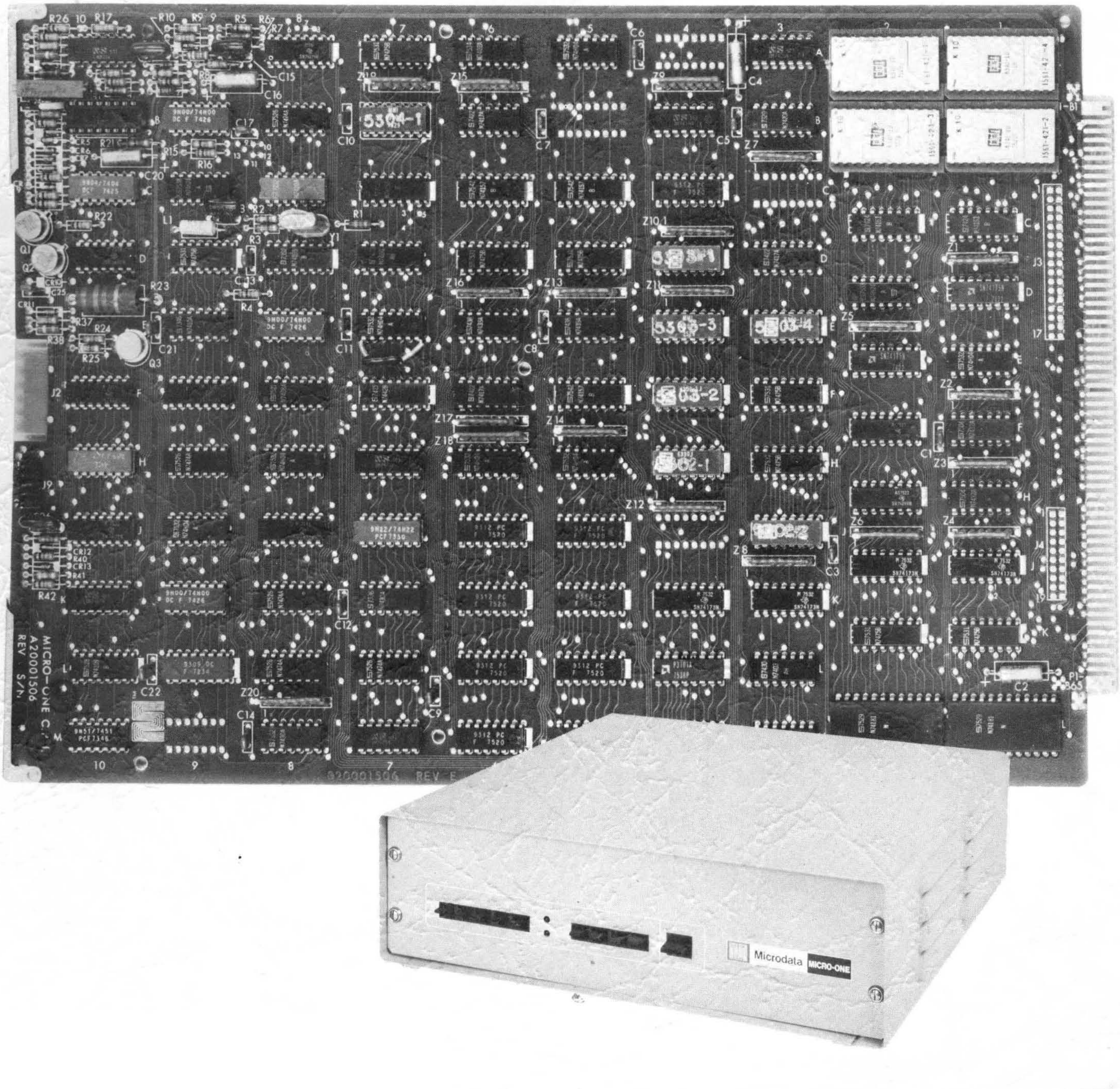
Microdata

Microdata MICRO-ONE USER'S MANUAL



Microdata MICRO-ONE

MICRO-ONE®

USER'S MANUAL

UM 20001506
OCTOBER, 1975

PROPRIETARY INFORMATION

The information contained herein is proprietary to and considered a trade secret of Microdata Corporation and shall not be reproduced in whole or part without the written authorization of Microdata Corporation.

™ Microdata

Microdata Corporation
17481 Red Hill Avenue
Irvine, California 92714
(714) 540-6730 TWX: 910-595-1764

98820 75 1011A

Printed in U.S.A.

MICRO ONE USER'S MANUAL

TABLE OF CONTENTS

Page

SECTION 1 ARCHITECTURE
1.0 SYSTEM ORGANIZATION 1-1
1.1 General Characteristics 1-1
1.2 Registers and File 1-6
1.2.1 T Register 1-6
1.2.2 M Register 1-7
1.2.3 N Register 1-7
1.2.4 L Register 1-7
1.2.5 U Register 1-7
1.2.6 R Register 1-7
1.2.7 LINK Register 1-7
1.2.8 I/O Control Register 1-7
1.2.9 File Registers 1-8
1.3 Memory Descriptions 1-8
1.3.1 Core Memory 1-8
1.3.2 MOS Memory 1-9
1.3.3 Control Memory 1-9
1.3.4 Memory Busy Delays 1-9
1.3.5 Memory Data Delays 1-9
1.3.6 Read-Only Memory Delays 1-9
1.4 Status and Condition Flags 1-10
1.4.1 Internal Status 1-10
1.4.2 Condition Flags 1-10
1.5 Byte I/O Interface 1-11
1.5.1 Byte I/O Bus 1-11
1.6 External Priority Interrupts 1-13
1.7 Real-Time Clock 1-13
1.8 Power-Fail/Automatic Restart 1-13
1.9 Arithmetic Functions 1-14

SECTION 2 MICROCOMMAND REPERTOIRE
2.0 GENERAL 2-1
2.1 Command Formats 2-1
2.1.1 Literal Commands 2-1
2.1.2 Operate Commands 2-1
2.1.3 Execute Command 2-2
2.1.4 Formats for Execute Commands 2-2
2.1.5 Literal Commands 2-3
2.1.6 Operate Commands 2-3
2.1.7 Terms and Symbols Used in the Command Descriptions 2-4
2.2 Microcommands - Formats, Descriptions, and Examples 2-4
2.2.1 LT Load T 2-4
2.2.2 LM Load M 2-5
2.2.3 LN Load N 2-5
2.2.4 LU Load U 2-6
2.2.5 LZ Load Zero Control 2-6

		Page
2.2.6	LS Load Seven Control	2-7
2.2.7	JP Jump	2-8
2.2.8	L Register Organization	2-8
2.2.9	LF Load File	2-10
2.2.10	AF Add to File	2-10
2.2.11	TZ Test If Zero	2-11
2.2.12	TN Test If Not Zero	2-13
2.2.13	CP Compare	2-14
2.2.14	K Control	2-15
2.2.15	Standard Output Functions	2-17
2.2.16	Standard Input Functions	2-17
2.2.17	A Add	2-23
2.2.18	S Subtract	2-26
2.2.19	R Read Memory W Write Memory	2-28
2.2.20	C Copy	2-33
2.2.21	O Or	2-33
2.2.22	X Exclusive Or	2-37
2.2.23	N And	2-39
2.2.24	H Shift	2-41
2.2.25	E Execute	2-42
SECTION 3	MICRO-ONE I/O AND MEMORY INTERFACE	
3.1	General Discussion	3-1
3.2	I/O Organization	3-1
3.3	Serial I/O Interface	3-1
3.4	Byte I/O Interface	3-1
3.4.1	Program-Controlled I/O	3-3
3.4.2	Concurrent I/O	3-3
3.4.3	External Priority Interrupts	3-3
3.4.4	Direct Memory Access Port	3-4
SECTION 4	BYTE I/O INTERFACE	
4.1	Introduction	4-1
4.2	Byte I/O Bus	4-1
4.2.1	Input Data Lines	4-1
4.2.2	Output Data Lines	4-1
4.2.3	Input Control Lines	4-1
4.2.4	Output Control Lines	4-3
4.2.4.1	Control Lines IO1X/ through IO3X/	4-3
4.2.4.2	Lines CPH1 and CPH2/.	4-4
4.2.4.3	Control Line MRST/	4-4
4.2.4.4	Control Line PROT/, PRIN/	4-5
4.2.4.5	Spare Lines	4-5
4.2.4.6	Control Line SELO/, SELI/.	4-5
4.3	Byte I/O Fundamentals	4-6
4.3.1	Device Addresses	4-6
4.3.2	Device Orders	4-7
4.3.3	Status Bytes	4-9
4.3.4	Function Bytes	4-9
4.4	Byte I/O Operations and Timing	4-10

		Page
4.4.1	Program Controlled I/O Operations	4-10
4.4.1.1	Address/Order Phase	4-11
4.4.1.2	Transfer Phase	4-12
4.4.1.3	Data Output Operations	4-12
4.4.1.4	Function Output Operations	4-12
4.4.1.5	Data Input Operations	4-13
4.5	Concurrent I/O Operation	4-14
4.5.1	Concurrent I/O Timing	4-14
4.6	External Interrupt Operation	4-16
4.6.1	Priority Determination	4-17
4.6.2	External Interrupt Requests	4-18
4.6.3	Interrupt Sequence and Timing	4-18
SECTION 5	MICRO-ONE CPU READ/WRITE MEMORY INTERFACE	
5.1	Processor and Memory Interface	5-1
5.1.1	Control Section	5-1
5.1.2	Memory Read Data Selection Logic	5-3
5.1.3	Memory Write Data Gating Logic	5-3
5.1.4	M and N Register Address Gating Logic	5-3
5.2	Memory Control Interface	5-3
SECTION 6	DIRECT MEMORY ACCESS PORT	
6.1	Introduction	6-1
6.2	Functional Description	6-1
6.2.1	DMA Interface	6-3
6.2.2	DMA Memory Control Logic	6-3
6.2.3	DMA Memory Read Data Receivers	6-4
6.2.4	DMA Memory Write Gating Logic	6-4
6.2.5	DMA Memory Address Gating Logic	6-4
6.3	DMA Port/Memory Control Interface Timing	6-4
6.3.1	Clock Signals	6-5
6.3.2	DMA Port Signals	6-5
6.3.2.1	DMA Request (DMAR/)	6-5
6.3.2.2	DMA Write (DMAW/)	6-5
6.3.2.3	Memory Busy (MBSY)	6-6
6.3.2.4	Memory Addresses	6-6
6.3.2.5	Write Data	6-6
6.3.2.6	Read Data	6-8
SECTION 7	SERIAL I/O INTERFACE	
7.1	Introduction	7-1
7.2	Use as Teletype Controller	7-1
7.2.1	General Operation	7-1
7.2.2	Character Assembly and Disassembly	7-1
7.2.3	Serial I/O Instructions	7-4
7.2.4	Teletype Interface Connection	7-4
SECTION 8	MICRO-ONE BACKPLANE CONNECTOR SIGNAL LIST	
8.1	I/O Controllers and DMA Interface Signal List	8-1
8.2	Serial TTY (J2)	8-1
8.3	Front Panel (Cable) Connector (J3)	8-1

Page

SECTION 9 I/O INTERFACE SIGNAL GLOSSARY

SECTION 10 OPERATOR CONTROLS
10.1 Consoles 10-1
10.1.1 System Console (Standard 1600 System Console) 10-1
10.1.2 Basic Console 10-1
10.2 Displays on System Console 10-1
10.2.1 Data Display 10-3
10.2.2 Run 10-3
10.2.3 Halt 10-3
10.2.4 Lock 10-3
10.2.5 Scan 10-3
10.2.6 Panel 10-3
10.2.7 Address Stop 10-3
10.3 Switches and System Console 10-3
10.3.1 Display Selector 10-3
10.4 Command Switches 10-4
10.4.1 Panel Switch 10-4
10.4.2 Sense Switches 10-4
10.4.3 Run 10-4
10.4.4 Step 10-4
10.4.5 Interrupt 10-4
10.4.6 Clock 10-4
10.4.7 Reset 10-5
10.4.8 On'Off-Lock 10-5
10.5 Address Sync 10-5
10.6 Register Display and Entry 10-5
10.6.1 Display 10-5
10.6.2 Enter 10-5
10.7 Operating Procedures -- System Console 10-6

SECTION 11 MICRO ONE CPU OPERATIONAL DESCRIPTION
11.1 General 11-1
11.1.1 Arithmetic Logic Unit and Multiplexer 11-1
11.1.2 Carry In 11-3
11.2 T. Register 11-7
11.3 File Registers 11-9
11.4 R and U Registers 11-9
11.5 L Register 11-9
11.5.1 L or K Destination in an Operate Command 11-12
11.6 Condition and Link Logic 11-14
11.7 Memory Address Registers (M and N) 11-14
11.8 Destination Register Clock Logic 11-14
11.9 Command Decode ROMs 11-18
11.10 Programmed Input/Output 11-19
11.11 Interrupts 11-19
11.12 Memory Sequencer (Core Memory Version) 11-22
11.13 Computer Clock and Run Control 11-26
11.14 Run/Halt Control 11-26
11.15 Computer Start Logic 11-29
11.16 Automatic Power Fail and Power on Detection Function 11-29

SECTION 12 SCHEMATICS

LIST OF ILLUSTRATIONS

Figure No.	Title	Page
1-1	Micro-One Block Diagram	1-2
1-2	Backplane Interconnections	1-12
3-1	Typical Micro-One Series I/O Configuration	3-2
4-1	Micro-One I/O Bus Lines	4-2
4-2	Relationship of Control Signals CPH1 and CPH2/	4-4
4-3	Data or Function Output Timing	4-12
4-4	Data or Status Input Timing	4-14
4-5	Concurrent I/O Timing	4-15
4-6	Typical Priority Scheme	4-17
4-7	Typical Selection Acknowledgment Scheme	4-18
4-8	External Interrupt Timing	4-19
4-9	Interrupt Sequencer States	4-20
5-1	CPU and DMA Memory Interface	5-2
5-2	Half Cycle Read	5-5
6-1	DMA/Processor Core Memory Interface	6-2
6-2	DMA Port/Memory Control Timing	6-7
7-1	Serial I/O Interface Circuit	7-2
7-2	Serial I/O Timing	7-3
10-1	MICRO 1600 System Control	10-2
11-1	Arithmetic/Logic Unit Block Diagram	11-2
11-2	2-Register Block Diagram	11-8
11-3	Fill Registers Block Diagram	11-10
11-4	R and U Register Block Diagram	11-11
11-5	L Register Block Diagram	11-13
11-6	Condition & Link Register	11-15
11-7	M and N Address Registers	11-16
11-8	Destination Register Clock Logic	11-17
11-9	Command Decode ROM's Block Diagram	11-18
11-10	Programmed I/O Block Diagram	11-20
11-11	Interrupts	11-21
11-12	Memory Controller Block Diagram	11-23
11-13	Clock Generation Logic, Block Diagram	11-27
11-14	Run/Halt Control Block Diagram	11-28
11-15	Computer Start Logic, Block Diagram	11-30
11-16	Power Fail Detect Block Diagram	11-31

LIST OF TABLES

Table No.	Title	Page
1-1	Microcommand Set	1-3
1-2	File Register O Flags	1-8
1-3	Internal Status Bits	1-10
2-1	Register Designation for Operate Commands	2-2
2-2	Standard I/O Control Codes	2-20
2-3		2-24
2-4		2-25
2-5		2-26
2-6		2-26
4-1	I/O Control States	4-4
4-2	Standard I/O Device Addresses	4-7
4-3	Standard Device Order	4-8
4-4	Typical Status Byte Definition	4-10
4-5	Interrupt Sequence States	4-20
8-1	Micro-One Backplane Connector Signal List	8-2
8-2	MOS Memory Interface Connector List	8-4
9-1	I/O Interface Signal Glossary	9-1
11-1	ALU MUX Addresses, ALU Modes, and Carry In	11-4

SECTION 1
ARCHITECTURE

SECTION 1

ARCHITECTURE

1.0 SYSTEM ORGANIZATION

The Micro-One Computer is a bus-organized system, constructed around a file of 15 programmable registers, which feature microprogrammed control. The basic elements of the system are shown in the block diagram of Figure 1-1. The system executes 15 basic microcommands with many variations, plus a code-variable execute command shown in Table 1-1. All microcommands are 16-bits long and are in one of three standard formats. Micro-One microprograms are established in a Read-Only-Memory (ROM) and thereafter become an integral part of the system's hardware. The microprogram can be changed by replacing the ROM devices. Commands read out of the ROM control all aspects of Micro-One's operation and are executed in a single 200 nanosecond machine clock cycle.

Micro-One's 8-bit Arithmetic/Logic Unit (ALU) performs all data manipulation, including: addition, subtraction, logical AND, logical OR, logical Exclusive OR, and 1-bit left and right shifts. The output of the logic network is the A-bus which is the input to the files and other system registers; all data byte movement occurs on this bus. The A-bus extends to the backplane and can be used for special I/O functions. The output of the register file is one of the inputs to the ALU, the other input is the B-bus. B-bus inputs are determined by the type of command, its options, and the I/O mode. B-bus inputs are the true and the complement outputs of the T register, the input bus, and the 8-bit literal contained in certain commands, and four external sense lines.

The memory data and address buses communicate between the core memory modules, the processor, and the Direct Memory Access (DMA) port. Either the processor or the DMA port may operate with the memory, with DMA having operational priority.

1.1 GENERAL CHARACTERISTICS

The heart of the Micro-One system is mounted on a single 8-1/2 x 12-inch printed circuit board which contains the basic CPU and the 1K ROM. There is a connection and interface control for a piggyback MOS memory of up to 8K bytes in 1K increments. Additional characteristics include:

- Core memory addressing to 64K bytes, or strap selectable to 32K bytes for operation with Micro-One/21 firmware.
- 1.0 Microsecond memory speed (full cycle)
- 8-Bit memory bytes
- Up to 1024 words of read only storage
- Two versions of control consoles

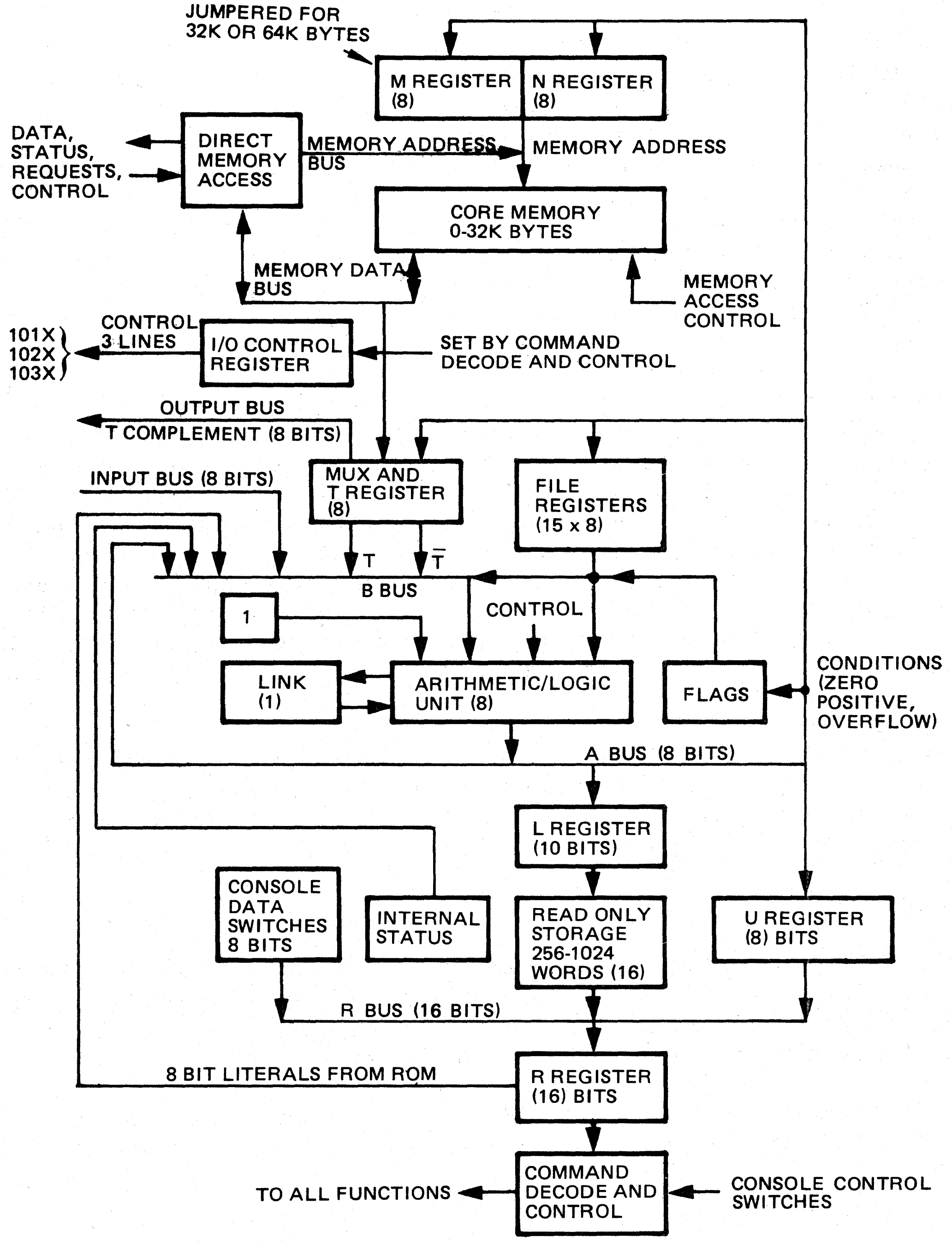


Figure 1-1. Micro-One Block Diagram

Table 1-1. Microcommand Set

No.	Name	Class Instruction	Code	Mnemonic	Literal to Register Subfunctions	Operations
0	Execute	Execute	0XXX	E	NA	0X is ORed with U Register
1	Literal to Register	Literal Class Commands	10XX	LZ	Load Zero	No operation
			11XX	LT	Load T	XX replace contents of T
			12XX	LM	Load M	XX replace contents of M
			13XX	LN	Load N	XX replace N & M is cleared.
			14XX	JP	Jump	to page 0.
			15XX	JP	Jump	to page 1.
			1CXX	JP	Jump	to page 2.
			1DXX	JP	Jump	to page 3.
			16XX	LU	Load U	XX replaces contents of U.
			17XX	LS	Load Seven	Internal Controls
2	Load File		2fXX	LF	N/A	f = file number
3	Add to File		3fXX	AF	N/A	f = file number
4	Test If Zero		4fXX	TZ	N/A	Skip on no bits match, if file f of the ones in the XX.
5	Test Not Zero		5fXX	TN	N/A	Skip on any bits match in file f of the ones in X.
6	Compare		6fXX	CP	N/A	Skip on f + XX>2^8 -1
						c Field (Binary)
7	Control	Operate Class Commands C field	7fc*r	K	destination register T,M,N,L,U N/A	0000 No operation 0001 Enter Sense Switches 0010 Shift Right Four Bits 0100 Enter Internal Status 0111 Enter Console Switches 1000 Clear I/O Mode 1001 Control Output 1010 Data Output 1011 Space Serial TTY

Table 1-1. Microcommand Set (Continued)

No.	Name	Class Instruction	Code	Mnemonic	Literal to Register Subfunctions	C field (binary)
7	Control	Operate Class Commands	7fc*r		N/A	1100 Concurrent Acknowledge 1101 Interrupt Acknowledge 1110 Data Input 1111 Spare
8	Add		8fc*r	A	N/A	0001 Modify Flags 0010 File + T 0100 Sum + 1 1000 Sum + Link Bit
9	Subtract		9fc*r	S	N/A	0001 Modify Flags 0010 File + T Complement 0100 Inhibit Increment 1000 Difference + Line
10	Read/Write Memory		Afc*r	R/S	N/A	00XX Transfer 01XX Decrement 10XX Add Link 11XX Increment XX1X Half Cycle XXX1 Write (Not Read)
11	Copy		Bfc*r	C	N/A	XXX1 Modify Flags XX1X Select T X1XX Select + 1 1XXX Select Link
12	OR		Cfc*r	O	N/A	XXX1 Modify Flags XX1X Select T X1XX Select T Complement 1XXX Linked Zero Test

Table 1-1. Microcommand Set (Continued)

No.	Name		Code	Mnemonic	Literal to Register Subfunctions	C field (binary)
13	Exclusive OR		Dfc*r	X		Same as OR
14	AND		Efc*r	N		Same as OR
15	Shift		Ffc*r	H		XXX1 Modify Flags XX1X Shift Right X1XX Insert ONE 1XXX Insert Link

NOTE: If* = 0, result of operation
is placed in file (f).

① f = file address

② c = sub op code field

③ * = inhibit file write

④ r = destination field

- TTL integrated circuitry
- Operating temperature 0 to 50°C; relative humidity 90%
- Compatibility with Micro 1600 interface controllers
- Power: +5V, 3A with 1K ROM
- Power fail detect and auto restart standard (requires full wave rectified 8V peak signal supplied from power supply)
- 120 Hz real-time-clock standard
- Serial TTY interface standard
- Single channel external interrupt
- Concurrent I/O using the programmed I/O bus
- Bidirectional memory data bus
- Separate 8-bit output and input data buses
- ROM memory sequencer which can be programmed for CORE OR MOS memory timing

1.2 REGISTERS AND FILE

There are eight CPU registers and 15 file registers. Each of the eight CPU registers has a specific use in the processor, while the files are used for general storage and flags.

1.2.1 T Register

The 8-bit T register serves as the operand register for most of the operate class commands, and as a buffer register for output and memory operations. Both the true and complement output of the T register can be gated to the B-bus as an operand. When both the contents of T and its complement are selected as operands, the effective operand is all 1-bits; if neither is selected the operand is all 0-bits.

The T register can be loaded directly from ROM using a Load T instruction, from core memory on a Read instruction, or it may be loaded from a file register from the input bus, or from itself (such as when incrementing T) by designating T as the destination register of an operate class command. All programmed outputs, including control and data bytes, go out via the T register.

1.2.2 M Register

The eight-bit M register contains the eight high-order bits of the processor memory address. This register is gated onto the Memory Address bus at all times except during a DMA operation. The M register can be loaded directly from ROM using a Load M command, or can be loaded by designating M as the destination register of an operate class command. The M register is cleared on a Load N command.

1.2.3 N Register

The eight-bit N register contain the eight low-order bits of the processor memory address. This register is gated onto the Memory Address bus at all times except during a DMA memory operation. The N register can be loaded directly from ROM using a Load N command, or by being designated as the destination register of an operate class command.

1.2.4 L Register

The 10-bit L register is the program counter and contains the read-only storage address of the next command to be executed, unless it is altered by a Jump command. The eight low-order bits of the L register serve as a counter which is incremented by one at each clock time when the processor is running unless a command execution delay is imposed. L is loaded by a Load L command, or as a destination register of an operate class command.

1.2.5 U Register

The eight-bit U register is used to modify the output of the read-only storage. For commands with Op Code 0 or nonliteral instructions with destination of 7, the contents of the U register are Inclusive-ORed with the eight high order bits of the ROM output as it is gated into the R register. This allows for dynamic modification and changing of operation codes and file register designators. U is loaded by a Load U command or as a destination register of an operate class command.

1.2.6 R Register

The 16-bit R register holds the present microcommand being executed. Its output is decoded and controls the operation of the processor at each clock time.

1.2.7 LINK Register

The one-bit LINK register holds the ALU'S high order carry from Add, Subtract, and Compare commands and the shifted off end bit from the Shift command.

1.2.8 I/O Control Register

This three-bit register generates the control signals for the I/O bus, Seven separate control signals can be developed by decoding the register outputs. It is loaded and cleared by a control command, placing the timing of I/O control signals under command control. There are three output modes and four input modes. The high-order bit of the register is the input flag. When this bit is a 1-bit, the input bus is substituted for the T register when it is selected, and is the source of data when executing an external I/O control command.

1.2.9 File Registers

Files consist of 15 eight-bit operational registers plus one File Zero register. All commands, except Load Register with Literal (Op-1), specify the file which will provide one input to the ALU. All file registers are functionally identical except for file register 0 which contains eight flags, and cannot be used for general storage. The flags of file register 0 are given in Table 1-2.

Table 1-2. File Register 0 Flags

Bit	Flag
0	Overflow Result Condition
1	Negative Result Condition
2	Zero Result Conditon
3	Concurrent I/O Request Line
4	Internal Interrupt
5	I/O Reply Line
6	Serial Teletype
7	External Interrupt Line

1.3 MEMORY DESCRIPTIONS

A brief functional description of the Micro-One's memories, memory busy, and memory data delays is provided in paragraphs 1.3.1 through 1.3.6.

1.3.1 Core Memory

The magnetic core memory of the Micro-One is organized into pluggable modules of 8K or 16K bytes. Addressed at the byte level, the memory is operated in read or write, and full or half-cycle operations. The full-cycle memory timing is five 200 ns clock cycles (1.0 microsecond); the half-cycle timing in the system is three clock cycles (600 ns). For a read operation, the accessed data is placed in the T register two clock cycles after the start of the memory operation. Full cycle regeneration of the data in the memory does not require the use of the T register and T may be modified by the microprogram before completion of the restore part of the cycle.

1.3.2 MOS Memory

The MOS memory of the Micro-One is organized into modules of 1K bytes with up to 8K bytes available. The memory is mounted piggyback on the Micro-One circuit board so that additional connectors are not required.

1.3.3 Control Memory

The read-only memory provides storage for commands and constants of the microprogram. Its output is gated into the R register where it controls system operation at the next clock time.

The ROM is always accessed for the next command while the current command is being executed. This look-ahead ability achieves faster command execution time. When the sequence of command execution is altered by a jump or skip, and additional cycle must be taken to perform an access before the next command is executed. When the unit is halted, the L register contains the address of the first command to be executed when operation is resumed.

Each command is executed in a single clock cycle time although execution may be delayed because of core memory or read-only memory operations. The system clock rate is 20 MHz, and the clock cycle is 200 nanoseconds.

1.3.4 Memory Busy Delays

When the memory is busy due to processor or DMA operations and a read/write command or a command to modify M or N registers awaits execution, a delay will occur until the memory operation is completed. These commands are executed on the last clock of the memory half or full cycle. If a DMA request is pending at the time a read or write memory command is to be executed, execution is delayed to give the DMA memory priority.

1.3.5 Memory Data Delays

Operate class commands which select the contents of either the T register or its complement during the first two cycles of a processor memory read operation are executed during the third cycle of the read operation. This allows time for the accessed byte to be placed in the T register.

1.3.6 Read-Only Memory Delays

An extra cycle is required for command execution for the following conditions because of the look-ahead nature of the read-only memory:

a. Jump command.

b. Test if Zero command when a skip occurs.

c. Test if not Zero command when a skip occurs.

d. Compare command when a skip occurs.

e. Operate class commands which have the L register designated as a destination.

1.4 STATUS AND CONDITION FLAGS

Status and Condition flags are described in the following paragraphs.

1.4.1 Internal Status

Eight internal status bits are provided in Micro-One to designate a particular internal interrupt condition. When any of the internal status bits is a 1-bit, the internal interrupt flag (bit-4) in file register 0 is also a 1-bit. This flag is tested by the microprogram to detect the presence of the internal interrupt condition. The internal status bits are entered via the B-bus into the selected file register by a control command, at which time the status bits are cleared. The assignments for the eight internal status bits are given in Table 1-3.

Table 1-3. Internal Status Bits

Bit	Interrupt Status
0	Console Interrupt
1	DMA termination
2	Real-Time Clock Interrupt
3	(Spare 0)
4	(Spare 1)
5	(Spare 2)
6	Console Step Switch
7	Power Fail/Restart Interrupt

1.4.2 Condition Flags

The Overflow, Negative and Zero conditions resulting from an operation involving the ALU can be stored in File Register 0. (See Table 1-2.) condition flags are updated for command 7 and for commands 8, 9, B-F if bit 4 is a 1-bit. These condition flags can be tested by the microprogram for implementing various conditional operations. Definitions of the condition flags follows:

a. Overflow - The Overflow Condition Flag stores the arithmetic overflow condition during an add, subtract or copy command. It stores the shifted off end bit during a shift command. Arithmetic overflow occurs when the result exceeds the range of the computer's 8-bit registers.

b. Negative - The Negative Condition Flag stores the high-order bit of the result on the A-bus since the 2's complement number system uses the most significant bit as the sign bit.

c. Zero - The Zero Condition flag stores the zero condition of the result. The zero test can be linked over multiple byte operations under control of the LINK modifier (bit 7) of operate instructions. When this bit is 1, the Zero Condition flag may not be set to indicate the zero condition of the current byte, but may only be reset to indicate a non-zero result. For this flag to indicate zero over multiple bytes it must be set by a zero result on the first operation which will have the LINK modifier zero, and not be reset by non-zero conditions on succeeding bytes which will have the LINK modifier a one.

1.5 BYTE I/O INTERFACE

The Micro-One provides an extremely fast elementary I/O capability. (The basic interconnections are shown in Figure 1-2.) The data paths and control functions are simple elements, sequenced from the control memory with flexible disciplines. With the fast (200 ns/step) control memory, firmware microprograms in control memory can react with a high degree of versatility in timing, data paths and I/O capabilities. This includes priority interrupts, fully-buffered data channels, macroprogrammable transfers, and special purpose communication multiplexer channels.

The byte I/O interface provides the facility for transferring bytes over a party line I/O bus under microprogram control. Standard Micro-One firmwave provides both programmed I/O and concurrent I/O transfer capability, along with a priority interrupt system. The basic I/O element is the Byte I/O bus.

1.5.1 Byte I/O Bus

Data transfers through the byte I/O interface are basically two-phase operations. During the first phase, a control byte is placed on the Byte I/O bus before the actual transfer of data. The control byte contains a device address specifying the address of one of the I/O controllers on the bus, and a device order code signifying the type of operation to be performed during the transfer (data, status, or function transfer, etc).

All controllers on the bus examine the device number, but only the addressed controller accepts the control byte and logically connects itself to the bus for the subsequent data byte transfer. During the second phase of the byte I/O operation, a single byte is transferred to or from the controller. After each byte transfer the controller disconnects itself from the bus.

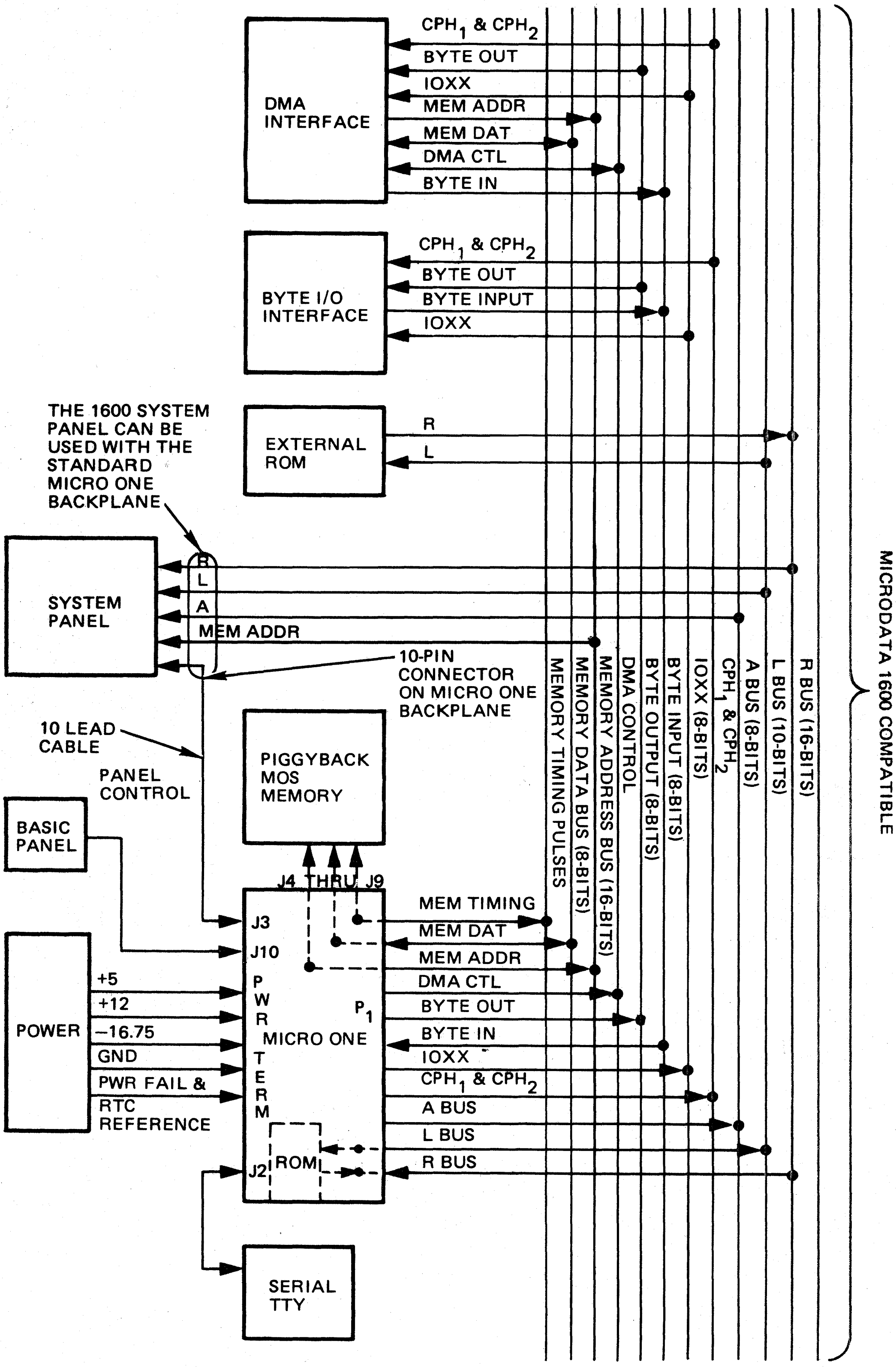


Figure 1-2. Backplane Interconnections

1.6 EXTERNAL PRIORITY INTERRUPTS

The Micro-One external interrupt system operates through the byte I/O interface in the computer mainframe. Interrupts can originate from device controllers or from an optional Priority Interrupt interface board connected to the Byte I/O bus. This interface board controls eight external interrupt signals.

The byte I/O interface contains a single external interrupt request line common to all controllers on the Byte I/O bus, and a priority line that progresses sequentially through all controllers on the bus. Each I/O controller receives priority from the preceding controller in the priority chain and, if it is not ready to request an interrupt, passes it along to the next controller in the line. When a controller has priority and is ready to request an interrupt the priority signal is halted and the Interrupt Request signal is activated. After the I/O request is acknowledged, the controller places an address byte on the I/O bus. The processor uses this byte to transfer program control to the proper interrupt servicing routine.

1.7 REAL-TIME CLOCK

The standard real-time clock function provides an internal interrupt at a 120 Hz rate. This can be used at the macroprogramming level as a real-time clock. The timing is derived from the power line where it is full-wave rectified. An optional input and jumper select is provided to make it possible to use an externally supplied real-time clock signal at rates other than 120 Hz.

When the timing signal occurs, it provides an internal interrupt by setting condition flag bit 4 and bit 2 of the internal status byte. The timing signal internal interrupt may be disabled and enabled by commands 1710 and 1720 respectively. The microprogram must detect the internal interrupt and take appropriate action. (Special real-time clock interrupt handling firmware is available with the Micro-One.)

1.8 POWER-FAIL/AUTOMATIC RESTART

The power-fail and automatic restart function provides the following:

a. An internal interrupt by setting condition flag bit 5 and bit 7 of the internal status byte upon detection of primary power loss.

b. A processor reset when the computer is halted after loss of primary power.

c. A processor rest for over 100 milliseconds after power is applied.

d. Automatic switch to run mode after the power-on reset period.

e. Power-restart interrupt immediately after automatic switch to run mode.

A power-fail interrupt detected while the processor is in the Run mode can be used to store processor registers and to halt the processor. The automatic processor reset that follows the halt and the one following power-on prevents any spurious operations in the core memory. At power-on, the processor reset clears the L register causing the processor to start at ROM location 0. The power-fail interrupt which occurs at this time can be detected and treated as a restart interrupt to cause a restoring of the processor registers. Standard power-fail/automatic restart interrupt firmware is available. The +12V power to the Micro-One must remain above +5V during a power-fail as the two voltages collapse.

1.9 ARITHMETIC FUNCTIONS

The Micro One uses a 2's complement binary number system. The registers and memory cells are 8 bits in length. For programming convenience, entering data, printing out, and preparing punched paper tape, the 8 bits are organized into two hexadecimal digits. The hexadecimal digits, with their decimal and binary equivalents, are as follows:

Decimal	Hexadecimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Throughout this document hexadecimal numbers are identified with single quotes:

'33'
'AA'

For addition functions, the two numbers are added directly with the carry out of the most significant bit going to Link, and overflow setting the overflow bit, if designated in the command.

For subtraction, one number is converted to a 2's complement and added to the other.

For single byte operations, with a 2's complement number system, the range of numbers is as follows:

Binary	Hexadecimal	Decimal	
01111111	'7F'	+127	POSITIVE
--------	----	----	
00000001	'01'	+ 1	
00000000	'00'	0	
11111111	'FF'	- 1	NEGATIVE
11111110	'FE'	- 2	
--------	----	----	
10000000	'80'	-128	

In general, arithmetic overflow occurs whenever the number range (+127 to -128) of the Micro One is exceeded on an arithmetic operation. As can be seen in the examples, the link bit may be set even though an overflow did not occur. This is the result of using a 2's complement number system.

SECTION 2

MICROCOMMAND REPERTOIRE

SECTION 2

MICROCOMMAND REPERTOIRE

2.0 GENERAL

This section contains description of all Micro-One commands. With each description is a diagram showing the format of the command and its operation code, given in hexadecimal. Above each diagram is the command's mnemonic code and the name of the command. Under each diagram is a description of the command, followed by a list of the registers and indicators that can be affected by the command. The timing of each command is one clock cycle (200 nsecs) unless the L register is designated as the destination of the result, in which case the command execution time is two cycles.

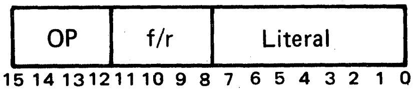
2.1 COMMAND FORMATS

There are three basic command formats. Each command is 16 bits long and is contained in a single read-only memory location.

The formats are literal commands, operate commands and execute commands.

2.1.1 Literal Commands

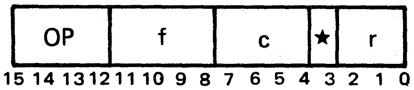
The literal class commands have the following format;



In this format the operation code occupies the four high-order bits. Bits 11-8 contain either a file register designator (f) or a register or control group designator (r). Bits 7-0 comtain an 8-bit literal which is transferred as an operand to the B bus.

2.1.2 Operate Commands

The operate class commands have the following format:



In this format the operation code occupies the four high-order bits. Bits 11-9 contain a file register designator (f) which specifies 1 of the 16 file registers to be used in command execution. Bits 7-4 contain control option bits (c) which are unique to the specific command. When bit 3 is a 1, the result of an operate class command is inhibited from being placed in the designated file register. Symbolically, this is specified to the program assembler by appending an * to the command mnemonic. The register designator (r) in bits 2-0 specifies a processor register destination to receive the result of the operation.

Since there is only one file register selected at a time, the only file register that can receive the result of a particular operate command is the same file register selected for the operand. The register's identifier is added as a second character of the command mnemonic. Table 2-1 contains the register codes.

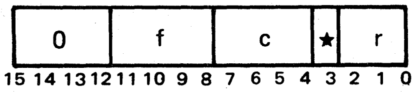
Table 2-1. Register Designators for Operate Commands

Designator	Mnemonic	Register
0		none
1	T	T Register
2	M	M Register
3	N	N Register
4	L	L Register-addresses: 000-0FF and 200-2FF
5	K	L Register-addresses; 100-1FF and 300-3FF
6	U	U Register
7	S	U Register ORed into command (except for Control command)

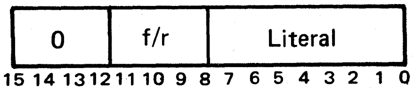
2.1.3 Execute Command

The execute command causes the contents of the U register to be ORed with the 8 high-order bits of the command to form an effective command. This operation is also performed when r=7 for the operate class commands. The execute command has zero bits in the four high order bits. The remainder of the command has the format required for the effective command to be executed.

2.1.4 Formats for Execute Commands



If U contains Operate command OP code.



If U contains Literal command OP code.

2.1.5 Literal Commands

The literal commands, listed by Op Code are as follows:

Op Code	Command
1	Load Register
2	Load File
3	Add to File
4	Test Zero
5	Test Not Zero
6	Compare

The literal commands are used to load constants into various Micro-One registers, to test for bit configurations and data values in file registers, and to load or add constants to file registers. Eight of the 16 bits are used as command, and the other 8 are available as data.

2.1.6 Operate Commands

The operate commands, listed by Op Code are as follows:

Op Code	Command
7	Control
8	Add
9	Subtract
A	Memory
B	Copy
C	OR
D	EXCLUSIVE OR
E	AND
F	SHIFT

The operate commands are used to control the flow of data in or out and through the Micro-One computer, and to perform the arithmetic and logic functions in the computer.

With this powerful command set, it is possible to implement all of the data handling and control functions of a larger computer.

2.1.7 Terms and Symbols Used in the Command Descriptions

(f_1)	Contents of file 1
$(f_1) \longrightarrow T$	Contents of file 1 to T register
....	Indeterminate value or function
'AA'	Hexadecimal number in flow chart
X'AA'	Hexadecimal constant in assembly language statement

Affected Register States

For each command, certain registers are modified. These are described in examples as affected registers.

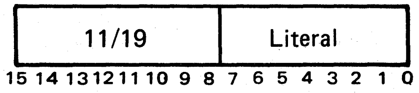
$\wedge$ LOGICAL AND
$\vee$ LOGICAL OR
$\veebar$ LOGICAL EXCLUSIVE OR

L '024'	Effective address of L register as used in examples. Because of the lookahead feature of the Micro-One, the actual L address is one higher than indicated in the examples.

2.2 MICROCOMMANDS - FORMATS, DESCRIPTIONS, AND EXAMPLES

The formats of the examples for each command have been selected to facilitate explanation of that particular command. Because of the difference in characteristics and utilization of the various commands, and associated data patterns, the example formats are different for each command category.

2.2.1 LT Load T



The contents of the 8-bit literal field are placed in the T register. The condition flags and Link register are not effected.

This command is used to provide constant data values, bit patterns for comparison tests, masks, and I/O control codes, which are most conveniently used in the T register.

Example: Load T with hexadecimal value 'AA'

L	Machine Code	Assembly Language	Flow Chart Notation
'024'	'11AA'	LT X'AA'	'AA' ⟶ T

Effected register states:

Register	Before	After
L	'024'	'025
T	.....	'AA'

Command execution time - 200 nsecs

2.2.2 LM Load M

12	Literal

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

The contents of the 8-bit literal field are placed in the M register. The condition flags and Link register are not effected.

This command is used to set the M register for accessing dedicated core locations. The M register is also modified by designation as destination register in operate commands.

Example: Load M with page address hexadecimal value '55'

L	Machine Code	Assembly Language	Flow Chart Notation
'134'	'1255'	LM X'55'	'55' → M

Effected register states

Register	Before	After
L	'134'	'135'
M	....	'55'

Command execution time - 200 nsecs

2.2.3 LN Load N

13	Literal

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

The contents of the 8-bit literal field are placed in the N register and the M register is cleared. The condition flags and Link register are not effected.

The command is used to set the N register for accessing dedicated core locating locations. If the location is in page 0 of core ('0000'-'00FF'), only this command is required to set both the M and N registers, since M is automatically cleared. If M is not to be page 0, then N must first be set, followed by M.

Example: Load N with address hexadecimal value "F" and set M = '00'

L	Machine Code	Assembly Language	Flow Chart Notation
'235'	'13FF'	LN X'FF'	'FF'→N '00'→M

Effected register states

Register	Before	After
L	'235'	'236'
M	---	'FF'
N	---	'00'

Command execution time - 200 nsecs

2.2.4 LU Load U

16	Literal
15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0

This command is used to place specific command codes into the U register, which is used in conjunction with general function EXECUTE class commands. The U register can also be modified by being designated as the destination register in an operate command. The differences in utilization of these two approaches for modifying the U register are described in a later paragraph which discusses U register applications.

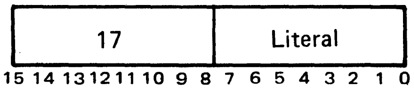
When modifying the U register, it is necessary to place at least one command between the modifying command and a command which uses the U register as an input. Otherwise an undefined value of U may be used.

2.2.5 LZ Load Zero Control

10	Literal
15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0

When this command or an operate command with destination 0 is executed, a pulse called CGLO of approximately 50 nsecs width is generated. CGLO is available on the I/O and option board connectors of the Micro-One. During the CGLO of a literal command, the literal value is on the A bus, which is available on the back plane. An 8-bit control latch can be set by this command and used for any purpose, such as enabling counters, interrupts, or control lines. Since the CGLO also occurs on dest. 0, logic must be provided to detect the literal command (OP1) and enable the CGLO only when an OP1 has occurred.

2.2.6 LS Load Seven Control



The 8 bits of the literal perform control functions as described below:

1700 - No operation

1701 - Enable serial teletype. The serial teletype input is gated into bit 6 of file register 0. The serial teletype value is available at all times.

1704 - Disable external interrupts: recognition of external interrupts is inhibited.

1708 - Enable external interrupts: recognition of external interrupts is enabled.

1710 - Disable real-time clock: the real-time clock and interrupt are disabled.

1720 - Enable real-time clock: the real-time clock and interrupt are enabled.

1740 - Spare

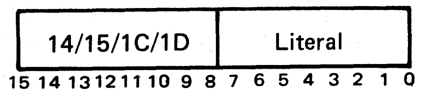
1780 - Halt: the processor is halted.

Command execution time - 200 nsecs.

Non-conflicting commands can be executed simultaneously. For example, enable external interrupts can be combined with enable real-time clock. The bits of the literal parts of the commands are ORed to produce the hexadecimal code.

Example:	Machine Code	Literal Bits	
Enable Interrupts	1708	0000	1000
Enable Real-Time Clock	1720	0010	0000
Composite Command	1728	0010	1000

2.2.7 JP Jump

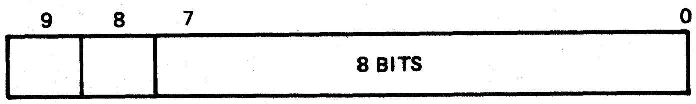


The contents of the 8-bit literal are placed in the 8 low-order bits of the L register; the content of bit 8 is placed in L_8 and the content of bit 11 is placed in L_9. The location of the next command to be executed is at the address specified by the new contents of the L register. The execution time of the command is two cycles. The jump operation codes for the four 256-word pages in read-only memory are as follows:

14	-	Jump to locations 000-0FF (page 0)
15	-	Jump to locations 100-1FF (page 1)
1C	-	Jump to locations 200-2FF (page 2)
1D	-	Jump to locations 300-3FF (page 3)

In order to fully explain this command, a detailed description of the L register follows.

2.2.8 L Register Organization



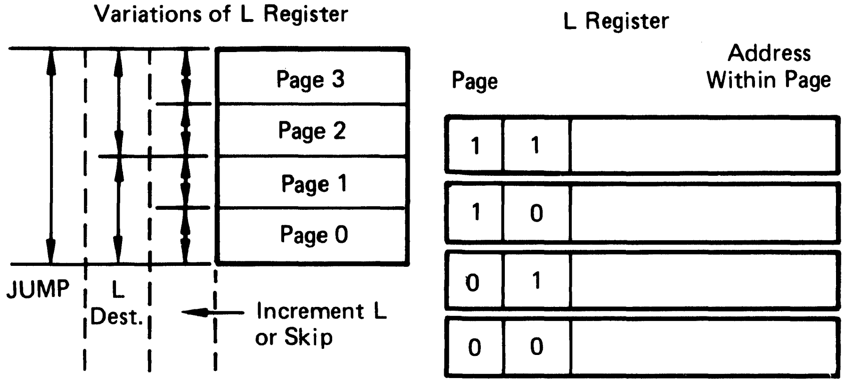
Bits 0 to 7 act somewhat like a counter as they are incremented like a counter after each command execution except conditional skips, jumps, or operate commands containing L or K as a destination. If the L count is at XFF, and the next command causes L to be incremented, the L count will go to X00, with no indication of a carry. If a command causes L to skip, L will go from XFF to X01 prior to execution of a microcommand.

To change pages, it is necessary to change bit 8 or 9. Bit 9 can only be changed with a jump (literal to L) command. With the jump command, any part of L can be reached.

Bit 8 can be changed with either a jump command or by designating the L register as the destination register in an operate command.

As shown in Table 2-1, a destination designator of 4 or 5 effects the L register. The designator 4 causes bit 8 to reset, and 5 causes bit 8 to set. In the assembly language mnemonics, a 4 is labeled L, and a 5 is labeled K.

The various methods of changing L are shown in the following read-only map outline.



Since L is always addressing the next command to be executed, any condition, such as a skip, jump, or L destination results in a clock cycle skip because the "next" command must be discarded for a new "next" command.

Examples:

L	Machine Code	Assembly Language	Flow Chart Notation
1) Jump to page 0 location '33'			
'021'	'1433'	JP X'033'	'033'→L Sometimes just shown as a line from one block to another in a flow chart
2) Jump to page 2 location '46'			
'150'	'1C46'	JP X'246'	'246'→L
3) Jump to page 3 location '31'			
'230'	'1D31'	JP X'331'	'331'→L

L Register States

Example	Before	After
1	'021'	'033'
2	'150'	'246'
3	'230'	'331'

Command execution time - 400 nsecs

2.2.9 LF Load File

2	f	Literal
15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0

The contents of the 8-bit literal field are placed in the file register designated by f. File register 0 cannot be loaded by this command. The condition flags and Link register are not affected.

This command is used for initializing or clearing the registers. It is also used for setting relative and absolute jump addresses into files. It can also be used as part of a table look-up routine. Another use is for setting indirect return addresses into files.

Example of load file command:

L	Machine Code	Assembly Language	Flow Chart Notation
'025'	'2355'	LF 3,X'55'	'55'→f3

Affected register states:

Register	Before	After
L	'025'	'026'
file 3	---	'55'

Execution time - 200 nsec

2.2.10 AF Add to File

3	f	Literal
15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0

The contents of the 8-bit literal field are added to the contents of the file register designated by f and the sum replaces the original contents of the file register. Subtraction is performed by placing the 2's complement of the number in the literal field. The condition flags and Link register are not affected. File 0 may not be selected by this command.

This command is used when it is desired to add a number other than 1 (in which case the operate class is used) to a file register. Specific cases are where a file is used for a pointer or to update the U register and changes of 2 or greater are required. Another use is to clear out higher order bits from a register. This command can also be used to set a flag bit in a file without resetting the other flag bits.

Examples:

1) Add '2A' to file 3 which contains '31'

2) Subtract '03' from file 5 which contains '54'

3) Set flag bit 6 in file 9 which has flag bit 1 set

Example Number	L	Machine Code	Assembly Language	Flow Chart Notation
1)	'015'	'332A'	AF 3,X'2A'	(f_3)+'2A'→f_3
2)	'105'	'35FD' ①	AF 5,X'FD'	(f_5)-'03'→f_5
3)	'250'	'3940' ②	AF 9,X'40'	(f_9)+'40'→f_A

① 2's complement of '03'

② Hexadecimal equivalent of bit 6 = 1

Affected register states:

Example Number	Register	Before	After
1)	L	'015'	'016'
	file 3	'31'	'5B'
2)	L	'105'	'106'
	file 5	'54'	'51'
3)	L	'250'	'251'
	file 9	'02'	'42'

Execution time - 200 nsecs

2.2.11 TZ Test If Zero

4	f	Literal
15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0

If, for all the 1-bits of the literal field, the corresponding bits of the file register designated by f are 0-bits, the next command is skipped. The condition flags, Link register and the file register are not effected. If the skip is taken, the timing of the command is two clock cycles.

This is a conditional branch type of command designed to test for the following conditions or functions existing in the referenced file register: negative or positive number, odd or even number, interrupt or internal status bits, sense switch bits, condition flags set or not set, teletype input bit set or not set. Since all of the selected bits must be 0, this is a logical AND type function. If a test bit is 0, the corresponding bit in the file does not affect the skip.

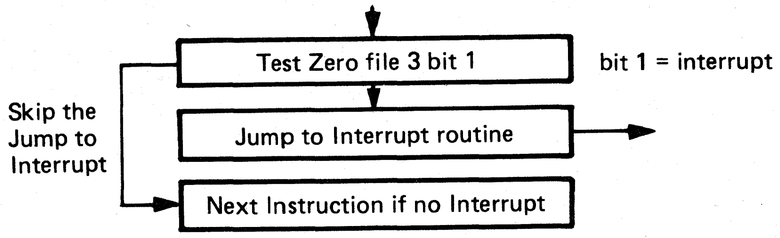
Bit Pattern Examples:

File Register	10001000	
Test Zero Literal	00111000	NO SKIP
File Register	11100111	
Test Zero Literal	00011000	SKIP
File Register	10110000	
Test Zero Literal	01001010	SKIP
File Register	00010000	
Test Zero Literal	00010000	NO SKIP

Since all bits tested must be 0, this command is good for testing for the occurence of any of a number of possibilities, such as testing for the presence of any of 3 interrupt flags.

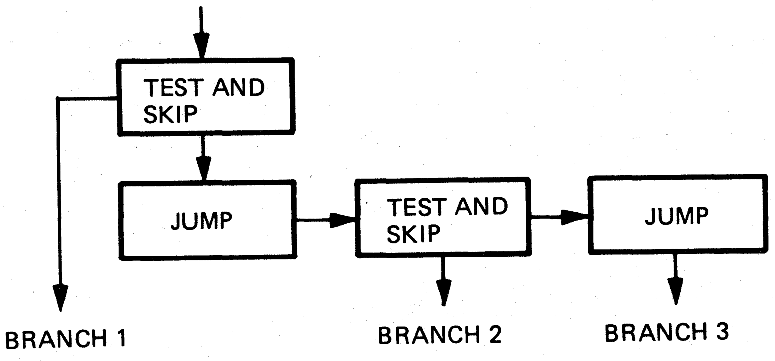
The conditional skip can be used for branching, the skip is followed by a jump command.

Example of Branch:

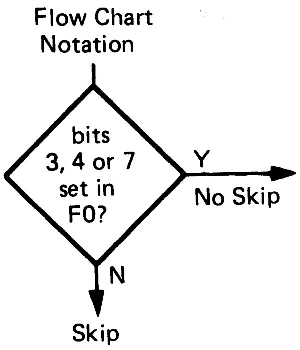

Test Zero file 3 bit 1
bit 1 = interrupt
Skip the Jump to Interrupt
Jump to Interrupt routine
Next Instruction if no Interrupt

A three-way branch can be implemented with two test and skip commands and two jump commands.

Example:



Example: Skip if bits 3, 4, and 7 are not set in file 0.

L	Machine Code	Mnemonic	Flow Chart Notation
'00E'	'4098'	TZ 0,·X'98'	

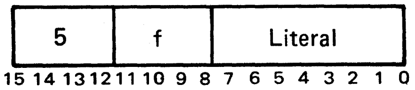
Effected register states:

	Register	Before	After
Case 1	L	'00E'	'010' SKIP
	F0	'43'	'43'
Case 2	L	'00E'	'00F' NO SKIP
	F0	'80'	'80'

Command execution time - 200 nsecs - no skip
400 nsecs - skip

This timing applies to test not zero, and compare, as well.

2.2.12 TN Test If Not Zero



If, for any bit of the literal field which is a 1-bit, the corresponding bit of the file register designated by f is also a 1-bit, the next command is skipped. The condition flags, Link register and file register are not effected. If the skip is taken, the timing of the command is 2 clock cycles.

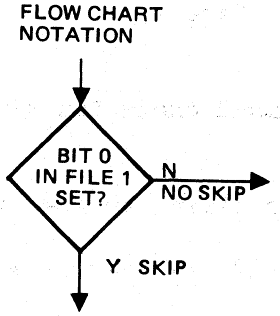
This command differs from the test zero command in two ways. First, it skips on 1's instead of 0's, and it skips on any 1 as opposed to all 0's on the test zero instruction.

If both tests (zero and not zero) were reduced to one bit comparisons, the only variation would be that one command produces the opposite result of the other. If a jump was wanted, the choice would then be, if the tested bit was a 1 or 0.

If multiple bits are tested, the test not zero is the MAX TERM, and test zero is the MIN TERM logic equivalent. Bit pattern examples for test not zero:

File register Test not zero literal	01101100 00110001	SKIP
File register Test not zero literal	01000001 00011010	NO SKIP
File register Test not zero literal	01100110 01101000	SKIP
File register Test not zero literal	11100111 00010000	NO SKIP

Example: Skip if bit 0 in file 1 = 1

L	Machine Code	Mnemonic	Flow Chart Notation
'01C'	'5101'	TN 1,X'01'	 FLOW CHART NOTATION BIT 0 IN FILE 1 SET? N NO SKIP Y SKIP

Effected register states:

	Register	Before	After	
Case 1	L F_1	'01C' '01'	'01E' '01'	SKIP
Case 2	L F_1	'01C' '80'	'01D' '80'	NO SKIP

Command execution time - 200 nsecs - no skip
- 400 nsecs - skip

2.2.13 CP Compare

6	f	Literal
15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0

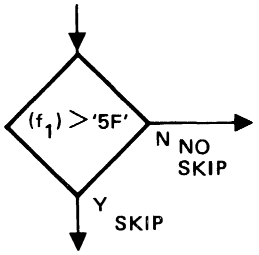
If the sum of the contents of the file register designated by f and the contents of the 8-bit literal is greater than 2^8-1, the next command is skipped. The condition flags, and file register are not effected. If the skip is taken, the timing of the command is 2 clock cycles. The Link stores the carry out of the adder. File 0 may not be selected by this command.

This command is used for looping control, and for data value testing. It is also used to test OP codes in instructions for selection of a particular class of Op codes, such as memory reference, having Op code greater than 5, for example. To test if the content of a file register exceeds a selected number, the 1's complement is placed on the literal part of the compare command.

EXAMPLE: SKIP IF $(f_1) > $'5F'

L	MACHINE CODE	MNEMONIC
'014'	'61A0'	CP 1 X'A0'

FLOW CHART CHART



AFFECTED REGISTER STATES:

	REGISTER	BEFORE	AFTER	
CASE 1	L F_1	'014' '52'	'016' '52'	NO SKIP
CASE 2	L F_1	'014' '66'	'015' '66"	SKIP

COMMAND EXECUTION TIME – 200 NANOSECONDS – NO SKIP
400 NANOSECONDS – SKIP

2.2.14 K Control

7	f	c	*	r

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

This command is used to control special data flow operations, and I/O functions. The prime usage is:

- Enter sense switches from panel to selected file register
- Shift selected file right 4 bit places
- Enter internal status to selected file register
- Set and clear the 3 I/O control flip flops (IOXX)

The prime functions of this command are determined by the value of the c field as follows:

c	Operation	Explanation
0	No operation	
1	Enter Sense Switches	Status of the 4 console sense switches are placed in the 4 high order bits of the file file register designated by f. The 4 low order bits are set to 1 bits. The status can also be placed in the designated destination register.
2	Shift File Right 4	The 4 high order bits of the file register designated by f are placed in 4 low order bits of the file register. The 4 high bits are set to 1 bits. The result can also be transferred to the designated destination register.
3	Unused	
4	Enter Internal Status	The 8 internal status bits are placed in the file register designated by f, and the designated destination register. The internal interrupt flag in file 0 is reset by this command, along with the console interrupt, real-time clock, memory parity, and power fail/restart. Console step is reset upon release of the console switch and spare bits are controlled according to their individual implementation in h hardware.
5	Unused	
6	Unused	
7	Enter Console Switches	The contents of the 8 low order console command switches are ANDed with 8 low order bits of the next command. File register 0 and the destination register 0 must be selected to prevent any modification of the file or register during execution of the control command. The command physically preceding this operation must not cause a read-only memory delay.

c	Operation	Explanation
8	Clear I/O Mode	The I/O control register is cleared. Data from the designated file or input bus ANDed with the designated file can be transferred to the designated file register and register (r).
9-F	Set I/O Modes	The I/O control register is set to equal the 3 low order bits of the c field. Data from the designated file or input bus ANDed with the designated file can be transferred to a designated file register and register (r). For all values of c, except 0,3,5,6,7, source data is placed in the designated files if bit 3=0 and also in the designated register. Destination r=7 is undefined for this command. In other words, the U register is not used.

2.2.15 Examples of Control Commands

C = 1 Enter sense switches into file 1

L	Machine Code	Mnemonic	Flow Chart Notation
'005'	'7110'	K 1,1	(SSW) $\rightarrow f_1$

Affected Register Status:

	Register	Before	After
Case 1	L	'005'	'006'
	file 1	---	'9F'
	Sense SW (Binary)	1001	1001
	File 0 (Bits 2-0)	---	010
Case 2	L	'005'	'006'
	file 1	---	'2F'
	Sense SW (Binary)	0010	0010
	File 0 (Bits 2-0)	---	000

C = 2 Shift file 1 right 4

L	Machine Code	Mnemonic	Flow Chart Notation
'012'	'7120'	K 1,2	F_1 SR4 $\rightarrow F_1$

Affected Register States:

Register	Before	After
L	'012'	'013'
file 1	'E0'	'FE'
file 0 (Bits 2-0)	---	010

C = 4 Enter internal status to file 1

L	Machine Code	Mnemonic	Flow Chart Notation
'1E3'	'7140'	K 1,4	Status ——▶ f_1

Affected Register Status:

Register	Before	After
L	'1E3'	'1E4'
file 1	---	'45'
Status	'45'	'40'
file 0 (Bits 2-0)	---	000

Note: Sense switch 4 can be tested by testing negative condition flag after entering SSW to file 0.

C = 7 Enter console switches into file 5

L	Machine Code	Mnemonic	Flow Chart Notation
'112'	'7070'	K 0,7	f5∧CSW ——▶ f5
'113'	'25FF'	LF 5,X'FF'	

Affected Register Status:

Register	Before	After
L	'112'	'114'
file 5	---	'A5'
Console SW	'A5'	'A5'
file 0 (Bit 2-0)	---	010

This command cannot be executed via the front panel because it requires a dynamic situation, and two separate functions entered on the front panel.

2.2.16 Standard Output Functions

The two output codes COXX and DOXX represent a two byte output sequence, where the first byte is for control, and the second byte is for data. A device select control byte is first placed in the T register (which is also the output bus) and then COXX is set and reset. Following this, a data value is placed in T and DOXX is set and reset.

2.2.17 Standard Input Functions

COXX and DIXX control codes are used for data input routines. A device select control byte is first placed in T, and COXX is set and reset. Following this, DIXX is set, data is input while DIXX is set and then DIXX is reset.

While DIXX is set, data can be entered in two ways:

1. Operate commands involving T get the input bus instead of T as long as IO3X is set. These commands are ADD, OR, COPY, EXCLUSIVE OR, and AND. Any of these can be used to input data while DIXX is set as long as T complement is not selected.

2. The control command with the c field = 8-F causes the input bus to be ANDed with the selected file register as long as IO3X is set. This method allows inputting on the same command that resets DIXX (providing the selected file has first been set to 'FF').

C = 8-F Input/Output Control

When c equals 8-F, the operations are associated with external I/O, and the 3 low order bits of c are placed in the I/O control register. On the same operation, data can be moved from the designated file register or the input bus ANDed with the designated file register as determined by the current contents of the I/O control register, to the designated file or destination register. The data source is specified as follows:

I/O Control Register Mode	Source
0-3	Designated file register
4-7	Input bus ANDed with designated file register

The values 4-7 correspond to the IO3X control flip-flop. This flip-flop must be set in order to transfer data from the input bit to the computer's internal registers. Other than this restriction, the three I/O control register bits can be used in any manner desired at the microprogramming level of the Micro One and as long as standard I/O interface modules are not used.

For purposes of standardization of common interface modules, and implementation of standard I/O software instructions, a convention for I/O codes have been adopted as shown in Table 2-2.

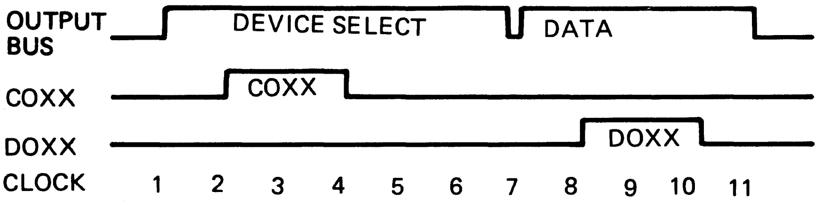
Table 2-2. Standard I/O Control Codes

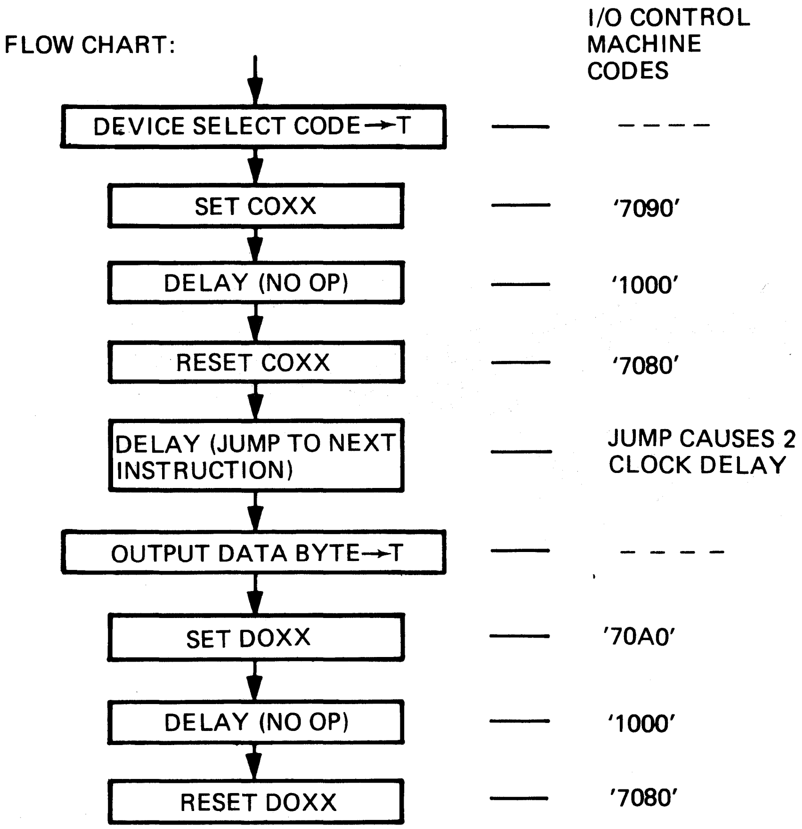
cField (Hex)	I/O Mode	IOXX 3 2 1	Control Activity	
8	0	0 0 0	None	
9	1	0 0 1	Control Output (COXX/)	Output Codes
A	2	0 1 0	Data Output (DOXX/)	
B	3	0 1 1	Space Serial Teletype	
C	4	1 0 0	Concurrent Acknowledge (CACK/)	
D	5	1 0 1	I/O Acknowledge (IACK/)	Input Codes
E	6	1 1 0	Data Input (DIXX/)	
F	7	1 1 1	Spare	

Note that the I/O mode is directly represented as the 3 LSB's of c field

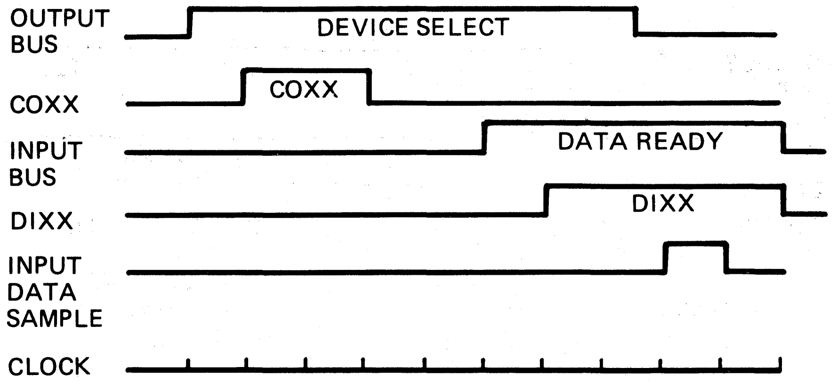
I/O Examples:

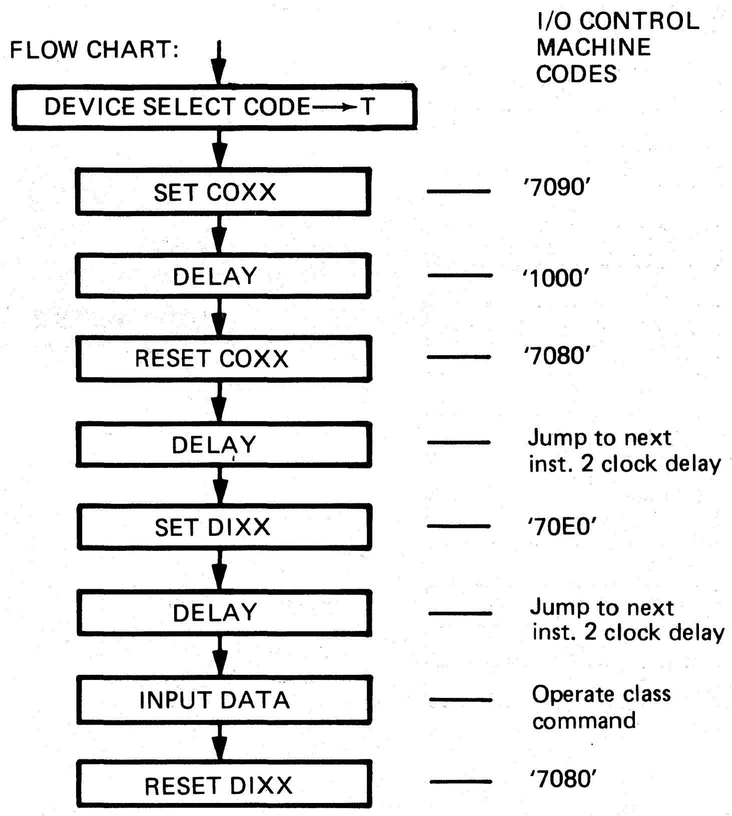
1. Generate following output waveform:





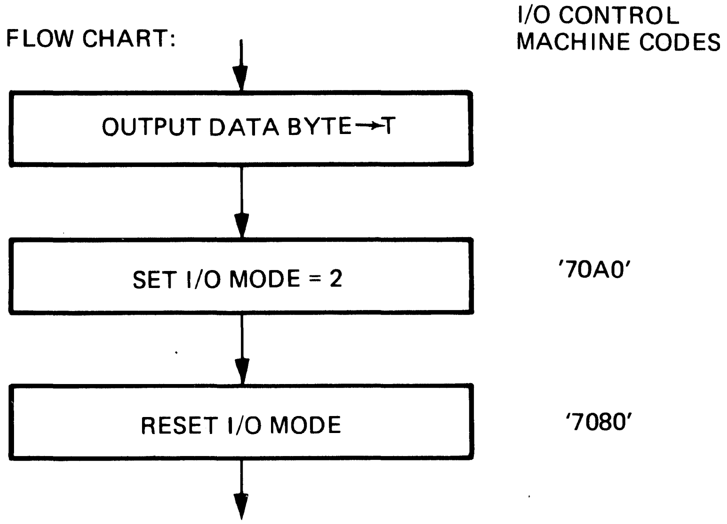
2. Input data according to following waveform:





For a very simple interface having only 3 data registers to set, a single byte sequence will suffice for outputtting data.

3. Output a byte to interface Latch No. 2, where only 3 interfaces latches exist in the system, using the simple interface technique mentioned above.

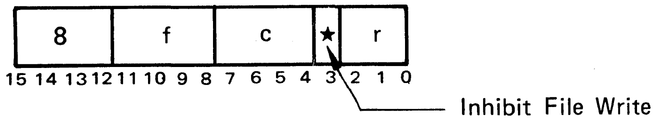


On an input cycle it is necessary to wait at least one clock cycle after generating DIXX to input data. The I/O controls are set in time at the completion of the control command. An input on the next clock would attempt to transfer data before the interface unit has the correct response data ready for input.

c field = B which is I/O mode 3 is used to set the serial teletype mode to SPACE, which ties up the I/O channel.

c field = D which is I/O mode 5 is used to acknowledge interrupts.

2.2.17 A Add



The selected operand is added to the contents of the file register designated by f. The sum is placed in the file register (f), if * is an 0 bit, and in the register designated by r.

The state of the carry out of the high order bit of the adder is placed in Link. File 0 may not be selected by this command. The c field controls selection of the operand, incrementing the result and modification of the condition flags is as follows:

c-bits
7 6 5 4

1 x x x Link Control: The content of LINK is added to the sum. The zero condition flag can be reset but cannot be set, providing a linked zero test over multiple bytes. A linked zero over multiple bytes functions as follows: Assume a 2-byte add is to be performed. Two file registers contain a 16-bit number to be added to another 16-bit number in core memory. The add is performed one byte at a time with the LINK used for carry into the second add. On the first byte addition the condition flags are modified. If the result of the first byte addition is not zero, then of course the entire addition results in a non-zero condition, so that the zero condition flag should not be set on the second byte add even if its result is zero. On the other hand, if the first add produces a zero condition, the second may not, therefore the zero condition flag should be resettable on the second byte add.

The add function can be used to move data from a file to another register by not selecting any input in the c field.

x 1 x x Add One: One is added to the sum.

x x 1 x Select T: The contents of the T register or the input bus are selected as the operand. If the T register is not selected, the operand is zero.

x x x 1 Modifying Condition Flags: The condition flags are updated according to the result.

Eight different examples have been selected to illustrate various c states, data values, and destination registers. Since the L register advances 1 unless it is the destination, its state will not be shown in the affected register state chart. File 1 will be used in all examples.

The various functions selected for each example are shown in Tables 2-3, 2-4, and 2-5.

Table 2-3.

The general form of the examples is --

Add the contents of file 1 to one or more of the following:

Link, 1, T

Destination register choices are

T, F_1, or N

Link is always updated.

Condition flags are updated on selected examples.

Add command uses file 1 for all examples
Table of functions selected for each example.

Table 2-4.

	c Field					Destination		
Example	Add Link	Add 1	Select T	Modify Cond. Flags	Hexadecimal Code for c Field	Selected Register Symbol	Binary Code	Hexadecimal Code
1. Add (file 1) to (T), put result in T and f_1, and update condition flags.	0	0	1	1	3	T, f_1	0001	1
2. Add (file 1) to (T), put result in T, update condition flags.	0	0	1	1	3	T	1001	9
3. Add (file 1) to T, put result in N, update condition flags.	0	0	1	1	3	N	1011	B
4. Add (file 1) to T, +1, put result in f_1 and N.	0	1	1	0	6	N, f_1	0011	3
5. Add (file 1) to (LINK), put result in f_1.	1	0	0	0	8	f_1	0000	0
6. Add one to f_1 and put result in f_1, update C.	0	1	0	1	5	f_1	0000	0
7. Add (f_1) to T and (LINK). Put result in f_1.	1	0	1	0	A	f_1	0000	0
8. Add (file 1) to (T) plus 1. Put result in T, f_1.	0	1	1	0	6	T, f_1	0001	1

Table 2-5.

The coding for the 8 Addition examples is shown below.

Example	Machine Code (Hex)	Assembly Language Mnemonics	Flow Chart Notation
1	8131	AT 1, T, C	$(f_1) + (T) \rightarrow T, f_1, C$
2	8139	AT* 1, T, C	$(f_1) + (T) \rightarrow T, C$
3	813B	AN* 1, T, C	$(f_1) + (T) \rightarrow N, C$
4	8163	AN 1, I, T	$(f_1) + (T) +1 \rightarrow N, f_1$
5	8180	A 1, L	$(f_1) + (L) \rightarrow f_1$
6	8150	A 1, I, C	$(f_1) + 1 \rightarrow f_1$
7	81A0	A 1, L, T	$(f_1) + T + (L) \rightarrow f_1$
8	8161	AT 1, L, T	$(f_1) + (T) + 1 \rightarrow T, f_1$

NOTE

If both Link and 1 are selected as inputs, they are ORed instead of added, thus the effective input is 1 regardless of the value of L.

Table 2-6. Effected Register State Chart

						Conditions		
Example		File	T	Link	N	Zero	Neg	Ovflow
1	Before	'65'	'9B'	....	....		....	
	After	00	00	1	....	1	0	0
2	Before	'65'	'15'	....	....		....	
	After	'65'	'7A'	0	....	0	0	0
3	Before	'65'	'65'	....	....		....	
	After	'65'	'65'	0	'CA'	0	1	1
4	Before	'65'	'00'	....	....		....	
'	After	'66'	'00'	0	'66'		....	
5	Before	'00'	....	1	....		....	
	After	'01'	....	0	....		....	
6	Before	'FF'	....	....	....		....	
	After	'00'	....	1	....	1	0	0
7	Before	'00'	'00'	1	....		....	
	After	'01'	'00'	0	....		....	
8	Before	'01'	'01'	....	....		....	
	After	'03'	'03'	0	....		....	

Table 2-6 shows the results for the eight ADDITION EXAMPLES:

Command execution time - 200 nsecs

2.2.18 S SUBTRACT

The complement of the selected operand plus one is added to the contents of the file register designated by f. The difference is placed in the file register (f) if * is a 0 bit, and in the register designated by r. The result is a 2's complement subtraction. The state of the carry out of the high order bit of the adder is placed in Link. File 0 may not be selected by this command. The c field controls selection of the operand, incrementing the result, and modification of the condition flags as follows:

c-bits
7 6 5 4 Operation

1 x x x Link control: The content of LINK is added to the sum. Selection of the LINK inhibits the automatic addition of one. The zero condition flag cannot be set, providing a linked zero test over multiple bytes. Refer to the add description for details on linked zero test.

x 1 x x Inhibit add one: If link control is not selected, one is automatically added to the result to produce a 2's complement subtraction. This control bit inhibits this addition, providing a 1's complement subtraction.

x x 1 x Select T: This complement of the contents of the T register are selected as the operand to the adder. If not selected, the operand consists of a 1-bit in each bit position.

x x x 1 Modify Condition Flags: The condition flags are updated according to the result.

Affected: F, LINK, Condition Flags, r.

If the input bus is enabled (I03X), this command will yield an unpredictable result because the complement of the input bus is not available.

Examples:

1. Subtract zero from file 1.

$(f_1) - 0 \longrightarrow f_1$

Machine Code	Mnemonic
9100	S 1

Effected register states:

Register	Before	After
Link	--	1
file 1	'00'	'00'

Even though 0 is subtracted from 0, since 2's complement adding is used there is a carry of 1 all through the adder to the Link.

2. Subtract T, 1 from file 1
Destination T Update condition flags

Machine Code	Mnemonic	Flow Chart Notation
'9179'	ST* 1,D,T,C	$(f_1) - T\text{-}1 \longrightarrow T, C$

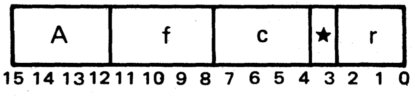
Effected register states:

Register	Before	After
f_1	'31'	'31'
T	'31'	'FF' ← 2's complement for -1
L	--	0
C	--	0 1 0

Zero Neg Overflow

Command execution time -- 200 nanoseconds.

2.2.19 R READ MEMORY W WRITE MEMORY



The primary function of this command is to initiate a core memory cycle in which one byte is transferred between the T register and core memory. The address in core is determined by the contents of the M and N registers. File 0 may not be selected by this command.

The lower two bits of the c field determine whether the memory operation is read or write and whether the operation is a full or half cycle.

The c-bits control the type of memory operation as follows:

c-bits 7 6 5 4	Memory Access Operation
x x 1 x	Half Cycle: If this bit is a 1-bit, a half cycle memory operation is performed; otherwise a full cycle operation is selected.
x x x 1	Write: If this bit is a 1-bit, a write memory operation is performed; otherwise a read operation is selected.

A full cycle takes 5 clock times.

A half cycle takes 3 clock times.

A full cycle read leaves the data in core unchanged.

A full cycle write causes the old data to be cleared so the new value is unaffected by the old.

A half cycle read leaves all ones in the core location.

A half cycle write ANDS the data to be written with the data already in core.

If a half cycle write into a particular memory cell was preceded by a half cycle read, the data value gets stored without modification since it is ANDed with 1's, left from the previous half cycle read.

A secondary function of this command is to simultaneously move data between registers while initiating the memory cycle.

The contents of the file register designated by f is unaltered, incremented, or decremented as controlled by the c field. The result is placed in the file register (f) if * is a 0-bit, and in the register designated by r. At the same time, a read (R) or write (W) memory operation is initiated as controlled by bit 4. If the operation is a memory read, the T register is cleared and the accessed data is set into the T register after two clock cycle times. Data to be written into memory must be placed in the T register during or before the write memory command, if the operation is a half cycle write, and by the first clock cycle time after the write memory command on a full cycle write. The condition flags and LINK are not affected. Execution of the memory command is delayed if the memory is in a busy condition from a previous R or W command or DMA operation.

The bits of the c field control the transfer of data from the file register as follows:

c-bits
7 6 5 4   Operation

0 0 x x   Transfer: The contents of the file register are transferred unaltered.

0 1 x x   Decrement: The contents of the file register minus one are routed as specified. If the M register is selected as the destination and the content of LINK is a 1-bit, the contents of the file register are transferred without being decremented. This provides a decrement with link control when M is the destination.

1 0 x x   Add Link: The content of LINK is added to the contents of the file register, and the sum is transferred as specified.

1 1 x x   Increment: The contents of the file register plus one are transferred as specified.

This data transfer feature permits setting up one of the registers directly involved with the memory access (M, N, or T) at the same time the memory cycle is initiated. There are some timing restrictions pertaining to modification of M, N, or T registers during a memory cycle. Some of the functions have logic interlocks to prevent errors, and some do not. These restrictions must be carefully considered with respect to data errors, and unexpected program time delays. The restrictions must be carefully considered with respect to data errors, and unexpected program time delays. The restrictions are as follows:

1. Attempting to change M or N while a memory cycle is in progress stops the computer clock until the memory cycle is over. No data errors result. Either M or N can be changed by the command initiating the memory cycle without causing delay.

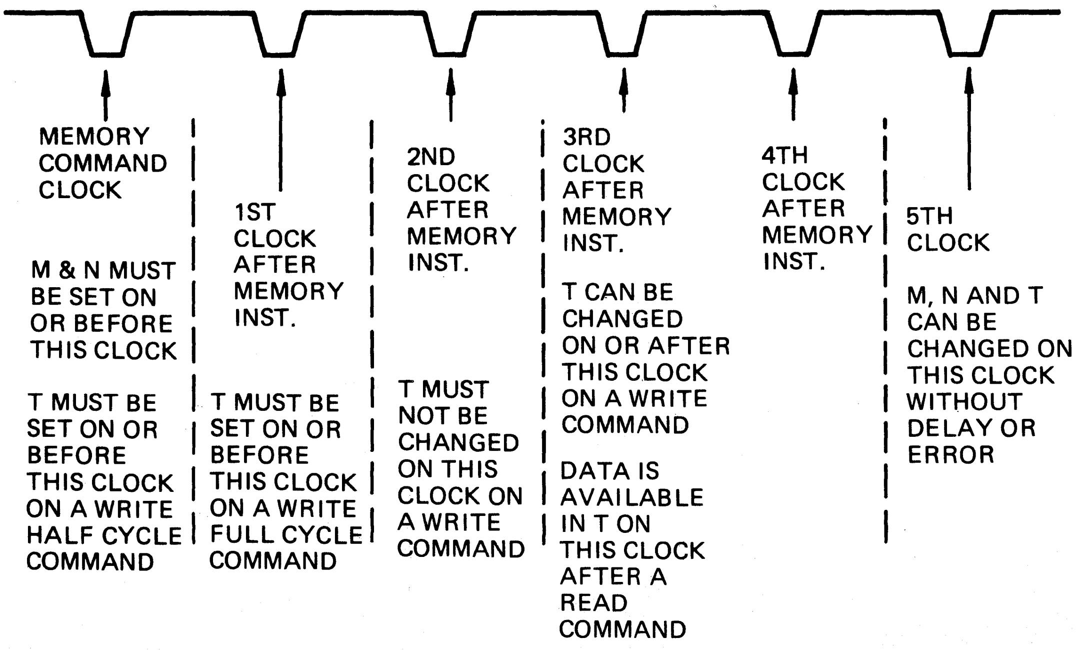
2. Accessing T during a read cycle causes the clock to stop until the new data value from core is correctly in T. This causes delay but no data error.

3. Changing T during a write cycle will cause a delay if it occurs during WTXX/ and it may cause a data error if it occurs on the clock immediately preceeding WTXX/.

The memory access restrictions are specifically defined in the following chart:

	Full Cycle Read	Full Cycle Write	Half Cycle Read	Half Cycle Write
Delay from changing M and N	Up to 4 clocks	Up to 4 clocks	Up to 2 clocks	Up to 2 clocks
Delay due to T access	Up to 2 clocks	0 clocks	Up to 2 clocks	0 clocks
Data in T available (on Read)	3rd clock after memory command		2nd clock after memory command	
T must be loaded by (on Write)		1st clock after memory cycle command		Memory Cycle Command
T must stay loaded until (on Write)		2 clocks after memory command		

Timing Diagram for Memory Accesses:



MEMORY COMMAND CLOCK

M & N MUST BE SET ON OR BEFORE THIS CLOCK

T MUST BE SET ON OR BEFORE THIS CLOCK ON A WRITE HALF CYCLE COMMAND

1ST CLOCK AFTER MEMORY INST.

T MUST BE SET ON OR BEFORE THIS CLOCK ON A WRITE FULL CYCLE COMMAND

2ND CLOCK AFTER MEMORY INST.

T MUST NOT BE CHANGED ON THIS CLOCK ON A WRITE COMMAND

3RD CLOCK AFTER MEMORY INST.

T CAN BE CHANGED ON OR AFTER THIS CLOCK ON A WRITE COMMAND

DATA IS AVAILABLE IN T ON THIS CLOCK AFTER A READ COMMAND

4TH CLOCK AFTER MEMORY INST.

5TH CLOCK

M, N AND T CAN BE CHANGED ON THIS CLOCK WITHOUT DELAY OR ERROR

Examples:

Example	Machine Code				Mnemonics		c Field Binary Functions and Codes for Memory Commands				c Field Hex. Code	General Description
	op	file	c	dest								
1) Full cycle write (file 1) + 1 → N, f_1	A	1	D	3	WN	1, I	Increment		Full cycle write		D	Full cycle write memory is initiated and N register is updated as well as f_1.
							1	1	0	1		
2) Half cycle read (file 2) → M, f_2	A	2	2	2	RM	2, H	Transfer		Half cycle read		2	Half cycle read memory is initiated while M register is updated directly from f_2.
							0	0	1	0		
3) Half cycle write (file 2) + (Link) → M, f_2	A	2	B	2	WM	2, L, H	Add Link		Half cycle write		B	Half cycle write memory is initiated while file 2 and M are updated by adding (LINK).
							1	0	1	1		
4) Full cycle write (file 3) → T, f_3	A	3	1	1	WT	3	Transfer		Full cycle write		1	Full cycle write memory is initiated, T is updated from f_3 on the same command.
							0	0	0	1		
5) Half cycle read (f_1) − 1 → N followed (f_3) + (T) → T, f_3					Inhibit file write ↓		Decrement		Half cycle read			Half cycle read memory is initiated, followed by T register access on the next instruction. This will cause a program delay until the third clock.
	A	1	6	B	RN*	1, D, H	0	1	1	0	6	
	8	3	2	1	AT	3, T	—	—	—	—	—	
6) Half cycle write followed by loading T (f_3) → T, f_3							Transfer		Half cycle write			Half cycle write memory is initiated, followed by loading T on next instruction. No time delay occurs, but data written into memory may be incorrect.
	A	0	3	0	W	0, H	0	0	1	1	3	
	8	3	0	1	AT	3	—	—	—	—	—	
7) Full cycle read, decrement (file 1) and transfer to M (f_1) − 1 → M, f_1							Decrement		Full cycle read			A full cycle read is initiated (f_1) is decremented and transferred to M. If (LINK) = 1 the contents of the file are transferred without being decremented.
	A	1	4	2	RM	1, D	0	1	0	0	4	

2.2.20 C COPY

B	f	c	★	r
15 14 13 12	11 10 9 8	7 6 5 4	3	2 1 0

The selected operand is placed in the file register designated by f, if * is a 0-bit, and in the register designated by r. The LINK is not affected. The c filed controls selection of the operand, incrementing the operand, and modification of condition flags as follows:

c-bits
7 6 5 4 Operation

1 x x x Link Control: The content of LINK is added to the sum. The zero condition flag can be reset but cannot be set, providing a linked zero test over multiple bytes.

x 1 x x Add One: One is added to the sum.

x x 1 x Select T: The contents of the T register or input bus are selected as the operand. If the T register is not selected, the operand is zero.

x x x 1 Modify condition flags: The condition flags are updated according to the result.

Affected: F, Condition Flags, r.

This command is used to transfer T to a selected file register, with the option of incrementing or adding LINK while transferring. It is also used for inputting data, because when the input control flip flop (I03X) is set during an input mode, operate commands selecting T get the input bus instead.

The command can be used to test the condition of T by selecting f_0 as the file register (which is unaffected) and setting the modify condition flag in the c field.

The command can also be used to clear one file and another selected register by not selecting any input in the c field.

Command Execution Time -- 200 nanoseconds.

File register 1 is used for all examples except setting condition flag example.
Examples of Copy Command:

2.2.21 O OR

C	f	c	★	r
15 14 13 12	11 10 9 8	7 6 5 4	3	2 1 0

Examples	Machine Code op file c dest	c field for Copy Commands Link	 Add 1	 Select T	 Mod. Cond. Flags	 Hex. Code	Destination for Copy Commands Selected Registers	 Binary Code	 Hex. Code	Mnemonics	General Discussion
$(T) \rightarrow f_1$	B 1 2 0	0	0	1	0	2	f_1	0000	0	C 1, T	(T) is transferred, unaltered to file 1.
$(T) + 1 \rightarrow f_1$, N	B 1 6 3	0	1	1	0	6	f_1, N	0011	3	CN 1, I, T	(T) is incremented and transferred to file 1, and to the N register.
$(T) + (LINK) \rightarrow f_1$	B 1 A 0	1	0	1	0	A	f_1	0000	0	C 1, T, L	(T) is added to (LINK) and transferred to f_1.
$0 \rightarrow f_1$, N	B 1 0 3	0	0	0	0	0	f_1, N	0011	3	CN 1	File 1 and N registers are cleared because no input is selected.
$(T) \rightarrow f_0$, C Set Condition Flags	B 1 3 0	0	0	1	1	3	f_0	0000	0	C 0, T, C	Condition flags are set according to the state of (T). File 0 can't be loaded by this instruction so is unchanged.
Set DIXX	7 0 E 0									K 0, X'E'	The input flip flop is set by the DIXX command, so the copy T command transfers the Input bus to file 1 and to T.
Delay	1 0 0 0									LZ X'00'	
$(T) \rightarrow f_1$, T	B 1 2 1	0	0	1	0	2	f_1, T	0001	1	CT 1, T	
Reset DIXX	7 0 8 0									K 0, 8	

The selected operand is logically inclusive-ORed on a bit-for-bit basis with the contents of the file register designated by f and the result is placed in the file register, if * is a 0-bit, and in the register designated by **r**. The LINK is not affected. The c field controls selection of the operand and modification of the condition flags as shown below:

c-bits 7 6 5 4	Operation
1 x x x	Link control: The zero condition flag can be reset but cannot be set, providing a linked zero test over multiple bytes. See the description of the add command for a detailed description of linked zero test.
x 1 x x	Select complement T: The complement of the contents of the T register is selected as the operand. If the T register is also selected, the effective operand contains a 1-bit in each bit position.
x x 1 x	Select T: The contents of the T register or Input bus are selected as the operand. If neither the T register nor the complement of the T register is selected, the operand is zero.
x x x 1	Modify Condition Flags: The condition flags are updated according to the result.

Affected: F, Condition Flags, r.

If both complement T and T are selected, the operand is all 1's. If the input bit is enabled (I03X), complement T must not be selected.

This command is used for the general function of logical ORing as needed in a microprogram. It also has the following specific applications: Setting flag bits without disturbing other bits (with the OR function it doesn't matter if the flag is already set since there is no carry); moving data from a file to another register by not selecting any operand; setting all 1's in a file register and/or one other selected register by selecting both T and T complement as operands; combining two numbers into one byte, such as for assembling hexadecimal digits into multiple digit numbers after the digits have been input to the computer as a string.

Bit pattern example of OR function:

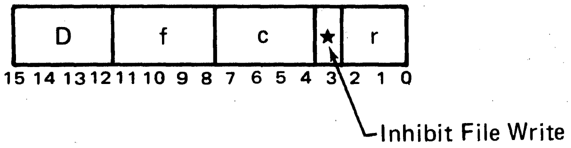
	Binary	Hexadecimal
file 1	01101000	'68'
T	00110100	'34'
Result	01111100	'7C'

Command Execution Time -- 200 nanoseconds.

File register 1 is used for all examples.
Examples of OR command:

Flow Chart Notation	Machine Code op file c dest	c field for OR commands					Destination for OR command results			Mnemonics	General Discussion
		Link	Select Comp. T	Select T	Mod. Cond. Flags	Hex. Code	Selected Registers	Binary Code	Hex. Code		
(f_1) V $(T) \rightarrow T$	C 1 2 9	0	0	1	0	2	1	1001	9	OT* 1, T	OR (file 1) with (T), inhibit file write put result in T.
(f_1) V $0 \rightarrow N, f_1$	C 1 0 3	0	0	0	0	0	N, f_1	0011	3	ON 1	Move (file 1) to N by ORing with 0 and putting result in N.
(f_1) V $(T) \rightarrow f_1$	C 1 2 0	0	0	1	0	2	f_1	0000	0	O 1, T	OR (file 1) with (T) and put result in file 1.
(f_1) V (T), $(\overline{T}) \rightarrow N$	C 1 6 B	0	1	1	0	6	N	1011	B	ON* 1,T,F	Set N = FF (all ones) by ORing (f_1) with T, $\overline{T}$ and putting result in N.
(f_1) V (T) $(\overline{T}) \rightarrow f_1$	C 1 6 0	0	1	1	0	6	f_1	0000	0	O 1,T,F	Set f_1 = FF by ORing f_1 with T, $\overline{T}$ and putting result in f_1.
(f_1) V $(T) \rightarrow$ Link, C	C 1 B 8	1	0	1	1	B	none	1000	8	O* 1,T,L,C	Perform conditional test on (f_1) V (T) without changing f_1 or T Select L to perform linked zero test with a previous command.

2.2.22 X EXCLUSIVE OR



The selected operand is logically exclusive-ORed on a bit for bit basis with the contents of the file register designated by f and the result is placed in the file register, if * is a 0-bit, and in the register designated by r. The LINK is not affected. The c field controls selection of the operand and the modification of the condition flags as shown below:

c-bits 7 6 5 4	Operation
1 x x x	Link Control: The zero condition flags can be reset but cannot be set, providing a linked zero test over multiple bytes. See the description of the Add command for a detailed description of linked zero test.
x 1 x x	Select Comlement T: The complement of the contents of the T register is selected as the operand. If the T register is also selected, the effective operand contains a 1-bit in each bit position.
x x 1 x	Select T: The contents of the T register or input bus are selected as the operand. If neither the T register nor the complement of the T register is selected, the operand is zero.
x x x 1	Modify Condition Flags: The condition flags are updated according to the result.

Affected: F, Condition Flags, r.

If both complement T and T are selected, this command produces the one's complement of the value in the file register. If the input bus is enabled (IO3X), complement T must not be selected.

This command is used for the following functions: general purpose exclusive OR; data comparison, ones complementing; and flipping selected bits such as controls and status flags.

Bit pattern example of exclusive OR.

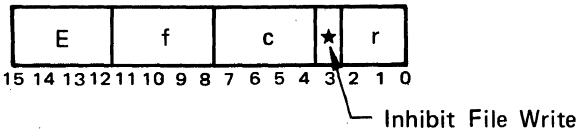
	Binary	Hexadecimal
file 1	01101100	'6C'
T	00011010	'1A'
Result	01110110	'76'

Command execution time -- 200 nanoseconds.

File register 1 is used for all examples.
Examples of Exclusive OR command:

	Machine Code	c field for OR commands					Destination for Exclusive OR command results				
Example Flow Chart Notation	op file c dest	Link	Select Comp. T	Select T	Mod. Cond. Flags	Hex. Code	Selected Registers	Binary Code	Hex. Code	Mnemonics	General Discussion
$(f_1) \veebar (T) \rightarrow T$	D 1 2 9	0	0	1	0	2	T	1001	9	XT* 1,T	Exclusive OR (file 1) with (T) inhibit file write, put result in T.
$(f_1) \veebar 0 \rightarrow N, F_1$	D 1 0 3	0	0	0	0	0	N, f_1	0011	3	XN 1	Move (file 1) to N by exclusive ORing with 0 (same result as OR), put result in N.
$(f_1) \veebar (T) \rightarrow f_1$	D 1 2 0	0	0	1	0	2	f_1	0000	0	X 1,T	Exclusive OR (file 1) with (T) and put result in file 1.
$(f_1) \veebar (T), (\overline{T}) \rightarrow T$	D 1 6 B	0	1	1	0	6	N	1001	9	XT* 1,T,F	Produce ones complement of (f_1) and place result in T.
$f_1 \veebar (T), (\overline{T}) \rightarrow f_1$	D 1 6 0	0	1	1	0	6	f_1	0000	0	X 1,T,F	Produce ones complement of (f_1) and put it back into f_1.
$(f_1) \veebar (T) \rightarrow$ Link, C	D 1 B 8	1	0	1	1	B	none	1000	8	X* 1,T,L,C	Perform conditional test and linked zero test on $(f_1) \veebar (T)$ without changing (f_1) or (T).

2.2.23 N AND



The selected operand is logically ANDed on a bit-for-bit basis with the contents of the file register designated by f and the result is placed in the file register, if * is a 0-bit, and in the register designated by r. The LINK is not affected. The c field controls selection of the operand and modification of the condition flags as shown below:

c-bits
7 6 5 4 Operation

1 x x x Link control: The zero condition flag can be reset but cannot be set, providing a linked zero test over multiple bytes. See the description of the add command for a detailed description of a linked zero test.

x 1 x x Select complement T: The complement of the contents of the T register is selected as the operand. If the T register is also selected, the effective operand contains a 1-bit in each bit position.

x x 1 x Select T: The contents of the T register or input bus are selected as the operand. If neither the T register nor the complement of the T register is selected, the operand is zero.

x x x 1 Modify condition flags: The condition flags are modified by execution of the command. Updated according to the result.

Affected: F, Condition Flags, r.

If both complement T and T are selected and AND command moves the data, unchanged from the selected file register to the designated destination register. If the input bus is enabled (IO3X), complement T must not be selected.

The AND command is used for the following functions: General purpose ANDing of files and T; resetting selected flag or status bits, without disturbing other flags; and marking out parts of a byte.

Bit pattern examples of the AND function.

	Binary	Hexadecimal
file 1	01101011	'6B'
T	10101101	'AD'
Result	00101001	'29'

Command execution time — 200 nanoseconds.

File register 1 is used for all examples.
Examples of AND Command:

Example	Machine Code f d i e o l s p e c t	c field for And commands					Destination for And command results				
Flow Chart Notation		Link	Select Comp. T	Select T	Mod. Cond. Flags	Hex. Code	Selected Registers	Binary Code	Hex. Code	Mnemonics	General Discussion
$(f_1) \wedge (T) \longrightarrow f_1$	E 1 2 0	0	0	1	0	2	f_1	0000	0	N 1,T	(f_1) is anded with (T). The result is put into f_1.
$(f_1) \wedge 0 \longrightarrow N, f_1$	E 1 0 3	0	0	0	0	0	N, f_1	0011	3	NN 1	(f_1) is anded with 0. The result (which is 0) is put into N, and f_1.
$(f_1) \wedge (T) \longrightarrow T$	E 1 2 9	0	0	1	0	2	T	1001	9	NT* 1,T	(f_1) is anded with (T). The result is put in T and inhibited from f_1.
$(f_1) \wedge (T), (\overline{T}) \longrightarrow N$	E 1 6 B	0	1	1	0	6	N	1011	B	NN* 1,T,F	(f_1) is anded with $(\overline{T})$ which is same as anding with FF (all ones). Result is put in N and inhibited from f_1.
$(f_1) \wedge (\overline{T}) \longrightarrow f_1$	E 1 4 0	0	1	0	0	4	f_1	0000	0	N 1,F	(f_1) is anded with $(\overline{T})$. The result is put into f_1.
$(f_1) \wedge (T) \longrightarrow$ Link, C	E 1 B 8	1	0	1	1	B	none	1000	8	N 1,T,L,C	(f_1) is anded with $(\overline{T})$. The result is not put in any register. Only the condition flags are set. Use of link results in multi byte zero test.

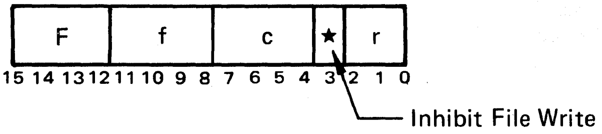
	Binary	Hexadecimal
file 1	01000010	'42'
T	10111111	'BF'
Result	00000010	'02'

reset a flag

file 1	10100101	'A5'
T	11010011	'D3'
(Select T complement)	(00101100)	('2C')
Result	00100100	'24'
file 1	10100101	'A5'
T, T complement	11111111	'FF'
Result	10100101	'A5'

Command Execution Time -- 200 nanoseconds.

2.2.24 H SHIFT



The contents of the file register designated by f is shifted left or right one bit position and placed in the file register, if * is a 0-bit, and in the register designated by r. The high order or low order bit which is shifted off is placed in LINK and in the overflow flag if the modify condition flag is selected. The c field controls the direction of shift, entry of an end bit, and modification of the condition flags as follows:

c-bit
7 6 5 4 Operation

1 x x x Link control: The content of the LINK is inserted into the vacated low order or high order bit position. The zero condition flag can be reset but cannot be set, providing a linked zero test over multiple bytes. See the description of the add command for a detailed description of the linked zero test.

x 1 x x Insert 1: A 1-bit is unconditionally inserted into the vacated low order or high order bit position; otherwise a 0-bit is inserted unless the contents of LINK is selected.

x x 1 x Shift right: if bit 5 is a 1-bit, the operation is a right shift; otherwise a left shift is performed.

c-bit 7 6 5 4	Operation
x x x 1	Modify condition flags: The zero and negative flags are updated according to the result. The content of the bit shifted out is placed in the overflow flag.

Affected: F, LINK, Condition Flags, r.

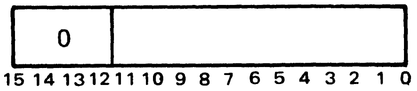
This command provides great flexibility for various shifting functions mechanized by microprogramming. These are as follows:

- Left or right shifting;
- End around carry or no end around carry;
- Arithmetic or logical shifts;
- Multiple byte shift register implementations in either file registers or core memory;
- Pattern rotations by successive shifting of 8 files, one bit at a time, and assembling into a 9th file;
- Set or reset link bit by shifting with no destination register.

Bit pattern examples of shift command.
All examples are for shift (f_1) and put result back in f_1.

Instruction	Sequence Number	File 1 Binary	Link	File 1 Hexadecimal	Condition Flags
Shift Right	before	01101001	0	'69'	----
	after	00110100	1	'34'	----
Shift Left	before	01101001	1	'69'	----
	after	11010010	0	'D2'	----
Shift Right Enter Link	before	00111000	1	'38'	----
	after	10011100	0	'9C'	----
Shift Left Enter 1	before	10001010	0	'8A'	----
	after	00010101	1	'15'	----
Shift Left Modify Condition Flag	before	11001011	0	'CB'	----
	after	10010110	1	'96'	011
Shift Right Modify Condition Flag	before	00000001	0	'01'	----
	after	00000000	1	'00'	101

2.2.25 E EXECUTE



The 8-bit contents of the U register are ORed with the 8 high order bits of the execute command to form an effective command. This provides a means of partially modifying the contents of a read only storage location. The ORing is performed before the output of the read only storage is gated into the R register. The meaning of bits present in positions 0-11 is dependent upon the desired effective operation code after the modification. Due to the look-ahead feature of the read-only memory, the new contents of the U register are not available until after one machine cycle following the transfer of data to it.

The execute command provides a means for program modification of a command. This capability is used for many different functions, three of which are as follows:

- Indexing of file registers in a program loop.
- Having a general purpose instruction which may take on different specific functions, such as load a register, add to the register, AND with the register, etc., depending on program variables.
- Selection of alternate file registers depending on program variables.

Sometimes a combination of two of the above is used.

The U register can be set with the load U command, or by being designated as the destination register of an operate class command, such as Add, Copy, etc.

For the file register indexing, a separate file register is designated as an index register. It is loaded with an initial value, then incremented, with the result being put in U each time through the loop, until the loop is exited.

Examples of execute commands:

U register	'84'	
Execute Command	'0021' ←	This command is stored in ROM ET 0, 2
Effective Command	'8421'	$\begin{cases} (f_4) + (T) \longrightarrow f_4, T \\ AT \qquad 4, T \end{cases}$

ınstruction codes for bit pattern examples of shift command
These examples are the same except for additional Destination Registers

		Machine Code	c field					Destination for Shift Command results				
Example	Flow Chart Notation	f d i e o l s p e c t	Insert Link	Insert 1	Shift Right	Mod. Cond. Flags	Hex. Code	Selected Registers	Binary Code	Hex. Code	Mnemonics	General Discussion
Shift right result to f_1, T.	$(f_1)@R \longrightarrow f_1$, T	F 1 2 1	0	0	1	0	2	f_1, T	0001	1	HT 1,R	(file 1) is shifted right one bit, link, or 1 are not inserted. The result is put in T and f_1.
Shift left result to f_1.	$(f_1)@L \longrightarrow F_1$	F 1 0 0	0	0	0	0	0	f_1	0000	0	H 1	(file 1) is shifted left one bit, link or 1 are not inserted. The result is put in f_1.
Shift right insert link result to f_1, N.	$(f_1)@_R$+LK $\longrightarrow f_1$, N	F 1 A 3	1	0	1	0	A	f_1, N	0011	3	HN 1,R,1	(file 1) is shifted right one bit, (Link) is inserted in vacated left hand bit. Result is put in f_1 and N.
Shift left insert 1 result to f_1, M.	$(f_1)@_L$+1 $\longrightarrow f_1$, M	F 1 4 2	0	1	0	0	4	f_1, M	0010	2	HM 1, I	(file 1) is shifted left 1 is inserted into the vacated right hand bit. Result is put in f_1 and M.
Shift left modify cond. flag. Result to f_1.	$(f_1)@_L \longrightarrow f_1$, C	F 1 1 0	0	0	0	1	1	f_1	0000	0	H 1,C	(file 1) is shifted left. The result is put into file 1. Condition flags are modified.
Shift right Modify cond. flag. Result to f_1.	$(f_1)@_R \longrightarrow F_1$, C	F 1 3 0	0	0	1	1	3	f_1	0000	0	H 1,R,C	(file 1) is shifted right. The result is put into file 1. Condition flags are modified.

Incrementing the U register value leaves the command the same, but changes the file register number to 5. If this continued to file F, the next increment would change the command to a subtract.

U Register	'F1"	
Executive Command	'0020'	This command is stored in ROM E 0,2
Effective command	'F120'	Shift Right file 1 H 1,R

The meaning of the c field of the lower two hexadecimal digits in the execute command changes with the OP code value in the U register. Therefore the c field is left as a digit in the MNEMONIC for the execute command.

Commands can also be modified by the U register by using the operate commands with a 7 in the destination register. This method is advantageous if there are two variable functions to be done in one loop, with one U register setting. For example, a program may be indexing through a set of files where it is necessary to add to a file, and shift the same file in the same program loop. This could be mechanized as follows:

$(f_F) + 1 \longrightarrow U, f_F$

--- NOP

$(f_0) + (T) \longrightarrow f_0$, Destination = 7 (OR U with command)

(F_0) @ $R \longrightarrow F_0$, Destination = 7

The coding for this is:

	Machine Code	Mnemonic	
	'8F46'	AU F, 1	
another command	----	---	
	'8027'	AS 0, T	Add to file 0
	'F027'	HS 0, R	Shift file 0

Assume U = '04' after the first command.

The effective commands following are:

'8427'	Add to file 4
'F427'	Shift file 4 right

This method of command modification has the limitation of no destination register since the destination register code position is tied up selecting U as a modifier to the command. The execute command does not have this restriction.

COMMAND REFERENCE TABLE

Command	Mnemonic	Operation Code		Comments
Load T	LT	11/19 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	
Load M	LM	12 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	
Load N	LN	13 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	
Load U	LU	16 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	
Load Zero	LZ	10 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	
Load Seven	LS	17 15 14 13 12 11 10 9 8	Literal 7 6 5 4 3 2 1 0	

1	7	0	0	No Op	
1	7	0	1	Enable Serial TTY	
1	7	0	2	Reset T_8	
1	F	0	2	Set T_8	
1	7	0	4	Disable	External Interrupts
1	7	0	8	Enable	
1	7	1	0	Disable	Real Time Clock
1	7	2	0	Enable	
1	7	4	0	Load Protect Bit	
1	7	8	0	Halt	

Mnemonic					
Command		Operation Code			Comments
		15 14 13 12	11 10 9 8	7 6 5 4 3 2 1 0	
JUMP	JP	14		Literal	000-OFF
		15		Literal	100-1FF
		1C		Literal	200-2FF
		1D		Literal	300-3FF
Load File	LF	2	f	Literal	
Add To File	AF	3	f	Literal	
Test Zero	TZ	4	f	Literal	
Test Not Zero	TN	5	f	Literal	
Compare	CP	6	f	Literal	

Mnemonic		Operation Code	Comments	Operand Field
Command				

Control K

7	f	c	★	r
15 14 13 12	11 10 9 8	7 6 5 4	3	2 1 0

0	No Op
1	Enter Sense SW
2	Shift Right 4
4	Enter Internal Status
7	Enter Console SW
8	Clear I/O
9	Set COXX (in MICRO ONE/20)
A	Set DOXX (in MICRO ONE/20)
B	Space Serial TTY
C	Set CACK (in MICRO ONE/20)
D	Set IACK (in MICRO)
E	Set DIXX (in MICRO)
F	Spare

Add Ar★

8	f	c	★	r
15 14 13 12	11 10 9 8	7 6 5 4	3	2 1 0

1	x	x	x	Link	L
x	1	x	x	Add 1	I
x	x	1	x	Select T	T
x	x	x	1	Modify Condition Flags	C

Subtract Sr★

9	f	c	★	r
15 14 13 12	11 10 9 8	7 6 5 4	3	2 1 0

1	x	x	x	Link	L
x	1	x	x	Decrement	D
x	x	1	x	Select T	T
x	x	x	1	Modify Condition Flags	C

Command	Mnemonic	Operation Code	Comments	Operand Field
Memory	Wr★ Rr★	A | f | c | ★ | r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	Decrement	D
		1 1 x x	Increment	I
		x x 1 x	Half Cycle Operation	H
		x x x 1	Write Operation (supplied by OP Code)	
Copy	Cr★	B | f | c | ★ | r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	Add 1	I
		x x 1 x	Select T	T
		x x x 1	Modify Condition Flags	C
OR	Or★	C | f | c | ★ | r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	$\overline{T}$	F
		x x 1 x	T	T
		x x x 1	Modify Condition Flags	C
Exclusive OR	Xr★	D | f | c | ★ | r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	$\overline{T}$	F
		x x 1 x	T	T
		x x x 1	Modify Condition Flags	C

Command	Mnemonic	Operation Code	Comments	Operand Field
AND	Nr★	E f c ★ r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	T	F
		x x 1 x	T	T
		x x x 1	Modify Condition Flags	C
Shift	Hr★	F f c ★ r		
		15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0		
		1 x x x	Link	L
		x 1 x x	Insert 1	I
		x x 1 x	Shift R	R
		x x x 1	Modify Condition Flags	C

SECTION 3

I/O AND MEMORY INTERFACE

SECTION 3

MICRO-ONE I/O AND MEMORY INTERFACE

3.1 GENERAL DISCUSSION

There are three primary input/output interfaces on the Micro-One computers for connecting external equipment to the interfaces: Byte I/O, Direct Memory Access and Serial I/O. In the Micro-One, I/O and memory operations are performed under control of microcommands. A standard set of I/O macro-instructions is used for performing program-controlled and concurrent I/O operations.

Information in Sections 3 through 8 discuss I/O and memory, and are arranged as follows: Section 3 provides general information on the various Micro-One I/O interface systems; Section 4 provides a detailed description of the programmed Byte I/O interface system, including programmed transfers, block automatic (concurrent) transfers, and external interrupts. Section 5 describes the CPU Memory interface; Section 6 contains a detailed description of the DMA port; Section 7 describes the Serial I/O interface, and Section 8 is an I/O connector signal list. Section 9 is an I/O and memory term glossary.

3.2 I/O ORGANIZATION

Figure 3-1 is a block diagram of a typical Micro-One serial computer system showing the three primary I/O interfaces: Serial I/O channel, Parallel Byte I/O channel, and DMA port. These three interfaces provide the system designer with the flexibility to structure efficient I/O systems for a wide range of applications. The serial I/O interface, although most commonly used with a teletype, can be used for other bit-serial devices as well. The byte I/O interface can be used by controller circuit boards that plug into the mainframe chassis. The DMA interface provides the method for external I/O devices to communicate directly with core memory. The user can design his own interface controller for the DMA port, or use standard Microdata DMA interfaces.

3.3 SERIAL I/O INTERFACE

The serial I/O interface is designed primarily for communicating with a full-duplex teletype. Character assembly and disassembly, with all timing and synchronization, are performed at the microprogram level. Two macro instructions, Input Byte Serially (IBS) and Output Byte Serially (OBS), are used for communicating with the serial channel device.

Note that the Micro-One firmware (-13) for these instructions was designed to operate with a 110-baud teletype. The designer can alter the timing of the serial channel for teletype (or other serial device) compatibility by performing a simple change in firmware.

3.4 BYTE I/O INTERFACE

The byte I/O interface provides for transfer of bytes over a party line I/O bus under microprogram control. The standard Micro-One computer firmware provides both programmed and concurrent I/O transfer capability, along with a priority interrupt system.

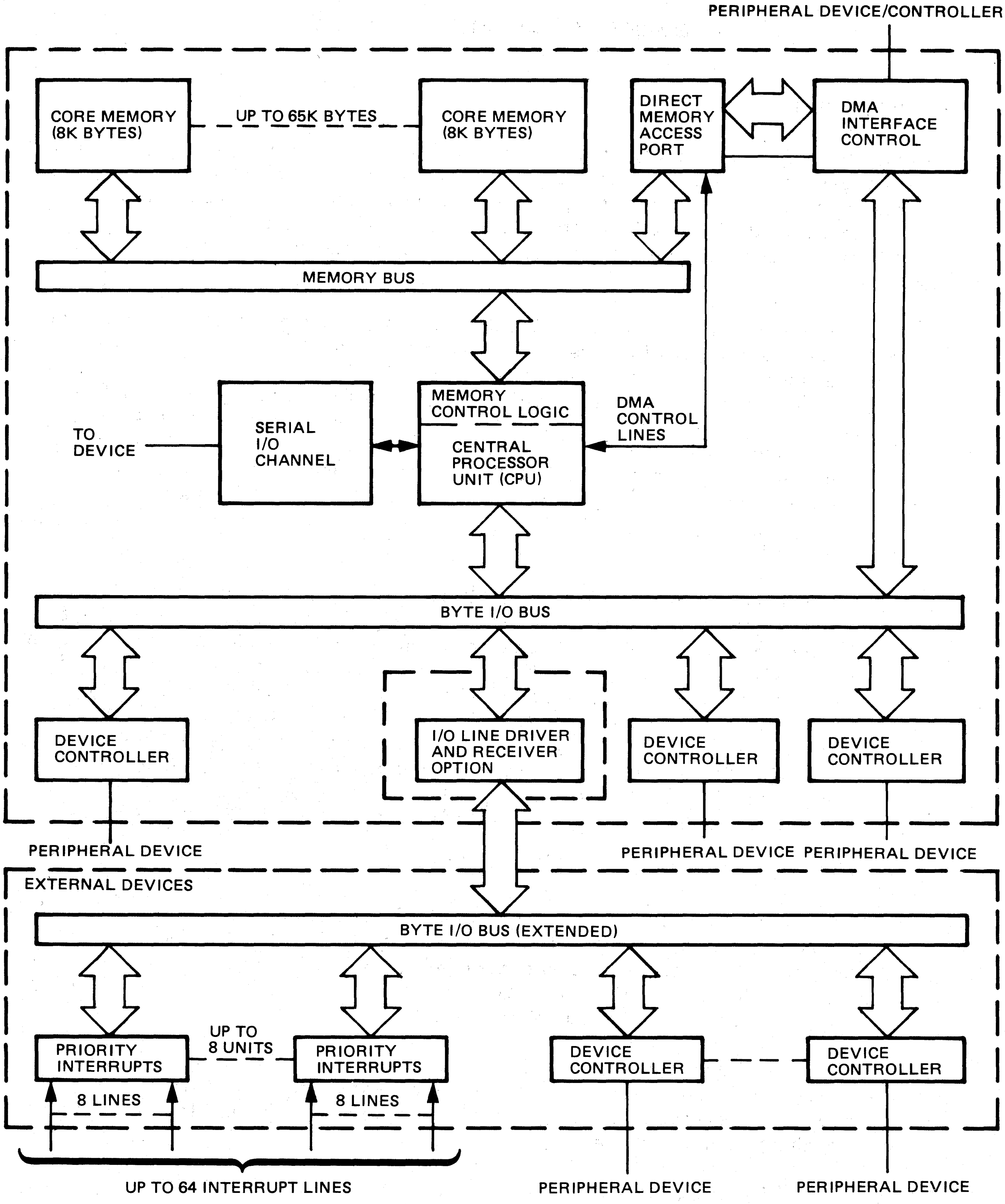


Figure 3-1. Typical Micro One Series I/O Configuration

Data transfers through the byte I/O interface are basically two-phase operations. First, a control byte is placed on the byte I/O bus before the actual transfer of data. The control byte contains an I/O controller device address and a device order code for the type of operation to be performed during the transfer (data transfer, status/function transfer, etc.)

All controllers on the bus examine the device number, but only the addressed controller accepts the control byte and logically connects itself to the bus for the subsequent data byte transfer. During the second phase of the byte I/O operation a single byte is transferred to or from the I/O controller. After each byte transfer, the controller disconnects itself from the bus.

3.4.1 Program-Controlled I/O

In standard firmware sets, such as the Micro-One/21, two basic instructions (one for input and one for output) are used for transferring information to and from controllers on the byte I/O bus under programmed control. These instructions permit transfers between the device controller and the A Register, B Register, or Memory. Up to eight types of input and eight types of output instructions may be defined for a particular controller. Generally these include function output, data output, status input, and data input, and are determined by a 3-bit device order in the control byte of the I/O instruction.

3.4.2 Concurrent I/O

The concurrent I/O feature provides the capability for automatic block transfers between core memory and I/O controllers connected to the byte I/O interface. The concurrent mode transfer rate is a function of the firmware set used in the computer. As an example, standard Micro-One/21 firmware performs concurrent transfers at rates up to 20,000 bytes per second.

Once started, the transfers are fully automatic and proceed without program intervention. Concurrent I/O operations take priority over instruction execution, and force a break in the execution of long instructions such as multiply, divide, and shifts to ensure that concurrent I/O servicing delays are not excessive. Concurrent I/O operations make use of pairs of two-byte address control words stored in dedicated core memory locations. One pair of address words is used by each controller. The control words, which contain the address of the current byte being transferred and the address of the last byte in the block, are initially set by the software program and thereafter are manipulated automatically by firmware for each byte transferred.

3.4.3 External Priority Interrupts

The external interrupt system of Micro-One series computers operates through the byte I/O interface in both the computer mainframe and expansion chassis. Interrupts can originate from device controllers or from the optional Priority Interrupt interface board connected to the byte I/O bus. The Priority Interrupt interface board provides control of eight external interrupt signals.

The byte I/O interface contains a single external interrupt request line, common to all I/O controllers on the byte I/O bus, and a priority line that is carried sequentially through all controllers on the bus. Each I/O controller receives priority from the preceding controller in the priority chain. Priority is passed a long to the next controller in line only if the previous controller is not ready to request an interrupt. When a controller receives priority and is ready to request an interrupt, it stops the progression of the priority signal and activates the interrupt request signal.

After receiving acknowledgment of the interrupt request, the interrupting controller places an address byte on the I/O bus that the processor uses to transfer program control to the proper interrupt servicing routine.

3.4.4 Direct Memory Access DMA Port

All Micro-One series computers contain a DMA Port through which data can be transferred between core memory and I/O devices at rates up to one million bytes per second. The DMA Port provides this high transfer rate and low access latency time, for use with high-speed, demand-type devices such as rotating memories.

The DMA Port consists of the memory bus and various DMA memory control lines which are available in the computer mainframe. It is up to an external DMA controller to manipulate the control lines and place data and addresses on the memory bus at the proper times when DMA transfers take place. The DMA controller may be designed and constructed by the user, or a standard Microdata DMA controller can be used.

The standard programmed I/O instructions are used to set up the DMA Controller and, if so designed, I/O device controller(s) attached to the DMA Controller with the parameters of the transfer. Program communication with these device controllers takes place over the byte I/O bus. Once the transfer is initiated, the DMA Controller and attached device controller supervise the transfer, and only minimum attention from the microprogram is required. Standard Microdata DMA Controllers can accommodate several external device controllers.

SECTION 4
BYTE I/O INTERFACE

SECTION 4

BYTE I/O INTERFACE

4.1 INTRODUCTION

The Byte I/O interface, to which the parallel-byte device controllers are connected, contains input control lines, input data lines, output control lines, output data lines and spare lines. The points of origin or destination in the CPU of the byte I/O interface lines are shown in Figure 4-1.

4.2 BYTE I/O BUS

The following paragraphs describe the I/O data and control lines of the byte I/O bus. Unless noted, descriptions apply to both internal and external buses.

4.2.1 Input Data Lines

Input data lines lD00/through ID07/ are terminated on the CPU input bus by lK pullups to +5V. The lines are driven by 7438 TTL power gates, or equivalent, with uncommitted collectors on each controller. When a gate switches on, the connected line swings to ground potential and places a logical 1 on the B bus. When the gate is switched off, the line swings to +5V. The input data lines are handled the same whether the device controller is located in the mainframe or in an external chassis.

4.2.2 Output Data Lines

Output data lines OD00/through OD07/ originate at the processor Output Data Register. Data or address information to be transferred over the output data lines is transferred from the A Register, the B Register, or Memory, into the Output Data Register and onto the lines. Lines OD00/through OD07/ are present at all CPU I/O backplane connectors.

To preserve the expansion capability of the byte I/O bus, each device controller on the bus is restricted to a single unit load (one TTL gate, 1.6 ma maximum) on each of the output data lines. Two loads are allowed if one load is a low power TTL gate such as a 74L04.

Output Data Register	ODOX/
Binary 1	0V
Binary 0	+4V nominal

4.2.3 Input Control Lines

The input control lines on the byte I/O bus are:

- ECIO/ - Concurrent I/O request
- IRPY/ - I/O Reply (spare)
- EINT/ - External interrupt

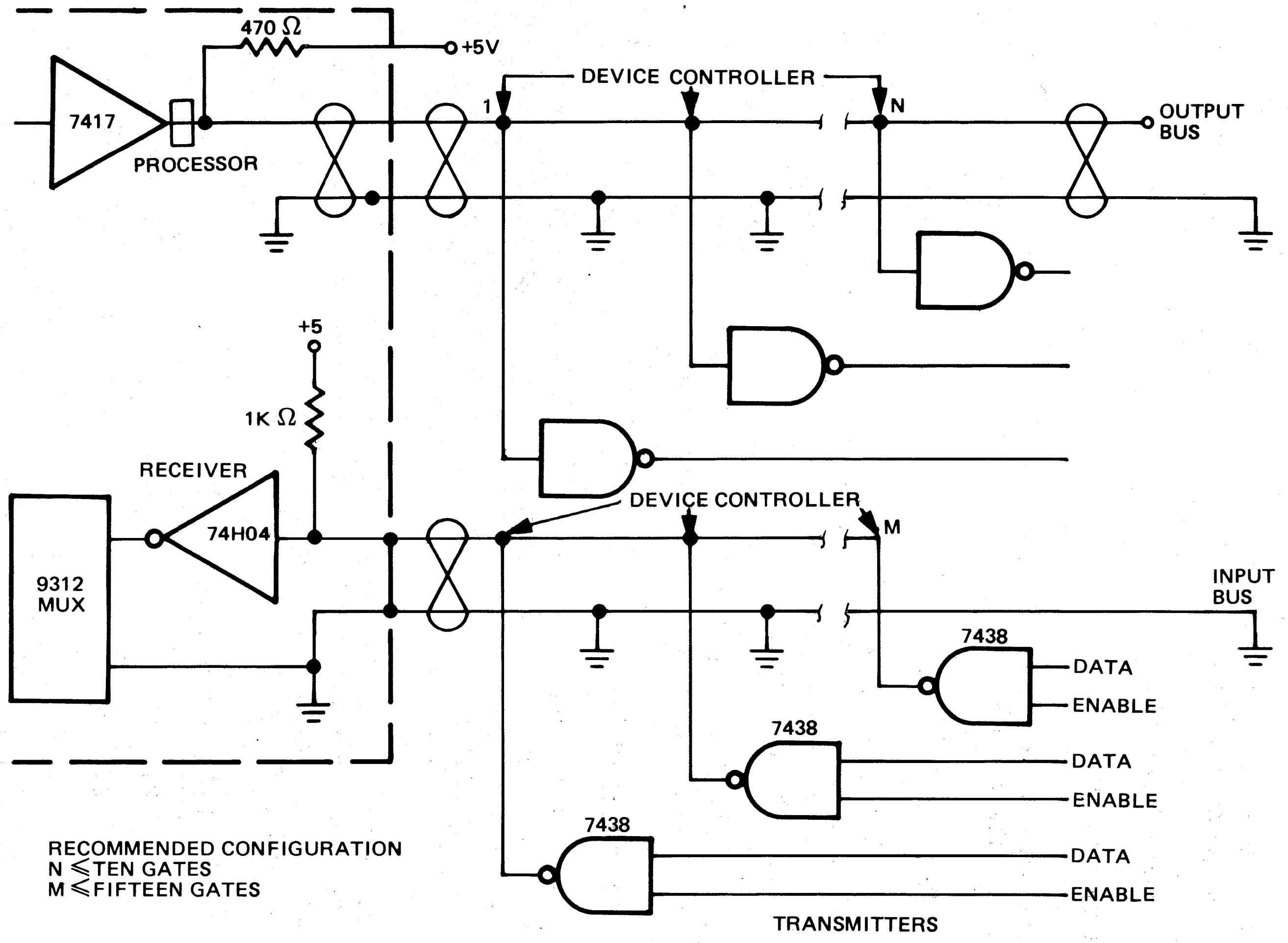


Figure 4-1. Micro One I/O Bus Lines

These lines are present in the I/O card connectors in the backplane. The lines are driven by 7438 TTL power gates (or equivalent) with uncommitted collectors in each controller. A 1K pullup resistor for each line is included in the processor, except for E1NT/ which has a 470 ohm pullup. All lines are active (indicate assertion) when they are at ground potential. For example, ground potential on the E1NT/ line causes an external interrupt request.

4.2.4 Output Control Lines

The output control lines on the byte I/O bus are:

- I01X/ -
- I02X/ - I/O control bits 1 - 3 from I/O Control Register
- I03X/ -
- CPH1 - Processor Clock
- CPH2 - Processor Clock (inverted & delayed 33 nsec from CPH1)
- MRST/ - Master Reset
- PRIN/ - Priority In
- PROT/ - Priority Out
- SELI - Select In
- SELO - Select Out

4.2.4.1 Control Lines I01X/ through I03X/

These lines are tied to the buffered inverted outputs of the I/O Control Register in the CPU which is set and reset at the microcommand level. Device controllers connected to the I/O bus decode these lines into eight assigned states, indicating various I/O control modes. Table 4-1 provides standard definitions of the eight control flip-flop states. Other definitions can be devised for systems not using standard Microdata firmware and I/O controllers.

Subsequent discussions refer to the conditions of I01X/ through I03X/ by the logic terms assigned to the eight states of these lines (COXX/, DOXX/, etc) which are decoded in the I/O controllers.

Table 4-1. I/O Control States

I/O Control Register State (Binary) IOIX = LSB IO3X = MSB	Control Definition	Logic Term
0	None	None
1	Control output	COXX/
2	Data output	DOXX/
3	Space serial Teletype	SP1
4	Concurrent I/O acknowledge	CACK/
5	Interrupt acknowledge	IACK/
6	Data input	DIXX/
7	Spare	SP3/

4.2.4.2 Lines CPH1 and CPH2/.

These lines provide processor clock signals to device controllers. Each line can be used independently as a 5 MHz square wave source, or the lines may be NANDed together to produce a 33-nanosecond clock pulse (CPH1 is inverted and delayed approximately 33 nanoseconds to form CPH2/). The relationship of the signals on lines CPH1 and CPH2/ is shown in Figure 4-2.

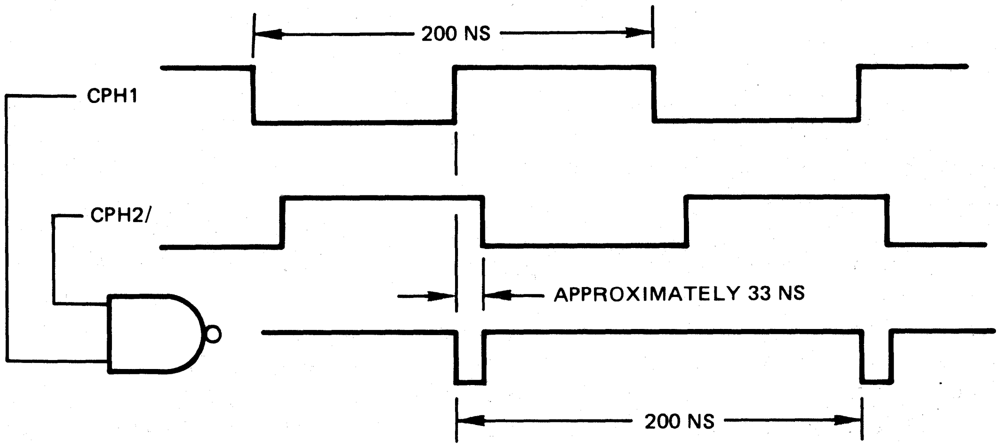


Figure 4-2. Relationship of Control Signals CPH1 and CPH2/

4.2.4.3 Control Line MRST/

This line is the master reset line which is activated by the front panel RESET switch, or by the power fail or restart. It is used to clear all control flip-flops to their initialized condition. Ground potential is applied to this line when the RESET switch is pressed.

4.2.4.4 Control Line PROT/, PRIN/

This line carries interrupt request priority from controller to controller along the CPU backplane. The line is labeled PROT/ (Priority Out) as it leaves the CPU and each controller, and enters each controller as term PRIN/ (Priority In).

Relative priority of each controller is determined by the positions of the controller boards. The first controller in the mainframe (nearest the CPU) has highest mainframe priority with the last controller having lowest mainframe priority.

4.2.4.5 Spare Lines

Spare lines SP2 and SP7 are applied to the byte I/O bus. They are provided only for special requirements and are not terminated in any way on the standard Micro-One.

4.2.4.6 Control Line SELO/, SELI/.

This line carries external interrupt and concurrent I/O select priority from controller to controller along the CPU backplane. It is labeled SELO/ (Select Out), as it leaves the CPU, and each controllers, and enters each controller as term SELI/ (Select In).

Selection priority of the controllers is determined by the positions of the controller boards in the mainframe and expansion chassis (para 4.2.4.4).

NOTES

- A controller must receive PRIN/ to make an external interrupt request. The requesting controller removes PROT/ from all lower priority controllers to lock out lower priority interrupt requests. The requesting controller must receive SELI/ from the preceding controller to respond to an interrupt acknowledgment (IACK/).

- Any controller can make a concurrent I/O request (ECIO/) at any time. Simultaneous concurrent requests are handled in order of priority as a requesting controller must receive SELI/ in order to respond to the concurrent I/O acknowledgment (CACK/). A controller requesting a concurrent I/O transfer will not pass SELO/ to the next controller until it has transferred one data byte.

- Descriptions of the PROT/, PRIN/, SELO/, and SELI/ functions are provided in paragraphs 4.6 through 4.6.3.

4.3 BYTE I/O FUNDAMENTALS

Though the flexibility of the byte I/O technique lends itself to individualized applications, certain standard conventions have been adopted for byte I/O operations in the Micro 1600 and Micro-One series computers. These conventions are described in the following paragraphs.

Byte-programmed I/O operations provide transfers of data, control, and status information over the byte I/O bus. This multiplex channel permits intermixed program and concurrent I/O transfers. More than one device on the bus can be operating in a concurrent block transfer mode at the same time. A maximum of 32 controllers can normally be addressed on the byte I/O bus.

The second byte of an I/O instruction is a control byte containing a 3-bit device order and a 5-bit device address.

DEVICE ORDER (f)	DEVICE ADDRESS (d)
7 6 5	4 3 2 1 0

The two-phase operations used with the Micro-One I/O concept are discussed in paragraph 1.6.

4.3.1 Device Addresses

Each I/O controller on the byte I/O bus is assigned a unique 5-bit device address. On Microdata device controllers, standard addresses are assigned by printed circuitry on the controller board. Other addresses may be assigned by cutting the etch and installing jumper wires.

Each device controller on the I/O bus determines if it is being addressed by comparing its assigned address to the 5-bit device number in the control byte sent to all controllers on the output data lines. The device address portion of the control byte appears on data lines OD00/ through OD04/. The assigned device address is also used to identify the I/O controller requesting an interrupt or concurrent I/O transfer. The processor acknowledges each request with signal IACK/ (for interrupts) or CACK/ (for concurrent I/O). On receiving the acknowledgment signal, the requesting controller places its address on input data lines ID01/ through ID05/. For concurrent I/O operations, the controller indicates the direction of data transfer by bit ID07/; a 1 indicates output, a 0 indicates input.

Table 4-2 lists the device addresses assigned to Microdata standard interface units. Customer-designed controllers should not use the addresses assigned to standard Microdata controllers which are to be used. Each priority interrupt group listed in the table is one set of eight priority interrupt levels on the optional (8-level) Priority Interrupt Board.

Table 4-2. Standard I/O Device Addresses

ADDRESS (HEXADECIMAL)	I/O DEVICE
00	Teletype (Model 2610 or Integral)
01	Asynchronous Modem or TTY/CRT Controller
02	High Speed Paper Tape Reader
03	High Speed Paper Tape Punch
04	Card Reader
05	Line Printer
06	Unassigned
07	Unassigned
08	Input/Output Expander (32 X 32)
09	Magnetic Tape (1 to 4 Drives)
0A	Magnetic Tape (1 to 4 Drives)
0B	8 Channel Asynchronous Modem Controller
0C	8 Channel Asynchronous Modem Controller
0D	8 Channel Asynchronous Modem Controller
0E	8 Channel Asynchronous Modem Controller
0F	Unassigned
10	Synchronous Modem Controller
11	Synchronous Modem Controller
12	Unassigned
13	Unassigned
14	Disc Controller (1 to 4 Drives)
15	Disc Controller (1 to 4 Drives)
16	DMA Channel Controller
17	Unassigned
18	Unassigned
19	Unassigned
1A	4 or 8 Channel Communications Controller
1B	4 or 8 Channel Communications Controller
1C	4 or 8 Channel Communications Controller
1D	4 or 8 Channel Communications Controller
1E	4 or 8 Channel Communications Controller
1F	ACM or Priority Interrupt Group 1

4.3.2 Device Orders

Accompanying the 5-bit device address in the control byte is a 3-bit device order specifying the I/O operation to be performed by the controller. The device order portion of the control byte appears on output data lines OD05/ through OD07/.

Table 4-3 provides a list of standard device orders. Not all device controllers use all orders listed in the table. Their use is dictated by controller design and device requirements. Many controllers have unique device order definitions.

Table 4-3. Standard Device Orders

DEVICE ORDER (Value of f)	OPERATION	DESCRIPTION
0	Data	Data order causes a data byte to be transferred between processor and addressed controller. Direction of transfer depends on the type of instruction (input or output).
1	Status/function	Status/function order causes a status byte to be transferred from addressed controller to processor, or a function byte to be transferred from processor to controller depending on the type of instruction (input or output).
2	Block input with interrupt	Block input with interrupt order notifies the addressed controller to proceed with a concurrent block input to memory, and to generate an interrupt at end of transfer unless controller has been subsequently disarmed. This order is sent with an output instruction.
3	Arm interrupt	Arm interrupt order permits addressed controller to make an external interrupt request on satisfying the interrupt condition. This order is sent with an input instruction.
4	Disconnect	Disconnect order causes the addressed controller from making an external interrupt request under any condition and releases priority to lower devices. This order is sent with an output instruction.

Table 4-3. Standard Device Orders (Continued)

DEVICE ORDER (Value of f)	OPERATION	DESCRIPTION
5	Disarm interrupt	Disarm interrupt order inhibits the addressed controller from making an external interrupt request under any condition and releases priority to lower devices. This order is sent with an output instruction.
6	Block output with interrupt	Block output with interrupt order notifies addressed controller to proceed with concurrent block output from memory, and to generate an interrupt at the end of the transfer unless the controller has been subsequently disarmed. This order is sent with an output instruction.
7	Not assigned	This order, if assigned, may perform any required function as interpreted by the individual controller. The order can be sent with either an input or output instruction to cause a corresponding byte transfer.

4.3.3 Status Bytes

In response to a status order from the processor (specified by the device order), the addressed I/O controller places a status byte on input data lines ID00/ through ID07/. Four of the status bits are common to most device controllers; the other four are device dependent and differ from controller to controller. The status byte is transferred into the A Register, the B Register, or Memory by an input instruction, generally with device order 1. The significance of each bit in a typical status byte is described in Table 4-4.

4.3.4 Function Bytes

Some device controllers perform a greater number of operations than are specified by the 3-bit device order. In these cases, a function byte is output to the controller by an I/O command to further specify the desired operation. The byte is defined as a function byte by proper coding of the device order field (f).

Table 4-4. Typical Status Byte Definition

BIT NO.	CONDITION	DESCRIPTION
0	Ready	Set to 1 when I/O controller is in a ready state and not performing a concurrent I/O operation.
1	Input flag	Set to 1 when I/O controller has a byte ready for input to the processor.
2	Output flag	Set to 1 when I/O controller is ready to receive a byte from the processor.
3	Error	Set to 1 when an error has occurred during a transfer operation. The error may be the result of timing, parity, or a device malfunction. The bit is cleared when the status byte is transferred.
4 - 7	Undefined	Unique for each I/O controller.

4.4 BYTE I/O OPERATIONS AND TIMING

The following paragraphs describe the program-controlled and concurrent I/O operations of the Micro-One computer. Timing diagrams are included for each operation, and typical controller logic for performing the operations is shown. For a description of the I/O instructions that pertain to the byte I/O operations, refer to Micro-One/21 Computer Reference Manual, publication No. RM2000 1551-421.

4.4.1 Program Controlled I/O Operations

The four basic types of program controlled I/O operations are:

- Output of data to I/O controller
- Output of function (control) information to I/O controller.
- Input of data to computer
- Input of controller status information to computer.

Each type of I/O operation consists of two phases; the address/order phase and the transfer phase. The address/order phase is the portion of the operation during which the desired controller is addressed and the device order bits are sent to the controller. The second phase transfers one byte of information between the I/O controller and the computer. Timing diagrams for output and input operations are shown in Figures 4-3 and 4-4, respectively.

4.4.1.1 Address/Order Phase

The address/order phase is identical for all types of I/O operations regardless of whether the information is an input command (IBA, IBB, IBM) or an output command (OBA, OBB, OBM), or if the type of information transferred is data, a controller function, or controller status.

All input and output instructions consist of either three or four bytes. The first byte specifies the direction of transfer and the source/destination within the CPU (A or B Register, or Memory). The second byte contains a 5-bit device address specifying the desired controller, and a 3-bit device order code containing control information for the controller. During the address/order phase, this second instruction byte is placed on the I/O bus and applied to the inputs of all device controllers. If the source/destination is memory, the instruction consists of four bytes, with the last two bytes specifying a core memory address.

Two hundred nanoseconds after the second byte is placed on the I/O bus, the I/O Control Register changes from state 0 to state 1. The controllers decode state 1 of IO1X/ through IO3X/ to produce control output signal COXX/. During COXX/, each controller examines lines OD00/ through OD04/ to determine which controller is being addressed. The controller whose address is on the lines connects itself for service and decodes and stores the device order code on lines OD05/ through OD07/. The I/O Control Register remains in the COXX/ state for at least 800 nanoseconds, after which it returns to state 0 (no function) and the address/order byte is removed from the output data lines. This marks the end of the address/order phase.

At this point, the addressed controller has been connected to the I/O bus and has received, decoded, and (in most cases) stored the function information contained in the device order code. The device order code may specify:

a. Transfer of data byte between the processor and controller.

b. Transfer of a status/function byte between the processor and controller.

c. One of several possible control functions to be performed by the controller (i.e., rewind magnetic tape, set up for concurrent transfer, enable interrupt, etc.).

If the device order code specifies transfer of data, the controller readies itself for a transfer to or from its data register. The direction of transfer is specified by the computer during the second phase.

A device order specifying a status/function transfer causes the controller to ready itself for transfer to or from its status/function register. In most standard controllers, the status/function register is used only to hold controller status information for transfer to the computer. In these controllers, all necessary control functions can be specified in the device order alone. Some controllers, however, require a wider range of control than is provided by the 3-bit device order. In these cases, the next byte output to the controller is sent to the status/function register where it specifies controller and/or device actions. When the device order specifies a device/controller function, the byte transferred during the second phase may be ignored by the controller.

4.4.1.2 Transfer Phase

Transfer of the data, function, or status byte takes place during the second (transfer) phase of the I/O operation. During the transfer phase the I/O Control Register changes to one of two states to specify the direction of transfer. For input operations, the controller places the byte being transferred onto the input data lines ID00/ through ID07/ and the CPU strobes the byte into the destination register or memory location. During output operations, the byte is applied to output data lines OD00/ through OD07/ by the computer and is strobed into the appropriate controller register. At the end of the transfer phase, the I/O Control Register returns to state 0.

4.4.1.3 Data Output Operations

The transfer phase of a data output transfer begins when the computer places the data byte on the output data lines. Two hundred nanoseconds later, the I/O Control Register changes to state 2. The controller decodes the state of IO1X/ through IO3X/ to produce data output signal DOXX/, indicating the presence of data on the lines. The controller then strobes the data byte into its data register. When the I/O Control Register returns to state 0, the controller disconnects itself from further service. A timing diagram for data output operations is shown in Figure 4-3.

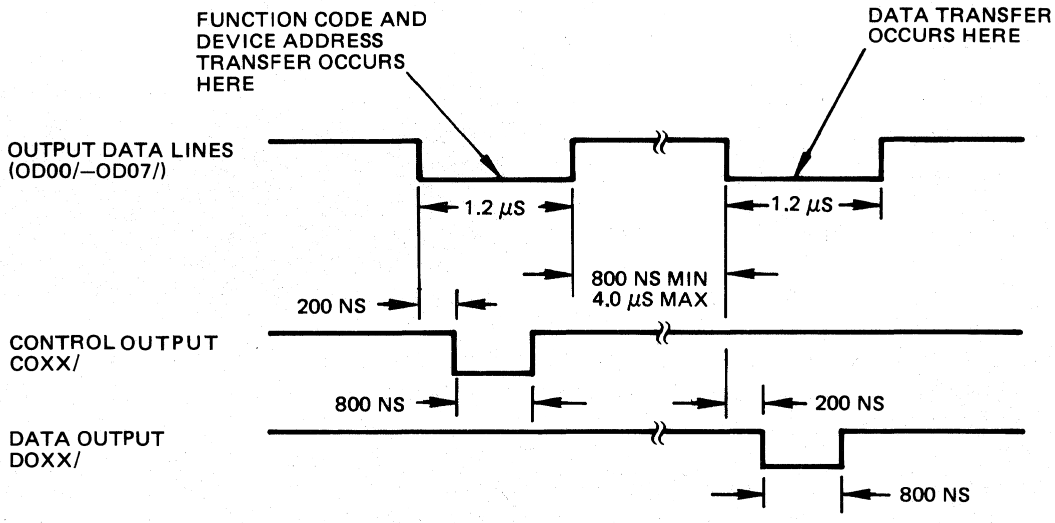

FUNCTION CODE AND DEVICE ADDRESS TRANSFER OCCURS HERE
DATA TRANSFER OCCURS HERE
OUTPUT DATA LINES (OD00/—OD07/)
1.2 μS
1.2 μS
800 NS MIN
4.0 μS MAX
200 NS
CONTROL OUTPUT COXX/
800 NS
200 NS
DATA OUTPUT DOXX/
800 NS

Figure 4-3. Data or Function Output Timing

4.4.1.4 Function Output Operations

The timing diagram of Figure 4-3 for a data output operation is also valid for a function output operation. The function output operation is typically used to control a discrete action in an I/O device requiring no data transfer. Rewinding tape is an example of such an action. The most efficient way to perform this operation is to issue a single instruction containing all the information necessary to alert the device and cause the tape to rewind.

In the Micro-One computer, the output byte instructions (OBA, OBB, OBM) are also used to perform the function output operation. The only difference in the instructions is the value of the device order (bits 5 through 7) in the second byte of the instruction.

When an output byte instruction is used for function output the device order code of the second byte designates the unique function in the I/O device to be controlled. The assignment of device order codes for function operations precludes the use of the same codes for data transfer operations.

The function output operation is executed exactly like the data output operation previously described. A data byte is transferred from either the A Register, the B Register, or Memory, depending on the output instruction used. This data byte is usually ignored by the device controller since the device order code of the control byte contains enough information to describe most function operations. However, should a controller require more function definition than is possible in the control byte, the data byte transferred during the function operation could be used to carry additional function information, as described in Paragraph 4.3.4.

4.4.1.5 Data Input Operations

The timing diagram for a typical data input operation is shown in Figure 4-4. Four hundred nanoseconds after the address/order byte is removed from the data output lines the I/O Control Register changes from state 0 to state 6. The controller decodes the state of IO1X/ through IO3X/ to produce data input control signal DIXX/. DIXX/ causes the controller to place the data byte (from the controller data register) on input data lines IDOO/ through ID07/. The data must be settled no later than 400 nanoseconds after DIXX/ goes low, and remains stable until DIXX/ again goes high. When DIXX/ goes low, IO3X/ is low. This causes selection of the input bus as a data source whenever the firmware command designates T Register as a source.

Note: The controller may place the data byte on the lines as early as the beginning of signal COXX/.

Input data must be removed from the input data lines no later than 400 nanoseconds after DIXX/ goes high. For normal operation on the external byte I/O bus with less than a 30-foot twisted pair cable, the DIXX/ signal can be used for gating or qualifying data applied to the input data lines.

The timing diagram shown in Figure 4-4 is also valid for a status input operation. A similar relationship exists between status and data input operations as exists between data and function output operations. In the Micro-One computer, the input byte instructions (IBA, IBB, IBM) are used both for data and status input operations. The only difference is the source of the data byte in the controller. In status input operations, the source of the input byte is the status/function register. To differentiate between the two operations, a device order code of 000 is used in the control byte for data transfer and a code of 001 is normally used for status transfer.

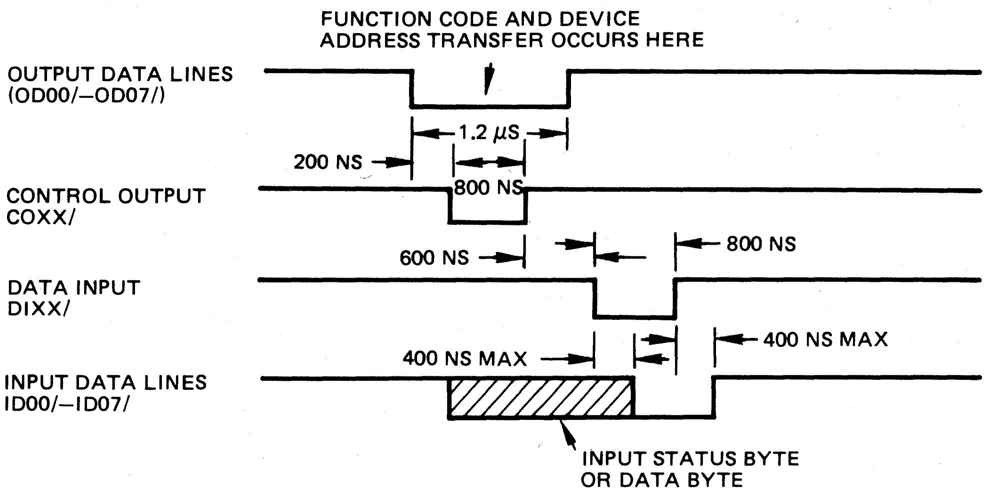

FUNCTION CODE AND DEVICE ADDRESS TRANSFER OCCURS HERE
OUTPUT DATA LINES (OD00/–OD07/)
1.2 μS
200 NS
800 NS
CONTROL OUTPUT COXX/
600 NS
800 NS
DATA INPUT DIXX/
400 NS MAX
400 NS MAX
INPUT DATA LINES ID00/–ID07/
INPUT STATUS BYTE OR DATA BYTE

Figure 4-4. Data or Status Input Timing

4.5 CONCURRENT I/O OPERATION

Concurrent I/O operation is the name given to the block transfer technique used in the standard Micro-One computer. The software program sets up starting and stopping addresses for the block transfer in dedicated memory locations and executes an input or output instruction to initiate the transfer; thereafter, firmware controls the data transfer operation automatically on demand from the device controller. In terms of controller design, the additional logic required for concurrent I/O operations can be thought of as an overlay to the program-controlled logic discussed earlier. All that is required is additional logic to recognize a data ready condition and to assert a concurrent I/O request at that time instead of waiting for an input or output instruction, as is the case with program-controlled operations. A device performing a concurrent I/O operation initiates its own data transfers when it is ready.

4.5.1 Concurrent I/O Timing

The timing for a typical concurrent I/O operation is shown in Figure 4-5. When the device controller is ready to transfer data, it causes the concurrent I/O request line (ECIO/) to go low. After recognizing the request, processor firmware causes it to respond by setting the I/O Control Register to state 4, producing the I/O acknowledge signal CACK/ in the controller. While CACK/ is low, the device controller applies an address byte containing its own address and a bit indicating the direction of transfer (input or output) to input data lines ID01/ through ID07/.

Concurrent I/O operations utilize four dedicated memory locations to hold the 16-bit starting and final addresses of the block to be transferred. In Micro-One computers, the first of these 4 bytes is at the location specified by the controller device address times 4. When the processor acknowledges a

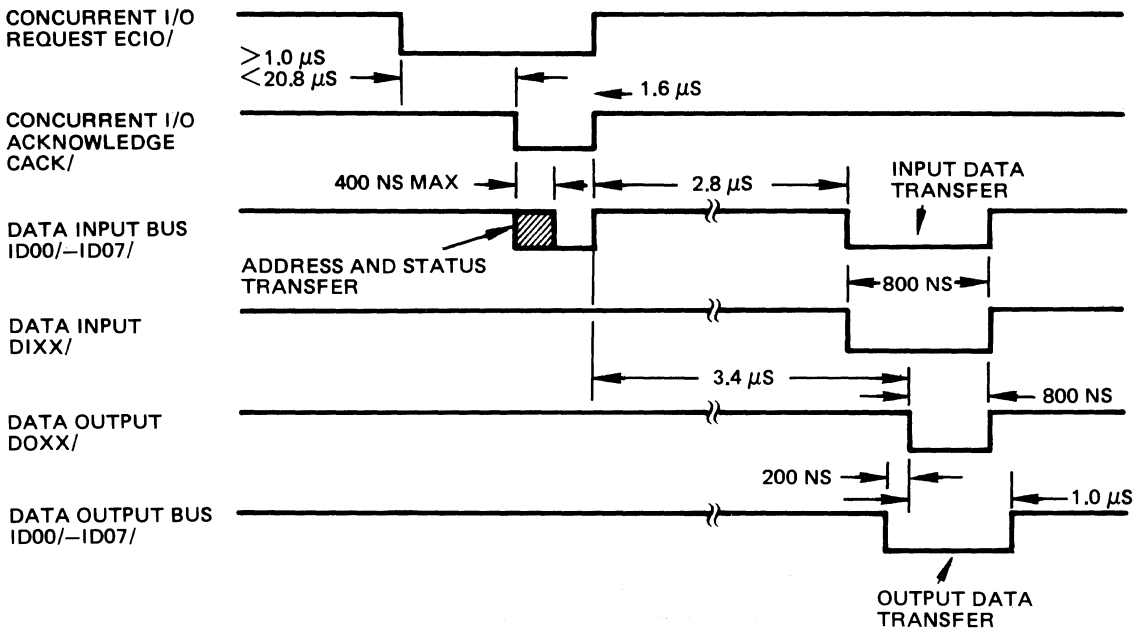
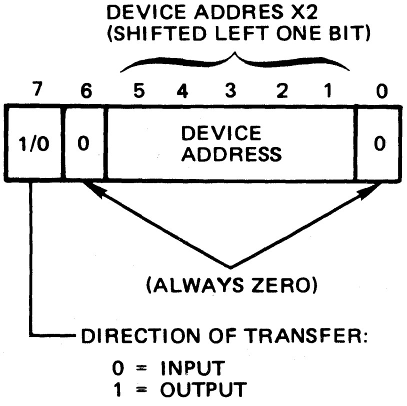

CONCURRENT I/O REQUEST ECIO/
>1.0 µS
<20.8 µS
1.6 µS
CONCURRENT I/O ACKNOWLEDGE CACK/
400 NS MAX
2.8 µS
INPUT DATA TRANSFER
DATA INPUT BUS ID00/–ID07/
ADDRESS AND STATUS TRANSFER
800 NS
DATA INPUT DIXX/
3.4 µS
800 NS
DATA OUTPUT DOXX/
200 NS
1.0 µS
DATA OUTPUT BUS ID00/–ID07/
OUTPUT DATA TRANSFER

Figure 4-5. Concurrent I/O Timing

concurrent I/O request with CACK/, the controller places a byte on data input lines ID00/ through ID07/. This byte contains the device address times 2 (shifted left 1 bit) in the lower 6 bits with bit 7 set or reset to specify the direction of transfer.

This byte is shown in the following diagram.



When the computer receives this byte, it obtains the actual dedicated memory address (4 X device address) by shifting the 2X address (supplied by the controller) left 1 more bit position. The direction bit (7) is shifted out and used by the computer to specify an input or output operation. Examples of device addresses, addresses supplied by the controller, and dedicated memory addresses are as follows:

Actual Device Address (Hex)	Address Supplied By Controller (Hex)	Dedicated Memory Address (Hex)
00	00	00 thru 03
01	02	04 thru 07
02	04	08 thru 0B
.	.	.
.	.	.
.	.	.
.	.	.
1F	3E	7C thru 7F

The firmware uses the address byte to initiate a normal input or output operation to or from memory. If an input is specified, the firmware asserts DIXX/ and the controller responds by placing a data byte on ID00/ through ID07/. The addresses of the block of memory locations to or from which the transfer takes place is specified by software. Prior to starting the block transfer, the program loads the 4 dedicated memory locations with a 16-bit starting (current) address and a 16-bit ending address. After each byte is transferred, the starting (current) address is incremented by the firmware. Thus, each byte is transferred to or from the next sequential location. After each transfer (but before the current address is incremented) firmware compares the current address with the final address. If the two are equal, the firmware tells the controller that the transfer is complete by essentially executing an output instruction with a device order of 4. Upon receipt of the second byte of this instruction (strobed by DOXX/), the controller issues an external interrupt to indicate to the CPU that the operation is complete.

4.6 EXTERNAL INTERRUPT OPERATION

Interface lines PROT/, PRIN/, SELO/, SELI/, EINT/, IACK/, and ID00/ through ID07/ are used by device controllers or the optional Priority Interrupt board on the byte I/O bus for external interrupt operations. Lines PROT/ and PRIN/ (paragraph 4.2.4.4) make up the hard-wired priority chain that determines the relative priority of each controller and Priority Interrupt board on the byte I/O bus. These lines determine priority for interrupt requests. Line EINT/ (paragraph 4.2.3) carries the interrupt request from the controller to the processor. I/O control register state 5 is decoded in the controller as interrupt acknowledge IACK/. Input data lines ID00/ through ID07/ carry an interrupt address byte from the interrupting controller to the processor in response to the interrupt acknowledge signal on line IACK/. The interrupt address byte is used by the processor to locate the entry address of the interrupt servicing subroutine.

4.6.1 Priority Determination

Interface units on the byte I/O bus are assigned priority for control of external interrupts and concurrent I/O request operations. The priority is achieved by the manner in which lines PRIN/, PROT/, SELI/, and SELO/ are used to link the interface units together. A typical example of priority wiring is shown in Figures 4-6 and 4-7. In these examples, three device controllers in the mainframe chassis are connected in the priority chain. The figures show that the priority of an interface unit is the same as the physical location of that interface on the byte I/O bus. With special priority wiring, however, the relative priorities can be independent of backplane positioning.

A device may make a concurrent I/O request at any time. However, to make an external interrupt request, the device must have priority in (PRIN/). Signal IS02 of Figure 4-6 on each interface unit inhibits propogation of PRIN/ if the interrupt servicing routine is not complete. This establishes a true-level priority among all interface units for generating an external interrupt. A controller never passes a low signal on line PROT/ if it is making a request or until the interrupt servicing routing is complete.

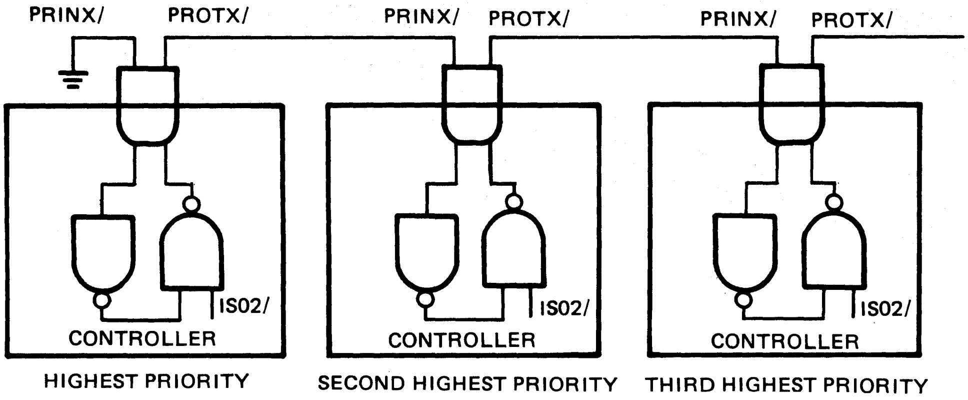


Figure 4-6. Typical Priority Scheme

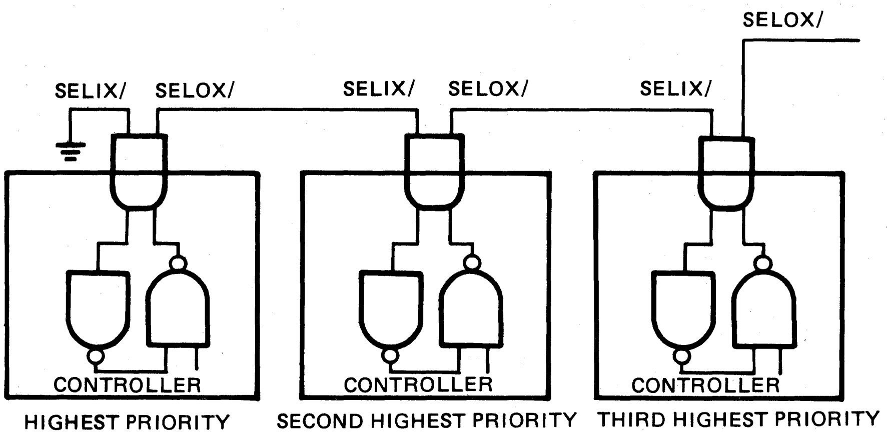


Figure 4-7. Typical Selection Acknowledgment Scheme

4.6.2 External Interrupt Requests

External interrupt requests from interface units are carried on line EINT/ to the processor. The internal microprogram recognizes the presence of an external interrupt request and responds as dictated by interrupt handling firmware. External interrupt line EINT/ can be used both by device controllers and by the optional Priority Interrupt interface board. The Priority Interrupt option provides the proper interface to the I/O bus, contains priority logic for each interrupt level, and permits processor control over the handling of interrupts. This standard option provides, on one circuit board, convenient hardware for 8 levels of system interrupts. Because the basic interrupt facility makes use of the byte I/O bus, all device controllers have access to the interrupt request line and can react to the firmware interrupt handling sequences in the processor (provided they operate according to the design guidelines given in paragraph 4.6.3.

Note: Requesting an interrupt removes priority for interrupt operations from all controllers lower on the priority chain

4.6.3 Interrupt Sequence and Timing

Figure 4-8 shows the timing for a typical external interrupt sequence. The sequential firmware, processor and I/O device operation is:

a. The I/O device controller lowers line EINT/ to signal a request for microprogram attention. The controller must receive priority signal PRIN/ from the higher priority controllers. The requesting controller does not pass the priority signal to lower controllers.

b. At the end of the macro instruction currently being executed (if not a privileged instruction like I/O or jump), the microprogram senses the interrupt request and jumps to a firmware subroutine to handle it.

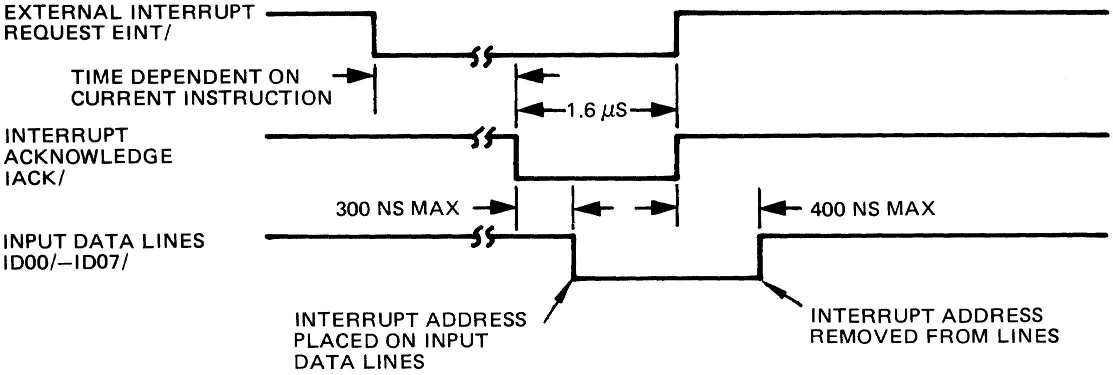

EXTERNAL INTERRUPT REQUEST EINT/
TIME DEPENDENT ON CURRENT INSTRUCTION
1.6 µS
INTERRUPT ACKNOWLEDGE IACK/
300 NS MAX
400 NS MAX
INPUT DATA LINES ID00/–ID07/
INTERRUPT ADDRESS PLACED ON INPUT DATA LINES
INTERRUPT ADDRESS REMOVED FROM LINES

Figure 4-8. External Interrupt Timing

c. The microprogram causes line IACK/ to go true to acknowledge the request. All controllers in the priority chain decode IACK/ and each requesting controller passes SELO/ down the chain to the lower priority controllers (SELO/ becomes SELI/ at the input to each controller).

d. The controller that issued the interrupt request does not pass SELO/ to the next controller. This prevents any lower priority controller that may have simultaneously requested an interrupt from responding to signal IACK/.

e. In response to the acknowledgement and receipt of SELI/, the requesting controller places a 6-bit interrupt address on input data lines (ID01/ through ID06/. The interrupt address specifies the location (in core memory page 1) of the 2-byte entry address for the interrupt servicing subroutine.

f. The processor accepts the 6-bit interrupt address and causes line IACK/ to go high.

g. The processor fetches the 2-byte interrupt subroutine entry address from the first 256-word page of memory using the interrupt address supplied by the controller as the lower 6-bits, and 01 as the upper two bits.

h. Using the 2-byte entry address, the microprogram executes a pseudo return jump or call instruction to the interrupt servicing subroutine at that address.

i. The interrupt servicing subroutine then proceeds to service the interrupt according to the macroprogram.

j. At the end of the servicing routine, priority is released by any of the 3 actions listed below. Any of these will cause the requesting controller to pass signal PROT/ to the lower priority controllers.

1. Rearming (if another request is expected)
2. A concurrent I/O request
3. Disarming (if no further requests are desired)

The interrupt sequencer in the controller contains two J-K flip-flops (and associated circuits) which generate the interrupt request (EINT/) and control the priority line PROT/ to the next controller. The 4 states of the flip-flops determine the priority interrupt status of the controller. These 4 states are illustrated in Figure 4-9 and described in Table 4-5.

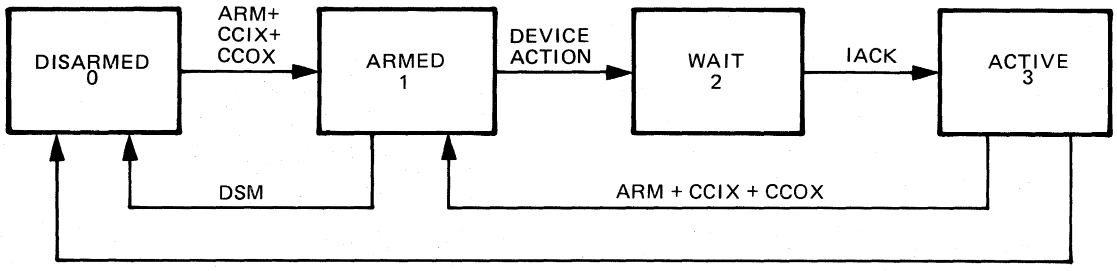


Figure 4-9. Interrupt Sequencer States

Table 4-5. Interrupt Sequencer States

State	Flip-Flop States IS02	Flip-Flop States IS01	Function
0	0	0	Disarmed state. Disregards any received interrupt and does not move to requesting state.
1	0	1	Armed state. Allows system to move to requesting state when peripheral conditions are met.
2	1	1	Wait state. Generates an external interrupt to the processor when priority in is received.
3	1	0	Active state. Inhibits propagation of priority to lower level priority controllers.

When the controller is initialized, the sequencer is set to state zero (disarmed). When disarmed, the controller cannot generate an interrupt request and always passes PROT/ to the next controller.

In order to allow an interrupt, the program must execute an instruction to arm the controller interrupt, setting the sequencer to state one (armed). In this state, the controller can generate EINT/ providing it has priority from the preceding controller on the priority chain.

When the controller is ready to interrupt the CPU, the sequencer advances to state two (wait), the priority line (PROT/) is removed from the next controller, and interrupt request EINT/ is generated. The firmware responds to the interrupt with the acknowledgment (IACK/). The first interrupting controller in the string that has SELI places its address on the input data lines. Its sequencer then advances to state three (active) and removes EINT/.

While the sequencer is in the active state, PROT/ is not passed to the next controller. This prevents a lower priority controller from generating an interrupt while the interrupt handling subroutine is in process. One of the functions of the interrupt subroutine is to execute an instruction to rearm the controller if another interrupt is expected, or to disarm the controller if no further interrupts are desired. The rearming or disarming normally takes place near the end of the subroutine and restores priority (PROT/) to the lower priority controllers.

Concurrent I/O operations are normally terminated with an end-of-operating interrupt to inform the CPU that the block of data has been transferred. The concurrent I/O request automatically arms the controller interrupt.

SECTION 5

MICRO-ONE CPU READ/WRITE MEMORY INTERFACE

SECTION 5

MICRO-ONE CPU
READ/WRITE MEMORY INTERFACE

5.1 PROCESSOR AND MEMORY INTERFACE

Figure 5-1 illustrates a portion of the processor and memory interface consisting of the following elements:

- Control Section
- Memory Read Data Selection Logic
- Memory Write Data Gating Logic
- M and N Register Address Gating Logic

The remainder of this section discusses each element in turn.

5.1.1 Control Section

The Control Section is the main control element of the Micro-One CPU. However, only that part of the Control Section pertaining to memory is discussed. One function of the Control Section is to decode the CPU memory microcommand OPA/. OPA/ requests memory service for the CPU, and eventually results in a Memory Busy (MBSY) condition while the memory services the CPU. While MBSY is high, due to memory use by the CPU, the DMA is prevented from issuing a request for memory service. While memory is being used by the DMA channel, the Control Section is inhibited from executing OPA/.

The Control Section also generates Timing Hold signal THLD/. In a DMA operation, THLD/ is used to stop the main CPU clocks while a DMA memory cycle is occurring. This essentially freezes execution of microcommands and prevents CPU memory request while memory is servicing the DMA channel. Signal THLD/ is generated by either of two sets of conditions relative to DMA operations:

1. Simultaneous OPA/ and DMAR/

2. OPA/ while MBSY/ is active

The third function of the Control Section which relates to memory operations is generation of Transfer Memory Clock signal LT2/. LT2/ clocks the memory read data through the Memory Read Data Selection Logic to the CPU T Register. When the DMA interface is using memory, LT2/ is inhibited. Inhibiting LT2/ prevents the data being transferred from memory to the controller from entering the CPU T Register.

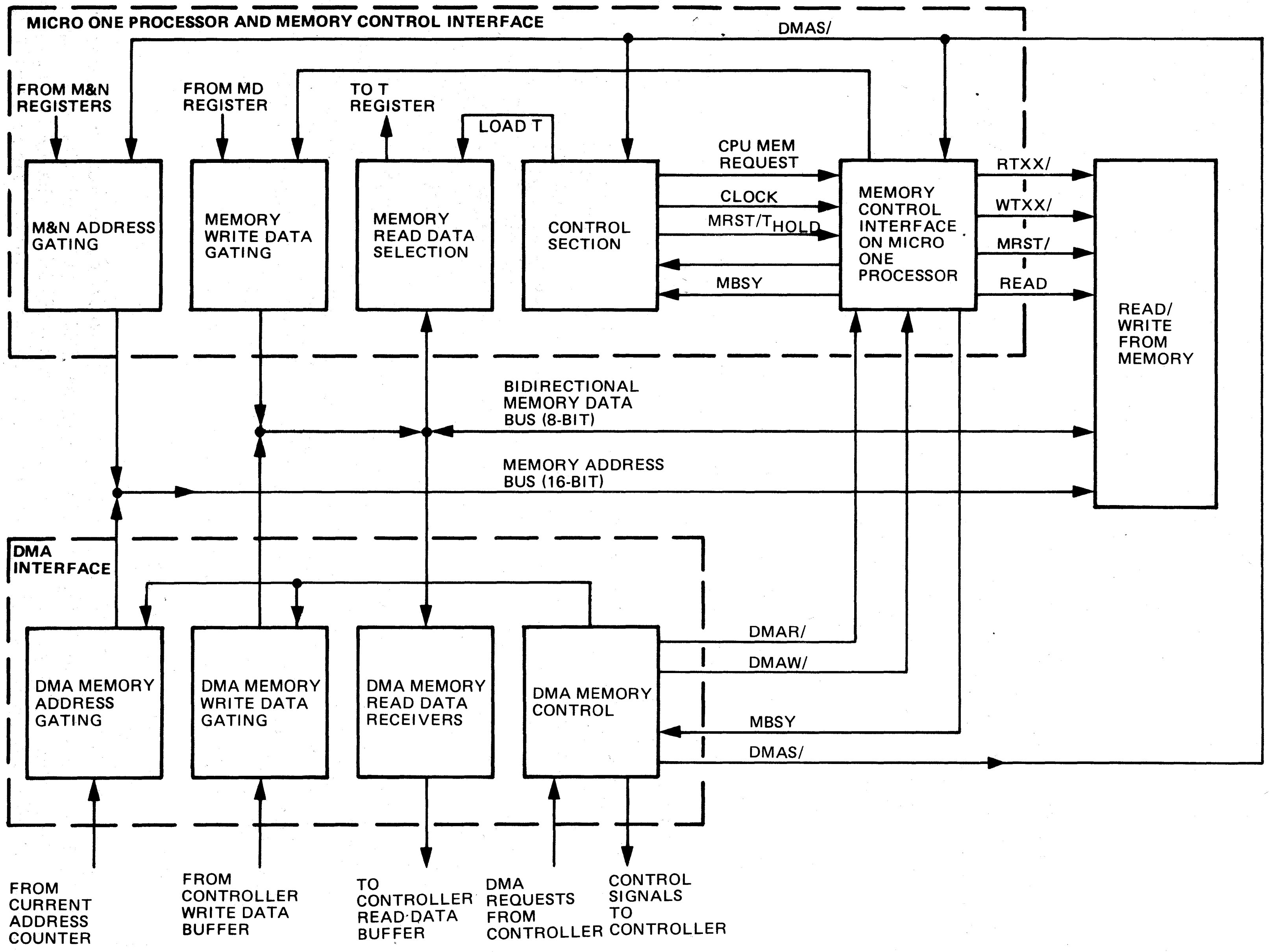

MICRO ONE PROCESSOR AND MEMORY CONTROL INTERFACE
DMAS/
FROM M&N REGISTERS
FROM MD REGISTER
TO T REGISTER
LOAD T
M&N ADDRESS GATING
MEMORY WRITE DATA GATING
MEMORY READ DATA SELECTION
CONTROL SECTION
CPU MEM REQUEST
CLOCK
MRST/T$_{HOLD}$
MBSY
MEMORY CONTROL INTERFACE ON MICRO ONE PROCESSOR
RTXX/
WTXX/
MRST/
READ
READ/ WRITE FROM MEMORY
BIDIRECTIONAL MEMORY DATA BUS (8-BIT)
MEMORY ADDRESS BUS (16-BIT)
DMA INTERFACE
DMA MEMORY ADDRESS GATING
DMA MEMORY WRITE DATA GATING
DMA MEMORY READ DATA RECEIVERS
DMA MEMORY CONTROL
DMAR/
DMAW/
MBSY
DMAS/
FROM CURRENT ADDRESS COUNTER
FROM CONTROLLER WRITE DATA BUFFER
TO CONTROLLER READ DATA BUFFER
DMA REQUESTS FROM CONTROLLER
CONTROL SIGNALS TO CONTROLLER

Figure 5-1. CPU and DMA Memory Interface (Detailed Block Diagram)

5.1.2 Memory Read Data Selection Logic

During CPU memory accesses (read mode), this element gates data read from memory into the CPU T Register. As explained in the preceding paragraph the Transfer Memory Clock LT2/ signal from the Control Section intiates the gating operation. To prevent read data gating to the CPU during DMA operations, DMAS/ inhibits generation of LT2/.

5.1.3 Memory Write Data Gating Logic

During CPU memory write operations this element gates the write data from the MD Register onto the Memory Data bus. Gating signal WRIT, originating in the Memory Control Interface, is inhibited during DMA operations.

5.1.4 M and N Register Address Gating Logic

During CPU memory accesses (read or write), this element gates the 16-bit memory address onto the Memory Address bus from the M and N Registers. During DMA operations, when the memory address is applied by the DMA interface, this function is inhibited.

5.2 MEMORY CONTROL INTERFACE

The Memory Control Interface is the principal memory controlling element of the system. Its primary functions are:

- Generating memory status and control signals,

- Monitoring requests for memory service from the DMA and CPU,

- When memory is not busy, determining which requesting device (CPU or DMA) is to receive access. Determination is based on DMA having highest priority and the CPU having second priority,

- Initiating memory read or write operations and timing out the memory cycles.

The Memory Control Interface generates the Memory Busy (MBSY) signal to the CPU and to the DMA interface. When memory is available, this signal goes low and either or both DMA and CPU can request service. If both elements request simultaneously, the DMA has priority and will receive memory service. The DMA memory request signal is DMAR/; CPU memory request signal is OPA/.

When the DMA or CPU request sequence occurs, the Memory Control Interface generates the appropriate memory control signals (listed below) and times out the memory cycles, as shown in Figure 5-2.

- RTXX/ - Start read portion of cycle,
- WTXX/ - Start write portion of cycle,
- READ - Low level = clear/write or half-cycle write;
  High level = read/restore or half-cycle read.
- MBSY/ - Active low during all memory sequences. Used internally by the CPU and externally to control DMA interface devices.

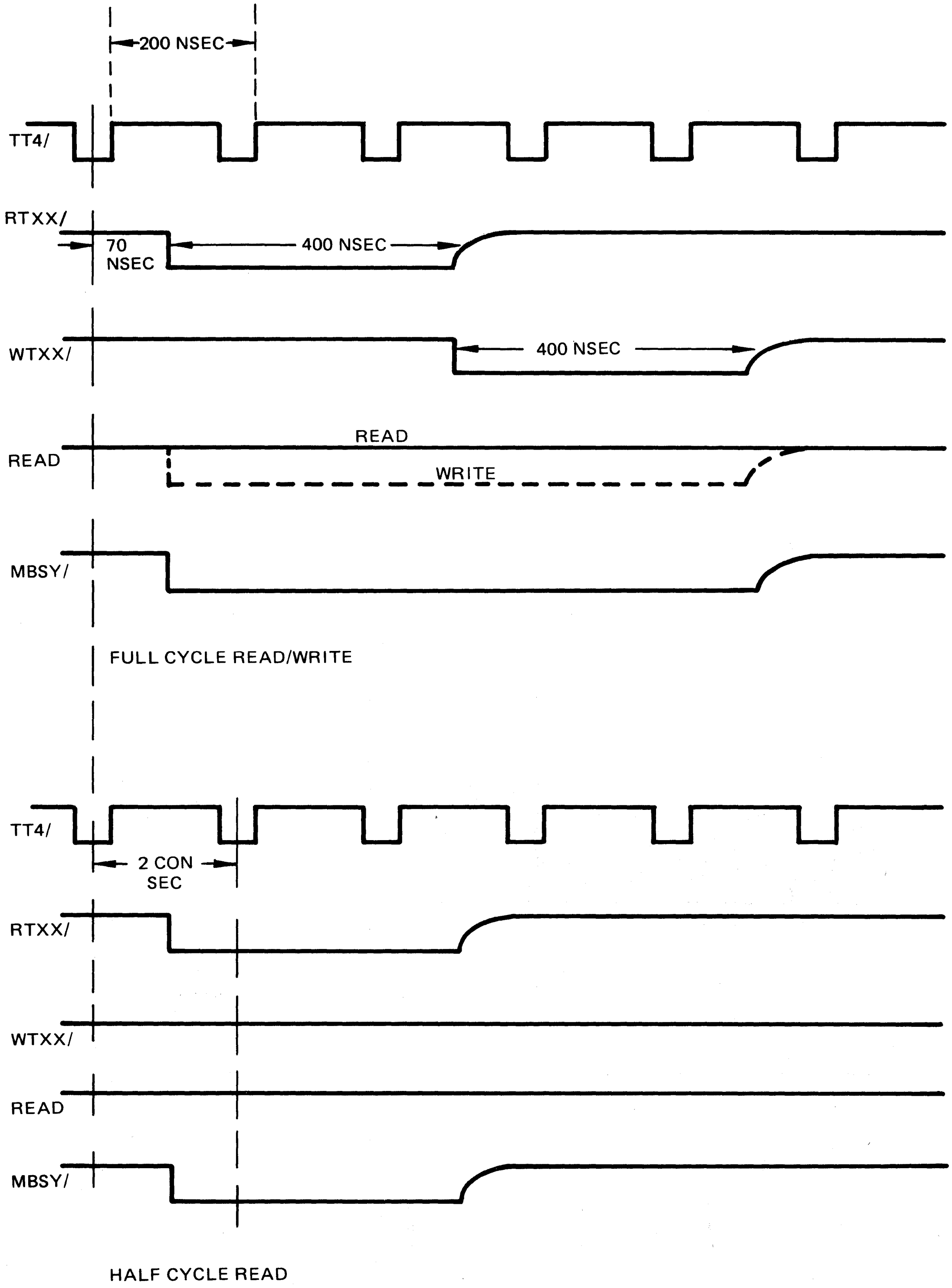
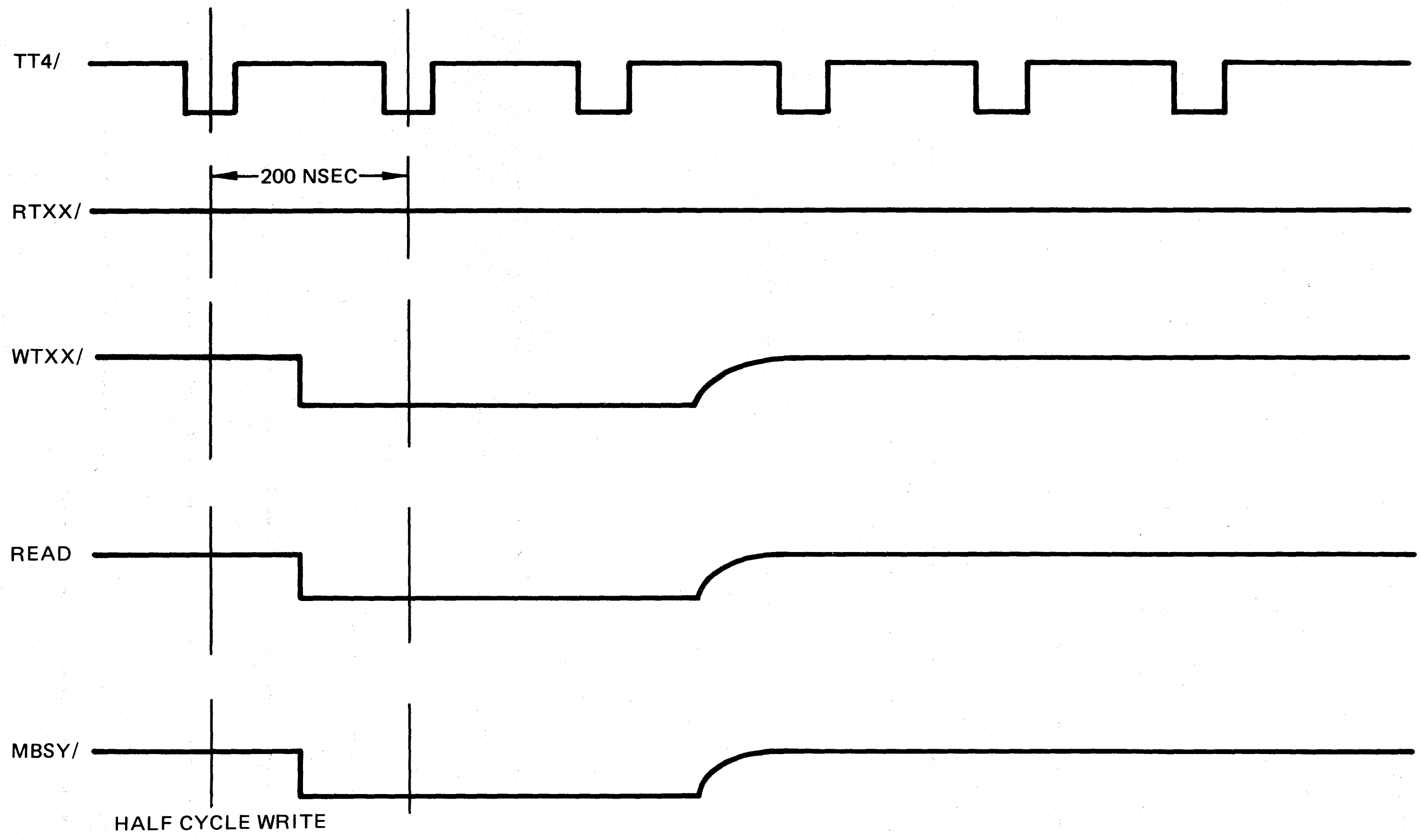


Figure 5-2. Half Cycle Read (Sheet 1 of 2)



NOTE: FOR MOS MEMORY OPERATIONS, THE WAVEFORMS FOR RTXX/, WTXX/, READ, AND MBSY CAN BE REPROGRAMMED IN THE MICRO ONE MEMORY SEQUENCER ROM. MINIMUM PULSE WIDTH IS 200 NANOSECONDS. MAXIMUM CYCLE TIME FOR THE SEQUENCER IS 1.6 MICROSECONDS.

Figure 5.2. Memory Timing Signals

SECTION 6

DIRECT MEMORY ACCESS PORT

SECTION 6

DIRECT MEMORY ACCESS PORT

6.1 INTRODUCTION

The Direct Memory Access (DMA) Port is a channel to/from the Micro-One core memory through which data may be transferred at very high speeds without involving the Micro-One CPU. DMA transfers are controlled entirely by an external DMA interface and occur at the rate of the external device up to full memory speed. At maximum memory speed, transfers take place at 1 million bytes per second using back-to-back, 1.0 microsecond memory cycles. The DMA Port consists of the set of lines used for DMA control and data transfer. These lines are available at all backplane card slots.

A simplified functional block diagram of the Micro-One DMA system is provided in Figure 6-1. The external peripheral device is conventionally controlled by a device controller connected to the byte I/O bus. Function output and status input are accomplished via the byte I/O bus. This includes commands to intiate the DMA transfer. As indicated in Figure 6-1, these latter functions may be performed within the DMA interface.

The DMA interface logic/buffers and device controller may be on one printed circuit board or on separate boards. If separate boards are used they are interconnected by cables attached to the rear of the boards. A separate DMA interface is available from Microdata.

6.2 FUNCTIONAL DESCRIPTION

In systems utilizing the Micro-One's DMA capability, core memory is shared by the DMA interface and the CPU. When either element requires a read or clear/write memory cycle it issues a request for memory service. To ensure minimum latency time in answering DMA requests and to prevent the CPU from interrupting DMA operations, the DMA interface always has priority over the CPU for memory service. When the DMA interface requests memory service it essentially freezes any CPU activities from occurring.

The memory responds to a DMA interface request as soon as the current CPU memory cycle is completed. The DMA interface then selects the read or clear/write mode and gates a 16-bit address onto the memory address bus from a DMA interface address counter or the device controller. The data byte is then transferred.

This process is repeated every time the device controller is ready to do a data transfer. Sequential addresses are gated onto the lines until an entire, predetermined block has been transferred. Determining the end of the DMA block transfer is made in the interface or device controller by two 16-bit registers; the starting (current) address register, and the final address register, which are loaded with the desired limits of the block prior to transfer. Following the transfer of each byte, the current address register is incremented to the next sequential address and compared to the final address register.

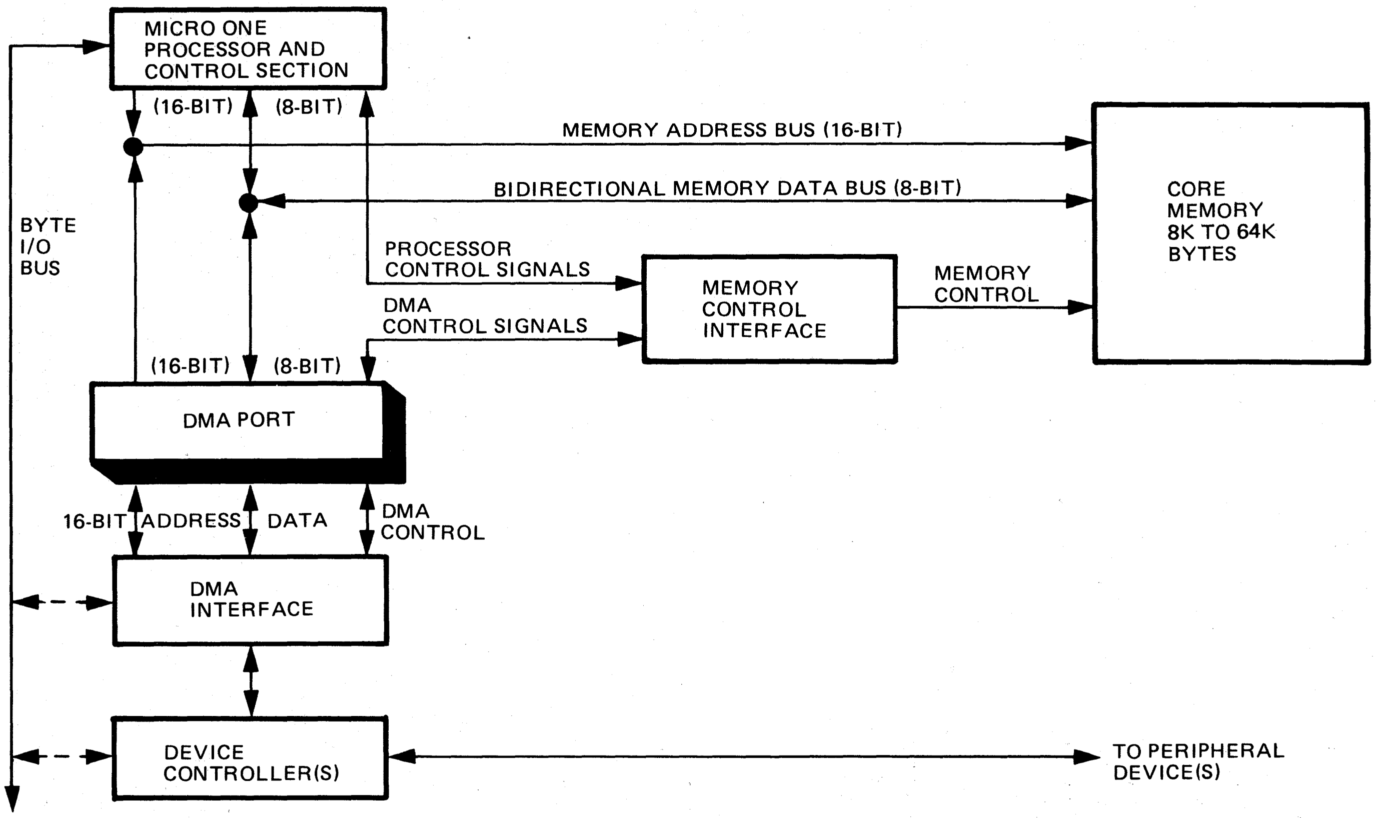


Figure 6-1. DMA/Processor Core Memory Interface (Simplified Block Diagram)

An end-of-block interrupt should be generated in the DMA interface or device controller. This may be either an external interrupt via the I/O bus or an internal DMA interrupt which sets bit 1 of the CPU's internal status register.

The following paragraphs describe the various elements in the DMA system and their functions in DMA operations. (See Figure 5-1.)

6.2.1 DMA Interface

The DMA interface provides interface and controls required for use of the Micro-One DMA capability. A standard Microdata DMA interface or an individually designed interface may be used.

The elements of the DMA interface are:

- DMA Memory Control Logic
- DMA Memory Read Data Receivers
- DMA Memory Write Gating Logic
- DMA Memory Address Gating Logic

6.2.2 DMA Memory Control Logic

When the DMA interface has been readied for DMA operation this element monitors contoller requests for DMA transfers. After receiving a controller request, the DMA Memory Control Logic intitiates a DMA memory cycle sequence as soon as the CPU memory cycle is completed (MBSY = low level). The DMA memory cycle sequence consists of a DMA request (DMAR/), DMA selection (DMAS/), and read or write mode selection (DMAW or DMAW/, respectively). These signals originate in the DMA Memory Control section and are sent to the Memory Control Interface in the Micro-One.

Signal DMAS/ performs the following functions in both the DMA interface and the Memory Control Interface:

- Gates memory address from current address counter of DMA controller onto memory address bus.
- For memory write operations, gates write data from DMA controller onto bidirectional memory data bus.

Signal DMAW is set low (DMAW/) by the DMA Memory Control if a clear/write cycle is being requested. It is set high (DMAW) if a read cycle is requested.

6.2.3 DMA Memory Read Data Receivers

Data read from memory during a DMA transfer is buffered from the bidirectional data bus by eight receivers in the DMA interface. The outputs of the receivers are applied to the controller.

6.2.4. DMA Memory Write Gating Logic

This element gates the 8-bit data byte from the controller onto the bidirectional data bus and is written into memory during memory write operations. The data byte is gated onto the bus by DMA Selection signal DMAS/.

6.2.5 DMA Memory Address Gating Logic

This element gates the 16-bit address stored in the current address register (contained in either the DMA interface or the controller) onto the memory address bus. Address gating occurs at DMAS/ time.

6.3 DMA PORT/MEMORY CONTROL INTERFACE TIMING

To simplify the timing of essential DMA Port/Memory Control Interface signals, Figure 6-2 illustrates a DMA clear/write memory cycle, followed immediately by a DMA read/restore cycle, followed by a Micro-One processor read/restore memory cycle. The sequences shown are not necessarily typical, but serve to define all DMA signal and timing requirements.

At the start of the timing sequence, the memory is not busy and the DMA request (DMAR/) and CPU request for memory service OPA occur during the same clock period (t_0 to t_1). Since the DMA channel has priority over the CPU for memory service, the DMA is granted service before the CPU in cases of simultaneous requests. The DMA receives memory service on the TT4/ computer clock pulse t_1, provided the timing requirements of paragraphs 6.3.1 and 6.3.2 are met.

TT4/ clock pulses occur every 200 nsecs, and 5 pulses (1 microsecond total) occur during each complete read/restore or clear/write memory cycle. Therefore, the DMA is granted service for the second memory cycle (read/restore) on clock pulse t_6, which occurs 1 microsecond after t_1 (start of the first memory cycle). The processor memory service request is not answered until the end of the second DMA cycle (t_{11}). The CPU has been locked out of memory service for 2 consecutive memory cycles by the DMA. Since the DMA is not requesting memory service for the third memory cycle starting at time t_{11}, the processor is allowed to perform its read/restore operation at this time.

During the two DMA cycles, the processor operation freezes. Clock Stop signal THLD/ stops or inhibits certain computer clocks so that the memory type microcommand OPA is not executed. OPA remains true until THLD/ is removed at time t_{10} to t_{11}, allowing DMA use of memory during THLD/. It is assumed that a non-memory type microcommand will be decoded and executed during clock period t_{11} to t_{12}.

6.3.1 Clock Signals

The clock signals TT4/, CPH1, and CPH2/ are used by the Micro-One processor, Memory Control interface, and the DMA interface to time and synchronize all memory operations. These clocks are generated by a single crystal oscillator clock generator located on the CPU control board. TT4/ is the main reference clock signal for all timing requirements and is used to generate CPH1 and CHP2/. CPH1 and CPH2/ are phased clock signals, available to the DMA interface at any backplane connector at which the DMA interface board is installed. Combining CPH1 and CPH2/ produces a clock pulse similar to, and almost in phase with, TT4/. The clock developed from CPH1 and CPH2/ is used in the DMA interface to time the channel's control functions in synchronism with the processor and Memory Control interface.

6.3.2. DMA Port Signals

The following paragraphs describe the DMA Port signals generated in the DMA interface, along with their timing requirements. Figure 6-2 is referenced as an aid in conveying the timing relationships.

6.3.2.1 DMA Request (DMAR/).

DMAR/ directly generates THLD/ during clock pulse t_1 if OPA is present. CSTP/ in turn, inhibits processor execution of OPA and fetching of another microcommand. DMAR/ also enables setting of the Memory Sequencer on clock periods t_1 and t_6 to begin each DMA memory cycle. The MBSY and THLD/ signals are held true during the DMA cycles by the Memory Sequencer.

DMAR/ must go low not later than 60 nsec before the leading edge (low-to-high) of TT4/ pulse t_1, and remain true until 20 nsecs after the trailing edge of t_6 in order to initiate a DMA cycle of time t_1. It must return to the false (high) state before the leading edge of time t_2. Once DMAR/ goes high after initiating a DMA memory cycle, it must remain high until after memory becomes not busy (MBSY = low level) at the end of the current DMA cycle.

The DMAR/ line must be driven by a power gate (SN7438 or equivalent) with an uncommitted collector. The terminating resistor is located in the processor, allowing the line to swing between +5V and virtually OV.

6.3.2.2 DMA Write (DMAW/)

When DMAW/ is true (low level), it is applied to the Memory Control sequencer to enable a clear/write memory cycle. (See Figure 6-2, time t_0.) A false (high level) state of DMAW/ enables a read /restore DMA memory cycle as shown at time t_6.

DMAW/ must be stable in the desired state no later than 55 nsecs before the trailing edge of TTY/ clock pulse t_1. It must remain true until the memory goes not busy (MBSY = low level) after the trailing edge of clock pulse t_5.

The DMAW/ line must be driven by a power gate (SN7438 or equivalent) with an uncommitted collector. The terminating resistor is located in the processor which allows the line to swing between +5V and virtually 0V.

6.3.2.3 Memory Busy (MBSY)

MBSY is generated by the Memory Control interface to inform the DMA interface when memory is not busy so a DMA request can be issued. MBSY is used in the DMA interface to inhibit DMAR/ and DMAS/ until MBSY goes false (low level).

MBSY will go true (high) at the DMA interface about 40 nsecs after the trailing edge of the TT4/ clock pulse on which the memory cycle begins (t_1, t_6, and t_{11} of Figure 6-2) for A CPU memory cycle, and is 80 nanoseconds for DMA cycles. It remains true for four clock periods during a full memory cycle and goes false no later than 80 nsec after the trailing edge of the fifth clock pulse. During half memory cycles (initiated and controlled by the processor only), MBSY remains true for two clock periods, going false not more than 70 nsecs after the third clock pulse.

MBSY line must be terminated in a single unit load (one TTL gate input) which is equivalent to a maximum load of 2 ma.

6.3.2.4 Memory Addresses

The DMA interface must have the current address on the Memory Address bus at least 15 nsecs before the leading edge of TT4/ pulse on which the DMA memory cycle begins (t_1 and t_6). The address must remain stable on the bus until MBSY goes false (low)

Memory address lines M07A/ through M00A/ and N07A/ through N00A/ are terminated on the computer backplane. The lines must be driven by power gates (type SN7438 or equivalent) with uncommitted collectors. Because of the termination network, the lines are allowed to swing between virtually 0V and +3.6V.

6.3.2.5 Write Data

During clear/write operations the DMA interface must place the desired write data on the bidirectional memory data bus no later than the leading edge of the t_3 clock pulse (Figure 6-2). The data must remain stable until MBSY goes false (low).

Bidirectional memory data lines MD07 through MD00 are terminated on the computer backplane. The memory write data must be driven to the memory data bus by power gates (type SN7438 or equivalent) with uncommitted collectors. Because of the termination network the lines are allowed to swing between virtually 0V and +3.6V.

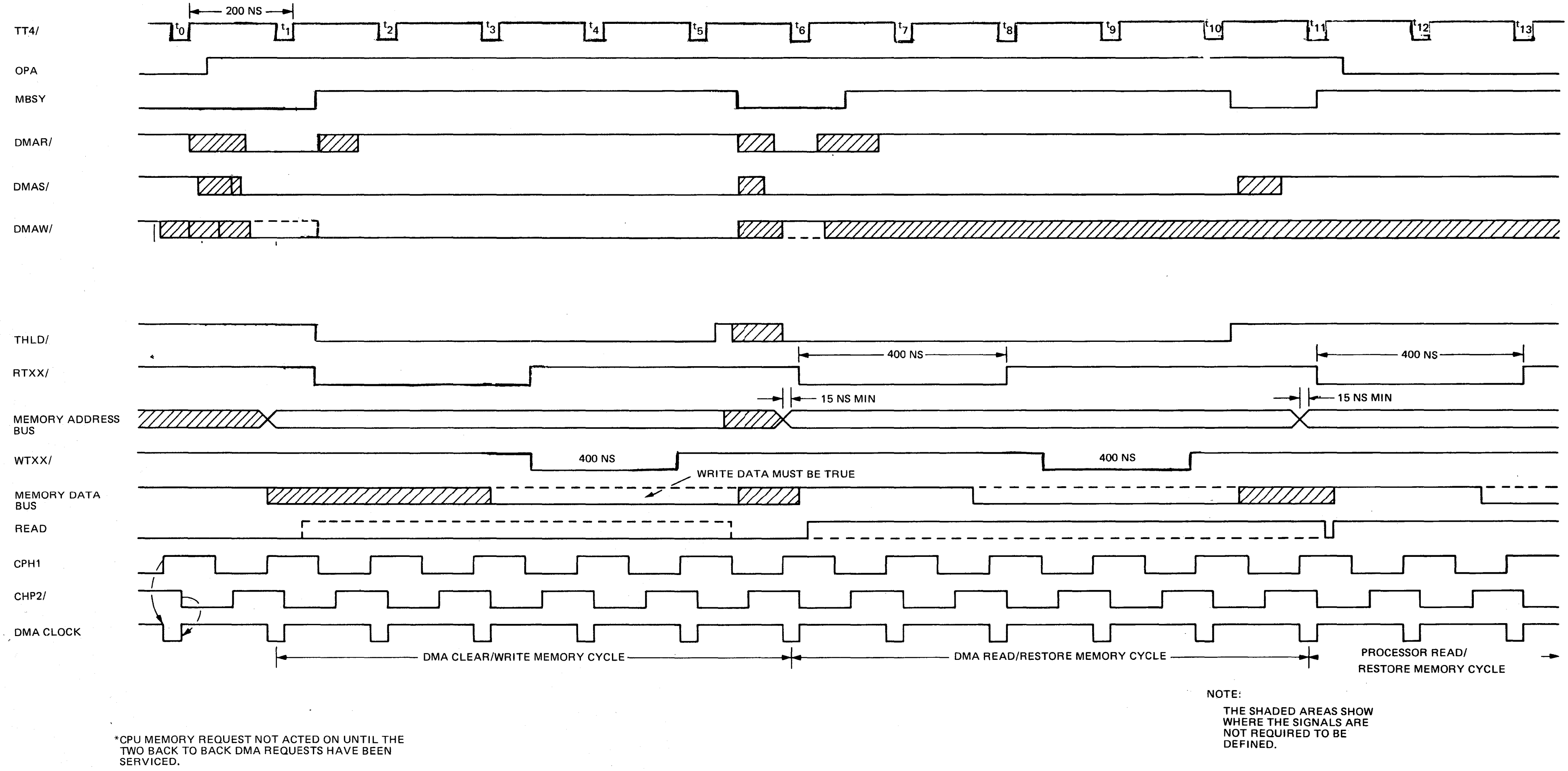


Figure 6-2. DMA Port/Memory Control Timing

6.3.2.6 Read Data

The data read from memory during a read/restore cycle will be available on the bidirectional memory data bus 445 nsec after the trailing edge of the TT4/ clock pulse on which the cycle started (t_6 of Figure 6-2). The data will remain stable for gating into the DMA interface until the trailing edge of the DMA clock pulse corresponding to TT4/ clock pulse t_{10}.

Read data, received from the bidirectional memory data bus, must be buffered by gates whose inputs load each line with only one unit load (one TTL gate input) that is equivalent to a maximum load of 2 ma.

SECTION 7
SERIAL I/O INTERFACE

SECTION 7

SERIAL I/O INTERFACE

7.1 INTRODUCTION

The Serial I/O interface is an optional feature of M-1 series computers. It is a hardware/firmware option using microprogramming to control a serial device such as a teletype or modem.

7.2 USE AS TELETYPE CONTROLLER

The following paragraphs describe operation of the serial I/O channel with a 4-wire, full-duplex, 20 ma teletype. A cable is provided with the serial channel to connect directly to the teletype.

7.2.1 General Operation

The 4-wire I/O interface circuit is shown in Figure 7-1. The transmit portion of the circuit contains a 20-ma current source that can be turned on or off depending on the state of the I/O control register. When the I/O control register is in any state other than state 3, output of gate 9E is high, emitter follower Q3 conducts, and approximately 20 ma of current flows through resistor R24. This current holds the teletype in the mark condition. When the I/O control register is set to state 3 by a microcommand, the output of gate 9E is low, emitter follower Q3 cuts off, and no current flows to the teletype.

The receive portion of the interface circuit contains a low-pass filter network connecting the teletype distributor to bit 6 of File Register 0 where it may be sensed by microcommands. One side of the teletype distributor is connected to -16.75 volts through resistor R23. The other side of the distributor is connected to 3M, which forms bit 6 of File Register 0. When the teletype sends a mark signal, the output of 3M is held low and a 0 bit appears in bit 6 of File Register 0. When the teletype sends a space signal a 1 bit appears in bit 6 of File Register 0.

7.2.2 Character Assembly and Disassembly

Teletype character assembly, disassembly, synchronization, and timing is accomplished by a firmware routine initiated by the macro instructions for the serial I/O interface. Figure 7-2 illustrates the timing for transmission or reception of 110-band teletype characters.

Note: The Micro One/13 is the only standard firmware set which has these serial I/O macro instructions.

During an input operation the firmware program searches for the leading edge of the start bit by continuously testing bit 6 of File Register 0.

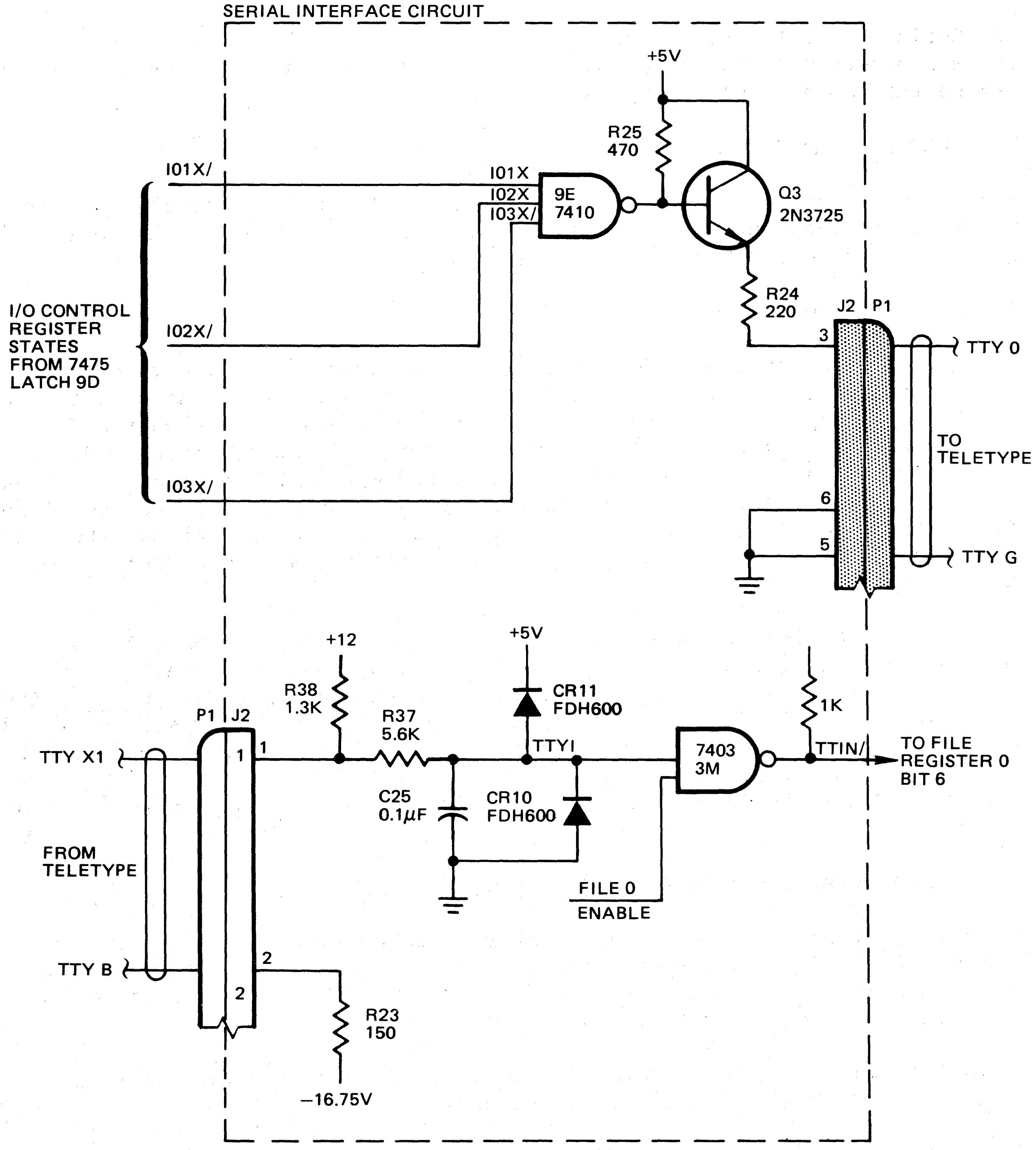
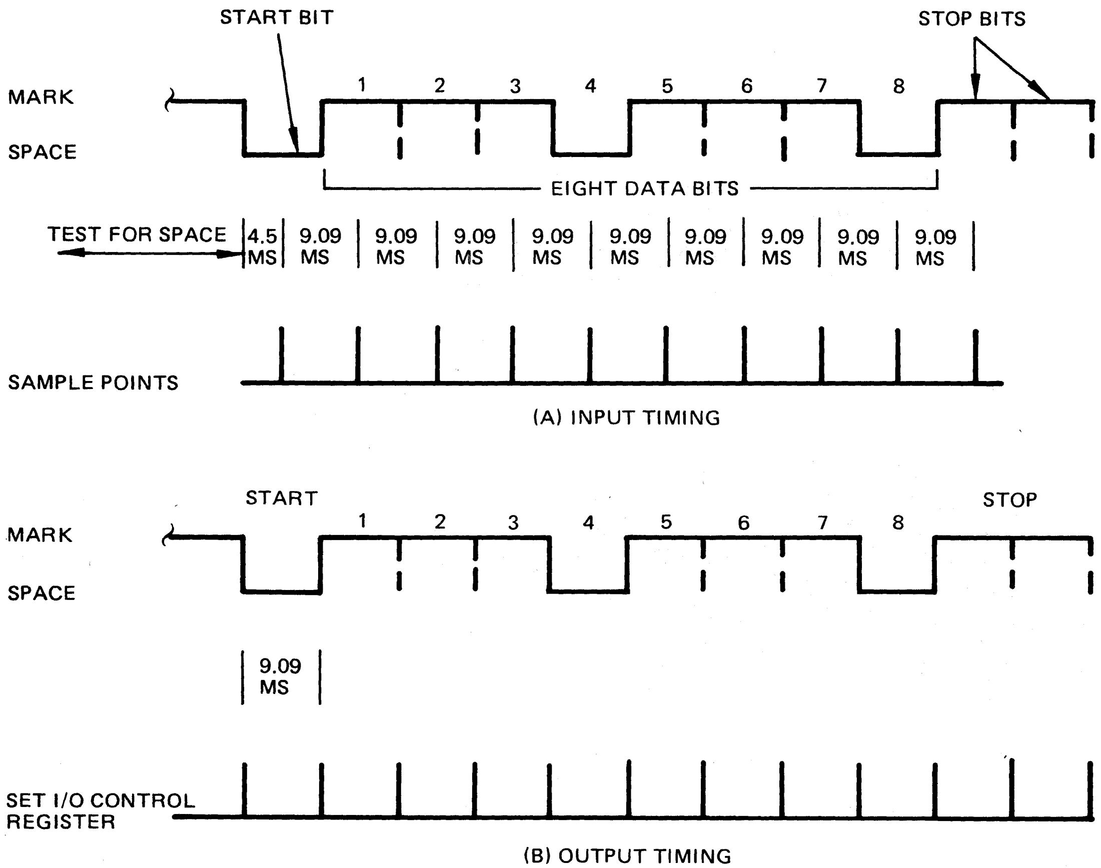


Figure 7-1. Serial I/O Interface Circuit (For ASR 33 TTY)



(A) INPUT TIMING

(B) OUTPUT TIMING

Figure 7-2. Serial I/O Timing

Once a space level is detected the firmware program delays 4.5 milliseconds and samples the input every 9.09 milliseconds, shifting each bit into the least significant byte of the A Register (File Register 4). The initial delay of 4.5 milliseconds, after detecting the leading edge of the start bit, causes sampling to occur in the middle of each bit. The firmware routine exists after eight bits have been assembled.

During an output operation the firmware program sets the I/O control register to the appropriate mark or space condition every 9.09 milliseconds according to the start and stop bits and the data to be serially transmitted. Before the first information bit is transferred, the I/O control register is set to mode 3 to transmit the start bit. The firmware program for transmitting a teletype character remains active for 11 intervals (100 milliseconds) to assure the proper stop interval before the next character is transmitted.

7.2.3 Serial I/O Instructions

Two macro instructions affect the operation of the serial I/O interface:

Input Byte Serially (IBS), and Output Byte Serially (OBS).

The Input Byte Serially instruction transfers an 8-bit character from the teletype into the eight low-order bits of the A Register. The execution of this instruction terminates when a complete teletype character has been received for proper operation, execution of the instruction must be started before the start of the teletype character. Once the instruction is started, the computer becomes tied up until a teletype character is received. The execution time of the instruction extends approximately 84 milliseconds after the leading edge of the teletype character start bit. When the program echoes input characters back to the teletype the effective input rate cannot exceed five characters per second (no input can be handled during the 100 milliseconds required for output).

The Output Byte Serially macro instruction disassembles the eight low-order bits of the A Register and transfers them serially, as a teletype character, through the serial I/O interface. During the execution of this instruction the eight low-order bits of the A Register are set to 1's, the eight high-order bits remain unchanged.

7.2.4 Teletype Interface Connection

The standard Teletype Model ASR-33TY with 20 ma loop interface is directly compatible with all Microdata TTY controllers.

SECTION 8

MICRO-ONE BACKPLANE CONNECTOR SIGNAL LIST

SECTION 8

MICRO-ONE BACKPLANE CONNECTOR SIGNAL LIST

8.1 I/O CONTROLLERS AND DMA INTERFACE SIGNAL LIST

This section contains a signal list of the Micro-One backplane connectors which can be used for I/O controllers and the DMA interface. (See Table 8.1.)

8.2 SERIAL TTY (J2)

1. TTYXI
2. TTY B
3. TTY O
4.
5. TTY G

6 TTY G

8.3 FRONT PANEL (CABLE) CONNECTOR (J3)

1.
2. ES07/
3. ES04/
4. ES05/
5. HLTL/
6. INTF/
7. ES06/
8. RUNF/
9. STPF/
10. CLKF/

Table 8-1. Micro-One Backplane Connector Signal List

PIN	SIGNAL NAME	SIGNAL TYPE	PIN	SIGNAL NAME	SIGNAL TYPE
A1	GND	--	B1	+5VDC	1.
A2	GND	--	B2	+5VDC	1.
A3	SPARE	--	B3	-16.75VDC	1.
A4	-16.75VDC	1.	B4	SPARE	--
A5	+12VDC	1.	B5	SPARE	--
A6	CHP1	2.	B6	SPARE	--
A7	SPARE	--	B7	M04A/	5.
A8	-16.75VDC	1.	B8	SPI1/	3.1KΩ
A9	N07A/	5.	B9	M06A/	5.
A10	OD05/	4.-470Ω	B10	OD01/	4.-470Ω
A11	M01A	5.	B11	A04L/	2.
A12	M02A/	5.	B12	MS1	7.
A13	A00L/	2.	B13	M03A/	5.
A14	A01L/	2.	B14	M00A/	5.
A15	N06A/	5.	B15	RS00	3.-470Ω
A16	L00X	2.	B16	MS2	7.
A17	L11X (GND)	2.	B17	RS04	3.-470Ω
A18	L04X	2.	B18	M05A/	5.
A19	RS01	3.-470Ω	B19	LIOX	
A20	L01X	2.	B20	RTXX/	4.-470Ω
A21	WTXX/	4.-470Ω	B21	RS05	3.-470Ω
A22	L05X	2	B22	CPH2/	4.-470Ω
A23	RS02	3.-470Ω	B23	READ	4.-470Ω
A24	L02X	2.	B24	CGLO/	2
A25	L06X	2.	B25	RS06	3.-470Ω
A26	OD02/	4.-470Ω	B26	OD06/	4.-470Ω
A27	RS03	3.-470Ω	B27	MD07	6.-470Ω
A28	L03X	2.	B28	MD03	6.-470Ω
A29	A02L/	2.	B29	RS07	3.-470Ω
A30	L07X	2.	B30	MD05	6.-470Ω
A31	I02X/	4.-470Ω	B31	I01X/	4.-470Ω
A32	ID04/	3.-1KΩ	B32	ID00/	3.-1KΩ
A33	CPEN/	3.-1KΩ	B33	L08X	2.
A34	A03L/	2.	B34	MD00	6.-470Ω
A35	MD01	6.-470Ω	B35	L09X	2.
A36	MD04	6.-470Ω	B36	RS08	3.-470Ω
A37	RS09	3.-470Ω	B37	OD04	4.-470Ω
A38	EINT/	3.-470Ω	B38	A05L/	2.
A39	SPARE		B39	OD00/	4.-470Ω
A40	AENP		B40	MD06	6.-470Ω
A41	RUNX	2.	B41	A06L/	2.
A42	RS10	3.-470Ω	B42	ECIO/	3.-1KΩ
A43	RS11	3.-470Ω	B43	MD02	6.-470Ω
A44	DMAR/	3.-1KΩ	B44	MRST/	6.-1KΩ
A45	DMAS/	3.-1KΩ	B45	MS3	7.
A46	SP2	SPARE	B46	RTCI	
A47	RS13	3.-470Ω	B47	M07A/	5.
A48	RS14	3.-470Ω	B48	A07L/	2.
A49	CONT (GND)	GND	B49	DMAT/	SPARE
A50	N03A/	5.	B50	IRPY/	3.-1KΩ

Table 8-1. Micro-One Backplane Connector Signal List (continued)

PIN	SIGNAL NAME	SIGNAL TYPE	PIN	SIGNAL NAME	SIGNAL NAME
A51	RS12	3.-470Ω	B51	NOOA/	5.
A52	SELO/	8.	B52	SELI/	8.
A53	N04A/	5.	B53	N01A/	5.
A54	N05A/	5.	B54	PRIN/	8.
A55	PROT/	8.	B55	N02A/	5.
A56	DMAW/	3.-1KΩ	B56	SPI2/	3.- NO PLP.
A57	MBSY	2.	B57	SPIO/	3.- NO PLP.
A58	OD07/	4.-470Ω	B58	OD03/	4.-470Ω
A59	RS15	3.-470Ω	B59	ID05/	3.-1KΩ
A60	ID01/	3.-1KΩ	B60	ID07/	3.-1KΩ
A61	ID06/	3.-1KΩ	B61	IO3X/	4.-470Ω
A62	ID03/	3.-1KΩ	B62	ID02/	3.-1KΩ
A63	-16.75 VDC	1.	B63	-16.75 VDC	1.
A64	GND		B64	+5 VDC	1.
A65	GND		B65	+5 VDC	1.

Signal Types:

1. POWER

2. TTL OUT

3. OPEN COLLECTOR INTO TTL (Pull Up)

4. TTL OPEN COLLECTOR OUT (Pull Up)

5. TR1 STATE OUT

6. BIDIRECTIONAL TTL OPEN COLLECTOR

7. JUMPER

8. THROUGHPUT

Table 8-2. MOS Memory Interface Connector List

J3		J4	
1	M06A/	1	MD04
2	MD00	2	MD02
3	RTXI/	3	MD01
4	M03A/	4	MD06
5	M04A/	5	GND
6	RFSH	6	GND
7	READ/	7	GND
8	MBSY/	8	GND
9	WTXI/	9	GND
10	GND	10	N03A/
11	GND	11	
12		12	
13	CPH2/	13	
14	-16.75V	14	N00A/
15		15	N01A/
16	MD05	16	N04A/
17	MD03	17	N02A/
18	N07A/	18	N05A/
19	M01A/	19	+5V
20	M02A/	20	+5V
21	MSEX/		
22	M00A/		
23	N06A/		
24	GND		
25	M05A/		
26	GND		
27	RTXX/		
28	WTXX/		
29	-16.75V		
30	-16.75V		
31	READ		
32	+5V		
33	+5V		
34	MD07		

SECTION 9

I/O INTERFACE SIGNAL GLOSSARY

SECTION 9

I/O INTERFACE SIGNAL GLOSSARY

This section contains a glossary of the signals used to interface byte I/O controllers and DMA Port interfaces to the Micro-One computer. The list is arranged alphabetically by signal mnemonic. The origin for each signal (computer or controller), the connector pin number, and its function, are provided in Table 9-1.

NOTE: A slash (/) at the end of a signal or line mnemonic denotes that the line is low when the function specified by the mnemonic is occurring.

Table 9-1. I/O Interface Signal Glossary

SIGNAL MNEMONIC	PIN NO.	FUNCTION	ORIGIN	
			CPU	CONT
CPH1	A6	Processor Clock. 5.0 MHz square wave.	X	
CPH2/	B22	Processor Clock. Inverted version of CPH1, delayed 33 nsec.	X	
DMAR/	A44	DMA Request. DMAR/initiates a memory cycle for DMA.		X
DMAS/	A45	DMA Select. DMAS/ selects memory for a DMA operation by enabling DMA controller access to memory.		X
DMAW/	A56	DMA Write. Causes the data byte on memory data lines MD00 through MD07 to be written into memory.		X
ECIO/	B42	Concurrent I/O Request. Low signal from I/O device requesting a concurrent I/O transfer. ECIO/ appears in CPU as bit 3 of File Register 0 where it acts as an interrupt to the macroprogram and initiates a firmware routine for handling a concurrent transfer.		X
EINT/	A38	External interrupt. Low signal from I/O device requesting interruption of the macroprogram. EINT/ appears in CPU as bit 7 of File Register 0 where it initiates a firmware routine for transferring control to a macroprogram interrupt handling routine.		X

Table 9-1. I/O Interface Signal Glossary (continued)

SIGNAL MNEMONIC	PIN NO.	FUNCTION	ORIGIN	
			CPU	CONT
ID00/	B32	Data Input Bit 0. Connects to CPU via B Bus gating.		X
ID01/	A60	Data Input Bit 1. Connects to CPU via B Bus gating.		X
ID02/	B62	Data Input Bit 2. Connects to CPU via B Bus gating.		X
ID03/	A62	Data Input Bit 3. Connects to CPU via B Bus gating.		X
ID04/	A32	Data Input Bit 4. Connects to CPU via B Bus gating.		X
ID05/	B59	Data Input Bit 5. Connects to CPU via B Bus gating.		X
ID06/	A61	Data Input Bit 6. Connects to CPU via B Bus gating.		X
ID07/	B60	Data Input Bit 7. Connects to CPU via B Bus gating.		X
I01X/	B31	Bit 1 of I/O Control Register	X	
I02X/	A31	Bit 2 of I/O Control Register	X	
I03X/	B61	Bit 3 of I/O Control Register	X	
MBSY	A57	Memory Busy. MBSY is a status signal from memory indicating that memory is busy.	X	
MD00	B34	Bidirectional data line (bit 0) to memory for DMA operation.	X	X
MD01	A35	Bidirectional data line (bit 1) to memory for DMA operation.	X	X
MD02	B43	Bidirectional data line (bit 2) to memory for DMA operation.	X	X
MD03	B28	Bidirectional data line (bit 3) to memory for DMA operation.	X	X
MD04	A36	Bidirectional data line (bit 4) to memory for DMA operation.	X	X

Table 9-1. I/O Interface Signal Glossary (continued)

SIGNAL MNEMONIC	PIN NO.	FUNCTION	ORIGIN	
			CPU	CONT
MD05	B30	Bidirectional data line (bit 5) to memory for DMA operation.	X	X
MD06	B40	Bidirectional data line (bit 6) to memory for DMA operation.	X	X
MD07	B27	Bidirectional data line (bit 7) to memory for DMA operation.	X	X
M00A/	B14	Bit 0 of upper half of memory address (used by DMA)	X	X
M01A/	A11	Bit 1 of upper half of memory address (used by DMA)	X	X
M02A/	A12	Bit 2 of upper half of memory address (used by DMA)	X	X
M03A/	B13	Bit 3 of upper half of memory address (used by DMA)	X	X
M04A/	B7	Bit 4 of upper half of memory address (used by DMA)	X	X
M05A/	B18	Bit 5 of upper half of memory address (used by DMA)	X	X
M06A/	B9	Bit 6 of upper half of memory address (used by DMA)	X	X
M07A/	B47	Bit 7 of upper half of memory address (used by DMA)	X	X
N00A	B51	Bit 0 of lower half of memory address (used by DMA)	X	X
N01A/	B53	Bit 1 of lower half of memory address (used by DMA)	X	X
N02A/	B55	Bit 2 of lower half of memory address (used by DMA)	X	X
N03A/	A50	Bit 3 of lower half of memory address (used by DMA)	X	X
N04A/	A53	Bit 4 of lower half of memory address (used by DMA)	X	X

Table 9-1. I/O Interface Signal Glossary (continued)

SIGNAL MNEMONIC	PIN NO.	FUNCTION	ORIGIN	
			CPU	CONT
N05A/	A54	Bit 5 of lower half of memory address (used by DMA)	X	X
N06A/	A15	Bit 6 of lower half of memory address (used by DMA)	X	X
N07A/	A9	Bit 7 of lower half of memory address (used by DMA)	X	X
MRST/	B44	Master Reset. Signal used to clear all control flip-flops in controllers.	X	
OD00/	B39	Output Data Bit 0	X	
OD01/	B10	Output Data Bit 1	X	
OD02/	A26	Output Data Bit 2	X	
OD03/	B58	Output Data Bit 3	X	
OD04/	B37	Output Data Bit 4	X	
OD05/	A10	Output Data Bit 5	X	
OD06/	B26	Output Data Bit 6	X	
OD07/	A58	Output Data Bit 7	X	
PRIN/	B54	Priority In. Low signal from preceding controller indicating I/O controller has priority to request interrupt operation. PRIN/ is passed serially from CPU to controller to controller, etc.	X	X
PROT/	A55	Priority Out. Low signal originating in CPU and passed from controller to controller carrying interrupt request priority. PROT/ becomes PRIN/ on input to each controller.	X	X
RTCI	B46	Real-Time Clock input, for user supplied RTC frequencies other than the standard power line freq (120 Hz)		X
SELI/	B52	Select in. A low signal from preceding controller which occurs after an interrupt or concurrent I/O request during acknowledge time.	X	X

Table 9-1. I/O Interface Signal Glossary (continued)

SIGNAL MNEMONIC	PIN NO.	FUNCTION	ORIGIN	
			CPU	CONT
		SELI/ indicates controller has priority to place its address on data lines for an interrupt or concurrent I/O operation. SELI/ is passed serially from CPU to controller to controller, etc.		
SELO/	A52	Select Out. Low signal originating in CPU and passed from controller to controller which passes select priority from controller to controller. SELO/ becomes SELI/ at input to each controller.	X	X
SPARE/	B49	Spare. Available for use by DMA controller or other customer-designed option as an internal CPU interrupt. Enters CPU at internal status register bit 1 where it is OR'ed with other internal status bits, and ultimately sets bit 4 of file register 0.		X

PANEL TERMS:

CLKF/, STPF/, RUNF/ INTF/	Panel CPU Control Switches
ES04/-ES07/	Panel Sense Switches
ANEP/	Enable Console Switches
RUN/	CPU in RUN MODE

TTY TERMS:

TTYXI	Serial TTY Input
TTY B	Serial TTY Reference Voltage
TTY O	Serial TTY Output
TTY G	Serial TTY Ground

SECTION 10

OPERATOR CONTROLS

SECTION 10

OPERATOR CONTROLS

10.1 CONSOLES

Two control console options are available: system console and basic console. These consoles differ in their number of displays and controls. This range of consoles permits the user to tailor the cost to meet the control and display capability required for a particular application. The system console is shown in Figure 10-1, and the basic console in Figure 10-2.

10.1.1 System Console (Standard 1600 System Console)

When using a standard Micro One backplane, the 1600 system panel can be used providing complete control and display facilities. It is primarily used for maintenance, system and firmware checkout. The console provides for display of the micro-one registers in addition to the functions of the basic console. The features include:

- Run and halt indicators
- Display of A bus
- Display of M, N, and L registers
- Display of read-only-memory output
- Four sense switches
- Six control switches, including run, step, interrupt, clock and reset
- Manual command execution

10.1.2 Basic Console

The basic console provides minimal control capability and is designed for dedicated system application where operator control is not required. The features include:

- Run and halt indicators
- Four sense switches
- Five control switches including run, step, interrupt, clock and reset

10.2 DISPLAYS ON SYSTEM CONSOLE

The following paragraphs define the usage of the displays on the system console.

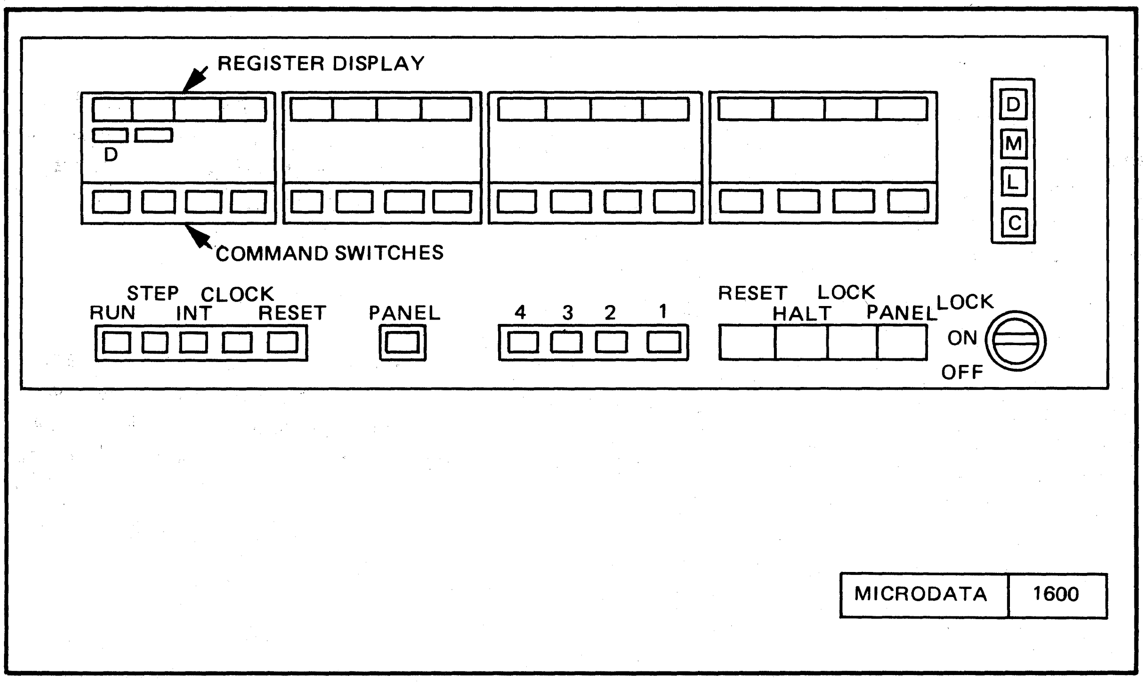


Figure 10-1. MICRO 1600 System Console

Figure 10-2. Micro One Basic Console

10.2.1 Data Display

The 16-bit data indicators (16 lamps on console) display the 8-bit A bus, Memory Address, 16-bit Control Memory output, or 12-bit control Memory Address as selected by the Display Selector switches.

10.2.2 RUN

The RUN indicator is on when the processor is running.

10.2.3 HALT

The HALT indicator is on when the power is applied and the processor is not running.

10.2.4 LOCK

The LOCK indicator is on when the panel is disabled.

10.2.5 PANEL

The PANEL indicator is on when the command switches are enabled and substituting for the control memory.

10.3 SWITCHES ON SYSTEM CONSOLE

System console switches are defined as follows:

10.3.1 Display Selector

The 4 interlocked switched located in the upper right corner select 1 of the 4 displays as follows:

D - Data: This 8-bit display is the processor's A bus. The data on the A bus when the processor is halted and in the panel enable mode depends on the setting of the command switches.

M - Memory Address: This 16-bit display is of the memory address lines. This is normally the contents of the M and N registers.

L - Control Memory Address: This 10-bit display is the contents of the L register.

C - Control Memory: This 16-bit display is of the output of the control memory. When the processor is halted the R register contains the same data.

10.4 COMMAND SWITCHES

These 16 locking switches are substituted for the control memory when the PANEL switch is in the down position. When the processor is halted, the switch setting is constantly clocked into the R register and depressing the CLOCK switch causes the command set in the switches to be executed. The command may also be executed repeatedly by depressing the RUN switch. These switches are used to gate registers onto the A bus for display and for entering data into register.

10.4.1 Panel Switch

This locking switch selects the source of commands. When in the normal up position the control memory is used and when in the down position the 16 command switches on the panel are substituted for the control memory.

10.4.2 Sense Switches

The four locking sense switches are available on the control panel. These switches may be read by an Enter Sense Switch command.

10.4.3 Run

This momentary contact switch places the processor in the run mode causing it to execute microcommands.

10.4.4 Step

This momentary contact switch places the processor in the run mode and as long as the switch is depressed causes an internal interrupt. The halt internal interrupt is bit 7 of the internal status. This switch is normally microprogrammed to cause a processor halt. Since the processor is forced to run when the switch is depressed, the computer can be microprogrammed to cause a single macro instruction to be executed.

10.4.5 Interrupt

This momentary contact switch places the processor in the run mode and causes an internal interrupt. The console interrupt is bit 0 of the internal status. This switch is normally microprogrammed to cause a console interrupt.

10.4.6 Clock

This momentary contact switch causes the processor to execute a single microcommand. If the processor is running at the time the switch is depressed, the processor will come to a forced halt following the current microcommand execution.

10.4.7 Reset

This momentary contact switch halts the processor and clears the L register, I/O control register and other control flip-flops. The reset is made available to I/O devices. Since the current microcommand execution will not be completed, the computer should not be stopped by this switch.

10.4.8 On-Off-Lock

A 3-position key lock switch enables and disables the panel. The key can be removed in any position. In the OFF position, the panel is inactive. In the ON position, the panel is active. In the LOCK position, power remains on, but the panel switches are not active except for the sense switches.

10.5 ADDRESS SYNC

A sync jack is mounted on the rear of the front panel for maintenance purposes. A positive pulse of 200 nsec duration is obtained when the contents of the L register are the same as the address set into command switches 14-0.

10.6 REGISTER DISPLAY AND ENTRY

Use of the register Display and Entry is discussed in the following paragraphs.

10.6.1 Display

The processor registers can be displayed directly be selecting the proper display selector or indirectly by use of commands set into the command switches to cause the register to be gated to the A bus where it can be displayed by selecting 'D'.

The R, U, MD and OD registers cannot be displayed, but the R register will hold the same information as on the R bus when the processor is halted. The M, N and L registers can be displayed by selecting them with the display selector.

The file registers, T register and LINK can be displayed indirectly by setting the commands shown below into the command switches and selecting the data display (A bus). Panel switch must be on.

Register	Command Setting
Selected File Register X	CX00
T Register	B020
LINK (AL)	B080
LINK (ML)	B082

10.6.2 Enter

Information can be entered into a register by executing a command from the panel. This requires turning on the PANEL switch, setting the command into the command switches and pressing the CLOCK switch. In addition, control functions such as interrupt enable or the file select can be performed by executing the appropriate command. The commands for placing the literal 'ZZ' in a register are shown as follows:

Register		Command
T		11ZZ
M		12ZZ
N		13ZZ
U		16ZZ
File Register X		2XZZ
L	(Page 0)	14ZZ
	(Page 1)	15ZZ
	(Page 2)	1CZZ
	(Page 3)	1DZZ

10.7 OPERATING PROCEDURES — SYSTEM CONSOLE

The following list of commands is a minimum that should be tried out when first becoming acquainted with the Micro-One.

1. Loading and stepping the L register

   a. Load L

      1) Set CLOCK, RESET

      2) Set PANEL on

      3) Select L display

      4) Set the following commands into the command switches and press the CLOCK switch one for eachｏ On the Micro-One manual loading of L causes L + 1 to be loaded into L.

Settings Switches	Display
14A9	0AA
1454	055
15FE	1FF
1C10	211
1DED	3EE

b. Step L

1) Set PANEL off

2) Set RESET

3) Select L display

4) Each time the CLOCK switch is pressed, the L count should increment, skip, or jump. If no ROM board is plugged in, the L count will step.

2. Test M and N

1) Set PANEL on

2) Display to M or N

3) Set the following command into the command switches and press the clock switch once for each.

1255	Load M	M = 55
13AA	Load N	N = AA, M = 0

Try other values and repeat

3. Test ROM and L register (with Micro One/10 firmware)

1) Set PANEL off

2) Set RESET

3) Select L, C

4)

L	C	
000	BF02	
001	2B00	Repeatedly press the CLOCK
002	2A00	
003	4010	

After this, the L value depends on computer register states, because of conditional skips and jumps.

4. Test the T register

1) Set PANEL on

2) Set DISPLAY to D (A bus)

3) Set the following sequences into the command switches and press the CLOCK switch.

11AA	CLOCK	Load T
B020		Display T = AA with copy T
1155	CLOCK	Load T
B020		Display T = 55 with copy T

Try other values and repeat

5. Test the File Registers

a. Load and Read each File.

1) Set PANEL on

2) Set DISPLAY to D (A bus)

3) To load file f, set the following command into the command switches and press the clock switch once:

2fXX   XX = data value

4) To read file f, set the following command into the command switches:

cf00

Load file f	2fXX	clock
Read file f	cf00	Do not clock

SECTION 11

MICRO-ONE CPU OPERATIONAL DESCRIPTION

SECTION 11

MICRO ONE CPU OPERATIONAL DESCRIPTION

11.1 GENERAL

This section describes, separately, each major function of the Micro-One processor. The individual descriptions include a verbal portion and a block diagram. The block diagrams contain references to the accompanying annotated logic schematics. The references identify integrated circuit chips by logic page number and chip number.

The breakdown of the operational description is as follows:

1. Arithmetic Logic Unit and Multiplexer
2. T Register
3. File Register
4. R and U Registers
5. L Register
6. Condition and Link Logic
7. Memory Address Registers (M and N)
8. Destination Register Clock Logic
9. Command Decode ROMs
10. Programmed Input/Output
11. Interrupts
12. Memory Sequencer
13. Computer Clock and Run Control
14. Computer Start Logic
15. Automatic Power Fail and Power On Detection Function

11.1.1 Arithmetic Logic Unit and Multiplexer

The Arithmetic Logic Unit (ALU) shown in figure 11-1 consists of an 8-bit, 2-input, programmable arithmetic logic function generator; an 8-bit latch to provide A bus and $\overline{A}$ bus outputs as well as to prevent logical racing. It further consists of logic to determine carry-in; and an 8-bit, 8-channel multiplexer to select 1 of 8 possible inputs to one port (B) of the ALU function generator. The function generator consists of 2 74181 chips, which

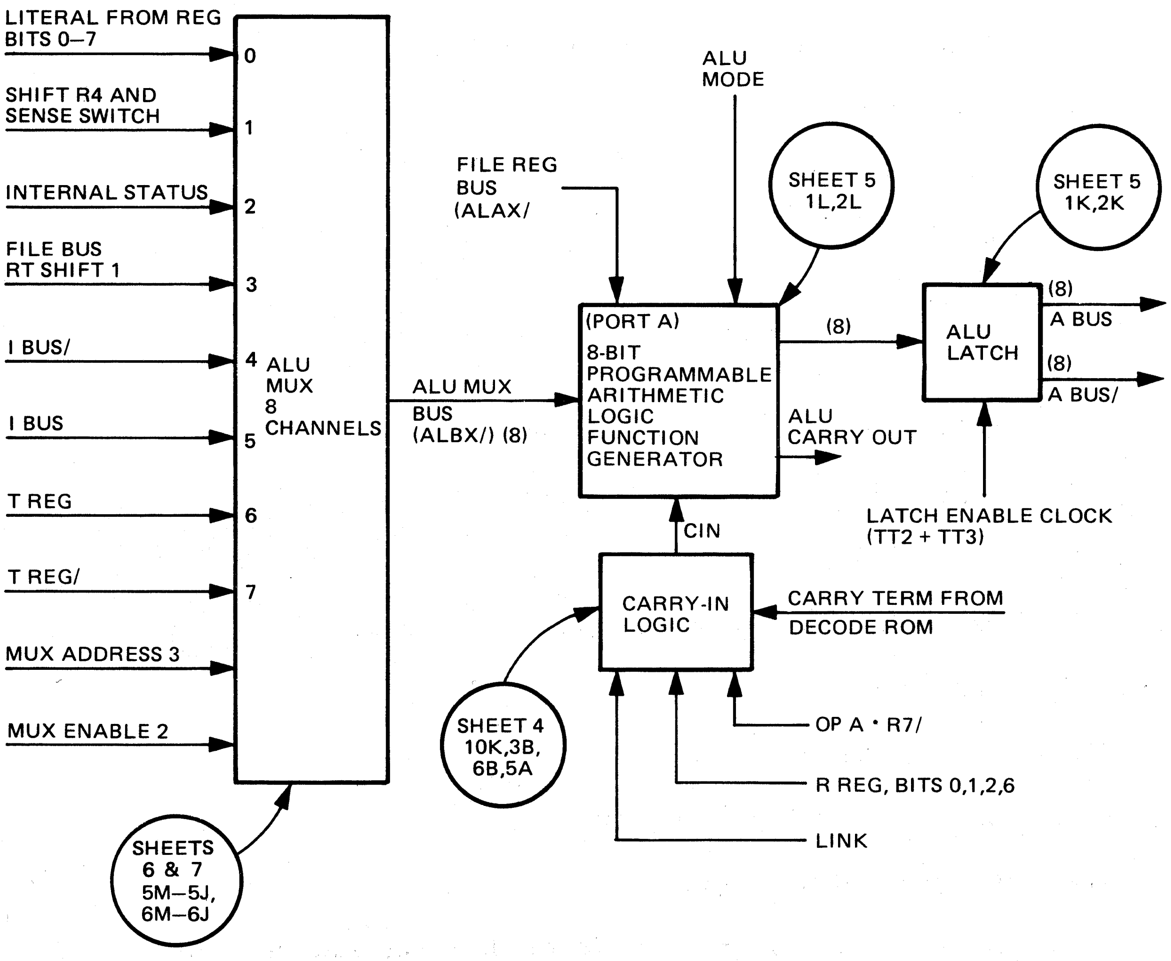


Figure 11-1. Arithmetic/Logic Unit Block Diagram

have 16 arithmetic and 16 logic functions, selectable by 5 ALU mode input lines (ALS1, ALS2, ALS3, ALS4, ALM). One input channel, A, consists of the file register bus and the other, B, is from the ALU MUX.

The ALU MUX channels are as follows:

Channel 0 - 8-Bit literal from R register, used for all literal class commands such as literal-to-register, literal-to-file, add-literal-to-file, and skip commands.

Channel 1 - Shift file right 4 to lower 4 bits, and 4 sense switches to upper 4 bits. For each of these, the alternate 4 bits are equal to 1's, which value is achieved by disabling either the upper or lower 4 bits of the ALU MUX (with EALBL or EALBU).

Channel 2 - Internal status consisting of console interrupt, DMA termination, real-time clock, step switch input, power fail interrupt and 3 spare inputs.

Channel 3 - Shift right input. Since the ALU function generator has the capability of left shifts only, the 8 lines from the file register are entered into the MUX and displaced one bit position to the right to generate right shifts.

Channel 4 - Input bus/.

Channel 5 - Input bus. Whenever T register is designated as a data source by a firmware command and I/O control bit IO3X is set, the input bus is actually selected by the ALU MUX instead of T. When T/ is designated, input bus/ is selected.

Channel 6 - T Register.

Channel 7 - T Register/. T or T/ may be designated by firmware commands as data sources.

The MUX addresses, and ALU modes for implementation of the firmware commands are shown in Table 11-1.

11.1.2 Carry In

Carry In (CIN) to the ALU is determined by the firmware operation as follows:

Firmware Operation	Carry In
1. Literal to register	Not applicable
2. Add to file	0
3. Test zero and test NOT zero	Not applicable

Table 11-1. ALU MUX Addresses, ALU Modes, and Carry In

Firmware Command		ALU MUX Address (3 lines)		ALU Mode (5 lines)		
Mode	Function	Address	MUX Channel	Mode (HEX)	Function	Carry In
1rxx	Literal to register	0	Literal	'1A'	B	N/A
2fxx	Literal to file	0	Literal	'1A'	B	N/A
3fxx	Add literal to file	0	Literal	'09'	A + B + C	0
4fxx	Test if Zero	0	Literal	'1E'	A ∧ B	N/A
5fxx	Test if not Zero	0	Literal	'1E'	A ∧ B	N/A
6fxx	Compare	0	Literal	'09'	A + B + C	0
7f1r	Enter Sense Switches	1	Enter Sense Switches	'15'	B/	N/A
7f2r	Shift Right 4	1	Shift Right 4	'15'	B/	N/A
7f4r	Enter Internal Status	2	Internal Status	'15'	B/	N/A
7f7o	Enter Console Switches	7	T/	'1F'	A	N/A
7f8r	Clear I/O, IO3x = 0	7	$\overline{T}$/	'1F'/	A	N/A
7f8r	Clear I/O, IO3x = 1	5	Input bus	'1E'	A ∧ B	
7F9r to 7fFr	Set I/O } IO3x = 0 States } IO3x = 1	7/5	$\overline{T}$//Input bus	'1F'/'1E'	A/A ∧ B	N/A
8f0r	File + 0 → r	7	NA	'0F'	A + C	0
8f2r	File + T → r	6/5	T/Input bus	'09'	A + B + C	0
8f4r	File + 1 → r	7	NA	'0F'	A + C	1
8f6r	File + T + 1 → r	6/5	T/Input bus	'09'	A + B + C	1
8f8r	File + Link → r	7	NA	'0F'	A + C	Link

A = FILE BUS INPUT
B = ALU MUX INPUT
+ = ADD
− = SUBTRACT

C = CARRY
∧ = AND
∨ = OR
∀ = EXCLUSIVE OR

A/ = A NOT
T = T REGISTER

Table 11-1. ALU MUX Addresses, ALU Modes, and Carry In (continued)

Firmware Command		ALU MUX Address (3 lines)		ALU Mode (5 lines)		
Mode	Function	Address	MUX Channel	Mode (HEX)	Function	Carry In
9f0r	File - 0 → r	7	NA	'00'	A - C/	1
9f2r	File - T → r	6/5	T/Input bus	'06'	A - B - C/	1
9f4r	File - 1 → r	7	NA	'00'	A - C/	0
9f6r	File - T - 1 → r	6/5	T/Input bus	'06'	A - B - C/	0
9f8r	File - Link → r	7	T/	'00'	A - C/	Link/
Af0r	read mem ; f → r	7	T/	'FF'	A	N/A
Af4r	read mem ; f - 1 → r	7	T/	'00'	A - C/	0
Af4m	read mem ; f - 1 + L → r	7	T/	'00'	A - C/	Link/
Af8r	read mem ; f + L → r	7	T/	'0F'	A + C	Link
AfCr	read mem ; f + 1 → r (L = link)	7	T/	'0F'	A + C	1
Bf0r	0 → r	7	NA	'0F'	A + C*	0
Bf2r	(T) → r	6/5	T/Input bus	'09'	A + B + C*	0
Bf4r	1 → r	7	NA	'0F'	A + C*	0
Bf6r	(T) + 1 → r	6/5	T or Input bus	'09'	A + B + C*	1
Bf8r	(T) + (L) → r	6/5	T or Input bus	'09'	A + B + C*	Link

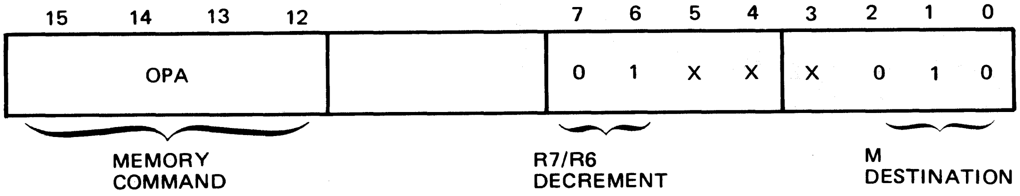
*FILE OUTPUT DISABLED

Table 11-1. ALU MUX Addresses, ALU Modes, and Carry In (continued)

Firmware Command		ALU MUX Address (3 lines)		ALU Mode (5 lines)		
Mode	Function	Address	MUX Channel	Mode (HEX)	Function	Carry In
Cf0r	(f) → r	7	NA	'1F'	A	N/A
Cf2r	(f) OR (T) → r	6/5	T or Input bus	'1B'	$A \vee B$	N/A
Cf4r	(f) OR (T/) → r	7/4	T/ or Input bus/	'1B'	$A \vee B$	N/A
Cf6r	'FF' → r	7	NA	'03'	'FF'	N/A
Df0r	(f) EXOR 0 → r	7	NA	'1F'	A	N/A
Df2r	(f) EXOR (T) → r	6/5	T or Input bus	'19'	$A \veebar B$	N/A
Df4r	(f) EXOR (T/) → r	7/4	T/ or Input bus	'19'	$A \veebar B$	N/A
Df6r	(f) EXOR 1 → r	7	NA	'10'	A/	N/A
Ef0r	(f) AND 0 → r	7	NA	'1C'	0	N/A
Ef2r	(f) AND (T) → r	6/5	T or Input bus	'1E'	$A \wedge B$	N/A
Ef4r	(f) AND (T/) → r	7/4	T/ or Input bus/	'1E'	$A \wedge B$	N/A
Ef6r	(f) AND 1 → r	7	NA	'1F'	A	N/A
Ff0r	Shift File Left Enter 0	3	Shift R1	'0C'	A + A + C	0
Ff4r	Enter 1	3	Shift R1	'0C'	A + A + C	1
Ff8r	Enter Link	3	Shift R1	'0C'	A + A + C	Link
Ff7r	Shift File Right Enter 0	3	Shift R1	'15'	B	N/A
Ff6r	Enter 1	3	Shift R1	'15'	B	N/A
FfAr	Enter Link	3	Shift R1	'15'	B	N/A

Firmware Operation	Carry In
4. Op Code 7	Not applicable
5. Op Code 8, 9, B, F	Determined by Carry In (CIN/) ROM which decodes the Op Code (bits 15-12) and the C field (bits 5, 6, 7) to result in CIN/ of LINK/, 0, or 1 as shown in Table 11-1
6. Op Codes C, D, E	Not applicable
7. Op Code A	When Op Code A (Memory) is processed, CARRY IN is determined by the CIN/ ROM for all C field conditions, except for C = 10XX in which case a special CIN source is selected when the M register is designated

The special Carry In for Memory command with decrement, and M destination, is as follows:



For this condition the Carry In term (CIN/) = LINK $\wedge$ $\underbrace{\text{R7/} \wedge \text{R6}}_{\text{DECREMENT}}$ $\wedge$ $\underbrace{\text{R2/} \wedge \text{R1} \wedge}_{\substack{\text{M} \\ \text{Destination}}}$ RO $\wedge$ $\underbrace{\text{OPA}}_{\substack{\text{Memory} \\ \text{Command}}}$

11.2 T. REGISTER

The T register is the hub of data flow through the Micro-One. It is a primary input to the ALU via the ALU MUX. It drives the Memory Data bus on Memory Write functions, and it drives the Output Data bus. T is loaded from either the CPU A bus on firmware operations, or the Memory Data bus on Memory Read functions. The T register is illustrated in Figure 11-2.

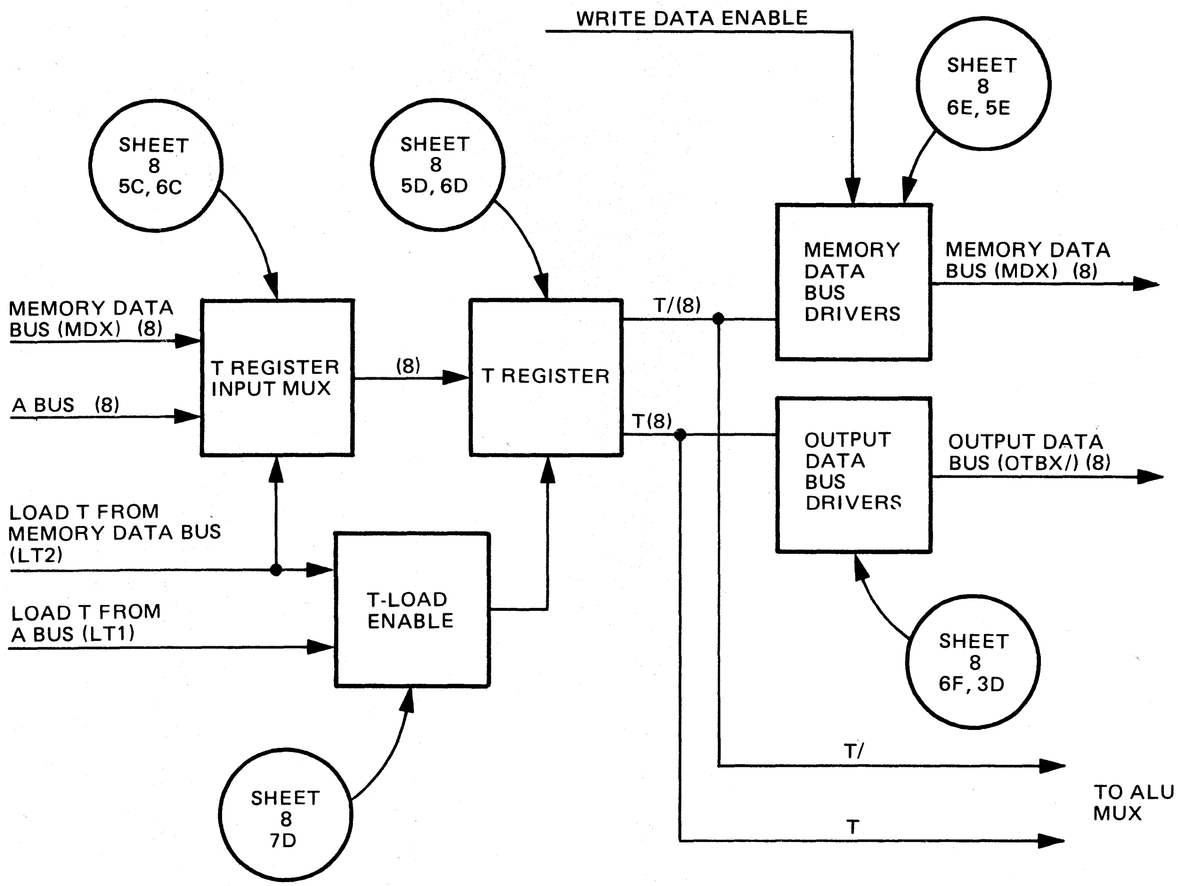


Figure 11-2. T-Register Block Diagram

Gated drives, activated by Memory Write (WRIT), drive the Memory Data bus while non-gated drivers drive the Output Data bus. T and T complement go directly to the ALU MUX.

The input to T is selected by a 2-channel MUX. Loading of T is strobed by either the firmware generated T destination strobe (LT1), or by a Load T strobe derived from the Memory Read strobe (LT2). The T register is a D type latch in which the output follows the input as long as the strobe is present.

11.3 FILE REGISTERS

There are 16 8-bit registers designated as File Registers in the Micro-One. See Figure 11-3. Of the 16, 15 are general purpose random-access and the remaining one (File 0) is for condition flags, interrupts and I/O flags. Two 16 X 4-bit chips are used for the general purpose files (address 0 being nonaccessible). An array of 2-input gates is used for File 0. When File Address 0 is selected, a File 0 Enable is generated by the File Control logic, otherwise a File Chip Enable is generated. The general purpose files and File 0 are tied together, open collecter, to form the File Register bus.

11.4 R AND U REGISTERS

The R register contains the 16-bit firmware command being executed. It is loaded at the beginning of each cycle by the TT4 phase clock. The R register is always loaded, even if the CPU is halted, except when Thold is active. The R and U registers are shown in Figure 11-4.

Thold temporarily prevents execution of a command; therefore, the command must be saved in R until Thold goes inactive and then is executed.

If the firmware command from the Control ROM has either a 0 Op Code (Execute) or is an operate command with 8 or higher op code and has a 7 destination, the contents of the U register are ORed with the upper 8 bits of the command prior to loading the R register. This is done by use of a 2-channel multiplexing latch for the upper 8 bits of R, which selects either the ROM output directly or the ROM output ORed with U.

The U register is loaded by a firmware command from the A bus.

11.5 L REGISTER

The L register, shown in Figure 11-5, is the Control ROM Address Register containing 10 bits, thus it can address 1K words of ROM or 4 pages. The lower 8 bits of the L register consist of a counter which can be clocked, parallel loaded, or reset. The upper two bits are latches which can be individually loaded, or reset. The lower 8 bits address a page size of ROM and cycle independently of the upper two bits when in the L count mode.

Unless L is loaded, L is incremented 1 count for each firmware instruction. Counting is suspended when in Halt or Thold. During idle periods, the counting continues.

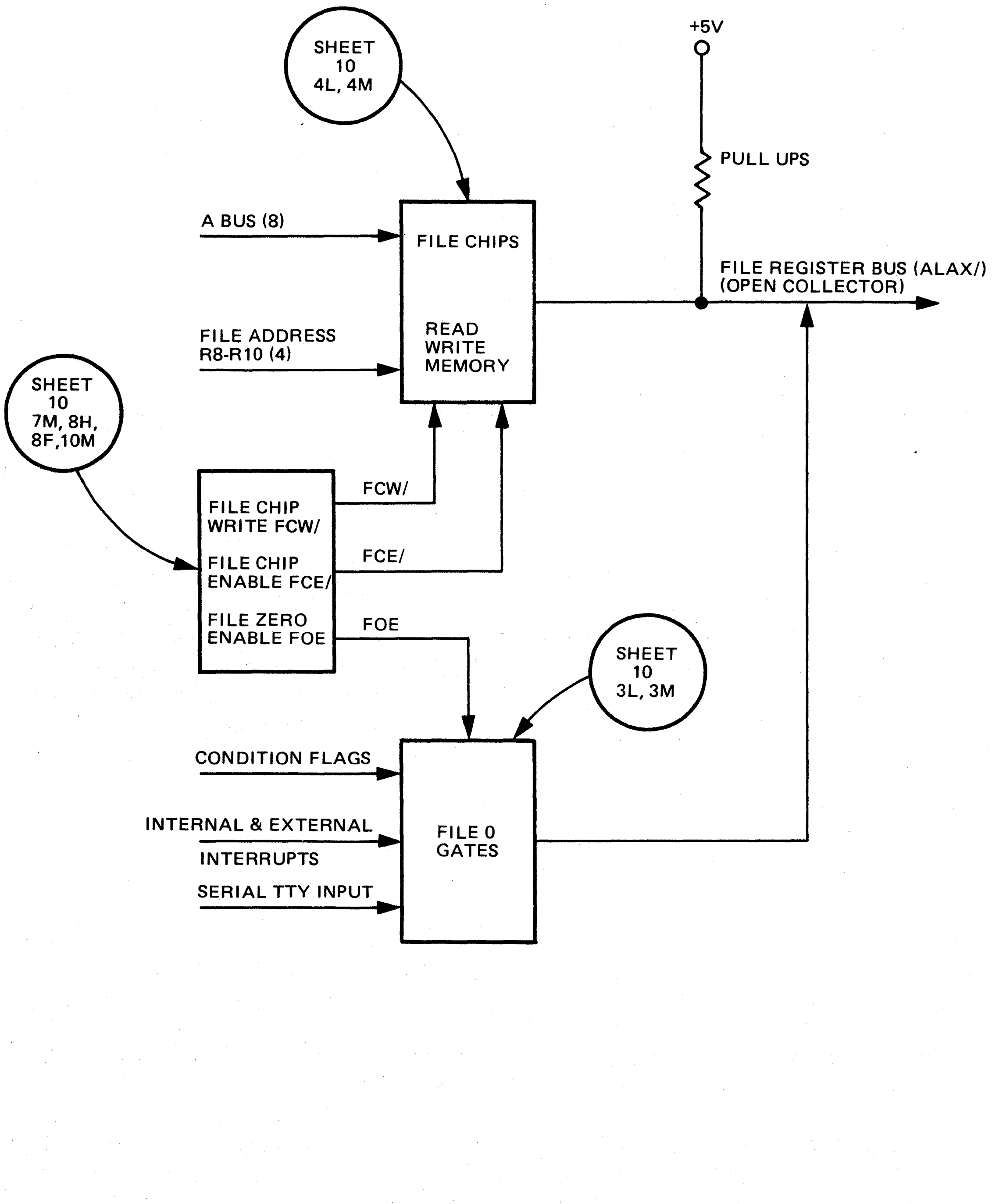


Figure 11-3. File Registers Block Diagram

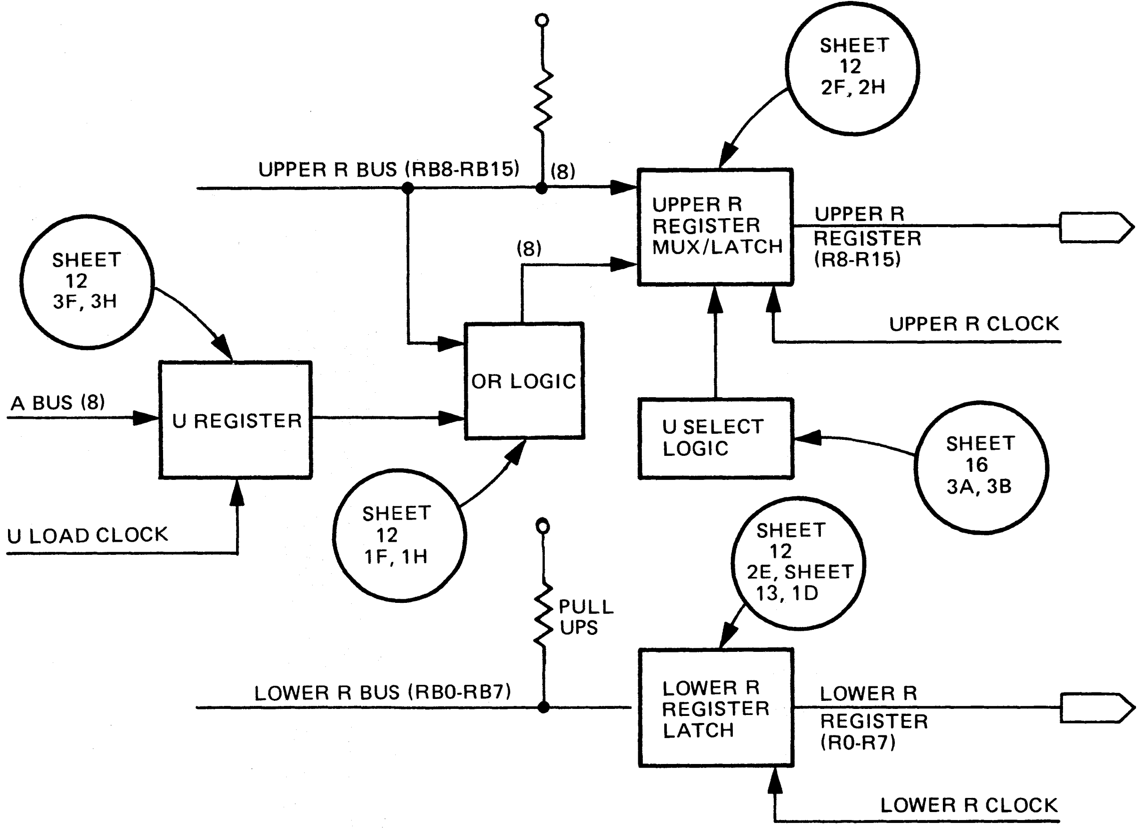


Figure 11-4. R and U Register Block Diagram

L is parallel loaded by the following commands:

Literal to L firmware command

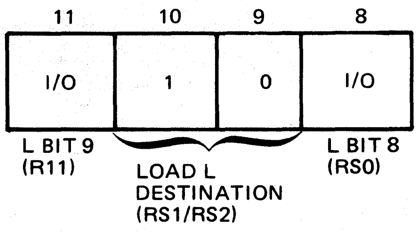
1 4 X X

1 5 X X

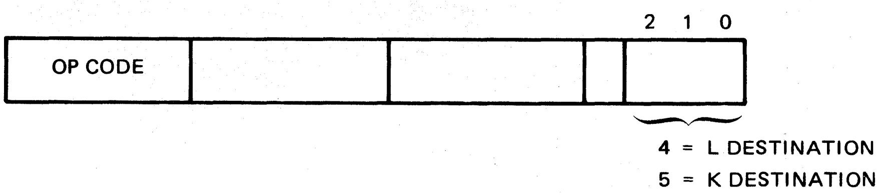
1 C X X

1 D X X

The literal is loaded into the lower 8 bits via the A bus. The upper 2 bits of L are loaded by 4, 5, C, or D as follows:



11.5.1 L or K Destination in an Operate Command



The lower 8 bits of L are loaded directly from the A bus. Bit 8 is loaded from bit 0 of the destination via RSO, providing odd or even page selection. Bit 9 is not affected by this operation, therefore the effective jump is confined to the half of the ROM at which the previous address was located.

When L is loaded, or a skip operation occurs, the instruction immediately after the load or skip is inhibited by the IDLE function. This is necessary because of the "look ahead" feature of the CPU during execution of a microcommand, the next microcommand is being fetched from the ROM.

When L is loaded from the system panel using the load L instruction, setting of IDLE causes clocking of L one count higher than the value set on the front panel.

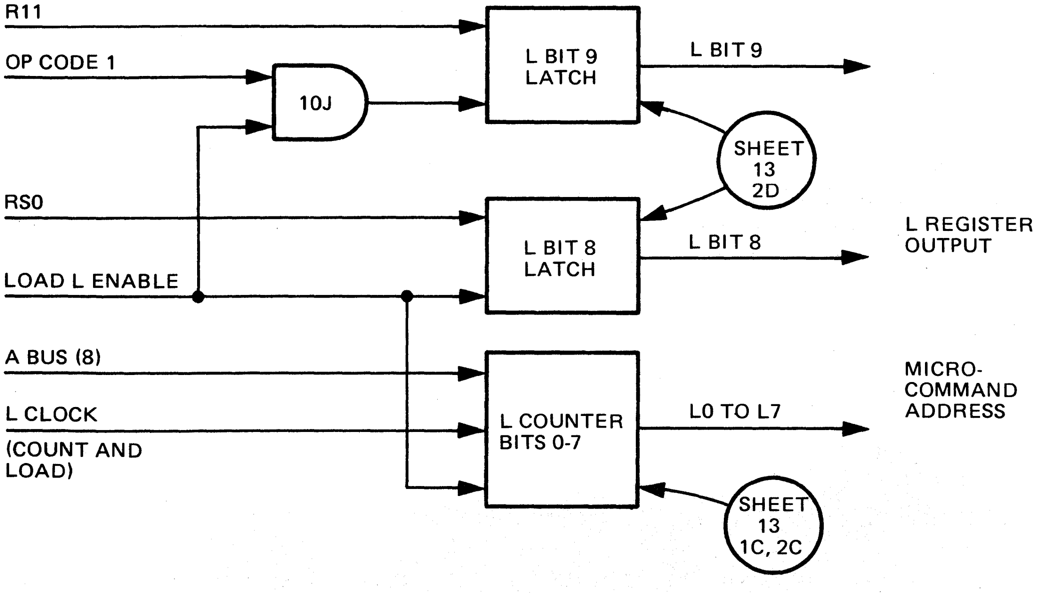


Figure 11-5. L Register Block Diagram

11.6 CONDITION AND LINK LOGIC

The Micro-One condition and link logic is shown in Figure 11-6. There are three ALU operational conditions maintained and utilized in the Micro-One: overflow, negative, and zero. The negative condition is simply bit 7 of the A bus and zero condition is simply the eight input AND of A bus/. Overflow is divided into two categories, arithmetic and shift. For shift overflow, overflow is the same as Link; namely, the bit shifted out of the file (ALA7 for left shift and ALA0 for right shift). For arithmetic operations, overflow is defined as when the carry in to the most significant bit does not equal carry out. Expressed logically:

$$\text{OVFL} = (\text{ALA7} \wedge \text{AB7} \wedge \text{A7/}) \vee (\text{ALA7/} \wedge \text{ALB7/} \wedge \text{A7})$$

In the Micro-One, a combination ROM and 8-channel multiplexer are used to generate both the arithmetic and shift type overflows. The ROM selects one of four overflow sources, based on the firmware command Op code and ALU mode. The four sources are A7, A7/, Shift out bit, or 0.

Zero, Overflow, and Negative condition terms are stored in latches which are updated on command from the firmware. Zero and overflow conditions are also used directly for the firmware skip tests. The zero condition latch has separate update logic because of the requirement for "reset but not set" on linked zero tests over multiple bytes.

Link is determined from either ALU carry out or the shifted out bit from the selected File depending on whether an arithmetic or shift operation is taking place. Link is unconditionally updated on both arithmetic and shift operations except for the Op Code 3 Add to File command for which neither Link or Condition flags are affected.

11.7 MEMORY ADDRESS REGISTERS (M AND N)

The M and N registers, shown in Figure 11-7, are each 8-bit tristate registers which contain the read/write memory addresses while a memory cycle is taking place. M and N are loaded from the A bus by firmware command. Whenever the N register is loaded by a 13XX command the M register is cleared to 0. This is accomplished in the Micro-One by disabling the M register output on a load N command without actually clearing M.

11.8 DESTINATION REGISTER CLOCK LOGIC

The Destination Register Clock Logic, shown in Figure 11-8, generates the load register strobes for M, N, U, T, L, and interrupt enable registers. Also, a Load 0 strobe is generated for external use.

Destination addresses are located in two places in the firmware command.

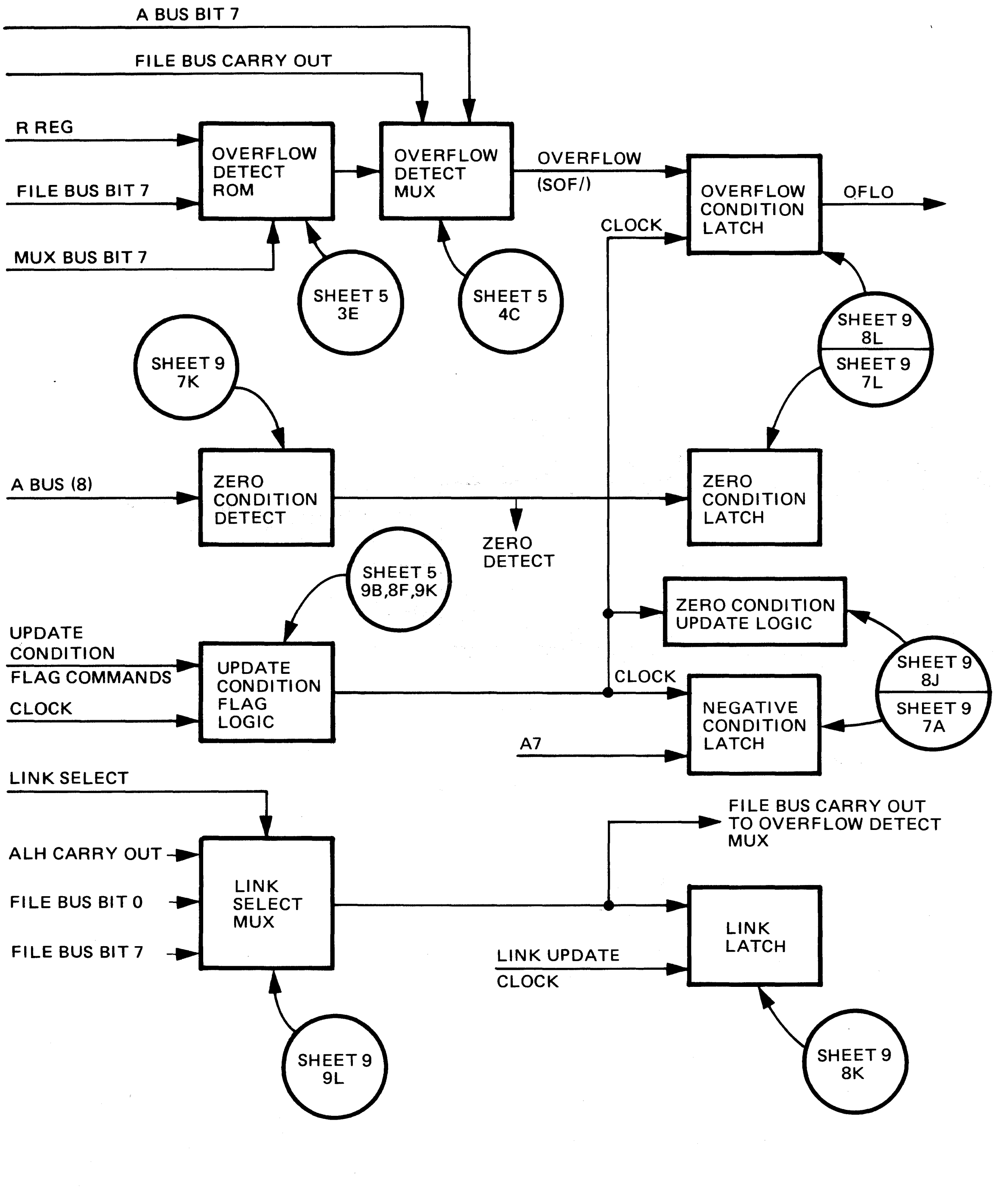


Figure 11-6. Condition & Link Register

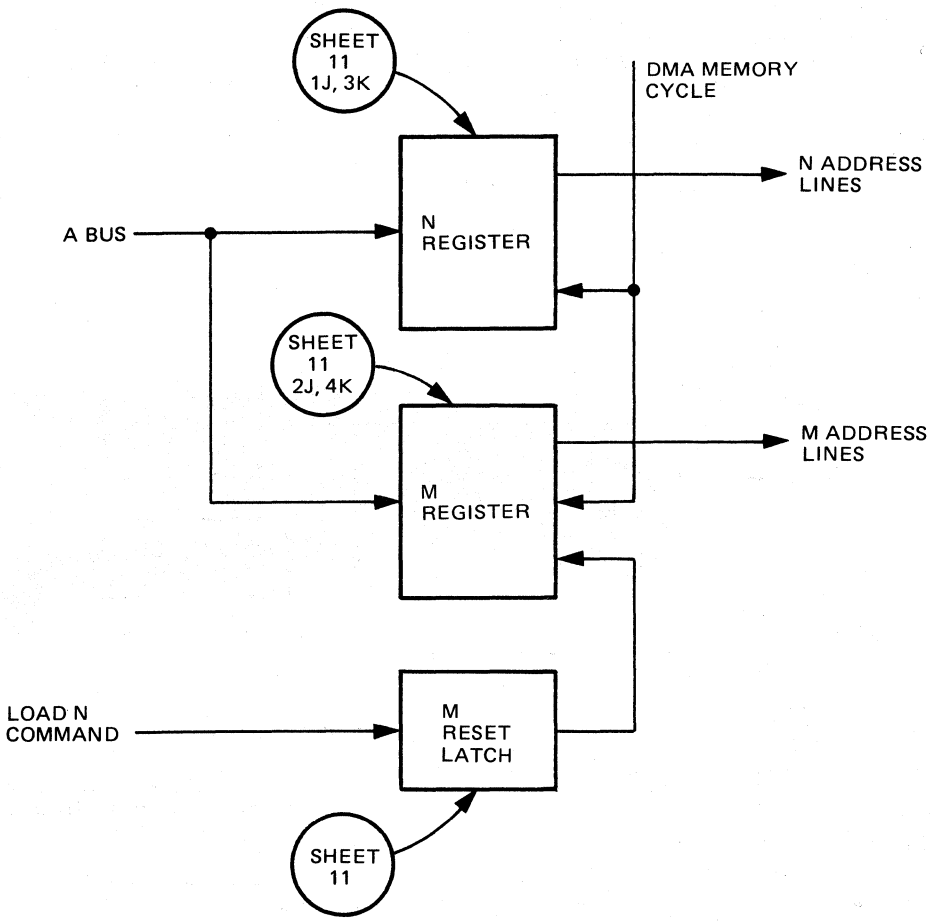


Figure 11-7. M and N Address Registers

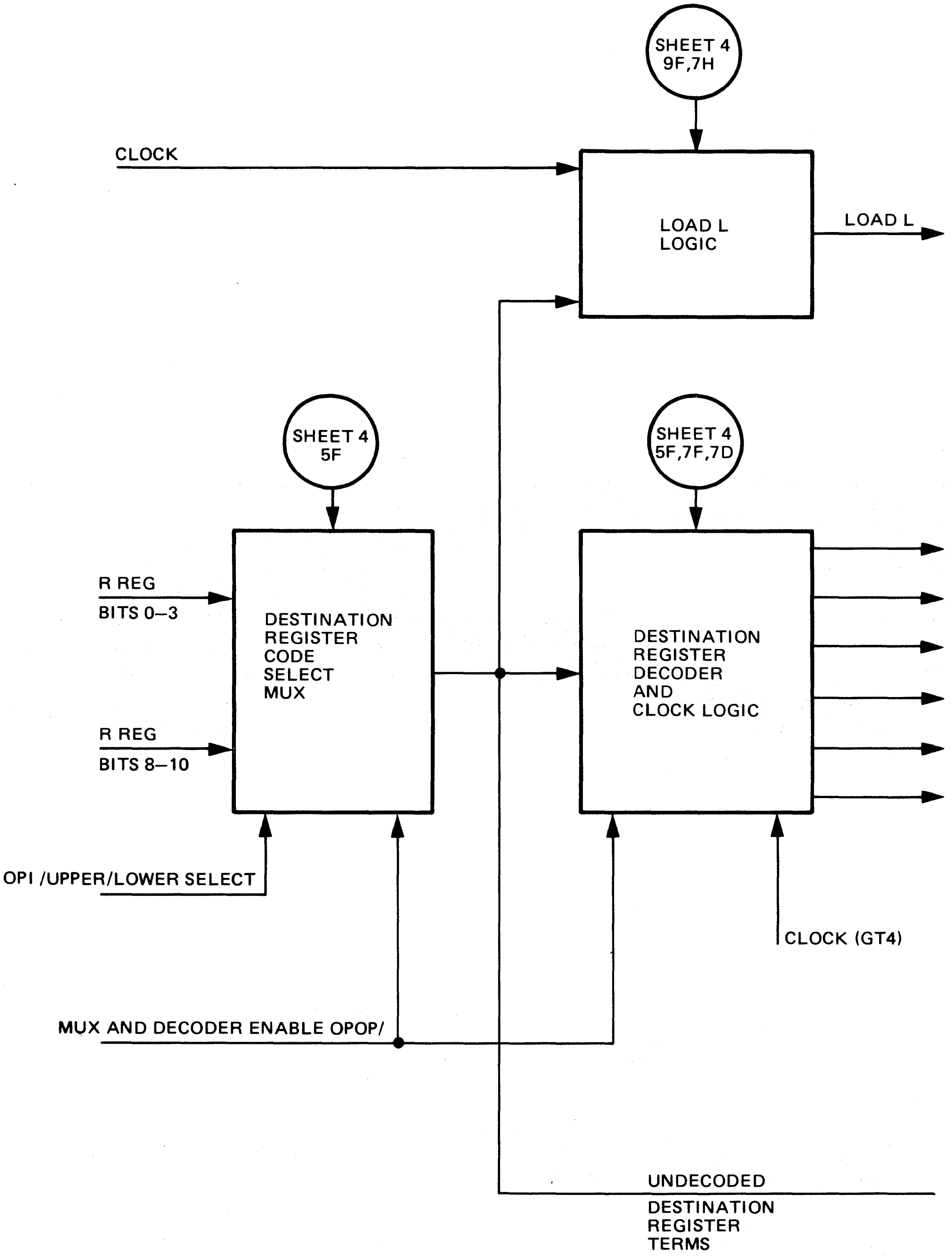
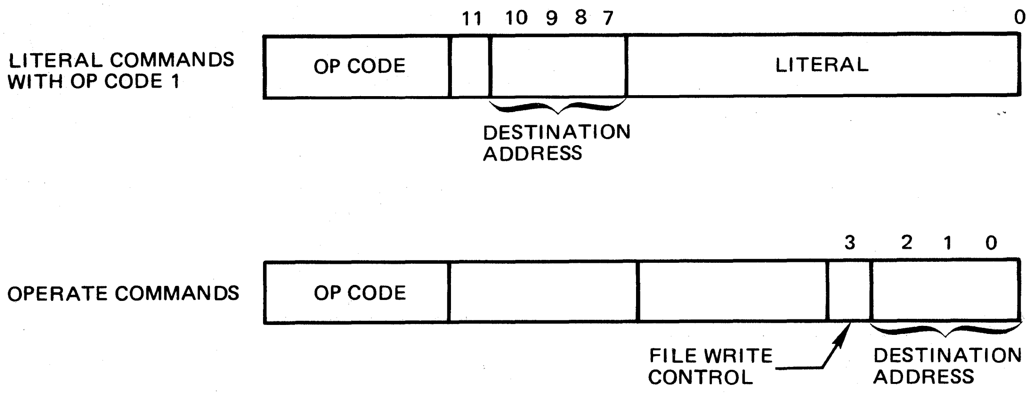


Figure 11-8. Destination Register Clock Logic



A 2-channel multiplexer is used to select the correct destination address according to the op code. Most of the destination register strobes are generated by a decoder having one of its inputs a GT4 clock. In some places, non-clocked or differently clocked destination register terms are needed. These terms, identified as RS0 to RS3, are provided for generation of Load L clock, file write clock, and indication of M, N, and T destination terms to generate T hold.

11.9 COMMAND DECODE ROMS

Firmware commands in the R register are decoded by read only memory chips to generate control terms. The ALU mode and MUX addresses are generated directly by the ROM's. Other terms are generated, such as 10 decoded OP code terms, which are used as inputs to discrete MSI and SSI logic functions to generate all of the register update control terms. In addition, the command decode ROM's are used to generate a carry-in and link, skip, and idle updates.

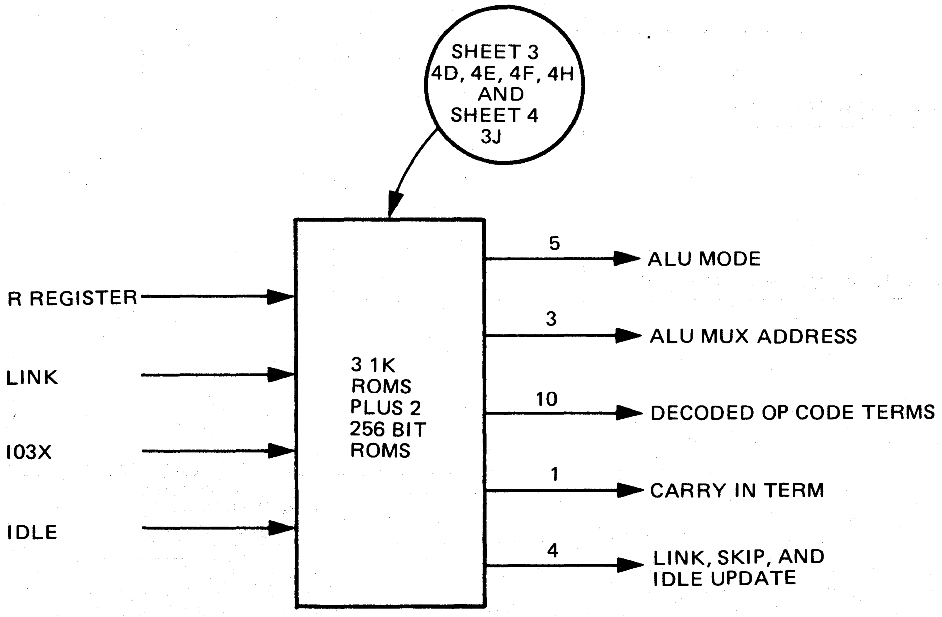


Figure 11-9. Command Decode ROM's Block Diagram

11.10 PROGRAMMED INPUT/OUTPUT

The programmed I/O, shown in Figure 11-10, consists of separate 8-bit input and output data buses. The output bus is driven from the T register through inverter drivers. Zero true logic is used. The input bus goes directly to the ALU input MUX. Both input and input/ go to the MUX. Input/output control is accomplished with three lines, identified as IO1X/, IO2X/, and IO3X/. These lines are set or reset by the firmware command:

15 12	11 8	7	6	5	4	3 0
7	FILE	1	IO3X	IO2X	IO1X	DESTINATION

Clock time for setting the registers is GT4. The IOXX/ register is a D type latch where the outputs follow the inputs as long as the clock is present.

The Serial Teletype output is driven by IO1X, IO2X, and IO3X/, and TTYØ from transistor driver.

11.11 INTERRUPTS

The Micro-One features two types of interrupts: internal and external. A block diagram of the Micro-One interrupt structure is shown in Figure 11-11. The internal interrupts consist of: console interrupt, power fail/restart, stepswitch interrupt, real-time clock, and four spares. The internal interrupts are input to the Micro-One individually via the internal status channel of the ALU MUX, and collectively via the eight input OR gate to bit 4 of File 0.

Internal interrupt latches are reset by either Master reset or the firmware command, enter internal status. The console interrupt consists of a clocked latch which is always enabled, and thus always responds when the console interrupt switch is depressed.

The power fail/restart interrupt consists of an RS latch which is set by power fail and held in a clamped-on state during power-on with release of clamping taking place immediately after releasing Master reset.

Real-time clock consists of an interrupt latch which is set each time a real-time clock pulse occurs, if the real-time clock enable latch is set. Real-time clock pulses are generated by a level sensor which has a full wave rectified power line signal as an input (120 Hz) or (100 Hz). The Step switch interrupt contains no latch and is tied directly to internal status input and to the internal interrupt input gate.

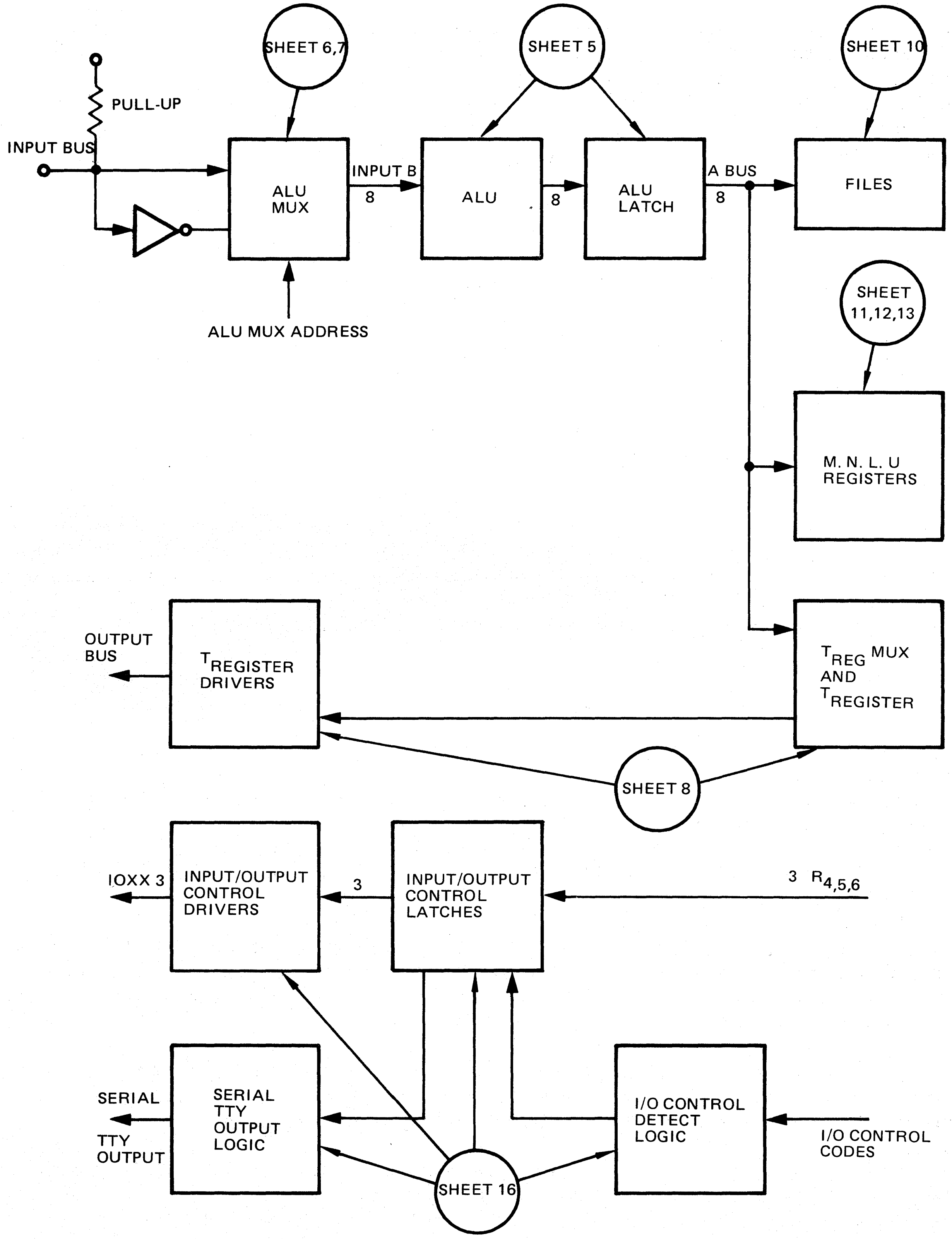


Figure 11-10. Programmed I/O Block Diagram

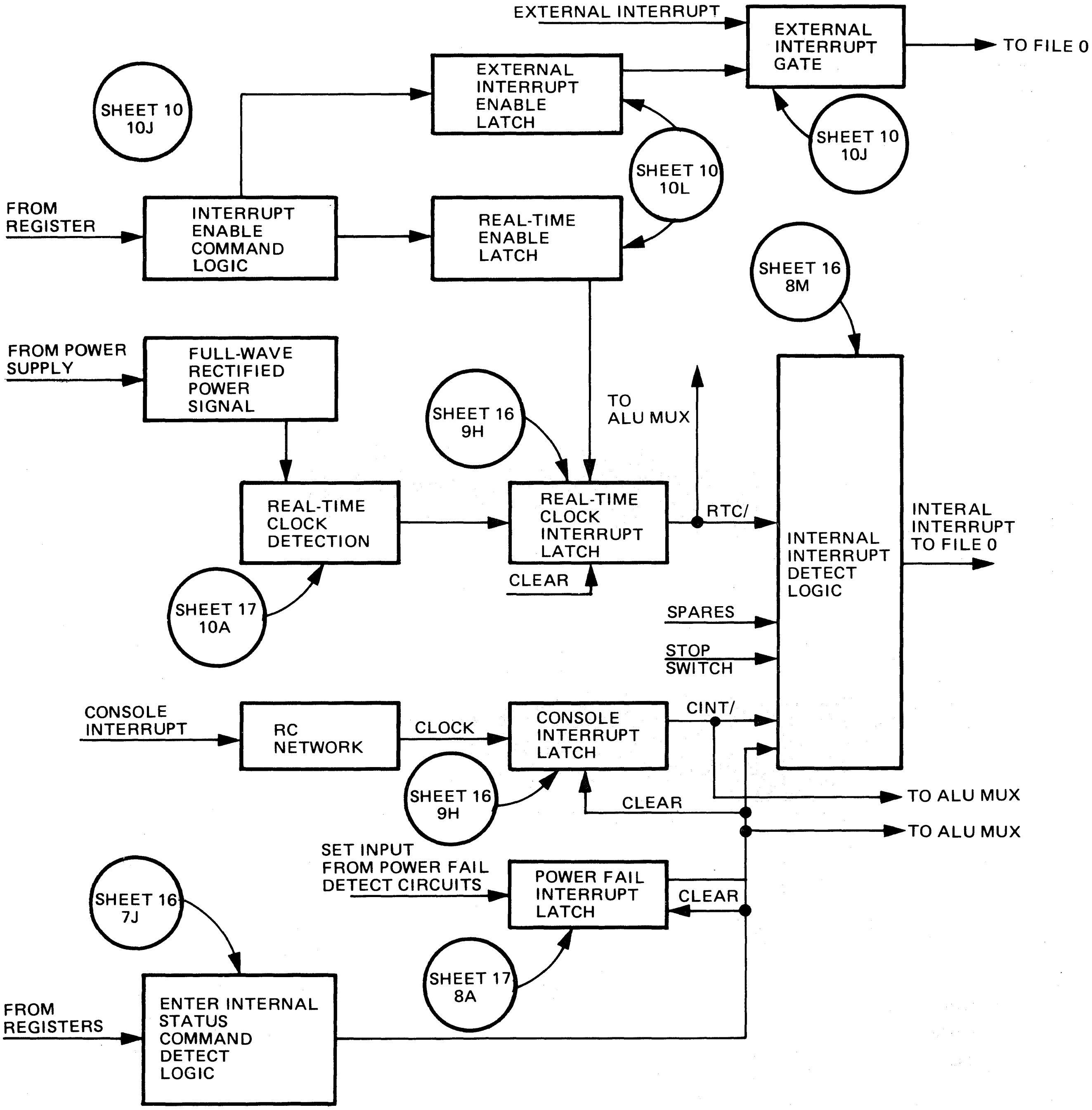


Figure 11-11. Interrupts

11.12 MEMORY SEQUENCER (CORE MEMORY VERSION)

Memory timing pulses are generated by a state sequencer (shown in Figure 11-12) which is organized around a ROM, a command latch, and a sequence counter. The six primary outputs from the sequencer consist of Memory Read Strobe (RTXX/), Memory Write Strobe (WTXX/), Memory Ready Command (READ), Memory Write (WRIT), Load T Strobe (LT2), and Memory Busy State (MBSY).

The inputs to the sequencer consist of: DMA Request (DMAR), DMA Write (DMAW), CPU Memory Request (OPA), CPU Write (R4), and CPU 1/2 cycle (R5). Commands are loaded any time Memory Busy is inactive. If there is no active command at load time the sequencer goes through one idle step and immediately loads again. There are five command input lines; therefore 32 different sequences are possible. When a command is loaded, memory busy becomes active, so the command remains loaded, and the sequence counter advances until memory busy becomes inactive. There are a maximum of eight 200 nanosecond steps possible for each command sequence. For Core Memory Control, the 32 command sequences are organized as follows in the ROM:

	CPU		DMA	
			READ	WRITE
CPU memory active	CPU full cycle read	CPU full cycle read	DMA read	DMA write
	CPU 1/2 cycle read	CPU 1/2 cycle read	DMA read	DMA write
	CPU full cycle write	CPU full cycle write	DMA read	DMA write
	CPU 1/2 cycle write	CPU 1/2 cycle write	DMA read	DMA write
CPU memory idle	idle	idle	DMA read	DMA write
	idle	idle	DMA read	DMA write
	idle	idle	DMA read	DMA write
	idle	idle	DMA read	DMA write

All DMA memory operations are full cycle.

The outputs from the ROM consist of RTXX/, WTXX/, READ/ and MBSY/

The 8-step sequence patterns for memory control are as follows:

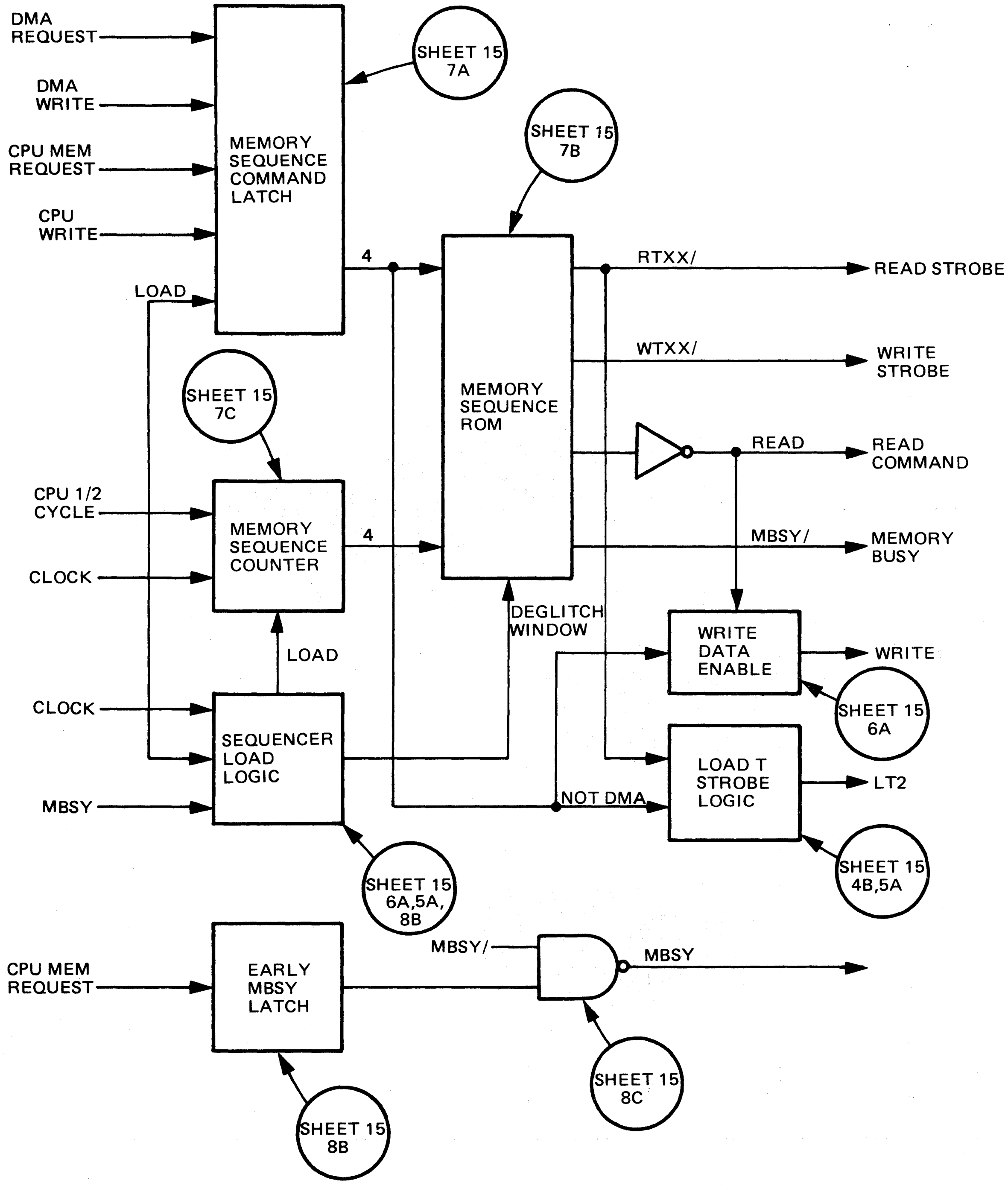


Figure 11-12. Memory Controller Block Diagram

Table 11-1.1. 8 Step 200 Nanosecond per Step Sequence Patterns for Memory Control

11.1.1 DMA or CPU Full Cycle Write

RTXX/	WTXX/	READ/	MBSY/
0	1	1	0
0	1	1	0
1	0	1	0
1	0	1	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

11.1.2 DMA and CPU Full Cycle Read

RTXX/	WTXX/	READ/	MBSY/
0	1	0	0
0	1	0	0
1	0	0	0
1	0	0	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

11.1.3 CPU 1/2 Cycle Write

RTXX/	WTXX/	READ/	MBSY/
1	0	1	0
1	0	1	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

11.1.4 CPU 1/2 Cycle Read

RTXX/	WTXX/	READ/	MBSY/
0	1	0	0
0	1	0	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Priority

Priority is established by the following methods:

1. When any memory sequence is in process, MBSY prevents loading a new command (either CPU or DMA).

2. On simultaneous DMA and CPU requests, the CPU request is held off by the T hold logic until the memory becomes unbusy, and another DMAR request is not present.

11.13 COMPUTER CLOCK AND RUN CONTROL

The clock generation logic (as shown in Figure 11-13) consists of a 20 MHz oscillator with a 4-bit shift register to generate a 4-phase 200 nanosecond clock. Each of the phases is approximately 50 nsec wide. The phases are identified as TT1/, TT2/, TT3/, and TT4/. The repeating cycle is maintained by reloading a binary 0111 pattern each time TT4/ = 0. Starting of the clock is achieved by entering a 0 into the serial input any time all phases are simultaneously high, which state will occur whenever the clock goes into improper operation.

Control of the CPU is established by enabling the clock phases under various operating conditions. The enabled clocks are as follows:

GT3 = RUNX ∧ TT3 Used to set Halt on Firmware Command and panel clock.

GT4 = RUNX ∧ TT4 Used for register update, Condition flag update, file update, I/O control register update, interrupt clocks.

Load L Clock = (RUNX/) ∧ (TT1 + TT2)/

LOAD R = TT4/ ∧ THOLD/

L COUNT CLOCK = TT4/ ∧ THLD/ ∧ HLT/

ALU LATCH UPDATE = TT2 + TT3

LOAD MEMORY SEQUENCER = TT4/ ∧ MBUSY/

11.14 RUN/HALT CONTROL

RUNX (see Figure 11-14) is made up of three terms as follows: RUNX = Halt/∧ ∧THOLD/ ∧ Idle/. Halt/ is the main run enable term of the CPU and is manually set by panel control, or automatically by the power fail/power on function. The Halt state can be caused by Firmware command, Panel Clock, or by Master Reset.

Thold (see Figure 11-14) is a temporary run disable which is caused by simultaneous occurrence of memory activity requests. There are four functions which cause Thold:

1. Attempting to alter M or N during a memory cycle.
2. Requesting a Memory cycle while one is in process.
3. Selecting T as a source before a read is complete.
4. DMA and CPU simultaneously requesting a memory access.
5. Changing T during a write strobe.

Thold stops all CPU firmware functions, including R register update and M, N, L, U, and File update and Input/Output register update.

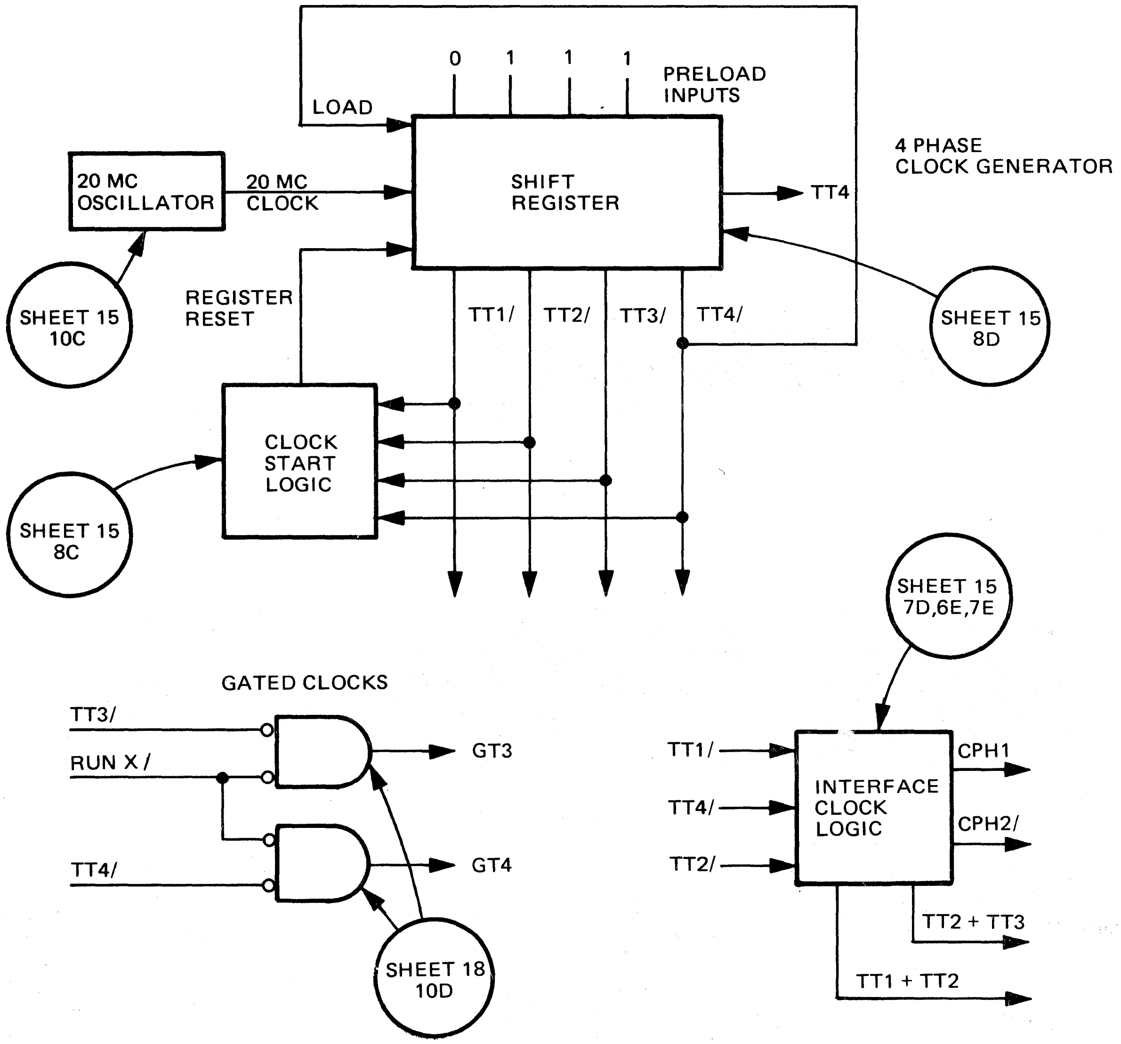


Figure 11-13. Clock Generation Logic, Block Diagram.

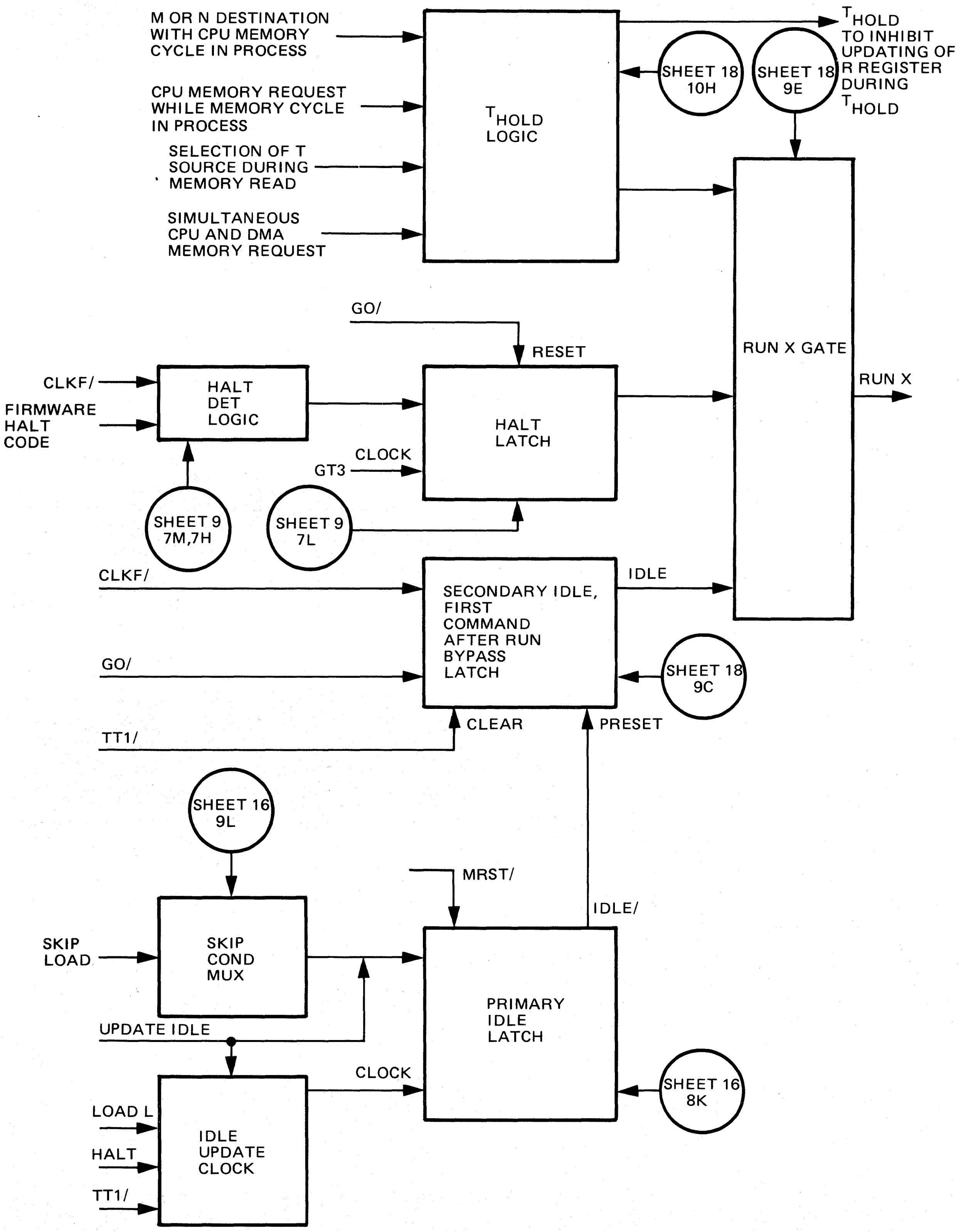


Figure 11-14. Run/Halt Control Block Diagram.

Idle (see Figure 11-14) is a temporary run-disable which occurs for one firmware clock cycle and is caused by a skip or load L action. During idle, the R register is still updated so that the next firmware command can be fetched, but all other functions are disabled. Idle disables Thold to prevent false Tholds on non-executed memory-related functions immediately following a jump command.

11.15 COMPUTER START LOGIC

There are five different inputs for starting the Micro One as shown in Figure 11-15.

1. Panel Interrupt
2. Panel Step Mode
3. Panel Run
4. Panel Clock
5. Power Fail/Restart Interrupt

All of these inputs share a common computer start latch and computer start pulse logic. The computer start latch is set asynchronously to the computer clock and remains active until reset by halt, or released by the power fail interrupt input. The computer start pulse is synchronized by clock phase TT3/, at which time GO/ is generated. GO/ is input to the clear side of the Halt latch. Halt/ is input to the computer start pulse logic to inhibit subsequent GO/ pulses.

Four of the computer start inputs are for panel control. These utilize a common RC pulse network, and separate diode isolators. The CPU differentiates between the four as follows:

Panel Interrupt and Panel Step mode are also input to the internal interrupt portion of the CPU, where they are recognized by the firmware. Panel clock is also input to the Halt Detect logic to generate halt after one Firmware Command has been executed, and to the First Command by-pass logic to prevent bypass when Panel Clock is activated as opposed to either of the other three panel start modes.

The Power Fail/Restart Interrupt generates the remaining start term. For either Power Fail or Restart, it is necessary to generate a GO/pulse. The Power Fail Interrupt will always be activated during these times, and will force the computer start latch to an active state where it will remain until released by acknowledging the Power Fail Interrupt. Multiple GO/ pulses are inhibited by the Halt/ input to the computer start pulse logic.

11.16 AUTOMATIC POWER FAIL AND POWER ON DETECTION FUNCTION

The Power Fail Detect logic (shown in Figure 11-16) consists of an RC filter and an analog level detector. The input is a full-wave rectified signal from the power supply. The level sensor has a feedback resistor to generate hysteresis. This hysteresis prevents fluttering of the power fail circuitry

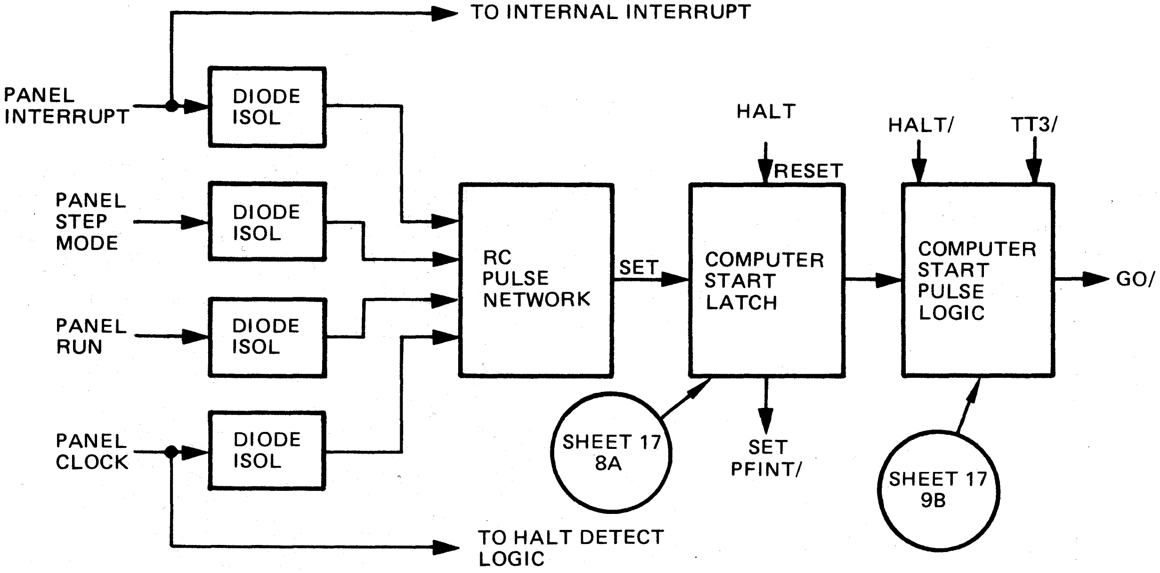


Figure 11-15. Computer Start Logic, Block Diagram

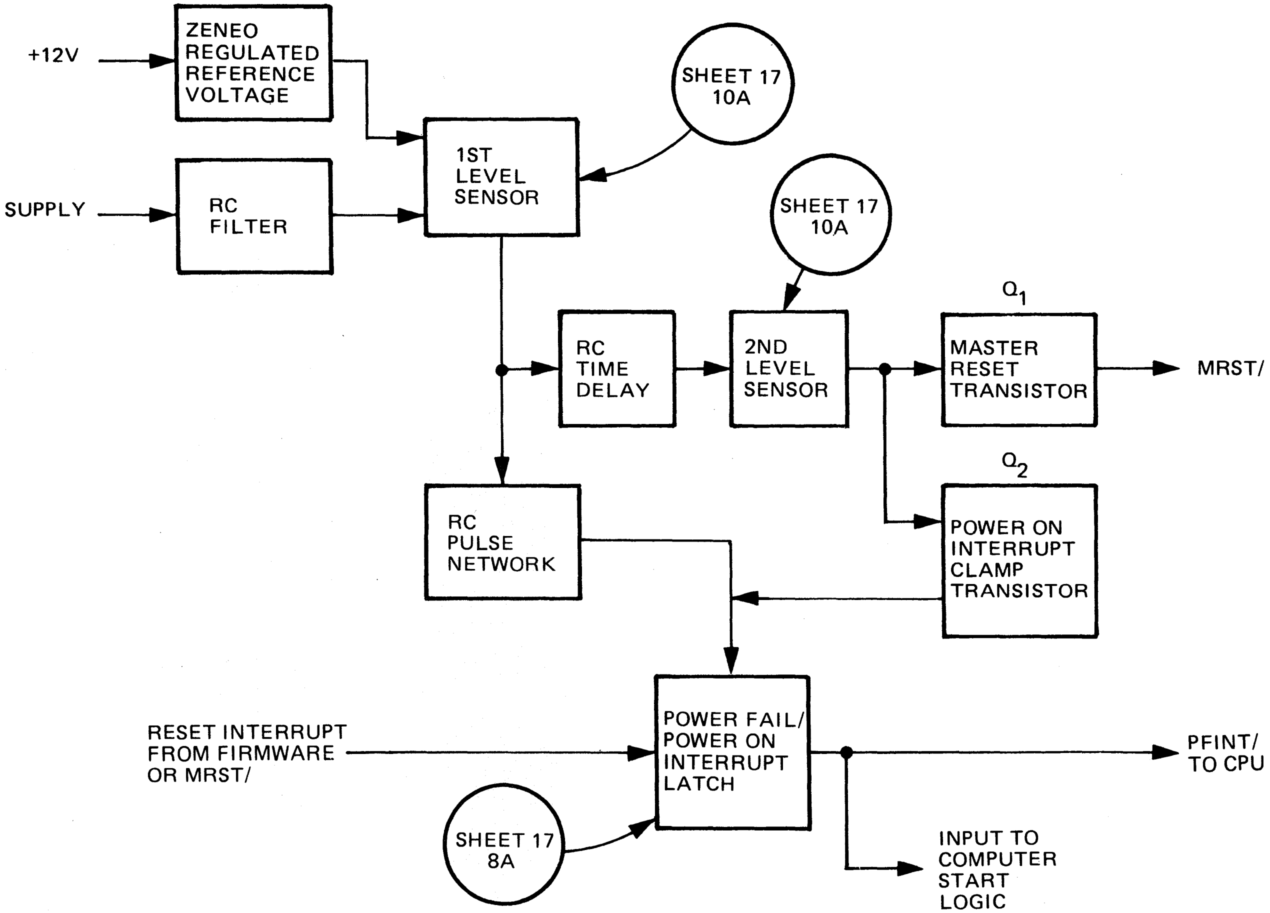

+12V
ZENEO REGULATED REFERENCE VOLTAGE
SUPPLY
RC FILTER
1ST LEVEL SENSOR
SHEET 17 10A
SHEET 17 10A
RC TIME DELAY
2ND LEVEL SENSOR
Q1
MASTER RESET TRANSISTOR
MRST/
Q2
POWER ON INTERRUPT CLAMP TRANSISTOR
RC PULSE NETWORK
RESET INTERRUPT FROM FIRMWARE OR MRST/
POWER FAIL/ POWER ON INTERRUPT LATCH
SHEET 17 8A
PFINT/ TO CPU
INPUT TO COMPUTER START LOGIC

Figure 11-16. Power Fail Detect Logic Block Diagram

at the power fail threshold levels. The input RC time constant is set so that the first missing power line pulse will be detected. When power fail is detected, two things occur: the power fail interrupt is immediately set, and after approximately a 2 msec delay, Master Reset is activated and remains activated while the power supply voltages decay. The function of setting of the CPU for power fail, such as saving the registers in core and coming to halt, is accomplished by software and firmware during the time between Power Fail Interrupt and Master Reset. During the initial part of power on, the Power Fail Interrupt is clamped to an active state. When power reaches a correct level (plus a delay) the level sensor switches output levels and starts a delay of approximately 100 milliseconds, at which time Master Reset is released, leaving Power Fail Interrupt in an active state, to be acknowledged by the firmware/software.

For manual operations, the Power Fail Interrupt can be made momentarily inactive by depressing the Master Reset switch.

SECTION 12

SCHEMATICS

REV	EO	ZONE	DESCRIPTION	DWN	CHKD	APPD	DATE
D	3612		PILOT RELEASE	J.N.	J.O.		4/8/75
E	3662		REVISED AND NEW RELEASED	M.F.	J.O.		4/28/75
F	4062		SEE E.O.	C.T.	J.O.		10/17/75
G	4172		SEE E.O.	C.T.	J.O.		10/17/75

COMPONENTS USED	LAST REF DES USED	REF DES NOT USED
RESISTOR	R42	R11
RES MODULE	Z20	
CAPACITOR	C27	
DIODE	CR14	
TRANSISTOR	Q3	
CHOKE	L1	
CRYSTAL	Y1	
E POINT	E13	

	10	9	8	7	6	5	4	3	
A	17 17 17 LM339N		15 11 17 17 74279	15 7475	15 15 15 15 7402	15 15 15 4 74H04	SPARE	16 7425	A
B	17 OSC	18 17 5 74H00	15 15 7474	15 5304 $\boxed{3}$	15 15 4 15 5 15 7417	SPARE	15 13 7410	16 4 7420	B
C	15 18 18 15 18 18 7404	18 74H74	15 15 7420	15 74161	8 9322	8 9322	5 9312	SPARE	C
D	18 17 18 7402	16 7475	15 74195	15 15 4 8 7400	8 7475	8 7475	3 5303-1	8 8 16 8 8 16 7417	D
E		18 16 7410	16 4 16 12 74H00	4 15 13 15 4 4 74H04	8 8 8 8 7438	8 8 8 8 7438	3 5303-3	5 5303-4	E
F	18 18 7402	4 4 10 4 7402	15 10 9 5 7408	4 7442	8 8 8 8 15 16 7417	4 9322	3 5303-2	12 7475	F
H	18 74S64	16 16 7474	10 10 10 9 4 9 74H04	4 9 7410	7 7 7 7 7 7 74H04	7 7 6 6 6 6 74H04	3 5302-1	12 7475	H
J	13 4 16 10 7402	18 18 9 74H04	9 9 18 9 7402	9 16 74H22	7 9312	6 9312	SPARE	4 5302-2	J
K	16 4 12 7427	5 16 9 9 74H00	16 9 7474	9 7430	7 9312	6 9312	11 74173	11 74173	K
L	10 74109	16 9 9309	9 9 7474	9 9 7474	7 9312	6 9312	10 3101A	10 10 10 10 7403	L
M	10 10 7451	SPARE	16 7430	9 10 7425	7 9312	6 9312	10 3101A	10 10 10 10 7403	M
	10	9	8	7	6	5	4	3	

PIN 1 (REF)

2	1	
14 6240	14 6240	A
14 6240	14 6240	B
13 74161	13 74161	C
13 13 74H74	13 74175	D
12 74175	12 10 12 10 74H04	E

12
74298
	12 12
	12 12
	7432
		F

12
74298
	12 12
	12 12
	7432
		H

11 74173	11 74173	J
5 7475	5 7475	K
5 74181	5 74181	L
2	1	

CURRENT REV	G	D	D	D	D	D	D	D	G	D	D	D	D	D	G	D	D	F
SHEET NO.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

$\boxed{5}$ SIGNAL NOT USED ON THIS BOARD

$\boxed{4.}$ FOR -21 FIRMWARE SET CUT ETCH BETWEEN E1 & E2 (COMPONENT SIDE)

$\boxed{3.}$ I.C. AT LOCATION 7B TO BE SELECTED ACCORDING TO NEED FOR MOS MEMORY (-2) OR CORE MEMORY (-1).

2. ALL CAPACITORS ARE .1µf, 25V.

1. ALL RESISTORS ARE 1/4 W, 5% VALUED IN OHMS.

NOTES: UNLESS OTHERWISE SPECIFIED

UNINCORPORATED ENGINEERING ORDERS

DRAFTSMAN	JODI NEWELL	DATE 3/27/75
CHECKER	L. Ovento	4-7-75
ENGINEER	Robert Stewart	4-8-75
APPD	W. Schrader	4-8-75
APPD	D Swift	4-8-75
APPD		6/24/75
APPD		6/24/75

SC 20001506

G

TITLE P.C. BOARD SCHEMATIC— MICRO-ONE C.P.U.

Microdata™
IRVINE, CALIFORNIA

DWG SIZE		REV
C	SC 20001506	G

SCALE ∽ IDENT CODE 52936 SHEET 1 OF 18

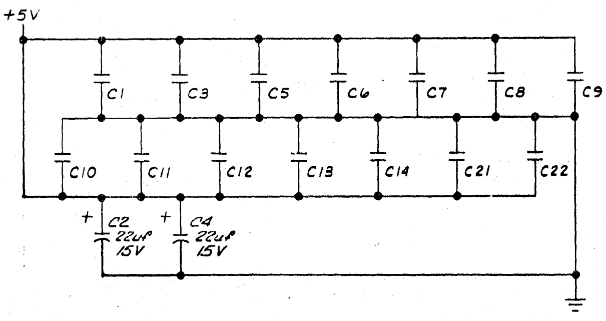
	SHT. NO.	A	P1	B	SHT. NO.	
		GND	1	+5V		
		GND	2	+5V		
			3	-16.75V	16	
	16	-16.75V	4			
	16	+12V	5			
	15	CPH1	6			
			7	M04A/	11	
	16	-16.75V	8	SPI1 /	7	
	11	N07A /	9	M06A/	11	
	8	ØD05 /	10	ØD01 /	8	
	11	M01A /	11	A04L /	5	
	11	M02A /	12			
	5	A00L /	13	M03A /	11	
	5	A01L /	14	M00A /	11	
	11	N06A /	15	RS00	12	
	13	L00X	16			
5		L11X (GND)	17	RS04	13	
	13	L04X	18	M05A /	11	
	12	RS01	19	L10X (GND)		5
	13	L01X	20	RTXX /	15	
	15	WTXX /	21	RS05	13	
	13	L05X	22	CPH2 /	15	
	12	RS02	23	READ	15	
	13	L02X	24	CGL0 /	4	
	13	L06X	25	RS06	13	
	8	ØD02 /	26	ØD06 /	8	
	12	RS03	27	MD07	8	
	13	L03X	28	MD03	8	
	5	A02L /	29	RS07	13	
	13	L07X	30	MD05	8	
	16	IØ2X /	31	IØ1X /	16	
	7	ID04 /	32	ID00 /	6	
	14	CPEN /	33	L08X	13	
	5	A03L /	34	MD00	8	
	8	MD01	35	L09X	13	
	8	MD04	36	RS08	12	
	12	RS09	37	ØD04 /	8	
	10	EINT /	38	A05L /	5	
			39	ØD00 /	8	
	9	AENP/	40	MD06	8	
	18	RUNX	41	A06L /	5	
	12	RS10	42	ECIØ /	10	
	12	RS11	43	MD02	8	
	15	DMAR /	44	MRST /	16	
5		DMAS /	45			
	17	PFDI	46	RTCI	17	
	12	RS13	47	M07A /	11	
	12	RS14	48	A07L /	5	
5		CØNT (GND)	49	DMAT /	6	
	11	N03A /	50	IRPY /	10	
	12	RS12	51	N00A /	11	
	2	SELØ /	52	SELI /	2	
	11	N04A /	53	N01A /	11	
	11	N05A /	54	PRIN /	2	
	2	PRØT /	55	N02A /	11	
	15	DMAW /	56	SPI2 /	7	
	15	MBSY	57	SPI0 /	6	
	8	ØD07 /	58	ØD03 /	8	
	12	RS15	59	ID05 /	7	
	6	ID01 /	60	ID07 /	7	
	7	ID06 /	61	IØ3X /	16	
	6	ID03 /	62	ID02 /	6	
		-16.75V	63	-16.75V		
		GND	64	+5V		
		GND	65	+5V		

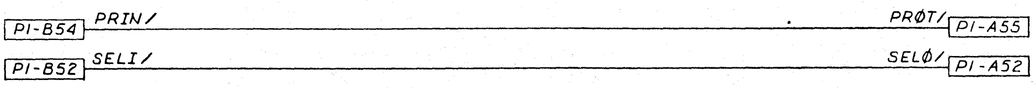
J2		SHT. NO.
1	TTYXI	16
2	TTYB	16
3	TTYØ	16
4		
5	TTYG	16
6	TTYG	16

J3		SHT. NO.
1	M06A/	11
2	MD00	8
3	RTX1 /	15
4	M03A/	11
5	M04A/	11
6	RFSH	15
7	READ/	15
8	MBSY /	15
9	WTX1 /	15
10	GND	
11	GND	
12		
13	CPH2 /	15
14	-16.75V	
15		
16	MD05	8
17	MD03	8
18	N07A /	11
19	M01A /	11
20	M02A /	11
21	MSEX /	15
22	M00A /	11
23	N06A /	11
24	GND	
25	M05A /	11
26	GND	
27	RTXX /	15
28	WTXX /	15
29	-16.75V	
30	-16.75V	
31	READ	15
32	+5V	
33	+5V	
34	MD07	8

J4		SHT. NO.
1	MD04	8
2	MD02	8
3	MD01	8
4	MD06	8
5	GND	
6	GND	
7	GND	
8	GND	
9	GND	
10	N03A/	11
11		
12		
13		
14	N00A /	11
15	N01A /	11
16	N04A /	11
17	N02A /	11
18	N05A /	11
19	+5V	
20	+5V	

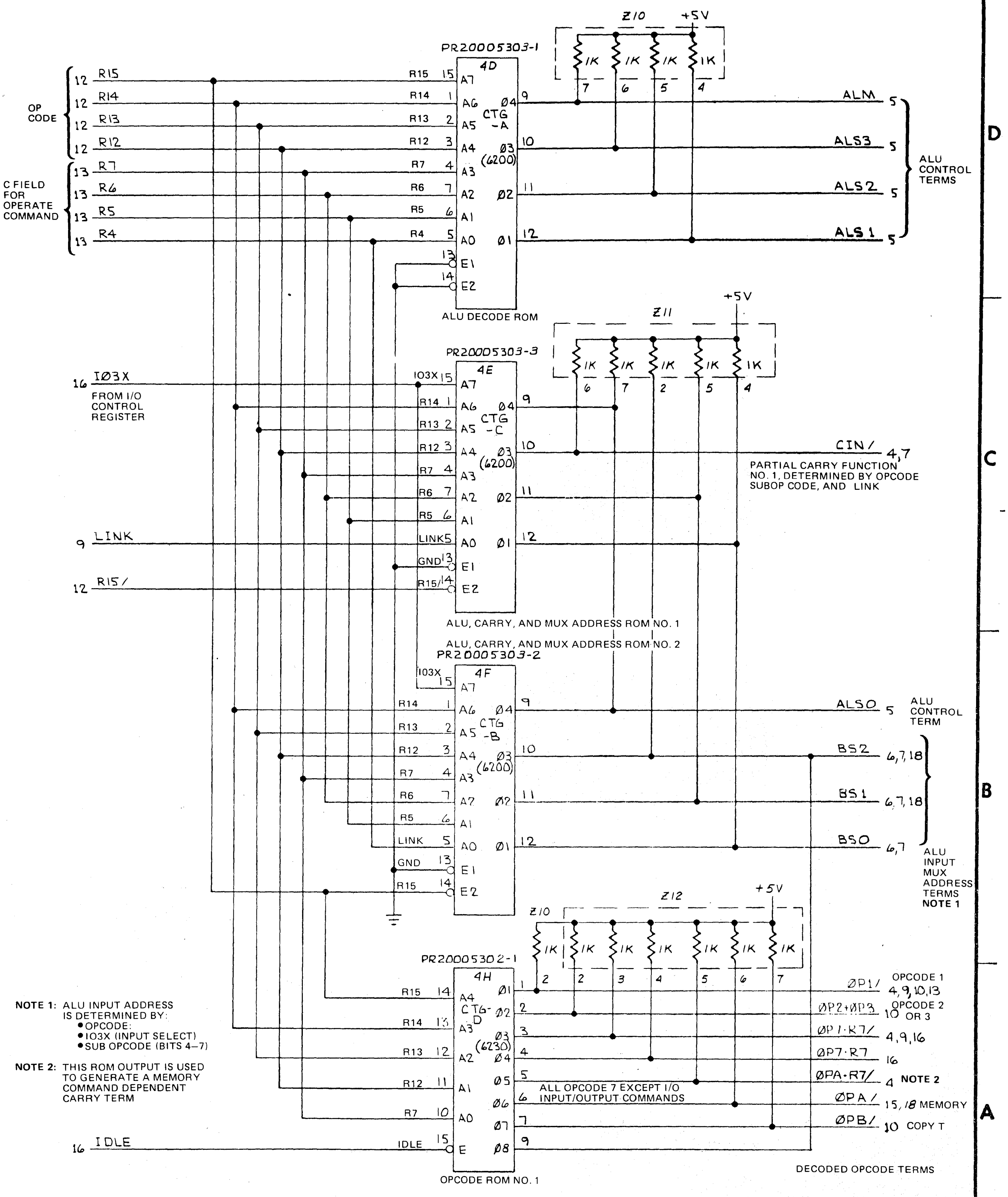
J9		SHT. NO.
1		
2	ES07	7
3	ES04	7
4	ES05	7
5	HLTL /	9
6	INTF /	16
7	ES06	7
8	RUNF /	17
9	STPF /	16
10	CLKF /	17





C	SC20001506	D
DWG SIZE	SHEET 2 OF 18	REV

FORM 9-130-2

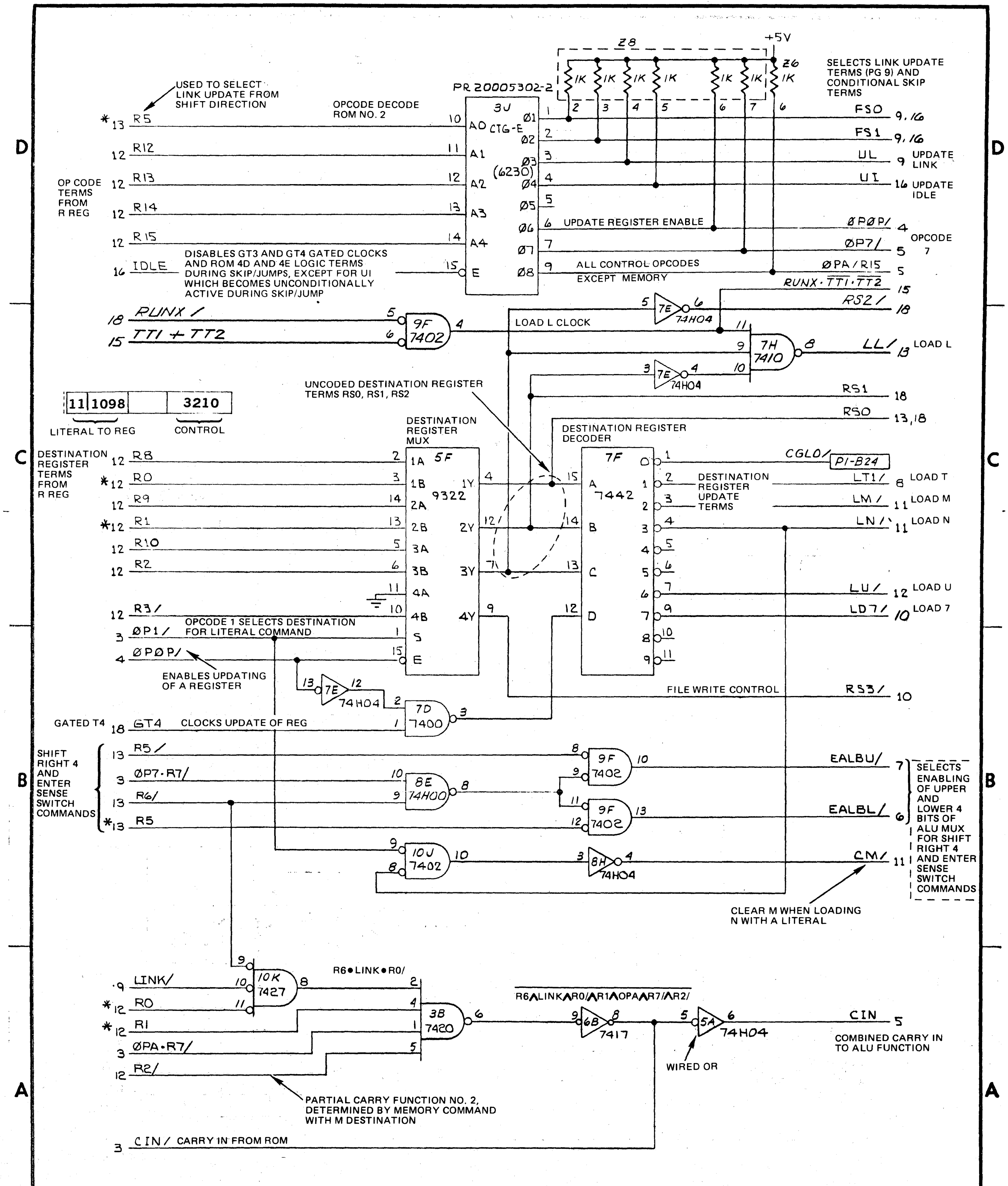


NOTE 1: ALU INPUT ADDRESS IS DETERMINED BY:
- OPCODE:
- IO3X (INPUT SELECT)
- SUB OPCODE (BITS 4–7)

NOTE 2: THIS ROM OUTPUT IS USED TO GENERATE A MEMORY COMMAND DEPENDENT CARRY TERM

CONTROL DECODE ROMS

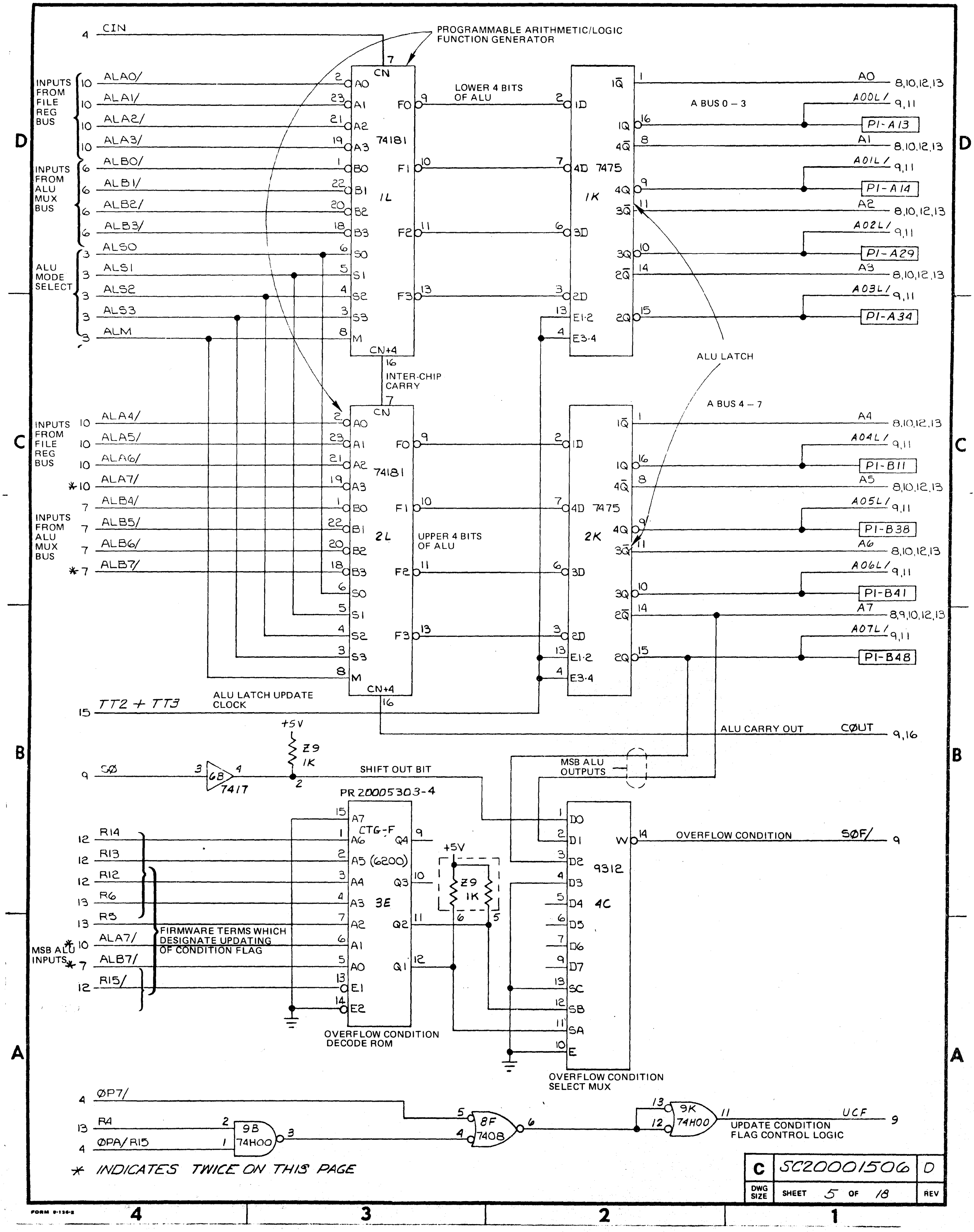
C	SC20001506	D
DWG SIZE	SHEET 3 OF 18	REV


Z8
+5V
Z6
1K 1K 1K 1K 1K 1K 1K
2 3 4 5 6 7 6
USED TO SELECT LINK UPDATE FROM SHIFT DIRECTION
SELECTS LINK UPDATE TERMS (PG 9) AND CONDITIONAL SKIP TERMS
OPCODE DECODE ROM NO. 2
PR 20005302-2
3U
CTG-E
(6230)
A0 A1 A2 A3 A4 E
Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7 Ø8
*13 R5
12 R12
12 R13
12 R14
12 R15
16 IDLE
OP CODE TERMS FROM R REG
DISABLES GT3 AND GT4 GATED CLOCKS AND ROM 4D AND 4E LOGIC TERMS DURING SKIP/JUMPS, EXCEPT FOR UI WHICH BECOMES UNCONDITIONALLY ACTIVE DURING SKIP/JUMP
UPDATE REGISTER ENABLE
ALL CONTROL OPCODES EXCEPT MEMORY
FS0 9,16
FS1 9,16
UL 9 UPDATE LINK
UI 16 UPDATE IDLE
ØPØP/ 4
ØP7/ 5
OPCODE 7
ØPA/R15 5
RUNX·TTI·TT2 15
RS2/ 18
18 RUNX/
15 TT1 + TT2
9F 7402
LOAD L CLOCK
7E 74H04
7H 7410
LL/ 13 LOAD L
RS1 18
RS0 13,18
11 1098 3210
LITERAL TO REG
CONTROL
UNCODED DESTINATION REGISTER TERMS RS0, RS1, RS2
DESTINATION REGISTER MUX
DESTINATION REGISTER DECODER
5F 9322
1A 1B 2A 2B 3A 3B 4A 4B S E
1Y 2Y 3Y 4Y
7F 7442
A B C D
CGLO/ P1-B24
DESTINATION REGISTER UPDATE TERMS
LT1/ 8 LOAD T
LM/ 11 LOAD M
LN/ 11 LOAD N
LU/ 12 LOAD U
LD7/ 10 LOAD 7
DESTINATION REGISTER TERMS FROM R REG
12 R8
*12 R0
12 R9
*12 R1
12 R10
12 R2
12 R3/
3 ØP1/
4 ØPØP/
OPCODE 1 SELECTS DESTINATION FOR LITERAL COMMAND
ENABLES UPDATING OF A REGISTER
7E 74H04
7D 7400
FILE WRITE CONTROL
RS3/ 10
GATED T4 18 GT4
CLOCKS UPDATE OF REG
SHIFT RIGHT 4 AND ENTER SENSE SWITCH COMMANDS
13 R5/
3 ØP7·R7/
13 R6/
*13 R5
8E 74H00
9F 7402
EALBU/ 7
EALBL/ 6
SELECTS ENABLING OF UPPER AND LOWER 4 BITS OF ALU MUX FOR SHIFT RIGHT 4 AND ENTER SENSE SWITCH COMMANDS
10J 7402
8H 74H04
CM/ 11
CLEAR M WHEN LOADING N WITH A LITERAL
·9 LINK/
*12 R0
*12 R1
3 ØPA·R7/
12 R2/
10K 7427
R6•LINK•R0/
3B 7420
$\overline{R6 \wedge LINK \wedge R0/ \wedge R1 \wedge OPA \wedge R7/ \wedge R2/}$
6B 7417
5A 74H04
CIN 5
COMBINED CARRY IN TO ALU FUNCTION
WIRED OR
PARTIAL CARRY FUNCTION NO. 2, DETERMINED BY MEMORY COMMAND WITH M DESTINATION
3 CIN/ CARRY IN FROM ROM
D C B A
4 3 2 1

* INDICATES TWICE ON THIS PAGE

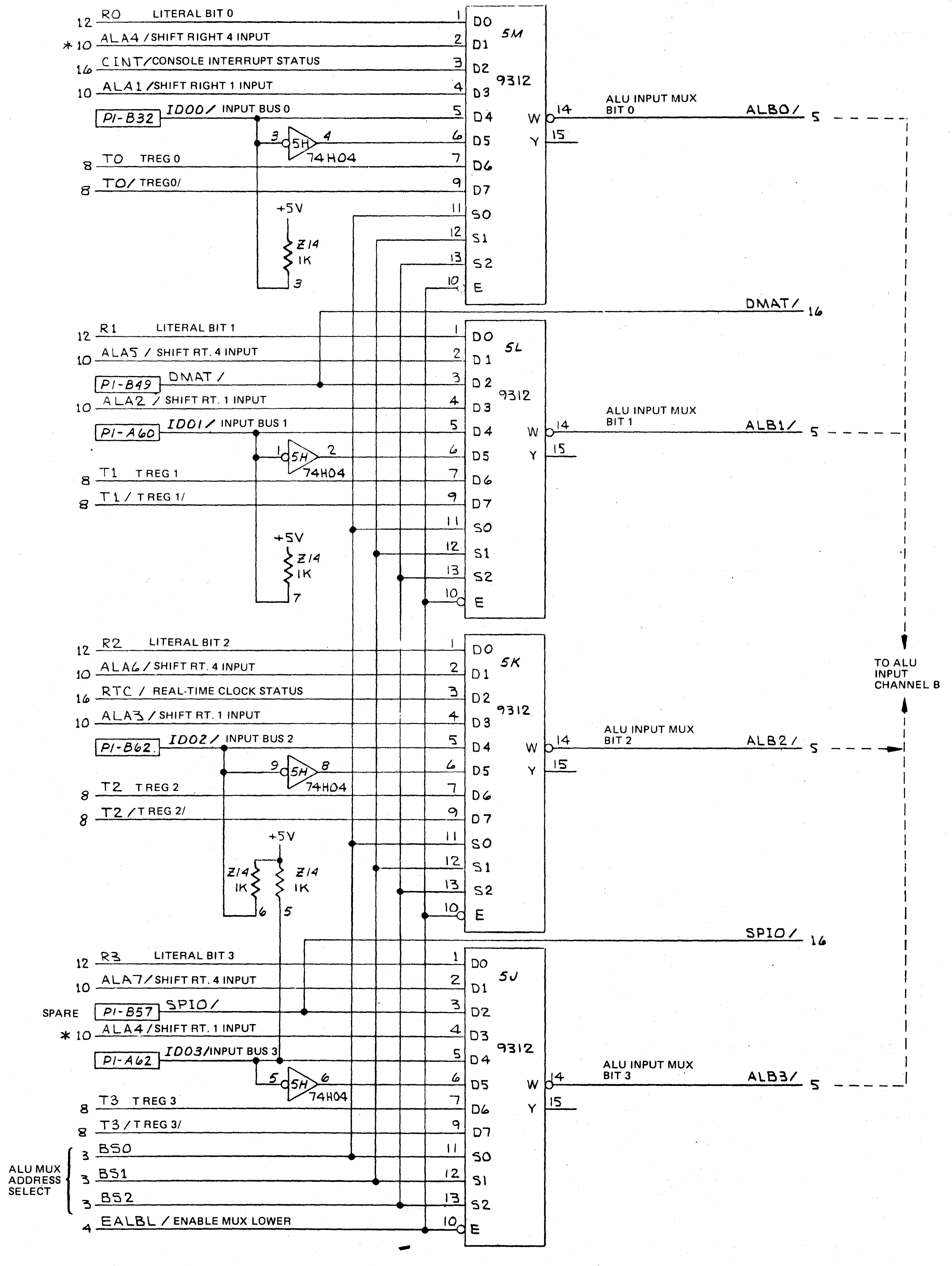
C	SC20001506	D
DWG SIZE	SHEET 4 OF 18	REV

FORM 9-130-2



* INDICATES TWICE ON THIS PAGE

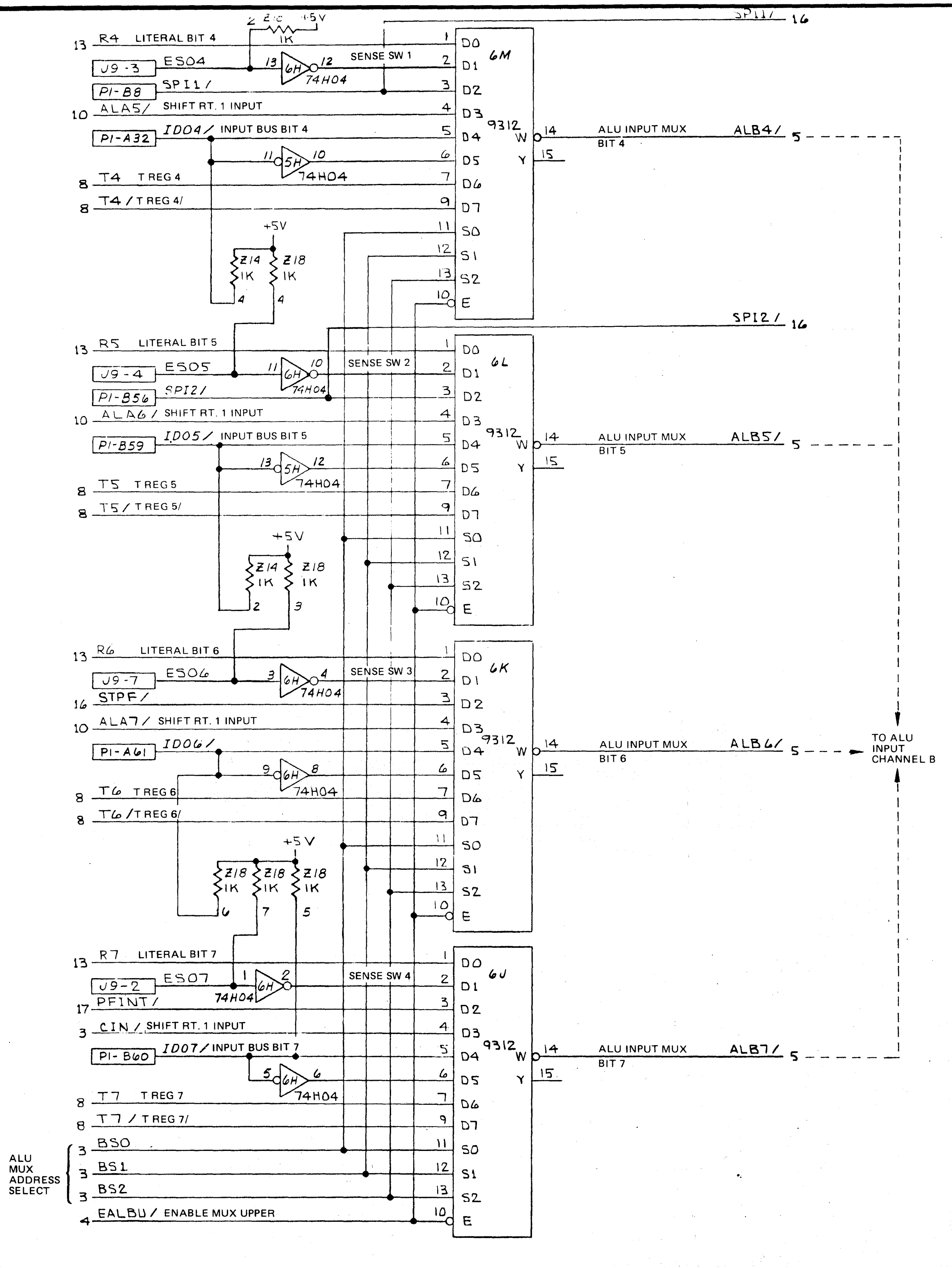
C	SC20001506	D
DWG SIZE	SHEET 5 OF 18	REV



ALU INPUT MUX BITS 0 – 3

* INDICATES TWICE ON THIS PAGE

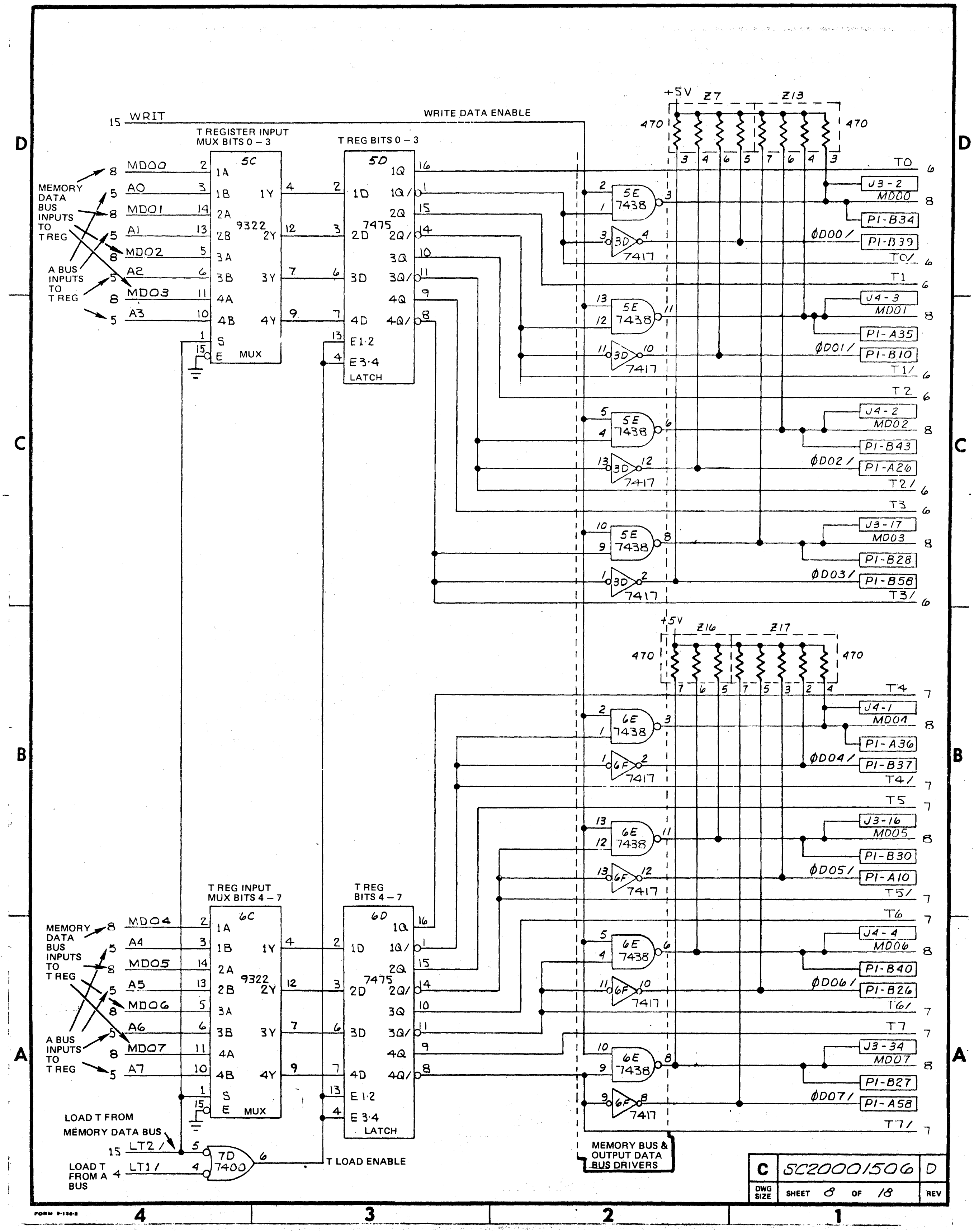
C	SC20001506	D
DWG SIZE	SHEET 6 OF 18	REV



ALU INPUT MUX BITS 4 – 7

C	SC20001506	D
DWG SIZE	SHEET 7 OF 18	REV

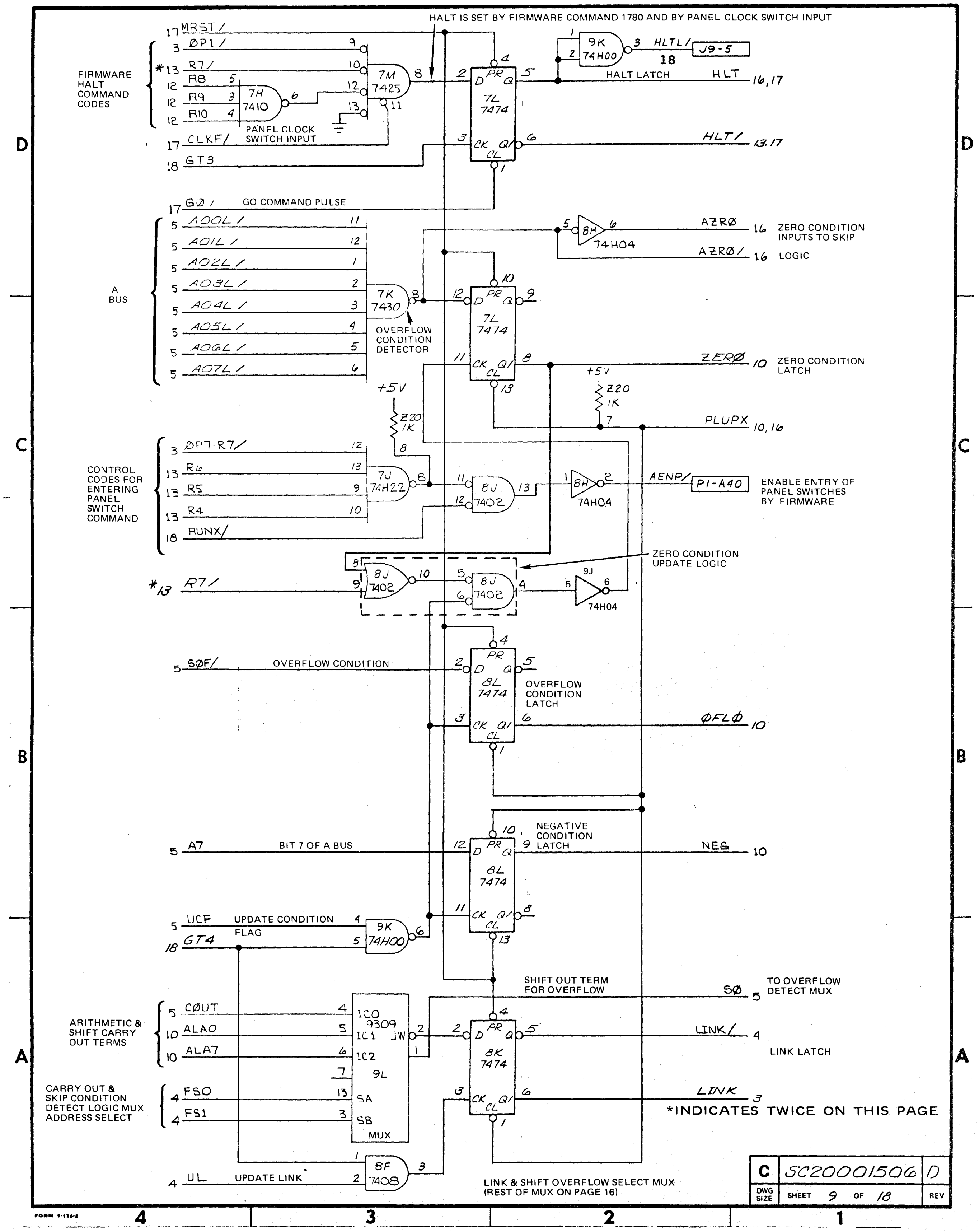
FORM 9-136-2



C	SC20001506	D
DWG SIZE	SHEET 8 OF 18	REV

DEFINITION OF TERMS

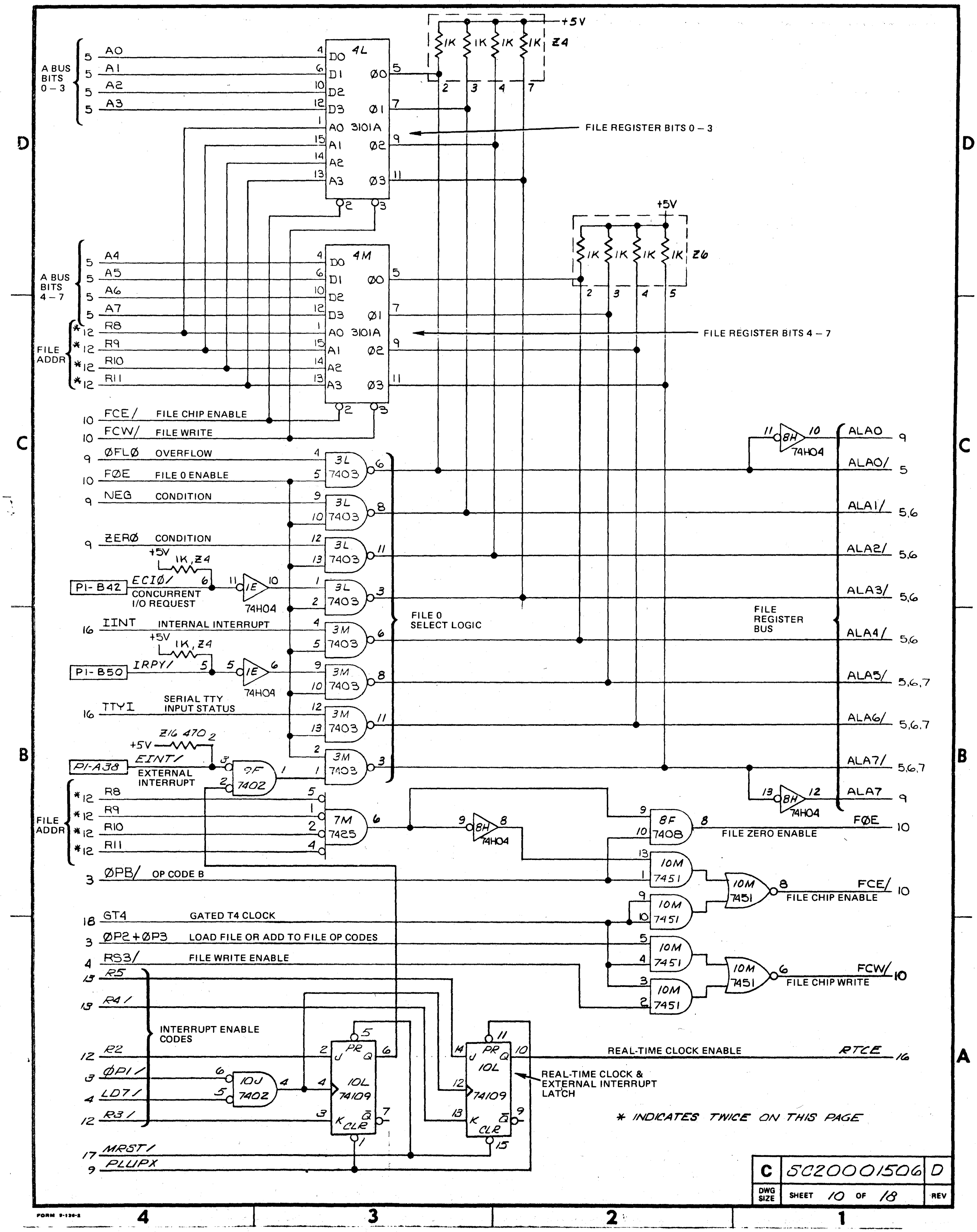
T'x'	T REGISTER
T'x'/	T REGISTER/
OTB'x'/	OUTPUT DATA BUS/
MD'x'	MEMORY DATA BUS



*INDICATES TWICE ON THIS PAGE

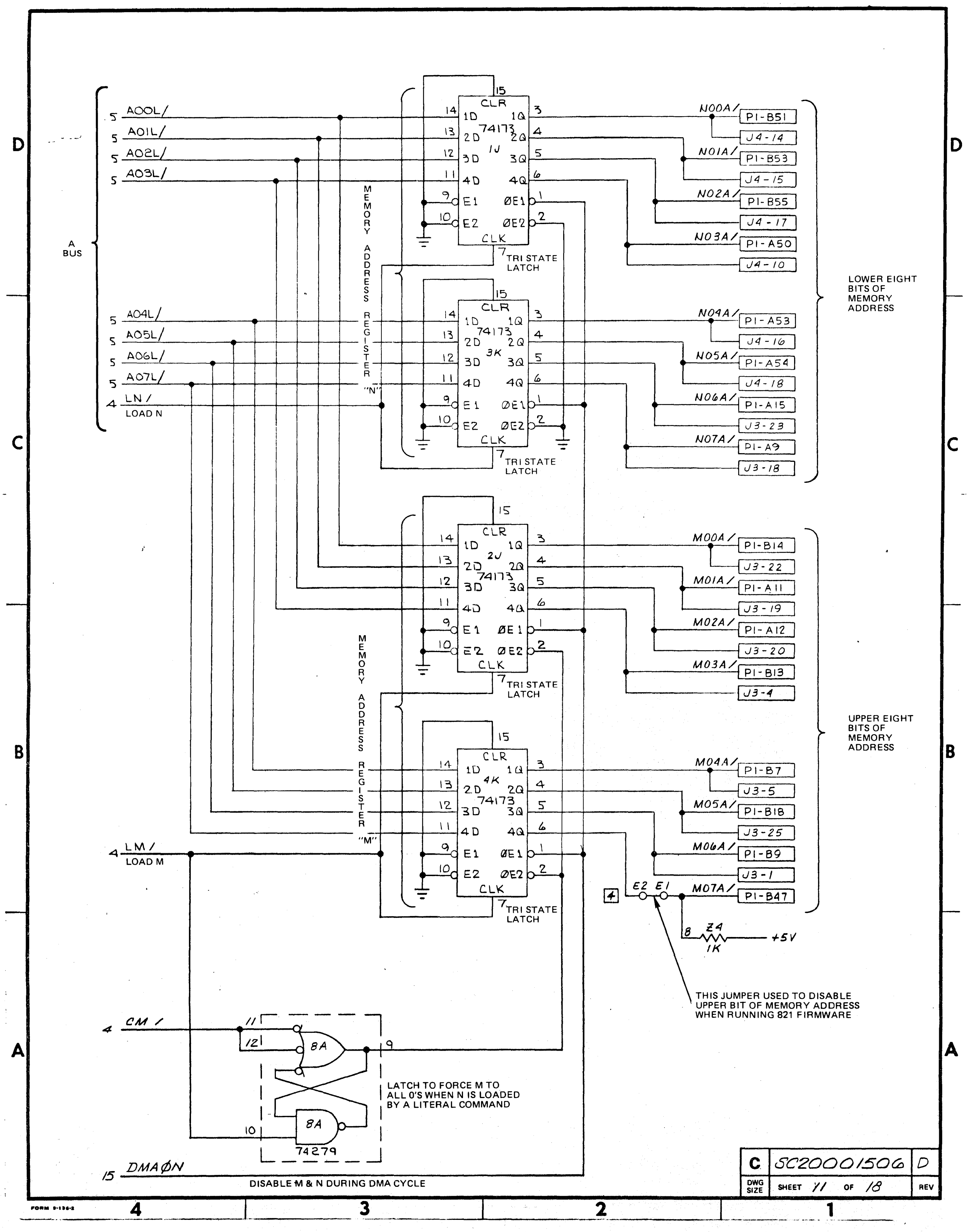
C	SC20001506	D
DWG SIZE	SHEET 9 OF 18	REV

FORM 9-136-2



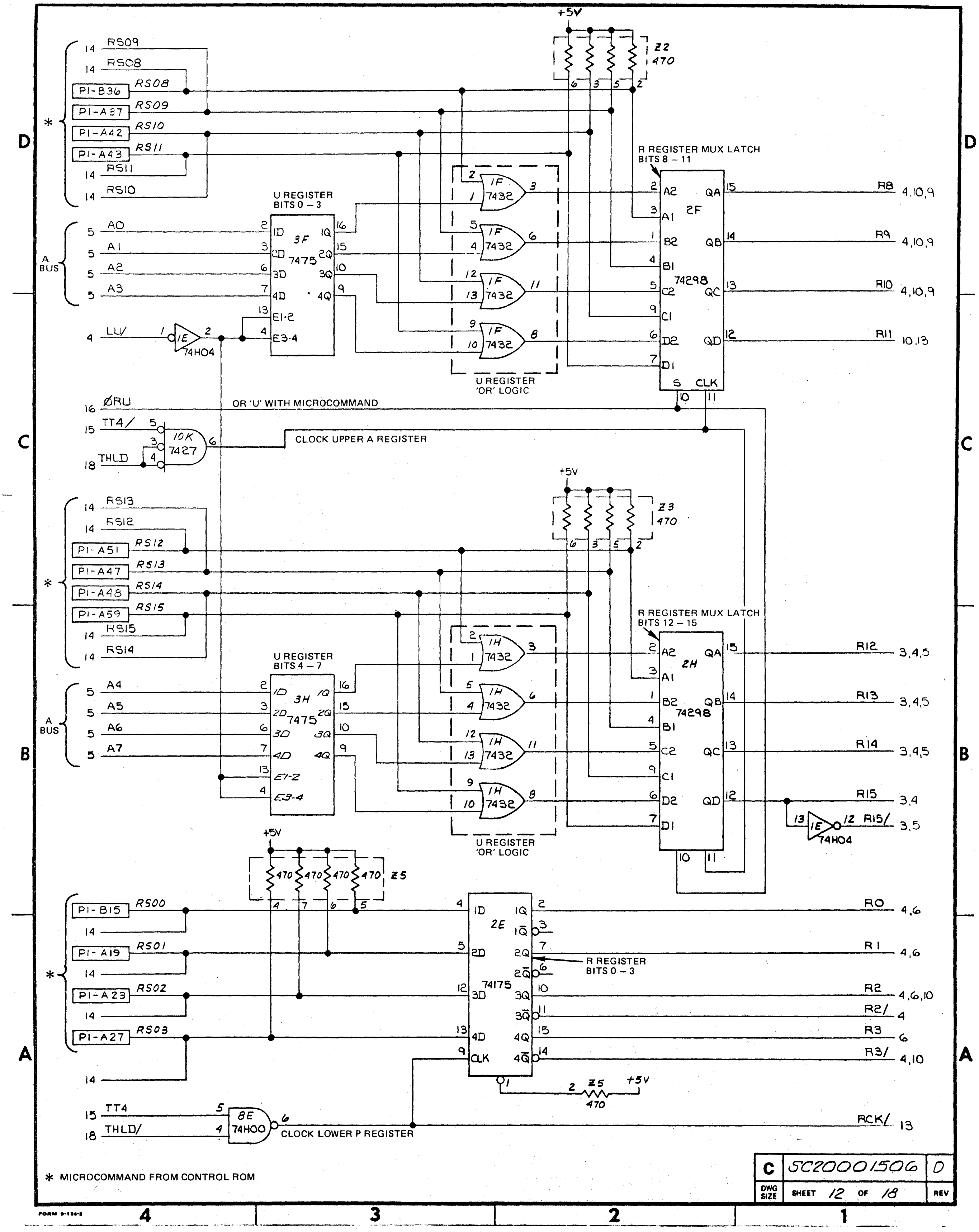
DWG SIZE		REV
C	5C20001506	D
	SHEET 10 OF 18	

FORM 9-139-2



C	SC20001506	D
DWG SIZE	SHEET 11 OF 18	REV

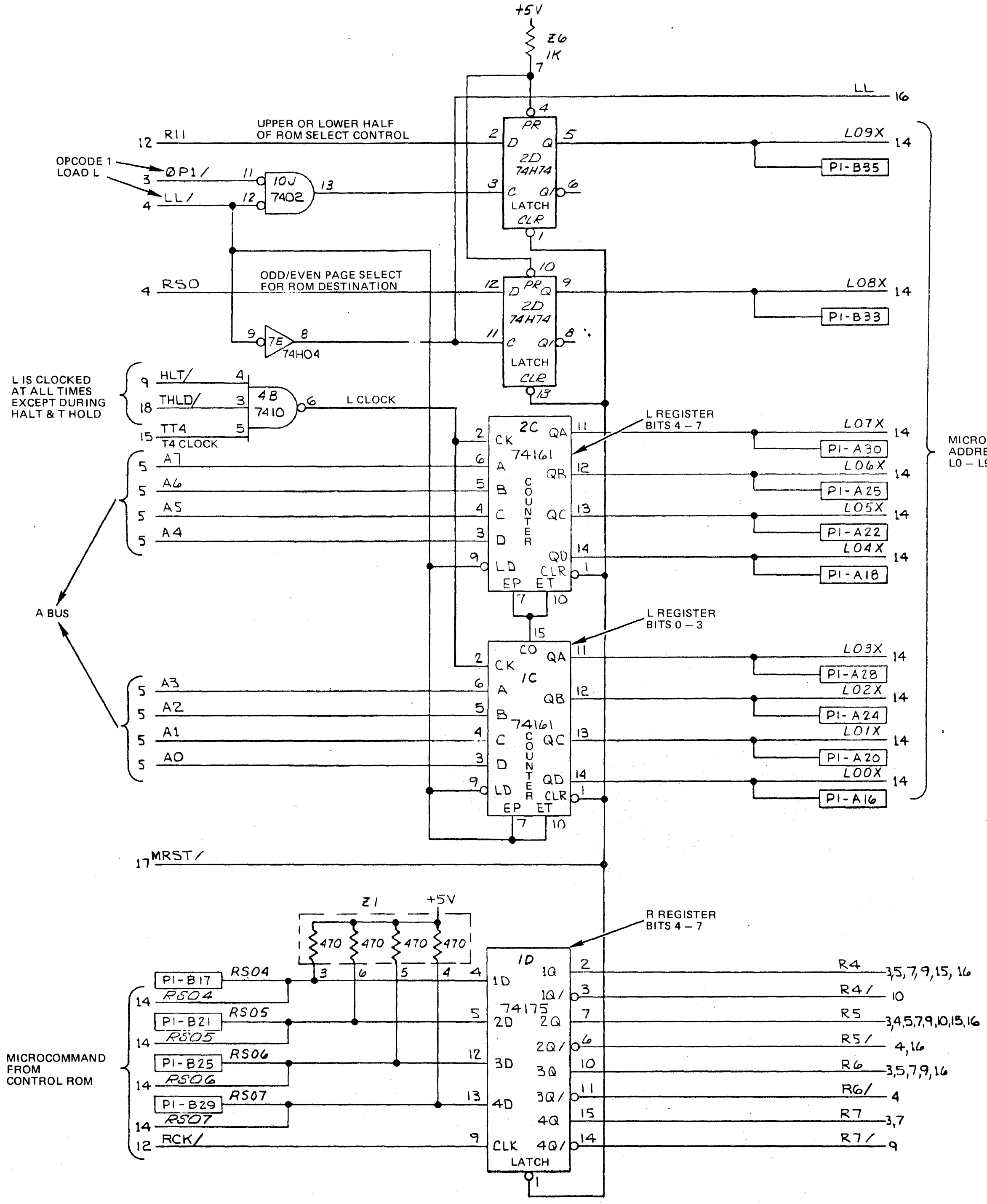
FORM 9-136-2



* MICROCOMMAND FROM CONTROL ROM

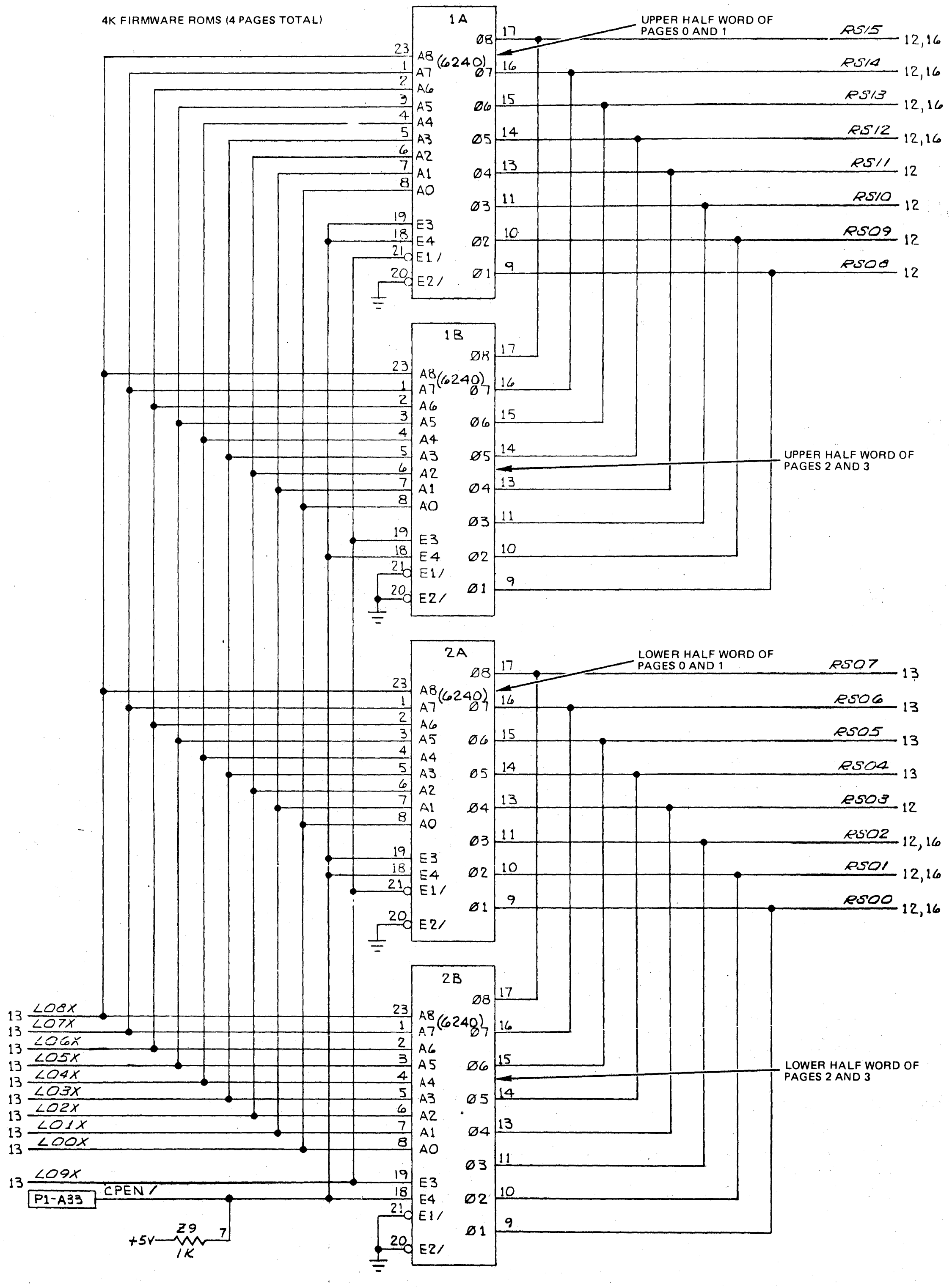
C	SC20001506	D
DWG SIZE	SHEET 12 OF 18	REV

FORM 9-135-2



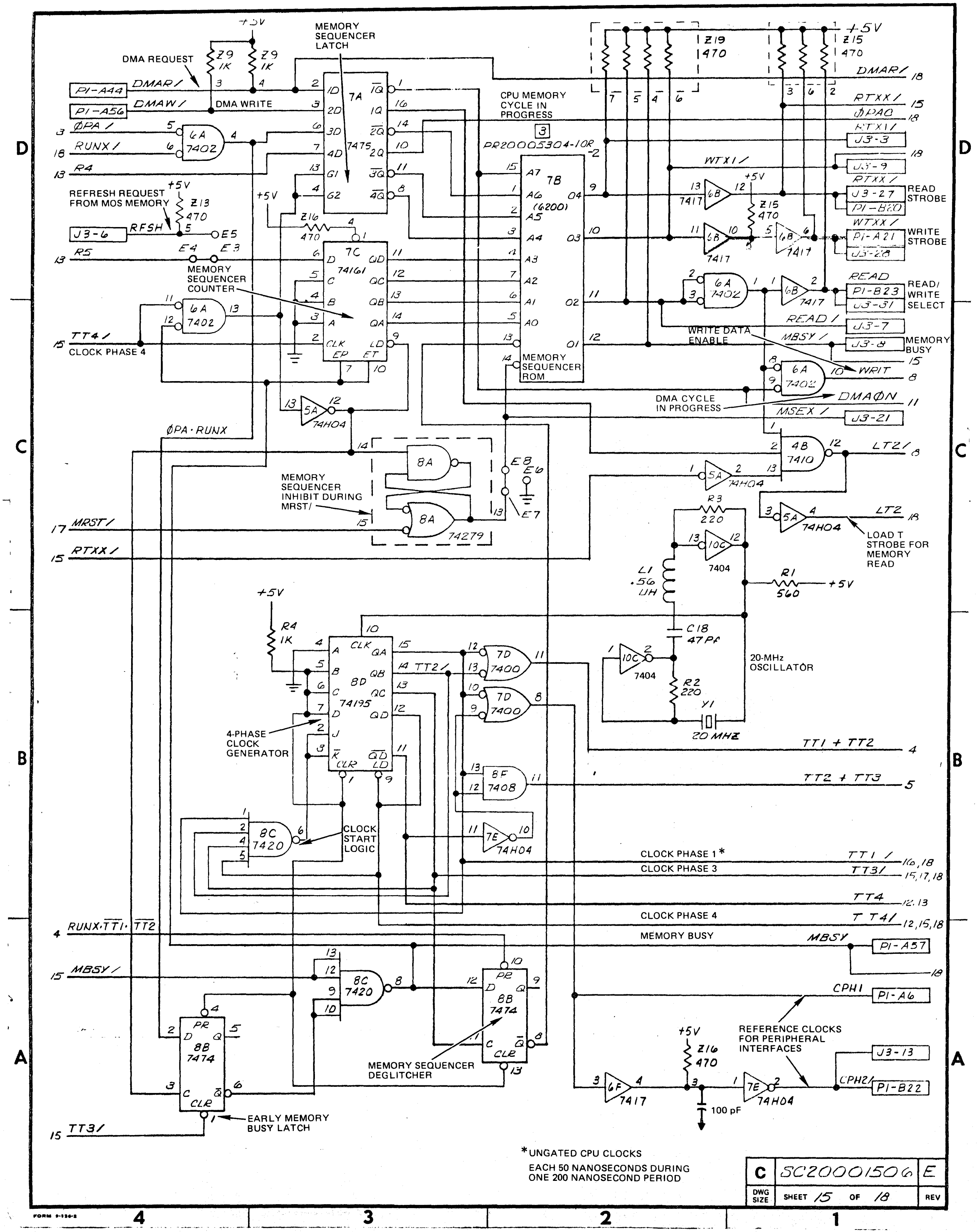
DWG SIZE		REV
C	SC20001506	D
	SHEET 13 OF 18	

4K FIRMWARE ROMS (4 PAGES TOTAL)



4K FIRMWARE ROMS (4 PAGES TOTAL)

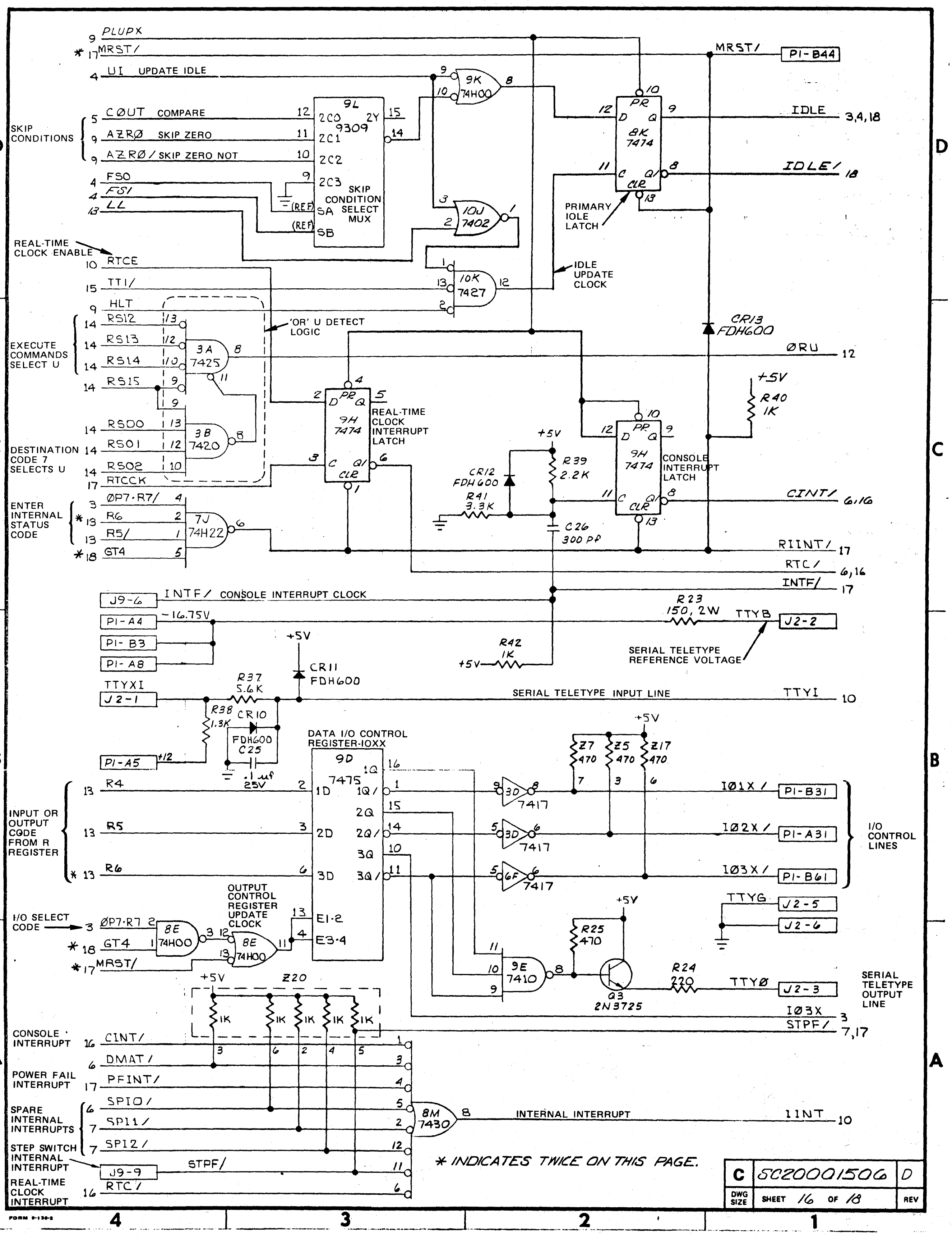
C	5C20001506	D
DWG SIZE	SHEET 14 OF 18	REV



*UNGATED CPU CLOCKS
EACH 50 NANOSECONDS DURING ONE 200 NANOSECOND PERIOD

C	SC20001506	E
DWG SIZE	SHEET 15 OF 18	REV

FORM 9-136-2

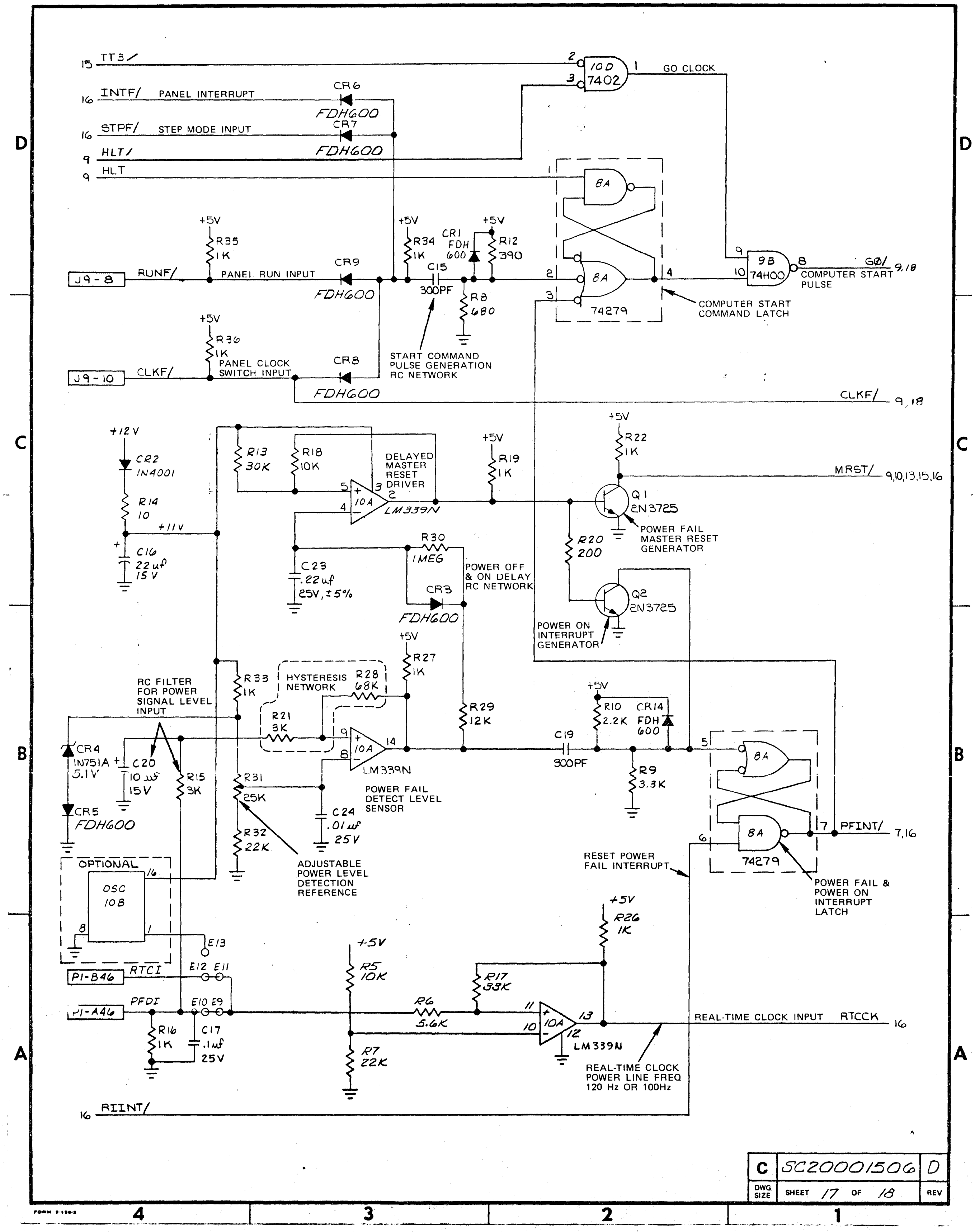


* INDICATES TWICE ON THIS PAGE.

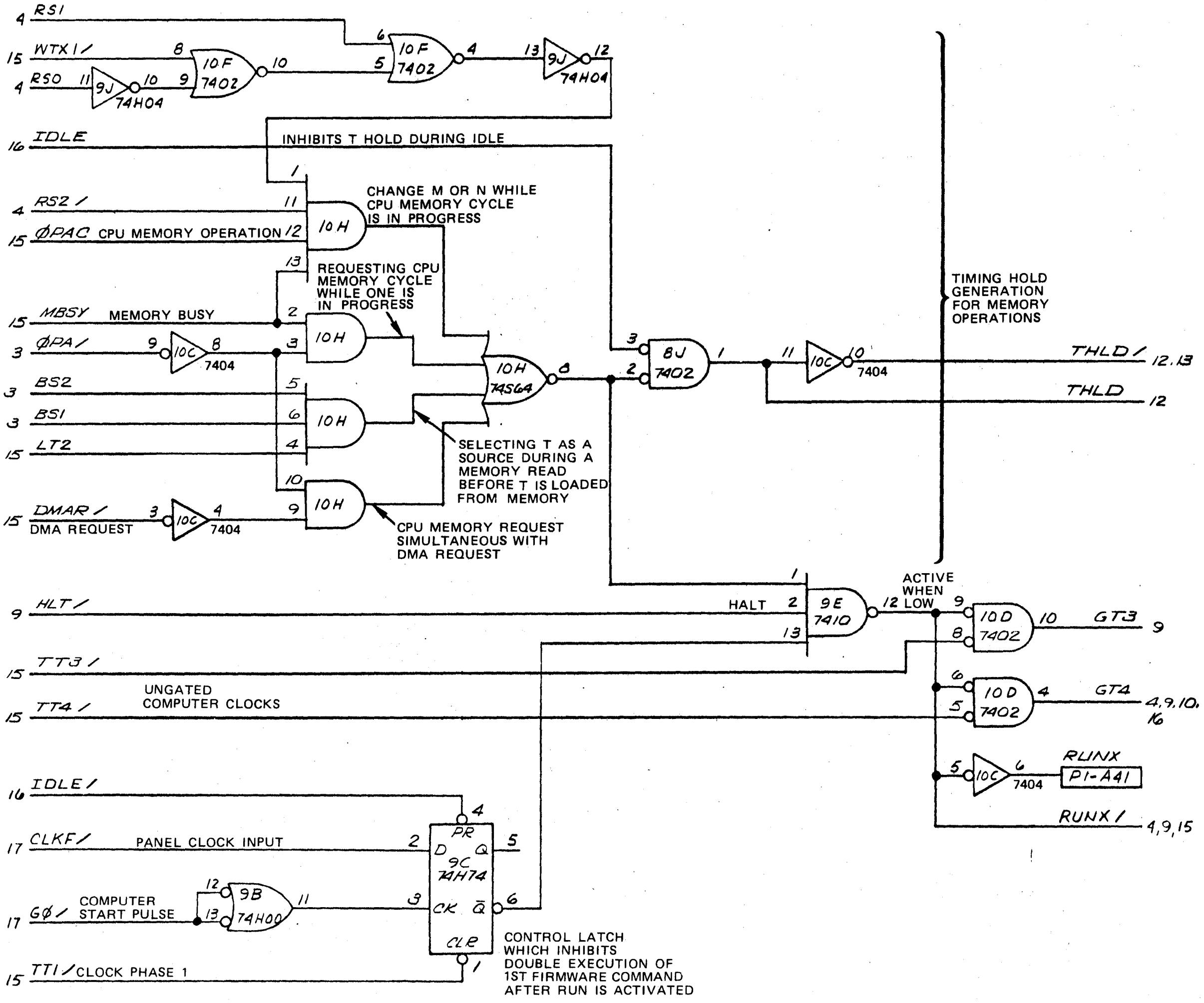
C 5C20001506 D

DWG SIZE SHEET 16 OF 18 REV

FORM 9-130-2



C	SC20001506	D
DWG SIZE	SHEET 17 OF 18	REV


4 RS1
15 WTX1/ 8
4 RS0 11 9J 10 9 10F 7402 10
74H04
6 5 10F 7402 4
13 9J 74H04 12
16 IDLE
INHIBITS T HOLD DURING IDLE
1
4 RS2/ 11
15 ØPAC CPU MEMORY OPERATION 12
10H
CHANGE M OR N WHILE CPU MEMORY CYCLE IS IN PROGRESS
13
REQUESTING CPU MEMORY CYCLE WHILE ONE IS IN PROGRESS
15 MBSY MEMORY BUSY 2
3 ØPA/ 9 10C 8 7404 3
10H
3 BS2 5
3 BS1 6
15 LT2 4
10H
10H 74S64 8
SELECTING T AS A SOURCE DURING A MEMORY READ BEFORE T IS LOADED FROM MEMORY
15 DMAR/ DMA REQUEST 3 10C 4 7404 10 9
10H
CPU MEMORY REQUEST SIMULTANEOUS WITH DMA REQUEST
3 2 8J 7402 1
11 10C 10 7404
THLD/ 12,13
THLD 12
TIMING HOLD GENERATION FOR MEMORY OPERATIONS
1
9 HLT/ HALT 2 9E 7410 12
13
ACTIVE WHEN LOW
9 8 10D 7402 10 GT3 9
15 TT3/
15 TT4/ UNGATED COMPUTER CLOCKS
6 5 10D 7402 4 GT4 4,9,10,16
5 10C 6 7404 RUNX P1-A41
RUNX/ 4,9,15
16 IDLE/ 4
17 CLKF/ PANEL CLOCK INPUT 2
PR D Q 9C 74H74 CK Q̄ CLR
5 6
17 GØ/ COMPUTER START PULSE 12 13 9B 74H00 11 3
15 TT1/ CLOCK PHASE 1 1
CONTROL LATCH WHICH INHIBITS DOUBLE EXECUTION OF 1ST FIRMWARE COMMAND AFTER RUN IS ACTIVATED

C	SC2000150 6	D
DWG SIZE	SHEET 18 OF 18	REV