2/1981

USERS MANUAL
Dual Density Controller
Model 1170

3/18/80

FILE MANAGEMENT FIRMWARE IMPROVEMENTS FOR THE MODEL 1170 CONTROLLER

Effective March, 1980, the firmware installed in the Model 1170 Controller has five new TEST commands and several protocol changes. Firmware EPROMS having these new commands are marked 1A, 3A, or 7A to differentiate them from the prior firmware EPROMS marked V1, V3, or V7. (All EPROMS also bear a socket indication designation.) The new TEST commands are:

TS/disk N: This command will cause the drive R/W heads to be positioned to Track N, where N is a decimal number on the range zero to 76. It is not necessary for a diskette to be loaded unless the drive is a PerSci Model 70.

TC/disk: This command is an enhancement of the TI command. It also will cause every address and data field of every track to be read and verified, but rather than terminating the test at the last sector of Track 76, it will cycle the test back to Track zero, repeating until terminated by diskette ejection, controller reset, or a hard error condition. Verification initiated by the TC command will start at the track-sector following that last addressed, unless the heads have unloaded (i.e., the controller time delay has expired) or the disk selection is changed.

TP: This command initiates a hash check of the controller EPROM code. The controller will reply ACK EOT if the test is successful or NAK XXXX EOT if not where XXXX is the hash check (hex) found.

TM 7800 Either of these two commands initiates a write/read test on a 1K block
TM 7900 of the controller 2K RAM. The controller will reply ACK EOT if the test is successful and NAK YYYY EOT if not where YYYY is the address (hex) of the first location failing the test.

RAM Chips	Address Space
U7/22	7800H → 78FFH
U8/23	7900H → 79FFH

CHANGE TO POWER UP/RESET PROTOCOL

Immediately following power up or a reset the controller will test the status of the serial port. If the optional components are installed and the DSR line (J2 Pin 11) is high, the controller will operate in the serial mode only; otherwise, the controller will default to parallel I/O operation only. NOTE THAT THIS IS A SIGNIFICANT CHANGE TO THE PREVIOUS PROTOCOL DESCRIBED IN SECTION 5.4 OF THE MANUAL. The sign on message will be sent to the port selected. Two wire (TXD AND RXD) operation with serial ASCII terminals is still feasible if Pin 11 of U15 is jumpered to ground to cause DSR to appear always true. (Remove U43 to operate parallel.)

CHANGE TO COPY COMMAND PROTOCOL

Prior firmware caused all of the allocated space of File Sets to be read and re-written in response to a COPY command even if no data was present. The new firmware limits this process to actual data only, (File Allocation space remains the same) offering a speed improvement in some cases.

PERSCI, INC.

MODEL 1170 DUAL-DENSITY CONTROLLER
USER'S MANUAL

June 1979

PREFACE

This manual is intended to provide the user of the Model 1170 Dual Density Controller with sufficient information for system integration of the controller and operation with any of the possible diskette drive combinations.

Key to the controller operation is the File Management Firmware (FMF) which this manual describes in detail. It is the policy of PerSci not to distribute program source listings or other details of this firmware, but PerSci will make corrections for flaws found by users and will issue updated versions as required.

Any new versions of the FMF will be made available to users who may:

- Purchase a new set of EPROMS
- Send in existing EPROMS for reprogramming for a fee
- Send in a diskette which will be returned with the Firware Object Code.

PerSci will also consider firmware modification or expansion to provide functional enhancements for the general user where feasible, but radical firmware changes and/or special disk-resident software for use with the controller "XECUTE" command require special charges.

PERSCI CONTROLLER SELECTION GUIDE

Following is a summary of pertinent questions for use in selection of a PerSci 1070 or 1170 Controller.

Question: Q1 - Is double density data storage desired?
(i.e., 505,856 data bytes 1 side; 1,011,712 data bytes/diskette)

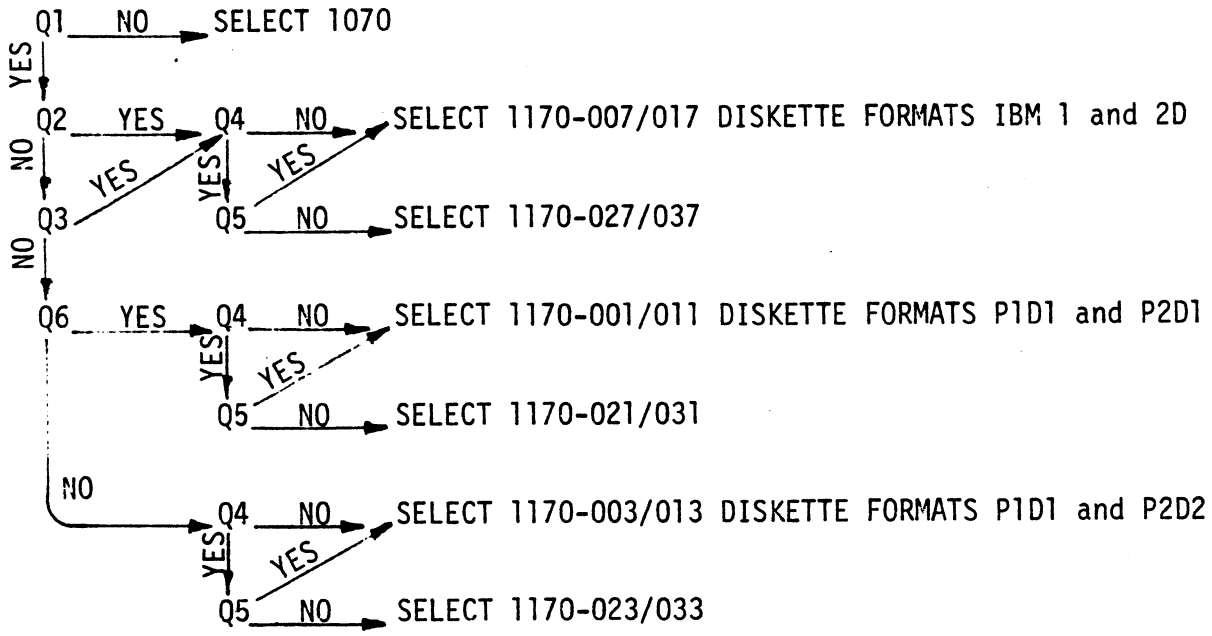
Q2 - Is IBM diskette 2D compatibility needed?

Q3 - Is dual density capability required?
(i.e., format, read, and write single (FM) or double (MFM) density)

Q4 - Will controller drive only Model 299B drives?

Q5 - Must system diskettes be interchangeable with diskettes from other controller and/or drive types?

Q6 - Are 128 data bytes/sectors required for host computer software?



NOTE: Actual double density only formats used by DASH 0X1, and 0X3 versions are identical but controller firmware used on -0X1 makes 256 byte sectors appear to host as 2 separate 128 byte sectors.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

APPENDICES

1.0 GENERAL

1.1 Introduction

The Model 1170 is the second intelligent controller developed by PerSci and can operate in either single density (FM) or double density (MFM) modes. It continues the concept pioneered by the Model 1070 of providing an advanced disk operating system resident in the controller firmware, resulting in a minimum software burden to the host computer.

The controller Z80A-CPU, 2 to 9K bytes of RAM, 8K bytes of EPROMs, and other support circuitry are mounted on two S100 printed circuit boards. Connections and space are provided for an optional RS232 serial asynchronous interface along with an eight-bit parallel interface which is designed to exchange data via the S100 bus I/O ports.

All signal and data pins are compatible with the S100 bus as are the connections for +8 volts and ±16 volts DC power.

The controller program resides in ROM on the controller board and performs the file management functions normally associated with a microcomputer disk operating system. These include: diskette format initialization; maintaining and searching an index of files on each diskette; allocation and de-allocation of diskette space; blocking and unblocking of both fixed and variable length records; error detection and retry; creating, deleting, renaming, copying of files; and even performing diagnostic testing of the diskette drives. These file management functions are specified by means of a high-level controller command language which requires only minimal unique routines in the host computer software (for example, 168 bytes in a typical 8080-based minicomputer). This program is so interdependent with the logic design of the controller that it is best referred to as "firmware," and the entire set of programs written for PerSci controllers will be referred to as the File Management Firmware. Firmware commands for the 1170 are an extension of the set used by the PerSci Model 1070 controller.

The 1170 is initially available with three versions of File Management Firmware, the standard (F2-7) version offering operation with the IBM 3740 (single density) or Diskette 2D (double density) formats. Two alternate firmware versions are designed for double density only operation. All three will work with PerSci drives in the diskette select mode supporting four Model 70, two Model 277, or two Model 299 drives.

Future firmware versions will be available which take advantage of the address decoding ability of the Model 299 drives so that

the 1170 can operate up to 8 drives (32 diskette sides) with a capabity of over 16 Megabytes of storage. All three work with the PerSci Model 299 to automatically adjust for insertion of single- or double-sided diskettes.

The development of the microprocessor has created new uses for, and demands on, the floppy disk drives invented early in this decade. Rapid design evolution has answered many of the demands, but with little, if any, widely accepted drive or media standards. Therefore, the following paragraphs present a brief summary of key aspects of diskette technology and usage.

The basic floppy disk drive is comprised of a motor rotating an 8-inch diameter diskette kept in its protective envelope at 360 RPM with a single gap head used for both writing and reading. Two separate gaps offset to either side of the read/write gap are used to erase or "trim" the head track when writing. The diskettes are interchangeable between drives and when loaded into the drive, the plane of the diskette is penetrated slightly by the read/write head which has a pad opposite the head to ensure the media remains in contact. This pad can be lifted usually to "unload" the head even if the diskette remains in the drive. The read/write head is stepped incrementally to one of 77 tracks, usually by a stepper motor subject to mechanical wear, and offering no "feedback" sensor to indicate track location. Just enough logic circuitry is provided to control the stepper motor and the read/write head.signals.

The PerSci Model 70, 270, 272, and 277 drive designs have improved this basic and useful concept by:

a. Making the read/write Head Positioner into a true servo system powered by a linear DC motor which operates with a track sensor to provide fast, accurate placement of the read/write head.

b. Providing an automatic diskette loading system which does not deform diskettes.

c. Designing an intergrated and efficient package that can accommodate two diskettes in the same volume previous units required to handle one diskette.

d. Providing advanced electronics for the detection of data.

The PerSci Model 299A/B and 288 designs have maintained these features, while adding the capability of working with two inserted two-sided diskettes doubling the storage surface available.

However, even PerSci designs require a controller serving to interconnect one or more drives to a host computer.

1.2 Dumb versus Intelligent Controllers

The minimum capability for a "dumb" controller serving one or more drives requires that the design:

a. Control the read/write head positioner generating step pulses at the required rate along with a direction signal.

b. Accept data bytes from the host and convert these to serial encoded timed pulses which when sent to the drive cause the write head current to reverse.

c. Accept encoded pulses from the drive and convert these to data bytes for transfer to the host.

d. Synchronize these data transfers of the preceding three requirements between the host and the diskette timing.

These requirements are satisfied by several LSI chips now available such as the Western Digital FD1771/91 Series, but still needed are rather complex real-time programs for the host which consume considerable CPU time just "waiting" for data transfer synchronization.

The 1170 Controller eliminates this requirement, allowing the host to perform major diskette functions such as initialization, diskette copying, and self tests, such as verification by issue of a single command, leaving the host then free for other uses. The addition of the optional serial option enables the 1170 to be used alone with a CRT display, keyboard, or teletype, as a file storage and retrieval unit.

1.3 The Media

The floppy diskette is a flexible, oxide-coated 7.88-inch diameter plastic disk kept in an 8-inch square protective envelope or cover. The diskette (and cover) have a hole for the drive spindle hub, and an aperture for the read/write heads to run in contact with the diskette. A small set of holes are offset in the envelope so that an index hole in the diskette may be detected.

Diskettes having one index hole are intended for use as "soft" sectored formats such as used with the 1170. Other diskettes have more than one index hole and are <u>not</u> usable with the 1170. The exact location of the diskette cover index hole is used as a means of indicating if the diskette is intended for single- or two-sided use. The result is that the PerSci Model 299 drives can work with single- or two-sided diskettes, but the Model 70 and 277 drives can accept single-sided ones only.

Diskettes are available from a number of vendors, pretested in a variety of formats.

1.4 Recording Scheme

A read/write head is writing to the diskette when it is being driven to magnetic saturation. Receipt of a "WRITE" pulse will then cause the head to reverse saturation. These flux reversals or transitions are then detected by the same head (with write drive current removed) and a read amplifier to generate pulses which are returned to the controller as read data. (The read amplifier output may or may not be processed by a data separator between amplifier output and controller. The 1170 requires a separate data separator for single density, but not for double density.)

1.5 Single/Double-Density Codes

Serial data is exchanged between controller and drives as pulses. Pulses from the drive to controller are generated by a read amplifier detecting reversals of magnetic field in the diskette surface, while pulses from controller to drive are the output of the controller encoder conversion of NRZ data. If the drive and diskette media were ideal, then the drive output data pulses would have the same timing and frequency of the controller output. They are not, so the controller must encode data subject to the limitations of the media and diskette drive.

The 1070/1170 single-density code is double frequency (or FM) where a magnetic flux transition occurs for every clock and every data bit, which results in packing densities very near the media limit at the inner tracks. Unfortunately, this code is only 50% efficient because one-half the available space is used for clocks required for synchronization during reading.

MFM code is a more efficient code which packs twice as many bytes per track inch by recording clocks only when data is not present. (See Figure 1.1.) The resultant diskette media has no more flux-transitions/inch than the single density one, but the read amplifier/controller combination must detect data subject to the same order of media distortion as single density with one-half the critical tolerances. Two approaches to the problem are possible — greatly improve the Drive Amplifier, which has been done with the PerSci 299B and 288, and/or Write Precompensation in the controller. The latter approach involves a systematic predistortion of the encoded pulses to the drive, so that peak shift distortion is compensated.

The 1170 Controller is capable of encoding data for recording with or without write compensation dependent upon the Encoded EPROM used. Thus, it may be used with the 299B either way as required for interchangeability with other system diskette drives. However, the write precompensation encoder should be used with the PerSci 70, 270, 272, 277, or 299A drives.

1.6 Soft-Sectored Formats

Most drives now available are designed to use the floppy diskette as a single-sided media having 77 tracks. With the advent to two-sided media and drives, diskettes are now often considered a media having 77 cylinders to point up the increased capacity without having to move the read/write heads. Each track has a continuous or unformatted capacity in excess of 5,200 bytes (FM) or 10,400 (MFM) if the diskette index hole is used as a start reference for data storage. For practical applications, the tracks are divided into arc sections (or sectors) which are used to store and retrieve fixed length blocks of data. Diskettes are available with additional (usually 32) holes around the drive spindle hole to establish sectors. These are known as hard-sectored diskettes. Soft-sectored diskettes use special address fields written during an initialization to divide the track into sectors. In addition, most formats specify a special index sector which is not used for data storage. A diskette (soft sectored) format then may be specified by:

- The number of tracks or cylinders.
- Number of data and special sectors per track.
- The recording code used.
- Definition of special bytes used for sector identification and error checks.

Figure 1.2 is a summary of the 1170 Controller diskette formats in these terms. The data sectors are divided into two subsectors: The Address Field and the Data Field. Each of these fields has a unique "Mark" which serves to enable the read logic to achieve byte synchronization when reading. The Data Field is completely rewritten any time a sector is written to, but the Address Field is only written during the initialization.

1.7 Diskette Initialization and Use

Before the 1170 can use a diskette, it must be initialized into an acceptable format. Many vendors provide such diskettes to which the 1170 can read or write data using its INPUT, OUTPUT, and TEST commands. To make full use of the File Management Firmware, however, a volume name and other parameters must be present in the data field of the first sector of Track Zero (i.e., Outer Track) which are written by the FMF KILL command.

1.8 IBM Compatibility

The PerSci 1070 and 1170 with FMF2-7 will generate on single-sided diskettes formats meeting all requirements of IBM Product Reference Literature GA21-9190-1, with the exception of data field

contents. Address fields, and the Data Mark at the start of data fields are identical and the CRC bytes added at the end of the field are compatible. It is this format that is generally referred to as the IBM 3740 type. Diskettes of this type generated on IBM machines have been read on PerSci drives using the 1070 and the 1170 INPUT, OUTPUT, and TEST commands.

The 1170 with FMF2-7 will generate on two-sided diskettes formats meeting all requirements of IBM Product Reference Literature GA21-9257-1, with the exception of data field contents. Address fields and the Data Marks at the start of the data fields are identical and the CRC bytes added are compatible. It is this format that is known as the 2D (IBM Part No. 1766872). Diskettes of this type procured from IBM have been read on PerSci 299 drives using the 1170 INPUT, OUTPUT, and TEST commands.

NOTE

IBM Diskette 2D (P/N 1766872) are best read on the PerSci 299B which has a read amplifier which can be set for MFM encoded diskettes not using write precompensation.

NOTE

IBM Diskette 2D (P/N 1669045) diskettes with 1024 byte sectors are not compatible with FMF2-7.

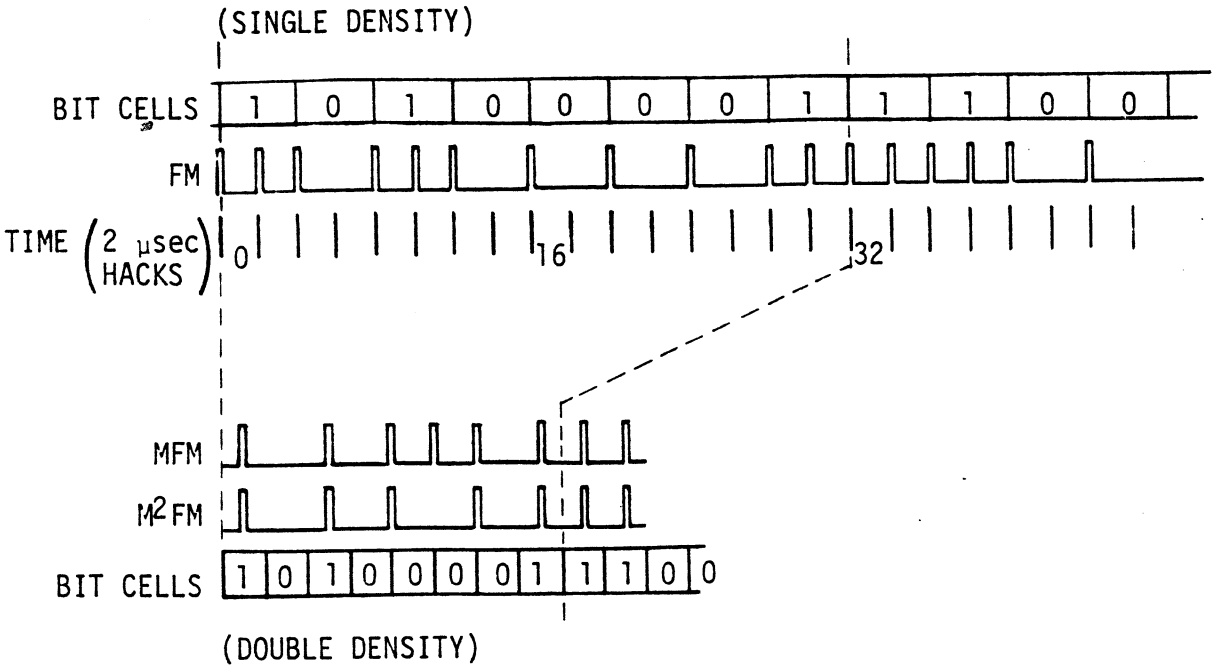


Figure 1.1 FM, and MFM Encoding Compared

FM Code

FM Code has a pulse for every clock and data bit which at a data rate of 250K bits/secs results in flux transition densities ranging from 3250 ppi at the outer track to 6500 ppi at the inner track.

MFM Code

MFM Code is a more efficient code having one pulse per data bit and clock pulses only between adjacent data zeros.

DISKETTE FORMATS				TRACK/CYLINDERS SECTOR TYPE				
DISKETTE NAME				CYLINDER ZERO		CYLINDER 1→76		GENERATING CONTROLLERS
PERSCI	IBM	IBM P/N	SIDES	SIDE 0	SIDE 1	SIDE 0	SIDE 1	
P1	1	2305830	1	FM	--	FM	--	1070/1170-007
P1D1	-	--	1	MFM	--	MFM	--	1170-0X1, 0X3
P1D2	-	--	1	FM	--	MFM	--	1170-0X7
P2S0	-	--	2	FM	FM	FM	FM	1070 ONLY
P2S1	-	--	2	FM	FM	FM	FM	1170-0X7
P2D1	-	--	2	MFM	MFM	MFM	MFM	1170-0X1, 0X3
P2D2	2D	1766872	2	FM	MFM	MFM	MFM	1170-0X7 X = 0→6

DISKETTE DESCRIPTION

1. Each diskette has 77 tracks per side with twenty-six data sectors, an index sector, and pre-index gap as shown.

2. All sectors of a track use the same code (i.e., FM or MFM).

3. All data tracks/cylinders (i.e., 1-76) of a diskette use sectors of the same code.

4. Pre-index gap, index, address fields, and post data bytes are written only when initializing diskette using "KILL" command. Byte value shown for data is written during initialization.

Figure 1.2 PERSCI DISKETTE FORMATS

SECTOR DETAILS

ADDRESS FIELD CONTENTS

FM (SINGLE DENSITY) BYTES

NO.	HEX VALUE	
6	00	PREAMBLE
1	FE	ADDRESS MARK*
1	0 → 4C	TRACK NUMBER
1	0 → 01	SIDE NUMBER
1	01 → 1A	SECTOR NUMBER
1	00	
2	XX	CRC BYTES
11	FF	POST ADDRESS

MFM (DOUBLE DENSITY) BYTES

NO.	HEX VALUE	
12	00	PREAMBLE
3	A1	ADDRESS MARKS*
1	FE	
1	0 → 4C	CYLINDER NUMBER
1	0 → 01	SIDE NUMBER
1	01 → 1A	SECTOR NUMBER
1	01	
2	XX	CRC BYTES
22	4E	POST ADDRESS

DATA FIELD CONTENTS

FM (SINGLE DENSITY) BYTES

NO.	HEX VALUE	
6	00	PREAMBLE
1	FB	DATA MARK*
128	E5	DATA
2	XX	CRC BYTES
1	FF	
26	FF	POST DATA

MFM (DOUBLE DENSITY) BYTES

NO.	HEX VALUE	
12	00	PREAMBLE
3	A1	DATA MARKS*
1	FB(1)	
256	40	DATA
2	XX	CRC BYTES
1	4E	
53	4E	POST DATA

(1) THIS BYTE IS F8 FOR SIDE ONE, CYLINDER ZERO

INDEX SECTOR CONTENTS

FM (SINGLE DENSITY) BYTES

NO.	HEX VALUE	
40	FF	
6	00	
1	FC	INDEX MARK*
26	FF	

MFM (DOUBLE DENSITY) BYTES

NO.	HEX VALUE	
80	4E	
12	00	
3	C2	INDEX MARKS*
1	FC	
50	4E	

PRE INDEX GAP

FM (SINGLE DENSITY) BYTES

NO.	HEX VALUE
247	FF

MFM (DOUBLE DENSITY) BYTES

NO.	HEX VALUE
598	4E

*INDEX, ADDRESS, AND DATA MARK CODES HAVE ONE OR MORE CLOCK PULSES DELETED

Figure 1.2 PERSCI DISKETTE FORMATS (Continued)

2.0 THE 1170 CONTROLLER

2.1

The 1170 Controller is basically a dedicated microprocessor operating with its own RAM/EPROM via address and data lines not accessible to its host. Communication with the host by means of one of two ports (parallel or optional serial) on the Processor Logic PCB and with the diskette drives by means of special logic on the Disk Port PCB. (See block diagram of Figure 2.1.)

The Processor Logic PCB has a Z80A-CPU with provision for up to 8K bytes of EPROM (2716) and up to 9K bytes of RAM (2114). This PCB also carries the two interface ports and a crystal oscillator (8 MHz) with Frequency Divider which provides the basic 4 MHz clock used with the CPU and the clocks used by the Encoder Logic in writing to the diskette drives.

This board was designed to accommodate three types of EPROMS (2708, Single Voltage 2716, and Three Voltage 2716), but the single power (INTEL) type 2716 has been adopted as standard.

The Disk Port PCB has the special logic for selecting a diskette drive, controlling the position of the read/write head, and encoding or decoding diskette serial data pulses in either single (FM) or double (MFM) density modes. A socket is provided on this board for the 2708 in the Encoder which controls write encoding and precompensation.

2.2 Interfaces

Two alternative interfaces are provided by the controller design. The first is a parallel 8-bit wide port with the necessary hardware components included as standard while the second is an optional serial-by-bit operating at one of sixteen switch-selectable baud rates.

2.2.1 S100 Parallel Ports (8-Bit Bytes)

The parallel port is designed to meet the requirements for data transfer using 8080 or Z80 type microprocessor IN and OUT instructions. A status/control port is provided so that the S100 host microprocessor program can test or "poll" the controller to ensure it is ready for data transfer via the data port and to enable selected bytes to be used as control rather than data.

The data port uses a Z80A-PIO which interrupts the controller CPU whenever a data exchange occurs. It is this feature of the design that requires that no host IN or OUT instruction should be made to the data port unless the status port is ready. (See Section 5.0.)

2.2.2 Serial (RS232C) Interface

The serial interface is an option identical to that offered by the 1070 controller. It is installed when the necessary baud rate oscillator and other logic are installed on the PCB and if the proper firmware is used. It offers the ability to use the 1170 with almost any CRT terminal as a "stand-alone" device. It is designed primarily for communication using the ASCII character set as eight bits (no parity) and one stop bit for all transmission speeds.

2.3 Double-Density Write Precompensation

Model 70, 277, and 299A Series Floppy Disk Drives perform best with double-density recording using a technique called "write precompensation." This is an empirically derived procedure of distorting the signal at the time of writing so that the signal read back is corrected for inherent media distortion. The newer read amplifiers, such as that of the PerSci 299B, however, perform better without this "write precompensation." The 1170 Controller may be converted to either mode of operation by insertion of the proper EPROM at U27 of the Data Port Board.

2.4 Firmware Options

The 1170 firmware has three functions. These are:

- Host/Controller Data Exchange

  The firmware must coordinate data transfer between controller and the host via one of two ports. The firmware code for the serial port is quite similar to that for the 1070, but the code for the parallel data port makes full use of the Z80A-PIO interrupt facilities.

- Controller/Diskette Drive Data Exchange

  Read and write data exchange between controller ram and diskette is quite firmware dependent. It is this portion of the firmware that controls diskette format.

- File Management Function

  This part of the firmware offers an expansion of the Command Set first used in the PerSci 1070 extended to new drives and more efficient diskette formats. Changes in one or more of these functions results in changed controller operation. The parameters of the first firmware (Series FMF2.X) offered are defined in the next section.

2.5 Versions of the 1170 Controller

As the previous sections have indicated, the first offering of the 1170 Controller has an optional serial port, data encoding with or without write precompensation, and three initial versions (with more to follow) of File Management Firmware. These variations alone result in 12 different operational versions of the controllers. Alternate versions of firmware and/or RAM capacity are contemplated which add to the permutations. PerSci differentiates between these variations of the controller by assigning each a dash number. Figure 2-0 is a summary of the 1170 dash numbers and should be used in inquiries to PerSci.

2.6 Operation

The 1170 operates in one of two modes: Diskette Operations or Data I/O Operations. In the first mode, the controller CPU is fully occupied with diskette read/write operations and will turn off the host data port. In the second mode, the data port is enabled. This means that the controller will exchange data with the host up to one sector in length and will then wait for a period before resuming the exchange. The waiting period is primarily a function of the diskette access time (i.e., time required for the diskette sector sought to come to the R/W head), which can be as high as 166 millisec (at 360 RPM) with an average of 83 millisec. This access time is the limiting restriction on overall data rates and can be reduced by a careful choice of interleave sequences for the diskette format. (See Appendix B.)

2.7 Data Error Detection and Control

The 1170 provides several important safeguards for data storage and retrieval. These include the following.

2.7.1 Write Protect

The 1170 will not permit data writes to write-protected diskettes (i.e., notched) in drives having the optional detector.

2.7.2 CRC Checks

When formatting or writing data fields to diskettes, the 1170 appends two CRC (Cyclic Redundancy Check) characters as the last two bytes. These two bytes are then used in any read operation (address or data field) as a check that all bits were read correctly. The polynomial used is:

$$G(X) = X^{16} + X^{12} + X^5 + 1 \quad .$$

The polynomial and method are identical to that used for the IBM 3740 and 2D diskettes.

2.7.3 Verify Checks

All read operations of address or data fields are also checked to see that the proper bytes of the address and/or data marks were found. (For example, the first four of the 264 bytes read back in a MFM data field must be: A1, A1, A1, FB.)

2.7.4 Data Lost Check

Any read or write operation starts with a search for the proper address field. If no address or data mark is found for one complete revolution of the diskette, this is taken as an indication that all data for the track is lost. A "lost" return usually means that the diskette will have to be reformatted and/or the read circuitry of the controller has failed.

2.8 Future Developments

The Model 1170 with Series FMF2 firmware is designed to operate with a 2K RAM which provides the required memory for five open files and other data. The Processor Logic PCB is designed to accommodate up to eighteen 2114 RAM chips resulting in a RAM capacity of 9216 bytes. This would enable the complete reading of 26 sectors of one track (6656 bytes) in one diskette revolution with a consequent increase in data average storage and retrieval rates. The additional RAM could also be used to increase the number of open files. These and other features will be included in the Series FMF3 firmware to be issued.

CONTROLLER 200640-	ASSEMBLY PCB MATERIAL LISTS		CONTROLLER CONFIGURATION					COMPATIBLE DRIVES					REMARKS
			FMF 2.V					MODEL				DATA (3) SEPARATOR REQ	
	200 644-	200 647-	V	EPROMS NO.	EPROMS TYPE	RS 232 OPTION?	EN-CODER	70	277	299A	299B		
-000	-000	-001	-	0	-	NO	E II						MINIMUM CONFIGURATION (NO EPROMS)
001	-004		1	3	INTEL				044	013	25002/16002	NO	DOUBLE DENSITY ONLY - 128 BYTE SECTORS
002			-	3	2716				044	013	25002/16002	NO	BLANK (UNPROGRAMMED) EPROMS
003			3	3					044	013	25002/16002	NO	DOUBLE DENSITY ONLY - 256 BYTE SECTOR
004			4	3				NU	NU	NU	NA	NO	SAME AS 001 DRIVING 1 TO 8 299 DRIVES
005			-	4							NA	YES	BLANK EPROMS
006			6	3				NU	NU	NU	NA	NO	SAME AS 003 DRIVING 1 TO 8 299 DRIVES
007			7	4				017	008/042/013	001/014	25102/16102	YES	STANDARD FOR IBM 2D DISKETTES OR S.D.
008			8	4				NU	NU	NU	NA	YES	SAME AS 007 DRIVING 1 TO 8 299 DRIVES
-009	-004	-001				NO							
-010	-003	-001	-	0	-	YES							MINIMUM CONFIGURATION WITH SERIAL I/O
011			1S	3	INTEL				044	013	25002/16002	NO	-01X UNITS CORRESPOND TO -00X UNITS
012			-	3	2716						25002/16002	NO	WITH ADDITION OF RS 232 OPTION
013			3S	3					044	013	25002/16002	NO	
014			4S	3				NU	NU	NU	NA	NO	
015			-	4						—	NA	YES	
016			6S	3				NU	NU	NU	NA	NO	
017			7S	4					002/013/042	001/014	25102/16102	YES	
018			8S	4				NU	NU	NU	NA	YES	
-019	-003	-001				YES	E II						
-020	-000	-002	-	0		NO	EZ	NOT USEABLE			25001/16001	NO	-02X UNITS CORRESPOND TO -00X UNITS
021	-004		1	3							25001/16001	NO	WITH EXCEPTION OF ENCODER
022			-	3							25001/16001	NO	EZ ENCODER DOES NOT
023			3	3							NA	NO	WRITE PRECOMPENSATE
024			4	3							NA	YES	
025			-	4							NA	NO	
026			6	3							25101/16101	YES	
027			7	4							NA	YES	
028			8	4									
-029	-004	-002				NO							
-030	-003	-002	-	0		YES					25001/16001	NO	-03X UNITS CORRESPOND TO -02X UNITS
031			1S	3							25001/16001	NO	WITH ADDITION OF RS 232 OPTION
032			-	3							25001/16001	NO	
033			3S	3							NA	NO	
034			4S	3							NA	YES	
035			-	4							NA	NO	
036			6S	3							25101/16101	YES	
037			7S	4									
038			8S	4									
-039	-003	-002				YES	EZ	NOT USEABLE					

(1.) CONTROLLERS WITH FMF 2.1, 2.3, AND 2.7 REQUIRE DRIVES WITH DISK SELECT DRIVE ADDRESSING

(2.) FMF 2.4, 2.6, AND 2.8 REQUIRE 299 DRIVES USING BINARY DECODING OF ADDRESS LINES (E.G. 299B OPTIONS)

(3.) DATA SEPARATOR IS SINGLE DENSITY DOUBLE F (ASSEMBLY 200591)

NA = NO DASH NUMBER ASSIGNED

NU = NOT USEABLE

Figure 2-0. 1170 Controller Configuration

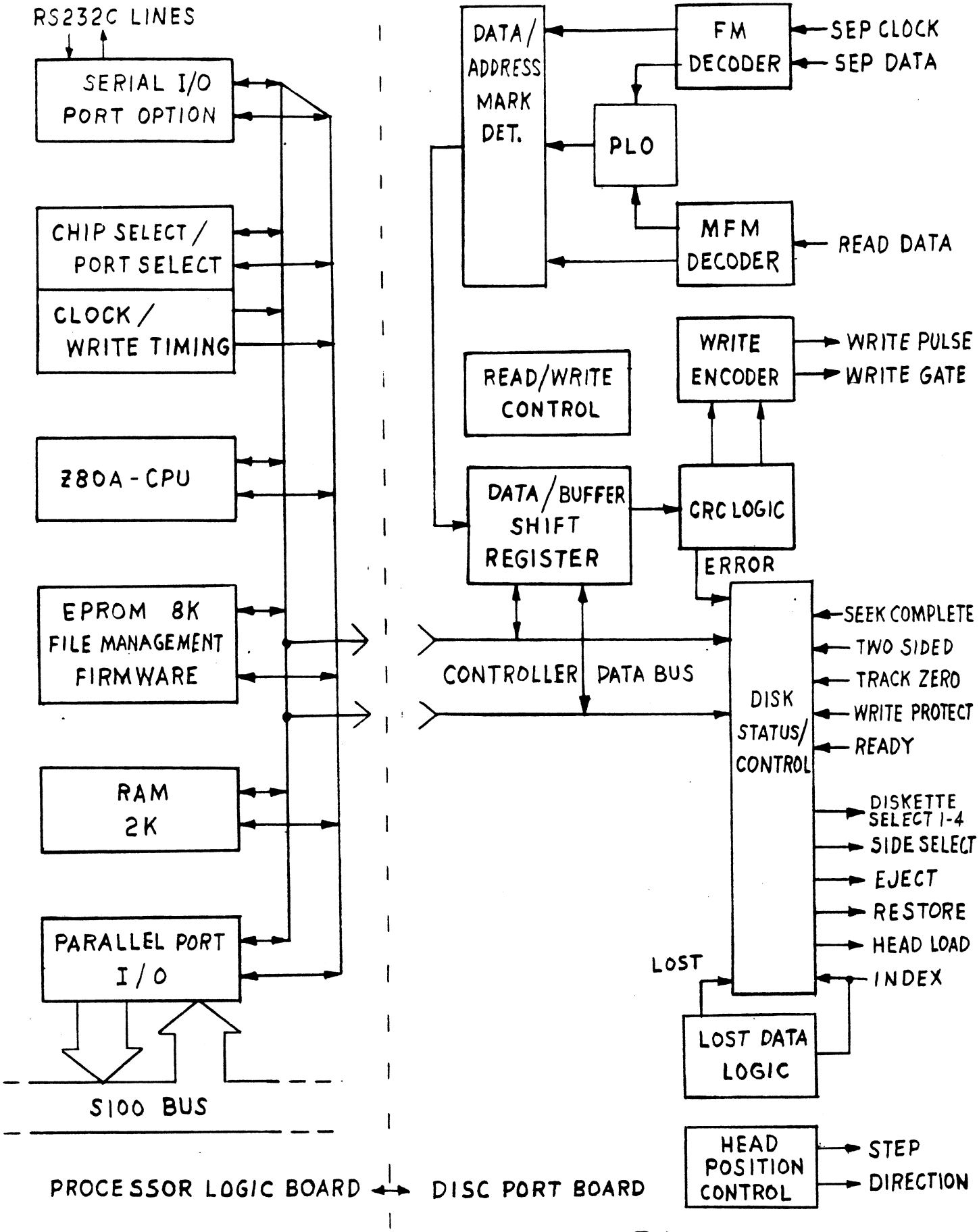


FIGURE 2.1 BLOCK DIAGRAM

3.0 CONTROLLER FIRMWARE

3.1 Series 2 File Management Firmware

First firmware for the 1170 is specified as Series FMF2.X, and is an extension of the Series FMF1, used with the 1070. To date, this series has four versions, all resulting in different diskette formats and/or different aspects to the host. However, each responds to the same commands with minor variations and each may be used on an 1170 with any of the allowable drive combinations.

3.1.1 Version 1 (F-2.1)

This version is designed to enable conversion of existing host operating systems based on 128 byte sectors to double-density operation. While each data field of a track sector is actually 256 bytes in length, the controller causes it to logically appear to the user as two 128-byte sectors. This feature results in slightly slower operation, but does not reduce storage capacity. (It is necessary to read a 256-byte sector before a 128-byte sector can be written.)

If additional RAM (2K) is added to the controller, optional programs are available to enable the conversion of PerSci 1070 formatted single-density to double-density diskettes or vice versa.

3.1.2 Version 3 (F-2.3)

This version operates with diskette formats identical to V1, but the user "sees" sectors 256 bytes in length. This version does not have the operational speed penalty of the Version 1.

3.1.3 Version 5 (F-2.5)

This version was intended to operate with IBM 2D diskette formats, but was found flawed after issue and is no longer issued. (Track zero on the second side is single density.) It has been replaced by Version 7, to which it is identical in every other respect.

3.1.4 Version 7 (F-2.7)

Version 7 is the only version to operate with either the IBM 3740 or IBM 2D diskette formats (single density or double density) and is the standard issued by PerSci with the 1170 unless otherwise specified.

Controllers with F-2.7 will operate interchangeably with single- or double-density diskettes and can be easily used to convert

diskettes of one density to another with the copy command. The controller will perform all commands automatically adjusting itself to the diskette format and code. Diskettes with 1070-generated formats may also be used. The 1170 will automatically convert index track content to match that used by the Version 7 Firmware and will reconvert to 1070-compatible content (V command) and eject the diskette (if the drive has the eject option).

3.2 Series F-2 and Two-Sided Diskettes

The 1170 Controller will automatically work with single- or two-sided diskettes inserted in PerSci 299 Drives.* The controller will cause two-sided diskettes to appear to the host as an extended diskette having twice as many sectors per track (or cylinder). This scheme minimizes head positioner change for long files and keeps the drive address of the Controller Commands consistent. It does restrict this series (F-2) to maximum of two 299 drives.

3.3 Data Storage Organization

Controllers with Series FMF2 store or retrieve diskette data by blocks one sector in length using the INPUT and OUTPUT commands. Higher level operation with the controller treats data as files where the user assigns a file name at time of first recording (SAVE command) or reserves file space in advance with the ALLOCATE command. The space allocation for each file is recorded in the index (i.e., Track 0) track of the diskette, leaving the remaining 76 tracks available for file storage.

Each file allocation is made up of contiguous sectors with a mimimum of one physical sector per file to a maximum of all sectors available on a diskette. (See Figure 3.0.) First file on a newly initialized diskette starts at Track 1, Sector 1. With subsequent files, space is allocated immediately after previous allocations. The controller will not permit allocations in excess of available diskette space. Files can be deleted (DELETE command) when desired but the file space will not be available for new files until a GAP command is executed to compress remaining allocations. Note that one GAP command will do all necessary compression following a number of DELETE commands. Each diskette thus serves as an independent storage entity which can be transferred from drive to drive without need for the host to keep track of file location.

*A two-sided diskette inserted in a 70 or 277 drive will result in a NOT READY indication.

The 0 track of each disk is reserved by the controller for holding an index identifying the volume and all files held on that volume.

The first 128 bytes of Sector 1 are reserved as a volume header formatted as follows:

	BYTES	1-8	VOLUME NAME
	BYTES	9-11	VERSION
	BYTE	12	TYPE
*	BYTES	13-14	1ST DATA SECTOR ON DISK
*	BYTES	15-16	LAST AVAILABLE SECTOR ON DISK
*	BYTES	17-18	FIRST FREE SECTOR ON DISK
	BYTE	19	FF IF FM DATA FIELDS TRACKS 1-76 00 IF MFM DATA FIELDS TRACKS 1-76
	BYTES	20-25	DATE CREATION
	BYTES	26-31	SPARE
	BYTE	32	INTERLEAVE SEQUENCE

The remainder of the 0 track is divided into 32 byte slots formatted as follows:

	BYTES	1-8	FILE NAME
	BYTES	9-11	VERSION
	BYTE	12	TYPE
*	BYTES	13-14	START OF ALLOCATION
	BYTES	15-16	EXTENT LENGTH (128 BYTE SECTORS)
*	BYTES	17-18	END OF DATA SECTOR (RELATIVE TO START OF FILE)
	BYTE	19	END OF DATA OFFSET
	BYTES	20-25	DATE CREATION
	BYTES	26-31	DATE LAST UPDATE
	BYTE	32	SPARE

*All items marked thus are held as 128 byte sector offsets from the start of disk.

Figure 3.0. Index File Format

All PerSci File Management Firmware reserve the outer zero track/ cylinder for use an an index (i.e., table of contents) file to the data files stored on the remaining cylinders. The first 128 bytes of the first sector are used as a volume header, while the remaining track capacity is divided into slots 32 bytes in length. Each slot contains the pertinent data on the diskette files, and the number of slots limits the possible files on a diskette.

3.4 Data/File Access Methods

The controller provides four methods for diskette data storage and retrieval. These are as follows.

3.4.1 Direct Access

Any data field of any sector of any track may be written or read directly bypassing the file management features of the firmware. This is the only method of direct access to the index tracks of the diskettes.

The remaining three methods are features of the File Management Firmware.

3.4.2 Stream Access

The stream access method permits an entire file to be read or written as a continuous stream of data bytes (as if the diskette file were a very high-speed paper tape). Stream access is the simplest access method to use, requiring only a single controller command to read (load) or write (save) an entire file. It is ideally suited to the storage and retrieval of executable programs or any other use in which paper tape or cassette tape is conventionally used. Stream access is performed using the "LOAD" and "SAVE" controller commands.

3.4.3 Punctuated Access

The punctuated access method treats an open file as a sequence of variable-length records separated by punctuation marks (the controller uses the ASCII record separator character "RS" for this). A punctuated file may be positioned at its beginning or end, and variable-length records may be read or written in sequence, one at a time. Records may span sector boundaries on the diskette, but this is made transparent by the controller. Punctuated access is most appropriate for the storage of text files (e.g., source programs or word processing files) or for any application in which sequential access to variable-length records is desirable. Because of its dependency on a unique punctuation character ("RS") to delimit records, punctuated access is not well suited to the storage of arbitrary binary information.

3.4.4 Relative Access

The relative access method treats an open file as a byte-addressable memory. A relative file may be positioned at its beginning, end, or to any desired byte position within the file. Any number of bytes may then be read or written. Relative read and write operations may span sector boundaries, but this is made transparent by the controller. Relative access is ideal for data-base oriented applications in which random access is required.

Punctuated and relative access methods require that a file name has been assigned and space allocated using the ALLOCATE command and that the target file has been opened with the FILE command. Data storage and retrieval then proceeds using the READ and WRITE commands.

Punctuated and relative access can be done with files created using the SAVE command.

When a file is opened, the controller automatically brings the target data sectors of the file from diskette to its RAM so that READ and WRITE operations can proceed without delay for any number of bytes so long as a sector boundary is not crossed. The controller will continue in this mode until a file is CLOSED when it will write the RAM data to diskette. Thus, a QUERY command will not reflect the true status of open files for the diskette.

The number of file allocations possible for each diskette is a function of its recording (single or double density) the index track, and if it is single or double sided. (See Figure 3-1.) For all FMF2 versions, a maximum of five files across all diskettes can be OPEN at any one time, and all files must be closed before a COPY or GAP command can be performed. (Each OPEN file requires 256 bytes of RAM of the available 2K. COPY and GAP require all available RAM as a buffer memory.)

3.5 File References

A file reference identifies a particular file or group of files. File references may be either unique or ambiguous: a unique file reference identifies one file uniquely, while an ambiguous file reference may be satisfied by several different files.

File references consist of four components: a name of up to eight characters (NNNNNNNN), a version consisting of a period followed by up to three characters (.VVV), a type consisting of a colon followed by a single character (:T), and a disk number consisting of a slant followed by a digit between 0 and 3 (/D). The version, type, and drive components are optional and are set off from the

name by means of their unique leading punctuation characters (NNNNNNNN.VVV:T/D). A missing name, version, or type is assumed to be blank, and a missing disk number is assumed to refer to the current default drive.

The following are examples of valid unambiguous file references:

```
MONITOR             MASTER/2          STARTREK.BAS/1
MONITOR.SRC         MASTER:$          STARTREK.XQT
MONITOR.OBJ:A       MASTER.ONE        STARTREK:Z/O
```

The special characters "?" and "*" may be used to make a file reference ambiguous so that it may match a number of different files. The "?" is used as a "wild-card" character which matches any character in the corresponding position in a file reference. Thus, the ambiguous file reference:

PER????.BA?

matches all of the following unambiguous file references:

PERFECT.BAL PERSCI.BAS PERQ.BAX

The character "*" is used to denote that all character positions to the right are wild-cards unless otherwise specified. The following examples illustrate the flexibility which this facility provides:

Reference	Equivalent to	Ambiguous Reference Matches
MONITOR.*	MONITOR.???:?	all files with name MONITOR
*.BAS	????????.BAS:?	all files with version .BAS
Z*	Z???????.???:?	all files starting with Z
*	????????.???:?	all files on the diskette

3.6 Controller Commands

Controller commands consist of a single command letter followed (in most cases) by one or more command parameters. Parameters must not contain embedded spaces, must be set off from one another by spaces, and may optionally be set off from the command letter by spaces.

CONTROLLER COMMAND SUMMARY

Command	Command Syntax	Command Function Summary
Allocate	A file sectors	Allocates an empty file "file" of "sectors" sectors.
Copy	C file1 file2 sectors	Copies files matching "file1" to same or different diskette, optionally renaming according to "file2" and reallocating according to "sectors."

Command	Command Syntax	Command Function Summary
Delete	D file	Deletes all files matching "file."
Eject	E /disk	Ejects diskette.
File	F unit file	Opens "file" and associated with "unit."
	F unit	Closes the open file associated with "unit."
	F	Closes all open files.
Gap	G /disk	Compresses allocations on "drive" to eliminate gaps.
Header	H volume/disk	Rename diskette.
Input	I track sector /disk	Reads specified sector.
Kill	KK volume/disk seq	Initializes diskette with interleave "seq."
	KK volume/disk	Deletes all files on diskette without initializing.
Load	L file	Reads entire file "file" as a stream.
Mode	M date:density : options/disk	Sets current date, I/O options, density and/or default drive.
Name	N file1 file2	Renames file "file1" in accordance with "file2."
Output	O track sector /drive	Writes specified sector.
Position	P unit sector byte	Positions the open file associated with "unit."
	P unit	Reports current position of file associated with "unit."
Query	Q file	Reports index information for files matching "file."

Command	Command Syntax	Command Function Summary
Read	R unit bytes	Relative read of file associated with "unit."
	R unit	Punctuated read of file associated with "unit."
Save	S file	Creates new file "file" by writing as a stream.
Test	T option/disk	Executes a diagnostic test on drive "drive."
Vale	V /disk	Convert diskette header to 1070 compatible and eject.
Write	W unit bytes	Relative write to file associated with "unit."
	W unit	Punctuated write to file associated with "unit."
Xecute	X file option	Loads file "file" into controller RAM and executes it.
Yourstatus	Y /disk	Read drive status.
Zap	Z unit	Writes end-of-data mark at present position of file associated with "unit."

NOTE

Numeric command parameters (byte, bytes, sector, sectors, seq, track) may be either decimal or hexadecimal. Hexadecimal parameters must be prefixed by a letter (such as "H" or "X;" for example, the commands:

```
A  FNAME  32
A  FNAME  X20
```

will both allocate a file whose length is 32 (decimal) sectors.

NOTE

H, V, and Y commands and ".D" parameter for mode command do not exist in 1070 firmware.

3.6.1 Allocate Command (A File/Disk Sectors)

The ALLOCATE command creates a new, empty file with the specified allocation (decimal or hex number of sectors).

Example:

```
A  BIGFILE  1000
```

3.6.2 Copy Command (C file1 file2 sectors)

The COPY command copies one or a collection of files from a diskette volume to the same or a different diskette volume. The copied files may have the same or different names as the original files, and may have the same or different allocations. The COPY command cannot be used if there are any open files.

Examples:

```
C  ALPHA  BETA
C  ALPHA/0  */1
C  ALPHA/0  BETA/1  100
C  */0  */1
C  A*/0  B*/1
```

The first example makes a duplicate of the file ALPHA on the same diskette (default drive), calling the duplicate BETA. The second example copies the file ALPHA from drive 0 to drive 1, leaving the name and allocation unchanged. The third example also copies ALPHA from drive 0 to drive 1, but changes the name to BETA and gives the new file an allocation of 100 sectors (which may be larger or smaller than ALPHA). The fourth example copies all files from drive 0 to drive 1, preserving all file names and allocations. The last example copies only files with names starting with "A" from drive 0 to drive 1, changing the first character of each file name from "A" to "B."

3.6.3 Delete Command (D file/disk)

The DELETE command deletes a file or a collection of files from a diskette.

Examples:

```
D  GEORGE
D  *.OBJ/1
D  XZ??/2
```

The first example deletes a single file GEORGE from the default drive. The second example deletes all files on drive 1 which have version .OBJ. The last example deletes all files on drive 2 which have two to four character names starting with "XZ".

3.6.4 Eject Command (E/disk)

The EJECT command causes the diskette to be ejected from the specified drive. Note that this command is effective only if the diskette drive is equipped with the Remote Eject feature.

Examples:

```
E  /2
E
```

3.6.5 File Command (F unit file/disk)

The FILE command opens and closes diskette files. A file must be open before punctuated or relative access is permitted by the controller. An open file is associated with a logical unit number between 1 and 5 (a maximum of five files may be open at one time).

Examples:

```
F  2  MASTER/1
F  2
F
```

The first example opens the file MASTER on drive 1 and associates it with logical unit 2. The second example closes the open file associated with logical unit 2. The third example closes all open files.

3.6.6 Gap Command (G /disk)

The GAP command compresses the allocations on a diskette volume to eliminate any gaps in the allocations caused by prior file deletions. The GAP command cannot be used if there are any open files.

Examples:

```
G  /3
G
```

3.6.7 Input Command (I track sector /drive)

The INPUT command reads a single specified sector of a diskette volume. The sector is specified by decimal track number (0-76), the sector number may be as high as 52 for 2.7 and 104 for 2.1 firmware.

decimal sector number (1-26), and drive number. (The track and sector number may also be hexadecimal.)

Examples:

```
I  43  10  /1
I  1   1
```

3.6.8 Kill Command (KK volume/disk seq)

The KILL Command rewrites the Volume Header (TK-0 sec.1) renaming the diskette and deletes all other index file data. Optionally, the command also initializes (formats) the entire diskette, erasing all previously recorded information thereon and writing new sector headers on each track. The diskette may be initialized with any one of thirteen sector interleave sequences to enhance read/write performance.

Examples:

```
KK SCRATCH/3
KK BACKUP  1
KK MASTER  9
```

The first example deletes all files on drive 3, labels the volume SCRATCH, but does not initialize each track. The second example initializes the diskette on the default drive without interleave. The last example initializes with interleave sequence 9.

3.6.9 Load Command (L file/disk)

the LOAD command reads a diskette file in its entirety as a stream.

Examples:

```
L  BASIC
L  EDITOR/3
```

3.6.10 Mode Command (M date.density*:options/disk)

The MODE command may be used to set the current date, the default diskette, and/or various controller options. The current date is entered as a six-character value (the format YYMMDD is suggested, but not required by the controller). The default diskette is

*Density only for Version F-2.7.

entered as the character "/" followed by a drive number (0-3); this becomes the drive which is used for all subsequent file references and commands which do not include an explicit drive number. The options are entered as the character ":" followed by a single hexadecimal digit (0 through F) whose bits are microcoded as follows:

Option	Meaning
:8	Suppress nonfatal error message
:4	Simultaneous head load NOT available
:2	Keep heads loaded continuously
:1	Model 70 drives in use

NOTE

At initial power up, the controller assumes an option code of zero by default, that drives with the simultaneous head load feature are in use, and that diskette initialization mode is double density.

Single- or double-density mode may be set (F-2.7 only) as follows:

.S Initialize Diskette Single Density (FM)
.D Initialize Diskette Double Density (MFM).

Examples:

```
M 770819.D
M /1
M :8.S
```

The first example sets a date and double density mode, while the last example informs the controller that the controller and/or drive do not support simultaneous head load, that nonfatal error messages are to be suppressed, and sets single-density mode for diskette initialization.

3.6.11 Name Command (N file1 file2)

The NAME command modifies the name, version, and/or type of a file. The wild-card characters "?" and "*" are used to indicate that selected portions of the file reference are to be left unchanged, as illustrated in the examples. The file reference may be ambiguous if all files satisfying the reference are to be renamed. The 1170 will reply with all renamed files followed by ACK EOT.

Examples:

```
N  ALPHA BETA
N  BACKUP.2  *.3
N  XRATED  R*
```

The first example changes the file ALPHA to BETA. The second example changes BACKUP.2 to BACKUP.3, while the third changes XRATED to RRATED.

3.6.12 Output Command (O track sector /disk)

The OUTPUT command writes a single-specified sector of a diskette volume. Its parameters are identical to those for the INPUT command. Controller will output data in compatible code and length to diskette format.

Examples:

```
0  43  10  /1
0  1  1
```

3.6.13 Position Command (P unit sector byte)

The POSITION command permits open files to be positioned at the beginning, end, or at any specified byte position. The command may also be used to report the current position of an open file.

Examples:

```
P  2  213  88
P  2  213
P  2  0
P  2  65535
P  2
```

The first example positions the open file associated with logical unit 2 to byte 88 in sector 213 of the file. The second example positions the file to byte 0 of sector 213. The third example positions the file at its beginning, and the fourth example positions the file at its end-of-data (note that the controller does not permit a file to be positioned beyond its end-of-data). Finally, the last example simply reports the current position of the file.

3.6.14 Query Command (Q file/disk)

The QUERY command lists the following index information for one, some, or all files on a diskette volume:

- Name, version, and type
- Start of allocation (decimal track and sector)
- Length of allocation (decimal number of sectors)
- End of data (decimal sector and byte offset)
- Date of creation
- Date of last update.

This information is preceded by a heading which lists the volume name, next available track and sector, volume interleave, density, and date initialized.

CAUTION

Index information will not be current for OPEN files.

Examples:

```
Q  ALPHA/2
Q  *.SRC
Q  *
```

Sample QUERY listing:

```
DEMO.DSK 06-07 09 D 770215
FMF11.OBJ;3         01-01 0032 0031 082 770430
TEXTED              02-07 0025 0024 000 770503
DOCUMENT.TXT        03-06 0079 0078 001 770503 770618
```

3.6.15 Read Command (R unit bytes)

The READ command reads an open file by means of either the relative or punctuated access method (i.e., fixed-length or variable-length records).

Examples:

```
R  2  80
R  2
```

The first example reads a fixed-length record of 80 bytes from the current position of the open file associated with logical unit 2. The second example reads a variable-length record delimited by a record separator character ("RS").

NOTE

The maximum length of a fixed-length read is 65535 bytes (HFFFF).

3.6.16 Save Command (S file/disk)

The SAVE command creates a new file by writing a stream of data onto the diskette. The resulting file receives an allocation of the minimum number of sectors needed to accommodate the length of the stream.

Examples:

```
S BASIC
S EDITOR/3
```

3.6.17 Test Command (T option/drive)

The TEST command performs one of several diagnostic tests on the specified drive. Available tests are: V (random seek-verify test), R (random seek-read test), and I (incremental seek-read test).

Test V is a high-speed random-seek test. It performs a sequence of seeks to a randomly-selected track, reads the first encountered sector header on that track, and verifies that the correct track has been reached.

Test R is a random-seek-read test. It performs a seek to a randomly selected track, then reads a particular randomly selected sector on that track, and verifies that both the sector header and sector data are correct (using the CRC in each case).

Test I is an incremental-read test. It reads and verifies both the sector header and sector data of each sector on the diskette, starting at track 0 sector 1 and proceeding incrementally through track 76 sector 26. This test may be used to verify diskette format.

Once initiated, tests V and R run indefinitely until the controller is reset or until a hard disk error is encountered which persists for nine successive retries. Test I makes a single pass over the diskette, reading each sector once, and then terminates.

Examples:

```
T V/1
T R/0
T I
```

3.6.18 Write Command (W unit bytes)

The WRITE command writes an open file by means of either the relative or punctuated access method (i.e., fixed-length or variable-length records). If data is written beyond the end-of-data of the

file, the end-of-data is moved accordingly. The controller will not permit data to be written beyond the last sector allocated to the file.

Examples:

```
W  2  80
W  2
```

The first example writes a fixed-length record of 80 bytes to the open file associated with logical unit 2, starting at the current position of the file. The second example writes a variable-length record to the file, followed by a record separator character ("RS").

NOTE

The maximum length of a fixed-length write is 65535 bytes (HFFFF hex).

3.6.19 Xecute Command (X file option)

The XECUTE command loads an executable diskette file into controller RAM and executes it. This permits diskette-resident routines to extend the effective command repertoire of the controller. The option is a decimal or hex parameter which is passed to the routine.

Note that the XECUTE command is not required for normal use of the controller, but was included to facilitate special applications of the controller. For further details, contact PerSci.

Examples:

```
X  DRIVTEST  1
X  CONVERT
```

3.6.20 Zap Command (Z unit)

The ZAP command truncates an open file by establishing the end-of-data at the current position of the file. Note that this command does not affect the allocation of the file, only its end-of-data position. Zapping a file so that only desired data is preserved will increase WRITE command operation speed.

Example:

```
Z  2
```

3.6.21 Vale Command (V/disk) - F-2.7 Only

The Vale command converts the volume label (track 0, Sector 1) of a single-sided, single-density diskette to be compatible with that used by the PerSci 1070 controller. It will also cause the diskette to be ejected if the drive has that option.

3.6.22 Yourstatus Command (Y/disk)

The status for the addressed disk as defined following is read, converted to hex equivilent, and returned to the host. The bit meaning for status byte read is

(MSB) Bit 7 true for data "lost" error
6 True for data "CRC" error
5 False for write protect (drive must have option)
4 False if R/W head is at track zero
3 False if diskette is two sided
2 False if index hole is being detected at instant of reading status
1 False if R/W head positioning servo has completed seek
0 False if disk is ready for data exchange

3.6.23 Header Command (H volume/disk)

The HEADER command will rename the diskette without disturbing the existing index file data. This command enables relabeling of diskettes with no other consequences.

NOTE

All F2.7 commands will determine data code (FM or MFM) by reference to Volume Header (Byte 19 of Track zero, Sector 1).

If Byte 19 = 00, then MFM code is assumed.

If Byte 19 = FF, then FM code is assumed.

If Byte 19 = E5, then FM code is assumed with a 1070 format (see Section 3.1.4)

Any other value will result in a Format Error response to commands other that INPUT, OUTPUT, and the TEST options. For these commands, if Byte 19 is ambiguous, the controller will attempt opperation in the code set by the MODE Command.

FMF SERIES	DISKETTE TYPE	DISKETTE ORGANIZATION ADDRESSABLE BY HOST					
		DATA TRACK	SECTORS TRACK	BYTES SECTOR	MAX NO. FILES	MAX OPEN FILES	DATA BYTES/ PER DISKETTE
2.1/2	P1D1	1→76	52	128	204	5	505,856
2.1/2	P2D1	1→76	104	128	412	5	1,011,712
2.3/4	P1D1	1→76	26	256	200	5	505,856
2.3/4	P2D1	1→76	52	256	408	5	1,011,712
2.7/8	P1	1→76	26	128	100	5	252,928
2.7/8	P1D2	1→76	26	256	200	5	505,856
2.7/8	P2S	1→76	52	128	200	5	505,856
2.7/8	P2D2	1→76	52	256	200	5	1,011,712

Figure 3.1 FILE MANAGEMENT FIRMWARE DISKETTE ORGANIZATION

4.0 INSTALLATION

4.1 Physical Description

The controller consists of two printed circuit boards designed for installation next to each other in a standard S100 bus mainframe. Each board has its own voltage regulators for the S100 power distribution system. Two ribbon cables, captive to one board, are used to make necessary signal interconnections between the boards. The processor logic PCB also has an edge connector for use with a twenty-six pin ribbon cable connector for use with E.I.A. RS232 Data Terminal Equipment. The Disk Port PCB has a fifty-pin edge connector for the Drive Ribbon Cable.

4.2 Controller Configuration

Various options are provided for the 1170 controller which may be selected by connecting jumpers on the processor PCB. These are:

4.2.1 Reset Connections (Points L,M, N)

The controller PB Reset may be connected to a host reset at Pin 75 (Jumper L-M) or at Pin 54 (Jumper L-N). Either of these will also require a jumper or diode at CR2. The diode should be used if it is desired to isolate the controller PB reset from also resetting the host. For best results, a Germanium diode should be used.

The host reset should drive Pin 54 or 75 to ground for at least 10 milliseconds.

4.2.2 EPROM Type Selection

The 1170 is designed to use three types of EPROMS, (2708s, INTEL 2716s, or TI 2716s). Jumper Points A, B, C, D, E, F, H, W, X, Y, and Z are provided to enable this feature which will be set at PerSci. (For details, see PerSci Drawing No. 200648.)

4.2.3 Serial Port Jumpers

Jumper J-K. This jumper is factory installed to enable the 1170 to operate with RS232 serial two wire devices by grounding the CLEAR-TO-SEND input of the 8251. It should be removed for operation with devices which use this signal.

T1, T2, T3, T4, T5, T6. The six connections of the RS232 interface are etched at the connector so that the traces may be cut and the connections reversed if desired.

4.3 Diskette Drive Configuration

The 1170 controller using FMF2-1, 3, 5, and 7 will accommodate two Model 277 drives, four Model 70s, or two Model 299 drives with minimum changes to the controller.

Address logic for the Model 70, 277, and 299A drives is set by jumpers on a select module on the biggest PCB of the drives. The select logic of the Model 299 is controlled by jumpers on the PCB itself.

Following are necessary jumpers to set drive addresses.

CAUTION

When two or more drives are "daisy chained," only the drive at the end of the ribbon cable should have the resistor pack termination module installed.

Drives will normally be delivered following checkout at PerSci set as Drive 1 and with termination module installed.

Model 277	Select Module Jumpers (U11)
Drive 1 (Side 0 and 1)	2 to 13, 4 to 11
Drive 2 (Side 2 and 3)	1 to 14, 6 to 9

In addition, the connection at Pin 6 of the Model 277 drive PCB edge connector for the 50-pin cable should be cut to prevent it from being grounded by the drive.

Model 70	Select Module Jumpers (U5)
Drive 1 (Side 0)	7 to 8, 3 to 12
Drive 2 (Side 1)	7 to 8, 4 to 11
Drive 3 (Side 2)	7 to 8, 5 to 10
Drive 4 (Side 3)	7 to 8, 6 to 9

Connection of Model 70 drives as Drive 3 or 4 can only be done with controllers having A or later revision PCBs. Also required with Drive 3 is a jumper from Point A to B of the Disk Port PCB. Drive 4 requires a jumper from Point C to Point D.

Model 299A	Select Module Jumpers (U1)
Drive 1 (Disks 0 and 1)	1 to 14, 2 to 13
Drive 2 (Disks 2 and 3)	4 to 11, 3 to 12

U2, U3, and U5 should not have any chips installed. A 7432 should be installed at U6.

Model 299B	PCB Jumpers*
Drive 1 (Disks 0 and 1)	Factory standard (trace link C11-C12)
Drive 2 (Disks 2 and 3)	Cut trace C11-C12, Jumper C1-C2

With the Model 299 drives, reference is made to disks rather than sides because the controller treats a two-sided diskette as a "super" diskette having twice the number of sectors per track as a single-sided one. Note that it is possible to mix the model types handled by the controller so long as no redundant sides/ disks exist. The 1170 controller will accommodate three to eight Model 299 drives with FMF2-4, 6, and 8.

4.4 Installation

a. Turn off S100 bus power.

b. Insert the two PCBs in adjacent connectors of the S100 bus. Preferred position of the Disk Port PCB is in front (i.e., on component side) of the Processor Logic PCB.

c. Connect 50-conductor ribbon cable from drive to controller Disk Port PCB.

CAUTION

Ensure that conductors of cable are in proper order so that controller output Pin 50 goes to drive Pin 50.

d. Connect ribbon cable to Serial I/O Port of Processor Logic PCB (if serial I/O is desired and optional components are installed).

e. Turn on power.

4.4.1 S100 Port Addresses

The controller Data Port address should be set to one of the possible 128 left where the least significant bit of the address must be zero. The Status/Control Port address will then be the next higher number. The selection is made by putting jumpers at Points A1 thru A7 on the Processor Logic PCB. Each point is associated with an address line of the S100 bus and should be connected if the bit of the desired address is high. For example, if the desired Data Port address is 8E (with resultant Status/Control Port at 8F), then jumpers should be installed at Points A7, A3, A2, and A1.

*Factory standard links C3-C4, C5-C6, C7-C8, and C9-C10 should also be present.

CAUTION

Controllers will usually be delivered with jumpers at A7 and A6 setting the Data, Status/Control Ports to C0, C1, but this is subject to change without notice.

4.5 Phase-Locked Loop (PLL) Adjustment

The Phase-Locked Loop circuitry of the Disk Port PCB is primarily designed for use in reading double-density data, but is also is used in conjunction with a PerSci single-density data separator in a drive to read single-density data. It is preset at checkout at PerSci and should not need readjustment unless some component has been replaced.

The PLL output is a 1 MHz frequency signal (the sawtooth waveform at TP7) which is twice the clock signal of the signal read back from the diskette with MFM data. The PLL is also designed so that its output will follow minor variations of frequency and phase of the recovered signal, thus compensating for diskette speed variations and media distortion.

In the following paragraphs, a simple four-step procedure is given for adjustment of the PLL, which requires an oscilloscope having two traces, one of which can be used to synchronize the display. Oscilloscope bandwidth should be 25 MHz or better.

The procedure steps should be followed in sequence as each adjustment is influenced by the previous steps.

4.5.1 Initial Set Up

a. Connect A probe to TP7 of the Disk Port PCB. Set A trace sensitivity to 5V/DIV.

b. Connect B probe to TP (base of Q4 for X1, X2, and X3 PCBs). Set B trace sensitivity to 5V/DIV.

c. Probe ground leads should be connected to ground TP8 on PCB.

d. Set scope sweep to 1/10 μsec/DIV.

4.5.2 Coarse Phase Adjustment

a. Using A trace, adjust pot R32 so that sawtooth at TP7 is 1.0 microsecond (rate equals 1.0 MHz).

b. Execute TI command, and using B trace only, adjust pot R19 so that sawtooth waveform at TP12 is balanced about ground. (That is, maximum positive and negative voltage excursions are about equal.)

4.5.3 Coarse Frequency Adjustment

a. Disconnect B probe.

b. Execute TI command, and using A trace adjust pot R32 so that PLL output at TP7 will lock (i.e., synchronize) to read data.

4.5.4 Frequency Range Test and Final Adjustment

a. Move B probe to TP10. Set B trace sensitivity to 1V/DIV.

b. Verify that pot R32 may be used to adjust voltage at TP10 over the range of plus (+) 400 millivolts to minus (-) 400 millivolts while PLL is locked to data.

c. Adjust R32 to that TP10 voltage is balanced about zero volts.

4.5.5 Final Phase Adjustment

a. Move B probe to TP1. Set B trace sensitivity 5V/DIV.

b. Adjust pot R19 so that positive-going trailing edge of Read Data (TP1) is centered on the positive-going ramp of the PLL output (TP7).

4.6 The 1170 Controller and the 299B Drive

The 1170 Write Encoder is designed so that the write precompensation for MFM code is selectable by change of the 2708 EPROM at U27 on the Disk Port PCB. Double-density (MFM) operation with the Model 70, 299A, and 270 series drives requires write precompensation and EPROMs marked EII should be used.

The read detection circuitry of the Model 299B drive may be operated in one of two modes: with or without Read Compensation.

The 299B with Read Compensation should be used with diskettes formatted and written without precompensated MFM codes, and with controllers that do not precompensate their output write pulses. Either mode may be used with single-density (FM) codes.

If the system diskettes are to be:

a. Read by PerSci Model 70, 299A, or 270 Series Drives as well as a 299B, or

b. Read by other drives not having Read Compensation

then the 299B should be set for no Read Compensation and the controller should use precompensation. Following is a tabular summary of compatible drives and controllers.

SYSTEM DISKETTE/COMPONENT COMPATIBILITY

Mode	Diskette Formatted/ Written	Drives*	1170 Controller Encoder EPROM Disk Port (U27)
Ƶ	MFM	299B with Read Compensation	EƵ
II	Precompensated MFM	70/270 Series/299A 299B without Read Compensation	EII

*NOTE

Drives must have been manufactured and/or qualified as double-density (MFM) units.

Users upgrading to 299B only operation can probably convert their existing diskette library to Mode Ƶ formats by using a Mode Ƶ system to copy diskettes.

The new diskettes will be formatted/written pure MFM, and it is probable that the 299B will be able to successfully read the older diskettes. (The controller will make up to 9 retries on any errors and will effectively "filter" the precompensated formats.)

299B READ AMPLIFIER CONFIGURATION

299B	Cut/Jumper
With Read Compensation	Cut H3-H4/Jumper H1-H2*
Without Read Compensation	Cut H1-H2/Jumper H3-H4

*Factory standard.

4.7 Serial I/O Baud Rates

The RS232 Serial Interface Option includes an on-board speed selection switch with the following settings:

Switch Setting	Transmission Speed (BPS)	Switch Setting	Transmission Speed (BPS)
0	50	8	1,800
1	75	9	2,000
2	110	A	2,400
3	134.5	B	3,600
4	150	C	4,800
5	300	D	7,200
6	600	E	9,600
7	1,200	F	19,200

NOTE

The controller outputs serial characters with one start bit, eight data bits (no parity), and one stop bit for all transmission speeds.

5.0 SYSTEM CONSIDERATIONS

5.1 Parallel Port Function

The 1170 parallel Data Port is designed for data transfer using the IN and OUT instructions of an 8080 or Z80 CPU. It provides fast coordinated data exchange independent of data content with a minimum burden to the host software. Further, there is no requirement to cause the host CPU to hang in "wait" status to synchronize its operation with the controller for data exchange.

5.2 Parallel Handshake

The Status/Control Port is provided to coordinate data exchange via the Data Port/IN instruction to the Status Port will read a byte, three bits of which are of significance.

5.2.1 Status Byte Bits

Bit 7 (MSB) — If high, the controller data port holds a byte which is an ASCII control character.

Bit 6 — If high, the controller data port holds a byte for the host. If bit 7 is low, then this byte is not a control character.

Bit 1 — If high, then the controller is ready to accept a byte from the host. The byte may be either a control character or data byte. (Note that this is a change from the PerSci 1070.)

Bits 5, 4, 3, 2, 0 — These lines are not driven by the controller and may be ambiguous.

NOTE

- Reading the Status Port will not affect either port.
- If Bit 7 is high, then Bit 6 should be high. Bit 6 and 1 can be high at the same time.
- If Bit 6 is high, then the controller Data Port should be read before the host attempts an OUT instruction to it.
- An IN (Read) instruction to the Data Port will cause Status Port Bit 6 (and Bit 7 if byte is a control character) to go low.

NOTE (CONTINUED)

- An OUT (Write) instruction to the Data or Control Ports will cause Status Port Bit 1 to go low.

- An OUT (Write) instruction to the Control Port does not have to be repeated to the Data Port as required by the 1070.

CAUTION

No IN instruction should be performed to the Data Port unless Bit 6 is high.

No OUT instruction should be performed to the Data Port unless Bit 1 is high.

5.3 Serial Data Port

The Serial Data Port is an option which if installed provides the controller with a serial-by-bit data exchange capability meeting the voltage levels of RS232C. This port is considerably slower than the parallel and is limited in that it should be used only with ASCII encoded data. (See Section 4 for possible baud rates.) The controller will output serial characters with one start bit, eight data bits (no parity), and one stop bit for all transmission speeds.

5.4 Power Up/Reset Conditions

The Controller Firmware (FMF2 versions) will operate with either the parallel or serial ports, but not both after initialization. When it is "powered up" or reset it will output a message as follows:

PerSci 1170 V F2-X CR LF EOT

where X will identify the FMF version.

A parallel bus host must read the first byte (ASCII "P") within 30 seconds or the controller will default to Serial I/O Mode only.

CAUTION

Default to Serial I/O Mode only may occur even with controllers which do not have the necessary optional hardware installed!

Note that this first ASCII "P" will be read by the parallel host even if default has occurred and that default must be deduced if the Status Port fails to indicate succeeding outputs.

5.5 Interface Protocol

The interface protocol between the microcomputer and the controller consists of sequences of ASCII characters and makes use of standard ASCII communications controls. The protocol for the simplest controller commands (ALLOCATE, EJECT, FILE, KILL, MODE, NAME, TEST, XECUTE, ZAP) is the following:

```
Microcomputer sends:  command-text [EOT]
Controller sends:     ACK  EOT
```

The protocol for controller commands which return informational text (COPY, DELETE, GAP, POSITION, QUERY) is the following:

```
Microcomputer sends:  command-text [EOT]
Controller sends:     informational-text  CR  LF  ACK  EOT
```

The protocol for controller commands which read data from diskette (INPUT, LOAD, READ) is the following:

```
Microcomputer sends:  command-text [EOT]
Controller sends:     SOH  diskette-data  ACK  EOT
```

The protocol for controller commands which write data to diskette (OUTPUT, SAVE, WRITE) is the following:

```
Microcomputer sends:  command-text [EOT]
Controller sends:     ENQ  EOT
Microcomputer sends:  diskette-date [EOT]
Controller sends:     ACK  EOT
```

Finally, the controller may terminate any command at any time with a fatal error diagnostic message, using the following protocol:

```
Controller sends:     NAK  fatal-error-msg  CR  LF  EOT
```

Note that no ACK will be transmitted by the controller in this case.

Underlined characters are "flagged" as control by Status Port. [ ] indicates characters written to the Control Port.

5.6 Error Diagnostic Messages

The controller issues two classes of error diagnostic messages: fatal and nonfatal. Fatal error diagnostic messages are always preceded by a NAK and followed by an EOT. They indicate the premature and unsuccessful termination of a controller command. The various fatal error diagnostic messages are listed below:

- COMMAND ERROR ON DRIVE #n

  Indicates that the controller received an invalid command or command parameter.

- DUP FILE ERROR ON DRIVE #n

  Indicates that an attempt was made to create a new file with the same name as an existing file on the same diskette.

- HARD DISK ERROR ON DRIVE #n

  Indicates that a seek, read, or write error occurred which could not be successfully resolved in nine retries.

- NOT FOUND ERROR ON DRIVE #n

  Indicates that the specified file could not be found in the index of the specified diskette.

- OUT OF SPACE ERROR ON DRIVE #n

  Indicates that an attempt was made to exceed the capacity of a diskette, an index track, or a file allocation.

- READY ERROR ON DRIVE #n

  Indicates that an attempt was made to access a diskette drive which is not in ready status.

- UNIT ERROR

  Indicates that an attempt was made to read, write, or position a logical unit number with which no open file is associated, or that an attempt was made to use the COPY or GAP commands with one or more files open.

- FORMAT ERROR

  Indicates diskette format is not suitable for command.

The clause "ON DRIVE #n" is omitted in the case of errors not associated with a particular drive.

Note that each fatal message begins with a unique letter, so that an interfacing program need only analyze the first character following a NAK to determine the type of fatal error.

Nonfatal error diagnostic messages are issued for soft disk errors. They are not preceded by a NAK, and they contain the following information:

- Type of disk operation (seek, read, or write)
- Error retry number (1 to 9)
- Track and sector at which error occurred
- Type of error (protect, verify, CRC, or lost)

Multiple error-type indications may be received on a single nonfatal error message, and their meanings are as follows:

- Protect: a write was attempted on a write-protected disk
- Verify: the desired sector header could not be found
- CRC: the sector header and/or data failed the CRC test
- Lost: controller could not detect address or data marks for one complete diskette revolution.

During the transmission of diskette data (LOAD, SAVE, READ, WRITE, INPUT, and OUTPUT commands), nonfatal error messages are suppressed. They may also be suppressed under all circumstances by means of the MODE command.

5.7 Sample Drive Program

For further guidance in the system integration of the 1170, a suggested flow chart for a host driver program is provided as Appendix A of this manual along with the assembly listing coded for an 8080-based host. The flowchart is, and the program almost, identical to that used in the PerSci 1070 Users Manual.

5.8 Controller Timing

5.8.1 Parallel Port data exchanges must meet the requirements shown by Figure 5-0, where all timing references are specified at the midpoint of the rising or falling edge of the signals. Rise and fall times should not exceed 50 nanoseconds.

5.8.2 RESET* should be driven to ground for a minimum of 10 milliseconds.

5.8.3 The controller will exchange blocks of data (256 bytes double density/128 bytes single density) at the maximum baud rate of 19,200 bps for the serial port and up to 40,000 bytes/sec for the parallel port. It will then pause for short intervals to read or write data to the diskette drive and other housekeeping tasks. The time consumed for these pauses is a function of the diskette format interleave, head position, etc., and is very application sensitive.

5.8.4 The lack of a "handshake" protocol for the serial port requires the user to ensure that data sent at higher baud rates to the controller is not lost. This is best done by the user inserting pauses of 1/3 second or more between blocks of transmitted data.

5.8.5 Diskette Interleave

The controller provides 13 different sequences for sector addresses when initializing a diskette with the KILL command. (See Appendix B.) Selection of interleave sequence can greatly affect operational speed when storing or retrieveing files two or more sectors in length. The optimum sequence will greatly depend upon the host CPU speed, the driver program, file access method used, etc., and experimentation is recommended.

5.8.6 First users of the 1170 report data exchange rates of 2 to 12 times the rate possible with the PerSci 1070 for similar applications. The 1170 is faster because:

- It writes/reads twice as much data per diskette revolution (in double density 256 bytes/sector mode).
- It exchanges data with the S100 bus via a Z80A-P10 using the powerful interrupt system provided.
- The controller CPU is a Z80A operating at 4.0 MHz doubling the rate for housekeeping task processing.
- The index format for the diskette was redesigned to minimize calculations to locate file sectors.

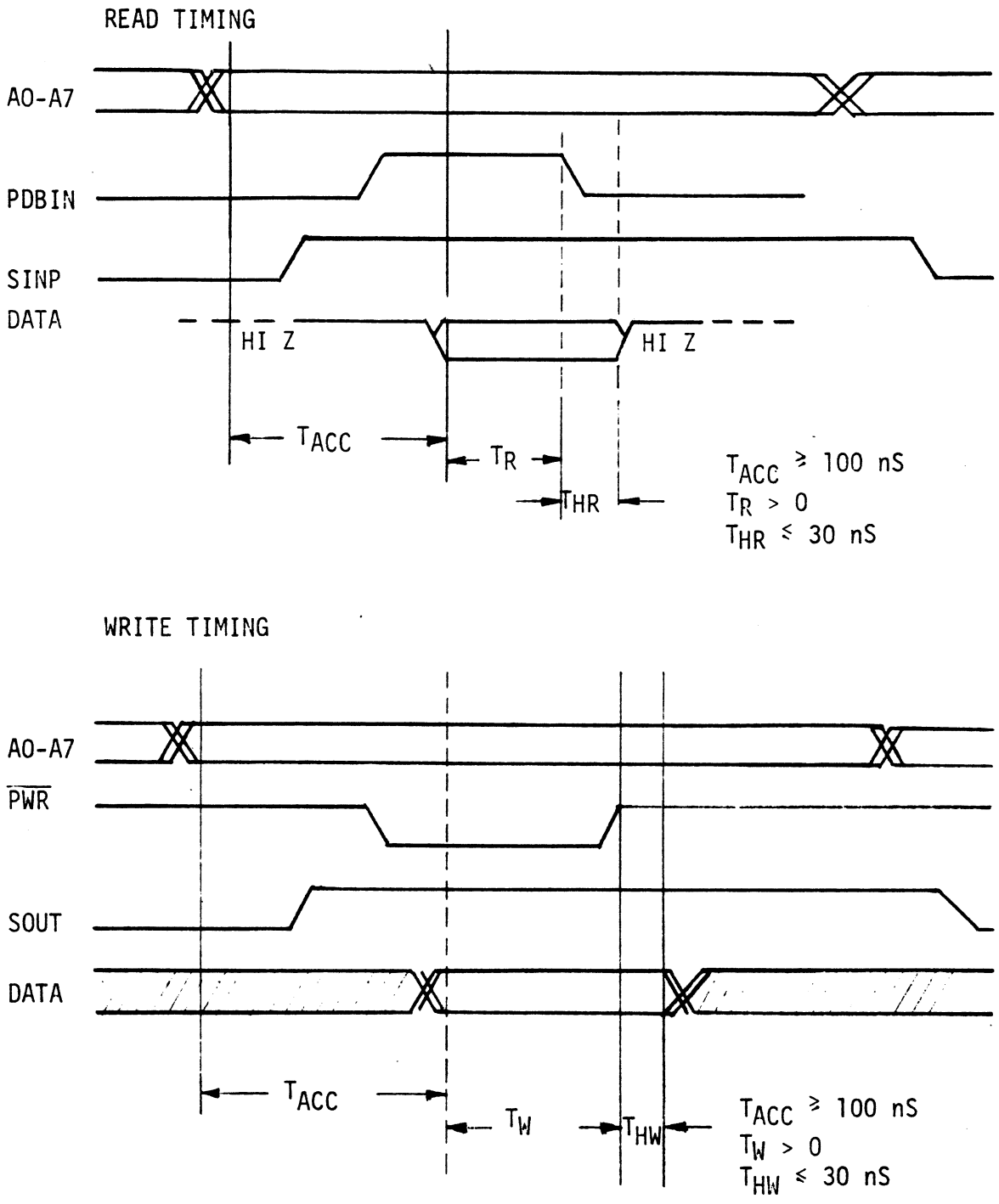
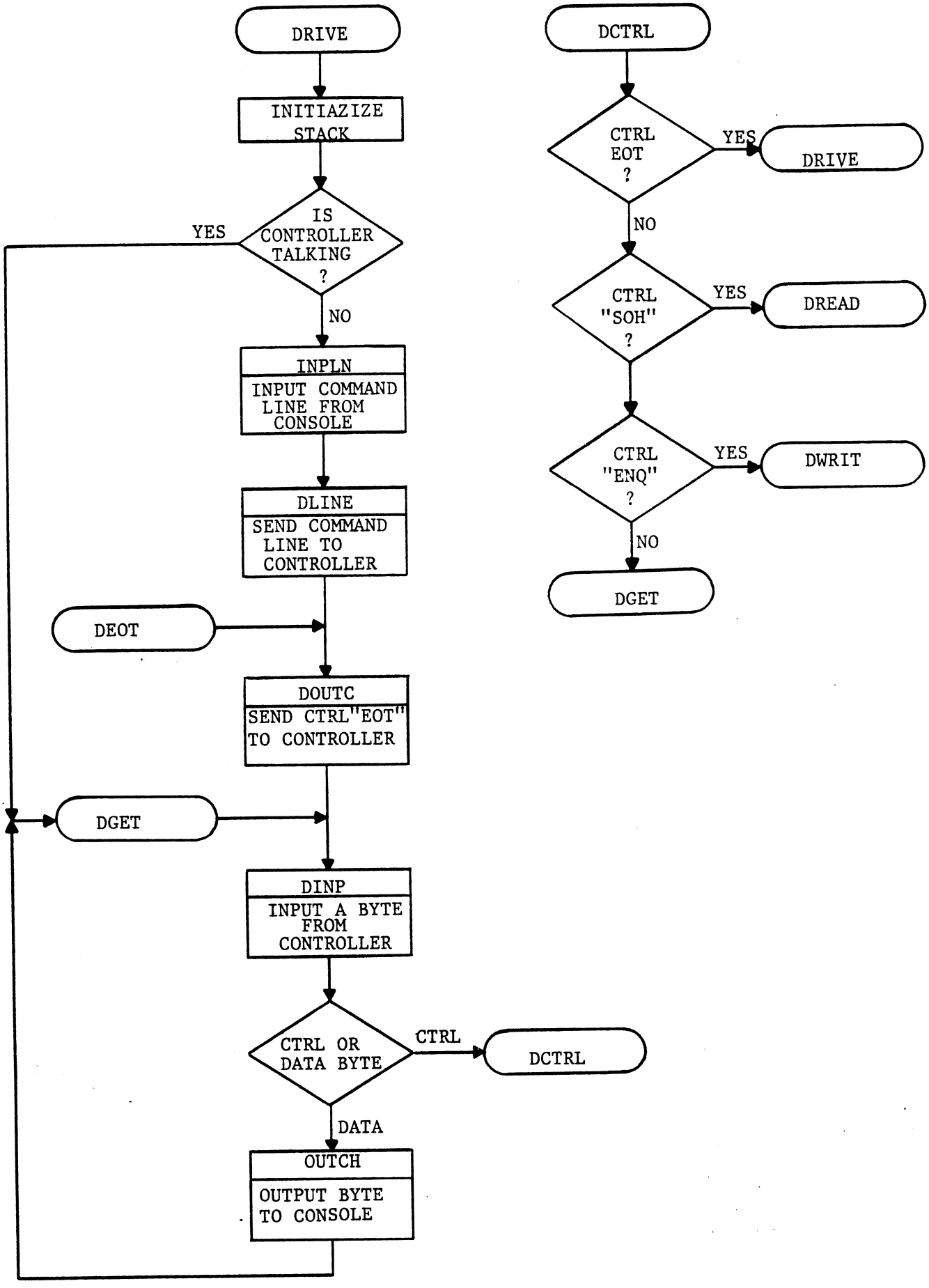


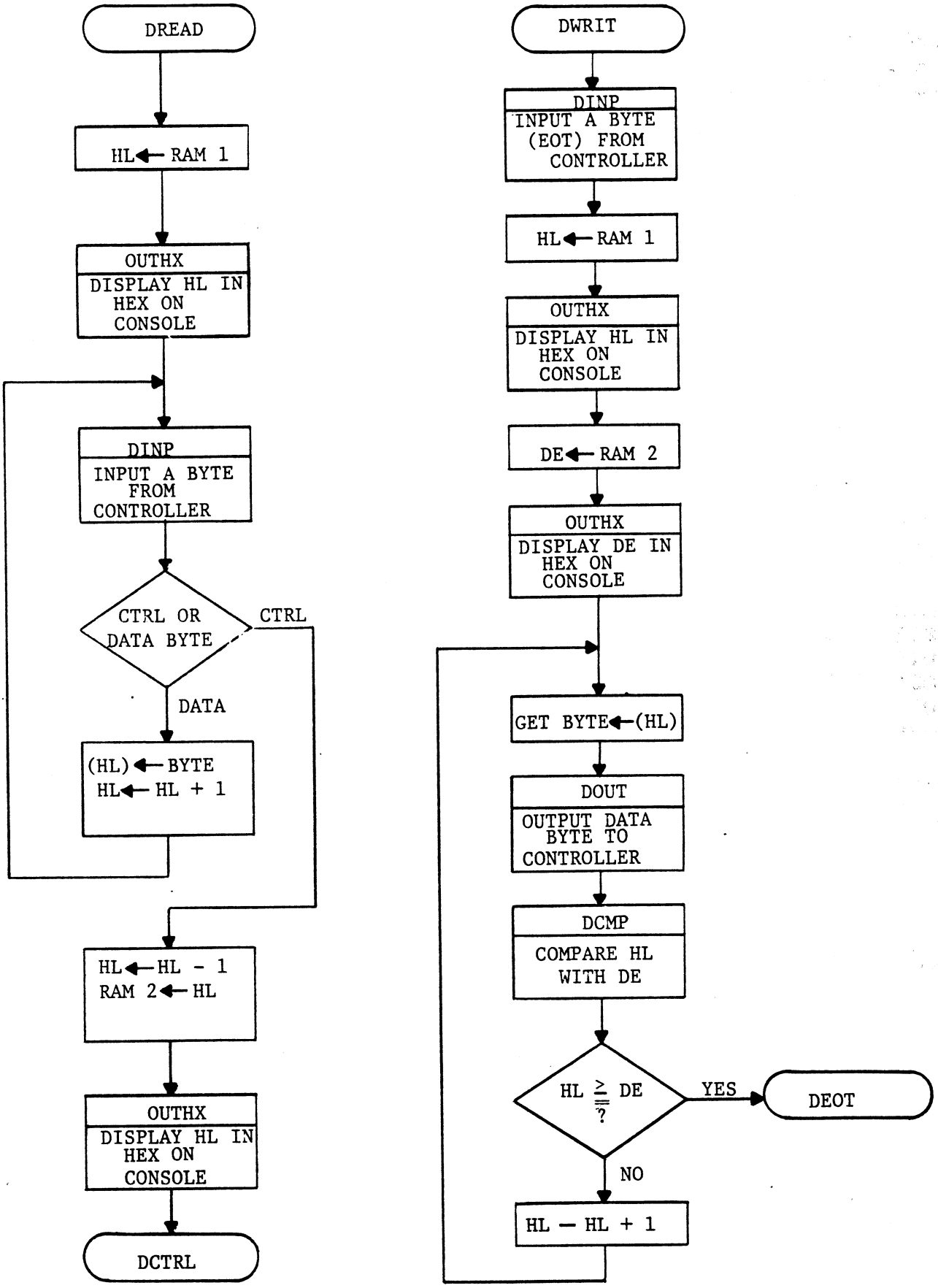
Figure 5.0 READ TIMING AND WRITE TIMING

APPENDIX A

Sample Driver Program Flowchart
Sample 8080 or Z80 Driver Program



SAMPLE DRIVE PROGRAM FLOWCHART



SAMPLE DRIVE PROGRAM FLOWCHART

```
                        ;
                        ; THIS IS THE BASIC DRIVER ROUTINE WHICH SENDS CONSOLE
                        ; COMMANDS TO THE CONTROLLER, CONTROLLER MESSAGES TO
                        ; THE CONSOLE, AND CONTROLS THE TRANSMISSION OF FILES
                        ; AND RECORDS BETWEEN THE CONTROLLER AND HOST COMPUTER
                        ;
                        ; THIS ROUTINE IS VIRTUALLY IDENTICAL TO THE DRIVER
                        ; FOR THE MODEL 1070 CONTROLLER. THE DIFFERENCES
                        ; ARE INDICATED.
                        ;
0000    31 0199         DRIVE:  LXI     SP,STACK        ;INITIALIZE STACK
0003    DBC1                    IN      DSTAT   ;GET 1170 STATUS
0005    E6C0                    ANI     0C0H    ;IS 1170 TALKING?
0007    C2 0015                 JNZ     DGET    ;YES, CLEAN UP
000A    CD 00A8                 CALL    INPLN   ;INPUT CONSOLE LINE
000D    CD 0072                 CALL    DLINE   ;SEND LINE TO 1170
0010    3E04            DEOT:   MVI     A,EOT   ;SEND 'EOT' TO 1170
0012    CD 008D                 CALL    DOUTC   ;AS CONTROL BYTE
0015    CD 007C         DGET:   CALL    DINP    ;INPUT BYTE FROM 1170
0018    DA 0021                 JC      DCTRL   ;CONTROL OR DATA BYTE?
001B    CD 014A                 CALL    OUTCH   ;DATA, SEND TO CONSOLE
001E    C3 0015                 JMP     DGET    ;NEXT BYTE
0021    FE04            DCTRL:  CPI     EOT     ;WHICH CONTROL CHAR?
0023    CA 0000                 JZ      DRIVE   ;'EOT', COMMAND IS DONE
0026    FE01                    CPI     SOH
0028    CA 0033                 JZ      DREAD   ;'SOH', READ 1170
002B    FE05                    CPI     ENQ     ;
002D    CA 0050                 JZ      DWRIT   ;'ENQ', WRITE TO 1170
0030    C3 0015                 JMP     DGET    ;ELSE IGNORE (ERROR)
                        ;
                        ;
                        ; THIS ROUTINE CONTROLS A DISK READ INTO RAM.
                        ;
0033    2A 00A4         DREAD:  LHLD    RAM1    ;GET RAM STARTING ADDR
0036    CD 0103                 CALL    OUTHX   ;DISPLAY ON CONSOLE
```

```
0039  CD 007C     DREAL:  CALL    DINP    ;INPUT BYTE FROM 1170
003C  DA 0044             JC      DREAX   ;CONTROL OR DATA BYTE?
003F  77                  MOV     M,A     ;DATA, STORE IN RAM
0040  23                  INX     H       ;INCREMENT RAM ADDR
0041  C3 0039             JMP     DREAL   ;NEXT BYTE
0044  F5           DREAX:  PUSH    PSW     ;CONTROL, SAVE BYTE
0045  2B                  DCX     H       ;DECREMENT RAM ADDR
0046  22 00A6             SHLD    RAM2    ;SAVE RAM ENDING ADDR
0049  CD 0103             CALL    OUTHX   ;DISPLAY ON CONSOLE
004C  F1                  POP     PSW     ;GET CONTROL BYTE
004D  C3 0021             JMP     DCTRL   ;ANALYZE IT
                    ;
                    ;
                    ; THIS ROUTINE CONTROLS A DISK WRITE FROM RAM.
                    ;
0050  CD 007C     DWRIT:  CALL    DINP    ;INPUT BYTE FROM 1170
0053  D2 0050             JNC     DWRIT   ;SHOULD BE AN 'EOT'
0056  2A 00A4             LHLD    RAM1    ;GET RAM STARTING ADDR
0059  CD 0103             CALL    OUTHX   ;DISPLAY ON CONSOLE
005C  EB                  XCHG
005D  2A 00A6             LHLD    RAM2    ;GET RAM ENDING ADDR
0060  CD 0103             CALL    OUTHX   ;DISPLAY ON CONSOLE
0063  EB                  XCHG            ;START IN HL, END IN DE
0064  7E           DWRIL:  MOV     A,M     ;GET BYTE FROM RAM
0065  CD 0087             CALL    DOUT    ;SEND DATA TO 1170
0068  CD 013A             CALL    DCMP    ;COMPARE ADDR TO END
006B  D2 0010             JNC     DEOT    ;AT END, SEND 'EOT'
006E  23                  INX     H       ;ELSE INCREMENT RAM ADDR
006F  C3 0064             JMP     DWRIL   ;DO NEXT BYTE
                    ;
                    ;
                    ; THIS ROUTINE SENDS A COMMAND TO THE CONTROLLER.
                    ;
0072  CD 0124     DLINE:  CALL    GETCH   ;GET CHAR FROM BUFFER
0075  D8                  RC              ;EXHAUSTED, ALL DONE
```

```
0076    CD 0087             CALL    DOUT    ;SEND CHAR TO 1170
0079    C3 0072             JMP     DLINE   ;CONTINUE
                    ;
                    ;
                    ; THIS ROUTINE INPUTS A BYTE FROM THE 1170 AND
                    ; SETS CARRY IF IT IS A CONTROL BYTE.
                    ;
007C    DBC1        DINP:   IN      DSTAT   ;GET 1170 STATUS BYTE
007E    E6C0                ANI     0C0H    ;RECEIVE DATA AVAILABLE?
0080    CA 007C             JZ      DINP    ;WAIT FOR IT
0083    17                  RAL             ;SET CARRY IF CONTROL
0084    DBC0                IN      DDATA   ;FETCH DATA BYTE
0086    C9                  RET             ;DONE
                    ;
                    ;
                    ; THIS ROUTINE OUTPUTS A DATA BYTE TO THE 1170.
                    ;
0087    CD 0093     DOUT:   CALL    DOUTW   ;WAIT UNTIL READY
008A    D3C0                OUT     DDATA   ;SEND TO 1170
008C    C9                  RET             ;DONE
                    ;
                    ;
                    ; THIS ROUTINE OUTPUTS A CONTROL BYTE TO THE 1170.
                    ;
008D    CD 0093     DOUTC:  CALL    DOUTW   ;WAIT UNTIL READY
0090    D3C1                OUT     DSTAT   ;WRITE 1170 STATUS BYTE
                    ;********************* DELETED "OUT  DDATA" INSTRUCTION
                    ;********************* FROM 1070 VERSION
0092    C9                  RET             ;DONE
                    ;
                    ;
                    ; THIS ROUTINE WAITS FOR THE 1170 TRANSMIT BUFFER TO
                    ; BE EMPTY AND READY FOR ANOTHER BYTE. IT ALSO
                    ; ARBITRATES IF THE 1170 AND HOST TRY TO TRANSMIT TO
                    ; ONE ANOTHER AT THE SAME TIME.
```

```
                        ;
0093    F5      DOUTW:  PUSH    PSW     ;SAVE BYTE TO SEND
0094    DBC1            IN      DSTAT   ;GET 1170 STATUS BYTE
0096    E6C0            ANI     0C0H    ;IS 1170 TRANSMITTING?
0098    C2 00A2         JNZ     DOUTX   ;YES, BREAK THE TIE
009B    DBC1            IN      DSTAT   ;GET 1170 STATUS AGAIN
009D    E602            ANI     02H     ;IS XMIT BUFFER EMPTY?
                        ;********************* THAT MASK DIFFERS FROM 1070 VERSION.
009F    CA 0094         JZ      DOUTW+1 ;NO, WAIT UNTIL IT IS
                        ;********************* THAT JUMP DIFFERS ALSO.
00A2    F1      DOUTX:  POP     PSW     ;RESTORE BYTE TO SEND
00A3    C9              RET             ;ALL DONE
                        ;
                        ;
                        ; SYMBOLIC EQUIVALENCES
                        ;
00C0                    DDATA   = 0C0H  ;CONTROLLER DATA PORT
00C1                    DSTAT   = 0C1H  ;CONTROLLER STATUS PORT
0004                    EOT     = 04H   ;ASCII 'EOT'
0001                    SOH     = 01H   ;ASCII 'SOH'
0005                    ENQ     = 05H   ;ASCII 'ENQ'
                        ;
                        ;
                        ; RAM WORKING STORAGE
                        ;
00A4    0000    RAM1:   .WORD   0       ;SAVE/LOAD START ADDR
00A6    0000    RAM2:   .WORD   0       ;SAVE/LOAD END ADDR
                        ;
                        ;
                        ;
                        ;
                        ; THIS ROUTINE INPUTS A LINE FROM THE CONSOLE DEVICE
                        ; INTO A RAM BUFFER, AND PROCESSES BACKSPACE AND
                        ; LINE-DELETE FUNCTIONS.
```

```
                        ;
00A8  CD 00F8   INPLN:  CALL CRLF       ;CR/LF TO CONSOLE
00AB  3E3E              MVI     A,'>'   ;GET COMMAND PROMPT
00AD  CD 014A           CALL    OUTCH   ;SEND TO CONSOLE
00B0  21 0156           LXI     H,IBUFF ;GET BUFFER ADDRESS
00B3  22 0176           SHLD    IBUFP   ;INITIALIZE POINTER
00B6  0E00              MVI     C,0     ;INITIALIZE COUNT
00B8  CD 0140   INPLI:  CALL    INPCH   ;GET CHAR FROM CONSOLE
00BB  E67F              ANI     7FH     ;STRIP PARITY BIT
00BD  FE20              CPI     ' '     ;TEST IF CONTROL CHAR
00BF  DA 00D2           JC      INPLC   ;YES, GO PROCESS
00C2  77                MOV     M,A     ;ELSE STORE IN BUFFER
00C3  3E20              MVI     A,32    ;GET BUFFER SIZE
00C5  B9                CMP     C       ;TEST IF FULL
00C6  CA 00B8           JZ      INPLI   ;YES, LOOP FOR CONTROL CHAR
00C9  7E                MOV     A,M     ;RECALL CHAR
00CA  23                INX     H       ;INCR POINTER
00CB  0C                INR     C       ;INCR COUNT
00CC  CD 014A   INPLE:  CALL    OUTCH   ;ECHO CHARACTER
00CF  C3 00B8           JMP     INPLI   ;GET NEXT CHAR
00D2  FE08      INPLC:  CPI     08H     ;TEST IF BACKSPACE
00D4  CA 00E9           JZ      INPLB   ;YES, KILL CHAR
00D7  FE1B              CPI     1BH     ;TEST IF ESCAPE
00D9  CA 00F3           JZ      INPLK   ;YES, KILL LINE
00DC  FE0D              CPI     0DH     ;TEST IF RETURN
00DE  C2 00B8           JNZ     INPLI   ;NO, IGNORE
00E1  79                MOV     A,C     ;GET COUNT
00E2  32 0178           STA     IBUFC   ;SAVE IT
00E5  CD 00F8           CALL    CRLF    ;SEND CR/LF TO CONSOLE
00E8  C9                RET             ;DONE
00E9  2B        INPLB:  DCX     H       ;DECREMENT POINTER
00EA  0D                DCR     C       ;DECREMENT COUNT
00EB  F2 00CC           JP      INPLE   ;IF NOT EMPTY, ECHO CHAR
00EE  23                INX     H       ;IF EMPTY, UNDO DECR
00EF  0C                INR     C       ;
```

```
00F0  C3 00B8            JMP     INPLI   ;GET NEXT CHARACTER
00F3  AF          INPLK: XRA     A       ;KILL LINE BY SETTING..
00F4  32 0178            STA     IBUFC   ;..COUNT TO ZERO
00F7  C9                 RET             ;DONE
                  ;
                  ;
                  ; THIS ROUTINE SENDS A CR LF SEQUENCE TO THE CONSOLE
                  ;
00F8  3E0D        CRLF:  MVI     A,0DH   ;GET A CR
00FA  CD 014A            CALL    OUTCH   ;DISPLAY IT
00FD  3E0A               MVI     A,0AH   ;GET A LF
00FF  CD 014A            CALL    OUTCH   ;DISPLAY
0102  C9                 RET             ;DONE
                         ;
                  ;
                  ;
                  ; THIS ROUTINE OUTPUTS THE CONTENTS OF THE H-L REGISTE
                  ; AS A FOUR-DIGIT HEXADECIMAL NUMBER ON THE CONSOLE.
                  ;
13  3E20        OUTHX:  MVI     A,' '   ;OUTPUT A SPACE
0105  CD 014A            CALL    OUTCH   ;
0108  7C                 MOV     A,H     ;GET TOP HALF OF WORD
0109  CD 010D           CALL    OUTH1   ;DISPLAY IN HEX
010C  7D                 MOV     A,L     ;SAME WITH BOTTOM HALF
010D  F5          OUTH1: PUSH    PSW     ;SAVE LOW-ORDER DIGIT
010E  1F                 RAR             ;SHIFT HIGH-ORDER DIGIT
010F  1F                 RAR
0110  1F                 RAR
0111  1F                 RAR
0112  CD 0116           CALL    OUTH    ;ADD BIAS AND DISLAY
0115  F1                 POP     PSW     ;GET OTHER DIGIT
0116  E60F        OUTH:  ANI     0FH     ;EXTRACT DIGIT
0118  C630              ADI     '0'     ;ADD ASCII ZONE BITS
011A  FE3A               CPI     '9'+1   ;TEST IF ALPHA CHAR
011C  DA 014A            JC      OUTCH   ;NO, DISPLAY AS-IS
```

```
011F  C607          ADI   'A'-'9'-1     ;ELSE ADD BIAS FOR A-F
0121  C3 014A       JMP   OUTCH   ;NOW DISPLAY IT
                    ;
                    ; THIS ROUTINE OBTAINS A CHARACTER FROM THE RAM BUFFER
                    ; AND SETS THE CARRY FLAG IF EXHAUSTED.
                    ;
0124  E5            GETCH: PUSH   H         ;SAVE REGS
0125  2A 0176       LHLD  IBUFP   ;GET POINTER
0128  3A 0178       LDA   IBUFC   ;GET COUNT
012B  D601          SUI   1       ;DECREMENT WITH CARRY
012D  DA 0138       JC    GETCX   ;BUFFER EMPTY
0130  32 0178       STA   IBUFC   ;REPLACE COUNT
0133  7E            MOV   A,M     ;GET CHARACTER
0134  23            INX   H         ;INCREMENT POINTER
0135  22 0176       SHLD  IBUFP   ;SAVE POINTER
0138  E1            GETCX: POP    H         ;RESTORE H-L
0139  C9            RET           ;DONE (CARRY IF NO CHAR)
                    ;
                    ;
                    ; THIS ROUTINE COMPARES D-E WITH H-L.
                    ;
013A  7C            DCMP:  MOV    A,H     ;GET MOST SIGNIF
013B  BA            CMP   D       ;COMPARE MSBYTE
013C  C0            RNZ           ;NON-ZERO, DONE
7D                  MOV   A,L     ;ELSE COMPARE LSBYTE
013E  BB            CMP   E       ;
013F  C9            RET           ;DONE
                    ;
                          ;
                    ;
                    ; THESE ROUTINES PERFORM I/O VIA THE SYSTEM CONSOLE
                    ; DEVICE, PASSING THE CHARACTER IN THE A REGISTER. THEY
                    ; MUST BE CODED TO WORK WITH THE PARTICULAR CONSOLE
                    ; I/O INTERFACE ARRANGEMENT OF EACH MICROCOMPUTER. THE
                    ; TWO ROUTINES MUST NOT MODIFY ANY REGISTERS OTHER THAN
```

```
                        ; THE A REGISTER.
                        ;
0140    DB00            INPCH:  IN      00      ;GET CONSOLE STATUS
0142    E601                    ANI     01H     ;TEST DATA WAITING
0144    C2 0140                 JNZ     INPCH   ;NO, WAIT FOR IT
0147    DB01                    IN      01      ;GET CONSOLE CHARACTER
0149    C9                      RET             ;ALL DONE
                        ;
                        ;
014A    F5              OUTCH:  PUSH    PSW     ;SAVE OUTGOING CHAR
014B    DB00                    IN      00      ;TEST TRANSMIT BUFFER EMPTY
014D    E680                    ANI     80H     ;
014F    C2 014B                 JNZ     OUTCH+1 ;WAIT FOR EMPTY
0152    F1                      POP     PSW     ;GET SAVED CHAR
0153    D301                    OUT     01      ;SEND TO CONSOLE
0155    C9                      RET             ;ALL DONE
                        ;
                        ;
                        ; RAM WORKING STORAGE
                        ;
0156                    IBUFF:  .BLKB   32      ;INPUT TEXT BUFFER
0176                    IBUFP:  .BLKB   2       ;INPUT POINTER
0178                    IBUFC:  .BLKB   1       ;INPUT COUNTER
0179                            .BLKB   32      ;STACK AREA
0199                    STACK   = .             ;TOP OF STACK
                        ;
                        ;
                                .END            ;END OF ASSEMBLY
```

+++++ SYMBOL TABLE +++++

CRLF	00F8	DCMP	013A	DCTRL	0021	DDATA	00C0
DEOT	0010	DGET	0015	DINP	007C	DLINE	0072
DOUT	0087	DOUTC	008D	DOUTW	0093	DOUTX	00A2
DREAD	0033	DREAL	0039	DREAX	0044	DRIVE	0000
DSTAT	00C1	DWRIL	0064	DWRIT	0050	ENQ	0005
EOT	0004	GETCH	0124	GETCX	0138	IBUFC	0178
IBUFF	0156	IBUFP	0176	INPCH	0140	INPLB	00E9
INPLC	00D2	INPLE	00CC	INPLI	00B8	INPLK	00F3
INPLN	00A8	OUTCH	014A	OUTH	0116	OUTH1	010D
OUTHX	0103	RAM1	00A4	RAM2	00A6	SOH	0001
STACK	0199						

APPENDIX B

Optional Sector Interleave Sequence
Sector Details

OPTIONAL SECTOR INTERLEAVE SEQUENCE

DISK RECORD SEQUENCES

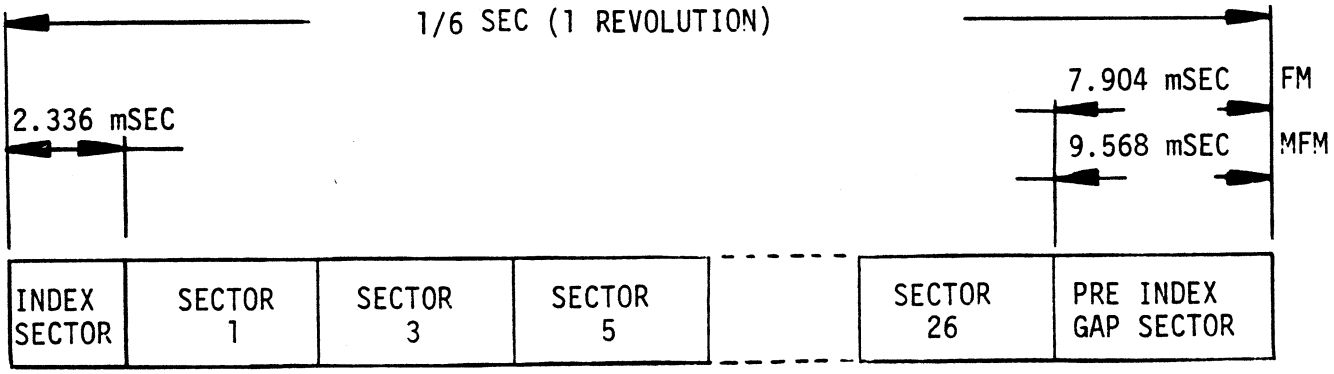
BLANK	01	02	03	04	05	06	07	08	09	10	11	12	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	3	4	5	6	7	8	9	10	11	12	13	14
3	3	5	7	9	11	13	15	17	19	21	23	25	2
4	4	7	10	13	16	19	22	25	2	2	2	2	15
5	5	9	13	17	21	25	2	2	11	12	13	14	3
6	6	11	16	21	26	2	9	10	20	22	24	26	16
7	7	13	19	25	2	8	16	18	3	3	3	3	4
8	8	15	22	2	7	14	23	26	12	13	14	15	17
9	9	17	25	6	12	20	3	3	21	23	25	4	5
10	10	19	2	10	17	26	10	11	4	4	4	16	18
11	11	21	5	14	22	3	17	19	13	14	15	5	6
12	12	23	8	18	3	9	24	4	22	24	26	17	19
13	13	25	11	22	8	15	4	12	5	5	5	6	7
14	14	2	14	26	13	21	11	20	14	15	16	18	20
15	15	4	17	3	18	4	18	5	23	25	6	7	8
16	16	6	20	7	23	10	25	13	6	6	17	19	21
17	17	8	23	11	4	16	5	21	15	16	7	8	9
18	18	10	26	15	9	22	12	6	24	26	18	20	22
19	19	12	3	19	14	5	19	14	7	7	8	9	10
20	20	14	6	23	19	11	26	22	16	17	19	21	23
21	21	16	9	4	24	17	6	7	25	8	9	10	11
22	22	18	12	8	5	23	13	15	8	18	20	22	24
23	23	20	15	12	10	6	20	23	17	9	10	11	12
24	24	22	18	16	15	12	7	8	26	19	21	23	25
25	25	24	21	20	20	18	14	16	9	10	11	12	13
26	26	26	24	24	25	24	21	24	18	20	22	24	26

Numbers at column top are interleave sequences specified by kill command.

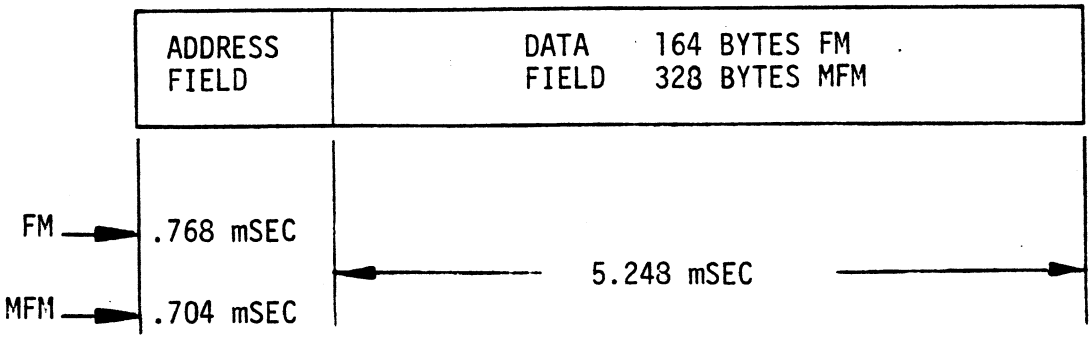
Columns show sector sequence for specified interleave.

N.B. "Blank" sequence changes index track only.

Sector Details



DISKETTE TRACK FORMAT (FM OR MFM)
INTERLEAVE SEQUENCE OF 02 SHOWN
FOR 26 SECTORS



SECTOR DETAILS

REVISIONS			200648 C
REV	DESCRIPTION	DATE	APPROVED
A	MFG RELEASE		
B	ECO 365		
C	ECO 381		

J3: A0 A1 A2 A3 A4 A5 A6 A7 B 2Φ GND DDF

J2: D0 D1 D2 D3 D4 D5 D6 D7

WAIT PB RESET CLK CARRY TORO NT RD

U5 74LS175 · U14 74LS175 · U6 74LS138 · U4 74LS299 · U41 74LS165 · U27 2708 · U28 74LS164 · U8 9401 · U3 Z80A-PIO · U32 74LS175 · U7 74LS00 · U13 7404 · U12 74LS74 · U15 74LS74 · U16 74LS10 · U17 74LS02 · U18 74LS00 · U29 74LS10 · U30 74LS136 · U31 74LS00 · U40 74LS136 · U42 74LS74 · U43 74LS74 · U44 74LS74 · U45 74LS74 · U19 74LS74

R1 1K · R2 1K · R3 1K · R4 1K · R5 1K · R40 1K · R45 1K

TP9

J1: 34 DIRECTION · 36 STEP · 40 WRITE GATE · 38 DWA · 26 DS1 · 28 DS2 · 30 DS3 · 18 DS4 · 32 DS5 · 2 HEAD LOAD · 16 EJECT · 14 RESTORE · 1 ALL ODD +NO. PINS GND. SHIELD · 49 · 22 READY · 10 SEEK COMPL · 20 INDEX · 6 TWO SIDE · 42 TROO · 44 WRITE PROT. · 48 SEP DATA · 50 SEP CLOCK · 46 READ DATA

U1 7406 · U11 7406 · Q11 2N4400 · R46 1K · R47 1K · VR2 100 IN747 · B A C D E F

U2 220/330

P1: 1 +8V UNREG · 51 · 2 +16V UNREG · 52 -16V UNREG · 50 · 100 GND

G1 LM323 · G2 7812 · G3 7905 · C9 C10 C11 .01 C12 C13 C14 C15 C16 C17

U23 7404 · U22 74LS74 · U21 74LS74 · U20 74LS74 · U35 74LS74 · U37 74LS136 · U38 74LS136 · U40 74LS136 · U24 74LS136 · U26 74LS136 · U39 74LS164 · U25 74LS164 · U36 74LS00 · U34 74LS02 · U9 1458 · U33 1458 · U10 LM306

R6 1K · R7 1K · R8 1K · R9 10K · R10 1K · R39 1K · C1 100PF

TP1 TP2 TP3 TP4

R18 4.3K · R19 2K · R20 1K · R21 180 · R22 100 · R43 100 · R23 10K · R41 3.9 MEG · R12 100 · R13 150 · R15 510 · R16 150 · R17 27 · R11 1K · R14 1K · VR1 IN751 · C2 · C3 · C4 100 PF · C8

Q1 2N4400 · Q2 2N4402 · Q3 2N4402 · Q4 2N4400 · Q5 2N4400 · Q6 2N5460 · Q7 2N4400 · Q8 2N4402 · Q9 2N4402 · Q10 2N4400

CR1 IN3064 · CR2 IN3064 · CR5 IN3064 · CR7 IN3064 · CR8 IN3064

R24 22K · R25 3.3K · R26 1K · R27 1.2K · R28 1K · R29 2.2K · R30 1K · R31 5.1K · R32 2K · R33 1K · R34 1.5K · R35 5.1K · R36 1.5K · R37 1K · R38 1K · R42 10K · C5 .022 · C6 510PF · C7 100PF

TP7 TP8 TP10 TP11 TP12

(NOTES CONTINUED ON SHT 2)
2 ALL CAPACITORS IN MICROFARADS.
1. ALL RESISTORS IN OHMS, 5%, 1/4 W.
NOTES: UNLESS SPECIFIED:

DISC PORT (PCB ASSY 200647)

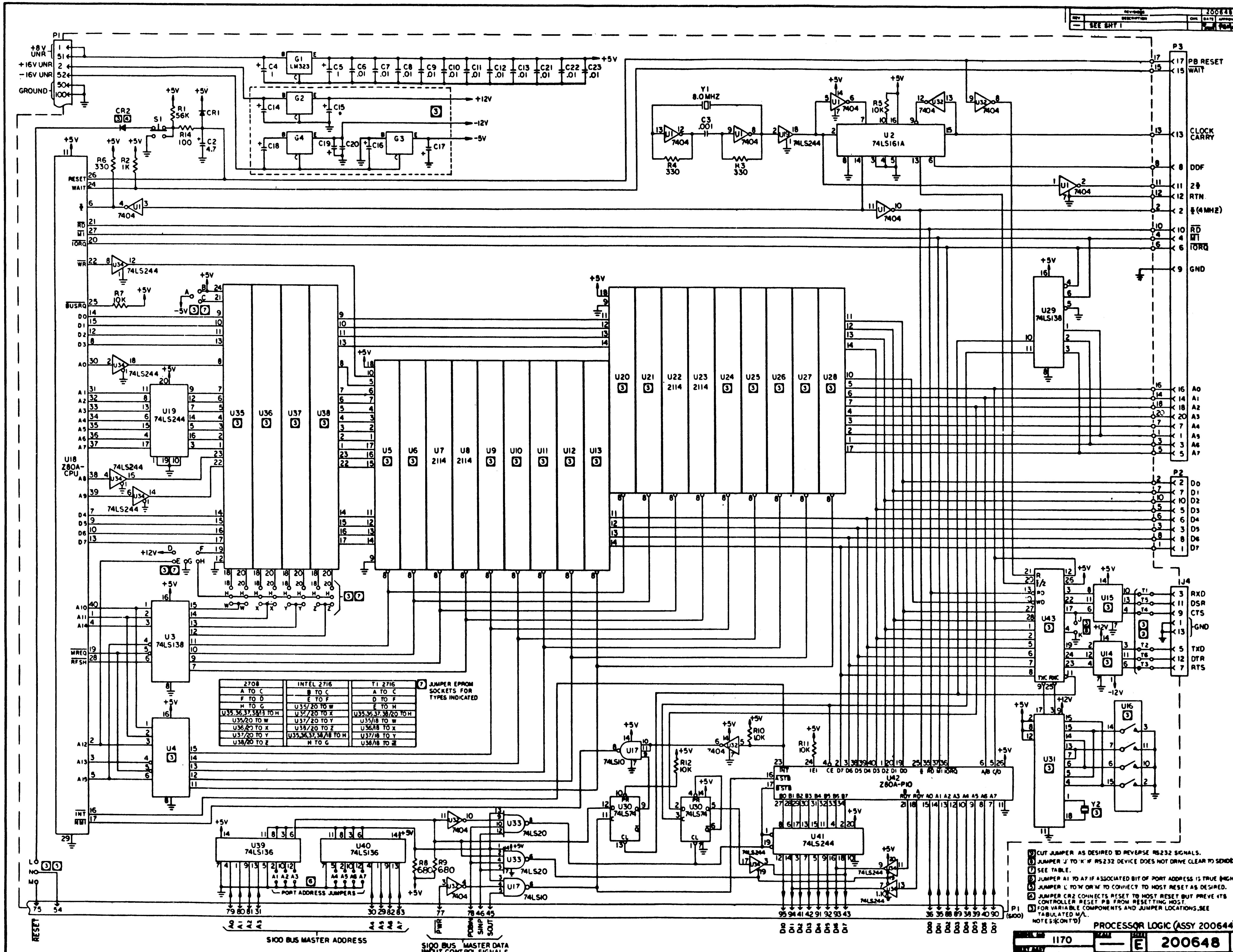
PerSci Incorporated

SCHEMATIC, PCB, COMPOSITE, DUAL DENSITY CONTROLLER

MODEL NO. 1170 · SIZE E · 200648 C

NEXT ASSY 200644/200647

200648 C1



2708	INTEL 2716	TI 2716
A TO C	B TO C	A TO C
F TO D	E TO F	D TO F
H TO G	U35/20 TO W	E TO H
U35,36,37,38/13 TO H	U36/20 TO X	U35,36,37,38/20 TO H
U35/20 TO W	U37/20 TO Y	U35/18 TO W
U36/20 TO X	U38/20 TO Z	U36/18 TO X
U37/20 TO Y	U35,36,37,38/18 TO H	U37/18 TO Y
U38/20 TO Z	H TO G	U38/18 TO Z

(7) JUMPER EPROM SOCKETS FOR TYPES INDICATED

(9) CUT JUMPER AS DESIRED TO REVERSE RS232 SIGNALS.
(8) JUMPER 'J' TO 'K' IF RS232 DEVICE DOES NOT DRIVE CLEAR TO SEND(CTS).
(7) SEE TABLE.
(6) JUMPER A1 TO A7 IF ASSOCIATED BIT OF PORT ADDRESS IS TRUE (HIGH).
(5) JUMPER 'L' TO 'N' OR 'M' TO CONNECT TO HOST RESET AS DESIRED.
(4) JUMPER CR2 CONNECTS RESET TO HOST RESET BUT PREVENTS CONTROLLER RESET PB FROM RESETTING HOST.
(3) FOR VARIABLE COMPONENTS AND JUMPER LOCATIONS, SEE TABULATED M/L.

NOTES:(CONT'D)

PROCESSOR LOGIC (ASSY 200644)

1170	—	E	200648	C
200644/200647			SHT 2 OF 2	

200648 C

PERSCI, INC.

12210 Nebraska Ave
W Los Angeles
CA 90025