PET COMMUNICATION WITH THE OUTSIDE WORLD

commodore

PET™
2001 Series

personal computer

Introduction

This supplement contains general background information on the PET interfaces and lines to the "outside world."

This information is relevant to users who wish to interface with any device external to the PET computer itself. The device could be a printer, a source of digital data such as a seismograph or other scientific instrument, a second cassette tape deck or an extra memory to increase the power of the PET still further.

This document is intended to provide essential information for the experienced user anxious to explore the interface possibilities of the PET.

PET INTERFACES AND LINES 3

COMMANDS AND OPERATIONS FOR PERIPHERAL DEVICES 15

THE IEEE-488 BUS— A SHORT DESCRIPTION 43

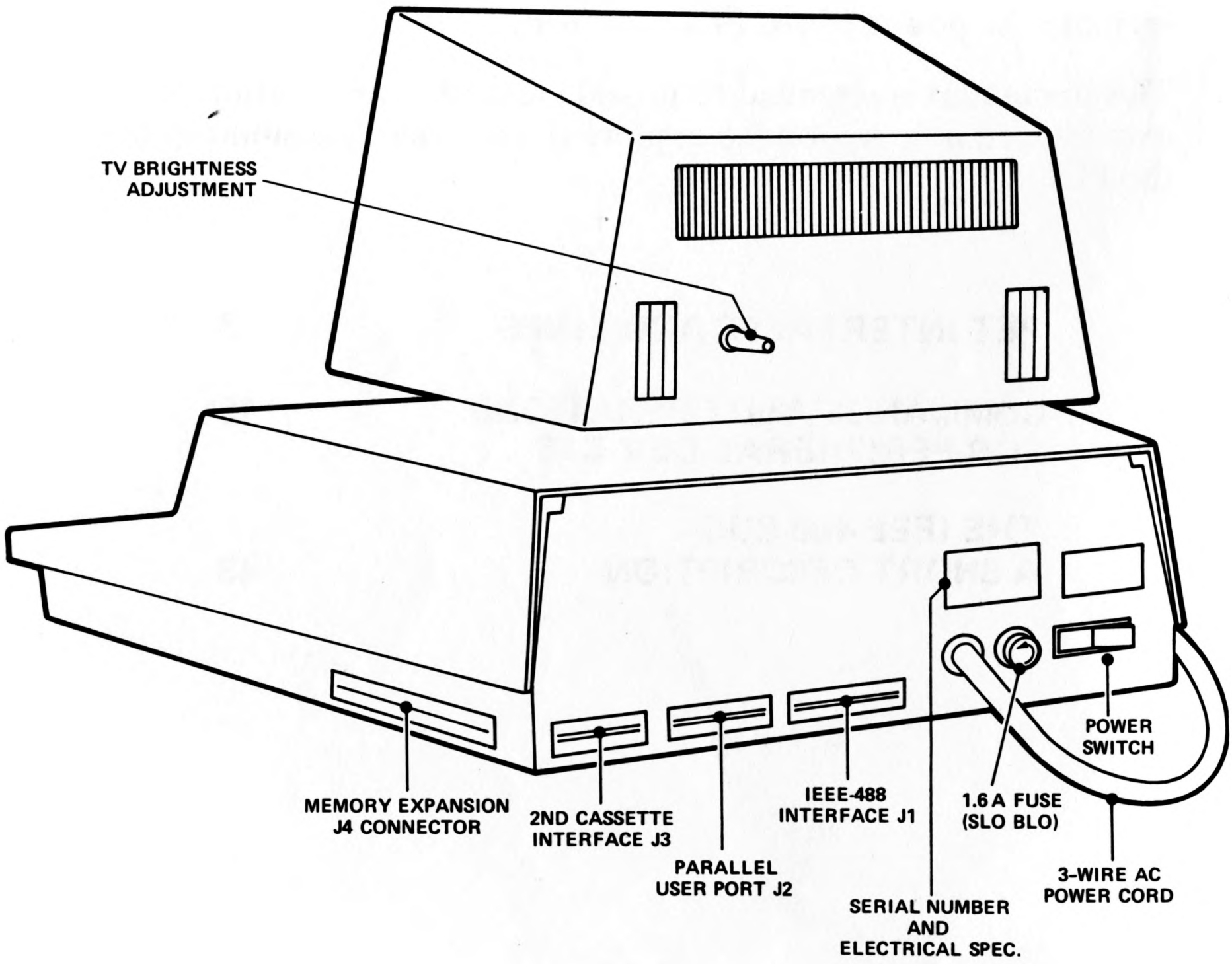


Figure 1-1. Rear view of PET 2001 showing switch, fuse, line cord and interfacing connectors

PET Interfaces and Lines

TABLE OF CONTENTS

1.	**Connector Orientation**	4
2.	**IEEE-488 Interfaces (Connector J1)**	5
	2.1 Receptacles for the IEEE-Interface	6
	2.2 IEEE-488 Connectors	6
3.	**Parallel User Port (Connector J2)**	7
	3.1 Versatile Interface Adapter	8
	3.2 Programming the User Port	10
4.	**Second Cassette Interface (Connector J3)**	10
5.	**Memory Expansion Connector (Connector J4)**	12

1. Connector Orientation

As indicated in Figure 1-1, there are four connectors provided, accessible through slots in the rear and side of the PET that enable the user to interface the computer with external devices.

As outlined in Figure 1-2, edge card connectors are utilized which are in fact, direct extensions of the PET main logic assembly board itself. There are two contacts to each position of the connector. The contact identification convention for J1 and J2 is also illustrated in Figure 1-2.

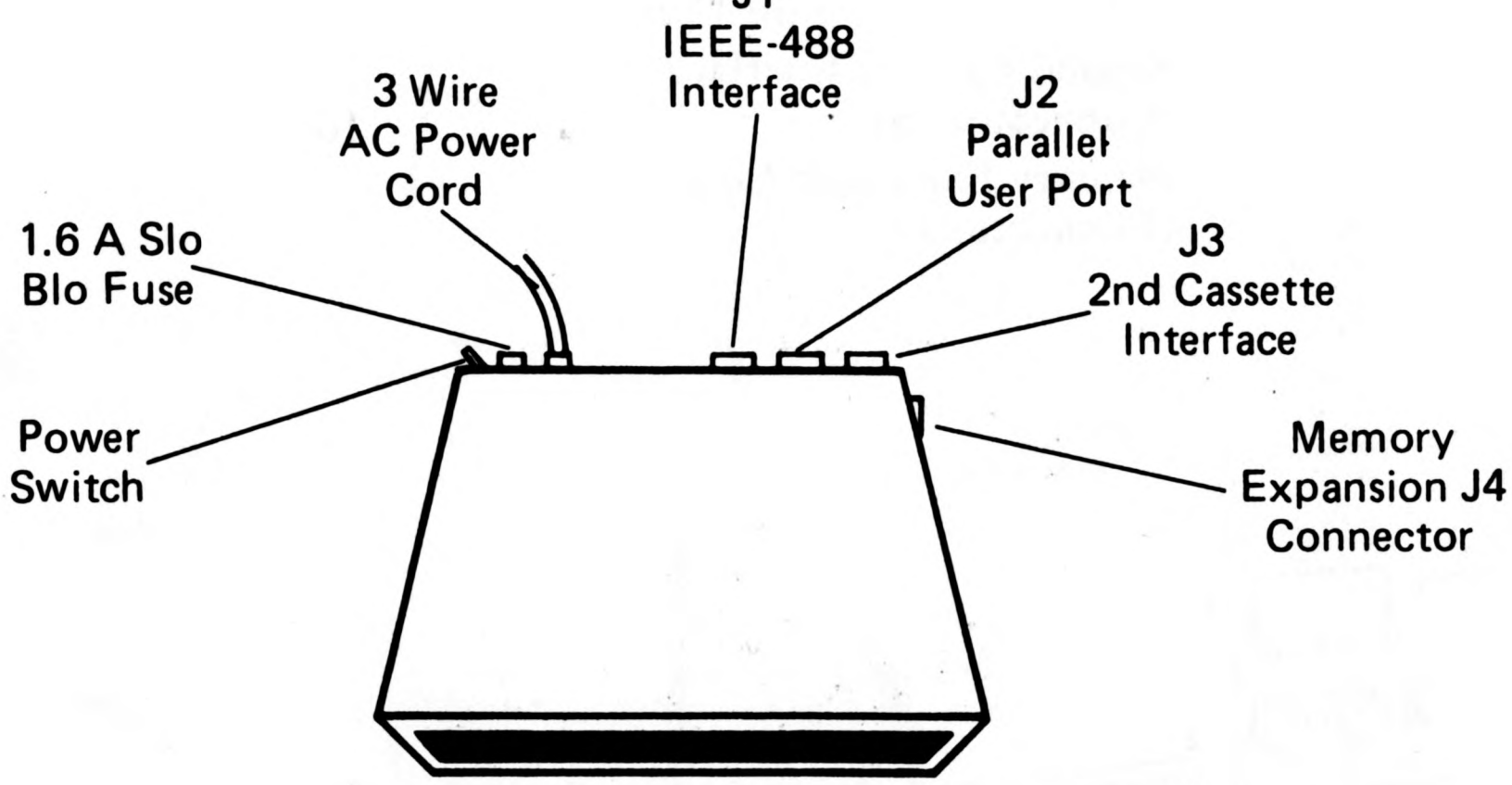


Figure 1-1. Simplified top view of PET showing switch, fuse, line cord and interfacing connectors.

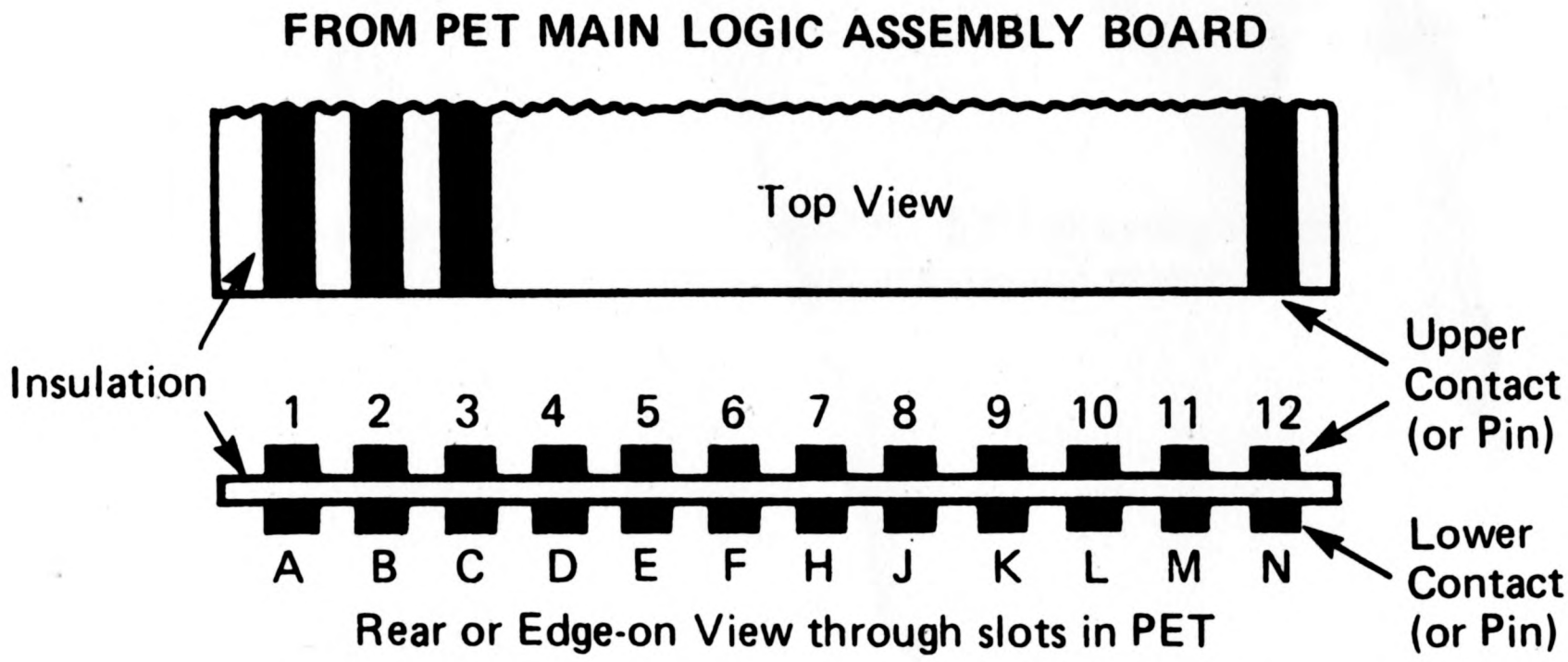


Figure 1-2. Simplified views of edge connectors J1 and J2 to illustrate contact identification convention.

2. IEEE-488 Interfaces (Connector J1)

The standard IEEE-488 connector is not used on the PET. Instead, a standard 12 position, 24 contact edge connector with .156 inch spacing between contact centers is provided. This permits it to be compatible with all of the other connections to the PET.

Keying slots are located between pins 2-3 and 9-10.

Table 2-1 shows the PET contact identification characters, the connection for a standard IEEE connector, the IEEE mnemonics and the signal definitions.

Electrical drive capability and line impedance matching is in accordance with IEEE-488 specifications.

Table 2-1. PET contact identification characters. IEEE-488 identification characters, associated labels and descriptions.

Blue Connector

PET Pin Characters	Standard IEEE Connector Pin Numbers	IEEE Signal Mnemonic	Signal Definition/Label
Upper Pins			
1	1	DI01	Data input/output line #1
2	2	DI02	Data input/output line #2
3	3	DI03	Data input/output line #3
4	4	DI04	Data input/output line #4
10	5	EOI	End or identify
11	6	DAV	Data valid
12	7	NRFD	Not ready for data
13	8	NDAC	Data not accepted
14	9	IFC	Interface clear
	10	SRQ	Service request Not Implement
15	11	ATN	Attention
16	12	GND	Chassis ground and IEEE cable shield drain wire
Lower Pins			
5	13	DI05	Data input/output line #5
6	14	DI06	Data input/output line #6
7	15	DI07	Data input/output line #7
8	16	DI08	Data input/output line #8
	17	REN	Remote enable (Grounded in Pet)
9	18	GND	DAV ground

Table 2-1 continued on next page.

Table 2-1. PET contact identification characters (continued)

PET Pin Characters	Standard IEEE Connector Pin Numbers	IEEE Signal Mnemonic	Signal Definition/Label
Lower Pins			
H	19	GND	NRFD ground
J	20	GND	NDAC ground
K	21	GND	IFC ground
L	22	GND	SRQ ground
M	23	GND	ATN ground
N	24	GND	Data ground (DI01-8)

2.1 Receptacles for the IEEE Interface

A list of frequently used 12 position, 24 contact receptacles that are suitable for connection to the PET edge card connector J1 and J2 is shown here:

Table 2-2. Receptacles recommended for PET IEEE-488 connectors or parallel user port.

Manufacturer	Part Number
Cinch	251-12-90-160
Sylvania	6AG01-12-1A1-01
Amp	530657-3
Amp	530658-3
Amp	530654-3

2.2 IEEE-488 Connectors

The IEEE-488 standard receptacles are *not* directly connectable to the PET edge connector; some of these are shown in Table 2-3, and belong to the Cinch Series 57 or Champ Series (Amphenol). Also shown are their matching plugs.

Table 2-3. IEEE standard connectors

Connector Manufacturer	Identifier	Description
Cinch	5710240	Solder-plug
Cinch	5720240	Solder-receptacle
Amp	552301-1	Insulation displacement plug
Amp	552305-1	Insulation displacement receptacle

Commodore has available a 1 meter long IEEE-488 dual connector-PET edge connector, cable. Please contact your local dealer or Commodore for price and delivery.

3. Parallel User Port (Connector J2)

The lines for this interface are brought out from the PET main logic board to a 12 position, 24 contact edge connector with .156 inch spacing between contact centers. See Table 2-2 for suitable mating connectors.

Keying slots are located between pins 1-2 and 10-11.

Table 3-1 shows the PET pin identification characters, the corresponding labels and their descriptions.

Note that the connections 1-12, the top line of contacts (see Figure 1-2), are primarily intended for use by the PET service department or qualified dealers. When using the incorporated ROM diagnostic, a special connector is used; this jumpers some of the top contacts to the bottom contacts. ***It is strongly advised that the top connectors 1-12 be used only with extreme caution.***

Table 3-1. Parallel user port information.
PET pin identification characters, the corresponding signal labels and their descriptions.

Pin Identification Character	Signal Label	Signal Description
1	Ground	Digital ground.
2	T.V. Video	Video output used for external display, used in diagnostic routine for verifying the video circuit to the display board.
3	IEEE-SRQ	Direct connection to the SRQ signal on the IEEE-488 port. It is used in verifying operation of the SRQ in the diagnostic routine.
4	IEEE-EOI	Direct connection to the EOI signal on the IEEE-488 port. It is used in verifying operation of the EOI in the diagnostic routine.
5	Diagnostic Sense	When this pin is held low during power up the PET software jumps to the diagnostic routine, rather than the BASIC routine.

Table 3-1 continued on next page.

Table 3-1. Parallel user port information (continued).

Pin Identification Character	Signal Label	Signal Description
6	Tape #1 READ	Used with the diagnostic routine to verify cassette tape #1 read function.
7	Tape #2 READ	Used with the diagnostic routine to verify cassette tape #2 read function.
8	Tape Write	Used with the diagnostic routine to verify operation of the WRITE function of both cassette ports.
9	T.V. Vertical	T.V. vertical sync signal verified in diagnostic. May be used for external TV display.
10	T.V. Horizontal	T.V. horizontal signal verified in diagnostic may be used for TV display.
11, 12	GND	Digital ground.
A	GND	Digital ground.
B	CA1	Standard edge sensitive input of 6522VIA.
C	PA0	Input/output lines to peripherals, and can be programmed independently of each other for input or output.
D	PA1	
E	PA2	
F	PA3	
H	PA4	
J	PA5	
K	PA6	
L	PA7	
M	CB2	Special I/O pin of VIA.
N	GND	Digital ground.

3.1 Versatile Interface Adapter

The lines on the bottom side of the user port connector originate from a Versatile Interface Adapter (VIA MOS Technology part #6522).

The signals CA1, PA∅-7 and CB2 are directly connected to a standard

6522 VIA located at hexadecimal address E84Ø. (Decimal address 59456).

The parallel port consists of eight programmable bi-directional I/O lines PAØ-7, an input handshake line for the eight lines, CA1, which can also be used for other edge-sensitive inputs and a very powerful connection, CB2. This has most of the abilities of CA1, but can also act as the input or output of the VIA shift register.

A detailed specification for the VIA is attached. All signals on the VIA that are not connected to the user port are utilized by the PET for internal controls. Please note that the user should avoid interfacing with these signals in any way.

Table 3-2 shows the decimal and hexadecimal addresses in the PET associated with the VIA.

Table 3-2. VIA 6522 Decimal and Hexadecimal addresses in PET.

Decimal	Hexa-Decimal	$E840+	Addressed Location
59456	E84Ø	ØØØØ	Output register for I/O port B.
59457	E841	ØØØ1	Output register for I/O port A with handshaking.
59458	E842	ØØ1Ø	I/O Port B Data Direction register.
59459	E843	ØØ11	I/O Port A Data Direction register.
59460	E844	Ø1ØØ	Read Timer 1 Counter low order byte Write to Timer 1 Latch low order byte.
59461	E845	Ø1Ø1	Read Timer 1 Counter high order byte. Write to Timer 1 Latch high order byte and initiate count.
59462	E846	Ø11Ø	Access Timer 1 Latch low order byte.
59463	E847	Ø111	Access Timer 1 Latch high order byte.
59464	E848	1ØØØ	Read low order byte of Timer 2 and reset Counter interrupt. Write to low order byte of Timer 2 but do not reset interrupt.

Table 3-2 continued on next page

Table 3-2. VIA 6522 Decimal and Hexadecimal addresses in PET (continued).

Decimal	Hexa-Decimal	$E840+	Addressed Location
59465	E849	1ØØ1	Access high order byte of Timer 2; reset Counter interrupt on write.
59466	E84A	1Ø1Ø	Serial I/O Shift register.
59467	E84B	1Ø11	Auxiliary Control register.
59468	E84C	11ØØ	Peripheral Control register.
59469	E84D	11Ø1	Interrupt Flag register (IFR).
5947Ø	E84E	111Ø	Interrupt Enable register.
59471	E84F	1111	Output register for I/O Port A, without handshaking.

3.2 Programming the User Port

Data lines PAØ-7 are individually programmed to function for input or output as required. This is done by using a software POKE 59459 command to place a number into the data direction register. This number must contain zeros on bits corresponding to lines where inputs are required and ones where outputs are required. Table 4-3 shows a practical example of input/output selection.

The programming need only be carried out at the beginning. From then on POKE 59471 can be used to drive the pins programmed as outputs, and PEEK(59471) will read all the inputs.

Table 4-3. Parallel user port example.
Programming of lines PAØ-7 for input/output operation.

Command Statement	Binary Representation	Lines	Mode
POKE 59459,255	11111111	PAØ-7	Output
POKE 59459,Ø	ØØØØØØØØ	PAØ-7	Input
POKE 59459,24Ø	11110000	PAØ-3	Input
		PA4-7	Output

4. Second Cassette Interface (Connector J3)

This interface is brought out from the PET main logic board to a 6 position, 12 contact edge connector with .156 inch spacing between contact centers (See Figure 4-1).

A keying slot is located between pins 2-3.

This port is intended for use with the Commodore second cassette system only. Any other connections are made at the risk of the user. Please note that the +5 volts is *not* intended for use as an external power supply.

Table 4-1 shows the PET pin identification characters, labels and descriptions. Table 4-2 shows some typical receptacles that are suitable for the second cassette connector.

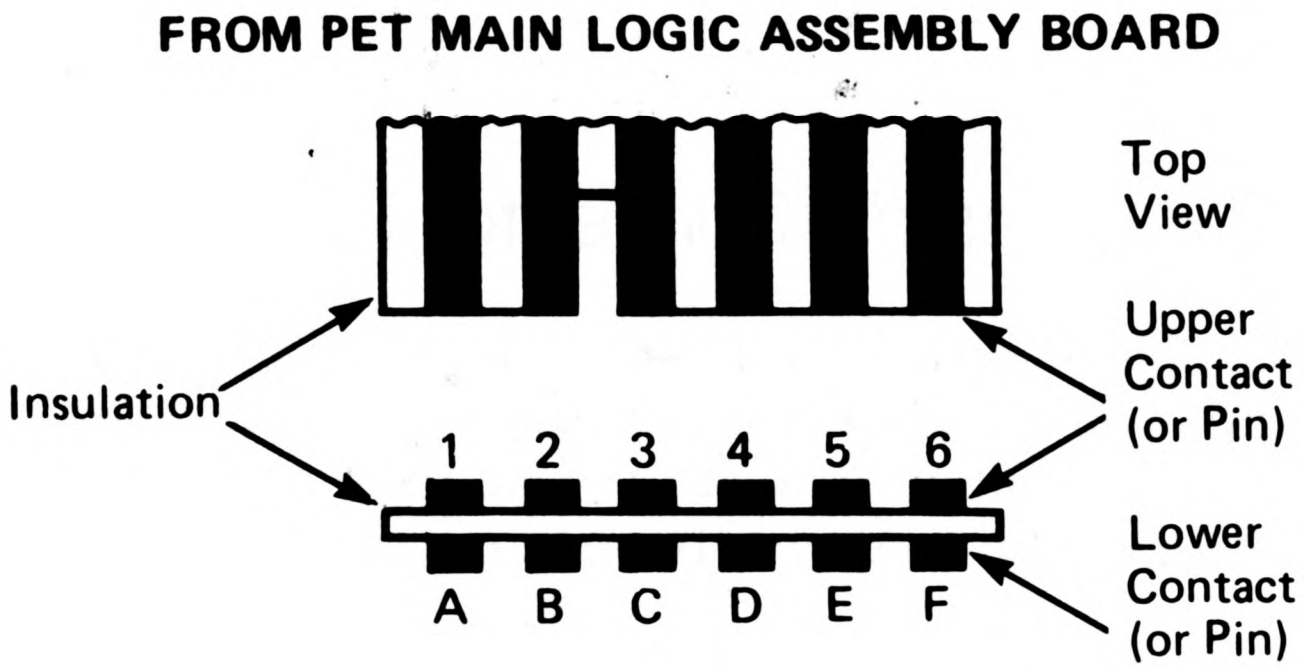


Figure 4-1. Simplified view of edge connector J3 with contact identification.

Table 4-1. Second cassette interface port.
PET pin identification characters, labels and associated descriptions.
Note A-1, B-2, etc., imply a pin A to pin1, pin B to pin 2, connection.
In some special units, pins 1 through 6 were not connected.

Pin Identification Characters	Label	Description
A-1	GND	Digital ground.
B-2	+5	Positive 5 volts to operate cassette circuitry only.
C-3	Motor	Computer controlled positive 6 volts for cassette motor.
D-4	Read	Read line from cassette.
E-5	Write	Write line to cassette.
F-6	Sense	Monitors closure of mechanical switch on cassette when any button is pressed.

Table 4-2. A selection of suitable receptacles for connecting with the PET second cassette edge connector J3.

Manufacturer	Identifier
Sylvania	6AJØ7-6-1A1-Ø1
Viking	2KH6/1AB5
Viking	2KH6/9AB5
Viking	2KH6/21AB5
Amp	53Ø692-1
Sullins	ESM6-SREH
Cinch	25Ø-Ø6-9Ø-17Ø

5. Memory Expansion Connector (Connector J4)

The memory expansion connector provides access to the buffered and decoded input/output lines from the 6502 microprocessor. Figure 6-1 shows a simplified view of the 40 position–80 contact edge connector used. The spacing between contact centers is 0.1 inch.

Note that the 40 top edge "B" connections (or pins) are ground returns for the corresponding 40 lower edge "A" connections.

Table 5-1 shows the PET pin numbers, line labels and line descriptions.

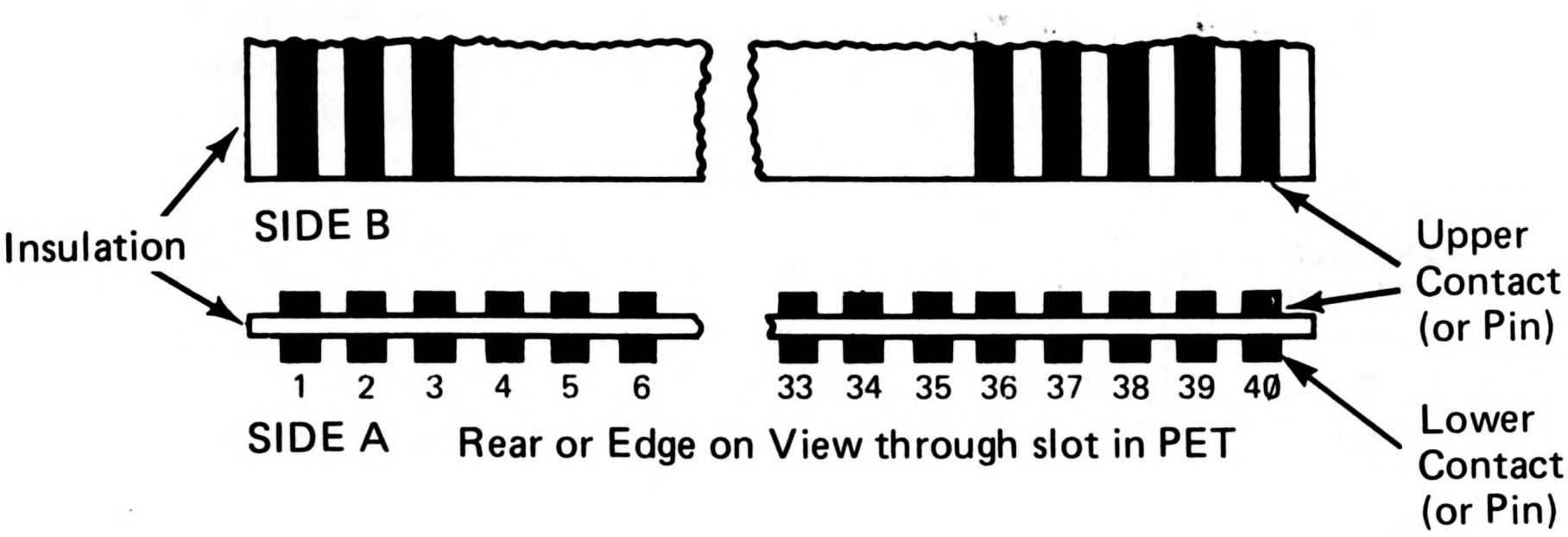


Figure 5-1. Simplified view of edge connector J4 with contact identification. All side B contacts grounded.

Table 5-1. Memory expansion connector. PET pin numbers. Line labels and line descriptions.

Side A Pin Numbers	Line Labels	Line Description
A1	BAØ	Address bit Ø, used for memory expansion. Buffered.
A2	BA1	Address bit 1, used for memory expansion. Buffered.
A3	BA2	Address bit 2, used for memory expansion. Buffered.
A4	BA3	Address bit 3, used for memory expansion. Buffered.
A5	BA4	Address bit 4, used for memory expansion. Buffered.
A6	BA5	Address bit 5, used for memory expansion. Buffered.
A7	BA6	Address bit 6, used for memory expansion. Buffered.
A8	BA7	Address bit 7, used for memory expansion. Buffered.
A9	BA8	Address bit 8, used for memory expansion. Buffered.
A1Ø	BA9	Address bit 9, used for memory expansion. Buffered.
A11	BA1Ø	Address bit 1Ø, used for memory expansion. Buffered.
A12	BA11	Address bit 11, used for memory expansion. Buffered.
A13	NC	No connection.
A14	NC	No connection.
A15	NC	No connection.
A16	$\overline{\text{SEL 1}}$	4K byte page address select for memory locations 1ØØØ-1FFF.
A17	$\overline{\text{SEL 2}}$	4K byte page address select for memory locations 2ØØØ-2FFF.
A18	$\overline{\text{SEL 3}}$	4K byte page address select for memory locations 3ØØØ-3FFF.
A19	$\overline{\text{SEL 4}}$	4K byte page address select for memory locations 4ØØØ-4FFF.
A2Ø	$\overline{\text{SEL 5}}$	4K byte page address select for memory locations 5ØØØ-5FFF.
A21	$\overline{\text{SEL 6}}$	4K byte page address select for memory locations 6ØØØ-6FFF.

Table 5-1 continued on next page.

Table 5-1. Memory expansion connector. PET pin numbers. Line labels and line descriptions (continued).

Side A Pin Numbers	Line Labels	Line Description
A22	$\overline{\text{SEL 7}}$	4K byte page address select for memory locations 7ØØØ-7FFF.
A23	$\overline{\text{SEL 9}}$	4K byte page address select for memory locations 9ØØØ-9FFF.
A24	$\overline{\text{SEL A}}$	4K byte page address select for memory locations AØØØ-AFFF.
A25	$\overline{\text{SEL B}}$	4K byte page address select for memory locations BØØØ-BFFF.
A26	NC	No connection.
A27	$\overline{\text{RES}}$	Reset for 6502 microprocessor. Note: connected to 74LS00 output.
A28	$\overline{\text{IRQ}}$	Interrupt request line to the microprocessor.
A29	BØ2	Buffered phase 2 clock.
AØ	R/W	Buffered read/write from 6502 microprocessor.
A31	NC	No connection.
A32	NC	No connection.
A33	BDØ	Data bit Ø. Buffered.
A34	BD1	Data bit 1. Buffered.
A35	BD2	Data bit 2. Buffered.
A36	BD3	Data bit 3. Buffered.
A37	BD4	Data bit 4. Buffered.
A38	BD5	Data bit 5. Buffered.
A39	BD6	Data bit 6. Buffered.
AØ	BD7	Data bit 7. Buffered.

Commands and Operations for Peripheral Devices

TABLE OF CONTENTS

1. **Additional BASIC Commands** 17

2. **Input/Output Command Parameters** 17
   - 2.1 Logical Files 17
   - 2.2 Device Numbers 18
   - 2.3 Secondary Addresses 19
   - 2.4 File Names 20

3. **Tape Cassette Operation for Files** 20
   - 3.1 File Recording Technique 21
   - 3.2 File Headers 21
   - 3.3 Tape Buffers 22
   - 3.4 Multiple Files 22

4. **Logical File I/O Operations: General** 23

5. **Opening Files** 24
   - 5.1 Examples of **OPEN** Statements 24
   - 5.2 **LOAD** 25
   - 5.3 **VERIFY** 26
   - 5.4 **SAVE** 27
   - 5.5 IEEE-488 Special Features 27
   - 5.6 IEEE-488 **OPEN** Considerations 27

6. **Tape File Operation Modes** 28
   - 6.1 **OPEN** for Write or Tape from PET 28
   - 6.2 **OPEN** for Read from Tape to PET 28

7. **Data Input: General** 31
   - 7.1 **INPUT#** – String and Variable Input 31
     - 7.1.1 Example of **INPUT#** Statement 31
   - 7.2 **GET#** – Character Transfers 32
   - 7.3 Tape Input 32
   - 7.4 IEEE-488 Device Input Sequences 32
   - 7.5 Input Buffer Limitations 32

8. **Data Output: General** 33
   - 8.1 **PRINT#** 33
     - 8.1.1 Examples of **PRINT#** Statement 35
   - 8.2 IEEE-488 Bus Output 36
   - 8.3 CMD Command 36
     - 8.3.1 Examples of CMD Command 37

9. Closing Files 37
9.1 Example of **CLOSE** Statement 38
9.2 Tape File Closure 38
9.3 IEEE-488 Named Device Closure 38

10. Error Detection: General 38
10.1 Status Word (ST) 38
10.2 IEEE Device Errors 39
10.3 Tape Unit Errors 39
10.4 Examples of ST Use 40

11. Polling Techniques 41

12. Default Parameters 41

1. Additional BASIC Commands

By this time, the user is probably familiar with the use of the commands **INPUT** and **PRINT**. **INPUT** permits the entry of data from the input keyboard and **PRINT** permits the output or display of data. These commands are common to all forms of BASIC.

To add flexibility to the PET computer system, several commands have been added to classical BASIC in the PET, and future Commodore products will take advantage of the resulting extra capability. In general, enhanced flexibility of data interchange between the PET and peripheral devices is possible, thanks to the use of these extra commands.

To communicate with any device, a combination of the additional commands is used:

a)	**OPEN/CLOSE**	Open or close logical file.
b)	**PRINT#**	Write data from PET to I/O device.
c)	**CMD**	Same as PRINT# but leaves IEEE device an active listener on bus after execution of command.
d)	**INPUT#**	Read data from I/O device to PET.
e)	**GET#**	PET accepts one character from I/O device.

2. Input/Output Command Parameters

In order to use the additional commands referred to in 1, above, four parameters must be taken into consideration:

a) Logical file number (**LF**)
b) Device number (**D**)
c) Secondary address (**SA**)
d) File-name (**FN**)

These parameters can appear, for example, when using the **OPEN#** command in the form of the statement:

OPEN#LF,D,SA,FN

These parameters are defined and explained in 2.1 through 2.4. The default values for these parameters are listed in Tables 12-1 and 12-2, on pages 41 and 42.

2.1 Logical Files

Files are used to store and retrieve data, as for example in the case of

a magnetic tape or disc file. A convenient extension of this idea is to regard any device which can receive and/or generate data as a logical file. To the PET operating system, data might just as well have come from, or be going to, a storage system such as magnetic tape.

For example, imagine that an external digital voltmeter is set up so that it can transmit voltage readings upon request to the PET via the IEEE data bus. Sometime during the "voltmeter program" the programmer will have to open a file and assign a logical file number to the voltmeter. Once this has been done, the PET can use a "read" command (**INPUT#**) which uses the logical file number to refer to the voltmeter. When no further data is required from the voltmeter, the logical file can be closed.

More generally, the advantages offered by the use of logical files are:

a) Every device number secondary address combination can be associated with its own unique logical file number within a program.

b) Multiple files within a single device can be referred to by means of distinct logical file numbers. This approach is to be used in the newly developed disc storage system for the PET.

c) Once a logical file number has been defined in an **OPEN** statement within a program, only this single number need be used in the following input/output statements. This eliminates the need for further restatement of device number, secondary address (where used) and file name (where used).

Although it is permissible to identify and use many logical files in a given program, the PET operating system has to keep track of the files that are currently in use in the program. The greatest number of files that can be controlled by the PET at one time is ten. Note that in the present version of the operating system, exceeding ten will result in loss of PET operation; this can be restored by switching the computer off and on. A logical file number can be any integer in the range 1 through 255.

2.2 Device Numbers

All devices which the PET communicates with are assigned device numbers. The first four of these are reserved for the following peripherals:

	Device Number	Device
	0	Keyboard
Default —	1	Cassette 1 panel mounted
	2	Cassette 2 add-on
	3	Video screen

All other devices are automatically assumed by the PET to be IEEE devices, and control is transferred to the device which will have been allocated a number within the range 4 through 3Ø. Except in special cases, a specific number would be allocated to each IEEE device to allow the PET and a particular device to communicate using the parallel IEEE-488 bus.

On many IEEE devices, the allocation of the device number is made by means of a switch, or in the case of less expensive products, by the connection of jumpers.

2.3 Secondary Addresses

The concept of a secondary address may be new to those people who have never worked with the IEEE bus. The use of a secondary address permits an intelligent peripheral to function in any one of the number of modes. For example, in the PET 2020 printer, there are six secondary addresses:

Secondary Address	Operation
Default — 0	Normal printing
1	Printing under format statement control
2	Transfer data from PET to format statement
3	Set variable lines per page
4	Use expanded diagnostic messages
5	Byte data for programmable character

In short, by changing the secondary address used to communicate with a given physical device, its operating characteristics can be totally changed, if so desired. Many of the IEEE devices have their own particular secondary address conventions which must be followed. Specific data on these conventions can be obtained by consulting the manual for that particular device.

The PET tape units have a special set of secondary address rules:

Secondary Address	Operation
Default — 0	Tape is being opened for "read"
1	Tape is being opened for "write"
2	Tape is being opened for "write" with an "end of tape" header being forced when the file is closed

The secondary address can have values over the range 0 through 31.

2.4 File Names

In random storage devices where there is more than one file to be accessed, the use of names to identify files is mandatory. In the case of tapes, a file name is desirable, even if there is only one file on the tape, since it facilitates the identification of tapes.

For the two cassette tape units of the PET, a file name may be any combination of up to 128 characters.

When a file name is searched for, it is matched on an ascending character basis.

Assume that an eight character file name COUNTING was specified when writing. If desired, this could be searched for with an abbreviated name such as COU. The first file header that is found with these three consecutive characters will then be opened and positioned on. In principle, this could include unwanted file names such as COUNT or COUNTRY, as well as the wanted COUNTING.

It is, therefore, advisable to specify the *complete* file name in order to avoid errors.

For other devices which use named files, the individual description of the device should be consulted in order to ascertain the specific requirements for file name usage.

3. Tape Cassette Operation for Files

The PET devotes special attention to the two tape cassette units that can be attached to it. The tape units are specially modified so that the PET has control over the motor movement of the cassette.

It can also sense when the **Play**, **Rewind**, or **Fast Forward** buttons have been pushed; this is done by means of a single switch mounted in the tape unit.

Note that the same switch is used to sense all three buttons: if *any* of the three is pushed, the PET will think that you pushed the **Play** button and will respond accordingly.

Because of the type of mechanism used in the tape unit, the user must rewind, fast forward, stop, load and eject tapes. He must also put the unit into the write mode by pushing the record button either simultaneously with, or before the **Play** button is pressed.

The PET has total control over the movement of the tape once the appropriate buttons have been pushed to engage the motor.

Messages displayed throughout the program will tell the user when it is necessary for him to initiate the function of play or record. Logic dictates the times when the user should rewind and fast forward.

The two tape units of the PET are handled independently, and the various control lines permit the reading of data from cassette #1, the reading of data from cassette #2, motor control of cassette #1, motor control for cassette #2 and a common write line.

3.1 File Recording Technique

The data structure embodied in tape files will ensure the maximum memory utilization and maximum reliability of recording.

To accomplish this, the PET records data at two audio frequencies in two consecutive blocks of data. All of the data is totally repeated, and by means of error checking techniques incorporated in the PET software, it is possible for most audio dropouts to be corrected by the operating system utilizing the redundant data stored in the second data block.

In order to correct for (a), the fact that tape units record at different speeds, and (b), the normal drag characteristics of tapes, a speed correlation technique is used during reading. To correct for the individual start/stop characteristics on the tape and synchronize correctly between each block on tape, a single tone is written between blocks. This signal is used to synchronize both position and speed of the tape. Varying lengths of tone are used at the beginning and between the data blocks of the tape. By writing about ten seconds of the tone on each opening of a file, the PET automatically corrects for normal leader. Individual tape blocks are separated by shorter tone durations.

3.2 File Headers

An important assumption underlying the tape system design was that the user would often want to record more than one file of data on a tape. In order to facilitate this and to allow for proper label checking, the first physical data recorded on tape for any operation is a file header. This file header looks exactly the same as the normal data block, except that the first character of every block on tape contains an identification character which enables the operating system to differentiate between program blocks, data blocks, file headers and end of tape headers.

The PET allows for up to 128 characters of a file name to be stored in the file header. This is the name which is searched for and matched on in the various OPEN/CLOSE options.

3.3 Tape Buffers

Another basic premise in the design of the tape operating system was that the user would want to write tape independently of what is occurring on tape at a given moment. This is accomplished in the operating system by permanently assigning a block of memory as a data buffer for each cassette tape. A 192 character buffer is located at decimal address 634 for cassette #1, followed by a 192 character buffer at decimal address 826 for cassette #2. The tape file header is written into the buffer first and then written on tape.

Data files are accumulated in the tape buffer until 192 characters are exceeded, then the contents are either written on tape for write, or if the program is reading tape, the next block of data is read into the buffer. Tape file headers and all data blocks are, therefore, 192 characters long.

Tape buffers are not used in the case of program files, since these are written onto the tape directly from the memory in which the program resides. In order to accommodate the variable memory location, the file header for a program file contains the beginning and ending address for the program. The full program is written onto tape in the usual form of two consecutive redundant blocks.

3.4 Multiple Files

In order to have multiple files on tape, the user needs the ability to add files to a tape and also know when the tape is at its end. It is, therefore, important that the operating system give an "end of file" and "end of tape" indication.

In the case of data files, an "end of file" marker is appended after the last data character. This is available to the user in a status word on reading; the "end of file" marker is automatically inserted when a write file is closed.

In the case of program files, because all the data is always contained in a single block, the end of the block signifies the end of the program.

To signify that the end of tape has been reached, a special separate file header is written. When this is encountered during a search for files, the PET automatically stops the tape and indicates "file not found" to the user. A typical multiple file tape could contain first a data file, then a program file, followed by an "end of tape" header as illustrated in the example of Figure 3-1.

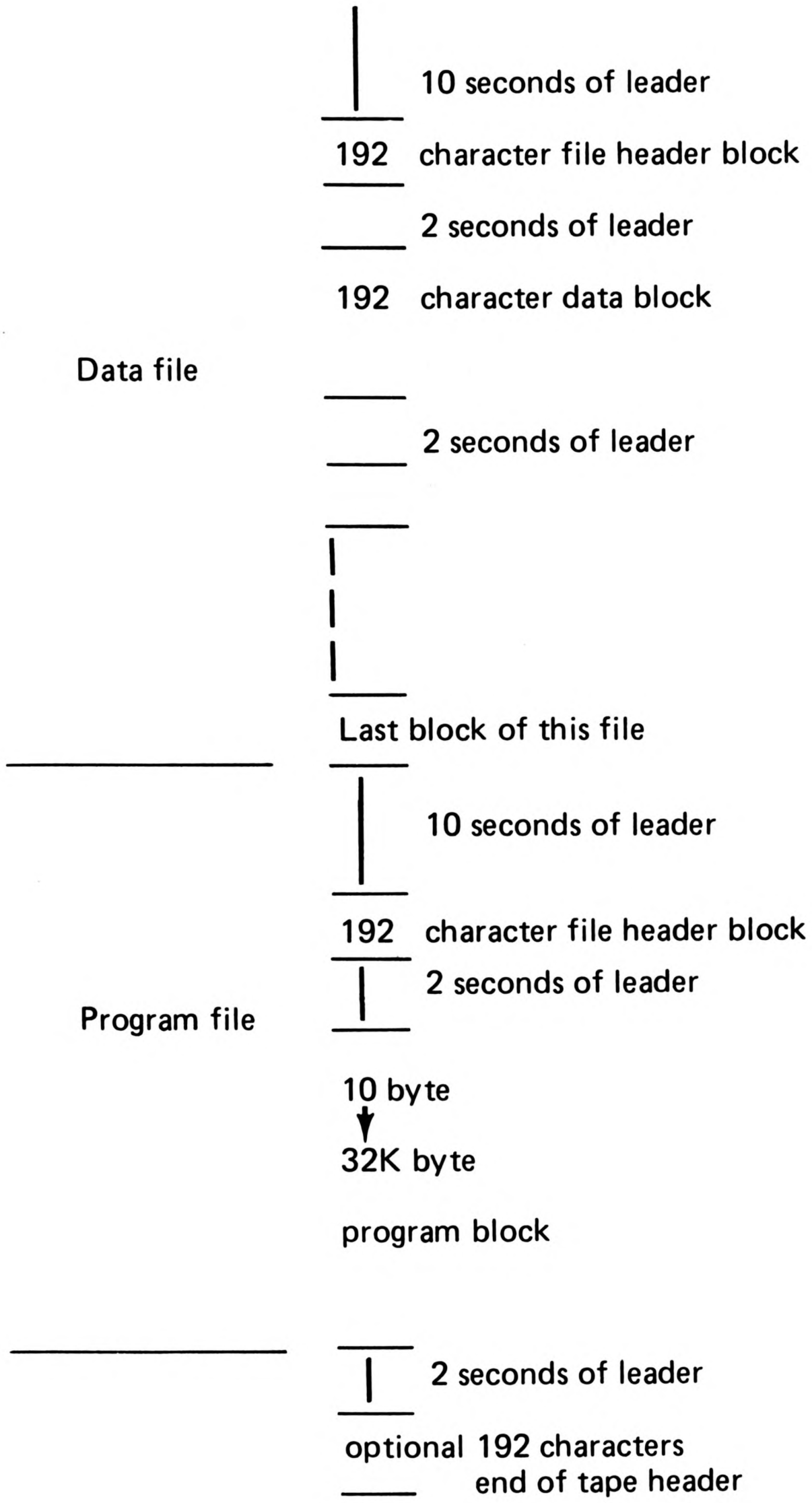


Figure 3-1. An example of multiple file structure.

4. Logical File I/O Operations: General

These operations can be subdivided into three steps:

a) Open the file — tell the PET everything it needs to know about the file.

b) Read data from, or write data to the logical files.

c) Close the file – allow the PET to clear up the device and terminate the active file.

These steps are discussed in detail in Sections 5 through 9.

5. Opening Files

In order to tell BASIC about the file you want to operate on, it is first necessary to *open* the file. This is done by the following statement:

OPEN logical file, device, secondary address, file name

More specifically, the statement consists of the command **OPEN** followed by the logical file number, then the device number to which the file is assigned, then the secondary address data (if any) communicated during the interaction of BASIC with the file, and last, the name of the physical file (if any).

This statement, or expression, is interpreted by BASIC, and could, therefore, use *computed* logical file numbers, device numbers or secondary address data. This capability is extremely useful when handling multiple file devices such as discs.

The keyword **OPEN** and the logical file numbers are essential in order to open a file; that is address a device in preparation for a "read" (**INPUT#**) or a "write" (**PRINT#**).

The device number is optional; if not entered, the default value "1" will be used (see Section 2.3 and Tables 12-1, 12-2.

A file name is optional, though preferred, for the tape units; a name would be essential for a disc storage unit, however.

5.1 Examples of OPEN Statements

The statement **OPEN 1,2,1** is interpreted by the operating system as saying:

Parameter	
(LF)	Logical file #1 has been opened
(D)	Logical file #1 has been assigned to tape unit #2
(SA)	Tape unit #2 has been instructed to write on tape
(FN)	A file name has not been assigned to the tape record

Similarly, **OPEN 3** is interpreted as saying: (F)

Parameter

(LF)	Logical file #3 has been opened
(D)	Logical file #3 has been assigned to tape unit #1 (default "1")
(SA)	Tape unit #1 has been instructed to read from tape (default "0")
(FN)	No file name referred to

If the PET 2020 printer is assigned "4" as a device number, then **OPEN 12,4,1** is interpreted as:

Parameter

(LF)	Logical file #12 has been opened
(D)	Logical file #12 has been assigned to device #4
(SA)	Printer has been instructed to print under format statement control (see Section 2.3)
(FN)	File name not applicable

NOTE: The current version of PET has a problem with **OPEN** *for tape files. The opening of the tape file is automatic, but the tape header may not always be written at the beginning of the tape buffer; this implies that the operating system does not always correctly initialize the buffer pointer. For consistent and reliable operation of the tape file header, the following statements should be used:*

1) For tape #1: **POKE** 243,122
   POKE 244,2
2) For tape #2: **POKE** 243,58
   POKE 244,3

These should be written prior to each **OPEN** for write.

This problem will be resolved in due course as a set of modified ROMs will shortly be available. However, the two **POKE**s will not cause any PET malfunction if the new ROMs are installed.

5.2 LOAD

A special case of the **OPEN** command is the **LOAD** of a named file; a **LOAD** is done with the following statement:

LOAD name, device number

The operating system automatically generates an **OPEN** using the appropriate secondary addresses for "load." This **OPEN** causes the loading device to search for a program name. After the program is

found, it is automatically read from the device and loaded into memory starting at an address specified in the file header. Any reading errors on the first pass through that program are automatically fixed on the second pass, if possible.

At the end of the load cycle, a checksum of the total program is made. If a checksum error, or if an unrecoverable read error occurred, the operating system automatically prints **?LOAD ERROR** and stops the load program.

If the program load was from direct mode, the clear function is performed at the end of the load, thereby, initializing all variables.

If the **LOAD** is called from a program, then the PET treats this **LOAD** as an overlay. The new program is loaded into the space used by the previous program, but the values of all of the variables are maintained from the previous program. This allows for one program to call another and pass parameters to the called programs.

The only restriction on this is that all called programs must fit in the same, or less space as the first program.

Because BASIC totally replaces the current program, it is not directly possible to have a single main program and several subroutine overlays, however, by including the main program with each overlay, the same effect is obtained with some loss of speed.

The combination of the use of named files and overlays allows the writing of very large structured programs of appreciable complexity.

5.3 VERIFY

This instruction is a special case of **LOAD**. It should be used after every program **SAVE**.

The command causes BASIC to go through all the steps of a program **LOAD**, with the exception that the data does not get loaded into memory, but, instead, gets compared with memory. If either first or second pass errors occur, the PET will type out **?VERIFY ERROR** which means that the program should be saved again before it is lost. On **VERIFY**, the status word has the following meanings (see Section 10.1 for details):

Code	Meaning
4	Short block
8	Long block
16	Any mismatches
32	Checksum **ERROR** on tape

5.4 SAVE

SAVE also performs an automatic open and close. The **SAVE** is specified by the statement:

SAVE name, device number

If the physical device is one of the two tape units, the operating system automatically initiates a tape header and opens a tape file with the appropriate name. The file header is written with the beginning and ending address.

If the device is an IEEE-488 device, a special open message is sent indicating that the PET is sending a program file.

The program is then written directly from its memory locations to the tape or the IEEE-488 bus.

If the **SAVE** is on tape, a checksum is computed and also saved and then the whole program is written again to give the redundant recording. At the end of the program, the tape is automatically stopped and positioned for the next data.

5.5 IEEE-488 Special Features

In the tape, the program beginning and ending address are stored in and retrieved from the tape file header.

In order to more efficiently use the IEEE-488 data, the starting address of the program is sent as the first two bytes of data on a **SAVE** and retrieved from those positions on a **LOAD**.

5.6 IEEE-488 OPEN Considerations

If the **OPEN** command selects a device which has a value of 4 or more, the operating system assumes that the device is an IEEE-488 device.

If the **OPEN** does not specify a file name, then nothing is communicated on the IEEE-488 bus. However, if a file name is specified, the operating system sends a listen attention sequence to the device number specified in the **OPEN** along with a secondary address which is the OR of hexadecimal "FØ" and the secondary address specified in the **OPEN** statement.

Commodore-supplied peripherals, such as the floppy disc storage system, will use this secondary address and also the file name, which is then transmitted to the listening device in order to transfer data later to the open file.

6. Tape File Operation Modes

Tape files can be opened for two distinct purposes:

a) In order to write from the PET onto tape.
b) In order to read from tape to the PET.

6.1 OPEN for Write on Tape from PET

The flow diagram of Figure 6-1 outlines the PET-user interaction and PET function when opening a file for write on tape. The initial block shows that there are two ways of opening the file:

a) **OPEN** for write—data tape.
b) **SAVE**—write a program tape.

Note that if the tape file is opened directly from the keyboard, then the message **WRITING NAME** is displayed. If the file is opened under program control, and the **Play** and **Record** buttons are depressed previously, then no message appears on the screen. In this manner, any display material placed there by the current program is not disturbed.

6.2 OPEN for Read from Tape to PET

The flow diagram of Figure 6-2 outlines the PET-user interaction and PET function when opening a file for reading on tape. The initial block shows that there are two ways of opening the file:

a) **OPEN** for read data tape.
b) **LOAD** program into memory.

Note that if the file is opened directly, that is from the keyboard, then the messages **PRESS PLAY, SEARCHING FOR NAME** and **FOUND NAME** are displayed. If **LOAD** was used, then the BASIC variables of the loaded program are initialized.

If the file is opened under program control and provided that the **Play** button had been pressed previously, no messages appear on the video screen in order not to disturb material displayed by the current program. Initialization of the BASIC variables does not occur.

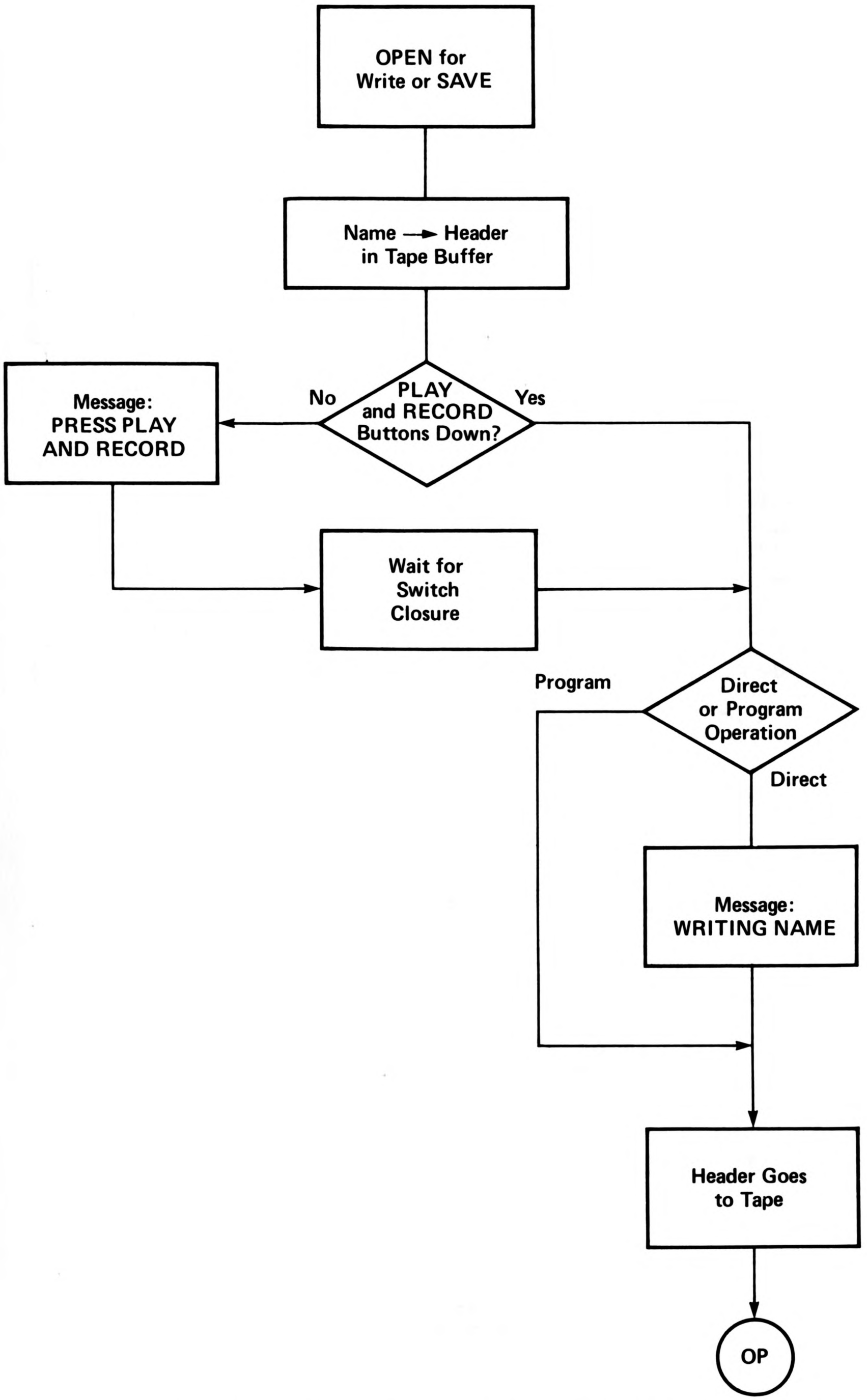
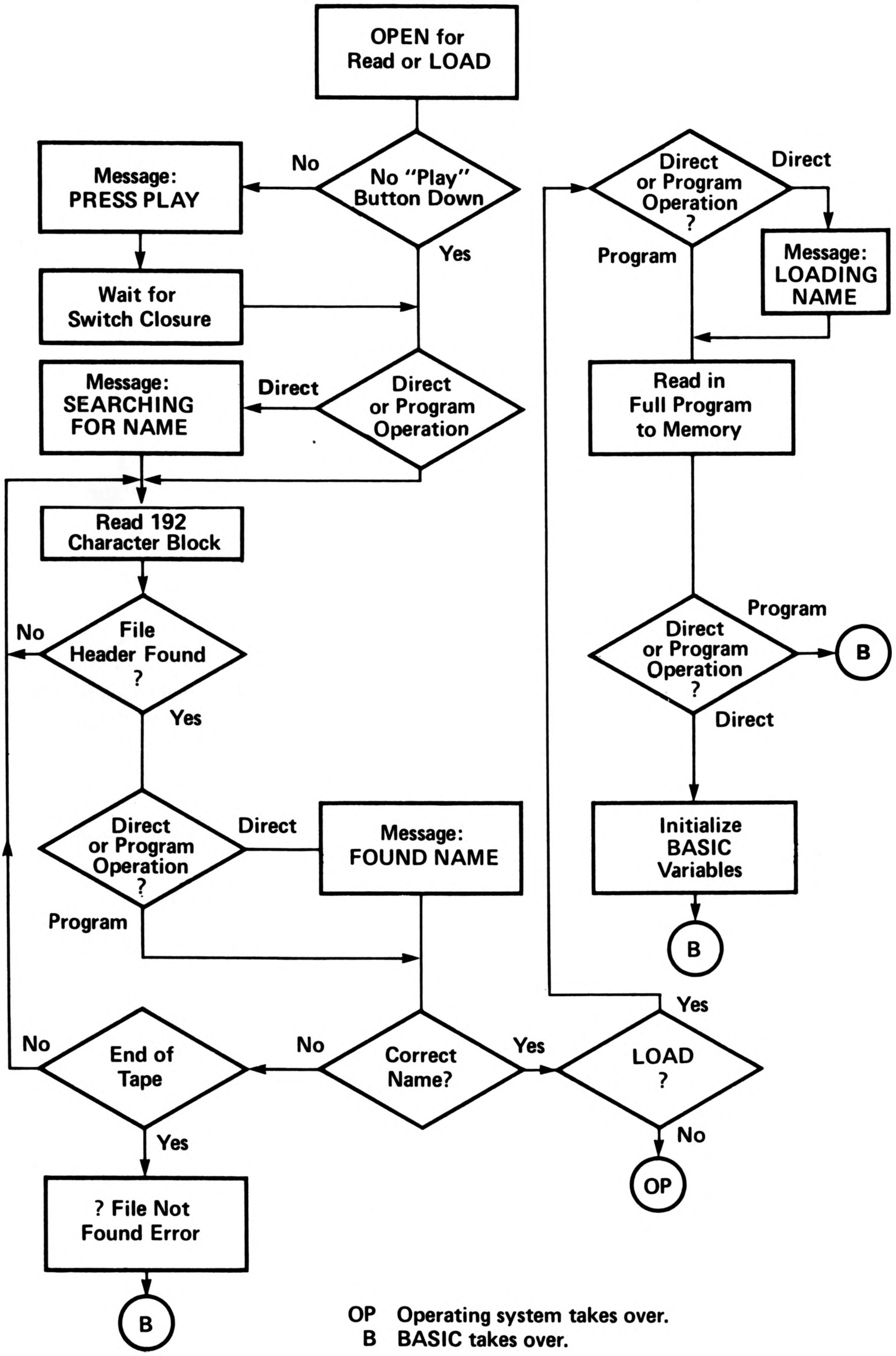


Figure 6-1. OPEN for write from PET: PRINT#, CMD or SAVE.
OP = operating system takes over.



OP Operating system takes over.
B BASIC takes over.

Figure 6-2. OPEN for read to PET: INPUT# or LOAD. OP = Operating system takes over. B = BASIC takes over.

7. Data Input: General

The use of the word "input" in this context implies input of data *to* the PET *from* any device.

7.1 INPUT# — String and Variable Input

INPUT# is the command used to initiate data transfer from I/O devices to the operating system. The statement format is:

INPUT# logical file number, A,A$,B,B$, etc.

Where A,A$,B and B$ are numerical and string variables to be inputted (read) from the selected logical file to the operating system one character at a time.

Because the rules for the BASIC interpreter apply to these input statements, all carriage returns, commas, terminate fields, nulls, preceeding blanks (except in strings), and other control characters are automatically deleted.

It is not always possible to mix both numeric and alphabetic data on an I/O device. If a numeric field is specified, only numeric data in the standard form expected by BASIC is accepted, otherwise a **?BAD DATA ERROR** message is displayed.

If there is any ambiguity about the data coming in, the user should input only to strings and then use the various string manipulation commands to process the data into the appropriate variables.

7.1.1 Example of **INPUT#** *Statement*

If X represents a series of 50 numbers stored on a tape file named **Vector** and we assume that the **Play** button has just been depressed on tape unit #1. Then the following program will read the 50 numbers one at a time and display them on the video screen.

10 OPEN 1,1,0, "VECTOR"	Open logical file #1. Assign file to cassette 1. Open tape for "read". Look for physical file named **VECTOR.**
20 FOR K=1 to 50	Read 50 numbers one at a time from cassette 1.
30 INPUT#1,X	
40 PRINT X	Display numbers on video screen.
50 NEXT K	
60 CLOSE 1	When fifty numbers have been read, close logical file #1.

7.2 GET# — Character Transfers

Not all devices transfer data in a form which is acceptable directly to BASIC. There is a series of binary data and combinations which BASIC ignores and although many IEEE devices do correctly respond with data formats which are acceptable to BASIC, not all do.

In addition, in some cases, it is desirable for the programmer to have immediate access to characters as they are transferred to the system. **GET#** fetches from the IEEE-488, or a tape device, a single character at a time, putting a character in a field specified following the **GET#**. The form is:

GET# logical file, field

7.3 Tape Input

When reading from the tape file, the data comes to the user I/O independent. Each time BASIC starts on **INPUT#** or **GET#** from a logical device which was opened for read on tape 1 or 2, a special subroutine is called, which initiates tape input.

As each character is requested from BASIC, it is fetched from the appropriate tape buffer. When the buffer is empty, the tape input routine suspends the user program and reads the next data block from tape into the buffer and then transfers the next character to BASIC. If a read error occurs, it is noted in the status word.

When the end of file mark is encountered in the buffer, the end of file position of the status word is set on and carriage returns are forced automatically out until the command is finished.

At the end of a command, BASIC calls another routine which reinitializes the input to be the keyboard and tells the end of file operation that a command is complete.

7.4 IEEE-488 Device Input Sequences

All **INPUT#** or **GET#** commands go through the same sequence. When the command is first encountered, the IEEE-488 input initiation routine is called, which sends a talk attention sequence to the device and secondary address which was specified for that logical file in the **OPEN** sequence. At the end of the attention sequence, the PET establishes itself in a listener mode and attempts to wait for a **DAV** signal indicating a single character has been received. If the **DAV** is is received within 65 milliseconds, that character is handed to BASIC and/or to the other program calling the IEEE-488 routine. Each time the IEEE-488 routine is called, it will go through the same sequence

of getting a single character while waiting for a time out to occur. If the bus does not respond in 65 milliseconds, then the IEEE-488 routine will automatically terminate the sequence; giving a read error in the status word to indicate that it has terminated the sequence.

If during the course of reading the character, the IEEE-488 routine senses an EOI line, it will indicate the end of information in the status word and will continue to return carriage returns, until the command it has been currently operating under has been terminated. At the end of the command, BASIC calls a termination subroutine which re-initializes the device to the keyboard and sends an untalk to the IEEE-488 bus, thereby, freeing the bus for the next command.

7.5 Input Buffer Limitations

Although data is transferred from the operating system one character at a time, in order to edit, BASIC accumulates these characters into an 80 column input buffer. This buffer must be terminated by a carriage return.

On current PETs, should more than 80 characters be read, the operating system will malfunction, as the operating system variables are overwritten. The PET can be made to function again by switching the line supply off and on.

Although this problem will be resolved in future versions, the 80 column limitation will still apply. This constraint must be kept in mind when using tape and disc file systems.

This means that carriage returns must be written on tapes, discs, or other I/O devices in such a way that not more than 80 characters per field are written without being separated by carriage returns.

If an I/O device sends more than 80 characters, the **GET** command can be used to build your own string without running into the buffer limitation.

8. Data Output: General

The use of the words "print" and "write" refers to data output *from* the PET *to* any device.

8.1 PRINT#

The command **PRINT#** must be followed by the logical file number, and then a comma to separate the data that would follow **PRINT**:

PRINT# logical file number, data

Data is transferred a single character at a time to the physical device correlated with the logical file specified in the relevant **OPEN** statement. Many of the file delimiters such as commas are automatically deleted by BASIC; although this does not greatly effect the printing, it should be remembered that when reading back from tape or another I/O device that file delimiters must be forced. This forcing can be done by inserting a **CHR$(44)** or "," between fields or by only printing single fields in each **PRINT#** statement which will force carriage returns between fields. Example:

instead of writing

```
PRINT#LF,A;B$;C$
```

which will be sent as:

```
AB$C
```

with no delimiters:

```
PRINT#LF,A;CHR$(44);B$;CHR$(44);C$
```

or:

```
PRINT#LF,A",";B$;",";C$
```

which will output: (Note: **CR** means carriage return)

```
A,B$,C$,CR
```

or:

```
PRINT#LF,A
PRINT#LF,B$
PRINT#LF,C$
```

which will output:

```
A  CR  B$  CR  C$  CR
```

Because BASIC always formats outputs to any devices as though it were outputting to the screen, **PRINT#LF,A,B** has several skip characters between the values of **A** and **B**, while **A ; B** does not have any extra skips.

An exception to this rule is the tape where the first skip on output is suppressed.

Note: Although both the **INPUT#** *and* **PRINT#** *commands operate in virtually the same way as their equivalent* **INPUT** *and* **PRINT** *state-*

ments do in BASIC, the abbreviated command **?** *which can be used in place of* **PRINT**, *does not apply to* **PRINT#**. **?#** and **PRINT#** are *recognized and reduced to two different token characters when processed by BASIC.* **?#** *will look like* **PRINT#** *when listed but gives* **?SYNTAX ERROR** *when an attempt is made to execute it.*

8.1.1 Examples of **PRINT#** *Statement*

This program will print the series of numbers 1,2,3,. . .50, one at a time on the PET 2020 printer.

```
10 OPEN 5,4,0
20 FOR K=1 to 50
30 PRINT#5,K
40 NEXT K
50 CLOSE 5
```

Open logical file #5. Assign logical file #5 to device #4 (2020 printer) in normal print mode corresponding to secondary address "0".

Print the series of fifty numbers on printer.

Close logical file #5.

To write the above series of numbers on a cassette in tape unit #2, only the **OPEN** line would have to be modified, if the same logical file numbers were chosen:

```
10 OPEN 5,2,1
20 FOR K=1 to 50
30 PRINT#5,K
40 NEXT K
50 CLOSE 5
```

Open logical file #5. Assign logical #5 to device #2 (tape unit #2) with a write without "end of tape" designation corresponding to secondary address '1'.

Record the series of fifty numbers on tape.

Close logical file #5.

In the above cassette example, the data would be accumulated in a 192 character buffer one character at a time. When the capacity of the buffer is exceeded, then data entry is suspended, the tape started, and the buffer contents written to tape. The buffer is initialized to accept up to 192 characters and then the program is allowed to proceed.

Note: Not all tape units currently operate with the same **Start/Stop** *characteristic as defined for the original tape operating system. In order to obtain reliable operation of the tape recorders, the 192 characters of the buffer should be monitored by the program. Prior to transferring 192 characters, the programmer should turn on the appropriate cassette motor and then wait for at least .1 second before transferring the last character.*

There are several ways to accomplish this. The simplest is to just **POKE 59411,53** *for cassette #1 and* **POKE 59456,207** *for cassette #2 after every* **PRINT** *statement, this keeps the motor on all of the time and eliminates the problem.*

On the other hand, if your programs have time consuming functions like human input, sorting, or other long program run times, you should not run the motor all the time, but obtain the delay either putting a delay loop before each print, or turning the motor off with a **POKE 59411,61** *for cassette #1 or a* **POKE 59456,223** *for cassette #2 before the long function and turning it back on after it.*

8.2 IEEE-488 Bus Output

The **PRINT#** command causes BASIC to call an output subroutine which initializes an IEEE-488 device for output. The first step in the command is that the PET reassigns its normal output from the screen device to the physical device that was chosen for the logical file in the open routine. A listen command is sent on the IEEE bus to the physical device and a secondary address specified for that logical file in the **OPEN**.

BASIC then hands one character at a time to another subroutine which proceeds to transfer that character over the bus with the PET acting as a talker and all addressed devices responding listeners.

When BASIC has finished the **PRINT#**, another subroutine in the operating system is called and the PET sends an "unlisten" command to the entire bus and restores the primary address to the screen. This frees the whole bus for the next operation.

This unlisten sequence also sends an EOI signal on the bus, along with the last character sent from BASIC. To accomplish this, each character is stored in a buffer prior to transmission by the IEEE routines and the previous character is sent.

8.3 CMD Command

Normally, each print command deals only with one logical device and at the end of the command the entire bus is unlistened. In some instances, it is advisable to have more than one device on the bus; in order to facilitate this, the special command **CMD** is provided. **CMD** is virtually identical to **PRINT#**, except that at the end of the data transfer, the unlisten routine is not called, thereby leaving the device on the bus as a listener.

The operating system continues to treat the last device to be commanded by **CMD** as the primary output device for BASIC. **PRINT** or **LIST** commands are then directed to this primary device, rather than to the video screen. More specifically, the **CMD** of the printer device, followed by **LIST**, results in a hard copy *printed* listing, instead of a video screen listing. However, since neither the **CMD** nor **LIST** command terminate bus operation for the device, a **PRINT#** is required to terminate a **CMD** command.

8.3.1 Examples of **CMD** *Command*

To list:.

```
OPEN 3,4          where 4 is the printer device number.
CMD 3
LIST              will list just the same as the screen, except
                  on the printer.
```

to print and write to a disc at the same time:

```
*CMD 3            where logical file 3 is open to the printer.
 PRINT#15,A,B,C   where 15 is the floppy disc logical file number
                  (previously opened)
```

will result in A,B, and C being stored on the floppy but also being displayed on the printer.

To monitor an input device:

```
**CMD 3           turn on printer
  INPUT#15,A,B,C  read from floppy
```

This will result in the data coming from the floppy being transferred to A, B and C but also being printed as they are being transferred.

9. Closing Files

Any logical files which have been opened during a program should preferably be closed when no longer required, and in the case of tape or disc files, must be closed before the program ends. The following should be kept in mind:

a) If the total number of logical files currently open exceeds ten, then loss of PET operation will result.

b) If a logical file assigned to a tape unit is not closed, no "end of file" mark will be recorded at the end of the physical tape file. If this tape is then loaded into memory, the PET will have no way of knowing the file has ended, and if unwanted and erroneous data is present from a previous recording, it will also be read into memory.

---

*Must be given each time because **PRINT#** unlistens the bus.

**Need not be given each time, more code may be included between the instructions.

9.1 Example of CLOSE Statement

To close any file, the following simple statement is sufficient:

CLOSE logical file

If it is required to close logical file number 5, then this becomes:

CLOSE 5

9.2 Tape File Closure

If a file had been opened on the tape, there are two operations that occur: an "end of file" marker is recorded in the next data character, then the tape buffer is forced out onto the cassette.

If during **OPEN** the "end of tape" option was chosen, an "end of tape file" header block is also forced out on the cassette.

9.3 IEEE-488 Named Device Closure

For IEEE-488 devices, which were opened with file names, a special listener command sequence, with the special secondary address of hexadecimal EØ OR'ed with the secondary address from the **OPEN** is sent. This allows devices such as disc files to be closed by the peripheral controller.

10. Error Detection: General

The basic concept of the PET operating system is that the user will be allowed to operate in a free-form format; reading and writing on tapes, discs, and printers, in the manner that is most comfortable for him. Because I/O is totally free-form, it is important that the operating system should have a means of informing the user when transmission errors or end of data conditions occur. Sections 10.1 through 10.4 deal with error detection methods available to PET users.

10.1 Status Word (ST)

In order to facilitate **Input/Output** operation error detection, the PET uses the "status word" concept in which a byte in memory is manipulated by each of the I/O operations for the PET, and can be sampled by the programmer at any time by calling the function ST. Each bit in the status word has a general meaning for all operations and a specific meaning for the individual I/O device.

Table 10-1 shows the errors as a function of the ST word value for the tape cassette units, IEEE read/write operations, tape verify and load operations.

Table 10-1. Status Word (ST) values correlated with tape cassette, unit and IEEE bus read/write errors.

ST Bit Position	ST Numeric Value	Cassette Read	IEEE R/W	Tape Verify + Load
0	1		Time out on write	
1	2		Time out on read	
2	4	Short block		Short block
3	8	Long block		Long block
4	16	Unrecoverable read error		Any mismatch
5	32	Checksum error		Checksum error
6	64	End of file	EOI line	
7	-128	End of tape	Device not present	End of tape

10.2 IEEE Device Errors

There are basically three errors that can occur during an IEEE-488 transfer. First, the entire bus does not respond to an attention sequence. If this occurs, the IEEE-488 subroutine sets the **device not present** bit (7 or -128). The PET also terminates the current program with a **?DEVICE NOT PRESENT ERROR**. If the bus responds correctly to the attention, but when the PET goes to write the first character to the bus and the physical device is not present as indicated by having NRFD or NDAC low, the PET, again, gives a device not present indication.

The second error occurs during the process of transferring data to the device. The bus does not respond in the appropriate times and/or if it ceases to respond by means of bringing NRFD and NDAC both high, a write error indication is given in bit Ø.

The third error occurs when during read on an IEEE-488, the IEEE device has not sent DAV in less than 65 milliseconds; bit 1 of the status word is then set. Whenever the EOI line is encountered, the subroutine sets the bit 6 on in the status word and continues to force carriage returns.

10.3 Tape Unit Errors

The cassette only checks data on read. The errors detected are:

1) **SHORT BLOCK (4).** When reading a block from tape, a spacer tone was encountered before the expected number of bytes has been read from that block. Possible cause: attempting to read a short load file as a data record.

2) **LONG BLOCK (8).** When reading a block from tape, a spacer tone was not encountered after the expected number of bytes had been read from that block. Possible cause: reading a long load file as data.

3) **UNRECOVERABLE READ ERROR (16).** Cause: more than 31 errors on the first block of redundant blocks—or—an error that could not be corrected because it occurred in the same place in both blocks.

4) **CHECKSUM ERROR (32).** After a LOAD or reading of data, a checksum is computed over the bytes in RAM and compared to a byte received from the input device. If they do not match, this bit is set.

5) **END OF FILE (64).** This bit is set when the end of data file mark is encountered in a tape record.

6) **END OF TAPE (–128).** An EOT record was read.

10.4 Examples of ST Use

As you can see, there is no status that the PET detects for the writing of tapes, nor errors detected for printing to and reading from the screen. There is an error on writing data out to the IEEE-488 and there is also a series of errors detected on inputting from the IEEE-488 or from tape.

The normal programming technique is to follow an **INPUT#** or a **GET#** by either a test or storage of the value of status. Because this is only a single byte of memory and the status changes on each new I/O command, the status is very transient.

```
100 INPUT#2,A
110 INPUT#5,B
120 IF ST=0 THEN 200
```

This code only checks the result of the transfer of data from logical file 5. The results of reading logical file 2 is forever lost. Similarly:

```
100 INPUT#2,A
110 PRINT A
120 IF ST=0 THEN 200
```

In this case, the **ST** reflects the print status, rather than the results of reading #2.

A correct way to use **ST** is the following:

```
100 INPUT#2,A,B,C
110 IF ST=0 THEN 200          process normally
120 IF ST=64 THEN 300         end of data processed with no errors
130 IF ST=2 THEN 400          time out with no errors
```

Each error can now be processed with the following:

```
140 IF ST AND mask THEN    Mask represents the bit being tested
```

11. Polling Techniques

One technique to poll slow IEEE-488 devices such as sampling devices, which take many minutes to respond, is to use the **INPUT#** from the device; then if the status indicates time out, process other routines or loop on the **INPUT#** until no error occurs. If there are no errors, the correct data has been finally read and one can process that data information.

By using this sampling technique, a whole series of slow devices can be serviced, along with running a foreground program by use of the real time clock (TI,TI$) and the **INPUT#**/timeout error sequence, to occasionally poll devices.

12. Default Parameters

Table 12-1. Default values.

Parameter	Default Value	Default Operation
Device #	D=1	Cassette #1 selected
Secondary address	SA=Ø	On tape files open for read On IEEE-488 devices, no secondary address is sent.

Table 12-2. Example of default parameters.

Statement	Equivalent (Default) Parameter Values	Operation
OPEN#1	OPEN 1,1,Ø	Open logical file #1 for cassette #1 read no file name
OPEN#1,2	OPEN#1,2,Ø	Open logical file #1 for cassette #2 read no file name
OPEN#1,2,1	OPEN#1,2,1	Open logical file #1 for cassette #2 write no file name
OPEN#1,2,1, "DAT"	OPEN#1,2,1, "DAT"	Open logical file #1 for cassette #2 write file named "DAT"

The IEEE-488 Bus —A Short Description

TABLE OF CONTENTS

1. Introduction to the IEEE-488 Bus 44
1.1 Bus/Device Connection 44
1.2 The Data Bus 45
1.2.1 Data Transmission Modes 45
1.3 The Transfer Bus 45
1.3.1 The Handshake Procedure 46
1.3.2 PET/IEEE Bus Timing Constraints 49
1.4 The Management Bus 49

2. IEEE Signals and Definitions 49
2.1 Logic Level Convention 49

3. Status Word (ST) 51

4. IEEE-488 Register Addresses 51

1. Introduction to the IEEE-488 Bus

This bus consists essentially of 16 signal lines that are divided functionally into three groups, those are:

a) The data transmission bus
b) The control bus
c) The management bus

Furthermore, the IEEE bus can support three classes of device:

a) Talkers: at any given moment, only one device is permitted to transmit data to the data bus.
b) Listeners: as many devices as required may receive data simultaneously from the bus.
c) Controller: the PET is the *only* controller allowed on the IEEE bus.

The function and mode of operation of the data, control and management busses, are discussed in Sections 1.2 through 1.4.

1.1 Bus/Device Connection

The line-pin connections for the 12 position, 24 contact edge card connector, emanate from the PET main assembly board (see Table 1-1). For further information, please refer to Figure 1-2 in "PET Interfaces and Lines."

Certain physical limitations should be noted when connecting devices to the IEEE bus:

a) The maximum advisable bus extension from the PET is 20 meters.
b) The maximum interdevice spacing is 5 meters.
c) The maximum number of devices is 15.

Table 1-1. IEEE bus group, label and contact identification number.

PET Contact Identification	Bus	IEEE Label	PET Contact Identification	Label Description
1	DATA	DI01	1	Data INPUT/OUTPUT LINE #1
2		DI02	2	Data INPUT/OUTPUT LINE #2
3		DI03	3	Data INPUT/OUTPUT LINE #3
4		DI04	4	Data INPUT/OUTPUT LINE #4
5	MANAGER	EOI	5	End or identify
6	TRANSFER	DAV	6	Data valid
7		NRFD	7	Not ready for data
8		NDAC	8	Data not accepted

Table 1-1. IEEE bus group, label and contact identification number (continued)

PET Contact Identifi-cation	Bus	IEEE Label	PET Contact Identifi-cation	Label Description
9	MANAGER	IFC	9	Interface clear Same as PET reset
10		SRQ	10	Service request
11		ATN	11	Attention
12		SHIELD	12	Chassis ground and IEEE cable shield
A	DATA	DI05	13	Data INPUT/OUTPUT LINE #5
B		DI06	14	Data INPUT/OUTPUT LINE #6
C		DI07	15	Data INPUT/OUTPUT LINE #7
D		DI08	16	Data INPUT/OUTPUT LINE #8
E	MANAGER	REN	17	Remote enable (REN) always ground in the PET
F	GROUNDS	GND6	18	DAV ground
H		GND7	19	NFRD ground
J		GND8	20	NDAC ground
K		GND9	21	IFC ground
L		GND10	22	SRQ ground
M		GND11	23	ATN ground
N		LOGIC GND	24	Data ground (DI01-8)

1.2 The Data Bus

This bus is comprised of 8 bi-directional lines that transmit the active low data signals DI01-8. The slowest device in use on the bus at a given time controls the rate of data transfer; the mode of transfer is one byte at a time, bit parallel.

Peripheral addresses and control information are also transmitted on the data bus. They are differentiated from data by ATN (true) during their transfer.

The most significant bit (MSB) is on line DI08.

For an explanation of signal abbreviations such as DI01-8, see Section 2.

1.2.1 Data Transmission Modes

All possible bit patterns are valid on the data bus when sending data to devices.

1.3 The Transfer Bus

This three line bus controls the transfer of data over the data bus. The signals transmitted are used in the handshake procedure outlined in Section 1.3.1.

These signals are:

a) NRFD Not ready for data
b) NDAC Data not accepted
c) DAV Data valid

Note that the talker originates the DAV signal and the listeners the NRFD and NDAC signals.

See Table 2-1 for detailed description of signals.

1.3.1 The Handshake Procedure

When a talker transmits a data byte to one or more listeners, this control procedure is used in order to ensure that the operation is successful.

The essential function of the handshake is to ensure:

a) All listeners are ready to accept data.
b) That there is valid data on the data bus.
c) That the data has been accepted by all listeners.

The transfer of data occurs at a rate determined by the slowest active device on the bus; this allows the interconnection of devices which handle data at different speeds.

The sequence of events that occur during the transfer of a data byte from the talker to the listeners is shown in the flow diagram of Figure 1-2.

Figure 1-1 shows the relative timing of transfer bus signals during a typical handshake; the bracketed numbers in the following sequence refer to the changes in signal logic levels in the Figure:

1) NRFD goes high (false) indicating that all listeners are ready for the next byte of data.

2) The talker puts the next data byte on the data bus and allows the data signals to settle. This could happen before, after or during (1).

3) The talker tests NRFD, when it is found to be high, the talker makes DAV low (true) to inform listeners that the bus data is now valid.

4) As soon as a single listener detects that DAV is low, that listener sets NRFD low; data is now accepted by all the individual listeners at their own rate, each of whom release NDAC as they accept the data.

5) NDAC goes high (false) when the slowest of the listeners have accepted the data.

6) The talker sets DAV high (false) indicating that the data bus signals are now invalid.

7) The listeners note that DAV has gone high and sets NDAC low (true) completing the handshake. When each listener has processed the data, they release NRFD. This terminates the sequence for the first data transfer. The sequence will repeat again, beginning at (a), until all required data transfers have been completed.

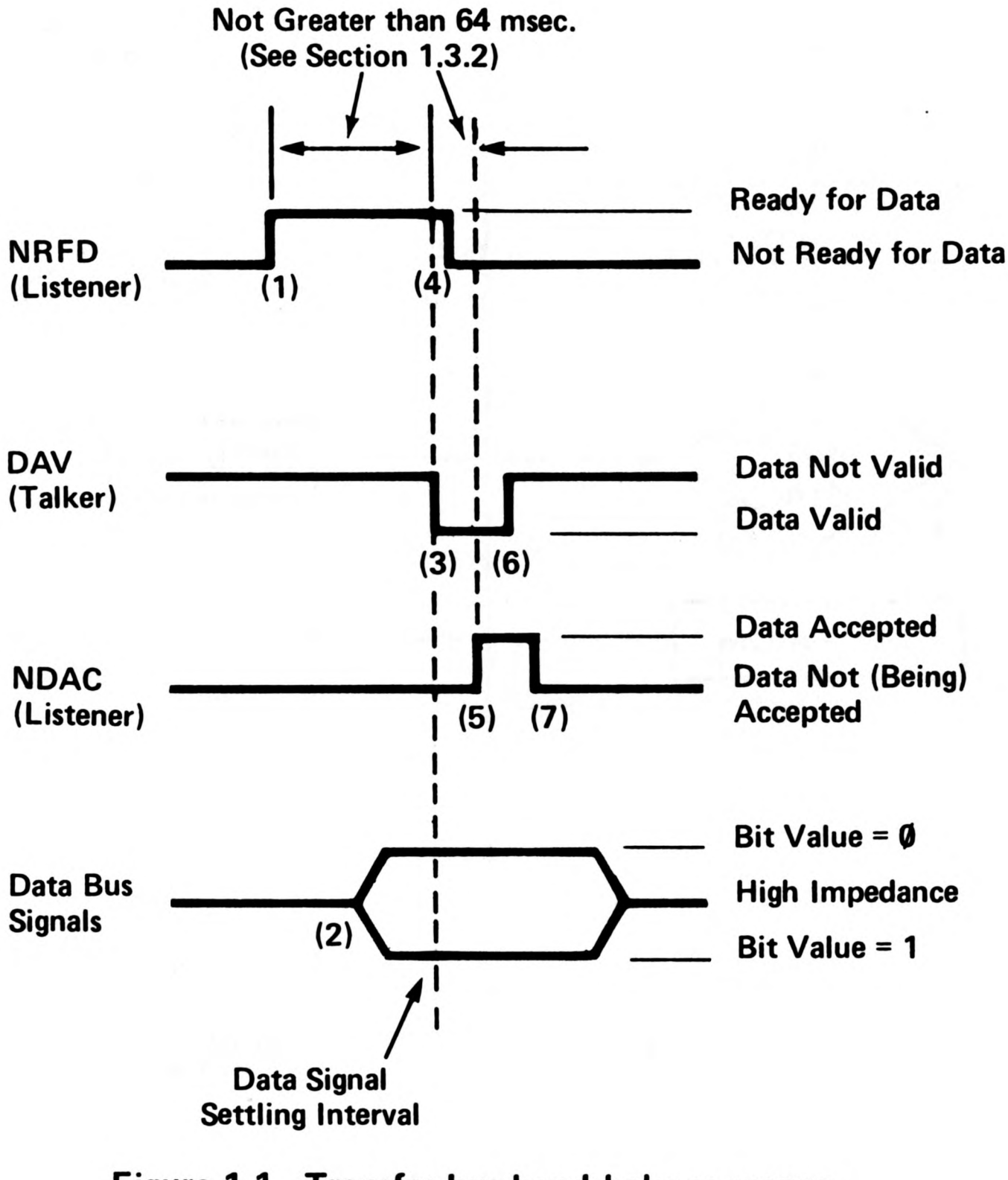


Figure 1-1. Transfer bus handshake sequence. Numbers (1)–(7) refer to Section 1.3.1.

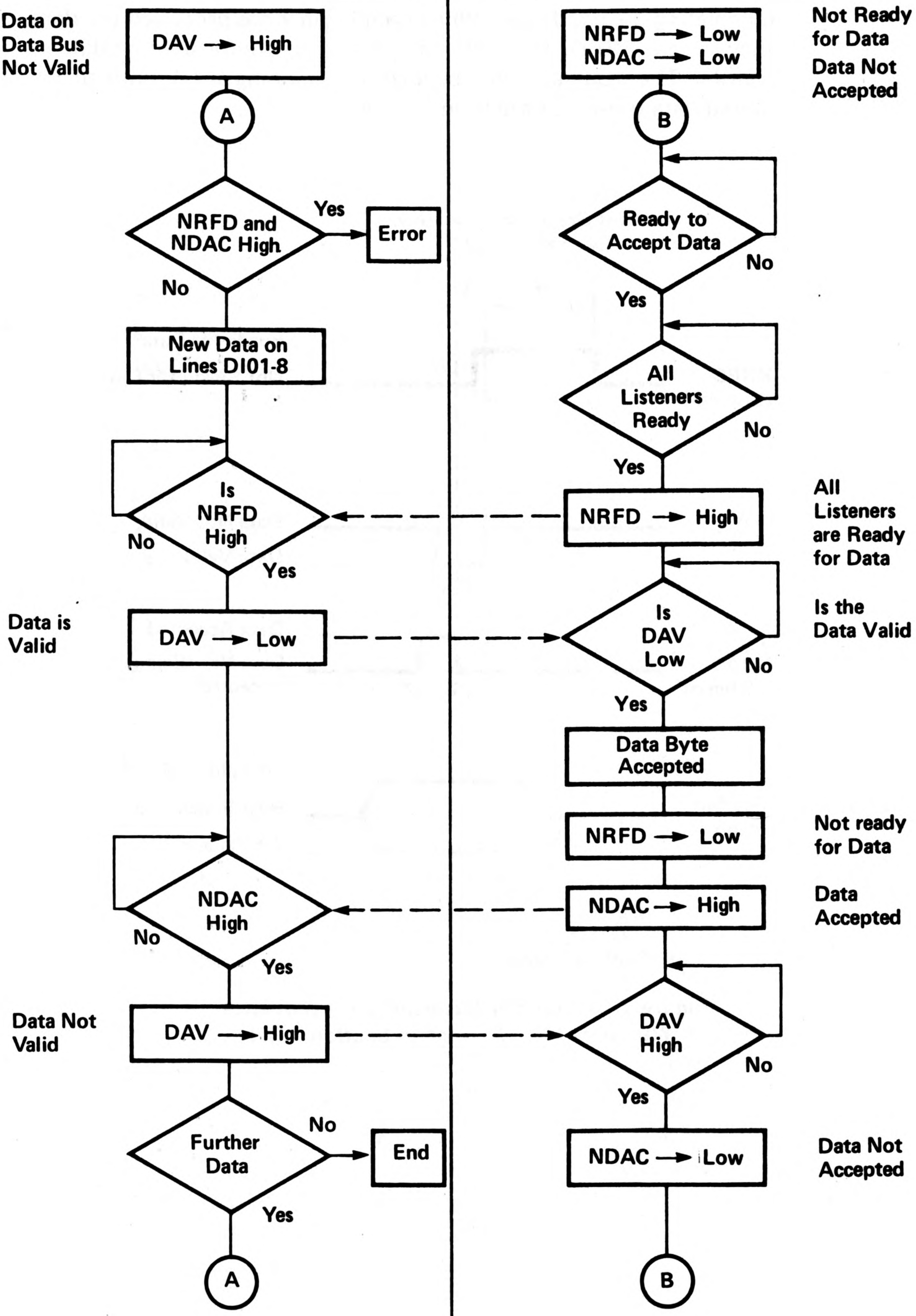


Figure 1-2. Sequence of events during a data byte transfer from the talker to the listeners. Broken lines indicate the testing of transfer bus signal logic levels.

1.3.2 PET/IEEE Bus Timing Constraints

The following limitations should be noted in order to avoid a loss of data:

a) When PET is a listener, it expects DAV to go low within 64 milliseconds after it has set NRFD high.

b) When PET is a talker, it expects NDAC to go high within 64 milliseconds after it has set NRFD high.

If these limitations are exceeded, the PET ceases to transfer and sets the appropriate status word (ST). See Table 3-1.

1.4 The Management Bus

This group of five signal lines controls the state of the data bus and defines its signals; these can be concerned with data, addresses, or control information (device commands).

The five management signals are:

a)	**ATN**	Attention	Assigns devices to act as listeners or talkers.
b)	**EOI**	End or identify	Indicates that last data byte is being transferred.
c)	**IFC**	Interface clear	Initializes the data bus. Talkers and listeners set idle. Same signal as reset in the PET.
d)	**SRQ**	Service request	Device tells controller that service is required. Not implemented in BASIC but available in PET.
e)	**REN**	Remote enable	Permanently tied to ground in the PET.

2. IEEE Signals and Definitions

The 16 transmission lines of the IEEE-488 bus are each assigned a specific signal. Table 2-1 gives the bus group, name, abbreviation and functional description for each of these signals.

2.1 Logic Level Convention

The "true" or logical "1" is low with common collector type outputs. This allows any device to hold the bus in the "true" or logical "1" state.

Table 2-1. IEEE-488 bus signal.

Bus Group	Signal Abbrev.	Name	Functional Description
Manager	ATN	Attention	The PET (controller) sets this signal low while it is sending commands on the data bus. When ATN is low, only peripheral addresses and control messages are on the data bus. When ATN is high, only preveiously assigned devices can transfer data.
Transfer	DAV	Data Valid	When DAV is low, this signifies that data is valid on data bus.
Manager	EOI	End or Identify	When the last byte of data is being transferred, the talker has the option of setting EOI low. The PET always sets EOI low while the last data byte is being transferred from the PET.
Manager	IFC	Interface Clear	The PET sends its internal reset signal as IFC low (true) to initialize all devices to the idle state. When PET is switched on or reset, IFC goes low for about 100 milliseconds.
Transfer	NDAC	Data Not Accepted	This signal is held low (true) by the listener while reading. When the data byte has been read, the listener sets NDAC high. This signals the talker that data has been accepted.
Transfer	NRFD	Not Ready for Data	When NRFD is low (true), one or more listeners are not ready for the *next* byte of data. When all devices are ready, NRFD goes high.
Manager	SRQ	Service Request	Not implemented in BASIC, but available to the PET user.
Manager	REN	Remote Enable	REN is held low by the bus controller. The PET has a pin grounded that keeps REN permanently low.

Table 2-1 continued on next page.

Table 2-1. IEEE-488 bus signal (continued)

Bus Group	Signal Abbrev.	Name	Functional Description
Data	DI01-8	Data input/ output lines 1 through 8	These signals represent the bits of information on the data bus. When a DI0 signal is low, it represents 1 and when high Ø.
General	GND	Ground	Ground connections: There are six control and management signal ground returns, one data signal ground return and one chassis shield ground lead.

3. Status Word (ST)

ST is a BASIC variable which can be used to check the outcome of **Input/Output** operations. **ST** can have certain values over the range –128 to 127. Table 1-4 shows the status code that appertains to the IEEE-488 bus.

Table 3-1. ST status code for IEEE-488 bus.

ST	Error	Explanation
1	Time out on listener	The IEEE device has not responded within the 65 milliseconds time out inverval.
2	Time out on talker	The IEEE device has *not* provided an active "data valid" signal (DAV low) within the 65 millisecond time out interval.
64	End or identify (EOI)	EOI has gone low (true), on the last byte of data being transferred on IEEE bus. Note that all devices do not generate an EOI signal. Consult relevant instrument manual.
–128	Device not present	Device did not respond when addressed; this generates an error message and the operating system returns the PET to BASIC command level.

4. IEEE-488 Register Addresses

Table 4-1 shows the IEEE-488 hardware addresses for the PET. An attempt to control the bus by means of the **PEEK** and **POKE** commands will fail, if the time out intervals for the 488 devices are exceeded.

Table 4-1. IEEE-488 hardware addresses and signal information.

Hex Address	Decimal Address	Bits	IEEE	Mode
E820	59424	0–7	DI01-8	Input
E822	59426	0–7	DI01-8	Output
E821	59425	3	NDAC	Output
E823	59427	3 7	DAV SRQ	Input
E810	59408	6	EOI	Input
E840	59456	0 1 2 6 7	NDAC NRFD ATN NRFD DAV	Input Output Output Input Output

Table 4-2 gives the code assignments for the command mode of operation on the IEEE bus.

Table 4-2. Code assignments for "Command Mode" of operation.

(SENT AND RECEIVED WITH ATN TRUE)

b7 / b6 / b5 / Bits ②	b4 ↓	b3 ↓	b2 ↓	b1 ↓	COLUMN → ROW ↓	0 0 0 0	① MSG	0 0 1	MSG	0 1 0	MSG	0 1 1	MSG	1 0 0	MSG	1 0 1	MSG	1 1 0	MSG	1 1 1	MSG
						0		1		2		3		4		5		6		7	
	0	0	0	0	0	NUL		DLE		SP		0		@		P				p	
	0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
	0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
	0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
	0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
	0	1	0	1	5	ENQ	PPC ③	NAK	PPU	%		5		E		U		e		u	
	0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
	0	1	1	1	7	BEL		ETB		'		7		G		W		g		w	
	1	0	0	0	8	BS	GET	CAN	SPE	(		8		H		X		h		x	
	1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
	1	0	1	0	10	LF		SUB		*		:		J		Z		j		z	
	1	0	1	1	11	VT		ESC		+		;		K		[		k		{	
	1	1	0	0	12	FF		FS		,		<		L		\		l		|	
	1	1	0	1	13	CR		GS		-		=		M		]		m		}	
	1	1	1	0	14	SO		RS		.		>		N		^		n		~	
	1	1	1	1	15	SI		US		/		?		O		_		o		DEL	

MLA ASSIGNED TO DEVICE

MTA ASSIGNED TO DEVICE

MEANING DEFINED BY PCG CODE

④

ADDRESSED COMMAND GROUP (ACG)

UNIVERSAL COMMAND GROUP (UCG)

LISTEN ADDRESS GROUP (LAG)

TALK ADDRESS GROUP (TAG)

PRIMARY COMMAND GROUP (PCG)

SECONDARY COMMAND GROUP (SCG)

NOTES:
① MSG = INTERFACE MESSAGE
② b_1 = DI01 . . . b_7 = DI07
③ REQUIRES SECONDARY COMMAND
④ DENSE SUBSET (COLUMN 2 THROUGH 5). ALL CHARACTERS USED IN BOTH COMMAND & DATA MODES.