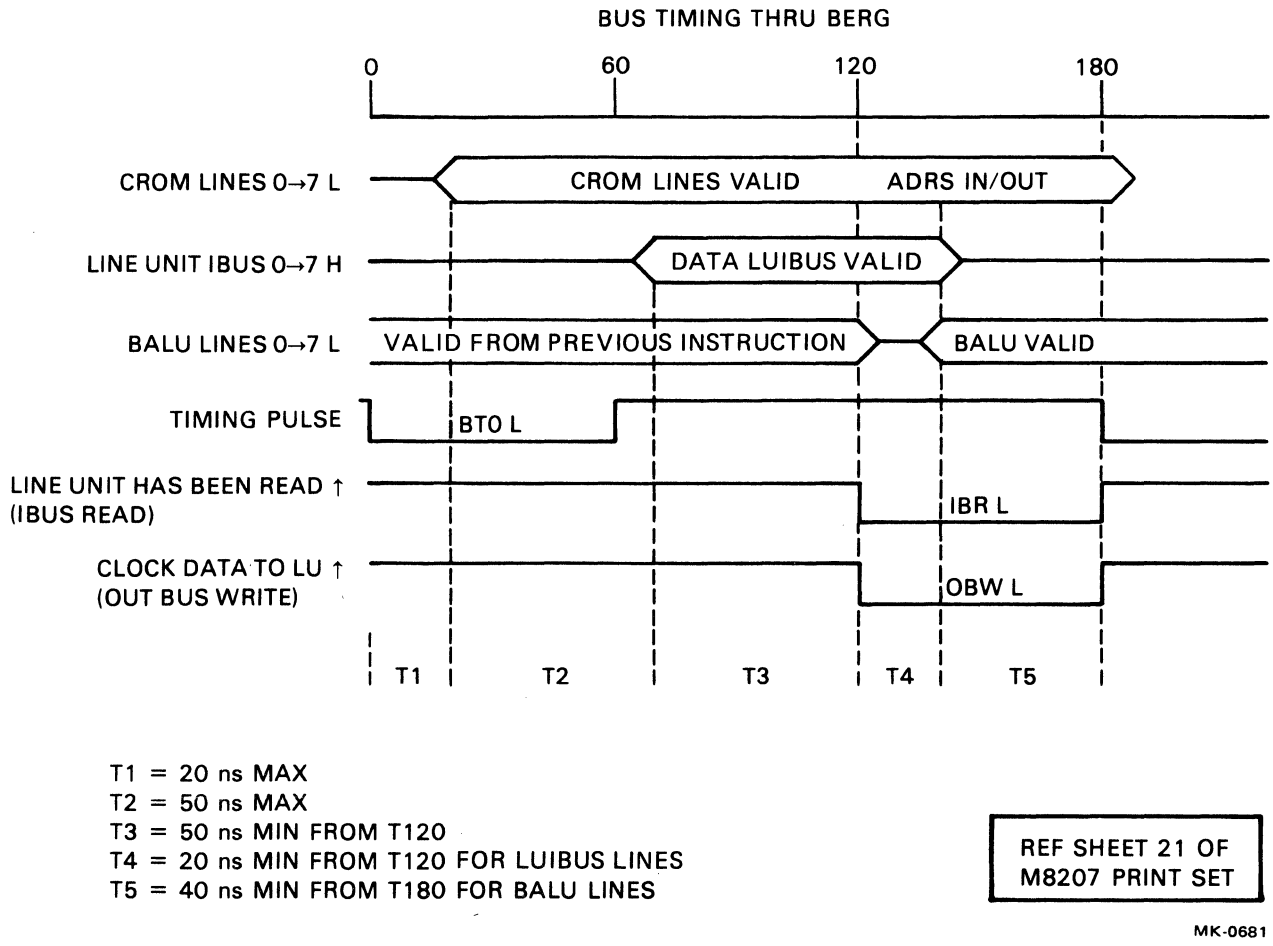


Figure 2-12 Timing Through BERG Port

2.11 BRANCH CONTROL (Figure 2-13)

Three branch instructions are defined by Source Decoder input bits CROM <15:13>:

- Branch Immediate (JMP I)
- Branch Memory (JMP MEM)
- Branch Register (JMP BRG)

When a branch condition is established, BRANCH INST L is asserted. This provides the strobe input to the 8-to-1 branch control multiplexer, which is driven directly by instruction register CROM <10:8> condition bits. The control MUX is disabled during a move instruction as defined by the Source Decoder. In this case BRANCH INST L is unasserted, thereby removing the required strobe input to enable the control MUX.

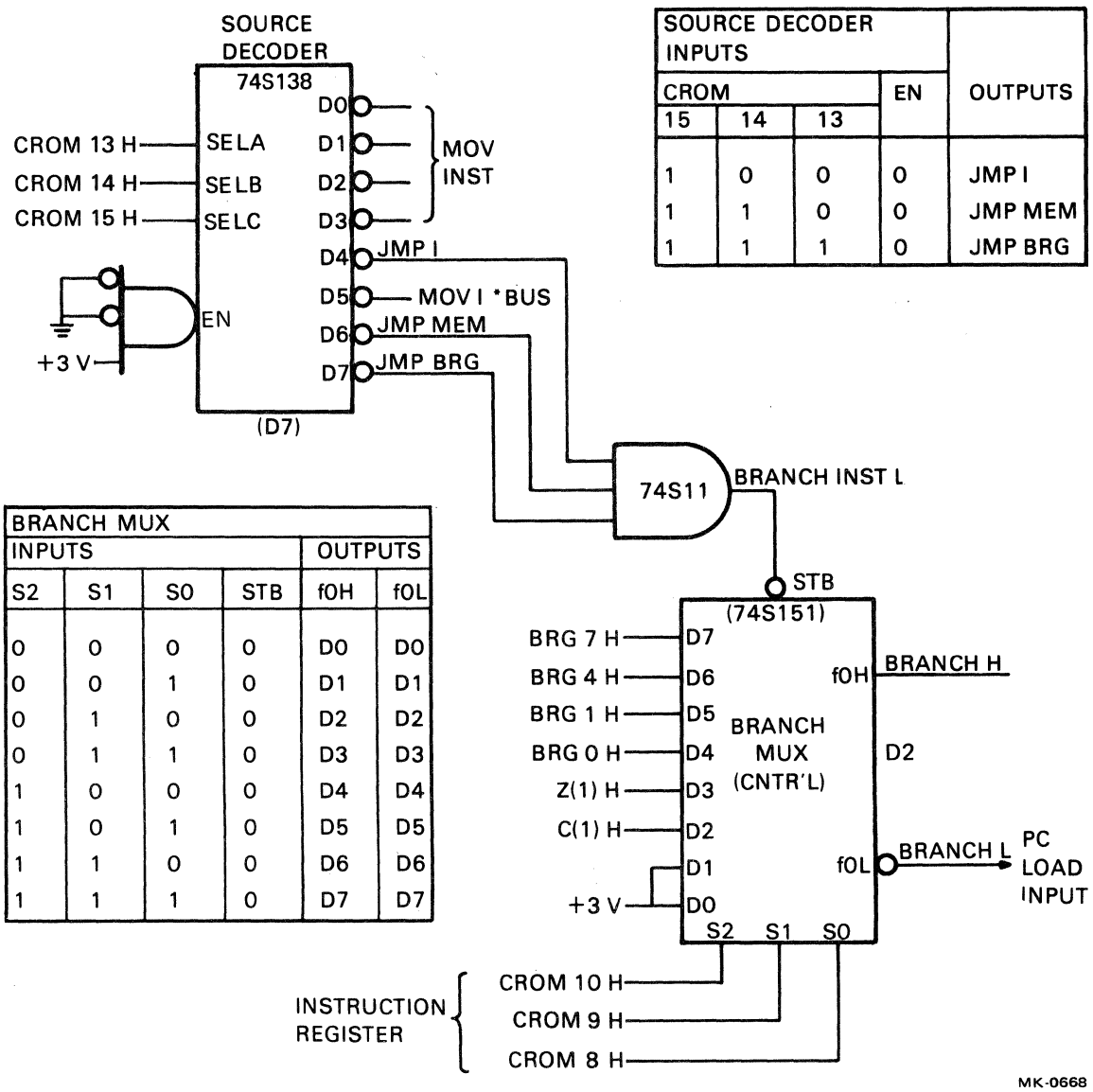
Branch MUX control inputs are:

- +3 Volts (inputs D0, D1) - branch always instructions
- C(1) H bit (input D2) - branch on carry

- Z(1) H bit (input D3) – branch on Z (ALU all ones)
- BRG 0, 1, 4, 7 (H) bits (inputs D4 – D7)

BRG 0 (Branch Field) – During this branch, the upper six bits of the PC are always loaded from the PCR (OUT*13); the page and field bits are changed to the contents of register 13.

BRG 1, 4, 7 – These branches use bits 11 and 12 of the IR for paging bits, but cannot change the field bits. The only way to change the field bits is with Branch 0, unless the field is incremented into.



SOURCE DECODER INPUTS				
CROM			EN	OUTPUTS
15	14	13		
1	0	0	0	JMP I
1	1	0	0	JMP MEM
1	1	1	0	JMP BRG

BRANCH MUX					
INPUTS				OUTPUTS	
S2	S1	S0	STB	f0H	f0L
0	0	0	0	D0	D0
0	0	1	0	D1	D1
0	1	0	0	D2	D2
0	1	1	0	D3	D3
1	0	0	0	D4	D4
1	0	1	0	D5	D5
1	1	0	0	D6	D6
1	1	1	0	D7	D7

Figure 2-13 Branch Control Logic

2.12 BRANCH REGISTER (BRG) (Figure 2-14)

The BRG is an 8-bit register which is used as a temporary storage register. It provides four distinct modes of operation used under branch conditions:

- Parallel Loading
- Shift Right
- Shift Left (ground input locks out shift left mode)
- Inhibit Clock

These modes of operation are determined by the states of SO and SI inputs from the Desination ROM. Refer to Figure 2-13 and note that the operations occur on the positive-going edge of CLK MAR BRG L.

2.13 DESTINATION ROM (DROM) (Figure 2-15)

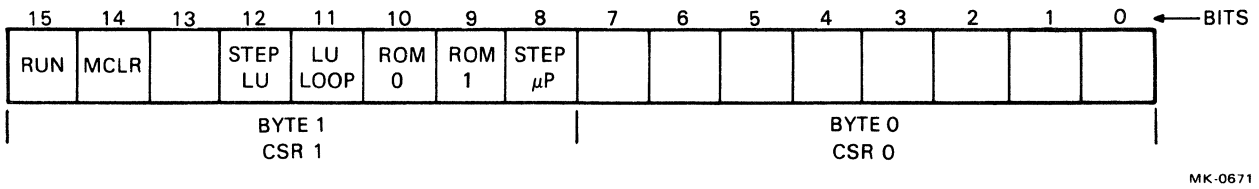
The DROM provides control lines SO and SI to the BRG and increment inputs to the MAR. DROM also provides three additional control outputs determined by CROM input bits <12:8>:

- BRANCH FIELD H output is used as the select input to a 74S157 MUX in the PC circuitry.
- SP and MEM outputs are gated into a 74S158 MUX to generate: (1) the write enable input, WRITE MEM L, for the memory, and (2) the write enable input, WRITE SP L, for the SP.

2.14 CONTROL AND STATUS REGISTERS (CSR0 and CSR6)

The control and status registers are primarily defined by the microcode. Byte 1 of CSR0 is defined by the hardware. When written to by the CPU, the microprocessor can write into the Multiport RAM without affecting the hardware register. CSR6 is only defined when used with byte 1, bit 9, and bit 10 of CSR0.

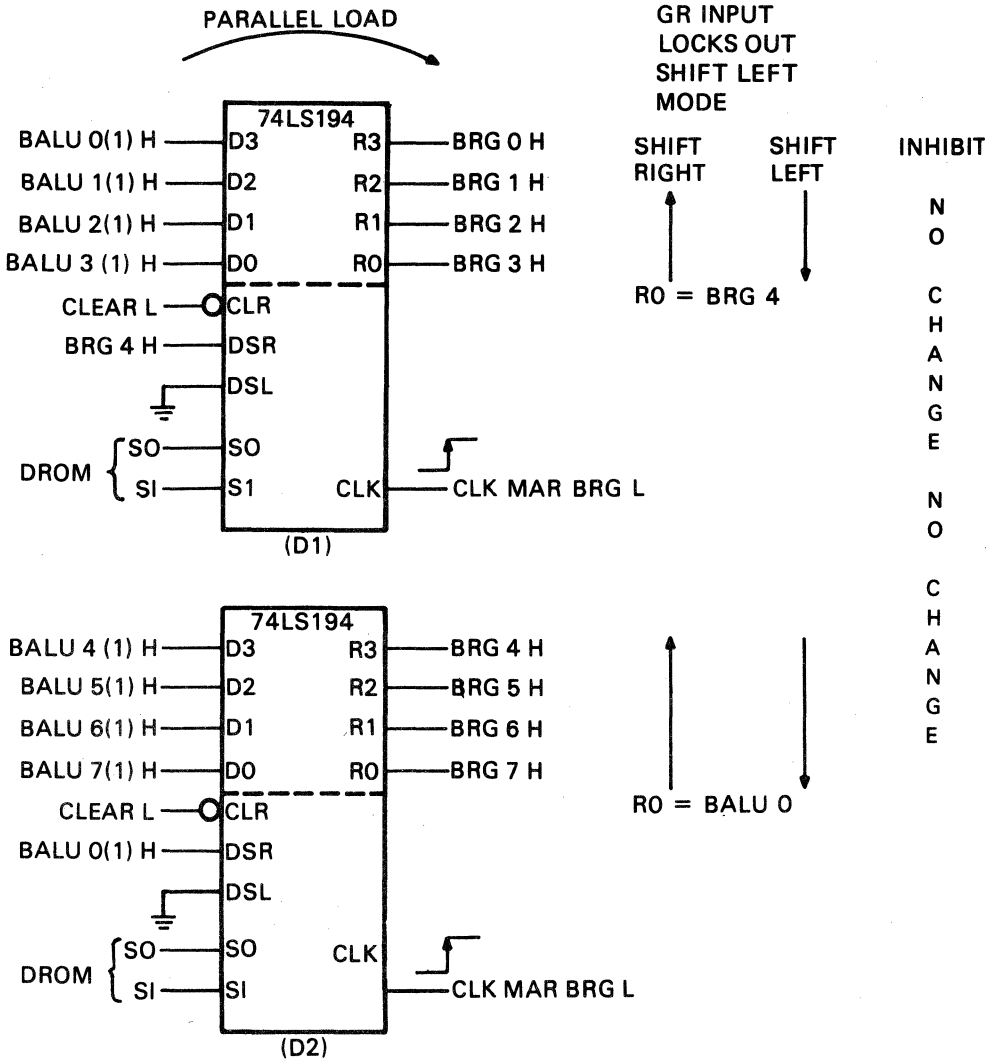
CSR0 - Address 76XXX0



The M8207 uses the floating address area. Actual setting of the address depends on the microcode variation option, of which the M8207 is a part. The space requires eight bytes or four words for CSR0 through CSR6 locations. The XXX in the address is switch selectable and should be set up when the module is installed in the system. Bits and associated CSR1 (high byte of CSR0) are controlled in the following manner. Note that RUN bit 15 can be disabled via the RUN INHIBIT switch (see D16 of Print Set).

NOTE

Switch 8 of the Interrupt Switch Pack is used to disable ALL CSRs.



INPUTS										OUTPUTS				
	MODE			SERIAL		PARALLEL								
CLR	SI	SO	CLK	LEFT	RIGHT	D0	D1	D2	D3	R0	R1	R2	R3	
H	H	H	↑	X	X	D0	D1	D2	D3	D0	D1	D2	D3	①
H	L	H	↑	X	H/L	X	X	X	X	H/L	R0	R1	R2	②
H	H	L	↑	L	X	X	X	X	X	R1	R2	R3	L	③
H	L	L	X	X	X	X	X	X	X	R0	R1	R2	R3	④

1 PARALLEL LOADING – OUT = THE LEVEL OF STEADY STATE INPUTS (BALU BITS)

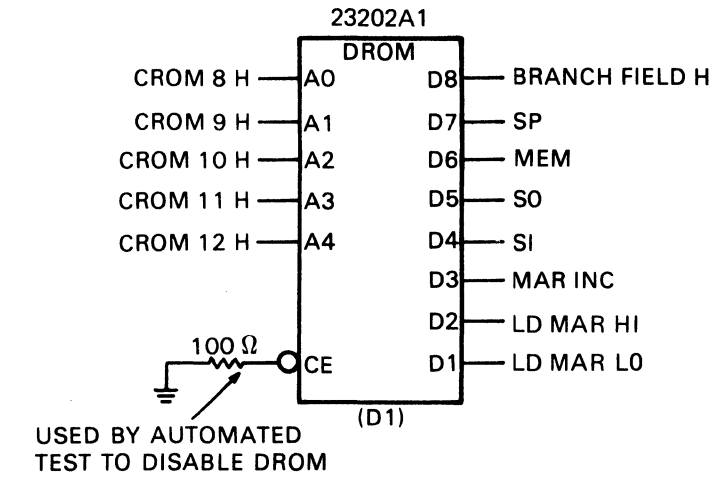
2 SHIFT RIGHT – OUT = R0 IS AT LEVEL OF DSR INPUT R0–R3 SHIFTED RIGHT

3 SHIFT LEFT – OUT = R0–R3 SHIFTED LEFT, DSL AT LOW LEVEL (GND IN) AT VACATED R3 BIT POSITION

4 INHIBIT CLOCK – OUT = NO CHANGE

MK-0670

Figure 2-14 Branch Register



	MICROINSTRUCTIONS	
CROM BITS	BRANCH	MOVE
12,11	BRANCH ADDR BITS	MAR FUNCTION
10,9,8	CONDITION CODE	DESTINATION FIELD

MK-0669

Figure 2-15 Destination ROM

- *BIT 8 (STEP μP)* steps the microprocessor through one instruction cycle.
- *BITS 9, 10, 11, 12 and 15* functions are asserted on the positive-going edge of CSR0 clock input signal.
- *BIT 9 (ROM I)* when cleared generates CROM enable 0 + 1K, 2 + 3K, and 4 + 5K inputs to the control ROM circuitry. Bit 9 when set enables CSR6 to be used as the maintenance register.
- *BIT 10 (ROM O)* is ANDed with A1 and A2 to provide the select input to the data select multiplexers at the UNIBUS data interface. When select input is unasserted, A PORT data is directed to the UNIBUS interface. When select input is asserted, ROM data is selected to verify the contents of the ROMs.
- *BIT 11 (LU LOOP)* connects line unit serial line back to its serial line in (optional with the line unit).
- *BIT 12 (STEP LU)* shifts transmitters/receivers (optional with the line unit).
- *BIT 13* reads/writes in the Multiport RAM; does not perform any hardware functions.
- *BIT 14 (MASTER CLEAR)* clears both the microprocessor and line unit.
- *BIT 15 (RUN)* controls the microprocessor clock via the RUN STEP L signal input to the system clock circuitry (set on Master Clear).
- *BITS 11, 12, 14, and 15* are directly connected to the line unit via the J1 connector. (See D16 of Print Set.)

CSR6 – Address 76XXX6

In maintenance mode, an instruction can be loaded with bit 9 of CSR0. In CROM display, the contents of the ROM can be verified with bit 10 of CSR0 in a set state.

CHAPTER 3
SERVICE

3.1 SCOPE
Maintenance procedures used in servicing an M8207 module are discussed in this chapter.

NOTE
The M8207 performs no useful function without application firmware supplied only by DIGITAL.

There are no built-in maintenance features such as error checking or self-diagnosis.

3.2 MAINTENANCE PHILOSOPHY
The M8207 module is the Field Service Field Replaceable Unit (FRU). Isolation to the module must be done through the use of DEC Diagnostics and/or any microdiagnostics that may have been incorporated in the Control Read Only Memory (CROM). In some applications, ROMs are provided for replacement by a Field Change Order when an error is detected in CROM.

3.2.1 Preventive Maintenance
There is no specific Preventive Maintenance (PM) schedule for the M8207 module. Voltages and connections should be checked during system PM or when problems exist. While special tools are required for PM, standard test equipment may be used. The diagnostics include:

- Microprocessor basic diagnostics (not requiring CROMs) are provided
- Functional diagnostics based on ROM code application

3.2.2 Corrective Maintenance
Standalone diagnostics are provided for checkout of the basic M8207 Microprocessor. These diagnostics check out the M8207 logic and determine whether the microprocessor is capable of running a microprogram from CROM.

Microinstructions are clocked into the Maintenance Instruction Register (MIR) from the UNIBUS, allowing one microinstruction to be executed at a time. Although individual microinstructions are executed at full microprocessor speed, the time between instructions depends upon Central Processing Unit (CPU) execution time.

Basic M8207 diagnostics are designed to run on the basic microprocessor logic with or without CROMs installed.

Functional diagnostics are provided for specific DIGITAL supplied applications implemented in the CROMs. These diagnostics are application oriented, implementing and testing features provided in

microcode as well as in the M8207 hardware. They should be run whenever possible to ensure total option functionality. The functional diagnostics perform a Cyclic Redundancy Check (CRC) on the CROM contents, before starting the microprocessor. CROMs can be diagnosed in 2K segments and may be replaced if required.

NOTE
A 2K segment consists of 2 chips, each 2K- by 8-bits.

When problems occur, the voltage levels should be checked on the backplane,* and all connections should be checked for proper installation and seating. In systems where the M8207 is newly installed or add-ons have been implemented, proper bus configuration must be ensured. Diagnostics should then be run starting with the basic tests and continuing on to the functional diagnostics.

If a customer's application does not work, but all diagnostics run successfully, and the system checks out, the M8207 should be replaced. Diagnostics should then be run on the new M8207. Upon successful completion of the diagnostic, the customer's application can be retried. The possibility that faulty microcode or software drivers exist should be considered.

NOTE
The M8207 will only be supported when used in a DIGITAL supplied subsystem, including DIGITAL supplied microcode.

3.3 MICROPROCESSOR USE WITH LINE UNITS

M8207 diagnostics should be run before any line unit diagnostics. This is necessary because the M8207 provides all access and control for the line unit and could erroneously cause a line unit to appear faulty.

As with M8207 basic tests, microinstructions are loaded and clocked at the CPU execution speed. Execution of individual microinstructions occurs at the microprocessor cycle rate.

3.4 LINE UNIT PORT CABLE CONSIDERATION

The line unit cable (BC08S-1 or BC08R-1) is susceptible to damage from frequent bending and chafing. This cable may be responsible for apparent line unit failures and should be replaced if in doubt.

3.5 M8207 LINE UNIT PORT

Only DEC supplied line units are supported on the line unit port.

3.6 MICRODIAGNOSTICS

Microdiagnostics are not part of the M8207 module. They must be incorporated as part of the application microcode. Recommended microdiagnostic tests are provided in Appendix A. A microdiagnostic failure, in general, is reason to run CPU based diagnostics or replace the indicated module. It should be noted that the microdiagnostic cannot be relied upon for 100% logic failure detection.

The microdiagnostic relies upon a basically operational microprocessor. Some faults in a microprocessor could cause the microdiagnostic to report incorrect error information or, in extreme cases, not to report at all.

*The M8207 microprocessor draws 7 amps at +5 volts. This should be considered, especially when adding a load to a regulator. Nine slot SPC backplanes, such as the DD11-D, permit configurations with power supply overload capability.

Refer to Appendix A for error codes and a description of the internal and interface diagnostics. The intent of the microdiagnostic is to:

- Identify a faulty unit
- Provide error information indicative of the fault

The first intent can help the customer isolate a faulty option under system operation. This could allow reasonable system operation to continue while placing a call for service. And, if the customer can identify the failing subsystem, the Field Service time needed to diagnose the problem would be reduced.

The second intent may indicate which module is defective, effectively reducing diagnostic and repair time.

3.7 ROM REPLACEMENT

The Programmable ROMs (PROMs) on the M8207 are 2K- by 8-bits. The last four bytes on the PROM contain information about the PROM. The first of the last four bytes contains the ASCII Version number. The second byte contains the ROM number (0, 1, 2, 3, 4, or 5). The third and fourth bytes contain a negative CRC-CCITT word for that PROM. The two static diagnostics written for the M8207 do not check the PROMs. The functional diagnostics written for a specific option associated with the M8207 should be used for verification of ROM contents and ROM placement checking.

Figure 3-1 is an extract of the M8207 module component layout; component side view of upper connector pins and ROM portion. Pin 1 is identified on each chip to ensure proper orientation and installation of a good chip should it become necessary. A ROM Kit including ROM replacement procedures is supplied with the shipment.

Table 3-1 shows the relationship of the E numbers and address range of chips 0 through 5.

Table 3-1 Chip Address Range

Chip Location	Chip No.	Byte	Address Range
E03	0	LOW	$0000-3777_8$
E02	1	HIGH	$0000-3777_8$
E04	2	LOW	$4000-7777_8$
E01	3	HIGH	$4000-7777_8$
E05	4	LOW	$10000-13777_8$
E14	5	HIGH	$10000-13777_8$

CAUTION

Care should be exercised when replacing ROMs.

- **Avoid touching chip surface**
- **Handle ROMs by placing fingers on either ends of the chip**
- **Do not bend pins during insertion**

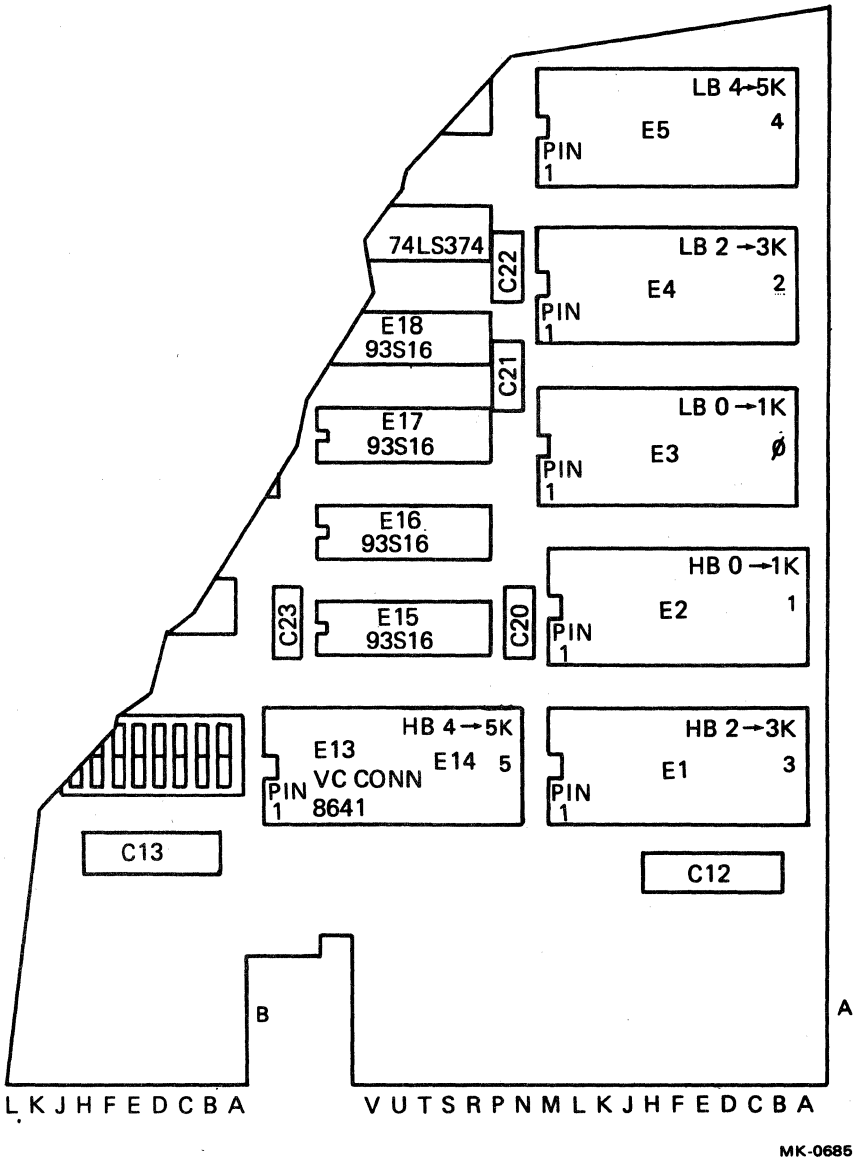


Figure 3-1 Extract of M8207 Module Component Layout

Refer to Figure 3-1 and check for:

- Proper chip orientation
- Proper chip location

Following is a sample test to check out ROM 0 for CRC/CCITT (ROM contents) and ROM position.

```
        MOV     #0,ROMN         ;ROM NUMBER
        MOV     #0,CADDR        ;GET STARTING ADDR.

        MOV     #-1,CWORD       ;INIT CRC WORD.

10$     JSR     PC,GWORD        ;GET FIRST BYTE.

        MOVB    @SEL6,WORDT     ;STORE FIRST BYTE.
        INC     CADDR           ;UPDATE ADDR.
        JSR     PC,GWORD        ;GET A BYTE.
        MOVB    @SEL6,WORDT+1   ;STORE IN HIGH BYTE OR
                                WORDT
        INC     CADDR           ;UPDATE ADDR.
        CMP     CADDR,#3777+1   ;AT END?
        BEQ     20$             ;YES,EXIT LOOP.

        JSR     PC,CRCR         ;NO-CALCULATE CRC ON THIS
                                WORD.
        BR      10$             ;LOOP

20$     COM     CWORD           ;STORED CRC WORD IS
                                COMPLEMENT.
        CMP     CWORD,WORDT     ;EQUAL?
        BEQ     30$
        ERROR                   ;ROM CRC WORD BAD.

30$     MOV     #3775,CADDR     ;SET ADDRESS FOR ROM
                                NUMBER.
        JSR     PC,GWORD        ;READ ROM
        CMPB    ROMN,WORDT      ;IS ROM NUMBER CORRECT?
        BEQ     40$
        ERROR

40$
        END TST
```

```
;*****************************************************************************
;*
;* GWORD          ROUTINE TO READ A WORD FROM ROM ON THE M8207
;*          CALL=JSR PC,GWORD
;*          ENTER WITH ADDRESS INCADDR
;*          EXIT WITH DATA IN SEL6
;*****************************************************************************
;
```

```
GWORD:  CLR     @SEL0                   ;INIT
        MOVB    CADDR+1,@SEL2           ;SET HIGH BYTE OF ADDRESS
        JSR     R5,ROMCLK
        .WORD   121053                  ;MOV IBUS* 2 TO OBUS*13
        BIC     #377,1$                 ;STRIP ADDR FIELD.
        BISB    CADDR,1$                ;ADD IN IMM ADDR.
        JSR     R5,ROMCLK               ;GO DO BRANCH

1$      .WORD   100000                  ;BRANCH EXT PUTS ADDR. IN
                                        PCREG.
        BIS     #2000,@SEL0

        RTS PC                          ;EXIT

;*************************************************************************
;*
;*ROMCLK          ROUTINE TO SINGLE STEP AN INSTRUCTION IN THE
;*                M8207
;*                CALL=JSR R5, ROMCLK
;*                   .WORD INSTR
;*                      RETURNS HERE
;*************************************************************************

ROMCLK:

        BISB    #2,@SEL1                ;SET ROMI
        MOV     (R5)+,@BSEL 6           ;SET INSTRUCTION
        BISB    #3,@BSEL1               ;CLOCK INSTR.
        BICB    #7,@BSEL1               ;CLEAR
        RTS     R5

;*************************************************************************
;*CRCR        ROUTINE TO TAKE 16 BITS OF DATA AND CONVERT THEM
;*            INTO PART OF A SERIAL STREAM THAT WE'RE CALCULATING
;*            A CRC-CCITT WORD FROM.
;*            CALL=JSR PC,CRC
;*            ENTER WITH 16 BITS IN WORDT
;*            EXIT WITH 16 BIT WORD IN CWORD
;*            NOTE::  CWORD MUST HAVE BEEN INITED TO -1 PRIOR TO
;*            FIRST CALLING THIS ROUTINE
;*************************************************************************

CRCR:   MOV     R1,-(SP)                ;SAVE GEN REGS R1,R2
        MOV     R2,-(SP)
        MOV     #16.,R2
```

```
10$     CLC
        ROR     CWORD
        ROR     WORDT
        BVC     20$

        MOV     #102010,R1              ;CRC/CCITT POLYNOMIAL
        BIC     CWORD,R1
        BIC     #102010,CWORD
        BIS     R1,CWORD

20$     DEC     R2
        BGT     10$
        MOV     (SP)+,R2                ;RESTORE GEN REGS

        MOV     (SP)+,R1

        RTS PC                          ;EXIT
```

APPENDIX A
MICRODIAGNOSTIC TESTS

A.1 INTRODUCTION

The microdiagnostics contained in this Appendix were written for the M8207 Microprocessor. The following tests are recommended for all M8207 applications.

NOTE

The microdiagnostics provided in this section are EXAMPLES and are not to be construed as the ONLY IMPLEMENTATION.

A.2 HARDWARE REQUIREMENTS

The following hardware is required to run the microdiagnostics.

- Central Processing Unit (CPU)
- M8207
- 16K Memory

A.3 RELATED SOFTWARE

Internal microdiagnostics need no external supporting software. Interface microdiagnostics for PDP-11s require "CZDMTC0 DMP11 FCTNL TST#1.

A.4 INTERNAL MICRODIAGNOSTICS

- Internal microdiagnostics are written in KDA assembler format that run on initialization within the DMP11 subsystem. These diagnostics are go/nogo type.

- Errors are isolated to the module level. When an error is detected, the microdiagnostic attempts to insert an error number in BSEL4 and enter an infinite loop. The CPU software may test the run bit and error number at the end of the test, or a visual inspection can be performed by evaluation of the Control and Status Registers (CSRs).

- Good and bad data are entered in BSEL 3 and 5 when applicable.

- The current test number is entered in BSEL 6.

- Test numbers are not in sequential order.

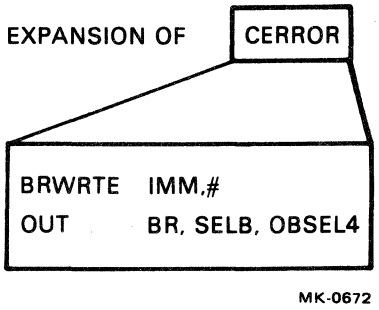
NOTE

If a hardware failure prevents the microdiagnostics from reporting an error indication, some or all of the above will not occur.

A.5 INTERNAL MICRODIAGNOSTIC TEST PROCEDURES

In the following microdiagnostic tests, CERROR is a macro that, when expanded, generates a current error number, outputs this number to BSEL 4, and increments the error number to identify the next error.

Example:



MK-0672

A.5.1 Branch Test #1 - Branch False and Branch Always

Errors in this test may be caused by the failure of a BRWRTE instruction and/or the Arithmetic Logic Unit (ALU), as well as the branch not operating properly.

```
START:

            BRWRTE   IMM,0              ;
            OUT      BR,SELB,OBSEL1     ;ZERO MASTER CLEAR BIT
                                        ;
DBRANC:     BRWRTE   IMM,143            ;SHOW BRANCH TEST
            OUT      BR,SELB,OBSEL6     ;
            CERROR                      ;ERROR #1 TO BSEL 4 IN CASE OF FAILURE
DBR.F:      BRWRTE   IMM,0              ;LOAD 0 IN BREG
            Z        DBRERR             ;SHOULD NOT BRANCH, Z BIT CLEAR.
            BRWRTE   IMM,376            ;SET ALL BUT BIT 0.
            BR0      DBRERR             ;NOW BRANCH ON BIT 0, IF ERROR.
            BRWRTE   IMM,375            ;SET ALL BUT BIT 1.
            BR1      DBRERR             ;ONLY AN ERROR WILL CAUSE A BRANCH.
            BRWRTE   IMM,357            ;SET ALL BUT BIT 4.
            BR4      DBRERR             ;ONLY AN ERROR WILL CAUSE A BRANCH.
            BRWRTE   IMM,177            ;SET ALL BUT BIT 7.
            BR7      DBRERR             ;ONLY AN ERROR WILL CAUSE A BRANCH.
                                        ;
            ALWAYS DBRFE                ;NOW BRANCH TO END OF TEST.
                                        ;IF BRANCH ALWAYS FAILS, END UP IN
                                        ;ERROR ROUTINE
DBRERR:                                 ;
            ALWAYS DBRERR               ;LOOP HERE ALWAYS
                                        ;
DBRFE:                                  ;END OF TEST
                                        ;
```

A.5.2 Branch Test #2 - Branch True

Errors in this test may be caused by the same failures defined in Test #1.

```
DBR.T:    BRWRTE   IMM,200          ;SET BIT 7
          BR7      DBR.T2           ;BR IF KNOWN SET
          ALWAYS   DBRTER           ;ELSE GOTO ERROR.
DBR.T2:   BRWRTE   IMM,20           ;SET BIT 4.
          BR4      DBR.T3           ;BR IF KNOWN SET.
          ALWAYS   DBRTER           ;ELSE GOTO ERROR.
DBR.T3:   BRWRTE   IMM,2            ;SET BIT 1
          BR1      DBR.T4           ;BR IF KNOWN SET.
          ALWAYS   DBRTER           ;ELSE GOTO ERROR.
DBR.T4:   BRWRTE   IMM,1            ;SET BIT 0
          BR0      DBR.T5           ;BR IF KNOWN SET.
          ALWAYS   DBRTER           ;ELSE GOTO ERROR.
DBR.T5:   SP       IMM,377,SP17     ;SET ALL BITS, CAUSE Z BIT SET.
          Z        DBR.T6           ;BR IF Z BIT SET.
          ALWAYS   DBRTER           ;
DBR.T6:   SP       BR,INCA,SP17     ;SET "C" (CARRY BIT).
          C        DBR.T7           ;BR IF C BIT SET.
          ALWAYS   DBRTER           ;ELSE GOTO ERROR.
DBR.T7:   ALWAYS   DBR.T8           ;DOUBLE CHECK THAT BR ALWAYS WORKS

DBRTER:                             ;
          CERROR                    ;ERROR #2 IF BRANCH TEST (TRUE) FAILS
          ALWAYS   DBRTER           ;LOOP HERE
DBR.T8:                             ;END OF TESTS
```

A.5.3 Extended Branch Test - Branch Field

This test performs branches to all fields and returns.

```
DBREX0    BRWRTE   IMM,135          ;SET UP TEST NUMBER IN BSEL6
          OUT      BR,SELB,OBSEL6   ;
          JMPEXT   DFLD0            ;BRANCH TO FLD 0
          CERROR                    ;ERROR #3 IF BRANCH EXT FIELD 0 FAILS
          ALWAYS   DBREXR           ;ERROR
DBREX1:   JMPEXT   DFLD1            ;BRANCH TO FIELD 1
          CERROR                    ;ERROR #4 IF BRANCH EXT FIELD 1 FAILS
          ALWAYS   DBREXR           ;ERROR
DBREX2:   JMPEXT   DFLD2            ;BRANCH TO FIELD 2
          CERROR                    ;ERROR #5 IF BRANCH EXT FIELD 2 FAILS
          ALWAYS   DBREXR           ;ERROR
DBREX3:   JMPEXT   DFLD3            ;BRANCH TO FIELD 3
          CERROR                    ;ERROR #6 IF BRANCH EXT FIELD 3 FAILS
          ALWAYS   DBREXR           ;ERROR
DBREX4:   JMPEXT   DFLD4            ;BRANCH TO FIELD 4
          CERROR                    ;ERROR #7 IF BRANCH EXT FIELD 4 FAILS
          ALWAYS   DBREXR           ;ERROR
DBREX5:   JMPEXT   DFLD5            ;BRANCH TO FIELD 5
          CERROR                    ;ERROR #10 IF BRANCH EXT FIELD 5 FAILS
DBREXR:   ALWAYS   DBREXR           ;LOOP ON ERROR
DBREX:                              ;END OF TEST
```

A.5.4 IBUS/OBUS Test

This test exercises the IBUS/OBUS (*) registers. Patterns are set up in SP1 and SP2. Control is then transferred to subroutine DWRIO.

Patterns used: First pass - SP1 = 125; SP2 = 252
Second pass - SP1 = 252; SP2 = 125
Third pass - SP1 and SP2 = 0

Time per pass: approximately 16.8 ms

```
DIOT:   BRWRTE  IMM,125         ;GET CONSTANT 125
        SP      BR,SELB,SP1     ;PUT INTO SP1
        CERROR                  ;ERROR #11 IF PATTERN 125/252 FAILS IN
                                ;SCRATCHPAD TST
        BRWRTE  IMM,252         ;GET CONSTANT 252
        SP      BR,SELB,SP2     ;PUT INTO SP2
        CALLSB  SP0,DWRI0       ;GO TO SUB DWRIO TO R/W IBUS/OBUS REGS
        CERROR                  ;ERROR #12 IF PATTERN 252/125 FAILS IN
                                ;SCRATCHPAD
DIOT1:  BRWRTE  IMM,252         ;GET CONSTANT 252
        SP      BR,SELB,SP1     ;PUT INTO SP1
        BRWRTE  IMM,125         ;GET CONSTANT 125
        SP      BR,SELB,SP2     ;PUT INTO SP7
        CALLSB  SP0,DWRIO       ;GO TO SUB DWRIO TO R/W IBUS/OBUS REGS
        CERROR                  ;ERROR #13 IF PATTERN 0 FAILS IN
                                ;SCRATCHPAD
DIOT2:                          ;
        BRWRTE  IMM,0           ;NOW WRITE ALL IBUS/OBUS REGS
        SP      BR,SELB,SP1     ;TO ZERO BY CALLING
        SP      BR,SELB,SP2     ;ROUTINE "DWRIO" WITH ZEROS IN
        CALLSB  SP0,DWRIO       ;
DIOT3:  ALWAYS  DSCRTH          ;
```

The DWRIO subroutine writes/reads out patterns in OBUS/IBUS(*). Enter with two patterns; one in SP1, the other in SP2. Return address in SP0 (must be in same page and field). If an error is detected, hang in this routine.

NOTE

In comparing the register against S/B, use a one's complement subtract. This allows branching on the Z bit to be set if the register and S/B are equal.

```
DWRIO:  BRWRTE  SELA,SP1        ;GET FIRST PATTERN
        OUT     BR,SELB,OINCON  ;WRITE IN BSEL0
        OUT     BR,SELB,OOCON   ;BSEL2
        OUT     BR,SELB,OPORT1  ;BSEL4
        OUT     BR,SELB,OPORT3  ;BSEL6
        OUT     BR,SELB,OIDAT2  ;OTHER SIDE #6
        OUT     BR,SELB,OUTDA2  ;#2
        OUT     BR,SELB,IBA2    ;#4
        OUT     BR,SELB,OBA2    ;#6
        BRWRTE  SELA,SP2        ;GET NEXT PATTERN
        OUT     BR,SELB,OLINEN  ;BSEL3
        OUT     BR,SELB,OPORT2  ;BSEL5
```

DWRI0 (Cont):

```
        OUT       BR,SELB,OPORT4     ;BSEL7
        OUT       BR,SELB,OIDAT1     ;OTHER SIDE #1
        OUT       BR,SELB,OUTDA1     ;#3
        OUT       BR,SELB,IBA1       ;#5
        OUT       BR,SELB,OBA1       ;#7
        BRWRTE    IBUS,INCON         ;GET CONTENTS OF BSEL0
        NODST     BR,SUBOC,SP1       ;COMPARE IT AGAINST S/B
        Z         DWRI01             ;BR IF OK
        CERROR                       ;ERROR #14 IF INCON FAILS
        ALWAYS    DWRER              ;ELSE GO TO ERROR
DWRI01: BRWRTE    IBUS,OCON          ;READ BSEL2
        NODST     BR,SUBOC,SP1       ;COMPARE AGAINST S/B
        Z         DWRI02             ;BR IF OK
        CERROR                       ;ERROR #15 IF OCON FAILS
        ALWAYS    DWRER              ;READ BSEL4
DWRI02: BRWRTE    IBUS,PORT1         ;COMPARE AGAINST S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRI03             ;ELSE GO TO ERROR
        CERROR                       ;ERROR #16 IF PORT 1 FAIL
        ALWAYS    DWRER              ;READ BSEL6
DWRI03: BRWRTE    IBUS,PORT3         ;COMPARE AGAINST S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRI04             ;ELSE GO TO ERROR
        CERROR                       ;ERROR #17 IF PORT 3 FAILS
        ALWAYS    DWRER              ;READ OTHER SIDE #1
DWRI04: BRWRTE    IBUS,INDAT2        ;COMPARE AGAINST S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRI05             ;ELSE ERROR
        CERROR                       ;ERROR #20 IF INDAT2 FAILS
        ALWAYS    DWRER              ;READ #3
DWRI05: BRWRTE    IBUS,IODAT2        ;COMPARE AGAINST S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRI06             ;ELSE ERROR
        CERROR                       ;ERROR #21 IF IODAT2 FAILS
        ALWAYS    DWRER              ;READ #5
DWRI06: BRWRTE    IBUS,IIBA2         ;COMPARE S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRI07             ;ELSE ERROR
        CERROR                       ;ERROR #22 IF IIBA2 FAILS
        ALWAYS    DWRER              ;READ #7
DWRI07: BRWRTE    IBUS,IOBA2         ;COMPARE S/B
        NODST     BR,SUBOC,SP1       ;BR IF OK
        Z         DWRIL1             ;ELSE ERROR
        CERROR                       ;ERROR #23 IF IOBA2 FAILS
        ALWAYS    DWRER              ;READ BSEL1
DWRIL1:                              ;
DWRIL2: BRWRTE    IBUS,LINENM        ;COMPARE S/B
        NODST     BR,SUBOC,SP2       ;BR IF OK
        Z         DWRIL3             ;ELSE ERROR
        CERROR                       ;ERROR #24 IF LINENM FAILS
        ALWAYS    DWRER              ;READ BSEL5
DWRIL3: BRWRTE    IBUS,PORT2         ;COMPARE S/B
        NODST     BR,SUBOC,SP2       ;BR IF OK
```

DWRIL3 (Cont):

```
            Z         DWRIL4          ;ELSE ERROR
            CERROR                    ;ERROR #25 IF PORT2 FAILS
            ALWAYS    DWRER           ;READ BSEL7
DWRIL4:     BRWRTE    IBUS,PORT4      ;BR IF OK
            NODST     BR,SUBOC,SP2    ;ELSE ERROR
            Z         DWRIL5          ;READ OTHER SIDE #0
            CERROR                    ;ERROR #26 IF PORT4 FAILS
            ALWAYS    DWRER           ;COMPARE S/B
DWRIL5:     BRWRTE    IBUS,INDAT1     ;BR IF OK
            NODST     BR,SUBOC,SP2    ;ELSE ERROR
            Z         DWRIL6          ;READ #2
            CERROR                    ;ERROR #27 IF INDAT1 FAILS
            ALWAYS    DWRER           ;COMPARE S/B
DWRIL6:     BRWRTE    IBUS,IODAT1     ;BR IF OK
            NODST     BR,SUBOC,SP2    ;ELSE ERROR
            Z         DWRIL7          ;READ #4
            CERROR                    ;ERROR #30 IF IODAT1 FAILS
            ALWAYS    DWRER           ;COMPARE S/B
DWRIL7:     BRWRTE    IBUS,IIBA1      ;BR IF OK
            NODST     BR,SUBOC,SP2    ;ELSE ERROR
            Z         DWRIL8          ;READ #6
            CERROR                    ;ERROR #31 IF IIBA1 FAILS
            ALWAYS    DWRER           ;COMPARE S/B
DWRIL8:     BRWRTE    IBUS,IOBA1      ;BR IF OK, ELSE FALL THRU FOR ERROR
            NODST     BR,SUBOC,SP2    ;PICK UP #0
            Z         DWREE           ;TRY CLEARING RUN BIT
            CERROR                    ;ERROR #32 IF IOBA1 FAILS
DWRER:                                ;
            ALWAYS    DWRER           ;HANG FOREVER ON ERROR
                                      ;
                                      ;
DWREE:                                ;
            RTNSUB    SP0,P0          ;EXIT SUBROUTINE
```

A.5.5 Scratchpad Test

The scratchpad test floats a zero through all scratchpads.

NOTE

As an overhead, main memory locations 0-15 will be tested. To do this: (1) set up memory (loop point), (2) do memory to scratchpad, (3) compare memory to scratchpad, (4) see if test is completed, (5) modify memory, (6) loop back to step 2.

```
DSCRTH:     BRWRTE    IMM,123
            OUT       BR,SELB,OBSEL6
DSP.Z1:     BRWRTE    IMM,<377-17>        :DO LOOP IS 15 TIMES.
            SP        BR,SELB,SP0         ;
            NODST     IMM,0,LDMAR         ;SET MEMORY ADDR ZERO
            NODST     IMM,0,LDMAPG        ;LOAD MAR UPPER
            MEMINC    IMM,177             ;PUT A 177 INTO THE FIRST LOC.
```

```
DSP.Z2:   MEMINC  IMM,377              ;MOVE ALL ONES TO MEMORY
                                       ;UPDATE MAR ADDR.
          SP      BR,INCA,SP0          ;SEE IF DONE
          Z       DSPZ31               ;IF DONE (=377) NEXT STEP
          ALWAYS  DSP.Z2               ;ALSO LOOP
DSPZ31:   MEM     IMM,0                ;LOAD TEST ADDR LOC WITH 0
                                       ;DO MEMORY TO SP NOW
DSP.Z3:   NODST   IMM,0,LDMAR          ;SET MEM ADD TO ZERO
          SP      MEMX,SELB,SP0,INCMAR ;OK FOR THE NEXT 16 INSTRUCTIONS
          SP      MEMX,SELB,SP1,INCMAR ;NOW COPY
          SP      MEMX,SELB,SP2,INCMAR ;MEMORY INTO SCRATCHPAD
          SP      MEMX,SELB,SP3,INCMAR ;REGS THIS TEST,
          SP      MEMX,SELB,SP4,INCMAR ;NOTICE AS YOU GO THROUGH,
          SP      MEMX,SELB,SP5,INCMAR ;IT DEPENDS HEAVILY UPON
          SP      MEMX,SELB,SP6,INCMAR ;MAIN MEMORY BEING OK
          SP      MEMX,SELB,SP7,INCMAR ;IF MAIN MEMORY IS BAD,
          SP      MEMX,SELB,SP10,INCMAR;TROUBLE LIES IN TWO AREAS
                                       ;1. TEST DOES NOT COMPLETE,
          SP      MEMX,SELB,SP11,INCMAR;DUE TO SHIFTS, ETC, PROPER
          SP      MEMX,SELB,SP12,INCMAR;COMPLETION MAY NOT BE
          SP      MEMX,SELB,SP13,INCMAR;ACHIEVED, THEREBY CAUSING
          SP      MEMX,SELB,SP14,INCMAR;PROBABLE "HANG"
          SP      MEMX,SELB,SP15,INCMAR;2. SP ERROR MAY BE INDICATED
          SP      MEMX,SELB,SP16,INCMAR;WHEN REALLY A MEMORY WAS
          SP      MEMX,SELB,SP17,INCMAR;BAD.

                                       ;DO COMPARISON BETWEEN MEMORY+SP
          NODST   IMM,0,LDMAR          ;START AT MEMORY LOC. 0.

          COMP    MEMI,SP0             ;NOW YOU MUST CHECK OUT
          Z       DSP.C1               ;THE SCRATCHPADS AGAINST THE
          CERROR                       ;ERROR #33 IF SP0 BAD
          ALWAYS  DSPERR               ;CONTENTS OF THE MEMORY. SINCE
DSP.C1:   COMP    MEMI,SP1             ;THIS IS A SCRATCHPAD TEST,
          Z       DSP.C2               ;ASSUME THAT MEMORY IS GOOD.
          CERROR                       ;ERROR #34 IF SP1 BAD
          ALWAYS  DSPERR               ;YOU'LL FIND OUT LATER IF THAT
DPS.C2:   COMP    MEMI,SP2             ;WAS A BAD ASSUMPTION.
          Z       DSP.C3               ;
          CERROR                       ;ERROR #35 IF SP2 BAD
          ALWAYS  DSPERR               ;
DSP.C3:   COMP    MEMI,SP3             ;
          Z       DSP.C4               ;
          CERROR                       ;ERROR #36 IF SP3 BAD
          ALWAYS  DSPERR               ;
DSP.C4:   COMP    MEMI,SP4             ;
          Z       DSP.C5               ;
          CERROR                       ;ERROR #37 IF SP4 BAD
          ALWAYS  DSPERR               ;
DSP.C5:   COMP    MEMI,SP5             ;
          Z       DSP.C6               ;
          CERROR                       ;ERROR #40 IF SP5 BAD
          ALWAYS  DSPERR               ;
```

```
DSP.C6:  COMP     MEMI,SP6          ;
         Z        DSP.C7            ;
         CERROR                     ;ERROR #41 IF SP6 BAD
         ALWAYS   DSPERR            ;
DSP.C7:  COMP     MEMI,SP7          ;
         Z        DSPC10            ;
         CERROR                     ;ERROR #42 IF SP7 BAD
         ALWAYS   DSPERR            ;
DSPC10:  COMP     MEMI,SP10         ;
         Z        DSPC11            ;
         CERROR                     ;ERROR #43 IF SP10 BAD
         ALWAYS   DSPERR            ;
DSPC11:  COMP     MEMI,SP11         ;
         Z        DSPC12            ;
         CERROR                     ;ERROR #44 IF SP11 BAD
         ALWAYS   DSPERR            ;
DSPC12:  COMP     MEMI,SP12         ;
         Z        DSPC13            ;
         CERROR                     ;ERROR #45 IF SP12 BAD
         ALWAYS   DSPERR            ;
DSPC13:  COMP     MEMI,SP13         :
         Z        DSPC14            ;
         CERROR                     ;ERROR #46 IF SP13 BAD
         ALWAYS   DSPERR            ;
DSPC14:  COMP     MEMI,SP14         ;
         Z        DSPC15            ;
         CERROR                     ;ERROR #47 IF SP14 BAD
         ALWAYS   DSPERR            ;
                                    ;
DSPC15:  COMP     MEMI,SP15         ;
         Z        DSPC16            ;
         CERROR                     ;ERROR #50 IF SP15 BAD
         ALWAYS   DSPERR            ;
DSPC16:  COMP     MEMI,SP16         ;
         Z        DSPC17            ;
         CERROR                     ;ERROR #51 IF SP16 BAD
         ALWAYS   DSPERR            ;
DSPC17:  COMP     MEMI,SP17         ;
         Z        DSPC18            ;
         CERROR                     ;ERROR #52 IF SP17 BAD
DSPERR:
         ALWAYS   DSPERR            ;LOOP ALWAYS.
DSPC18:                             ;
                                    ;
                                    ;GET MEMORY ADDRESS FROM
         LDMA     MEMX,SELB         ;MEMORY. THIS ADDRESS IS THE
         BRWRTE   MEMX,SELB         ;LAST ADDRESS MODIFIED
         BR0      DSP.M2            ;IF BIT 0 IS SET NORMAL.
DSP.M1:                             ;UPDATE MEMORY ADDR IN MEMORY AND
         MEM      IMM,377           ;REWRITE ALL ONES INTO LAST ADDR.
         LDMA     IMM,20            ;POINT TO ADDR. OF ADDR.
         SP       MEMX,SELB,SP0     ;INCREMENT ADDR.
         SP       BR,INCA,SP0       ;
         MEM      SELA,SP0,LDMAR    ;WRITE IT BACK AND LOAD NEW ADDR.
```

```
DSP.M1 (Cont):
            BRWRTE  IMM,20              ;LOOK TO SEE IF YOU'RE DONE
            SP      BR,SUBOC,SP0        ;
            Z       DSP.EX              ;YES THIS EXIT
            MEM     IMM,177             ;
            ALWAYS  DSP.Z3              ;
DSP.M2:     BROTAT                      ;ROTATE RIGHT
            MEM     BR,SELB             ;
            ALWAYS  DSP.Z3              ;
DSP.EX:                                 ;
```

A.5.6 ALU Test

The ALU test tests all ALU functions.

```
ADD                 A AND B*
ADDC                A AND B* W/CARRY
SUBC                A-B* W/BORROW
INCA                A+1*
APLUSC              A+CARRY*
TWOA                A+A
TWOAC               A+A+CARRY*
DECA                A-1 *
SELA                SELECT A SIDE OF ALU
SELB                SELECT B SIDE OF ALU
AORNB               A<OR NOT>B
AANDB               A<AND>B
AORB                A<OR>B
AXORB               A<EXCLUSIVE OR>B
SUB                 -B(TWO'S COMPLEMENT)
SUBOC               A-B (ONE'S COMPLEMENT)
DALU:       BRWRTE  IMM,151             ;
            OUT     BR,SELB,OBSEL6      ;

DALU.1:     SP      IMM,1,SP1           ;SP1=1
            BRWRTE  IMM,377             ;BREG=377
            BRWRTE  BR,ADD,SP1          ;BREG=0,SP1=1,C=1
            BRWRTE  BR,ADDC,SP1         ;BREG=2,SP1=1,C=0

            BRWRTE  BR,ADD,SP1          ;BREG=3
            SP      IMM,3,SP3           ;SP3=3
            BRWRTE  BR,SUB,SP3          ;BREG=0
            SP      BR,TWOA,SP3         ;SP3=6
            SP      BR,DECA,SP3         ;SP3=5
            SP      IMM,377,SP17        ;SP17=377
            SP      BR,INCA,SP17        ;SP17=0,C=1 ;SETS CARRY
            SP      BR,APLUSC,SP1       ;SP1=2,C=0
            SP      BR,DECA,SP17        ;SP17=377,C=1 ;SETS CARRY
            SP      BR,TWOAC,SP1        ;SP1=5, C=0
            SP      BR,TWOAC,SP1        ;SP1=10 (NO CARRY)
            SP      BR,APLUSC,SP1       ;SP1=10 (NO CARRY)
            BRWRTE  IMM,0               ;BRG=0
            SP      BR,ADDC,SP1         ;SP1=10 (NO CARRY)
            BRWRTE  IMM,11              ;BRG=11,SP17=377
```

DALU.1 (Cont):

```
        SP          BR,INCA,SP17        ;SP17=0, C=1
        SP          BR,SUBC,SP1         ;1-B W/C SP1=377
        SP          IMM,6,SP6           ;SP6=6
        BRWRTE      IMM,3               ;BR6=3
        SP          BR,SUBOC,SP6        ;SP6=2 (ONE'S COMP SUB)
        BRWRTE      IMM,0               ;BRG=0
        BRWRTE      BR,SELA,SP6         ;BRG=2
        SP          BR,INCA,SP1         ;
                                        ;SP1=0,SP17=0,SP3=5
        BRWRTE      BR,ADD,SP1          ;BRG=2
        BRWRTE      BR,ADD,SP17         ;BRG=2
        BRWRTE      BR,ADD,SP3          ;BRG=7
        SP          IMM,7,SP7           ;SP7=7
        BRWRTE      BR,SUB,SP7          ;BRG=7
        SP          BR,SELB,SP0         ;SP0=0
        BRWRTE      BR,SELB,SP0         ;SP0=0
        SPBR        BR,DECA,SP0         ;BR=377
        Z           DALU.3              ;IF BR=377 NEXT TEST. ELSE
                                        ;SOMEWHERE AN ALU M FUNCTION
                                        ;OCCURRED.
        CERROR                          ;ERROR #53 IF ALU BAD
DALU.2: ALWAYS      DALU.2              ;HANG HERE, ONE ALU ERROR.
```

The following section is broken out for logic testing of the ALU AORNB, AANDB, AORB, and AXORB functions.

```
DALU.3: BRWRTE      IMM,253             ;BINARY 10 101 011 (253)
        SP          BR,SELB,SP0         ;PUT IN SP 0 (253)
        BRWRTE      IMM,221             ;BINARY 10 010 001 (221)
        BRWRTE      BR,AANDB,SP0        ;AND BRG = 201
        SP          BR,SELB,SP10        ;SP10=201;10 000 001
        BRWRTE      IMM,160             ;BIG=160 01 110 000
        SP          BR,AORB,SP10        ;SP10=361 11 110 001
        BWRITE      BR,AORNB,SP10       ;BR=377 11 111 111
        SP          BR,SELB,SP0         ;SP0=377
        BWRITE      IMM,377             ;BREG=377
        SP          BR,AXORB,SP0        ;SP0=0, BR=377
        SP          BR,DECA,SP0         ;SP0=377 Z BIT SET
        Z           DALU.4              ;BRANCH IF Z SET
        CERROR                          ;ERROR #54 IF ALU BAD
        ALWAYS      DALU.2              ;ELSE GO TO ERROR
DALU.4:                                 ;END TEST
```

A.5.7 Memory Test (Moving Inversion)

This memory test uses the modified moving inversion.

```
DMEMS:  BRWRTE      IMM,222             ;TEST NUMBER.
        OUT         BR,SELB,OBSEL6      ;
        NODST       IMM,0,LDMAR         ;LOAD LOW MAR TO ZERO.
        NODST       IMM,0,LDMAPG        ;HIGH MAR ZEROED.
        SP          IMM,0,SP0           ;NOW ZERO WHOLE MEMORY.
```

```
DMEMS1:  SPBR     IBUS,MARH,SP1      ;SEE IF THROUGH MEMORY
         BR4      DMEMS3             ;IF OVERFLOW BIT SET, YES
         MEM      MEM1,SELA,SP0      ;ELSE WRITE ZERO INTO MEM.
         ALWAYS   DMEMS1

DMEMS3:  BRWRTE   IMM,200            ;GET STARTING PATTERN
         SP       BR,SELB,SP1        ;PUT INTO SP1
                                     ;SP0 WILL CONTAIN OLD PATTERN

DMEML:   NODST    IMM,0,LDMAR        ;NOW LOAD ADDR ZERO
         NODST    IMM,0,LDMAPG

DMEML2:  BRWRTE   IBUS,MARH
         BR4      DMEML4
         BRWRTE   MEMX,SELB          ;READ MEM LOCATION
         NODST    BR,SUBOC,SP0       ;SEE IF OLD PATTERN IS RETAINED
         Z        DMEML3             ;IF YES WILL BRANCH
         OUT      BR,SELB,OBSEL5     ;WRITE BAD DATA IN SEL5
         BRWRTE   BR,SELA,SP0        ;GET GDDAT
         OUT      BR,SELB,OBSEL3     ;PUT IN BSEL3
         CERROR                      ;OLD PATTERN WAS NOT RETAINED
         ALWAYS   DMEMER             ;POSSIBLE MEMORY DUAL ADDRESS ERROR.
                                     ;ERROR #55 IF MEMORY BAD

DMEML3:  MEM      SELA,SP1           ;WRITE NEW PATTERN
         BRWRTE   MEM1,SELB          ;READ NEW PATTERN
         NODST    BR,SUBOC,SP1       ;WRITE OK?
         Z        DMEML2             ;YES LOOP.
         OUT      BR,SELB,OBSEL5     ;BAD-BDDAT TO BSEL5
         BRWRTE   BR,SELA,SP1        ;GET GDDAT
         OUT      BR,SELB,OBSEL3     ;PUT IN BSEL3
         CERROR                      ;ERROR #56 IF MEMORY BAD
         ALWAYS   DMEMER

DMEML4:  BRWRTE   SELA,SP1           ;GET NEW PATTERN
         BR0      DMEME              ;IF BIT TESTED, EXIT
         SP       BR,SELB,SP0        ;ELSE MAKE IT THE OLD PATTERN
         BRSHFT                      ;THEN ROTATE PATTERN RIGHT
         SP       BR,SELB,SP1        ;THEN WRITE NEW PATTERN
         ALWAYS   DMEML

DMEMER:  ALWAYS   DMEMER             ;LOOP HERE ALWAYS ON ERROR

DMEME:                               ;ESCAPE HERE ON TEST COMPLETE
```

A.5.8 Memory Test (Dual Addressing)

This test is broken into two parts. Part one ensures that addresses within a page will not dual address against each other. Part two ensures that the pages do not dual address against one another. This method is used because 12-bit addressing is implemented and memory storage is only 8-bits wide.

```
DOATST:  BRWRTE   IMM,132            ;TEST NUMBER
         OUT      BR,SELB,OBSEL6
         BRWRTE   IMM,0               ;PICK UP CONSTANTS
         SP       BR,SELB,SP2
```

```
DOATST (Cont):
            NODST   IMM,0,LDMAR         ;LOAD ADDRESS ZERO.
            NODST   IMM,0,LDMAPG

DOL1:       MEMINC  SELA,SP2            ;WRITE EACH MEMORY LOCATION WITH ITS
                                        ;OWN ADDRESS.
            SP      BR,INCA,SP2         ;UPDATE ADDR.
            BRWRTE  IBUS,MARH           ;MAKE SURE YOU DON'T REACH OVERFLOW.
            BR4     DO1                 ;IF YOU DO, EXIT THIS LOOP.
            BRWRTE  IMM,0
            ALWAYS  DOL1                ;ELSE DO NEXT ADDRESS.

DO1:        NODST   IMM,0,LDMAR         ;LOAD ADDRESS ZERO AGAIN.
            NODST   IMM,0,LDMAPG
            BRWRTE  IMM,0               ;PICK UP CONSTANTS.
            SP      BR,SELB,SP1

DOL2:       BRWRTE  MEMI,SELB           ;READ MEMORY
            NODST   BR,SUBOC,SP1        ;READ MEMORY ADDRESS, SEE IF IT
                                        ;CONTAINS ITS OWN ADDRESS BY
                                        ;COMPARING IT AGAINST OUR COUNTER.
            Z       DO2                 ;IT WILL BE 377 AFTER SUBTRACT.
            OUT     BR,SELB,OBSEL5      ;PUT EXPECTED INTO BSEL2
            BRWRTE  BR,SELA,SP1
            OUT     BR,SELB,OBSEL3
            CERROR                      ;DUAL ADDRESS ERROR #57
                                        ;MOST PROBABLY HERE.

DOL3:       ALWAYS  DOL3

DO2:        SPBR    BR,INCA,SP1         ;UPDATE ADDRESS.
            BRWRTE  IBUS,MARH           ;SEE IF YOU'RE THROUGH.
            BR4     DOA                 ;WHEN OVERFLOW SET/
            BRWRTE  SELA,SP1
            ALWAYS  DOL2                ;LOOP UNTIL DONE.

DOA:        NODST   IMM,0,LDMAR         ;LOAD ADDRESS ZERO AGAIN.
            NODST   IMM,0,LDMAPG

            BRWRTE  IMM,0               ;PICK UP CONT
            SP      BR,SELB,SP0

DOA1:       MEM     SELA,SP0            ;NOW WRITE THE CURRENT PAGE
                                        ;NUMBER INTO THE 1ST ADDRESS OF EACH
                                        ;PAGE-LATER READ THEM BACK.

            SPBR    BR,INCA,SP0         ;UPDATE PAGE NUMBER
            BR4     DOA2                ;ON OVERFLOW, EXIT THIS LOOP.
            NODST   BR,SELA,SP0,LDMAPG
            BRWRTE  IMM,0
            ALWAYS  DOA1

DOA2:       BRWRTE  IMM,0               ;PICK UP CONSTANTS.
            SP      BR,SELB,SP0
```

```
DOA3:    NODST    BR,SELA,SP0,LDMAPG
         NODST    MEMX,SUBOC,SP0     ;SEE IF CONTENTS AGREE
                                     ;WITH WHAT WAS EXPECTED
                                     ;FOR IT.
         Z        DOA5

         OUT      BR,SELB,OBSEL3     ;ERROR-PAGE NUMBER IN BSEL2
                                     ;FOR ALL TO SEE.
         BRWRTE   MEMX,SELB          ;READ BAD DATA
         OUT      BR,SELB,OBSEL5     ;PUT INTO BSEL5
         CERROR                      ;PAGE DUAL ADDRESS ERROR #60

DOA4:    ALWAYS   DOA4

DOA5:    SPBR     BR,INCA,SP0        ;UPDATE PAGE NUMBER.
         BR4      DOA6               ;ON OVERFLOW EXIT.
         ALWAYS   DOA3               ;ESX

DOA6:

DIADON:  BRWRTE   IMM,305            ;CODE SPECIFYING COMPLETION OF
         OUT      BR,SELB,OBSEL6     ;INTERNAL MICRO-DIAGNOSTICS.
```

Microdiagnostics END, user INIT code starts.

A.5.9 Field Branch Extended Tests

The following tests are used by the Extended Branch Tests (Section A.5.3). If each test does not return to the relevant Extended Branch Test given in Section A.5.3, the program will loop indefinitely.

FIELD 0 Branch Extended Test

```
DFLD0:   JMPEXT   DBREX1             ;RETURN TO TEST
         CERROR                      ;ERROR #61 IF OBREX1 FAILS
5$:      ALWAYS   5$                 ;LOOP ON ERROR
```

This routine is placed in address range 000 - 777.

FIELD 1 Branch Extended Test

```
DFLD1:   JMPEXT   DBREX2             ;RETURN TO TEST
         CERROR                      ;ERROR #62 IF DBREX2 FAILS
5$:      ALWAYS   5$                 ;LOOP ON ERROR
```

This routine is placed in address range 1000 - 1777.

FIELD 2 Branch Extended Test

```
DFLD2:   JMPEXT   DBREX3             ;RETURN TO TEST
         CERROR                      ;ERROR #63 IF DBREX 3 FAILS
5$:      ALWAYS   5$                 ;LOOP ON ERROR
```

This routine is placed in address range 2000 - 2777.

FIELD 3 Branch Extended Test

```
DFLD3:   JMPEXT   DBREX4          ;RETURN TO TEST
         CERROR                   ;ERROR #64 IF DBREX4 FAILS
5$:      ALWAYS   5$              ;LOOP ON ERROR
```

This routine is placed in address range 3000 - 3777.

FIELD 4 Branch Extended Test

```
DFLD4:   JMPEXT   DBREX5          ;RETURN TO TEST
         CERROR                   ;ERROR #65 IF DBREX5 FAILS
5$:      ALWAYS   5$              ;LOOP ON ERROR
```

This routine is placed in address range 4000 - 4777.

FIELD 5 Branch Extended Test

```
DFLD5:   JMPEXT   DBREX           ;RETURN TO TEST
         CERROR                   ;ERROR #66 IF DBREX FAILS
5$:      ALWAYS   5$              ;LOOP ON ERROR
```

This routine is placed in address range 5000 - 5777.

A.6 INTERFACE DIAGNOSTICS

Interface diagnostics are only run after completion of internal diagnostics. These diagnostics are started by an external exercisor, the CPU code designed to interface with this code. Any attempt to run the interface diagnostic code without the proper external software could prove fatal to these tests. All DMP11 UNIBUS registers are redefined.

TEST A - UNIBUS Window Register Test

- Write all ones in the registers
- Enter after call
- Exit when CPU puts a zero into BSEL 0
- 11 code should look at data when BSEL 7 is equal to 377

```
DTSTA:   BRWRTE   IMM,377         ;PICK UP ALL ONES CONSTANT
         OUT      BR,SELB,OINCON  ;PUT INTO ALL WINDOWS
         OUT      BR,SELB,OOCON
         OUT      BR,SELB,OLINEN
         OUT      BR,SELB,OPORT1
         OUT      BR,SELB,OPORT2
         OUT      BR,SELB,OPORT3
         OUT      BR,SELB,OPORT4

DTSTAL:  SP       IBUS,INCON,SP0  ;READ BSEL0
         Z        DTSTAL          ;LOOP WHILE = 377
```

TEST B - UNIBUS Under Register Test

- Write all zeros into the register
- Enter from last test completion

- Write all registers to zero (except main)

- Loop until PDP-11 verifies data, and either puts code 377 into bit 7 set (remaining bits = 0), or into BSEL 0 to loop on the last two tests

```
DTSTB:    BRWRTE   IMM,0              ;PATTERN=0
          OUT      BR,SELB,OINCON     ;WRITE EACH WINDOW TO ZERO
          OUT      BR,SELB,OOCON
          OUT      BR,SELB,OLINEN
          OUT      BR,SELB,OPORT1
          OUT      BR,SELB,OPORT2
          OUT      BR,SELB,OPORT3
          OUT      BR,SELB,OPORT4

DTSTBL:   SPBR     IBUS,INCON,SP0     ;READ BSEL0
          BR7      DTSTBE             ;IF
          ALWAYS   DTSTBL             ;LOOP UNTIL BIT 7 SET

DTSTBE:   SPBR     IBUS,INNCON,SP0    ;REREAD BSEL0 IN CASE OF CHANGE
          Z        DTSTC              ;IF 377 THEN NEXT TEST
          BR7      DTSTA              ;IF BIT 7 SET LOOP LAST TWO TESTS
          ALWAYS   DTSTBL             ;ELSE, LOOP FOR PDP-11 INDICATOR
```

TEST C - Interrupt the PDP-11 at Vector XX0

- To enter this test, the PDP-11 code must have exited Test B
- The PDP-11 must set up interrupt vector XX0
- The PDP-11 must set BSEL 3 equal to 377
- The microcode, upon seeing 377, generates an interrupt
- The microcode then exits test

```
DTSTC:    SPBR     IBUS,LINENM,SP0    ;READ BSEL3
          Z        DTSTC1             ;WHEN PDP-11 WRITES IT TO 377,
                                      ;THEN ADVANCE
          ALWAYS   DTSTC              ;ELSE, LOOP PDP-11 NOT READY!

DTSTC1:   BRWRTE   IMM,200            ;PICK UP INTO ENABLE BIT
                                      ;WRITE INTERRUPT ENABLE
          OUT      BR,SELB,OBR        ;PUT IT IN INTERRUPT REG
                                      ;UCPU NOW INTERRUPTS
```

TEST D - Interrupt the CPU at Vector XX4

- To enter this test, the CPU must have exited Test C
- The PDP-11 must set up the interrupt vector at XX4
- The 11 code must set BSEL 3 equal to 0
- The microcode upon seeing BSEL 3 equal to 0, generates an interrupt at XX4
- The microcode then exits test

```
DTSTD:    SPBR     IBUS,LINENM,SP0    ;LOOK AT BSEL3
          Z        DTSTD              ;WAIT TILL IT TURNS ZERO (OR NON-377)
```

DTSTD (Cont):

```
        BRWRTE  IMM,300         ;PICK UP I.E. AND INTR 4 BITS
        OUT     BR,SELB,OBR     ;PUT IN INTERRUPT REG
                                ;RUCPU NOW INTERRUPTS
```

TEST E - Non-Processor Request (NPR) IN/OUT Test

- To enter this test, the CPU must first have run Test D
- The PDP-11 code must set the address of the input data area in BSEL 4 and 5 (ADDRXXXXX4)
- The PDP-11 code must set the address of the out data area in BSEL 6 and 7 (ADDRXXXXX6)
- The PDP-11 code must set BSEL 0 to a 0
- The micro-CPU reads the data from one address in the block via NPR, and outputs the data to one location equal to the OUT BLOCK
- The micro-CPU performs the previous step 16 times
- When completed, the micro-CPU will put a 377 into BSEL 0. If an error is detected, the microcode will "hang"
- The PDP-11 puts a 260 into BSEL 0 to exit test, (or) goes back to Step 4 (CPU code must set BSEL 0 to 0)

```
DTSTE:  SPBR    IBUS,INCON,SP0      ;WAIT TILL PDP-11 IS READY
        Z       DTSTE               ;IF = 377, THEN LOOP
        SPBR    IBUS,INCON,SP0      ;IT CHANGED! REREAD, IF=200 THEN
        BR7     DTSTF5              ;EXIT TEST
        OUT     IBUS,PORT1,IBA1     ;READ IN NPR REG, PUT INTO
        OUT     IBUS,PORT2,IBA2     ;ADDR REG
        BRWRTE  IMM,1               ;PICK UP NPR REQUEST BIT
        OUT     BR,SELB,ONPR        ;CAUSE NPR

DTSTE1: BRWRTE  IBUS,NPR            ;READ BACK NPR REG
        BR0     DTSTE1              ;LOOP WHILE NPR

        OUT     IBUS,PORT3,OBA1     ;WRITE OUTPUT ADDRESS
        OUT     IBUS,PORT4,OBA2
        BRWRTE  IBUS,INDAT1         ;READ LOW BYTE OF INPUT DATA
        OUT     BR,SELB,OUTDA1      ;PUT IN LOW BYTE OF DATA
                                    ;TO BE OUTPUTTED
        BRWRTE  IBUS INDAT2         ;READ HIGH BYTE OF INPUT DATA
        OUT     BR,SELB,OUTDA2      ;PUT IN HIGH BYTE OF DATA
                                    ;TO BE OUTPUTTED
        BRWRTE  IMM,21              ;SET OUT NPR
        OUT     BR,SELB,ONPR        ;DO IT
```

```
DTSTE2:  BRWRTE   IBUS,NPR          ;READ BACK NPR REG
         BR0      DTSTE2           ;LOOP WHILE NPR IN PROGRESS
         BRWRTE   IMM,377          ;PICK UP 377
         OUT      BR,SELB,OINCON

         ALWAYS   DTSTE

DTSTF5:  ALWAYS   INIT             ;END TO TESTS.
```

APPENDIX B
INTEGRATED CIRCUIT DESCRIPTIONS

Selected Integrated Circuits (ICs) shown in the *M8207 Print Set* are provided in this Appendix as an aid in troubleshooting to the IC level. The descriptions include one or more of the following: pin/signal designations, equivalent logic block diagrams, logic symbols, and truth/function tables.

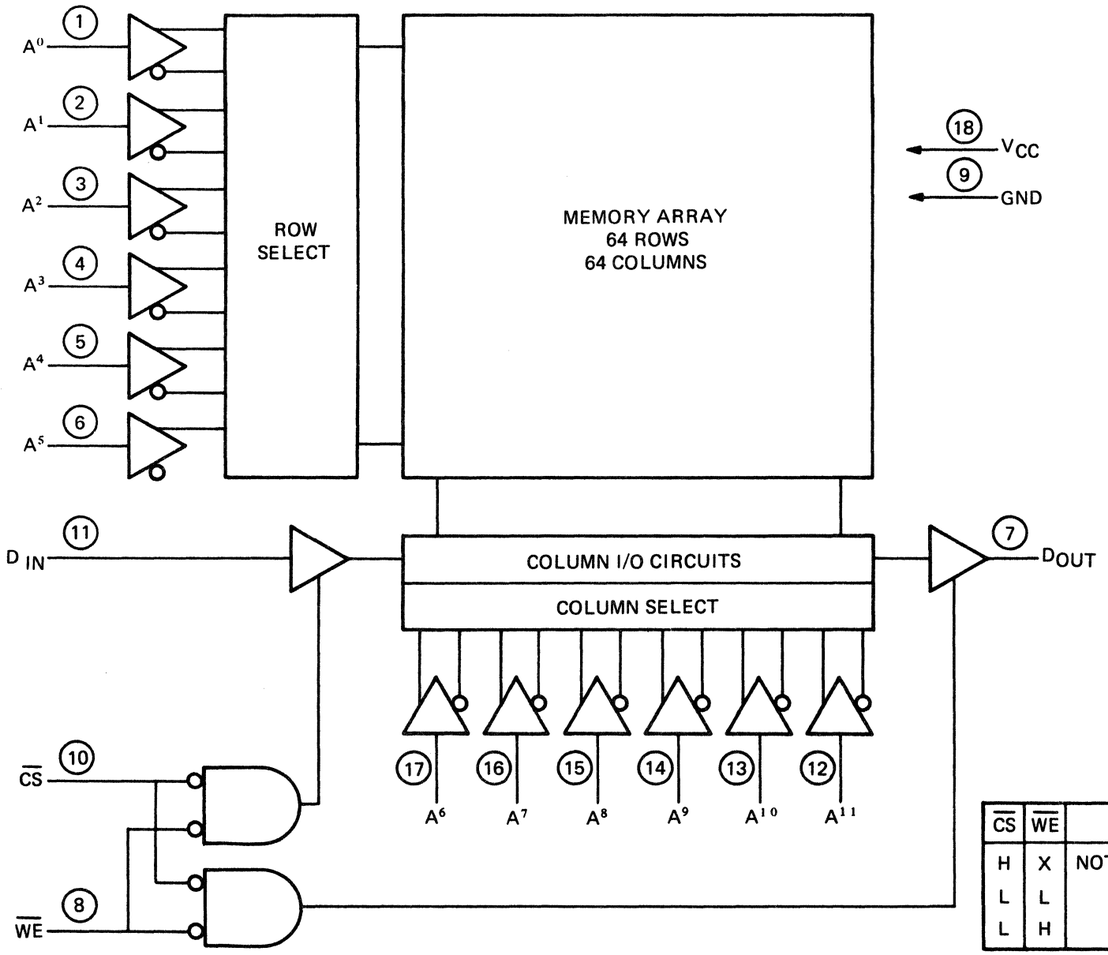
The following ICs are included:

2147 - 4096 × 1-BIT STATIC RAM
74S138 - DECODER MULTIPLEXER
74S151 - DATA SELECTOR/MULTIPLEXER
74LS153 - DUAL 4-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER
74S157 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER
74S158 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER
74LS174 - HEX D-TYPE FLIP-FLOPS WITH CLEAR
74S175 - QUAD D-TYPE FLIP-FLOPS WITH CLEAR
74S181 - 4-BIT ARITHMETIC LOGIC UNIT, ACTIVE HIGH DATA
74S189 - 64-BIT RANDOM ACCESS MEMORY 16 × 4
74LS194 - 4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER
74S240 - OCTAL INVERTER AND LINE DRIVER WITH 3-STATE OUTPUT
74S241 - OCTAL BUFFER AND LINE DRIVER WITH 3-STATE OUTPUT
74S251 - DATA SELECTOR/MULTIPLEXER WITH 3-STATE OUTPUTS
74LS257 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER
74S273 - OCTAL D-TYPE FLIP-FLOP WITH CLEAR
74LS374 - OCTAL D-TYPE TRIGGERED FLIP-FLOPS
8136 - 6-BIT UNIFIED BUS COMPARATOR
82S112 - HIGH SPEED 8 × 4 MULTIPORT RAM
82S191 - 16,384-BIT 2K × 8 BIPOLAR PROM
93S16 - 4-BIT BINARY COUNTER
DC013 - DEC UNIBUS CHIP

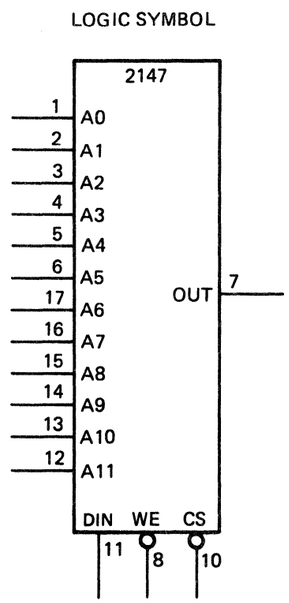
2147 – 4096 × 1-BIT STATIC RAM

The 2147 is a 4096-bit static random access memory organized as 4096 words by 1-bit using HMOS technology. The data is read out nondestructively and has the same polarity as the input data. A data input and a separate three-state output are used.

BLOCK DIAGRAM



LOGIC SYMBOL



TRUTH TABLE

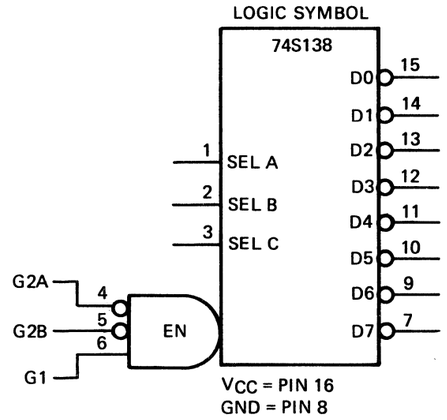
$\overline{CS}$	$\overline{WE}$	MODE	OUTPUT	POWER
H	X	NOT SELECTED	HIGH Z	STANDBY
L	L	WRITE	HIGH Z	ACTIVE
L	H	READ	D_{OUT}	ACTIVE

MK-0688

74S138 – DECODER MULTIPLEXER

The 74S138 decodes one of eight lines depending on the conditions at the three binary select inputs and the three enable inputs. Two active-low and one active-high enable inputs reduce the need for external gates or inverters when expanding.

LOGIC SYMBOL



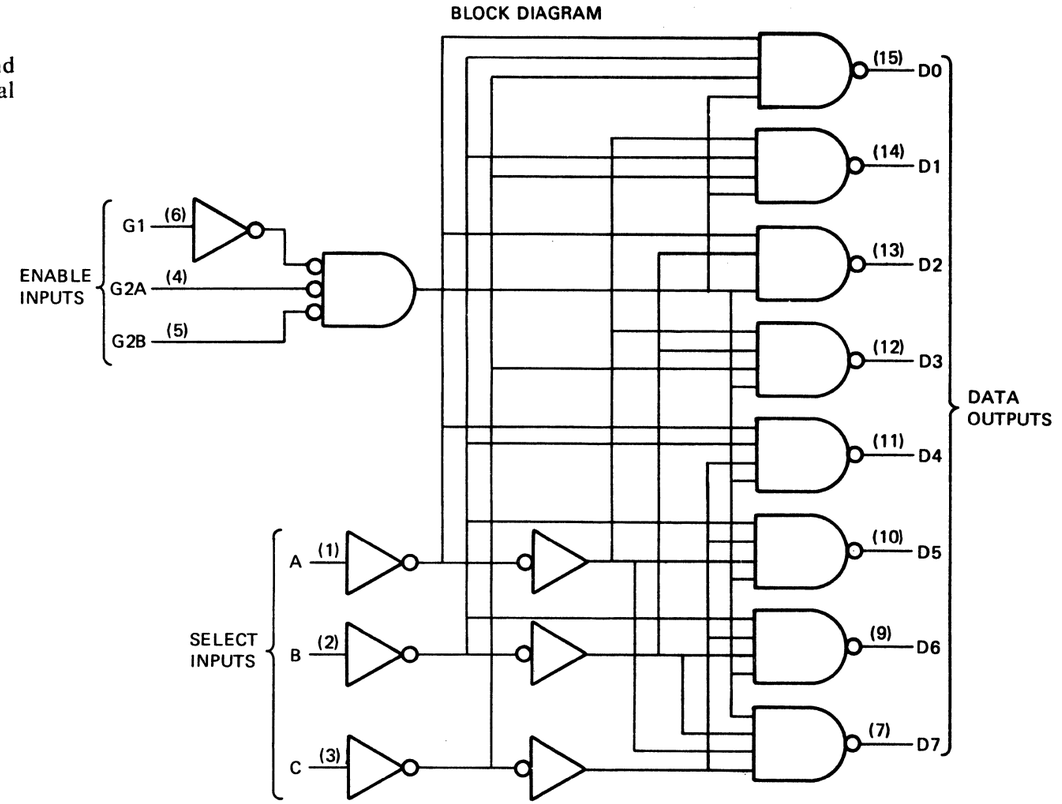
V_{CC} = PIN 16
GND = PIN 8

FUNCTION TABLE

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2*	C	B	A	D0	D1	D2	D3	D4	D5	D6	D7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

*G2 = G2A + G2B
H = HIGH LEVEL, L = LOW LEVEL, X = IRRELEVANT

BLOCK DIAGRAM



IC-74138B

74S151 - DATA SELECTOR/MULTIPLEXER

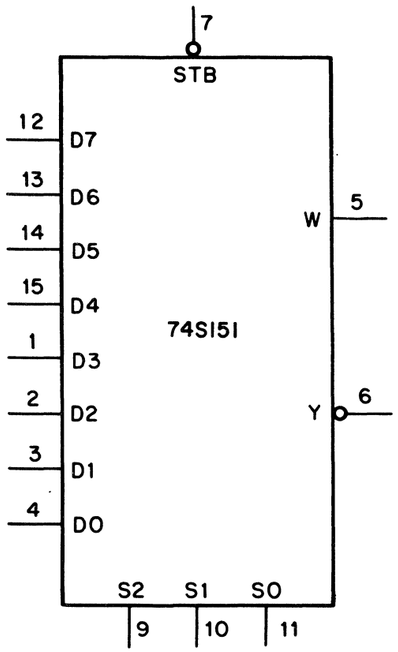
The 74S151 monolithic data selector/multiplexer contains full on-chip binary decoding to select the desired data source. The 74S151 selects one of eight data sources and has a strobe input which must be at a low logic level to enable the device. A high level at the strobe forces the f0 output high, and the f1 output (as applicable) low.

74S151 TRUTH TABLE

Inputs											Outputs	
S2	S1	S0	STB	D0	D1	D2	D3	D4	D5	D6	D7	W	Y
X	X	X	1	X	X	X	X	X	X	X	X	0	1
0	0	0	0	0	X	X	X	X	X	X	X	0	1
0	0	0	0	1	X	X	X	X	X	X	X	1	0
0	0	1	0	X	0	X	X	X	X	X	X	0	1
0	0	1	0	X	1	X	X	X	X	X	X	1	0
0	1	0	0	X	X	0	X	X	X	X	X	0	1
0	1	0	0	X	X	1	X	X	X	X	X	1	0
0	1	1	0	X	X	X	0	X	X	X	X	0	1
0	1	1	0	X	X	X	1	X	X	X	X	1	0
1	0	0	0	X	X	X	X	0	X	X	X	0	1
1	0	0	0	X	X	X	X	1	X	X	X	1	0
1	0	1	0	X	X	X	X	X	0	X	X	0	1
1	0	1	0	X	X	X	X	X	1	X	X	1	0
1	1	0	0	X	X	X	X	X	X	0	X	0	1
1	1	0	0	X	X	X	X	X	X	1	X	1	0
1	1	1	0	X	X	X	X	X	X	X	0	0	1
1	1	1	0	X	X	X	X	X	X	X	1	1	0

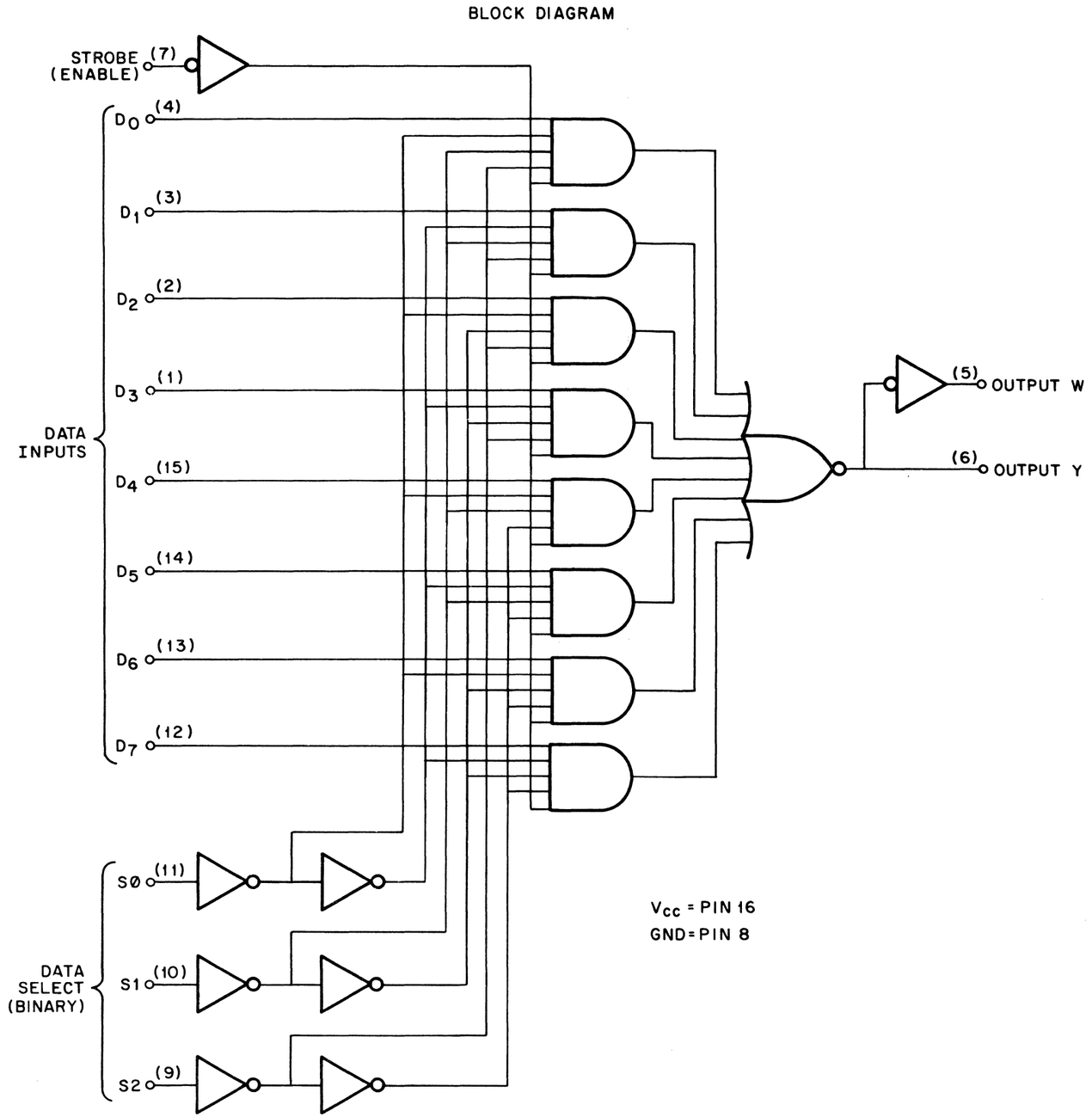
When used to indicate an input, X = irrelevant.

LOGIC SYMBOL



IC-74151B

BLOCK DIAGRAM



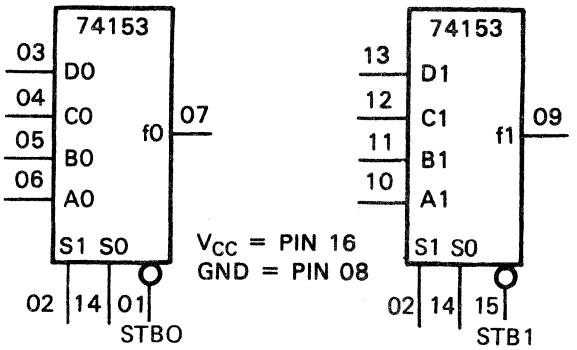
V_{CC} = PIN 16
GND = PIN 8

IC-74151A

74LS153 - DUAL 4-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

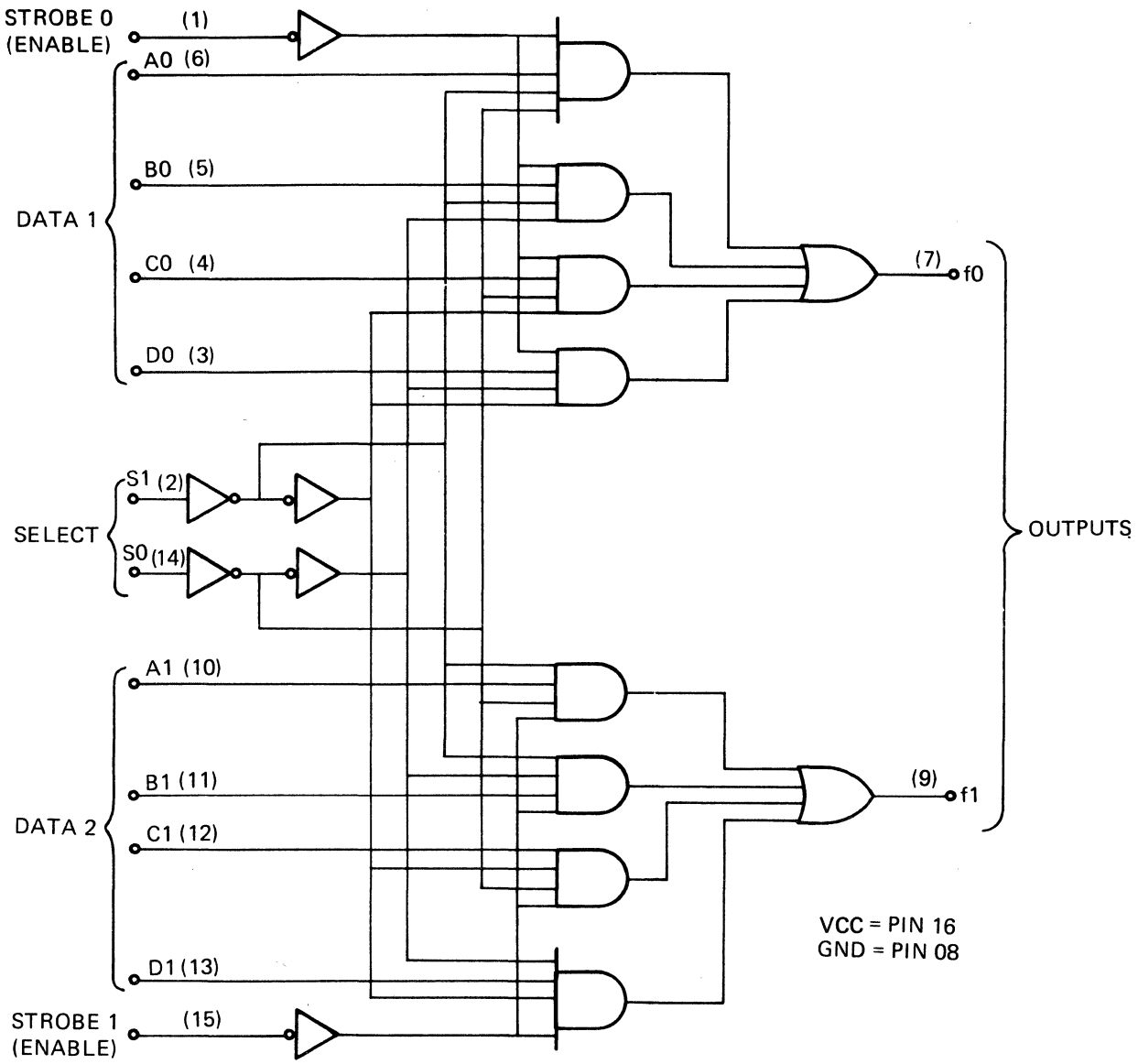
The 74LS153 monolithic data selector/multiplexer contains inverters and drivers to supply fully complementary, on-chip, binary decoding data selection to the AND-OR invert gates. Separate strobe inputs are provided for each of the two four-line sections.

LOGIC SYMBOL



ADDRESS INPUTS		DATA INPUTS				STROBE OUTPUT	
S1	S0	A	B	C	D	STB	f
X	X	X	X	X	X	H	L
L	L	L	X	X	X	L	L
L	L	H	X	X	X	L	H
L	H	X	L	X	X	L	L
L	H	X	H	X	X	L	H
H	L	X	X	L	X	L	L
H	L	X	X	H	X	L	H
H	H	X	X	X	L	L	L
H	H	X	X	X	H	L	H

ADDRESS INPUTS S0 AND S1 ARE COMMON TO BOTH SECTIONS: H=HIGH LEVEL, L=LOW LEVEL, X=IRRELEVANT

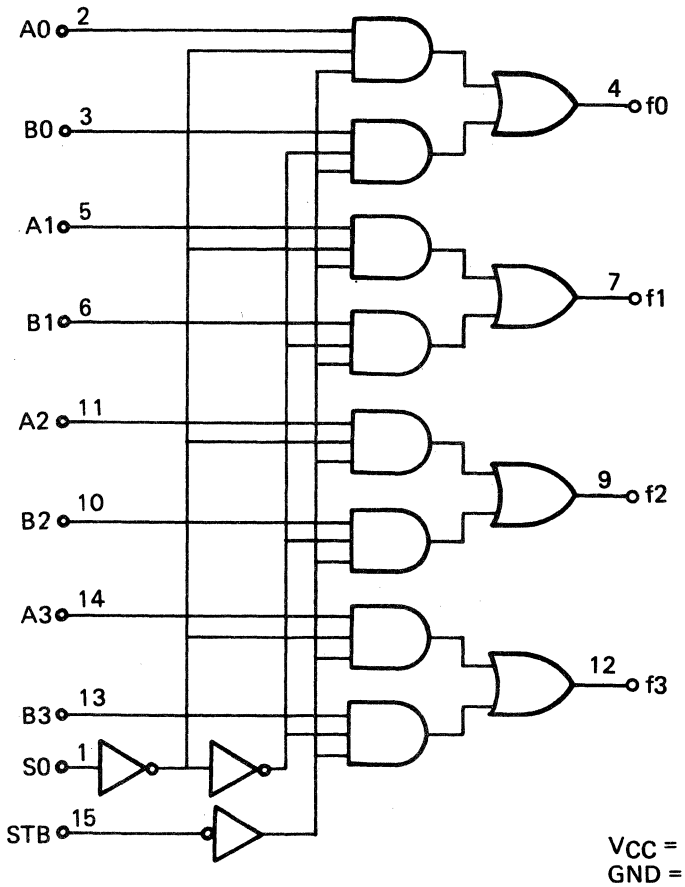

STROBE 0
(ENABLE)
(1)
A0 (6)
B0 (5)
DATA 1
C0 (4)
D0 (3)
(7)
f0
S1 (2)
SELECT
S0 (14)
OUTPUTS
A1 (10)
B1 (11)
DATA 2
C1 (12)
(9)
f1
VCC = PIN 16
GND = PIN 08
D1 (13)
STROBE 1
(ENABLE)
(15)

IC-74153

74S157 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

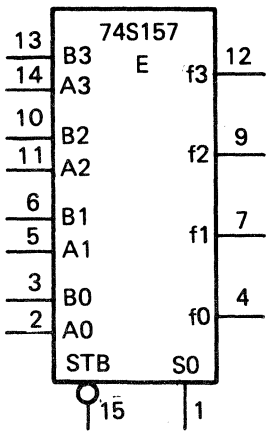
The 74S157 monolithic data selector/multiplexer contains inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input is provided. A 4-bit word is selected from one of two sources, and routed to the four outputs (true data).

BLOCK DIAGRAM



V_{CC} = PIN 16
GND = PIN 8

LOGIC SYMBOL



TRUTH TABLE

INPUTS			OUTPUT
STROBE	SELECT	A B	
H	X	X X	L
L	L	L X	L
L	L	H X	H
L	H	X L	L
L	H	X H	H

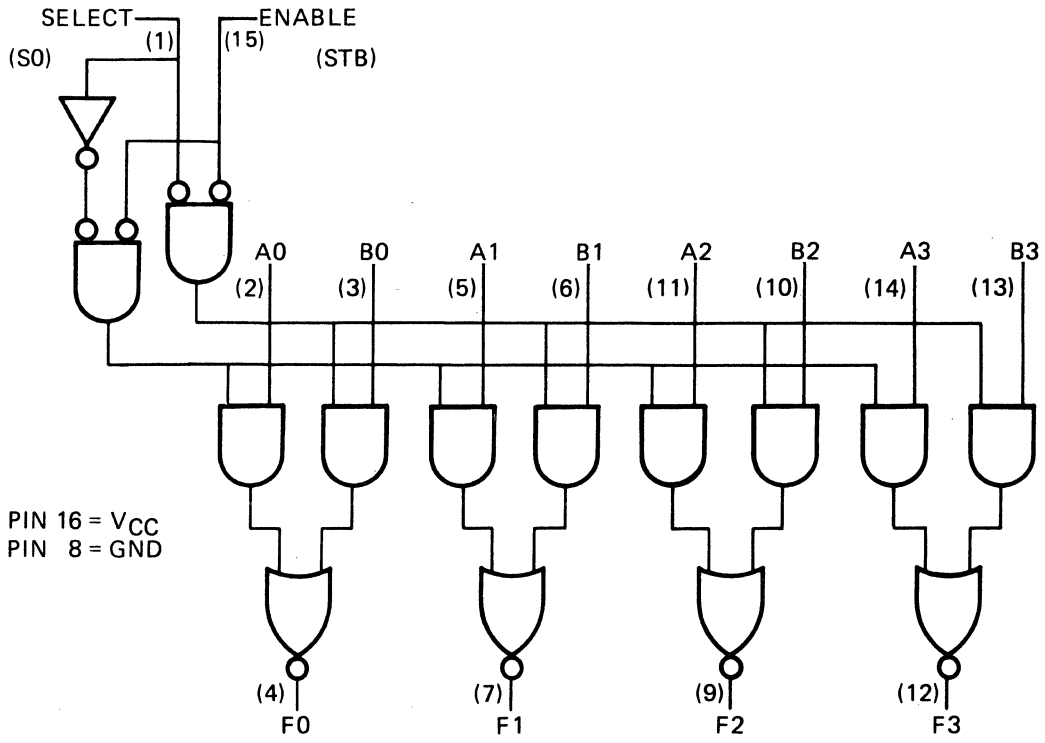
H = HIGH LEVEL, L = LOW LEVEL
X = IRRELEVANT

IC-74157

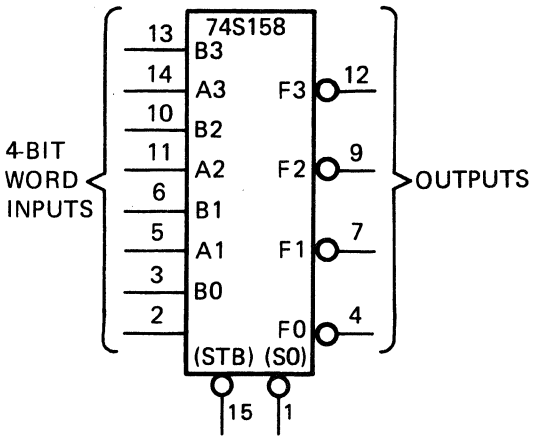
74S158 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

The 74S158 monolithic data selector/multiplexer contains inverters and drivers to supply full on-chip data selection to the four output gates. A separate strobe input is provided. A 4-bit word is selected from one of two sources, and routed to the four outputs (inverted data).

BLOCK DIAGRAM



LOGIC SYMBOL



FUNCTION TABLE

INPUTS				OUTPUT Y
STROBE	SELECT	A	B	f
H	X	X	X	H
L	L	L	X	H
L	L	H	X	L
L	H	X	L	H
L	H	X	H	L

H = HIGH LOGIC LEVEL
L = LOW LOGIC LEVEL
X = IRRELEVANT

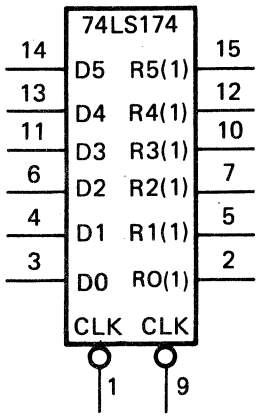
MK-0658

74LS174 - HEX D-TYPE FLIP-FLOPS WITH CLEAR

These monolithic, positive-edge-triggered flip-flops utilize TTL circuitry to implement D-type flip-flop logic. All have a direct clear input.

Information at the D inputs meeting the setup time requirements is transferred to the R outputs on the positive-going edge of the clock pulse. Clock triggering occurs at a particular voltage level and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the D input signal has no effect at the output.

LOGIC SYMBOL

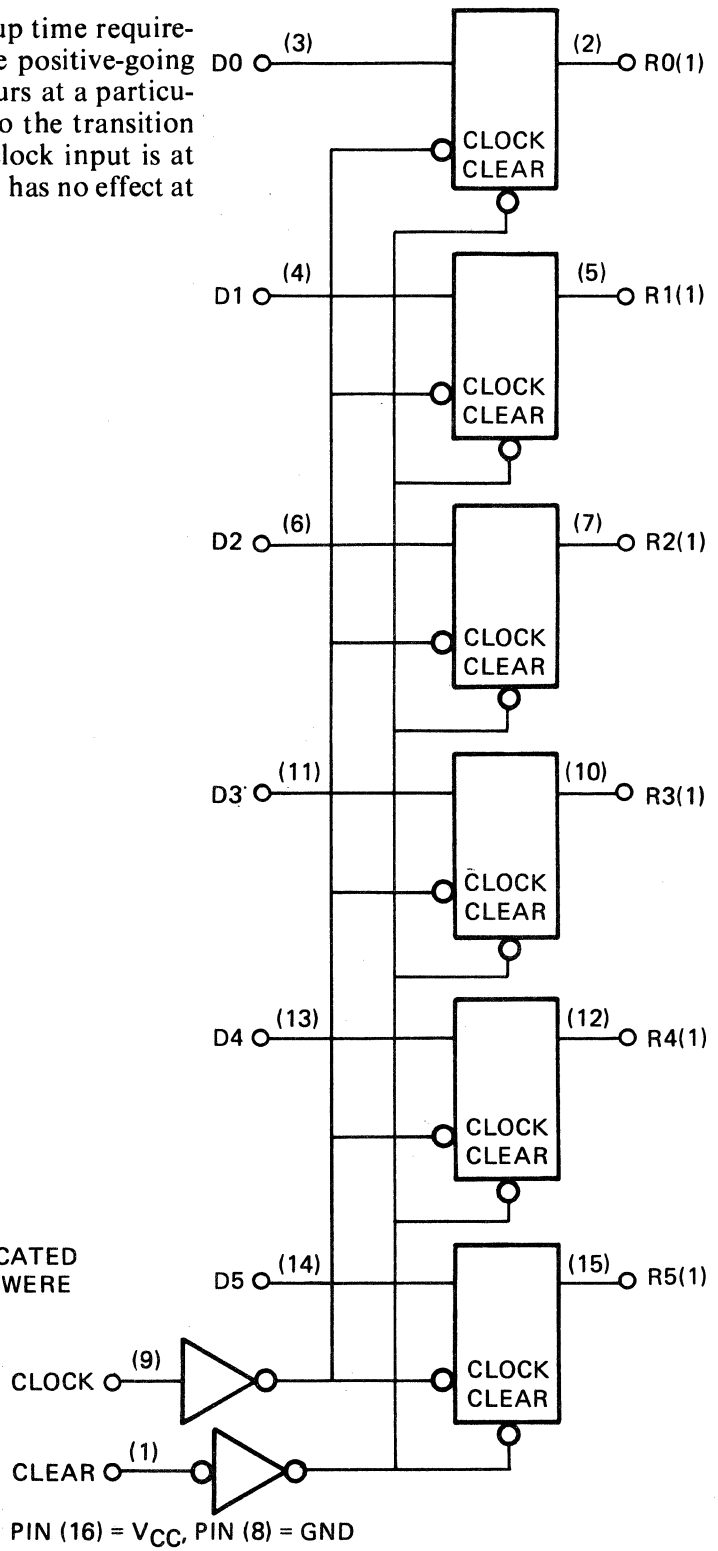


FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUTS
CLEAR	CLOCK	D	R
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

H–HIGH LEVEL (STEADY STATE)
L–LOW LEVEL (STEADY STATE)
X–IRRELEVANT
↑–TRANSITION FROM LOW TO HIGH
Q_0–THE LEVEL OF R BEFORE THE INDICATED STEADY STATE INPUT CONDITIONS WERE ESTABLISHED

BLOCK DIAGRAM



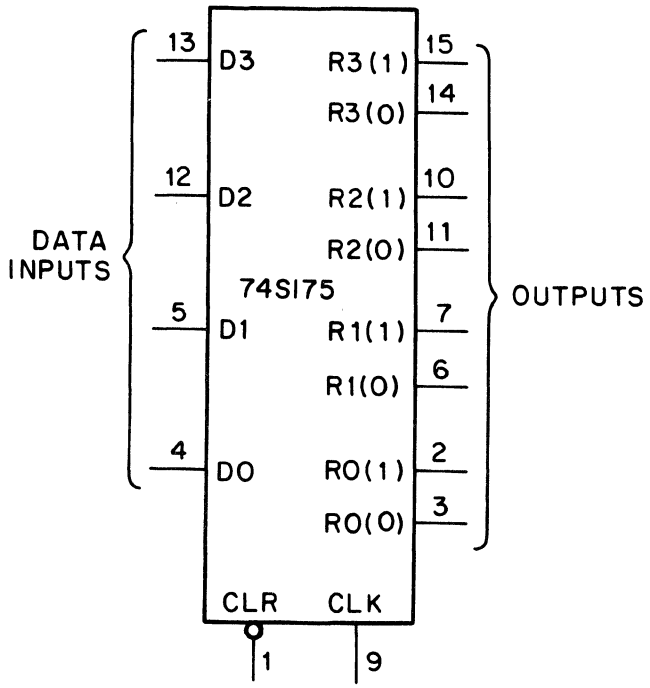
PIN (16) = V_{CC}, PIN (8) = GND

MK-0660

74S175 - QUAD D-TYPE FLIP-FLOPS WITH CLEAR

These monolithic, positive-edge-triggered flip-flops utilize TTL circuitry to implement D-type flip-flop logic. All have a direct clear input and feature complementary outputs from each flip-flop. Information at the D inputs meeting the setup time requirements is transferred to the R outputs on the positive-going of the clock pulse. Clock triggering occurs at a particular voltage level and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the D input signal has no effect at the output.

LOGIC SYMBOL

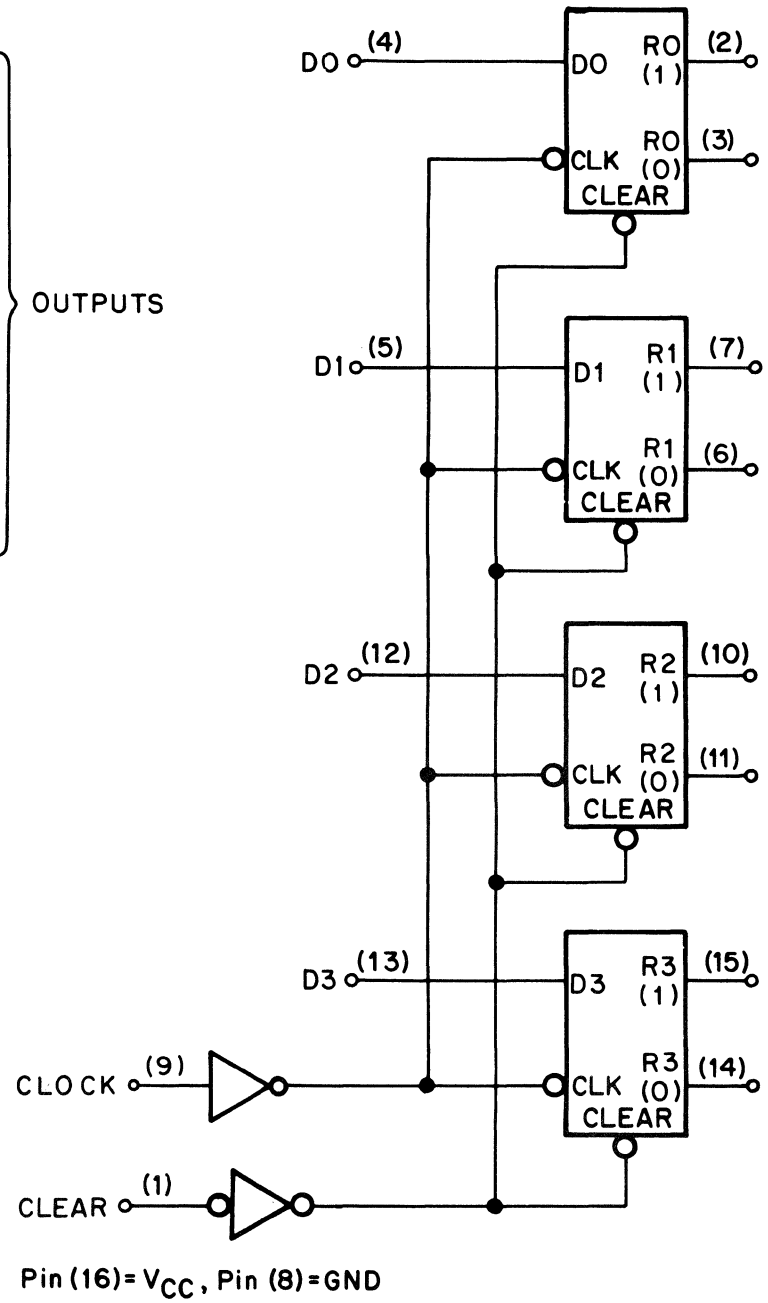


FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUTS	
CLEAR	CLOCK	D	R	$\bar{R}$
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\bar{Q}_0$

H = HIGH LEVEL (STEADY STATE)
L = LOW LEVEL (STEADY STATE)
X = IRRELEVANT
↑ = TRANSITION FROM LOW TO HIGH LEVEL
Q_0 = THE LEVEL OF R BEFORE THE INDICATED STEADY-STATE INPUT CONDITIONS WERE ESTABLISHED.

BLOCK DIAGRAM

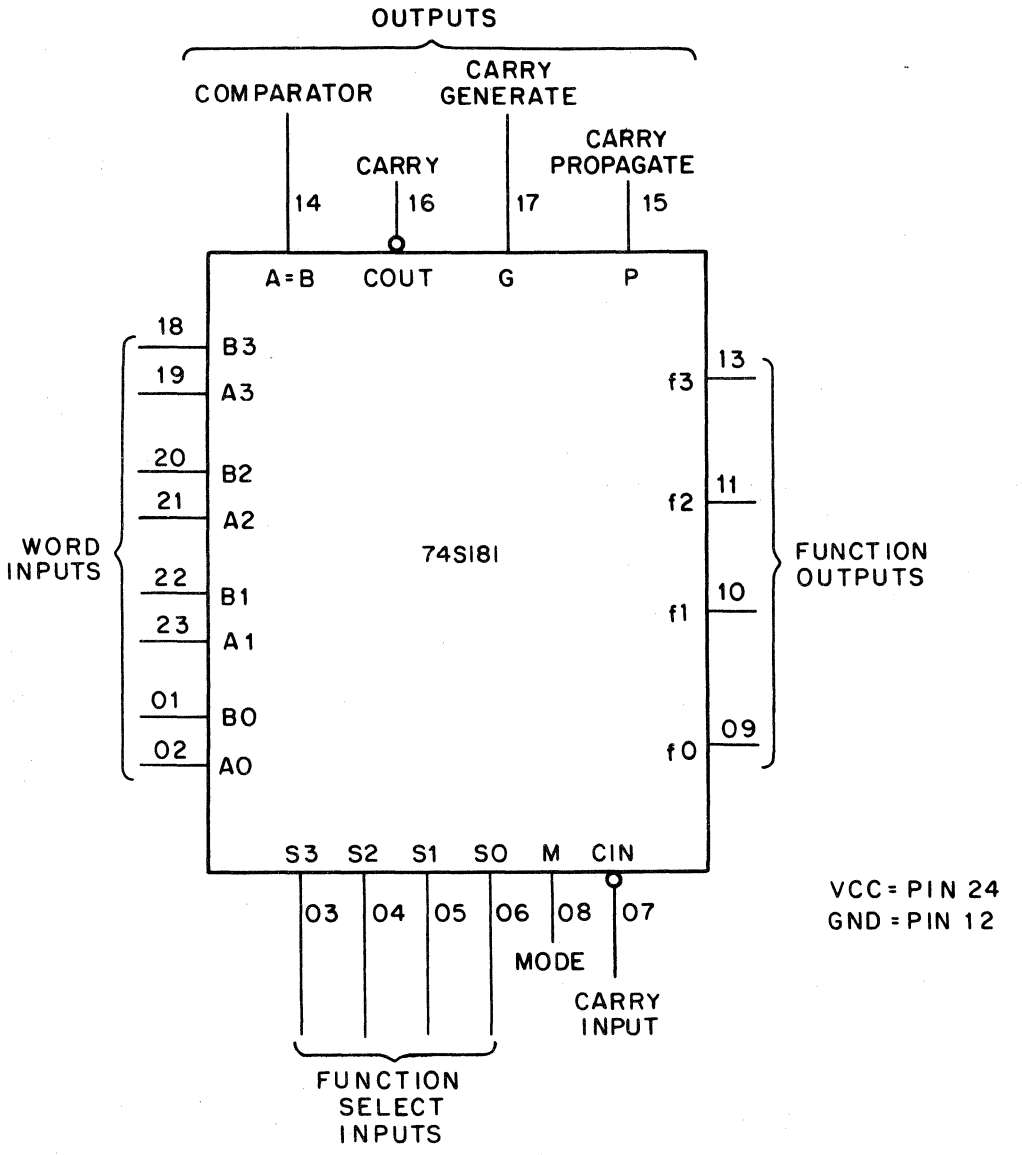


Pin (16) = V_{CC}, Pin (8) = GND

IC-74175B

74S181 – 4-BIT ARITHMETIC LOGIC UNIT, ACTIVE HIGH DATA

The 74S181 performs up to 16 arithmetic and 16 logic functions. Arithmetic operations are selected by four function-select lines (S0, S1, S2, and S3) with a low-level voltage at the mode control input (M), and a low-level carry input. Logical operations are selected by the same four function-select lines except that the mode control input (M) must be high to disable the carry input.



74181
TABLE OF LOGIC FUNCTIONS

Function Select				Output Function	
S3	S2	S1	S0	Negative Logic	Positive Logic
L	L	L	L	$f = \overline{A}$	$f = \overline{A}$
L	L	L	H	$f = \overline{A}\overline{B}$	$f = \overline{A + B}$
L	L	H	L	$f = \overline{A} + B$	$f = \overline{A}B$
L	L	H	H	f = Logical 1	f = Logical 0
L	H	L	L	$f = \overline{A + B}$	$f = \overline{AB}$
L	H	L	H	$f = \overline{B}$	$f = \overline{B}$
L	H	H	L	$f = \overline{A \oplus B}$	$f = A \oplus B$
L	H	H	H	$f = A + \overline{B}$	$f = A\overline{B}$
H	L	L	L	$f = \overline{A}B$	$f = \overline{A + B}$
H	L	L	H	$f = A \oplus B$	$f = \overline{A \oplus B}$
H	L	H	L	f = B	f = B
H	L	H	H	f = A + B	f = AB
H	H	L	L	f = Logical 0	f = Logical 1
H	H	L	H	$f = A\overline{B}$	$f = A + \overline{B}$
H	H	H	L	f = AB	f = A + B
H	H	H	H	f = A	f = A

With mode control (M) high: C_{in} irrelevant
For positive logic: logical 1 = high voltage
logical 0 = low voltage
For negative logic: logical 1 = low voltage
logical 0 = high voltage

74181
TABLE OF ARITHMETIC OPERATIONS

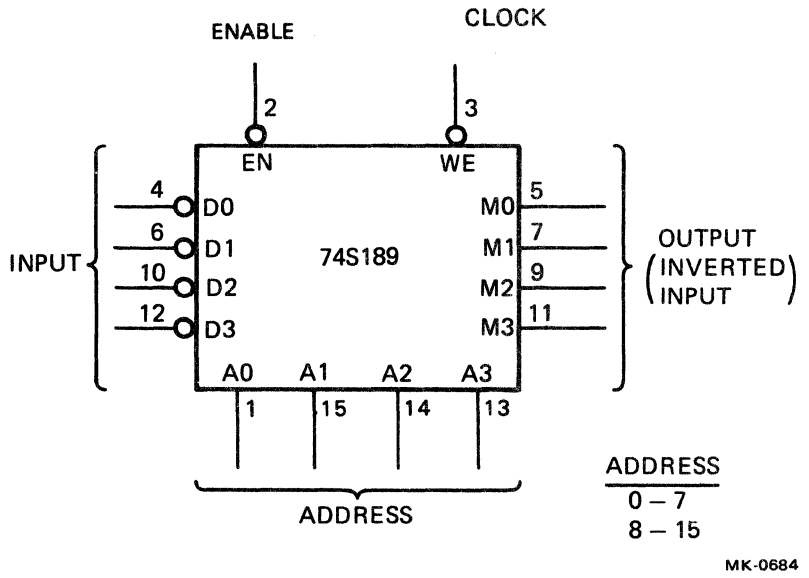
Function Select				Output Function	
S3	S2	S1	S0	Low Levels Active	High Levels Active
L	L	L	L	f = A minus 1	f = A
L	L	L	H	f = AB minus 1	f = A + B
L	L	H	L	$f = A\overline{B}$ minus 1	$f = A + \overline{B}$
L	L	H	H	f = minus 1 (2's complement)	f = minus 1 (2's complement)
L	H	L	L	$f = A$ plus $[A + \overline{B}]$	$f = A$ plus $A\overline{B}$
L	H	L	H	f = AB plus $[A + \overline{B}]$	$f = [A + B]$ plus $A\overline{B}$
L	H	H	L	f = A minus B minus 1	f = A minus B minus 1
L	H	H	H	$f = A + \overline{B}$	$f = A\overline{B}$ minus 1
H	L	L	L	f = A plus [A + B]	f = A plus AB
H	L	L	H	f = A plus B	f = A plus B
H	L	H	L	$f = A\overline{B}$ plus [A + B]	$f = [A + \overline{B}]$ plus AB
H	L	H	H	f = A + B	f = AB minus 1
H	H	L	L	f = A plus A†	f = A plus A†
H	H	L	H	f = AB plus A	f = [A + B] plus A
H	H	H	L	$f = A\overline{B}$ plus A	$f = [A + \overline{B}]$ plus A
H	H	H	H	f = A	f = A minus 1

With mode control (M) and C_{in} low
† Each bit is shifted to the next more significant position.

IC-74181B

74S189 - 64-BIT RANDOM ACCESS MEMORY 16 × 4

The 74S189 is a 64-bit random access memory organized as 16-words by 4-bits providing inverted three-state outputs.

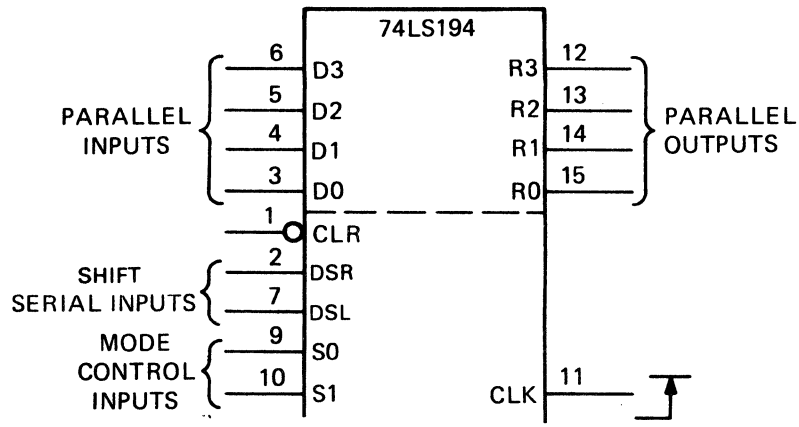


74LS194 - 4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER

The 74LS194 bidirectional shift register features parallel inputs, parallel outputs, right-shift and left-shift serial inputs, operating mode control inputs, and a direct overriding clear line. The register has four distinct modes of operation:

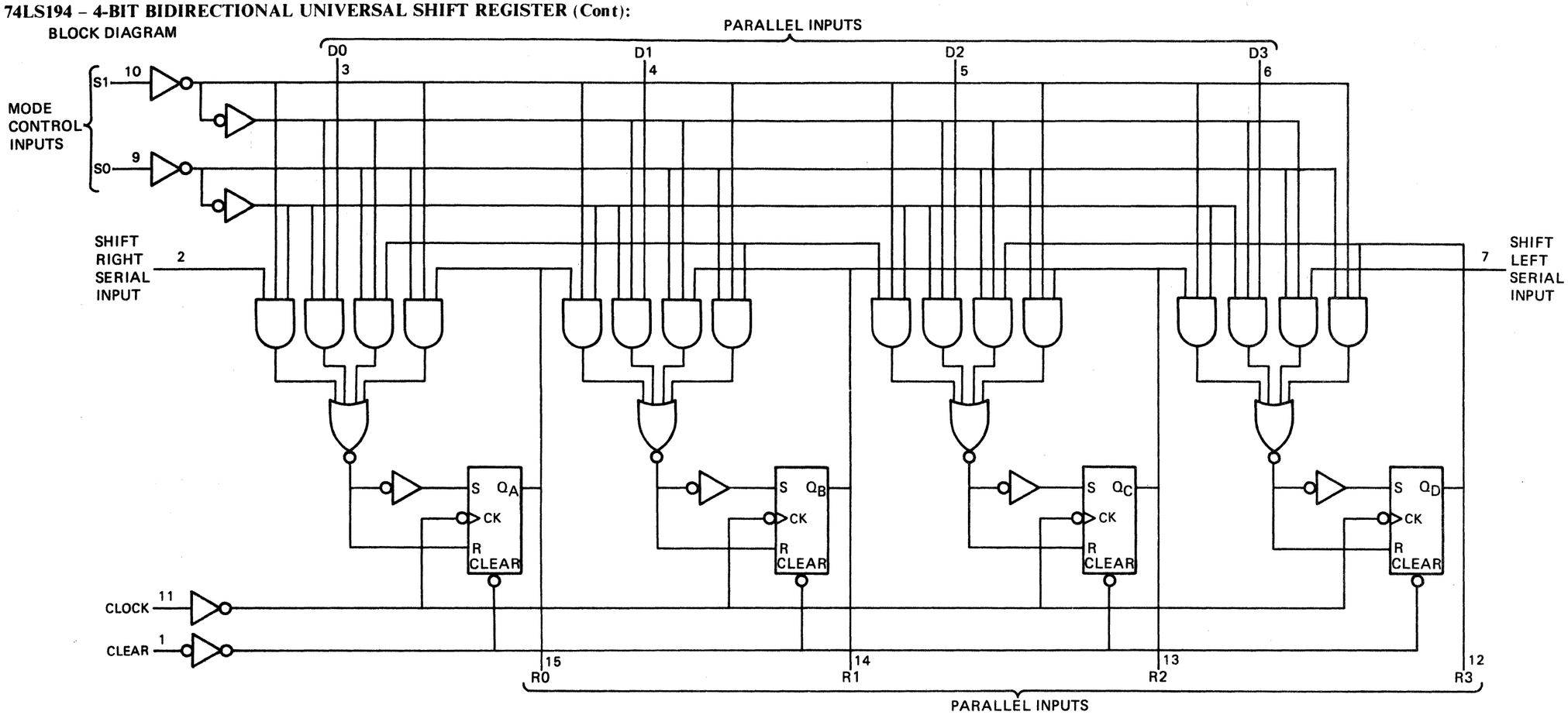
- Parallel (broadside) load
- Shift right (in the direction R0 toward R3)
- Shift left (in the direction R3 toward R0)
- Inhibit clock (do nothing)

LOGIC SYMBOL



74LS194 – 4-BIT BIDIRECTIONAL UNIVERSAL SHIFT REGISTER (Cont):

BLOCK DIAGRAM



FUNCTION TABLE

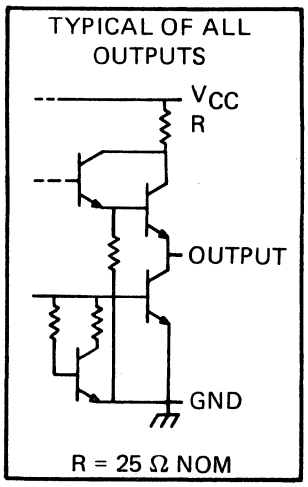
CLEAR	MODE		CLOCK	SERIAL		PARALLEL				OUTPUTS			
				LEFT	RIGHT	D0	D1	D2	D3	R0	R1	R2	R3
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	H	↑	X	X	a	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

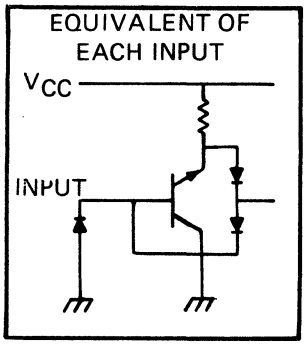
H = HIGH LEVEL (STEADY STATE)
L = LOW LEVEL (STEADY STATE)
X = IRRELEVANT (ANY INPUT, INCLUDING TRANSITIONS)
↑ = TRANSITION FROM LOW TO HIGH LEVEL
a, b, c, d = THE LEVEL OF STEADY STATE INPUT AT INPUTS D0, 1, 2, 3 RESPECTIVELY
Q_{A0} Q_{B0} Q_{C0} Q_{D0} – LEVEL OF R0, R1, R2, R3 RESPECTIVELY BEFORE THE INDICATED STEADY STATE INPUT CONDITIONS WERE ESTABLISHED
Q_{An}, Q_{Bn}, Q_{Cn}, Q_{Dn}, = THE LEVEL OF R0, R1, R2, R3 RESPECTIVELY BEFORE THE MOST RECENT TRANSITION OF THE CLOCK

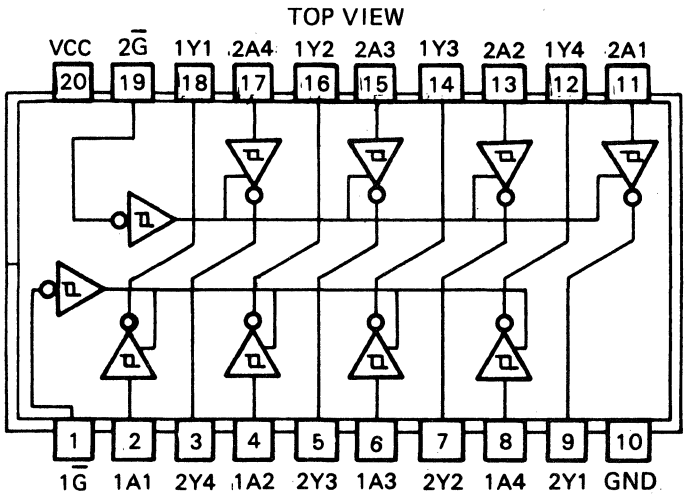
MK-0676

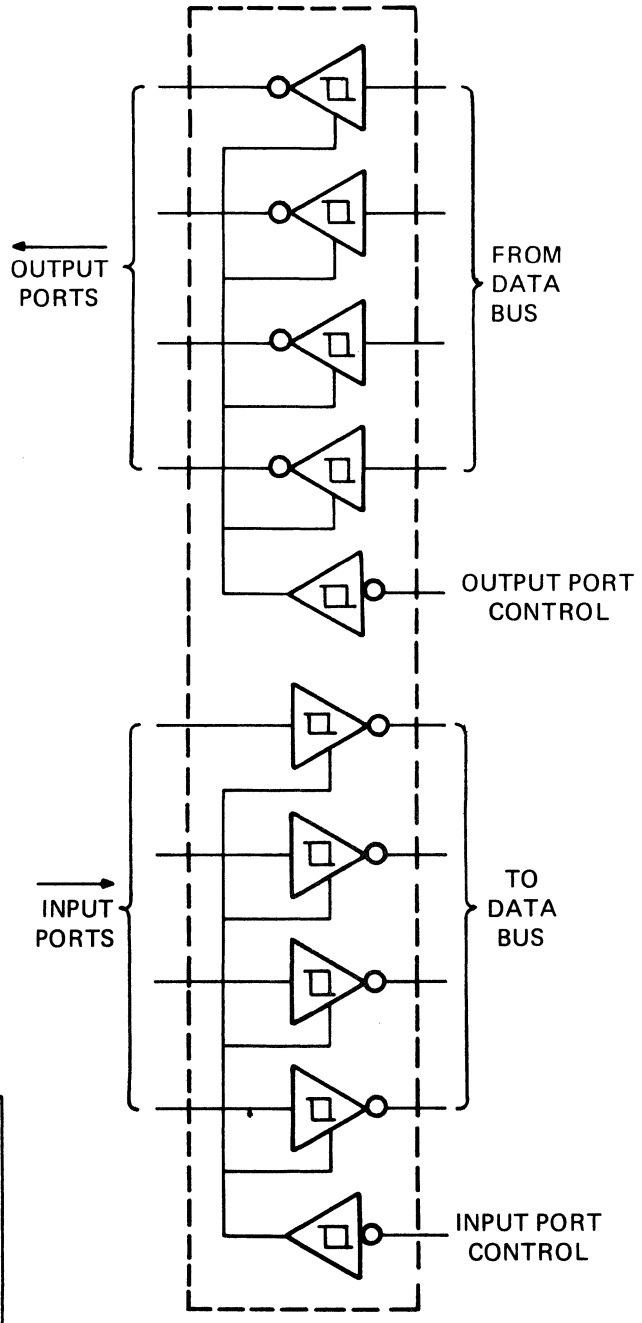
74S240 - OCTAL INVERTER AND LINE DRIVER WITH 3-STATE OUTPUT

These octal inverters and line drivers provide inverting outputs, and $\overline{G}$ (active-low output control) inputs. $\overline{G}$ selects four inverting buffers.







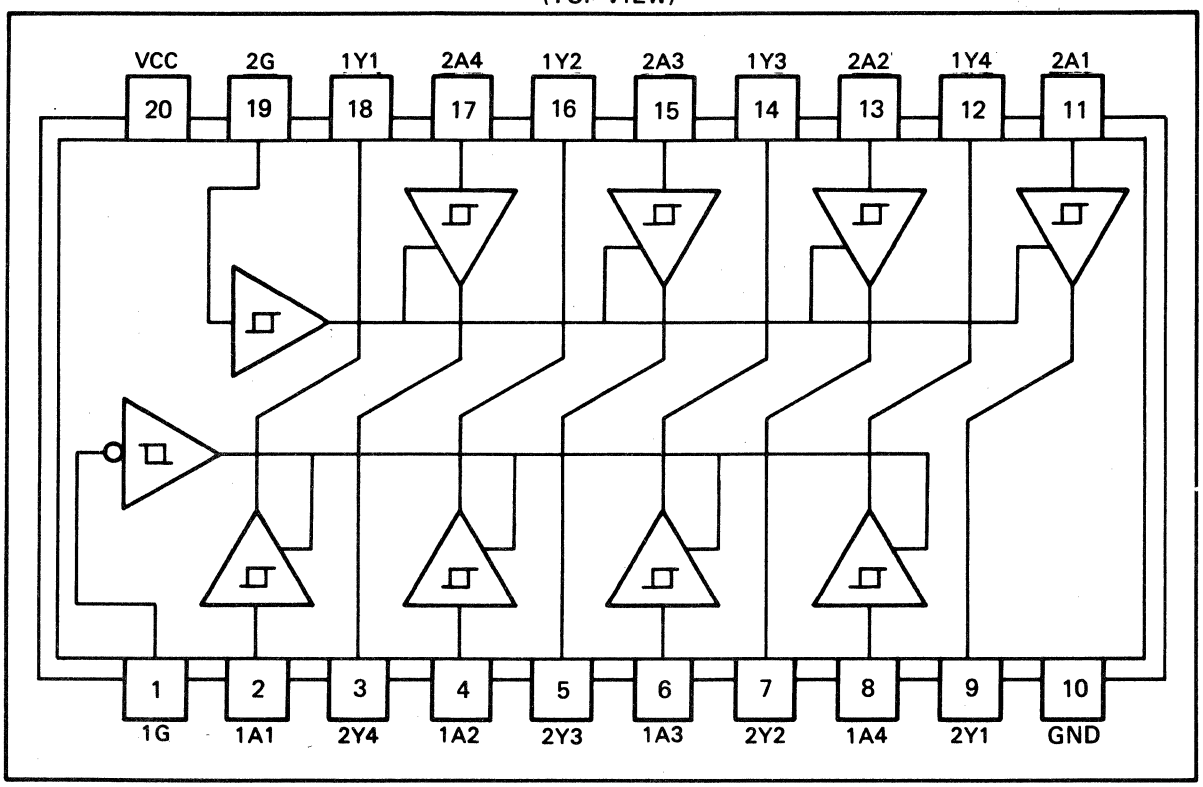


MK0674

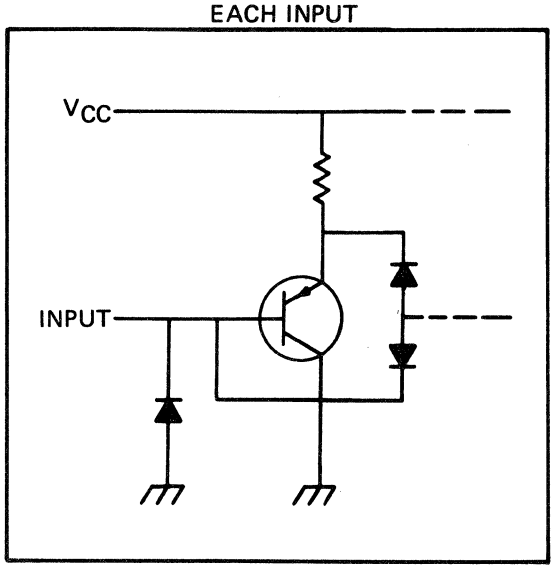
74S241 - OCTAL BUFFER AND LINE DRIVER WITH 3-STATE OUTPUT

These octal buffers and line drivers provide noninverting outputs, and complementary $\overline{G}$ and G selects for each of four buffers.

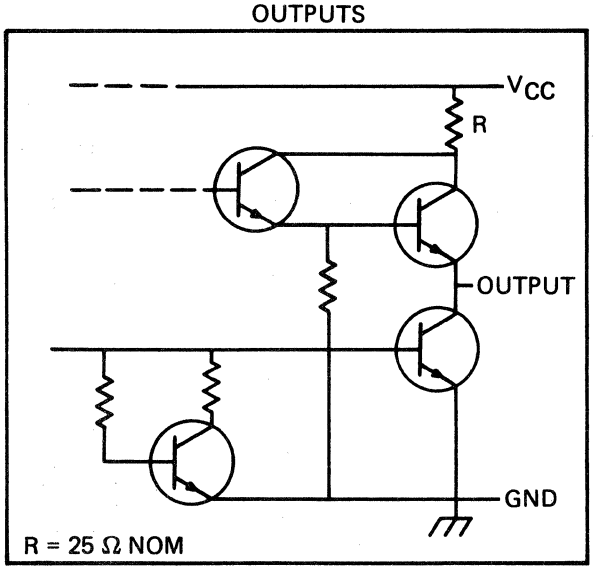
(TOP VIEW)



EQUIVALENT OF EACH INPUT



TYPICAL OF ALL OUTPUTS

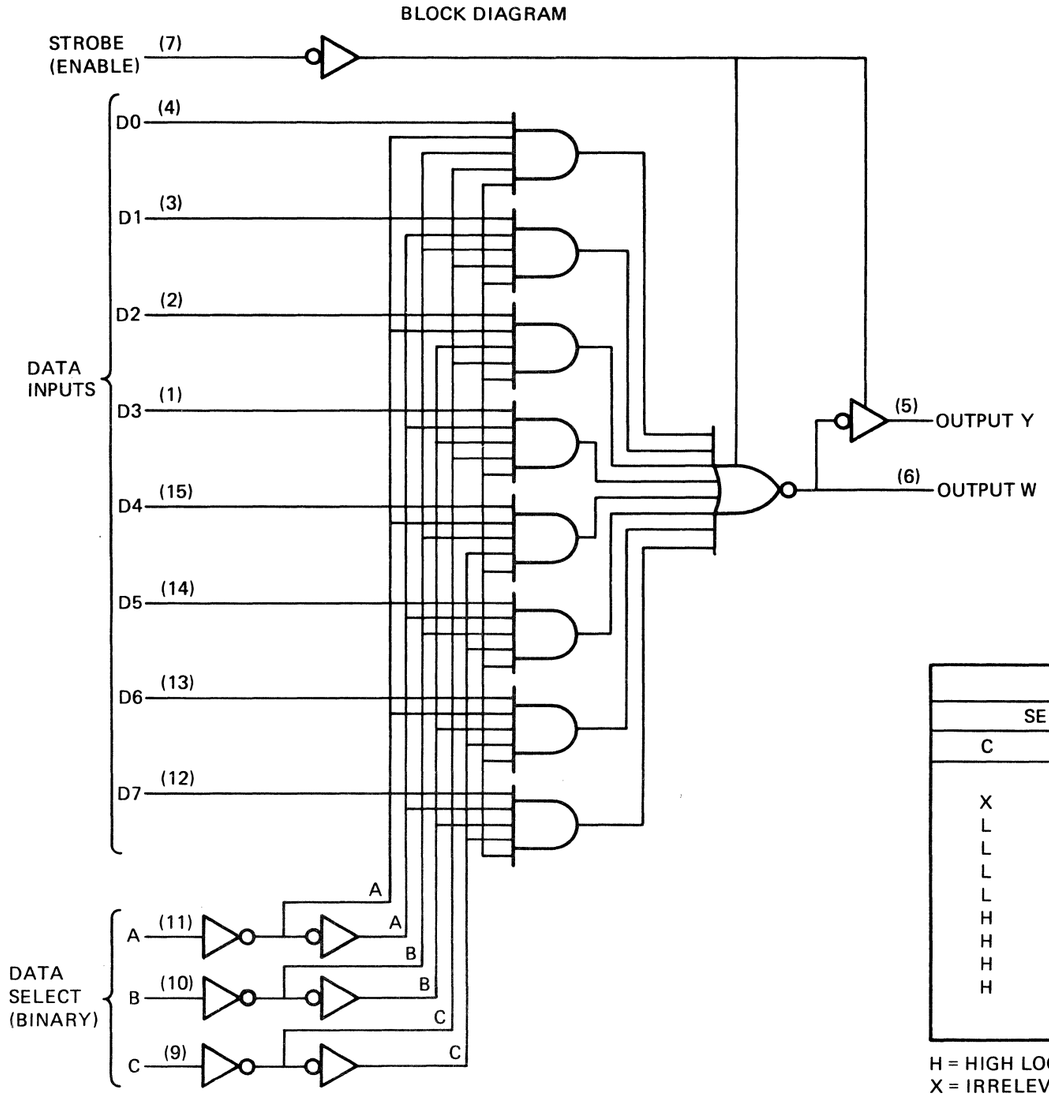


MK-0686

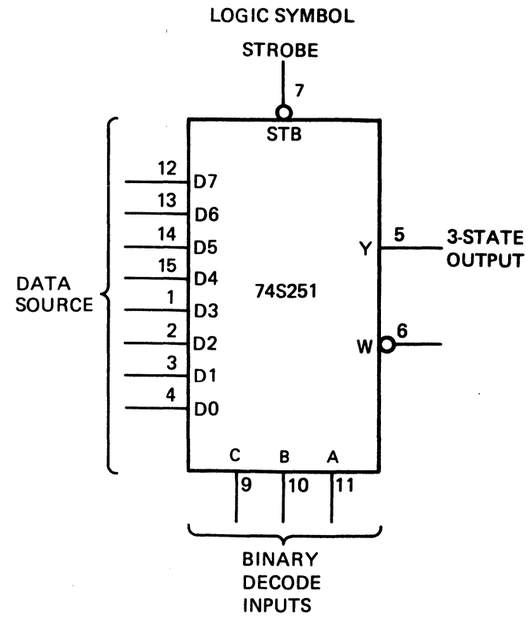
74S251 – DATA SELECTOR/MULTIPLEXER WITH 3-STATE OUTPUTS

The 74S251 monolithic data selector/multiplexer contains full on-chip binary decoding to select one of eight data sources, and features a strobe-controlled three-state output. The strobe must be at a low logic level to enable the device. The three-state outputs permit a number of outputs to be connected to a common bus. When the strobe input is high, both outputs are in a high-impedence state in which both the upper and lower transistors of each totem pole output are off, and the output neither drives nor loads the bus significantly. When the strobe is low, the outputs are activated and operate as standard TTL totem pole outputs.

BLOCK DIAGRAM



LOGIC SYMBOL



FUNCTION TABLE

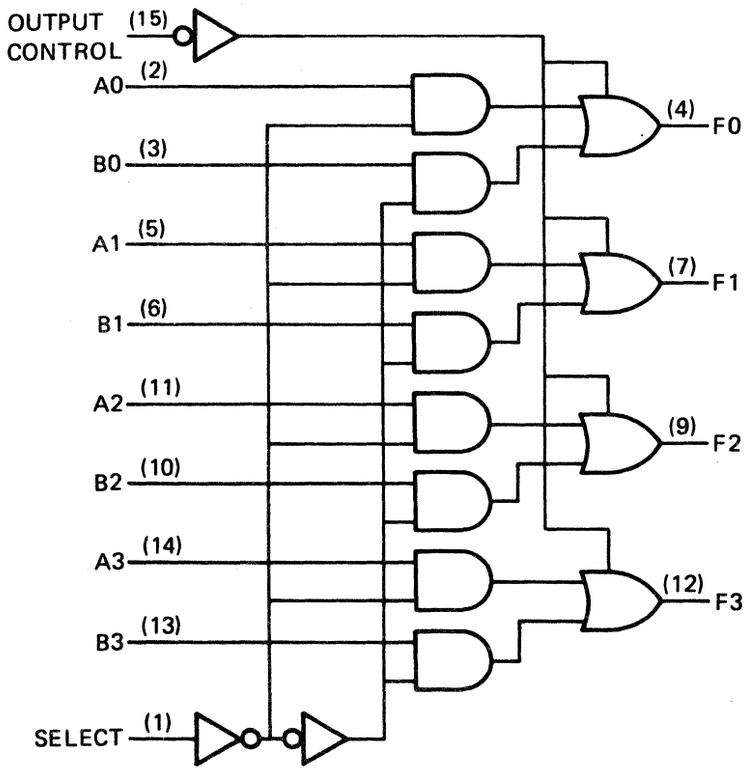
INPUTS				OUTPUTS	
SELECT			STROBE		
C	B	A	S	Y	W
X	X	X	H	Z	Z
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

H = HIGH LOGIC LEVEL, L = LOW LOGIC LEVEL
X = IRRELEVANT, Z = HIGH EMPEDANCE (OFF)
D0, D1, THRU D7 = THE LEVEL OF THE RESPECTIVE D INPUT MK-0678

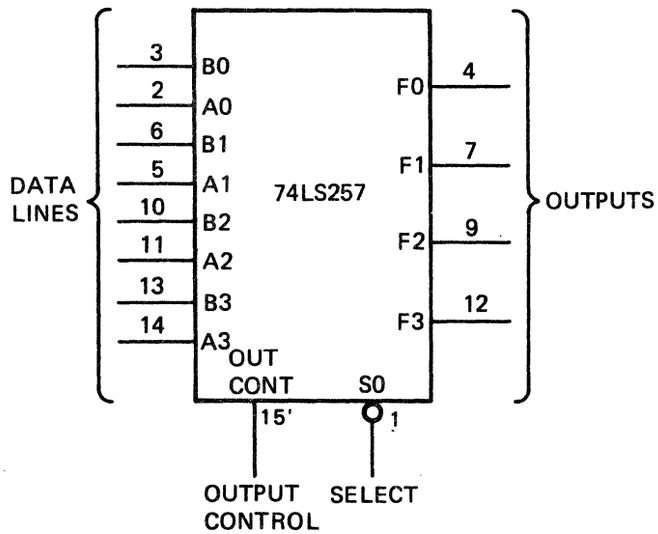
74LS257 - QUAD 2-LINE-TO-1-LINE DATA SELECTOR/MULTIPLEXER

The 74LS257 multiplexer features three-state outputs that can interface directly with and drive data lines of bus-organized systems. With all but one of the common outputs disabled (at a high-impedance state) the low impedance of the single enabled output will drive the bus line to a high or low logic level.

BLOCK DIAGRAM



LOGIC SYMBOL



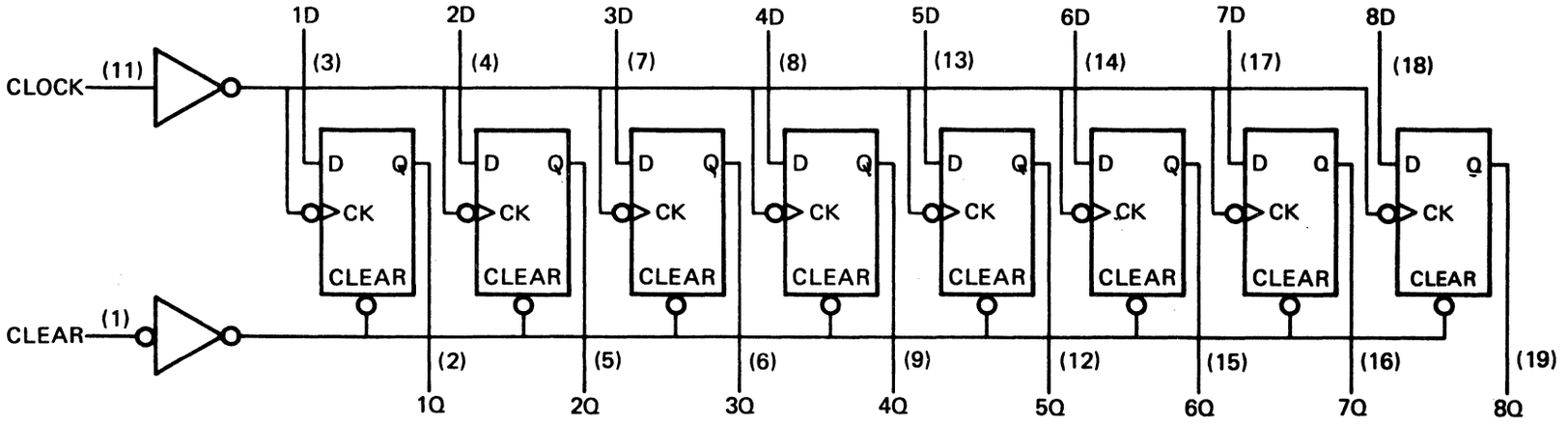
MK-0687

74S273 - OCTAL D-TYPE FLIP-FLOP WITH CLEAR

These monolithic, positive-edge-triggered flip-flops utilize TTL circuitry to implement D-type flip-flop logic with a direct clear input.

Information at the D inputs meeting the setup time requirements is transferred to the Q outputs on the positive-going edge of the clock pulse. Clock triggering occurs at a particular voltage level and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the D input signal has no effect at the output.

BLOCK DIAGRAM



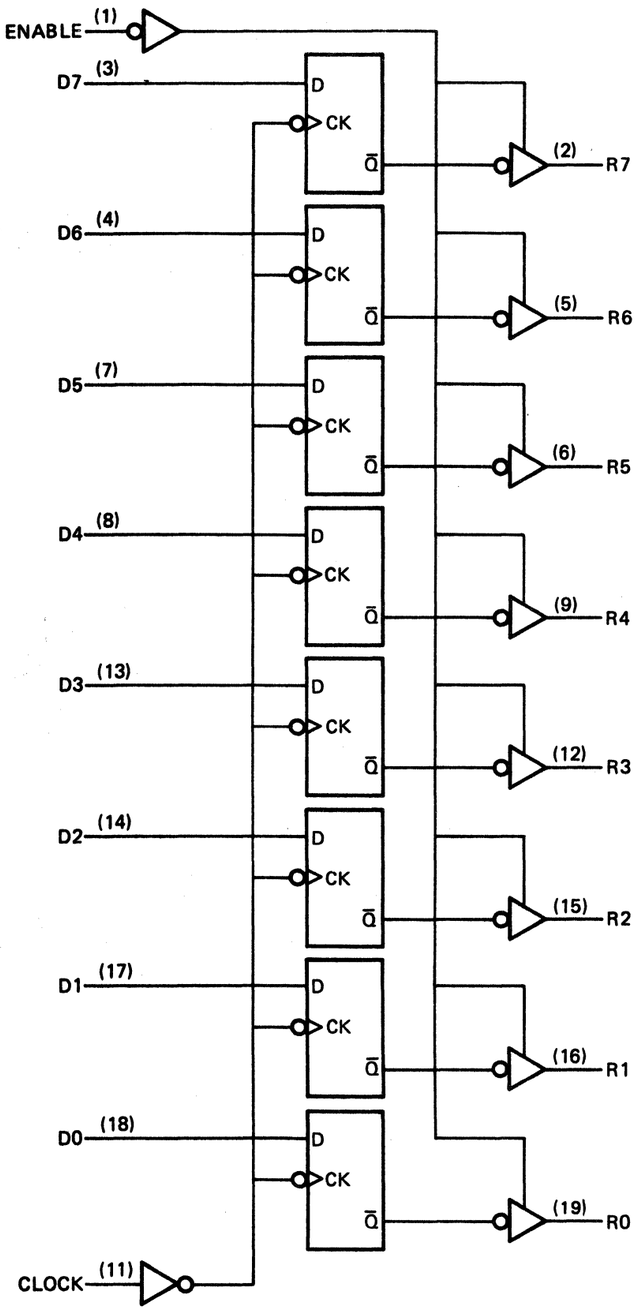
FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUT
CLEAR	CLOCK	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

H = HIGH LEVEL (STEADY STATE)
L = LOW LEVEL (STEADY STATE)
↑ = TRANSITION FROM LOW TO HIGH LEVEL
X = IRRELEVANT (ANY INPUT, INCLUDING TRANSITIONS)
Q_0 = LEVEL OF Q BEFORE THE INDICATED STEADY-STATE INPUT CONDITIONS WERE ESTABLISHED

MK-0683

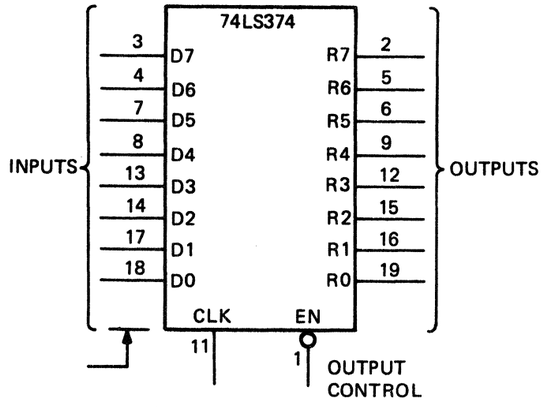
POSITIVE EDGE TRIGGERED FLIP-FOOP



74LS374 – OCTAL D-TYPE EDGE-TRIGGERED FLIP-FLOPS

The eight flip-flops of the 74LS374 are edge-triggered, D-type flip-flops. On the positive transition of the clock, the R outputs will be set to the logic states that were set up at the D inputs. A buffered output control input can be used to place the eight outputs in either a normal logic state (high or low logic levels) or a high-impedance state. In the high-impedance state, the outputs neither load nor drive the bus lines significantly. The output control does not affect the internal operation of the flip-flops; the old data can be retained, or new data can be entered, while the outputs are off.

LOGIC SYMBOL



FUNCTION TABLE

OUTPUT CONTROL	CLOCK	D	OUTPUT
L	↑	H	H
L	↑	L	L
L	L	X	Q_0
H	X	X	Z

H = HIGH LEVEL
L = LOW LEVEL
↑ = TRANSITION FROM LOW TO HIGH LEVEL
X = IRRELEVANT (ANY INPUT INCLUDING TRANSITIONS)
Z = OFF (HIGH IMPEDANCE) STATE OF A THREE-STATE OUTPUT
Q_0 = LEVEL OF R BEFORE THE INDICATED STEADY-STATE INPUT CONDITIONS WERE ESTABLISHED

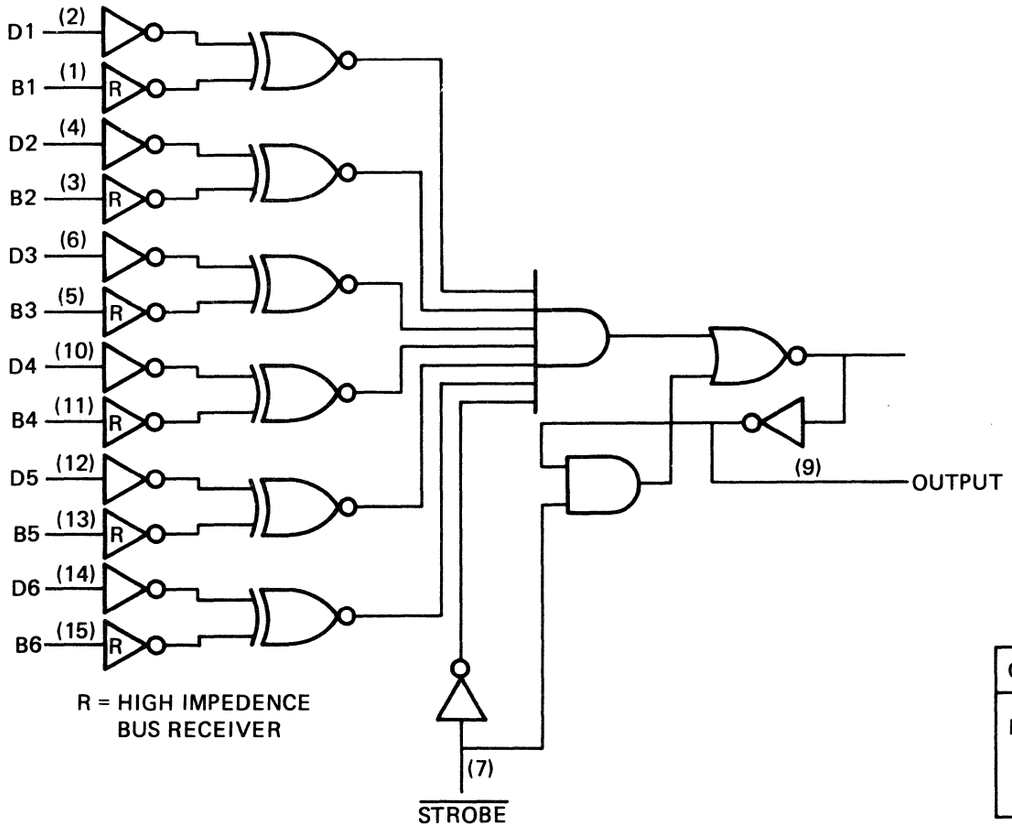
MK-0682

8136 - 6-BIT UNIFIED BUS COMPARATOR

The 8136 compares two binary words of 2- to 6-bits in length, and indicates matching (bit for bit) of the two words. The 8136 has open-collector outputs which go to the high state upon equality, and it is expandable to n bits by collector-ORing. The device has an output latch which is strobe controlled.

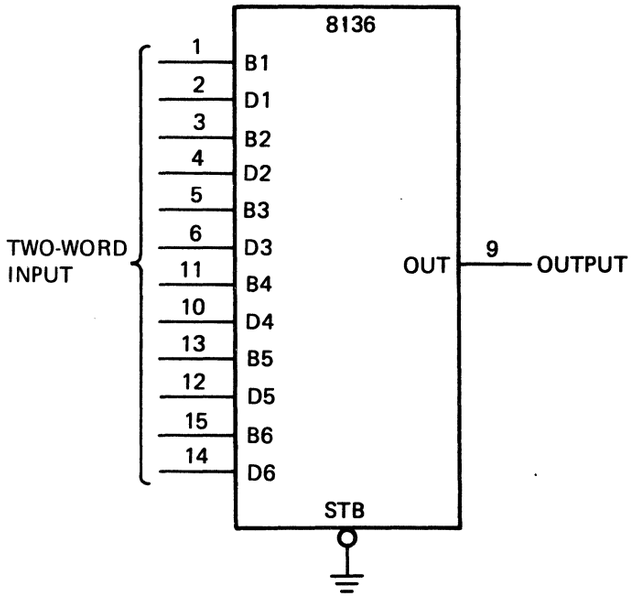
The transfer of information to the output occurs when the STROBE input goes from a logic "1" to a logic "0" state. Inputs may be changed while the STROBE is at the logic "1" level, without affecting the state of the output.

BLOCK DIAGRAM



R = HIGH IMPEDENCE BUS RECEIVER

LOGIC SYMBOL



TRUTH TABLE

CONDITION	$\overline{\text{STROBE}}$	OUTPUT
D = B, D ≠ B	H	$O_{N\,1}$*
D = B	L	H
D ≠ B	L	L

* LATCHED IN PREVIOUS STATE

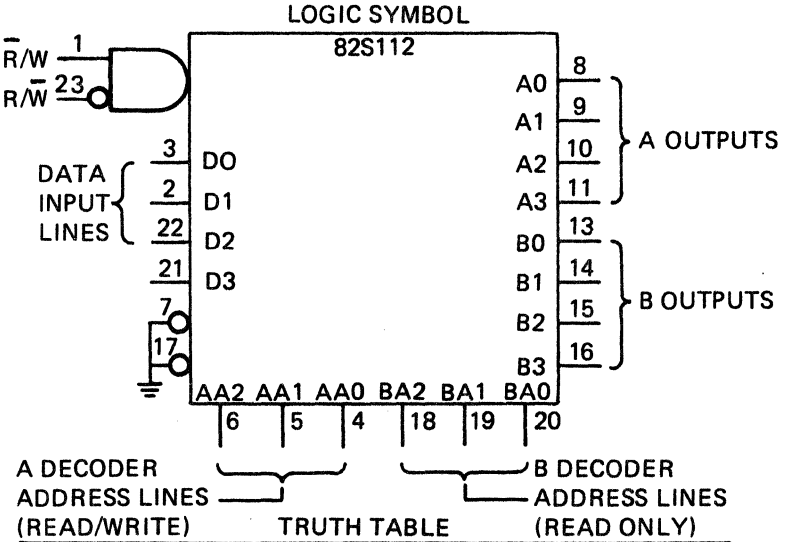
MK-0667

82S112 - HIGH SPEED 8 × 4 MULTIPORT RAM

The 82S112 is a Schottky TTL 32-bit multiport memory organized in eight words of 4-bits each. The device is ideally suited for high speed accumulators and buffer memories.

Stored data is addressed through two independent sets of three input decoders, and read out when the corresponding output enable line is low. Two separate word locations can, therefore, be read at the same time by enabling both the A and B output drivers. In addition, data can be read and written at the same time by utilizing the "A" address to specify the loction of the word to be written, and the "B" address to specify the word to be read.

LOGIC SYMBOL



TRUTH TABLE

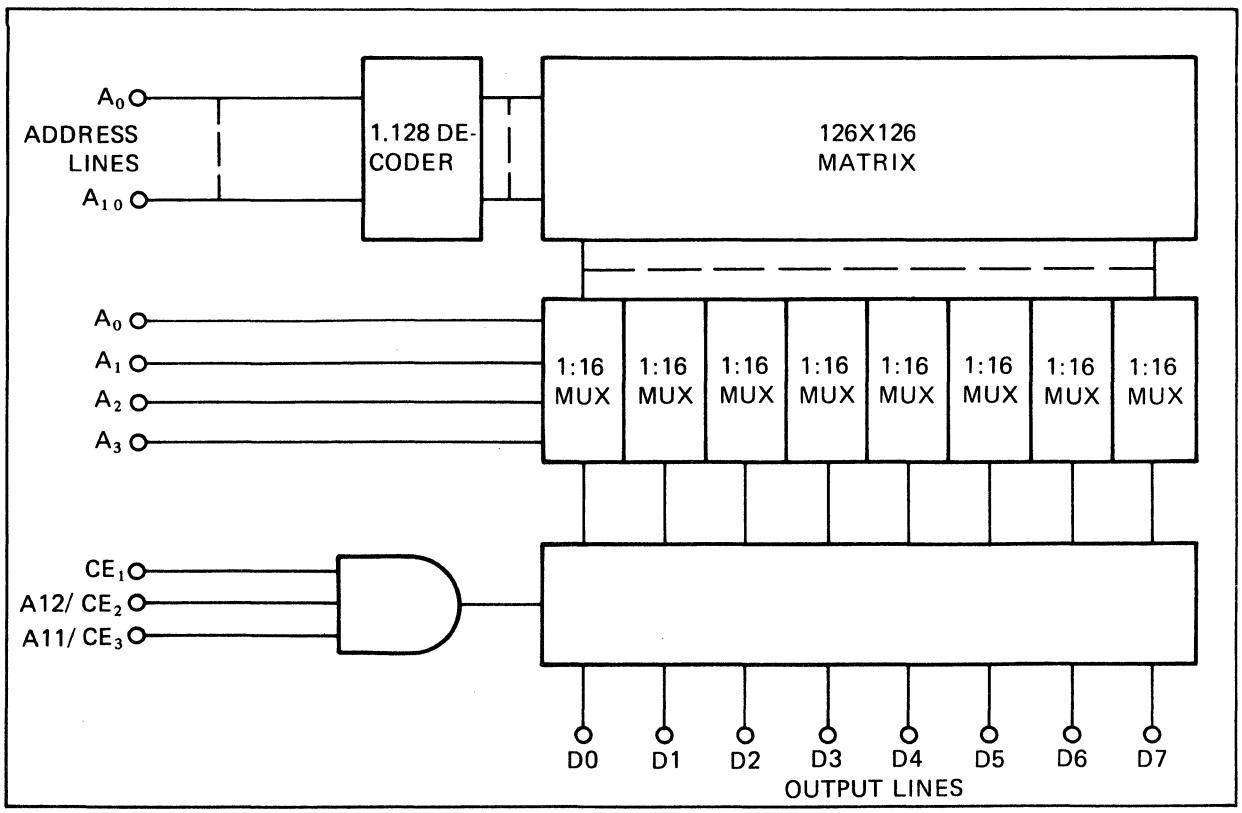
$\overline{R/W}$	R/$\overline{W}$	A OUTPUT ENABLE	B OUTPUT ENABLE	MODE	OUTPUTS A	OUTPUTS B
0	x	1	1	Outputs Disabled	"1"	"1"
0	x	1	0	Read	"1"	Data
0	x	0	1	Read	Data	"1"
0	x	0	0	Read	Data	Data
1	1	1	1	Read	"1"	"1"
1	1	1	0	Read	"1"	Data
1	1	0	1	Read	Data	"1"
1	1	0	0	Read	Data	Data
1	0	1	1	Write	"1"	"1"
1	0	1	0	Write	"1"	Data "B" Address
1	0	0	1	Write	Data Being Written	"1"
1	0	0	0	Write	Data Being Written	Data "B" Address

MK-0673

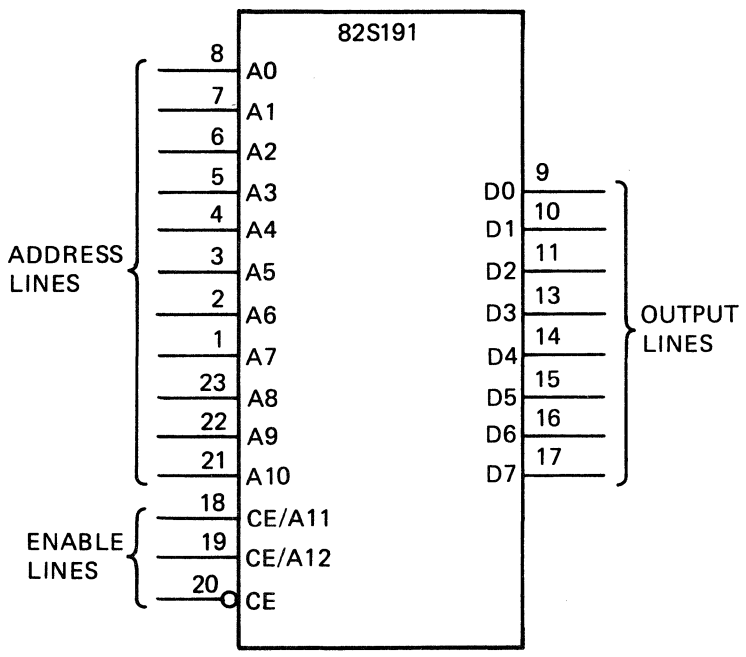
82S191 - 16,384-BIT 2K × 8 BIPOLAR PROM

The 82S191 is field programmable, supplied with all outputs at logical low. Outputs are programmed to a logic high level at any specified address by fusing a Ni-Cr link matrix. This device includes on-chip decoding and three chip enable inputs.

BLOCK DIAGRAM



LOGIC SYMBOL



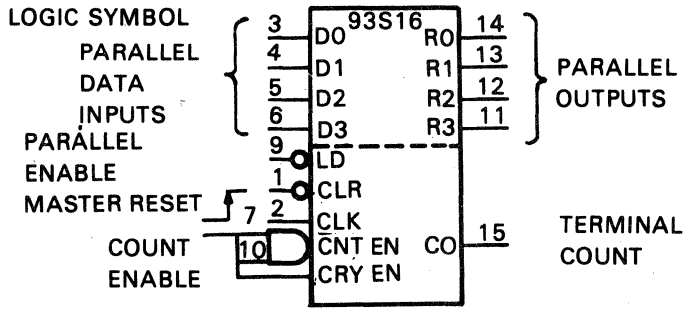
MK-0677

93S16 - 4-BIT BINARY COUNTER

The 93S16 is a fully synchronous 4-bit binary counter. With the parallel enable (LD) low, data on the inputs is parallel loaded on the positive clock transition. When LD is high and both count enables are also high, counting will occur on the low-to-high clock transition.

The terminal count state 1111 is decoded and ANDed with CRY EN in the terminal count (CO) output. If CRY EN is high and the counter is in its terminal count state, then CO is high.

The counter has an asynchronous master reset (CLR). A low on the CLR input forces the outputs low independent of all other inputs. The only requirements on the LD, CNT EN, CRY EN, and D0 - D3 inputs is that they meet the setup time requirements before the clock low-to-high transition.

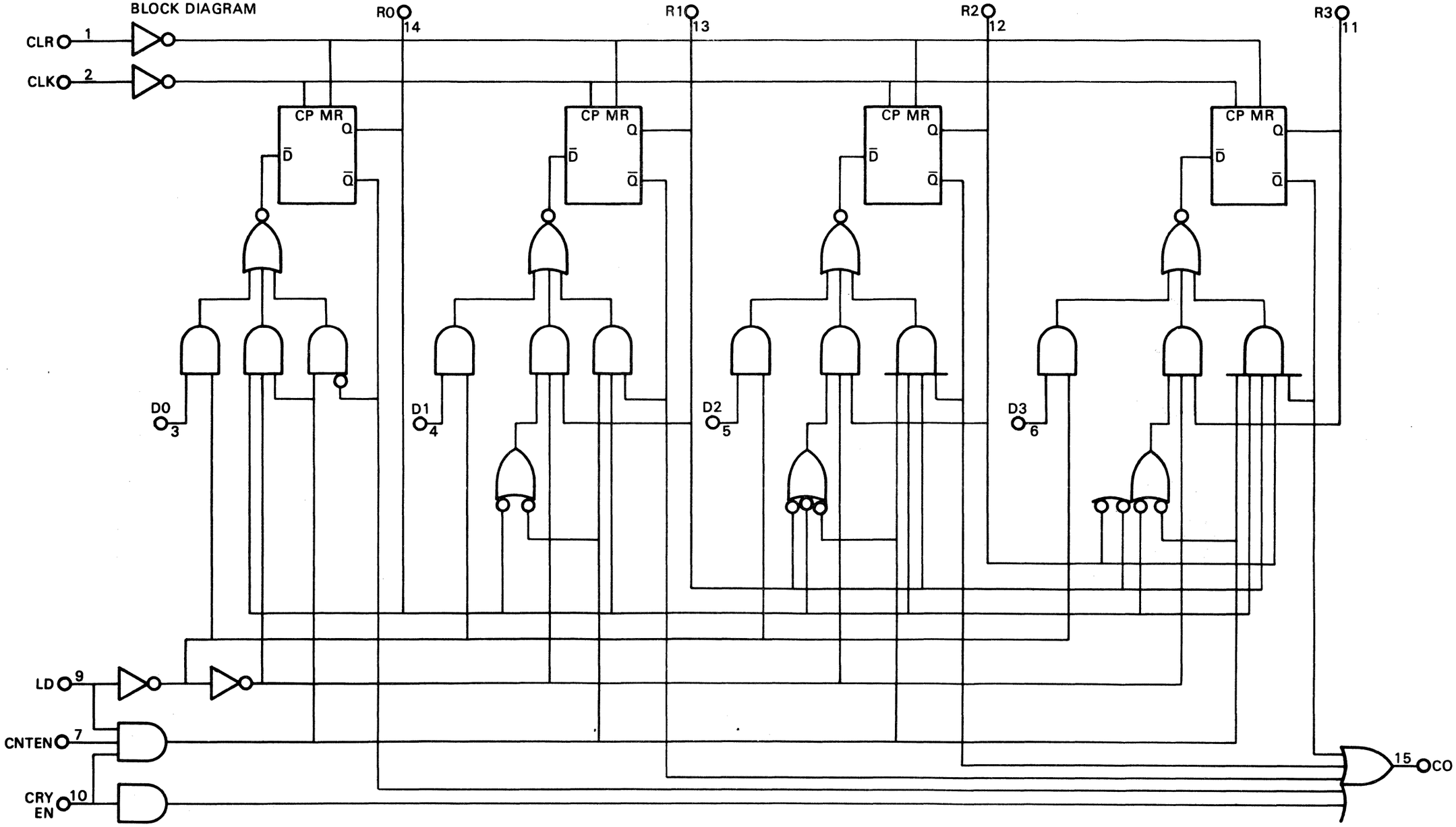


FUNCTION TABLE

INPUTS									OUTPUTS			
CLK	CLR	LD	ENT 7	EN 10	D0	D1	D2	D3	R0	R1	R2	R3
X	L	X	X	X	X	X	X	X	L	L	L	L
↑	H	L	X	X	D_0	D_1	D_2	D_3	D_0	D_1	D_2	D_3
↑	H	H	L	L	X	X	X	X	NC	NC	NC	NC
↑	H	H	L	H	X	X	X	X	NC	NC	NC	NC
↑	H	H	H	L	X	X	X	X	NC	NC	NC	NC
↑	H	H	H	H	X	X	X	X	COUNT			

H = HIGH
L = LOW
X = DON'T CARE
NC = NO CHANGE
D_I = MAY BE EITHER HIGH OR LOW
↑ = LOW-TO-HIGH TRANSITION

BLOCK DIAGRAM

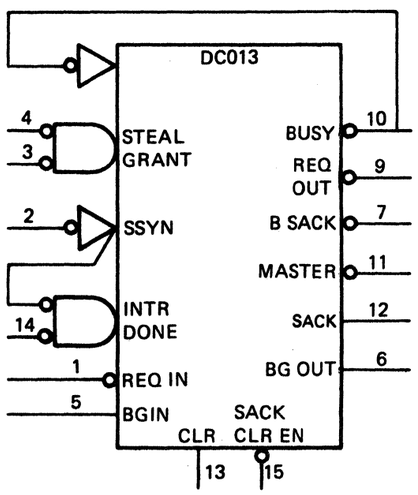


IC-93S16

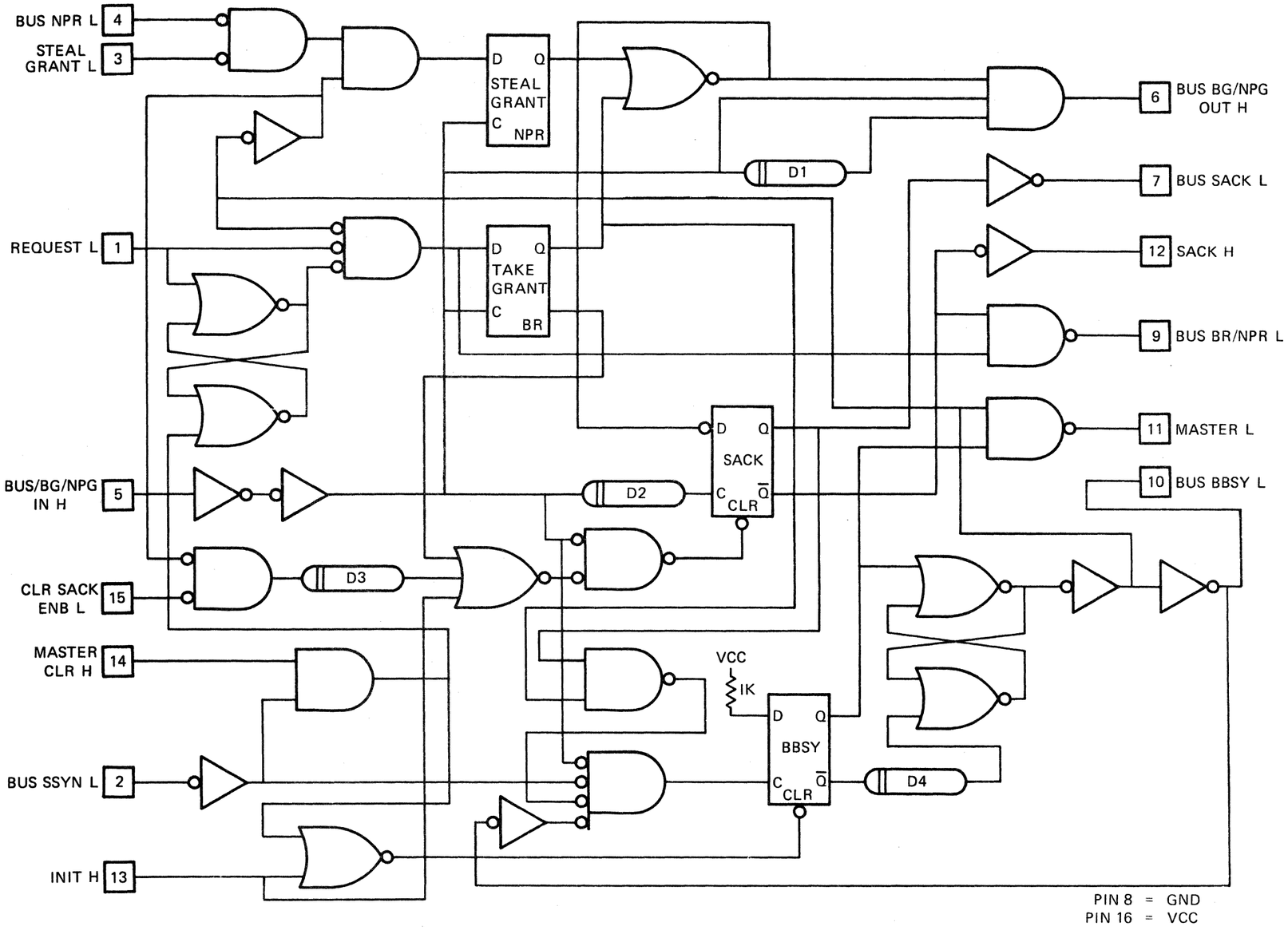
DC013 - DEC UNIBUS CHIP

The DEC DC013 UNIBUS chip is an LSI integrated circuit designed specifically for the UNIBUS and is useful in performing interrupt and NPR control. Its main function permits a device to make bus requests and to gain control of the UNIBUS (become bus master).

LOGIC SYMBOL



MK-0659



MK-0665

OPERATION – Refer to the logic diagram on page B-28.

REQUEST (pin 1) is asserted to initiate a BR or NPR. The request latch ensures only one priority transfer sequence is initiated for each assertion of REQUEST. Once the latch is set, REQUEST must be negated and asserted again before another cycle may begin, provided MASTER CLR (pin 14) is not grounded.

If the device is not asserting BUS BBSY (pin 10), the device is not already bus master, the D input to the TAKE GRANT flip-flop is asserted and one input to the BUS REQUEST driver is enabled.

BUS REQUEST (pin 9) is asserted while SACK flip-flop is in the reset state and at the same time as REQUEST is asserted by the device.

BUS BBSY (pin 10) is input to the grant circuitry; any grant received while this device is asserting BUS BBSY is passed.

TAKE GRANT AND STEAL GRANT flip-flops are clocked by the assertion of BUS GRANT IN (pin 5).

If the device is asserting BUS REQUEST, the next assertion of BUS GRANT IN causes the TAKE GRANT flip-flop to become set.
or
If STEAL GRANT (pin 3) is asserted and some device is asserting BUS NPR (pin 4) while the BBSY flip-flop is reset, the STEAL GRANT flip-flop is set.

Either flip-flop (TAKE GRANT or STEAL GRANT) set disables an input to the BUS GRANT OUT driver, preventing the grant from being passed and enabling the D input to the SACK flip-flop.

If the TAKE GRANT and SACK flip-flops are set, one input to the clock of the BBSY flip-flop is enabled.

If the device is not requesting the BUS, the REQUEST from the previous grant has not been negated and reasserted.
or
If the device is already asserting BUS BSSY, the TAKE GRANT flip-flop is cleared by the next assertion of BUS GRANT IN.

If the STEAL GRANT flip-flop is also cleared by the assertion of BUS GRANT IN, the grant is passed as an assertion of BUS GRANT OUT (pin 6).

The delay (D1) at the input to the BUS GRANT OUT driver ensures that the TAKE GRANT and STEAL GRANT flip-flops have had time to settle before any grants are passed to the next device of the system.

Assertion of BUS GRANT IN is also used to clock the SACK flip-flop. Delay (D2) at the input of the SACK flip-flop ensures that the TAKE GRANT and STEAL GRANT flip-flops have had time to respond to the assertion of BUS GRANT IN before the SACK flip-flop is clocked.

If either flip-flop (TAKE GRANT or STEAL GRANT) are set when the SACK flip-flop is clocked, the SACK flip-flop will set. When this occurs, BUS SACK (pin 7) is asserted, allowing the arbitrator to negate the grant after a minimum of 75 ns delay, and BUS REQUEST is negated.

If the TAKE GRANT flip-flop is set when the SACK flip-flop is set, one input to the clock input of the BBSY flip-flop is enabled. The SACK (pin 12) output is also asserted; it is available to the device as an inversion of BUS SACK.

SACK flip-flop may be reset by asserting CLR SACK ENB (pin 15) at the same time that BUS BBSY is asserted and BUS GRANT IN is negated. Delay (D3) at the output of the CLR SACK ENB gate ensures that the BUS INTR (external signal) driven by the MASTER signal (pin 11) is asserted before SACK flip-flop is cleared.

The SACK flip-flop may be cleared by the negation of BUS GRANT IN while the TAKE GRANT flip-flop is reset. The SACK flip-flop remains set while CLR SACK ENB is negated, keeping the arbitrator disabled.

BBSY flip-flop will set when the following clock input conditions are satisfied:

- TAKE GRANT and SACK flip-flops are set
- BUS GRANT IN, BUS SSYN (pin 2) and BUS BSSY are negated (not necessarily in that order)

BUS BSSY is asserted when the BBSY flip-flop is set. The BBSY driver input enables the MASTER output to:

- Disable the D inputs to the TAKE GRANT and STEAL GRANT flip-flops
- Allow the SACK flip-flop to be cleared

MASTER may be used by the master device to initiate a data transfer or an interrupt sequence. The MASTER signal is asserted when BUS BSSY has been asserted and is negated before the negation of BUS BSSY. Once BUS BSSY has been asserted, the device has gained control of the UNIBUS and the data transfer or interrupt sequence may proceed.

The MASTER signal is available to drive external gating, e.g., it may be used to enable the appropriate data lines and BUS INTR in an interrupt sequence.

When the requesting device has completed its transfer, it asserts MASTER CLR. When both BUS SSYN and MASTER CLR are asserted (signifying that the transfer is completed), the BBSY flip-flop is reset and the latch on the REQUEST signal is returned to the state which disabled the REQUEST input to the DC013. The MASTER signal is negated as soon as the BBSY flip-flop is reset.

The delay (D4) at the $\overline{Q}$ output of the BBSY flip-flop ensures that the negation of BUS BSSY is delayed for negation of the MASTER signal which is used to drive BUS INTR. Delay (D4) is guaranteed to a minimum of 80 ns.

APPENDIX C
M8207 MICROPROCESSOR JUMPER CONFIGURATION

The M8207 microprocessor jumper configurations for both the DMP11 and DMR11 options are provided in this appendix. The M8207 module comes in two versions:

- M8207-YA for DMP11 options, and
- M8207-RA for DMR11 options.

Jumper locations are shown in Figure C-1 and functions are discussed in Table C-1.

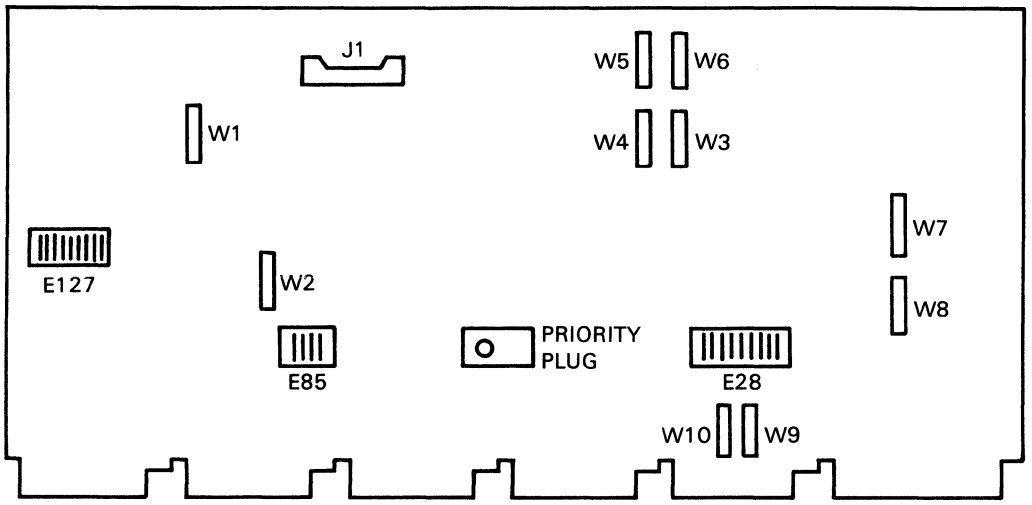


Figure C-1 M8207 Microprocessor Jumper Locations

Table C-1 M8207 Jumper Chart

Jumper Number	Normal Configuration	Function
W1	Always IN	Microprocessor Clock Enable – When removed it disables the microprocessor clock. Removed only for automatic module testing at factory.
W2	Always IN	Bus ac Low Enable – When removed it disables a program-asserted ac low signal passed onto the UNIBUS.
W4, W6, W7, W9	IN for this option only	For DMP11 option (M8207-YA).
W3, W5, W8, W10	IN for this option only	For DMR11 option (M8207-RA).

GLOSSARY

A Port
Read/write input to the Multiport Random Access Memory.

Arithmetic Logic Unit (ALU)
Allows the microprocessor to perform arithmetic and logic operations.

BERG Port
An 8-bit port that allows the microprocessor to communicate with other devices without using the UNIBUS.

Bit-Stuff Protocol
Zero insertion by the transmitter after any succession of five continuous ones designed for bit oriented protocols such as IBM's Synchronous Data Link Control (SDLC).

B Port
Read address input of the Multiport Random Access Memory (Read Only Port).

Branch Register (BRG)
Temporary data storage register used for branch determination and shifting right.

Buffered Arithmetic Logic Unit (BALU)
Operations performed by the ALU are buffered by the BALU and directed to data memory, respective registers, and the BERG Port.

Comite Consultatif Internationale de Telegraphie et Telephone (CCITT)
An international consultative committee that sets international communications usage standards.

Control ROM (CROM)
Plug-in Control Read Only Memory, used as the instruction memory for the microprocessor.

CPU
Central Processing Unit.

CSR
Control and Status Registers.

Data Multiplexer (DMUX)
An 8-bit wide, 8-to-1 multiplexer used to select data for the B input of the ALU.

DATI
Data In Transactions (Read).

DATO
Data Out Transactions (Write Word).

DATOB
Data Out Byte Transactions (Write Byte).

Destination ROM (DROM)
Controls the operand as defined by the destination of the instruction in the instruction register.

Digital Data Communications Protocol (DDCMP)
DIGITAL's standard communications protocol for character oriented protocol.

Direct Memory Access (DMA)
Permits input/output transfers directly into or out of memory without passing through the processor's general registers.

Electronic Industries Association (EIA)
Sets electronic industry standards for interfacing signals between units in the U.S.

Field Replaceable Unit (FRU)
Refers to a faulty unit not to be repaired in the field. Unit is replaced with a good unit, and faulty unit is returned to predetermined location for repair.

Full Duplex
Simultaneous two-way independent transmission in both directions.

Function ROM (FROM)
Controls up to 16 functions performed by the ALU.

Half Duplex
An alternate, one way at a time independent transmission.

IBUS/OBUS
Microprocessor NPR data, NPR address registers, and BERG Port registers.

IBUS*/OBUS*
Microprocessor NPR control and miscellaneous registers, and CSRs.

Instruction Register (IR)
Contains the instruction that is being executed. Outputs are used to control the microprocessor.

Line Unit Input Data Bus (LU IBUS)
Provides a path to the DMUX via the BERG Connector.

Main Memory (MEM)
Data storage area for the microprocessor 4K × 8 RAM; cannot be accessed directly by the CPU.

Maintenance Instruction Register (MIR)
Provides a destination for an instruction that can be loaded by the CPU during maintenance.

Memory Address Register (MAR)
Controls the data memory for buffered arithmetic and logic operations to main memory.

Multiport RAM (MP RAM)
Contains all M8207 control and status registers between the microprocessor and the CPU processor.

Non-Processor Request (NPR)
Direct memory access type transfers. See DMA.

Program Counter (PC)
A 14-bit counter used to control the address of the Control ROM directly.

Program Counter Register (PCR)
Contains upper six bits to be loaded into the PC on a branch condition, and provides extra field addressing with future expansion capability.

RAM
Random Access Memory.

ROM
Read Only Memory.

R232C
EIA Standard single-ended interface levels to modem.

RS 422
EIA Standard differential interface levels to modem.

RS 423
EIA Standard single-ended interface modem.

RS 449
EIA Standard connections for RS 422 and RS 423 to modem interface.

Scratchpad Memory (SP)
Read/write memory used for temporary storage of data.

Source ROM (SROM)
Defines the type of instruction to be executed and the source of the data used for the instruction.

System Clock
Basic timing generated by a 33.33MHz crystal providing microprocessor timing functions.

V.35 – CCITT Standard
Differential current mode type signal interface for high speed modems.

"Y" Version
Defines microcode stored in the Control ROMs.

INDEX

M8207 Microprocessor
Technical Manual
EK-M8207-TM-002
(MK)

Reader's Comments

Your comments and suggestions will help us in our continuous effort to improve the quality and usefulness of our publications.

What is your general reaction to this manual? In your judgement is it complete, accurate, well organized, well written, etc? Is it easy to use? _____

What features are most useful? _____

What faults or errors have you found in the manual? _____

Does this manual satisfy the need you think it was intended to satisfy? _____

Does it satisfy *your* needs? _____ Why? _____

☐ Please send me the current copy of the *Technical Documentation Catalog*, which contains information on the remainder of DIGITAL's technical documentation.

Name _____ Street _____
Title _____ City _____
Company _____ State/Country _____
Department _____ Zip _____

Additional copies of this document are available from:

Digital Equipment Corporation
444 Whitney Street
Northboro, MA 01532

Attention: Printing and Circulation Services (NR2/M15)
Customer Services Section

Order No. **EK-M8207-TM-002**

Fold Here

Do Not Tear – Fold Here and Staple

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 33 MERRIMACK, NH

POSTAGE WILL BE PAID BY ADDRESSEE

Digital Equipment Corporation
Educational Services/Quality Assurance
12 Crosby Drive, BU/E08
Bedford, MA 01730

Digital Equipment Corporation • Bedford, MA 01730