COMMODORE SUPERPET COMPUTERS

System Overview©

commodore
COMPUTER

System Overview:

Commodore SuperPET

F. D. Boswell
T. R. Grove
K. I. McPhee
J. B. Schueler
J. W. Welch

Disclaimer

Table of Contents

Waterloo Computing Systems Newsletter

The software described in this manual was implemented by Waterloo Computing Systems Limited. From time-to-time enhancements to this system or completely new systems will become available.

A newsletter is published periodically to inform users of recent developments in Waterloo software. This publication is the most direct means of communicating up-to-date information to the various user. Details regarding subscriptions to this newsletter may be obtained by writing:

Waterloo Computing Systems Newsletter
Box 943,
Waterloo, Ontario, Canada
N2J 4C3

Chapter 1

Introduction

This manual is intended to provide an introduction to the Commodore SuperPET personal computing system. This is accomplished in the next two chapters which give an overview of the hardware and of the Waterloo microSoftware packages.

The manual also includes chapters on topics which apply to several of the software packages. Three chapters are concerned with the file system used by the Waterloo microSoftware. There is also a chapter on the full-screen editor which can be used to enter or to update text or programs. In addition, there is a chapter on the SETUP program which is used to set up the RS232 serial line for communication.

The appendices include an overview of the Waterloo Library, a description of the floating-point emulation used in Waterloo microSoftware, and a document describing usage of the screen with the computing system.

A new user of the system is advised to read the first three chapters of this manual and then to refer to a language reference manual for details on how to use a specific language. Portions of this manual may be referred to subsequently as directed by the individual's need for information.

Chapter 2

Commodore SuperPET Hardware

2.1 Introduction

This chapter provides an introduction to the hardware of the Commodore SuperPET computer. The presentation consists of two sections. The first is an external view of the personal computer and the second gives the internal details of the hardware architecture. The capabilities of the system may be understood by reading only the first part. The second section is intended to supply additional information to those requiring a more thorough understanding of the hardware.

2.2 External View

Simply, the Commodore SuperPET consists of a keyboard, a CRT (cathode ray tube) screen, two microprocessor computing systems, and a number of connectors (called ports) to which external devices may be connected.

Microprocessor Selection

A three-stage toggle switch controls the manner in which the microprocessor system is selected.

(a) In the position marked "6502", the MOS Technology 6502 microprocessor system is selected.

(b) In the position marked "6809", the Motorola MC6809 microprocessor system is selected. In order to run the Waterloo microSoftware, this switch position should be selected.

(c) In the position marked "PROG", the microprocessor system to be selected is controlled by the software in the computer.

Thus, the personal computer may be viewed as two systems which are packaged together.

Memory

The system memory consists of read/write memory (RAM) into which programs and data may be loaded and/or executed. With each microprocessor there is also some individual read-only memory (ROM) containing non-changing information. With the 6502 processor, this ROM contains the Commodore BASIC package as described in the manual for that software. With the 6809 microprocessor, the ROM contains a library of useful routines utilized by the Waterloo microSoftware.

The read/write memory consists of 32768 bytes (32K) of "standard" memory which is always accessible by a computer program. Additionally, there is 65536 bytes (64K) of read/write memory which is organized into 16 "banks" of 4096 bytes (4K) each. At any one time only one bank is directly accessible by a computer program. A facility exists whereby the accessible bank can be changed by an executing program. In the Waterloo microSoftware, this feature is used extensively. Software programs, such as interpreters, are loaded into the "bank-switched" RAM and the entire "standard" RAM memory is thus available to store the user programs (written in one of the microlanguages) and the associated data.

The Waterloo microSoftware interpreters do not modify this "bank-switched" RAM once an interpreter has been loaded. The instructions in the interpreter can now be protected from accidental modification by setting the memory three-stage toggle switch to "READ". This action will prevent the "bank-switched" RAM from being modified. Setting the switch to "R/W" permits this memory to be modified and setting the switch to "PROG" enables programs to set the memory protection as desired.

Keyboard

The keyboard of the Commodore SuperPET is a 73-key typewriter-style keyboard with standard upper/lower case, numeric keypad, and full cursor control. The depression of one of these keys can be detected by the software executing in the personal computer. This software typically causes a character to be displayed on the screen.

Thus, different software programs may view the keyboard in different ways. The Waterloo microAPL system may view a key as representing an APL operator. The Commodore BASIC system may interpret the same character as a graphics character.

Screen

The screen is an integrated green phosphor display with 25 lines, each containing 80 characters. Each character is displayed using an 8 by 8 dot matrix of which only 7 by 7 dots are normally used for the character. Normally, the hardware displays lines with 2 extra blanked rows of dots separating each line. These separation lines may be omitted (by the software control, see the appendix on Screen Usage) in order to provide better graphics support.

There are four different fonts available for displaying characters. A font represents the collection of character representationsto be displayed. The fonts are:

(a) Commodore graphics font
(b) Commodore upper/lower case Roman font
(c) Waterloo upper/lower case Roman font
(d) Waterloo APL font

These fonts are described in the appendix on Screen Usage. Each software package selects the font appropriate for its purposes. Thus, Waterloo microAPL uses the last font and Commodore BASIC uses the first two fonts. A font can also be selected from the software, enabling a microFORTRAN program to use the Commodore graphics font in order to present graphical output.

Ports

There are a number of input/output ports to which may be connected external devices. The RS232 port may be used to connect asynchronous serial devices which communicate using this interface. A SETUP program (described in the following chapter) permits this port to be used with varying characteristics such as different communication speeds. The RS232 port is typically used to connect to host computing systems or to serial devices such as printers.

There is also an IEEE-488 port which can be used to connect devices which communicate using this interface. Such devices include Commodore disk drives (2040, 4040, 8050) and printers (2022, 4022). A number of these devices may be connected in parallel in order that more than one be simultaneously available to the system.

There is also a user port consisting of an 8-bit parallel interface. This is described in the Commodore manual.

Additionally, there are two cassette ports which may be used with the 6502 microcomputer system, but are not currently supported by the 6809 software. These allow cassette units to be attached to the system as file storage devices.

2.3 Internal View

This section is intended to provide an introduction to the internal view of the hardware of the Commodore SuperPET. Other documentation should be consulted for a more detailed description of the various components. It is assumed that the reader of this section has some hardware knowledge.

Memory Layout

Both microcomputer systems view memory as a 64K address space in which the following address ranges (hexadecimal) include:

Range	*Contents*
0000-7FFF	RAM
8000-8FFF	Screen Memory
9000-9FFF	One 4K Bank of "bank-switched" RAM
A000-E7FF	Processor-dependent ROM
E800-EFFF	Control Registers (I/O Addresses)
F000-FFFF	Processor-dependent ROM

When a processor switch occurs (either manually with the toggle switch or via program control) the following events take place:

(1) Memory is re-mapped to contain the appropriate ROM locations.

(2) The processor starts execution. The 6809 begins execution at the address contained in memory location FFFE. The 6502 begins execution at the address contained in memory location FFFC.

Bank Switching

Only one of the 16 4K banks is mapped into the address range 9000-9FFF at any moment. The current bank may be changed by placing the bank number of the new bank (0-F) in the bank-select register (see description of Bank-Select Latch).

Addresses in the I/O Page

The following devices or control registers are found in the input/output page:

address	*register*
E810-E813	PIA
E820-E823	PIA
E840-E84F	VIA
E880-E881	CRT Controller
EFF0-EFF3	ACIA (6551)
EFF4-EFF5	ACIA (6850)
EFF8*	System Latch
EFFC	Bank-Select Latch

* Also requires the bank-select latch, bit 7 to be 1.

Bank-Select Latch (EFFC)

This register is used to select the bank of memory accessible in memory locations 9000 to 9FFF. It is a one-byte latch:

bit	*use*
0-3	bank number
4-6	not used
7	must be 1 when accessing System Latch; should be zero otherwise.

Thus, the storing of the hexadecimal value 87 (10000111 in binary) causes bank seven to become accessible.

System Latch (EFF8)

This device is used to select (under program control) the microprocessor system and the memory protection for the bank-switched memory.

bit	*use*
0	select CPU (1=6502, 0=6809)
1	memory protect (1=read/write, 0=read/only)
2	not used
3	Diagnostic Sense
4-7	not used

The settings are ignored when the toggle switches are not set to "PROG", as described previously. Bit (3) should be set to 1 before a switch is made to the 6502 processor. This will cause the standard software in that processor to not write on all RAM memory, destroying its contents.

Notes

Chapter 3

Waterloo Micro Software Description

3.1 Overview of Waterloo Software

An extensive software package has been developed to satisfy many of the educational requirements at the University of Waterloo. This portable software is particularly suited to microcomputers, but identical versions will be available on medium and large-scale computing facilities. Thus a user is not limited by the capacity of the micro; the identical program will run without modification on larger and faster equipment.

The package consists of interpreters for various languages, an editor, an operating system (supervisor) and an assembly language development system.

(a) Language interpreters are included for four (4) programming language dialects known as

Waterloo microBASIC
Waterloo microPASCAL
Waterloo microFORTRAN
and Waterloo microAPL

These language interpreters have been designed specifically for educational use in the teaching of computer programming. The design of the interpreters features good error diagnosis and debugging capabilities which are useful in educational and other program-development environments.

Waterloo microBASIC includes ANS Minimal BASIC, with a minor exception, and several extensions such as structured programming control, long names for variables and other program entities, character-string manipulation, callable procedures and multi-line functions, sequential and relative file capabilities, integer arithmetic, debugging facilities, and convenient program entry and editing facilities.

Waterloo microPASCAL is an extensive implementation of Pascal, corresponding very closely to draft proposals being produced by the International Standards Organization (ISO) Pascal committee. The ISO draft language is a refinement of the language originally defined by Wirth, varying only in minor aspects. This implementation includes sophisticated features such as text file support, pointer variables, and multi-dimensional arrays. A significant feature of Waterloo microPASCAL is its powerful interactive debugging facility.

Waterloo microFORTRAN is a special dialect designed for teaching purposes. It has many of the characteristics and much of the flavour of normal FORTRAN, but varies significantly from established standards for that language. This language processor has many of the important characteristics of the WATFIV-S compiler which is widely used on IBM computers, plus some features from the new FORTRAN-77 definition. Examples of language features supported are FORMAT, subroutines and functions, multi-dimensional arrays, extended character-string manipulation, structured programming control and file input/output. In addition, the interpreter provides a powerful interactive debugging facility.

Waterloo MicroAPL is intended to be a complete and faithful implementation of the IBM/ACM standard for APL with respect to the syntax and semantics of APL statements, operators and primitive functions, input and output forms, and defined functions. System commands, system variables and system functions are those consistent with a single user environment. There are no significant design limitations on the rank or shape of arrays or the length of names. The shared variable processor is omitted. Extensions include system functions supporting files of APL arrays. APL equivalents of the BASIC features PEEK, POKE and SYS are included.

(b) A text editor, known as Waterloo microEDITOR, is suitable for creating and maintaining both program source and data files. It is a traditional line-oriented text editor with powerful text searching and substitution commands including global change. Full-screen support and special function keys allow text to be altered, inserted and deleted on the screen without entering commands. Facilities for repeating and editing previously issued commands further enhance the useability of this editor.

(c) File-oriented Assembler and Linker programs, known as the Waterloo 6809 Assembler and Linker, are included which support development of general-purpose Motorola 6809 machine-language programs. The Assembler supports syntax and directives for Motorola 6809 assembly language and includes powerful macro capabilities. In addition, the Assembler supports pseudo opcodes for structured programming control, long names (labels) for meaningful identification of program segments and data, and the ability to include definitions from separate files. The Assembler produces both a listing and a relocatable object file.

The Linker allows the combination of an arbitrary number of relocatable object files to produce an absolute loadable and executable program file. Since it is disk-oriented, the Linker is capable of building programs which are larger than the RAM workspace available. The Linker supports the building of programs in segments or banks for operation in bank-switched RAM memory, as well as supporting the building of programs for operation in normal RAM memory.

(d) The Waterloo microSUPERVISOR is an operating system designed for single-user microcomputer environments. It includes Monitor, Library and Serial Line Communications support as described below.

A Monitor program is included which supports the loading of Linker-produced program files into bank-switched RAM memory or normal RAM memory. The Monitor also provides facilities which are useful for debugging machine-language programs. These include commands to display and alter RAM memory and 6809 microprocessor registers, utilizing full-screen features for ease of use. In addition, a Monitor command permits disassembly of Motorola 6809 microprocessor instructions into assembly-language mnemonic form.

A Library of functions and procedures is included for general use by other programs included in the Package. The Library includes support functions for input/output operations to the keyboard, screen and peripheral devices. Other elements of the Library provide floating-point arithmetic, fundamental trigonometric functions, and several general-purpose utility functions.

A Serial Line Setup program is included which permits the selection of programmable characteristics, such as baud rate, of RS-232 serial lines. In addition, this program includes support for establishing communication with a host computer, through a serial line, for the purpose of accessing its files and peripheral devices.

3.2 Invoking Waterloo Software

When the Commodore SuperPET is turned on, the following operations should be carried out:

- the memory protection switch should be set to R/W
- the processor switch should be set to 6809
- the Commodore SuperPET is turned on (switch at back)
- the Waterloo distribution diskette should be inserted in the leftmost drive of the disk unit

At this point, the main menu will be displayed on the computer screen.

A Waterloo software component may be selected by entering the (abbreviated) name of that component as indicated in the menu. For example, to invoke Waterloo microFORTRAN, depress the "f" key followed by the RETURN key.

Other 6809 programs can also be loaded and placed into execution by entering the name of the file containing that program. When the file name does not contain a device specification, the system attempts to locate the file on the leftmost drive of the disk unit. The format of file names is given in a later chapter in this manual. The user should refer to the Waterloo 6809 Assembler manual for details about the creation of 6809 programs.

Notes

Chapter 4

File System – Generally

4.1 Introduction

This chapter introduces the notion of *files* and *devices* which are used to store, retrieve and display data. For example, a disk unit may be used to store a *file* of student marks. This file is accessible by other programs in order to produce summaries of the data such as marks reports. A device such as a printer may also be treated as if it were a file, although it is only useful for displaying data; an attempt to read information from this device will cause an error to be detected.

When programming in one of the Waterloo languages, the specific file name of a file or device is specified when that file is connected (usually with an OPEN statement) to the executing program. The remainder of the program may be written without knowledge of the specific file or device accessed. In this way, programs may be written in a device-independent fashion. The Waterloo software also provides the capability to use specific features of particular devices - the program, however, may not produce the required results on other devices which do not have these extended features.

A Waterloo language processor should be viewed as being connected to a number of devices. Some devices (such as disks) may be used to store a number of files.

Other devices (such as printers or a screen) have limited, special-purpose uses.

When a microcomputer is connected to a large computing system, the large computing system is treated as a device named "host". Files may be stored on and retrieved from such a system, provided the host system is executing a communication program controlling access to files on that system. At present, there exist communication programs (called "HOSTCM") for the IBM Series/I computer, IBM 370/303X/43XX/ computers with VM/CMS, and DEC PDP-11 computers with RSTS/E. These programs are not included with the Commodore SuperPET packages, but will be available from Waterloo.

4.2 Types of Files

A file should be viewed as a number of *records,* where each record is a sequence of zero or more characters. The Waterloo software supports three different *types* of files, depending upon the records which are to compose the file:

(a) *Text*

A "text" file consists of variable-sized records, containing only "printable" characters. When the type of a file is not explicitly given, this file type is automatically selected by the system.

(b) *Variable*

A "variable" file consists of variable-sized records, containing arbitrary characters. This file type should be specified in preference to "text" when the records contain data that is in a format that cannot be displayed on a screen.

(c) *Fixed*

A "fixed" file consists of fixed-sized records, containing arbitrary characters. When a program attempts to write a record smaller than the record size for the file, the system will automatically extend the record by adding space characters.

These file types are entered, within pararentheses, as the first part of a file name. The types may be abbreviated by deleting letters from the end:

```
(t),(te),(tex),(text)
```

The preceding are all valid specifications for a text file.

4.3 File Names

The general format of a file name is as follows:

```
(type:size)device . file-designator
```

The following defaults apply when not specified:

- The default *type* is "text".
- The default record *size* is 80 characters for fixed files; none is required for other types of files.
- The default *device* is the diskette drive attached to a Commodore SuperPET.
- There is no default *file-designator*.

The format of a file-designator depends upon the device on which a file resides. These are given with the description of the appropriate device. A file-designator is not required when the device is not used for actual file storage (i.e., a printer).

The *size* option in the file name gives the maximum number of characters in a record for that file. An attempt to write more than this maximum will cause the record to be truncated to the maximum. When the size is not specified, no maximum is enforced for variable or text files.

With most devices on which files may reside, a simple sequence of alphabetic characters is sufficient for file-designator part of the file name:

```
host.myfile
disk.student
(fixed:30)host.data
(text)data
```

The preceding examples illustrate several file names. On some devices, both upper- and lower-case characters can be specified; on others, they are treated as if only upper-case characters were entered.

Chapter 5

Files With Commodore SuperPET

5.1 Introduction

This chapter describes files on devices that are directly connected to the Commodore SuperPET system. It does not describe files or devices which may be present on host systems. As explained in the previous chapter, the general format of a file name is as follows:

(type:size)device.file-designator

Within this chapter, the "device.file-designator" portion of the file name will be described. This name will be referred to as the "filename".

5.2 Disk Files, Generally

The Commodore disk *units* used with this system each have two *drives*. A flexible disk can be placed in each drive. Multiple files can be stored on each disk and must be identified individually with names. A file-designator can be up to 16 characters long and can include blanks or other special characters:

```
disk.myfile
(text)disk.datafile
```

The preceding are two simple file names for files on the device "disk".

Multiple disk units can be attached to the system. Each unit is identified by a number or *address*. Normally the first disk unit attached to the system has its address set to 8. The two disks within each unit are designated by the numbers 0 and 1 for the rightmost and leftmost drives respectively:

```
disk8/0.myfile
disk8/0.datafile
disk9/1.goophy
```

The first two names are equivalent to the preceding example. The last refers to a file "goophy" on the leftmost drive of a second disk unit.

The general format of a file name is thus:

```
disk / drive . file-designator, format
```

where

(a) "disk" refers to the diskette unit, the first unit being "disk" or "disk8". By default, "disk" is used when this option is not specified.

(b) "drive" refers to a number 0 (right drive) or 1 (left drive). When the option is not specified, drive 0 is used.

(c) "file-designator" consists of 1 to 16 arbitrary characters, including special characters and spaces.

(d) The file format, if specified, should be "seq" (sequential) or "rel" (relative). When not specified, "seq" is used.

The difference between "rel" and "seq" files is the format in which data is stored on diskettes and the input/output functions which can be performed upon them. Each of the three file types (text, fixed, variable) may be used in either file format (seq or rel).

With "seq" format, the data is stored in a compact fashion. With "text" files, each record is terminated by a special character to mark the end of the record. "Variable" files have two extra characters inserted at the start of each record to contain the length of the record. In "fixed" files, the data in each record is stored without the addition of any extra characters.

In "rel" format, the data is stored in the manner described in the preceding paragraph. However, space is reserved for the largest possible record in each case. Thus, the "rel" files will, in general, occupy more space than an equivalent "seq" file. When not specified, the default size of 80 characters is used for the record size.

The advantage in using "rel" format is that the records in the file may be accessed in a random order. Thus, it is possible to read the ninth, second, and fourteenth records, successively. It is possible to access records in a file of "seq" format only in the order in which they were written.

5.3 Disk Files, Enhanced

Each disk file which is open is assigned to a *channel* or *secondary address*. Although a maximum of five files can be simultaneously open per disk unit, there are sixteen possible channels, numbered 0 through 15. Channel 0 is reserved for system use with operations such as listing file directories. Channel 1 is reserved for system use with operations such as BASIC's STORE command. Channel 15 is used for reading error status messages from the disk unit and writing special commands to the disk unit. Use of channel 15 for special disk commands is described in a later section. Channels 2 through 14 are available for general use with data files.

The general form of a disk filename specification follows:

disk *addr* - *channel* / *drive* . *file-designator* , *format*
where,
addr is the unit address (0-30),
channel is the file channel (0-15),
drive is the drive number (0 or 1),
file-designator is a string of up to 16 characters,and
format is the file format (seq, rel).

Defaults have been assigned to most of the parameters of the general specification to simplify ordinary use. These are shown in the following table of default values

parameter	*default*
addr	8
channel	automatically assigned
drive	0
file-designator	no default
format	seq

Examples follow which illustrate filenames and their meanings.

filename	*meaning*
charlie	file "charlie" on drive 0 of disk unit 8
disk.charlie	same as above
disk/1.charlie	file "charlie" on drive 1 of disk unit 8
disk9.charlie	file "charlie" on drive 0 of disk unit 9
disk	error/command channel of disk unit 8
disk-15	same as above
(f:80)customer,rel	relative file "customer" on drive 0 of disk unit 8
(f)customer,rel	same file as above; record size not specified for existing file (default of 80)

Note that if no file-designator is specified, it is assumed that the command channel 15, is to be used.

The ability to specify a channel number is provided to allow direct control of this parameter by sophisticated users having a detailed knowledge of the disk unit and the IEEE-488 bus. In general, it is recommended that this parameter *not* be specified.

5.4 The Printer

The Commodore printer can be considered as a special output-only file. Simply use the device name

printer

to specify the filename. Straight-forward printing can be accomplished using the filename printer in this form.

5.5 The Terminal Keyboard

The keyboard may be treated as a special device named "keyboard". As keys are depressed, the corresponding characters are transferred to the "keyboard buffer" by the Waterloo software. When the device "keyboard" is read, data in the keyboard buffer is transmitted to the program and is removed from the keyboard buffer. When the buffer is empty, a single character with hexadecimal value 00 is transmitted to the program.

When a program writes data to the "keyboard" device, the data is added to the keyboard buffer. This provides a mechanism whereby commands and/or data can be supplied to a program as if it had been entered by a user.

The keyboard buffer is 40 characters in length. When the buffer is full, any other characters transmitted to the "keyboard" device are ignored.

5.6 The Terminal Screen

The terminal screen may be treated as a special device named "terminal". When written to, data is transmitted to the screen, at the current position of the cursor.

When an attempt to read from this device occurs, the keyboard buffer (see Terminal Keyboard) is processed until a character corresponding to the RETURN key is encountered. This processing involves displaying the characters upon the screen at the position indicated by the cursor. When a RETURN character is encountered, the characters on the current line, as indicated by the cursor, are transmitted to the program. The space characters at the end of the line are not transmitted.

5.7 Serial Devices

The Commodore SuperPET has a port which allows serial communication with ASCII devices, such as printers or terminals, or a host computer. This port can be used for *host* communications (discussed in following chapters) if the personal computer is connected to another computer with this interface. The port can be designated for use with *auxiliary* devices such as ASCII printers or "slave" terminals (terminals which are not used to control the computer, but provide additional input/output capability).

Note that the Commodore printer does *not* connect to this port. Like the Commodore disk, it communicates using IEEE-488 protocol and is attached to the IEEE-488 data bus.

If the port is used for an auxiliary device, it is referenced by the device name:

```
serial
```

For example, an ASCII printer can be connected to this port and referenced with the filename, serial.

5.8 IEEE-488 Devices

The Commodore SuperPET is equipped with an IEEE-488 data bus port and Commodore disks and printers can be attached to it. Filenames for specifying these devices have been described in earlier sections. Other types of devices are available, such as plotters and laboratory monitoring equipment, which can communicate using the IEEE-488 bus. To reference such devices, which are connected to your computer, specify a filename of the following form

ieee *addr* - *channel*

where, addr is the device primary address (0-30),
and channel is the secondary address (0-30),

Default values for addr and channel are both 0. The following example filename refers to an IEEE-488 device with primary address 10 using secondary address (channel) 1:

```
ieee10-1
```

The Waterloo microSystems software handles the IEEE-488 data bus line-driving protocol for this type of communication. However, the programmer must handle any special requirements or protocol of the devices being used. Note that the Commodore disk and printer should *not* be referenced with this type of filename if it is desired that the software automatically handle the special characteristics of these devices.

5.9 Commodore Disk Command Support

The Commodore disk units used with this system contain microprocessors which are programmed to support a number of commands for managing the disk. The commands supported by the disk are described in the Commodore manual which is supplied with the disk unit. Since the Commodore manual is written for use with Commodore PET BASIC, the reader must adapt the examples given as illustrated below. These commands can be used with programs by opening the command channel 15 of the disk unit as a file and writing command strings to this special file.

Although Waterloo microBASIC statements are used to illustrate the use of disk commands, similar techniques can be applied using other Waterloo micro languages.

The following example illustrates use of the VALIDATE disk command to create a Block Availability Map and validate the directory of drive 0 on disk unit 9.

```
open  #2, 'disk9', output
print #2, 'VALIDATE 0'
```

In this example, "#2" is a file number that is associated with a file by the OPEN statement. The file number is used in the print statement to reference the file.

Similar techniques can be used to issue other commands to the disk. Substitute the appropriate command string for 'VALIDATE 0' in the above example.

Note:

Certain Commodore diskette commands (for example, INITIALIZE, NEW, DUPLICATE, or VALIDATE) cause the diskette unit to close *all* files on that unit although the computer system still considers them open. Such operations should not be performed while files are open on the diskette unit. Unlike Commodore PET BASIC, closing of the command channel does not cause the diskette unit to close all its files.

5.10 Commodore Printer Formatting Support

The Commodore printer used with this system contains a microprocessor which is programmed to support output formatting and allows definition of new characters. Like the disk, the printer has several channels or secondary addresses upon which it can receive data. When the filename "printer" is used, data is transferred on channel 0 resulting in the data being printed exactly as it is sent to the printer. Other channels provide special functions as described in the Commodore manual supplied with your printer. Since the Commodore manual is written for use with Commodore PET BASIC, the reader must adapt the examples given as illustrated below. These channels can be opened for output by specifying a filename of the following form:

printer *addr* - *channel*

where
addr is the printer address (0-30), and
channel is a number between 0 and 6 (2022) or 10 (4022).

Normally, only one printer is attached to a personal computer at address 4. If *addr* is not included in a printer filename, it is assumed by default to be 4. The Commodore printer attaches to the IEEE-488 data bus. Other terms sometimes used for *addr* and *channel* are *primary address* and *secondary address*, respectively.

The following example illustrates definition of a format string with Waterloo microBASIC using printer channel 2.

```
open  #5, 'printer-2', output
print #5, '$99.99'
close #5
```

In this example, "#5" is a file number that is associated with a file by the OPEN statement. The file number is used in other statements to reference the file.

Commodore has defined their own coding scheme for lower and upper case alphabetic characters, square brackets, backslash, uparrow and underscore characters. This coding scheme does not conform to the standard 7-bit ASCII code used by the Waterloo microSystems software. The Waterloo software automatically translates these characters as they are transmitted to the printer so that characters are printed in the form that they are typed at the keyboard and appear on the screen.

Note: A side-effect of this translation is that special format characters A, Z and S used in field format definitions output to channel 2 must be replaced by their lowercase counterparts a, z and s.

Notes

Chapter 6

File System – Host

6.1 Introduction

As mentioned previously, the Commodore SuperPET may be connected to a host computer system in order to access files on the host system. There must be a communications program (HOSTCM) executing on the host to control access to these files. This program will vary from host to host and consequently is not distributed as part of the Commodore software.

6.2 Communicating with the Host

In order that the Commodore SuperPET can communicate using the communication protocol as the host computer expects, a Host Setup program is available from the main menu of the Waterloo software. The use of this program is described in a subsequent chapter.

6.3 Files on Host Systems

The Waterloo languages and microEditor provide the capability to access host files as if these files were found on devices directly attached to the Commodore SuperPET. The languages require that a filename be specified (usually in an OPEN statement) before a file may be accessed. The format of a file name is given as

```
(type:size)host . file-designator
```

where the "type" and "size" options are as described in the chapter on general file systems. The string "host" indicates that the file resides on a host computing system. The file-designator portion represents a file name in the format of the host system. It is beyond the scope of this manual to provide the format for the host systems. This should be obtained by consulting documentation for the host system.

Generally, most host systems accept a filename of at least 6 characters in upper case:

```
host.DATA
(f:55)host.SPECS
(t)host.MYPROG
```

The preceding are examples of typical host file names.

Notes

Chapter 7

Waterloo microEditor

7.1 Introduction

Waterloo microEditor is a text editor suitable for creating and maintaining program source and data. It is a full-screen editor and utilizes program function keys. This chapter has two main sections.

(1) The tutorial introduction (subsection 7.2) is intended to provide an informal introduction to the editor.

(2) Subsections 7.3 and 7.4 provide a precise reference manual defining all the features of the editor.

7.2 Tutorial Introduction

It is essential that the reader work through this tutorial with the computer, in order that examples and experiments can be tried.

7.2.1 Introduction

The Waterloo microEditor may be invoked directly from the main menu. In addition, it is an integral part of the Waterloo microFORTRAN and microPASCAL systems and is automatically invoked when these systems are selected from the main menu. The editor may also be invoked with the EDIT command in the Waterloo microBASIC system.

When Waterloo microEditor is invoked the workspace is initially empty so there are no lines between the <beginning of file> line and the <end of file> line.

Note the two bright squares which appear on the screen. The top one is called the *text cursor* and the bottom one is called the *command cursor*. The text cursor marks the *current line*.

Many of the editor functions are performed using one of the 11 PF-keys (program function keys) PF0-PF9 and PF. . These keys are obtained by pressing shift and the appropriate numeric keypad key (0-9,.) simultaneously. For example PF8 is obtained by pressing shift and 8 on the numeric keypad simultaneously.

The CRSR (left) and CRSR (right) keys will move the cursor left and right on the line in order to facilitate corrections. The TAB key may also be used to tab to the right on the line. The INST (insert) key will make room for new characters on the line and the DEL (delete) key will remove characters from the line. The rubout key (labelled REPEAT) replaces characters with blanks. The erase-to-end-of-line key (labelled ESC) does exactly what its name suggests. The RUN key will execute the contents of the workspace when the editor is being used in conjunction with a language processor. The STOP key will generally terminate the current editor function.

All keys automatically repeat their function if held down.

7.2.2 Entering Data

PF0 will create a new line on which text can then be typed. Experiment by creating several lines of text.

Note that when PF0 was depressed the command cursor disappeared and anything typed then appeared on the newly created line. When there is no command cursor the editor is said to be in *screen mode*.

7.2.3 Making Changes

In order to change text simply use the CRSR (up)/(down) key to move the cursor up or down to the appropriate line and then type over what is there with whatever is desired. (PF7 and PF4 have the same function as CRSR (up) and CRSR (down).) The CRSR (left), CRSR (right), INST, and DEL keys are particularly useful in this application.

In screen mode, depressing the STOP key causes the current line to revert back to its original state. By "original state" is meant the contents of that line when the text cursor was last moved to that line. This key is useful to undo undesired changes to the current line.

Note that a new line can be added after any line in the file by moving the cursor to that line, depressing PF0, and then entering the new line.

7.2.4 Deleting Lines

In order to delete a line, position the cursor on it using the CRSR (up/down) key and then depress PF2.

7.2.5 Command Mode

Many functions of the editor are invoked by typing a command as distinct from using a PF-key. In order to enter a command, the editor must be in *command mode*. If there is only one cursor on the screen then the editor is in screen mode and PF5 must be depressed in order to enter command mode. A second cursor will then appear in the command area. All PF-key functions are available in command mode as well as screen mode.

The editor can be put in screen mode by depressing PF8. Screen mode is also entered when a new line is created by using PF0.

7.2.6 Positioning the Current Line

The editor can be positioned at a new current line simply by typing the line number of the desired current line. For example, typing a 6 makes the sixth line in the workspace the current line. Line 0 is the <beginning of file> line. In order to specify the last line in the file a "$" (dollar sign) may be used. Simple arithmetic expressions can be used. For example, $-6 specifies the seventh-to-last line in the file.

The current line can be moved relative to its current position by specifying a + (plus sign) or − (minus sign) followed by the number of lines to move. For example, +6 moves the current line ahead six lines. PF-keys may also be used to position the current line.

PF6 moves back one line (−1 command)
PF3 moves ahead one line (+1 command)
PF9 moves back one screen (−20 command)
PF. moves ahead one screen (+20 command)

7.2.7 Locating Lines by Content

The editor can easily search the workspace for a string. For example, the command

```
/sometext
```

causes the *first* line in the workspace containing the string "sometext" to become the current line.

The line containing the *next* occurrence of the word "sometext" can be located by typing

```
+/sometext
```

Similarly, the line containing the *previous* occurrence can be located using

```
-/sometext
```

7.2.8 Global Changes

Often it is desireable to make a "global" change to the workspace. This involves changing *all* occurrences of a particular string of text to another string of text. For example, we may wish to change all occurrences of the word "line" to "string". This is done by entering the command

```
*c*/line/string
```

7.2.9 Clearing the Workspace

The workspace can be cleared using the global delete command

```
*d
```

Note that the simple delete command

```
d
```

deletes only the current line.

7.2.10 Saving the Workspace

Once editing is completed, it is necessary to save the workspace on a diskette file. This is done with the "put" command. For example, the command

```
p  mytext
```

causes the workspace to be saved on the diskette under the name "mytext". After the operation is completed, a message is displayed stating the number of lines saved.

The file can be retrieved by first clearing the workspace with the "*d" command and then recalling the file with the "get" command as follows:

```
*d
g  mytext
```

7.3 The Full-screen Display and PF-Keypad

The full-screen display provides a "window" through which a block of lines in a file may be seen. Using the program function keys, text may be altered, inserted or deleted directly on the screen in an easy and user-efficient manner.

The full-screen display divides the screen into three areas:

(1) a "window" on the file,

(2) a one-line area for command input, and

(3) a one-line area where messages are displayed.

The layout for the display is shown below:

```
.-------------------------------------------------------------.
|                                                             |
|                                                             |
|                                                             |
|                   "window"  on the file                     |
|                      (text  area)                           |
|                                                             |
|                                                             |
|                                                             |
|-------------------------------------------------------------|
|                   command  entryarea                        |
|-------------------------------------------------------------|
|                  message display  area                      |
'-------------------------------------------------------------'
```

One of the lines displayed in the text area is referred to as the *current line* and is designated by the *text cursor*. Editor commands typically refer to the current line in some way, as detailed in the command descriptions.

Program function (PF) keys are provided which execute commonly-used editor commands. PF-keys are obtained by pressing shift and the appropriate numeric keypad key simultaneously. The numeric keypad is arranged in the following manner:

7 move current line up one in window	8 enter screen mode	9 move window up one screen in file
4 move current line down one in window	5 enter command mode	6 move window up one line in file
1	2 delete current line	3 move window down one line in file
0 insert blank line after current line		• move window down one screen in file

The CRSR (up/down) key provides the same function as PF7 and PF4.

The editor has two modes of operation. In *command mode* there is a cursor in the *command area* where commands may be typed while the current line is marked by the text cursor in the text area. In *screen mode,* only the text cursor appears and the user may move it anywhere in the text area and modify the file. PF-keys may be used in screen mode or command mode. Commands may only be typed in when in command mode. PF8 switches the editor to screen mode and PF5 switches the editor to command mode.

7.4 Command Syntax and Semantics

7.4.1 Specifying Individual Lines

A line in the file is specified by its "<line-id>", where a <line-id> is one of the following:

<line-specifier>	the line indicated by <line-specifier>
<line-specifier>+n	the line n lines after the line indicated by <line-specifier>
<line-specifier>−n	the line n lines before the line indicated by <line-specifier>

A <line-specifier> is one of the following:

$	the last line in the file
.	the current line
n	the "nth" line in the file
+	the line after the current line
+n	the "nth" line after the current line
−	the line previous to the current line
−n	the "nth" line previous to the current line
/text/	the first line in the file that contains the search string "text"
+/text/	the first line following the current line that contains the search string "text"
−/text/	the first line preceding the current line that contains the search string "text"

A <line-specifier> may also be null, in which case it indicates the current line.

7.4.2 Search Strings

A search string is simply a character string for which the editor searches the workspace or current line. Search strings are typically enclosed in slashes (/) although the closing slash may be omitted if it is the last character on the command line.

The special search string "//" represents the last non-null search string that was used in any command.

There are some character sequences (known as meta-characters) which have special meanings. These are:

%/	represents a "/"
%↑	represents the beginning of a line
%$	represents the end of a line
%.	represents any character
%*	represents zero or more repetitions of the character which immediately precedes the "%*"
%%	represents a "%"

There is one additional meta-character which applies only to the replacement text in a change command (Section 2.6.3). It is "%&", and it represents the text that was matched by the search string of the change command (if any).

Examples

/%↑abc	represents "abc" occurring at the beginning of a line
/abc%$	represents "abc" occurring at the end of a line
/a%*bc	represents 0 or more "a"'s followed by "bc" ("bc", "abc", "aabc", "aaabc", ...)
/a%.c	represents an "a" followed by any character, followed by a "c" ("aac", "abc", "acc", "adc", ...)
/ab%.%*d	represents "ab" followed by any string of any length (including the empty string) followed by a "d"

7.4.3 Specifying Line Ranges

Many of the editor commands may be applied not only to a single line, but also to a number of lines. The command descriptions in the next section use the notion of a "<line-range>", where a <line-range> is any of the following:

<line-id>	the line indicated by <line-id>
<line-id>,<line-id>	all lines from the first <line-id> to the second <line-id> if either of the <line-id>'s is omitted, then "." (the current line) is assumed
*	all the lines in the file (equivalent to the range 1,$)
*/text/	all lines containing "text"
<line-id>↑	the 20 lines preceding <line-id> (equivalent to the range ".−20,."; displays the previous screenful when used alone)
<line-id>&	the 20 lines following <line-id> (equivalent to the range ".,.+20"; displays the next screenful when used alone)

Examples

5,10	lines 5 to 10 inclusive
,+5	the current line and the five lines following it (could equivalently be expressed as .,.+5 or .,+5 or ,.+5)
−5,+5	the five lines preceding the current line, the current line and the five lines following the current line (11 lines total)
−/abc/,+/def/	all lines from the first line found containing "abc", when searching backwards from the current line, to the first line found containing "def" when searching forward from the current line

*/%↑test/	all lines starting with "test"

7.4.4 Commands

In the descriptions that follow, the <line-id>'s and <line-ranges>'s are optional. If they are not specified "." is used. In the syntax model for each command, the lower-case characters represent the minimal truncation of the command. For example, in its syntax model, the directory command is specified as "diRECTORY" indicating that "di" is an acceptable abbreviation.

7.4.4.1 (Number-sign)

Syntax:	#
Description:	The number-sign command ("#") displays the current line-number.

7.4.4.2 (Question-mark)

Syntax:	?
Description:	The question-mark command ("?") recalls the last command so that it may be altered and resubmitted.

7.4.4.3 Bye

Syntax:	bye
Description:	The bye command terminates the current session.

7.4.4.4 Change

Syntax:

```
<line-range>cHANGE/text1/text2/
<line-range>cHANGEn/text1/text2/
<line-range>cHANGE*/text1/text2/
cHANGE
```

Description: There are two fundamental types of change commands available in the editor. The first, illustrated by the first three models above, changes the text denoted by "text1" to the text denoted by "text2" in all the lines denoted by the <line-range>. The first form changes only the first occurrence of "text1" to "text2", whereas the second and third forms change the "nth" and all occurrences, respectively. The second type of change command, illustrated by the fourth model above, allows screen editing of the current line.

Examples:

1)	c/a/b	change the first occurrence of "a" to "b"
2)	c*/y/z	change all occurrences of "y" to "z"
3)	c	display the current line for alteration and replacement

7.4.4.5 Copy

Syntax:

```
copY <input-filename>  <output-filename>
copY <input-filename> to <output-filename>
```

Description: This command may be used to produce a copy of the file specified by <input-filename> in a file specified by <output-filename>.

Examples:

```
copy  disk.test.data  disk/1.test.backup
copy  source.file,prg to disk9.source.file,prg
copy  "data file 1" to "data file 2"
```

7.4.4.6 Date

Syntax:

```
datE <date-string> <time-string>
datE
```

Description: This command sets or displays the current date and time. When a date and time is specified the current date and time are set, otherwise they are displayed. <date-string> may be any character string up to 11 characters long. The date should be enclosed in quotation characters (") if it is to include space characters. <time-string> must be in the format *hh:mm:ss:tt* where *hh* is hours, *mm* is minutes, *ss* is seconds and *tt* is ticks (60ths of a second).

Examples:

```
date 81/06/30 10:45
date "SEP 25 1980"
```

7.4.4.7 Delete

Syntax: <line-range>dELETE

Description: The delete command deletes all lines in the specified line range.

Examples:

1)	*d	delete all lines
2)	1,10d	delete lines 1 through 10 inclusive

7.4.4.8 Directory

Syntax: diRECTORY <devicename>

Description: The directory command lists the names of all the files that reside in the specified directory.

Examples:

di disk	list the directory of the default diskette
di disk9/1	list the directory of diskette unit 9, drive 1

7.4.4.9 Get

Syntax:

```
<line-id>gET <filename>
<line-id>gET
```

Description: The get command causes the entire contents of the specified file to be inserted following the line specified by <line-id>. If no filename is specified then the current filename is assumed. If a filename is specified then it becomes the current filename.

Example:

$g demo	insert the file named "demo" after the last line in the file

7.4.4.10 Help

Syntax:	hELP
Description:	This command causes a summary of the editor commands to be displayed.

7.4.4.11 Input

Syntax:	<line-id>iNPUT
Description:	The input command sets the current line to <line-id>, and then puts the editor into input mode. Subsequent text is entered into the file, until a line containing only a dot (".") is entered.

Examples:

1)	i	enter input mode at the current line
2)	3i	enter input mode after the third line

7.4.4.12 List

Syntax:	<line-range>
Description:	Text may be viewed ("listed") simply be specifing the range of text to be listed. In other words, the "list" command is the default command if no other is given. The current line is set to the last line in the line range. In the special case where <line-range> is not specified (i.e. the null command), the next line in the file is displayed (and becomes the new current line).

Examples:

1)	*	list the entire file; the current line becomes $
2)	1	list the first line and set the current line to that line
3)	1,.	list all lines from the first line to the current line; the current line remains the same

7.4.4.13 Mount

Syntax:	mouNT <devicename>
Description:	This command causes the named device to be prepared for access. This command should be issued when a new diskette is placed in a 2040 disk drive.

7.4.4.14 Name

Syntax:	nAME <filename>
	nAME
Description:	This command sets or displays the current filename which get and put commands use by default. When no filename is specified the current filename is displayed. When a filename is specified it becomes the current filename. Note that a put or get command which specifies a filename will cause the current filename to be set to the filename specified.

7.4.4.15 Put

Syntax:	<line-range>pUT <filename>	
	<line-range>pUT	
Description:	The put command copies all the lines specified by the <line-range> into the file specified by the <filename>. If the <line-range> is not specified then the entire file is copied (equivalent to "*p <filename>"). If a filename is not specified then the current filename is assumed. If a filename is specified then it becomes the current filename.	
Example:	p demo	put the entire file into the file named "demo"

7.4.4.16 Rename

Syntax:

```
renAME <filename> <filename>
renAME <filename> to <filename>
```

Description: This command causes the file specified by the first filename to have its name changed to the second filename.

7.4.4.17 Run

Syntax:

```
run
```

Description: When the editor is being used in conjunction with a language processor the run command executes the file. The RUN key has the same function.

7.4.4.18 Scratch

Syntax:

```
scrATCH <filename>
```

Description: This command causes the named file to be deleted.

7.4.4.19 Tabset

Syntax:

```
tabSET <tab-1> <tab-2> <tab-3> ... <tab-n>
tabSET
```

Description: This command sets or displays the current tab stops. These define the positions to which the cursor jumps when the tab key is pressed. When tab stops are specified the current tabs are set, otherwise they are displayed. A maximum of 10 tab stops may be specified.

7.4.4.20 Talk

Syntax: tALK

Description: The system becomes a terminal of the host system ("pass-thru" mode). Talk mode is terminated with the STOP key. The HOST SETUP program should have been previously invoked in order to create a suitable communications environment for use with the host.

7.4.4.21 Time

Syntax:

```
timE <date-string> <time-string>
timE
```

Description: This command is identical to the DATE command, except that only the time is displayed when no arguments are specified in the command.

Chapter 8

Host Setup Program

8.1 Introduction

The Commodore SuperPET is equipped with an RS232 line connection. This may be used to connect the personal computer to a remote computer called a *host*. Once connected in this way, the Commodore SuperPET may be used as a terminal to the host computer. By issuing the TALK command in the Waterloo microEditor, the user may directly communicate with the host computer using the keyboard. A program executing in the Commodore SuperPET may also directly communicate with the host by reading from and writing to the device "serial".

Alternatively, the host computer may be used as a device of the personal computer. When used in this latter fashion for file storage, a communications program must be executing on the host computer. This communication program controls access to files on the host. The program is not supplied with the Commodore SuperPET software but may be obtained for various computer systems from Waterloo. The RS232 line may also be used to connect auxiliary devices (such as an ASCII printer) to the Commodore SuperPET.

Host computers and other devices communicate using the serial line in various ways. In order to inform the Commodore SuperPET of the manner in which a host expects to communicate, a SETUP program may be invoked from the main menu.

The SETUP program displays the various communication options, together with the default values for the options. These values may be changed to those expected by the host by typing new values in place of the inappropriate default values. This is accomplished by positioning to the default value using either the TAB key or the cursor positioning keys and then entering the new value. When all the new values are entered, depressing the RETURN key causes the Commodore SuperPET to be reset to reflect the new line options and the main menu is then displayed.

If a response to an option is invalid, a "?" character replaces the erroneous value and the options are displayed again. At this point the value may be corrected and the RETURN key depressed another time.

A user should consult the manager of the host system for the appropriate values to be entered for the SETUP options. The default options are designed to be used with VM/SP on an IBM/370 computer.

8.2 SETUP Options

When the SETUP program is invoked from the main menu, the following is displayed:

```
Baud          2400
Parity        EVEN
Stopbits      1
Prompt        11
Lineend       0D
Response      13
```

In this section, each option will be described and the valid responses outlined. When an auxiliary device is connected or when the microcomputer is used as a terminal, only the first three options are significant.

Baud

The baud rate establishes the speed at which information is transmitted on the serial line. The following values are permissible:

50	75	110	135	150	300	600
1200	1800	2400	3600	4800	7200	9600

Parity

There are four possible entries for the parity option: even, odd, mark, space.

Stopbits

Either one or two "extra" bits are sent between characters of information. Consequently, the valid responses are 1 or 2.

Prompt

The prompt string is the character sequence sent by the host to indicate to the microprocessor to send another line of input data. The prompt string may be up to four characters in length and is entered in hexadecimal. The default value is hexadecimal 11, sometimes called the XON character.

Lineend

The lineend character is appended to each line sent to the host in order to signal the end of a line. The value of this option is a single character, entered in hexadecimal. This character is used by the host to determine the end of a line being received from the microcomputer. The default value is hexadecimal 0D, the ASCII carriage-return character.

Response

The response character is sent by a host whenever the host has received a line of data. The value of this option is entered as a single character, in hexadecimal. This is the last character of a stream received from the host, in response to a lineend character received from the microcomputer. The default value is hexadecimal 13, sometimes

Commodore SuperPET Overview

Appendices

Appendix A

Floating Point Emulation

There is no hardware supplied with the Commodore SuperPET to perform calculations using floating point numbers. The Waterloo microSoftware is written in a manner that these operations are performed by the library software. This appendix provides an overview of the format of these numbers.

Each floating point number is stored using five bytes (or 40 bits). The first bit is used for the sign of the number (1 implies a negative number). The next eight bits are the exponent (base 2) in excess 128 notation. The mantissa is the remaining 31 bits and represents a 32-bit binary fraction in which the first bit is always one. Zero is represented as 40 zeroed bits.

The representation described permits approximately nine digits of accuracy. As is well-known, the potential error will increase as more calculations are performed using these numbers. However, the total error is usually insignificant compared to the final result of most computations. It is beyond the scope of this manual to analyze these problems in detail.

All calculations are performed internally using five-byte mantissae. This tends to minimize the error in extended calculations.

Appendix B

Waterloo Library

When using the 6809 microcomputer system, there is a collection of library routines available to be used with various software programs. These routines are described in the Waterloo 6809 Assembler manual. The locations of these routines are found, with the routine name, in the file "WATLIB.EXP" found on the distribution diskette.

Appendix C

Screen Handling

In this appendix, several features of the screen are discussed.

Fonts

As described previously, there are four different fonts available:

(1) Commodore upper/lower Roman font
(2) Commodore graphics font
(3) Waterloo upper/lower Roman font
(4) Waterloo APL font

At any point, exactly one of these is the current font used to display characters on the screen.

The character representations available in a font may be viewed by writing a program which displays, as character values, the numbers from 0 to 255. This could be accomplished in Waterloo microFORTRAN with the following program:

```
do  i = 0,  255
     print,  i,  char(i)
enddo
end
```

Similar programs can be written in the other programming languages.

Font Selection

The fonts are organized into pairs: the Commodore fonts and the Waterloo fonts. To select the Waterloo fonts, the following one-byte hexadecimal values should be transmitted to the following locations:

location	*value*
E880	0C
E881	30

The Commodore fonts may be selected by transmitting the following one-byte hexadecimal values to the following locations:

location	*value*
E880	0C
E881	10

This transmission may be accomplished by using the "POKE" facility found in the various languages. Once the correct pair of fonts has been enabled, one of the pair may be selected by transmitting one of the following one- byte hexadecimal values:

location	*value*	
E84C	0C	(select font 1)
E84C	0E	(select font 2)

Removing Extra Display Lines

As described previously, each character is displayed using an 8 by 8 dot matrix. By default, there are two extra blanked rows of dots between each line. Display of these two extra rows may be eliminated by transmitting one-byte hexadecimal data to the following locations (in the order given):

location	*value*
E880	04
E881	28
E880	05
E881	05
E880	07
E881	21
E880	09
E881	07

This transmission may be accomplished by using the "POKE" feature found in the various languages. The display of the two extra lines may be enabled by transmitting one-byte hexadecimal data to the following locations (in the order given):

location	*value*
E880	04
E881	20
E880	05
E881	03
E880	07
E881	1D
E880	09
E881	09