MODULE REPAIR GUIDE NO. 2

EDITED BY CUSTOMER ENGINEERING DIVISION

7180/7279 REPAIR

1. INTRODUCTION

This MRG contains technical information concerning the microprocessor for the PCS-II and 2210 minifloppy disk drives. It includes descriptions of the basic components of the microprocessor, an explanation of the instruction set (software), a theory of operation (hardware), schematic drawings and a troubleshooting section.

TABLE OF CONTENTS

Printed in U.S.A.

WANG LABORATORIES, INC.
ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

MODULE REPAIR GUIDE NO. 2A

EDITED BY CUSTOMER ENGINEERING DIVISION

March 10, 1978

CORRECTIONS AND UPDATES TO MODULE REPAIR GUIDE NO. 2

1. GENERAL

Module Repair Guide No. 2 contained a lower revision microprogram than the program currently being used. This addendum contains a listing of the updated program along with a few additional pages from MRG No. 2 that have had some major typographical corrections made. The following pages, including the listing of the microprogram, should be inserted into MRG No. 2 and the old pages removed.

2. 7180/7279 MICROPROCESSOR

2.1 SIMPLIFIED THEORY OF OPERATION

Being similar to most general purpose microprocessors, a typical Wang Labs' disk microprocessor is comprised of the following elements (Refer to Figure 1):

A Read-Only-Memory (ROM); used to control all disk microprocessor operations.

A Random-Access-Memory (RAM); normally used as a transitional working register.

An Arithmethic/Logic Unit (ALU)

Two general purpose registers: The A register, and the K register.

Two Status registers; ST_0 and ST_1, actually control indicators which sense and set various disk and disk microprocessor conditions.

The 7180 board contains the entire microprocessor logic for the PCS-II disk drives. Figure 1 shows the simplified block diagram for the 7180 board.

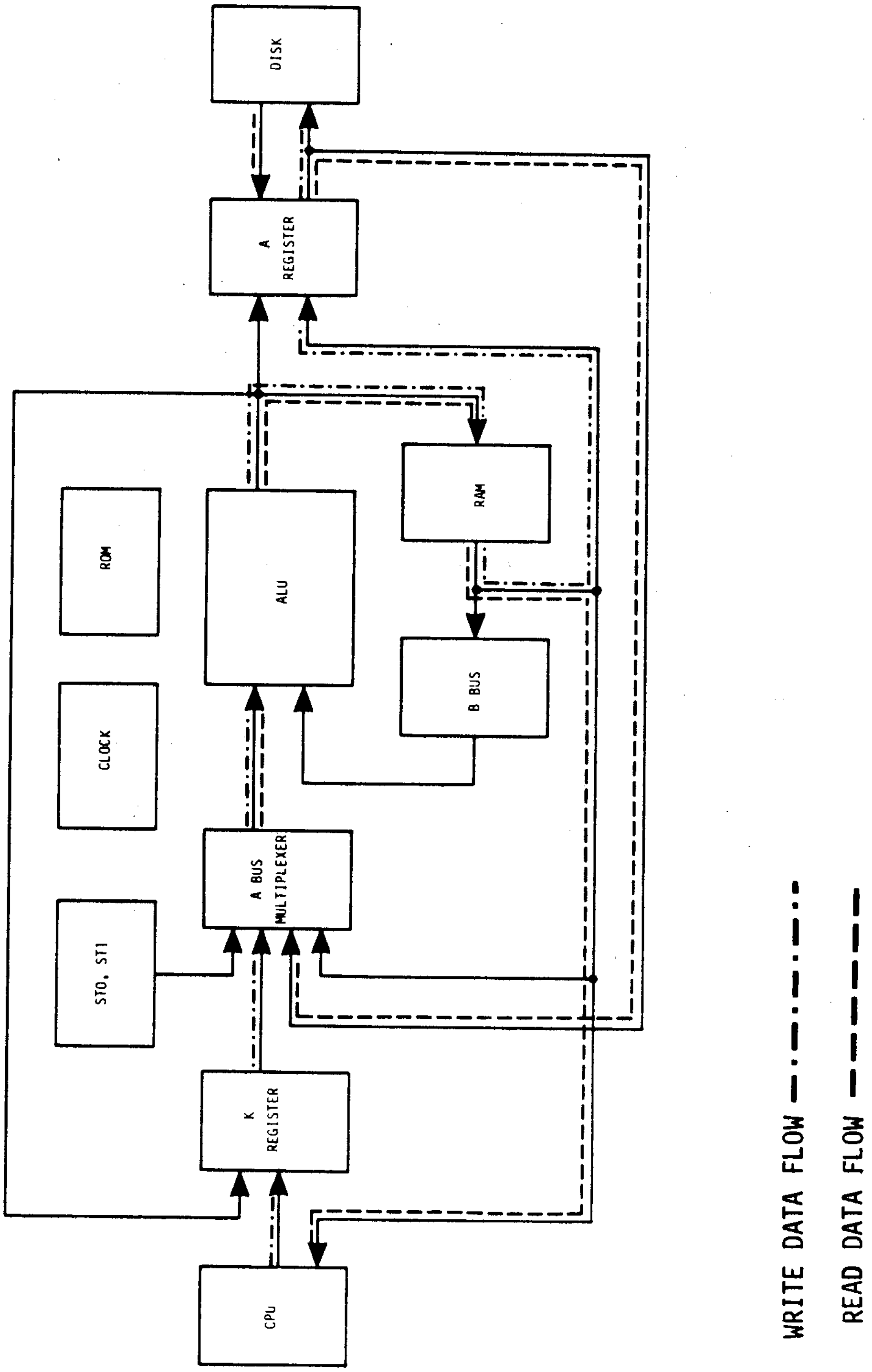


FIGURE 1 SIMPLIFIED BLOCK DIAGRAM

Read/Write Data Flow

When data is written on the disk (refer to Figure 1), data is strobed into the K register from the CPU and clocked into the A bus multiplexer. The output of the multiplexer is applied to the ALU and sent to the RAM. The RAM outputs the data to the A register which sends the data to be written to the disk.

When data is read from the disk, it is applied to the A register, the bus multiplexer and on to the ALU. The ALU applies the data to the RAM and finally to the CPU.

The extra lines shown in Figure 1 are not used during a read/write sequence but are used for data manipulation and housekeeping functions before, during and after the read/write sequence.

2.1.1 READ ONLY MEMORY

The ROM is the heart of the microprocessor and contains the microprogram for the microprocessor. The 7180 utilizes two INTEL 2708 Programmable Read Only Memory (PROM) Integrated Circuits. Each PROM contains a 1024 x 8 bit matrix.

Since a ROM Instruction requires 16 bits, both PROMs are simultaneously selected to provide a 16 bit output. With this configuration, the total ROM capacity is 1024 bytes or steps. See Figure 2.

The steps in the ROM are expressed in hexadecimal notation. The steps are 0000_{16} to $03FF_{16}$ (0-1023). The ROM is addressed by an instruction counter (IC) which normally incremeмts the ROM one step after an instruction has been decoded and performed. Ten bits, ICq-o are applied to the ROM from the IC, the address. The IC can cause the ROM to branch from the increment operation to any address by a branch instruction.

The 16 bit ROM output, RI_{15} - RI_0, is latched into D-latches and becomes bits R_{15} - R_0.

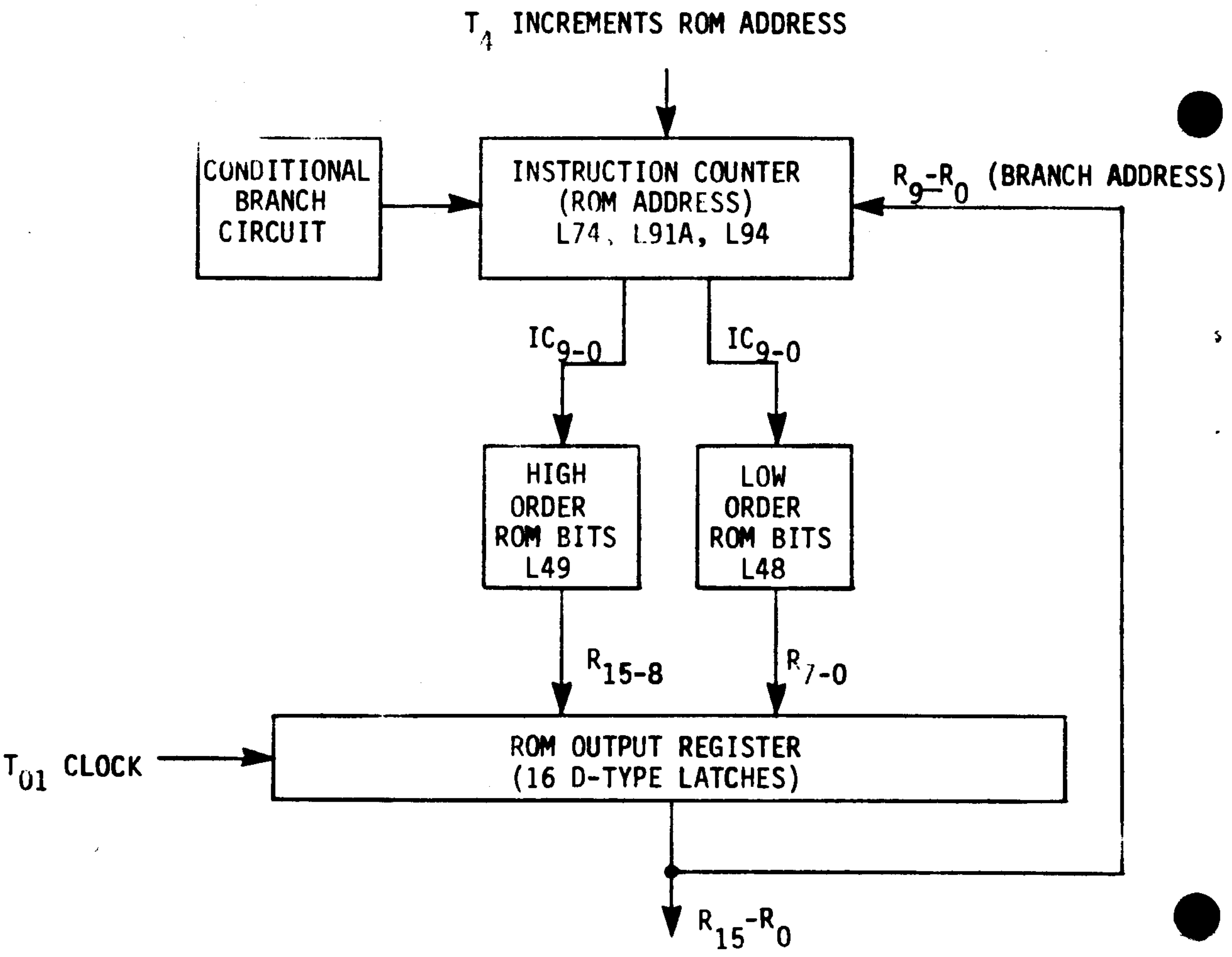

T4 INCREMENTS ROM ADDRESS
CONDITIONAL BRANCH CIRCUIT
INSTRUCTION COUNTER (ROM ADDRESS) L74, L91A, L94
R9-R0 (BRANCH ADDRESS)
IC9-0
IC9-0
HIGH ORDER ROM BITS L49
LOW ORDER ROM BITS L48
R15-8
R7-0
T01 CLOCK
ROM OUTPUT REGISTER (16 D-TYPE LATCHES)
R15-R0

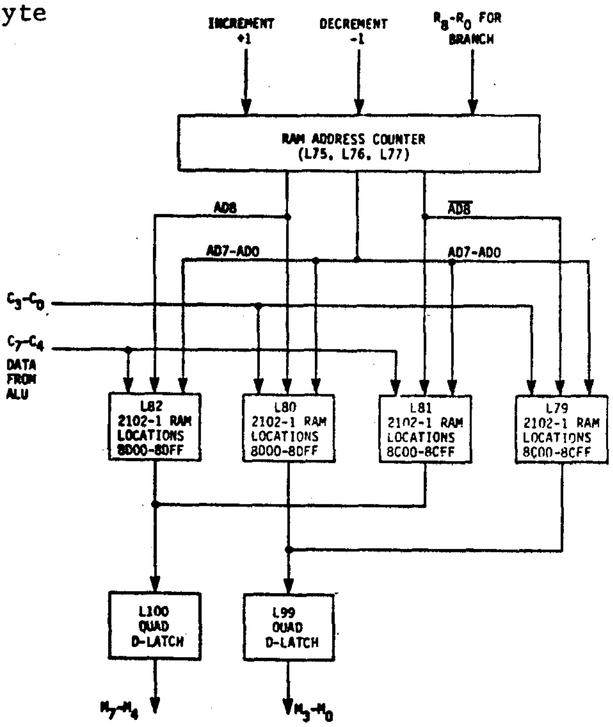
FIGURE 2 READ ONLY MEMORY

2.1.2 RANDOM ACCESS MEMORY

The RAM consists of four 2101-1 Integrated Circuits with a capacity of 256 x 4 bits resulting in a total storage capacity of 512 bytes. See Figure 3. The RAM address counter addresses the RAM with AD_8 - AD_0 bits and can increment, decrement or preset the RAM to any address. Information can only be loaded into the RAM from the ALU, C_7 - C_0, but the output can be transferred to the A register, ALU or the CPU. The RAM is divided into two sections: Locations $8C00_{16}$ - $8CFF_{16}$ (these are locations 0-255 - the reason for this notation will be explained later) are used for the read/write buffer and various locations between $8D00_{16}$ - $8DFF_{16}$ (256-511) are used as a work buffer. Refer to the RAM allocation chart below.

RAM ALLOCATION

LOCATION	DESCRIPTION
*C00 - 8CFF	Read/write buffer
8DOF	Zero sent to the 2200
8D10	2200 Address byte #1
8D11	2200 Address byte #2
8D12	2200 Address byte #3
8D20	Header byte #1 (track from disk)
8D21	Header byte #2 (sector from disk)
8D25	Disk track #1 (track currently under head)
8D26	Disk track #2 (track currently under head)
8D30	Error count
8D31	Internal status
8D32	Address sent to 2200
8D33	Format retries
8D40	Disk #2 operation count
8D41	Disk #1 operation count
8DD4 - 8DE7	20 Bytes of zeroes
8DE8	03 Byte
8DE9	Header byte #1 (track desired)
8DEA	Header byte #2 (sector desired)
8DEB - 8DFE	20 Bytes of zeroes
8DFF	03 Byte


INCREMENT +1
DECREMENT -1
R_8-R_0 FOR BRANCH
RAM ADDRESS COUNTER (L75, L76, L77)
AD8
$\overline{\text{AD8}}$
AD7-AD0
AD7-AD0
C_3-C_0
C_7-C_4 DATA FROM ALU
L82 2102-1 RAM LOCATIONS 8D00-8DFF
L80 2102-1 RAM LOCATIONS 8D00-8DFF
L81 2102-1 RAM LOCATIONS 8C00-8CFF
L79 2102-1 RAM LOCATIONS 8C00-8CFF
L100 QUAD D-LATCH
L99 QUAD D-LATCH
M_7-M_4
M_3-M_0

2.1.3 ARITHMETIC/LOGIC UNIT

Two 74181 integrated circuits, designed to perform specific arithmetic or logical operations, as directed by the ROM Microinstruction sequence, comprise the ALU.

This ALU responds to sixteen instructions. Basic ALU inputs consist of the A bus, the B bus, a Carry-In bit, and a function select code decoded from the ROM.

The A bus is the output of a multiplexer, incorporating the A register, the K register, and Status registers St_0 or ST_1 as selectable inputs.

The B bus is the output of a two-part register, incorporating the eight low order ROM bits (R_7 - R_0), or the eight bit RAM output (M_7 - M_0).

The ALU output is the C bus (C_7 - C_0); data on this bus can be stored in the A register, K register, RAM, or can be transferred to the status registers ST_0 or ST_1. Again, ALU manipulations are directed by the ROM instruction set.

2.1.4 REGISTER STRUCTURE

1) K Register:

The general purpose K register stores data from either the controlling CPU (KS_7 - KS_0), or from the disk microprocessor ALU (C_7 - C_0); i.e., whichever source is selected by the ROM for input to the K register.

When commanded by the ROM, the contents of the K register is transferred to the AUL.

2) A Register:

The general purpose A register stores data from (1) the disk microprocessor RAM (M_7 - M_0), which is data to be written on the disk, (2) the disk data (a READ), (3) the cyclic redundancy check (CRC) circuitry (verifies disk data accuracy), and (4) the ALU (C_7 - C_0). The contents of the A register may be used by the ALU, or may be used as a transitional register for each data byte written on the disk.

3) Status Registers (ST_0, ST_1):

Status register ST_0 reports eight disk/CPU conditions to the disk microprocessor, as follows:

Word Ready STO-1 (Bit 1) - The 4th bit of a four-bit binary counter, used to indicate when each byte is ready to be transferred from disk to RAM, via the A register and ALU (WRITE). Word Ready is also used to indicate when each byte is ready to be transferred from RAM to disk via the A register.

Address Bit 8 STO-2 (Bit 2) - The 9th (highest order) RAM address bit; (2^9 = 512 byte addresses).

40 ms Delay STO-3 (Bit 3) - The output of an integrated circuit (active for 40 ms). This delay is used to allow time for proper disk access.

HM STO-4 (Bit 4) - A signal indicating to the microprocessor that the motor of Disk Drive #1 has been turned on and that the R/W head has been loaded.

CAX STO-5 (Bit 5) - Made up of the 2200 Addresses Bus terms AB_6 and AB_8. CAX distinguishes a select address from a data transfer operation.

Calculator Input Strobe STO-6 (Bit 6) - A strobe from 2200 to disk microprocessor which must accompany each address or data byte from the 2200 to the K register. This status bit indicates that one byte is ready to be transferred from the K register to RAM via the ALU.

HM_2 STO-7 (Bit 7) - A signal indicating to the microprocessor that the motor of Disk Drive #2 has been turned on and that the R/Q head has been loaded.

Calculator Busy STO-8 (Bit 8) - The Ready/Busy indicator from the 2200 CPU.

These bits are sampled by the ALU via the A Bus inputs when ROM bits R_9 and R_8 = 10_B (R_9 = 1; R_8 = 0).

Status register ST_1 reports four disk/disk microprocessor conditions back to the microprocessor as follows:

SMO ST1-1 (Bit 1) - This bit is the index pulse from the disk. (There is one pulse for every 10 sectors on the disk). It is used to determine sector 0 during a format operation.

ST1-2 through 4 (Bits 2, 3, 4) - These bits are not used in a PCS-II disk microprocessor.

SECTOR MARK PULSE ST1-5 (Bit 5) - Ultimately the sector mark pulse (SM) from the disk. (There is one pulse for each sector on the disk). It is used to denote the beginning of each sector.

Tract 00 ST1-6 (Bit 6) - A line from the Disk Drive which indicates when the R/W head is positioned at track zero (the outer most track).

Carry ST1-7 (Bit 7) - This is the carry bit for arithmetic operations. It is gated to the ALU for ROM microinstructions specifying carry, and receives that resultant carry.

ST1-8 (Bit 8) - This bit is not used in a PCS-II disk microprocessor.

These bits are sampled by the ALU via the B Bus inputs when ROM bits R_9 and $R_8 = 11_2$.

2.2 THE INSTRUCTION SET - HARDWARE CONTROL VIA SOFTWARE

2.2.1 GENERAL

Control is implemented via 16-bit ROM instructions, as explained in Section 2.1.1. This 16-bit output (bits $R_{15} - R_0$) is distributed throughout the disk microprocessor, and is subsequently decoded as instructions to perform specific logic operations and data manipulations.

ROM output is formatted such that type of operation, registers involved, destination of resultant data, information source(s), and other control factors are defined by a single 16-bit instruction. The assembled sequence of "microinstructions" is referred to as the microprogram for the peripheral's microprocessor.

There are 26 basic instructions in the 7180 instruction set with each instruction having variables resulting in literally thousands of unique instructions available for data manipulation. The microprocessor hardware is designed so that it can execute every instruction with the end result predictable in every case.

Each of the instructions are comprised of two parts: the operation code and the operand. The operation code is defined as a portion of a computer instruction that indicates which action is to be performed by the computer. ROM bits R_{15} - R_8 are used as the operation code. Operand is defined as any one of the quantities entering into or arising from an operation. ROM bits R_7 - R_0 comprise the operand.

Operation code bits indicate the following, depending on which instruction category (to be explained later) is applicable:

1) Function to be performed.

2) Register (A, K, ST_0, ST_1) used in performing this function.

3) Whether to increment/decrement RAM address, or allow current RAM address to remain unchanged.

4) Selection of destination for resultant information (function performed). That is, store these results either in a register (A, K, ST_0, ST_1) or back into the current RAM location.

5) Whether high order bits (7-4) or low order bits (3-0) of the register designated (A, K, ST_0, ST_1) will be used for comparisons involved in conditional branch instructions.

Operand code bits indicate the following (again depending on which instruction category is applicable):

1) Use operand (via microprocessor ALU B Bus) as mask bits, along with bits (via microprocessor ALU A Bus) from a register f(A, K, ST_0, ST_1) designated by ROM bits R_8 and R_9.

2) Use operand bits $R_7 - R_4$ as a mask for 4-bit Conditional Branch instructions.

3) Use operand bits $R_3 - R_0$ or $R_7 - R_0$ as a destination branch address for Conditional Branch instructions.

4) Use operand bits $R_7 - R_0$ plus operation code bits R_8 and R_9 as a destination address for Unconditional Branch instructions.

5) Use operand bits $R_7 - R_0$ plus operation code bit R_8 as a new RAM address to be preset via Load Auxiliary instruction in the microprogram.

2.2.2 INSTRUCTION CATEGORIES

The 7180/7279 disk microprocessor instruction set can be arranged into five major categories.

1) Register Instructions:

An operation using 8 bits of RAM output ($M_7 - M_0$) contained at the current RAM address, and 8 bits contained in a register (A, K, ST_0, ST_1) designated by ROM bits R_9 and R_8. The results of such instructions are either stored into the current RAM location or stored back into the register designated by ROM bits R_9 and R_8. The RAM address can be incremented, decremented or remain unchanged by bits R_{11} and R_{10}.

There are eight register instructions as explained below:

INSTRUCTION	EXPLANATION
NOOP	No operation. Used as a filler command and has no effect but to cause the ROM address to increment.
B to M	B to memory. Transfers contents of B (buffer register A,1 K, STO or ST1) to memory and causes RAM address to increment, decrement or remain unchanged.
M to B	Memory to B. Transfers contents of memory to B (buffer register A, K, STO or ST1) to memory and causes RAM address to increment, decrement or remain unchanged.
Add w/o carry	Binary add without carry bit. A binary add of the contents of B (buffer register A, K, STO or ST1) and RAM with result to RAM or B and causes RAM address to increment, decrement or remain unchanged.
OR	Logical OR function. ORs the contents of B (buffer register A, K, STO or ST1) with RAM with result to RAM or B and causes RAM address to increment, decrement or remain unchanged.
XOR	Exclusive OR function. Same as OR function except contents of B and RAM are exclusive ORed.

INSTRUCTION	EXPLANATION
Add with carry	Binary add with carry bit. Same as add without carry instruction.
AND	Logical AND function. ANDs the contents of B (buffer register A, K, STO or ST1) with contents of RAM with result to RAM or B and causes RAM address to increment, decrement or remain unchanged.

2) Immediate Instructions:

An operation using 8 bits of ROM output (R_7 - R_0) contained at the current ROM address, and 8 bits contained in a register (A, K, ST_0, ST_1) designated by ROM bits R_8 and R_9. The results of such instructions are stored back into the designated register (R_8, R_9).

There are four immediate instructions as explained below:

INSTRUCTION	EXPLANATION
OR Immediate	Logical OR function. The contents of a register (A, K, STO or ST1) are ORed with the eight least significant bits of ROM with the result stored back into the register.
XOR Immediate	Exclusive OR function. Same as OR Immediate except contents are exclusive ORed.

INSTRUCTION — EXPLANATION

Add w/carry Imm. — Binary add with carry bit. The binary add w/carry bit of a register (A, K, STO or ST1) with the eight least significant bits of ROM with the result stored back into the register.

AND Immediate — Logical AND function. Same as OR Immediate except contents are ANDed.

3) Branch Instructions:

These instructions are divided into two categories: conditional branch and unconditional branch. The conditional branch instructions allow from 0 to ± 15 microprogram step jumps; or from 0 to ± 255 microprogram step jumps (depending on which conditional branch instruction is performed). The unconditional branch instructions causes a jump to any step within the microprocessor's microprogram.

There are ten branch instructions as explained below:

INSTRUCTION — EXPLANATION

Br if Reg. = 0 — Branch if register = 0. Conditional branch to step indicated by the eight least significant bits of ROM if register (A, K, STO or ST1) equals zero. Maximum number of steps is + 255.

Br. if Reg. ≠ 0 — Branch if register ≠ 0. Same as above if register does not equal zero.

Br. if True L,H

Branch if True Low or Branch if True High. Conditional branch to step indicated by ROM bits R_3 - R_0 if the 4 least significant bits (Low) or the 4 most significant bits (High) of register (A, K, STO or ST1) equal any corresponding true ROM bits R_7 - R_4. Maximum number of branch steps is +15.

Br. if False L,H

Branch if False Low or Branch if False High. Conditional branch to step indicated by ROM bits R_3 - R_0 if the 4 LSB (Low) or the 4 MSB (High) of register f(A, K, STO or ST1) equal any corresponding false ROM bits R_7 - R_4. Maximum number of branch steps is +15.

Br. if = Mask

Branch if equal to Mask. Conditional branch to step indicated by ROM bits R_3 - R_0 if either the 4 LSB or 4 MSB (selected by a bit of the instruction code) of a register (A, K, STO or ST1) equal the mask of ROM bits R_7 - R_4. Maximum # of branch steps is +15.

Br. if ≠ Mask

Branch if not equal to Mask. Same as above if register does not equal mask.

UB to steps 0-255

Unconditional Branch to steps 0-255. In the form of HEX 88YY, causes the microprogram to branch to the address contained in YY to one of the first 256 steps of the microprogram (steps 0-255).

UB to steps 256-511

Unconditional Branch to steps 256-511. In the form of HEX 89YY, causes the microprogram to branch to the address contained in YY to one of the second 256 steps of the microprogram (steps 256-511).

UB to steps 512-767

Unconditional Branch to steps 512-767. In the form of HEX 8AYY, causes the microprogram to branch to the address contained in YY to one of the third 256 steps of the microprogram (steps 512-767).

UB to steps 768-1023

Unconditional Branch to steps 768-1023. In the form of HEX 88YY, causes the microprogram to branch to the address contained in YY to one of the fourth 256 steps of the microprogram (steps 768-1023).

4) RAM Address Instructions:

These two instructions allow the ROM outputs R_8 - R_0 to preset the RAM address.

INSTRUCTION

EXPLANATION

Load Aux. (0-255)

Load Auxiliary. Enables the ROM bits R_8 - R_0 to preset the RAM to any address between 0-255 (the data buffer). The instruction takes the form of HEX 8CXX where XX is the RAM address.

Load Aux. (256-511)	Load Auxiliary. Enables the ROM bits R_8 - R_0 to preset the RAM to any address between 256 and 511 (the work buffer). The instruction takes the form of 8DXX where XX is the RAM address.

5) Control Instructions:

Control instructions initiate disk functions such as disk head movement, read, write, format, etc. These instructions are decoded directly from ROM outputs to peripheral interfacing hardware in the microprocessor. The ALU is not involved in these instructions.

There are 12 control functions used in the PCS-II microprocessor.

INSTRUCTION	EXPLANATION
Control 1	Takes the form of HEX ECXX where EC01 - Turn on read gate (RDG) EC02 - Turn on write gate (WTG) EC04 - Not used EC08 - Not used EC10 - Head direction Select (HD DIR) EC20 - Preset CRC (PRC) EC40 - Head step (HD ST) EC80 - Clear drive #2 (HM_2) ED00 - Clear drive #1 (HM_1)
Control 2	Takes the form of HEX FCXX where FC01 - Select drive #1 (DK_1) FC02 - Select drive #2 (DK_2) FC04 - Set drive #1 (DK_1) FC08 - Not used FC10 - Strobe to 2200 (IBS)

FC20 - 40 ms delay

FC40 - Not used

FC80 - Not used

2.2.3 INSTRUCTION SET SUMMARY

2.2.3.1 Introduction

Tables 1, 2 and 3 summarize the 26 instructions of the 2210/PCS-II microinstruction set. Table 1 lists all the instructions and is divided into three major columns. The first column labeled "instruction category" contains the instruction names arranged by category as explained in Section 2.2.2. The second major column is labeled "operation code bits" and lists in binary the eight most significant bits of ROM output. This is the first half of any instruction, the operation code. The third major column is labeled "operand code bits" and lists in binary the eight least significant bits of ROM output. This is the second half of any instruction, the operand.

Referring to the second major column, it is noted that ROM bits R_{15} - R_{12} are fixed ones and zeroes for the register instructions but that bits R_{11} - R_8 may vary. The I/D heading under ROM bits R_{11} and R_{10} indicates increment or decrement of RAM address depending upon the status of these two bits.

It is at this point that Table 2 is required; Table 2 is an expansion of all the abbreviations used in Table 1. Referring to Table 2, the I/D bits are expanded into the four possible configurations. For example, if I/D bits R_{11} and R_{10} are a one and zero respectively for a register instruction, the result of the operation is stored in RAM and the RAM address is decremented one location.

TABLE 1

PCS-II/2210 DISK MICROPROCESSOR INSTRUCTION SET

	OPERATION CODE BITS-(R_{15}-R_8)								OPERAND CODE BITS-(R_7-R_0)
INSTRUCTION CATEGORY	15	14	13	12	11	10	9	8	7 6 5 4 3 2 1 0
REGISTER INSTRUCTIONS	INSTRUCTION CODE				I/D		REG.		OPERAND
NO-OP	0	0	0	0	ID	ID	B		Not Used
B to Memory	0	0	0	1	ID	ID	B		Not Used
Memory to B	0	0	1	0	ID	ID	B		Not Used
Binary ADD w/carry	0	0	1	1	ID	ID	B		Not Used
OR	0	1	0	0	ID	ID	B		Not Used
XOR - Exclusive OR	0	1	0	1	ID	ID	B		Not Used
Binary ADD wo/carry	0	1	1	0	ID	ID	B		Not Used
AND	0	1	1	1	ID	ID	B		Not Used
IMMEDIATE INSTRUCTIONS	INSTRUCTION CODE						REG.		IMMEDIATE OPERAND (B BUS)
OR Immediate	1	1	0	0	1	0	B		I I I I I I I I
XOR Immediate	1	1	0	1	1	0	B		I I I I I I I I
Binary ADD wo/carry Immed	1	1	1	0	1	0	B		I I I I I I I I
AND Immediate	1	1	1	1	1	0	B		I I I I I I I I
BRANCH INSTRUCTIONS (Conditional; 8 bit)	INSTRUCTION CODE						REG.		BRANCH ADDRESS (B BUS)
Branch if Register = 0	1	1	0	0	1	1	B		Y Y Y Y Y Y Y Y
Branch if Register ≠ 0	1	1	0	1	1	1	B		Y Y Y Y Y Y Y Y
MASK BRANCH INSTRUCTIONS (Conditional; 4 bit)	INSTRUCTION CODE					H/L	REG.		MASK BRANCH ADDRESS
Branch IF True	1	0	1	0	0	S	B		M M M M Y Y Y Y
Branch IF False	1	0	1	1	0	S	B		M M M M Y Y Y Y
Branch IF = Mask	1	0	0	0	0	S	B		M M M M Y Y Y Y
Branch IF ≠ Mask	1	0	0	1	0	S	B		M M M M Y Y Y Y
UNCONDITIONAL BRANCH	INSTRUCTION CODE								BRANCH ADDRESS
TO STEPS 0-255 (0000-00FF)	1	0	0	0	1	0	0	0	Y Y Y Y Y Y Y Y
TO STEPS 256-511 (0100-01FF)	1	0	0	0	1	0	0	1	Y Y Y Y Y Y Y Y
TO STEPS 512-767 (0200-02FF)	1	0	0	0	1	0	1	0	Y Y Y Y Y Y Y Y
TO STEPS 768-1023 (0300-03FF)	1	0	0	0	1	0	1	1	Y Y Y Y Y Y Y Y
RAM ADDRESS INSTRUCTIONS	INSTRUCTION CODE								RAM ADDRESS LOADED
LOAD AUX (DATA BUFFER)	1	0	0	0	1	1	0	0	X X X X X X X X
LOAD AUX (WORK BUFFER)	1	0	0	0	1	1	0	1	X X X X X X X X
CONTROL INSTRUCTIONS	INSTRUCTION CODE								CONTROL OPERAND
Control 1	1	1	1	0	1	1	Z	Z	Z Z Z Z Z Z Z Z
Control 2	1	1	1	1	1	1	Z	Z	Z Z Z Z Z Z Z Z

Again referring back to Table 1, the last two bits of the operation code (bits R_9 and R_8) determine what register is to be used in the operation. This column is headed by "REG." which is an abbreviation for register. Observing Table 2, the two bits decode one of the four registers. For example, if ROM bits R_9 and R_8 are both low during a register instruction, the A register is the selected register.

Table 3 allows the reader to disregard Tables 1 and 2 since Table 3 is a breakdown of all the instructions in hexadecimal form. For example, if a 1700_{16} code is encountered in the program, the reader would be forced to convert the hexadecimal code to binary (0001011100000000), refer to Table 1 for the type of instruction, and finally to Table 2 to determine the register used and whether or not the RAM address is incremented or decremented. However, by utilizing Table 3, the reader can instantly determine that a 1700_{16} code is a B to M instruction involving register ST1 and incrementing the RAM address.

TABLE 2

EXPLANATION OF LETTER DESIGNATIONS
FOR INSTRUCTION SET BITS

I/D = Increment/decrement of RAM Address

00 = Result to original RAM address; no increment or decrement
01 = Result to original RAM address then increment (+1)
10 = Result to original RAM address then decrement (-1)
11 = Result to selected register (B) and increment RAM address

REG. = Selected register

00 = A register
01 = K register
10 = Status Reg. 0
11 = Status Reg. 1

I = Immediate Operand (R_0 - R_7 Mask)

M = Mask; a unique configuration of binary bits.

Y = Branch Address

S = High order or Low order 4-bits of MASK Branch Instruction.

0 = Low order (Bits 0 - 3)
1 = High order (Bits 4 - 7)

X = New R. A. M. Address for LOAD Aux

Z = Control Operand

Control 1 (ECZZ$_{16}$) X or (EDZZ$_{16}$:

EC01 - Turn on Read Gate; RDG - (CNTRL1 AND R_0)
EC02 - Turn on Write Gate; WTG - (CNTRL1 AND R_1)
EC04 - Not used - (CNTRL1 AND R_2)
EC08 - Not used - (CNTRL1 AND R_3)
EC10 - Head Direction Select; HD DIR - (CNTRL1 AND R_4)
EC20 - PRC - (CNTRL1 AND R_5)
EC40 - Head Step; HD ST - (CNTRL1 AND R_6)
EC80 - Clear drive #2; HM_2 - (CNTRL1 AND R_7)
ED00 - Clear drive #1; HM_1 - (CNTRL1 AND R_8)

Control 2 (FCZZ)

FC01 - Select Drive #1 - (CNTRL2 AND R_0)

FC02 - Select Drive #2 - (CNTRL2 AND R_1)

FC04 - Set Drive #1 - (CNTRL2 AND R_2)

FC08 - Not used

FC10 - Strobe to 2200 - (CNTRL2 AND R_4)

FC20 - 40 ms. Delay - (CNTRL2 AND R_5)

FC40 - Not used - (CNTRL2 AND R_6)

FC80 - Not used - (CNTRL2 AND R_7)

TABLE 3
EXPANDED BREAKDOWN OF MICROPROCESSOR OPERATION CODES

B To M

1.	A	K	St0	Stl	
	1000	1100	1200	1300	No RAM I/D
	1400	1500	1600	1700	AD + 1
	1800	1900	1A00	1B00	AD - 1

M To B

2.	A	K	St0	Stl	
	2000	2100	2200	2300	No RAM I/D
	2400	2500	2600	2700	AD + 1
	2800	2900	2A00	2B00	AD - 1

Add With Carry (RAM)

3.	A	K	St0	Stl	
	3000	3100	3200	3300	No RAM I/D
	3400	3500	3600	3700	AD + 1
	3800	3900	3A00	3B00	AD - 1
	3C00	3D00	3E00	3F00	AD + 1 Result to B

OR (RAM)

4.	A	K	St0	St1	
	4000	4100	4200	4300	No RAM I/D
	4400	4500	4600	4700	AD + 1
	4800	4900	4A00	4B00	AD - 1
	4C00	4D00	4E00	4F00	AD + 1 Result to B

Exclusive OR (RAM)

5.	A	K	St0	St1	
	5000	5100	5200	5300	No RAM I/D
	5400	5500	5600	5700	AD + 1
	5800	5900	5A00	5B00	AD - 1
	5C00	5D00	5E00	5F00	AD + 1 Result to B

Add Without Carry (RAM)

6.	A	K	St0	St1	
	6000	6100	6200	6300	No RAM I/D
	6400	6500	6600	6700	AD + 1
	6800	6900	6A00	6B00	AD - 1
	6C00	6D00	6E00	6F00	AD + 1 Result to B

AND (RAM)

7.	A	K	St0	St1	
	7000	7100	7200	7300	No RAM I/D
	7400	7500	7600	7700	AD + 1
	7800	7900	7A00	7B00	AD - 1
	7C00	7D00	7E00	7F00	AD + 1 Result to B

OR Immediate

8.	C8II	C9II	CAII	CBII

9. D8II D9II DAII DBII

Add Without Carry Immediate

10. E8II E9II EAII EBII

AND Immediate

11. F8II F9II FAII FBII

Branch Commands

12.	A	K	St0	St1		
	80MY	81MY	82MY	83MY	= Mask L	
	84MY	85MY	86MY	87MY	= Mask H	
	90MY	91MY	92MY	93MY	≠ Mask L	
	94MY	95MY	96MY	97MY	≠ Mask H	Can only branch within a ±15 step area.
	A0MY	A1MY	A2MY	A3MY	True L	
	A4MY	A5MY	A6MY	A7MY	True H	
	B0MY	B1MY	B2MY	B3MY	False L	
	B4MY	B5MY	B6MY	B7MY	False H	
	CCYY	CDYY	CEYY	CFYY	= 0	Can branch within a ±255 step area.
	DCYY	DDYY	DEYY	DFYY	≠ 0	

13. Unconditional Branch = 88YY steps 0 - 255 in microprogram
89YY steps 256-511 in microprogram
8AYY steps 512-767 in microprogram
8BYY steps 768-1023 in microprogram

14. Load Auxiliary = 8CXX - RAM 0 - 255 (DATA BUFFER)
8DXX - RAM 256-511 (WORK BUFFER)

15. Control 1 = ECZZ or EDZZ (See Table 2)
Control 2 = FCZZ (See Table 2)

2.2.3.2 Instruction Set Examples

1) Register Instructions

The following two examples are typical of Register Instruction decoding:

EXAMPLE: $10XX_{16}$ (B to M)

	HEX 1				HEX 0				n/a				n/a			
Binary Value	0	0	0	1	0	0	0	0	X	X	X	X	X	X	X	X
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

Where: 0 = low (±0V)
1 = high (+5V)
X = don't care

In the above register instruction, $R_{15} - R_{12}$ identify a Register-to-Memory command (sometimes referred to as B-to-M, in notation form). ROM bits R_{11} and R_{10} indicate that RAM address will not be incremented and that the contents of the A register (designated by ROM bits $R_{9,8} = 00_2$) will be stored at the current RAM location. ROM bits $R_7 - R_0$ are not used in register instructions. The above information can be verified by using Tables 1 and 2. However, by referring to Table 3, the $10XX_{16}$ code is immediately recognized as a B to M with the A register and no RAM increment or decrement.

EXAMPLE: $65XX_{16}$ (Add with Carry)

	HEX 6				HEX 5				n/a				n/a			
Binary Value	0	1	1	0	0	1	0	1	X	X	X	X	X	X	X	X
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

For this instruction, R_{15} - R_{12} designate that a Binary Add with Carry will be performed between the 8 bits at the current RAM location, and the 8-bit contents of the register designated by ROM bits R_9, R_8. In this case, R_9 and R_8 = 01_2, designating the 8-bit register. The Binary Add with Carry is performed, and R_{11} and R_{10} indicate that the result is stored at the current RAM location and then the RAM address is incremented. Referring to Table 3, the 65XX code is decoded as an Add with Carry instruction involving the K register and incrementing the RAM address.

Generally speaking, a Register instruction is an operation performed between an eight bit register (A, K, ST_0, ST_1) designated by R_9, R_8 and the eight bits stored at the current RAM address.

2) Immediate Instructions

The following example is typical of Immediate Instruction decoding:

EXAMPLE: $F8AA_{16}$ (AND Immediate)

	HEX F				HEX 8				HEX A				HEX A			
Binary Value	1	1	1	1	1	0	0	0	1	0	1	0	1	0	1	0
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

All Immediate Instructions are performed using the 8 bits contained in a register (A, K, ST_0, ST_1) designated by R_9 and R_8, and the 8-bit mask presented in R_7 - R_0. The results of Immediate Instructions are transferred back to the designated register (A, K, ST_0, ST_1).

For the example Immediate Instruction F8AA, ROM bits R_{15} - R_{10} designate an AND IMMEDIATE operation. The result of this AND IMMEDIATE is stored back into designated register A (R_9, R_8 = 00_2). This can be verified by using Table 3.

The 8-bit mask present in bits R_7 - R_0 (10101010) are the bits that are ANDed in the ALU with the present contents of RAM.

As an example, if the RAM contains a bit configuration of 11110000 and this AND Immediate is executed, the final result in the A register would be a bit configuration of 10100000 (10101010 ANed with 11110000 = 10100000).

3) Branch Instructions

The various Branch instructions are explained individually:

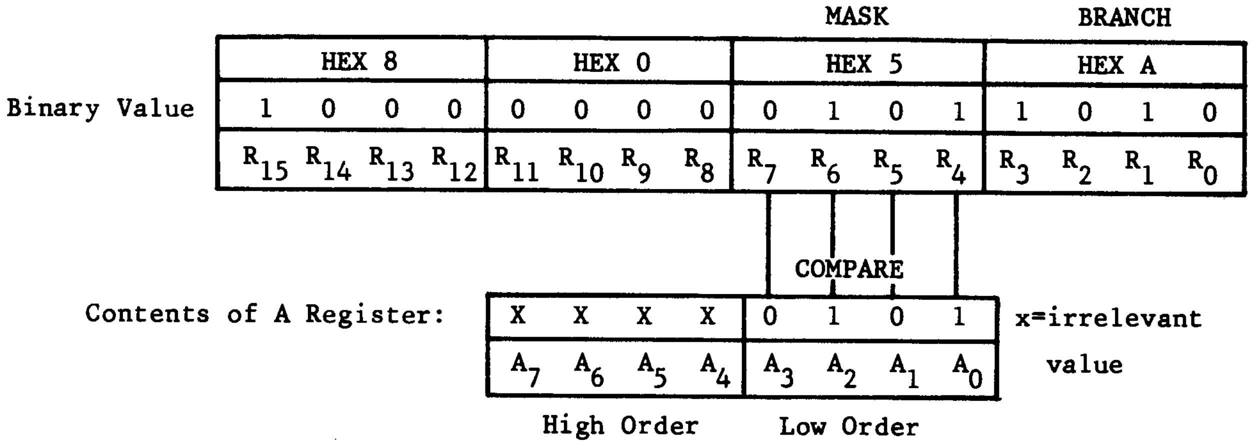
First, there are eight 4-bit Branch instructions; they cause a branch from 0 to + 15 steps from the current microprogram step if a specified condition is met. These eight 4-bit instructions belong to the conditional category of Branch instructions.

Four ROM bits (R_3 - R_0) determine the Branch address; therefore, there are 16 (2^4) unique combinations of those binary bits possible. Thus (for example) if the current ROM address is 0000_{16}, and the Branch address specified by R_3 - R_0 is 1111_2, a jump of +15 steps is performed, branching the microprogram to ROM address $000F_{16}$. Conversely, if the current ROM address is $000F_{16}$ and the Branch address specified by R_3 - R_0 is 0000_2, a jump of -15 steps is performed, branching the microprogram to ROM address 0000_{16}. These 4-bit conditional branch instructions compare the contents of a register designated by R_9 and R_8 to a mask presented by bits R_7 - R_4. If the conditions of the branch are met, branch to a step (ROM address) designated by R_3 - R_0 is made. If the conditions of the branch are not met, the microprogram continues sequentially to the next step in that routine.

Since these instructions can only compare four bits of a register specified by R_9 and R_8 with the four mask bits (R_7 - R_4), either the four high order bits or the four low order bits of the designated register is specified by ROM bit R_{10}. If R_{10} = high, or logical "one", compare bits 7-4 of the designated register with mask bits R_7 - R_4; if R_{10} = low, or logical "zero", compare bits 3-0 of the designated register with mask bits R_7 - R_4.

The following examples are typical of each four bit conditional branch instruction.

EXAMPLE: $805A_{16}$ (Br A = ML)



									MASK				BRANCH			
	HEX 8				HEX 0				HEX 5				HEX A			
Binary Value	1	0	0	0	0	0	0	0	0	1	0	1	1	0	1	0
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

Contents of A Register:	X	X	X	X	0	1	0	1	x=irrelevant value
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
	High Order				Low Order				

This instruction causes a branch by changing the current set up of IC_7 - IC_0. Actually, the only bits of ROM address that change from the current address are IC_{0-3}. This ROM address modification takes place only if the low order bits of the A register (A_3 - A_0) are identical to the mask presented in R_7 - R_4. With the A register in the state indicated above, the condition sought is met and a branch is performed.

If the current ROM step (address) was:

$$\begin{array}{lcc} \text{Binary:} & \underbrace{0\quad 0\quad 0\quad 0} & \underbrace{0\quad 1\quad 1\quad 1} \\ \text{HEX} & 0 & 7 \end{array}$$

The new address would be:

$$\begin{array}{lcc} \text{Binary:} & \underbrace{0\quad 0\quad 0\quad 0} & \underbrace{1\quad 0\quad 1\quad 0} \\ \text{HEX} & 0 & A \end{array}$$

Since the new address = 010_{10} (=A_{16})
And the previous address = 07_{10} (=7_{16})

Subtract -

Branch to address 0A (a jump of +3) occurs when specified conditions are met.

The above example microinstruction, 805A, is a typical "Branch if A = Mask Low" conditional branch in the microprocessor.

EXAMPLE: 8426_{16} (Br. A = MH)

			MASK	BRANCH
	HEX 8	HEX 4	HEX 2	HEX 6
Binary Value	1 0 0 0	0 1 0 0	0 0 1 0	0 1 1 0
	R_{15} R_{14} R_{13} R_{12}	R_{11} R_{10} R_9 R_8	R_7 R_6 R_5 R_4	R_3 R_2 R_1 R_0

COMPARE

x = irrelevant value

Contents of A register:	0	0	1	0	X	X	X	X
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

This instruction causes a branch to the ROM address designated by $R_3 - R_0$ if the high order bits of the A register ($A_7 - A_4$) are equal to the mask presented in $R_7 - R_4$. Again, the specified condition is met and the branch is performed. The ROM address is changed in the same manner as the previous $805A_{16}$ example.

The example microinstruction 8426 is a typical "Branch if = Mask High" conditional branch in the microprocessor.

EXAMPLE: $A0CF_{16}$ (Br A = TL)

			MASK	BRANCH
	HEX A	HEX 0	HEX C	HEX F
Binary Value	1 0 1 0	0 0 0 0	1 1 0 0	1 1 1 1
	R_{15} R_{14} R_{13} R_{12}	R_{11} R_{10} R_9 R_8	R_7 R_6 R_5 R_4	R_3 R_2 R_1 R_0

x = irrelevant value

X	X	X	X	1	1	X	X
A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0

This instruction causes a branch only if the configuration of TRUE (ON; logic '1') bits presented in the R_7 - R_4 mask match the configuration of TRUE (ON; logic '1'; high) bits contained in the four low order bits (3-0) of a register designated by ROM bits R_8 and R_9. ROM bits R_{15} - R_{11} designate Branch if True; bit R_{10} (=0) designates four low order bits of the register selected by R_8 and R_9, which in this case is the A register.

Generally speaking, this is called a Branch if True Low (Branch if A register low order TRUE bits match R_7 - R_4 TRUE bits). If any one of the TRUE mask bits R_7 - R_4) do not match up to a corresponding true bit in A_3 - A_0, no branch occurs.

ROM address is modified for branch in the same manner as the previous $805A_{16}$ example.

Since only TRUE mask bits are being compared, any other TRUE bits in the selected register not corresponding to TRUE bits in the mask are ignored.

If "Branch IF True High" is the instruction, the four high order bits of the A register would be used to compare with the TKUE mask bits presented in R_7 - R_4.

EXAMPLE: $B03F_{16}$ (A = FL)

			MASK	BRANCH
	HEX B	HEX 0	HEX 3	HEX F
Binary Value	1 0 1 1	0 0 0 0	0 0 1 1	1 1 1 1
	R_{15} R_{14} R_{13} R_{12}	R_{11} R_{10} R_9 R_8	R_7 R_6 R_5 R_4	R_3 R_2 R_1 R_0

Contents of A Register:	X	X	X	X	0	0	X	X	X = irrelevant
	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	

This instruction causes a branch if the configuration of false (low; 0) bits presented in the $R_7 - R_4$ FALSE mask bits (bits R_7 and R_6 in this case) match up to corresponding low order FALSE bits in the A register (A_3 and A_2 in this case).

A "Branch IF False High" instruction follows the same format as the above example. The only difference being that the FALSE high order bits of the selected register (the A register in this example) would be compared.

The two remaining conditional branch instructions have the capability to branch 0 to ± 255 steps from the current step in the microprogram. This is achieved by changing $IC_7 - IC_0$. Unlike the 4 bit conditional branch instructions which actually can modify only the four low order ROM address bits ($IC_3 - IC_0$), all 8 ROM address bits can be changed to the address contained in bits $R_7 - R_0$. These instructions cause a branch only if all eight bits of the register designated by R_9 and R_8 are either =0 or ≠0.

These two instructions are as follows:

BRANCH IF REGISTER =0	1 1 0 0 1	B B	Y Y Y Y Y Y Y Y
BRANCH IF REGISTER ≠0	1 1 0 1 1	B B	Y Y Y Y Y Y Y Y

where B = selected register

EXAMPLE: DD2C$_{16}$ (Branch IF Reg ≠ 0)

	HEX D				HEX D				HEX 2				HEX C			
Binary Value	1	1	0	1	1	1	0	1	0	0	1	0	1	1	0	0
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

The example causes a branch from the current step, to step 002C or 012C, (depending on whether the current step lies within steps 0000 - 00FF or within 0100 - 01FF) if register K ('B' = 01) is any value other than zero.

The last Branch Instruction category is the Unconditional Branch. These instructions cause a direct branch to any step in the microprogram, subject to no conditions.

These two instructions are as follows:

UNCONDITIONAL BRANCH 88YY
(to 0000 thru 00FF) 1 0 0 0 1 0 0 0 Y Y Y Y Y Y Y Y

UNCONDITIONAL BRANCH 89YY
(to 0100 thru 01FF) 1 0 0 0 1 0 0 1 Y Y Y Y Y Y Y Y

UNCONDITIONAL BRANCH 8AYY
(to 0200 thru 02FF) 1 0 0 0 1 0 1 0 Y Y Y Y Y Y Y Y

UNCONDITIONAL BRANCH 8BYY
(to 0300 thru 03FF) 1 0 0 0 1 0 1 1 Y Y Y Y Y Y Y Y

EXAMPLE: 89C5$_{16}$ (UB)

	HEX 8				HEX 9				HEX C				HEX 5			
Binary Value	1	0	0	0	1	0	0	1	1	1	0	0	0	1	0	1
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

This example causes a direct branch to step 01C5 in the microprogram without meeting any logic conditions.

4) Ram Address Instructions

Load auxiliary instructions unconditionally preset a RAM address specified by R_7 - R_0 (AD_7 - AD_0), using R_8 (AD_8) as a chip enable.

The LOAD AUXILIARY instructions are as follows:

LOAD AUXILIARY 8CXX
(DATA BUFFER): 1 0 0 0 1 1 0 0 X X X X X X X X

LOAD AUXILIARY 8DXX
(WORK BUFFER): 1 0 0 0 1 1 0 1 X X X X X X X X
Where X = new RAM address bits.

EXAMPLE: $8D20_{16}$ (Load Auxiliary; Work Buffer)

	HEX 8				HEX D				HEX 2				HEX 0			
Binary Value	1	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

This example presets RAM address to 20_{16}; bit $R_8 = 1$ selects the Work Buffer RAM integrated circuits. Location 20 in the Work Buffer is the location for storage of the track header byte read from each disk sector. $8CXX_{16}$ presets to any location within the R/W Data Buffer in RAM.

5) Control Instructions

Control instructions are used to perform operations external to the microprocessor, such as sending a strobe to the 2200 CPU, or loading a R/W head in a Shugart disk drive.

The Control instructions are as follows:

```
CONTROL 1   1 1 1 0 1 1 0 0 Z Z Z Z Z Z Z Z   ECZZ
            1 1 1 0 1 1 0 1 Z Z Z Z Z Z Z Z   EDZZ

CONTROL 2   1 1 1 1 1 1 0 0 Z Z Z Z Z Z Z Z   FCZZ
```

EXAMPLE: $FC10_{16}$ (Control 2)

	HEX F				HEX C				HEX 1				HEX 0			
Binary Value	1	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0
	R_{15}	R_{14}	R_{13}	R_{12}	R_{11}	R_{10}	R_9	R_8	R_7	R_6	R_5	R_4	R_3	R_2	R_1	R_0

This instruction causes one strobe and one byte of data input to be sent to the CPU, as specified by the Control Operand 10_{16}.

2.2.3.3 Typical Microprogram

The following microprogram is the prime routine for the PCS-II disk microprocessor. The prime routine consisting of 27 steps is initiated whenever the disk unit is turned on, the RESET key on the 2200 is depressed or the format button is depressed. Refer to text at end of program for a more detailed explanation of some of the actions that occur in the program.

SAMPLE MICROPROGRAM: PCS-II DISK PRIME ROUTINE

STEP	HEX		
(IN HEX	CODE	INSTRUCTION	COMMENT
0000	FC04	CNTRL 2/04	Select Disk #1
0001	F800	A AND IMM	Clear A Register (Immediate Operand = 0's).
0002	8D00	Load Auxiliary	Select RAM address 00 in the Work Buffer.
0003	1400	A to M(+1)	Take zeroes in A register and transfer to current RAM location then increment (+1) to next RAM address. This clears locations 00-FF of the Work Register.

0004	A223	BOTL	(Branch to 0003 IF STO-2 (AD_9) is ON). This causes a loop which increments and clears each RAM location in the work buffer (256-511) per step 0003 comments.
0005	1400	A to M(+1)	Same as step 0003 except clear data buffer (RAM locations 0-255). Note that when RAM address =111111111, adding (incrementing) RAM address brings the clear operation to RAM location 000000000 (MSB carry not used).
0006	B2D5	BOFL	(Branch to step 0005 IF STO-2 (AD_8) is OFF). This causes a loop similar to steps 0003 and 0004. Per comments for step 0005 the RAM data buffer is cleared.
0007	C803	A OR IMM	Immediate Operand = HEX 03; therefore, HEX 03 is transferred to the A register.
0008	8DE8	LOAD AUXILIARY	RAM address is preset to location $E8_{16}$ (or 488_{10}) which is the 1st HEX 03 byte for disk sector format.
0009	1000	A to M(N)	Transfer HEX 03 to current RAM location (Work Buffer, step 488_{10}). No change (N) in RAM address.

000A	8DFF	LOAD AUXILIARY	RAM address is preset to location FF_{16} (or 511_{10}) which is the HEX 03 byte for disk sector Write operations; this is also the second HEX 03 in sector format.
000B	1000	A to M(N)	HEX 03 (per comments for step 000A) transferred to RAM work buffer location 511_{10}.
000C	A74E	B1TH	Branch to step 000E IF bit 7 (carry) is ON, indicating that Format Button has been depressed.
000D	88C8	U.B.	Assuming FORMAT has not been initiated, unconditionally branch to step 00C8 to continue PRIME routine.
00C8	FB00	ST1 AND IMM	Immediate Operand $=00_{16}$ to clear carry.
00C9	0000	NOOP	No operation is performed.
00CA	8D30	LOAD AUXILIARY	Location of Error Count.
00CB	F800	A AND IMM	Clear A register.
00CC	F900	K AND IMM	Clear K register.
00CD	1400	A to M(+1)	Clear Error Count.
00CE	E901	K ADD IMM	Add 1 to the K register.

00CF	919D	Br K = ML	Branch to step 00CD if the low order bits of the K register do not equal 9. This clears all the status locations in RAM.
00D0	8DE9	LOAD AUXILIARY	Location of track address.
00D1	1400	A to M(+1)	Clear track address.
00D2	FC00	CNTRL 2/00	Clear drive select.
00D3	8D0F	LOAD AUXILIARY	Location of 00 byte to be sent to CPU.
00D4	89E1	U.B.	Branch to step 01E1.

The comments provide a basic explanation of what the instruction does, however the reasons are not always evident. This paragraph helps explain the reasons behind the steps executed in a prime routine.

STEP	REASON
0000	Disk #1 is always selected. It is assumed that disk #1 will be used.
0001	Clears the A register. A clearing process has now begun to set all pertinent registers to zero.
0002	Selects location 00 of the RAM's work buffer area. The next instruction will start a routine that will clear the 256 locations in the RAM.
0003	Takes the zeroes from the A register and transfers them to location 00 of the RAM then increments RAM address.

0004 Branch 0 = TL. Branch to step 0003 if STO-2 is on (AD8). The 23 of the code A223 indicates a mask of 2 and a branch to 3. The AD8 is always turned on by a Load Aux. command and this was done in step 0002. L78-3 turns on AD8 during a Load Aux. to the work buffer. This branch command causes a loop between steps 3 and 4 to clear the 256 locations per step 3 comment.

0005 Same as step 0003 except clear data buffer (RAM locations 0-255). Note that when RAM address = 111111111, adding (incrementing) RAM address brings the clear operation to RAM location 000000000 (LMSB carry not used).

0006 (Branch to step 0005 IF bit 2 of ST_0 (AD_8) is OFF). This causes a loop similar to steps 0003 and 0004. Per comments for step 0005 the RAM data buffer is cleared.

0007 The value HEX 03 is transferred to the empty A register by an OR Immediate. (The 03 is used when formatting a disk; this code is written in every sector as an aid in locating data on a write and read.)

0008 The RAM address is preset to location 8DE8 (488) which is the first HEX 03 byte for disk sector format.

0009 Transfers the 03 to the present RAM location without increment or decrement.

000A The RAM address is preset to location 8DFF (511) which is the 03 byte for disk write; this is also the second 03 in sector format.

000B Same as step 0009 but location 8DFF (511).

000C Branch to step 000E if bit 7 of ST1 register is on (the carry bit). The carry bit has several uses, one of which is being a flag to indicate that the format button was depressed. If the carry bit is on, the program branches to step 1030 to continue the format routine, otherwise continue.

000D Assuming format has not been initiated, unconditionally branch to step 00C8 to continue prime routine.

00C8 When the microprogram was first written, steps C8 and C9 contained
the operations that steps D3 and D4 now contain; however, due to
necessary modifications in the microprogram, codes were added to
00D2 steps C8 through D2 for use in another routine. Consequently, the
unconditional branch to C8 from 0D was never changed to read
"branch to D3." Therefore steps C8 through D2 are executed during
a prime routine but are not necessary.

00D3 This Load Aux. presets the RAM address to location 8D0F preparing to recieve a strobe from the 2200.

00D4 Branch to step 01E1 and wait 10 sec. for 2200 strobe. If no strobe within 10 sec. turn motor off.

2.2.4 MODEL PCS-II/2210 MICROPROGRAM

This section contains the complete PCS-II/2210 disk microprogram. To aid in following the Format, Write and Read routines, these routines are listed in abbreviated form below.

FORMAT - WITH MOTOR OFF

1) Prime 0000-000B.
2) Format 000C, 000E-000F.
3) 400 MSEC Delay 0230-0236.
4) Format continued 0010, 0012.
5) Turn motor #1 on with 1 sec. delay 018F-0196, 0198.
6) Zero Head 001F-0025, 0027.
7) Look for sector 0 to write 01CB-01D0.
8) Write format 0031, 0034-003A, 0013-0016, 003B-0040.
9) Check and increment sectors for format write 0041-004A, 0033.
10) When sector 9 formatted, increment track 0046.
11) Check and increment tracks for format write 004D-0056.

12) Branch to write next track 0055.

13) Look for sector 0 to read 0057, 01CC-01D0.

14) Read header bytes and check for error 0031-0032, 0058-006B.

15) Set status loc. 80 bit 006C-0070.

16) Read format 0075-0079, 0017-001B, 007A-007D, 0080-0088.

17) Check CRC 0089-008D.

18) Check and increment sectors for format read 008F-0096, 0030-0032.

19) When sector 9 read decrement track 0092.

20) Check and decrement tracks for format read 0097-00A1.

21) Branch to read next track 00A2.

22) Stop format when at track 0 009C-009E.

23) Prime 0000-000D, 01DE-01EE.

WRITE-WITH MOTOR OFF

1) Prime 0000-000D.

2) Stop motor IF no 2200 strobe within 10 sec. 01DE-01EE.

3) Three initial address bytes from 2200 00D5-00D7, 00C2-00D4, 01DE-01EE, 00D5-00E0.

4) Check for illegal address 00E1-00E6, 01B5-01BC.

5) Address conversion 00EA-00F6, 00F3-01F8, 00F7-00FF, 0240-0250.

6) Select Desired Disk turn motor ON 0100-010B.

7) 1 Sec. motor on delay 0190-0197.

8) Check for platter 010C-0110.

9) Select appropriate track address & zero head (if needed) 0111-0125.

10) Zero head (if needed) 001F-0026.

11) Step head to desired track 0126-013B.

12) Head moved previously? Possible retry 013C-0140.

13) Answer last address byte 0143-0147.

14) Data from CPU (write) 014C-0153, 01C0-01C2.

15) Accept LRC byte from CPU and compare 01C3-01C9, 0157-0158, 0028-002A.

16) Read header bytes and store in memory 0058-0062.

17) Compare header bytes with requested address 0063-00D4.

18) Write routine 017B-018D.

19) First and last (error) strobe to CPU 00C4-00D4.

20) Clear drives if no 2200 strobe within 10 sec. 01DE-01EE.

READ-WITH MOTOR ON

1) Prime 0000-000D.

2) Stop motor IF no 2200 strobe within 10 sec. 01DE-01EE.

3) Three initial address bytes from 2200 00D5-00D7, 00C2-00D4, 01DE-01EE, 00D5-00E0.

4) Check for illegal address 00E1-00E6, 01B5-01BC.

5) Address conversion 00EA-00F6, 01F3-01F8, 00F7-00FF, 0240-0250.

6) Select desired disk 0100-0109.

7) Clear disk if drive is not accessed within 8 sec. of last operation of that drive 0200-0214.

8) Check for platter 010C-0110.

9) Select appropriate track address and zero head (if needed) 0111-1014.

10) Zero head (if needed) 001F-0026.

11) Step head to desired track 0126-013B.

12) Head moved previously. Possible retry 013C-0140.

13) Answer last address byte 0143-0147, 0148-014B, 0157-0158 0028-002A.

14) Read header bytes and store in memory 0058-0062.

15) Compare header bytes with requested address 0063-0073.

16) Read routine 0075-0088.

17) Check CRC 0089-008E.

18) RAM location 0F contains a 00 byte 0159-0163.

19) Send data and LRC on read 0164-016D.

20) Clear drives if no 2200 strobe within 10 sec. 01DE-01EE.

	STEP	CODE	KEY	COMMENT
PRIME	0000	FC04	CNTRL-2	SELECT DISK #1
	0001	F800	A AND IM	CLEAR A REG.
	0002	8D00	LA (ADDR = 01XX)	M TO 256
	0003	1400	A TO M(+1)	CLEAR 256-511
	0004	A223	BOTL	BR. AD8 ON
	0005	1400	A TO M(+1)	CLEAR 0-255
	0006	B2D5	BOFL	BR. AD8 OFF
	0007	C803	A OR IM	03 TO A REG.
	0008	8DE8	LA (ADDR = 01XX)	M TO 488
	0009	1000	A TO M(N)	03 TO 488
	000A	8DFF	LA (ADDR = 01XX)	M TO 511
	000B	1000	A TO M(N)	2ND 03 TO 511
FMT.	000C	A74E	B1TH	BR. CARRY ON
NO FMT.	000D	89DE	UB (IC = 01XX)	BR. TO BEGINNING
	000E	FB00	ST1 AND IM	CLEAR CARRY
	000F	8A30	UB (IC = 10XX)	BR. TO 400 MSEC. DELAY
	0010	A742	B1TH	BR. CARRY ON
	0011	8800	UB (IC = 00XX)	BR. TO PRIME
	0012	898F	UB (IC = 01XX)	BR. TO 1 SEC. DELAY
WRITE FORMAT CONT.	0013	E901	K ADD IM	K+1
	0014	DD13	BR K <> 0	BR. <> 819.2 MSEC.
	0015	EC26	CNTRL-1	PRESET CRC
	0016	883B	UB (IC = 00XX)	
READ ROUTINE CONT.	0017	F900	K AND IM	CLEAR K REG.
	0018	E901	K ADD IM	K+1
	0019	95B8	BR K <> MH	BR. <> 563.2 MSEC.
	001A	F900	K AND IM	CLEAR K REG.
	001B	887A	UB (IC = 00XX)	
ZERO HEAD	001C	FFFF		
	001D	FFFF		
	001E	FFFF		
	001F	FC23	CNTRL-2	START 40 MSEC.
	0020	A240	BOTL	BR. 40 MSEC. ON
	0021	B7D5	B1FH	BR. = TRACK 00
	0022	EC10	CNTRL-1	SET DIRECTION
	0023	EC50	CNTRL-1	STEP -> TRACK 00
	0024	881F	UB (IC = 00XX)	
	0025	A747	B1TH	BR. CARRY ON
	0026	8926	UB (IC = 01XX)	BR. READ/WRITE
	0027	89CB	UB (IC = 01XX)	
	0028	A718	B1TH	BR. SECTOR MARK ON
	0029	B7E9	B1FH	BR. SECTOR MARK OFF
	002A	8858	UB (IC = 00XX)	
	002B	8A20	UB (IC = 10XX)	
	002C	FFFF		
	002D	FFFF		
	002E	FFFF		
	002F	FFFF		

```
0030   B7E0   B1FH               BR. SECTOR MARK OFF
0031   A584   BKTH               BR. TO WRITE FORMAT
0032   8858   UB (IC = 00XX)     BR. TO READ FORMAT

FORMAT WRITE
0033   B7E3   B1FH               BR. SECTOR MARK OFF
0034   8DD4   LA (ADDR = 01XX)   WRITE FORMAT
0035   EC26   CNTRL-1            WRITE GATE ON
0036   EC26   CNTRL-1            PRESET CRC
0037   F900   K AND IM           CLEAR K REG.
0038   E901   K ADD IM           K+1
0039   DD38   BR K <> 0          BR. <> 819.2 MSEC.
003A   8813   UB (IC = 00XX)
003B   A22B   BOTL               BR. AD8 ON
003C   B2DC   BOFL               BR. AD8 OFF
003D   E901   K ADD IM           K+1
003E   B5BD   BKFH               BR. <> 103.4 MSEC.
003F   F900   K AND IM           CLEAR K REG.
0040   EC24   CNTRL-1            STOP WRITE

SECTORS COUNTED
DURING
FORMAT WRITE
0041   0000   NOOP(N)
0042   0000   NOOP(N)
0043   8DEA   LA (ADDR = 01XX)   SECTOR LOCATION
0044   2000   M TO A(N)          SECTOR LOC. TO A
0045   D809   A XOR IM
0046   800D   BR A = ML          BR. = SECTOR 9
0047   2000   M TO A(N)          SECTOR TO A
0048   E801   A ADD IM           SECTOR +1
0049   1000   A TO M(N)          NEXT SECTOR TO M
004A   8833   UB (IC = 00XX)     BR. TO WRITE NEXT SECTOR

TRACKS COUNTED
DURING FORMAT WRITE
004B   0000   NOOP(N)
004C   0000   NOOP(N)
004D   1800   A TO M(-1)         SECTOR 0 TO M
004E   2000   M TO A(N)          TRACK LOC. TO A
004F   E801   A ADD IM           TRACK +1
0050   A426   BATH               BR. = TRACK 32
0051   1000   A TO M(N)          NEXT TRACK TO M
0052   EC44   CNTRL-1            STEP -> TRACK 34
0053   FC23   CNTRL-2            START 40 MSEC.
0054   A244   BOTL               BR. 40 MSEC. ON
0055   89CB   UB (IC = 01XX)     BR. TO SECTOR 0 NEXT TRACK
0056   9031   BR A <> ML         BR. TRACK <> 35
0057   89CC   UB (IC = 01XX)     BR. TO READ SECTOR 0

READ HEADER
BYTES AND
STORE IN MEMORY
0058   8D20   LA (ADDR = 01XX)   TRACK BYTE LOCATION
0059   F900   K AND IM           CLEAR K REG.
005A   E901   K ADD IM           K+1
005B   95CA   BR K <> MH
005C   EC15   CNTRL-1            READ GATE ON
005D   EC35   CNTRL-1            PRESET CRC
005E   A21E   BOTL               BR. WRDY = 1
005F   1400   A TO M(+1)         TRACK BYTE TO M
```

```
0060   B2E0   BOFL                   BR. WRDY = 0
0061   1400   A TO M(+1)             SECTOR BYTE TO M
0062   EC35   CNTRL-1                PRESET CRC
-----------------------------------------------------------------
COMPARE HEADER BYTES WITH REQUESTED ADDRESS
0063   8D20   LA (ADDR = 01XX)       TRACK BYTE
0064   2100   M TO K(N)              TRACK BYTE TO K
0065   8DE9   LA (ADDR = 01XX)       TRACK ADDRESS
0066   5D00   K XOR M(+1)RTB         1
0067   DDA3   BR K <> 0              TRACK ERROR
0068   2100   M TO K(N)              SECTOR ADDRESS TO K
0069   8D21   LA (ADDR = 01XX)       SECTOR BYTE
006A   5D00   K XOR M(+1)RTB         2
006B   DDA8   BR K <> 0              SECTOR ERROR
-----------------------------------------------------------------
006C   8D31   LA (ADDR = 01XX)       STATUS LOCATION
006D   C980   K OR IM                CRC ERROR BIT
006E   4900   K OR M(-1)             CRC ERROR TO M
006F   F900   K AND IM               CLEAR K REG.
0070   A745   B1TH                   BR. CARRY ON
0071   8D10   LA (ADDR = 01XX)       1ST ADDRESS BYTE
0072   2100   M TO K(N)              1ST BYTE TO K
0073   B5B5   BKFH                   BR. IF READ
0074   897B   UB (IC = 01XX)         BR. TO WRITE
-----------------------------------------------------------------
READ ROUTINE INCLUDING FORMAT
0075   F900   K AND IM               CLEAR K REG.
0076   8C00   LA (ADDR = 00XX)       DATA BUFFER
0077   E901   K ADD IM               K+1
0078   95B7   BR K <> MH             BR. <> 563.2 MSEC.
0079   8817   UB (IC = 00XX)
007A   EC25   CNTRL-1                PRESET CRC
007B   A21B   BOTL                   BR. WRDY = 1
007C   1400   A TO M(+1)             DATA TO MEMORY
007D   8880   UB (IC = 00XX)         BR. TO CHECK AD8
007E   C904   K OR IM
007F   88C4   UB (IC = 00XX)         BR. SEND CRC ERROR
0080   A225   BOTL                   BR. AD8 ON
0081   B2E1   BOFL                   BR. WRDY = 0
0082   1400   A TO M(+1)             DATA TO MEMORY
0083   A225   BOTL                   BR. AD8 ON
0084   887B   UB (IC = 00XX)         BR. TO WAIT WRDY
0085   A215   BOTL                   BR. WRDY = 1 (1ST CRC)
0086   1400   A TO M(+1)             CRC TO M
0087   B2E7   BOFL                   BR. WRDY = 0 (2ND CRC)
0088   EC10   CNTRL-1                STOP READ
-----------------------------------------------------------------
CHECK CRC
0089   2401   M TO A(+1)             1ST CRC TO A
008A   DC2B   BR A <> 0              BR. IF CRC ERROR
008B   2001   M TO A(N)              2ND CRC TO A
008C   DC2B   BR A <> 0              BR. IF CRC ERROR
008D   A74F   B1TH                   BR. CARRY ON
-----------------------------------------------------------------
008E   8959   UB (IC = 01XX)
008F   8DEA   LA (ADDR = 01XX)       SECTOR LOCATION
```

1. XOR REQUESTED TRACK WITH TRACK READ
2. XOR REQUESTED SECTOR WITH SECTOR READ

SECTORS COUNTED FORMAT READ

```
0090    2000    M TO A(N)             SECTOR TO A
0091    D809    A XOR IM
0092    8007    BR A = ML             BR. = SECTOR 9
0093    2000    M TO A(N)             SECTOR TO A
0094    E801    A ADD IM              SECTOR +1
0095    1000    A TO M(N)             NEXT SECTOR TO M
0096    8830    UB (IC = 00XX)        BR. TO READ NEXT SECTOR
```

TRACKS COUNTED FORMAT READ

```
0097    1800    A TO M(-1)            SECTOR 0 TO M
0098    2000    M TO A(N)             TRACK LOC. TO A
0099    E8FF    A ADD IM              TRACK -1
009A    1000    A TO M(N)             NEXT TRACK TO M
009B    D8FF    A XOR IM
009C    DC9F    BR A <> 0             BR. <> TRACK 0
009D    FB00    ST1 AND IM            CLEAR CARRY
009E    8800    UB (IC = 00XX)        BR. TO PRIME
009F    EC54    CNTRL-1               STEP HEAD -> TRACK 0
00A0    FC23    CNTRL-2               START 40 MSEC.
00A1    A241    BOTL                  BR. 40 MSEC. ON
00A2    89CC    UB (IC = 01XX)        BR. TO SECTOR 0 NEXT TRACK
```

ERROR ROUTINE

```
00A3    EC04    CNTRL-1               STOP READ
00A4    8D31    LA (ADDR = 01XX)      STATUS LOCATION
00A5    2000    M TO A(N)             STATUS TO A
00A6    C808    A OR IM               TRACK ERROR BIT
00A7    1000    A TO M(N)             TRACK ERROR TO M
00A8    EC04    CNTRL-1               STOP READ
00A9    F900    K AND IM              CLEAR K REG.
00AA    8D30    LA (ADDR = 01XX)      ERROR COUNT
00AB    2000    M TO A(N)             ERROR COUNT TO A
00AC    E801    A ADD IM              A+1
00AD    1400    A TO M(+1)            ERROR COUNT TO M
00AE    0000    NOOP(N)
00AF    0000    NOOP(N)
00B0    B7B2    B1FH                  BR. CARRY OFF
00B1    89A3    UB (IC = 01XX)
00B2    8424    BR A = MH             BR. 32 ERRORS
00B3    8828    UB (IC = 00XX)        BR. REREAD SECTOR
00B4    2000    M TO A(N)             STATUS TO A
00B5    F900    K AND IM              CLEAR K REG.
00B6    B478    BAFH                  BR. FORMAT BYTE ERROR
00B7    887E    UB (IC = 00XX)
00B8    A08B    BATL                  BR. TRACK ERROR
00B9    C902    K OR IM               SECTOR ERROR BIT
00BA    88C4    UB (IC = 00XX)        SECTOR ERROR TO 2200
00BB    A419    BATH                  BR. IF HEAD MOVED
00BC    C810    A OR IM
00BD    1000    A TO M(N)             HEAD MOVED STATUS TO M
00BE    8D10    LA (ADDR = 01XX)      1ST ADDRESS BYTE
00BF    2000    M TO A(N)             1ST ADDRESS BYTE TO A
```

```
00C0   C820   A OR IM
00C1   8911   UB (IC = 01XX)          BR. TO ZERO HEAD

REINITIALIZE
00C2   0000   NOOP(N)
00C3   FB00   ST1 AND IM              CLEAR CARRY
00C4   8D30   LA (ADDR = 01XX)        ERROR LOCATION
00C5   F800   A AND IM                CLEAR A REG.
00C6   F90F   K AND IM                MASK OUT HIGH ORDER
00C7   1400   A TO M(+1)              ERROR COUNT
00C8   E910   K ADD IM
00C9   9597   BR K <> MH
00CA   8DE9   LA (ADDR = 01XX)        TRACK LOCATION
00CB   1400   A TO M(+1)              CLEAR TRACK
00CC   F90F   K AND IM                MASK OUT HIGH ORDER
00CD   FC00   CNTRL-2                 CLEAR SELECT
00CE   8D32   LA (ADDR = 01XX)        *
00CF   1100   K TO M(N)
00D0   A612   BOTH                    BR. NOT REINITIALIZE
00D1   89D8   UB (IC = 01XX)
00D2   B672   BOFH                    WAIT KBD
00D3   FC13   CNTRL-2                 STROBE TO 2200
00D4   89DE   UB (IC = 01XX)

3 ADDRESS BYTES
FROM 2200
00D5   B6D5   BOFH                    LOOK END STROBE
00D6   A618   BOTH                    BR. NOT REINITIALIZE
00D7   88C2   UB (IC = 00XX)          BR. TO REINITIALIZE
00D8   0400   NOOP(+1)                MEMORY +1
00D9   1100   K TO M(N)               ADDRESS BYTE TO M
00DA   E801   A ADD IM                A+1
00DB   B67B   BOFH                    WAIT KBD
00DC   FC13   CNTRL-2                 STROBE TO 2200
00DD   A03F   BATL                    BR. = 3RD BYTE
00DE   89EE   UB (IC = 01XX)          LOOK NEXT BYTE
00DF   A24F   BOTL                    BR. 40 MSEC. ON
00E0   0000   NOOP(N)                 DELAY

CHECK FOR
ILLEGAL
ADDRESS
00E1   8D10   LA (ADDR = 01XX)        1ST ADDRESS BYTE
00E2   2400   M TO A(+1)              1ST BYTE TO A
00E3   9007   BR A <> ML              BR. ILLEGAL ADDRESS
00E4   2400   M TO A(+1)              2ND BYTE TO A
00E5   9407   BR A <> MH              BR. ILLEGAL ADDRESS
00E6   89B5   UB (IC = 01XX)          BR. NOT ILL. ADDR.

ILL.
ADDR.
TO
2200
00E7   F900   K AND IM                CLEAR K REG.
00E8   C901   K OR IM                 01 -> K
00E9   88C4   UB (IC = 00XX)          ILL. ADDR. TO 2200

00EA   8D11   LA (ADDR = 01XX)        2ND ADDRESS BYTE
00EB   2400   M TO A(+1)              2ND BYTE TO A
00EC   2100   M TO K(N)               3RD BYTE TO K
00ED   1800   A TO M(-1)
00EE   1100   K TO M(N)
00EF   F800   A AND IM                CLEAR A REG.
```

*ADDRESS, ERROR OR REINT TO 2200

TRACK + SECTOR CONVERSION

```
00F0    C8F6    A OR IM                 F6 -> A
00F1    C9FF    K OR IM                 FF -> K
00F2    FB00    ST1 AND IM              CLEAR CARRY
00F3    8D11    LA (ADDR = 01XX)
00F4    3C00    A PWC M(+1)RTB          RESULT TO A
00F5    3D00    K PWC M(+1)RTB          RESULT TO K
00F6    89F3    UB (IC = 01XX)
00F7    8DE9    LA (ADDR = 01XX)
00F8    2100    M TO K(N)
00F9    E901    K ADD IM                K+1
00FA    1100    K TO M(N)
00FB    88EF    UB (IC = 00XX)
00FC    E80A    A ADD IM                A+10
00FD    8DEA    LA (ADDR = 01XX)        SECTOR LOC.
00FE    1000    A TO M(N)
00FF    8A40    UB (IC = 10XX)
```

SELECT DESIRED DISK AND TURN MOTOR ON (LOAD HEAD)

```
0100    2000    M TO A(N)               1ST BYTE TO A
0101    A417    BATH                    BR. DISK #2
0102    B275    BOFL                    BR. MOTOR OFF (#1)
0103    FC01    CNTRL-2                 SELECT #1
0104    8A00    UB (IC = 10XX)
0105    FC01    CNTRL-2                 SELECT #1
0106    8990    UB (IC = 01XX)          BR. TO 1 SEC. DELAY
0107    B6BA    BOFH                    BR. MOTOR OFF (#2)
0108    FC02    CNTRL-2                 SELECT #2
0109    8A10    UB (IC = 10XX)
010A    FC02    CNTRL-2                 SELECT #2
010B    8990    UB (IC = 01XX)
```

CHECK FOR PLATTER

```
010C    FC23    CNTRL-2                 START 40 MSEC.
010D    A24F    BOTL                    BR. 40 MSEC. ON
010E    88E7    UB (IC = 00XX)          ERROR
010F    B7ED    B1FH                    BR. SECTOR MARK OFF
0110    A240    BOTL                    BR. 40 MSEC. ON
```

SELECT APPROPRIATE TRACK ADDRESS AND CHECK IF HEAD SHOULD BE ZEROED

```
0111    0000    NOOP(N)
0112    8DE9    LA (ADDR = 01XX)        TRACK ADDRESS
0113    2100    M TO K(N)               TRACK TO K
0114    0000    NOOP(N)
0115    A41C    BATH                    BR. DISK #2
0116    8D25    LA (ADDR = 01XX)        DISK #1 TRACK LOC.
0117    A42E    BATH                    TRACK ERROR, ZERO HEAD
0118    2000    M TO A(N)               TRACK LOC. TO A
0119    A48D    BATH                    ALREADY ZEROED HEAD
011A    8821    UB (IC = 00XX)          BR. TO ZERO HEAD
011B    0000    NOOP(N)
011C    8923    UB (IC = 01XX)
011D    8925    UB (IC = 01XX)
011E    F800    A AND IM                CLEAR A REG.
011F    1000    A TO M(N)               SET TRACK LOC. TO 00
```

```
0120   8821   UB (IC = 00XX)       BR. TO ZERO HEAD
0121   0000   NOOP(N)
0122   0000   NOOP(N)
0123   8D26   LA (ADDR = 01XX)     DISK #2 TRACK LOC.
0124   8917   UB (IC = 01XX)

STEP HEAD TO DESIRED TRACK
0125   F87F   A AND IM             MASK OUT 80 BIT
0126   C980   K OR IM              80 -> K (HEAD ZEROED)
0127   1100   K TO M(N)
0128   D87F   A XOR IM
0129   3C00   A PWC M(+1)RTB       RESULT TO A
012A   A74D   B1TH                 BR. CARRY ON
012B   D8FF   A XOR IM
012C   89D1   UB (IC = 01XX)
012D   E801   A ADD IM             A+1
012E   EC00   CNTRL-1              SET DIRECTION -> TRACK 34
012F   EC40   CNTRL-1              STEP HEAD
0130   FC23   CNTRL-2              START 40 MSEC.
0131   A241   BOTL                 BR. 40 MSEC. ON
0132   E8FF   A ADD IM             A-1
0133   CC39   BR A = 0
0134   B7B6   B1FH                 BR. CARRY OFF
0135   892E   UB (IC = 01XX)
0136   EC10   CNTRL-1              SET DIRECTION -> TRACK 00
0137   EC50   CNTRL-1              STEP HEAD
0138   8930   UB (IC = 01XX)
0139   FB00   ST1 AND IM           CLEAR CARRY
013A   FC23   CNTRL-2              START 40 MSEC.
013B   A24B   BOTL                 BR. 40 MSEC. ON

HEAD MOVED PREVIOUSLY?
013C   8D31   LA (ADDR = 01XX)     STATUS LOCATION
013D   2000   M TO A(N)            STATUS TO A
013E   0000   NOOP(N)
013F   0000   NOOP(N)
0140   B4E3   BAFH                 BR. HEAD NOT MOVED

ANSWER LAST ADDRESS BYTE
0141   8828   UB (IC = 00XX)       BR. READ/WRITE
0142   88C2   UB (IC = 00XX)       BR. TO REINT.
0143   8D10   LA (ADDR = 01XX)     1ST ADDRESS BYTE
0144   2800   M TO A(-1)
0145   B675   BOFH                 WAIT KBD
0146   FC13   CNTRL-2              STROBE TO 2200
0147   A44C   BATH                 BR. IF WRITE

DATA FROM CPU (WRITE)
0148   A628   BOTH                 WAIT 2200 STROBE
0149   B6D9   BOFH                 WAIT END STROBE
014A   B6E2   BOFH                 REINITIALIZE
014B   8957   UB (IC = 01XX)
014C   F800   A AND IM             CLEAR A REG.
014D   8C00   LA (ADDR = 00XX)     DATA BUFFER
014E   A62E   BOTH                 WAIT 2200 STROBE
014F   B6DF   BOFH                 WAIT END STROBE
```

```
0150   A612   BOTH                 BR. NOT REINT.
0151   88C2   UB (IC = 00XX)       BR. TO REINT.
0152   1100   K TO M(N)            DATA BYTE TO M
```

SEND DATA AND LRC ON READ

```
0153   89C0   UB (IC = 01XX)       BR. TO GENERATE LRC
0154   0000   NOOP(N)
0155   0000   NOOP(N)
0156   0000   NOOP(N)
0157   EC04   CNTRL-1              STOP READ
0158   8828   UB (IC = 00XX)       BR. TO WRITE
0159   F900   K AND IM             CLEAR K REG.
015A   8D10   LA (ADDR = 01XX)     1ST ADDRESS BYTE
015B   2800   M TO A(-1)
015C   B67C   BOFH                 WAIT KBD
015D   FC13   CNTRL-2              STROBE TO 2200
015E   0000   NOOP(N)              DELAY
015F   0000   NOOP(N)              DELAY
0160   0000   NOOP(N)              DELAY
0161   8C00   LA (ADDR = 00XX)     DATA BUFFER
0162   A48E   BATH                 BR. IF COMPARE
0163   F800   A AND IM             CLEAR A REG.
0164   B674   BOFH                 WAIT KBD
0165   FC13   CNTRL-2              STROBE TO 2200
0166   0000   NOOP(N)              DELAY
0167   0000   NOOP(N)              DELAY
0168   6C00   A ADD M(+1)RTB       GENERATE LRC
0169   B2D4   BOFL                 BR. AD8 OFF
016A   1000   A TO M(N)            LRC TO MEMORY
016B   B67B   BOFH                 WAIT KBD
016C   FC13   CNTRL-2              LRC TO 2200
016D   89DE   UB (IC = 01XX)       BR. TO BEGINNING
```

DATA FROM 2200 FOR WRITE COMPARE

```
016E   A62E   BOTH                 WAIT 2200 STROBE
016F   B6DF   BOFH                 WAIT END STROBE
0170   5D00   K XOR M(+1)RTB       1  (RESULT TO K)
0171   CD73   BR K = 0             BR. DATA COMPARE
0172   C8FF   A OR IM
0173   A225   BOTL                 BR. AD8 ON
0174   896E   UB (IC = 01XX)       BR. TO COMPARE
0175   A625   BOTH                 WAIT 2200 STROBE
0176   B6D6   BOFH                 WAIT END STROBE
0177   8009   BR A = ML            BR. NO ERROR ON COMPARE
0178   88E7   UB (IC = 00XX)       BR. SEND ERROR
0179   F900   K AND IM             CLEAR K REG.
017A   898A   UB (IC = 01XX)
```

```
017B   EC04   CNTRL-1              STOP READ
017C   F800   A AND IM             CLEAR A REG.
017D   F900   K AND IM             CLEAR K REG.
017E   E901   K ADD IM             K+1
017F   959E   BR K <> MH           BR. <> 460.8 MSEC.
```

1. XOR DATA READ WITH DATA FROM 2200

```
WRITE ROUTINE
0180   89D3   UB (IC = 01XX)
0181   EC26   CNTRL-1            WRITE GATE ON
0182   EC26   CNTRL-1            PRESET CRC
0183   B7E6   B1FH               BR. SECTOR MARK ON
0184   EC00   CNTRL-1            STOP WRITE
0185   88E7   UB (IC = 00XX)     BR. TO SEND ERROR
0186   A223   BOTL               BR. AD8 ON
0187   0000   NOOP(N)
0188   B7E8   B1FH               BR. SECTOR MARK OFF
0189   EC00   CNTRL-1            STOP WRITE
018A   B2D5   BOFL               BR. AD8 OFF
018B   0000   NOOP(N)
018C   F900   K AND IM           CLEAR K REG.
018D   88C4   UB (IC = 00XX)     LAST BYTE TO 2200
018E   0000   NOOP(N)
------------------------------------------------------------
1 SEC. MOTOR
ON DELAY
018F   FC05   CNTRL-2            SELECT DISK #1
0190   F900   K AND IM           CLEAR K REG.
0191   FC23   CNTRL-2            START 40 MSEC.
0192   A242   BOTL               BR. 40 MSEC. ON
0193   E901   K ADD IM           K+1
0194   9511   BR K <> MH
0195   9141   BR K <> ML         BR. <> 1 SEC.
0196   A748   B1TH               BR. CARRY ON
0197   890C   UB (IC = 01XX)     BR. READ/WRITE
------------------------------------------------------------
COUNT FORMAT RETRIES
0198   881F   UB (IC = 00XX)     BR. FORMAT
0199   0000   NOOP(N)
019A   0000   NOOP(N)
019B   A74D   B1TH               BR. CARRY ON
019C   88E7   UB (IC = 00XX)     ERROR TO 2200
019D   8D33   LA (ADDR = 01XX)   FORMAT RETRIES
019E   2000   M TO A(N)
019F   E801   A ADD IM           A+1
01A0   1000   A TO M(N)
01A1   A043   BATL               BR. = 4 RETRIES
01A2   89B0   UB (IC = 01XX)
------------------------------------------------------------
FLASH FORMAT LIGHT
01A3   F800   A AND IM           CLEAR A REG.
01A4   E801   A ADD IM           A+1
01A5   FC23   CNTRL-2            START 40 MSEC.
01A6   A246   BOTL               BR. 40 MSEC. ON
01A7   90F4   BR A <> ML
01A8   F800   A AND IM           CLEAR A REG.
01A9   FC00   CNTRL-2            TURN OFF DISK #1
01AA   E801   A ADD IM           A+1
01AB   FC23   CNTRL-2            START 40 MSEC.
01AC   A24C   BOTL               BR. 40 MSEC. ON
01AD   90FA   BR A <> ML
01AE   FC01   CNTRL-2            SELECT DISK #1
01AF   89A3   UB (IC = 01XX)
------------------------------------------------------------
```

```
RETRY FORMAT
01B0    F800    A AND IM               CLEAR A REG.
01B1    8DE9    LA (ADDR = 01XX)       TRACK LOC.
01B2    1400    A TO M(+1)             CLEAR TRACK
01B3    1400    A TO M(+1)             CLEAR SECTOR
01B4    881F    UB (IC = 00XX)

ILLEGAL ADDRESS?
01B5    B017    BAFL                   BR. <> ILL. ADDR.
01B6    88E7    UB (IC = 00XX)         BR. TO ERROR
01B7    B0EC    BAFL                   BR. <> ILL. ADDR.
01B8    F800    A AND IM               CLEAR A REG.
01B9    C8A2    A OR IM
01BA    3C00    A PWC M(+1)RTB         RESULT TO A
01BB    A746    B1TH                   BR. CARRY ON
01BC    88EA    UB (IC = 00XX)         TO CONVERSION

GENERATE LRC
01BD    FFFF
01BE    FFFF
01BF    FFFF
01C0    6C00    A ADD M(+1)RTB         GENERATE LRC
01C1    A223    B0TL                   BR. AD8 ON
01C2    894E    UB (IC = 01XX)         BR. TO NEXT BYTE
01C3    A623    B0TH                   WAIT 2200 STROBE
01C4    B6D4    B0FH                   WAIT END STROBE
01C5    8D34    LA (ADDR = 01XX)       LRC LOCATION
01C6    1100    K TO M(N)              LRC TO MEM.
01C7    5C00    A XOR M(+1)RTB         *  (RESULT TO A)
01C8    DCCA    BR A <> 0              BR. LRC ERROR
01C9    8957    UB (IC = 01XX)         BR. TO WRITE
01CA    88E7    UB (IC = 00XX)         BR. TO SEND ERROR

LOOK FOR INDEX AND SECTOR FMT.
01CB    C980    K OR IM                80 -> K (HEAD ZEROED)
01CC    F980    K AND IM               MASK OUT
01CD    B3ED    B1FL                   BR. INDEX OFF
01CE    A31E    B1TL                   BR. INDEX ON
01CF    B7EF    B1FH                   BR. SECTOR MARK OFF
01D0    8831    UB (IC = 00XX)

01D1    CC3C    BR A = 0
01D2    8934    UB (IC = 01XX)
01D3    F900    K AND IM               CLEAR K REG.
01D4    E901    K ADD IM               K+1
01D5    9594    BR K <> MH
01D6    8DF9    LA (ADDR = 01XX)
01D7    8981    UB (IC = 01XX)

INITIALIZE REPLY
01D8    F900    K AND IM               CLEAR K REG.
01D9    C9C0    K OR IM                C0 -> K
01DA    1100    K TO M(N)              C0 -> MEM.
01DB    B67B    B0FH                   WAIT KBD
01DC    FC13    CNTRL-2                STROBE TO 2200
01DD    89F9    UB (IC = 01XX)

01DE    FC00    CNTRL-2
01DF    8D0F    LA (ADDR = 01XX)
```

*LRC BYTE FROM 2200 WITH LRC BYTE GENERATED

CLEAR DRIVES IF NO
STROBE WITHIN 10 SEC.

```
01E0   B6DF   BOFH                 BR. STROBE ON
01E1   C9F8   K OR IM
01E2   B6DF   BOFH                 BR. STROBE ON
01E3   FC23   CNTRL-2              START 40 MSEC.
01E4   B6DF   BOFH                 BR. STROBE ON
01E5   A244   BOTL                 BR. 40 MSEC. ON
01E6   E801   A ADD IM             A+1
01E7   B6DF   BOFH                 BR. STROBE ON
01E8   DCE2   BR A <> 0            BR. <> 10 SEC.
01E9   E901   K ADD IM             K+1
01EA   B6DF   BOFH                 BR. STROBE ON
01EB   DDE2   BR K <> 0
01EC   ED88   CNTRL-1              STOP MOTOR
01ED   8D0F   LA (ADDR = 01XX)
01EE   A62E   BOTH                 BR. NO STROBE
01EF   88D5   UB (IC = 00XX)
```

CONVERSION
CONTINUED

```
01F0   FFFF
01F1   FFFF
01F2   FFFF
01F3   8D11   LA (ADDR = 01XX)
01F4   1400   A TO M(+1)
01F5   1100   K TO M(N)
01F6   B7B8   B1FH                 BR. CARRY OFF
01F7   88F7   UB (IC = 00XX)
01F8   88FC   UB (IC = 00XX)
```

CLEAR DISK IF DRIVE IS NOT ACCESSED
WITHIN 8 SEC. OF LAST OPERATION OF
THAT DRIVE

```
01F9   0000   NOOP(N)
01FA   0000   NOOP(N)
01FB   0000   NOOP(N)
01FC   89ED   UB (IC = 01XX)
01FD   FFFF
01FE   FFFF
01FF   FFFF
0200   8D40   LA (ADDR = 01XX)
0201   2100   M TO K(N)
0202   F900   K AND IM
0203   1500   K TO M(+1)
0204   2100   M TO K(N)
0205   E901   K ADD IM             K+1
0206   1100   K TO M(N)
0207   A529   BKTH                 BR. = 8 SEC.
0208   890C   UB (IC = 01XX)
0209   F900   K AND IM             CLEAR K REG.
020A   1100   K TO M(N)
020B   A41E   BATH                 BR. DISK #2
020C   EC80   CNTRL-1              CLEAR DISK #2
020D   890C   UB (IC = 01XX)
020E   ED00   CNTRL-1              CLEAR DISK #1
020F   890C   UB (IC = 01XX)
```

```
0210   8D41   LA (ADDR = 01XX)
0211   2100   M TO K(N)
0212   F900   K AND IM
0213   1900   K TO M(-1)
0214   8A04   UB (IC = 10XX)
```

COUNT # OF ERRORS

```
0215   FFFF
0216   FFFF
0217   FFFF
0218   FFFF
0219   FFFF
021A   FFFF
021B   FFFF
021C   FFFF
021D   FFFF
021E   FFFF
021F   FFFF
0220   EC04   CNTRL-1
0221   8D30   LA (ADDR = 01XX)      SECTOR ERROR LOC.
0222   2000   M TO A(N)
0223   F800   A AND IM              CLEAR ERROR
0224   1000   A TO M(N)
0225   8D38   LA (ADDR = 01XX)      CRC ERROR LOC.
0226   2000   M TO A(N)
0227   E801   A ADD IM              A+1
0228   1000   A TO M(N)
0229   841B   BR A = MH             BR. = 16 ERRORS
022A   8828   UB (IC = 00XX)        BR. TO RETRY
022B   887E   UB (IC = 00XX)        BR. TO CRC ERROR
022C   FFFF
022D   FFFF
022E   FFFF
022F   FFFF
```

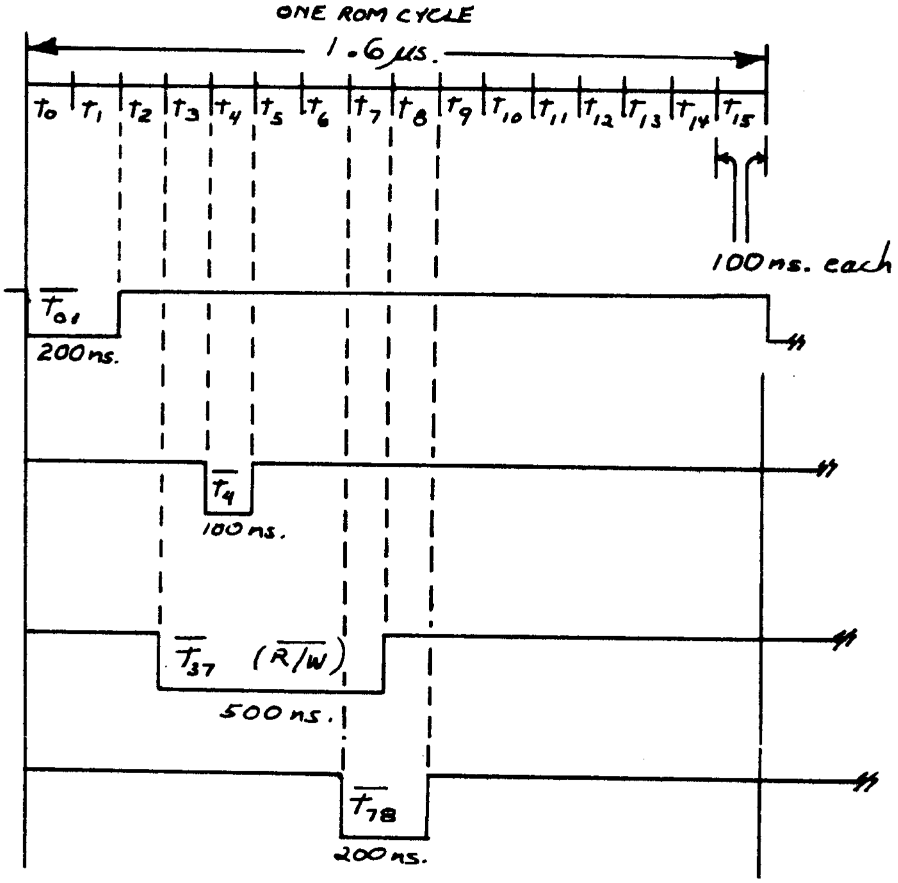
400 MSEC. DELAY

```
0230   F900   K AND IM              CLEAR K REG.
0231   FC23   CNTRL-2               START 40 MSEC.
0232   FB00   ST1 AND IM            CLEAR CARRY
0233   A242   BOTL                  BR. 40 MSEC. ON
0234   E901   K ADD IM              K+1
0235   91A1   BR K <> ML
0236   8810   UB (IC = 00XX)
```

```
0237   FFFF
0238   FFFF
0239   FFFF
023A   FFFF
023B   FFFF
023C   FFFF
023D   FFFF
023E   FFFF
023F   FFFF
```

CONVERSION CONTINUED

```
0240   809A   BR A = ML
0241   808C   BR A = ML
0242   8046   BR A = ML
0243   B0BE   BAFL
0244   8058   BR A = ML
0245   806A   BR A = ML
0246   D806   A XOR IM
0247   1000   A TO M(N)
0248   8D10   LA (ADDR = 01XX)
0249   8900   UB (IC = 01XX)
024A   D80E   A XOR IM
024B   8A47   UB (IC = 10XX)
024C   D80C   A XOR IM
024D   8A47   UB (IC = 10XX)
024E   6000   A ADD M(N)
024F   6000   A ADD M(N)
0250   8A48   UB (IC = 10XX)
0251   FFFF
0252   FFFF
0253   FFFF
0254   FFFF
0255   FFFF
0256   FFFF
0257   FFFF
0258   FFFF
0259   FFFF
025A   FFFF
025B   FFFF
025C   FFFF
025D   FFFF
025E   FFFF
025F   FFFF
0260   FFFF
0261   FFFF
0262   FFFF
0263   FFFF
0264   FFFF
0265   FFFF
0266   FFFF
0267   FFFF
0268   FFFF
0269   FFFF
026A   FFFF
026B   FFFF
026C   FFFF
026D   FFFF
026E   FFFF
026F   FFFF
```


ONE ROM CYCLE
1.6 µs.
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15}
100 ns. each
$\overline{T_{01}}$
200 ns.
$\overline{T_4}$
100 ns.
$\overline{T_{37}}$ $(\overline{R/W})$
500 ns.
$\overline{T_{78}}$
200 ns.

2.3.3 INPUT TO MICROPROCESSOR FROM CPU

Data from the 2200 enters the microprocessor via the K register L65 and L67 from here it is loaded into RAM. This is accomplished in the following manner. To begin, R_8 and R_9 select the K register inputs of the A Bus multiplexer L51-L54 thus presenting the K register contents to the 74181 ALU L62 and L63. ALUs are capable of performing several logic and arithmetic functions selected by L37 pins 2, 4, 10, 12 and L34 pin 6. The configuration of these pins is dependent on the output of L35 and L36 which decode ROM bits R_{15} - R_{11} to determine the instruction and the ALU function necessary to perform this instruction. In this case the ALU chips will direct the data to its outputs (C_7 - C_0) unprocessed. From here the data is available for storage in the RAM L79 - L82. Data will be written in a location designated by the address contained in L75, L76 and L77. These can be incremented, decremented, or loaded in block form, thus the ROM can directly control RAM address. To actually write the data, a line designated R/W must be changed; this is done by L8 and L21. L8, L13, L14, L21, L34, L39, L50, L64 and L66, make up the ROM instruction decoder. Here is a list of Ics and their function in this decoder:

L39	Pin 6	- Indicates to send the result to the selected register and increment RAM address.
L39	Pin 5	- Increment RAM address.
L39	Pin 7	- Decrement RAM address.
L39	Pin 9	- Control Command.
L39	Pin 10	- Immediate Instruction.
L50	Pin 4	- A register clock.
L50	Pin 5	- K register clock.
L50	Pin 7	- Status register 1 clock.
L50	Pin 9	- Load Auxiliary.
L50	Pin 10	- Unconditional Branch.
L50	Pin 11	- Branch if the high order register bits meet the condition.
L50	Pin 12	- Branch if the low order register bits meet the condition.

With the data in RAM the processor is now ready to accept more data or process present data.

2.3.4 OUTPUT OF MICROPROCESSOR TO DISK

Data from the microprocessor makes its way to the disk in the following manner. At T_{01} the data in the current RAM address is loaded into the RAM output register; from here it is available for input to the A register L26 and L55. With write gate (WTG) on, these inputs to the A register are selected via L40-L43 and A register clock (ACU), L106 (L103) pin 11, loads the first byte of data in the A register. This data is now converted, from parallel to serial by L57, L69, L70, L89 and L91 (L90) and sent to the disk to be written in the double frequency (2F) format via L57 and L66. After the first byte, the A register is clocked by Word Ready (WDRDY) which occurs every time 8 bits have been converted from parallel to serial.

2.3.5 INPUT FROM DISK TO MICROPROCESSOR

Information from the disk is first converted from serial to parallel by L45 and L46. Again WDRDY clocks the data into the A register each time 8 bits have been assembled into a parallel word. From the A register the data must be loaded into RAM. This is accomplished by R_8 and R_9 selecting the A register inputs to the A Bus Multiplexer L51-L54 thus presenting the data to the ALU L62 and L63. Again the ROM instruction decoder has put the RAM in a write mode through L8 and L21 and the ALU Function Decoder L35 and L36 has allowed the data to pass through the ALU unchanged. When this data appears on the C Bus it is immediately stored in the memory location indicated by L75, L76, and L77. Memory address is then incremented and the processor is ready to accept the next byte from the disk.

2.3.6 OUTPUT FROM MICROPROCESSOR TO CPU

Once data from the disk is loaded into RAM it is ready to be transferred to the 2200 CPU. This is the path that the transfer follows. Since the RAM is normally in a read mode, the data can be accessed by simply changing the RAM address. This presents data to the RAM output register L99 and L100; at T_{01} it is loaded, thus a new byte of data can be loaded every ROM cycle (1.6 us). From the RAM output register the data is buffered through L98 and L101 which are strobed by L9 and L105 (L102). Since these are under ROM control, the data transfer between the processor and th 2200 is synchronized. Synchronization is necessary because the processor is faster than the 2200; the processor must wait for the 2200 to become ready which is accomplished in the microprogram. When the 2200 indicates that it is ready to accept another byte, the above process is repeated. This continues until all data is transferred.

2.3.7 DISK CONTROL OPERATIONS

All disk operations are controlled by the ROM with two instructions: Control 1 and Control 2. The control commands are listed below with the logic gates that decode them and a short description of each.

CNTRL1 AND R0 L93-2

Turns the Read Gate (RDG) on to enable the processor to read from the disk.

CNTRL1 AND R1 L93-5

Turns the Write Gate (WTG) on to enable the processor to write on the disk.

CNTRL1 AND R2

NOT USED.

CNTRL1 AND R3

NOT USED.

CNTRL1 AND R4 L93-12

Head Direction Select (HD DIR). A high signal sets the head direction toward track 34 while a low signal sets the head direction to track 0. This control signal must be used with a disk selected while the head is loaded and in conjunction with head step.

CNTRL1 AND R5 L9-8

PRC. Used as a reset command before a read or write to clear the CRC register and preset logic that will be used for the read/write.

CNTRL1 AND R6 L1-12

Head Step (HDST). This is the output of a one-shot which moves the head one track. A disk must be selected, the head loaded and a direction chosen to properly move the head.

CNTRL1 AND R7 L17-8	Clear disk #2 (HM_2).
CNTRL1 AND R8 L17-6	Clear disk #1 (HM_1).
CNTRL2 AND RO L92-3	Select disk #1 (DK1)
CNTRL2 AND R1 L92-6	Select disk #2 (DK2).
CNTRL2 AND R2 L92-10	Set disk #1 (DK1).
CNTRL2 AND R3	NOT USED.
CNTRL2 AND R4 L105-12 (L102-12)	Strobe to 2200. Not applicable to this section.
CNTRL2 AND R5 L7-6	40 ms delay. Because of the speed of the ROM cycle, delays are necessary to halt the microprogram while mechanical actions take place. Any number of 40 ms delays can be used to create longer delays.
CNTRL2 AND R6	NOT USED.
CNTRL2 AND R7	NOT USED.

2.3.8 INDEX/SECTOR SEPARATOR

The index/sector separator consists of L68 and L88 (L87). Sector pulses are outputed at L88-1 (L87-1) and the index pulse at L88-10 (L87-10). From there they are applied to L52 and L54 as bits of the ST1 register. They are then manipulated by the microprogram to determine appropriate sector locations.

2.3.9 CYCLIC REDUNDANCY CHECK

The CRC circuit is comprised of L27-L30. During a write command, data being written is shifted through the CRC circuitry to develop a unique 16-bit code that is written on the disk in two bytes at the end of the 256 byte data transfer. When the 256 bytes of data are read, the data is shifted through the CRC circuitry and a 16-bit CRC code is stored. This code is then compared with the CRC read from the disk and if the data was read correctly, the two 16-bit codes will be equal. No provision is made to determine if data was correctly written, therefore, if a CRC error develops on a read, it cannot be determined if data was written or read incorrectly.

2.3.10 FORMAT ROUTINE

When the format button is depressed, L3 ( ) is preset by pin 2. This action turns the carry bit on, which is used as an indicator by the microprogram. PF (prime/format) is generated which resets the ROM IC to step 0000. From this point, the microprogram clears the hardware for the next 12 steps which are also used during a prime routine. The 13th step of the program (step 000C) checks to see whether the carry bit is on or off. Because the carry bit was turned on, the program branches to 1030 and begins a 400 ms delay to determine whether the carry bit was legitimately turned on. If a noise spike generated the format signal, the spike will be cleared by one of the 13 T_4 signals applied to L3-1 during the 13 steps of the prime routine. When the 400 ms delay is completed with the carry bit still on, disk #1 is selected, the motor is turned on the head is loaded and stepped to track zero. When the correct sector is found, sector zero, on track zero, the write gate is turned on and the following data is written on the disk from these memory locations:

302 BYTES PER SECTOR

LOCATION

01D4 to 01E7	01E8	01E9	01EA	01EB to 01FE	01FF	0000-00FF		
20 Bytes 00	03 Byte	Track Address	Sector Address	20 Bytes 00	03 Byte	256 Bytes 00	CRC#1	CRC#2

Twenty bytes of zeroes are written immediately after the sector mark is decoded to allow for the mechanical tolerances of the sector pick-up from drive to drive. A byte of 03 is written and is used as an indicator on a read to allow the logic time to decode the track and sector addresses. The second set of 20 bytes of zeroes is written so that on subsequent read or write commands, the logic has time to determine whether a read or write is to be executed. The second 03 byte is written and used as an indicator that data follows on a read or write. When the second 03 is decoded on a read command, the next 256 bytes (plus 2 bytes of CRC) is read data; on a write command, start writing 256 bytes of data (plus 2 bytes of CRC).

When the processor has written one sector with the above data, the process is repeated for all sectors in the track, then for all tracks on the disk. After the 35 tracks are formatted, there is a read format routine (also used during a read) that checks the track and sector addresses along with the CRC codes. This read format routine is done beginning with track 34 and ending with track 0. When the disk has been verified after a complete format routine, there is an unconditional branch to step 0000 where the microprogram does its initializing before cycling on step 01EB waiting for a 2200 strobe.

2.4 TROUBLESHOOTING PROCEDURES

2.4.1 INTRODUCTION

A special extender board has been designed which allows use of a 7069 lightboard for troubleshooting the 7180 and 7279 microprocessors. The purpose of the light board is to observe internal microprocessor conditions and to exercise, via manual control, all microprocessor functions. This section explains the set up procedure for the extender and light board and also information on how to use the light board to facilitate repair of the 7180 and 7279.

2.4.2 EQUIPMENT REQUIRED

QTY.	DESCRIPTION	PART #
1	Extender Board	210-7176
2	24 Pin Ribbon Cable	220-3014
1	Mini-diskette Ribbon Cable Extender	N/A
1	Mini-diskette Power Calbe Extender	N/A

2.4.3 LIGHTBOARD INSTALLATION PROCEDURE

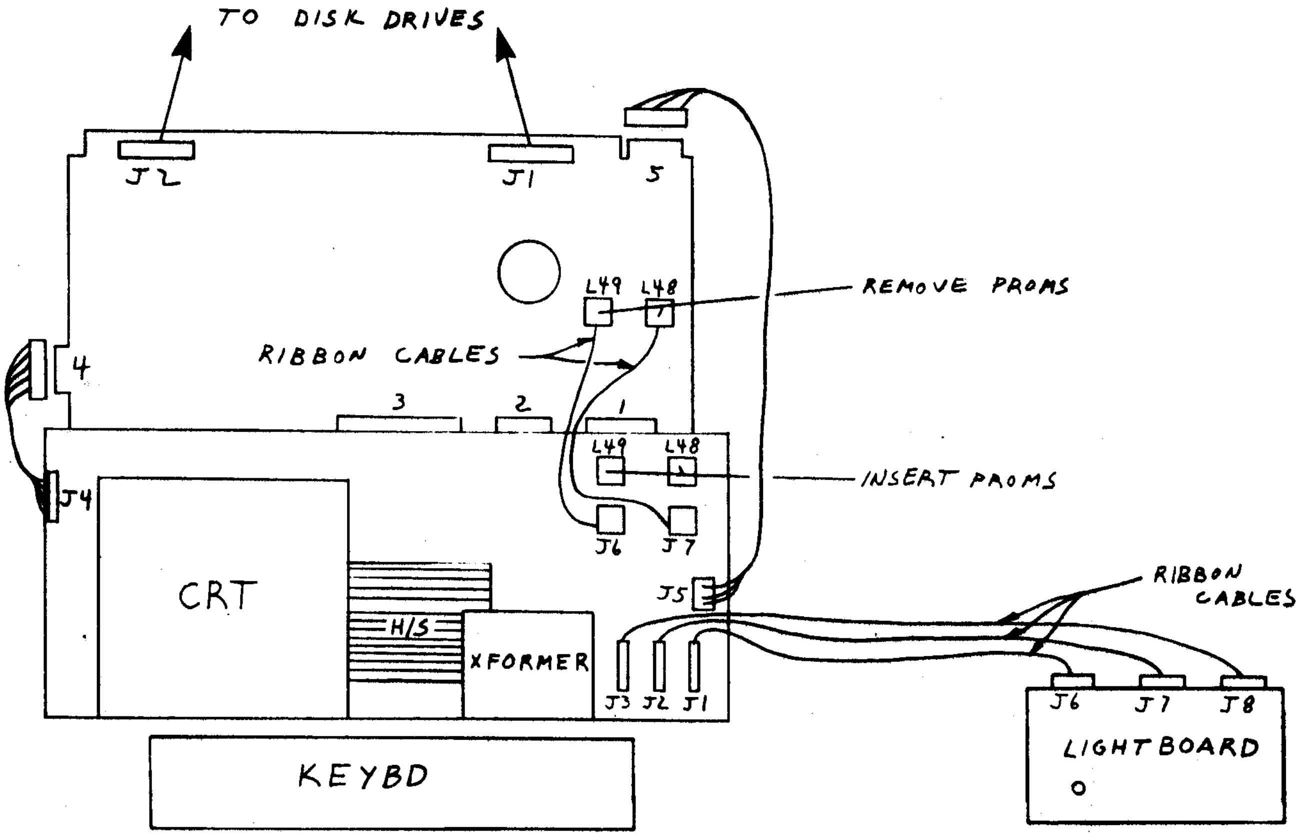
Remove the two PROMS (L48, L49) from the 7180 (7279) and insert them into their appropriate connectors on the extender board L48, L49. Reference schematic of 7176. Plug the two 24 pin ribbon cables into J6 and J7 of the extender. Install the extender in the test unit. Insert 7180 (7279) into the extender with the component side facing the front of the unit (opposite of its normal position) attach J4 and J5 (wired connectors) of the extender to connectors 4 and 5 of the 7180 (7279). Plug the 24 pin ribbon cable that is connected to J6 into the L49 PROM connector on the 7180 and the other ribbon cable J7 into the L48 PROM connector. Attach the lightboard cables J6, J7, J8 (on the 7069) to J1, J2 and J3 (on the 7176) respectively.

210-7176 EXT.	7069 PCB	7180 (7279) PCB
J1	J6	
J2	J7	
J3	J8	
J4		Connector 4

J5	Connector 5
J6	L49
J7	L48

NOTE: 1) Pin 1 of each lightboard cable fingerboard must plug into pin 1 of each female connector on the extender board.

2) The two 24 pin ribbon cables have to be used to allow access to the instruction counter.



2.4.4 OPERATIONAL USE OF TEST UNITS

Each set of light indicators (labeled accordingly) represent current outputs of the following:

1. Designated Registers
   A register (A_7 - A_0 indicators)
   K register (K_7 - K_0 indicators)
   ST_0, ST_1 registers (C_7 - C_0 indicators)

2. ALU Output (C_7 - C_0 indicators)

3. ROM Output (R_{15} - R_0)

4. ROM Address (IC_8 - IC_0)

5. RAM Output Register (M_7 - M_0)

6. RAM Address (AD_8 - AD_0)

Item 3 is actually dual purpose; indicators R_{15} - R_0 can also represent a manually set ROM instruction, to be used in lieu of (supercedes) 7180 ROM outputs. Such manually introduced commands to the microprocessor are set on ROM bit switches S_{15} - S_0 (See pictorials and schematics of 7069). The ROM switch at schematic coordinates H, 14 must be in the UP position to allow indicators R_{15} - R_0 to display manual settings of ROM bit switches S_{15} - S_0. Use of this feature follows in proceeding text.

Item 4, ROM Address indicators IC_9 - IC_0 can be used in conjunction with the Compare Switch to halt the microprogram at any step manually preset on switches SI_9 - SI_0; AUTO/STEP in AUTO mode). Halt occurs when the indicators IC_{9-0} = switch settings. Two other switches control test unit operation: the AUTO/STEP toggle switch and the STEP pushbutton microswitch. With AUTO/STEP in the STEP mode, the microprocessor will complete one cycle each time the STEP pushbutton is depressed. When the AUTO/STEP switch is in the AUTO mode, and the STEP pushbutton is depressed once, the microprocessor begins to cycle continuously.

If a Halt was performed at a desired step by using the COMPARE switch (DOWN = ON), the microprogram will continue if the STEP Pushbutton is depressed. To disable the COMPARE halt function, turn the COMPARE switch OFF (UP). The COMPARE feature is useful for stopping the microprocessor so that key points in the microprogram may be monitored. Monitoring key points in the program sometimes reveals exactly where the microprocessor is failing. Also, some failures occur only during full speed ("on the fly") operation and may not occur during manual stepping of given routine.

Again, note that when manually stepping through the microprogram, the IC may not continue past certain locations. This condition could be normal if the present command is a conditional branch command (e.g. a situation where the microprogram branches on itself until a condition is met).

It may be desirable to do a single command repeatedly, particularly if the command is suspected of intermittent failure. To accomplish this, place AUTO/STEP in the AUTO mode, ROM switch OFF (disable ROM output; enable S_{15} - S_0 manually simulated command), and depress the STEP pushbutton. The manually set ROM command will execute repetitively.

Generally speaking, a good procedure for manual checkout of the microprocessor would be to manipulate data from register to register, using Register commands and Immediate commands. Control Commands verify communication to CPU or Shugart disk drive. Load Auxiliary commands will verify contents and proper addressing of RAM.

Hands-on use is the most valuable tool for developing solid approaches to microprocessor troubleshooting with these test units.

2.4.5 7180/7279

The following items are required to aid in the repair of the 7180/7279 board:

1) The outline of the microcode steps involving a format, write and read (Section 2.2.4).
2) The microprogram (Section 2.2.4).

By using the outline of the microcode steps and the A = B comparator output test point on the 7069 light board, (wire side) the subroutine in which a failure occurs can be located.

TROUBLESHOOTING PROCEDURE

1. Check board visually for shipping or handling damage.
2. Load the board with tested PROMs (if applicable) and RAMs.
3. Check voltages with oscilloscope for noise and proper level.
4. Operate system (attempt a format, write and read) to check for failure.
   a. Flex board lightly while operating system to check for possible opens or shorts.
   b. Observe 2200 for any error codes.
5. Insure potentiometers R43 and R56 are adjusted so that a 4 usec negative pulse is obtained from each one-shot. R43 controls L47 pin 9 and R56 controls L47 pin 7. (Perform a verify on the diskette to adjust these pots).
6. Install lightboard and recheck voltages to insure against intermittent errors due to increased load.
7. Set the light board switches as follows:
   a. S/R switch must be down or in ROM position.
   b. AUTO/STEP switch to AUTO (up).
   c. ON/OFF switch to ON (Compare switch).
   d. SI9-0 switches should be set within the failing routine, as listed in the abbreviated microprogram in Section 2.2.4. Preferably the SI switches should be set between the starting or lowest step and the point of failure.

The board is now ready for troubleshooting. As an example, it will be assumed that a 7180 board is failing during a format command (for this example, the board fails at step 01CD). The basic approach to locating the step that is failing is to use the abbreviated format routine listed in Section 2.2.4.

Set the IC switches at some address in the middle of the format routine, for instance, step 001F. Depress the format button on the disk. When the program reaches step 001F and stops (with the comparator switch on the light board in the ON position), this indicates that all steps up to but not including step 001F are good. Next, set the IC switches on the light board to 0041. Depressing the format switch again causes the disk to begin formatting, however, because step 01CD is faulty (no index pulse), the disk will hang-up and never reach step 0041.

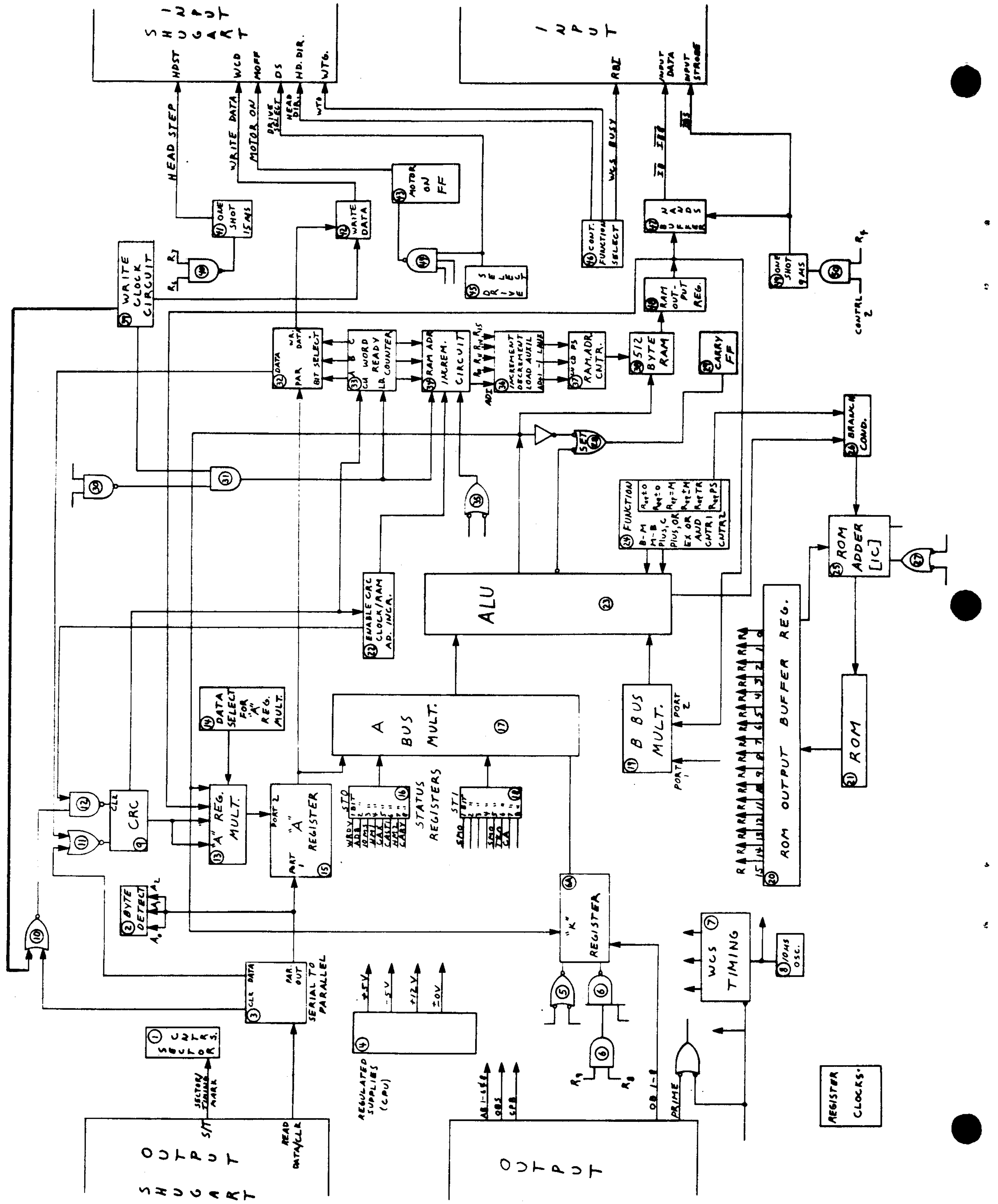
As a result, the problem is known to be somewhere between steps 001F and 0041, and with the same logical approach as above, it will be found that step 01CD is failing.

2.5 MODEL 7180/7279 BLOCK DIAGRAM AND SCHEMATICS

Each block of the diagram is numbered; the integrated circuits which comprise each numbered block are listed below on three pages. The block diagram and the IC listing should be used to further comprehend the "HARDWARE OPERATIONAL THEORY" section of this publication. For the 7279, only the ICs with different numbers are referenced in parenthesis.

BLOCK #	INTEGRATED ICRCUITS UTILIZED
1	L68, L88-10 (L87-10), L88-1 (L87-1), L31-5
2	L56-8, L58-13, L46-15/14
3	L32-10; L32-8, L57-11, L57-5
4	(CPU)
5	L66-8
6	L66-6, L21-13, L5-8, L8-8
6A	L65, L67
7	L11, L13, L25, L23-11, L14-8, L9-6, L23-3
8	L13-8, L13-2, L13-6
9	L27, L28, L29, L30-6, L30-8; L30-11
10	L106-6 (L103-6)
11	L106-8 (L103-8)
12	L56-11
13	L40, L41, L42, L43
14	L44-3, L44-11, L44-6, L44-8, L106-3 (L103-3), L89-11 (L88-11), L57-8, L30-3
15	L26, L55
16	L24, L54, L53, L52, L51
17	L54, L53, L52, L51
18	L24, L54, L53, L52, L51
19	L95, L97, L96
20	L59, L60, L61
21	L49, L48
22	L72-5 (Both Enables)
23	L63, L62

BLOCK #	INTEGRATED ICRCUITS UTILIZED
24	L36, L35, L34-2, L37-2, L37-12, L37-4, L37-10, L34-6, L37-6
25	L94, L74, L91A (L91)
26	L7-8, L4-6, L2-8, L6-8, L20-8, L20-6
27	L20-11
28	L2-6, L34-4
29	L3-51, L4-3, L4-11
30	L56-3
31	L57-3
32	L69-5/6
33	L70
34	L71-12, L71-6, L58-1, L71-8, L90-8 (L89-8), L89-6 (L88-6), L89-3 (L88-3), L90-6/5 (L89-6/5)
35	L56-6
36	L88-4, L23-6, L78-6, L8-6-
37	L75, L76, L77, L78-3, L78-8, L73-6
38	L8-6, L82, L80, L81, L79, L21-1
39	L75-10
40	L7-11
41	L1-12
42	L66-3, L57-11, L32-8
43	L64-6, L64-8
44	L64-6, L64-8
45	L17-6/8
46	L78-11, L93-2/5/12
47	L98-6, L98-8, L98-11, L98-3, L101-8, L101-6, L101-11, L101-3
48	L100, L99-2/5/12/15
49	L105-12 (L102-12), L108-3 (L105-3), L87-6 (L86-6)
50	L9-3


SHUGART INPUT
HDST
WCD
MOFF
DS
HD. DIR.
WTG.
INPUT
RBI
INPUT DATA
INPUT STROBE
HEAD STEP
WRITE DATA
MOTOR ON
DRIVE SELECT
HEAD DIR.
WTG
WCS BUSY
IB IBB
BS
41 ONE SHOT 15 MS
42 WRITE DATA
43 MOTOR ON FF
44
49
45 DRIVE SEL
46 CONT. FUNCTION SELECT
47 BUFFERS AND DRIVERS
48 RAM OUTPUT REG.
59 ONE SHOT 9 MS
60
R4
CONTRL 2
R1
R2
51 WRITE CLOCK CIRCUIT
32 DATA PAR BIT SELECT WR. DATA
33 A B C CLK WORD READY LB COUNTER
34 RAM ADR INCREM. CIRCUIT
R8 R9 R14 R15
ADI
36 INCREMENT DECREMENT LOAD AUXIL. ADI -1 LAUX
CD PS
37 RAM ADR CNTR.
38 512 BYTE RAM
29 CARRY FF
26 BRANCH COND.
28 SET
30
31
35
24 FUNCTION
B-M
M-B
PLUS, C
PLUS, OR
EX OR
AND
CNTR 1
CNTR 2
Rref 0
Rref 0
Rref = M
Rref = M
Rref TR
Rref PS
23 ALU
25 ROM ADDER [IC]
27
21 ROM
20 ROM OUTPUT BUFFER REG.
RA 15 RA 14 RA 13 RA 12 RA 11 RA 10 RA 9 RA 8 RA 7 RA 6 RA 5 RA 4 RA 3 RA 2 RA 1 RA 0
19 B BUS MULT.
PORT 1
PORT 2
17 A BUS MULT.
16 ST0
18 ST1
BIT
WRDY
ADR
10MS
HM
CAR
CAST
HML
CARY
SMO
SMO
TZO
CA
STATUS REGISTERS
15 "A" REGISTER
PORT 1
PORT 2
14 DATA SELECT FOR "A" REG. MULT.
13 "A" REG. MULT.
9 CRC
CLR
10
11
12
22 ENABLE CRC CLOCK/RAM AD. INCR.
2 BYTE DETECT
A0
A1
3 CLK DATA PAR OUT
SERIAL TO PARALLEL
1 SECTOR COUNTERS
SECTOR TIMING MARK
4
+5V
-5V
+12V
±0V
REGULATED SUPPLIES (CPU)
64 "K" REGISTER
5
6
6
R9
R8
7 WCS TIMING
8 10MHS OSC.
AB 1-16
OBS
SPB
OB 1-8
PRIME
OUTPUT
REGISTER CLOCKS
S/T
READ DATA/CLK
OUTPUT SHUGART

2.6 7180/7279 SIGNAL MNEMONICS

A0-A7 : Output of the A register 8 bit data path for R/W data.

AB 1-6, 8 : Address Bus for disk selection.
AB 6,8 for 7279

ABS (7180 only) : Address Bus strobe.

ACU : A register clock.

AD_0-AD_8 : Output of RAM address counter.

ADI : RAM address increment.

AND : Logical and instruction decoded.

BR : Unconditional Branch.

BR_0, Br_1, Br_3 : Directly control the ROM address for sequential addressing as well as addressing for Branch command.

B to M : Select register to memory instruction decoded.

C_0 - C_7 : Output from ALU 8 bit data path known as C Bus. Carries data to selected registers as indicated by micro-instruction.

CA : Carry F/F output (used as status bit)

CABYS : Calculator busy.

CAPM : Calculator prime.

CASTIS : Calculator strobe to disk microprocessor.

CAX : Made up from terms AB_6 and AB_8. They distinguish the select address from a data transfer operation.

CKC : Clear 7180 clock.

CLKG : Clock input from 7180 lightboard (if used)

CNTRL1,2 : Control command generated by microprogram to access peripheral functions.

CPB : Calculator ready/busy to microprocessor.

$DK_{1,2}$: Disk select lines from microprocessor.

EXC OR : Logical Exclusive OR instruction decoded.

FH : Indicates adder output is not equal.

FMT : Format pushbutton.

HDIR : Select R/W head direction; in or out.

HDST : Head step.

$HM_{1,2}$: Turns disk drive motor on.

IB_1 - IB_8 : Input bus to CPU.

IBS : Input bus strobe.

IC_0 - IC_9 : ROM address bits.

IM_1 : Index/sector from selected disk drive.

K_0 - K_7 : The K register output. High order and low order bits from CPU or ALU microprocessor.

KR : K register clock.

KS_0 - KS_7 : The K register inputs. High order and low order bits of an 8 bit word received from the CPU.

LAUX : Load auxiliary instruction decoded.

M : Controls the MODE CONTROL INPUT on the ALU for determining if the ALU will perform a logic function or an arithmetic function.

M_0 - M_7 : Memory output.

M to B : Memory to selected register instruction decoded.

NOOP : No operation 1.6 microsecond delay.

OB_1 - OB_8 : Output bus from CPU.

OBS : Output bus strobe.

OR : Logical OR instruction decoded.

OSC : 10 megahertz oscillator.

PF : Prime/format.

PLUS NC : Add without carry instruction decoded.

PLUS WC : Add with carry instruction decoded.

PRC : Clears CRC

PRMS : Prime signal.

R_0 - R_{15} : 16 bit ROM output. Makes up microinstruction for microprocessor.

RB (7180 only) : Microprocessor ready/busy to CPU.

RCD : Read clock and data from disk.

RDG : Read gate. Selects either C Bus or A register data as input to memory.

REG = 0 : Branch if selected register = 0

REG ≠ 0 : Branch if selected register = 0

REG = M : Branch if selected register = ROM mask.

REG ≠ M : Branch if selected register = ROM mask.

REG TR : Branch if true bits in selected register match true bits in mask.

REG FS : Same (False instead of True)

$RM_0 - RM_7$: Originates from a multiplexed selection of ROM bits or RAM bits, the B bus to the ALU.

RS : ROM/Switches from light board (if used).

R/W : Input to RAM, low for write and high for read mode.

$S_0 - S_3$: Select lines to ALU for determining the output.

SM : Sector pulses from disk.

SMO : Index pulse from disk.

ST1 : Clock for Carry F/F; results in setting of Carry F/F if TK 0 (bit 6, C_6) is active.

T_{01} : RAM/ROM buffer registers clock time.

T_{37} : RAM R/W time.

T_4 : ROM address increment/branch time.

T_{78} : RAM address increment/decrement time; also clocks certain status bits.

TK_0 : Track zero sensing indicator for selected disk drive.

WCD : Write clock and data.

WCLK : Clock for write data.

$WCLK_1$: Clock 1 (for parallel to serial conversion during a write).

WP : Write protect sensing indicator from disk drive.

WRDY : Indicates 8 bits are ready /or a R/W.

WTG : Write gate.

40MS : 40 millisecond delay /or disk access.