USER'S MANUAL

FOR

MSC 8009

PRELIMINARY
2-26-81

MSC 8009
TABLE OF CONTENTS

SECTION 1
INTRODUCTION

SECTION 2
Z80 PROCESSOR

SECTION 3
MULTIBUS

SECTION 4
MEMORY

SECTION 5
SERIAL I/O INTERFACE

SECTION 6
FLOPPY-DISK FORMATTER/CONTROLLER

SECTION 7
INTERRUPT

SECTION 8
THEORY OF OPERATION

SECTION 9
ARITHMETIC PROCESSOR UNIT

APPENDICES

DRAWINGS

SECTION 1

INTRODUCTION

1.1 SCOPE

The Monolithic Systems Corporation MSC 8009 is a single-board OEM computer that is directly compatible with the industry standard MULTIBUS*. As software development is a major cost of any computer system, the 8080 software compatibility of the MSC 8009 provides an advanced, high-speed, next-generation system without incurring the costs and delays associated with developing new software. New systems can now be designed taking full advantage of the Z80A* instruction set and optional floating-point arithmetic unit.

The MSC 8009 operates in a multimaster system with either parallel or serial priority resolution. An on-board bus plus the MULTIBUS structure allows other operations to proceed on the MULTIBUS while the Z80A processor uses local memory and I/O devices. Since the on-board memory and I/O resources are extensive, bus access is usually needed only for communication between tasks. System throughput with multiple masters is greatly enhanced because of the light MULTIBUS traffic load. MULTIBUS compatibility means that the MSC 8009 can be used either to expand existing SBC 80-based systems or as the basis of a new design.

An on-board floppy-disk formatter/controller in addition to two serial ports will increase the computing power for most applications. Via a 1793 Floppy-Disk Formatter/Controller, the floppy-disk interface of the MSC 8009 offers a soft-sector format that can be made IBM compatible with the proper software. Variable length sectors and the self-clocking feature of the 1793 means more data per track. The system uses Z80 block I/O instructions to transfer data. Write pre-compensation reduces error rate; and the data separator is crystal controlled. Programmable stepping rates from 3 to 15 milliseconds lets the MSC 8009 operate with drives having different track-to-track access time.

*MULTIBUS is a registered trademark of Intel Corporation

Z80A is a registered trademark of Zilog Inc.

1.2 SPECIFICATIONS

PROCESSOR:

Z80A (4 MHz)

MULTIBUS COMPATIBILITY:

Full MULTIBUS control logic permits up to 16 bus masters (including other MSC CPU's) to share the system bus.

BUS EXCHANGE MODES:

Three bus modes allow exchange of bus master every cycle, every instruction (allows test and set), or never. The program sets the bus modes for optimum control of multi-processing systems.

CYCLE TIME:

The execution of the fastest Z80A instruction require 1.25 microseconds.

FLOPPY DISK INTERFACE:

Format:

Accomodates single- and double-density formats that are compatible with IBM soft-sector configuration.

Read Mode:

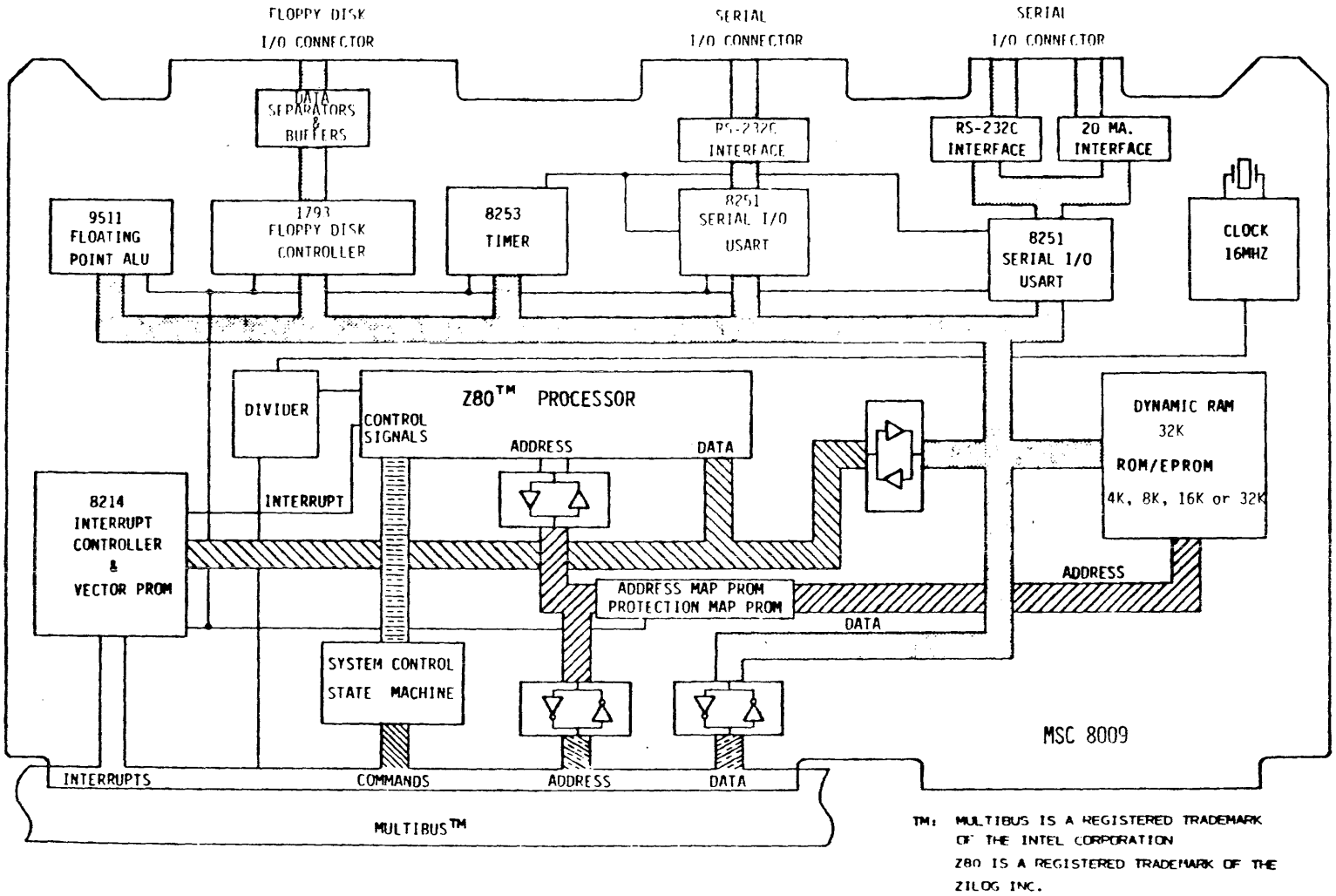
Single/multiple sector read with automatic search or entire track read. Either selectable 128 byte or variable length sector.

Write Mode:

Single/multiple sector write with automatic search. Entire track write capability for diskette formatting.

Supporting Software:

CP/M



TM: MULTIBUS IS A REGISTERED TRADEMARK OF THE INTEL CORPORATION
Z80 IS A REGISTERED TRADEMARK OF THE ZILOG INC.

MSC 8009 MICROCOMPUTER

Capability:

With proper drive options, up to eight 4- or 8-inch disk drives intermixed can be controlled.

Model Compatibility:

The models listed can be accommodated. However, other drives can be used and have not been included.

Shugart SA-400
Shugart SA-450
Shugart SA-800
Shugart SA-850

MEMORY CAPACITY:

RAM - 32K bytes, dynamic

EPROM - The MSC 8009 is shipped with four sockets that accomodate most 8-bit wide memory devices with standard 24-pin callout. Capacities include:

1) To 32K using 8K x 8 masked-ROM elements.
2) To 16K using 4K x 8 EPROM elements.
3) To 8K using 2K x 8 EPROM elements.
4) To 4K using 1K x 8 EPROM elements.

MEMORY MANAGEMENT:

RAM and ROM addressing under PROM control on 1K-byte boundaries. Dual-address map option provides two, complete address maps selectable under program control.

MEMORY PROTECTION:

A protection PROM allows the selection of system resources that are available to other MULTIBUS masters. Any or all of the RAM, ROM or I/O subsystems may be protected from external access.

MEMORY REFRESH:

The memory refresh cycles are automatic and nearly transparent.

DIRECT MEMORY ACCESS:

Another bus master may access the on-board memory or I/O devices (if allowed by the protection PROM) in 625 to 750 nanoseconds.

FLOATING POINT ARITHMETIC:

Two optional Arithmetic Processing Units are available -- the 9511 and the 9512. The 9511 provides 32-bit precision fixed or floating point operations including transcendental functions. For increased performance, the 9512 offers 32- and 64-bit arithmetic operations.

INTERRUPTS:

Vectored, 8-levels of priority interrupts plus Non-Maskable Interrupt (NMI). All on-board interrupt sources are open collector so that a number of devices can share the same level.

Two serial ports provide a programmable interface that can be used for either synchronous or asynchronous operation. One port can be configured for either EIA RS-232-C, TTL or opto-isolated 20 mA loop signals. The other port can be configured for EIA RS-232-C or TTL only. Signal specifications are:

Synchronous:

5- to 8-bit character; one or two programmable SYNC characters.

Asynchronous:

5- to 8-bit character; 1, 1-1/2, or 2 stop bits; choice of parity or error detection.

BAUD Rates:

75 through 9600 BAUD, software selectable.

TIMERS:

One 16-bit counter/timer with six operational modes.

MODE 0 - Interrupt
MODE 1 - Programmable/Retriggerable One-shot
MODE 2 - Pulse Rate Generator
MODE 3 - Squarewave Generator
MODE 4 - Software Triggered Strobe
MODE 5 - Hardware Triggered Strobe
MODE 6 - Baud Rate Generator for one serial port

INTERFACE SIGNALS:

MULTIBUS:

All signals conform with MULTIBUS specifications.

Floppy Disk I/O:

All signals are TTL compatible.

Serial I/O:

Signals fulfill either EIA RS-232-C, TTL or 20 mA current loop convention depending on the MSC 8009 configuration.

Interrupt Requests:

All signals are TTL compatible.

Timer:

All signals are TTL compatible.

POWER REQUIREMENTS:

The MSC 8009 operates with MULTIBUS power supply voltages of +5V, -5V, +12V, and -12V.

PHYSICAL DIMENSIONS:

12 in.(Width) X 6.75 in.(Height) X 0.5 in.(Depth).

1.3 FUNCTIONAL DESCRIPTION

The MSC 8009 design is based on the Z80A microprocessor, which is fully upward compatible with the popular 8080A. The Z80A executes all 8080 instructions without modification. In fact, there are cases where 8080 programs in ROM and PROM can simply be plugged into the MSC 8009 with a significant improvement in performance. However, care must be taken when certain programming techniques have been used in 8080 application programs due to the 4-MHz clock rate of the Z80A. Polling loops or delay-timing routines may require adjustment. An additional flag appears in the flag register for the BCD subtraction feature. Flag differences may be significant in rare cases where uncommon programming techniques have been employed.

1.4 Z80A PROCESSOR

The Z80A processor itself offers features that are beyond those of the 8080 or 8085. The designer or programmer can use these features to reduce system size or further increase the speed of application programs. These features include:

(1) 80 additional instructions
(2) Double compliment of registers
(3) Block transfer I/O instructions
(4) Index registers
(5) BCD subtraction
(6) Two additional interrupt modes
(7) Non-Maskable Interrupt (NMI)
(8) Block search and block move instructions

1.5 ARITHMETIC PROCESSING UNIT (Optional)

The MSC 8009 will accept one of two available Arithmetic Processor Units (APU).

1.5.1 9511 APU

The 9511 enhances the computational capability of the MSC 8009. It is capable of performing 32-bit operations using floating point as well as fixed-point data formats. Data transfers are to or from an internal stack. Commands are issued to a second I/O address to perform an operation on the data that is contained in this stack. The status of the 9511 can be read out at any time from the same I/O address. The results are then made available for retrieval, or an additional command may be entered. If the 9511 is busy, it will cause the Z80 to wait if the CPU tries to access the 9511.

Some of the arithmetic and transcendental functions in addition to control and conversion commands that can be performed with the 9511 include:

(1) Basic Arithmetic Operations (Addition,, Subtraction, Multiplication and Division).
(2) Trigonometric Functions.
(3) Logarithmic Functions (Common and Natural).
(4) Constant Pi (π)
(5) Exponential Functions (e^{x} or X^{y}).
(6) Stack Control (single, double or floating).
(7) Square Root.
(8) Change Signs (single, double or floating).

1.5.2 9512 APU

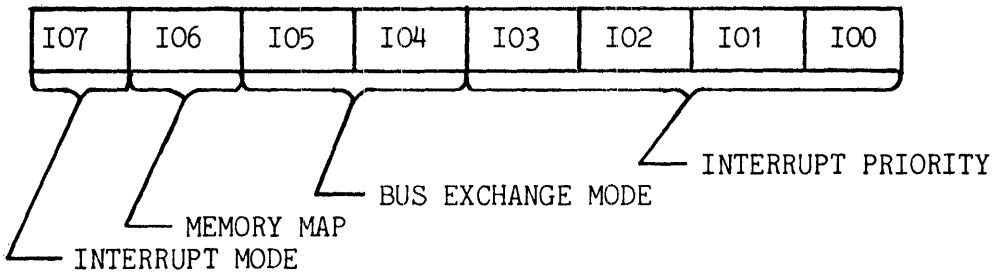
The 9512 provides single-precision (32-bit) and double-precision (64-bit) add, subtract, multiply and divide operations. All operand result, status and command information transfers take place over an 8-bit bidirectional data bus. Using programmed I/O, the user can handle all data transfers between the Z80 and the 9512. Operands are pushed onto an internal stack by the Z80; and a command is issued to perform an operation on the data stack. By popping the stack, the final result is then made available to the Z80.

1.6 BUS INTERFACE

The MSC 8009 uses an internal bus for access to the on-board memory, arithmetic processor, floppy-disk formatter/controller and I/O. Since these on-board tasks do not require the MULTIBUS, the MSC 8009 is able to perform internal operations and not interfere with the MULTIBUS activities. This means that up to 16 bus masters (including other MSC 8009's) can share the MULTIBUS and provide the user with the benefits of multiprocessing.

1.7 MODE STATUS REGISTER

An internal 8-bit register sets up the MSC 8009 via software control for the following indicated modes of operation.



1.7.1 Bus Exchange Modes

Through program control, the user can use bits 4 and 5 (IO4 and IO5) to select one of three bus modes for optimum control of a multiprocessing system. These modes are:

IO5	IO4	BUS OPERATION
0	0	Bus released to higher priority on every data transfer.
1	0	Bus exchange only on M1 processor cycle (Fetch Instruction).
X	1	MSC 8009 keeps the bus.

X = don't care

1.8 READ/WRITE MEMORY

In the basic system, the dual-ported dynamic RAM provides the MSC 8009 with up to 32K-byte storage capacity. The sixteen-chip memory array is partitioned into two, 16K-byte sections. Either the address lines of the Z80A processor or the MULTIBUS (Direct Memory Access) address these arrays. The refresh cycle of the dynamic RAM is automatic. So that requested memory operations can be performed with minimum time delay, the refresh cycle is hidden.

1.8.1 Refresh Cycle

If either a memory read or write operation of the dynamic RAM is in progress, the refresh cycle is inhibited. When the memory cycle is complete, a refresh cycle may be initiated. The refresh control logic attempts to initiate a refresh during an M1 machine cycle, making the refresh transparent. However, if an M1 cycle is not available, the refresh is still inserted.

1.9 READ ONLY MEMORY

Four ROM/EPROM sockets accommodate the most popular 8-bit wide, 24-pin memory devices. These devices normally hold the commonly used subroutines, standard support software and programs for specific applications. By using the 2708 and 2758 (1K X 8EPROM), the MSC 8009 can hold up to 4K bytes of program storage. The 2516 or 2716 (2K X 8 EPROM) provides 8K bytes while either the 2532 or 2732 4K X 8 EPROM offers the user 16K bytes. For 32K byte capacity, the user can use the 8K X 8 masked ROMs.

Since each ROM/EPROM device may vary in power configuration, hardware jumpers provide the required voltages. Different chips may be intermixed, but three of the four sockets must contain the same devices. For example, standard support software may require three 2716's (2K X 8 EPROM) and one 8K X 8 Masked ROM -- total capacity of 14K bytes.

1.9.1 Protection PROM

A special PROM protects any or all RAM, ROM or I/O subsystems from external access. This device generates an internal enabling signal that permits the bus to address the MSC 8009 only when called for. A pin on the auxilliary connector totally disables access to the MSC 8009 board, allowing multiple processors to be paged into the same memory space.

1.10 DUAL MAP CONFIGURATION (Optional)

The optional DUAL MAP feature provides the MSC 8009 with two complete address maps. The system always powers up using the first map. Since the system has to have a ROM at location zero to start properly, a ROM should be mapped into low memory. For example, the user can switch to a map that contains RAM in low memory. This is achieved through a bit sent with the same output instruction used for programming the interrupt controller (See paragraph 1.7). A "0" in bit position six represents a start up condition, and a "1" switches the map. This capability permits the use of software written for 8080 and Z80A processors that previously could not be run on a single-board computer.

1.11 I/O INTERFACE

Two serial I/O interfaces provide the MSC 8009 with serial-data communication channels that are programmable and will operate either synchronously or asynchronously based on the current serial-data transmission protocols. The floppy disk interface can support up to eight soft-sector drives. Since an address decoder PROM defines the MSC 8009 address structure, any I/O device on-board may be assigned any of the 256 device address codes used by the 8080 or Z80A instruction set.

1.11.1 Serial I/O Interfaces

The two serial I/O interfaces are designed around two software-programmable devices -- an 8251 USART (Universal/Asynchronous Receiver/Transmitter) and an 8253 Programmable Timer. The user can configure one channel only for either EIA RS-232-C, TTL or opto-isolated 20 mA current-loop operation. The other channel can be configured for either EIA RS-232-C or TTL operation.

NOTE: For current loop operation, the external device supplies the current source for proper operation.

The user's program selects the mode of operation, data format, control character format, parity and BAUD rate. The 8251 provides full duplex, double-buffered transmit and receive capabilities. Also, parity, overrun and framing error detection are incorporated within the USART. One section of the 8253 supplies the BAUD rate clock under program control to all channels optionally or additional sections can be used for different baud rates on the other serial channels.

1.12 FLOPPY DISK INTERFACE

An onboard chip allows the MSC 8009 to interface with up to eight 4- or 8-inch drives intermixed. The commands listed in the following Table Can be used to perform the desired floppy-disk operation.

Restore
Seek
Step
Step In
Step Out
Read Sector
Write Sector
Read Address
Read Track
Write Track

1.12.1 Disk Format

Disks may be formatted to be compatible with either Shugart drives, IBM 3740 or System 34 formats with Sector lengths of 128, 256, 512, or 1024 bytes.

Basically a recorded sector on disk consists of two fields -- the ID FIELD and the DATA FIELD. The ID FIELD contains the ID Address Marks, track number, side number, sector number, sector length, and the Cyclic Redundant Check bits (CRC). The information contained within the ID FIELD must be found within four revolutions of the disk; otherwise, a "record-not-found" condition will be set up. The sector length defines the number of bytes per sector.

The DATA FIELD consists of its Address mark, data and CRC bits. For double-density operation, there will be three bytes preceding the Address Marks of the ID FIELD and DATA FIELD with the clock transition missing between bits 4 and 5.

1.12.2 Status Register

An 8-bit Status Register, internal to the formatter/controller chip, provides the user with the status of the floppy-disk operation. During the execution of a command, a status bit will be set, which can be monitored. This bit is reset when the current instruction terminates. The remaining bits are updated in accordance to the instruction.

1.12.2.1 Address Marks

For synchronization, each track has a unique combination of data and clock bits called "Address Marks". These four distinct marks are:

Index
ID Address
Data Address
Deleted Address

1.12.2.2 Cyclic Redundancy Check Characters (CRC)

All information beginning with an address mark and up to the actual CRC characters themselves establishes the contents of the Cyclic Redundancy Check characters. Each field recorded on disk is appended with two CRC character bytes. These bytes are generated from a cyclic permutation of the data bits starting with bit zero of the address mark and terminating with bit zero of the last byte within a field (excluding CRC bytes). When a field is read, the data bits are divided by the same general polynomial. A nonzero remainder indicates invalid data, while a zero remainder denotes that correct data has been read.

1.13 INTERRUPT

The MSC 8009 provides vectoring for eight levels of priority interrupt in addition to a Non-Maskable Interrupt (NMI). Three operating modes nd priority assignments may be configured anytime during system opertion via software control. These modes include:

(1) MODE 0 generates RST instructions, identical with 8080 vectoring.

(2) MODE 1 vectors all levels to location 38H (RST7) independent of the interrupt vector hardware. This mode is useful for debugging purposes because all interrupts execute a restart at location 38H.

(3) MODE 2 uses a PROM to specify the eight, low-order bits of interrupt-vector address for each of the eight, priority levels. The Z80A interrupt vector register defines the eight, high-order bits that are set by the program. Consequently, any interrupt can be vectored to any location in memory under program control.

1.13.1 Non-Maskable Interrupt

The Non-Maskable Interrupt (NMI) feature of the Z80A provides the MSC 8009 with an effective and efficient method of dealing with system power failure or for diagnosing certain hardware and software problems. A power-fail service routine can be added to an existing SBC 80 program without major changes because the NMI functions independently of existing interrupt hardware.

A pin of the SBC auxiliary connector P2 is assigned to NMI. Also, NMI may be connected to pin 33 of P1 (MULTIBUS connector), but this is a reserved bus pin and may conflict with some applications. NMI can be used to detect error conditions or abnormal system situations, but should not be used in the course of normal program execution. The power-fail module can make use of NMI so that system status is saved and used for restart information. NMI always vectors to location 66H.

1.14 SYSTEM CONTROL

The majority of MSC 8009 logic is synchronous with the master clock. Additional independent system-timing functions include power up initialization and a Watchdog Timer.

1.14.1 System Clock

The MSC 8009 uses a standard, master-clock frequency of 16 MHz to derive the 8-MHz Bus clock. A number of MSC 8009 functions are synchronized with the master clock (K1115A device).

1.14.2 Power Up

When powering up the MSC 8009 for operation, the -5V must be stable prior to applying +12V. During the power down sequence, the +12V must be removed before the -5V.

An open-collector driver produces the initializing signal INIT/. This signal is held for at least 50 milliseconds after power up, conditioning the MSC 8009 for operation. In place of the driver, an external device can also be used to initialize the system via the MULTIBUS.

1.14.3 Memory And I/O Addressing

A bipolar, fusible-link PROM defines the memory address and I/O structure, for the MSC 8009. Jumpers and switches have been eliminated to increase reliability and provide more flexibility in memory allocation. Any memory device may be assigned to start on any 256-byte boundary in the 64K addressable space. The memory and I/O allocations of the standard figure are suitable for most applications. However, if care is taken to avoid overlapping device addresses, the address-decoder PROM contents may be reprogrammed to satisfy even the most unusual memory or I/O address mapping requirements. Two complete address maps may be stored in an optional extended PROM.

1.14.4 Watchdog Timer

The Watchdog Timer is set to 4 seconds so that certain program and system failures can be detected and evaluated. An example is the referencing of non-existent memory or I/O devices. The Watchdog Timer may be disconnected from NMI/. If desired, the timing can be changed by replacing one capacitor.

SECTION 2

Z80 PROCESSOR

2.1 SCOPE

This section discusses and summarizes the Z80 instruction set that is used in programming the MSC 8009. The instructions are grouped according to the type of instruction. Each instruction is described using the assembly language mnemonic operation code; the instruction name; the symbolic operation; a description; the binary fields and pattern that make up the machine instruction; and the number of memory cycles, machine states and affected flags.

The mnemonic operation codes used here differ in some cases from those used by Zilog so as to be compatible with commonly used 8080 (Intel) mnemonics. The equivalent Zilog operation codes appear to the upper right of each instruction description.

2.2 INSTRUCTION AND DATA FORMAT

The program instruction may be 1, 2, 3 or 4 bytes (8 bits = 1 byte) in length. A multiple-byte instruction is stored in successive memory locations with the first byte address as the instruction address. The exact format depends on the operation that is to be performed.

As with instructions, multi-byte data are stored in successive memory locations with the least-significant byte first.

2.3 ADDRESS MODES

There are four modes of addressing data that are stored either in memory or in processor registers.

1) DIRECT Mode uses byte 2 and byte 3 of the instruction to define the desired memory location of the data. Byte 2 contains the low-order bits of the address and byte 3, the high-order bits.

2) REGISTER Mode uses a 2 or 3 bit field in the instruction to specify the register or register pair containing the data.

3) REGISTER INDIRECT Mode uses a 2 or 3 bit field in the instruction to define a register pair that contains the memory address of the data location. The first register holds the high-order bits of the address and the second, the low-order bits.

4) IMMEDIATE Mode uses additional instruction bytes to store the data. Data can be either 8 bits or 16 bits in length (least-significant byte first followed by the most-significant byte).

2.3.1 Branching Instructions

Normally, the execution of an instruction procedes sequentially through increasing memory address. However, a branching instruction can specify the address of the next instruction to be executed in one of two ways. Jumps and Calls are branching instructions.

1) The branch instruction holds the address of the next instruction that is to be executed, where byte 2 contains the low-order address and byte 3 the high-order.

2) The branch instruction defines a register pair that contains the address of the next instruction to be executed. The first register provides the high-order bits of the address and the second, the low-order bits.

An exception to these two cases is the 'RST' instruction.

2.3.2 Restart Instruction (RST)

This is a special 1 byte call instruction that is usually used during interrupt sequences. A 3 bit field in the RST instruction determines one of eight fixed vector addresses. The program control is transferred to the instruction that has an address eight times the contents of this 3 bit field.

2.4 FLAG

The flag register (F and F') supply information regarding the Z80 status at any specific time. Bit position for each flag is shown below:

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

where:

C = CARRY FLAG
N = ADD/SUBTRACT FLAG
P/V = PARITY/OVERFLOW FLAG (1 indicates even parity).
H = HALF-CARRY FLAG
Z = ZERO FLAG
S = SIGN FLAG
X = NOT USED

Each of the two Z80 Flag Registers contain 6 bits of status information that are either set or reset by CPU operations. Bits 0, 2, 6 and 7 are testable and used with conditional Jump, Call or Return instructions. Flags H and N are associated with BCD arithmetic and are not directly testable.

2.4.1 Carry Flag (C)

Either an ADD instruction that generates a carry or a SUBTRACT instruction that generates a borrow, sets the CARRY FLAG. This flag is then reset if a carry is not generated or a borrow does not occur. This is convenient for extended precision arithmetic. Also, the 'DAA' instruction sets the CARRY FLAG if the conditions for a decimal adjustment are fulfilled.

For Rotate and Shift instructions, the CARRY FLAG provides a link between the least-significant and most-significant bit for any register or memory location. For example, the carry holds the last bit shifted from bit 7 of the register or memory location during a Rotate or Shift Left instruction. For a Rotate or Shift Right, the carry represents bit 0 of the register or memory location.

Logical instruction AND, OR, and XOR reset the CARRY FLAG. Also, the CARRY FLAG can be set through instruction 'SCF' or complemented with 'CCF'.

2.4.2 Add/Subtract Flag (N)

The Decimal Adjust Accumulator instruction (DAA) uses the ADD/SUBTRACT FLAG to distinguish between 'ADD' and 'SUBTRACT' operations. For all 'ADD' operations, this flag is reset and for all 'SUBTRACT' operations, this flag is set.

2.4.3 Parity/Overflow Flag (P/V)

The PARITY/OVERFLOW FLAG is set to a specific state that depends on the operation.

2.4.3.1 Arithmetic Operations

An overflow condition sets this flag to indicate that the result in the accumulator is either greater than the maximum number (+127) or less than the minimum number (-128). The sign bit of the operand denotes the overflow condition.

For addition, operands with different signs never cause an overflow condition. When adding operands with like signs, a sum with a different sign sets the PARITY/OVERFLOW FLAG. The following example illustrates this situation.

```
ADDEND:  +120 (decimal) = 0111 1000 (binary)
AUGEND:  +105 (decimal) = 0110 1001 (binary)

   SUM:  +225 (decimal) = 1110 0001 (binary)
```

The binary sum represents -95, which is incorrect, therefore the PARITY/OVERFLOW FLAG is set.

An overflow can occur for operands of unlike signs in a subtraction operation. Consider the following example.

```
   MINUEND:  +127 (decimal) = 0111 1111 (binary)
SUBTRAHEND:(-)-64 (decimal) = 1100 0000 (binary)

DIFFERENCE:  +191 (decimal) = 1011 1111 (binary)
```

The minuend sign has changed from a positive to a negative, giving an incorrect difference -- an overflow condition.

Another way to predict overflow is to observe the carry to and from the sign bit. If there is a carry in and no carry out, or if there is no carry in and a carry out, then overflow has occurred.

2.4.3.2 Logical Operations

The PARITY/OVERFLOW FLAG finds use as a parity indicator for logical and rotate operations. The number of "1" bits in a byte are counted and an odd number sets the flag to "0" -- odd parity. If the total is even, the parity flag is set to "1". Also, when inputting a byte from an I/O device, certain input instructions affect the parity flag to indicate the parity of the incoming data.

2.4.4 Half-Carry Flag (H)

The carry and borrow status between accumulator bits 3 and 4 during an 8-bit arithmetic operation affects the condition of the HALF-CARRY FLAG. The Decimal Adjust Accumulator instruction (DAA) uses this flag to correct the result of a packed BCD add or subtract operation. This flag is set according to the following table.

H	ADD	SUBTRACT
1	There is a carry from bit 3 to bit 4.	There is no borrow from bit 4.
0	There is no carry from bit 3 to bit 4.	There is a borrow from bit 4.

2.4.5 Zero Flag (Z)

If the result when executing an appropriate instruction is 0, the ZERO FLAG is set. The ZERO FLAG is always "1" if the resulting byte in the accumulator is 0 when performing an 8-bit arithmetic operation. A non-0 result resets the flag.

When performing block-compare (search) instructions, a subtract is performed without affecting the accumulator contents. The Z flag is set to a "1" if an equivalence is found between the accumulator value and the memory location pointed to by the contents of the register pair HL. The ZERO FLAG indicates the complemented state of specific bits when testing a bit in a memory location or register.

If the result of B - 1 is 0 when inputting or outputting a byte between a memory location and I/O device using a block I/O instruction, the ZERO FLAG is set, otherwise it is reset. Also when using the input instruction INP r, the ZERO FLAG is set to indicate a 0 byte input.

2.4.6 Sign Flag (S)

The SIGN FLAG stores the state of the most-significant bit of the accumulator (bit 7). When the processor performs arithmetic operations on signed numbers, binary two's complement notation is used to represent and process numeric information. A "0" in bit 7 denotes a positive number and a "1" signifies a negative number. The binary equivalent of a positive number is stored in bits 0 thru 6 for a total range of 0 to 127. The two's complement of the positive number represents a negative number with a total range of -1 to -128. When inputting a byte from an I/O device to a register, input instruction INP r, the SIGN FLAG indicates either positive (S = 0) or negative (S = 1) data.

2.5 OP CODE SUMMARY

The following symbols, abbreviations and mnemonics will be used to describe the instruction set with exceptions noted where appropriate. So that the information is easy to use, a shorthand notation is employed for describing the assembler format of the instruction and its actual operation. All capital letters and special characters in the mnemonic descriptions are required. The lower-case characters indicate a class of values that can be inserted in the instruction at that point. A single, lower-case letter indicates an eight-bit quantity or register; while a double, lower-case letter denotes a sixteen-bit quantity or register. A symbol enclosed in parenthesis under the NOTATION heading indicates that the value whose address is specified is used.

2.5.1 Mnemonic Operand Symbols

2.5.1.1 8-Bit Operation

REGISTER	PATTERN DESIGNATION	DESCRIPTION
A	111	Accumulator
B	000	B Register
C	001	C Register
D	010	D Register
E	011	E Register
H	100	H Register
L	101	L Register
I		Interrupt Vector Register
R		Refresh Register
M	110	Memory Location addressed by HL pair

BYT	Number of bytes.
CYC	Number of cycles.
PG	Section 2 page number.
S	Number of states.
b	A bit position in an 8-bit byte, where the bits are numbered from right to left 0 to 7.
e	Relative 8-bit address (-126 to +129).
n	Any 8-bit absolute value.
i	An index register reference, either X or Y
d	An 8-bit index displacement where -128<d<127
r	8-bit register
rr	B for the BC register pair, D for the DE pair, H for the HL pair, and SP for the stack pointer.
p	interrupt vector number (0-7)

Vector	Address
0	00H
1	08H
2	10H
3	18H
4	20H
5	28H
6	30H
7	38H

s	Any of r (defined above) or M.
IFF	Interrupt flip-flop.
CY	Carry flip-flop.
ZF	Zero flag.
v[n]	Bit n of the 8-bit value or register v.
v[n-m]	Bits n through m of the 8-bit value of register v.
Iv	An input operation on port v.
Ov	An output operation on port v.
w<=v	The value of w is replaced by the value of v.
w<=>v	The value of w is exchanged with the value of v.

2.5.1.2 16-Bit Operation

PSW	AF Register Pair processor status word
B	BC Register Pair
H	HL Register Pair
SP	Stack Pointer Register
PG	Section 2 page number.
PC	Program Counter
IX	X-Index Register
IY	Y-Index Register
zz	B for the BC register pair, D for the DE pair.
nn	Any 16-bit value, absolute or relocatable.
qq	B for the BC register pair, D for the DE pair, H for the HL pair, and PSW for the A/Flag pair.
tt	B for the BC register pair, D for the DE pair, SP for the stack pointer, and X for index register IX.
uu	B for the BC register pair, D for the DE pair, SP for the stack pointer, and Y for index register IY.
vvH	The most-significant byte of the 16-bit value or register vv.
vvL	The least-significant byte of the 16-bit value or register vv.

2.5.2 Flag Symbols

X - Flag affected.
* - Flag unaffected.
P - Parity flag affected according to parity result of operation.
V - Overflow flag affected according to the overflow result.
0 - Flag reset.
1 - Flag set.
? - Flag unspecified.

2.5.3 8-Bit Data Transfer Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
MOV r,r'	22	r<=r'	1	4	1	*	*	*	*	*	*
MOV r,M	22	r<=(HL)	2	7	1	*	*	*	*	*	*
MOViR d,r	22	(i+d)<=r	5	19	3	*	*	*	*	*	*
MOV M,r	23	(HL)<=r	2	7	1	*	*	*	*	*	*
MOVRi r,d	23	r<=(i+d)	5	19	3	*	*	*	*	*	*
MVI r,n	24	r<=n	2	7	2	*	*	*	*	*	*
MVI M,n	24	(HL)<=n	3	10	2	*	*	*	*	*	*
MVIi d,n	25	(i+d)<=n	5	19	4	*	*	*	*	*	*
LDA nn	25	A<=(nn)	4	13	3	*	*	*	*	*	*
STA nn	26	(nn)<=A	4	13	3	*	*	*	*	*	*
LDAX zz	26	A<=(zz)	2	7	1	*	*	*	*	*	*
STAX zz	26	(zz)<=A	2	7	1	*	*	*	*	*	*
LDAI	27	A<=I	2	9	2	X	X	0	IFF	0	*
LDAR	27	A<=R	2	9	2	X	X	0	IFF	0	*
STAI	28	I<=A	2	9	2	*	*	*	*	*	*
STAR	28	R<=A	2	9	2	*	*	*	*	*	*

2.5.4 16-Bit Data Transfer Group

INSTRUCTION			NOTATION	CYC	S	BYT	FLAG					
OP CODE		PG					S	Z	H	P/V	N	C
LXI	rr,nn	29	rr<=nn	3	10	3	*	*	*	*	*	*
LXIi	nn	29	i<=nn	4	14	4	*	*	*	*	*	*
LBCD	nn	30	B<=(nn+1) C<=(nn)	6	20	4	*	*	*	*	*	*
LDED	nn	30	D<=(nn+1) E<=(nn)	6	20	4	*	*	*	*	*	*
LHLD	nn	30	H<=(nn+1) L<=(nn)	5	16	3	*	*	*	*	*	*
LIiD	nn	32	IiH<=(nn+1) IiL<=(nn)	6	20	4	*	*	*	*	*	*
LSPD	nn	30	SPH<=(nn+1) SPL<=(nn)	6	20	4	*	*	*	*	*	*
SBCD	nn	31	(nn+1)<=B (nn)<=C	6	20	4	*	*	*	*	*	*
SDED	nn	31	(nn+1)<=D (nn)<=E	6	20	4	*	*	*	*	*	*
SHLD	nn	31	(nn+1)<=H (nn)<=L	5	16	3	*	*	*	*	*	*
SIiD	nn	32	(nn+1)<=IiH (nn)<=IiL	6	20	4	*	*	*	*	*	*
SSPD	nn	31	(nn+1)<=SPH (nn)<=SPL	6	20	4	*	*	*	*	*	*
SPHL		33	SP<=HL	1	6	1	*	*	*	*	*	*
SPIi		33	SP<=Ii	2	10	2	*	*	*	*	*	*
PUSH	qq	34	(SP-1)<=qqH (SP-2)<=qqL SP<=SP-2	3	11	1	*	*	*	*	*	*
PUSHi		34	(SP-1)<=iH (SP-2)<=iL SP<=SP-2	3	15	2	*	*	*	*	*	*
POP	qq	35	qqH<=(SP+1) qqL<=(SP) SP<=SP+2	3	10	1	*	*	*	*	*	*
POPi		35	iH<=(SP+1) iL<=(SP) SP<=SP+2	4	14	2	*	*	*	*	*	*

2.5.5 Exchange, Block Transfer and Search Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
XCHG	36	HL<=>DE	1	4	1	*	*	*	*	*	*
EXAF	36	PSW<=>PSW'	1	4	1	*	*	*	*	*	*
EXX	36	BC DE HL<=>BC'DE'HL'	1	4	1	*	*	*	*	*	*
XTHL	37	H<=>(SP+1) L<=>(SP)	5	19	1	*	*	*	*	*	*
XTIi	37	IiH<=>(SP+1) Iil<=>(SP)	6	23	2	*	*	*	*	*	*
LDI	38	(DE)<=(HL) DE<=DE+1 HL<=HL+1 BC<=BC-1	4	16	2	*	*	0	X	0	*
LDIR	38	Repeat LDI until BC=0.	5/4	21/16	2	*	*	0	0	0	*
LDD	39	(DE)<=(HL) DE<=DE-1 HL<=HL-1 BC<=BC-1	4	16	2	*	*	0	X	0	*
LDDR	39	Repeat LDD until BC=0	5/4	21/16	2	*	*	0	0	0	*
CCI	41	A-(HL) HL<=HL+1 BC<=BC-1	4	16	2	X	X	X	X	*	*
CCIR	41	Repeat CCI until A=(HL) or BC=0	5/4	21/16	2	X	X	X	X	1	*
CCD	40	A-(HL) HL<=HL-1 BC<=BC-1	4	16	2	X	X	X	X	1	*
CCDR	40	Repeat CCD until A=(HL) or BC=0	5/4	21/16	2	X	X	X	X	1	*

2.5.6 8-Bit Arithmetic And Logical Group

INSTRUCTION			NOTATION	CYC	S	BYT	FLAG					
OP CODE		PG					S	Z	H	P/V	N	C
ADD	r	42	A<=A+r	1	4	1	X	X	X	V	0	X
ADD	M	42	A<=A+(HL)	2	7	1	X	X	X	V	0	X
ADDi	d	43	A<=A+(i+d)	5	19	3	X	X	X	V	0	X
ADI	n	43	A<=A+n	2	7	2	X	X	X	V	0	X
ADC	r	43	A<=A+r+CY	1	4	1	X	X	X	V	0	X
ADC	M	44	A<=A+(HL)+CY	2	7	1	X	X	X	V	0	X
ADCi	d	45	A<=A+(i+d)+CY	5	19	3	X	X	X	V	0	X
ACI	n	44	A<=A+n+CY	2	7	2	X	X	X	V	0	X
SUB	r	45	A<=A-r	1	4	1	X	X	X	V	1	X
SUB	M	46	A<=A-(HL)	2	7	1	X	X	X	V	1	X
SUBi	d	47	A<=A-(i+d)	5	19	3	X	X	X	X	1	X
SUI	n	46	A<=A-n	2	7	2	X	X	X	V	1	X
SBB	r	47	A<=A-r-CY	1	4	1	X	X	X	V	1	X
SBB	M	48	A<=A-(HL)-CY	2	7	1	X	X	X	V	1	X
SBBi	d	49	A<=A-(i+d)-CY	5	19	3	X	X	X	V	1	X
SBI	n	48	A<=A-n-CY	2	7	2	X	X	X	V	1	X
ANA	r	54	A<=A&r	1	4	0	X	X	1	P	1	0
ANA	M	54	A<=A&(HL)	2	7	1	X	X	1	P	0	0
ANAi	d	55	A<=A&(i+d)	5	19	3	X	X	0	P	1	0
ANI	n	55	A<=A&n	2	7	2	X	X	1	P	0	0
ORA	r	56	A<=A!r	1	4	1	X	X	1	P	0	0
ORA	M	56	A<=A!(HL)	2	7	1	X	X	1	P	0	0
ORAi	d	57	A<=A!(i+d)	5	19	3	X	X	1	P	0	0
ORI	n	56	A<=A!n	2	7	2	X	X	1	P	0	X
XRA	r	57	A<=A^r	1	4	1	X	X	1	P	0	0
XRA	M	58	A<=A^(HL)	2	7	1	X	X	1	P	0	0
XRAi	d	59	A<=A^(i+d)	5	19	3	X	X	1	P	0	0
XRI	n	58	A<=A^n	2	7	2	X	X	1	P	0	0
CMP	r	49	A-r	1	4	1	X	X	X	V	1	X
CMP	M	50	A-(HL)	2	7	1	X	X	X	V	1	X
CMPi	d	51	A-(i+d)	5	19	3	X	X	X	V	1	X
CPI	n	50	A-n	2	7	2	X	X	X	V	1	X
INR	r	51	r<=r+1	1	4	1	X	X	X	V	0	*
INR	M	52	(HL)<=(HL)+1	3	10	1	X	X	X	V	0	*
INRi	d	52	(i+d)<=(i+d)+1	6	23	3	X	X	X	V	0	*
DCR	r	52	r<=r-1	1	4	1	X	X	X	V	1	*
DCR	M	53	(HL)<=(HL)-1	3	11	1	X	X	X	V	1	*
DCRi	d	53	(i+d)<=(i+d)-1	6	23	3	X	X	X	V	1	*

2.5.7 General Purpose Arithmetic And Control Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
DAA	60	Convert A to packed BCD after an add or subtract of packed BCD operands	1	4	1	X	X	X	P	0	X
CMA	60	A<=#A	1	4	1	*	*	1	*	1	*
NEG	61	A<=-A	2	8	2	X	X	X	V	1	X
CMC	61	CY<=#CY	1	4	1	*	*	*	*	0	X
STC	61	CY<=1	1	4	1	*	*	0	*	0	1
NOP	62	No Operation	1	4	1	*	*	*	*	*	*
HLT	62	Halt	1	4	1	*	*	*	*	*	*
DI	62	IFF<=0	1	4	1	*	*	*	*	*	*
EI	62	IFF<=1	1	4	1	*	*	*	*	*	*
IM0	63	Interrupt Mode 0	2	8	2	*	*	*	*	*	*
IM1	63	Interrupt Mode 1	2	8	2	*	*	*	*	*	*
IM2	63	Interrupt Mode 2	2	8	2	*	*	*	*	*	*

2.5.8 16-Bit Arithmetic Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
DAD rr	64	HL<=HL+rr	3	10	1	*	*	X	*	0	X
DADC rr	64	HL<=HL+rr+CY	4	15	2	X	X	X	V	0	X
DSBC rr	65	HL<=HL-rr-CY	4	15	2	X	X	X	V	1	X
DADX tt	65	IX<=IX+tt	4	15	2	*	*	X	*	0	X
DADY uu	65	IY<=IY+uu	4	15	2	*	*	X	*	0	X
INX rr	66	rr<=rr+1	1	6	2	*	*	*	*	*	*
INXi	67	i<=i+1	2	10	2	*	*	*	*	*	*
DCX rr	66	rr<=rr-1	1	6	1	*	*	*	*	*	*
DCXi	67	i<=i-1	2	10	2	*	*	*	*	*	*

2.5.9 Rotate And Shift Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
RLC	68	A[n+1]<=A[n] A[0]<=A[7] CY<=A[7]	1	4	1	*	*	0	*	0	X
RAL	68	A[n+1]<=A[n] A[0]<=CY CY<=A[7]	1	4	1	*	*	0	*	0	X
RRC	69	A[n]<=A[n+1] A[7]<=A[0] CY<=A[0]	1	4	1	*	*	0	*	0	X
RAR	69	A[n]<=A[n+1] A[7]<=CY CY<=A[0]	1	4	1	*	*	0	*	0	X
RLCR r	70	r[n+1]<=r[n] r[0]<=r[7] CY<=r[7]	2	8	2	X	X	0	P	0	X
RLCR M	70	(HL)[n+1]<=(HL)[n] (HL)[0]<=(HL)[7] CY<=(HL)[7]	4	15	2	X	X	0	P	0	X
RLCRi d	71	(i+d)[n+1]<=(i+d)[n] (i+d)[0]<=(i+d)[7] CY<=(i+d)[7]	6	23	2	X	X	0	P	0	X
RALR r	71	r[n+1]<=r[n] r[0]<=CY CY<=r[7]	2	8	2	X	X	0	P	0	X
RALR M	72	(HL)[n+1]<=(HL)[n] (HL)[0]<=CY CY<=(HL)[7]	4	15	2	X	X	0	P	0	X
RALRi d	72	(i+d)[n+1]<=(i+d)[n] (i+d)[0]<=CY CY<=(i+d)[7]	6	23	2	X	X	0	P	0	X
RRCR r	73	r[n]<=r[n+1] r[7]<=r[0] CY<=r[0]	2	8	2	X	X	0	P	0	X
RRCR M	73	(HL)[n]<=(HL)[n+1] (HL)[7]<=(HL)[0] CY<=(HL)[0]	4	15	2	X	X	0	P	0	X
RRCRi d	74	(i+d)[n]<=(i+d)[n+1] (i+d)[7]<=(i+d)[0] CY<=(i+d)[0]	6	23	2	X	X	0	P	0	X

2.5.10 Rotate And Shift Group (cont'd)

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
RARR r	74	r[n]<=r[n+1] r[7]<=CY CY<=r[0]	2	8	2	X	X	0	P	0	X
RARR M	75	(HL)[n]<=(HL)[n+1] (HL)[7]<=CY CY<=(HL)[0]	4	15	2	X	X	0	P	0	X
RARRi d	75	(i+d)[n]<=(i+d)[n+1] (i+d)[7]<=CY CY<=(i+d)[0]	6	23	2	X	X	0	P	0	X
SLAR r	77	r[n+1]<=r[n] r[0]<=0 CY<=r[7]	2	8	2	X	X	0	P	0	X
SLAR M	78	(HL)[n+1]<=(HL)[n] (HL)[0]<=0 CY<=(HL)[7]	4	15	2	X	X	0	P	0	X
SLARi d	78	(i+d)[n+1]<=(i+d)[n] (i+d)[0]<=0 CY<=(i+d)[7]	6	23	2	X	X	0	P	0	X
SRAR r	79	r[n]<=r[n+1] r[7]<=r[7] CY<=r[0]	2	8	2	X	X	0	P	0	X
SRAR M	79	(HL)[n]<=(HL)[n+1] (HL)[7]<=(HL)[7] CY<=(HL)[0]	4	15	2	X	X	0	P	0	X
SRARi d	80	(i+d)[n]<=(i+d)[n+1] (i+d)[7]<=(i+d)[7] CY<=(i+d)[0]	6	23	2	X	X	0	P	0	X
SRLR r	76	r[n]<=r[n+1] r[7]<=0 CY<=r[0]	2	8	2	X	X	0	P	0	X
SRLR M	76	(HL)[n]<=(HL)[n+1] (HL)[7]<=0 CY<=(HL)[0]	4	15	2	X	X	0	P	0	X
SRLRi d	77	(i+d)[n]<=(i+d)[n+1] (i+d)[7]<=0 CY<=(i+d)[0]	6	23	2	X	X	0	P	0	X
RLD	80	A[0-3]<=(HL)[4-7] (HL)[4-7]<=(HL)[0-3] (HL)[0-3]<=A[0-3]	5	18	2	X	X	0	P	0	X
RRD	81	(HL)[0-3]<=(HL)[4-7] (HL)[4-7]<=A[0-3] A[0-3]<=(HL)[0-3]	5	18	2	X	X	0	P	0	X

2.5.10 Bit Set, Reset And Test Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
BIT b,r	82	ZF<=#r[b]	2	8	2	?	X	1	?	0	*
BIT b,M	82	ZF<=#(HL)[b]	3	12	2	?	X	1	?	0	*
BITi b,d	83	ZF<=#(i+d)[b]	5	20	4	?	X	1	?	0	*
SETB b,r	83	r[b]<=1	2	8	2	*	*	*	*	*	*
SETB b,M	84	(HL)[b]<=1	4	15	2	*	*	*	*	*	*
SETi b,d	84	(i+d)[b]<=1	6	23	2	*	*	*	*	*	*
RES b,r	85	r[b]<=0	2	8	2	*	*	*	*	*	*
RES b,M	85	(HL)[b]<=0	4	15	2	*	*	*	*	*	*
RESi b,d	85	(i+d)[b]<=0	6	23	2	*	*	*	*	*	*

2.5.11 Program Transfer Group

INSTRUCTION			NOTATION	CYC	S	BYT	FLAG					
OP CODE		PG					S	Z	H	P/V	N	C
JMP	nn	86	PC<=nn	3	10	3	*	*	*	*	*	*
JZ	nn	86	If0, then JMP else continue	3	10	3	*	*	*	*	*	*
JNZ	nn	87	If not 0	3	10	3	*	*	*	*	*	*
JC	nn	88	If carry	3	10	3	*	*	*	*	*	*
JNC	nn	87	If not carry	3	10	3	*	*	*	*	*	*
JPO	nn	88	If parity odd	3	10	3	*	*	*	*	*	*
JPE	nn	89	If parity even	3	10	3	*	*	*	*	*	*
JP	nn	89	If sign positive	3	10	3	*	*	*	*	*	*
JM	nn	90	If sign negative	3	10	3	*	*	*	*	*	*
JO	nn	90	If overflow	3	10	3	*	*	*	*	*	*
JNO	nn	91	If not overflow	3	10	3	*	*	*	*	*	*
JMPR	e	92	PC<=e where -126<e-PC<+129	3	12	2	*	*	*	*	*	*
JRZ	e	92	If 0, then JMPR else continue	3	12	2	*	*	*	*	*	*
JRNZ	e	92	If not 0	3	12	2	*	*	*	*	*	*
JRC	e	93	If carry	3	12	2	*	*	*	*	*	*
JRNC	e	93	If not carry	3	12	2	*	*	*	*	*	*
DJNZ	e	94	B<=B-1 If B=0 then continue else JMPR	3	13	2	*	*	*	*	*	*
PCHL		91	PC<=HL	1	4	1	*	*	*	*	*	*
PCIi		91	PC<=Ii	2	8	2	*	*	*	*	*	*

2.5.12 Call And Return Group

INSTRUCTION			NOTATION	CYC	S	BYT	FLAG					
OP CODE		PG					S	Z	H	P/V	N	C
CALL	nn	94	(SP-1)<=PC\H (SP-5)<=PC\L SP<=SP-2 PC<=nn	5	17	3	*	*	*	*	*	*
CZ	nn	95	If 0, then CALL else continue	3/5	10/17	3	*	*	*	*	*	*
CNZ	nn	95	If not 0	3/5	10/17	3	*	*	*	*	*	*
CC	nn	96	If carry	3/5	10/17	3	*	*	*	*	*	*
CNC	nn	96	If not carry	3/5	10/17	3	*	*	*	*	*	*
CPO	nn	97	If parity odd	3/5	10/17	3	*	*	*	*	*	*
CPE	nn	97	If parity even	3/5	10/17	3	*	*	*	*	*	*
CP	nn	98	If sign positive	3/5	10/17	3	*	*	*	*	*	*
CM	nn	98	If sign negative	3/5	10/17	3	*	*	*	*	*	*
CO	nn	99	If overflow	3/5	10/17	3	*	*	*	*	*	*
CNO	nn	99	If not overflow	3/5	10/17	3	*	*	*	*	*	*
RET		100	PC\H<=(SP+1) PC\L<=(SP) SP<=SP+2	3	10	1	*	*	*	*	*	*
RZ		102	If 0, then RET else continue	1/3	5/11	1	*	*	*	*	*	*
RNZ		102	If not 0	1/3	5/11	1	*	*	*	*	*	*
RC		102	If carry	1/3	5/11	1	*	*	*	*	*	*
RNC		103	If not carry	1/3	5/11	1	*	*	*	*	*	*
RPO		103	If parity odd	1/3	5/11	1	*	*	*	*	*	*
RPE		103	If parity even	1/3	5/11	1	*	*	*	*	*	*
RP		104	If sign positive	1/3	5/11	1	*	*	*	*	*	*
RM		104	If sign negative	1/3	5/11	1	*	*	*	*	*	*
RO		104	If overflow	1/3	5/11	1	*	*	*	*	*	*
RNO		105	If no overflow	1/3	5/11	1	*	*	*	*	*	*
RETI		101	Return from interrupt	4	14	2	*	*	*	*	*	*
RETN		101	Return from Non Maskable Interrupt	4	14	2	*	*	*	*	*	*
RST	n	100	(SP-1)<=PC\H (SP-2)<=PC\L PC<=8*n where 0<=n<8	3	11	1	*	*	*	*	*	*

2.5.13 Input/Output Group

INSTRUCTION		NOTATION	CYC	S	BYT	FLAG					
OP CODE	PG					S	Z	H	P/V	N	C
IN n	109	A<=In	3	11	2	*	*	*	*	*	*
INP r	109	r<=I(C)	3	12	2	X	X	0	P	0	*
INI	111	(HL)<=I(C) B<=B-1 HL<=HL+1	4	16	2	?	X	?	?	1	*
INIR	111	Repeat INI until B=0	5/4	21/16	2	?	1	?	?	1	*
IND	110	(HL)<=I(C) B<=B-1 HL<=HL-1	4	16	2	?	X	?	?	1	*
INDR	110	Repeat IND until B=0	5/4	21/16	2	?	1	?	?	1	*
OUT n	106	On<=A	3	11	2	*	*	*	*	*	*
OUTP r	106	O(C)<=r	3	12	2	*	*	*	*	*	*
OUTI	107	O(C)<=(HL) B<=B-1 HL<=HL+1	4	16	2	?	X	?	?	1	*
OUTIR	107	Repeat OUTI until B=0	5/4	21/16	2	?	1	?	?	1	*
OUTD	108	O(C)<=(HL) B<=B-1 HL<=HL-1	4	16	2	?	X	?	?	1	*
OUTDR	108	Repeat OUTD until B=0	5/4	21/16	2	?	1	?	?	1	*

2.6 OPCODE FORMAT AND DESCRIPTION

2.6.1 8-Bit Data Transfer Group

This group of instructions transfers data to and from registers and memory. Condition flags are not affected by any instruction in this group.

LD rd,rs

MOV r,r' Move register to register

r<=r'

Move contents of source register rs to destination register rd.

0	1	rd	rd	rd	rs	rs	rs

cycles: 1 states: 4
flags: none

LD (HL),rs

MOV M,r Move register to memory

r<=(HL)

Move contents of register rs to the memory location addressed by the HL register pair.

0	1	1	1	0	rs	rs	rs

cycles: 2 states: 7
flags: none

LD rd,(HL)

MOV r,M Move memory to register

r<=(HL)

Move contents of the memory location, whose address is in registers H and L, to register rd.

0	1	rd	rd	rd	1	1	0

cycles: 2 states: 7
flags: none

LD (IX+i),rs
LD (IY+i),rs

MOVXR i,r
MOVYR i,r Move register to indexed memory

(IX + i)<=r
(IY + i)<=r

Move contents of source register to the memory location addressed by the sum of the index register and the two's complement displacement i.

1	1	*	1	1	1	0	1	DD/FD
0	1	1	1	0	rs	rs	rs	
i								

cycles: 5 states: 19
flags: none

*0=IX 1=IY

LD rd,(IX+i)
LD rd,(IY+i)

MOVRX rd,i
MOVRY rd,i Move indexed memory to register

r<=(IX + i)
r<=(IY + i)

Move contents of memory location addressed by the sum of the index register and the two's complement displacement i to the destination register.

1	1	*	1	1	1	0	1	DD/FD
0	1	rd	rd	rd	1	1	0	
i								

cycles: 5 states: 19
flags: none

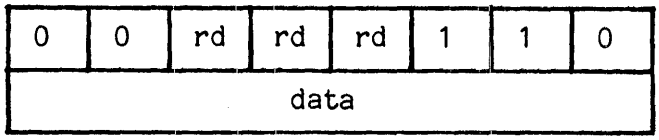
*0=IX 1=IY

LD rd,d8

MVI r,n Move immediate to register

r<=n

Move contents of byte 2 of the instruction to register rd.



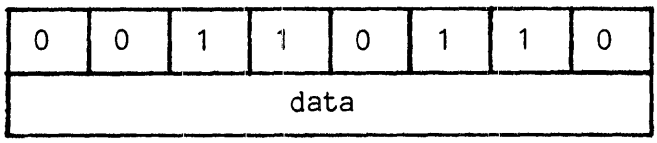
cycles: 2 states: 7
flags: none

LD (HL),d8

MVI M,n Move to memory immediate

(HL)<=n

Move contents of byte 2 of the instruction to the memory location addressed by the HL register pair.



36

cycles: 3 states: 10
flags: none

LD (IX+i),d8
LD (IY+i),d8

MVIX i,n
MVIY i,n Move immediate to indexed memory

(IX + i)<=n
(IY + i)<=n

Move byte 4 of the instruction to the memory location addressed by the sum of the index register and the two's complement displacement.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	1	0	1	1	0	36
i								
data								

cycles: 5 states: 19
flags: none

*0=IX 1=IY

LD A,(a16)

LDA nn Load accumulator direct

A<=(nn)

Move contents of the memory location, addressed by byte 2 and byte 3 of the instruction, to the A register.

0	0	1	1	1	0	1	0	3A
low-order addr								
high-order addr								

cycles: 4 states: 13
flags: none

LD (a16),A

STA nn Store accumulator direct

(nn)<=A

Move contents of the accumulator to the memory location addressed by byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0	32
low-order addr								
high-order addr								

cycles: 4 states: 13
flags: none

LD A,(rp)

LDAX zz Load accumulator indirect

A<=(zz)

Move contents of the memory location, addressed by the register pair rp, to register A. Note: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	rp	rp	1	0	1	0

cycles: 2 states: 7
flags: none

LD (rp),A

STAX zz Store accumulator indirect

(zz)<=A

Move contents of register A to the memory location addressed by the register pair rp. Note: Only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	rp	rp	0	0	1	0

cycles: 2 states: 7
flags: none

LDAI Load accumulator interrupt vector

A<=I

Load the interrupt vector register into the accumulator.

1	1	1	0	1	1	0	1	ED
0	1	0	1	0	1	1	1	57

cycles: 2 states: 9
flags: S Z H P/V N C
        X X 0 IFF 0 *

*The P/V flag is set to the value of the interrupt enable flip-flop.

LDAR Load accumulator refresh register

A<=R

Load the refresh register into the accumulator.

1	1	1	0	1	1	0	1	ED
0	1	0	1	1	1	1	1	5F

cycles: 2 states: 9
flags: S Z H P/V N C
        X X 0 IFF 0 *

*The P/V flag is set to the value of the interrupt enable flip-flop.

LD I,A

STAI Store accumulator interrupt vector

I<=A

Store the contents of the accumulator in the interrupt vector register.

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	1	1	47

cycles: 2 states: 9
flags: none

LD R,A

STAR Store accumulator refresh register

R<=A

Store the contents of the accumulator in the refresh register.

1	1	1	0	1	1	0	1	ED
0	1	0	0	1	1	1	1	4F

cycles: 2 states: 9
flags: none

2.6.2 16-Bit Data Transfer Group

LD BC,d16
LD DE,d16
LD HL,d16
LD SP,d16

LXI r,nn Load register pair immediate

rr<=nn
(r)<=n

Move byte 3 of the instruction into the high-order register (rh) of the register pair rp. Move byte 2 of the instruction into the low-order register (rl) of the register pair rp.

0	0	rp	rp	0	0	0	1
low-order data							
high-order data							

cycles: 3 states: 10
flags: none

LD IX,d16
LD IY,d16

LXIX
LXIY Load index register immediate

IX<=nn
IY<=nn

Load the immediate data in byte 4 to the high-order half of the index register and the immediate data in byte 3 to the low-order half.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	0	0	0	0	1	21
low-order data								
high-order data								

cycles: 4 states: 14
flags: none

*0=IX 1=IY

LD BC,(a16)
LD DE,(a16)
LD HL,(a16)
LD SP,(a16)

LBCD nn
LDED nn
LSPD nn Load register pair direct

rp<=(a16)

Move contents of memory location nn into the low-order byte of the register pair and the contents of the next higher memory location into the high-order byte of the register pair.

1	1	1	0	1	1	0	1	ED
0	1	rp	rp	1	0	1	1	
low-order addr								
high-order addr								

cycles: 6 states: 20
flags: none

LD HL,(a16)

LHLD nn Load H and L direct

L<=(nn)
H<=(nn + 1)

Move contents of the memory location, addressed by byte 2 and byte 3 of the instruction, to register L. Move contents of the memory location at the next higher address to register H.

0	0	1	0	1	0	1	0	2A
low-order addr								
high-order addr								

cycles: 5 states: 16
flags: none

LD (a16),BC
LD (a16),DE
LD (a16),SP

SBCD nn
SDED nn
SSPD nn Store register direct

(nn)<=rp

Move contents of register pair low-order byte to memory location nn and high-order byte in the next higher memory location.

1	1	1	0	1	1	0	1	ED
0	1	rp	rp	0	0	1	1	
low-order addr								
high-order addr								

cycles: 6 states: 20
flags: none

LD (a16),HL

SHLD nn Store HL direct

(nn)<=L
(nn + 1)<=H

Move contents of register L to the memory location addressed by byte 2 and byte 3. Move contents of register H to the next higher memory location.

0	0	1	0	0	0	1	0	22
low-order addr								
high-order addr								

cycles: 5 states: 16
flags: none

LD IX,(a16)
LD IY,(a16)

LIXD nn
LIYD nn Load index register direct

IXl<=(nn)
IXh<=(nn + 1)

Move contents of memory location nn into the index register low-order byte and the contents of memory location nn + 1 into the index register high-order byte.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	0	1	0	1	0	2A
low-order addr								
high-order addr								

cycles: 6 states: 20
flags: none

*0=IX 1=IY

LD (a16),IX
LD (a16),IY

SIXD nn
SIYD nn Store index register direct

(nn)<=IXl
(nn + 1)<=IXh

Move contents of index register low-order byte to memory location nn and index register high-order byte to memory location nn + 1.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	0	0	0	1	0	22
low-order addr								
high-order addr								

cycles: 6 states: 20
flags: none

*0=IX 1=IY

LD SP,HL

SPHL Move HL to SP

SP<=HL

Move contents of registers H and L (16 bits) to register SP.

1	1	1	1	1	0	0	1	F9

cycles: 1 states: 6
flags: none

LD SP,IX
LD SP,IY

SPIX
SPIY Move index register to stack pointer

SP<=IX
SP<=IY

Move index register contents to stack pointer.

1	1	*	1	1	1	0	1	DD/FD
1	1	1	1	1	0	0	1	F9

cycles: 2 states: 10
flags: none

*0=IX 1=IY

PUSH BC
PUSH DE
PUSH HL
PUSH AF

PUSH qq Push

(SP - 1)<=qqH
(SP - 2)<=qqL
SP<=SP - 2

Move contents of the high-order register of register pair rp to the memory location whose address is one less than the content of register SP. Move contents of the low-order register of register pair rp to the memory location whose address is two less than the content of register SP. Decrement the contents of register SP by 2. Note: Register pair rp = SP may not be specified.

1	1	rp	rp	0	1	0	1

cycles: 3 states: 11
flags: none

rp: B=00 D=01 H=10 PSW=11

PUSH IX
PUSH IY

PUSHX
PUSHY Push index register

SP - 1<=IXh
SP - 2<=IXl
SP<=SP - 2

Push the contents of the index register high-order byte into the memory location SP - 1 and push the contents of index register low-order byte into memory location SP - 2 and decrement the stack pointer by 2.

1	1	*	1	1	1	0	1	DD/FD
1	1	1	1	0	1	0	1	E5

cycles: 3 states: 15
flags: none

*0=IX 1=IY

POP BC
POP DE
POP HL
POP AF

POP qq Pop

qqL<=SP
qqH<=SP + 1
SP<=SP + 2

Move contents of the memory location, whose address is specified by the content of register SP, to the low-order register of register pair rp. Move contents of the memory location, whose address is one more than the contents of register SP, to the high-order register of register pair rp. Increment the contents of register SP by 2.

Note: Register pair rp = SP is not valid.

1	1	rp	rp	0	0	0	1

cycles: 3 states: 10
flags: none except POP PSW affects all flags

rp: B=00 D=01 H=10 PSW=11

POP IX
POP IY

POPX
POPY Pop index register from the stack

IXl=SP
IXh=SP + 1
SP<=SP + 2

Pop the contents of the memory location addressed by the stack pointer into the contents of the next higher memory location into the high-order byte of the index register and increment the stack pointer by 2.

1	1	*	1	1	1	0	1	DD/FD
1	1	1	1	0	0	0	1	E1

cycles: 4 states: 14
flags: none

*=IX 1=IY

2.6.3 Exchange, Block Transfer And Search Group

EX DE,HL

XCHG Exchange HL with DE

HL<=>DE

Exchange the contents of the HL register pair with the contents of register pair DE.

1	1	1	0	1	0	1	1	EB

cycles: 1 states: 4
flags: none

EX AF,AF'

EXAF Exchange accumulator and flags

PSW<=>PSW'

Exchange the processor status word (consisting of the accumulator and flags) with the alternate processor status word.

0	0	0	0	1	0	0	0	08

cycles: 1 states: 4
flags: none

EXX

EXX Exchange registers

BC<=>BC'
DE<=>DE'
HL<=>HL'

Exchange the B, C, D, E, H and L registers with the alternate set.

1	1	0	1	1	0	0	1	D9

cycles: 1 states: 4
flags: none

EX (SP),HL

XTHL Exchange stack top with H and L

L<=>SP
H<=>SP + 1

Exchange the contents of the L register with the contents of the memory location addressed by the contents of register SP. Exchange the contents of the H register with the contents of the memory location addressed by one greater than the contents of register SP.

1	1	1	0	0	0	1	1	E3

cycles: 5 states: 19
flags: none

EX (SP),IX
EX (SP),IY

XTIX
XTIY Exchange stack top with index register

IXL<=SP
IXH<=SP + 1

Exchange the contents of the memory location addressed by the SP with the index register low-order byte and exchange the memory location one greater than the SP with the index register high-order byte.

1	1	*	1	1	1	0	1	DD/FD
1	1	1	1	0	0	1	1	E3

cycles: 6 states: 23
flags: none

*0=IX 1=IY

LDI Load and double increment

```
DE<=HL
DE<=(DE + 1)
HL<=(HL + 1)
BC<=(BC - 1)
```

Move the memory location addressed by HL to the memory location addressed by DE; then increment pointers HL and DE and decrement byte counter BC.

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	0	0	A0

cycles: 4 states: 16

flags:	S	Z	H	P/V	N	C
	*	*	0	X	0	*

P/V flag = 0 when BC - 1 = 0
= 1 when BC - 1 =/ 0

LDIR Load and double increment and repeat

```
DE<=HL
DE<=(DE + 1)
HL<=(HL + 1)
BC<=(BC - 1)
PC<=(PC - 2 until BC = 0)
```

Move the memory location addressed by HL to the memory location addressed by DE; then increment pointers HL and DE and decrement byte counter BC and repeat the instruction until BC = 0.

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	0	0	B0

cycles: 5/4 states: 21/16

flags:	S	Z	H	P/V	N	C
	*	*	0	0	0	*

LDD Load and double decrement

DE<=HL
DE<=(DE - 1)
HL<=(HL - 1)
BC<=(BC - 1)

Move the memory location addressed by HL to the memory location addressed by DE; then decrement pointers DE and HL and decrement byte counter BC.

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	0	0	A8

cycles: 4 states: 16
flags: S Z H P/V N C
        * * 0   X 0 *

P/V flag = 0 when BC - 1 = 0
         = 1 when BC - 1 =/ 0

LDDR Load, double decrement and repeat

DE<=HL
DE<=(DE - 1)
HL<=(HL - 1)
BC<=(BC - 1)
PC<=(PC - 2 until BC = 0)

Move the memory location addressed by HL to the memory location addressed by DE; then decrement pointers DE and HL and decrement byte counter BC and repeat the instruction until BC = 0.

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	0	0	B8

cycles: 5/4 states: 21/16
flags: S Z H P/V N C
        * * 0   0 0 *

CCD Compare memory contents and decrement

A - (HL)
HL<=(HL - 1)
BC<=(BC - 1)

Subtract the contents of the memory location addressed by HL from the accumulator; decrement the memory pointer HL and the byte counter BC.

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	0	1	A9

cycles: 4 states: 16
flags: S Z H P/V N C
X X X X 1 *

P/V flag = 0 when BC - 1 = 0
= 1 when BC - 1 =/ 0

CCDR Compare memory contents, decrement and repeat

A = (HL)
HL<=(HL - 1)
BC<=(BC - 1)
PC<=(PC - 2 until BC = 0)

Subtract the contents of the memory location addressed by HL from the accumulator; decrement the memory pointer HL and the byte counter BC and repeat the instruction until BC = 0 or until A = (HL).

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	0	1	B9

cycles: 5/4 states: 21/16
flags: S Z H P/V N C
X X X X 1 *

CCI Compare memory contents and increment

A - (HL)
HL<=HL + 1
BC<=BC - 1

Subtract the contents of the memory location addressed by HL from the accumulator; increment the memory pointer HL and decrement the byte counter BC.

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	0	1	A1

cycles: 4 states: 16
flags: S Z H P/V N C
X X X X * *

P/V flag = 0 when BC - 1 = 0
= 1 when BC - 1 =/ 0.

CCIR Compare memory contents, increment and repeat

A = (HL)
HL<=HL - 1
BC<=BC - 1
PC<=PC - 2 until BC = 0

Subtract the contents of the memory location addressed by HL from the accumulator; increment the memory pointer HL and decrement the byte counter BC and repeat the instruction until BC = 0 or until A = (HL).

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	0	1	B1

cycles: 5/4 states: 21/16
flags: S Z H P/V N C
X X X X 1 *

P/V flag = 0 when BC - 1 = 0
= 1 when BC - 1 =/ 0

2.6.4 8-Bit Arithmetic And Logical Group

2.6.4.1 Arithmetic Instructions

This group of instructions performs arithmetic operations between data in accumulator and data registers or memory.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Half-Carry flags according to the standard rules.

All subtraction operations are performed via two's complement arithmetic and set the Carry flag to one to indicate a borrow and clear it to indicate no borrow.

ADD A,rs

ADD r Add register to accumulator

A<=A + r

Add register rs to the accumulator. Place the result in the accumulator.

1	0	0	0	0	rs	rs	rs

cycles: 1 states: 4
flags:

S	Z	H	P/V	N	C
X	X	X	V	0	X

ADD A,(HL)

ADD M Add memory to accumulator

A<=A + (HL)

Add the contents of the memory location addressed by the HL register pair to the contents of the accumulator. Place the result in the accumulator.

1	0	0	0	0	1	1	0

86

cycles: 2 states: 7
flags:

S	Z	H	P/V	N	C
X	X	X	V	0	X

ADD A,(IX+i)
ADD A,(IY+i)

ADDX i
ADDY i Add indexed memory to accumulator

A<=A + (IX + i)
A<=A + (IY + i)

Add the contents of the memory location addressed by the sum of index register and two's complement offset i, to the accumulator.

1	1	*	1	1	1	0	1	DD/FD
1	0	0	0	0	1	1	0	86
i								

cycles: 5 states: 19
flags: S Z H P/V N C
        X X X   V 0 X

*0=IX 1=IY

ADC A,rs

ADC r Add register with carry to accumulator

A<=A+r + CY

Add the contents of register rs and the contents of the Carry bit to the contents of the accumulator. Place the result in the accumulator.

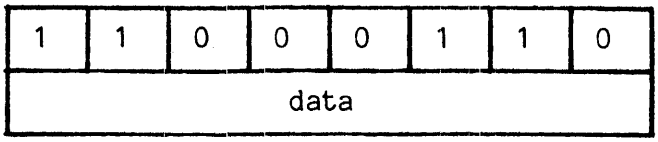
1	0	0	0	1	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
        X X X   V 0 X

ADI n Add immediate to accumulator

A<=A + n

Add the contents of the second byte of the instruction to the contents of the accumulator. Place the result in the accumulator.



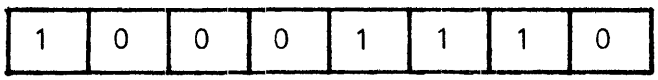
cycles: 2 states: 7
flags: S Z H P/V N C
X X X V 0 X

ADC A,(HL)

ADC M Add memory with carry to accumulator

A<=A + (HL) + CY

Add the contents of the memory location addressed by the HL register pair and the contents of the CY flag to the accumulator. Place the result in the accumulator.



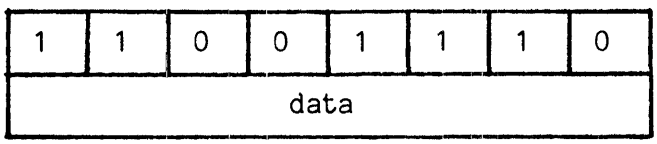
cycles: 2 states: 7
flags: S Z H P/V N C
X X X V 0 X

ADC A,n

ACI n Add immediate with carry to accumulator

A<=A + n + CY

Add the contents of the second byte of the instruction and the contents of the CY flag to the contents of the accumulator. Place the result in the accumulator.



cycles: 2 states: 7
flags: S Z H P/V N C
X X X V 0 X

ADCX i
ADCY i Add indexed memory with carry to accumulator.

A<=A +(IX + i) + CY
A<=A +(IY + i) + CY

Add the contents of the Carry bit and the memory location addressed by the sum of the index register and two's complement displacement i to the accumulator.

1	1	*	1	1	1	0	1	DD/FD
1	0	0	0	1	1	1	0	8E
i								

cycles: 5 states: 19
flags: S Z H P/V N C
        X X X   V 0 X

*0=IX 1=IY

SUB r Subtract register from accumulator

A<=A - r

Subtract the contents of register rs from the contents of the accumulator. Place the result in the accumulator.

1	0	0	1	0	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
        X X X   V 1 X

SUB M Subtract from accumulator

A<=A - (HL)

Subtract the contents of the memory location addressed by the HL register pair from the contents of the accumulator. Place the result in the accumulator.

1	0	0	1	0	1	1	0	96

cycles: 2 states: 7
flags:

S	Z	H	P/V	N	C
X	X	X	V	1	X

SUI n Subtract immediate from accumulator

A<=A - n

Subtract the contents of the second byte of the instruction from the contents of the accumulator. Place the result in the accumulator.

1	1	0	1	0	1	1	0	D6
data								

cycles: 2 states: 7
flags:

S	Z	H	P/V	N	C
X	X	X	V	1	X

SUBX i
SUBY i Subtract indexed memory from accumulator

A<=A - (IX + i)
A<=A - (IY + i)

Subtract the contents of the memory location addressed by the sum of the index register and the two's complement displacement i from the accumulator.

1	1	*	1	1	1	0	1	DD/FD
1	0	0	1	0	1	1	0	96
i								

cycles: 5 states: 19
flags: S Z H P/V N C
        X X X   V 1 X

*0=IX 1=IY

SBB r Subtract register with borrow from accumulator

A<=A - r - CY

Subtract the contents of register rs and the contents of the CY flag from the accumulator. Place the result in the accumulator.

1	0	0	1	1	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
        X X X   V 1 X

SBB M Subtract memory with borrow from accumulator

A<=A - (HL) - CY

Subtract the contents of the memory location addressed by the HL register pair and the contents of the CY flag from the accumulator. Place the result in the accumulator.

1	0	0	1	1	1	1	0	9E

cycles: 2 states: 7
flags:

S	Z	H	P/V	N	C
X	X	X	V	1	X

SBI n Subtract immediate with borrow

A<=A - n - CY

Subtract the contents of the second byte of the instruction and the contents of the CY flag from the accumulator. Place the result in the accumulator.

1	1	0	1	1	1	1	0	DE
data								

cycles: 2 states: 7
flags:

S	Z	H	P/V	N	C
X	X	X	V	1	X

SBC A,(IX+i)
SBC A,(IY+i)

SBBX i
SBBY i Subtract indexed memory with borrow from accumulator

A<=A - (IX + i) - CY
A<=A - (IY + i) - CY

Subtract the Carry bit and the contents of the memory location addressed by the sum of the index register and the two's complement displacement i from the accumulator.

1	1	*	1	1	1	0	1	DD/FD
1	0	0	1	1	1	1	0	9E
i								

cycles: 5 states: 19
flags: S Z H P/V N C
X X X V 1 X

CP rs

CMP r Compare register with accumulator

A - r

Subtract the contents of register rs from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = (rs). The CY flag is set to 1 if (A) < (rs).

1	0	1	1	1	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
X X X V 1 X

CMP M Compare memory with accumulator

A - (HL)

Subtract the contents of the memory location addressed by the HL register pair from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. The Z flag is set to 1 if (A) = (HL). The CY flag is set to 1 if (A) < (HL).

1	0	1	1	1	1	1	0	BE

cycles: 2 states: 7

flags:	S	Z		H		P/V	N	C
	X	X		X		V	1	X

CPI n Compare immediate with accumulator

A - n

Subtract the contents of the second byte of the instruction from the accumulator. The accumulator remains unchanged. The condition flags are set by the result of the subtraction. The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).

1	1	1	1	1	1	1	0	FE
data								

cycles: 2 states: 7

flags:	S	Z		H		P/V	N	C
	X	X		X		V	1	X

CMPX i
CMPY i Compare accumulator with indexed memory

A - (IX + i)
A - (IY + i)

Subtract the contents of the memory location addressed by the sum of the index register and the two's complement displacement i, from the accumulator. The accumulator remains unchanged.

1	1	*	1	1	1	0	1	DD/FD
1	0	1	1	1	1	1	0	BE
i								

cycles: 5 states: 19
flags: S Z H P/V N C
        X X X   V 1 X

*0=IX 1=IY

INR r Increment register

r<=r + 1

Increment the contents of register rd by one. Note: All condition flags except CY are affected.

0	0	rd	rd	rd	1	0	0

cycles: 1 states: 4
flags: S Z H P/V N C
        X X X   V 0 *

INC (HL)

INR M Increment memory

(HL)<=(HL) + 1

Increment the contents of the memory location addressed by the HL register pair by one. Note: All condition flags except CY are affected.

0	0	1	1	0	1	0	0	34

cycles: 3 states: 10
flags: S Z H P/V N C
X X X V 0 *

INC (IX+i)
INC (IY+i)

INRX i
INRY i Increment indexed memory

(IX + i)<=(IX + i) + 1
(IY + i)<=(IY + i) + 1

Increment the contents of the memory location addressed by the sum of the index register and the two's complement displacement i.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	1	0	1	0	0	34
i								

cycles: 6 states: 23
flags: S Z H P/V N C
X X X V 0 *

DEC rd

DCR r Decrement register

r<=r-1

Decrement the contents of register rd by one. Note: All condition flags except CY are affected.

0	0	rd	rd	rd	1	0	1

cycles: 1 states: 4
flags: S Z H P/V N C
X X X V 1 *

DCR M Decrement memory

(HL)<=(HL) - 1

Decrement the contents of the memory location addressed by the HL register pair by one. Note: All condition flags except CY are affected.

0	0	1	1	0	1	0	1	35

cycles: 3 states: 11
flags: S Z H P/V N C
X X X V 1 *

DCRX i
DCRY i Decrement indexed memory

(IX + i)<=(IX + i) - 1
(IY + i)<=(IY + i) - 1

Decrement the contents of the memory location addressed by the sum of the index register and the two's complement displacement i.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	1	0	1	0	1	35
i								

cycles: 6 states: 23
flags: S Z H P/V N C
X X X V 1 *

*0=IX 1=IY

2.6.4.2 Logical Group

This group of instructions performs logical (Boolean) operations between data in accumulator and data in registers or memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Half-Carry, and Carry flags according to the standard rules.

AND rs

ANA r AND register with accumulator

A<=A v r

Logical AND the contents of register rs with the contents of the accumulator. Place the result in the accumulator. The CY flag is cleared and AC is set.

1	0	1	0	0	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
        X X 1 P 0 0

AND (HL)

ANA M AND memory with accumulator

A<=A v (HL)

Logical AND the contents of the memory location addressed by the HL register pair with the contents of the accumulator. Place the result in the accumulator. The CY flag is cleared and H is set.

1	0	1	0	0	1	1	0

A6

cycles: 2 states: 7
flags: S Z H P/V N C
        X X 1 P 0 0

ANI n AND immediate with accumulator

A<=A v n

Logically AND the contents of the second byte of the instruction with the contents of the accumulator. Place the result in the accumulator. The CY flag is cleared and H is set.

1	1	1	0	0	1	1	0	E6
data								

cycles: 2 states: 7
flags:

S	Z		H		P/V	N	C
X	X		1		P	0	0

AND (IX+i)
AND (IY+i)

ANAX i
ANAY i AND indexed memory with accumulator

A<=A v (IX + i)
A<=A v (IY + i)

Logical AND the contents of the memory location addressed by the sum of the index register and the two's complement displacement i with the accumulator.

1	1	*	1	1	1	0	1	DD/FD
1	0	1	0	0	1	1	0	A6
i								

cycles: 5 states: 19
flags:

S	Z		H		P/V	N	C
X	X		1		P	0	0

*0=IX 1=IY

OR rs

ORA r OR accumulator with register

A<=A ^ r

Logical OR the contents of register rs with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.

1	0	1	1	0	rs	rs	rs

cycles: 1 states: 4
flags: S Z H P/V N C
        X X 1 P 0 0

OR (HL)

ORA M OR accumulator with memory

A<=A ^ (HL)

Logical OR the contents of the memory location addressed by the HL register pair with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.

1	0	1	1	0	1	1	0

B6

cycles: 2 states: 7
flags: S Z H P/V N C
        X X 1 P 0 0

OR d8

ORI data OR accumulator with immediate data

A<=A ^ n

Logical OR the contents of the second byte of the instruction with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.

1	1	1	1	0	1	1	0
data							

F6

cycles: 2 states: 7
flags: S Z H P/V N C
        X X 1 P 0 X

ORAX i
ORAY i OR accumulator with indexed memory

A<=A ^ (IX + i)
A<=A ^ (IY + i)

Logical OR the accumulator with the contents of the memory location addressed by the sum of the index register and the two's complement displacement i.

1	1	*	1	1	1	0	1	DD/FD
1	0	1	1	0	1	1	0	B6
i								

cycles: 5 states: 19
flags: S Z H P/V N C
        X X 1 P 0 0

*0=IX 1=IY

XRA r Exclusive OR accumulator with register

A<=A ⊕ r

Logical Exclusive OR the contents of register rs with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.

1	0	1	0	1	rs	rs	rs

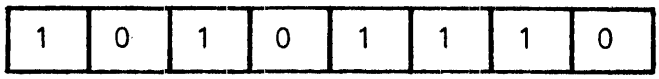
cycles: 1 states: 4
flags: S Z H P/V N C
        X X 1 P 0 0

XOR (HL)

XRA M Exclusive OR accumulator with memory

A<=A ⊕ (HL)

Logical Exclusive OR the contents of the memory location addressed by the HL register pair with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.



cycles: 2 states: 7
flags:

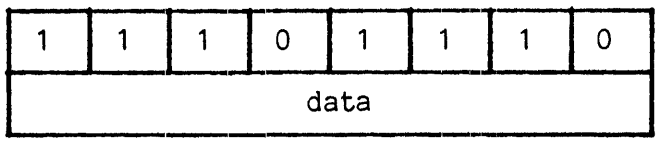
S	Z		H		P/V	N	C
X	X		1		P	0	0

XOR d8

XRI n Exclusive OR accumulator with immediate data

A<=A ⊕ n

Logical Exclusive OR the contents of the second byte of the instruction with the contents of the accumulator. Place the result in the accumulator. The CY and H flags are cleared.



cycles: 2 states: 7
flags:

S	Z		H		P/V	N	C
X	X		1		P	0	0

XRAX i
XRAY i Exclusive OR accumulator with indexed memory

A<=A ⊕ (IX + i)
A<=A ⊕ (IY + i)

Logical Exclusive OR the accumulator with the contents of the memory location addressed by the sum of the index register and the two's complement displacement i.

1	1	*	1	1	1	0	1	DD/FD
1	0	1	0	1	1	1	1	AE
i								

cycles: 5 states: 19
flags:

S	Z		H		P/V	N	C
X	X		1		P	0	0

2.6.5 General Purpose Arithmetic And Control Group

DAA

DAA Decimal adjust accumulator

Adjust the 8-bit binary number in the accumulator to form two 4-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least-significant 4 bits of the accumulator is greater than 9 or if the H flag is set, 6 is added to the accumulator.

2. If the value of the most-significant 4 bits of the accumulator is now greater than 9, or if the C flag is set, 6 is added to the most-significant 4 bits of the accumulator.

NOTE: All flags are affected except Subtract flag.

0	0	1	0	0	1	1	1	27

cycles: 1 states: 4
flags: S Z H P/V N C
X X X P 0 X

CPL

CMA Complement accumulator

A<=#A

Complement the contents of the accumulator (zero bits become one, one bits become zero).

0	0	1	0	1	1	1	1	2F

cycles: 1 states: 4
flags: S Z H P/V N C
* * 1 * 1 *

NEG Negate accumulator

A<= - A

Subtract the contents of the accumulator from zero (one's complement).

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	0	0	44

cycles: 2 states: 8
flags:

S	Z	H	P/V	N	C
X	X	X	V	1	X

CMC Complement carry

CY<=#CY

Complement the CY flag. No other flags are affected.

0	0	1	1	1	1	1	1	3F

cycles: 1 states: 4
flags:

S	Z	H	P/V	N	C
*	*	*	*	0	X

STC Set carry

CY<=1

Set the CY flag to one. No other flags are affected.

0	0	1	1	0	1	1	1	37

cycles: 1 states: 4
flags:

S	Z	H	P/V	N	C
*	*	0	*	0	1

NOP No operation

No operation is performed. The registers and flags are unaffected.

0	0	0	0	0	0	0	0	00

cycles: 1 states: 4
flags: none

HLT Halt

Stop the program counter upon completion of the instruction, and enter the 'HALT' state. The registers and flags are unaffected.

0	1	1	1	0	1	1	0	76

cycles: 1 states: 4
flags: none

EI Enable interrupts

IFF<=1

Enable the interrupt system following the execution of the next instruction.

1	1	1	1	1	0	1	1	FB

cycles: 1 states: 4
flags: none

DI Disable interrupts

IFF<=0

Disable the interrupt system immediately following the execution of the DI instruction.

1	1	1	1	0	0	1	1	F3

cycles: 1 states: 4
flags: none

IM0 Interrupt Mode 0

Place the processor interrupt system in interrupt Mode 0. (Allow interrupt device to force any instruction onto data bus.)

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	1	0	46

cycles: 2 states: 8
flags: none

IM1 Interrupt Mode 1

Place the processor interrupt system in interrupt Mode 1. (Allow interrupt device to force restart to location 38H.)

1	1	1	0	1	1	0	1	ED
0	1	0	1	0	1	1	0	56

cycles: 2 states: 8
flags: none

IM2 Interrupt Mode 2

Place the processor interrupt system in interrupt Mode 2. (Allow interrupt device to use interrupt vector registers and data bus to form call address.)

1	1	1	0	1	1	0	1	ED
0	1	0	1	1	1	1	0	5E

cycles: 2 states: 8
flags: none

2.6.6 16-Bit Arithmetic Group

ADD HL,BC
ADD HL,DE
ADD HL,HL
ADD HL,SP

DAD rr Double add

(HL)<=(HL) + rr

Add the contents of the register pair rp to the contents of the HL register pair. Place the result in the HL register pair. Note: The Carry flags are affected. C is set if there is a carry from bit 15; otherwise it is reset.

0	0	rp	rp	1	0	0	1

cycles: 3 states: 10

flags:	S	Z	H	P/V	N	C
	*	*	X	*	0	X

ADC HL,BC
ADC HL,DE
ADC HL,HL
ADC HL,SP

DADC rr Double add with carry

(HL)<=(HL) + rr + CY

Double precision add the register pair and carry bit to the HL register pair.

1	1	1	0	1	1	0	1	ED
0	1	rp	rp	1	0	1	0	

cycles: 4 states: 15

flags:	S	Z	H	P/V	N	C
	X	X	X	V	0	X

SBC HL,BC
SBC HL,DE
SBC HL,HL
SBC HL,SP

DSBC rr Double Subtract with carry

(HL)<=(HL) - rr - CY

Double precision subtract the carry bit and the register pair from the HL register pair. C is set if no borrow occurs from bit 15.

1	1	1	1	1	1	0	1	ED
0	1	rp	rp	0	0	1	0	

cycles: 4 states: 15
flags: S Z H P/V N C
X X X V 1 X

ADD IX,BC
ADD IX,DE
ADD IX,IX
ADD IX,SP
ADD IY,BC
ADD IY,DE
ADD IY,IY
ADD IY,SP

DADX tt
DADY uu Double add index register

IX<=(IX + tt)
IX<=(IX + uu)

Double precision add the register pair to the index register. H set by carry from bit 11, C set by carry from bit 15.

1	1	*	1	1	1	0	1	DD/FD
0	0	rp	rp	1	0	0	1	

cycles: 4 states: 15
flags: S Z H P/V N C
* * X * 0 X

*0=IX 1=IY

NOTE: rp register pair code

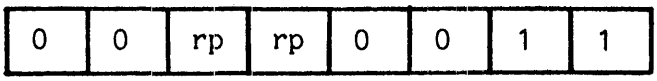
B	BC	00
D	DE	01
X	IX	10
SP	SP	11

INC BC
INC DE
INC HL
INC SP

INX rr Increment register pair

rr<=(rr + 1)

Increment the contents of the register pair rp by one. Note: No condition flags are affected.



cycles: 1 states: 6
flags: none

NOTE:

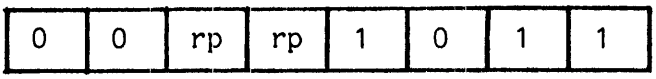
rp	register pair	code
B	BC	00
D	DE	01
H	HL	10
SP	SP	11

DEC BC
DEC DE
DEC HL
DEC SP

DCX rr Decrement register pair

rr<=(rr - 1)

Decrement the contents of the register pair rp by one. Note: No condition flags are affected.



cycles: 1 states: 6
flags: none

NOTE:

rp	register pair	code
B	BC	00
D	DE	01
H	HL	10
SP	SP	11

INX X
INX Y Increment index register

IX<=(IX + 1)
IX<=(IX + 1)

Increment the 16-bit index register. Do not affect the flags.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	0	0	0	1	1	23

cycles: 2 states: 10
flags: none

*0=IX 1=IY

DCX X
DCX Y Decrement register pair

IX<=(IX - 1)
IX<=(IX -1)

Decrement the 16-bit index register. Do not affect the flags.

1	1	*	1	1	1	0	1	DD/FD
0	0	1	0	1	0	1	1	2B

cycles: 2 states: 10
flags: none

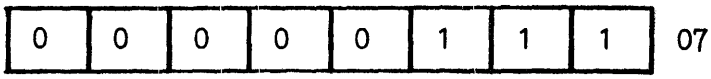
*0=IX 1=IY

RLCA

RLC Rotate accumulator left to carry

```
A(n + 1)<=A n
A 0<=A 7
CY<=A 7
```

Rotate the 8-bit accumulator left one position shifting the high-order bit into the low-order position and into the Carry flag. The previous carry bit is lost.



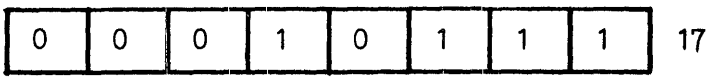

```
cycles: 1   states: 4
flags:  S Z   H   P/V N C
        * *   0    *  0 X
```

RLA

RAL Rotate accumulator left through carry

```
A(n + 1)<=A n
A 0<=CY
CY<=A 7
```

Rotate the 8-bit accumulator and Carry flag left. (9-bit rotate)

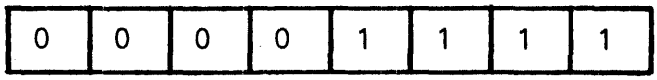



```
cycles: 1   states: 4
flags:  S Z   H   P/V N C
        * *   0    *  0 X
```

RRC Rotate accumulator right to carry

```
A n<=A(n + 1)
A(7)<=A(0)
CY<=A(0)
```

Rotate the 8-bit accumulator right one position shifting the low-order bit into the high-order position and into the Carry flag. The previous carry bit is lost.



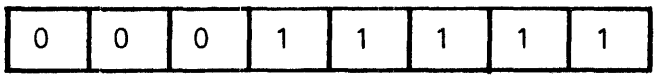
cycles: 1 states: 4

flags:	S	Z	H	P/V	N	C
	*	*	0	*	0	X

RAR Rotate right through carry

```
A n<=A(n + 1)
A 7<=CY
CY<=A 0
```

Rotate the 8-bit accumulator and Carry flag right. (9-bit rotate)



cycles: 1 states: 4

flags:	S	Z	H	P/V	N	C
	*	*	0	*	0	X

RLCR r Rotate register left to carry

```
r(n + 1)<=r n
r 0<=r 7
CY<=r 7
```

Rotate the 8-bit register left one position shifting the high-order bit into the low-order position and into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	0	0	0	r	r	r	

cycles: 2 states: 8
flags:

S	Z	H	P/V	N	C
X	X	0	P	0	X

RLCR M Rotate memory register left to carry

```
(HL)(n + 1)<=(HL) n
(HL) 0<=(HL) 7
CY<=(HL) 7
```

Rotate the 8-bit memory register left one position shifting the high-order bit into the low-order position and into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	0	0	0	1	1	0	06

cycles: 4 states: 15
flags:

S	Z	H	P/V	N	C
X	X	0	P	0	X

RLCRX i
RLCRY i Rotate indexed memory register left to carry

(IX + i)(n + 1)<=(IX + i) n
(IX + i) 0<=(IX + i) 7
CY<=(IX + i) 7

Rotate the 8-bit memory location addressed by the sum of the index register and the two's complement displacement i, left one position shifting the high-order bit into the low-order position and into the Carry flag. The previous carry bit is lost.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	0	0	0	1	1	0	06

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0 P 0 X

*0=IX 1=IY

RALR r Rotate register left thru carry

r(n + 1)<=r n
r 0<=CY
CY<=r 7

Rotate the 8-bit register and Carry flag left one position (9-bit rotate).

1	1	0	0	1	0	1	1	CB
0	0	0	1	0	r	r	r	

cycles: 2 states: 8
flags: S Z H P/V N C
        X X 0 P 0 X

RALR M Rotate memory register left thru carry

(HL)(n + 1)<=(HL) n
(HL) 0<=CY
CY<=(HL) 7

Rotate the 8-bit memory location and Carry flag left one position (9-bit rotate).

1	1	0	0	1	0	1	1	CB
0	0	0	1	0	1	1	0	16

cycles: 4 states: 15
flags: S Z H P/V N C
        X X 0 P 0 X

RL (IX+i)
RL (IY+i)

RALRX i
RALRY i Rotate indexed memory register left thru carry

(IX + i)(n + 1)<=(IX + i) n
(IX + i) 0<=CY
CY<=(IX + i) 7

Rotate the Carry flag and 8-bit memory location addressed by the sum of the index register and two's complement displacement i, left one position. (9-bit rotate)

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	0	1	0	1	1	0	16

cycles: 6 states: 23
flags: P Z H P/V N C
        X X 0 P 0 X

RRCR r Rotate register right to carry

r n<=r(n + 1)
r 7<=r 0
CY<=r 0

Rotate the 8-bit register right one position shifting the low-order bit into the high-order position and into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	0	0	1	rs	rs	rs	

cycles: 2 states: 8
flags: S Z H P/V N C
        X X 0 P 0 X

RRCR M Rotate memory register right to carry

(HL) n<=(HL)(n + 1)
(HL) 7<=(HL) 0
CY<=(HL) 0

Rotate the 8-bit memory register contents right one position shiftin the low-order bit into the high-order position and into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	0	0	1	1	1	0	0E

cycles: 4 states: 15
flags: S Z H P/V N C
        X X 0 P 0 X

RRCRX i
RRCRY i Rotate indexed memory register right to carry

(IX + i) n<=(IX + i)(n + 1)
(IX + i) 7<=(IX + i) 0
CY<=(IX + i) 0

Rotate the 8-bit memory location addressed by the sum of the index register and the two's complement displacement i, right one position shifting the low-order bit into the high-order position and into the Carry flag. The previous carry bit is lost.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	0	0	1	1	1	0	0E

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0 P 0 X

*0=IX 1=IY

RARR r Rotate register right thru carry

r n<=r(n + 1)
r 7<=CY
CY<=r 0

Rotate the 8-bit register and Carry flag right. (9-bit rotate)

1	1	0	0	1	0	1	1	CB
0	0	0	1	1	rs	rs	rs	

cycles: 2 states: 8
flags: S Z H P/V N C
        X X 0 P 0 X

RARR M Rotate memory register right thru carry

(HL) n<=(HL)(n + 1)
(HL) 7<=CY
CY<=(HL) 0

Rotate the 8-bit memory location and Carry flag right one position. (9 bit rotate)

1	1	0	0	1	0	1	1	CB
0	0	0	1	1	1	1	0	1E

cycles: 4 states: 15
flags: S Z H P/V N C
        X X 0 P 0 X

RR (IX+i)
RR (IY+i)

RARRX i
RARRY i Rotate indexed memory register right thru carry

(IX + i) n<=(IX + i)(n + 1)
(IX + i) 7<=CY
CY<=(IX + i) 0

Rotate the Carry flag and 8-bit memory location addressed by the sum of the index register and two's complement displacement i, right one position. (9-bit rotate)

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	0	1	1	1	1	0	1E

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0 P 0 X

*0=IX 1=IY

SRLR r Shift right logical

r n<=r(n + 1)
r 7<=0
CY<=r 0

Shift the 8-bit register right one position, shifting a zero into the high-order position and shifting the low-order bit into the Carry flag. The previous carry is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	1	1	rs	rs	rs	

cycles: 2 states: 8
flags:

S	Z		H		P/V	N	C
X	X		0		P	0	X

SRLR M Shift right logical memory

(HL) n<=(HL)(n + 1)
(HL) 7<=0
CY<=(HL) 0

Shift the 8-bit memory contents right one position, shifting a zero into the high-order position and shifting the low-order bit into the Carry flag. The previous carry is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	1	1	1	1	0	3E

cycles: 4 states: 15
flags:

S	Z		H		P/V	N	C
X	X		0		P	0	X

SRLRX i
SRLRY i Shift right logical indexed memory

(IX + i) n<=(IX + i)(n + 1)
(IX + i) 7<=0
CY<=(IX + i) 0

Shift the 8-bit memory contents addressed by the sum of the index register and the two's complement displacement i, right one position. A zero is shifted into the high-order position and the low-order bit is shifted into the Carry flag. The previous carry is lost.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	1	1	1	1	1	0	3E

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0 P 0 X

*0=IX 1=IY

SLAR r Shift left arithmetic

r(n + 1)<=r n
r 0<=0
CY<=r 7

Shift the 8-bit register left one position, shifting a zero into the low-order position and shifting the high-order bit into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	0	0	rs	rs	rs	

cycles: 2 states: 8
flags: S Z H P/V N C
        X X 0 P 0 X

SLAR M Shift left arithmetic memory

(HL)(n + 1)<=(HL) n
(HL) 0<=0
CY<=(HL) 7

Shift the 8-bit memory location left one position, shifting a zero into the low-order postion and shifting the high-order bit into the Carry flag. The previous carry bit is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	0	0	1	1	0	26

cycles: 4 states: 15
flags: S Z H P/V N C
        X X 0 P 0 X

SLARX i
SLARY i Shift left arithmetic indexed memory

(IX + i)(n + 1)<=(IX + i) n
(IX + i) 0<=0
CY<=(IX + i) 7

Shift the memory location addressed by the sum of the index register and the two's complement displacement i, left one position. A zero is shifted into the low-order position and the high-order bit is shifted into the Carry flag. The previous carry bit is lost.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	1	0	0	1	1	0	26

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0 P 0 X

*0-IX 1=IY

SRAR r Shift right arithmetic

r n<=r(n + 1)
r 7<=r 7
CY<=r 0

Shift the 8-bit register right one position, shifting the low-order bit into the Carry flag. The high-order bit is retained and copied into the next lower-order bit. The previous carry is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	0	1	rs	rs	rs	

cycles: 2 states: 8
flags: S Z H P/V N C
        X X 0 P 0 X

SRAR M Shift right arithmetic memory

(HL) n<=(HL)(n + 1)
(HL) 7<=(HL) 7
CY<=(HL) 0

Shift the 8-bit memory contents one position, shifting the low-order bit into the Carry flag. The high-order bit is retained and copied into the next lower-order bit. The previous carry is lost.

1	1	0	0	1	0	1	1	CB
0	0	1	0	1	1	1	0	2E

cycles: 4 states: 15
flags: S Z H P/V N C
        X X 0 P 0 X

SRARX i
SRARY i Shift right arithmetic indexed memory

(IX + i) n<=(IX + i)(n + 1)
(IX + i) 7<=(IX + i) 7
CY<=(IX + i) 0

Shift the 8-bit memory location addressed by the sum of the index register and the two's complement displacement i, right one position, shifting the low-order bit into the Carry flag. The high-order bit is retained and copied into the next lower-order bit. The previous carry is lost.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	0	1	0	1	1	1	0	2E

cycles: 6 states: 23
flags: S Z H P/V N C
        X X 0   P 0 X

*0 IX 1=IY

RLD Rotate left digit

A(0 - 3)<=(HL)(4 - 7)
(HL)(4 - 7)<=(HL)(0 - 3)
(HL)(0 - 3)<=(A)(0 - 3)

Rotate the high-order 4 bits of the memory location into the low-order 4 bits of the accumulator and rotate the low-order 4 bits of the memory location into the high-order 4 bits and rotate the low-order 4 bits of the accumulator into the low-order 4 bits of the memory location.

1	1	1	0	1	1	0	1	ED
0	1	1	0	1	1	1	1	6F

cycles: 5 states: 18
flags: S Z H P/V N C
        X X 0   P 0 X

RRD Rotate right digit

(HL)(0 - 3)<=(HL)(4 - 7)
(HL)(4 - 7)<=A(0 - 3)
A(0 - 3)<=(HL)(0 - 3)

Rotate the low-order 4 bits of the accumulator into the high-order 4 bits of the memory location and rotate the high-order 4 bits of the memory location into the low-order 4 bits of the memory location and rotate the low-order 4 bits of the memory location into the low-order 4 bits of the accumulator.

1	1	1	0	1	1	0	1	ED
0	1	1	0	0	1	1	1	67

cycles: 5 states: 18

flags:	S	Z	H	P/V	N	C
	X	X	0	P	0	X

2.6.8 Bit Set, Reset And Test Group

BIT b,rs

BIT b,r Bit test register

ZF<=#r b

Set the Zero flag if the register bit is zero, else reset the Zero flag.

1	1	0	0	1	0	1	1	CB
0	1	b	b	b	rs	rs	rs	

cycles: 2 states: 8
flags: S Z H P/V N C
        ? X 1 ? 0 *

BIT b,(HL)

BIT b,M Bit test memory

ZF<=#(HL) b

Set the Zero flag if the memory bit is zero, else reset the Zero flag.

1	1	0	0	1	0	1	1	CB
0	1	b	b	b	1	1	0	

cycles: 3 states: 12
flags: S Z H P/V N C
        ? X 1 ? 0 *

BITX b,i
BITY b,i Bit test indexed memory

ZF<=#(IX + i) b
Z<=#(IY + i) b

Set the Zero flag if the bit in the memory location addressed by the sum of the index register and the two's complement displacement i, is a zero, else reset the Zero flag.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
0	1	b	b	b	1	1	0	

cycles: 5 states: 20
flags: S Z H P/V N C
        ? X 1 ? 0 *

*0=IX 1=IY

SETB b,r Set bit register

r b<=1

Set bit in 8-bit register.

1	1	0	0	1	0	1	1	CB
1	1	b	b	b	rs	rs	rs	

cycles: 2 states: 8
flags: none

SET 6,(HL)

SETB b,M Set bit memory

(HL) b<=1

Set bit in 8-bit memory location.

1	1	0	1	1	0	1	1	CB
1	1	b	b	b	1	1	0	

cycles: 4 states: 15
flags: none

SET b,(IX+i)
SET b,(IY+i)

SETX b,i
SETY b,i Set bit indexed memory

(IX + i) b<=1
(IY + i) b<=1

Set bit in 8-bit memory location addressed by sum of index register and two's complement displacement.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	CB
i								
1	1	b	b	b	1	1	0	

cycles: 6 states: 23
flags: none

*0=IX 1=IY

RES b,rs

RES b,r Reset bit register

r b<=0

Reset bit in 8-bit register

1	1	0	0	1	0	1	1	CB
1	0	b	b	b	rs	rs	rs	

cycles: 4 states: 8
flags: none

RES B,(HL)

RES b,M Reset bit memory

(HL) b<=0

Reset bit in 8-bit memory location.

1	1	0	0	1	0	1	1	CB
1	0	b	b	b	1	1	0	

cycles: 4 states: 15
flags: none

RES b,(IX+i)
RES b,(IY+i)

RESX b,i
RESY b,i Reset bit indexed memory

(IX + i) b<=0
(IY + i) b<=0

Reset bit in 8-bit memory location addressed by sum of index register and two's complement displacement.

1	1	*	1	1	1	0	1	DD/FD
1	1	0	0	1	0	1	1	
i								
1	0	b	b	b	1	1	0	

cycles: 6 states: 23
flags: none

*0=IX 1=IY

2.6.9 Program Transfer Group

JP a16

JMP nn Jump

PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3.

1	1	0	0	0	0	1	1	C3
low-order addr								
high-order addr								

cycles: 3 states: 10
flags: none

JP Z,a16

JZ nn Jump on zero

if ZF=1, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Zero flag is set.

1	1	0	0	1	0	1	0	CA
low-order addr								
high-order addr								

cycles: 3 states: 10
flags: none

JP NZ,a16

JNZ nn Jump on non-zero

if ZF=0, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Zero flag is reset.

1	1	0	0	0	0	1	0	C2
low-order addr								
high-order addr								

cycles: 3 states: 10
flags: none

JP NC,a16

JNC nn Jump on no carry

if CY=0, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Carry flag is reset.

1	1	0	1	0	0	1	0	D2
low-order addr								
high-order addr								

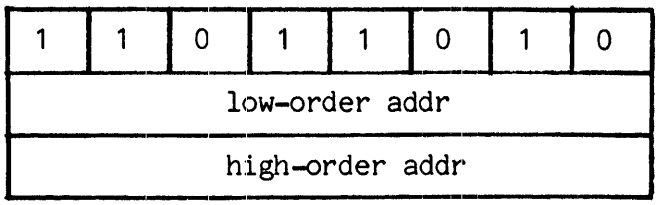
cycles: 3 states: 10
flags: none

JP CY,a16

JC nn Jump on carry

if CY=1, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Carry flag is set.



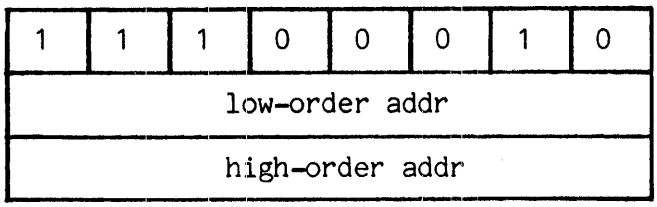
cycles: 3 states: 10
flags: none

JP O,a16

JPO nn Jump on parity odd

if PN=0, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is reset.

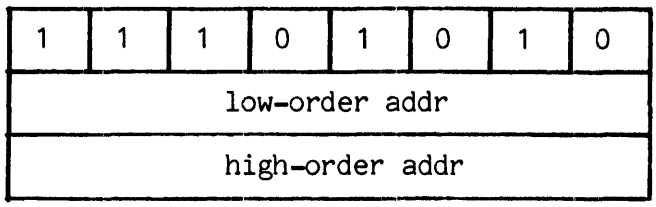


cycles: 3 states: 10
flags: none

JPE nn Jump on parity even

if P/V=1, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is set.

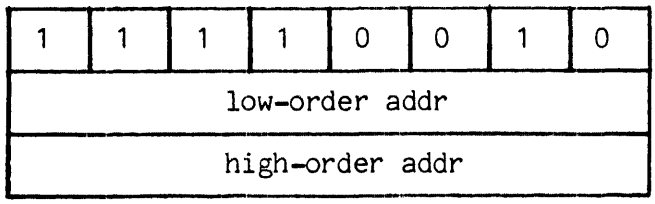


cycles: 3 states: 10
flags: none

JP nn Jump on plus

if S=0, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Sign flag is reset.



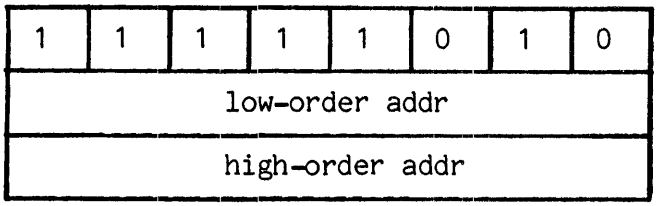
cycles: 3 states: 10
flags: none

JP M,a16

JM nn Jump on minus

if S=1, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Sign flag is set.



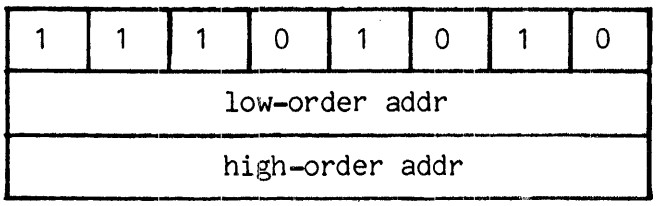
cycles: 3 states: 10
flags: none

JP PE,a16

JO nn Jump on overflow

if P/V=1, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is set.



cycles: 3 states: 10
flags: none

JP PO,a16

JNO nn Jump on no overflow

if P/V=0, PC<=nn

Load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is reset.

1	1	1	0	0	0	1	0	E2
low-order addr								
high-order addr								

cycles: 3 states: 10
flags: none

JP (HL)

PCHL Load program counter from HL

PC<=(HL)

Load the program counter with the content of the HL register pair.

1	1	1	0	1	0	0	1	E9

cycles: 1 states: 4
flags: none

JP (IX)
JP (IY)

PCIX
PCIY Load program counter from index register

PC<=(IXH)
PC<=(IXL)

Load the program counter with the content of the 16-bit index register.

1	1	*	1	1	1	0	1	DD/FD
1	1	1	0	1	0	0	1	E9

cycles: 2 states: 8
flags: none

*0=IX 1=IY

JMPR e Jump relative

PCL<=PCL + e

Add the 8-bit two's complement relative address to the lower 8 bits of the program counter.

0	0	0	1	1	0	0	0	18
relative addr - 2								

cycles: 3 states: 12
flags: none

JRZ e Jump relative on zero

if ZF=1
PCL<=PCL + e

If the Zero flag is set, add the 8-bit two's complement relative address to the lower 8 bits of the program counter.

0	0	1	0	1	0	0	0	28
relative addr - 2								

cycles: 3 states: 12
flags: none

JRNZ e Jump relative on non-zero

if ZF=0
PCL<=PCL + e

If the Zero flag is reset, add the 8-bit two's complement relative address to the lower 8 bits of the program counter.

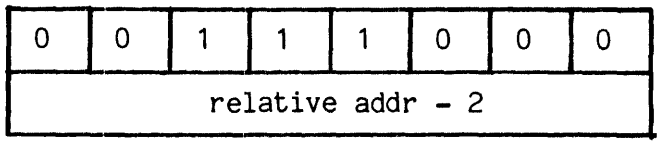
0	0	1	0	0	0	0	0	20
relative addr - 2								

cycles: 3 states: 12
flags: none

JRC e Jump relative on carry

if CY=1
PCL<=PCL + e

If the Carry flag is set, add the 8-bit two's complement relative address to the lower 8 bits of the program counter.

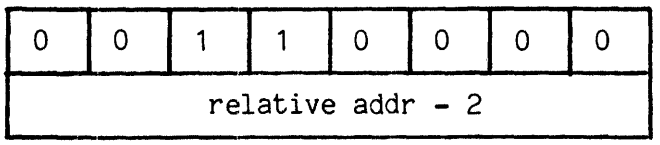


cycles: 3 states: 12
flags: none

JRNC e Jump relative on no carry

if CY=0
PCL<=PCL + e

If the Carry flag is reset, add the 8-bit two's complement relative address to the lower 8 bits of the program counter.

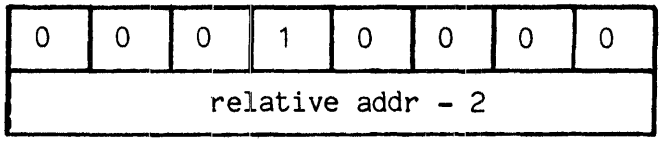


cycles: 3 states: 12
flags: none

DJNZ e Decrement and jump relative on non-zero

```
B<=B - 1
if B<>0
PCL<=PCL + e
```

Decrement the B register and if the result is not zero, add the 8-bit two's complement relative address to the lower 8 bits of the program counter.



cycles: 3 states: 13
flags: none

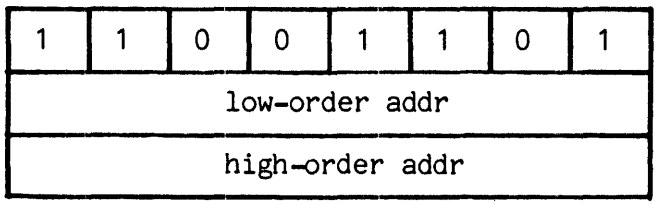
2.6.10 Call And Return Group

CALL a16

CALL nn Call

(SP - 1)<=PCH
(SP - 5)<=PCL
SP<=(SP - 2)
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3.



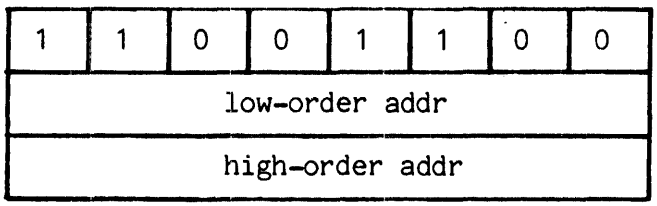
cycles: 5 states: 17
flags: none

CALL Z,a16

CZ nn Call on zero

if z=1
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Zero flag is set.



cycles: 3/5 states: 10/17
flags: none

CALL NZ,a16

CNZ nn Call on non-zero

if z=0
(SP - 1)<=PCL
(SP - 2)<=PCH
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Zero flag is reset.

1	1	0	0	0	1	0	0	C4
low-order addr								
high-order addr								

cycles: 3/5 states: 10/17
flags: none

CALL NC,a16

CNC nn Call on no carry

if c=0
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Carry flag is reset.

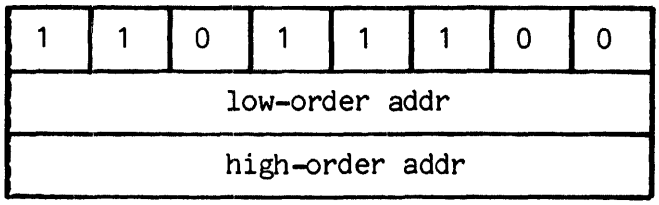
1	1	0	1	0	1	0	0	D4
low-order addr								
high-order addr								

cycles: 3/5 states: 10/17
flags: none

CC nn Call on carry

if c=1
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Carry flag is set.

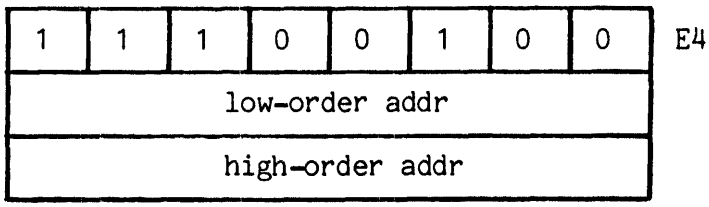


cycles: 3/5 states: 10/17
flags: none

CPO nn Call on parity odd

if P/V=0
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is reset.



cycles: 3/5 states: 10/17
flags: none

CPE nn Call on parity even

if P/V=1
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is set.

1	1	1	0	1	1	0	0	EC
low-order addr								
high-order addr								

cycles: 3/5 states: 10/17
flags: none

CP nn Call on plus

if S=0
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Sign flag is reset.

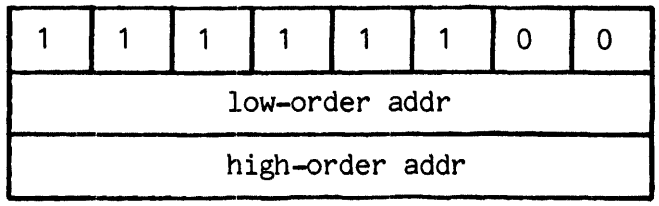
1	1	1	1	0	1	0	0	F4
low-order addr								
high-order addr								

cycles: 3/5 states: 10/17
flags: none

CM nn Call on minus

if S=1
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Sign flag is set.

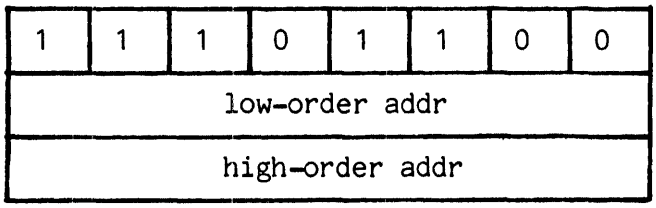


cycles: 3/5 states: 10/17
flags: none

CO nn Call on overflow

if P/V=1
(SP - 1)<=PCH
(SP - 2)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is set.

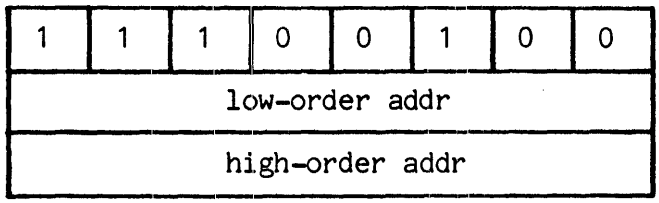


cycles: 3/5 states: 10/17
flags: none

CNO nn Call on no overflow

if P/V=0
(SP - 1)<=PCH
(SP - 1)<=PCL
PC<=nn

Push the program counter onto the stack and load the program counter with the 16-bit address in byte 2 and byte 3 if the Parity/Overflow flag is reset.

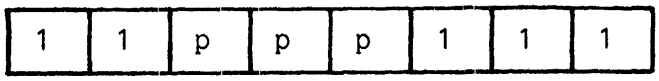


cycles: 3/5 states: 10/17
flags: none

RST p Restart

(SP - 1)<=PCH
(SP - 2)<=PCL
PCH<=0
PCL<=8p

Push the program counter onto the stack and load the program counter with one of 8 interrupt vector addresses.



cycles: 3 states: 11
flags: none

RET Return

PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)

Pop the top of the stack into the program counter. The previous contents of the program counter are lost.

1	1	0	0	1	0	0	1	C9

cycles: 3 states: 10
flags: none

RETN Return from non-maskable interrupt

PCH<=(SP + 1)
PCL<=(SP)
SP<(SP + 2)
IFF1<=IFF2

Pop the top of the stack into the program counter and copy the state of interrupt flip-flop 2 back into flip-flop 1.

1	1	1	0	1	1	0	1	ED
0	1	0	0	0	1	0	1	45

cycles: 4 states: 14
flags: none

RETI Return from interrupt

PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
IFF1<=0
IFF2<=0

Pop the top of the stack into the program counter and reset the interrupt flip-flops 1 and 2.

1	1	1	0	1	1	0	1	ED
0	1	0	0	1	1	0	1	4D

cycles: 4 states: 14
flags: none

RZ Return on zero

if ZF=1
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)

Pop the top of the stack into the program counter if the Zero flag is set.

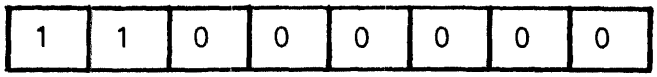
1	1	0	0	1	0	0	0	C8

cycles: 1/3 states: 5/11
flags: none

RNZ Return on non-zero

```
if ZF=0
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Zero flag is reset.



C0

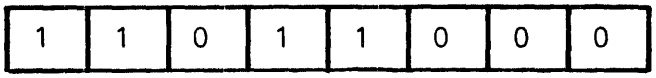
cycles: 1/3 states: 5/11
flags: none

RET C

RC Return on carry

```
if C=1
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Carry flag is set.



D8

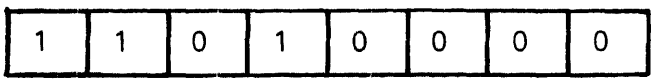
cycles: 1/3 states: 5/11
flags: none

RET NC

RNC Return on no carry

```
if CY=0
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Carry flag is reset.



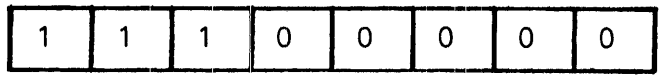
D0

cycles: 1/3 states: 5/11
flags: none

RPO Return on parity odd

```
if P/V=0
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Parity/Overflow flag is reset.



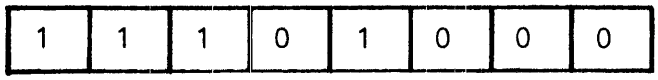
E0

cycles: 1/3 states: 5/11
flags: none

RPE Return on parity even

```
if P/V=1
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Parity/Overflow flag is set.



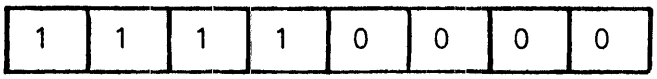
E8

cycles: 1/3 states: 5/11
flags: none

RP Return on plus

```
if S=0
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Sign flag is reset.



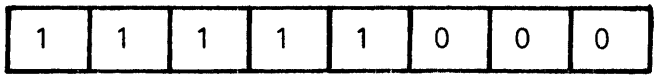
F0

cycles: 1/3 states: 5/11
flags: none

RM Return on minus

```
if S=1
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Sign flag is set.



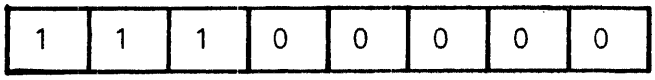
F8

cycles: 1/3 states: 5/11
flags: none

RO Return on overflow

```
if P/V=1
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Parity/Overflow flag is set.



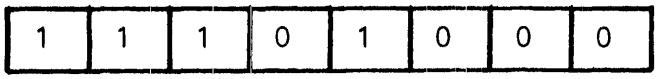
E0

cycles: 1/3 states: 5/11
flags: none

RNO Return on no overflow

```
if P/V=0
PCH<=(SP + 1)
PCL<=(SP)
SP<=(SP + 2)
```

Pop the top of the stack into the program counter if the Parity/Overflow flag is reset.



E8

cycles: 1/3 states: 5/11
flags: none

2.6.11 Input/Output Group

IN A,(aδ,

IN n Input to accumulator

A<= (In)

Input 8-bit data from input port to accumulator.

1	1	0	1	1	0	1	1	DB
port addr								

cycles: 3 states: 11
flags: none

ᵀN rd,(C)

INP r Input to register

r<=I (C)

Input 8-bit data from input port addressed by the C register to register rd.

1	1	1	0	1	1	0	1	ED
0	1	rd	rd	rd	0	0	0	

cycles: 3 states: 12
flags:

S	Z	H	P/V	N	C
X	X	0	P	0	*

INI Input to memory and increment

(HL)<=I (C)
B<=B - 1
(HL)<=(HL) + 1

Input 8-bit data from input port addressed by the C register to the memory location addressed by the HL register pair. Decrement the B register and increment the HL register pair.

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	1	0	A2

cycles: 4 states: 16
flags:

S	Z	H	P/V	N	C
?	X	?	?	1	*

Z = 0 when B - 1 =/ 0
Z = 1 when B - 1 = 0

INIR Input to memory, increment and repeat

(HL)<= (C)
B<=B - 1
(HL)<=(HL) + 1
PC<=PC - 2 until B=0

Input 8-bit data from input port addressed by the C register to memory location addressed by HL register pair. Increment the HL register pair. Decrement the B register and repeat until the B register becomes zero.

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	1	0	B2

cycles: 5/4 states: 21/16
flags:

S	Z	H	P/V	N	C
?	1	?	?	1	*

IND Input to memory and decrement

(HL)<=I (C)
B<=B - 1
(HL)<=(HL) - 1

Input 8-bit data from input port addressed by the C register to the memory location addressed by the HL register pair. Decrement the HL register pair and the B register.

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	1	0	AA

cycles: 4 states: 16
flags: S Z H P/V N C
? X ? ? 1 *

Z = 0 when B - 1 =/ 0
Z = 0 when B - 1 = 0

INDR Input to memory, decrement and repeat

(HL)<= (C)
B<=B - 1
(HL)<=(HL) - 1
PC<=PC - 2 until B=0

Input 8-bit data from input port addressed by the C register to the memory location addressed by the HL register pair. Decrement the HL register pair and the B register and repeat until B becomes zero.

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	1	0	BA

cycles: 5/4 states: 21/16
flags: S Z H P/V N C
? 1 ? ? 1 *

OUT (a8),A

OUT n Output accumulator

On<=A

Output 8-bit data to output port from accumulator.

1	1	0	1	0	0	1	1	D3
port addr								

cycles: 3 states: 11
flags: none

OUT (C),rs

OUTP r Output register

O(C)<=r

Output 8-bit data to output port addressed by the C register from register rs.

1	1	1	0	1	1	0	1	ED
0	1	rs	rs	rs	0	0	1	

cycles: 3 states: 12
flags: none

OUTI Output memory and increment

```
 O(C)<=(HL)
B<=B - 1
(HL)<=(HL) + 1
```

Output 8-bit data to output port addressed by the C register from the memory location addressed by the HL register pair. Decrement the B register and increment the HL register pair.

1	1	1	0	1	1	0	1	ED
1	0	1	0	0	0	1	1	A3

cycles: 4 states: 16

flags:	S	Z	H	P/V	N	C
	?	X	?	?	1	*

Z = 0 when B - 1 =/ 0
Z = 1 when B - 1 = 0

OUTIR Output memory, increment and repeat

```
 (C)<=(HL)
B<=B - 1
(HL)<=(HL) + 1
PC<=PC - 2 until B=0
```

Output 8-bit data to output port addressed by the C register from memory location addressed by HL register pair. Increment the HL register pair. Decrement the B register and repeat until the B register becomes zero.

1	1	1	0	1	1	0	1	ED
1	0	1	1	0	0	1	1	B3

cycles: 5/4 states: 21/16

flags:	S	Z	H	P/V	N	C
	?	1	?	?	1	*

OUTD Output memory and decrement

O(C)<=(HL)
B<=B - 1
(HL)<=(HL) - 1

Output 8-bit data from output port addressed by the C register from the memory location addressed by the HL register pair. Decrement the HL register pair and the B register.

1	1	1	0	1	1	0	1	ED
1	0	1	0	1	0	1	1	AB

cycles: 4 states 16
flags: S Z H P/V N C
? X ? ? 1 *

Z = 0 when B - 1 =/ 0
Z = 1 when B - 1 = 0

OUTDR Output memory, decrement and repeat

(C)<=(HL)
B<=B - 1
(HL)<=(HL) - 1
PC<=PC - 2 until B=0

Output 8-bit data to output port addressed by the C register from the memory location addressed by the HL register pair. Decrement the HL register pair and the B register and repeat until B becomes zero.

1	1	1	0	1	1	0	1	ED
1	0	1	1	1	0	1	1	BB

cycles: 5/4 states: 21/16
f a-s: S Z H P/V N C
? 1 ? ? 1 *

SECTION 3

MULTIBUS

3.1 SCOPE

This section describes the MULTIBUS convention and how the MSC 8009 operates in a MULTIBUS environment.

3.2 MULTIBUS CONVENTION

The MULTIBUS convention is a set of standard signal lines that interconnect a family of system modules. These modules include processors, memories and I/O interfaces. Some modules, such as the MSC 8009, may be a combination of all three. The physical structure of the system bus takes the form of a backplane that system modules plug into. Interconnections between modules are normally made via printed-circuit lines or wire-wrapped connections on a backplane.

Twenty address lines, sixteen bidirectional data lines, eight parallel interrupt lines, bus control signals, data transfer signals and power distribution lines make up the MULTIBUS. Many compatible modules use only eight data lines and sixteen address lines, including the MSC 8009.

The Control section consists of the memory control (MDRC/ and MWTC/), I/O control (IORC/ and IOWC/), Bus Contention Resolution controls (BCLK/, BPRN/, BPRO/, BREQ/ and BUSY/), Handshaking controls (XACK/ and AACK/), Interrupt lines (INT0/ thru INT7/), constant clock (CCLK/) and the initialization signal (INIT/).

A system bus convention may be considered as three buses -- the Data bus, the Address bus, and the Control bus. The Data bus provides the path over which data is transmitted between sources and destinations. Data travels on the Data bus between the CPU and memory, between CPU and I/O devices, between memory and I/O devices, or even between peripheral devices.

The Address bus specifies the sources and destinations of I/O devices or memories located externally to the CPU. When transferring data between a CPU register and an external I/O device or memory, the CPU places the addresses, and then the data transfer takes place.

The Control bus establishes direction and timing of the data to or from the selected I/O device or memory location. Separate control signals are provided for memory read (MRDC/), memory write (MWTC/), I/O read (IORC/), and I/O write (IOWC/) operations. At a minimum, the slave will respond with XACK/ to indicate the completion of a particular read or write operation.

3.3 MULTIBUS CONTROL

All modules connected to the MULTIBUS behave in a master/slave relationship. At any given time, only one device has control of the MULTIBUS, and this device is referred to as the Bus Master. The Bus Master drives the address bus and the control lines. Also, it can initiate data transfers with other devices on the bus called slaves. Either the master or slave can drive the data bus depending on the direction of the data transfer. A slave never drives the address bus. It merely responds to addresses that are asserted onto the bus by a Bus Master. Examples of a slave module are memory and simple I/O interfaces that cannot control the bus.

A bus arbitration scheme grants control of the bus to only one Bus Master when more than one device requests MULTIBUS control. Generally, Direct Memory Access (DMA) has priority over a processor. Data may be lost in a transfer from an unstoppable device such as a disk if access is delayed more than a few microseconds. When two processors vie for bus control, one is assigned a lower priority, and bus arbitration grants control based on priority. The position of each module in a serial-priority resolution configuration determines it's priority. If parallel-priority resolution is used, the order of connection to the priority resolution module establishes priority.

3.3.1 Bus Contention Resolution

Since the MULTIBUS can interconnect several devices that are capable of being a Bus Master, two or more of these devices could request bus control simultaneously. Thus, MULTIBUS control must be allocated so that important activities are performed more quickly than unimportant activities. Four MULTIBUS signals resolve bus contention -- Bus Priority In (BPRN/), Bus Priority Out (BPRO/), Bus Request (BREQ/) and Bus Busy (BUSY/). Each signal is synchronized with the Bus Clock BCLK/ to insure that two modules do not have control of the bus simultaneously.

3.3.1.1 Serial Bus Priority

In serial-bus configuration, the MSC 8009 is connected between two neighboring units, and it receives BPRN/ from the unit with the highest-priority and sends BPRO/ to the lower-priority unit (See Figure 3-1). A common bus ties BCLK/ and BUSY/ to all units, and BREQ/ is not used in a serial-bus contention techniques. When the MSC 8009 requests control of the MULTIBUS, it sends a 'high' on BPRO/ to all lower-priority modules to indicate that the bus is not available. To insure that the highest-priority module's request is passed to the lowest-priority unit, the MSC 8009 essentially 'ORs' its internal request with BRPN/ from the higher-priority unit to generate BPRO/. The highest-priority modules BPRN/ must be wired to a logic "low" condition.

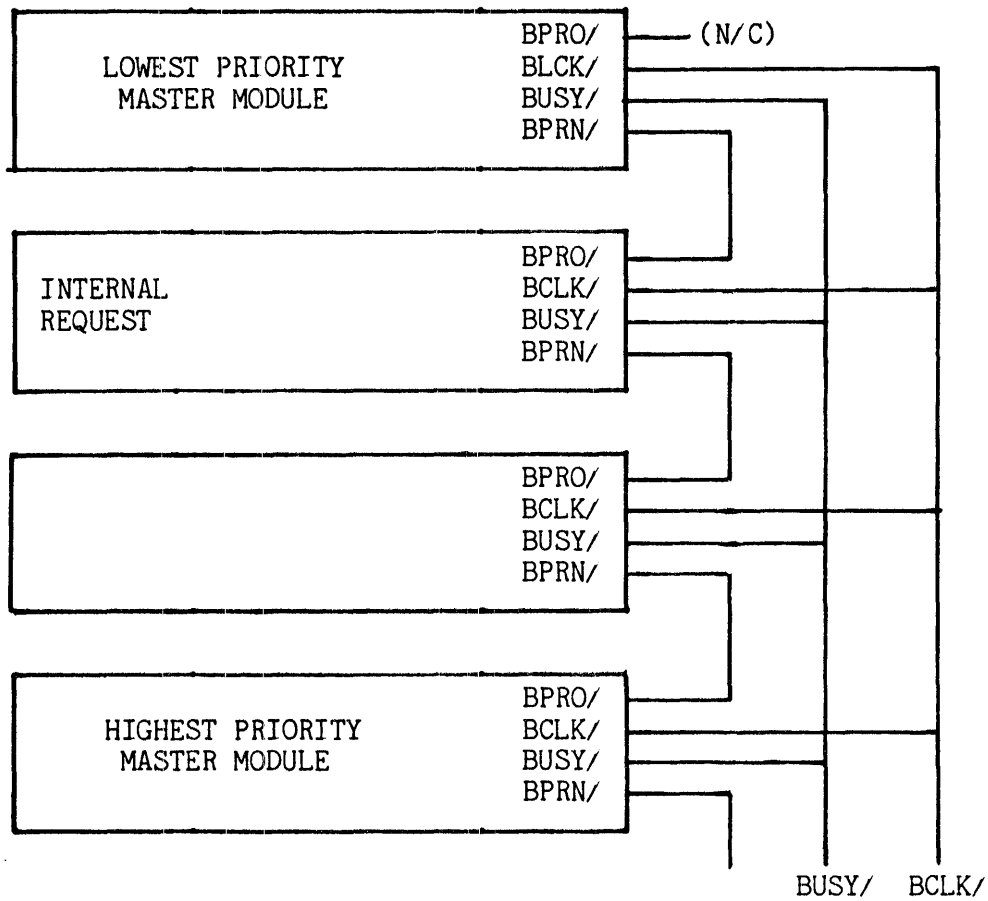


Figure 3-1
SERIAL-BUS CONTENTION CONFIGURATION

Once a request has been made, the MSC 8009 waits until BPRN/ goes 'low' (no higher-priority request is pending) and BUSY/ goes 'high' indicating that the current Bus Master has relinquished MULTIBUS control. The MSC 8009 now asserts BUSY/ to take control and initiate the requested data transfer operation. A problem still exists, however, if two Bus Masters request the bus. If both see BUSY/ go 'high' simultaneously and the higher-priority modules' request signal has not propogated down the chain to the lower-priority module, then both could conceively try to gain control of the MULTIBUS. Synchronizing all of the bus contention lines to a single clock resolves this problem. The Bus Clock (BCLK/) has a minimum period of 100 nanoseconds. The data on the bus contention lines should change, if necessary, on the high-to-low transition of the clock. In fact, to avoid the problem of two bus masters attempting to gain bus control simultaneously, a module should bid for bus control (setting BPRO/ 'high') on one clock cycle, but do not attempt to claim control of the bus (setting BUSY/ 'low') until the next clock cycle. This provides sufficient time for all higher-priority requests to propogate down the serial chain.

NOTE: It is not recommended to chain more than three devices with a BCLK/ of 100 ns. To support more devices, either increase the pulse-width of BCLK/ or use the parallel scheme.

3.3.1.2 Parallel Bus Priority

The parallel bus configuration requires an external printed-circuit board wired into the backplane. This board accepts the bus request signals from all bus masters and determines which signal has the highest priority. The arbitration circuit on this board consists of a parallel priority encoder such as a 74148 followed by a data selector chip such as 74S138. In turn, these circuits grant the MULTIBUS to the bus master having the highest priority.

3.3.1.3 Bus Exchange Modes

Bits 4 and 5 of the Board Configuration register (U22) lets the user program three bus-exchange modes (Refer to paragraph 1.4.1). The normal mode of operation (both bits are '00') allows the bus master to be changed every operation. This mode permits fast DMA response.

If the pattern for bits 5 and 4 is '10' respectively, the bus to change bus control with each instruction. This mode lets read-modify-write instructions (i.e., INR M) to be used to test and set flag in a multiprocessor environment so that external resources may be shared without software delays.

When bit 4 is a '1', the MSC 8009 keeps the bus once it gains control. This mode prevents other bus masters from acquiring the bus, which can be useful at system initialization time.

NOTE: This mode must be used with caution. Other processors or DMA devices may time out if they cannot gain bus control.

3.3.2 Acknowledge Signals

The MULTIBUS uses two types of acknowledge -- Transfer Acknowledge (XACK/) and Advance Acknowledge (AACK/). The Z80 timing characteristics and the slave unit dictate the use of these acknowledge signals. Care must be taken to insure that all modules on the MULTIBUS meet all timing requirements.

The MSC 8009 with the Z80A processor runs faster than the 8080 master modules with which AACK/ was intended for. Jumpers on existing memory and I/O boards that generate AACK/ in advance of XACK/ may have to be changed to assure proper operation if the CPU is changed from an 8080A to a Z80A. On the standard MSC 8009, AACK/ is disconnected and may be incorporated via jumper 66 to 67.

NOTE: AACK/ does not replace XACK/. XACK/ must always be used.

3.3.2.1 Transfer Acknowledge (XACK/)

When a memory or I/O port completes a read/write operation, it responds with XACK/. If the requested operation is a read, XACK/ signifies that the requested data is available on the MULTIBUS. The MSC 8009 accepts and loads the requested data upon receipt of XACK/. The Z80 samples the WAIT/ input approximately one cycle prior to the actual transfer of data. If WAIT is 'low', the Z80 enters one or more Wait states until WAIT/ goes 'high'. Then the Z80 readies the data during the next cycle. As shown in Figure 3-2, the slave device has up to Twsp (command to Z80 Wait sample point) to indicate valid data on the MULTIBUS when the Z80 needs it for entry after a no wait state.

For a write operation, the slave places a 'low' on XACK/ after accepting and latching the data into the addressed I/O port or memory. At this time, the MSC 8009 terminates the command, ready for another transaction.

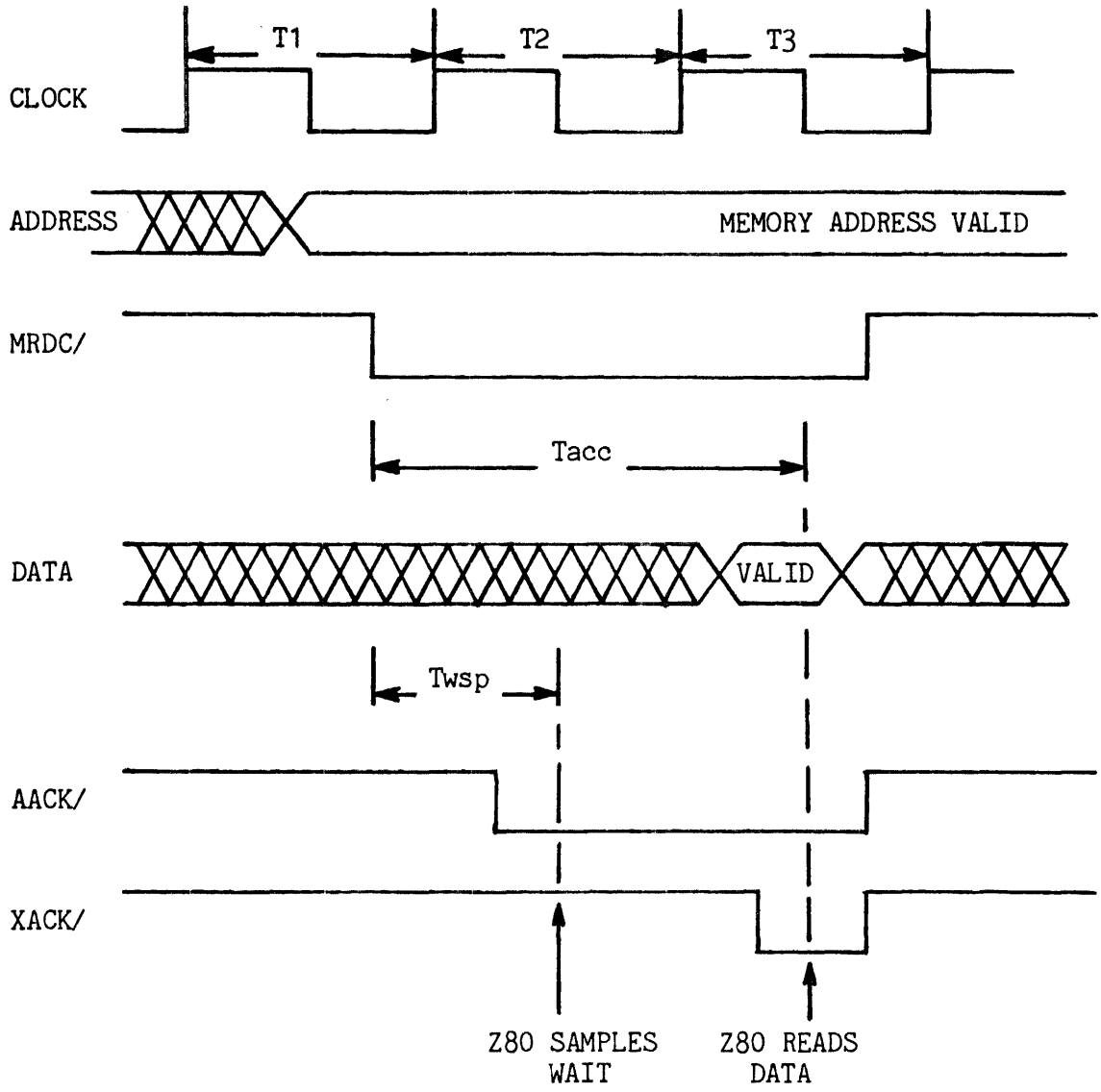


Figure 3-2
Z80 READ OPERATION

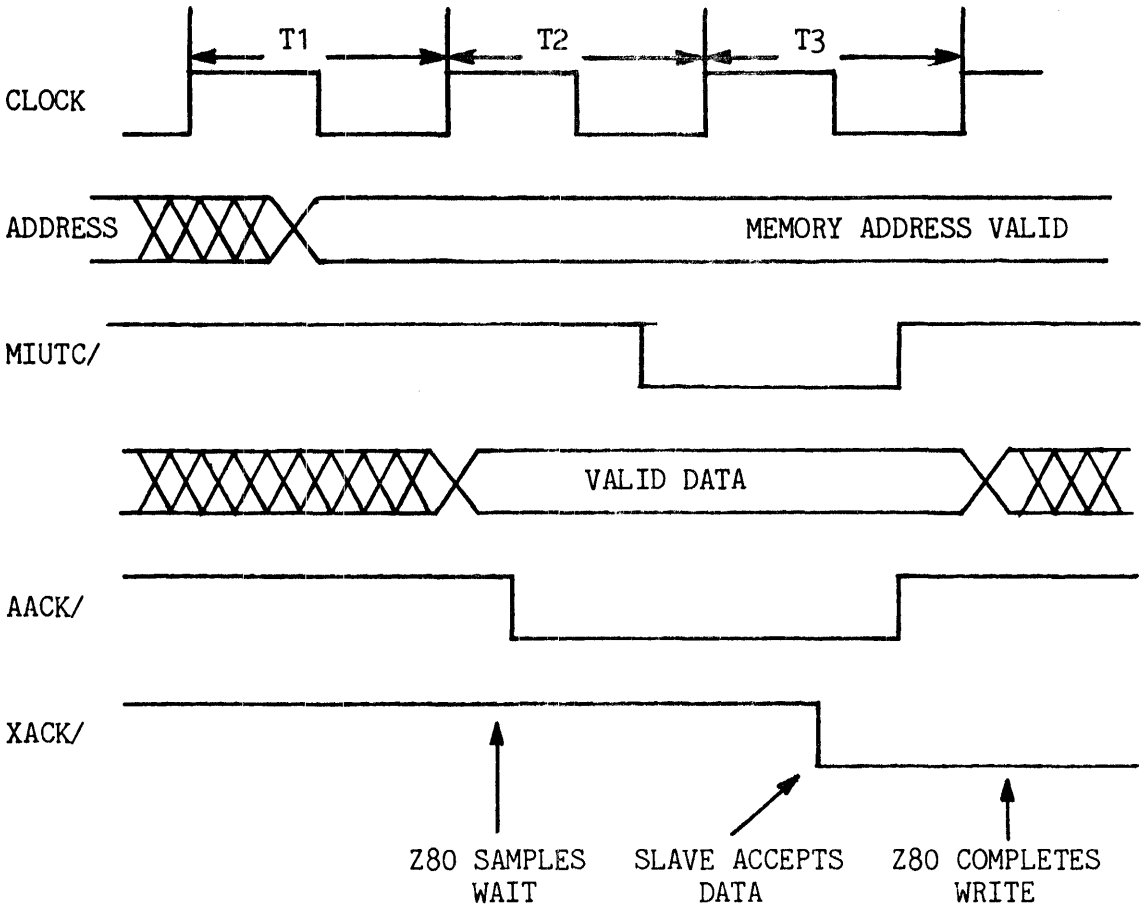


Figure 3-3
Z80 WRITE OPERATION

3.3.2.2 Advance Acknowledge (AACK/)

To speed read/write operations, some MULTIBUS devices use the optional Acknowledge signal AACK/ to reduce the number of WAIT states. If the slave module accepts data after the WAIT sample point (See Figure 3-3) and prior to the termination of a write command, AACK/ can be used to eliminate an extra Wait state. This signal is useful when XACK/ is too late for detection by the Z80 at the Wait sample point. Nevertheless, valid data is available on the MULTIBUS when the Z80 needs it; or the slave will accept the data before the Z80 completes a write operation.

During a read, many systems have a well-defined time lag between the receipt of the acknowledgement and the actual acceptance of data from the MULTIBUS. An Advance Acknowledge signal (AACK/) can be used for these conditions. This signal speeds throughput of certain systems such as the MSC 8009, and may occur prior to XACK/ by a time period equal to the time lag. Since the master module may not actually sample data as soon as acknowlege is received, a time lag could result. The master module must have consistent time lag in order to properly use AACK/ in cases where the time lag is not well defined, the speed improvement offered by AACK/ may have to be abandoned in favor of the slower, but reliable XACK/ signal.

3.4 SPECIFICATIONS

The MSC 8009 is designed to use the MULTIBUS convention. All signals listed in paragraph 3.3.3 are interfaced via a single, 86-pin connector (P1).

3.4.1 Electrical Characteristics

A 'low' (nominal 0V) on the active-low bus indictes a logic '1'. A slash (/) following the signal mnemonic indictes that the bus signal is active low. Timing and electricl specifications for all signals sent by the MSC 8009 are listed in Tables 3-1 and 3-2

3.4.2 Mechanical Characteristics

The bus connector is an 86-pin edge connector (two rows of 43 pins) with 0.156-inch spacing between pins. Board thickness of the MSC 8009 is nominally 0.062-inches, and spacing between installed boards should be 0.6-inches. Odd-numbered pins are located on the component side or top side of the board, and even-numbered pins on the circuit side or bottom. When viewing the board with the bus connector P1 down, pin 1 is located on the component side to the left.

1) Input Voltage Levels: High 2.0V to 5.0V
   Low 0.0V to 0.8V

2) Output Voltage Levels: High 2.4V to 5.25V
   Low 0.0V to 0.45V

3) Leakage Current of an Input: 0.04 mA

4) Leakage Current of an Output: 0.1 mA

5) Maximum Bus Capacitance: 300 pf on any one line

6) The National DS 8303 and DS 8304 octal bidirectional transceivers have excellent characteristics and fully comply with this specification.

Table 3-1
MULTIBUS LEVEL SPECIFICATIONS

3.4.3 Signal Description

Bus connector P1 provides the signal path between the MSC 8009 and the MULTIBUS for the following signals.

ADR0/ thru ADRF/	Sixteen address lines identify either memory location or I/O port to be accessed. Bit 0 (ADR0/) is least significant.
DAT0/ thru DAT7/	Eight bidirectional data lines transmit or receive data to and from addressed memory location or I/O port. Bit 0 (DAT0/) is least significant.
BCLK/	Bus Clock synchronizes bus contention logic when the MSC 8009 controls the MULTIBUS.
CCLK/	Clock signal available for other system modules.
INIT/	System Initialization signal conditions the MSC 8009 and any other module to a known state.
IORC/	Read Command signal indicates the address of an I/O port is on the MULTIBUS address lines, and data read via that port is to be placed onto the MULTIBUS data lines.
IOWC/	Write Commands allows the MSC 8009 to send data via MULTIBUS to the I/O port designated on the address lines.
MRDC/	Memory Read command indicates that the address of the memory location is on the MULTIBUS and data readd from that location is to be placed on the MULTIBUS.
MWTC/	Memory Write command transfers data from the data lines of the MULTIBUS into the memory location contained on the address lines.
XACK/	Transfer Acknowledge signal tells the MSC 8009 that the memory or I/O operation is complete.

AACK/	Optional Advance Acknowledge signal notifies the MSC 8009 that data will be available for the Z80. This signal must be used with caution because newer processors are faster than the 8080 for which many systems were designed.
INT0/ thru INT7/	Activating one of these eight, interrupt-request lines produced an interrupt operation. An interrupt must be held until the software releases the requesting module. INT0/ has the highest priority, and INT7/ has the lowest.
BPRN/	Bus Priority signal indicates that there is no master module with higher priority request and MULTIBUS control.
BPRO/	Bus Priority Out signal indicates that neither the MSC 8009 nor any higher priority master wants control of the MULTIBUS (for serial arbitration).
BUSY/	Busy signal signifies that some bus master has control of the MULTIBUS when asserted.
BREQ/	Bus request signal is asserted when the MSC 8009 wants control of the MULTIBUS.
NMI/	Non Maskable Interrupt is not a standard MULTIBUS signal, but can be connected to the MULTIBUS. It has highest interrupt priority.

3.4.4 Data Transfer Timing

The MULTIBUS data transfer operates on a handshaking principle so that no one module depends on another's internal timing. When the MSC 8009 issues a command to either read from or write into the location specified on the address lines, the MSC 8009 waits in a state of suspension until the slave module acknowledges that the transfer is complete.

The sequence of a data transfer operation is as follows (Refer to Figure 3-4):

1) When the MSC 8009 gains control of the MULTIBUS, the memory or I/O address is asserted onto the adddress lines.

2) If data is to be written into the slave, data is placed on the data bus.

3) After address and data lines have become stable for at least 100 nanoseconds, one of the following commands is then asserted:

   Memory Write Command (MWTC/)
   Memory Read Command (MRDC/)
   I/O Write Command (IOWC/)
   I/O Read Command (IOC/)

4) The command remains on the MULTIBUS until the request slave module acknowledges. For a write operation, this occurs when the slave accepts the data. If the requested operation is a read, acknowledgement comes when the slave has stable data on the MULTIBUS. To prevent an errant module from delaying the system operation, the MSC 8009 has a Watchdog Timer that reclaims control after 10 milliseconds.

5) When the MSC 8009 receives the acknowledge, the command is then removed from the MULTIBUS. In turn, the slave must remove its acknowledge within 100 nanoseconds.

The maximum data transfer rate for the MULTIBUS is 5 MHz. There is a 50-nanosecond address and data set up requirement, and the command signal must be at least 100 nanoseconds in addition to a 50-nanosecond data-hold requirement. These yield a theoretical minimum data-transfer time of 200 nanoseconds if the memory-access time is zero. Due to bus-arbitration timing and memory-access time, the actual rate is more on the order of 1 MHz.

3.5 MSC 8009 CONFIGURATION

There are many applications that use the standard MULTIBUS. The following list summarizes the MSC 8009 as a MULTIBUS module:

MODULE TYPE: Master or Slave.

PRIORITY: Depends on location within the priority structure of the system.

DATA BUS: Uses lower 8-bits only (DAT0 thru DAT7). Data lines DAT8/ thru DATF/ are not connected.

INHIBITS: INH1/ and INH2/ are not used.

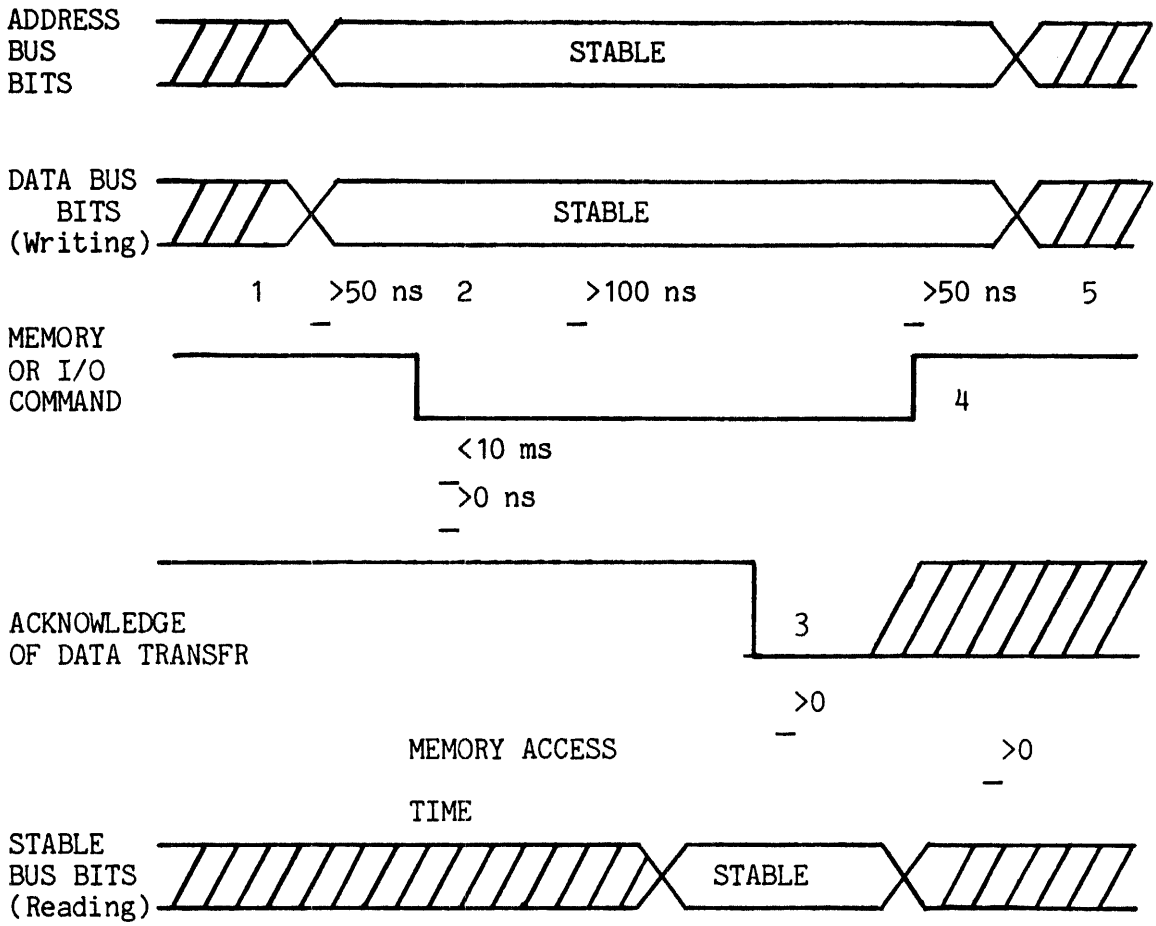
CLOCK: Standard frequency for both BCLK/ and CCLK/ is 8 MHz.

ACKNOWLEDGE: Jumpers 66 and 67 disconnect AACK/ from the MULTIBUS when modules designed for slower processors are used.

POWER SUPPLIES:

1) -10V is not used.
2) -12V is used with RS-232-C Serial I/O option.
3) +12V, +5V and -5V are used.

LEVEL SPECIFICATIONS: TTL Compatible



MEMORY OR I/O COMMAND = IORC, IOWC, MRDC AND MWTC
ACKNOWLEDGEMENT OF DATA TRANSFER = XACK AND AACK

Figure 3-4
MULTIBUS DATA TRANSFER TIMING

SIGNAL	DESCRIPTION	MIN.	MAX.	REMARKS
BLCK	Bus Clock	100ns	DC	35-65% duty cycle
ADR0/-ADRF/	Address Line Set Up	50ns		Relative to command assertion
DAT0/-DATF/	Write Data Set Up	50ns		Relative to command assertion
ADR0/-ADRF/	Address Line Hold	50ns		Relative to command removal
DAT0/-DATF/	Write Data Hold	50ns		Relative to command removal
DAT0/-DATF/	Read Data Set Up	0ns		Relative to XACK/
DAT0/-DATF/	Read Data Hold	0ns		Relative to command removal
XACK/, AACK/	Acknowledge Hold		100ns	Relative to command removal
XACK/	Acknowledge Delay		10ns	With optional Watch-dog Timer enabled
DAT0/-DATF/	Read Data Access	0ns		Maximum is DC with Watchdog Timer disabled
MRDC/, MWTC	Command Pulse Width	100ns		
IORC/, IOWC/				
INH1/, INH2	Inhibit Delay	100ns		Relative to address assertion
XACK/	Slave Acknowledge Delay	*		Inhibiting Slave must delay acknowledge

Table 3-2
MULTIBUS ELECTRICAL REQUIREMENTS

BUS SIGNALS	DRIVER		RECEIVER		TERMINATED ON BACKPLANE
	LOCATION	DRIVE	LOCATION	SOURCING	
INIT/	Master	Open Coll 32 mA	Any	1.8 mA	2.2K Ohm
BCLK/ CCLK/	Master	TTL 48 mA	Master	2.0 mA	220/330 Ohm
BREQ/	Master	TTL 10 mA	Any	2.0 mA	1.0K Ohm
BPRN/	Any	TTL 16 mA	Master	2.0 mA	None
BPRO/	Master	TTL 32 mA	Master	2.0 mA	None
BUSY/	Master	Open Coll 20 mA	Master	2.0 mA	1.0K Ohm
MRDC/ MWTC/	Master	Tri-State 32 mA	Slave	2.0 mA	1.0K Ohm
IORC/ IOWC/	Master	Tri-State 32 mA	Slave	2.0 mA	1.0K Ohm
XACK/ AACK/	Slave	Tri-State 16 mA	Master	2.0 mA	510 Ohm
DAT0/- DATF/	Any	Tri-State 15 mA	Any	0.5 mA	2.2K Ohm
ADR0/ ADRF/	Master	Tri-State 15 mA	Slave	0.5 mA	2.2K Ohm
INT7/- INT0/	Any	Open Coll 16 mA	Master	2.0 mA	1.0K Ohm

Table 3-2 (cont.)
MULTIBUS ELECTRICAL REQUIREMENTS

SECTION 4

MEMORY

4.1 SCOPE

This section describes the possible memory configurations of the MSC 8009, and how each can be employed to the user's advantage. In the basic system, a sixteen-chip dynamic RAM array provides up to 32K-byte storage capacity. These arrays are addressed by either the address lines of the Z80A processor or the MULTIBUS (Direct Memory Access).

Four on-board ROM/EPROM sockets allow up to 32K bytes of storage for commonly used subroutines, standard support software and specific user applications. These sockets accommodate the most popular 8-bit wide, 24-pin memory devices.

4.2 RAM CONFIGURATION

Divided into two 16K-byte sections, the sixteen-chip memory array comes unloaded, or in one of the following configurations using industry standard, 16-pin devices.

CAPACITY	ORGANIZATION	REQUIRED NUMBER
32K	16K X 1	16
16K	16K X 1	8

NOTE: If the device supplied is replaced with another, care must be taken to insure pin compatibility, and that the access time from CHIP ENABLE is less than 200 nanoseconds for use with the Z80A.

Separate MULTIBUS compatible memory cards such as the MSC 4602 (up to 64K bytes of RAM, 8K bytes of EPROM); the MSC 8103 (up to 8K bytes of RAM, four ROM/EPROM sockets); or the MSC 8104 (up to 32K bytes of RAM, four ROM/EPROM sockets) are available for special applications that require more than 32K bytes.

4.2.1 Addressing

Memory space allotment for the MSC 8009 is zero to 64K and can be accessed either directly by the Z80A or remotely from another bus master via the MULTIBUS. Inasmuch as on-board RAM access does not require use of the MULTIBUS, concurrent operations requiring the MULTIBUS can be performed.

During RAM operation, the fourteen address bits required to define one of 16,384 address locations are multiplexed onto the internal MA BUS (MA0 thru MA6). These bits are latched into on-chip address latches under control of two signals -- Row Address Strobe (RAS) and Column Address Strobe (CAS). RAS 1/ and RAS 2/ designate the 16K byte section as defined by signal RMBNK (RAM bank) from the program PROM. A 3480 RAS/CAS Timing chip (U67) generates all necessary control and timing for these two signals.

4.2.2 Memory Read

The execution of any program is a sequence of read and write cycles that transfer a byte of unique information to and from an addressed location in memory. There are two operations that require data to be transferred from memory -- instruction FETCH cycle and a data or read command. The contents of the program counter, which points to the current instruction, provides the memory address for the FETCH cycle while the address for a read command can have several origins. Also, data read during the read command is placed in a designated location, and the FETCH cycle places the data into the instruction register.

During the read cycle, the data output lines MRD0 thru MRD7 go from a high-impedance state to an active state so that the data read from the selected memory location is available on the internal data bus. This data remains on the bus until CAS returns to a "high" level. The memory data is latched so that refresh cycles can occur. The latch output is placed on the internal data bus and sent either to the MULTIBUS or the Z80A, depending upon the operation.

4.2.3 Memory Write

The combination of WRT/ and CAS/ latches the data that is to be written into a selected memory location in an on-chip register while RAS/ is active. The signals WRT/ and CAS/ go "low" simultaneously to strobe the data on the I/O BUS (IO0 thru IO7) into the memory elements. This data is placed on the I/O BUS by either the MULTIBUS (DAT0/ thru DAT7/) or the Z80A (D0 thru D7).

4.2.4 Refresh Cycle

Refresh of the dynamic storage cell requires each row address to be cycled within two milliseconds, or a minimum of 128 cycles every two milliseconds. When either a memory read or write operation is in progress, the refresh cycle is inhibited. When the designated cycle is complete, a refresh may begin following a short-time delay.

If a refresh has just taken place, another refresh will not start for approximately six microseconds. During the next ten microseconds, an M1 cycle (M1/ goes "low") will generate a refresh. On the other hand, if no M1 cycle occurs during the ten microseconds, a refresh is then forced.

CHIP	ORGANIZATION	EPROM/ROM SOCKET	U28 JUMPER POSTS					
			Gnd	-5V	+12V	+5V	LAA	LAB
2708	1K X 8	U101	11-12	4-13	1-14	-	-	-
(EPROM)		U102 thru U104	11-8	4-9	1-10	-	-	-
2716	2K X 8	U101	11-12	-	-	5-9	2-14	-
(EPROM)		U102 thru U104	11-8	-	-	5-13	2-10	-
TMS2532	4K X 8	U101	-	-	-	5-9	2-14	3-12
(EPROM)		U102 thru U104	-	-	-	5-13	2-10	3-8

NOTE: There are other 24-pin compatible EPROM/ROM that can be used. Ground, +5V, -5V, +12V and -12V in addition to LAA, LAB and LAC (used with 8K X 8 chips) are available on the JUMPER POSTS of U28 (See Drawing 305-0261-000, Sheet 7 for POST identification). Before installing a new device, remove all jumpers from U28.

Table 4-1
PROGRAM MEMORY JUMPER CONNECTIONS

4.3 EPROM/ROM CONFIGURATION

Four EPROM/ROM sockets accommodate the most popular 8-bit wide, 24-pin memory devices. Table 4-1 lists some of the devices that can be used with jumper-post call-out for socket U28. Different chips may be intermixed, but three of the four sockets must contain the same EPROM/ROM (U102 thru U104) and the fourth with another type (U101). For example, commonly used subroutines or standard support software may be installed in U102 thru U104 using three 2716's (2K X 8 EPROM) with one Masked ROM (8K X 8) in U101 -- total capacity of 18K bytes.

The use of a 4 MHz, Z80A requires memories that have a maximum access time of approximately 275 nanoseconds. This means that fast ROM, fusible-link PROM or VMOS EPROM must be used for program storage to eliminate wait states. One such chip is the Signetics 82S2708 (bipolar 1K X 8 PROM). This unit is compatible with the 2708 and has a maximum access time of 70 nanoseconds. Selected EPROM parts are available from Texas Instruments and Motorola too. Generally, the speed penalty imposed by a wait state on the M1 cycle is insignificant for system applications. The MSC 8009 is set up to insert one wait state on M1 cycles. This allows the popular 450-nanosecond PROM to be used.

4.3.1 EPROM/ROM Addressing

A PROM in conjunction with two decoders control all I/O and address mapping -- reducing the amount of discrete logic. Inasmuch as the on-board PROM/EPROM or RAM may be addressed for a unique 256-byte boundary in the 64K addressable space, any one of 256 device codes may be assigned to the I/O devices on the MSC 8009. The memory and I/O allocation for a standard configuration fulfills a number of applications. However, the PROM content may be altered to perform special memory and I/O mapping needs.

The eight, high-order address bits (LA8 thru LAF) are routed to a high-speed 256 X 8 PROM. This PROM may be mapped for any combination of the eight, most-significant address bits into any other pattern. This means that memories can be relocated to begin at an arbitrary 256-byte page boundary within the addressing space. The low-order addressing bits (LA0 thru LAF) go to the appropriate memory devices, allowing individual word selection.

The I/O instruction (one of 256 port addresses) reference all I/O ports that exist in the addressing space rather than being memory mapped. Decoder U21 accepts the levels on pins 7 thru 9 (D2 thru D4, respectively) of U54 and translates them into an enabling signal (CS0 thru CS5) when IOSEL/ is asserted.

To retrieve data from the EPROM (or ROM), the processor asserts a 16-bit address on the L/A BUS. This address selects one of four ROM/EPROM devices (ROMSEL 1 thru ROMSEL 4), and the data contained in the addressed location is then sent to the processor via the I/O BUS -- ROM/EPROM outputs are bussed together.

4.3.2 Memory Protect

A "true" on EXTRQ/ (pin 11 of U53) indicates that an off-board master can access the MSC 8009 memory or I/O devices. The Protect PROM (U53) selects which MSC 8009 addresses may be accessed from off the board.

4.4 DUAL MAP CONFIGURATION (OPTIONAL)

A 512 X 8 PROM and a flip-flop make up the Dual Map feature. This combination provides the MSC 8009 with two, complete address maps; and the system always powers up under the first map. A proper start requires location "0" to be in ROM. Then the program switches to the other map that may contain RAM in low memory. Bit 6 on the I/O BUS sets the MAP flip-flop U22 with an I/O output instruction. This same instruction programs the interrupt controller as discussed in Section 7. An output device code D7H with bit 6 set switches the system to the alternate address map. If bit 6 is reset, an output to D7H returns the system to the main-address map.

Selecting the alternate map without the proper PROM disables all on-board resources. Care must be taken not to interfere with the interrupt system that is controlled by bit 7, and bits 0 thru 3.

Primarily, the eight-output signals of PROM U54 select ROM, RAM, and I/O port (See Table 4-2). PROM U45 accepts and uses the output signal RAMSEL/ to generate the RAM request signal RMRQ/, which initiates the requested memory cycle.

SELECT SIGNAL	ADDRESS SPACE			
	4K	8K	16K	32K
ROMSEL 1	0000-03FF	0000-07FF	0000-0FFF	0000-1FFF
ROMSEL 2	0400-07FF	0800-0FFF	1000-1FFF	2000-3FFF
ROMSEL 3	0800-0BFF	1000-17FF	2000-2FFF	4000-5FFF
ROMSEL 4	0C00-0FFF	1800-1FFF	3000-3FFF	6000-7FFF

(a) PROM/ROM Allocation

PROM ADDRESS	SELECT SIGNAL	I/O PORT
DC-DF	CS0	TIMER
CE-CF	CS1	SERIAL (USART) #2
ED-EF	CS2	SERIAL (USART) #1
D7	CS3	INTERRUPT
D4-D5	CS4	APU
C0-C7	CS5	FLOPPY DISK INTERFACE
C4	CS6	FLOPPY DISK INTERFACE

(b) RAM & I/O Allocation

Table 4-2
MEMORY ALLOCATION

SECTION 5

SERIAL I/O INTERFACE

5.1 SCOPE

This section explains the serial I/O interface operation that provides the MSC 8009 with two serial-data communication channels. Using program control, the user can operate the MSC 8009 with synchronous or asynchronous byte-oriented, serial-data-transmission protocols via these I/O interfaces. Both channels can be configured for EIA RS-232-C and TTL; and only one can be used for opto-isolated 20mA current loop: The 8251 USART (Universal Synchronous/Asynchronous Receiver/Transmitter) lets the user select the data format using programming techniques with the 8253 Programmable Timer providing the transmit and receive clock (BAUD rate) for the USART.

5.2 CONFIGURING THE SERIAL I/O PORT

Normally, the MSC 8009 will be shipped configured per customer's requirement. In the event that the user requires another configuration, the following kits can be ordered.

301-0072-016	EIA RS-232-C
301-0072-017	20 mA current loop
301-0072-018	TTL

These kits have all the necessary components that are needed for the designated configuration for the 75188 power jumper requirement and component placement, refer to Table 5-3.

5.2.1 Terminal/Communication Configuration

Interfacing information for connecting the MSC 8009 to Data Communication Equipment (DCE) and Data Terminal Equipment (DTE) is given in Tables 5-1 and 5-2 respectively.

5.2.2 Programmable Timer Configuration

The 8253 Programmable Timer provides the MSC 8009 with three 16-bit timers, each programmed for a specific mode of operation. Each counter has separate mode configurations and counting operations -- binary or BCD.

MSC 8009 EDGE CONNECTORS		RS-232 Standard DB-255 (Female)	
PIN NO.	J2/3 SIGNALS	PIN NO.	RS-232-C SIGNAL
1	NC		
2*	+Rx20MA	14	SBA-Secondary Transmitted
3	RS-232-Rx	3	3 BB Received Data
4	RS-232 CLK OUT	15	DB-Transmission Signal Element Timing
5	RS-232-TxD	2	BA-Transmitted Data
6*	+Tx20MA	16	SBB-Secondary Received Data
7	RS-232 CTS	5	CB-Clear to Send
8	RS-232 CLK IN	17	DO-Received Signal Element Timing
9	RS-232 RTS	4	CA-Request to Send
10*	-Rx20MA	18	Unassigned
11	RS-232 DTR	20	CD-Data Terminal Ready
12	NC		
13	Signal Ground	7	AB-Signal Ground
14	*RS-232 DSR	6	CC-Data Set Ready
15	ON	8	CF-Received Line Signal Detector
16-20	NC		
21*	-Tx20MA	11	Unassigned
22-23	NC		
24*	-RDCTL	13	SCB-Secondary Clear To Send
25*	+RDCTL	25	Unassigned
26	NC		

1) Pins marked with an astrisk are for current-loop operation only. When the I/O port is configured for RS-232-C, they are not connected. Also, these pins are not connected on edge-connector J2.

2) Pin 14 can be used as the RS-232-C "DB Transmission Signal Element Timing" depending on the jumper configuration.

Table 5-1
SERIAL I/O CABLE CONNECTION FOR
DATA COMMUNICATIONS EQUIPMENT (DCE)

MSC 8009 EDGE CONNECTORS		RS-232 Standard DB-255 (Female)	
PIN NO.	J2/3 SIGNALS	PIN NO.	RS-232-C SIGNAL
1	NC		
2*	+Rx20MA	14	SBA-Secondary Transmitted Data
3	RS-232 RXD	2	BA-Transmitted Data
4	RS-232 CLK OUT	15	DB-Transmission Signal Element Timing
5	RS-232 TxD	3	BB-Received Data
6*	+Tx20MA	16	SBB-Secondary Received Data
7	RS-232 CTS	4	CA-Request to Send
8	RS-232 CLK IN	17	DO-Received Signal Element Timing
9	RS-232 RTS	5	CB-Clear to Send
10*	-Rx20MA	18	Unassigned
11	RS-232 DTR	6	CC-Data Set Ready
12	NC		
13	Signal Ground	7	AB-Signal Ground
14	*RS-232 DSR	20	CD-Data Terminal Ready
15	ON	8	CF-Received Line Signal Detector
16-20	NC		
21*	-Tx20MA	11	Unassigned
22-23	NC		
24*	-RDCTL	13	SCB-Secondary Clear To Send
25*	+RDCTL	25	Unassigned
26	NC		

1) Pins marked with an astrisk are for current-loop operation only. When the I/O port is configured for RS-232-C, they are not connected. Also, these pins are not connected on edge-connector J2.

2) Pin 14 can be used as the RS-232-C "DB Transmission Signal Element Timing" depending on the jumper configuration.

3) This configuration may be implemented with mass termination connectors and flat ribbon cable.

Table 5-2
SERIAL I/O CABLE CONNECTION FOR
DATA COMMUNICATIONS EQUIPMENT (DTE)

COMPONENTS			
COMPONENT	RS-232-C	CURRENT LOOP	TTL
U9A/B	75189	*	75189
U10A/B	75188	*	74LS00
U11	*	74LS04	*
U12	*	4N33	*
U13	*	4N33	*
U14	*	4N33	*
75188 POWER JUMPERS			
PINS	RS-232-C	CURRENT LOOP	TTL
43,44,45	43-44	N/A	44-45
45,46,47	46-47	N/A	45-46

NOTE: 1. Each serial I/O port has a set of pins (A and B).
2. For detail of other jumpers, refer to either specific paragraph or Appendix B of this manual.

Table 5-3
SERIAL I/O PORT CONFIGURATION

Each counter has a set of three signal lines (See Table 5-3) -- CLK INPUT (clock), GATE INPUT and OUTPUT. CLK INPUT sets up the counting rate (2 MHz maximum); and GATE INPUT enables the counting action, which generates the resultant OUTPUT signal. The use of GATE INPUT depends upon the mode (0 thru 5) selected by software.

COUNTER	FUNCTION	SIGNAL
0	Floppy Disk Clock	OUT 0
0	Input	GATE 0
0	Clock	CLK 0
1	Port #2 Clock	OUT 1
1	Input	GATE 1
1	Clock	CLK 1
2	Port #1 Clock	OUT 2
2	Input	GATE 2
2	Clock	CLK 2

Table 5-4
PROGRAMMABLE TIMER SIGNALS

5.2.2.1 BAUD Rate Configuration

Counter 2 of the 8253 (U17) supplies the transmit and receive clock for each 8251 USART of the MSC 8009. Normally, a 1 MHz clock (jumper 48 to 49) is used as the "I/O CLK". For 2 MHz operation, jumper 48 to 49 is removed; and a jumper is inserted between 50 and 51. The MSC 8009 is capable of 9600-baud asynchronous operation.

NOTE: All configurations (EIA, TTL and current loop) require either jumper 48 to 49 or 50 to 51, but not both for the I/O clock.

5.2.3 Clock Configurations

The following table lists available clocks and required jumpers.

JUMPERS	CLOCK
42A to 40A*	Programmable Timer (8253)
42A to 41A	RS-232 TXCLK OUT (Port #1)
39A to 40A	RS-232 RXCLK IN (Port #1)
97A to 98A	DSR/=TXC (Port #1)
98A to 99A*	TXC=RXC (Port #1)
42B to 41B	TXCLK OUT (Port #2)
39B to 40B	RXCLK IN (Port #2)
97B to 98B	DSR/=TXC (Port #2)
98B to 99B	TXC=RXC (Port #2)

*Standard MSC 8009 Configuration

To use the Programmable Timer as the clock, insert jumper 40 (A,B depending on Port) to 42 for USART operations, and jumper 41 to 42 makes it available for external applications. If the USART operation requires an external clock, insert jumper 39 to 40. Jumper 98 to 99 permits the same clock to be used for both transmitting and receiving functions. Jumper 97 to 98 allows RS-232-C DSR for transmit clock (TXC).

5.2.3 EIA RS-232-C Configuration

Both ports can be set up for EIA RS-232-C operation. In this configuration, all signals fulfill the EIA specifications, including clock signals. However, care must be taken when installing the jumpers that involve power (+5V and +12V). To configure the MSC 8009 for the RS-232-C convention, it requires the installation of a 75189 (U9A or B depending on the port) and a 75188 (U10A or B depending on the port) in addition to the following jumpers:

JUMPERS	CLOCK
40A to 42A	Port 1 Internal Clock
40B to 42B	Port 2 Internal Clock
43A to 44A	-12V for U10A
43B to 44B	-12V for U10B
46A to 47A	+12V for U10A
46B to 47B	+12V for U10B
96A to 97A	Port 1 Data Set Ready (DSR/)
96B to 97B	Port 2 Data Set Ready (DSR/)
98A to 99A	TXC=RXC (Port 1)
98B to 99B	TXC=RXC (Port 2)

5.2.4 TTL Configuration

Replacing the 75188 devices with 75LS00 lets the MSC 8009 accept TTL compatible signals. Care must be taken when inserting jumpers that are associated with power. If improperly installed, power application could inflict damage to the drivers. The TTL configuration requires the following jumpers:

JUMPERS	CLOCK
40A to 42A	Port 1 Internal Clock
40B to 42B	Port 2 Internal Clock
44A to 45A	+5V input to U10A
44B to 45B	+5V input to U10B
45A to 46A	+5V power for U10A
45B to 46B	+5V power for U10B
96A to 97A	Port 1 Data Set Ready (DSR/)
96B to 97B	Port 2 Data Set Ready (DSR/)
98A to 99A	TXC=RXC (Port 1)
98B to 99B	TXC=RXC (Port 2)

5.2.5 Current Loop Operation

Only Port 1 has Current Loop capabilities. When configuring Port 1 for current loop operation, correct signal polarity must be observed. This configuration requires three 4N33 Darlington Optical Isolators in U12 thru U14, and a 74LS04 Inverter in U11 as well as jumper 98A to 99A (TXC=RXC).

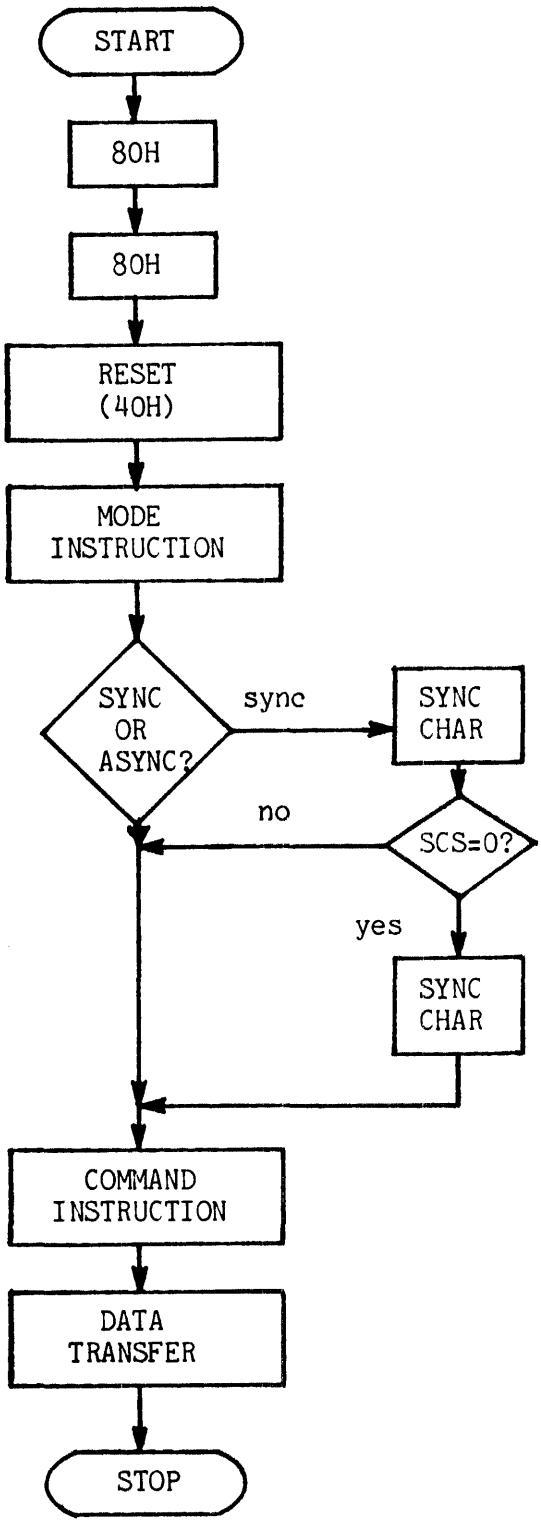
NOTE: Only the six signals applied to the opto-isolators are available on one connector (J3). The internal BAUD rate clock is not available and external BAUD rate to the 8251 is also not available.

Maximum open-circuit voltage for the 4N33 is 30V DC with less than 40 mA closed-circuit current. The external device must supply the current source for operating in current loop mode. An isolated device should not be connected to ground on the computer board.

5.2.6 Interrupt Configuration

Pins 3 (A,B) and 4 (A,B) make the inverse of USART-generated signals TXRDY and RXRDY available for an interrupt operation. An open-collector driver lets the user jumper these signals into the MULTIBUS interrupt system using pins 72 thru 87. The odd-number pins assert these signals onto the MULTIBUS; and the even-numbers go to the Z80 processor. Since the drivers are open-collector, a number of interrupts can be handled on the same level.

Flow Diagram:



Source Program:

```
LDI     A,80H       ;80H IN ACCUMULATOR
OUT     0EDH        ;SEND OUT 1st 80H
XTHL                ;TIME DELAY
XTHL
OUT     0EDH        ;SEND OUT 2nd 80H
XTHL                ;TIME DELAY
XTHL
LDI     A,40H       ;RESET COMMAND
OUT     0EDH        ;RESET 8251
XTHL                ;TIME DELAY
XTHL
 .       .           .
 .       .           .
Select Mode Now (Assume Standard
Asynchronous)
 .       .           .
 .       .           .
LDI     A,0CDH      ;SELECT MODE
OUT     0EDH        ;SEND OUT MODE INSTRUCTION
XTHL                ;TIME DELAY
XTHL
 .       .           .
 .       .           .
If synchronous mode is selected,
insert SYNC character definition
at this time
 .       .           .
 .       .           .
LDI     A,037H      ;COMMAND INSTRUCTION
OUT     0EDH        ;SEND OUT COMMAND INSTRUCTION
XTHL                ;TIME DELAY
XTHL
 .       .           .
 .       .           .
MSC 8009 is ready to send or
receive data
```

Figure 5-1
CONTROL WORD SEQUENCE

5.3 PROGRAMMING THE SERIAL I/O INTERFACE

The ensuing paragraphs give the recommended procedure for programming the serial I/O interface of the MSC 8009. The sequence of operation is illustrated in Figure 5-1 -- assuming that the hardware assuming that the hardware configuration is assembled as described in the preceding paragraphs.

5.3.1 Initialization

On power up, the MSC 8009 system-reset signal INIT forces the 8251 USART into an "idle" condition. Loading the Control Word register with three consecutive zero instructions, or two consecutive 80 H, followed by one 40 H command instructions via software will also reset USART (Refer to Figure 5-1). Whenever setting or changing modes, the software approach is recommended. Succeeding the initialization sequence, the next control-word entry must be a Mode instruction. The following table lists the standard port addresses for selecting the USART functions.

PORT	ADDRESS	READ	WRITE	USART
1	ED,EF	Status	Control or Mode	U19A
1	EC,EE	Data In	Data Out	U19A
2	CF	Status	Control or Mode	U19B
2	CE	Data In	Data Out	U19B

5.3.2 Clock Set

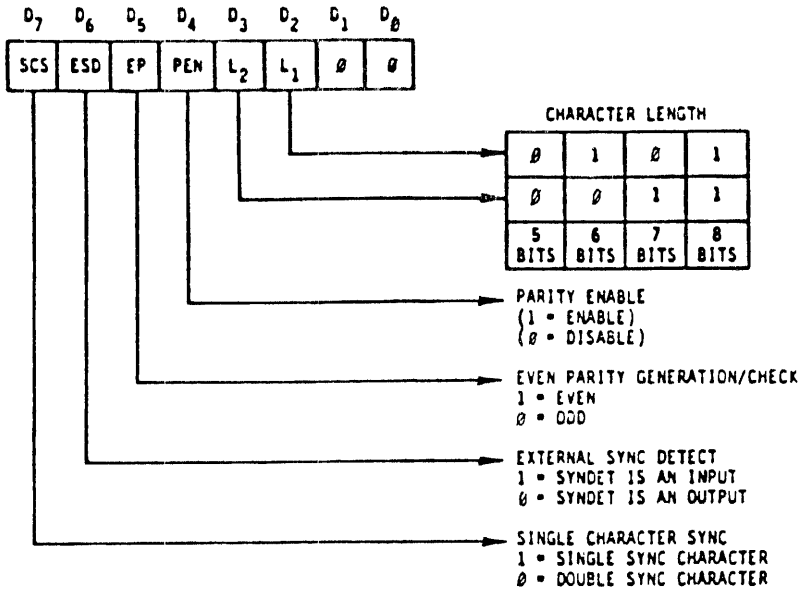
The BAUD rate should be set before the 8251 control word. Paragraphs 5.2.1.1 and 5.5.3 provide more detail in BAUD rate generation.

5.3.3 Control Word Programming

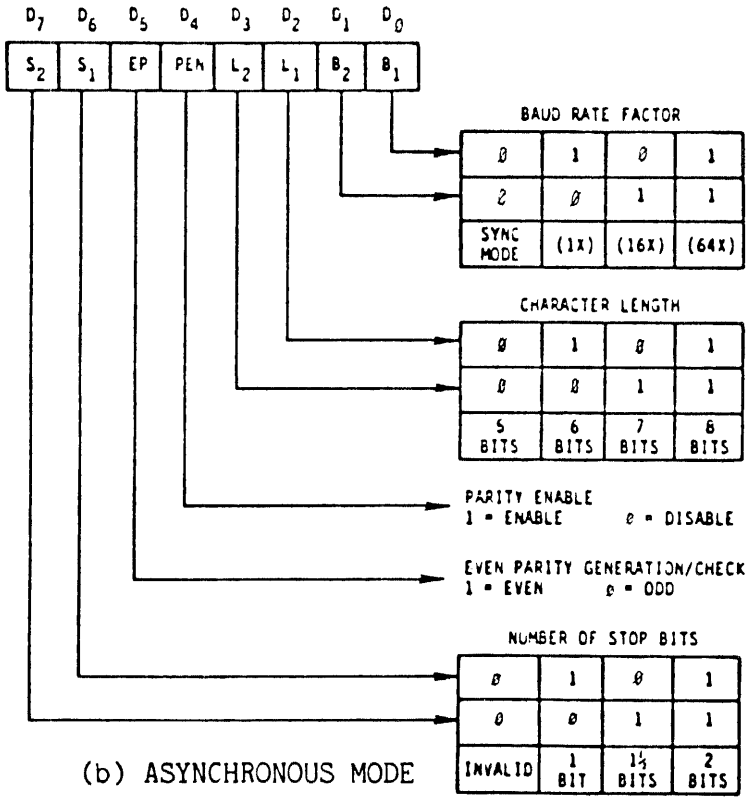
A set of control words, which are sent to the USART, define the mode and communication format. These formats are:

1 MODE INSTRUCTION
2 COMMAND INSTRUCTION

The Mode Instruction is loaded first. If a synchronous mode of operation is to be executed, one or two SYNC-character control word(s) must follow the Mode Instruction (See Figure 5-1). The last control word is a Command Instruction.



(a) SYNCHRONOUS MODE



(b) ASYNCHRONOUS MODE

Figure 5-2
MODE INSTRUCTION CONTROL WORD FORMAT

5.3.3.1 Mode Instruction

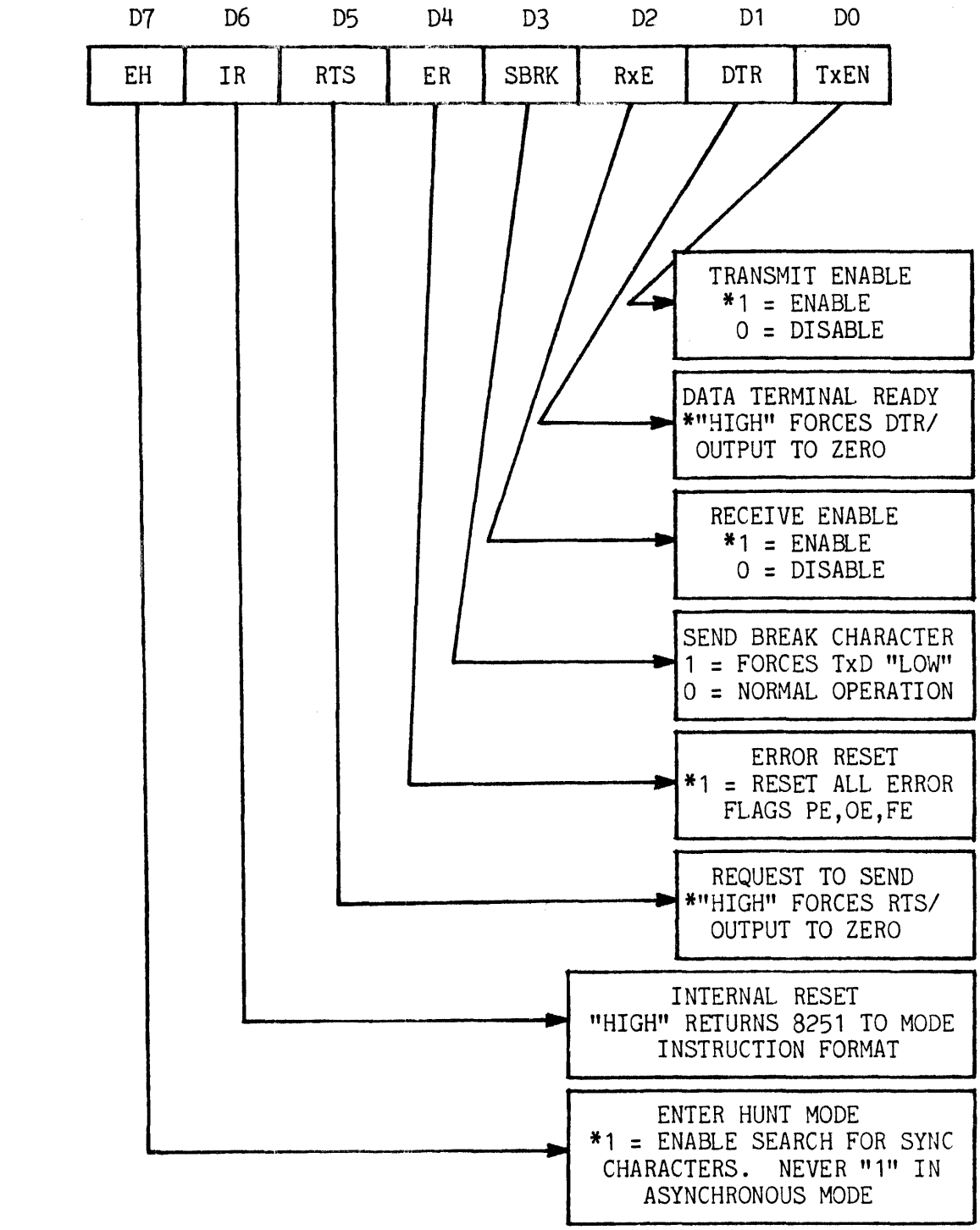
The Mode Instruction, shown in Figure 5-2, defines the operational characteristics of the serial I/O interface. These characteristics include synchronous or asynchronous operation, BAUD rate factor, character length, parity, number of stop-bits and number of SYNC characters. A Mode Instruction must follow a reset operation. Once the Mode Instruction has been received, SYNC characters or Command Instructions must then be inserted, depending on the contents of the Mode Instruction.

5.3.3.2 Command Instruction

Once the Mode Instruction has defined the operational characteristics and SYNC character(s) have been loaded (if in SYNC mode), the 8251 USART is now ready to receive the Command Instruction and begin data communications. A Command Instruction (See Figure 5-3) controls the specific operation that the Mode Instruction designates. These operations are:

1) ENABLE TRANSMIT (TxEN)
2) ENABLE RECEIVE (RxEN)
3) ERROR RESET (ER)
4) MODEM CONTROL

If the preceeding Mode Instruction is in a synchronous format, the Command Instruction will be interpreted as a SYNC character. Following the SYNC character(s) or Asynchronous Mode Instruction, subsequent control words will then be considered as an update to the Command Instruction. A Command Instruction may occur any time prior to reset during the data block. To guarantee that the command is acceptable, it is recommended that two 80H commands be sent. Bit 6 (40H) is then set in the Command Instruction to modify the Mode Instruction. This bit initiates an internal reset that conditions the USART for receiving a new Mode Instruction. Figure 5-1 shows this reset procedure and Control Word sequence. With I/O clock at 1 MHz and CPU running at 4 MHz, it is possible for the CPU to return and access the 8251 before the bus completes the previous operation. Therefore, a short delay is recommended between each command (pair of XTHL).



*Normally "1"

Figure 5-3
COMMAND INSTRUCTION CONTROL WORD FORMAT

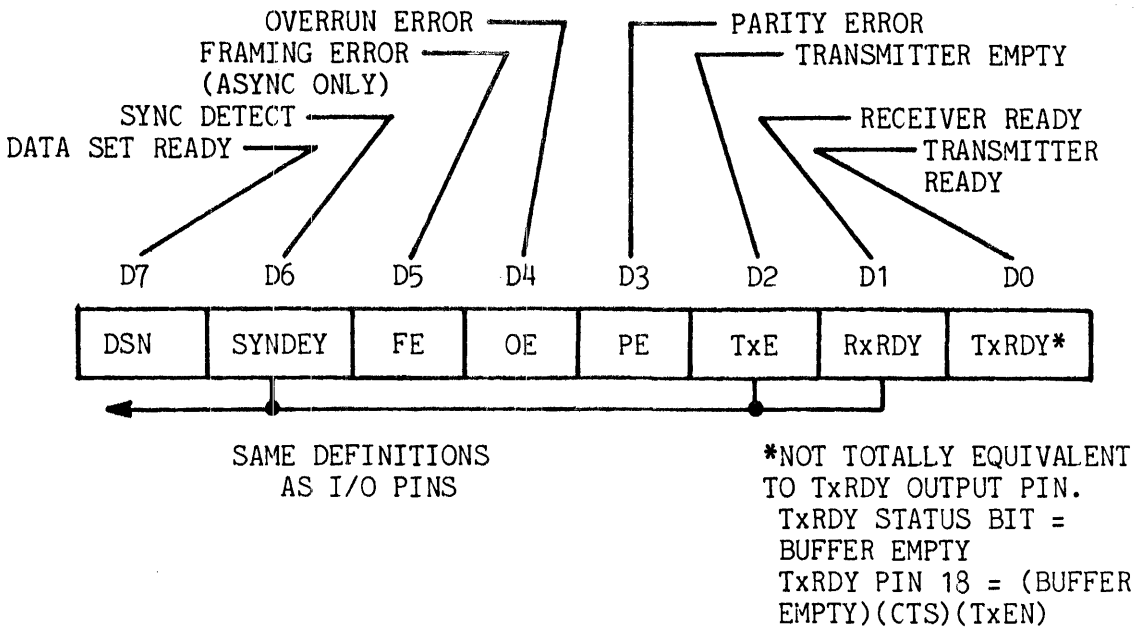


Figure 5-4
STATUS WORD FORMAT

5.3.4 Status Word Format

Frequently, the Z80 must be aware of any errors or other conditions that require a CPU response. The 8251 USART allows the CPU to read the device status at anytime. Since many of the Status register bits (See Figure 5-4) have the same meanings as the external output pins, the USART can be used in both polling and interrupt environments. To examine the Status register, the CPU issues a read command and places a "high" on the C/D input.

NOTE: Status update can have a maximum delay of 16 clock periods. Therefore, a software delay should be inserted before attempting to read the status register after any operation that affects it.

5.3.4.1 Parity Error

A parity error sets bit 3 (PE Flag), which does not inhibit USART operation. The setting of bit 4 or the ER bit of a subsequent Command Instruction clears the PE Flag.

5.3.4.2 Overrun Error

The OE Flag (bit 4) denotes that the processor failed to read a data character prior to availability of the succeeding bit. Although the setting of the OE Flag does not inhibit USART operation, the previously received character is overwritten and lost. The ER bit (bit 4) of a subsequent Command Instruction resets this bit.

5.3.4.3 Framing Error

If a valid STOP bit is not detected at the end of a character, the FE Flag or bit 5 is set. This action does not inhibit USART functions. The ER bit (bit 4) of a subsequent Command Instruction resets the FE Flag.

NOTE: Asynchronous mode only.

5.4 DATA COMMUNICATION

The 8251 USART provides the MSC 8009 with a serial I/O interface for either synchronous or asynchronous data communications.

5.4.1 Asynchronous Transmission

Before a character is placed onto the I/O bus, the 8251 USART sets TxRDY to signal the CPU that the USART is ready to accept information for transmission. This ready signal (TxRDY) is automatically reset when the Z80 sends a character to the USART. Also, the USART adds the START bit and the requested number of STOP bits (bits 6 and 7 of the Mode Instruction) to each character. If there is a request for parity consideration (bits 4 and 5 of the Mode Instruction), these bits are then inserted before the STOP bits. The USART asserts the character in a serial format onto the TxD line at the programmed BAUD rate. WHen the CPU stops sending data to the USART, TxD output will remain "high" unless a BREAK (continuous "low") has been programmed.

5.4.2 Asynchronous Receive

A high-to-low transition on the RxD input line triggers a START bit operation, and the 8251 automatically confirms the validity of the START bit. If the bit is valid, the bit counter begins to count the received bits. This counter defines bit locations so that error conditions can be flagged in the Status register. Receipt of the STOP bit transfers the serial data into a parallel I/O register and sets the RxRDY signal. Now the CPU can read the data.

5.4.3 Synchronous Transmission

As in the asynchronous transmission, the TxD output remains "high" (marking) until the CPU sends the first character -- usually a SYNC character. After a Command Instruction sets TxEN and CTS/ goes "low", the first character is transmitted serially. The rising edge of TxC (clock) shifts the data out on TxD at the programmed rate.

Once transmission has begun, the serial-data must continue at TxC (clock) rate to maintain synchronization. If the 8251 does not receive data before the Transmitter Buffer becomes empty, the SYNC character(s) that were loaded after the Mode Instruction (Refer to Figure 5-1) are automatically inserted into the data stream. These inserted Sync characters maintain synchronization until the CPU sends new data for transmission. If SYNC character(s) must be transmitted, the USART asserts TxEMPTY at the center of the list data bit (See Figure 5-5) to notify the CPU that the Transmission Buffer is empty and SYNC characters are being transmitted. The next character from the CPU automatically resets TxEMPTY.

5.4.4 Synchronous Receive

For synchronous reception, a Sync mode must be programmed, and the ENTER HUNT bit (bit 7) of the Command Instruction set. These conditions cause the receiver circuit of the 8251 to enter a hunt mode -- looking for character synchronization.

Incoming data on the RxD input is sampled on the leading edge of RxC/ and the resultant stored in the Receiver Buffer. The Receiver Buffer content is then compared with the first SYNC character after each bit has been loaded until a match is found. If two SYNC characters are programmed, the next character received is also compared. When the SYNC character(s) that have been programmed are detected, the 8251 exits the HUNT mode and goes into character synchronization. In event that synchronization is lost, the CPU may command the receiver to enter the HUNT mode with a Command Instruction that has bit 7 set (ENTER HUNT bit).

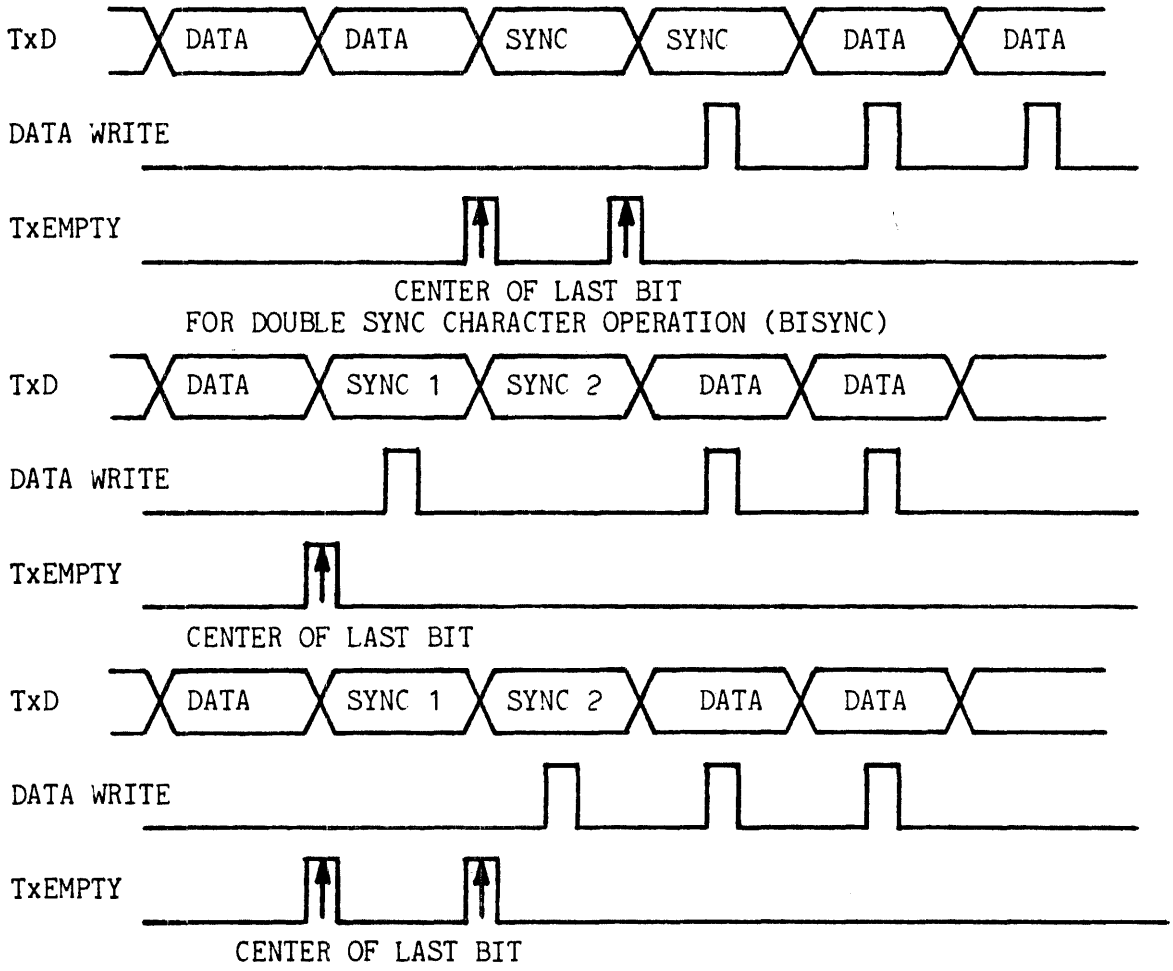


Figure 5-5
SYNC CHARACTER TRANSMISSION

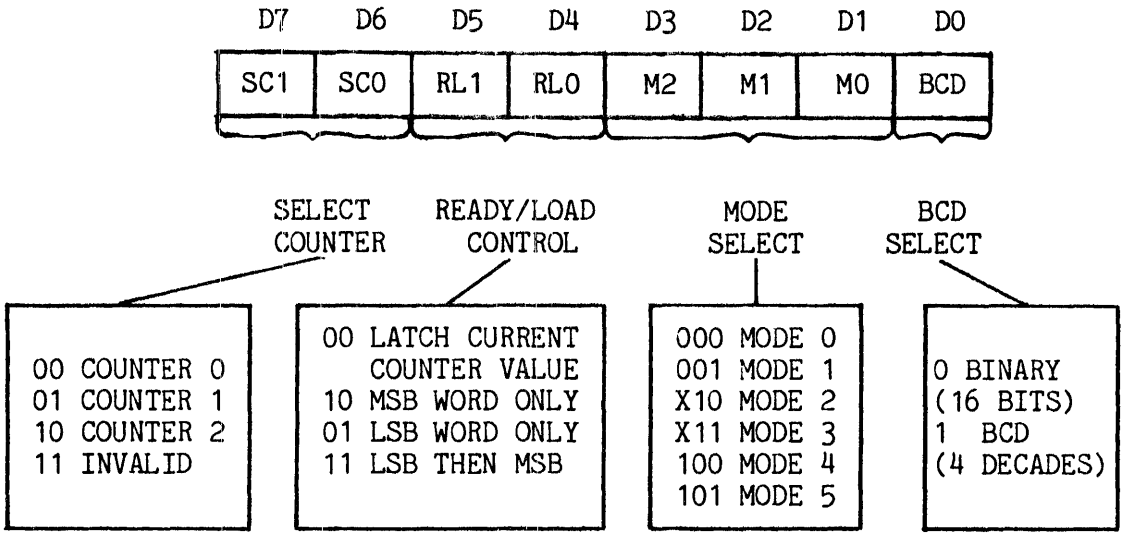


Figure 5-6
8253 INTERVAL TIMER CONTROL WORD FORMAT

5.5 TIMER INTERFACE

The 8253 Programmable Timer does not require a hardware or software reset. To program the Timer, it requires a single Control Word that is followed by one or two register counter words. The Control Word (See Figure 5-6) specifies the counter, the number of words that are to be read or loaded as well as in what order, operational mode of the selected counter and counting operation -- BCD or binary.

The status of this device is not available for examination. The standard port addresses are shown in Table 5-5. Each 16-bit counter register can be referenced as two sequential 8-bit word

READ	WRITE	ADDRESS
Read 0	Load 0	DC
Read 1	Load 1	DD
Read 2	Load 2	DE
Invalid	Control	DF

Table 5-5
8253 TIMER PORT ADDRESSES

5.5.1 Mode Definitions

The processor loads the appropriate Control Word (See Figure 5-6) into the Control Word register (for each counter separately) to establish the mode of operation. The three counters are identical The modes of operation are:

MODE 0 - Output (Interrupt) on the count 0
MODE 1 - Programmable/Retriggerable One Shot
MODE 2 - Pulse Rate Generator
MODE 3 - Square Wave Generator (BAUD rate)
MODE 4 - Software-Triggered Strobe
MODE 5 - Hardware-Triggered Strobe

For the timing relationship of each mode, refer to Figures 5-7 thru 5-12.

NOTE: 1) All references to "high" and "low" signals in the following paragraphs refer to the level at the chip. Refer to Drawing 305-0261-000, Sheet 9 for information regarding inverting buffers.

2) The WR pulse is the write strobe for a command or data byte to be latched into the timer. When two bytes are to be transferred the operation is finished after the second WR pulse.

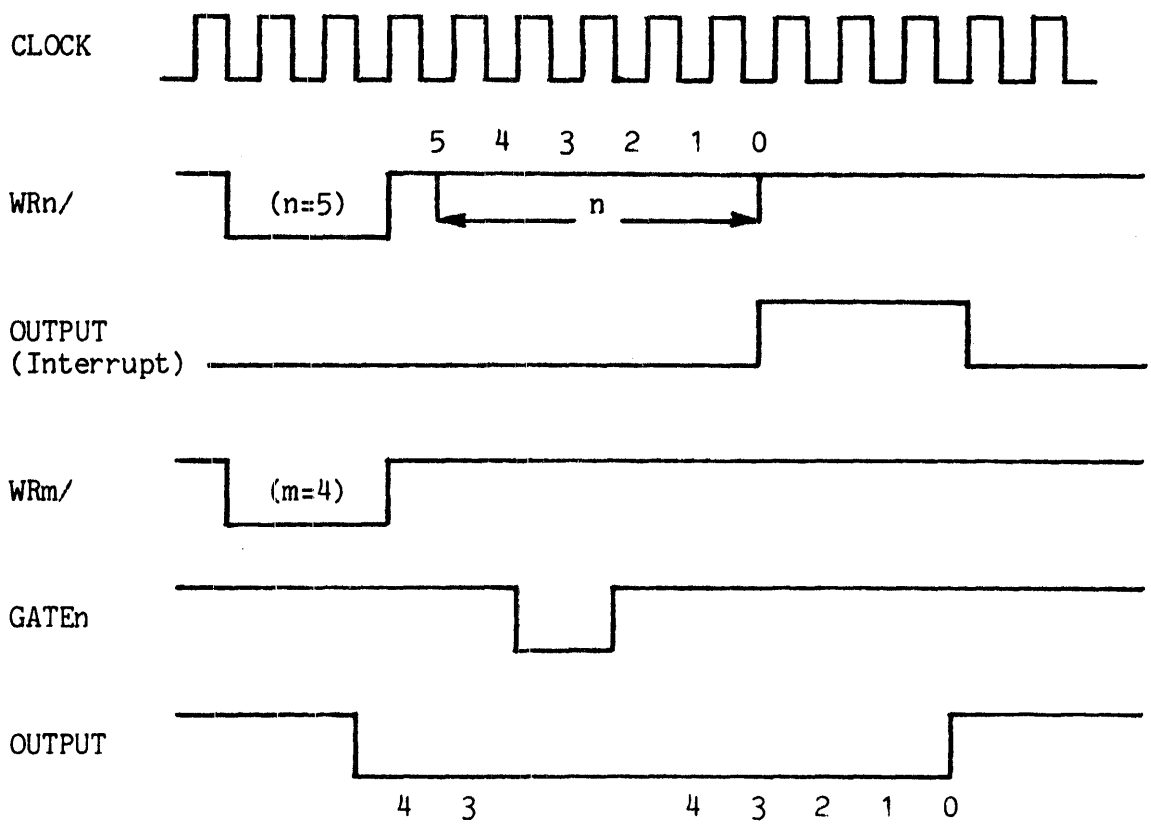


Figure 5-7
MODE 0 TIMING DIAGRAM

5.5.1.1 MODE 0 - Interrupt On Terminal Count

The initial MODE SET operation forces the output "low". When a counter is loaded with a count value, it begins counting. The output remains "low" until the terminal count sets it "high". It remains in the "high" state until the trailing edge of the second WR pulse loads in new count data. If the data is loaded during the counting process, the first WR stops the count. Counting starts with the falling clock edge after the second WR triggering the new count data. If GATE is asserted while counting. the count is terminated for the duration of GATE. The falling edge of CLK following the removal of GATE Restarts counting from the full count value. See Figure 5-7 for the timing of MODE 0.

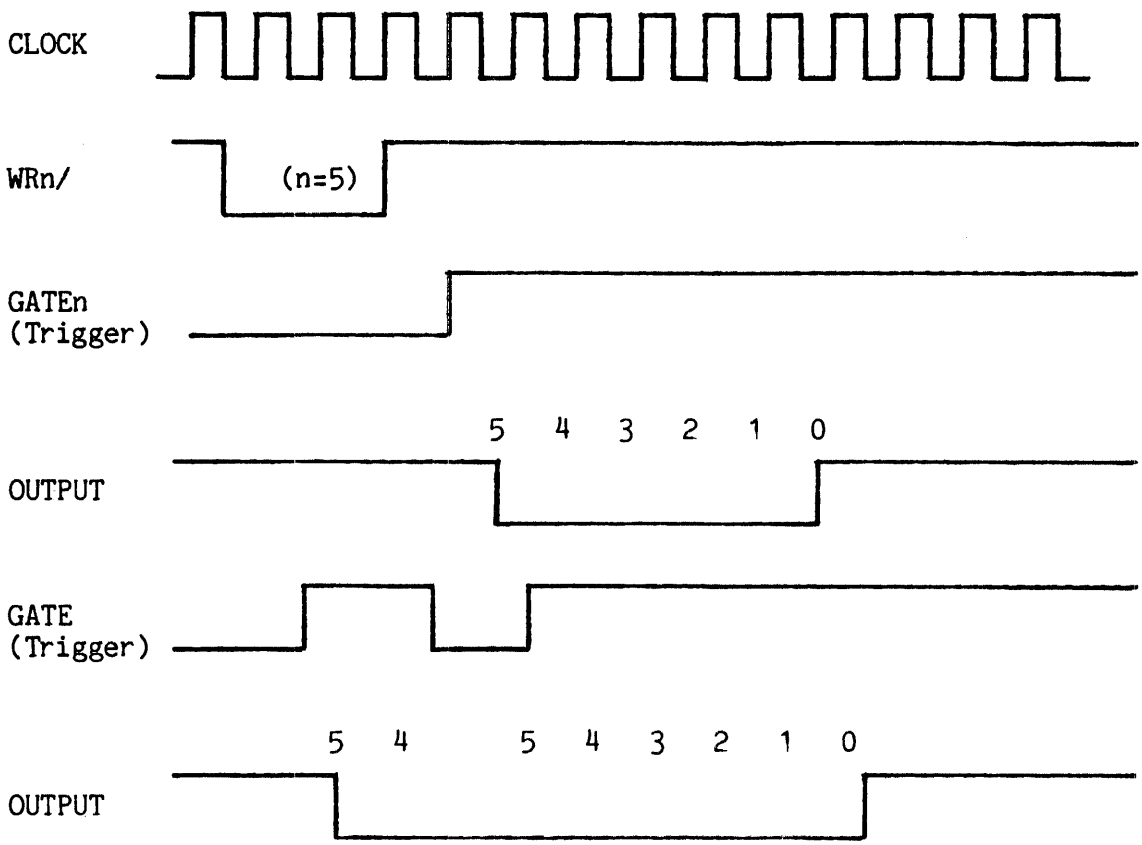


Figure 5-8
MODE 1 TIMING DIAGRAM

5.5.1.2 MODE 1 - Programmable One Shot

The trailing edge of CLOCK following the rising edge of GATE sets the output "low" (See Figure 5-8). The output is set "high" at the terminal count. The output pulse is not affected if new data is loaded while the one-shot is running. The assertion of a trigger pulse while the one-shot is running, resets and retriggers the one-shot The output will remain "low" for the full count value after the leading edge of TRIGGER.

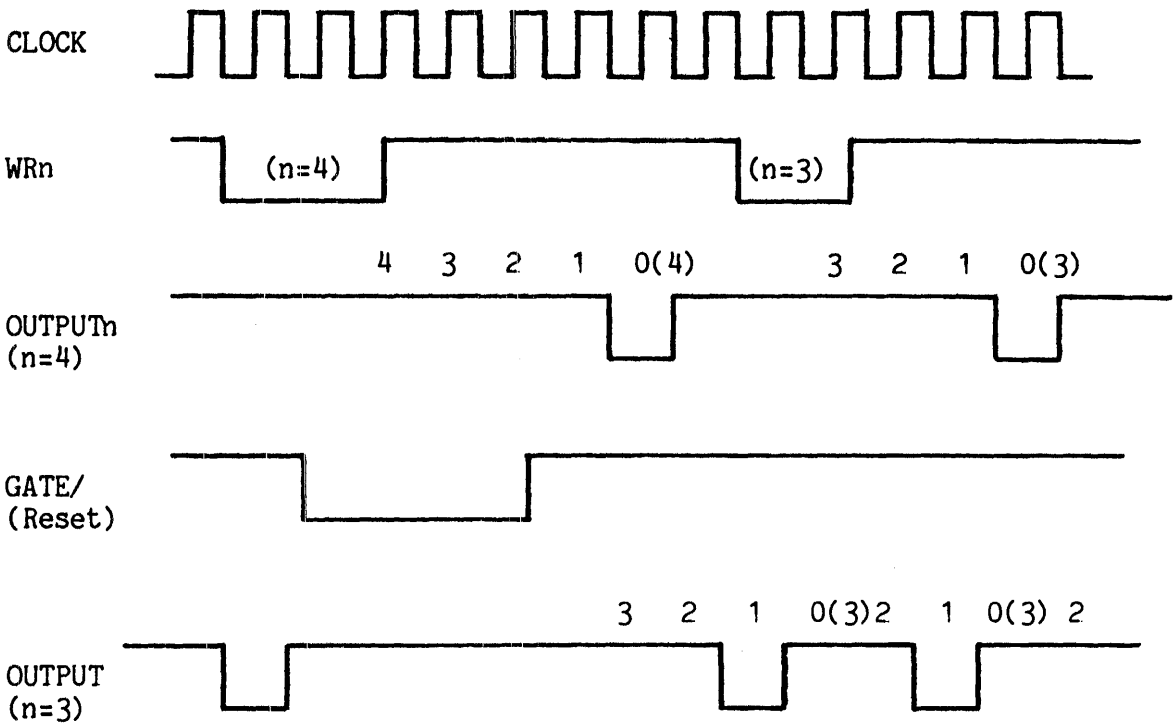


Figure 5-9
MODE 2 TIMING DIAGRAM

5.5.1.3 MODE 2 - Rate Generator

The Rate Generator is a variable modulus counter. The output goes "low" for one, full clock period as shown in Figure 5-9. The count data sets the time between output pulses. Changes in count data are reflected in the output as soon as the new data has been loaded into the count registers. The output remains "high" for the duration of a "low" GATE input. Normal operation resumes on the falling edge of CLOCK following the rising edge of GATE.

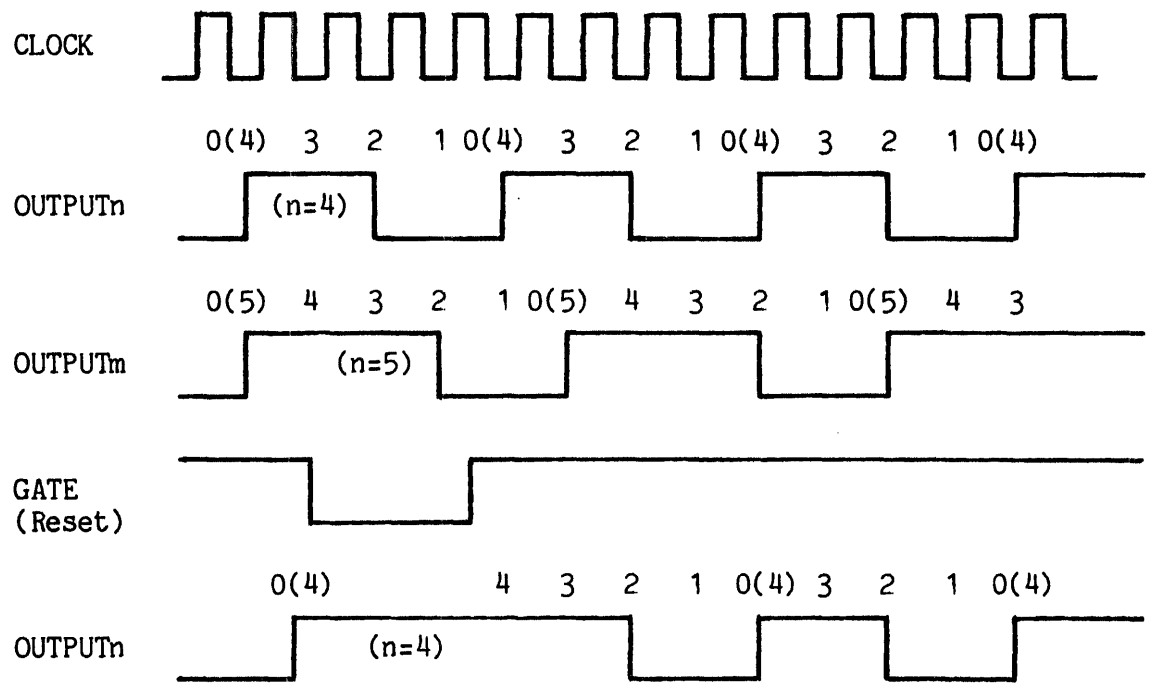


Figure 5-10
MODE 3 TIMING DIAGRAM

5.5.1.4 MODE 3 - Square Wave Generator

MODE 3 resembles MODE 2 except the output is "high" for half of the count and "low" for the other half (for even values of data). For odd values of count data, the output is "high" one clock cycle longer than when it is "low". In other words, the "high" period is for N+1 or (N+1)/2 clock cycles and the "low" period is for N-1 or (N-1)/2 clock periods, when N is the decimal value of the count data (Refer to Figure 5-10). Changes in count data are reflected in the output as soon as the new data has been loaded into the count registers. The output remains "high" during the loading of new count data. Counting resumes with that data after the second WR.

The output will be held in the "high" state while GATE (RESET) is "low". Counting starts from the full count value after GATE rises.

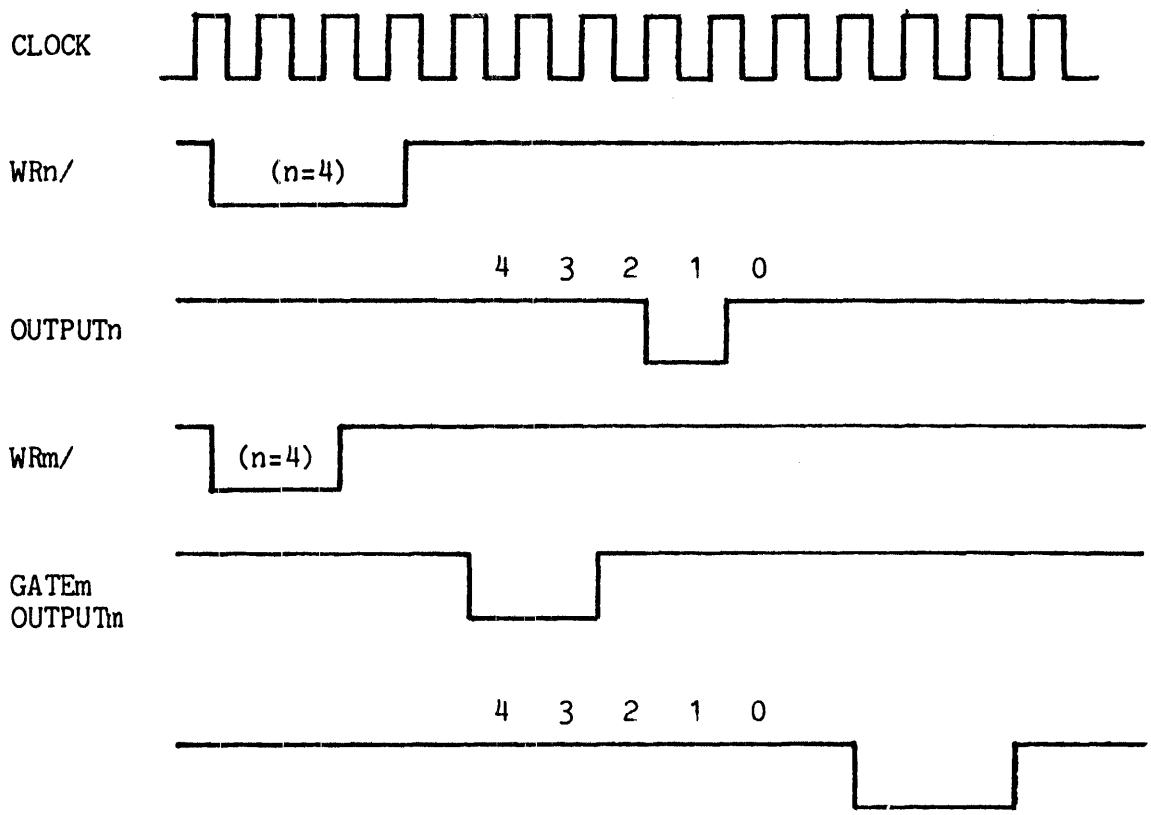


Figure 5-11
MODE 4 TIMING DIAGRAM

5.5.1.5 MODE 4 - Software Triggered Strobe

The output goes "high" when MODE 4 is set, and counting begins after the second byte of data has been loaded. When the terminal count is reached, the output goes "low" for one clock period (See Figure 5-11). Changes in count data are reflected in the output as soon as the new data has been loaded into the count registers. During the loading of new data, the output is held "high" and counting is inhibited. The output is held "high" for the duration of GATE. The counters are reset and counting begins from the full data value after GATE is removed.

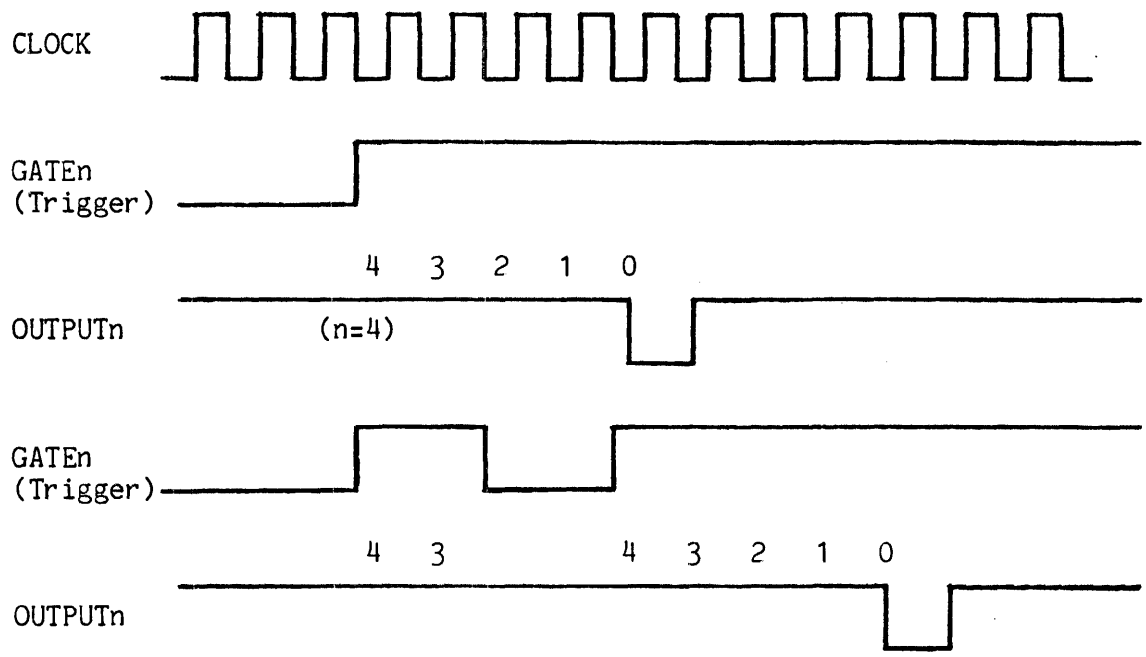


Figure 5-12
MODE 5 TIMING DIAGRAM

5.5.1.6 MODE 5 - Hardware Trigger Strobe

Loading MODE 5 sets the output "high". Counting begins when count data is loaded and GATE goes "high". After terminal count is reached, the output goes "low" for one clock period. Subsequent trigger pulses restart the counting sequence with the output pulsing "low" on terminal count following the last leading edge of the trigger input. See Figure 5-12 for the timing diagram of MODE 5 operation.

5.5.2 On-The-Fly Readout

Bits RL1 and RL0 of the Control Word (Refer to Figure 5-6) can be employed to control a status read of any selected counter without interrupting the counting sequence. If RL1 and RL0 are anything other than "00", the least-significant bit (LSB) or the most significant bit MSB or both can be read on-the-fly. However, the validity of the reading cannot be guaranteed unless the counter operation is inhibited via the GATE or CLK input prior to the attempted read operation. If RL1 and RL0 are "00", a counter can be reliably read on-the-fly because the current value is retained in an internal storage register before being read out.

5.5.3 BAUD Rate Generator

Counter 2 of the 8253 Programmable Timer generates both the transmit and receive clocks (TxC and RxC respectively) for the USART. Therefore, Counter 2 operates in MODE 3 (Square-Wave Generator), and the counter is loaded with the count down values listed in Table 5-7. Figure 5-13 illustrates the flow diagram with source program for setting up the BAUD rate generator.

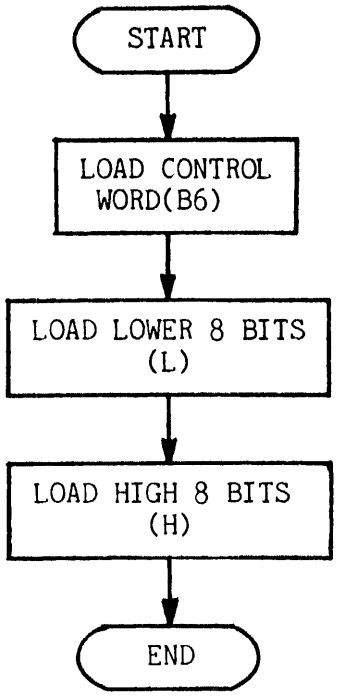
Either Monolithic System's MSC 8301 Uniform Monitor (2K bytes stand alone monitor) or MSC 8303 MSOS (6K byte disk monitor) includes a code that automatically sets the required BAUD rate for the user's terminal. Since the source code is available with these software packages, they provide useful exampes for the system designer.

BAUD RATE		COUNTER REGISTER 2	
1 MHz Clock	2 MHz Clock	Upper Byte (H)	Lower Byte (L)
4800	9600	00*	0D*
2400	4800	00	1A
1200	2400	00	34
600	1200	00	68
300	600	00	D0
150	300	01	A0
110	220	02	38
75	150	03	40
-	110	04	10

*Hex Notation

Table 5-6
8253 TIMER REGISTER BAUD RATE VALUES

Flow Diagram:



Source Program:

```
MVI    A,0B6H    ;LOAD CONTROL WORD B6H
OUT    0DFH      ;XFR TO COUNTER 2
MVI    A, xx     ;LOAD LOW 8-BITS OF BAUD RATE
OUT    0DEH      ;XFR TO COUNTER 2
MVI    A, xx     ;LOAD HIGH 8-BITS OF BAUD RATE
OUT    0DEH      ;XFR TO COUNTER 2
```

Figure 5-13
BAUD RATE GENERATOR ROUTINE

SECTION 6

FLOPPY-DISK FORMATTER/CONTROLLER

6.1 SCOPE

This section provides the information the user needs to understand and program the onboard floppy-disk controller of the MSC 8009.

6.2 DESCRIPTION

One chip gives the MSC 8009 a means for communicating with up to eight 5- or 8- inch floppy-disk drives. In addition, the Z80 block I/O instructions allow the user to quickly transfer data to and from the drive via the formatter/controller. Another instruction will not be accepted until the exchange of data is complete. To set up the MSC 8009 for floppy-disk operation, see Appendix C of this manual for the required jumper configuration.

6.3 CONTROL REGISTERS

There are six registers that provide status, temporary storage and control -- five internal to the formatter/controller chip and one external. These registers and addresses are:

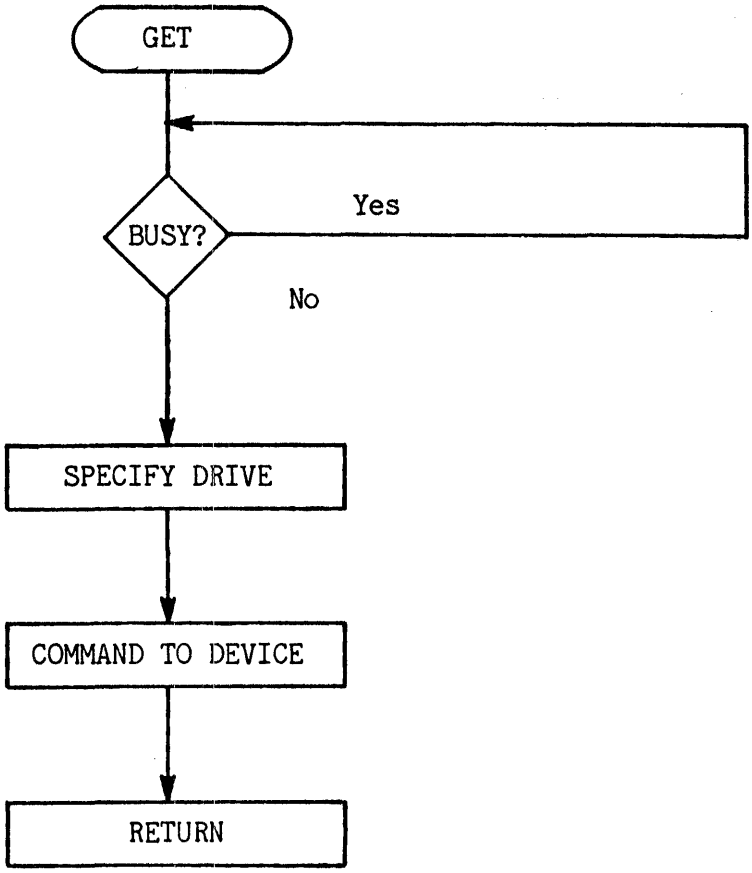
ADDRESS (HEX)	REGISTER	OPERATION Read	Write
C0	COMMAND		X
C0	STATUS	X	
C1	TRACK	X	X
C2	SECTOR	X	X
C3	DATA	X	X
C4	UNIT	X	X
C7	DATA (delay)	X	X

A read or write operation to I/O addresses 0C0H thru 0C4H should not be attempted during execution of a command. The only exception is the Status register (Refer to paragraph 6.3.1).

BIT	NAME	DESCRIPTION
7	COMP	A "1" enables write precompensation.
6	DRIVE SELECT	A "1" denotes a 5-inch drive; and a "0" specifies an 8-inch drive.
5	RECORDING DENSITY	A "0" selects single-density mode (FM); and a "1" signifies double-density (MFM).
4	HEAD SELECT	This line can be used to select the desired head.
3	DC MOTOR CONTROL	This bit can be used either with drives having separate DC motor control capability or as a unit select bit (UNIT 3).
2	UNIT 2	A "1" selects Drive 2.
1	UNIT 1	A "1" selects Drive 1.
0	UNIT 0	A "1" selects Drive 0.

Table 6-1
DRIVE DESIGNATION

Flow Diagram:



Source Program:

```
 1                      ;GET A COMMAND
 2                      ;ENTRY: NOTHING NEEDED
 3                      ;EXIT: COMMAND & UNIT READY
 4                      ;
 5   0000 CD 0F 00      GET:    CALL    STAT        ;DEVICE NOT BUSY
 6   0003 3A 01 00              LDA     01H         ;A=UNIT  DESIGNATION
 7   0006 D3 C4                 OUT     0C4H        ;SET UP DRIVE
 8   0008 2A 10 00              LHLD    COMM        ;COMMAND LOC.
 9   000B 7E                    MOV     A,M         ;GET COMMAND
10   000C D3 C0                 OUT     0C0H        ;TO DEVICE:
11   000E C9                    RET                 ;DONE
12   000F               STAT:   DS      1           ;STATUS ROUTINE
13   0010               COMM:   DS      1           ;COMMAND
```

Figure 6-1
COMMAND ROUTINE

6.3.1 Command Register

The execution of a command requires the loading of two 8-bit registers -- the UNIT register and the COMMAND register. One approach to calling a command is shown in Figure 6-1.

6.3.1.1 Unit Register

Drive requirements such as unit designation, head selection, recording density and write precompensation are entered into a register that is external to the formatter/controller chip. These drive specifications are sent on I/O address 0C4H. If a sequence of instructions are to be performed on one drive, the selected drive does not have to be set up for each instruction once it has been designated. For a description of each bit in this register, see Table 6-1.

6.3.1.2 Command Register

The instruction that is to be executed is held in a register onboard the formatter/controller chip. The commands are sent on I/O address 0CH0; and commands will be ignored if the chip is busy.

6.3.2 Status Register

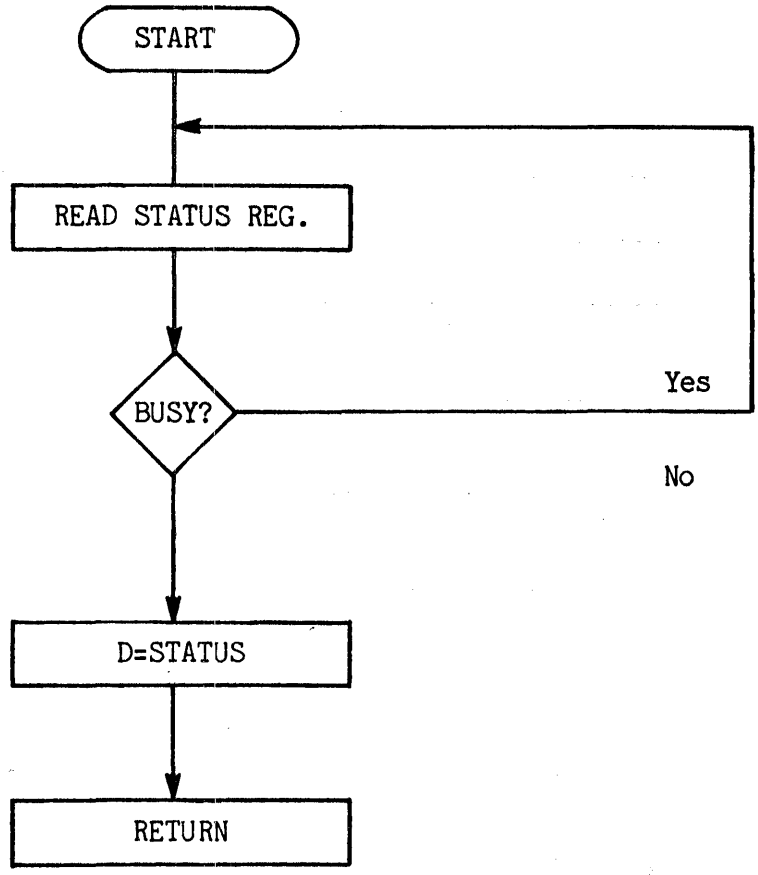
The status of a command just executed will be held in an 8-bit register called the STATUS register. Access (read only) to this register can be attempted any time.

Since Bit 0 or the Busy bit is a logic "1" during the execution of a command, this bit can be monitored to determine if a command is in process; and the other bits hold invalid information. Bits 1 thru 7 should be read only when the command is complete. At this time, the meaning of these bits will be a function of the previously executed command.

NOTE: If two sequential read operations of the Status register are attempted following the execution of a command, the value of the second bits read will be a valid Type 1 status (Refer to Table 6-2).

Figure 6-2 gives a sample routine that reads the Status register. For the definition of each bit, see Tables 6-2 (Head Position Status) and 6-3 (Read/Write Status).

Flow Diagram:



Source Program:

```
1                       ;READ STATUS REGISTER
2                       ;ENTRY: NOTHING NEEDED
3                       ;D HAS FINAL STATUS
4                       ;
5   0000 DB C0          STATUS: IN      0C0H        ;READ STAT
6   0002 57                     MOV     D,A         ;D=STATUS
7   0003 E6 80                  ANI     80H         ;BUSY?
8   0005 CA 00 00               JZ      STATUS
9   0008 C9                     RET                 ;DONE
```

Figure 6-2
STATUS REGISTER READ

BIT	NAME	DESCRIPTION
7	NOT READY	The drive is not ready if a "1".
6	WRITE PROTECT	If a "1", the Write Protect is activated.
5	HEAD LOADED	If a "1", the head is loaded and engaged.
4	SEEK ERROR	If a "1", the selected track was not verified. Reset when updated.
3	CRC ERROR	If a "1", the CRC byte was invalid.
2	TRACK 00	If a "1", the head is positioned at Track 00.
1	INDEX	The index mark from the drive is set if a "1".
0	BUSY	A command is in progress if a "1".

Table 6-2
HEAD POSITION STATUS

BIT	NAME	DESCRIPTION
7	NOT READY	A logic "1", indicates the drive is not ready.
6	WRITE PROTECT	For a write operation, a "1" indicates a Write protect condition. It will be "0" for a read operation. Reset when updated.
5	RECORD TYPE	For read, it indicates a record type code from the DATA FIELD address mark. For write, it will always be "0". Reset when updated.
4	RECORD NOT FOUND (RNF)	One indicates the desired track, sector or side was not found. Reset when updated.
3	CRC ERROR	If Bit 4 is a "1" an error was detected. Reset when updated.
2	LOST DATA	A "1" indicates the Z80 did not respond to the Data Request signal in one-byte time. Reset to zero when updated.
1	DATA REQUEST	If a "1", the Data Register is either full during a read or empty during a write. Generally this bit will be a "1" if Bit 2 is a "1". A "0" denotes a successful command. Reset when updated.
0	BUSY	If a "1", a command is in progress.

Table 6-3
READ/WRITE STATUS

6.3.3 Track Register

The 8-bit TRACK register holds the track number of the current Read/Write command. The data transfer to and from this register is on I/O address 0C1H. If the update flag is "1", the register is either incremented by one every time the head is stepped in toward Track 76; or decremented by one when the head is stepped out toward Track 00.

When the Verify flag is a "1", the contents of the Track register will be compared with the recorded track number during either a disk read or write operation. No verification will be performed if the flag is "0". Using this flag the user can quickly determine if the head is or is not on the proper track.

6.3.4 Sector Register

The 8-bit SECTOR register holds the address of the desired sector position. Register contents are compared with the recorded sector number during either disk read or write operations. The data is either loaded in or transferred from the Sector register via I/O address 0C2H.

6.3.5 Data Register

The transfer of data to and from the formatter/controller chip takes place on I/O addresses 0C3H and 0C7H. the register asociated with adress 0C3H holds the support data needed for the Seek command; and th other address (0C7H) is used when transferring data to and from the disk.

For a Seek operation, the number of the the desired track must be entered into the Data register prior to executing the command. Unless there is a misstep or a similar fault, the track number is not normally set into the Data register. If such a fault does occur, the user can read the current track number from the Track register of I/O address 0C1H. The desired track then can be entered into the Data register on I/O address 0C3H; and the Read/Write head can be positioned to the proper track via a Seek command. Essentially this procedure eliminates the need of performing a "Restore". Figures 6-4 and 6-5 illustrate two ways of performing a Seek operation.

For read and write operations, I/O address 0C7H is used. In other words, address 0C7H is used where the request of data is under control of the interface.

6.4 COMMAND STRUCTURE

The commands that are acceptable by the formatter/controller chip are summarized in Table 6-4. For presentation purposes, the commands will be segregated into the following groups.

Type 1	Head Positioning
Type 2	Sector Operation Group
Type 3	Track Operation Group
Type 4	Reset Interrupt

6.4.1 Head Positioning Commands (Type 1)

A "0" in Bit 7 of the command word identifies the commands that are used to position the Read/Write head. These commands are:

Restore	Find Track 00.
Seek	Find the designated track.
Step	Generate one stepping pulse in the direction of the previous stepping command.
Step In	One step in toward Track 76.
Step Out	One step out toward Track 00.

With reference to Table 6-4, the following paragraphs discuss the use of each bit as related to the head-positioning commands.

Bits 1 and 0 (Stepping Rate)

These bits provide the user with a choice of four stepping rates. These selections are:

BIT 1	BIT 0	STEPPING RATE
0	0	3 ms
0	1	6 ms
1	0	10 ms*
1	1	15 ms

*The "10" bit pattern is recommended for Shugart 800.

COMMAND	BITS							
	7	6	5	4	3	2	1	0
Restore	0	0	0	0	h	V	r1	r0
Seek	0	0	0	1	h	V	r1	r0
Step	0	0	1	u	h	V	r1	r0
Step In	0	1	0	u	h	V	r1	r0
Step Out	0	1	1	u	h	V	r1	r0
Read Sector	1	0	0	m	S	E	C	0
Write Sector	1	0	1	m	S	E	C	a0
Read Address	1	1	0	0	0	E	0	0
Read Track	1	1	1	0	0	E	0	0
Write Track	1	1	1	1	0	E	0	0
Reset Interrupt	1	1	0	1	0	0	0	0

(a) Command Word Format

FLAG	DESCRIPTION
r0,r1	Specifies stepping motor rate
a0	Data Address Mark
V	Verification of destination track
h	Head load at beginning of command
u	Update Flag
E	Designate 15 ms delay
C	Side Compare Flag
S	Side Select Flag
m	Single/multiple designation

*For Shugart 800 drive, set "r1r0 = 10" (10 ms).

(b) Bit Description

Table 6-4
FLOPPY-DISK COMMAND SUMMARY

Bit 2 (Verify Flag)

If this bit is a "1" the track address will be read automatically for verification.

Bit 3 (Head Load)

Whenever this bit is a "1", the read/write head will be loaded at the start of the command. For a seek operation, the actual time that the head is loaded depends upon this bit, and the Verify Flag (Bit 2). If this bit is a "1", the head is loaded; then moved to the desired track. If this bit is a "0", the head moves to the designated track; then is loaded whenever the Verify Flag is a "1".

Bit 4 (Update Flag)

When this bit is a "1" the track register is increment by one for each step. The track register is not updated when the bit is "0".

Bit 5 thru 7 (Command)

These bits designate which command is to be performed.

6.4.2 Sector Commands (Type 2)

Two commands make up this group. They are:

Read Sector: The head is loaded upon receipt of READ SECTOR; and the search for the proper recorded ID field is initiated. When the correct field is found, the recorded data from the data field is read and sent to the Z80. If the Data Address Mark is not found within 30 bytes (single density) or 43 bytes (double density), the Record-Not-Found bit (Refer to Table 6-3) is set; and the operation is terminated.

Write Sector: After the head has been loaded and positioned at the correct track and sector, the unit requests data from the Z80. The formatter/controller counts off either 11 bytes if single density or 22 bytes if double density from the Cyclic Redundacy Characters (CRC). If the request for data has been serviced, the write gate is activated. If the request has not been serviced, the command is terminated; and the Lost Data bit is set (Refer to Table 6-3).

Either six bytes (if single density) or 12 bytes (if double density) of zeros are written onto the disk when write becomes active. At this time, the Data Address Mark will be written if Bit 0 is a "0".

The data is written into the data field of the sector; then another request is made for more information from the Z80. If the request is not serviced in time for continuous writting, the Lost Data bit will be set; and a byte of zeros is written onto the disk. However, the command is not terminated. Following the last recorded data byte, a two-byte CRC is computed internally and recorded on the disk followed by one byte of "1's" in either FM (single density) or MFM (double density) format. The write gate is then deactivated.

Prior to loading either sector command, the sector number must be entered into the Sector register -- sample routines are shown in Figures 6-6 and 6-7. The definition of each bit will be given in the following discussion (Refer to Table 6-4).

Bit 0 (Data Address Mark)

If a "0", the Data Address Mark will be recorded. During a Sector Read, the mark must be found within either 30 bytes (single density) or 43 bytes (double density). The type of Data Address Mark encountered is recorded in Bit 5 of the Status register as shown below:

1	Deleted Data Mark
0	Data Mark

Bit 1 (Side Select Compare)

This bit is used for double-sided disk operation only. When it is a "1", the least-significant bit of the Side field is read and compared with Bit 3 (Side Compare Flag).

Bit 2 (Head Load Delay)

If this bit is a "1", there will be an automatic 15-millisecond delay in loading the Read/Write head, allowing it to be fully engaged. Thus, when the head is not loaded, this bit should be "0" prior to reading from the disk. Otherwise, data can be read incorrectly. However, in the case of consecutive reads, a "0" (no delay) will speed up the access since the head will be already loaded.

Bit 3 (Side Compare Flag)

This bit is for double-sided disk operation only. If Bit 1 is a "1", this bit will be compared with the side number recorded in the ID FIELD. If a comparision is not made within five index pulses, the command will be terminated; and Bit 4 (Record-Not-Found) of the Status register will be set.

Bit 4 (Multiple Sectors)

A "0" indicates that a single sector is to be read or written, while a "1" denotes that multiple sectors are to be read or written. Since the Z80 is fast enough to handle most transfers of successive vectors, it is recommended that a "0" be used in this bit.

Bits 5 thru 7 (Command)

These bits designate the command that is to be executed.

6.4.3 Track Commands (Type 3)

This group of instructions may be used as tools for diagnostic and disk formatting. The three commands are:

Read Address: READ ADDRESS provides a means for establishing the location of the Read/ Write head. The six bytes of data that contains this information is read from the disk and transferred to the Z80 during the execution of this command. For systems having multiple drives or sides, this command also offers a means for determining which drive or side is selected.

NOTE: At the completion of the command, the Track address is entered into the Sector register -- destroying the previous Sector address. Thus, prior to executing a Read Address operation, the Sector register must be loaded with the desired Sector address (Refer to Figure 6-8).

Read Track: READ TRACK is mainly for diagnostic applications. This command can be used as a means to procure data for hard copy. Gaps, address marks, and all data are loaded into the Data register and transferred to the Z80 through I/O address 0C7H. The user can now quickly inspect the disk for valid formatting and data fields as well as address marks. It should be pointed out however, that byte synchronization is not performed until an ID Address Mark is detected. Thus, data prior to the address mark will not be valid.

Write Track: One application of WRITE TRACK is disk formatting. The actual recording begins when the first byte is loaded into the Data register on I/O address 0C7H. From this point, it is up to the software to have data available when the formatter/controller requests it. If the Data register has not been loaded before encountering the next index pulse, the command will terminate; and Bit 2 (Lost Data) of the Status register will be set.

Bit 2 is the only controlling bit (See Table 6-4). This bit defines the head-loading delay as described in the Sector command section. Figure 6-7 is an example routine using the Track instructions.

6.4.4 Reset Interrupt (Type 4)

There are situations where certain conditions can cause the formatter/controller chip and interface circuits to malfunction or freeze. Through the use of the RESET INTERRUPT (Refer to Table 6-4), the current command or condition can be terminated; and the Busy bit will be reset. One approach to a Interrupt Reset is outlined in Figure 6-9.

6.4.5 Write Precompensation

Write precompensation is a techniques where the data is recorded in a direction opposite of the anticipated bit shift. Normally, this is required for double-density recording on 8-inch drives. As a general rule write precompensation is performed on tracks 44 thru 77; but may be required on all tracks if specified by the drive manufacturer. To enable the write precompensation feature, Bit 7 of the Unit Designation byte (I/O address 0C4H) must be a "1".

6.5 FORMATTING THE DISK

To format the disk, the Read/Write head should be positioned over the the desired track; and a Write Track command issued. For every byte of data written, a data request will be generated. This sequence should continue from one index mark to the next. Normally, the data pattern loaded into the Data register will be recorded with a normal clock pattern (See Table 6-5). However, a data pattern of F5 thru FE in the data register will be interrupted as Data Address marks with missing clocks or CRC generation. For instance, the formatter/controller chip interprets an FE in a single-density operation as an ID address mark; and CRC will be initialized. An F7 pattern generates two CRC characters in both single- or double-density operation. Thus patterns F5 thru FE must not appear in the gaps, Data fields or ID fields. Also, an F7 pattern must be used to generate the CRC's.

DATA PATTERN (HEX)	SINGLE DENSITY	DOUBLE DENSITY
00 thru F4	Write 00 thru F4 with CLK = FF	Write 00 thru F4, in MFM
F5	Not Allowed	Write A1* in MFM
F6	Not Allowed	Write C2** in MFM
F7	Generate 2 CRC bytes	Generate 2 CRC bytes
F8 thru FB	Write F8 thru FB, Clk=C7, Preset CRC.	Write F8 thru FB, in MFM
FC	Write FC with Clk=D7	Write FC in MFM
FD	Write FD with Clk=FF	Write FD in MFM
FE	Write FE, Clk=C7, preset CRC.	Write FE in MFM
FF	Write FF with Clk=FF	Write FF in MFM

* Missing clock transition between bits 4 and 5.
** Missing clock transition between bits 3 and 4.

Table 6-5
FORMATTER/CONTROLLER CONTROL BYTES

Other items that should be considered when formatting a disk include:

1. Sector length must be either 128, 256, 512, or 1,024 bytes. The following table lists the hex code that identifies the number of bytes per sector.

SECTOR LENGTH (Hex)	Number of Bytes (Decimal)
00	128
01	256
02	512
03	1024

2. Gap sizes must be in accordance to Table 6-6. These sizes are minimum values as required for proper operation of the formatter/controller device.

6.5.1 Shugart Drives

Either one long record or several small records can be written on the Shugart drives. An index pulse starts each track; and a unique recorded identifier (ID field) precedes each record. The format for a recorded sector is shown in Figure 6-3.

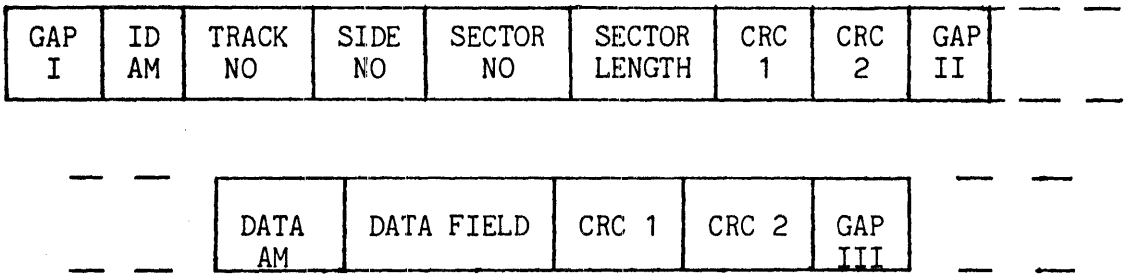


Figure 6-3
DISK SECTOR FORMAT

6.5.1.1 Gaps

A number of bytes containing no data is used to separate each field on a track from the adjacent fields. These areas are referred to as gaps; and they are provided to allow the updating of each field without effecting adjacent fields. There are four unique gaps on each track. The definition of the gaps is given in Table 6-6.

GAP	NAME	BYTES	DESCRIPTION
1	Post Index	32	Gap 1 is between Index Address mark and ID Address mark for Sector 1. It is not affected by the updating process.
2	ID	17	Gap 2 is between the ID field and the Data field, and it may vary in size slightly after the Data field has been updated.
3	Data Gap	33	Gap 3 is between the Data field and the next ID field. As with the ID Gap, it may vary in size slightly after updating the adjacent Data field.
4	Pre-Index	320	Gap 4 is between the last Data field on a track and the Index Address Mark. Initially this gap is nominally 320 bytes; however, due to write frequency tolerances and disk speed, this gap may again change slightly in length.

Table 6-6
GAP DEFINITIONS

6.5.1.2 Address Marks

Each track has a unique combination of data and clock bits referred to as. ADDRESS MARKS. These bit patterns identify the start of the ID and Data fields; and they are used to synchronize the interface circuitry with the first byte of each field. The four types of marks are described in Table 6-6:

MARK	DATA	CLOCK	LOCATION
Index	FC	D7	Beginning of each track
ID Address	FE	C7	Beginning of each ID field
Data Address	FB	C7	Beginning of Data field
Deleted Address	F8	C7	Beginning of deleted Data field

Table 6-7
ADDRESS MARK DEFINITION

6.5.1.3 Cyclic Redundancy Check Character (CRC)

Each field recorded on disk is appended with two CRC bytes. These bytes are generated from a cyclic permutation of the data bits starting with bit zero of the address mark and terminating with bit zero of the last byte within a field (excluding CRC bytes). When a field is read, the data bits are divided by the same general polynomial. A nonzero remainder indicates invalid data, while a zero remainder denotes that correct data has been read.

6.5.1.4 Setting Up The Disk

The program illustrated in Figure 6-10 should be considered as a starting point for formatting a disk to operate on Shugart or similar drive.

6.5.2 IBM Format

The IBM 3740 and the IBM system 34 will be briefly outlined in the following paragraphs.

6.5.2.1 IBM 3740 (Single Density)

To format a disk for the IBM single-density format with 128 bytes per sector, a Write Track command is used; and the following values should be entered into the Data register. For every byte written on disk, there will be one data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
40	FF
6	00
1	FC (Index Mark)
26*	FF
6	00
1	FE (ID Address Mark)
1	Track Number
1	Side Number (00 thru 01)
1	Sector Number (1 thru 1A)
1	00
1	F7 (2 CRC's written)
11	FF
6	00
1	FB (Data Address Mark)
128	Data (IBM uses E5)
1	F7 (2 CRC's written)
27	FF
247**	FF

* Write bracketed field 26 times
** Continue writing nominally 247 bytes

5.2.2 IBM System 34 (Double Density)

The following values should be loaded into the Data register for the double- density format with 256 bytes per sector. For every byte written, there will be a data request.

NUMBER OF BYTES	HEX VALUE OF BYTE WRITTEN
80	4E
12	00
3	F6
1	FC (Index Mark)
50*	4E
12	00
3	F5
1	FE (ID Address Mark)
1	Track Number (0 thru 4C)
1	Side Number (0 or 1)
1	Sector Number (1 thru 1A)
1	01
1	F7 (2 CRC's written)
22	4E
12	00
3	F5
1	FB (Data Address Mark)
256	DATA
1	F7 (2 CRC's written)
54	4E
598**	4E

* Write Bracketed field 26 times
** Continue writing for nominally 598 bytes

Flow Diagram:

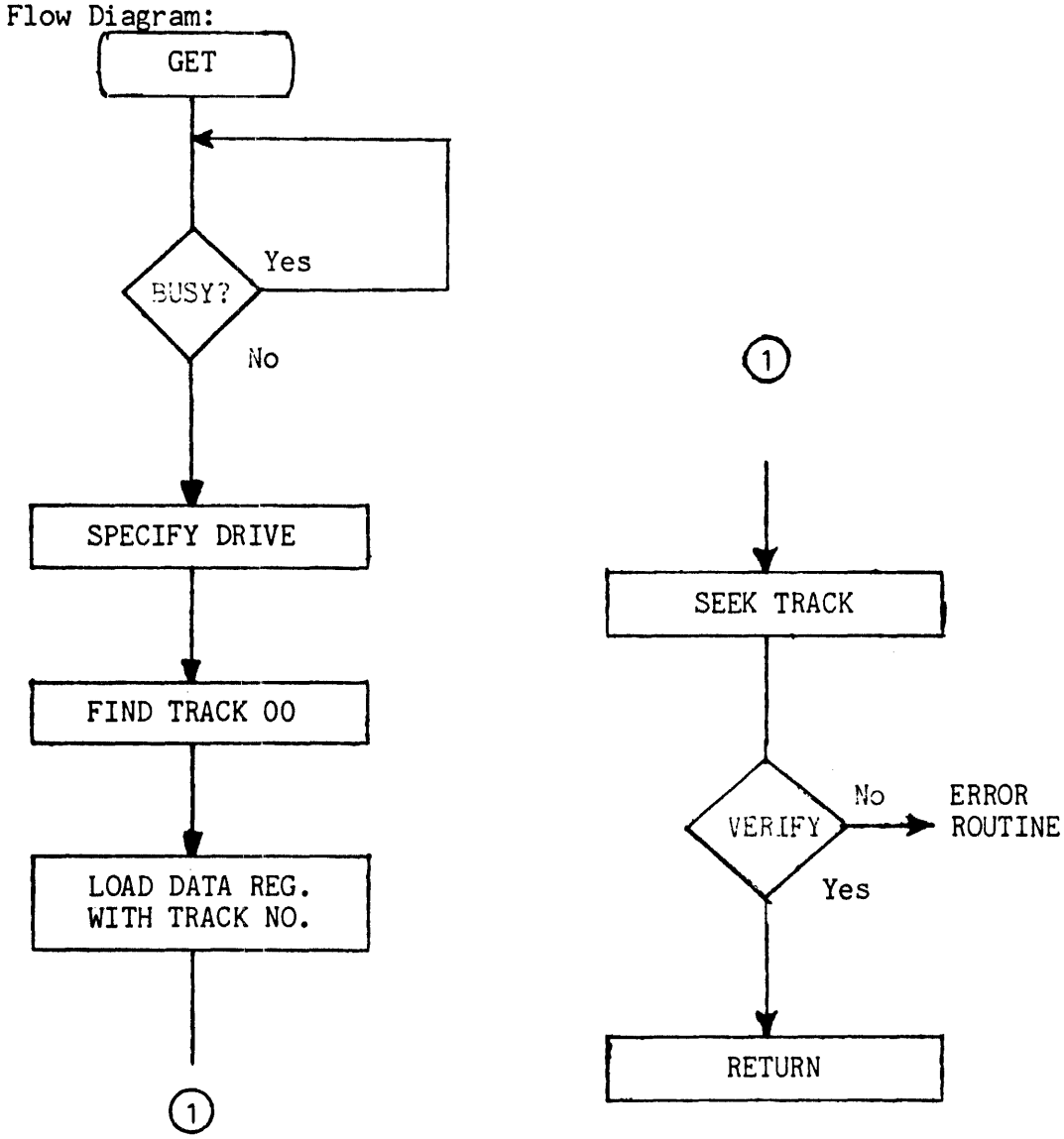


Figure 6-4
TRACK SEEK
FROM TRACK 00

Source Program:

```
 1                      ;TRACK SEEK FROM TRACK 00
 2                      ;ENTRY: TRACK ID VERIFY
 3                      ;EXIT: TO ERROR ROUTINE IF NO VERIFICATION
 4                      ;
 5  0000 CD 1D 00       GET:    CALL    STAT        ;GET STATUS, CHIP BUSY
 6  0003 3A 01 00               LDA     01H
 7  0006 D3 C4                  OUT     0C4H        ;SET UP UNIT 0
 8  0008 3A 02 00               LDA     02H         ;A=RESTORE INSTR.
 9  000B D3 C0                  OUT     0C0H        ;RESTORE
10  000D 7E                     MOV     A,M         ;A=TRACK NUMBER
11  000E D3 C3                  OUT     0C3H        ;LOAD TRACK REG.
12  0010 3A 16 00               LDA     16H         ;A=SEEK INSTR.
13  0013 D3 C0                  OUT     0C0H        ;SEEK & VERIFY
14  0015 DB C0                  IN      0C0H        ;READ STATUS
15  0017 E6 10                  ANI     10H         ;VERIFY
16  0019 CA 1E 00               JZ      ERR         ;NO VERIFICATION
17  001C C9                     RET                 ;VERIFIED
18  001D                STAT:   DS      1
19  001E                ERR:    DS      1
```

Figure 6-4 (cont'd.)
TRACK SEEK
FROM TRACK 00

Flow Diagram:

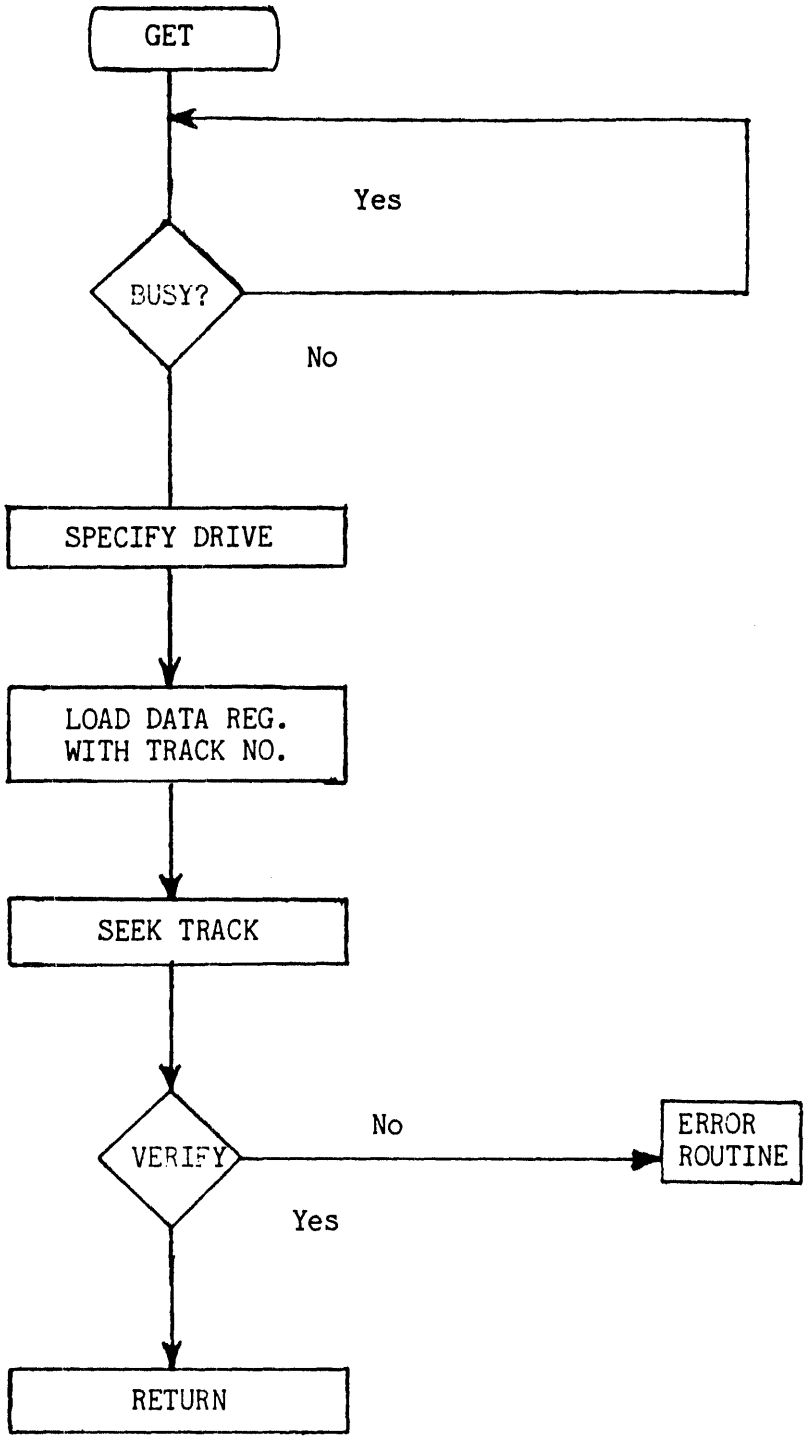


Figure 6-5
TRACK SEEK

Source Program:

```
 1                          ;TRACK SEEK ROUTINE
 2                          ;TRACK REG. HAS CURRENT HEAD POSITION
 3                          ;ENTRY: TRACK ID WITH VERIFICATION
 4                          ;
 5   0000 CD 18 00      GET:    CALL    STAT        ;GET STATUS, CHIP BUSY?
 6   0003 3A 01 00              LDA     01H
 7   0006 D3 C4                 OUT     0C4H        ;SET UP UNIT 0
 8   0008 7E                    MOV     A,M         ;TRACK NO.
 9   0009 D3 C3                 OUT     0C3H        ;LOAD DATA REG.
10   000B 3A 14 00              LDA     14H
11   000E D3 C0                 OUT     0C0H        ;SEEK & VERIFY
12   0010 DB C0                 IN      0C0H        ;READ STATUS
13   0012 E6 10                 ANI     10H         ;VERIFY
14   0014 CA 21 00              JZ      ERR         ;NO VERIFICATION
15   0017 C9                    RET                 ;VERIFIED
16   0018               STAT:   DS      9
17   0021               ERR:    DS      10
```

Figure 6-5 (cont'd.)
TRACK SEEK

Flow Diagram:

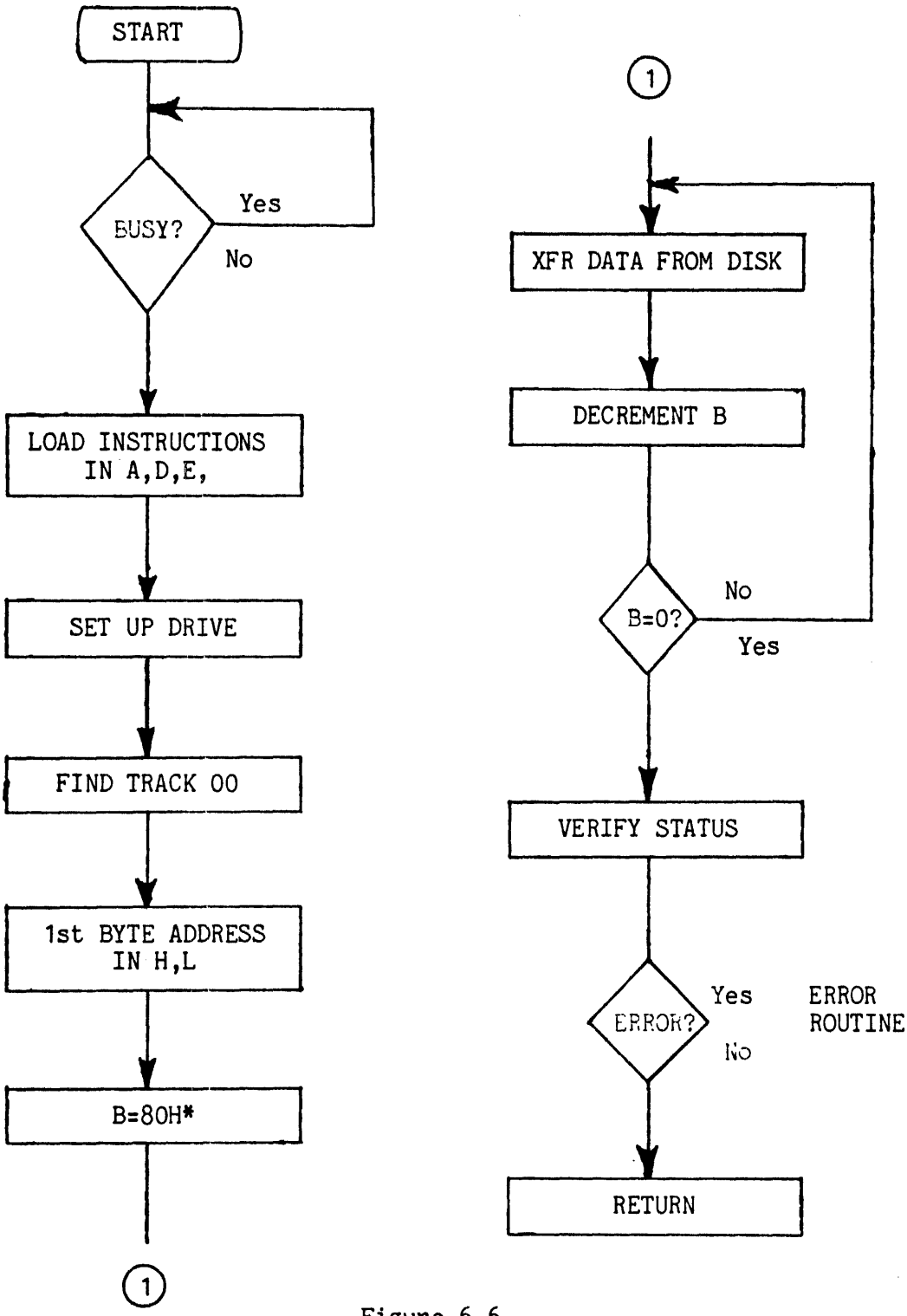


Figure 6-6
SECTOR READ ROUTINE

Source Program:

```
 1                          ;SECTOR READ ROUTINE
 2                          ;ENTRY: NOTHING NEEDED
 3                          ;SET B=80H IF SINGLE DENSITY
 4                          ;SET B=00H IF DOUBLE DENSITY
 5   0000 CD 23 00      GET:    CALL   STAT       ;GET STATUS, CHIP BUSY?
 6   0003 11 06 80              LXI    D,8006H    ;D=RESTORE,E=SECTOR READ
 7   0006 3A 01 00              LDA    01H        ;A=UNIT
 8   0009 D3 C4                 OUT    0C4H       ;SET UP DRIVE
 9   000B ED 51                 OUTP   D          ;RESTORE
10   000D 7E                    MOV    A,M        ;A=TRACK NO.
11   000E D3 C3                 OUT    0C3H       ;DATA REG.=TRACK NO.
12   0010 7E                    MOV    A,M        ;A=SECTOR
13   0011 D3 C2                 OUT    0C2H       ;SECTOR
14   0013 E1                    POP    H          ;1ST BYTE ADDRESS
15   0014 01 80 C7              LXI    B,0C780H   ;B=80H*,C=IO
16   0017 ED 59                 OUTP   E          ;READ SECTOR
17   0019 ED BA                 INDR              ;XFR DATA TO CPU
18   001B DB C0                 IN     0C0H       ;READ STATUS
19   001D F6 00                 ORI    00H        ;VERIFY STATUS
20   001F C2 24 00              JNZ    ERR        ;ERROR EXIT
21   0022 C9                    RET               ;DONE
22   0023               STAT:   DS     1
23   0024               ERR:    DS     1
```

Figure 6-6 (cont'd.)
SECTOR READ ROUTINE

Flow Diagram:

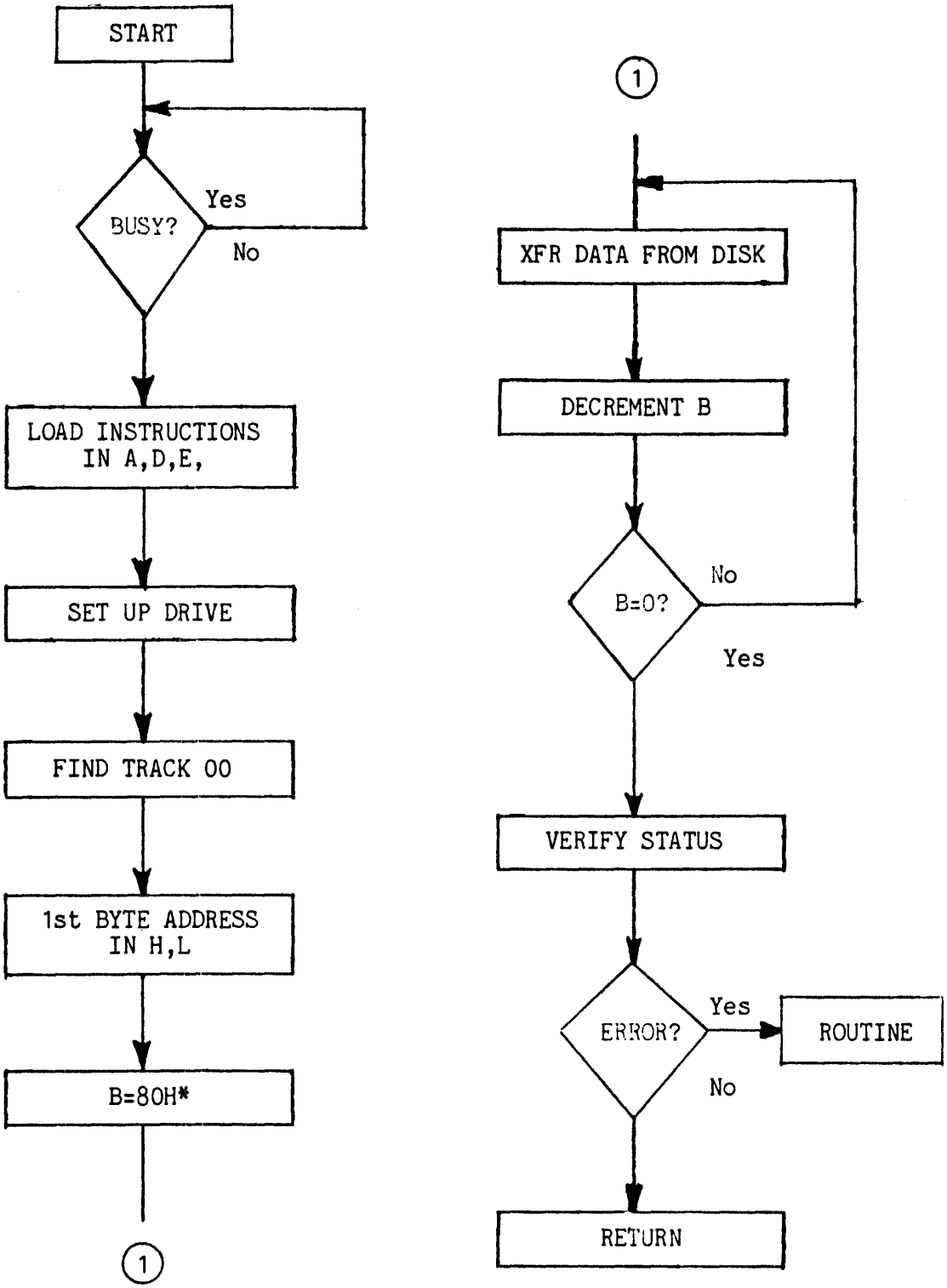


Figure 6-7
SECTOR WRITE ROUTINE

Source Program:

```
 1                          ;SECTOR WRITE ROUTINE
 2                          ;ENTRY: NOTHING NEEDED
 3                          ;SET B=80H IF SINGLE DENSITY
 4                          ;SET B=00H IF DOUBLE DENSITY
 5   0000 CD 23 00      GET:    CALL    STAT        ;GET STATUS, CHIP BUSY?
 6   0003 11 06 A0              LXI     D,0A006H    ;D=RESTORE,E=SECTOR WRITE
 7   0006 3A 01 00              LDA     01H         ;A=UNIT
 8   0009 D3 C4                 OUT     0C4H        ;SET UP DRIVE
 9   000B ED 51                 OUTP    D           ;RESTORE
10   000D 7E                    MOV     A,M         ;A=TRACK NO.
11   000E D3 C3                 OUT     0C3H        ;DATA REG.=TRACK NO.
12   0010 7E                    MOV     A,M         ;A=SECTOR
13   0011 D3 C2                 OUT     0C2H        ;SECTOR
14   0013 E1                    POP     H           ;1ST BYTE ADDRESS
15   0014 01 80 C7              LXI     B,0C780H    ;B=80H*,C=IO
16   0017 ED 59                 OUTP    E           ;WRITE SECTOR
17   0019 ED BB                 OUTDR               ;XFR DATA TO CPU
18   001B DB C0                 IN      0C0H        ;READ STATUS
19   001D F6 00                 ORI     00H         ;VERIFY STATUS
20   001F C2 24 00              JNZ     ERR         ;ERROR EXIT
21   0022 C9                    RET                 ;DONE
22   0023               STAT:   DS      1
23   0024               ERR:    DS      1
```

Figure 6-7 (cont'd.)
SECTOR WRITE ROUTINE

Flow Diagram:

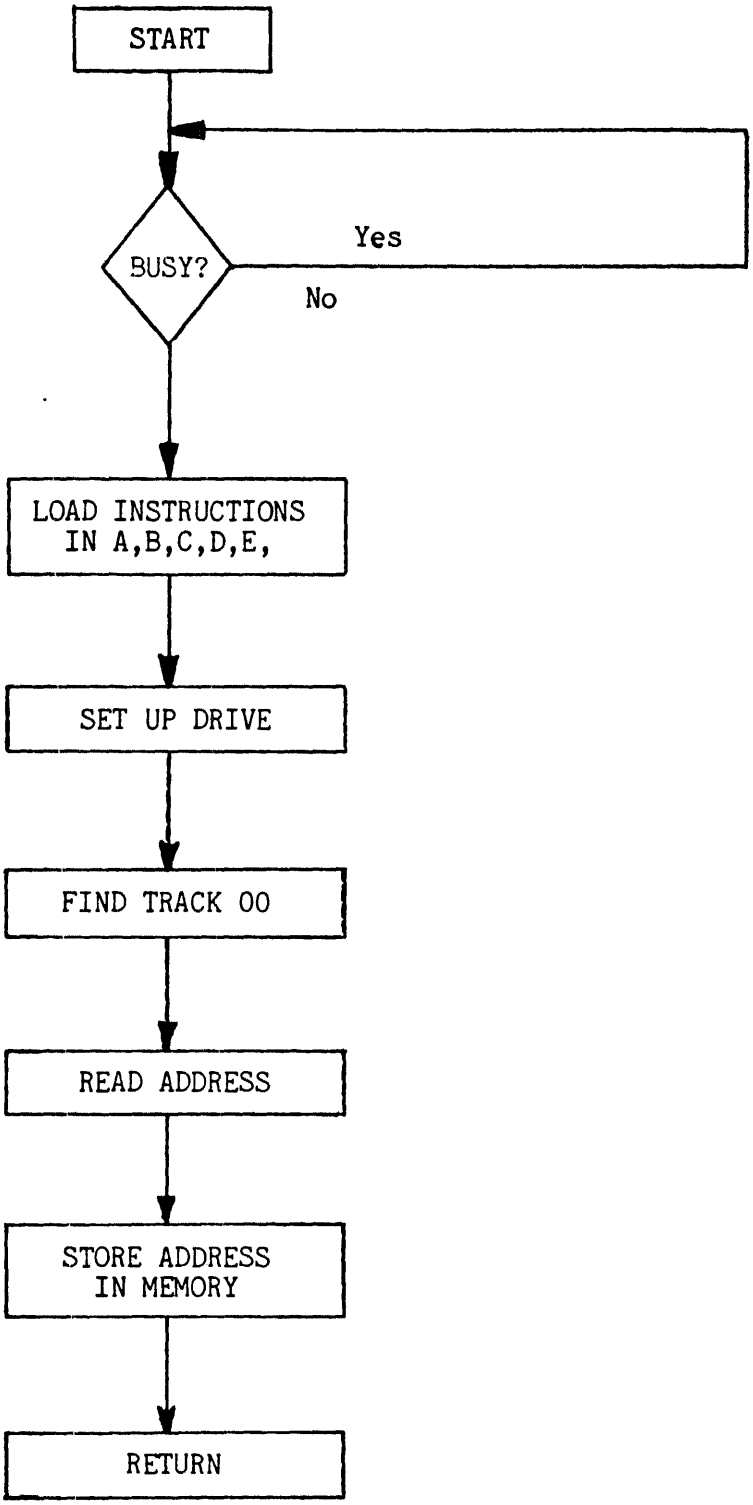


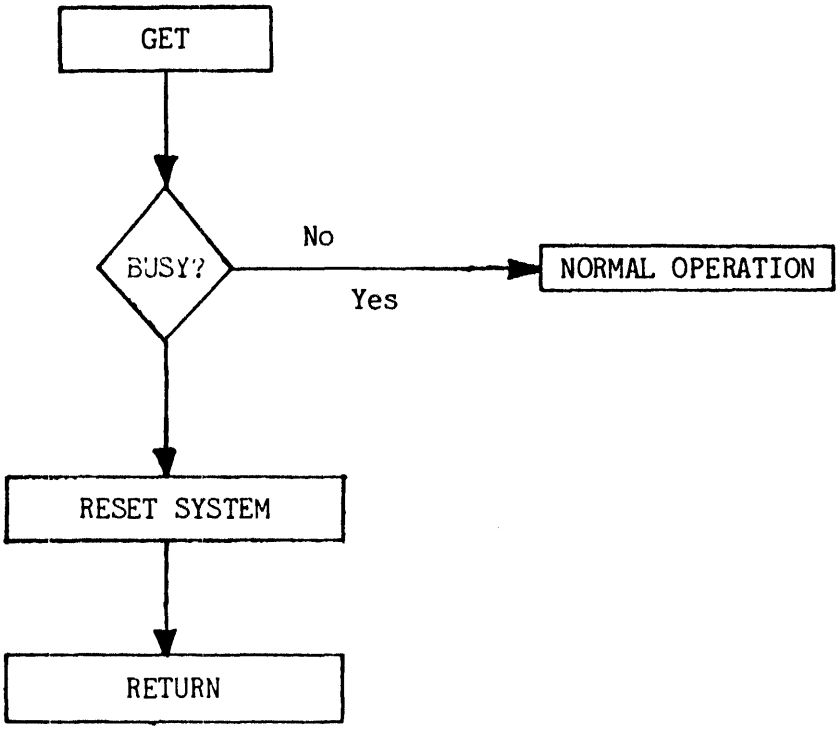
Figure 6-8
READ ADDRESS ROUTINE

Source Program:

```
 1                      ;READ ADDRESS ROUTINE
 2                      ;ENTRY: NOTHING NEEDED
 3                      ;
 4  0000 CD 2C 00  GET:    CALL   STAT       ;GET STATUS, CHIP BUSY?
 5  0003 01 02 C0          LXI    B,0C002H   ;B=RESTORE,C=COMMAND ADD.
 6  0006 11 12 D0          LXI    D,0D012H   ;D=SEEK,E=READ ADDRESS
 7  0009 3A 01 00          LDA    01H        ;A=UNIT
 8  000C D3 C4             OUT    0C4H       ;SET UP DRIVE
 9  000E ED 41             OUTP   B          ;RESTORE
10  0010 7E                MOV    A,M        ;A=TRACK
11  0011 D3 C3             OUT    0C3H       ;DATA REG.=TRACK
12  0013 ED 51             OUTP   D          ;SEEK & VERIFY
13  0015 DB C0             IN     0C0H       ;READ STATUS
14  0017 E6 10             ANI    10H        ;VERIFY
15  0019 CA 2D 00          JZ     ERR1       ;ERROR
16  001C E1                POP    H          ;1ST BYTE ADDRESS
17  001D ED 59             OUTP   E          ;READ ADDRESS
18  001F 01 06 C7          LXI    B,0C706H   ;B=06H,C=IO ADDRESS
19  0022 ED BA             INDR              ;XFR ADDRESS TO MEMORY
20  0024 DB C0             IN     0C0H       ;READ STATUS
21  0026 E6 80             ANI    080H       ;CHECK CSC BIT
22  0028 CA 2E 00          JZ     ERR2
23  002B C9                RET               ;DONE
24  002C               STAT:   DS     1
25  002D               ERR1:   DS     1
26  002E               ERR2:   DS     1
```

Figure 6-8 (cont'd.)
READ ADDRESS ROUTINE

Flow Diagram:



Source Program:

```
 1                      ;RESET INTERRUPT
 2                      ;ENTRY: NOTHING NEEDED
 3                      ;SYSTEM RESET
 4                      ;
 5   0000 CD 10 00      GET:    CALL    STAT        ;GET STATUS
 6   0003 F5                    PUSH    PSW         ;SAVE
 7   0004 E6 80                 ANI     80H         ;CHECK BUSY
 8   0006 C2 0A 00              JNZ     INTR        ;INTERRUPT IF BUSY
 9   0009 F1                    POP     PSW         ;NORMAL OPERATION
10   000A 3A 00 00      INTR:   LDA     0D0H
11   000D D3 C0                 OUT     0C0H        ;INTERRUPT
12   000F C9                    RET                 ;DONE
13   0010               STAT:   DS      1
```

Figure 6-9
RESET INTERRUPT ROUTINE

Flow Diagram:

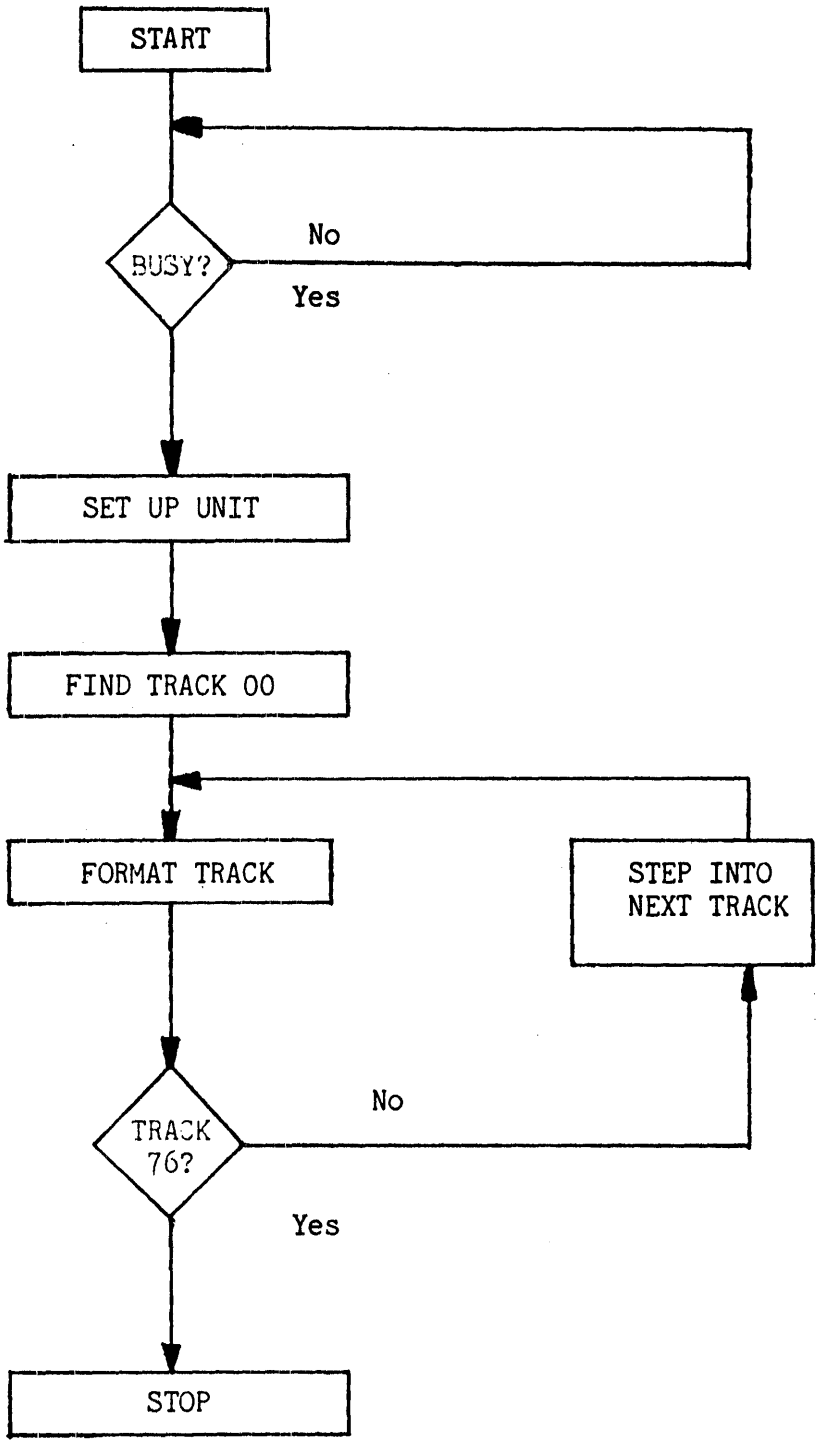


Figure 6-10
DISK INITIALIZATION ROUTINE

Source Program:

```
 1                      ;DISK INITIALIZATION (256 BYTES/SECTOR)
 2                      ;ENTRY:  BLANK DISK
 3                      ;
 4  0000 CD 81 02       GET:    CALL    STAT        ;GET STATUS, CHIP BUSY
 5  0003 3A 01 00               LDA     01H         ;A=UNIT
 6  0006 D3 C4                  OUT     0C4H        ;SET UP DRIVE
 7  0008 3A 02 00               LDA     02H         ;A=RESTORE INSTR.
 8  000B D3 C0                  OUT     0C0H        ;RESTORE
 9  000D 06 21          LOOP:   MVI     B,21H       ;B=21
10  000F 0E C7                  MVI     C,0C7H      ;C=C7
11  0011 2A 2B 00               LHLD    ADD         ;1st BYTE OF FORMAT
12  0014 3A F0 00               LDA     0F0H        ;A=WRITE TRACK INSTR.
13  0017 D3 C0                  OUT     0C0H        ;WRITE TRACK
14  0019 ED B3                  OUTIR               ;FORMAT SECTOR
15  001B DB C1                  IN      0C1H        ;A=TRACK NO.
16  001D FE 76                  CPI     76H         ;A-76
17  001F CA 2A 00               JZ      STOP        ;INIT. DONE
18  0022 3A 5A 00               LDA     5AH         ;A=STEP IN INSTR.
19  0025 D3 C0                  OUT     0C0H        ;STEP IN
20  0027 C3 0D 00               JMP     LOOP        ;FORMAT NEXT TRACK
21  002A 76             STOP:   HLT                 ;DONE
22  002B                ADD:    DS      256H        ;FORMAT
23  0281                STAT:   DS      1
```

Figure 6-10 (cont'd)
DISK INITIALIZATION ROUTINE

SECTION 7

INTERRUPT

7.1 SCOPE

The MSC 8009 interrupt system provides eight levels of priority interrupts plus a Non-Maskable Interrupt.

This section details the programming of the Interrupt system for full priority-interrupt servicing. Simple applications do not require this complexity. Typically, a subset of the procedures outlined in this section will suffice.

7.2 Z80 INTERRUPT CONTROL

The Z80 interrupt system operates in one of three modes.

MODE 0 generates eight, RST (RESTART) instructions (8080 interrupt response mode).

MODE 1 is useful for debugging purposes because all interrupts execute a restart at location 0038H (RST7). This mode does not use vector PROM.

MODE 2 uses a PROM to specify the eight, low-order bits of an interrupt-vector address for each of the eight, priority levels. The Z80 interrupt-vector register defines the eight, high-order bits that are set by the program. This means that any interrupt can be vectored to any memory location. The page address must be set into the Z80 interrupt vector register by the user's program.

The Z80 makes use of several instructions to control the interrupt processing. These intructions are:

MNEMONIC	OPERATION
IM0	Sets Interrupt MODE 0
IM1	Sets Interrupt MODE 1
IM2	Sets Interrupt MODE 2
EI	Enables Interrupt
DI	Disables Interrupt

The enable and disable instructions, EI and DI, are used to set or clear the INTERRUPT ENABLE latch in the processor. The state of this latch may be tested by loading the refresh or interrupt-vector register into the accumulator. When executed, these instructions cause the state of the Z80 Interrupt Enable latch to be loaded into the Parity flag.

NOTE: With the 8214 Interrupt chip on the MSC 8009 board, a normal practice would be to use it to control the enabling and disabling of interrupts. In this case, the Z80's Interrupt Enable flag would ordinarily be left enabled at all times. Thus testing the flag in the Z80 yields no useful information.

7.3 8214 INTERRUPT CONTROLLER

The 8214 Interrupt Controller and its associated Interrupt Vector PROM must be set up to allow the desired interrupt response. This is achieved by performing an "OUT 0D7H" instruction with the following information in the accumulator. The most-significant bit must be set to correspond to the Z80's Interrupt mode.

MODE	INSTRUCTION	MOST-SIGNIFICANT BIT
0	IM0	0
1	IM1	Don't Care
2	IM2	1

The lower-four bits (0 thru 3) define the priority level, which corresponds to the lowest interrupt number that is disabled. The remaining bits (4 thru 6) must remain in the desired state. This becomes automatic if the user keeps a copy of the register setting in memory -- the reason for "MASK" in the software example of paragraph 7.3.3. The following paragraphs discuss the use of these bits in more detail.

DATA BITS	PURPOSE	FUNCTION
DAT7	Interrupt Control	Mode Latch Control ("1" for Z80 Mode 2; "0" for Z80 Mode 0)
DAT6	Address Control	Address Map Switch ("1" for Alternate; "0" for Main Map)
DAT5	Bus Mode	Select Bus Exchange mode (See paragraph 3.3.1.3)
DAT4	Bus Mode	Select Bus Exchange mode (See paragraph 3.3.1.3)
DAT3	Interrupt Control	Status Group Select ("0" for priority "1" for none)
DAT2	Interrupt Control	B2 (This line is used to determine which level is enabled)
DAT1	Interrupt Control	B1 (This line is used to determine which level is enabled)
DAT0	Interrupt Control	B0 (This line is used to determine which level is enabled)

Table 7-1
DATA BIT FUNCTIONS
FOR OUTPUT TO INTERRUPT CONTROLLER
(Device Code D7)

7.3.1 Initialization

The initialization of the interrupt system must be accomplished in several places, including

1) Within the Z80 processor.

2) The 8214 Priority-Interrupt Controller and mode latch.

3) The I/O device that generates the interrupts.

After a power up, the logic states are indeterminate, and an initialization procedure must allow for this. The following discussion is based on MULTIBUS signal names, polarities and priorities. The inversions and rearrangements created by the 8214 and the vector PROM may be ignored unless improper operation is suspected. For troubleshooting, refer to the schematic in the appendix.

The outputs of the 8214 are latched to prevent the loss of interrupts when the Z80 produces several M1 cycles. These cycles occur during those instructions that use the index registers.

Normally, the Z80 interrupt is enabled after the 8214, and no spurious power up interrupt will occur because the enabling of the 8214 clears all pending interrupts. If the order of enabling interrupts is reversed, a spurious interrupt can happen. Notice that a device holding down an interrupt line will generate a new interrupt immediately following the enabling of the 8214. Therefore, no device will be stranded.

7.3.2 MULTIBUS Interrupt

The MULTIBUS interrupts are numbered INT0/ (highest priority) thru INT7/ (lowest priority). The least-significant four-bits (STATUS GROUP, B2, B1, and B0) define the priority number. To illustrate, consider the following listing.

OUTPUT TO INTERRUPT CONTROLLER	PRIORITY STRUCTURE
0	Everything Off
8	Everything On
7	Disable 7; Enable 6 thru 0
6	Disable 7 and 6; Enable 5 thru 0
5	Disable 7 thru 5; Enable 4 thru 0
4	Disable 7 thru 4; Enable 3 thru 0
3	Disable 7 thru 3; Enable 2 thru 0
2	Disable 7 thru 2; Enable 1 and 0
1	Disable 7 thru 6; Enable 0

The formula is as follows:

> When an interrupt occurs on level "n", send out a mask of "n" and re-enable the processor interrupt to enable the higher levels.

Output to the interrupt controller is device code D7 HEX.

7.3.3 Programming Multi-Level Interrupts

To successfully operate a multi-level, priority-interrupt scheme the user must insure that an interrupt restores the system to the exact state which existed prior to the interrupt. This requires that the program keep track of the level last sent to the 8214, since there is no way to read this back from the 8214. The following description can be used to develop interrupt software.

1) Initialization of interrupt system:

```
        MVI     A,DMASK   ;DESIRE MASK
        STA     MASK      ;SAVE FOR EXACT VALUE RESTORATION
        OUT     0D7H      ;SETUP 8214
        EI                ;ENABLE Z80 INTERRUPT
         .
         .
         .
User Code
         .
         .
         .
```

2) At beginning of interrupt code:

```
PUSH    PSW       ;SAVE ACC
LDA     MASK      ;SET OLD MASK
PUSH    PSW       ;SAVE IT TOO
ANI     0F0H      ;CLEAR LO BITS
ORI     LEVEL     ;SET INTERRUPT LEVEL
STA     MASK      ;SAVE NEW MASK
OUT     0D7H      ;SET UP 8214
EI                ;RE-ENABLE Z80
User interrupt service
        .
        .
        .
```

NOTE: The user must save and restore any register that is used including memory locations. The machine must terminate in the same condition it started in.

3) At end of interrupt code

```
DI*               ;TEMPORARILY IGNORE INTERRUPTS
POP     PSW       ;SET OLD MASK
STA     MASK      ;KEEP IT
OUT     0D7H      ;RESORE 8214
POP     PSW       ;GET OLD ACCUMULATOR
EI*               ;DO INTERRUPTS AGAIN
RTI               ;RETURN TO INTERRUPT PROCESS
```

In the previous code*, DI and EI may be deleted, but it is easier to understand and debug machine activity with them included.

7.4 8214 PRIORITY INTERRUPT CONTROLLER

The 8214 is an eight-level, priority-interrupt controller. The encoder portion of the 8214 accepts up to eight, active-low interrupt requests (R0/ thru R7/) and determines which has the highest priority -- R7/ having the highest priority. Once priority is established, the 8214 then compares that priority with a software-created, current-status register on B0/ thru B2/. If the incoming request is of a higher priority than the interrupt currently being serviced, an interrupt request to the processor is generated. Vector information that identifies the interrupting device is also generated.

7.4.1 Address And Bit Assignments

The 8214 is accessed by an "I/O Write To" device code D7. The three least-significant data bits are strobed into B0 thru B2 of the 8214. The most-significant data bit sets the Mode flip flop.

Data 7 equals "1" sets the PROM for MODE 2.
Data 7 equals "0" sets the PROM for MODE 0.

7.4.2 Vectors

The highest-priority MULTIBUS interrupt line is INT0/. It is connected with R7/ -- wirewrap post 86 on the MSC 8009 board.

PRIORITY REQUEST	RST	D7	D6	D5	D4	D3	D2	D1	D0
		1	1	A2	A1	A0	1	1	1
INT7/	7	1	1	1	1	1	1	1	1
INT6/	6	1	1	1	1	0	1	1	1
INT5/	5	1	1	1	0	1	1	1	1
INT4/	4	1	1	1	0	0	1	1	1
INT3/	3	1	1	0	1	1	1	1	1
INT2/	2	1	1	0	1	0	1	1	1
INT1/	1	1	1	0	0	1	1	1	1
INT0/	0	1	1	0	0	0	1	1	1

CAUTION: RST 0 will vector the program counter to location 0 (zero) and execute the same routine as the RESET input.

MODE 2 Interrupt

Interrupt Level	PROM Vector
INT0/	XXF0
INT1/	XXF2
INT2/	XXF4
INT3/	XXF6
INT4/	XXF8
INT5/	XXFA
INT6/	XXFC
INT7/	XXFE

"XX" portion of the vector is the contents of the Z80 interrupt vector register.

7.5 NON-MASKABLE INTERRUPT (NMI)

The Non-Maskable Interrupt (NMI) has higher priority than a normal interrupt and is sampled at the same time as the interrupt line. NMI cannot be disabled under software control. Its usual function is to provide immediate response to important signals such as an impending power failure. The Z80 response to NMI is similar to a normal memory read operation. The only difference being that the content of the data bus is ignored while the processor automatically stores the program counter in an external stack and jumps to location 0066 HEX. The service routine for NMI must begin at this location if this interrupt is to be used.

Two jumpers are associated with NMI. One jumper (64 and 65) allows the MSC 8009 Watchdog Timer to generate NMI if an acknowledgement is not received within 10 milliseconds during a memory or I/O operation. The other jumper (70 to 71) connects NMI to the MULTIBUS through P1, pin 33; permitting an exgternally generated NMI to be recognized. NMI/ may also be found on auxilliary connector P2, pin 4.

CAUTION: This connection to P1, pin 33 is not compatible with the MULTIBUS standard; and this connection may not be made if INTA/ is being used.

SECTION 8

THEORY OF OPERATION

8.1 SCOPE

This section discusses the theory of operation and system control logic of the MSC 8009, single-board computer. This discussion is intended to assist the user in understanding the overall system so that the MSC 8009 capabilities can be efficiently employed.

8.2 SYSTEM DESCRIPTION

There are five basic elements that make up the MSC 8009 -- the processor, the arithmetic processor, the disk formatter/controller, the memories, and the I/O. These elements are interconnected by three buses -- an Address bus, a Data bus, and a Control bus. These buses can extend beyond the MSC 8009 board via edge connector P1, and these buses are MULTIBUS* compatible. It is through the external MULTIBUS that the MSC 8009 interfaces with other processors, memories or I/O elements for developing of a larger system.

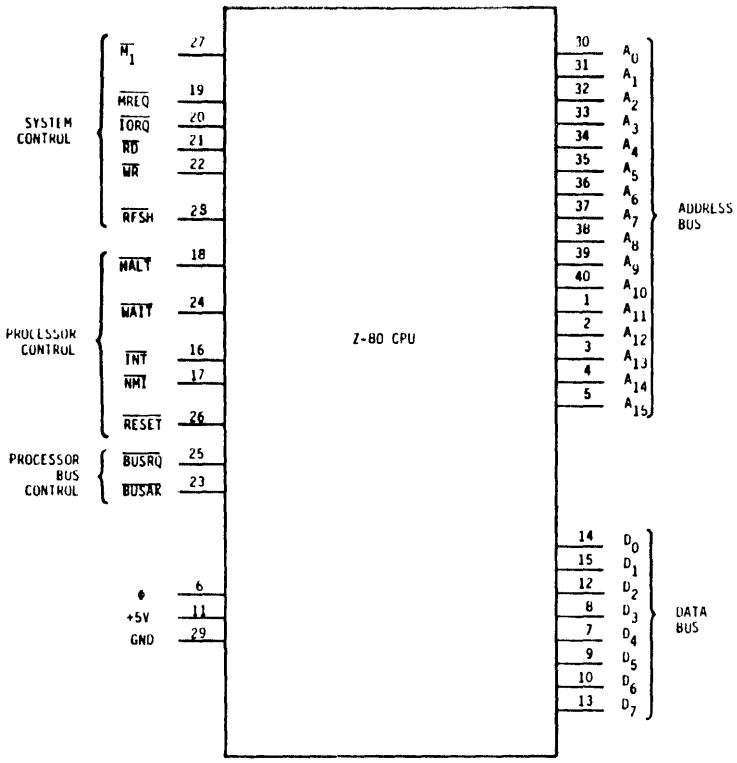
8.2.1 Local Control Bus

Unlike the Address and Data buses, the local control is not a single, well-defined bus; but actually a variety of signals within the MSC 8009 as well as the external signals from the MULTIBUS. This local control bus determines the source and destination of the address- and data-bus information.

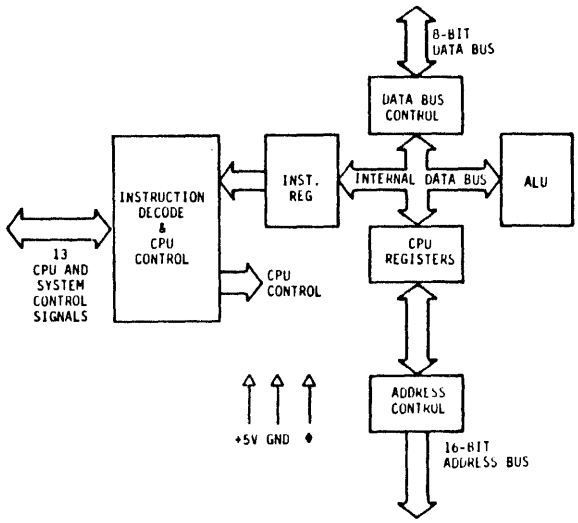
PROM U38 accepts and decodes the Z80-generated signals MRQ/, IORQ/, RD/ and WT/ to generate local-control signals LIORC/ (I/O Read), LIOWC/ (I/O Write), LMRDC/ (Memory Read), and LMWTC/ (Memory Write). When the MSC 8009 wants access to the MULTIBUS or another processor on the MULTIBUS, transceiver U44 asserts the forementioned control signals onto the bus as IORC/, IOWC/, MRDC/ and MWTC/.

8.2.1.1 MULTIBUS Control

Transceiver U44 permits other processors on the MULIBUS to access the MSC 8009 through the use of control signals IORC/, IOWC/, MRDC/ AND MWTC/. When the internal-bus state machine places a "low" on EXTC and a "high" on PROCA/, U44 accepts and converts these bus signals to the four, local-control signals described in paragraph 8.2.1.



(a) PIN CONFIGURATION



(b) BLOCK DIAGRAM

Figure 8-1
Z80 MICROPROCESSOR

8.2.2 Local Address Bus

The Local-address bus or L/A BUS is a sixteen-bit internal-address bus that is unidirectional in nature and addresses the onboard memory and I/O devices. Tri-state drivers U46, 47 and 55 multiplex the Z80 address lines onto the L/A BUS when an internal operation is being executed (PROCA/ is "low"). A high-speed address PROM (U54) in conjunction with two, three-to-eight decoders (U21 and U27) encode the L/A BUS lines as ROMSEL/, RAMSEL and IOSEL/ signals (Refer to Table 4-2). Also , the data selectors U41, 49, 57 and 55 convert the sixteen lines of the L/A BUS into seven-bit address lines as required by the onboard RAM array.

When executing an I/O operation, the lower-eight Z80 address lines containing the port address are duplicated on the upper-eight MULTIBUS address lines. This assures compatibility with the 8080 microprocessor series and always places the same I/O port address on both halves of the address bus.

8.2.2.1 MULTIBUS Addressing

The bus transceivers U63 and U64 allow either another processor to address the onboard memory (DMA) and I/O devices; or the Z80 address another external unit on the MULTIBUS. The internal-bus state machine generates PROCA/ to define direction of the data flow and EXTRADR as an enable for the transceivers.

8.2.3 Data Channel

The bidirectional-data bus of the MSC 8009 interconnects the onboard memory and I/O devices with either the Z80 (D0 thru D7) or the MULTIBUS (DAT0/ thru DAT7/). A "low" on PROCA/ lets the transceiver U48 transfer seven-bits of data to and from the Z80, depending on the state of RD/. A DMA transfer uses transceiver U65 to route data between the MULTIBUS and the MSC 8009.

8.2.4 Z80 Processor

Whereas the Z80 (U34) executes the instruction set as well as coordinating all bus transfers and internal functions, it is important that the user has complete knowledge of this microprocessor. This information may be obtained from either the user's manual or available data sheets. Figure 8-1 shows the Z80 pin assignments with signal designations.

During each major cycle of the processor, the Z80 places a sixteen-bit address on the ADDRESS BUS (A0 thru AF). Appropriate control signals are then generated to transfer eight bits of data (D0 thru D7) to and from the MSC 8009 using the bidirectional data bus (Refer to paragraph 8.2.3). The MSC 8009 control lines allow the Z80 to exchange data with both internal and external memory or I/O devices.

8.2.5 Floppy Disk Formatter/Controller

The Floppy Disk Formatter/Controller U321 performs all functions as needed to support both 8- and 5-inch drives intermixed with single- or double-density storage capabilities. When DDEN/ (Double Density Enable) is a logic "1", data will be either written or read based on frequency-modulated format (FM). Both clock and data are recorded serially on each track of the disk. A data pulse or flux transition between clock pulses indicates a logic "1"; a "0" denotes the absence of a pulse.

Data exchange between the Z80A and the formatter/controller chip is in parallel via the I/O BUS. A PROM provides the necessary control. This 8-bit data is either stored in one of the registers within the chip or translated into a serial format for transmission to the drive. If the operation is a read, the serial data from the drive will be converted into a parallel format for assertion onto the I/O BUS.

At the start of each instruction, the PAUSE/ line will be pulled to a "low" via the gating action of U304 (pins 1, 2 and 4 thru 6) -- "freezing" the Z80. PAUSE/ remains "low" until the requested operation is complete; and then PAUSE/ is released. However, Bit 0 (Busy Flag) of the Status register can be monitored to verify that the formatter/controller device is ready to accept another command.

8.2.6 Arithmetic Processing Unit (APU)

The Arithmetic Processing Unit (U15) provides the MSC 8009 with fixed- and floating-point arithmetic as well as transcendental functions. At the beginning of each command, the 9511 asserts a "low" on PAUSE/, "freezing" the Z80. PAUSE/ remains "low" until the 9511 completes the previous operation, and then PAUSE/ is released. However, the Status register can be monitored to verify that the 9511 is not busy (Busy Flag).

Data exchange between the Z80A and the 9511 is dispatched on the I/O BUS. Under PROM control, signals LA0, LIORC/, and LIOWC/ establish the following types of transfers.

LA0	LIORC	LIOWC	DESCRIPTION
0	1	0	Enter data byte into stack
0	0	1	Read data byte from stack
1	1	0	Enter command
1	0	1	Read status

8.2.7 System Clock

A packaged oscillator (U23) provides a 16-MHz signal as the basic clock for the MSC 8009. This signal is then fed into a four-bit counter which divides the basic-clock frequency by two, four, eight and sixteen. The 8-MHz clock is used for the State Machines and also made available on the MULTIBUS. Installed jumpers route the other clocks to the Z80A, APU, and several I/O circuitry (See Table 8-1).

JUMPER	CLOCK FREQUENCY	FUNCTION
68 to 69	8 MHz	MULTIBUS (CCLK/)
62 to 63	8 MHz	MULTIBUS (BLCK/)
*58 to 59	4 MHz	Z80A Processor
56 to 57	4 MHz	APU (MATHCLK)
*54 to 55	2 MHz	APU (MATHCLK)
52 to 53	2 MHz	Z80 Processor
50 to 51	2 MHz	I/O (IOCLK)
*48 to 49	1 MHz	I/O (IOCLK)

*Standard Configuration

Table 8-1
CLOCK RATE CONFIGURATION

An open-collector driver (U59) buffers the divide-by-two output (8-MHz) for jumper selection of MULTIBUS signals BCLK/ and/or CCLK/. For the Z80 processor, the selected clock is inverted and then applied to the processor. An active pull-up Q1 minimizes the rise-time delay of the clock due to heavy, internal-capacitive loading of the Z80. The 2- and 1-MHz clocks are available for the user to choose from for the time base of the interval timer. Also, either the 2- or 4-MHz clock is available for the APU operation.

8.3 SYSTEM OPERATION

Essentially, all Z80 cycles are a series of basic operations that include:

1) Memory Read or Write
2) I/O Read or Write

Normally, these operations require three to six clock cycles or T states. However, they can be lengthened to synchronize the Z80 with the speed of a peripheral. Each Z80 cycle is referred to as a M (Machine) cycle; and the first machine cycle (M1) of any instruction is the Fetch cycle. The following paragraphs detail how the MSC 8009 handles normal operations.

8.3.1 Wait Operation

A "low" on WAIT/ indicates to the Z80 that the addressed memory or I/O device is ready for a data transfer. While WAIT/ is active (low), the processor executes a Wait operation. The Wait operation can be terminated by one of several ways.

1) Watchdog Timer
2) Assertion of SAACK/ during either a read or write operation
3) Assertion of SAACK/ during either a M1 or Op Fetch cycle

For an Interrupt Acknowlege cycle, WAIT/ can not be released. This is achieved only when M1/ goes "low" (M1 cycle), causing INTA/ to go "low" (U43, pin 11), releasing WAIT/.

8.3.1.1 Watchdog Timer

If the Z80 initiates a data-transfer operation, and the selected memory or I/O device does not respond with an acknowledge, the processor could remain in a Wait state indefinitely. The Watchdog Timer (U58) forces the Wait condition to terminate within 3 milliseconds, preventing the occurance of an indefinite Wait state. When an instruction begins, either MRQ/ (Memory Request), IORQ/ (I/O Request), or EXTRQ (DMA) goes "low", triggering the Watchdog Timer. If neither XACK/ or AACK/ is received within the specified time period, pin 4 or U37 (timer output) goes "low", forcing WAIT/ "high". The Z80 can now proceed with another operation. If an acknowledge signal is received from the MULTIBUS, the Wait state is terminated immediately; and the Z80 can now execute the requested instruction. The beginning of each instruction triggers U58, causing pin 4 of U37 to remain "high".

8.3.2 OP Fetch Cycle

Each Fetch cycle requires five clock cycles. At the beginning of 1 2Fetch cycle, the address lines A0 thru AF contain the output levels of the Z80 Program counter (PC). At one-half clock cycle time later, MRQ/ (Memory Request) goes "low" (active), indicating that the data on the address bus is valid for a memory cycle. In turn, this signal activates the State Machine PROM (U32 and U33) to set up the memory address lines and prepare the memory for a read cycle. Simultaneously, RD/ goes "low" (active) to indicate that the Z80 wants data from the addressed location in memory.

During state T2 or the second clock cycle, pins 1 thru 6 of U30 forces WAIT/ "low" (Wait state) as a result of U42's gating action. At this time, the addressed memory is not ready for a data transfer. When SAACK/ goes "low" (Refer to paragraph 8.3.4), the clock signal sets U30, terminating the Wait state.

As soon as the information on the data bus (D0 thru D7) is received, the Z80 clears RD/ and MRQ/ on the rising edge of state T3. Since the major requirements of the Fetch cycle has now been fulfilled, it is desirable to execute a Refresh cycle during the T3 and T4 clock cycles. At this time, the refresh is transparent (Refer to paragraph 4.2.4).

8.3.3 Memory Read Or Write

A memory read or write cycle requires three clock periods unless WAIT/ is activated (low). Signals MREQ/ and RD/ perform the same functions for a read operation as described in the Fetch cycle. During the memory-write cycle, MREQ/ also becomes active one-half cycle after the leading-edge of T1. When the data bus or I/O BUS becomes stable (State T2), WT/ becomes active (low). The addressed memory location now receives data from the I/O BUS under control of U67. One-half cycle prior to the termination of State T3, MREQ/ and WT/ are reset, completing the Write cycle.

8.3.3.1 Address Decoding

The MSC 8009 uses a 512 X 8 PROM (U54) for address decoding and two, three-to-eight line decoders (U21 and U27) to generate all memory and I/O chip-select signals. The eight, high-order bits of the Address Bus (LA8 thru LAF) address the PROM that produces from these input lines two, four bit fields. One field selects the memory devices; and the other, the I/O devices (Refer to Table 4-2). Decoders U21 and U27 accept the PROM outputs and generate the select signals as required by the memory and I/O devices. One decoder (U21) supplies chip select CS0 thru CS5 to the I/O, APU and disk formatter/controller and the other decoder (U27) selects the requested ROM (ROMSEL1 thru ROMSEL4). Two bits from memory and one bit from the I/O field determine if the address is a valid on-board memory or I/O address (RAMSEL, ROMSEL and IOSEL).

The Address Decode PROM can be programmed to encode any combination of the eight, most-significant bits of the address into another bit pattern. This allows the memories to be mapped such that they can originate at any arbitrary 256-byte page boundary in the addressing space. Depending on the storage capacity, low-order address bits are routed to the appropriate memory element for individual word selection.

8.3.4 I/O Cycles

One Wait State (WAIT/ is active) is automatically inserted for an I/O operation. The purpose of this action is to provide sufficient time for an I/O port to decode its address, permitting the I/O devices to operate at Z80 speed. If another Wait is required, WAIT/ is again activated.

During State T2 of the I/O cycle, IORQ/ goes "low" (active). The Program PROM U38 accepts IORQ/ in conjunction with either RD/ or WT/, and generates either IORC/ (I/O Read) or IOWC/ (I/O Write) respectively. Signal IORC/ causes data to flow from the MULTIBUS onto the I/O BUS via the addressed I/O port. For I/O write, IOWC/ reverses the data flow through the selected I/O port.

8.3.4.1 Arithmetic Processor Cycles

Essentially, the APU (U15) acts as another I/O port when PROM U54 and the decoding device U21 generates chip-select signal CS1. The Arithmetic Processor then performs the requested operation that is represented by the encoding of LA0, LIORC, and LIOWC (See paragraph 8.2.5).

8.3.5 Disk Formatter/Controller

The formatter/controller device U321 and associated circuits perform all functions that are necessary for transferring data to and from the disk drive. At the beginning of each instruction, the combination of INTRQ (Interrupt Request) and DRQ (Data Request) from the formatter/controller will pull SRQ/ "low", causing PAUSE/ line to go "low".

8.3.5.1 Register Selection

There are five registers internal to the formatter/controller chip and one external that are involved with the various disk operations. Whether the internal or the external registers are used depends upon the signal level of either CS5/ or CS6/.

Drive selection update is accomplished when CS6/ and LIOWC/ go "low" -- loading the data on the I/O BUS into the eight latches of U309. The outputs of these latches are then sent to the designated drive (U0 thru U3) to define the head (HSL), recording density (DDEN), drive size (8"/) and write precomp (COME/). If LIOWR/ is "low" instead of LIOWC, the outputs of the U309 latches are loaded into U316 and then asserted onto the I/O BUS.

The Chip-select signal CS5/ in conjunction with either LIOWC/ (write) or LIORC (read) set up the formatter/controller for register operation. Address signals LA0 and LA1 select one of the five registers that will be used, depending on the operation. The interpretation of these signals is as follows:

LA1	LA0	Read (LIORC/)	Write (LIOWC/)
0	0	Status Register	Command Register
0	1	Track Register	Track Register
1	0	Sector Register	Sector Register
1	1	Data Register	Data Register

8.3.5.2 Clock Circuit

The correct clocks needed for writing and reading are generated by the multiplexing action of U317. The following list summarizes these clocks and their respective application.

DRIVE	RECORDING	CONTROLLER CLOCK	RECOVERY CLOCK
8 inch	Single Density	8 MHz	2 MHz
8 inch	Double Density	16 MHz	2 MHz
5 inch	Single Density	8 MHz	1 MHz
5 inch	Double Density	4 MHz	1 MHz

8.3.5.3 Data Exchange

The data exchange between the Z80 processor and the formatter/controller takes place on the I/O BUS -- I00 thru I07. During a write, the formatter/controller accepts and converts the parallel data on the I/O BUS into a serial format (WD) consisting of data, clock and unique address marks. For a read, the formatter/controller accepts the serial data on the RAW RD/ line (pin 27 of U321) and then transfers it onto the I/O BUS in parallel.

8.3.5.4 Disk Read

WARNING: Portions of this circuitry is PATENT PENDING.

During a read operation, RAW RD/ (U321, pin 27) indicates a flux reversal; and RCLK (U321, pin 26) represents the flux-reversal spacing. It is important that these signals are properly phased. In the MSC 8009, a counter-separator concept is used to achieve the necessary phasing.

The major components making up the counter-separator circuitry are U301 and U302. The last-stage output of U302 (pin 11) in effect is RCLK; and the other three stages of the counter provide the input gating for U301. When there are no flux changes, U302 will free run at the clock rate as set up by U317.

When a flux transition occurs, RAW RD/ will go "low" for two clock periods. Simultaneously, pin 12 of U301 will be "low", preventing U302 from counting. After two clock cycles, U302 is then allowed to continue counting, producing RCLK transitions.

8.3.5.5 Disk Write

The falling edge of WD will cause pin 7 of U318 to go "low"; and the outputs of the U312 latch configuration will be loaded into U310 (Parallel Access Shift Register). Following a short delay, pin 10 of U313 will go "low" -- resetting U318 and terminating the loading of U310. The pattern of these levels that are loaded into U310 determines the delay of WD/ sent to the drive. This delay is established by serially shifting the loaded pattern through U310 at a clock rate of either 4-MHz (jumper E6 to E5) or 8-MHz (jumper E6 to E7). For a normal write operation, COMPE/ will be "low", placing a "high" on pins 5 and 6 of U310, and a "low" on pins 7 and 4. This will enter the logic pattern of "0110" into the shift register.

The following circuit conditions are then set up via the latching action of U312 (pins 1 thru 6)

EARLY	LATE	DELAY
0	0	△
1	0	No Delay
0	1	△(2)
1	1	Illegal

where △(2) represents a specified delay. In turn, each of these conditions will enter a different pattern into U310.

8.3.5.6 Head Loading Delay

When a "low" is sent to the selected drive on HEAD/, the head will be loaded against the disk surface. There is an internal 15-millisecond delay within the formatter/controller chip so that the head can be fully engaged. If more time is required, it must be programmed through software. In other words this is accomplished by setting up Counter 0 of the 8253 Programmable Timer for MODE 2 operation (Refer to Section 5).

When bit 2 of the command word is zero for either a read or write operation, the built in 15-millisecond delay is disabled; and HLT is sampled immediately for a logic "1" from the 8253 (pin 10 of U17). The "high" on HLD will prevent a logic "1" from appearing on HLT until the preset pulse duration expires.

The clock for this operation is derived from U303, which accepts and divides the 1-MHz system clock by thirty-two. This concept allows the MSC 8009 to accomodate full-size disks (50 milliseconds) to mini drives (1.5 seconds).

8.3.6 Acknowledge Cycle

When either the MSC 8009 or some other device on the MULTIBUS initiates a data transfer, that device will assert either XACK/ or AACK/ to acknowledge the receipt of or the placement of data on the bus. Either signal terminates the Wait state, which releases the Z80 for another transaction.

A "low" on pin 9 of U45 indicates a RAM request, allowing the assertion of MACK/ to cause pin 9 of U31 to go "high". This starts the timing cycle for generating the acknowledge signals AACK/, XACK/ and SAACK/. For a ROM or I/O (LIORQ) request, either pin 7 or 9 of U45 will go "low" to initiate the acknowledge-signal timing immediately.

8.3.7 Interrupt Request

An eight-bit vector (INT0/ thru INT7/) from the MULTIBUS is applied to the Priority Interrupt Control circuit U62 (8214). This unit determines which input has the highest priority with INT7/ being assigned the lowest. Output INT/ goes "low", resetting latch U30. This action forces INT/ to the Z80 "low", initiating an interrupt sequence.

The Z80 samples INT/ (Interrupt) with the rising edge of the last clock that occurs during the completion of any instruction. This signal is accepted only when software control has not set the internal interrupt enable flip-flop of the Z80. During the Interrupt Request/Acknowledge cycle, the Z80 activates M1 concurrently with IORQ/. The Wait condition of the MSC 8009 is terminated immediately; and the Z80 resumes operation following two Wait states that the Z80 inserted during the Request/Acknowledge cycle. The Interrupt vector PROM at this time sends the Interrupt vector to the Z80.

8.3.7.1 Non-Maskable Interrupt

A unique feature of the Z80 is the Non-Maskable Interrupt or NMI, which is not a MULTIBUS signal. In the standard MSC 8009, the NMI/ signal is normally found on pin 33 of P1 or pin 4 of P2. This signal is useful for such purposes as power fail. If NMI/ conflicts with another application that may use pin 33, the jumper between posts 70 and 71 must be removed. The Watchdog Timer performs NMI by applying the signal LNMI/ to the Z80 via jumper 64 to 65. If this feature is not needed; or there are uses for the NMI feature, remove this jumper.

The Z80 samples NMI/ at the same time as the interrupt line INT/. However, NMI line has the highest, interrupt priority; and it can not be disabled under software control. The Z80 response to NMI is similar to a memory-read operation. The only difference is that the contents of the data bus is ignored while the Z80 automatically stores the data contents of the Program Control register and jumps to location 0066 HEX (CALL 0066H instruction). The service routine for NMI must start at this location, if NMI is to be used.

As previously mentioned,jumper 64 to 65 must be installed if the Watchdog Timer is to be used to generate NMI. If U58 is not retriggered within 10 milliseconds, the level (LNMI/) 13 of U58 will be inverted to generate the required negative-going edge for the Z80 NMI input.

8.3.8 HALT Request

Whenever the MSC 8009 receives the software instruction HALT (OP Code 76), the Z80 internally executes NOP instructions until an interrupt is received. If an interrupt (either NMI or INT) is received, and the internal Interrupt Enable flip-flop of the Z80 is set; the rising edge of the next clock will terminate the HALT operation. In turn, the next cycle will be an interrupt acknowledge corresponding to the type of interrupt that was requested. The purpose for the NOP operation is to provide MRQ so that DMA is still permitted. Each cycle during the HALT operation is a normal M1 cycle (Fetch) except that data read from the memory is ignored.

PIN NO.	SIGNAL	FUNCTION
6	EXTERNAL/	Clocks BREQ latch U50 when the MSC 8009 wants control of the MULTIBUS.
7	LREQ/	Sets up counter U31 for XACK and generates AACK/.
8	LIORQ/	Used for I/O operation to double AACK/
9	RMRQ/	Signifies that a memory cycle has been requested.
11	ACKEN/	Enables line drivers (U60) to apply AACK and XACK onto the MULTIBUS.
12	LOCMEM/	Indicates that Z80 is trying to access local RAM.
13	EXTDB/	Controls the direction of data flow between the MULTIBUS and the MSC 8009.
14	RMRD/	Enables local RAM data latches onto the I/O BUS.

Table 8-2
SYSTEM CONTROL PROM SIGNAL IDENTIFICATION

8.3.9 System Reset

When power is applied to the system, either a 556 (U70) generating an 100-millisecond pulse (approximately) via jumper 60 to 61; or INIT/ asserted onto the MULTIBUS by another device will initialize the MSC 8009. The signal INIT/ conditions the Z80 as well as the circuits of the MSC 8009 for immediate operation. The buffered signal via jumper 60 to 61 also can be used to reset all other devices that may be sharing the MULTIBUS.

8.4 SYSTEM CONTROL

The System Control PROM U45 accepts bus-control signals PROCA (Processor Address), LMRDC/ (Local Memory Read), LMWTC/ (Local Memory Write), LIORC/ (Local I/O Read) and IOWC/ (Local I/O Write), as well as the select signals RAMSEL/, ROMSEL, and IOSEL/. This PROM accepts and converts these signals to the eight control levels listed in Table 8-2. As an incidental function, U43 (pins 4, 5 and 6) combines CS4/ with LIOWC/ to generate LOAD/ -- the strobe for the 8214 Interrupt Controller U62 and Bus Mode Latch U22.

8.4.1 Bus State Machine

Two PROM (U32 and U33) and two octal latches (U39 and U40) control the data-transfer functions to and from the MULTIBUS. The following three signals in addition to the next state address (U32, pins 6 thru 8) MRQ/ and IORQ/ set up the state machine.

1) EXTERNAL/ is PROM generated (U45) and denotes that the address information on L/ABUS is invalid for MSC 8009 addressing.

2) EXTRQ/ becomes active or "low" when another MULTIBUS master is requesting access to the MSC 8009 memory or I/O.

3) BGRNT/ indicates that the MSC 8009 has MULTIBUS control.

The signals in this list are generated by the bus state machine:

1) PROCA signifies that the Z80 has control of the local buses. PROCA inactive indicates a DMA cycle.

2) LIOAD/ permits the lower-eight bits of address to be transferred to the upper-eight bits of the local-address bus.

3) LMADR/ sends upper-eight bits of the Z80 address to the upper 8-bits of the local-address bus.

4) EXTADR/ enables local-address bus and MULTIBUS.

5) BDSMS is used to release the MULTIBUS control (See paragraph 8.4.1.1).

6) LOCC/ allows the Z80 to generate local memory or I/O commands.

7) EXTC/ lets the local command bus assert the MULTIBUS command signals.

8) COUT/ places the output lines of the 8833 drivers (U44) at a "high" impedence, inhibiting the MULTIBUS command signals from being asserted onto the internal command bus.

8.4.1.1 Bus Exchange

After LOAD/ latches the level status of IO4 and IO5, the outputs on pins 2 and 14 of U22 define one of three bus modes (Refer to paragraphs 1.3.4.1). If both bits are "low", signal BDSMS (Bus Dismiss) from the state machine forces both pin 9 of U50 and pin 6 of U50 ". high" This is achieved through the gating action of U42, pins 4 thru 6. In turn, BPRO/ is asserted and BUSY/ unasserted, releasing the MULTIBUS control to another device. This same condition is set up when IO5 is "high"; and IO4 is "low", except M1/ is the controlling input of U29, pins 11 and 13. The last condition when IO4 is "high" holds pin 6 of U42 "low". In turn, BRPO/ remains "high", and BUSY/ is "low", allowing the MSC 8009 to keep control of the MULTIBUS.

8.4.2 Memory State Machine

A PROM (U69) and an octal latch (U71) make up the memory state machine. To start either a read or write cycle, LOCMEM/ (request from the bus state machine) or RD/ (request from the processor) is asserted in conjunction with a memory request (RMRQ/). A "low" is then applied to pin 10 of U69 via U36 (pin 1 thru 13) forcing pin 9 of U71 to go "high". The low-to-high transition triggers U58, preventing a double memory cycle (U36, pin 13 goes "high") as well as generating MACK/ for the acknowledge operation (Refer to paragraph 8.3.4). The signal on pin 12 of U71 is sent to the memory controller U67; and the requested memory cycle begins. This controller generates the necessary signals as required by the dynamic memory array. If the requested memory cycle is a read, pin 9 (ROWE1) gates the data from the addressed location into data latches U41, 49, 57 and 66.

The memory state machine becomes an arbitrating circuit in event that a refresh and a normal memory request occurs simultaneously. Priority is given to the requested memory cycle and allows that cycle to finish before another cycle begins. If the requested cycle is a read, the data is latched into the data latches, and then the system goes into a refresh.

8.4.2.1 Refresh

The refresh cycle is independent of other device or signal controls. If no memory cycle is in progress, the memory controller U67 asserts RFRQ/, requesting a refresh and causing pin 6 of U71 to go "low". In turn, the memory state machine responds with RFG; and the rising edge on pin 5, U70 initiates the Refresh cycle. Delay line U68 provides the timing signals as required by U67 to perform a refresh.

SECTION 9

ARITHMETIC PROCESSOR UNIT

9.1 SCOPE

This section presents the benefits and summarizes the command structure of the Arithmetic Processor Unit (APU). Two optional APU's are available -- the 9511 and the 9512. The 9511 provides 32-bit floating-point and fixed-point arithmetic operation including transcendental functions. For precise applications, the 9512 64-bit floating-point APU can be used.

9.2 CAPABILITIES

The Z80 transfers data to and from the APU (U15) using conventional I/O programming techniques. If the APU is busy and the Z80 tries to either access data or send another command, the APU asserts PAUSE/ forcing the Z80 to wait. When the APU completes the command in progress, the APU then accepts the new information and releases the Z80 for other transactions. This simplifies programming because successive operations can be performed without status checks. However, keep in mind that while the Z80 is waiting on the APU, the processor cannot perform an interrupt. If the application requires interrupts, the APU status register must be monitored and no attempt made to access data or send another command to the APU when it is busy. The additional programming consideration ensures that an interrupt will be serviced promptly. Otherwise, delays up to six milliseconds can be encountered.

9.2.1 I/O Addressing

Data is transferred to and from the APU on two I/O addresses. These addresses are:

	READ	WRITE
D4	Data	Data
D5	Status	Command

9.3 9511 ARITHMETIC PROCESSOR UNIT

9.3.1 Initialization

When power is first applied to the MSC 8009, the internal stack pointer may not be properly aligned for the correct word boundary. The procedure for correcting this situation is to load two bytes (16-bits) of data representing a numerical "1" (See Figure 9-1). In other words, the first byte will be "00000001"; and the second is eight zeros. Then perform a 16- to 32-bit floating-point conversion using the FLTS instruction (1DH) on this 16-bit word. Next, a read instruction is performed to determine whether the Top-Of-Stack (TOS) is a numerical "1" (00000001). If the byte is a numerical "1", another byte of data is read until the byte equals some other value. Now the stack is aligned for proper operation.

NOTE: The initialization procedure is also recommended when a software error unaligns the internal-stack pointer.

9.3.2 Stack Control

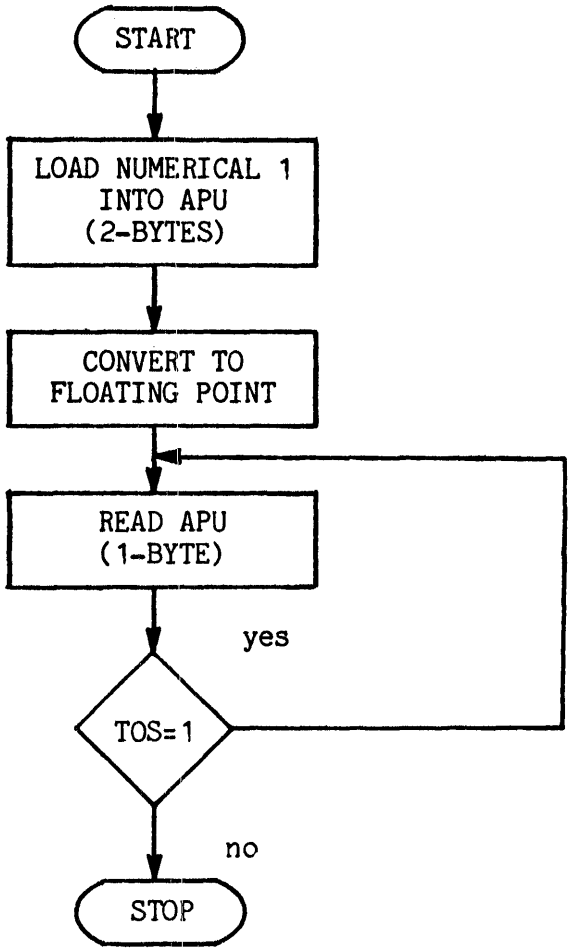
The I/O BUS interface with the 9511 includes access to an eight-level, sixteen-bit wide data stack that operates a push-down or FILO stack. Because fixed-point operands are sixteen-bits, eight such values may be maintained within the stack. For 32-bit operations (fixed or floating point), only four values can be stored. Data is transferred into the stack a byte at a time with the least-significant byte entered first and retrieved last. Thus, data should be moved to and from the 9511 in multiples that are equal to the number of bytes appropriate to the selected data format.

9.3.3 Data Format

The 9511 APU performs both 16- and 32-bit fixed-point operation. All fixed-point operands and results are in binary two's complement format. Numerical range for 16-bits is -32,768 to +32,767; and for the 32-bit format, the range is -2,147,483,648 to +2,147,483,647.

The floating-point format uses a 32-bit word with bit 31 (most-significant bit) indicating sign of the mantissa; bit 30 representing the sign of the exponent; bits 24 thru 29 forming the exponent, and bits 0 thru 23 providing the mantissa value.

Flow Diagram:



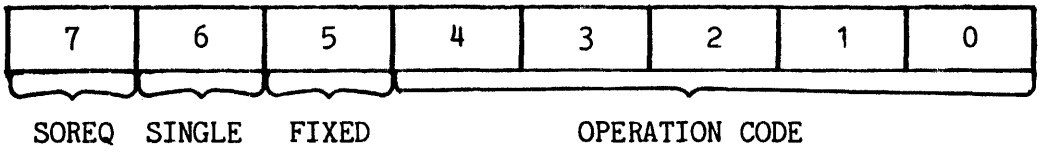
Source Program:

```
1   0000 ED A0          LDI   A,01H     ;BYTE 1 IN ACCUMULATOR
2   0002 D3 D4          OUT   0D4H      ;SEND OUT BYTE 1
3   0004 ED A0          LDI   A,00H     ;BYTE 2 IN ACCUMULATOR
4   0006 D3 D4          OUT   0D4H      ;SEND OUT BYTE 2
5   0008 EDA0           LDI   A,1DH     ;FLTS COMMAND IN
                                        ACCUMULATOR
6   000A D3 D5          OUT   0D5H      ;PERFORM FLTS
7   000C DB D4    HERE: IN    0D4H      ;TOS IN ACCUMULATOR
8   000E B9             CMP   01H       ;TOS=1?
9   000F 28 FB          JRZ   HERE      ;REREAD TOS IF "1"
```

Figure 9-1
9511 INITIALIZATION SEQUENCE

9.3.3.1 Command Format

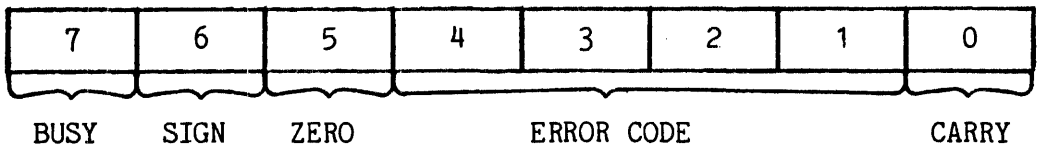
A single byte having the following format specifies the command that is to be executed.



Bits 0 thru 4 specify the operation; bits 5 and 6 identify the data format; and a "1" in bit 7 indicates that the command has been completed.

9.3.3.2 Status Register

An internal status register having the following format provides device status information to the MSC 8009.



If the BUSY BIT is a "1", the other status bits will not be defined. These bits as defined are valid only when BUSY is "0" -- the operation is complete.

SIGN (Negative = 1) represents the sign of the value that is contained in the top of the stack.

ZERO (value is zero = 1) denotes the top of the stack value is zero.

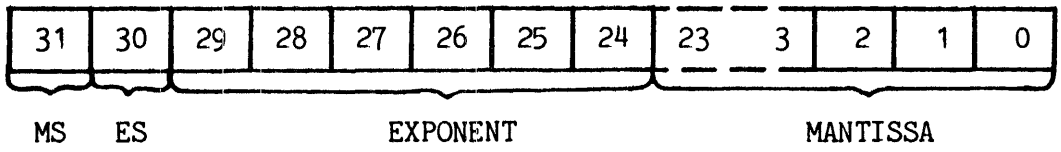
ERROR CODE shows the validity of the last operation based on the following codes:

0000 - no error
1000 - divide by zero
0100 - square root or log of a negative number
1100 - argument of inverse sine or cosine, or e^x too large
XX10 - underflow
XX01 - overflow

CARRY (carry/borrow = 1) indicates that the previous operation resulted in either a carry or borrow from the most-significant bit.

9.3.3.3 Floating Point Format

The range of values that the following format will represent is $+2.7 \times 10^{-20}$ to 9.2×10^{18} including zero.



The 32-bit floating-point value is made up of a 24-bit fractional value (mantissa); a 7-bit exponent value including sign (ES), expressed as a two's complement having a range of -64 to +65; and the most-significant bit representing the sign of the mantissa (positive is "0" and negative is "1") for a total of 32 bits. The binary point is assumed to be to the left of the most-significant mantissa bit (bit 23). All floating-point data values must be normalized. Bit 23 must be a "1", except for the value zero, when all bits are zero.

9.4 9511 INSTRUCTIONS

To make the following information easy to use, a shorthand notation will be employed to describe each instruction and its operation. For instance a single, upper-case letter will represent a sixteen-bit quantity, while a double, upper-case letter will denote a thirty-two bit quantity. A value enclosed in parenthesis under NOTATION will indicate floating-point. The following symbols and abbreviations will be used to describe the instruction set in the following tables and paragraphs.

A	TOS quantity prior to executing the instruction
B	NOS quantity prior to executing the instruction
BOS	Bottom Of Stack
C	Carry bit of the Status register
CC	Clock cycle
EF	Error field of the Status register
NOS	Next Of Stack
PG	Section 9 page numbers
R	Result of the executed operation
S	Sign bit of the Status register
SC	Stack content as defined in Table 9-1.
TOS	Top Of Stack
X	Status register bit is affected
*	Status register bit is unaffected
w<=v	Quantity "w" is replaced by quantity "v"
w<=#v	Quantity "w" is equal to the complement of the quantity "v"
w<=>v	Quantities "w" and "v" are exchanged.

9.4.1 Data And Stack Manipulation Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z ER C
CHSD	15	TOS<=0-AA	26-28	D	X X X *
CHSF	16	TOS<=#(AA)	16-20	D	X X X *
CHSS	16	TOS<=0-A	22-24	M	X X X *
FIXD	21	AA<=(AA)	90-336	G	X X X *
FIXS	21	A<=(AA)	90-214	O	X X X *
FLTD	22	(AA)<=AA	56-342	G	X X * *
FLTS	22	(AA)<=A	62-156	Q	X X * *
NOP	25	No Operation	4		* * * *
POPD	25	TOS<=BB BOS<=AA	12	B	X X * *
POPF	25	TOS<=(BB) BOS<=(AA)	12	B	X X * *
POPS	27	TOS<=B BOS<=A	10	P	X X * *
PTOD	26	TOS<=AA NOS<=AA	20	A	X X * *
PTOF	26	TOS<=(AA) NOS<=(AA)	20	A	X X * *
PTOS	27	TOS<=A NOS<=A	16	R	X X * *
PUPI	27	TOS<=π NOS<=(AA)	16	F	X X * *
XCHD	33	(AA)<=(BB)	26	C	X X * *
XCHF	33	AA<=(BB)	26	C	X X * *
XCHS	33	A<=B	18	L	X X * *

9.4.2 16-Bit Fixed-Point Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z ER C
SADD	29	TOS<=A+B	15-18	N	X X X *
SDIV	29	TOS<=B/A	84-94	S	X X X *
		If A=0	14	S	X X X *
SMUL	30	TOS<=[A X B]	84-94	S	X X X *
SMUU	31	TOS<=[A X B]	80-98	S	X X X *
SSUB	32	TOS<=B-A	30-32	N	X X X *

9.4.3 32-Bit Fixed-Point Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z ER C
DADD	17	TOS<=AA+BB	20-22	E	X X X X
DDIV	17	TOS<=BB/AA	196-210	H	X X X *
		If AA=0	18	H	X X X *
DMUL	18	TOS<=[AA X BB]	194-210	H	X X X *
DMUU	18	TOS<=[AA X BB]	182-218	H	X X X *
DSUB	19	TOS<=BB-AA	38-40	E	X X X X

9.4.4 32-Bit Floating-Point Primary Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z ER C
FADD	20	TOS<=(AA)+(BB)	54-368	H	X X X *
		If (AA)=0	18	E	X X X *
FDIV	20	TOS<=(BB)/(AA)	154-184	H	X X X *
		If (AA)=0	22	H	X X X *
FMUL	23	TOS<=(AA) x (BB)	146-168	H	X X X *
FSUB	23	TOS<=(BB)-(AA)	70-370	H	X X X *

9.4.5 32-Bit Floating-Point Derived Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z ER C
ACOS	14	TOS<=cos (AA)	6304-8284	J	X X X *
ASIN	14	TOS<=sin (AA)	6230-7938	J	X X X *
ATAN	15	TOS<=tan (AA)	4992-6536	I	X X * *
COS	16	TOS<=cos (AA)	3840-4878	I	X X * *
EXP	19	TOS<=e	3794-4878	I	X X X *
		If |(AA)|>1x2	34	I	X X X *
LOG	24	TOS<=log (AA)	4474-7132	I	X X X *
		If (AA)<0	20	I	X X X *
LN	24	TOS<=ln (AA)	4298-6956	I	X X X *
		If (AA)<0	20	I	X X X *
PWR	28	TOS<=(BB)	8290-12032	K	X X X *
SIN	30	TOS<=sin (AA)	3796-4808	I	X X * *
		If |(AA)|<2	30	I	X X * *
SQRT	31	TOS=/(AA)	782-870	G	X X X *
TAN	32	TOS<=tan (AA)	4894-5886	I	X X X *
		If |(AA)|<2	30	I	X X X *

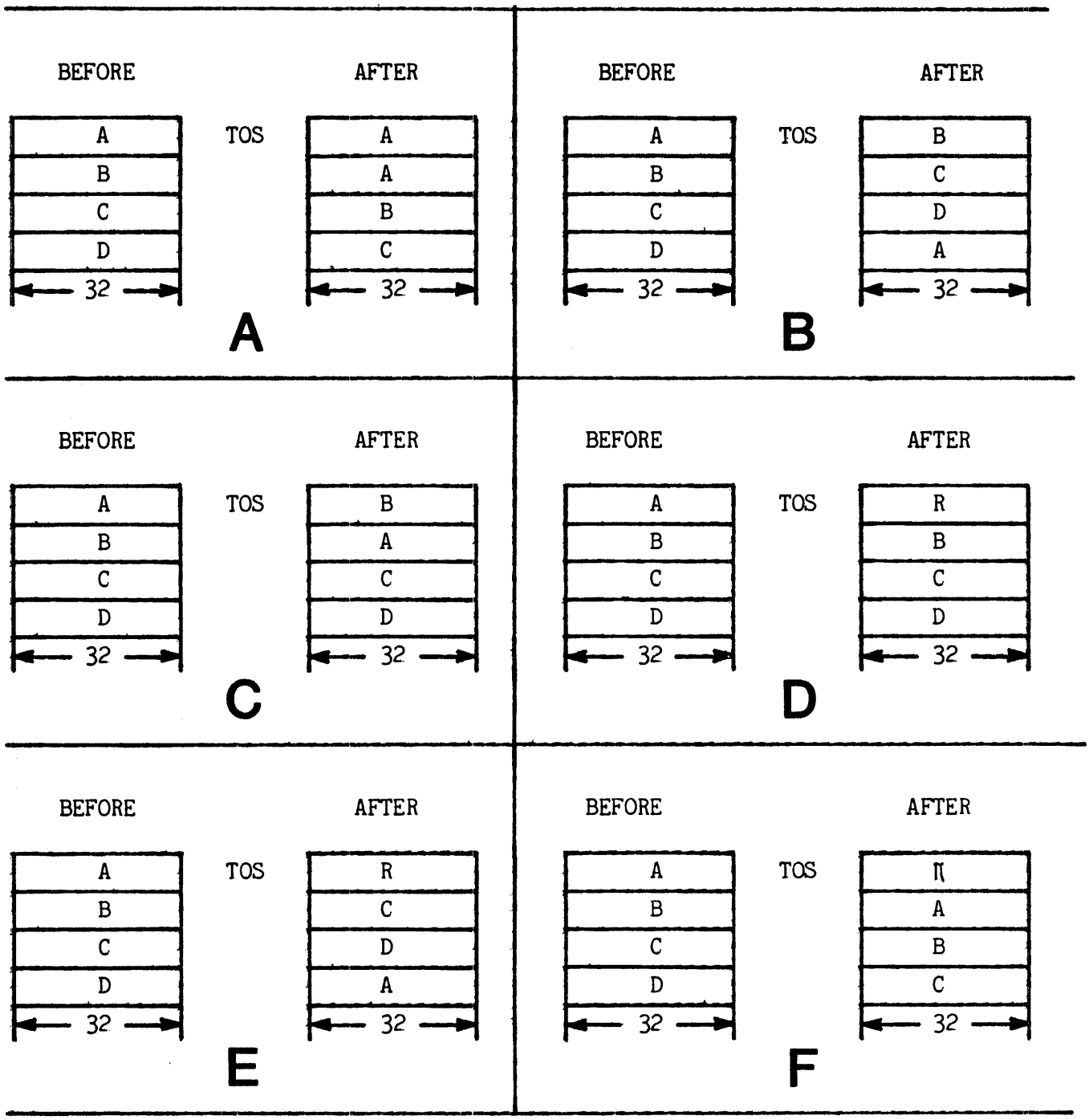


Table 9-1
STACK CONFIGURATIONS

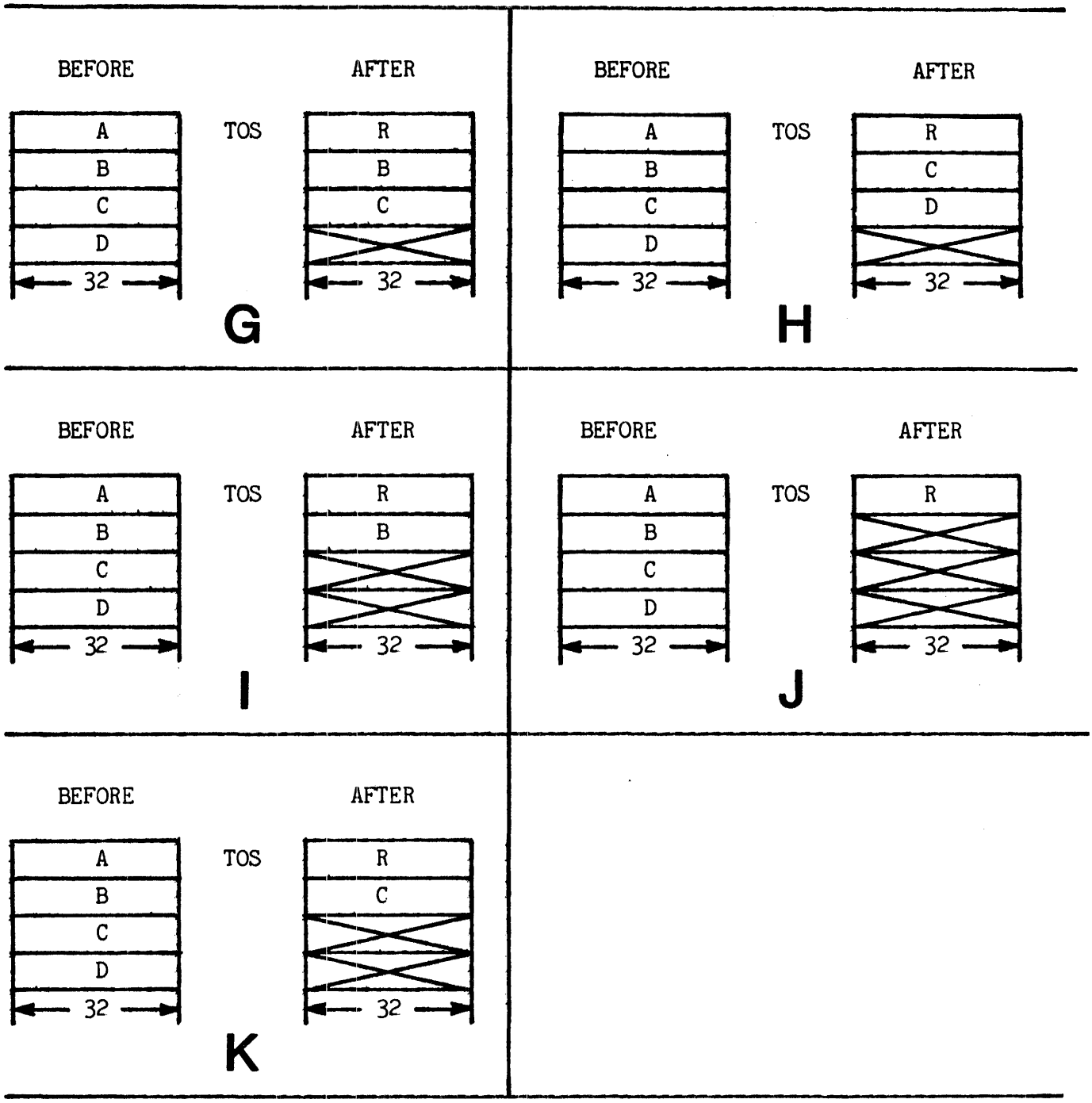


Table 9-1 (cont'd)
STACK CONFIGURATIONS

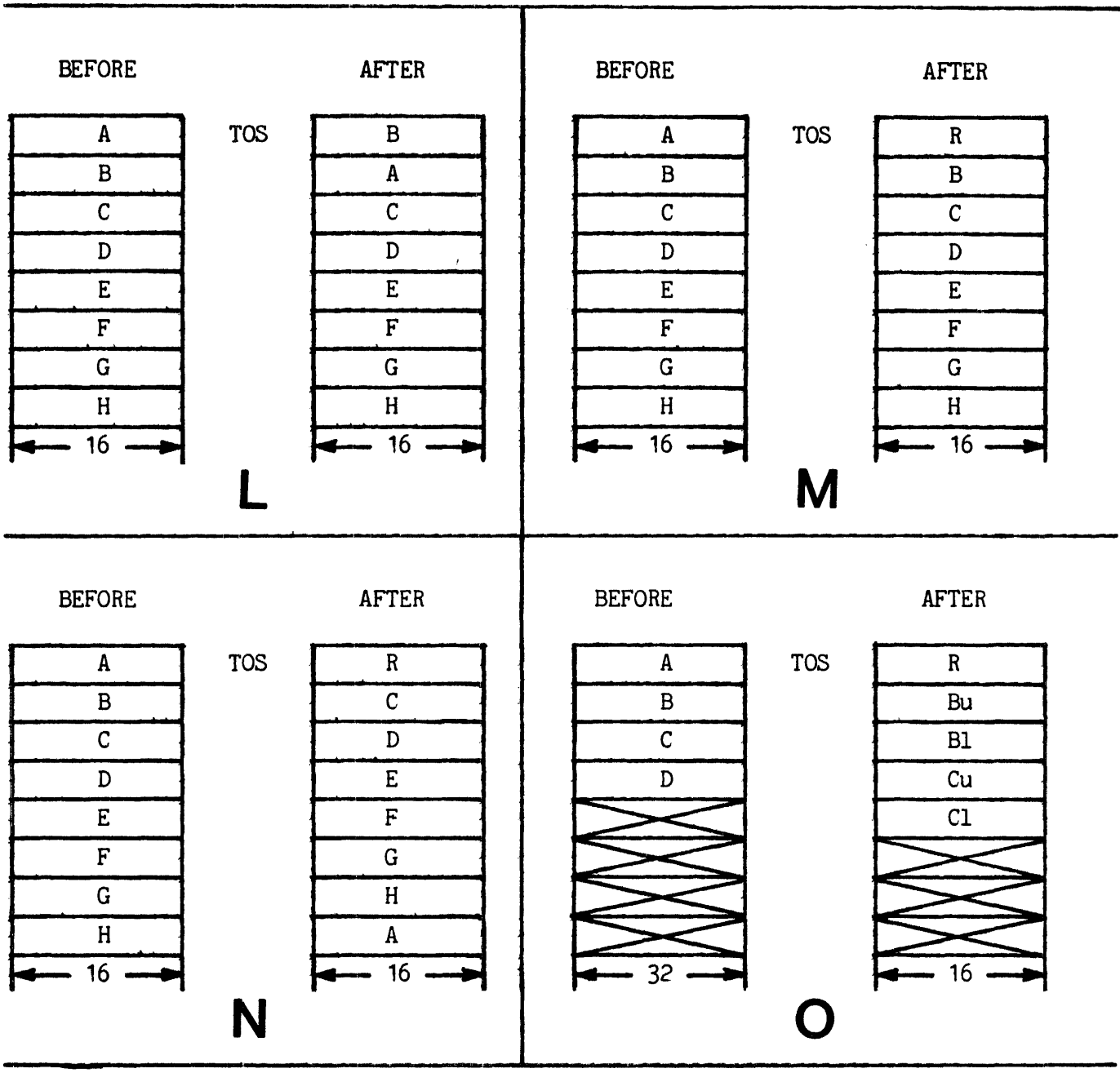


Table 9-1 (cont'd)
STACK CONFIGURATIONS

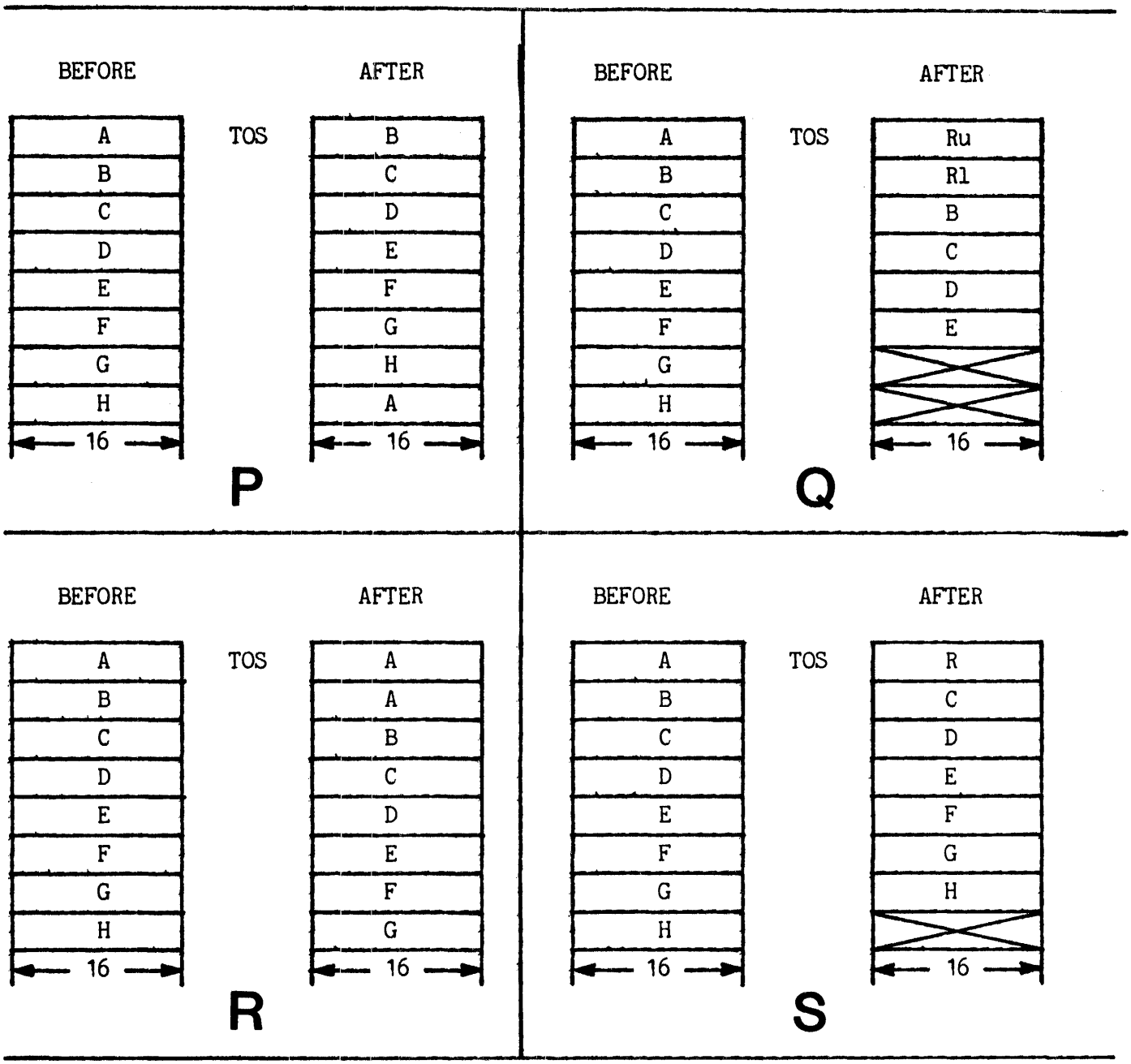


Table 9-1 (cont'd)
STACK CONFIGURATIONS

ACOS 32-Bit Floating-Point Inverse Cosine

The 32-bit floating-point operand A at TOS is replaced with the 32-bit floating-point inverse cosine of A; and the other initial operands in the stack are lost. The resultant value is in radians ranging from zero to 'PI'. All input-data values within the limits of -1.0 to +1.0, will be accepted. Values outside this range generate an error code of "1100" in the error field of the Status register.

86/06

Accuracy: Maximum relative error of 2×10^{-7} over the valid-input range.
Clock Cycles: 6304 to 8284
Status: S, Z, EF
Stack Contents: J

ASIN 32-Bit Floating-Point Inverse Sine

The 32-bit floating-point operand A at TOS is replaced with the 32-bit floating-point inverse sine of A; and the other initial operands in the stack are lost. The resultant value is in radians ranging from $-\pi/2$ and $+\pi/2$ radians. All input-data values within the range of -1.0 to +1.0 will be accepted. Values outside this range generate an error code of "1100" in the error field of the Status register.

85/05

Accuracy: Maximum relative error of 4×10^{-7} over the valid-input range.
Clock Cycles: 6230 to 7938
Status: S, Z, EF
Stack Contents: J

ATAN 32-Bit Floating-Point Inverse Tangent

The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point inverse tangent of A; and operand B (NOS) is unchanged. All other initial operands are lost. The resultant value is in radians ranging from $-\pi/2$ to $+\pi/2$. All input-data values must be in floating-point format.

87/07

Accuracy: Maximum relative error of 3×10^{-7} over the input-data range.
Clock Cycles: 4992 to 6536
Status: S, Z
Stack Contents: I

CHSD 32-Bit Fixed-Point Sign Change

The 32-bit fixed-point two's complement integer A at TOS is subtracted from zero; and the resultant R value replaces A at TOS. Other entries in the stack are not changed. Since no positive equivalent exists when integer A is input as the most-negative value possible, an overflow condition (XX01) will be set up in the error field of the Status register; and integer A is then reentered into TOS unchanged.

B4/34

Clock Cycles: 26 to 28
Status: S, Z, EF(overflow)
Stack Contents: D

CHSF 32-Bit Floating-Point Sign Change

The sign for the mantissa of the 32-bit floating-point operand A at TOS is inverted. The result replaces A at TOS. Other stack entries are unchanged. If A is input as zero (mantissa most-significant bit is zero), no change takes place.

95/15

Clock Cycles: 16 to 20
Status: S, Z
Stack Contents: D

CHSS 32-Bit Fixed-Point Sign Change

The 16-bit fixed-point two's complement integer operand A at TOS is subtracted from zero. The result replaces A at TOS. All other operands are unchanged. Since no positive equivalent exists when integer A is input as the most-negative value possible, an overflow condition (XX01) will be set up in the error field of the Status register; and the integer A is then reentered into TOS unchanged.

F4/74

Clock Cycles: 22 to 24
Status: S, Z, EF(overflow)
Stack Contents: M

COS 32-Bit Floating-Point Cosine

The 32-bit floating-point operand A at TOS is replaced by a 32-bit floating-point cosine of A; and the resultant value will be in radians. All initial operands on the stack are lost except B (NOS), which remains unchanged. Any input-data value that fulfills the data format will be accepted. All input values are scaled to fall within the range of $-\pi/2$ to $+\pi/2$ radians.

83/03

Accuracy: Maximum relative error of 5.0×10 for all input-data values in the range of -2π to $+2\pi$ radians.
Clock Cycles: 3840 to 4878
Status: S, Z
Stack Contents: I

DADD 32-Bit Fixed-Point Add

The 32-bit fixed-point two's complement integer operand A at TOS is added to the 32-bit fixed-point two's complement integer operand B at NOS. The sum replaces operand B; and then moved up into the TOS, while operand A is transferred to the bottom of the stack unchanged. The other operands in the stack remain unchanged except operand B, which is lost. If a carry is generated, it will be recorded in the Status register. If the result is too large and cannot be represented, the least-significant 32-bits of the sum are returned; and an overflow status (XX01) will be recorded.

AC/2C

Clock Cycles: 20 to 22
Status: S, Z, EF
Stack Contents: E

DDIV 32-Bit Fixed-Point Divide

The 32-bit fixed-point two's complement integer operand B at NOS is divided by the 32-bit fixed-point two's complement integer operand A at TOS. The 32-bit integer quotient replaces B; and then the stack is moved up so that the quotient will occupy TOS. A remainder will not be generated. Both operands A and B are lost; and the other operands in the stack remain unchanged. If A is zero, the quotient will be set to equal B; and a divide-by-zero error (XX01) will be designated in the Status register. If either A or B is the most-negative value possible, the quotient will be meaningless; and an overflow error (XX01) will be entered in the Status register.

AF/2F

Clock Cycles: 196 to 210 (A=0)
18 (A=0)
Status: S, Z, EF
Stack Contents: H

DMUL 32-Bit Fixed-Point Multiply, Lower

The 32-bit fixed-point two's complement integer operand A at TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at NOS. The 32-bit least-significant half of the product replaces B; and the stack is then moved up so that the value will occupy TOS. The most-significant half of the product will be lost as well as operands A and B. The other operands in the stack remain unchanged.

AE/2E

Clock Cycles: 194 to 210
Status: S, Z, EF (overflow)
Stack Contents: H

DMUU 32-Bit Fixed-Point Multiply, Upper

The 32-bit fixed-point two's complement integer operand A at TOS is multiplied by the 32-bit fixed-point two's complement integer operand B at NOS. The 32-bit most-significant half of the product replaces B; and the stack is then moved up so that the resultant will be at TOS. The least-significant half of the product is lost as well as operands A and B. All other original operands in the stack are not affected. If either A or B was the most-negative value possible, an overflow status is set into the Status register; and the product will be meaningless.

B6/36

Clock Cycles: 182 to 218
Status: S, Z, EF (overflow)
Stack Contents: H

DSUB 32-Bit Fixed-Point Subtract

The 32-bit fixed-point two's complement operand A at TOS is subtracted from the 32-bit fixed-point two's complement operand B at NOS. The difference replaces operand B; and the stack is moved up so that the differences will now be at TOS. Operand A is transferred to the bottom of the stack. All other operands in the stack are unchanged except B, which is lost.

AD/2D

Clock Cycles: 38 to 40
Status: S, Z, carry, EF (overflow)
Stack Contents: E

EXP 32-bit Floating-Point e^x

The base of the natural logarithm is raised to the exponent value that is specified by the 32-bit floating-point operand A at TOS. The result replaces A. All original operands of the stack are lost except B, which remains unchanged. All input-data values within the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$ will be accepted. Input values outside this range will cause a code of "1100" to be entered in the error field of the Status register.

8A/0A

Accuracy: Maximum relative error of 5.0×10^{-7} over the valid input-data range.

Clock Cycles: 3794 to 4878 ($|A| < 1 \times 2^5$)
34 ($|A| > 1 \times 2^5$)
Status: S, Z, EF
Stack Content: I

FADD 32-Bit Floating-Point Add

The 32-bit floating-point operand A at TOS is added to the 32-bit floating-point operand B at NOS. The result replaces B; and the stack is then moved up so that the sum occupies TOS. Operands A and B are lost; while the other operands in the stack remain unaffected. Exponent alignment before the addition and normalization of the result accounts for the variation in execution time. Both exponent overflow (XX01) and underflow (XX01) are recorded in the Status register. In either case, the mantissa is correct; and the exponent is offset by 128.

80/10

Clock Cycles: 54 to 24
Status: S, Z, EF
Stack Contents: H

FDIV 32-Bit Floating-Point Divide

The 32-bit floating-point operand B at NOS is divided by the 32-bit floating-point operand A at TOS. The quotient replaces B; and the stack is then moved up so that the quotient is at TOS. Operands A and B are lost, while the other operands in the stack remain unchanged. If operand A is zero, the result will be equal to B; and the divide-by-zero error (1000) will be recorded in the Status register. Either exponent overflow (XX01) or underflow (XX01) will be reported in the Status register. In either case, the mantissa portion of the result is correct; and the exponent portion is offset by 128.

93/13

Clock Cycles: 154 to 184 (A=0)
22 (A=0)
Status: S, Z, EF
Stack Content: H

FIXD 32-Bit Floating-Point to 32-Bit Fixed-Point Conversion

The 32-bit floating-point operand A at TOS is converted to a 32-bit fixed-point two's complement integer. The result replaces A. Both operands A and D (BOS) are lost, while the other operands are not affected. If the integer portion of A is larger than 32 bits when converted, an overflow error (XX01) will be recorded; and A will remain unchanged. However, operand D will still be lost.

9E/1E

Clock Cycles: 90 to 336
Status: S, Z, EF (overflow)
Stack Contents: G

FIXS 32-Bit Floating-Point to 16-Bit Fixed-Point Conversion

The 32-bit floating-point operand A at TOS is converted to a 16-bit fixed-point two's complement integer. The resultant value replaces the lower half of A; and the stack is then moved up two bytes so that the result is at TOS. Operands A and D (BOS) are lost, while operands B and C are unchanged. If B and C are 32-bit, they will appear as upper (u) and lower (l) halves on the 16-bit wide stack. If the integer portion of A is larger than 15 bits when converted, the overflow (XX01) status will be set; and A will remain unchanged. However, operand D will still be lost.

9F/1F

Clock Cycles: 90 to 214
Status: S, Z, EF (overflow)
Stack Contents: O

FLTD 32-Bit Fixed-Point to 32-Bit Floating-Point Conversion

The 32-bit fixed-point two's complement integer operand A at TOS is converted to a 32-bit floating-point number. The result replaces A at TOS. Operands A and D (BOS) are lost; and the other operands in the stack are unchanged.

9C/1C

Clock Cycles: 56 to 342
Status: S, Z
Stack Contents: G

FLTS 16-Bit Fixed-Point to 32-Bit Floating-Point Conversion

The 16-bit fixed-point two's complement integer A at TOS is converted to a 32-bit floating-point number. The lower-half of the result operand (Rl) replaces A; and the upper half (Ru) replaces H. The stack is then moved down so that Ru occupies TOS. Operands A, F, G, and H (see Stack Configuration Q) are lost; and operands B, C, D and E are unchanged.

9D/1D

Clock Cycles: 52 to 156
Status: S, Z
Stack Contents: Q

FMUL 32-Bit Floating-Point Multiply

The 32-bit floating-point operand A at TOS is multiplied by the 32-bit floating-point operand B at NOS. The normalized resultant value replaces B and then the stack is moved up so that the result occupies TOS. Operands A and B are lost; and the other operands in the stack are unchanged. Either exponent overflow (XX01) or underflow (XX01) will be reported in the Status register. In either case, the mantissa portion of the result will be correct; and the exponent portion will be offset by 128.

92/12

Clock Cycles: 146 TO 168
Status: S, Z, EF
Stack Contents: H

FSUB 32-Bit Floating-Point Subtraction

The 32-bit floating-point operand A at TOS is subtracted from the 32-bit floating-point operand B at NOS. The normalized difference replaces B; and the stack is then moved up so that the difference occupies TOS. The other operands in the stack are unchanged. Exponent alignment before the subtraction and normalization of the result account for the variation in execution time. Either exponent overflow (XX01) or underflow (XX01) will be reported in the Status register. In either case, the mantissa portion of the result will be correct; and the exponent portion is offset by 128.

91/11

Clock Cycles: 70 to 370 (A=0)
26 (A=0)
Status: S, Z, EF
Stack Contents: H

LOG 32-Bit Floating-Point Common Logarithm

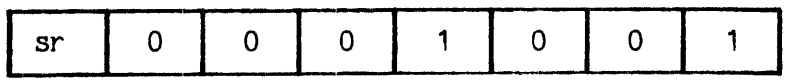
The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point common logarithm (Base 10) of A. All the original operands in the stack are lost; except B (NOS), which is unchanged. The LOG function accepts any positive input-data value that can be represented by the data format. If the LOG of a negative value is attempted, an error status of "0100" will be set in the Status register.

88/08

Accuracy: Maximum absolute error of 2.0×10^{-7} for the input range from 0.1 to 10; and a maximum relative error of 2.0×10^{-7} for positive values less than 0.1 or greater than 10.
Clock Cycles: 4474 to 7132 (A>0)
20 (A<0)
Status: S, Z, EF
Stack Contents: I

LN 32-Bit Floating-Point Natural Logarithm

The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point natural logarithm (Base e) of A. All operands in the stack are lost except B (NOS), which is unchanged. The LN function accepts all positive input-data values that can be represented by the data format. If LN of a negative number is attempted, an error status of "0100" will be set in the Status register.



89/09

Accuracy: Maximum absolute error of 2×10^{-7} for the input range from e^{-1} to e, and a maximum relative error of 2.0×10^{-7} for positive values less than e^{-1} or greater than e.
Clock Cycles: 4298 to 6956 (A>0)
20 (A<0)
Status: S, Z, EF
Stack Contents: I

NOP No Operation

The NOP command performs no internal-data manipulations. It may be used to set or clear the service request interface line without changing the stack content.

80/00

Clock Cycles: 4
Status: The status byte is reset to all zeros.

POPD 32-Bit Stack Pop

The 32-bit stack is moved up so that the original NOS becomes the new TOS. The original TOS rotates to the bottom of the stack. All operand values are unchanged. Both POPD and POPF perform the same operation.

B8/38

Clock Cycles: 12
Status: S, Z
Stack Contents: B

POPF 32-Bit Stack Pop

The 32-bit stack is moved up so that the original NOS becomes the new TOS. The original TOS rotates to the bottom of the stack. All operand values are unchanged. Both POPF and POPD perform the same operation.

98/18

Clock Cycles: 12
Status: S, Z
Stack Contents: B

POPS 16-Bit Stack Pop

The 16-bit stack is moved up so that the original value NOS becomes the new TOS. The previous value in TOS rotates to the bottom of the stack. All operand values are unchanged.

F8/78

Clock Cycles: 10
Status: S, Z
Stack Contents: P

PTOD 32-Bit TOS Onto Stack

The 32-bit stack is moved down; and the previous TOS is copied into the new TOS location. Operand D (BOS) is lost. All other operand values are unchanged. Both PTOD and PTOF execute the same operation.

B7/37

Clock Cycles: 20
Status: S, Z
Stack Contents: A

PTOF 32-Bit TOS Onto Stack

The 32-bit stack is moved down; and the previous TOS is copied into the new TOS location. Operand D (BOS) is lost. All other operand values are unchanged. Both PTOD and PTOF execute the same operation.

97/17

Clock Cycles: 20
Status: S, Z
Stack Contents: A

PTOS Push 16-Bit TOS Onto Stack

The 16-bit stack is moved down; and the previous TOS is copied into the new TOS location. Bottom of the stack operand is lost. All other operand values are unchanged.

F7/77

Clock Cycles: 16
Status: S, Z
Stack Contents: R

PUPI Push 32-Bit Floating-Point π

The 32-bit stack is moved down so that the previous TOS occupies the new NOS location. The 32-bit floating-point constant π is entered into the new TOS location. The operand in BOS is lost. The other original operands are unchanged.

91/1A

Clock Cycles: 16
Status: S, Z
Stack Contents: F

PWR 32-Bit Floating-Point X^Y

The 32-bit floating-point operand B at NOS is raised to the power that is specified by 32-bit floating-point operand A at TOS. The result replaces B; and the stack is moved up so that the result now occupies the TOS. All original operands are lost; except operand C (See Stack Configuration K) -- it is not changed. The PWR function accepts all input-data values that can be represented in the data format for operand A, and all positive values for operand B. If operand B is negative, an error pattern of "0100" will be entered into the Status register. The EXP and LN functions are used to generate PWR based on the relationship of B=EXP [A(LN B)]. Thus, if the term [A(LN B)] is outside the range of $-1.0 \times 2^{+5}$ to $+1.0 \times 2^{+5}$, an error pattern of "1100" will then be recorded. Also underflow (XX01) and overflow (XX01) conditions can occur.

8B/0B

Accuracy: The error performance for PWR is a function of the LN and EXP performance expressed by:

[(Relative Error) PWR] = [(Relative Error) EXP + [A (Absolute Error) LN]

The maximum relative error for PWR occurs when A is at its maximum value while [A(LN B)] is near 1.0×2^5; and the EXP error is also at its maximum. For most applications, the relative error for PWR will be less than 7.0×10^{-7}.

Clock Cycles: 8290 to 12032
Status: S, Z, EF
Stack Contents: K

SADD 16-Bit Fixed-Point Add

The 16-bit fixed-point two's complement integer operand A at TOS is added to 16-bit fixed-point two's complement integer operand B at NOS. The result sum replaces B; and the stack is then moved up so that TOS holds the final sum. Operand B is lost; and all the other operands are unchanged. If the addition generates a carry bit, it will be recorded in the Status register. If an overflow (XX01) occurs, it too will be entered into the Status register; and the sixteen least-significant bits of the sum are returned to TOS.

EC/6C

Clock Cycles: 16 to 18
Status: S, Z, C, EF
Stack Contents: N

SDIV 16-Bit Fixed-Point Divide

The 16-bit fixed-point two's complement integer operand B at NOS is divided by the 16-bit fixed-point two's complement integer operand A at TOS. The 16-bit integer quotient replaces B; and the stack is moved up so that the quotient occupies TOS. No remainder is generated. Both operands A and B are lost, while all other operands are unchanged. If A is zero, the final result will be set equal to B; and a divide-by-zero error (1000) will be recorded.

EF/6F

Clock Cycles: 84 to 14
Status: S, Z, EF
Stack Contents: S

SIN 32-Bit Floating-Point Sine

The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point sine of A where A is in radians. All operands are lost except B, which is unchanged. Any input-data value that can be represented by the data format will be accepted. All input values are scaled to fall within the interval $-\pi/2$ to $+\pi/2$ radians.

sr	0	0	0	0	0	1	0	82/02

Accuracy: Maximum relative error of 5.0×10^{-7} for input values in the range of -2π to $+2\pi$ radians.

Clock Cycles: 3796 to 4808
($|A| > 2^{-12}$)
30 ($|A| < 2^{-12}$)

Status: S, Z
Stack Contents: I

SMUL 16-Bit Fixed-Point Multiply, Lower

The 16-bit fixed-point two's complement integer operand A at TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at NOS. The 16-bit least-significant half of the product replaces B; and the stack is then moved up so that the result occupies TOS. The most-significant half of the product is lost as well as the original operands A and B. All other operands are unchanged. The overflow (XX01) status bit will be set if the discarded upper half was non-zero. If either A or B is the most-negative value that can be represented in the format, that value becomes the result; and an overflow (XX01) will be entered in the Status register.

sr	1	1	0	1	1	1	0	EE/6E

Clock Cycles: 84 to 94
Status: S, Z, EF
Stack Contents: S

SMUU 16-Bit Fixed-Point Multiply, Upper

The 16-bit fixed-point two's complement integer operand A at TOS is multiplied by the 16-bit fixed-point two's complement integer operand B at NOS. The 16-bit least-significant half of the result replaces B; and the stack is moved up so that the product occupies TOS. The least-significant half of the product is lost as well as operands A and B. All other operands are unchanged. If either A or B is the most-negative value that can be represented in the format, that value will be returned as the result; and an overflow (XX01) status will be set up in the Status register.

F6/76

Clock Cycles: 80 to 98
Status: S, Z, EF
Stack Contents: S

SQRT 32-Bit Floating-Point Square Root

The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point square root of A. Both the original operands A and D (BOS) are lost. The other operands in the stack are not changed. SQRT will accept any positive input-data value that can be represented by the data format. If A is negative, an error code of "0100" will be entered in the Status register.

81/01

Clock Cycles: 782 TO 870
Status: S, Z, EF
Stack Contents: G

SSUB 16-Bit Fixed-Point Subtract

The 16-bit fixed-point two's complement integer operand A at TOS is subtracted from 16-bit fixed-point two's complement integer operand at NOS. The difference replaces B; and the stack is then moved up so that the result occupies TOS. All operands are unchanged except operand B, which is lost. If the subtraction generates a borrow, it will be reported in the carry bit of the Status register. If A is the most-negative value that can be represented in the format range, an overflow status (XX01) will be entered; and the sixteen least-significant bits of the result are returned to TOS.

sr	1	1	0	1	1	0	1	ED/6D

Clock Cycles: 30 to 32
Status: S, Z, C
Stack Contents: N

TAN 32-Bit Floating-Point Tangent

The 32-bit floating-point operand A at TOS is replaced by the 32-bit floating-point tangent of A (in radians). All operands in the stack are lost except operand B, which is unchanged. The TAN function will accept any input-data value that can be represented in the data format. All input-data values are scaled to fall within $-\pi/4$ to $+\pi/4$ radians. TAN is unbounded for input values near odd multiples of $\pi/2$. In such cases, an overflow (XX01) will be set in the Status register. For angles less than 2π radians, TAN returns the original operand A as the tangent of A.

sr	0	0	0	0	1	0	0	84/04

Accuracy: Maximum relative error of 5.0×10 for input-data values in the range of -2π to 2π radians except for data values near odd multiples of $\pi/2$.

Clock Cycles: 48 to 5886 ($|A| > 2^{-12}$)
30 ($|A| < 2^{-12}$)
Status: S, Z, EF (overflow)
Stack Contents: I

XCHD Exchange 32-Bit Stack Operands

The 32-bit operand A at TOS and the 32-bit operand B at NOS are exchanged. After execution, B is at the TOS; and A is at the NOS. All operands are unchanged. XCHD and XCHF execute the same operation.

sr	0	1	1	1	0	0	1	B9/39

Clock Cycles: 26
Status: S, Z
Stack Contents: C

XCHF Exchange 32-Bit Stack Operands

The 32-bit operand A at TOS and the 32-bit operand B at NOS are exchanged. After execution, B is at the TOS; and A is at the NOS. All operands are unchanged. XCHF and XCHD execute the same operation.

sr	0	0	1	1	0	0	1	99/19

Clock Cycles: 26
Status: S, Z
Stack Contents: C

XCHS Exchange 16-Bit Stack Operands

The 16-bit operand A at TOS and the 16-bit operand B at NOS are exchanged. After execution, B is at the TOS; and A is at the NOS. All operands are unchanged.

sr	1	1	1	1	0	0	1	F9/79

Clock Cycles: 18
Status: S, Z
Stack Contents: L

9.6 9512 ARITHMETIC PROCESSOR UNIT

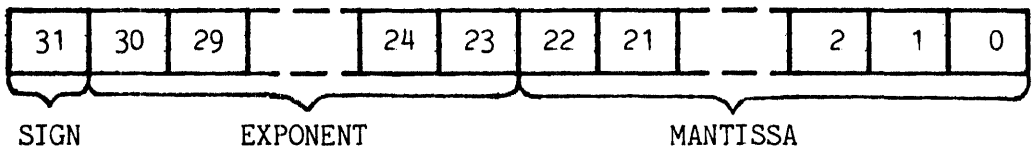
Communication between the onboard devices and the 9512 APU takes place on eight bidirectional I/O lines. These signals are gated to the internal 8-bit bus through appropriate interface and buffer circuitry. Multiplexing facilities exist for bidirectional communication between the 8- and the 17-bit buses. Both the Status and Command registers also interface with the 8-bit bus.

9.6.1 Stack Control

The 9512 performs the operands located at TOS and NOS; and the results are returned to the stack at NOS and then popped to TOS. These operands can be one of two formats -- single-precision floating-point (4 bytes) or double-precision floating-point (8 bytes). The results of an operation has the same format as the operands. In other words, operations using single-precision quantities always give a single-precision result, while double-precision quantities give double-precision results.

Operands are always entered into the stack least-significant byte first and the most-significant byte last. It should be noted that for single-precision operands four bytes should be pushed; but eight bytes must be pushed for double precision. The 9512 stack can accommodate either four single-precision or two double-precision quantities. Pushing more quantities than the stack can handle will result in loss of data. When the stack is popped for results, the most-significant byte is available first and least-significant byte last.

When the stack is read the result is transferred from the stack to the I/O lines. Reading the stack does not alter the data; it only adjusts the byte pointer. If the data popped exceeds the stack capacity, the internal byte pointer will wrap around and the original data will be reread. The 9512 can handle floating-point quantities in two different formats -- single- and double-precision. The single-precision quantities are 32-bits in length, as shown below.

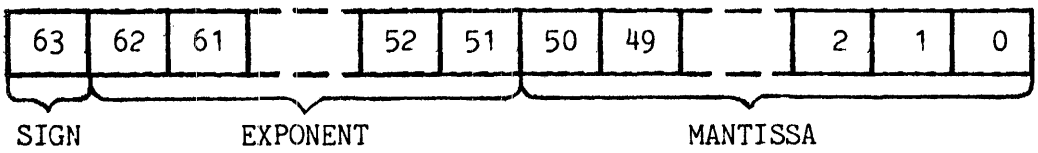


Bits 0 thru 22 in conjunction with bit 31 (SIGN) represents a signed fraction in sign-magnitude notation. A "1" in bit 31 signifies a negative number, and a "0" means positive. Bits 23 thru 30 represents a biased exponent.

There is an implied "1" beyond the most-significant bit (bit 22) of the mantissa. In other words, the mantissa is assumed to be a 24-bit normalized quantity; and the most-significant bit, which is always "1" due to normalization, is implied. The 9512 restores this implied bit internally before performing an arithmetic operation; normalizes the result, and strips the implied bit before returning the final value to the external data bus. The binary point is between the implied bit and bit 22 of the mantissa.

9.6.1.1 Double Precision

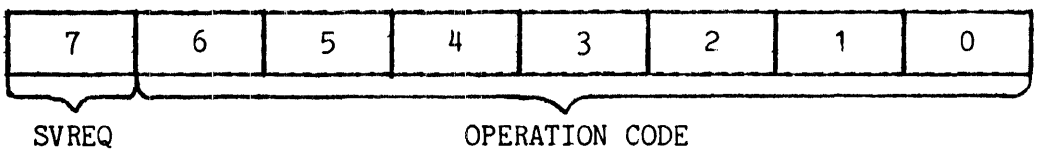
A double-precision quantity consists of the mantissa sign bit(S), an 11-bit biased exponent (E), and a 52-bit mantissa (M). The following diagram illustrates the double-precision format.



In this format; the mantissa is 52-bits in length, and the biased exponent is 11-bits. There is an implied one beyond the most-significant bit or bit 51 of the mantissa. In other words, the mantissa is assumed to be a 53-bit normalized quantity; and the most-significant bit, which will always be a one due to normalization, is implied. The 9512 restores this implied bit internally before performing arithmetic; normalizes the result and strips the implied bit before returning the result to the external data bus. The binary point is between the implied bit and bit 51 of the mantissa.

9.6.2 Command Format

The format of the 8-bit command is shown below.

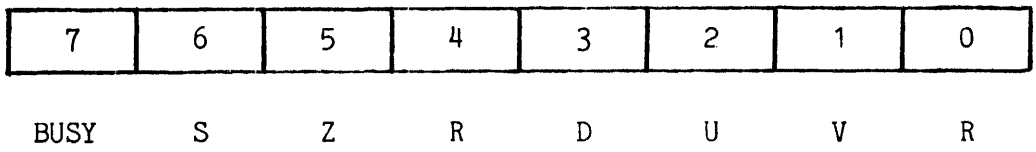


The command consists of eight bits. The least-significant seven bits specify the operation to be performed. The most-significant bit is the SERVICE REQUEST ENABLE (SVREQ) bit, and it must be a "1" if SVREQ is to go "high" following the execution of a command

The 9512 commands fall into three categories: Single-precision arithmetic, double-precision arithmetic and data manipulation. There are four arithmetic operations that can be performed with either single-precision (32 bits), or double-precision (64 bits) floating-point numbers -- add, subtract, multiply and divide. These operations require two operands. The 9512 assumes that these operands are located in the internal stack as Top Of Stack (TOS) and Next On Stack (NOS). The result will always be returned to the previous NOS, which becomes the new TOS. Results from an operation are of the same precision and format as the operands; and they will be found to preserve the accuracy. In addition to the arithmetic operations, the 9512 can perform eight data manipulating operations. These include changing the sign of a double- and single- precision operand located at TOS and NOS, as well as copying and popping single- and double- precision operands.

9.6.3 Status Register

An internal 8-bit status register having the following format provides device status information to the MSC 8009.



While bit 7 (BUSY) is a "1", the other status bits will not be defined. The definition of these bits (See Table 9-2) is valid only when bit 7 is "0" -- operation complete.

Bit 0 Reserved

Bit 1 indicates that an exponent over flow has occurred, otherwise the bit is cleared to zero.

Bit 2 shows that an exponent underflow has occurred, otherwise the bit is to zero.

Bit 3 denotes that an attempt to divide by zero is made, otherwise the bit is to zero.

Bit 4 Reserved

Bit 5 indicates that a command has been completed, and the result in TOS is all zeros. Otherwise, this bit is zero.

Bit 6 signifies a negative result in TOS, otherwise this bit will be zero.

Bit 7 indicates that the 9512 is in the process of executing a command. After the command is terminated, this bit is reset to zero.

Table 9-2
STATUS BIT DEFINITION

9.7 9512 INSTRUCTIONS

For a quick reference,a shorthand notation will be used in the following discussion and listing to describe each instruction and how it operates. A single lower-case letter denotes a single-precision operation; and a double lower-case letter represents a double-precision operation. All other symbols, mnemonics and abbreviations that will be used are:

a	TOS quantity prior to executing the instruction
b	NOS quantity prior to executing the instruction
BOS	Bottom Of Stack
CC	Clock cycle
D	Divide-by-zero bit of the Status register
NOS	Next Of Stack
PG	Section 9 page numbers
R	Result of the executed operation
S	Sign bit of the Status register
SC	Stack content as defined in Table 9-3.
TOS	Top Of Stack
U	Underflow bit of the Status register
V	Overflow bit of the Status register
X	Status register bit is affected
Z	Zero bit of the Status register
*	Status register bit is unaffected
w<= v	Quantity "w" is replaced by quantity "v"
w<=#v	Quantity "w" is equal to the complement of the quantity "v"
w<=>v	Quantities "w" and "v" are exchanged.

9.7.1 Data And Stack Manipulation Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z D U V
CHSD	43	TOS<=#aa	24	C	X X 0 0 0
CHSS	43	TOS<=#a	10	H	X X X 0 0
CLR	43	Clear Status	4	M	0 0 0 0 0
POPS	45	TOS<=b BOS<=a	14	G	X X * 0 0
PTOD	46	TOS<=aa NOS<=aa	40	A	X X 0 0 0
PTOS	46	TOS<=a NOS<=a	16	I	X X 0 0 0
If 'a' exp=0		TOS<=0 NOS<=a	16	I	X X 0 0 0
XCHS	49	a<=b	26	C	X X 0 0 0

9.7.2 Single Precision Operations

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z D U V
SADD	47	TOS<=a+b	58	F	X X 0 X X
SDIV	47	TOS<=b/a	228	S	X X X * *
SMU	48	TOS<=[a x b]	198	F	X X 0 X X
SSUB	48	TOS<=b-a	56	F	X X * * *

9.7.3 Double Precision Operation

INSTRUCTION		NOTATION	CC	SC	STATUS REG.
OP CODE	PG				S Z D U V
DADD	44	TOS<=aa+bb	578	B	X X 0 X X
DDIV	44	TOS<=bb/aa	4560	E	X X X X X
If 'aa' exp=0		TOS<=bb			
DMUL	45	TOS<=[aa x bb]	1748	B	X X 0 X X
DSUB	45	TOS<=bb-aa	578	B	X X 0 X X

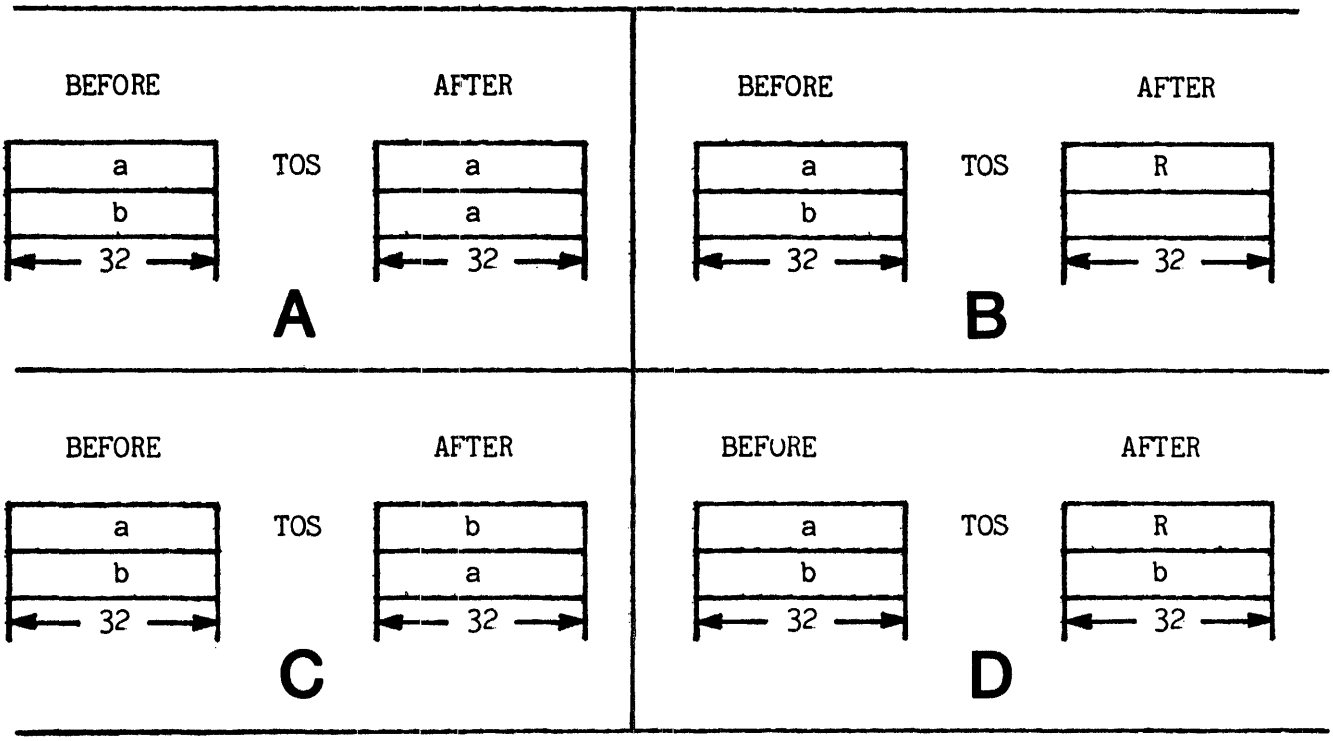


Table 9-2
STACK CONFIGURATIONS

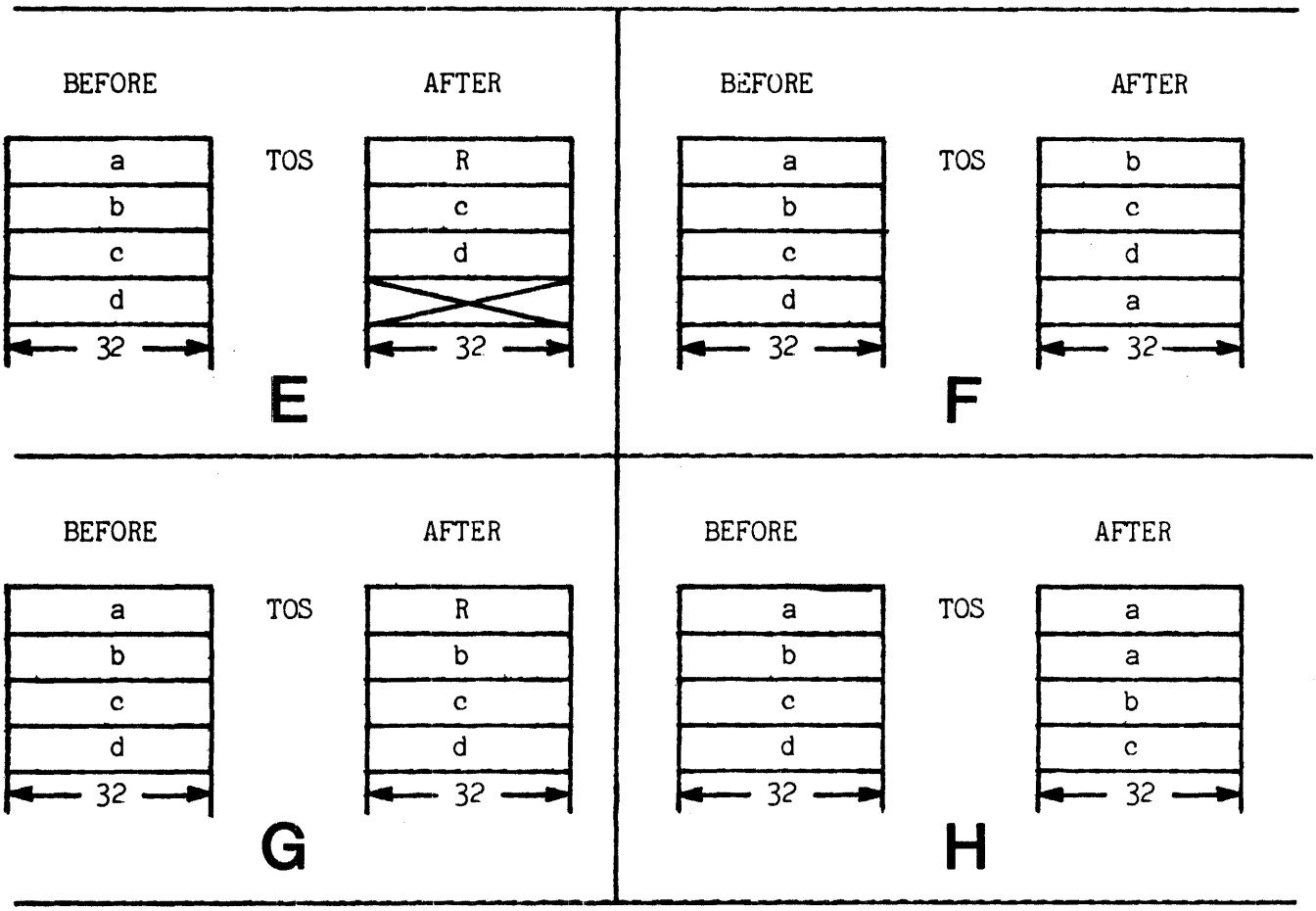


Table 9-2 (cont'd)
STACK CONFIGURATIONS

CHSD Change Sign, Double Precision

The sign of the double-precision operand A at TOS is complemented. The double-precision resultant is returned to TOS. If the double-precision operand A is zero, then the sign will not be affected. Status bits S and Z indicate the sign of the result and if the resultant value is zero. Status bits U, V and D are always zero.

sr	0	1	0	1	1	0	1	AD/2D

Clock Cycles: 24
Status: S, Z
Stack Contents: D

CHSS Change Sign, Single Precision

The signal of the single-precision operand A at TOS is complemented. The single-precision result is returned to TOS. If the exponent field of A is zero, the resultant value will be zero. Status bits S and Z indicate the sign of the result. If the resultant value is zero, status bits U, V and D will then be zero.

sr	0	0	0	0	1	0	1	85/05

Clock Cycles: 18
Status: S, Z
Stack Contents: G

CLR Clear Status

All status bits (S,Z,D,U,V) are set to zero; and the stack is not affected. Essentially this is a No Op command as far as operands are concerned.

sr	0	0	0	0	0	0	0	80/00

Clock Cycles: 4
Status: S, Z, D, U, V are always zero.

DADD Double-Precision Floating-Point Add

The double-precision operand A from TOS is added to the double-precision operand B from NOS. The result is rounded to obtain the final double-precision sum, which is returned to TOS. Status bits S, Z, U and V are used to report sign of the result; if the resultant value is zero; exponent underflow, and exponent overflow, respectively.

A9/29

Clock Cycles: 578
Status: S, Z, U, V
Stack Contents: B

DDIV Double-Precision Floating-Point Divide

The double-precision operand B at NOS is divided by the double-precision operand A at TOS. The result is rounded to obtain the final double-precision quotient R, which is returned to TOS. Status bits S, Z, D, U and V are used to report sign of the result; if the resultant value is zero; attempt to divide by zero; exponent underflow, and exponent overflow, respectively. If A is zero, the resultant value at TOS will equal the original value of operand B.

AC/2C

Clock Cycles: 4560
Status: S, Z, D, U, V
Stack Contents: B

DMUL Double-Precision Floating-Point Multiply

The Double-precision operand A at TOS is multiplied by the double-precision operand B at NOS. The result is rounded to obtain the final double-precision product, which is returned to TOS. Status bits S, Z, U and V are used to report sign of the result; if the result is zero; exponent underflow, and exponent overflow, respectively. Status bit D will be reset to zero.

AB/2B

Clock Cycles: 1748
Status: S, Z, U, V
Stack Contents: B

DSUB Double-Precision Floating-Point Subtract

The double-precision operand A at TOS is subtracted from the double-precision operand B at NOS. The result is rounded to obtain the final double-precision difference, which is returned to TOS. Status bits S, Z, U and V are used to report sign of the result; if the result is zero; exponent underflow, and exponent overflow, respectively. Status bit D will be reset to zero.

AA/2A

Clock Cycles: 578
Status: S, Z, U, V
Stack Contents: B

POPS Pop Stack, Single Precision

The single-precision operand A is popped from the stack in addition to being transferred to the bottom of the stack. Status bits S and Z are used to report the sign of the new operand at TOS; and if it is zero. The other status bits are reset to zero. Note that only the exponent field of the updated or new TOS is checked for zero.

87/07

Clock Cycles: 14
Status: S, Z
Stack Contents: F

PTOD Push Stack, Double Precision

The double-precision operand A at TOS is pushed back on to the stack. This effectively duplicates A two consecutive stack locations. Status bits S and Z are used to report sign of the new TOS and if the new TOS is zero. The other status bits are reset to zero.

AE/2E

Clock Cycles: 40
Status: S, Z
Stack Contents: A

PTOS Push Stack, Single Precision

This instruction effectively pushes the single-precision operand at TOS onto the stack. In other words, the original operand is duplicated in two locations of the stack. However, if the operand at TOS prior to the PTOS command has only its exponent field as zero, the new content of the TOS will be all zeros; and the contents of NOS will be a copy of the orginal TOS. Status bits S and Z are used to report the sign of the new TOS and if the content of TOS is zero. The other status bits are reset to zero.

86/06

Clock Cycles: 16
Status: S, Z
Stack Contents: H

SADD Single-Precision Floating-Point Add

The single-precision operand A at TOS is added to the single-precision operand B at NOS. The result is rounded to obtain the single-precision sum, which is returned to TOS. Status bits S, Z, U and V are used to report the sign of the result; if the result is zero; exponent underflow, and exponent overflow, respectively. Status bit D will be reset to zero.

sr	0	0	0	0	0	0	1	81/01

Clock Cycles: 58
Status: S, Z, U, V
Stack Contents: E

SDIV Single-Precision Floating-Point Divide

The single-precision operand B at NOS is divided by the single-precision operand A at TOS. The result is rounded to obtain the final quotient, which is returned to TOS. Status bits S, Z, D, U and V are used to report the sign of the result; if the result is zero; attempt to divide by zero; exponent underflow, and exponent overflow, respectively. If the exponent of the original operand A is zero, the resultant value in TOS will equal operand B.

sr	0	0	0	0	1	0	0	84/04

Clock Cycles: 228
Status: S, Z, U, V
Stack Contents: E

SMUL Single-Precision Floating-Point Multiply

The single-precision operand A at TOS is multiplied by the single-precision operand B at NOS. The result is rounded to obtain the final single-precision product, which is returned to TOS. Status bits S, Z, U and V are used to to report sign of the result; if the result is zero; exponent underflow, and exponent overflow, respectively. Status bit D is reset to zero.

83/03

Clock Cycles: 198
Status: S, Z, U, V
Stack Contents: E

SSUB Single-Precision Floating-Point Subtract

The single-precision operand A at TOS is subtracted from the single-precision operand B at NOS. The result is rounded to obtain the final single-precision difference, which is returned to TOS. Status bits S, Z, U and V are used to report the sign of the result; if the result is zero; exponent underflow, and exponent overflow, respectively. Status bits D will be reset to zero.

82/02

Clock Cycles: 56
Status: S, Z, U, V
Stack Contents: E

XCHS Single-Precision Stack Exchange

The 32-bit operand A at TOS and the 32-bit operand B at NOS are exchanged. After execution, B is at TOS; and A is at NOS. All operands are unchanged.

88/08

Clock Cycles: 26
Status: S, Z
Stack Contents: C

APPENDICES

Appendix A

MSC 8009 PIN ASSIGNMENT

Wire-wrap pins allow the user to configure the MSC 8009 for a specific application. These on-board pins are identified and described in Table-1 of this appendage.

PIN	DESCRIPTION	PIN	DESCRIPTION
1	NOT USED	31	NOT USED
2	NOT USED	32	NOT USED
3A,B	TXRDY/, 1TXRDY/	33	NOT USED
4A,B	RXRDY/, 1RXRDY/	34	NOT USED
5	4 MHz/	35	NOT USED
6	WRITE PRECOMP CLK INPUT	36	NOT USED
7	8 MHz	37	NOT USED
8	NOT USED	38	NOT USED
9	SVREQ/ FROM APU	39A,B	RXCLK IN
10	NOT USED	40A,B	RXC to the 8251(U19)
11	NOT USED	41A,B	TXCLK OUT
12	NOT USED	42A,B	CLK for the 75188 (U10)
13	NOT USED	43A,B	-12V
14	NOT USED	44A,B	U10-1 (75188)
15	NOT USED	45A,B	+5V
16	NOT USED	46A,B	U10-14 (75188)
17	NOT USED	47A,B	+12V
18	NOT USED	48	1MHz
19	NOT USED	49	I/O CLK
20	NOT USED	50	2 MHz
21	TSTCLK	51	I/O CLK
22	EXTRQ	52	2 MHz
23	RG 1793 (U321)	53	Z80 CLK
24	HEAD	54	2 MHz
25	HSL 74LS273 (U309)	55	APU CLK
26	NOT USED	56	4 MHz
27	NOT USED	57	APU CLK
28	NOT USED	58	4 MHz
29	NOT USED	59	Z80 CLK
30	NOT USED	60	onboard reset generator.

Table 1
MSC 8009 PIN CONFIGURATION

PIN	DESCRIPTION	PIN	DESCRIPTION
61	INIT/ P1-14	95	16K/64K REFEN
62	BUFFERED 8 MHz	96A,B	DSR/ for the 8251 (U19)
63	BCLK/ P1-13	97A,B	DSR/ J(2,3)-14
64	WATCHDOG TIMER OUTPUT	98A,B	TXC for the 8251 (U19)
65	LNMI	99A,B	RXC for the 8251 (U19)
66	AACK/ LOCAL	100	Enable for HEAD-START CKT
67	AACK/ P1-25	101	HEAD START CKT for 4MHz
68	BUFFERED 8 MHz	102	Bus state machine latch
69	CCLK/ P1-31	103	MRQ/
70	LNMI/	104	MRQ/
71	NMI/ P1-33	105	IORQ/
72	R0 of 8214 (U62)	106	IORQ/
73	INT7/ P1-36	107	Bus state machine latch IORQ/
74	R1 of 8214 (U62)	108	+5V
75	INT6/ P1-35	109	Pin 8 of RAM array
76	R2 of 8214 (U62)	110	+12V
77	INT5/ P1-38	111	+5V
78	R3 of 8214 (U62)	112	Pin 9 of RAM array
79	INT4/ P1-37	113	MA7
80	R4 of 8214 (U62)	114	-5V
81	INT3/ P1-40	115	Pin 1 of RAM array
82	R5 of 8214 (U62)	116	+5V
83	INT2/ P1-39	117	GND P1-15
84	R6 of 8214 (U62)	118	BPRN/
85	INT1/ P1-42	119	Unused bit of external
86	R7 OF 8214 (U62)	120	IRQ/ from 1793 (U321)
87	INT0/ P1-41	121	GND
88	ELEM SEL FOR REF	122	READY input to 1793 (U321)
89	4K REFEN	123	Ready input from J1 pin 22
90	16K/64K ELEM SEL FOR ROW	124	Data bit 3 of Unit Select port (0C4H)
91	16K/64K ROWEN	125	READY input from J1, pin 22
92	4K ELEM SEL for ROW	126	INDEX/ input to 1793 (U321)
93	4K ROWEN	127	Pin 24 of J1
94	16K/64K ELEM SEL for REF		

Table 1 (cont'd)
MSC 8009 PIN IDENTIFICATION

Appendix B

MSC 8009 JUMPER REQUIREMENT

By interconnecting the wire-wrap pins listed in Addendum A, the user can easily configure the MSC 8009 for a specific application. These pin/jumper combinations are described in Table 1 of this appendage.

WIRE-WRAP POST	FUNCTION
3A,3B	TXRDY/ output; normally tied to an interrupt pin.
4A,4B	RXRDY/ output; normally tied to an interrupt pin.
5-6	250ns Write Precomp
6-7	125ns Write Precomp
9	SVRE/(Service Request) output from the APU; normally tied to an interrupt pin when used.
21 (TSTCLK) 22 (EXTRQ)	Test clock; rising-edge on every local-bus transaction external request; low for every MULTIBUS request for onboard resources.
23-24	Connects pin 25 of a 1797 chip, when used, to the HEAD/ line going to the drive, since the pin is used for headselect.
24-25	Connects data-bit 4 to the HEAD/ line going to the floppy-diskdrive.
39A (NC) 40A-42A (TIMER INTERNAL CLK) 41A (NC) 96A-97A (DSR/ is on J2-14) 98A-99A (TXC = RXC)	Standard Configuration: These terminals configure the CLK and DSR signals associated with Serial Port #1. TXC and RXC are tied and supplied by the 8253 Baud Rate generator section.
39B (NC) 40B-42B (TIMER INTERNAL CLK) 41B (NC) 96B-97B (DSR/ is on J3-14) 98B-99B (TXC = RXC)	Standard Configuration: These terminals configure the CLK and DSR signals associated with Serial Port #2. TXC and RXC are tied and supplied by the 8253 Baud Rate generator section.

Table 1
MSC 8009 JUMPER CONFIGURATION

WIRE-WRAP POST	FUNCTION
39A (RXCLK from U2-8 NC) 41A-42A (TXCLK to J2-4) 96A (DSR/ not used) 97A-98A (TXC from J2-14) 99A (RXC not used)	Serial Port #1 high-speed synch modem (2400 - 19.2K Baud Rate) inputs. TXCLK is supplied to modem on J2-4; RXCLK is from the modem on J2-8, and DSR is from the modem on J2-14.
39B-40B (RXC from J3-8) 41B-42B (RXC to J3-4) 96B (DSR/ not used) 97B-98B (TXC from J3-14) 99B (RXC not used)	Serial Port #2 high-speed synch modem (2400 - 19.2K Baud Rate). inputs. TXCLK is supplied to modem on J3-4; RXCLK is from the modem on J3-8, and DSR is from the modem on J3-14.
43A-44A	-12V to pin 1 of 75188 (U10A) for RS-232-C voltage requirement of Serial Port #1.
45A (NC)	+5V Not used.
46A-47A	+12V to pin 14 of 75188 (U10A).
43B-44B	-12V to pin 1 of 75188 (U10B) for RS-232-C voltage requirement of Serial Port #2.
45B (NC)	+5V Not used.
46B-47B	+12V to pin 14 of 75188 (U10B).
48-49	Supplies 1 MHz to I/O devices and 8253 Baud Rate generator.
50-51	Supplies 2 MHz to I/O devices and 8253 Baud Rate generator.
52-53	Supplies 2 MHz to the Z80.
54-55	Supplies 2 MHz to the APU.
56-57	Supplies 4 MHz to the APU.

Table 1 (cont'd)
MSC 8009 JUMPER CONFIGURATION

WIRE-WRAP POST	FUNCTION
58-59	Supplies 4 MHz to the Z80.
60-61	Connects onboard reset generator to INIT/ line on P1-14 MULTIBUS.
62-63	Connects buffered 8 MHz to BCLK/ on P1-13 of the MULTIBUS.
64-65	Connects output of Watchdog timer to Z80, which generates NMI when Watchdog times out.
66-67	Connects AACK/ to MULTIBUS P1-25. Consult manual before using.
68-69	Connects buffered 8 MHz to CCLK/ on P1-31 of the MULTIBUS.
70-71	Connects NMI to P1-33 of MULTIBUS. For use with MSC Multibus chassis.
72-73 74-75 76-77 78-79 80-81 82-83 84-85 86-87	When installed, jumpers connect the interrupt-controller inputs of the MSC 8009 to the interrupt lines on the MULTIBUS (P1-35, P1-36, P1-37, P1-38, P1-39, P1-40, P1-41 and P1-42). Install with discretion in multiprocessor system.
88-89 90-91 92-93 94-95	Used to program memory-control circuit for 4K 16K or 64K elements. Posts 90-91 and 94-95 select 16K and 64K elements.
100-101	Enables a memory-headstart circuit that reduces RAM access time by a Wait state when the Z80 is running at 4 MHz. REMOVE this jumper when the Z80 is running at 2 MHz or when used in a multiprocessor system.

Table 1 (cont'd)
MSC 8009 JUMPER CONFIGURATION

WIRE-WRAP POST	FUNCTION
102-103 (MRQ/) 106-107 (IORQ/)	Routes MRQ/ and IORQ/ thru Bus State Machine latches. It may be advantageous to use these when using an in-circuit emulator. There will be an extra Wait state added to every processor transaction.
103-104 (MRQ/) 105-106 (IORQ/)	Standard jumpers that route MRQ/, IORQ/ and bypass Bus State Machine latches.
108-109	+5V to pin 8 of RAM array (5V elements only).
109-110	+12V to pin 8 of RAM array (Three-supply elements only).
111-112	+5V to pin 9 of RAM array (16K X 1 elements only).
112-113	Supplies MA7 to pin 9 of RAM (64K X 1 elements only).
114-115	-5V to pin 1 of RAM array (Three-supply elements only).
115-116	+5V to pin 1 of RAM array (Non-Three-Supply elements only).
117-118	Grounds BPRN/ (P1-15). Use with discretion on multimaster systems.
119	Unused external ROM bit, available for future design.

Table 1 (cont'd)
MSC 8009 JUMPER CONFIGURATION

WIRE-WRAP POST	FUNCTION
120	Floppy-disk interrupt request line.
121-122	Ground READY input for 5-inch drives only.
123-124	READY signal from J1, pin 22 for 8-inch drives only.
124-125	Unit 4 selection signal for 5-inch drives only.
126-127	INDEX signal from J1, pin 24 for 5-inch drives only.

Table 1 (cont'd)
MSC 8009 JUMPER CONFIGURATION

Appendix C

FLOPPY-DISK DRIVE JUMPER CONFIGURATION

This appendix provides the user with the information that is needed to properly configure the following Shugart disk drives for use with either the MSC 8009 -- single-board processor with onboard soft-sector inteface/controller; or the MSC 8101 -- hard-sector floppy-disk floppy-disk interface/controller board.

SA 400
SA 800/801
SA 850/851

JUMPER	DRIVE 0	DRIVE 1	DRIVE 2*	DRIVE 3*
A	PLUGGED	PLUGGED	PLUGGED	PLUGGED
B	PLUGGED	PLUGGED	PLUGGED	PLUGGED
C	PLUGGED	PLUGGED	PLUGGED	PLUGGED
D	OPEN	OPEN	OPEN	OPEN
DC	OPEN	OPEN	OPEN	OPEN
DS	PLUGGED	PLUGGED	PLUGGED	PLUGGED
DS1	PLUGGED	OPEN	OPEN	OPEN
DS2	OPEN	PLUGGED	OPEN	OPEN
DS3	OPEN	OPEN	PLUGGED	OPEN
DS4	OPEN	OPEN	OPEN	PLUGGED
HL	OPEN	OPEN	OPEN	OPEN
L	NOTE 2	NOTE 2	NOTE 2	NOTE 2
S	CUT RUN	CUT RUN	CUT RUN	CUT RUN
T1	PLUGGED	OPEN	OPEN	OPEN
T2	PLUGGED	PLUGGED	PLUGGED	PLUGGED
T3	PLUGGED	OPEN	OPEN	OPEN
T4	PLUGGED	OPEN	OPEN	OPEN
T5	PLUGGED	OPEN	OPEN	OPEN
T6	PLUGGED	OPEN	OPEN	OPEN
X	OPEN	OPEN	OPEN	OPEN
Y	PLUGGED	PLUGGED	PLUGGED	PLUGGED
Z	OPEN	OPEN	OPEN	OPEN
800	PLUGGED	PLUGGED	PLUGGED	PLUGGED
801	OPEN	OPEN	OPEN	OPEN

*NOTES:

1) Drive 0 should be installed at the end of the interconnecting cable between drives and interface board, because the line terminators are in this drive. Drives 2 and 3 are used with the MSC 8009 only.

2) Position of the "L" jumper depends on the power supply being used (Refer to the proper Shugart manual for information).

Table 1
SA 800/801
JUMPER/PLUG CONFIGURATION

JUMPER	DRIVE 0	DRIVE 1	DRIVE 2*	DRIVE 3*
C	PLUGGED	PLUGGED	PLUGGED	PLUGGED
DC	OPEN	OPEN	OPEN	OPEN
DS	PLUGGED	PLUGGED	PLUGGED	PLUGGED
DS1	PLUGGED	OPEN	OPEN	OPEN
DS2	OPEN	PLUGGED	OPEN	OPEN
DS3	OPEN	OPEN	PLUGGED	OPEN
DS4	OPEN	OPEN	OPEN	PLUGGED
FS	PLUGGED	PLUGGED	PLUGGED	PLUGGED
HLL	OPEN	OPEN	OPEN	OPEN
16-PIN PROGRAMMABLE SHUNT LOCATION H4 A	SHUNT			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 B	SHUNT			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 HL	OPEN			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 I	SHUNT			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 R	SHUNT			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 S	SHUNT			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 X	OPEN			
16-PIN PROGRAMMABLE SHUNT LOCATION H4 Z	OPEN			
IT	OPEN	OPEN	OPEN	OPEN
IW	PLUGGED	PLUGGED	PLUGGED	PLUGGED
DL	PLUGGED	PLUGGED	PLUGGED	PLUGGED
RM	OPEN	OPEN	OPEN	OPEN
RS	PLUGGED	PLUGGED	PLUGGED	PLUGGED
S1	OPEN	OPEN	OPEN	OPEN
S2	PLUGGED	PLUGGED	PLUGGED	PLUGGED
S3	OPEN	OPEN	OPEN	OPEN
TS	OPEN	OPEN	OPEN	OPEN
Y	PLUGGED	PLUGGED	PLUGGED	PLUGGED
1B	OPEN	OPEN	OPEN	OPEN
2B	OPEN	OPEN	OPEN	OPEN
3B	OPEN	OPEN	OPEN	OPEN
4B	OPEN	OPEN	OPEN	OPEN
-15, -5	SEE NOTE 1			
*2S	PLUGGED	PLUGGED	PLUGGED	PLUGGED
850	PLUGGED	PLUGGED	PLUGGED	PLUGGED
851	OPEN	OPEN	OPEN	OPEN
RES. PK (H3)	INSTALL	REMOVE	REMOVE	REMOVE

*NOTES: 1) Position of the jumpers depends on the power supply (Refer to the Shugart manual).
2) 2S is plugged for double-sided, open for single-sided.
3) Cut etch to pin 48 on ALL drives.

Table 2
SA 850/851
JUMPER/PLUG CONFIGURATION

JUMPER	PRIMARY DRIVE*	SECONDARY DRIVE
DS1 (2-13)	OPEN	OPEN
DS2 (3-12)	OPEN	JUMPERED
HS/HL (1-14)	OPEN	OPEN
HM/MH (7-8)	OPEN	OPEN
MX (5-10)	OPEN	OPEN
MX (6-9)	OPEN	OPEN
RES. PK	INSTALLED	REMOVED

*NOTE:

1) See Note 1 of Table 1 concerning drive requirements.

2) The above jumpers go into IC location 1F.

Table 3
SA 400
JUMPER/PLUG CONFIGURATION

Appendix D

9511 ARITHMETIC PROCESSOR APPLICATION NOTE

The following discussion details the 9511 commands, execution procedures, and performance as related to each instruction.

FUNCTIONAL DESCRIPTION

The 9511 is addressed as two ports selected by the LA0 line from the Z80 processor. When LA0 is "high", the Status register can be accessed via a read operation; and a command can be entered using a write. A "low" on LA0 causes a read instruction to access data from the Top-Of-Stack (TOS); and a write to enter data into TOS. Table 1 lists and defines the interface signals that control the 9511.

Data Stack

The internal data stack operates as a true push-down stack (FILO). In other words, when pushing data on the stack, the least-significant byte must be entered first and the most-significant byte last. When popping the stack to read the result of an operation, the most-significant byte will be available on the data bus first and the least-significant byte last.

The data stack consists of eight levels, where each level is 16-bits wide. Since single-precision fixed-point operands are 16-bits in length, the data stack can hold up to eight such values. For either double-precision (32 Bits) fixed-point or floating-point formats, up to four values can be maintained within the stack.

Data Entry

To enter data from the I/O BUS, it requires a "low" level on the chip-select line (CS4/), command data line (LA0), and the write request line (LIOWC). As each new data word is entered, the previously entered data is pushed "down"; and the new byte is placed on top of the stack. Data on the bottom of the stack prior to the entry procedure will be lost.

BIT	DESIGNATION	DESCRIPTION
0	CARRY	Previous operation resulted in either a carry or borrow from the most-significant bit. A logic "1" denotes a carry/borrow; and a logic "0" is a no carry/borrow.
1-4	ERROR CODE	A three-bit field indicates the validity of the results of the last operation. The error codes are: 0000 - No error 1000 - Divide by zero 0100 - Square root or log of a negative number. 1100 - Augment of inverse sine,cosine, or e to large. XX10 - Underflow XX01 - Overflow
5	ZERO	TOS value is zero if a logic "1".
6	SIGN	A logic "1" indicates a negative quantity in TOS.
7	BUSY	If a logic "1", the 9511 is currently executing a command.

Table 1
STATUS REGISTER DEFINITIONS

Data Extraction

Data is read from TOS when the chip select line (CS4/), command data line (LA0), and the read data line (LIORC/) go "low". As each byte at TOS is read from the stack, it will also be rotated to the bottom of the stack; then the next successive byte is pushed up into TOS.

Command Entry

After the appropriate number of bytes have been entered into the stack, a command can now be issued to perform the desired operation. The single-operand instructions operate on TOS only. However, some instructions, such as an add, operate on both TOS and NOS (Next On Stack) values; and they require two operands.

For a command entry, both the chip select (CS4/) and write data (LIOWC/) lines must be "low", and the command/data line must be "high". After the 9511 accepts the command, the Z80 can now execute other instructions concurrently with the 9511 command execution. If the Z80 issues another command to the 9511 during the execution of an APU instruction, the 9511 will not accept the new instruction until the current one is complete. Also, the Z80 will not be able to perform other programming tasks until this situation is ratified.

Pause Operation

A "high" on the PAUSE/ line signifies that the 9511 is in a quiescent state. If any of the following conditions exist, this signal will go "low".

1) A previously initiated operation is in progress; and either another command or stack access is attempted. PAUSE/ remains "low" until the current command is terminated; and then it goes "high" to allow the entry of the new instruction.

2) There is a request for data and the 9511 is not busy. PAUSE/ remains "low" for the time it takes to transfer one byte from TOS onto the I/O BUS.

3) If the 9511 is not busy, and a data entry has been requested; PAUSE/ will go "low". It remains "low" for the time that it requires to ascertain the preceding byte -- if any -- has been written into the stack. If so, PAUSE/ returns to a "high" immediately. If not, PAUSE/ remains "low" until the interface latch is released.

4) A status read will pull the PAUSE/ "low" for the time required to transfer the status of the interface latch; and then PAUSE/ will go "high", completing the status operation. It should be pointed out that status can be read even if the 9511 is busy.

NOTE: When PAUSE/ goes "low", the control signals present at that time must remain stable until PAUSE/ goes "high".

Status

An on-chip Status register provides a means for examining the status of the 9511. If the BUSY bit (Bit 7) is a logic "1", the other status bits will not be defined. A logic "0" means that the operation is complete; and the other status bits will be defined as listed in Table 2.

Read Status

To read the status register, a low is required on both the chip-select (CS4/) and read (LIORC/) lines in addition to a "high" on the command/data line (LA0). The status register information then gated onto the I/O BUS (I00 thru I07). The status of the 9511 can be read by the Z80 and time regardless if an operation is in progress or not.

PIN	SIGNAL	DESCRIPTION
1	GND	Power
2	+5V	Power
3	EACK/	Active "low" clears the END/ signal. If tied "low", the END/ output will be a pulse that is less than a clock period.
4	SVACK/	Active "low" resets SVREQ.
5	SVREQ	An active "high" output signal indicates that command execution is complete; and that the post-execution service was requested in the previous command byte. It is cleared by either SVACK/, RESET or end of a subsequent command that does not request service.
6	-	Not used.
7	-	Not used.
8-15	DB0-DB7	Eight bidirectional lines provide for transfer of commands, status and data between the 9511 and the CPU. The 9511 drives the data bus only when CS/ and RD/ are "low".
16	+12V	Power
17	PAUSE/	Active "low" output indicates that the 9511 has not yet completed the information transfer over the data bus. For further description refer to the paragraph entitled "Pause Operation".
18	CS/	An active "low" input signal conditions the read and write signals; thus, enabling the communication with the data bus.
19	WR/	The CS/ conditions the active "low" WR/ signal, indicating that information is to be transferred from the data bus into internal locations. RD/ and WR/ are mutually exclusive

Table 2
9511 INTERFACING SIGNALS

PIN	SIGNAL	DESCRIPTION
20	RD/	The active "low" signal is conditioned by CS/ and indicates that information is to be transferred from internal locations to the data bus. Rd/ and WR/ are mutually exclusive.
21	C/D	In conjunction with the RD/ and WR/ signals, the C/D control line establishes the type of transfers that are to be performed on the data bus.
22	RESET	An active "high" provides initialization for the 9511 chip. Reset terminates any operation progress, clears the status register and places the 9511 into the idle state. Stack contents are not affected. The RESET should be active for at least five clock periods following stable supply voltages and stable clock input. There is no internal power-on reset.
23	CLK	This is an input for an external timing source that may be asynchronous to the read and write control signals.
24	END/	An active "low", open-drain output indicates that execution of the previously entered command is complete. It can be used an an interrupt request and is cleared by EACK/, RESET or any read or write access to the 9511.

Table 2 (cont'd)
9511 INTERFACING SIGNALS

All derived functions except "Square Root" use Chesbyshev polynomial approximating equations. This approach helps minimize the internal microprogram; minimize the maximum error value; and provides a relatively even distribution of errors over the data range. To compute the various Chebyshev terms, the derived functions use the basic arithmetic operations that may produce error codes in the Status register as a result.

The 9511 commands, mnemonics, hex code and execution cycles are summarized in Table 3. For other details, refer to Section 9 of this manual. Speeds given in Table 3 are in terms of clock cycles. To arrive at the actual time value, multiply the clock cycles by the clock period. For example, the execution time for the SADD instruction is 16 to 18 clock cycles. In a 4-MHz system, this translates into 4 to 4.5 microseconds.

Where substantial variation of execution times could occur, the minimum and maximum values are given; otherwise the values are typical. Variations in the execution cycles reflect the data dependency of the algorithms. Some boundary conditions that will cause shorter execution times are not taken into account.

Total execution times may require allowances for operand transfer into the 9511, command execution and result retrieval from the 9511. Except for command execution, these times will be heavily influenced by such items as type of data; the control interface used; memory speed; CPU used; the priority allotted to DMA and interrupt operations; the size and number of operands to be transferred, and the use of chained calculations.

COMMAND MNEMONIC	HEX CODE (sr = 1)	HEX CODE (sr = 0)	EXECUTION CYCLES	SUMMARY DESCRIPTION
DATA AND STACK MANIPULATION OPERATIONS				
NOP	80	00	4	No Operation. Clear or set SVREQ.
FIXS FIXD	9F 9E	1F 1E	90 -214 90 -336	Convert TOS from floating point format to fixed point format.
FLTS FLTD	9D 9C	1D 1C	62 -156 56 -342	Convert TOS from fixed point format to floating point format.
CHSS CHSD	F4 B4	74 34	22 - 24 26 - 28	Change sign of fixed point operand on TOS.
CHSF	95	15	16 - 20	Change sign of floating point operand on TOS.
PTOS PTOD PTOF	F7 B7 97	77 37 17	16 20 20	Push stack. Duplicate NOS in TOS.
POPS POPD POPF	F8 B8 98	78 38 18	10 12 12	Pop stack. Old NOS becomes new TOS; and old TOS rotates to bottom.
XCHS XCHD XCHF	F9 B9 99	79 39 19	18 26 26	Exchange TOS and NOS.
PUPI	9A	1A	16	Push floating point constant π onto TOS. Previous TOS becomes NOS.

Table 3
9511 COMMAND SUMMARY

COMMAND MNEMONIC	HEX CODE (sr = 1)	HEX CODE (sr = 0)	EXECUTION CYCLES	SUMMARY DESCRIPTION
16-BIT FIXED-POINT OPERATIONS				
SADD	EC	6C	16 - 18	Add TOS to NOS. Result to Pop stack
SSUB	ED	6D	30 - 32	Subtract TOS from NOS. Result to NOS. Pop stack
SMUL	EE	6E	84 - 94	Multiply NOS by TOS. Lower result to NOS. Pop stack
SMUU	F6	76	80 - 98	Multiply NOS by TOS. Upper result to NOS. Pop stack
SDIV	EF	6F	84 - 94	Divide NOS by TOS. Result to NOS. Pop stack
32-BIT FIXED-POINT OPERATIONS				
DADD	AC	2C	20 - 22	Add TOS to NOS. Result to NOS. Pop stack
DSUB	AD	2D	38 - 40	Subtract TOS from NOS. Result to NOS. Pop stack
DMUL	AE	2E	194-210	Multiply NOS by TOS. Lower result to NOS. Pop stack
DMUU	B6	36	182-218	Multiply NOS by TOS. Upper result to NOS. Pop stack
DDIV	AF	2F	196-210	Divide NOS by TOS. Result to NOS. Pop stack

Table 3 (cont'd)
9511 COMMAND SUMMARY

COMMAND MNEMONIC	HEX CODE (sr = 1)	HEX CODE (sr = 0)	EXECUTION CYCLES	SUMMARY DESCRIPTION
32-BIT FLOATING-POINT PRIMARY OPERATIONS				
FADD	90	10	54 -368	Add TOS to NOS. Result to NOS. Pop stack
FSUB	91	11	70 -370	Subtract TOS from NOS. Result to NOS. Pop stack
FMUL	92	12	146-168	Multiply NOS by TOS. Result to NOS. Pop stack
FDIV	93	13	154-184	Divide NOS by TOS. Result to NOS. Pop stack

Table 3 (cont'd)
9511 COMMAND SUMMARY

COMMAND MNEMONIC	HEX CODE (sr = 1)	HEX CODE (sr = 0)	EXECUTION CYCLES	SUMMARY DESCRIPTION
32-BIT FLOATING-POINT DERIVED OPERATIONS				
SQRT	81	01	782-870	Square Root of TOS. Result to NOS.
SIN	82	02	3796-4808	Sine of TOS. Result to TOS.
COS	83	03	3840-4878	Cosine of TOS. Result to TOS.
TAN	84	04	4894-5886	Tangent of TOS. Result to TOS.
ASIN	85	05	6230-7938	Inverse Sine of TOS. Result to TOS.
ACOS	86	06	6304-8284	Inverse Cosine of TOS. Result to TOS.
ATAN	87	07	4992-6536	Inverse Tangent of TOS. Result to TOS.
LOG	88	08	4474-7132	Common Logarithm of TOS. Result to TOS.
LN	89	09	4298-6956	Natural Logarithm of TOS Result to TOS.
EXP	8A	0A	3794-4878	e raised to power in TOS. Result to TOS.
PWR	8B	0B	8290-12032	NOS raised to power in TOS. Result to NOS. Pop stack

Table 3 (cont'd)
9511 COMMAND SUMMARY

PROGRAMMING THE 9511

The following paragraphs provide example routines and algorithms that represent various ways of using the 9511. These programs however, should be considered as a useful starting point for the construction of 9511 supporting software for a particular application. They should not be considered as the only means of using the 9511. The comments that are included in the source program should help to make the code easier to adapt to a specific need.

Reading the Status Register

The contents of the Status register can be read at any time by inputting on I/O address 0D5H. Keep in mind that when the Busy bit (Bit 7) is a logic "1", the other bits in the Status register will not be valid. Therefore, the program should loop until such time as the Busy bit becomes "0". Figure 1 gives an example program showing how to do a status fetching routine.

Data Transfer

Data transfers to and from the 9511 occur on I/O address 0D4H. The data is held in a push-down stack within the 9511. Thus, the least-significant byte is written first and read last. A transfer of data can be attempted at any time. However, if the 9511 is busy, it will cause the Z80 to wait until the instruction in progress is complete. This could cause excessive interrupt latency if the operation in progress is lengthy. The maximum latency is slightly over six milliseconds. If this is unacceptable, then data transfers should be only attempted when the 9511 is not busy. The routines shown in Figures 2 and 3 illustrate the transfer of 32-bit data to and from the 9511.

Commands

Commands are sent on I/O address 0D5H. As in the data transfer operation, there may be excessive interrupt latency if the 9511 is busy. However, often there is a data transfer between commands; and it is worth considering whether to call the routine in Figure 4 or to send the command using in-line code. While the routine in Figure 4 always works, it is extremly slow if the 9511 is known to be idle.

Interrupt

Normally, it is necessary to use the 9511 in an environment where interrupts are enabled. It is unusual to use the 9511 as an interrupt device itself. There are two reasons for this. First, nearly all systems have one task only that is not I/O driven. This task can simply wait for the 9511, since the results are required to proceed further; and nothing is gained by an interrupt from the 9511. The second reason is that many of the 9511 commands do not require a great deal of time; and the overload that is associated with processing and dismissing an interrupt may take more time than simply waiting. The following technique can be used or those special cases where the system throughout can be improved.

Setting the most-significant bit of the command byte will cause an interrupt at the completion of the instruction -- assuming that the interrupt output has been properly jumpered within the interrupt hardware. This interrupt can be only released by sending another command with a "zero" in the most-significant bit. Since the 9511 has means for an interrupt reset using a vectored interrupt acknowledge, the MULTIBUS has no such provisions. The simplest command to use is the NOP (No Op); but, this will clear the Status register. Therefore, all flags needed should be checked prior to dismissing the interrupt.

Data Conversion

The only time that the programmer must actually consider the floating-point format is for conversion between it and another form such as ASCII strings. These conversions are very adaptable through the use of a subtract and status check. Constants can be developed for future recall using this conversion approach.

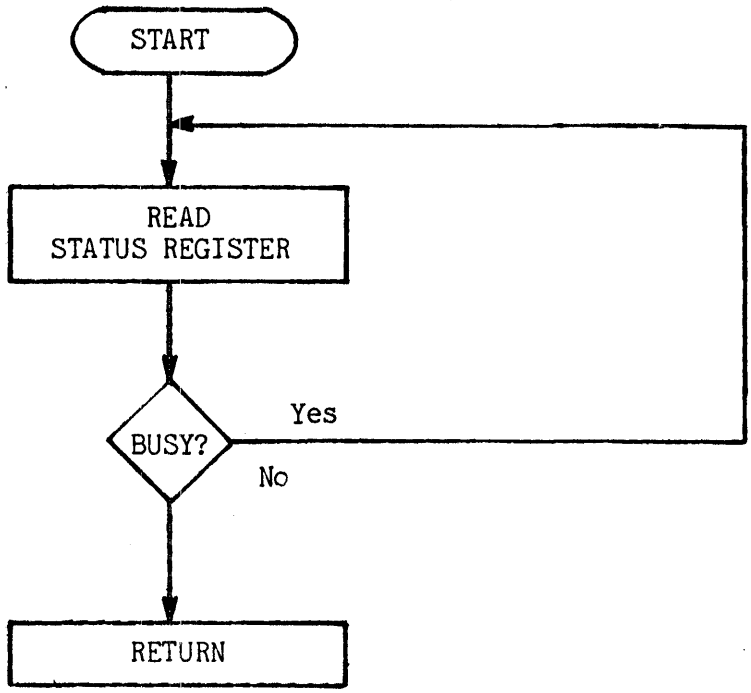
The data conversion technique to be described is not the fastest, but it will minimize the program's dependence on the floating-point format. This makes the user's program more adaptive to other formats such as that used by the 9511 Arithmetic Processor. Since the I/O device capability to handle unique data formats is the usual limit on speed of the data conversion, the speed of the code that is associated with 9511 is not too important.

The sample routines outlined in Figures 5 and 6 convert to and from ASCII strings held in memory. Adaption to BCD is simple. Integer binary can be handled directly by the 9511.

The format to be used is: +0.dddddd E+dd

The conversion to other formats is often times easier at the string level easier than inserting special cases at the binary floating-point level.

Flow Diagram:

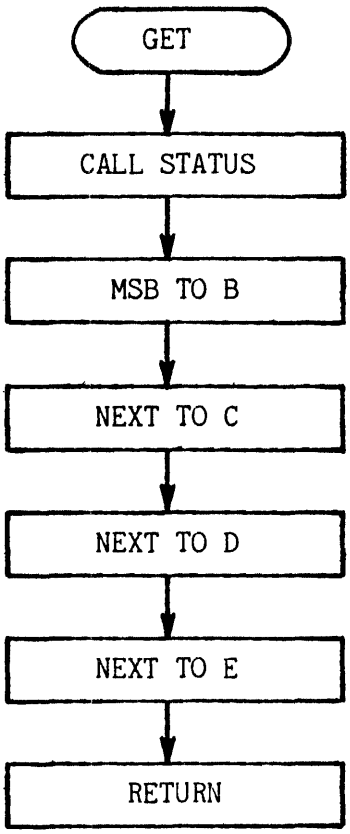


Source Program:

```
1                       ;STATUS REGISTER READ ROUTINE
2                       ;ENTRY: NOTHING NEEDED
3                       ;EXIT: A REG HOLDS STATUS
4                       ;9511 GUARANTEED NOT BUSY
5                       ;
6   0000 DB D5          STAT:   IN      0D5H        ;GET STATUS BIT
7   0002 B7                     ORA     A           ;SET Z80 FLAGS
8   0003 F0                     RP                  ;OK, VALID STATUS
9   0004 C3 00 00               JMP     STAT        ;STILL BUSY, TRY AGAIN
```

Figure 4
STATUS REGISTER READ ROUTINE

Flow Diagram:

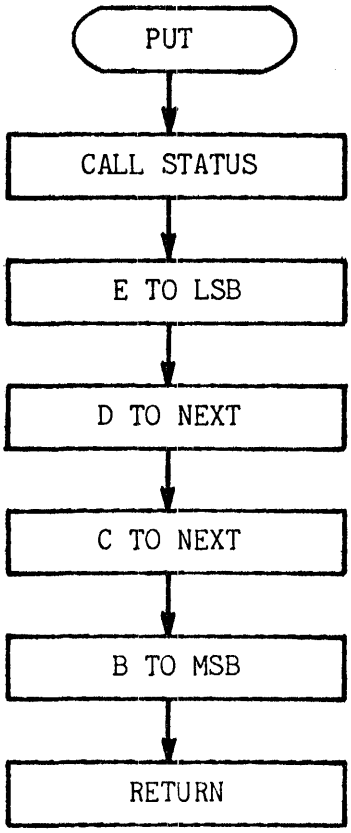


Source Program:

```
 1                      ;GETS DATA FROM 9511
 2                      ;ENTRY: NOTHING NEEDED
 3                      ;EXIT: A HAS STATUS, BCDE HAS FP NO.
 4                      ;9511 STACK IS POPPED 4 BYTES
 5                      ;
 6  0000 CD 12 00  GET:    CALL   STAT     ;GET STATUS, 9511 NOT BUSY
 7  0003 F5                PUSH   PSW      ;SAVE IT
 8  0004 DB D4             IN     0D4H
 9  0006 47                MOV    B,A      ;MOVE MSB
10  0007 DB D4             IN     0D4H
11  0009 4F                MOV    C,A      ;NEXT TO C
12  000A DB D4             IN     0D4H
13  000C 57                MOV    D,A      ;NEXT TO D
14  000D DB D4             IN     0D4H
15  000F 5F                MOV    E,A      ;MOVE LSB
16  0010 F1                POP    PSW
17  0011 C9                RET             ;DONE
18  0012               STAT    DS     1        ;CHECK BUSY
```

Figure 5
DATA TRANSFER ROUTINE

Flow Diagram:

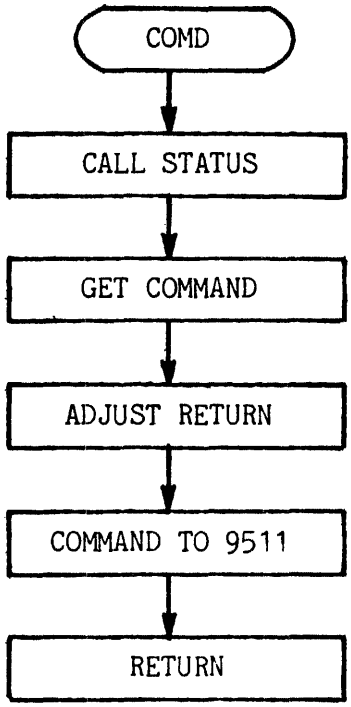


Source Program:

```
 1                      ;PUTS BDCE ON 9511 STACK
 2                      ;ENTRY: BCDE HAS FP NUMBER
 3                      ;EXIT: A HAS STATUS BEFORE PUSH
 4                      ;  9511 STACK PUSHED 4 BYTES
 5                      ;
 6  0000 CD 0F 00   PUT:    CALL   STAT      ;GET STATUS, 9511 NOT BUSY
 7  0003 F5                 PUSH   PSW       ;SAVE IT
 8  0004 7B                 MOV    A,E
 9  0005 D3 D4              OUT    0D4H      ;OUT LSB
10  0007 7A                 MOV    A,D
11  0008 D3 D4              OUT    0D4H      ;OUT NEXT
12  000A 78                 MOV    A,B
13  000B D3 D4              OUT    0D4H      ;OUT MSB
14  000D F1                 POP    PSW       ;RESTORE STATUS
15  000E C9                 RET              ;DONE
16  000F                STAT:   DS     1         ;CHECK BUSY
```

Figure 6
READ DATA ROUTINE

Flow Diagram:



Source Program:

```
 1                      ;DO COMMAND
 2                      ;ENTRY: COMMAND FOLLOWS CALL
 3                      ;EXIT: NOTHING CHANGED, COMMAND SENT
 4                      ;  9511 BUSY
 5                      ;
 6   0000 E3            COMD:   XTHL            ;GET RETURN ADDRESS
 7   0001 CD 0C 00              CALL    STAT     ;GET STATUS, 9511 NOT BUSY
 8   0004 F5                    PUSH    PSW      ;SAVE IT
 9   0005 7E                    MOV     A,M      ;GET COMMAND
10   0006 D3 D5                 OUT     0D5H     ;TO 9511
11   0008 F1                    POP     PSW
12   0009 23                    INX     H        ;ADJUST RETURN
13   000A E3                    XTHL
14   000B C9                    RET              ;DONE
15   000C               STAT:   DS      1        ;CHECK BUSY
```

Figure 7
COMMAND ROUTINE

Flow Diagram:

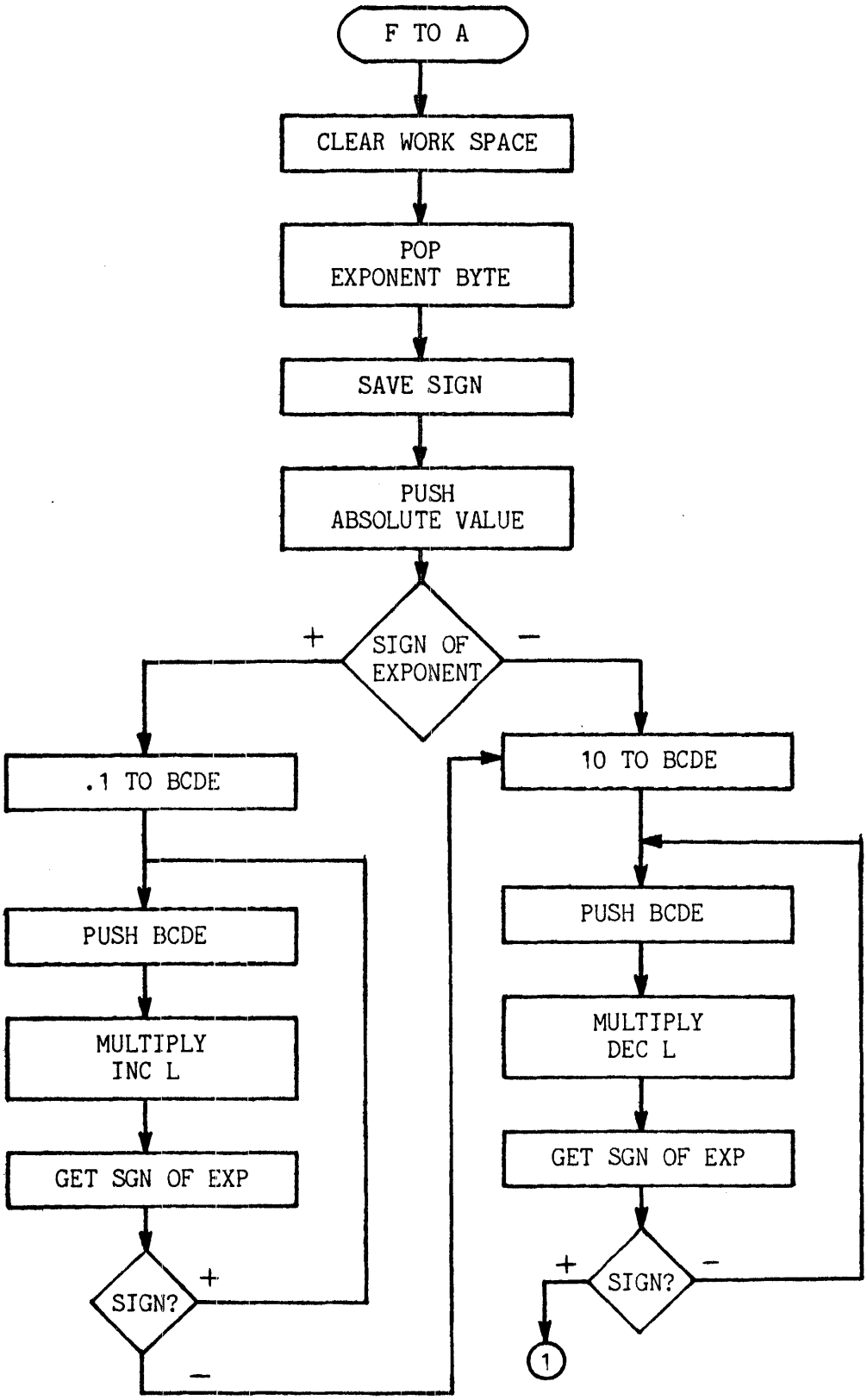
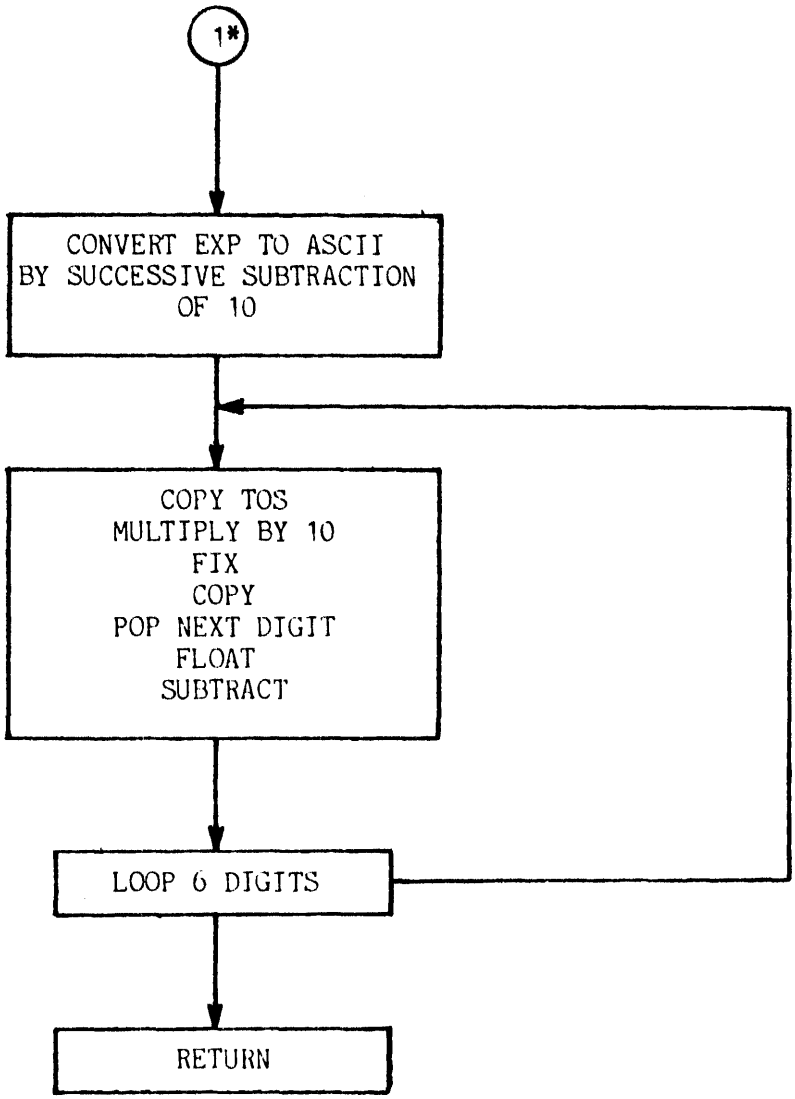


Figure 5
FLOATING POINT TO ASCII
CONVERSION ROUTINE



*NOTE: At this time, the exponent has been separated out. The number $0.1 < N < 1.0$ is in the 9511.

Figure 5 (cont'd)
FLOATING POINT TO ASCII
CONVERSION ROUTINE

Source Program:

```
 1                          ;FLOATING TO ASCII CONVERSION
 2                          ;ENTRY: IX POINTS TO BUFFER FOR ASCII
 3                          ;       NUMBER TO CONVERT IS IN 9511
 4                          ;EXIT: BUFFER FILLED, 9511 EMPTIED
 5                          ;      REGISTERS DESTROYED
 6                  00D4    FPDAT:  EQU     0D4H
 7                  0012    MPLY:   EQU     12H
 8                  001F    FIX16:  EQU     1FH
 9                  0037    DUP:    EQU     37H
10                  0077    DUP16:  EQU     77H
11  0000 21 00 FF           FTOA:   LXI     H,0FF00H  ;CLEAR DECIMAL EXPONENT
12  0003 DD 36 00 2B                MVIX    0,'+'     ;SET SIGN
13  0007 DD 36 01 30                MVIX    1,'0'     ;SET LEADING ZERO
14  000B DD 36 02 2E                MVIX    2,'.'     ;SET DECIMAL POINT
15  000F DD 36 09 45                MVIX    9,'E'     ;SET E
16  0013 DD 36 0A 2B                MVIX    10,'+'    ;SET EXP SIGN
17  0017 CD CF 00                   CALL    STAT      ;INSURE 9511 NOT BUSY
18  001A DB D4                      IN      FPDAT     ;GET EXPONENT BYTE
19  001C 47                         MOV     B,A       ;SAVE SIGN OF NUMBER
20  001D E6 7F                      ANI     07FH      ;MAKE ABSOLUTE
21  001F D3 D4                      OUT     FPDAT     ;PUT IT BACK
22  0021 78                         MOV     A,B       ;GET SIGN
23  0022 B7                         ORA     A         ;SET Z80 FLAGS
24  0023 F2 2A 00                   JP      FTOA1     ;NO, IS POSITIVE?
25  0026 DD 36 00 2D                MVIX    0,'-'     ;NO, SAY IT'S NEGATIVE
26  002A E6 40              FTOA1:  ANI     40H       ;EXTRACT EXPONENT SIGN
27  002C FA 49 00                   JM      FTOA3     ;IS MINUS
28  002F 01 00 0C                   LXI     B,0C00H
29  0032 11 CC CC                   LXI     D,0CCCCH  ;LOAD .1 TO BCDE
30  0035 CD D1 00           FTOA2:  CALL    PUT       ;.1 GOES TO 9511
31  0038 CD D0 00                   CALL    COMD      ;MULTIPLY BY .1
32  003B 12                         DB      MPLY
33  003C 2C                         INR     L
34  003D CD CF 00                   CALL    STAT      ;BUSY WAIT
35  0040 DB D4                      IN      FPDAT
36  0042 D3 D4                      OUT     FPDAT
37  0044 E6 40                      ANI     40H       ;GET EXPONENT SIGN
38  0046 CA 35 00                   JZ      FTOA2     ;STILL POSITIVE
39  0049 01 03 A0           FTOA3:  LXI     B,0A003H
40  004C 11 00 00                   LXI     D,0H      ;LOAD 10 GOES TO BCDE
```

Figure 5 (cont'd)
FLOATING POINT TO ASCII
CONVERSION ROUTINE

Source Program (cont'd):

```
41   004F CD D1 00          CALL   PUT        ;10 GOES TO 9511
42   0052 CD D0 00          CALL   COMD       ;MULTIPLY BY 10
43   0055 12                DB     MPLY
44   0056 2D                DCR    L          ;COUNT IT
45   0057 CD CF 00          CALL   STAT
46   005A DB D4             IN     FPDAT
47   005C D3 D4             OUT    FPDAT
48   005E E6 40             ANI    40H        ;GET SIGN OF EXPONENT
49   0060 C2 6E 00          JNZ    FTOA4      ;IF STILL NEGATIVE
50   0063 7D                MOV    A,L
51   0064 B7                ORA    A
52   0065 F2 8D 00          JP     FTOA5      ;IF DEC EXPONENT POSITIVE
53   0068 AF                XRA    A
54   0069 95                SUB    L          ;ABSOLUTE VALUE
55   006A DD 36 0A 2D       MVIX   10,'-'     ;CHANGE SIGN
56   006E CD D1 00    FTOA4: CALL   PUT
57   0071 CD D0 00          CALL   COMD       ;MULTIPLY BY 10
58   0074 12                DB     MPLY
59   0075 2D                DCR    L          ;COUNT IT
60   0076 CD CF 00          CALL   STAT
61   0079 DB D4             IN     FPDAT
62   007B D3 D4             OUT    FPDAT
63   007D E6 40             ANI    40H        ;GET SIGN OF EXPONENT
64   007F C2 6E 00          JNZ    FTOA4      ;IF STILL NEGATIVE
65   0082 7D                MOV    A,L
66   0083 B7                ORA    A
67   0084 F2 8D 00          JP     FTOA5      ;IF DEC EXPONENT POSITIVE
68   0087 AF                XRA    A
69   0088 95                SUB    L          ;ABSOLUTE VALUE
70   0089 DD 36 0A 2D       MVIX   10,'-'     ;CHANGE SIGN
71   008D 24          FTOA5: INR    H
72   008E DE 0A             SBI    10         ;EXTRACT MSD
73   0090 F2 8D 00          JP     FTOA5
74   0093 C6 3A             ADI    3AH        ;CONVERT TO ASCII
75   0095 DD 77 0C          MOVXR  12,A       ;STORE LSD
76   0098 7C                MOV    A,H
77   0099 B7                ORA    A
78   009A DD 77 0B          MOVXR  11,A       ;STORE MSD
79   009D 01 03 A0          LXI    B,0A003H
80   00A0 11 00 00          LXI    D,0H       ;FP 10
```

Figure 5 (cont'd)
FLOATING POINT TO ASCII
CONVERSION ROUTINE

Source Program (cont'd):

```
 81   00A3 2E 06              MVI    L,6       ;NO OF DIGITS TO DO
 82   00A5 DD E5              PUSHX            ;SAVE IX
 83   00A7 CD D0 00  FTOA6:   CALL   COMD      ;PUSH
 84   00AA 37                 DB     DUP       :COPY TOS TO NOS
 85   00AB CD D1 00           CALL   PUT       ;PUSH 10
 86   00AE CD D0 00           CALL   COMD      ;MPY
 87   00B1 12                 DB     MPLY
 88   00B2 CD D0 00           CALL   COMD      ;FIX
 89   00B5 1F                 DB     FIX16     ;FIX IT
 90   00B6 CD D0 00           CALL   COMD
 91   00B9 77                 DB     DUP16     ;SAVE IT
 92   00BA CD CF 00           CALL   STAT      ;BUSY CHECK
 93   00BD DB D4              IN     FPDAT
 94   00BF DB D4              IN     FPDAT     ;DECIMAL DIGIT
 95   00C1 F6 30              ORI    30H       ;MAKE ASCII
 96   00C3 DD 77 03           MOVXR  3,A       ;STORE DIGIT
 97   00C6 DD 23              INXX             ;BUMP TO NEXT DIGIT
 98   00C8 2D                 DCR    L
 99   00C9 C2 A7 00           JNZ    FTOA6     ;DO IT 6 TIMES
100   00CC DD E1              POPX             ;RESTORE IX
101   00CE C9                 RET              ;STOP
102   00CF            STAT:   DS     1         ;CHECK BUSY
103   00D0            COMD:   DS     1         ;COMD
104   00D1            PUT:    DS     1
```

Figure 5 (cont'd)
FLOATING POINT TO ASCII
CONVERSION ROUTINE

Flow Diagram:

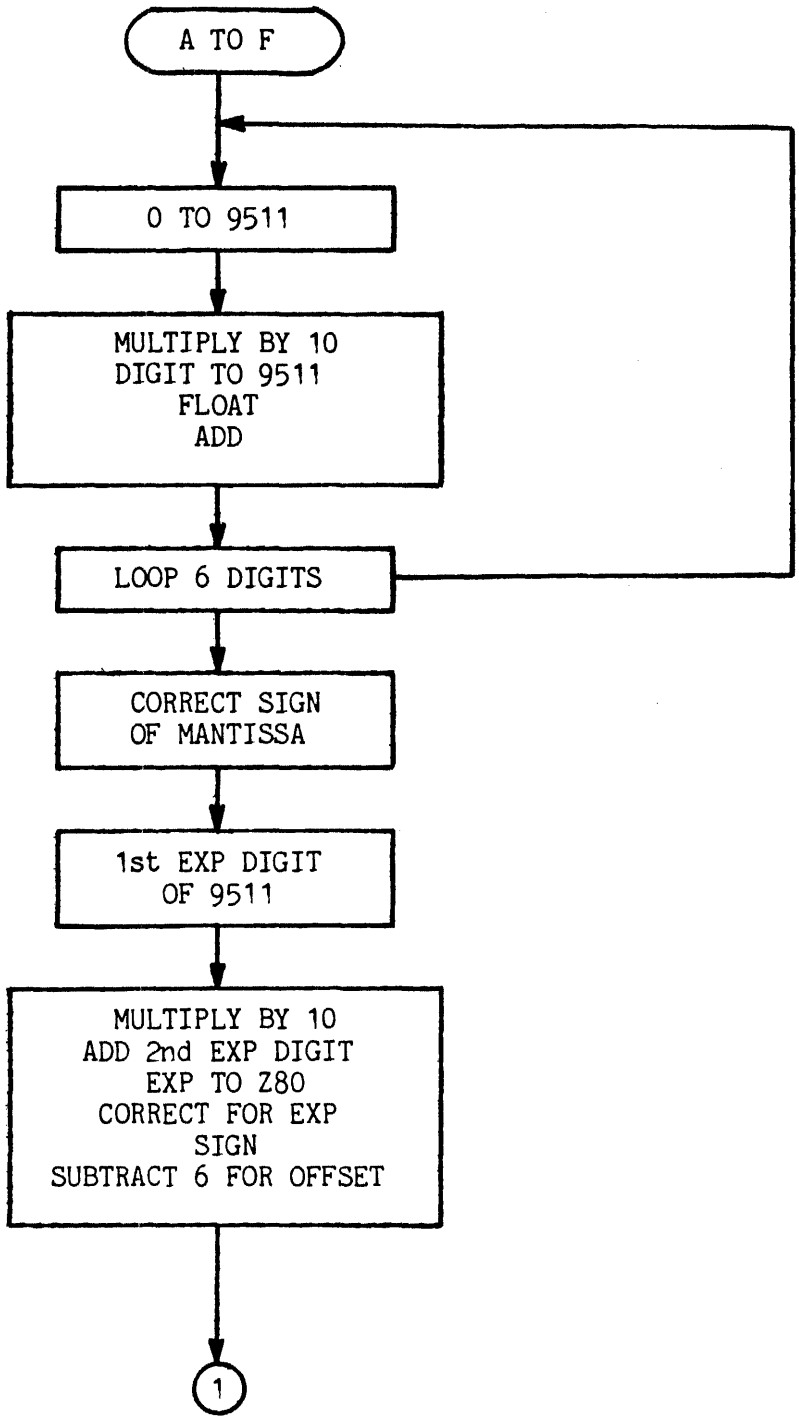


Figure 6
ASCII TO FLOATING POINT
CONVERSION ROUTINE

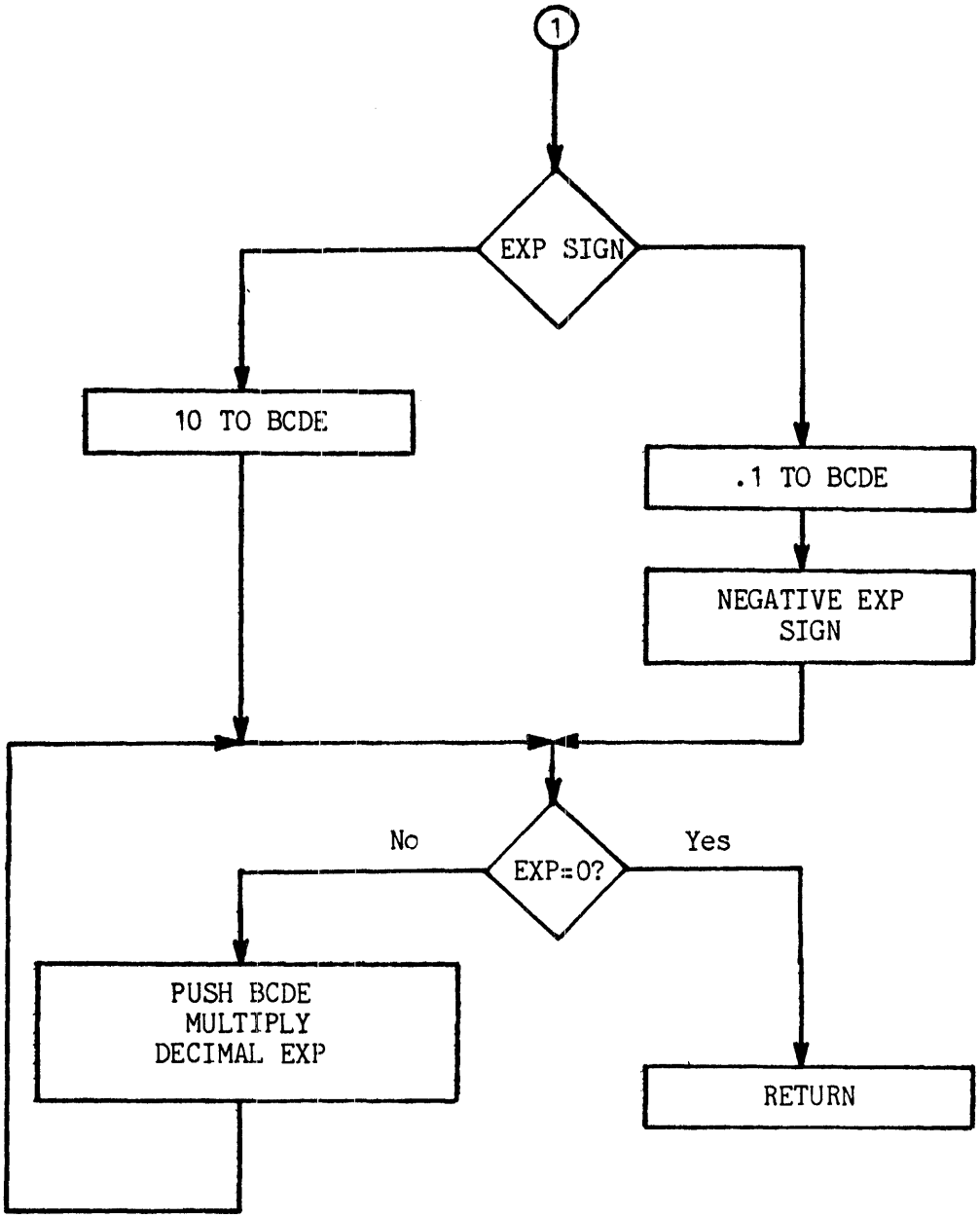


Figure 6 (cont'd)
ASCII TO FLOATING POINT ROUTINE

Source Program:

```
 1                          ;ASCII TO FLOATING POINT
 2                          ;ENTRY: IX POINTS TO BUFFER FOR ASCII
 3                          ;EXIT: 9511 FILLED
 4                          ;      REGISTERS DESTROYED
 5                          ;
 6                  00D4    FPDAT:  EQU     0D4H
 7                  0010    ADD:    EQU     10H
 8                  0012    MPLY:   EQU     12H
 9                  001D    FLT16:  EQU     1DH
10                  006C    ADD16:  EQU     6CH
11                  0076    MPLY16: EQU     76H
12  0000 01 00 00           ATOF:   LXI     B,0H
13  0003 11 00 00                   LXI     D,0H
14  0006 CD 90 00                   CALL    PUT         ;0 TO 9511
15  0009 01 03 A0                   LXI     B,0A003H
16  000C 11 00 00                   LXI     D,0H        ;FP 10
17  000F DD E5                      PUSHX               ;SAVE POINTER
18  0011 6E                         MOV     L,6         ;NO. OF DIGITS
19  0012 CD 90 00           ATOF1:  CALL    PUT         ;10 TO 9511
20  0015 CD 8E 00                   CALL    COMD
21  0018 12                         DB      MPLY        ;SHIFT
22  0019 DD 7E 03                   MOVRX   A,3         ;GET DIGIT
23  001C E6 0F                      ANI     0FH         ;MASK FROM ASCII
24  001E D3 D4                      OUT     FPDAT
25  0020 AF                         XRA     A
26  0021 D3 D4                      OUT     FPDAT       ;DIGIT TO 9511
27  0023 CD 8E 00                   CALL    COMD
28  0026 1D                         DB      FLT16       ;CONVERT TO FLOATING POINT
29  0027 CD 8E 00                   CALL    COMD
30  002A 10                         DB      ADD         ;ADD TO RUNNING SUM
31  002B 2D                         DCR     L
32  002C C2 12 00                   JNZ     ATOF1       ;DO ALL DIGITS
33  002F DD E1                      POPX
34  0031 DD 7E 00                   MOVRX   A,0         ;GET SIGN
35  0034 FE 2D                      CPI     '-'
36  0036 C2 3F 00                   JNZ     ATOF2       ;POSITIVE
37  0039 DB D4                      IN      FPDAT
38  003B F6 80                      ORI     80H
39  003D D3 D4                      OUT     FPDAT       ;FLIP SIGN OF 9511
40  003F DD 7E 0B           ATOF2:  MOVRX   A,11        ,FIRST DIGIT OF EXPONENT
```

Figure 6 (cont'd)
ASCII TO FLOATING POINT
CONVERSION ROUTINE

Source Program (cont'd):

```
41   0042 E6 0F              ANI     0FH
42   0044 D3 D4              OUT     FPDAT
43   0046 AF                 XRA     A
44   0047 D3 D4              OUT     FPDAT       ;TO 9511
45   0049 3E 0A              MVI     A,10
46   004B D3 D4              OUT     FPDAT
47   004D AF                 XRA     A
48   004E D3 D4              OUT     FPDAT       ;10 TO 9511
49   0050 CD 8E 00           CALL    COMD
50   0053 76                 DB      MPLY16      ;MULTIPLY
51   0054 CD 8F 00           CALL    STAT
52   0057 DD 7E 0C           MOVRX   A,12
53   005A E6 0F              ANI     0FH
54   005C D3 D4              OUT     FPDAT       ;2ND DIGIT
55   005E CD 8E 00           CALL    COMD
56   0061 6C                 DB      ADD16       ;ADD IT
57   0062 CD 8F 00           CALL    STAT
58   0065 DB D4              IN      FPDAT
59   0067 DB D4              IN      FPDAT       ;GOT ABSOLUTE EXP.
60   0069 DD 7E 0A           MOVRX   A,10
61   006C FE 2D              CPI     '-'         ;SIGN OF EXP.
62   006E C2 73 00           JNZ     ATOF3
63   0071 BF                 CMP     A
64   0072 3C                 INR     A
65   0073 DE 06      ATOF3:  SBI     6           ;CORRECT FOR DECIMAL POINT
66   0075 C8                 RZ
67   0076 6F                 MOV     L,A
68   0077 F2 82 00           JP      ATOF4
69   007A BF                 CMP     A
70   007B 3C                 INR     A
71   007C 01 00 0C           LXI     B,0C00H
72   007F 11 CC CC           LXI     D,0CCCCH    ;LOAD BCDE WITH .1
73   0082 CD 90 00   ATOF4:  CALL    PUT
74   0085 CD 8E 00           CALL    COMD
75   0088 12                 DB      MPLY        ;DO DECIMAL SHIFT
76   0089 2D                 DCR     L
77   008A C2 82 00           JNZ     ATOF4       ;UNTIL EXHAUSTED
78   008D C9                 RET
79   008E            COMD:   DS      1           ;COMMAND
80   008F            STAT:   DS      1           ;CHECK BUSY

81   0090            PUT:    DS      1
```

Figure 6 (cont'd)
ASCII TO FLOATING POINT
CONVERSION ROUTINE

Appendix E

MSC 8009 MEMORY CONTROLLER PIGGYBACK BOARD DESCRIPTION

MEMORY CONTROLLER PIGGYBACK BOARD (303-0292-00x) DESCRIPTION

The memory controller board has two functional blocks; a power-up reset generator, and the memory controller.

The power-up reset generator consists of U4,R1,LR1,C4 and C6. When power is first applied to the board, the output (pin 3) of U4, a 555 Timer IC, is low. After a time delay set by R1 and C4, the output goes high. This releases the reset condition that was asserted on the INIT line when power was applied.

The memory controller is made up of two parts, U2, a 74LS393 is the refresh interval timer. U1, a programmed 16R8 PAL and U5, a 74S112 dual flipflop, generate the memory control signals, and arbitrate between memory and refresh cycles. U3, a 74LS240 provides buffering and additional drive for the memory array.

A memory cycle may be started by either an externally generated signal, RMRQ/, or an internally generated refresh request. Refer to Fig. 1 for timing diagram of a memory cycle.

A refresh cycle will be started by one of two conditions. Either 14 microseconds have passed without a refresh cycle being requested, or a processor M1 cycle has just completed and a refresh cycle has not occurred in the last 14 microseconds.

A RMRQ/ memory cycle will be serviced before a refresh cycle if requests arrive at the same time. If a refresh cycle is in process when a RMRQ/ cycle is requested, the RMRQ/ will be held off until the refresh cycle is complete. This situation should only occur during DMA accesses, since refreshes are transparent to the processor.

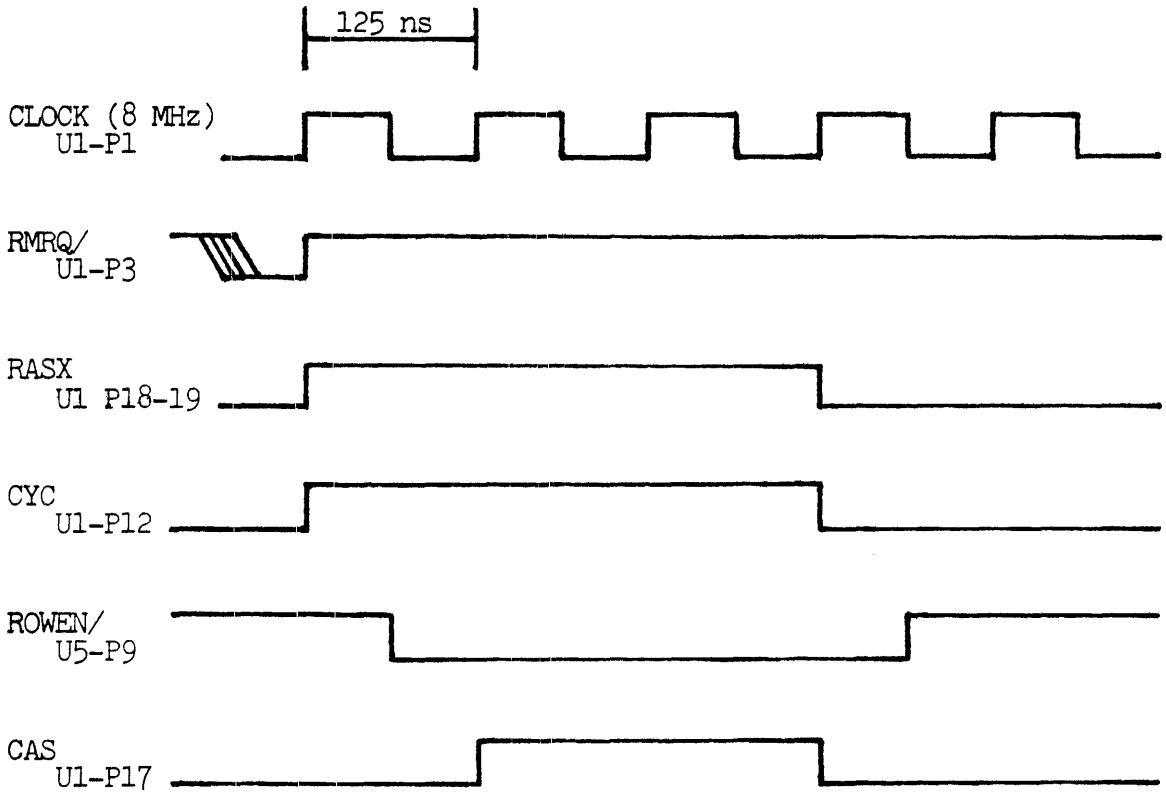
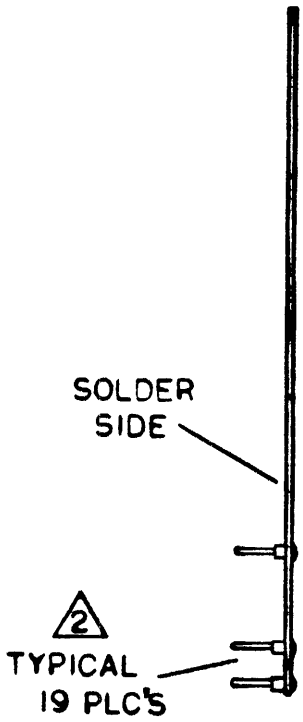
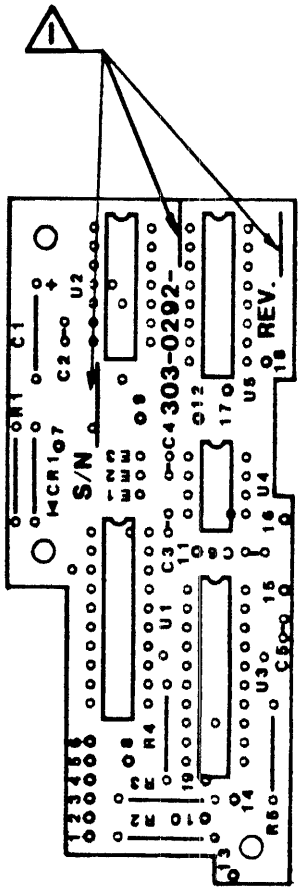


Figure 1-1
MEMORY CYCLE TIMING DIAGRAM





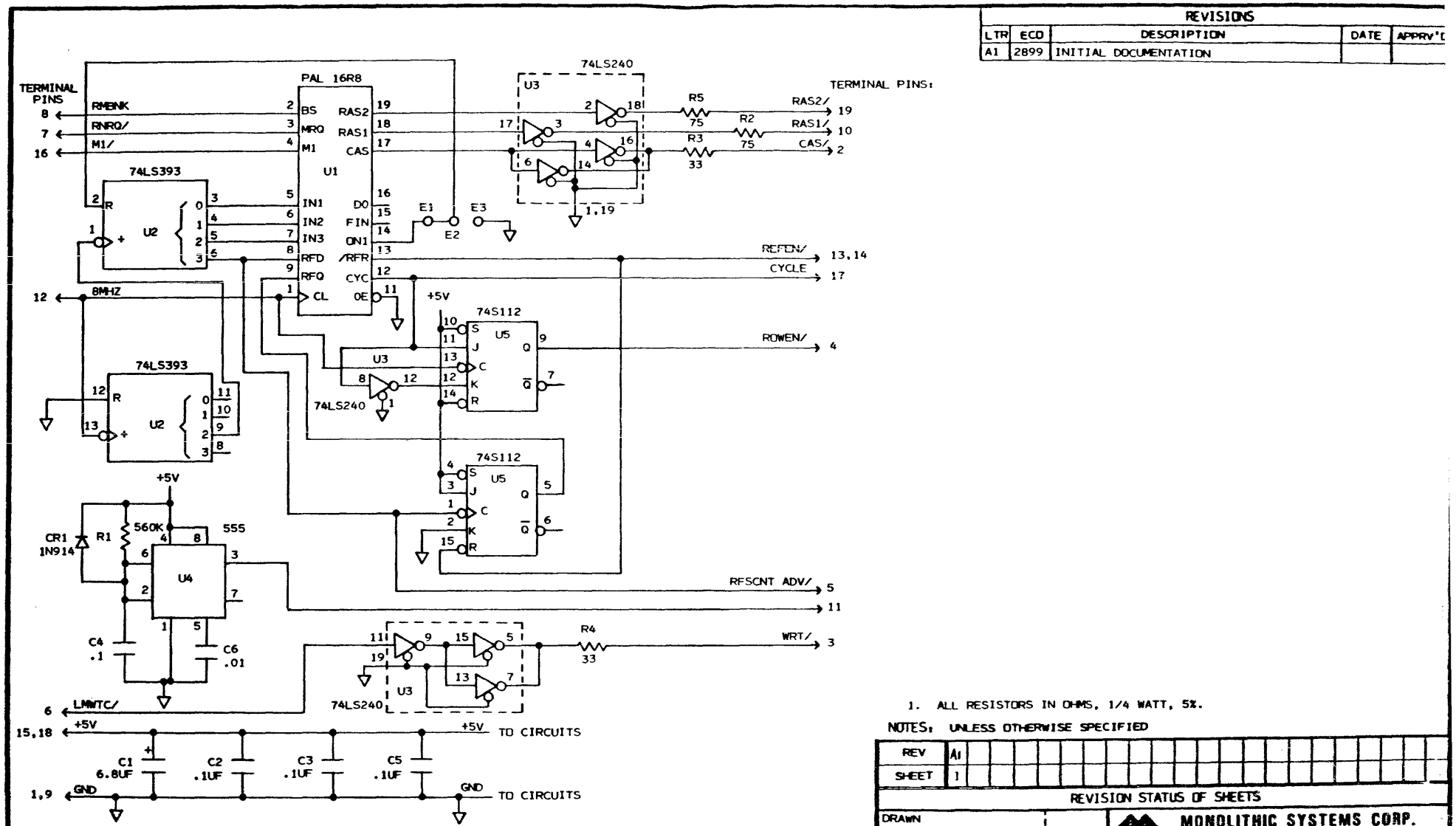
COMPONENT SIDE

(SEE SH. 1)

NOTES: UNLESS OTHERWISE SPECIFIED:

SIZE	CODE IDENT NO.	DWG NO.	
A	51513	303-0292-000	
SCALE	REV	SHEET	
	A1	2 OF 2	

MSF-1005 REV A



REVISIONS				
LTR	ECO	DESCRIPTION	DATE	APPRV'D
A1	2899	INITIAL DOCUMENTATION		

1. ALL RESISTORS IN OHMS, 1/4 WATT, 5%.

NOTES: UNLESS OTHERWISE SPECIFIED

REV	A1
SHEET	1

REVISION STATUS OF SHEETS

DRAWN	
CHECKED	
APPROVED	
APPROVED	

MONOLITHIC SYSTEMS CORP.
Englewood, Colorado 80112

TITLE: PCA SCHEMATIC
800X MEMORY CONTROLLER

SIZE	CODE IDENT NO.	DWG NO.	REV
B	51513	305-0292-000	A1

SCALE: | SHEET 1 of 1

LAST USED	
NOT USED	

REFERENCE DESIGNATIONS LAST USED/NOT USED

MSF 1052

REVISIONS				
ECO	LTR	DESCRIPTION	DATE	APPROVED
2864	B	RELEASED - REVISED PER ECO		
2878	C	REVISED PER ECO Rel'd 6/23/81	6-25-81	

(2) CUSTOMER SUPPLIED.

(1) FOR STAMPING INFORMATION, REFER TO 313-0003-006 & -007.

NOTES: UNLESS OTHERWISE SPECIFIED

REFER TO PRIME	
NEXT ASSY	
APPLICATION	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS ± DECIMALS XX ± .XXX ± ANGLES ±

MATERIAL: REFER TO PRIME

FINISH:

DO NOT SCALE DRAWING

CONTRACT NO.

APPROVALS	DATE
DRAWN	6/8/81
CHECKED	6/23/81
APPROVED	
APPROVED	

PHASE: RELEASED

REV	C	B												
SHEET	1	2												

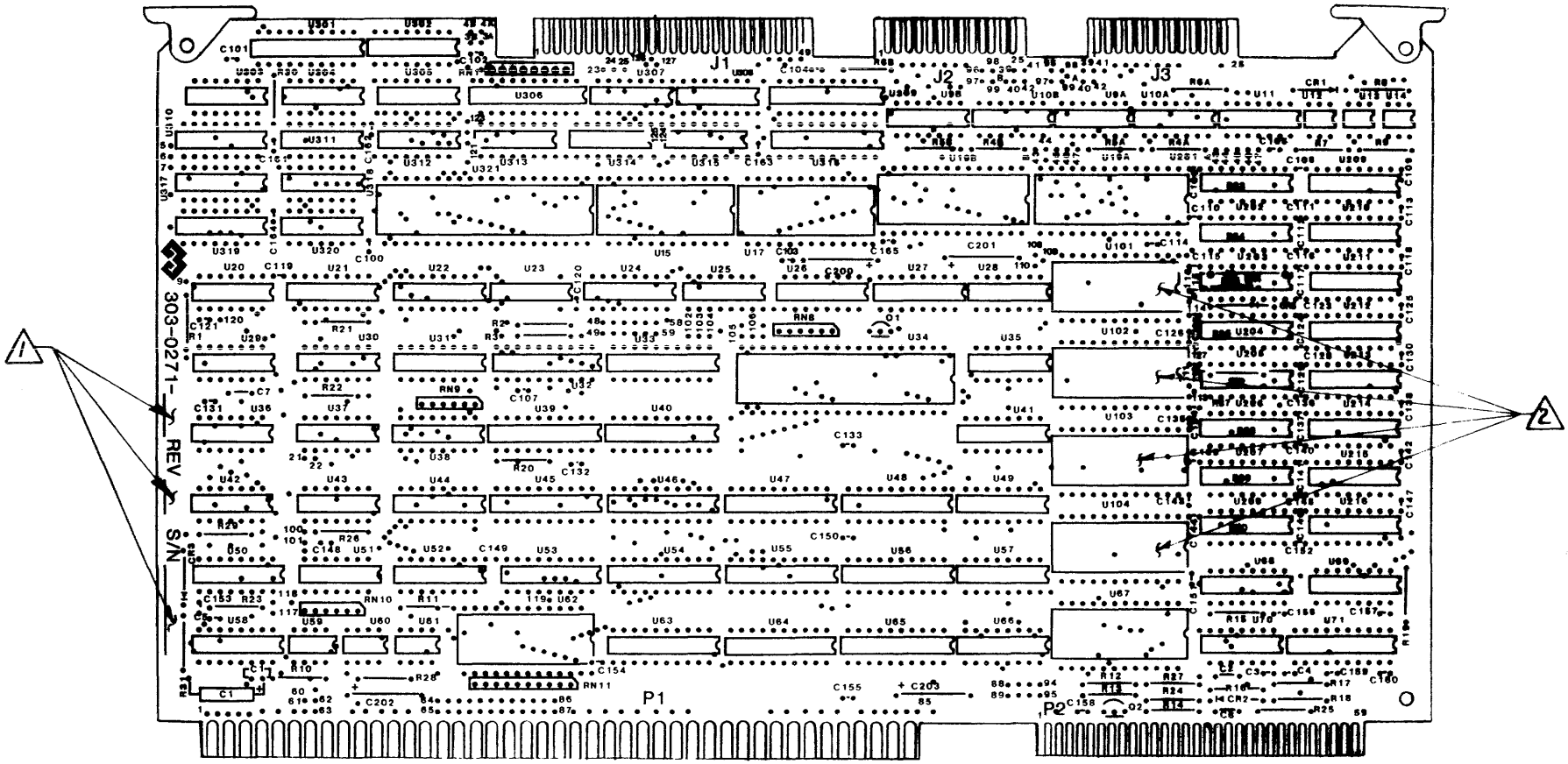
REVISION STATUS OF SHEETS

MONOLITHIC SYSTEMS CORP.
ENGLEWOOD, COLORADO 80112

TITLE: P.C.A. DRAWING
MSC 8009

SIZE	CODE IDENT NO	DRAWING NO	REV
B	51513	303-0271-000	C

SCALE | SHEET 1 OF 2



(SEE SH.1)

NOTES: UNLESS OTHERWISE SPECIFIED.

SIZE	CODE IDENT NO	DWG NO	REV
B	51513	303-0271-000	B
SCALE AIMAC		SHEET 2	

MSE-1008A

REVISIONS				
LTR	ECO	DESCRIPTION	DATE	APPRV'D
B	2864	RELEASED - REVISED PER ECO		
C	2878	REVISED PER ECO	6 30 91	

7. JUMPER TABLE

W/W PINS	8"(STD)	5"
121-122	--	JUMPER
122-123	JUMPER	--
124-125	--	*
126-127	--	JUMPER

* REFER TO MANUAL OR CONTACT MSC, WHEN USING 4, 5" DISK DRIVES.

8. MEMORY ARRAY JUMPER TABLE

	16K 5V TYPE	16K 4116 TYPE	64K 5V TYPE	64K MB8164 (+7,-2V)
JUMPERS	108-109	109-110	108-109	109-110
	111-112	111-112	112-113	114-115
	115-116	114-115	115-116	112-113
CR4,CR5	-	-	-	INSTALLED*
R32-R40	-	-	-	INSTALLED
U201-208	INSTALLED	INSTALLED	-	-
U209-216	INSTALLED FOR 32K	INSTALLED FOR 32K	INSTALLED	INSTALLED

* RUN UNDER DIODES MUST BE CUT WHEN DIODES ARE INSTALLED.

8. REFER TO MEMORY ARRAY JUMPER TABLE FOR JUMPER OPTIONS FOR DIFFERENT TYPES OF MEMORY ELEMENTS.

7. SEE TABLE FOR JUMPER OPTION FOR 5" & 8" FLOPPY DRIVE.

6. E E = JUMPER BLOCK AND E E = JUMPER WIRE.

5. EPROM/ROM MEMORY FOR THESE LOCATIONS ARE CUSTOMER SUPPLIED. FOR A COMPLETE LIST OF USABLE DEVICES. SEE USER MANUAL OR CONTACT MONOLITHIC SYSTEMS CORP., 84 INVERNESS CIRCLE EAST, ENGLEWOOD, COLORADO, 80112.

4. ALL CAPACITORS ARE IN UF.

3. PIN 1 IS COMPONENT SIDE, AND OPPOSITE PIN 2.

2. CAUTION: ONLY ONE, U9A OR U11 INSTALLED AT ONE TIME.

1. ALL RESISTANCES IN OHM, 5%, 1/4W.

NOTES: UNLESS OTHERWISE SPECIFIED

REV	C	B	B	B	B	B	B	B	B	B	B	B	B										
SHEET	1	2	3	4	5	6	7	8	9	10	11	12	13										

REVISION STATUS OF SHEETS

LAST USED	C203	CR5	R40	RN11	U126	U321			
NOT USED	C8-99, C166-199			RN2-7	E1,2,8,10-20,26-38	U1,8,16,18, U7-99	U105-200, U217-300		

REFERENCE DESIGNATIONS LAST USED/NOT USED

DRAWN 7/21/90

CHECKED

APPROVED

APPROVED

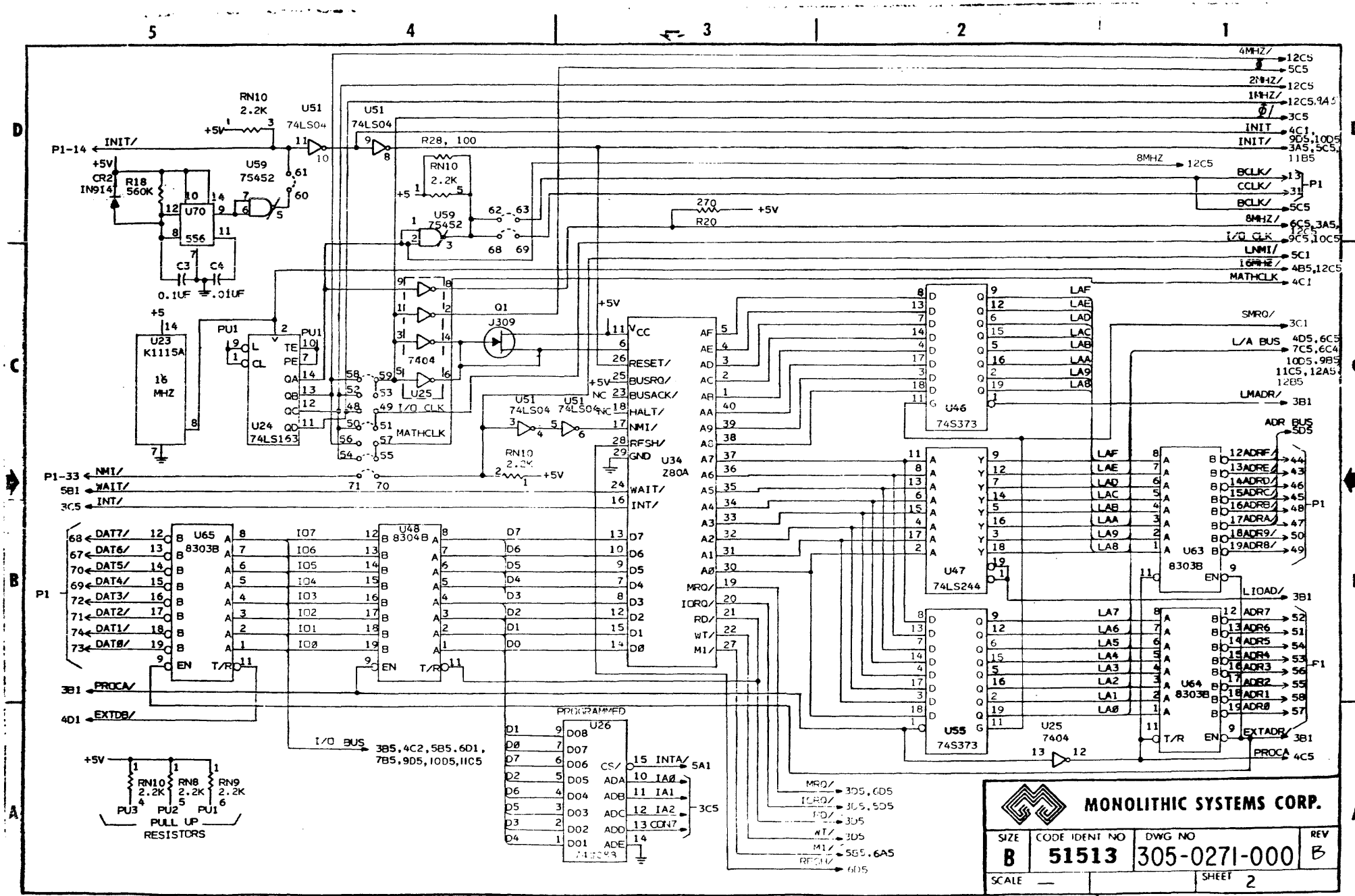
MONOLITHIC SYSTEMS CORP.
Englewood, Colorado 80112

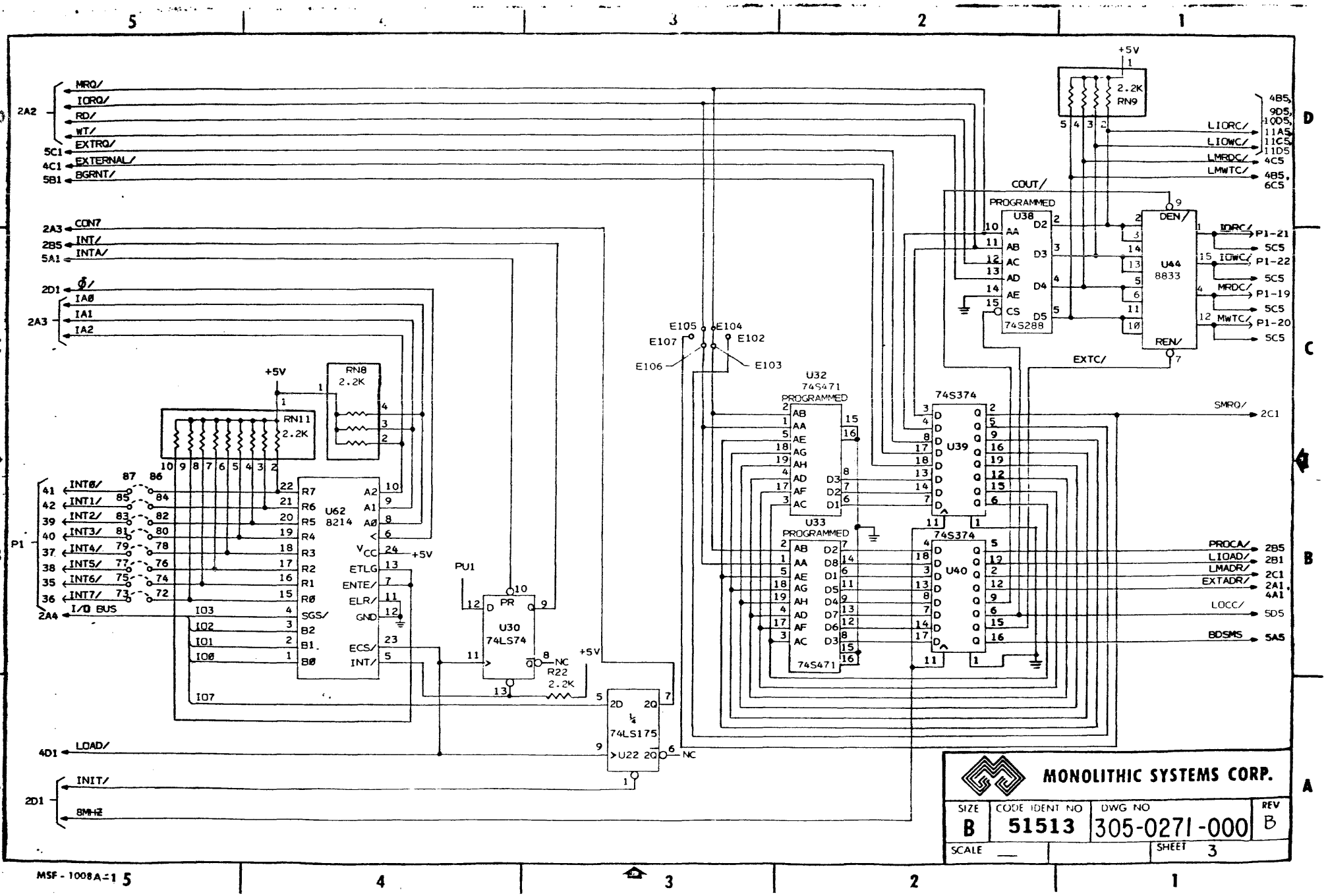
TITLE P.C. SCHEMATIC, MSC 8009

SIZE B | CODE IDENT NO. 51513 | DWG NO. 305-0271-000 | REV C

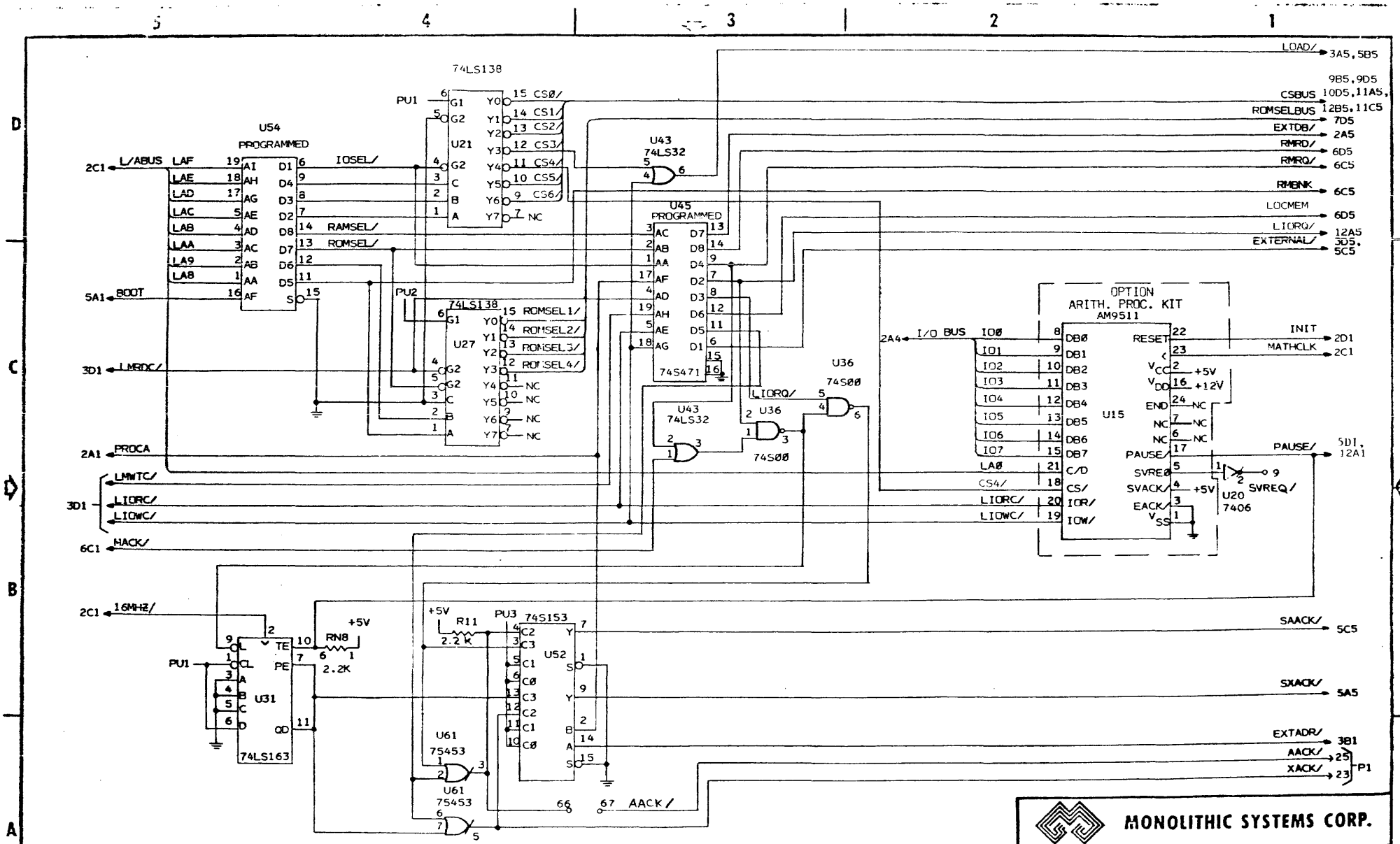
SCALE —— | SHEET 1 OF 13

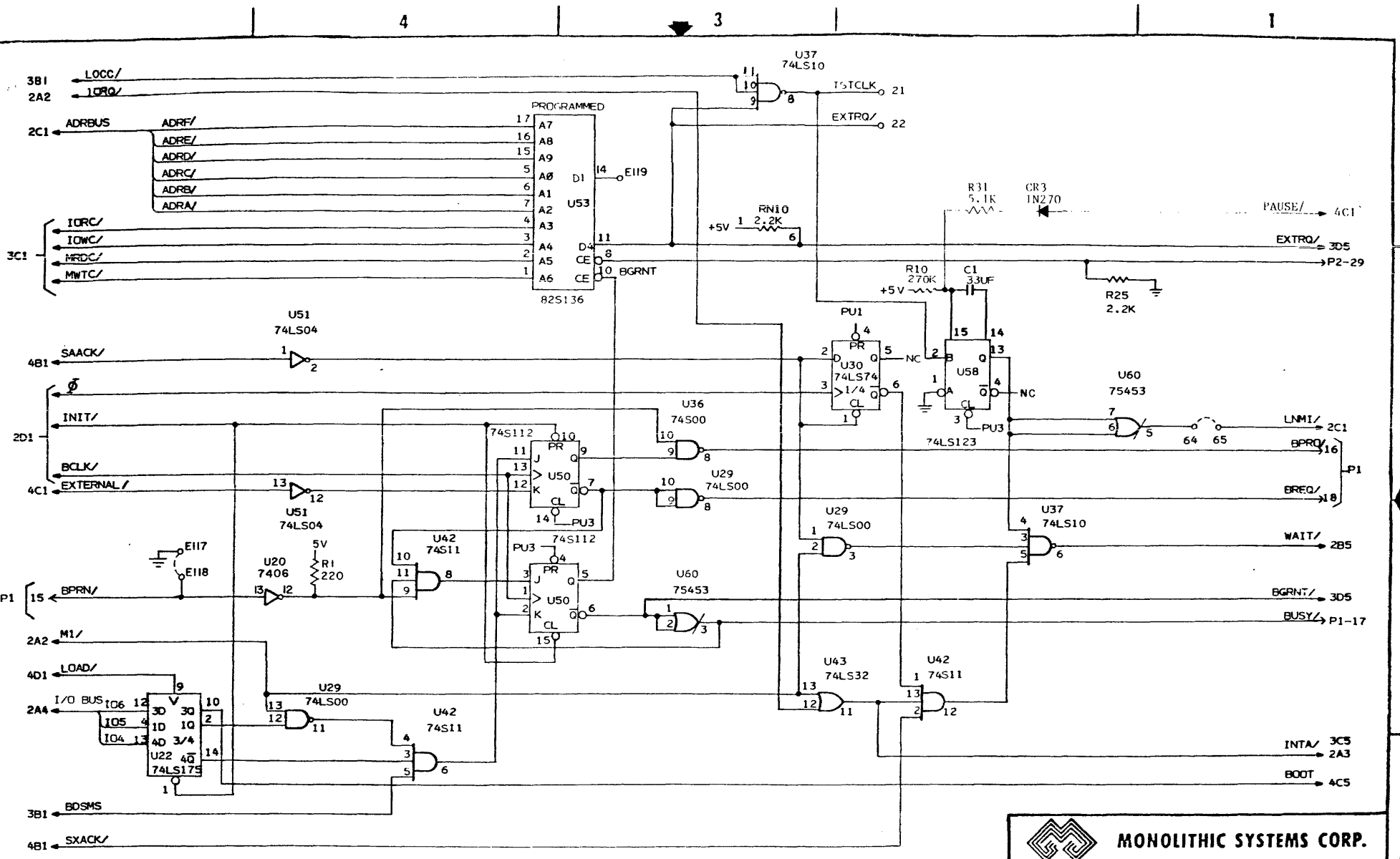
MSF 1052


MONOLITHIC SYSTEMS CORP.
SIZE B
CODE IDENT NO 51513
DWG NO 305-0271-000
REV B
SCALE —
SHEET 2
MSF-1008A-1
INIT/
NMI/
WAIT/
INT/
PROCA/
EXTDB/
I/O BUS
PULL UP RESISTORS
U23 K1115A 16 MHZ
U24 74LS163
U25 7404
U26 PROGRAMMED
U34 Z80A
U46 74S373
U47 74LS244
U55 74S373
U63 8303B
U64 8303B
U65 8303B
U48 8304B
U51 74LS04
U59 75452
U70 556
Q1 J309
MATHCLK
I/O CLK
4MHZ/
2MHZ/
1MHZ/
8MHZ
BCLK/
CCLK/
LNMI/
16MHZ/
SMRQ/
L/A BUS
LMADR/
ADR BUS 5D5
LIOAD/
EXTADR/
MRQ/
IORQ/
RD/
WT/
M1/
RFSH/

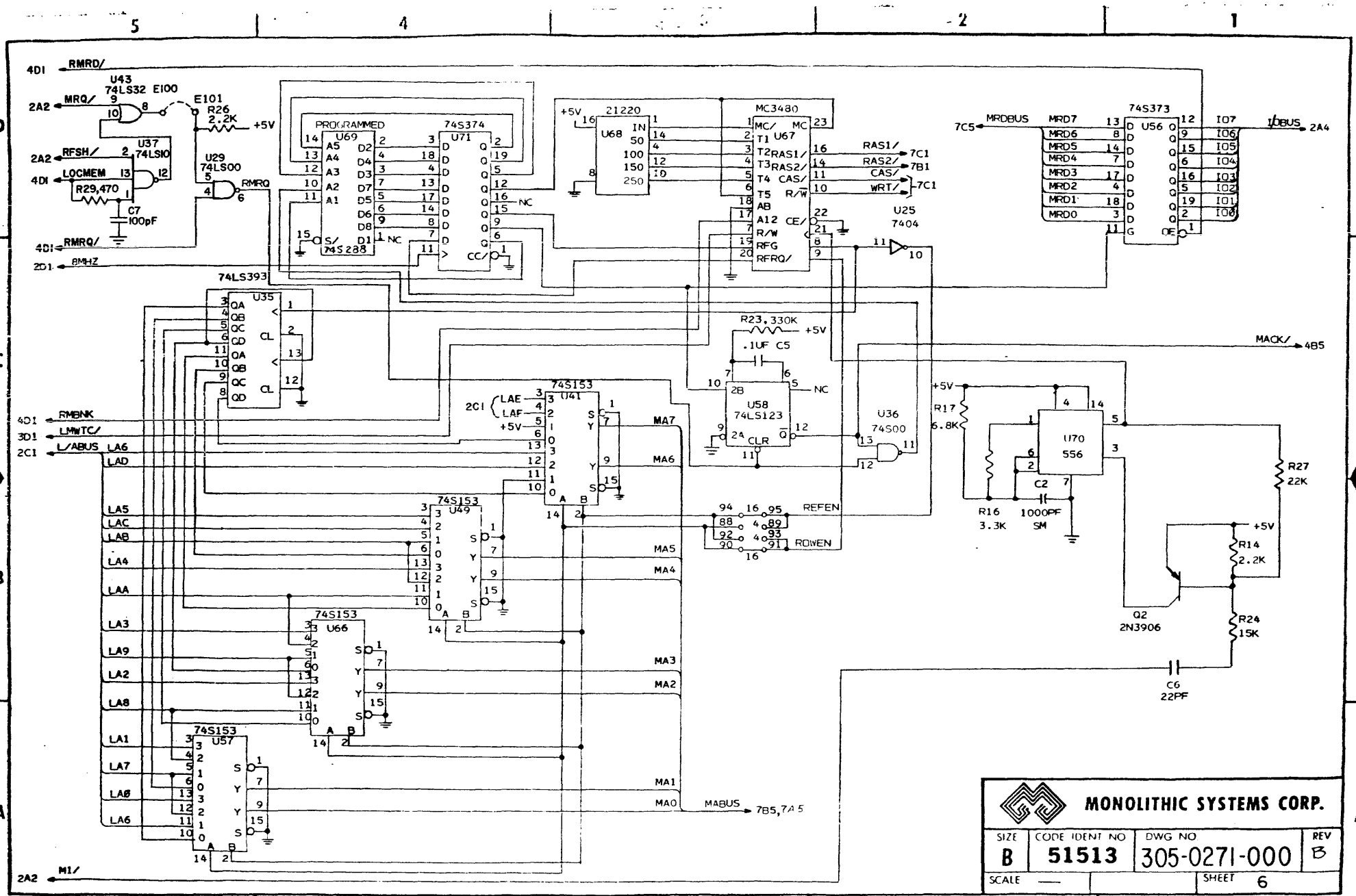


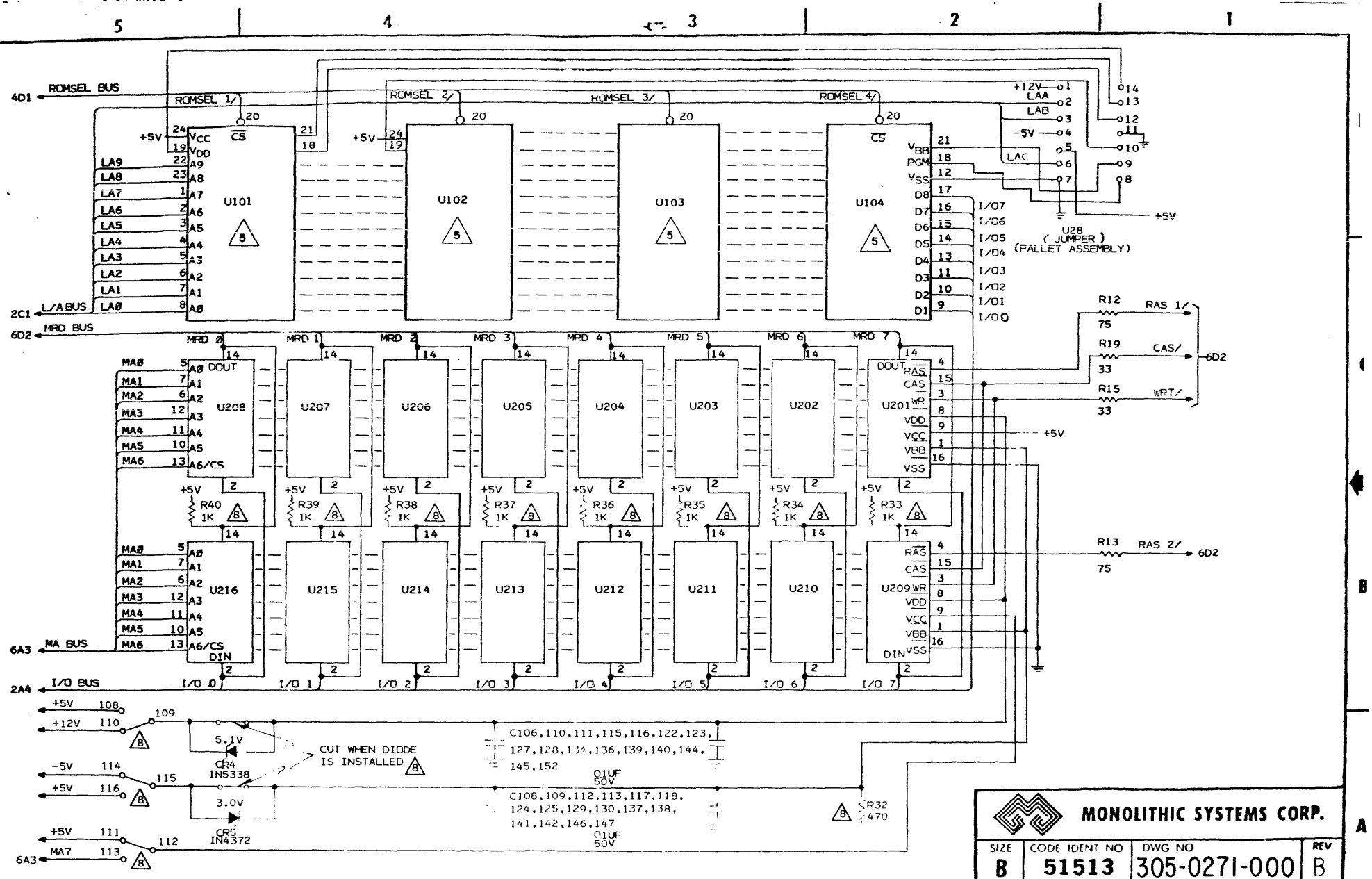
MONOLITHIC SYSTEMS CORP.			
SIZE	CODE IDENT NO	DWG NO	REV
B	51513	305-0271-000	B
SCALE —		SHEET 3	


74LS138
U54
PROGRAMMED
U21
U45
PROGRAMMED
U43
74LS32
U27
U36
74S00
U31
74LS163
U52
74S153
U61
75453
OPTION
ARITH. PROC. KIT
AM9511
U15
U20
7406
R11
2.2 k
RN8
2.2K
+5V
L/ABUS
LAF
LAE
LAD
LAC
LAB
LAA
LA9
LA8
IOSEL/
RAMSEL/
ROMSEL/
BOOT
PROCA
LMWTC/
LIORC/
LIOWC/
HACK/
16MHZ/
LMRDC/
I/O BUS
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
LA0
C/D
CS/
IOR/
IOW/
RESET
PAUSE/
SVREQ/
SVACK/
EACK/
INIT
MATHCLK
LIORQ/
AACK/
LOAD/
CSBUS
ROMSELBUS
EXTDB/
RMRD/
RMRQ/
RMBNK
LOCMEM
EXTERNAL/
SAACK/
SXACK/
EXTADR/
XACK/
MONOLITHIC SYSTEMS CORP.
SIZE B
CODE IDENT NO 51513
DWG NO 305-0271-000
REV B
SCALE —
SHEET 4
MSF - 1008A-1

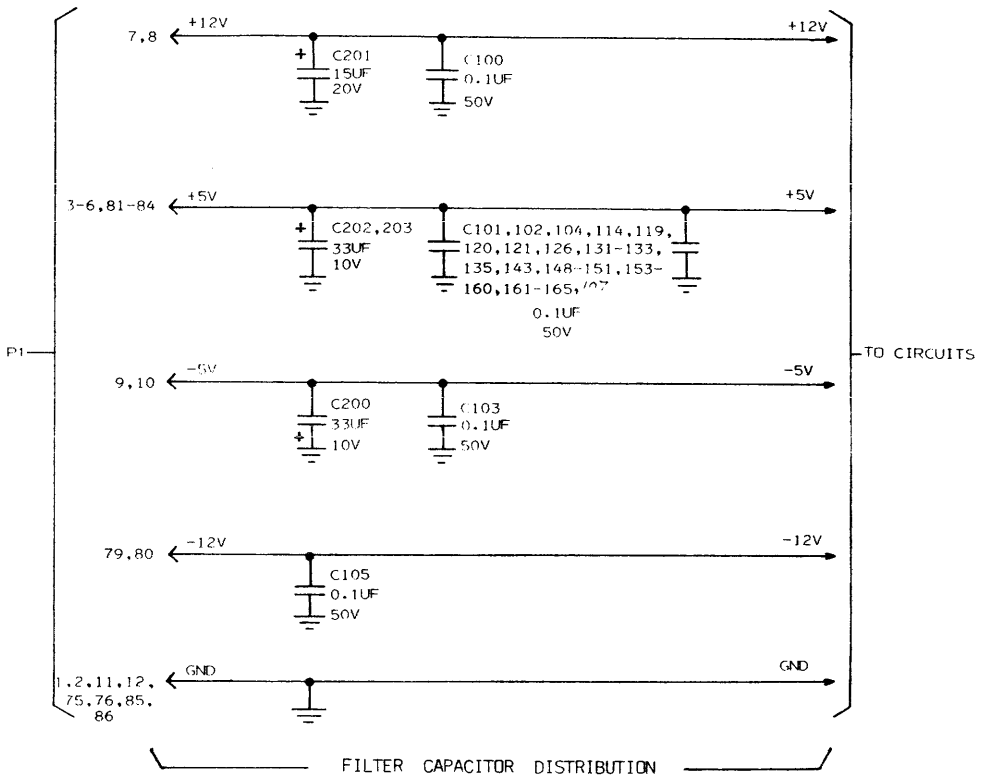


MONOLITHIC SYSTEMS CORP.			
SIZE B	CODE IDENT NO 51513	DWG NO 305-0271-000	REV B
SCALE —		SHEET 5	


4D1 RMRD/
U43 74LS32 E100
2A2 MRQ/
E101
R26 2.2K +5V
U37 74LS10
2A2 RFSH/
LOCMEM
R29,470
C7 100pF
U29 74LS00
RMRQ
4D1 RMRQ/
2D1 8MHZ
PROGRAMMED
U69
74S374
U71
74S288
NC
+5V
21220
U68
IN
50
100
150
250
MC3480
U67
MC/
T1
T2 RAS1/
T3 RAS2/
T4 CAS/
T5 R/W
AB
A12 CE/
R/W
RFG
RFRQ/
RAS1/ 7C1
RAS2/ 7B1
CAS/
WRT/ 7C1
U25 7404
74S373
U56
7C5 MRDBUS
MRD7
MRD6
MRD5
MRD4
MRD3
MRD2
MRD1
MRD0
IOBUS 2A4
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
74LS393
U35
QA
QB
QC
QD
CL
R23, 330K +5V
.1UF C5
MACK/ 4B5
U58 74LS123
2B
2A CLR
U36 74S00
4D1 RMBNK
3D1 LMWTC/
2C1 L/ABUS LA6
LAD
LA5
LAC
LAB
LA4
LAA
LA3
LA9
LA2
LA8
LA1
LA7
LA6
74S153
U41
2C1 LAE
LAF
MA7
MA6
U49
MA5
MA4
U66
MA3
MA2
U57
MA1
MA0
MABUS 7B5,7A5
REFEN
ROWEN
+5V
R17 6.8K
U70 556
R16 3.3K
C2 1000PF SM
R27 22K
R14 2.2K
R24 15K
Q2 2N3906
C6 22PF
2A2 M1/
MONOLITHIC SYSTEMS CORP.
SIZE B
CODE IDENT NO 51513
DWG NO 305-0271-000
REV B
SCALE —
SHEET 6
MSF - 1008A-1

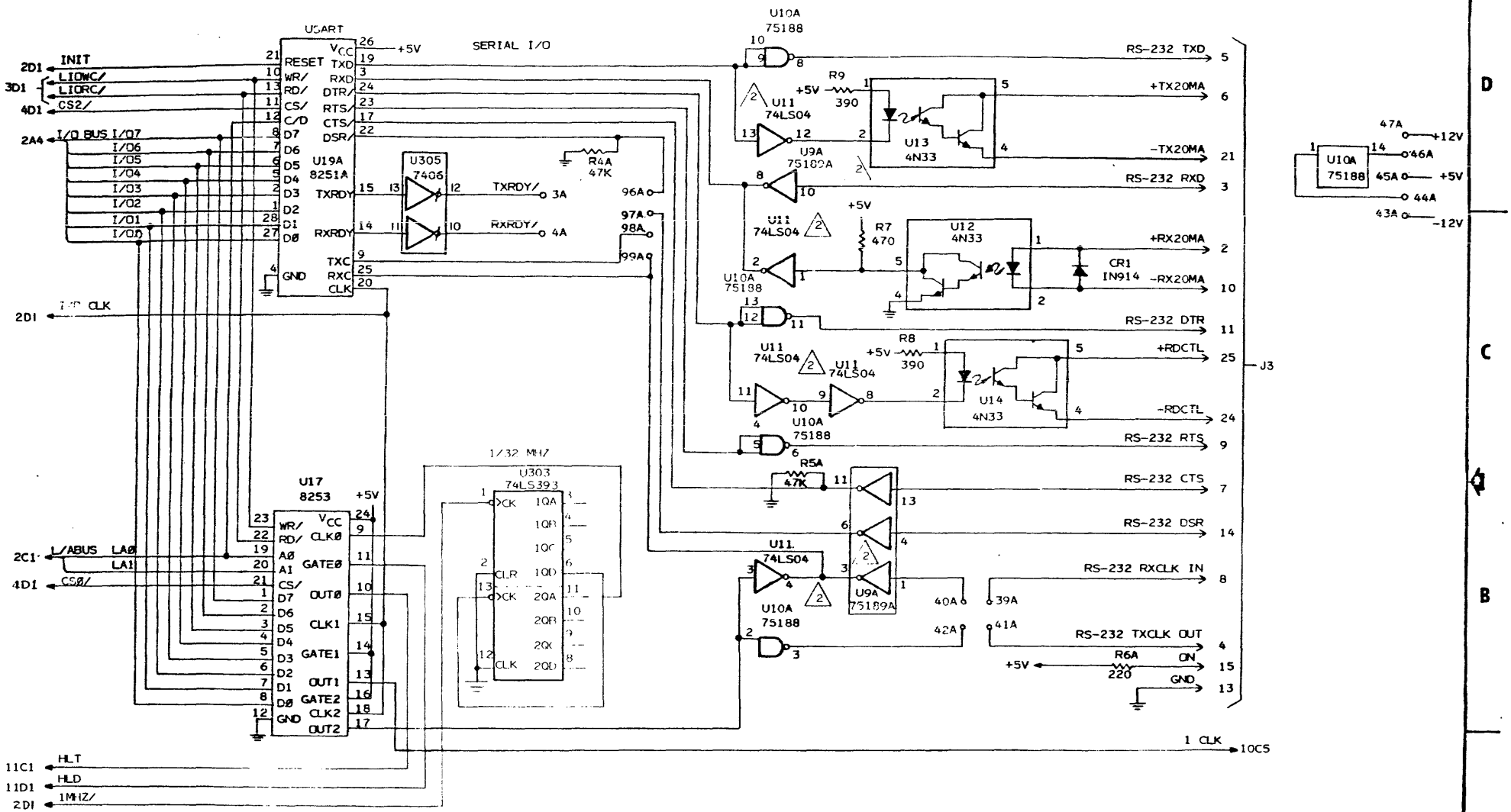


MONOLITHIC SYSTEMS CORP.			
SIZE	CODE IDENT NO	DWG NO	REV
B	51513	305-0271-000	B
SCALE —		SHEET 7	

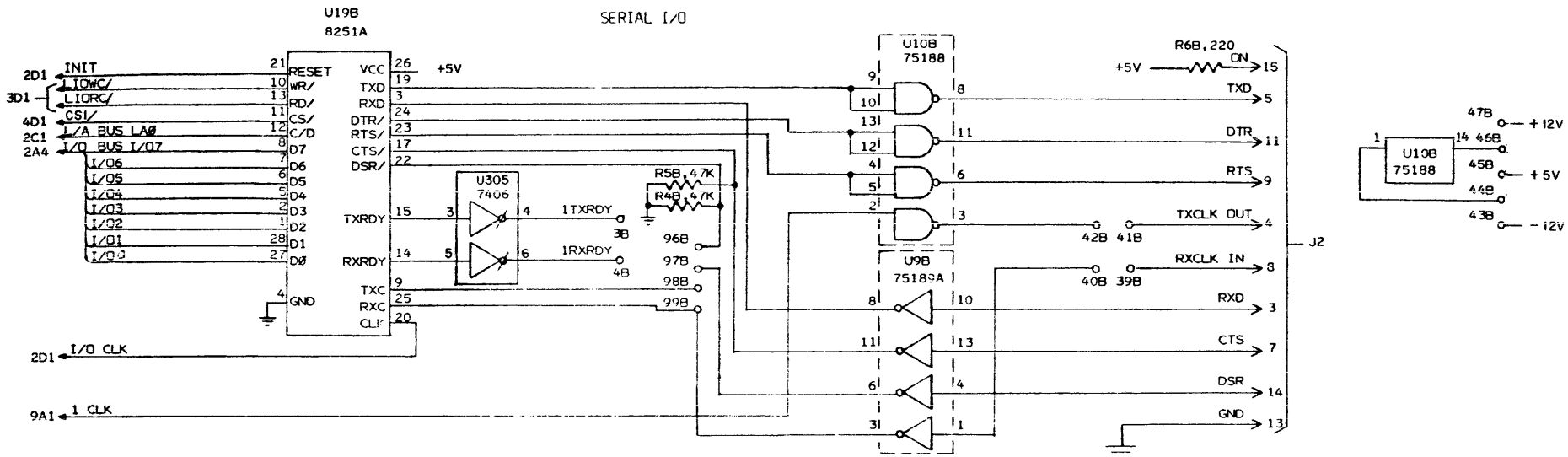


FILTER CAPACITOR DISTRIBUTION

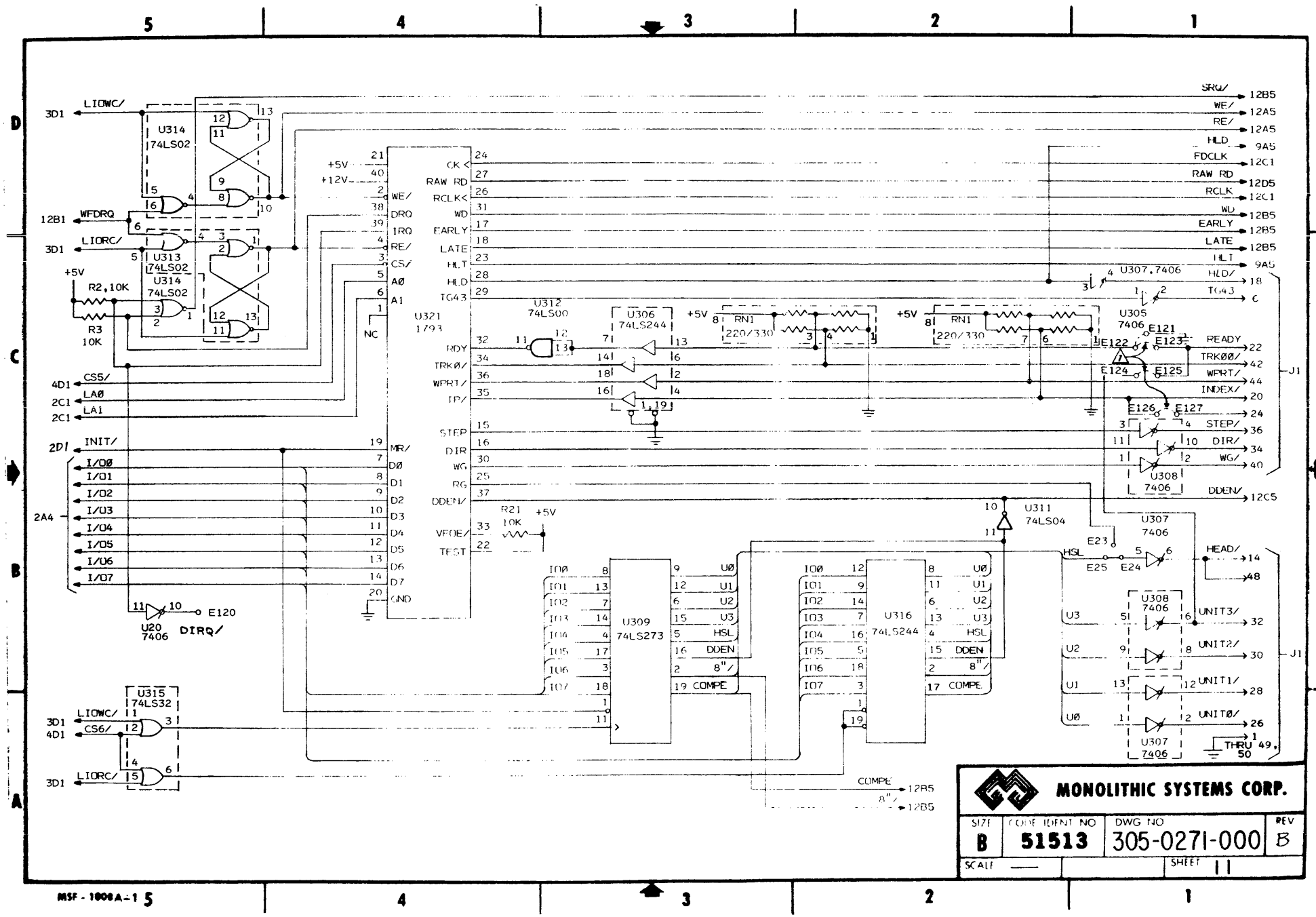
MONOLITHIC SYSTEMS CORP.			
SIZE B	CODE IDENT NO 51513	DWG NO 305-0271-000	REV B
SCALE —		SHEET 8	

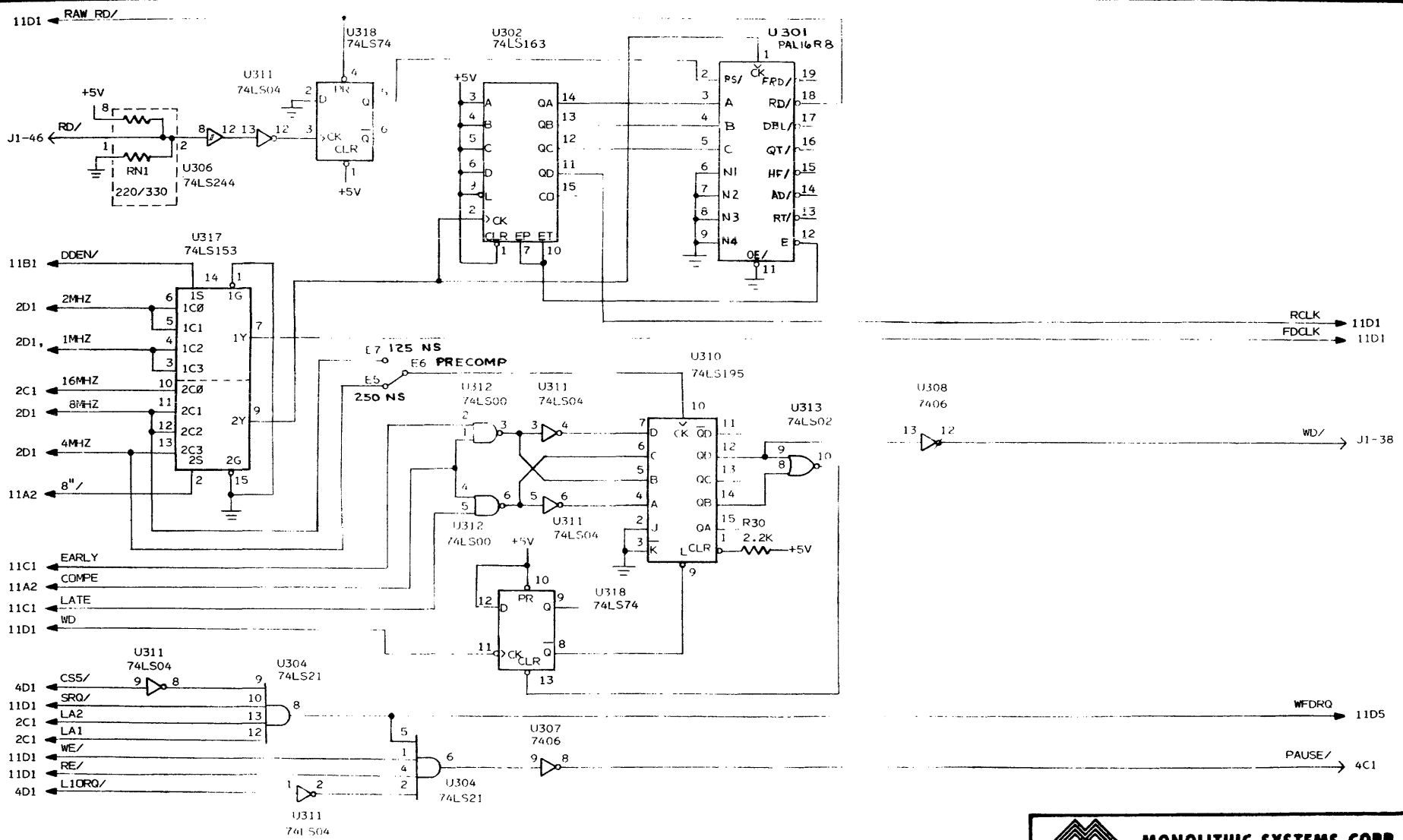


MONOLITHIC SYSTEMS CORP.			
SIZE B	CODE IDENT NO 51513	DWG NO 305-0271-000	REV B
SCALE —		SHEET 9	



MONOLITHIC SYSTEMS CORP.			
SIZE B	CODE IDENT NO 51513	DWG NO 305-0271-000	REV B
SCALE —		SHEET 10	


MONOLITHIC SYSTEMS CORP.
SIZE B
CODE IDENT NO 51513
DWG NO 305-0271-000
REV B
SCALE —
SHEET 11
MSF - 1000A-1
U314 74LS02
U313 74LS02
U315 74LS32
U321 1793
U312 74LS00
U306 74LS244
U316 74LS244
U309 74LS273
U311 74LS04
U307 7406
U308 7406
U20 7406
R2,10K
R3 10K
R21 10K
RN1 220/330
LIOWC/
LIORC/
WFDRQ
CS6/
CS5/
LA0
LA1
INIT/
DIRQ/
+5V
+12V
GND
CK
RAW RD
RCLK
WD
EARLY
LATE
HLT
HLD
TG43
WE/
DRQ
IRQ
RE/
CS/
A0
A1
NC
RDY
TRK0/
WPRT/
IP/
STEP
DIR
WG
RG
DDEN/
VFOE/
TEST
MR/
D0
D1
D2
D3
D4
D5
D6
D7
I/O0
I/O1
I/O2
I/O3
I/O4
I/O5
I/O6
I/O7
U0
U1
U2
U3
HSL
DDEN
8"/
COMPE
SRQ/
WE/
RE/
HLD
FDCLK
RAW RD
RCLK
WD
EARLY
LATE
HLT
HLD/
TG43
READY
TRK0/
WPRT/
INDEX/
STEP/
DIR/
WG/
DDEN/
HEAD/
UNIT3/
UNIT2/
UNIT1/
UNIT0/
THRU 49, 50
J1
E120
E121
E122
E123
E124
E125
E126
E127
E23
E24
E25



MONOLITHIC SYSTEMS CORP.

SIZE	DOCUMENT NO	DWG NO	REV
B	51513	305-0271-000	B
SCALE —		SHEET 12	

DEVICE TYPE NO.	REFERENCE DESIG.	NOTE	POWER & GROUND							REMARKS/SPARES	LINE NO.
			GND	+5V	-12	+12	-5				
I.C. 74S00	U36		7	14							1
I.C. 74LS00	U29,312		7	14						U312 - 1 SPARE	2
I.C. 74LS02	U313,314		7	14							3
I.C. 7404	U25		7	14							4
I.C. 74LS04	U51,311		7	14							5
I.C. 7406	U20,305,307,308		7	14						U307 - 1 SPARE U20 - 4 SPARES U305 - 1 SPARE	6
I.C. 74LS10	U37		7	14							7
I.C. 74S11	U42		7	14							8
I.C. 74LS21	U304		7	14							9
I.C. 74LS32	U43,315		7	14						U315, 2 SPARES	10
I.C. 74LS74	U30,318		7	14							11
I.C. 74S112	U50		8	16							12
I.C. 74LS123	U58		8	16							13
I.C. 74LS138	U21,27		8	16							14
I.C. 74S153	U41,49,52,57,66		8	16							15
I.C. 74LS153	U317										
I.C. 74LS163	U24,31,302		8	16							16
I.C. 74LS175	U22		8	16							17
I.C. 74LS195	U310		8	16							18
I.C. 74S244 OR 74LS244	U47,306,316		10	20						U306, 3 SPARES	19
I.C. 74LS273	U309		10	20							20
I.C. 74S288	U38		8	16						PROGRAMMED PROM 323-0029-001	21
I.C. 74S288	U69		8	16						PROGRAMMED PROM 323-0033-001	22
I.C. 74288	U26		8	16						PROGRAMMED PROM 323-0005-002/XXX	23
I.C. 74S373	U46,55,56		10	20							24
I.C. 74S374	U39,40,71		10	20							25
I.C. 74LS393	U35,303		7	14							26
I.C. 74S471	U32		10	20						PROGRAMMED PROM 323-0028-001	27
I.C. 74S471	U33		10	20						PROGRAMMED PROM 323-0028-004	28
I.C. 74S471	U45		10	20						PROGRAMMED PROM 323-0032-002	29
I.C. 75188	U10A,B		7	(SEE SHTS. 9 & 10)							30

DEVICE TYPE NO.	REFERENCE DESIG.	NOTE	POWER & GROUND							REMARKS/SPARES	LINE NO.
			GND	+5V	-12	+12	-5				
I.C. 75189A	U9A,B		7	14							31
I.C. 75452	U59		4	8							32
I.C. 75453	U60,61		4	8							33
I.C. K1115A	U23		7	14						16 MHZ CRYSTAL	34
I.C. 1793-02	U321		20	21		40					35
I.C. 21220	U68		8	16						DELAY LINE (5 TAPS)	36
I.C. MC3480	U67		12	24							37
I.C. 556	U70		7	14							38
I.C. Z80A	U34		29	11						CENTRAL PROCESSING UNIT	39
I.C. 8214	U62		12	24							40
I.C. 8251A	U19A,B		4	26							41
I.C. 8253	U17		12	24							42
I.C. 8303B/8287	U63,64,65		10	20							43
I.C. 8304B/8286	U48		10	20							44
I.C. 8833	U44		8	16							45
I.C. PAL16R8	U301		10	20							46
I.C. 4N33	U12,13,14										47
SPARE	U319,320										48
OPTIONS											
I.C. MK4116-3	U201-216		16	(SEE SHT. 7)						MEMORY ELEMENT 16K X 1 DYN. RAM	49
I.C. MB8164										MEMORY ELEMENT 64K X 1 DYN. RAM	50
I.C. 74S471	U54		10	20						PROGRAMMED PROM 323-0049-XXX	51
I.C. 74S472										PROGRAMMED PROM 323-0051-XXX	52
I.C. 82S136	U53		9	18						PROGRAMMED PROM 323-0050-XXX	53
I.C. AM9511	U15		1	2		16				ARITHMETIC PROCESSING UNIT	54
I.C. AM9512										ARITHMETIC PROCESSING UNIT	55
I.C. 74LS04	U11		7	14						1 SPARE	56
PALLET ASSY	U28	14 PIN									

MONOLITHIC SYSTEMS CORP.

SIZE	CODE IDENT NO	DWG NO	REV
B	51513	305-0271-000	B

SCALE —— SHEET 13