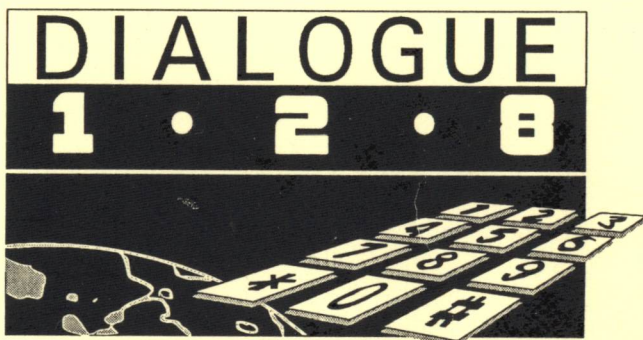
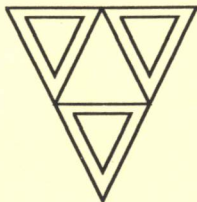


***Triple Point Software***



---

***DIALOGUE 128***

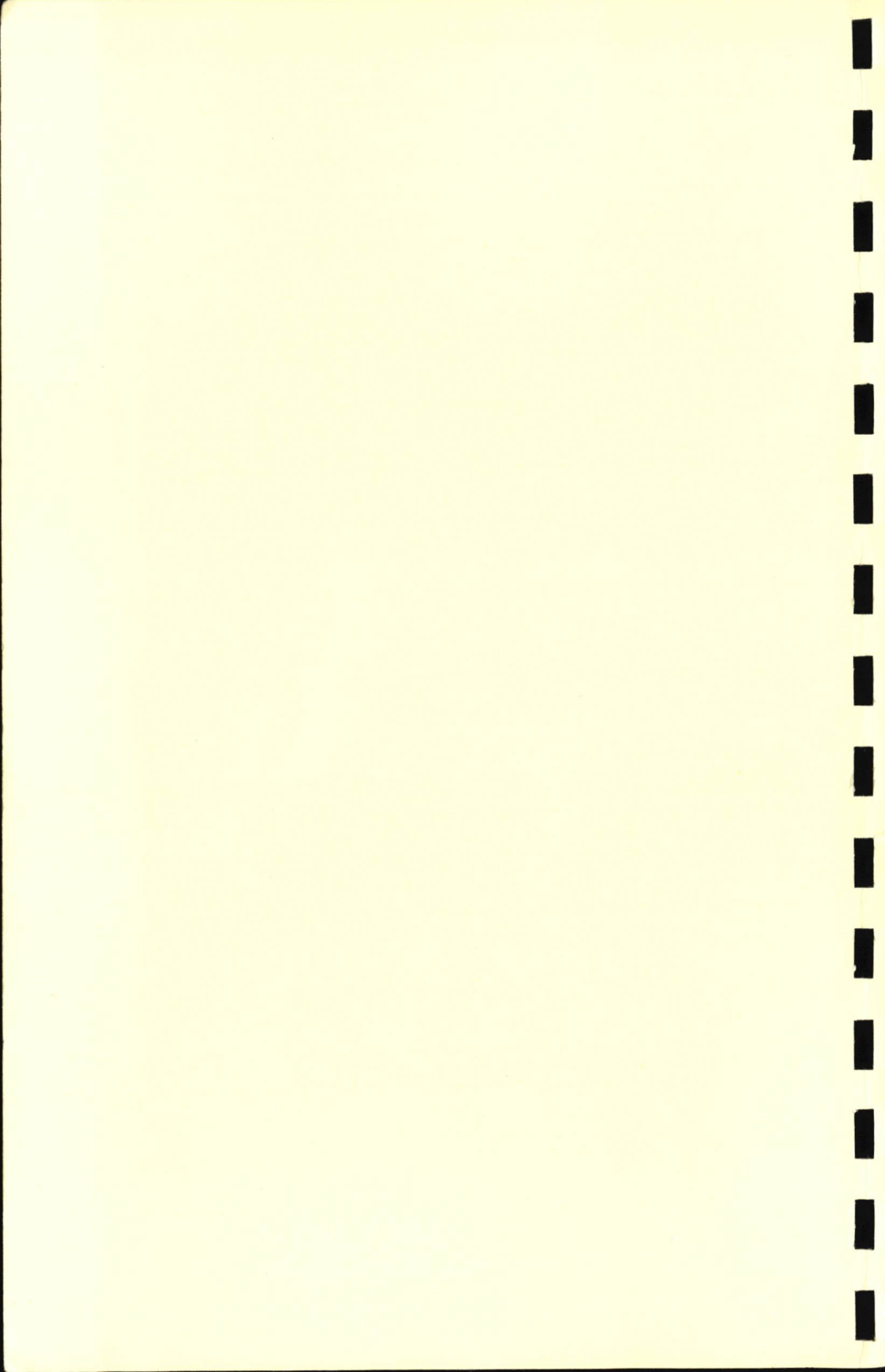
**Version 2.2**

---

**Superior Terminal Communications Software  
for the Commodore 128 and 128D computers**

**Written by Gary Farmaner  
COPYRIGHT © 1991 Triple Point Software**

---



## DIALOGUE 128 PHONE BOOK DELETE

There seems to be no command to delete an entry from the Dialogue 128 phone list, but you can edit the data to remove an item.

To begin, use 'C= C' to access the Configuration screen. Cursor over and down to "Set Default Files" and press RETURN. Read and remember the name of your phone number file, often "phone.num". Key RUN/STOP to exit the window, and R/S again to exit the Configuration screen.

Now use 'C= A' to access the Autodial screen. Cursor down to the line you want to delete. Press 'E' for edit (not CBM E as the manual states). You will see the system name on the status line, with the cursor at the beginning of the name. Cursor over to the end of the name and use DEL to delete all of it, then key RETURN. You will then see the system phone number. Cursor over to the end of the number, use DEL to remove it, and key RETURN. Key 'N' to bypass the function key definitions, and R/S to exit from the mini-config screen and get back to the phone number list.

Now you may have a list of bulletin boards and servers with a blank line in the middle. One way to get rid of the blank line is to rearrange the entries. Move the highlight to the next line below the blank one, and key 'R'. The entry will flash. Move to the blank line and key RETURN. The highlighted entry and the blank line will exchange places. Repeat this procedure until the blank line is at the end.

Finally, you need to save the revised phone book. Key 'S' to save, and the program will ask for the file name. Key in the part of the file name you remembered that comes before the period; for "phone.num" you enter "phone". The program will add the suffix ".num" and look for the file. Since it already exists, the program will ask if you want to replace the file. Reply 'Y' and the new file will be saved.

*Triple Point Software*



---

***DIALOGUE 128***

Version 2.2

---

**Superior Terminal Communications Software  
for the Commodore 128 and 128D computers**

**Written by Gary Farmaner  
COPYRIGHT © 1991 Triple Point Software**

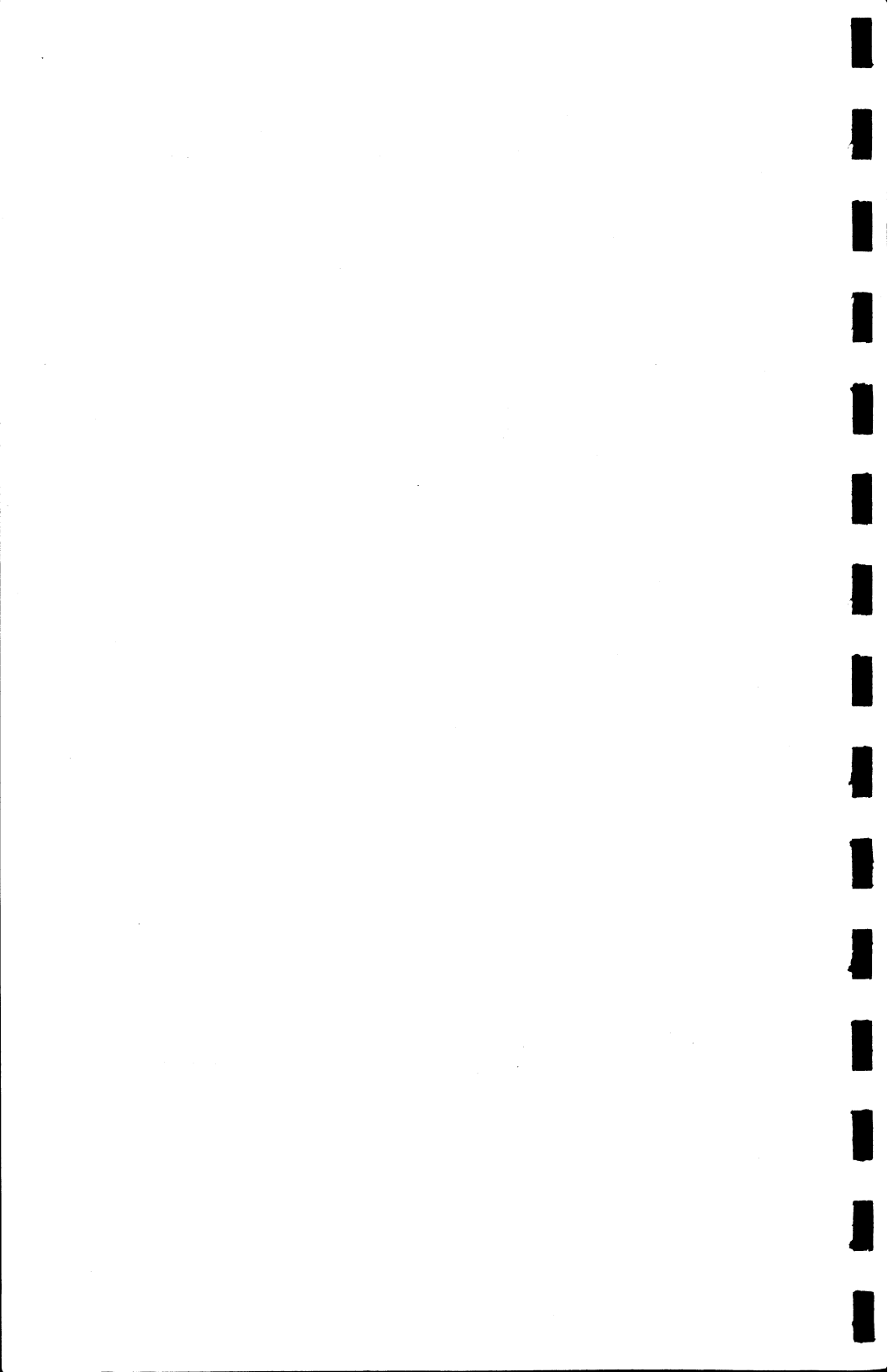
---

**Published by:  
*Triple Point Software*  
#369- 253 College Street  
Toronto, ON, Canada  
M5T 1R5**

---

**Dialogue manual written by Jeff Goebel and Gary Farmaner.  
For V2.2 it has been edited, updated and reset by Malcolm  
O'Brien. Copyright © 1991 *Triple Point Software*.**

**Artwork by Chester Chen  
Copyright © 1989**



---

# Dialogue 128

## Table of Contents

---

Introduction .....	1	■
Getting Started, Equipment Required, Creating a Work Disk		
Loading Dialogue .....	3	■
Layout of Dialogue .....	4	■
Status Line, Keyboard Equivalents		
Command List and Outline .....	7	■
Highlight bar, prompts, File Selection Mode		
Configuring Dialogue .....	14	■
Autodialing .....	19	■
Protocols, Uploading & Downloading .....	24	■
Transmit Modification Characters .....	29	■
Buffer Functions .....	30	■
The Buffer Editor .....	34	■
Macro Keys .....	36	■
Terminal Mode Commands .....	37	■
Auto-Exec Script Language .....	42	■
<hr/>		
Appendix A: Modem Support Files .....	52	
Appendix B: Dialogue Extension Files .....	56	
Character Editor	RLE Decoder/Viewer	
Translation Table Editor	Two Drive Copier	
Dvorak Keyboard	(RAMDOS-compatible)	
Appendix C: Control Codes .....	61	

---

---

## Dedication

I'd like to single out the following people for their enormous encouragement, help and/or support throughout the Dialogue 128 development process:

Bill Roberson, Jim Russell, Ed Flinn, Marte Brengle, Joe Buckley, Malcolm O'Brien, Kent Sullivan, Miklos Garamszeghy, Darrell Grainger, Jeff Goebel, Steve Punter, Mark Fellows, Doug Cotton, and Frank Hudson.

Thank you all,

Gary Farmaner

---

---

## An Introduction to Dialogue V2.2

*Dialogue* has many outstanding features. Among them are:

- VT52 and VT100 terminal emulation
- 300/600/1200/2400/3600/4800 baud rates allowing level 5 MNP protocol
- 7200-38,400 baud available in UART versions
- User-selectable variable baud rates from 0-600 supported
- Support of most major modem types through individual modem support files
- All commands accessible from keyboard, drop-down menus, joystick, or 1351 mouse
- 40/80-column selectable display (on 80-column screen)
- Optional 25- or 50-line mode
- All operations in FAST 2MHZ mode
- Full support of 1700 and 1750 RAM expanders, and 128 RAMDOS
- 64,000 character capture buffer (800/1600 lines)
- 9,000 character separate scrollbar review buffer
- Optional 512,000 character buffer with RAM expander (6,400/12,800 line)
- Multiple buffer configurations (1 to 8 separate capture buffers)
- Full-featured integrated text editor
- Extensive auto-dial/redial capabilities
- Multiple 30-entry phone directories
- Individually defined terminal parameters for each system
- Powerful auto-execute script language allows full unattended and automatic operation
- Punter C1, Ymodem Batch, Compuserve Quick B, Xmodem checksum/CRC/1K protocols
- Loadable extension files vastly extend *Dialogue's* future capabilities
- Partition and sub-directory support for 1581 disk drives and hard drives
- Burst mode support for 1571 and 1581 disk drives
- Colour graphics support for both CBM and IBM-type ANSI bulletin boards
- User-defined character sets - and fully integrated character editor included
- Disk is not copy-protected. Load *Dialogue* from any type of drive!

**Limited Warranty:** *Dialogue 128* is provided "as is" without software warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. *TriplePoint Software* warrants the media only, and will replace, free of charge, any disk found to be defective within 90 days of purchase. *Dialogue 128* is not copy-protected. You are encouraged to make a backup copy of the master disk and we recommend that you make several work disks. Once you've tried *Dialogue 128* you won't want to be without it! At *Triple Point Software*, we believe that copy-protection only serves to annoy the legitimate user. We show our respect for you, the customer, by disdaining copy-protection. You can demonstrate your respect for the author and publisher by refraining from distributing illegal copies. We appreciate your support.

**Copyrights:** Commodore 128 and 128D, and Amiga are all registered trademarks of Commodore Electronics Limited. CompuServe, GEOS, Hayes, VT100 and VT52, Touch-Tone, SwiftLink, IBM, Atari, Doodle, GENIE, CMD, and Xetec are trademarks or registered trademarks of their respective companies.

**Authors:** If you have written high-quality software that you would like to have published, Triple Point Software welcomes your submission.



---

## Getting Started

In addition to this manual, you should find the following in the *Dialogue* package:

- *Dialogue* diskette (5 1/4-inch 1541 format "flippy")
- *Dialogue* registration card

We urge you to complete, and return the registration card as soon as possible.

## Benefits of Registration

- Information on updates to *Dialogue*
- Announcements of new *Dialogue* extension files
- Recognition on major online services as a legitimate owner

## Equipment Required to Operate *Dialogue*

- Commodore 128 or 128D computer
- Commodore 1541 or 1571 disk drive or equivalent
- An 80-column video monitor (colour or monochrome)
- A compatible modem
- A blank work disk of any format
- Commodore model 1700 or 1750 RAM expansion unit (optional)
- Joystick, or Commodore 1351 proportional mouse (optional)

## About this manual:

There are so many commands and features of *Dialogue*, it was tough to squeeze them into a manual without confusing everybody. The result is a manual which describes the different areas of the program in segments, starting with generic keyboard information, followed by more specific command information. Almost all of the commands are listed by their function, and 'hot key'. We've spent a great deal of time attempting to make things fast and easy to find. If you read the manual once, you probably won't remember everything, but you'll almost definitely be able to quickly locate what you need to refresh your memory later.

As well, a 'quick reminder' references card is included listing all available commands, and their keyboard equivalents. If in doubt, commands are always available by pressing the **HELP** key on the keyboard.

---

## Creating a *Dialogue* Work Disk

*Dialogue* is supplied on a single-sided 1541 format diskette. This you can treat as the master disk. It is recommended that you do *not* continue to use the original disk, because it may become damaged. Supplied with *Dialogue* is a special 'Work Disk' creation program that installs a working copy of the files on a blank disk. Once the work disk has been made, you may store the original *Dialogue* disk away in a safe place. To create the work disk, place the original *Dialogue* disk in any 1541-compatible drive.

Enter **RUN"WORK DISK"** (or **RUN"WORK DISK",U#** where # is the drive's device number)

The work disk program provides full in-program documentation, and will guide you through the steps required. When complete, a working copy of *Dialogue* will reside on your new disk. Work disks may be any format - 1541, 1571, 1581, hard disk, etc.

When creating a work disk, the program will install a boot sector, making the work disk 'autobootable' from device 8. This means *Dialogue* will automatically load when the computer is turned on or reset.

## Loading *Dialogue*

Now that you have a work disk created, you can begin to use the terminal. From this point on, all activity is performed using the *Dialogue* work disk only. The original should be returned to the pocket in the front of the binder.

Enter **RUN"DIALOGUE"** (or **RUN "DIALOGUE",U#** where # is the drive's device number)

If the work disk is in device 8, typing **boot** or resetting the computer will automatically load *Dialogue*. Entering **boot u#** will also work if the work disk is in a drive other than device 8.

The *approximate* load times for various drive setups are:

Drive	Load time (minutes:seconds)
1581 w/JiffyDOS	0:24
1581	0:28
1571	0:57
IEEE	1:00
1541	2:10

**Note:** If you have a 1700 or 1750 RAM expansion unit, it will automatically be detected. If RAMDOS is installed in page 11, *Dialogue* will recognize this, and operate as normal, with full support of RAMDOS. If installed in another page, it will be relocated to page 11. Installation of RAMDOS requires that **ramdos128.bin4.3** (or **4.4**) be contained on the work disk.

When *Dialogue* has completed the load procedure, the main configuration window is displayed, and the system files are installed, along with the modem file you've chosen, and any other default files. The configuration window then disappears and the status line will reappear flashing text indicating this is the **Terminal mode**, and you're ready to use *Dialogue*!

---

## The Layout of *Dialogue*

### Terminal Mode

Terminal mode is the center of the *Dialogue* universe. This is where you'll interact with the other online systems, and with all the various features of *Dialogue* itself. Any characters that are received over the modem, will be displayed on the screen, and characters you type on the keyboard will be transmitted back out to the modem.

The top line in reversed text is called the status line. This line gives you an overview of the status of some important functions of the terminal.

The status line looks something like this:

```
C: 0:00:00  T: 0:00:00am  LR: 800 1  LC: 000  A:C:D:N:O:S  Terminal mode
```

From left to right, the particular sections of the status line have the following meanings:

**C: 0:00:00** This is the display of the elapsed 'Connect time' for each logon. When *Dialogue* detects a carrier, it will reset this clock to 0:00:00 and start counting. As an option, *Dialogue* can be programmed to save connect times of each system when you complete the online session. (See: Time Log Review. Page 22)

**T: 0:00:00am** This is the real-time clock, and can be used two ways. When the program is first loaded, the clock can be reset to zero to display the elapsed time since the load occurred. This would let you know how long you've been using *Dialogue*. Or you can set it to the 'real' time (or set the time from a realtime clock cartridge, if you have one installed). (See: Setting the Clock. Page 37)

The layout for both timers is **Hours:Minutes:Seconds** (HH:MM:SS).

**LR: 800 1 LC: 000** This is the default buffer status. It is composed of three sections. **LR:** Indicates the Lines Remaining in the current buffer. **1** This is the current buffer selected. *Dialogue* can have up to eight different buffers (numbered 1-8). The number of the currently selected buffer will be displayed here. This number is highlighted, and will flash whenever the capture buffer is open to receive incoming text. **LC:** Indicates the number of Lines Captured in the current buffer.

Buffer size and this display are determined by the settings in **Buffer configure** and **Buffer mode** when configuring *Dialogue*. (See: **Buffer Configure**. Page 16)

**A:C:D:N:O:S** - Flags

Each letter will flash, or be highlighted when its related condition is true (selected), and will appear as normal when the related condition is false (not selected).

**A** The 'A' begins flashing whenever an auto-exec script, or auto-logon script is being executed.

**C** Carrier detect. When *Dialogue* detects a carrier, the 'C' will appear highlighted. When the carrier is lost, it will return to normal.

---

**D** Disk buffer. If the buffer to disk option is active, 'D' will appear highlighted. When cancelled, the 'D' will return to normal. More information on the buffer to disk option is available on page 33.

- N** Non-visible characters. When *Dialogue* has been instructed to print those characters normally not visible, the 'N' will be flashing. During normal operation of terminal mode, the 'N' appears normal.
- O** Online. This flag is only significant for 'dumb' modems. For such modems (such as 1650/1660 compatibles), the 'O' will be highlighted whenever the modem is put online (off hook). It will appear normal when the modem is offline (on hook). When used with 'smart' modems (those that support the AT command set), the 'O' will always appear highlighted.
- S** Secondary characters. When calling Commodore colour boards, you may wish to use the graphics characters that require you to press the CBM key. Normally, *Dialogue* uses the CBM key to access the *Dialogue* commands. While in this special mode, the 'S' on the status line will flash, and the regular *Dialogue* CBM key commands will be disabled. All commands remain accessible from the drop-down menus. Further explanation of this mode is available on page 38. (See: **Toggle Secondary Commodore Characters. Page 38**)

#### "Terminal mode"

*Dialogue* prints various system messages, or error announcements in this area of the status line. The text will generally flash for a few seconds before being replaced with the words "Terminal mode".

#### The Keyboard

Since standard ASCII contains several characters that aren't normally supported by the C128 keyboard, *Dialogue* has to cheat, and 'trade' some generally unused keys to provide these additional characters.

To Produce:	ASCII Name	<i>Dialogue</i> Substitute Key(s):
_	Underline	Left Arrow
^	Caret	Up Arrow
\	Backslash	Pound Sterling
{	Open curly brace	Shift +
}	Close curly brace	Shift -
'	Single open quote	Shift Pound Sterling
	Vertical Bar	Shift @ <i>~ (single) ~</i>
~	Tilde	Shift *

When using CBM terminal mode, the traditional Commodore keyboard and character set is restored.

---

## Special Keys

There are several "non-printing" keys on the 128 keyboard that have been given unique functions within *Dialogue*.

**ESC** This key has several uses depending on which mode of the software is active. When ESC is pressed while in terminal mode, it will transmit an ESC over the modem. An ESC is a chr\$(27). In other areas, it performs various cursor functions. (See below)

**TAB** The TAB key has two functions, depending on whether you are active in terminal mode, or within the text editor. Pressing TAB while in terminal mode, transmits a CTRL-I. Some remote hosts may accept this as a valid TAB function. Others will ignore it. TAB is also used within the editor to advance to the next tab setting.

**CAPS LOCK** This key makes all letters appear as capitals without effecting the other keys. Its function is unchanged from normal 128 operation with the exception that *Dialogue's* CAPS LOCK function works with all the alphabet keys, including the 'Q', which originally did not shift, due to an internal 128 ROM bug.

**ALT** The ALT key is always used in conjunction with another key to enable various *Dialogue* functions (described below).

**HELP** Pressing HELP invokes *Dialogue's* drop-down menus. Most of the terminal's commands are listed. A highlight bar can be moved by using cursor keys.

**LINE FEED** Pressing LINE FEED from terminal mode transmits a chr\$(10) to the host.

**NO SCROLL** Effect depends on how Flow Control is set in **Configure**. Off - no effect. XON/XOFF - alternates CTRL-S and CTRL-Q. CTS/RTS - flips state of RTS line (for MNP modems).

**CBM logo key** This is the main command key of *Dialogue*. It is used in conjunction with other keys to activate many of *Dialogue's* commands.

**F1, F3, F5, F7** Function keys are user-defined. In VT modes they perform VT PF functions. In this case, you must press SHIFT along with the F key to access your definitions.

## Control characters (Control Codes)

Control characters are non-printable characters of ASCII with values less than 32. They are generally used to control some of the actions of a remote system. See Appendix C for a complete list of available Control codes and their ASCII values. To generate a control character in terminal mode press and hold the CONTROL key, and then press the desired code key. Consult the online help of each particular remote system for the specific control codes supported.

## Local Escape Codes

In many areas within *Dialogue*, certain escape codes are available. These codes are similar to the standard escape sequences mentioned in the C128 manual.

**ESC-P** deletes all the text on the current line up to the cursor position.  
**ESC-Q** deletes all text from the cursor position to the end of the screen line.  
**ESC-@** deletes all text from the cursor position to the end of the screen.  
**ESC-I** inserts a blank line.  
**ESC-D** deletes the current line.

---

## Commands and Features

This section will discuss the menu systems, and outline some of *Dialogue's* features and functions.

### The *Dialogue* Menu System

There are two ways you can activate the various functions of *Dialogue*. Commands can be accessed from the keyboard by pressing one or more command keys. Optionally, almost every function is available on drop-down menus that can be accessed directly from the keyboard, or with the use of a mouse, joystick, or trackball.

The menu system also doubles as a help menu. On the right side of each drop-down menu option, keyboard equivalents for each of the commands are shown. These are the fastest means of invoking a *Dialogue* function because they bypass the menu system altogether. Every function can be accessed by pressing one or two command keys. For the sake of simplicity, the *Dialogue* manual lists the commands by their keyboard equivalents.

The table below shows the conventions used to identify the command keys on the drop-down menus:

<u>Command Key</u>	<u>Display</u>	<u>Command Example</u>
CBM key	Ⓒ	Ⓒb - Toggle capture buffer on/off
ALT key	Ⓐ	Ⓐl - Load <i>Dialogue</i> extension module
SHIFT	↑	↑a - Cycle dial, and auto-exec in auto-dial window
CTRL	^	^d - Delete range in buffer editor
ESC	<	<i - Insert line in buffer editor
SPACE	-	- - Dial number in auto-dial window
Cursor key	CRSR	Ⓒ CRSR- Release cursor mode
CLR/HOME	CLR	Ⓒ CLR - Clear terminal mode screen
RUN/STOP	R/S	Ⓐ R/S - Send break signal to host

*Note: The drop-down menu system is actually two separate but visually identical systems. One appears when you use the keyboard, and one when you use the menu pointer. When the keyboard drop-down is active, the menu pointer has no effect. Similarly, when the menu pointer drop-down is active, the keyboard control is not available.*

### Keyboard Controlled Drop-Down Menus

To use the drop-down menu system from the keyboard, press the HELP key. The status line will be replaced by a menu banner (a set of categories) and the menu associated with the left-most category will drop down. The top menu choice on the drop-down portion will be reversed by a highlight bar (a line of reversed characters highlighting one of the menu options). This highlight bar can be moved up or down using the cursor keys. Moving up or down highlights new options, and left or right will switch *Dialogue* to the next or previous drop-down, wrapping around at either end. When the highlight bar is over the option you want to activate, press Return or Space to select. The drop-down menus will then disappear, and the function will be invoked as if you had used a keyboard command.

You can abort the drop-downs at any time without invoking a function by pressing the HELP key a second time, or hitting the RUN/STOP key.

*Note: In many areas of the manual, only the Return keypress option will be listed. In most cases, pressing Space works equally well.*

---

## Dialogue Menu Pointer Control

*Dialogue* can also access drop-down menus using a sprite-like pointer which is manipulated around the screen using a mouse, joystick, or trackball. When creating the work disk, you were given the option to configure *Dialogue* to use either a GEOS or Amiga style of pointer control. These two systems treat the drop-down menus quite differently.

**GEOS style pointer control** (1351 mouse, joystick, or trackball): The GEOS style of pointer control uses only one button to control everything. In the case of the 1350 or 1351 mouse, you use only the left button.

When the pointer is brought near the status line, it will change to the menu banner. To drop down a particular menu, point to the category heading and press the button once. The appropriate menu will then drop down, and remain displayed until the pointer moves out of the bounds of that menu, or until another item is selected. To select an option, move the pointer over it and press the button once.

**Amiga style pointer control** (1351 mouse only): This style uses both buttons of the 1351 mouse. When the **right** button is pressed and *held*, the banner will appear on the status line. Moving the pointer over a particular category will cause the associated menu to drop down. As you move the pointer over the various options, they will appear highlighted. If you move the pointer out of the bounds of the menu, the highlight bar disappears, but the drop-down menus stay visible on-screen until you let go of the right button.

To select a particular option, release the right button while that option is highlighted. If no menu option is highlighted when you let go, the menus vanish, and control is returned to the keyboard with no changes.

*Note: Since both these pointer systems use only slightly different methods to perform the same functions, a standard has been chosen to describe the options throughout the remainder of the manual. All mouse/joystick/trackball movements are referred to as 'the pointer', and the appropriate mouse or fire button as 'the button'.*

## The Highlight Bar Within Windows

For *Dialogue* functions other than the drop-down menus, like the Configuration Window, or Auto-dial Window, the highlight bar is generally active. It can be moved freely with the cursor keys, including the provision for the HOME key, which moves the highlight bar to the top left position of the window.

Using the pointer, you must position the arrow over the window option you want activated, and press the button once to move the highlight bar, and again to activate the function. In the cases where a list of options is to be toggled through, each subsequent press of the button will display the next selection. Values are cycled one direction only, generally increasing. For some options, it may make more sense to use the keyboard, where options can be toggled in either direction by using the SHIFT key.

When pointer mode is active, windows may have a 'close gadget' displayed in the top left corner of the window. It appears as a little box with a dot inside. Pressing the left mouse button or fire button while the tip of the pointer is aimed at the close gadget will cause *Dialogue* to exit the window. From the keyboard, most windows can be exited by pressing the RUN/STOP key.

---

## Prompts and Requester Boxes

*Dialogue* prompts can be broken down into three categories: graphic requester boxes, text/data input prompts, and filename input prompts.

*Dialogue* handles each prompt differently, and offers different options for each.

**Graphic Requester Boxes:** At certain times, a requester box will appear, offering one to three selection choices inside smaller graphic boxes. To choose, you may use the pointer (controlled by mouse or joystick), and click inside the box of your choice, or press the first letter of the option you desire on the keyboard. As an example, a simple Yes or No question like **Reset buffer?** would have a Yes box and a No box. To select Yes as your answer, you could either move the pointer to the Yes box and press the button, or simply type **y** on the keyboard.

All requester boxes will automatically choose the *second* option if you just press Return. In most cases, this is the safer of the two options (eg: No, don't reset my buffer.)

**Text input prompts:** As a part of many *Dialogue* commands, you will sometimes be requested to enter some text or numeric data. The responses and length limitations will vary from prompt to prompt, but four basic rules apply:

- Entering only a Return will leave the default/current data unchanged.
- Pressing CLR (SHIFT-HOME), will clear the current data, and leave the input blank.
- You will have full use of the cursor right/left keys, as well as insert and delete.
- Entering new data will erase the original, and replace it with your new information.

**Filename input prompts:** From any of the areas within *Dialogue*, where you are prompted to enter a filename, several powerful options are made available in addition to the four rules mentioned above:

• **Directing a file to a particular drive and device:**

In the Commodore world, there are three methods of file access: Single drives (like the 1571) use unique device numbers. Dual drives (like the 4040) and the Xetec hard drive have two or more 'drive numbers', but are addressable using one constant device number. The third method employs named logical subdivisions; i.e. partitions and sub-directories. The 1581 has partitions and the CMD hard drive has both partitions and sub-directories.

*Dialogue* displays the currently selected device number in parentheses within the filename prompt. Whenever you enter a filename, it is possible to direct the file to access any other device or drive number by including the required device and drive number as part of the filename.

**Examples:** To direct a file to a different device number, append a comma and the desired device number following the filename, as if loading a program into the 64 mode of the C128.

`testfile,9` directs the file to device 9  
`testfile,11` directs the file to device 11



---

To select a different drive number for use with a dual drive or hard drive, precede the filename with the appropriate drive number.

**1:testload** directs file to drive 1, from currently selected device  
**1:testload,9** directs file to drive 1, on device 9

Hitting Return on an empty prompt means no filename has been input. The function aborts.

If the file or device is not found, an error message will appear in a requester. You can retry, change device numbers, or abort. Choosing **Retry** will cause the computer to 'look again', in case you've changed the disk, or corrected the problem. If you choose to **Abort**, you'll be returned to the filename prompt, and allowed to enter another filename.

**Device Selection Mode:** You can also change the default device by entering a #. This will enable Device Selection Mode. All currently powered-up devices will be detected and displayed. Use the highlight bar to select the new default device. Or simply type #n (where n is the number of the desired device).

*Note: Once a device number has been specified, it will remain the default until a new device is specified. The active device is always shown within parentheses at every filename prompt, or within the DOS wedge prompt.*

#### • Viewing the disk directory

Entering \$ at the prompt, will display a three-column directory of the files on the disk. Depending on the command, or the device, only a specific type of file may be displayed.

**,\$9** for directory of device 9  
**\$1** for directory of drive 1

If the directory is longer than one screen, the word **<More?>** will appear on the status line between each page. Press Return (or Space) to continue, or RUN/STOP to abort the display.

#### • Checking the disk

Entering \$\$ can be used to display just the header of a disk, and blocks available. This allows you to quickly check to see if you have the right disk in the drive, or if it has sufficient free space.

#### • Access to DOS commands

If you enter the wedge symbol (>) from the filename prompt, you are dropped to the DOS wedge where any of the Commodore DOS 2.0 commands may be sent directly to the disk.

Refer to the *Dialogue* section on the DOS commands (page 40) for more information, or for complete details, consult your C128 and drive manuals about standard CBM DOS commands.

#### Special functions for 1581 users only

Since the Commodore 1581 disk drive has a more detailed file structure, including the use of partitions, additional features have been added to the filename prompts of *Dialogue* to support these. All other functions operate as described.

---

Users can select the name of a 1581 partition at any filename prompt by typing a / preceding the name of the chosen partition. Entering a / by itself will select the root directory. In either case, the 1581 will switch to the new partition, and the filename prompt will re-appear and ask for a filename to be entered. *You cannot include paths within the filename.* Files may only be read or loaded from the current directory.

examples:     /text   selects the partition text.  
               /       selects the root directory.

Optionally, entering // will display all the available 1581 partitions within the current directory. **File Selection Mode** (see below) is active, so use the cursor keys to move the highlight bar to the desired partition name and press Return. Mouse users can double-click the partition name.

Once a new partition is selected, entering \$ will re-display the list of available files, just as it would with any other drive. You may make use of **File Selection Mode** again to load or read any file.


### ***Dialogue File Selection Mode***

Whenever a directory is displayed as part of a load or read operation, a special **File Selection Mode** is invoked. A highlight bar automatically appears over the top left file in the directory. Using the cursor keys, move the highlight bar to the desired file and press Return. When using the mouse pointer, double-click the filename. The first click moves the highlight bar, and the second loads the file.

Pressing RUN/STOP will abort **File Selection Mode**, and return you to the standard filename prompt, where you can either press Return to abort, or manually type in a filename.

---

## CBM Key Commands

The bulk of *Dialogue* functions can be invoked by using the Commodore key. This is the key at the bottom left of the keyboard that looks like . Since this key is difficult to print, the manual refers to it as the CBM key. All CBM key functions are also available from the drop-down menus, and are accessible using the pointer. The CBM key appears as an underlined 'c' (c) on the drop-down menus.

To invoke a CBM key function from the keyboard while in terminal mode press and hold the CBM key, then press one of the appropriate keys as listed below:

Keypress:	Description of Function:	Page #
CBM-A	Enter the Auto-Dial Directory area	19
CBM-B	Toggle capture Buffer open/closed	30
CBM-C	Enter the <i>Dialogue</i> Configuration window	14
CBM-D	Download a file to disk using an error-detection protocol	25
CBM-E	Enter the buffer Editor	34
CBM-F	Display system Function key definitions	37
CBM-G	Grab current screen into buffer	31
CBM-H	Hook control. (Function varies with modem type)	37
CBM-I	Info. Displays statistics of last upload/download	27
CBM-L	Load disk file into capture buffer	33
CBM-M	Enter Macro definition window	35
CBM-N	Select active buffer Number	30
CBM-P	Dumps capture buffer to Printer	32
CBM-Q	Quit. Prompts <b>Quit Dialogue?</b>	39
CBM-R	Reset (clear) capture buffer (Prompts <b>Reset buffer?</b> )	31
CBM-S	Save capture buffer to disk	33
CBM-T	Transmit capture buffer over modem	32
CBM-U	Upload a disk file using an error-detecting protocol	25
CBM-V	View capture buffer	31
CBM-W	Waiting for call - 'host' mode	37
CBM->	DOS wedge. Disk and printer commands	40
CBM= CBM- <u>l</u>	Toggle split-screen conference mode. (Not in VT modes)	38
CBM- <u>l</u>	View scrollbar buffer	30
CBM-*	Set system clock	37
CBM-CRSR	Activate 'Release cursor mode'	39
CBM-RUN/STOP	Abort executing auto-exec script	48
CBM-CLR	Clear terminal mode screen	39
CBM-1-8 (top row)	Set 'current' buffer number (varies by setup)	30
CBM-SHIFT	Switch character set	13

---

## ALT Commands

*Dialogue* also uses the ALT key to access other functions. As with the CBM functions, ALT functions are also available from the drop-down menus. The ALT key is shown as a small underlined 'a' (a). To invoke an ALT key function from the keyboard while in terminal mode, press and hold the ALT key, and press one of the following keys:

Keypress:	Description of Function	Page #
ALT-A	Invoke an Auto-exec script file	42
ALT-B	Toggle 'Buffer to disk' option	33
ALT-I	Transmit modem Initialize string	40
ALT-L	Load a <i>Dialogue</i> extension file	40
ALT-N	Toggle Non-visible character mode	37
ALT-P	Dump current terminal mode screen to Printer	39
ALT-R	Reset capture buffer line transmit pointer	32
ALT-S	Toggle Secondary Commodore character mode	38
ALT-T	Transmit a single line from capture buffer	32
ALT-V	View a disk file	31
ALT 0-9	Invoke macro (numeric keypad only)	36
ALT RUN/STOP	Transmit a system BREAK signal	38

Almost all of the CBM and ALT key commands are also available from the drop-down menus. All of these commands are described in full detail on the pages shown.

### Incoming Control Codes and Terminal Emulations

*Dialogue* has several terminal emulation modes available, and is capable of handling several forms of control sequences sent by an online system.

**CBM mode** accepts Commodore codes, including those required to access many colour boards. CBM-SHIFT switches between uppercase/lowercase and uppercase/graphics.

**ASCII mode** the standard BBS mode. (accepts all standard *CompuServe* control codes.)

**IBM mode** supports most of the IBM ANSI standard including those codes required to view colour and graphics on IBM (compatible) bulletin board systems.

**VT100 mode** accepts a subset of the Digital Equipment Corporation VT100 terminal control sequences, including VT100 graphics. Not all VT100 modes are supported fully, but enough to work on most systems supporting VT100 with little compromise. Since line noise can put the terminal into/out of graphic mode, CBM-SHIFT switches modes. **VT52 mode** accepts standard VT52 control sequences. The control codes associated with these emulations are supported.

*Dialogue* will respond to the following incoming control codes:

**CTRL-G** In all modes, a received CTRL-G (chr\$(7)) causes *Dialogue* to ring a bell, and/or flash the screen once. This can be disabled from the **Configuration** menu. (See: **Bell Mode**, page 16)

**CTRL-L** This will cause *Dialogue* to clear the terminal screen, and place the cursor at the top left position. (CBM, IBM and ASCII modes only)

**CTRL-S/CTRL-Q** *Dialogue* supports XON/XOFF flow control when transmitting a buffer.

---

## Configuring *Dialogue*

### The Configuration Window      CBM

You modify many of *Dialogue*'s parameters within the Configuration window. Options include screen attributes, modem and RS-232 settings and establishing system default file-names. To open the window, press CBM-C or select **Configure** from the menu. The Configuration window will appear, showing the current settings and with the highlight bar active.

Pressing Return while an option is highlighted will either prompt for a filename, or increment through a list of various choices associated with that specific option. Pressing **Shift-Return** decrements through the list. To exit, press the RUN/STOP key.

When using the pointer, aim at a function and press the button once to move the highlight bar. Press the button again to activate the option, or cycle through the choices. To close the window, point to the small close gadget in the top left corner of the window, and press the button.

*Note: Several settings will not appear to have any effect until you return to terminal mode. For instance, pressing Return when over the 40/80 function, will toggle between 40- and 80-column display mode, but the display will not change until you exit.*

### Descriptions of Configuration Window Settings

**Screen columns:** Sets 40- or 80-column mode. 40-column mode is displayed using wider characters, but still uses the FAST mode and the C128's 80-column screen.

**Screen lines:** Sets 25- or 50-line mode. On some monitors, the 50-line mode may not be as clear as on other screens. Within the Configuration window, the 25-line screen and the 50-line screens have separate colour tables. Experiment with different colour variations to improve the legibility on your particular monitor.

**Newline mode:** Chooses which characters are required to place the cursor at the first position of the next line. For most systems CR/LF will be used.

**Wordwrap mode:** When set **On**, words that won't fit at the right side of the screen will drop to the next line. This is useful when the host system is sending lines longer than your screen setting, or when chatting to another caller. When **Off**, word wrap is handled by the host.

**Flow control:** This allows *Dialogue* to start or stop text flow as needed, like when you exit terminal mode for menu functions. Under normal conditions, the flow control used is standard XON/XOFF (CTRL-S/CTRL-Q). When using modems that support MNP protocol (up to level 5), the RTS/CTS flow control of the modem is used when connected to MNP systems.

**Delete character:** The ASCII code transmitted when Delete is pressed can be either chr\$(8) or chr\$(127). Shift-Delete will transmit the other.

**Delete mode:** This changes the way a delete will be *displayed*; not how it's transmitted. Selecting **Delete** will display a proper delete, where the character to the left of the cursor will be erased. Choosing **Backspace** sets a non-destructive mode, where the character to the left of the cursor is still deleted, but remains visible on the screen.

**Screen output:** This allows you to force or suppress the displaying of data on the screen while uploading or downloading with a protocol, and/or during buffer loads, saves and prints. Choose one of the four options (**None**, **Buffer**, **Protocol**, **Both**) by cycling through the list with the Return key, or the button.

---

**Menu pointer:** This option toggles the pointer on/off and turns on or off the close gadgets that appear on system windows. If you're not using a pointing device, leave it off.

**Pointer timeout:** Sets the number of idle seconds the pointer will wait before blanking out. Any pointer movement will cause the pointer to reappear where it was last displayed.

**Baud rate:** Cycles through 300/600/1200/2400/3600/4800 bits per second (plus 7200/9600/14,400/19,200/38,400 in UART versions). Optional baud rates are manually selectable by pressing **B**, and entering a new baud rate, or by using the menu option. 4800 baud is especially useful for level 5 MNP protocol on supporting modems.

**Fine tune:** (Not applicable for UART users) Some modems may require you to 'fine tune' the send and receive baud rates a tiny bit. If you experience continual garbage on the line, adjusting these may help. For normal operation, they should both be left at 0.

**Length/parity:** Parity is a control over the status of the eighth bit of a transmitted byte. The modes available within *Dialogue* are:

- |         |  |
|---------|--|
| 8/None  | - 8 data bits. No parity needed or possible. (Use with CBM and IBM boards) |
| 7/Odd   | - 7 data bits. 8th bit is set to make number of high bits an odd number.   |
| 7/Even  | - 7 data bits. 8th bit is set to make number of high bits an even number.  |
| 7/Space | - 7 data bits. 8th bit is always set low.                                  |
| 7/Mark  | - 7 data bits. 8th bit is always set high.                                 |

**Stop bits:** Sets for one or two bits to be sent after the transmitted byte.

**Local echo:** Local echo refers to whether the characters you type will be displayed on the screen as you type or if they need to be echoed back from the remote system.  
Full duplex = Echo Off; Half Duplex = Echo On.

**Terminal mode:** The terminal emulation mode for *Dialogue* to use. Several are available:

- |       |   |
|-------|---|
| CBM   | - PETSCII mode. Used to call Commodore colour boards.           |
| ASCII | - Normal terminal mode for use with most BBS systems.           |
| IBM   | - Special subset of IBM ANSI supporting PC graphics and colour. |
| VT100 | - For systems that support VT100, like VAX mainframes.          |
| VT52  | - For systems that support the VT52 control sequences.          |

**Transmit mode:** When transmitting text (function keys, macros, or buffer uploads) this mode helps determine how text is sent to the host. Since some systems can't handle continual streamed text without getting garbled, *Dialogue* has three options:

**Stream:** Characters are transmitted in a continuous stream separated by a time delay controlled by the **Transmit Delay** settings described below.

**Stream2:** Characters are streamed continually, and the **Transmit Delay** is inserted only at the end of each line.

**Reflect:** In this mode, *Dialogue* waits for each character to be echoed back before sending the next character. This mode is slower, but gives visual confirmation that the text you are transmitting is being received as it should be.

**Transmit delays :** This option, used in conjunction with either the **Stream** or **Stream2** modes, dictates the amount of delay between each transmitted character or line. Values are in 100ths of a second. The **Transmit delay** setting has no effect on text transmitted using **Reflect** mode.

---

**Cursor type:** Selects either block, or underline cursor.

**Cursor Flash:** Sets whether the cursor will flash, or remain constant.

**Printer device #:** Defines the device number of the printer. Values 4-7

**Secondary address:** The secondary address to use with the printer. Values 0-31

**Bell mode:** *Dialogue* has four different ways handling system bells.

**Sound:** Make a bell sound.

**Visible:** Briefly flash the screen.

**Both:** Generate both sound and visible bells.

**None:** Disable all bells.

**Buffer mode:** Chooses the method *Dialogue* uses to store characters in the buffer. **Line** mode stores full lines of displayed characters, and allows the use of the buffer editor.

**Character** mode displays only characters received, and permits more data to be stored in the buffer, but the editor isn't available.

**Buffer configure:** Here you set the configuration of the buffer memory, allowing either 1,2,4 or 8 completely individual buffer areas.

The following table shows you how the status line will look with an empty buffer, and buffer 1 selected, for the various buffer configurations.

Mode:	Buffer config:	Display:	Buffers:
Lines	1	LR: 800 1 LC: 000	1 x 800
Lines	2	LR: 400 1 LC: 000	2 x 400
Lines	4	LR: 200 1 LC: 000	4 x 200
Lines	8	LR: 100 1 LC: 000	8 x 100
Characters	1	R: 64000 1 C: 00000	1 x 64000
Characters	2	R: 32000 1 C: 00000	2 x 32000
Characters	4	R: 16000 1 C: 00000	4 x 16000
Characters	8	R: 08000 1 C: 00000	8 x 8000

When using the optional 1700 or 1750 RAM Expansion Units, several additional options exist, expanding *Dialogue's* advanced buffer capabilities even more. These larger buffer options are only available when the RAMDOS option is *not* active.

With the 1700			
Lines	RAM1	LR: 800 1 LC: 000	2 x 800
Lines	RAM2	LR: 1600 1 LC: 0000	1 x 1600
Characters	RAM1	R: 64000 1 C: 00000	2 x 64000
Characters	RAM2	R: 128000 1 C: 000000	1 x 128000
With the 1750			
Lines	RAM1	LR: 800 1 LC: 000	8 x 800
Lines	RAM2	LR: 6400 1 LC: 0000	1 x 6400
Characters	RAM1	R: 64000 1 C: 00000	8 x 64000
Characters	RAM2	R: 512000 1 C: 000000	1 x 512000

**Note:** Large RAM2 expansion buffers cannot be edited in the built-in editor.

---

**Set defaults:** This function pops up a sub-window with several options listed. Here you can enter default filenames (up to 12 characters each) for various system files. These files will be loaded automatically every time you run *Dialogue*. The modem file was entered when you created the *Dialogue* work disk. You'll learn more about the other files in the different sections of the manual. Setting these defaults is completely optional. If no filenames are specified, the word *None* will appear, and no files will be loaded.

All system filenames are tagged with a three letter extension when loading or saving, making them easier to identify in the directory. Do not include the extension when entering the filename, as *Dialogue* does this automatically.

Modem file	- The modem control file, for your modem.	.MOD
Number file	- The default phone directory file.	.NUM
Macro file	- The default macro key settings.	.MAC
Chr. set (25)	- Optional custom 25-line character set.	.CHR
Chr. set (50)	- Optional custom 50-line character set.	.CHR
Editor setup	- The default editor tabs, margins, and mode.	.EDT
Extension	- Optional extension module.	.EXT
Translation	- Optional custom translation table.	.TRN

**Colour Settings:** A secondary menu appears with various colour options.

Screen Colour - The main background screen colour.  
Text Colour - The colour of incoming text.  
Cursor Colour - The cursor, and status highlight colour.  
Status Line - The colour of the status line.  
Highlight Colour - The Highlight (Bold) text colour.  
System Text - *Dialogue* Text (windows and drop-downs).

*Note: All colour settings are duplicated for 25- or 50-line modes, and are set and stored separately. To set the 50-line mode colours, you must first activate 50-line mode.*

### Save System Configuration **S**

Pressing **S** will save the current settings to disk. Remember to do this to save your changes - before resetting the computer! Before saving, *Dialogue* will prompt for a device number. The last accessed device will be highlighted. To save to a new device, highlight the new device number and press Return (or double-click your selection). Otherwise, pressing Return will save the *configure.sys* file to the current device. In most cases the file will already exist, and you'll be asked to confirm that you wish to overwrite it with your new selections.

### Loading Default System Files **L**

Pressing **L** within the Configuration window, or selecting **Load Defaults** from the menus will cause the default system files to be reloaded from disk. A requester will ask **All Default Files?** Answering Yes will reinstall all the original values and the selected default files - just as if the program were newly loaded. If you answer No, only the list of defaults will be loaded. None of the actual files will be reloaded.

### Extra Functions Available from the Configuration Window

Fast 'hot-key' presses are provided for these functions and they are all available from the menus. Note that these options are all available to make temporary alterations. None of the changes will be saved with the configuration settings.



---

## Variable Baud Rates **B**

On some Commodore-based systems, it is possible to achieve baud rates in between 110 and 600 baud. This trick works with most dumb modems, and a few smart ones. Many people have claimed baud rates of up to 600, but the technique becomes increasingly unreliable with higher baud rates.

If the host you are connected to supports variable baud rates, it is important to first change the host system to the desired rate. Then, enter the configuration window and press **B**. *Dialogue* will prompt for a new baud rate between 50 and 9999. Consult the online system documentation to find out more about support for variable baud rates.

*Note: Modems cannot change speeds from 1200 or 2400 baud while online! The B command will still work, but you'll be unable to communicate with your modem.*

## Optional Character Set **C**

You are able change the internal character set used by *Dialogue*, to another style you have saved on disk. When you press **C**, or select **Load Character Set** using the pointer, you'll be prompted to enter the name of a new character set to install. There is no need to include the **.chr** extension when specifying a filename. Some optional character sets are provided with *Dialogue*, along with a complete character editor extension module. It allows you to create, or customize your own character sets (see Appendix B, **CHREDIT**). If you have a character set you like, you can preset *Dialogue* to load it automatically when the program is loaded (see **Set Defaults**, page 17).

*Note: Dialogue uses different character sets for 25-line mode, and 50-line mode. Each set must be loaded separately, by selecting the load option while the desired mode is active.*

## Loading a Modem File **F**

If for any reason, you wish to load an alternate modem file into *Dialogue*, pressing **F** will prompt for a new filename. Do not include the **.mod** extension when specifying a filename. See Appendix A for a complete list of modem files available.

## Modem Control **M**

(For 'smart' modem users only) Pressing **M** while in the Configuration window will cause *Dialogue* to prompt for a modem initialization string followed by a dial string. (**AT** is automatically sent before the string and should not be entered.) The default dial string is **DT**. The **T** should be replaced by **P** for those whose phone exchange doesn't support Touch-Tone.

## Loading a Translation Table **T**

When *Dialogue* is using a terminal emulation type other than **CBM**, *Dialogue* must translate the characters being received between **PETSCII** (the codes that the **C128** uses internally) to true **ASCII** (a standard set used on most other computers). You can replace *Dialogue's* translation table by loading one of your own. This allows you to customize *Dialogue* to adapt to just about any environment (eg: Atari **ATASCII** boards). The translation table is 512 bytes long. The first 256 are **PETSCII** to **ASCII**, and the second 256 are **ASCII** to **PETSCII**.

A complete translation table editor has been included as a loadable extension file with this version of *Dialogue*. See Appendix B for more information.

---

## Auto-dialing *Dialogue*

### The Auto-dial Directory Window      CBM **A**

Auto-dialing is a software-controlled dialing process, instead of you dialing the phone number manually. *Dialogue* uses 'modem support files', which are individually programmed to use the features of your particular modem. Since you'll probably be greeted with a busy signal some of the time, you'll quickly welcome *Dialogue's* powerful auto-dial capabilities. *Dialogue* features a 30-number phone directory from which you can select single or multiple numbers to auto-dial. Any number of these directories can be stored on disk. Press **CBM-A**, or select **Auto-dial** from the menus to display the auto-dial directory window.

### Entering a New Phone Number      CBM **E**

The window starts off empty, but has room for 30 numbers. To enter a new phone number for the directory, use the cursor keys to select a position, and then press the **E** key (which stands for **Edit** current position). Using the pointer, press once over the entry you wish to edit, which will highlight it, and then select **Edit entry** from the menus.

On the status line, *Dialogue* will prompt you to enter **System info**: You can enter up to 18 characters to describe the system. The name, and the hours of operation, are two choices. You have full use of the left/right cursor keys, as well as insert/delete to edit the data before pressing **Return**.

Next comes the **System number**: prompt. This is where the actual phone number of the on-line system is entered. The phone number data can be up to 16 characters in length. The prompt accepts all characters but be careful to limit them to those that are of use to your modem. For most modems, the following will apply:

- is ignored (generally used as a group separator, such as 555-5555).
- , will generate a two-second delay.

*Note for smart modem users: When the phone number begins with a non-numeric character, Dialogue does not use the dialing string set in the Configuration window. Rather, AT is sent, followed by the phone number string. This allows special modem modes to be selected. Of course, the number string must then contain the dial command before the digits (DT or DP).*

After pressing **Return** to confirm the entry, a requester asks if you wish to **Define function keys**?. If you choose **Yes**, *Dialogue* will prompt you for four inputs of text data, each linked to a function key (above the numeric keypad). When connected to a system in terminal mode, pressing one of these keys will cause *Dialogue* to transmit the corresponding text (maximum 20 characters each).

Each entry in the phone directory can have these four keys individually programmed. Use them for specific information like logon information, name, password, special online commands, message signatures, etc.

*Note: In VT100, or VT52 mode, these keys are re-defined to perform specific VT PF functions. To use your definitions, you must press the Shift key along with the appropriate function key.*

The next display you see will be a stripped-down version of the **Configuration** window. Here, you will have the opportunity to customize the major parameters of the individual system.

---

With other terminals you may have had to remember all the proper settings for any new system and set up the terminal accordingly before dialing each number. *Dialogue* remembers the settings for every number in the phone directory for you! Each can each be totally different, and *Dialogue* will make sure that the proper configuration is installed when you connect. This is made possible by use of a special subset of the **Configuration** window. Most of the regular system defaults from the main configuration window are shown here, along with some new items specific to auto-dialing a host system. The proper settings for these options require some knowledge of the system you want to call. Generally, the default settings will work. You are free to change or edit them at any time.

Screen width and baud rate are two regular options you may wish to set differently for each system. Not all systems you dial will conform to your preset defaults. If you're not sure of the baud rate, or screen size, it's always safe to try 300 baud and 80 columns first, and change them later.

Other parameters you may wish to change for each individual system are:

**Screen lines:** 25- or 50-vertical line screen display.

**Wordwrap:** If the host's output is wider than your active screen width.

**Flow control:** Options are **None**, **XON/XOFF** or **RTS/CTS**.

**Delete chr:** Choose **CHR5(8)** or **CHR5(127)** as the delete character.

**Local echo:** This determines whether characters you type are echoed back.

Half duplex = Local echo On. Full duplex = Local echo Off.

**Terminal mode:** Choose either **ASCII**, **IBM**, **VT100**, **VT52** or **CBM** emulation mode.

**Transmit Mode:** Some systems are not quick enough to accept **Stream**. Select **Reflect**.

**Transmit delays:** Timing for **Stream** modes. The larger the number, the slower text is sent.

The above are described in more detail on pages 14-15. (See: **The Configuration Window**)

The following special options pertain to the individual systems, and do not appear in the regular **Configuration** window.

**Carrier check:** How many seconds after dialing, to wait for a carrier before aborting.

**Save connect time:** Turn **On** to save elapsed time. (See: **Time Log Review**, page 22)

**Protocol:** Select the upload/download protocol supported by the system. Upon connect, this protocol will be installed as the default. You can change it any time via the **Protocol Parameters** window (page 25).

All of the above options are changed using the highlight bar using either **Return**, **Space**, or the button to toggle through the various choices. When you're content with the selections, press **RUN/STOP**, or click in the close gadget at the top left, and you'll be returned to the auto-dial window.

## **Saving The Directory Numbers** S

To use the number(s) you've defined the next time you load *Dialogue*, remember to *save* the directory entries. Press **S** to save the number file. *Dialogue* will prompt for a filename. There is no need to add the **.NUM** extension to the filename, as it will be appended automatically. After entering a filename, press **Return** and *Dialogue* will save the phone directory. If the filename exists on the disk, you'll be asked to confirm that you wish to erase the old file, and replace it with the new.

At this point, you may wish to set this number file as the **Default** file, so that it will automatically be loaded in each time *Dialogue* is loaded. This is done from within the **Configuration** menu. (See: **Set Defaults** - page 17)

---

## Loading Number Files L

Naturally, new directories can be loaded from disk at any time. You may store as many individual directories as your disk space permits, each with a different filename; although only one file can be programmed as the default to load in with *Dialogue*.

## Auto-Dialing a Number Return

Now that you have entered one, or several system numbers into the directory, you're ready to start *Dialogue* auto-dialing. Using the cursor keys, move the highlight bar over the number you want to dial, and press Return or Space. *Dialogue* will dial the number. A sub-window appears displaying various dial information, including a countdown timer indicating how long *Dialogue* is going to wait for a carrier tone, before aborting. This is the carrier delay value you specified when you defined the number.

If a carrier is detected, you'll automatically be transferred to **Terminal mode**. The pre-defined configuration for that system will be installed, the screen will clear, and any system info you defined will be displayed at the top of the screen. On the status line, the C (Carrier) will be highlighted. You are now online with the host system. Pressing Return will usually begin your session.

If no connection was established, *Dialogue* returns to the menu after the full countdown time has lapsed. If you wish to stop dialing at any time, pressing the RUN/STOP key, or moving the pointer to the close gadget at the **Dialing** window's top left, and pressing the button will abort.

*Note: Some modems have internal timers, and may 'timeout' internally before Dialogue has completed the countdown. This delay may be adjustable using the AT57 register.*

## Automatic Redial Shift Return

If the system you are calling is busy, you may want to have *Dialogue* keep trying. By holding the Shift when you press either Return or Space, *Dialogue* will keep dialing the number until it either connects, or you abort by pressing the RUN/STOP key. Since many of today's modems have speakers which let you hear when a line is busy, you are able to skip the remaining countdown time by pressing Space, or Return again; before *Dialogue* has completed its countdown. The call will be aborted, and redial immediately.

When you finally connect to a system, the **System Info** is displayed on the screen, and the screen starts flashing and/or ringing the bell. Pressing the CBM key once aborts this.

*Dialogue* is intelligent enough to detect a busy line very quickly with 'smart' modems that support the appropriate result codes. If your modem supports BUSY DETECT, you'll want to make sure that that option of your modem is active. Check your modem manual for ATX code information.

*Note: You can define the ATX code, along with any other modem settings (such as the AT57 registers described above) within the Modem Initialize String so that Dialogue sets your modem up for you each time. See page 40 for more information.*

## Spontaneous Number Dialing N

Occasionally you may want to dial a number once, but don't wish to retain it as one of your programmed numbers. Pressing N will prompt you for a phone number, and dial it directly.

---

Terminal values are set to the default selections as defined in the regular **Configuration** window.

### **Executing an Auto-Logon Script** **A**

*Dialogue* supports an elaborate script language which is described in more detail beginning on page 42. These scripts are stored on disk, and linked directly to the phone number of the desired system. Instead of pressing Return to dial the number normally, press the **A** key to connect using a programmed auto-exec script. In keeping with the above conventions, pressing **Shift-A** will redial continuously till a connection is established. When the connection is made, the auto-exec script will begin executing automatically. You can abort a logon auto-exec script at any time by pressing **CBM-RUN/STOP**.

### **Writing a Logon Auto-Exec Script** **W**

Pressing **W** while a phone number is highlighted will put you into the *Dialogue* buffer editor. This option will use the highest number buffer in your selected buffer configuration. Contents of this buffer will be lost so ensure that the buffer is empty (or erasable) before entering this command. See page 45 for full information on the language commands.

### **Printing the Auto-dial Directory** **P**

If a printer is connected, this will dump the displayed two-column directory to the printer. The printer device and secondary address are controlled within the **Configuration** window.

### **Rearrange Directory Positions** **R**

This function allows you to tidy up the directory, and put entries in any order. When the **R** key is pressed, the current highlighted position will flash. Move the highlight bar to the position you want the flashing entry to be located, and press Return. The two numbers will switch positions.

### **Time Log Review** **T**

*Dialogue* is capable of keeping track of the total accumulated connect time for any system. This feature is very helpful for estimating your online costs when using a pay per time system such as *CompuServe*. Pressing the **T** key while the highlight bar is over any correctly defined number will cause *Dialogue* to load the statistics from disk, and display the total connect time for that system. After the time has been displayed, you'll be asked whether you want to reset the time back to 00:00:00.

The accumulated connect time storage option is user selectable. Only those system numbers you have defined to save connect time will have a stored time log. The option exists within the subset of the special configuration window that is displayed when entering the information for a new phone listing (see page 19). Select **Save connect time On** to allow the accumulated connect time to be stored. The files will be created as needed, and you'll be prompted to supply an appropriate device number.

### **Multiple Number Auto-dialing** **M**

If there are several numbers you'd like to connect to in one session, *Dialogue* can cycle through them, attempting each one in sequence until it connects to one. Pressing the **M** key or selecting **Multiple Dial** from the menus will enter you into the **Multi-Dial** mode.

---

Within this mode, pressing Return while the highlight bar is over a number, cycles through a number of dial options. Each attribute will be displayed on the status line. The first option, **Dial**, flags the directory entry for standard redial. The second, **Exec**, is used to force the number to dial, and connect with an auto-exec script. The last option, **None**, clears any previous attributes. To clear all selections at any time, you can press the CLR key (SHIFT-HOME). When you have pre-selected all the numbers you wish to cycle through, pressing RUN/STOP will complete the process, and return you to the normal auto-dial mode.

### Dialing the Selected List D

To begin dialing through the queue, press the **D** key, or select **Dial List** from the menus. All the numbers you have selected as multi-dial will be displayed in the offset colour. When you connect to a system, its number is automatically deleted from the list, and returned to the normal colour. You can continue dialing the remaining numbers in the queue at any time by pressing the **D** key, or by selecting **Dial List** from the menus within the auto-dial window.

### Killing a Multi-dial Number K

Occasionally, it may be handy to discontinue dialing one of the numbers in your multi-dial list. This could be because the line is no longer a BBS, or the system is not answering. Whatever the case, pressing **K** at any time during the timer countdown will abort the dial, and clear the number from the list so that it won't be dialed again.

### Saving and Loading Multi-Dial Lists Alt S Alt L

To save the multi-dial list to disk, press ALT-S. Pressing ALT-L will load a previously defined dial list. These appear as **Save List** or **Load List** in the menus. In either case, the **.LST** extension is automatically appended to the filename.

*Note: Do not confuse these with the global Load and Save functions of the Auto-dial window. These options refer only to the multi-dial list. You must have a number directory loaded into memory first.*

---

## Protocols and Uploading/Downloading with *Dialogue*

### Terms for Terms

Sometimes telecommunications terms are too loosely applied. Downloading and uploading (two very prevalent terms) are often used to mean different things - occasionally even being used for their opposites! Therefore, we need to provide precise definitions:

**Upload:** The controlled transfer of a file from your computer to another system using an error-detection protocol.

**Download:** The controlled transfer of a file from another computer to yours using an error-detection protocol.

For our purposes, all other transfers of data, from buffers or disk files, are described as "transmitting data" or "dumping a buffer or file". Since such transfers do not involve any error-detection, they are not considered to be uploads or downloads.

### Protocols Defined and Explained

Telephone lines were intended to carry the human voice, not computer data. Modems do a marvelous job working within the limitations of the phone lines, but occasionally data being sent over the phone lines can get garbled, or distorted. During normal BBS typing, the occasional data error may appear as strange looking text or result in a character being dropped. While this may be slightly irritating, we can easily determine that an error has occurred and infer the correct data from its context.

However, when garbage occurs during a transfer of a program file without any protocol, the result is much worse. If even a single error occurs, it can render the entire transferred file useless. This would be frustrating - and expensive if you're paying for your connect time!

A protocol is a defined format that terminals use to send and receive disk files while ensuring that transmission errors are detected and eliminated. Most protocol formats include advanced methods to check that the data coming in is the same as the data that was transmitted.

When an error-detection protocol is used to transfer a file, data is sent in small 'packets'. A packet is simply a group of data being transmitted all at once. After each group is received, a mathematical 'checksum' is compared. If there is a discrepancy, the packet is sent again. If the data is received without error, only then is it saved to disk. This process is repeated until the entire file has been received correctly. Most protocols have a pre-defined packet size which is transmitted between each error checksum.

### Protocols Supported by *Dialogue*

*Dialogue* currently supports six error-detection protocols: **Punter C1**, **Xmodem checksum**, **Xmodem CRC**, **Xmodem 1K** (sometimes erroneously referred to as **Ymodem**), **Ymodem Batch** and **CompuServe Quick B**. These vary in their usage (see **Protocol Descriptions and Usage** below) and in their packet sizes.

You'll probably find that most of your file transfer requirements will be satisfied by one or more of these protocols. Most systems will use at least one of these and some will offer more than one. *CompuServe* supports five of the six. Additional protocols will probably be included in future versions of *Dialogue*.

---

## How to Upload or Download from the Host System

Before actually beginning a file transfer, you'll need to become familiar with the online system. Since every host will probably have slightly different commands and features, it is not possible for this manual to provide specific command instructions on how to upload or download from every type of system. Online help files should be available on each system to guide you through the required steps. In most cases, the **Usage:** notes given in **Protocol Descriptions and Usage** (see below) should be sufficient for you.

If you correctly prepare the online system to transfer a file (either an upload, or a download), it should somehow prompt you to begin the transfer process. Some examples are given in the **Usage:** notes. Once you've gotten to this stage, you're ready to activate *Dialogue's* upload or download functions.

### Uploading Files      CBM U

Press CBM-U or select **Upload** from the menus.

### Downloading Files      CBM D

Press CBM-D or select **Download** from the menus.

*Note: CompuServe B protocol is auto-execute and doesn't require the use of CBM-D or CBM-U.*

## The Protocol Parameters Window

A special window containing several protocol options will appear before commencing an upload or download. *It can be requested from any upload or download filename prompt by entering ?*. Make any changes that may be necessary and then press RUN/STOP (or click on the close gadget) to bring up the up/download filename prompt. Each option has several choices, which can be cycled through by pressing Return, or by clicking.

**Protocol:** [Options: None, Punter C1, Xmodem CRC, Xmodem Chk, Xmodem1K, Ymodem, CIS-QuickB]

A default protocol may be listed if you already selected a protocol from the auto-dial directory. Verify that the protocol you wish to use is selected.

**Translation:** [Options: Off, On] (*C1 protocol ignores this setting*)

If you're downloading text as a sequential file, it's probably in true ASCII format. It will definitely be in true ASCII if an Apple or IBM user uploaded it. (True ASCII is different from PETSII, and ASCII files won't be very usable if you download the file as is. If you do download an ASCII file without translating, all capitals will be lowercase and all lowercase letters will be capitals.) By choosing **Translation: On**, you instruct *Dialogue* to convert the text to PETSII as it is downloaded. This will allow most 64 or 128 word processors to read the file.

**Ymodem Filenames:** [Options: Prompt, Default]

Ymodem has the capability to receive the filenames automatically from the host system as part of the download. Since Ymodem is common on IBM-type systems where filenames are often restricted to eight characters or less, the **Prompt** option allows you to override the incoming default filenames. You'll be prompted to enter a new filename (which can be up to 16 characters in length). This option is an easy way to direct the incoming files to another drive or device. If **Default** is chosen, files are saved using the host's filenames. The function has no purpose when uploading using Ymodem or when using any other protocol.



---

**Filetypes:** [Options: Prompt, PRG, SEQ] (*Automatic with C1 protocol*)

Commodore computers can save files to disk using either PRG (program) or SEQ (sequential) format. When transferring files over the modem, you may have to manually specify which filetype to use. If **Prompt** is chosen, you'll be prompted to give the filetype for *each* file transfer. Before the transfer begins, a requester will appear prompting you for the filetype, SEQ or PRG. The file transfer will then begin.

If you set **Filetypes:** to PRG or SEQ, this requester will not appear and your selected filetype will be used for all transfers until the setting is changed again.

*Note: The SEQ filetype is automatically selected whenever translation is turned on. In the case of C1 transfers, Translation and Filetypes settings are both ignored.*

**Punter block size:** [Options: 40-255]

Punter 'block' size is variable. All other protocols ignore this setting. See **Protocol Descriptions and Usage** below for more information on block sizes.

When you've made the appropriate selections in the **Protocol Parameters** window, press the RUN/STOP key, or click on the close gadget at the top left of the window.

### **The Transfer Statistics Window**

During any *Dialogue* file transfer, a window appears displaying various pieces of useful information and error messages. Incoming data may, or may not, be visible as the transfer progresses, depending on the **Screen Output** option defined within the **Configuration** window. If that option has been set to either **Protocol** or **Both**, all transmitted data is displayed in a second window. In the case of an executable program, the incoming bytes will not be particularly informative. If you're downloading a text file, you'll actually be able to read the text as it's being received.

All file transfers can be aborted at any time, by pressing RUN/STOP, or by clicking on the close gadget. *Dialogue* will return to terminal mode with the text **Uload aborted** or **Dload aborted** flashing on the status line. The remote system may be slower at detecting an abort. To abort some host systems, you may have to press Return or CTRL-X a few times before continuing.

### **Notes on Downloading and Uploading to Other Terminals**

Professional online systems will automatically inform you when to commence an upload or download. However, on occasion, you'll probably want to send and receive files to and from friends using a terminal program. Or you may have a need to transfer files between two of your own computers with dissimilar characteristics; i.e., C128 and Amiga, C128 and PC, etc.

Both terminals (or computers) must use the same protocol to upload or download successfully. *Compuserve B protocol can not be used for such file transfers!* It is strictly Compuserve-specific; i.e., the necessary host functions are not implemented in B-capable terminal programs. If both people (or both computers) have terminals that support Ymodem Batch, that protocol is probably the best bet because of its batch attributes, which allow multiple files to be sent continuously.

To begin a file transfer between two terminals, the *uploader* always starts first. Since both users are using terminal programs, there are no 'start signal' messages to assist you. All the timing is done manually. The uploader starts the process by pressing CBM-U or selecting Upload File from the menus, and after waiting a sufficient amount of idle time, the downloader continues by pressing CBM-D, or selecting Download File from the menus.

---

When transferring files using C1 protocol, the letters GOO may appear on the downloader's screen when the upload process has begun. These letters are part of the error-detecting protocol, and can be ignored.

### Checking the Upload/Download Information      CBM

Since *Dialogue* returns to terminal mode after each upload or download, you may occasionally miss seeing the transfer statistics. At any time after an upload or download, you are still able to recall the statistics of the most recent upload or download by pressing CBM-I, or by selecting **Info** from the menus.

## Protocol Descriptions and Usage

### Punter C1

Developed in 1984 by Steve Punter of Mississauga, Ontario, C1 replaced his previous protocol for the old PET computers. For a long time, C1 was widely accepted as the standard transfer protocol on Commodore 8-bit systems and has recently found its way into the Amiga and MS-DOS worlds. With the introduction of protocols such as Xmodem 1K and Ymodem Batch, Punter C1 has lost some popularity in recent years except on 'colour boards'. Still, you will often find this protocol used on BBS's that are run on C64's and C128's.

C1 features a variable packet (block) size (40-256 bytes) and a dual checksum method using both additive checksum and cyclic redundancy check. Do not confuse 'block size' with actual disk blocks; they are totally unrelated. More commonly in the telecommunications world, what Steve Punter has chosen to call 'blocks' are referred to as 'packets'. Because of C1's variable block sizes, it is possible to transmit files even if phone lines are very poor, by sending smaller packets between each error check.

The C1 protocol transmits the block size as part of the first block. Only the *sender's* block size needs to be changed. Because of this, the option is used only when uploading files. When downloading, a different block size must be set on the host system before initializing the download. See the online system's help for more information.

**Usage:** C64/C128 BBS's typically offer the commands **LOAD** and **SAVE** to permit downloading and uploading respectively. In a typical situation you would enter the **LIST** command to the host to see a list of the files available for downloading. After deciding which file to download, you enter the **LOAD** command. The BBS will ask for the name of the file you want. Enter that data and you will usually see "Waiting for start signal. A to abort, B to change block size". Press **CBM-D** to download.

The **Protocol Parameters** window will appear. If Punter C1 is not the selected protocol, press **Return** or click until it appears on the **Protocol:** line. Then exit the window by pressing **RUN/STOP** or by clicking the close gadget. Finally, you are prompted for a filename to use on your disk. Enter a filename and the transfer will begin. Pressing **Return** at the filename prompt will abort the transfer and return you to terminal mode from which you can type **A** to cause the host to abort as well.

If you encounter C1 on an MS-DOS or Amiga system, remember that these computers have different filenaming conventions. MS-DOS allows eight alphanumeric characters with an optional three-character extension. Many special characters are not allowed. Stick with letters and numbers. All letters are uppercase in MS-DOS filenames. Amigas are much more flexible with filenames but avoid using spaces or slashes.

---

## Xmodem

Developed for CP/M systems in 1977 by Ward Christenson. Xmodem has become the accepted standard for microcomputer transfers. *Dialogue* supports three variations of Xmodem:

**Checksum:** The original protocol which uses a simple add-up checksum and a packet size of 128 bytes. It also has a packet size of 128 bytes.

**CRC (Cyclic Redundancy Checksum):** This increasingly popular variation uses a more reliable checksum method.

**Xmodem 1K:** An extension of Xmodem CRC developed by Chuck Forsberg in 1982 after Ward Christenson released an official protocol description for Xmodem. Xmodem 1K permits packet lengths of 128 and 1024 bytes. The larger packet size improves the efficiency of transfers, especially for large files. Some host systems incorrectly refer to Xmodem 1K as Ymodem.

**Usage:** Check the online help of the host system to find the commands to view available files and to upload and download. When using *CompuServe*, select a LIB and BROWSE the files. *CompuServe* will display each file in the LIB with a description. After each file and description is displayed you are presented with a Disposition! prompt.

To download the file, enter `dow pro:xmo`. *CompuServe* responds with: **Starting XMODEM send. Please initiate XMODEM receive and press <CR> when the transfer is complete.** Press `CBM-D` and a filename prompt will appear. If necessary, bring up the Parameters windows and make any adjustments desired. Be sure that the Protocol: selection is one of the Xmodem variations. Enter a filename, press Return and the transfer will begin. When *Dialogue* indicates that the download is complete, press Return to inform *CompuServe*. Uploading is accomplished in a similar fashion. At a LIB prompt, enter `upl pro:xmo`.

## Ymodem Batch

Developed by Chuck Forsberg in 1987, Ymodem improves on his own Xmodem 1K protocol by including the filename(s) within the data transferred. Ymodem permits large groups of files to be uploaded or downloaded in sequence.

**Usage:** Once again, you will need to check the online help files of the host to find the required commands for that system. After informing the host of your desire to send or receive files, use `CBM-U` or `CBM-D` respectively to initiate the transfer within *Dialogue*.

When downloading with Ymodem, you have the option to transfer the files continuously without interruption, or to stop and prompt for a new filename or filetype each time. These options are defined on the Protocol Parameters window. If you've chosen the filenames to be selected by default, the downloaded files will be saved to your disk without interruption.

When uploading using Ymodem Batch, use the special File Selection Mode. After selecting a file, your choice will be blanked out and you can select another. Use `RUN/STOP` (or the pointer) to exit. If desired, you can mix drives since the drive number is also buffered. You can select about 20 files in this manner (depending on filename lengths). If you fill the filename buffer, the transfer will autoexecute.

## CompuServe B

This is an auto-executing protocol created exclusively for *CompuServe* in the early 80's. It was designed to make uploading and downloading from *CompuServe* as simple as possible. In 1987 *CompuServe* made some enhancements to B protocol to improve its transfer charac

---

teristics and named the enhanced version Quick B. *Dialogue* supports this enhanced version. Quick B improves upon original B by allowing larger packet sizes. The *Dialogue* implementation requests 128-byte packets at 300 baud, 512-byte packets at 1200, and 1024-byte packets at 2400 and above.

**Usage:** Downloading with Quick B is very simple. You don't even have to use CBM-D. Just enter `dow pro:b` at a **Disposition!** prompt. *CompuServe* will request **Filename for your computer:**. Enter the filename and everything is automatic. Since the CBM commands are not used and *Dialogue* doesn't prompt you for a filename, *Dialogue* will put up a device number requester. This will allow you to direct the file to the desired disk drive. When the transfer is complete, you will be returned to terminal mode automatically.

### Transmit Modification Characters

At various times, *Dialogue* allows you to transmit a stream of text to the host without having to type it directly. This transmission style is controlled by the **Transmit Mode** and **Transmit Delays** options within the Configuration window. Text can be read and output directly from the screen from a pre-programmed function key, macro definition, or auto-exec script. To help control what is and isn't transmitted, several options are available:

**Shift-Return:** (Appears as: `C/R`) Normally, any text transmitted from *Dialogue* ends in a carriage return. It is possible to force extra carriage returns if required. *Dialogue* converts any `C/R` symbol into a `CHR$(13)`. When creating an auto-exec script, the `C/R` symbol is generated by pressing **CBM-Return** instead.

**CBM- +:** (Appears as: `->`) This symbol is used to suppress the carriage return at the end of the text transmission. Text is transmitted up to the point of the `->` and no carriage return is sent. The cursor remains on the same line. This is handy when defining macros, or function keys, to enter a portion of a command which may require additional parameters.

**Up Arrow:** (Appears as: `^`) The caret symbol (`^`) informs *Dialogue* that the following character is to be transmitted as a control code. Control codes are not echoed back to the screen. They are useful for transmitting system control codes like `XON/XOFF` (eg: `^s` sends a system halt). See Appendix C for a list of control codes. Transmissions that end with a control codes are *not* supplied with a carriage return automatically!

**ESC:** (Appears as: `<-`) Similarly, it is possible to transmit ESC codes by using the ESC key, followed by an appropriate key.

**Backslash sequences:** (Replaces `CBM-N` from previous versions. `CBM-N` is still supported.) At times, you may need to transmit a character not handled directly from the *Dialogue* keyboard. Defining a custom macro or function key to transmit these characters is easy. Simply enter the ASCII value of the character following a `\` symbol. *Dialogue* automatically converts the ASCII value into a character or control code and transmits it. (eg: `\065` will transmit the letter a). For the sake of convenience and readability, several special characters have been predefined:

<code>\b</code> Backspace ( <code>chr\$(8)/^h</code> )	<code>\U</code> Cursor up*
<code>\d</code> DEL ( <code>chr\$(127)</code> )	<code>\D</code> Cursor down*
<code>\e</code> Escape ( <code>chr\$(27)/^d</code> )	<code>\L</code> Cursor left*
<code>\I</code> Linefeed ( <code>chr\$(10)/^j</code> )	<code>\R</code> Cursor right*
<code>\r</code> Carriage return ( <code>chr\$(13)/^m</code> )	<code>\###</code> (where <code>###</code> is a decimal number)
<code>\t</code> Tab ( <code>chr\$(9)/^i</code> )	transmit <code>chr\$(###)</code> value
<code>\</code> Allows <code>\</code> character to be sent	

\* mostly used for program control within execs

---

## Buffer Functions

### The Scrollback Review Buffer

*Dialogue* has a built in 'scrollback' review buffer which is always active, memorizing everything that is printed to the terminal mode screen. It automatically retains the last 9,000 characters of terminal mode activity in its memory. This allows you to review (or 'grab') anything you may have missed the first time, even if you didn't have a capture buffer open.

#### Viewing the Scrollback Buffer      CBM

To view the scrollback buffer, press CBM and the up-arrow key or select **View Review** from the menus. When looking at the scrollback buffer, text is displayed newest to oldest, from the bottom up. All regular terminal activity is not displayed on the screen while you are reviewing the scrollback buffer.

#### Scrollback Buffer Options:

RUN/STOP	Exits scrollback mode, and returns to normal terminal activity.
CRSR	Cursor keys will scroll up or down through the buffer.
SPACE	Fast scroll mode. Defaults to whatever direction was last used.
+ / -	Pressing plus or minus displays the next or previous screen buffer page.
CBM-G	Grab a screen of text into capture buffer (described below).

*Note: Each time the screen width is changed, or an auto-exec is started or ended, internal pointers are reset, and the memory used by the scrollback buffer is cleared.*

### The Capture Buffer(s)

#### Buffer Open/Close Control      CBM

Pressing CBM-B once opens the active capture buffer, which begins memorizing all the incoming screen text. The active buffer number is highlighted and starts flashing on the status line, along with the words **Capture On**. The two status counters on the status line will update as the buffer fills, keeping track of the amount of free space remaining. The buffer size differs depending on the buffer configuration. (See: **Buffer Configure**, page 16)

If CBM-B is pressed a second time, the buffer closes, and text memorization stops. To identify whether a buffer is open or closed, check the status line. The active buffer number flashes while open.

#### Switching Current Buffers      CBM      (top row)

It is possible within *Dialogue* to have up to eight separate buffer areas available at one time. This enables you to store different things in completely separate storage areas. The active buffer number is always displayed on the status line, and flashes when the buffer is open. To change to any other buffer number, press and hold the CBM key while pressing a number from 1 to 8 on the top row number keys. Note that *only* the number keys on the top of the keyboard will have this effect. The numeric keypad is reserved for other functions. Optionally, you can select **Buffer Number** from the menus, or press CBM-N and respond to the prompt with a valid buffer number.

The size and number of active buffers is controlled from the **Configuration** window. Increasing the number of active buffers decreases the size of each. If text exists in a buffer, and you decrease its size, some text may be transferred to another buffer.

---

All of the following buffer functions are performed on the currently selected buffer only.

### Grabbing Screens to the Buffer    CBM **G**

Pressing **CBM-G** or selecting **Grab Screen** from the menus transfers the current screen text directly into the capture buffer. This allows you to 'grab' an important screen after it's been received, even if you didn't have the capture buffer turned on. The grab function can also be used in conjunction with the **Scrollback Buffer**, or the **View Disk File** function to grab any text that is currently being displayed on the terminal mode screen.

### Reset Buffer    CBM **R**

When you select **Reset Buffer**, either by pressing **CBM-R** or from the menus, a requester appears prompting you to confirm your choice. The buffer currently selected will then be cleared.

### View Buffer (to Screen)    CBM **V**

This option enters the **View Buffer** mode, and displays the current capture buffer on the terminal mode screen. During this mode, incoming text will not be displayed.

#### View Buffer Options

<b>RUN/STOP</b>	Aborts and returns to terminal mode.
<b>CRSR</b>	Both sets of cursor keys scroll up and down through the buffer.
<b>Space</b>	Pressing <b>Space</b> while viewing a buffer enables <b>Fast Scroll</b> mode, defaulting to whatever direction was last used.
<b>+ / -</b>	Pressing plus or minus pages to the next or previous buffer screen.
<b>HOME</b>	Pressing <b>HOME</b> goes to top of buffer instantly.
<b>SHIFT-CLR</b>	Pressing <b>SHIFT-CLR</b> advances to bottom of the buffer.

### View a Disk File (seq)    **Alt V**

This function allows you to view a disk file, on the terminal mode screen, without actually loading it into memory. You can also selectively open/close the capture buffer to capture the incoming text. During this mode, incoming terminal activity is not displayed. You'll be prompted to enter the filename to view. Entering a directory symbol (\$) will display only applicable **SEQ** files, and you'll be able to use the highlight bar to make your selection. Entering a ? at the filename prompt allows selection of **ASCII** to **PETSCII** translation (see **CBM-L** and **CBM-S**, page 34).

The disk file is read from disk, and displayed on the screen. Since the file is not loaded into memory, you do not have all of the same viewing capabilities listed above.

#### View Disk File Options (Available at any time during the view)

<b>Space</b>	Pauses viewing, <b>Space</b> again continues.
<b>ALT-P</b>	Prints current terminal mode screen display.
<b>CBM-B</b>	Toggles current capture buffer open/closed (see above).
<b>CBM-G</b>	Grabs current screen to capture buffer (see above).

### Transmit Buffer    CBM **T**

Pressing **CBM-T** or selecting **Transmit Buffer** from the menus will cause *Dialogue* to enter the **Transmit Buffer** mode. Here, you'll be able to send the buffer over the modem to the

---

host system. When transmitting a buffer file, no error-detecting protocol is used. The text is sent exactly as if you typed it yourself. The **Transmit Mode** and **Transmit Delay** are pre-selected from the settings within the **Configuration** window.

Most host message editors have some limit to the number of text lines that can be entered at one time. Some require you to break up long messages into several smaller ones, and other editors allow very large messages, but with only a certain number of lines permitted before saving each segment. Normally this could cause problems when you attempt to send a buffer that is larger than the allowed number of lines. (As an example, you have a 50-line message, but the receiving system only allows you to enter messages with a maximum of 40 lines.)

*Dialogue's* **Transmit Modification Characters** (see page 29) maximize your use of the **Transmit Buffer** option when sending messages to these types of systems.

Use the CBM- + keypress (which appears as >) as a delimiter within the buffer editor. For example, if the host will only accept 40 lines, you would put a > (no other characters) on the 41st line. When the > is encountered, *Dialogue* will stop transmitting, display **Transmit paused** on the status line, and return you to normal terminal mode. Here, you can set up the online system for receiving more text. This may require you to save a portion of the message before continuing, or to save a complete message, and enter the information for a second one. When you're ready to transmit the remaining portion of the buffer, press CBM-T again, and *Dialogue* will continue from where it left off; i.e. it will resume transmission beginning with the line following the >.

Pressing RUN/STOP aborts the transmission, and returns to terminal mode. An abort can be issued at any time during the transmit, or during the transmit pause. If the transmission is aborted, the pointer is reset, so any attempt to re-transmit the buffer will start at line one again.

### **Transmit Single Line of Buffer**     ALT T

*Dialogue* allows you to transmit a single line of the capture buffer at a time. This is especially handy for processing lists of commands online. Every time ALT-T is pressed, the next line from buffer 1 will be transmitted. Since RS-232 and DMA cannot occur simultaneously, you must insure that all RS-232 activity has stopped before using this command when the transmitting buffer is contained within the REU (**RAM1 Buffer Configure**).

*Note: By default, the transmitted text will come from buffer 1. You may wish to reset the function to transmit from another buffer. To do this, use the reset command described below.*

### **Reset Transmit Buffer Pointer**     ALT R

Pressing ALT-R resets the pointers used by the ALT-T command (as described above) to the beginning of the currently selected buffer. Once reset, all lines transmitted using the ALT-T function will come from the new buffer, even if the current buffer is later changed. In this way, it is possible to use one of the buffers to capture incoming text, while another is used to transmit data out.

### **Printing The Buffer**     CBM P

Pressing CBM-P or selecting **Print Buffer** from the menus will cause *Dialogue* to output the current buffer to the printer device (as set within the **Configuration** window). A countdown of lines being printed will be displayed on the status line as it prints.

---

The text being printed will always be displayed on screen as it's printing, regardless of the Screen Output option setting in the Configuration window.

### Loading Files Into The Buffer **CBM** **L**

This function prompts for a filename, and allows you to load a sequential (SEQ) file from disk, into the buffer, ready to be transmitted or edited. Note that loading text does not clear the buffer! It is appended to the end of the current contents of the buffer. You may wish to manually reset the buffer before loading.

Remember also that the buffers can be different sizes, depending on the Configuration settings. *Dialogue* will always load as much as it can into the buffer. Any leftover portion will be ignored.

Since the 128 uses a different form of ASCII from the rest of the non-Commodore world, an option has been provided to translate ASCII disk files into Commodore ASCII (commonly known as PETSCII). Entering a question mark at the filename prompt will produce an ASCII Translation? requester box. Answering Y to this question translates the incoming ASCII to PETSCII so that the text in the buffer will be correct with regard to upper/lower case.

### Save Buffer to Disk **CBM** **S**

This allows you to save the current buffer to disk, under the filename you supply when prompted. If the buffer is empty when you attempt to save it, you'll be alerted, and the command will abort.

Normally, buffers will be saved to disk in Commodore PETSCII. Before saving, you may choose to have the text converted to true ASCII instead. Entering a ? at the filename prompt calls up the ASCII Translation? requester. If you answer Y, the text will be converted to true ASCII as it's saved to disk. This is only advised if you plan on using the file(s) with other ASCII software, or on other computer systems that do not support Commodore's PETSCII. If the text is to be transmitted or printed using *Dialogue*, no translation is required.

**Note:** *Dialogue* also supports translation to and from true ASCII when uploading or downloading text. (See Protocol Parameters Window, page 25.)

### Buffer to Disk **Alt** **B**

On occasion, you may want to capture an unusually large amount of incoming text, but have insufficient memory space in the capture buffer. Pressing ALT-B or selecting Buffer to Disk from the menus will allow you to send data to a disk file. For this option to work properly, Flow Control must be turned on (either XON/XOFF or CTS/RTS).

When you open the buffer, you'll be prompted for a filename. If the file already exists, you'll be asked to confirm you wish to replace it. *You cannot add to an existing disk file.* When you press ALT-B a second time, the file will be closed, and text will no longer be memorized.

As data is received on the terminal mode screen, it will be stored in a regular buffer area. When the buffer fills, a flow control signal is sent to stop the host (or the MNP modem) from sending. The buffer is then saved to disk, cleared and a 'continue' flow control signal is sent. Text input continues, and the process repeats.

**Note:** *The C128 cannot access the serial disk drive and the RS-232 port at the same time, and attempting such an action will cause incoming text to become garbled!*



---

To permit error-free buffering to disk, it is necessary to stop the host from transmitting new data while the serial port is being accessed. *Dialogue* handles this by using the flow control codes as set within the **Configuration** window. Since not all host systems support XON/XOFF flow control (and not all users will have MNP modems), you will not always be able to use the **Buffer to Disk** option effectively. In such situations you will need to stop the host and save your buffer manually.

## The *Dialogue* Buffer Editor

Pressing CBM-E, or selecting **Editor** from the menus, activates the *Dialogue* full-screen buffer editor. You can use it to edit captured text, or to pre-compose messages for buffered transmission to an online system. The editor works in all display modes of *Dialogue*: 40/80 columns and 25/50 lines.

There are two available edit modes. **Mode 1** displays the capture buffer being edited using the full screen. **Mode 2** splits the screen in half, and displays the review buffer in the top half of the screen, and the editor screen in the bottom half. This option allows you to actually review the system while you work in the editor - an invaluable feature when responding to a long, complex message. You alternate modes within the editor by pressing CBM-M.

### Special Editing Keys

<b>TAB</b>	Tab works like a real tab, and moves the cursor to next tab position.
<b>SHIFT-TAB</b>	Sets or clears a tab stop at the current cursor position.
<b>HOME</b>	First press homes the cursor on screen. Second press advances to buffer top.
<b>SHIFT-CLR</b>	Once to set cursor on last line of page. Second press to advance to buffer end.
<b>HELP</b>	Selects the drop-down menus for the editor.
<b>RUN/STOP</b>	Pressing RUN/STOP exits editor, and returns to terminal mode.

### Changing Editor Environments

<b>ALT-A Auto-execute buffer:</b>	This will run the buffer as an exec from the editor. The buffer is transferred into the exec buffer. <i>Warning! If the exec uses the buffer that contains the exec code, save your work before issuing this command or you'll lose your source code!</i>
<b>ALT-L Load defaults:</b>	Loads stored tab stops, margin settings, and buffer mode.
<b>ALT-S Save defaults:</b>	Saves the current tab stops, margin settings, and buffer mode.

*Note: These settings can be set as an auto-load default file from the Configuration window.*

### Editor ESC Functions (Press ESC first, then the appropriate key)

<b>ESC-D</b>	<b>Delete line:</b> Deletes the entire line the cursor is currently on.
<b>ESC-I</b>	<b>Insert line:</b> Inserts a blank line at the position of the cursor.
<b>ESC-P</b>	<b>Clear to start:</b> Clear from current cursor position to start of line.
<b>ESC-Q</b>	<b>Clear to end:</b> Clear from current cursor position to end of line.
<b>ESC-@</b>	<b>Clear buffer:</b> Clear buffer from current cursor position to end of buffer.

*Note: An interesting side effect allows all the ESC and ALT key functions to be interchangeable. All ESC functions also work with the ALT key, and all ALT functions work if you press the ESC key first.*

---

## **Editor Set-Up and Operations** (Pressed while CBM key is held)

- CBM-<:** Sets left margin for text entry. Margins can be overruled by the cursor keys.
- CBM->:** Sets right margin for text entry. Margins can be overruled by the cursor keys.
- CBM-D:** Delete the line the cursor is currently on.
- CBM-L:** Load buffer with disk file. (Enter ? for ASCII translation option).
- CBM-I:** Insert a blank line at the position of the cursor.
- CBM-M:** Toggles buffer Mode. Normal or split screen (see above).
- CBM-R:** Sets a Range of lines in reverse, for subsequent CTRL commands (see below).
- CBM-W:** Toggles Wordwrap on/off (only within the editor).
- CBM-S:** Save buffer to disk. (Enter ? for ASCII translation option).
- CBM-INST:** Prompts for the number of lines to be INSERTed at current cursor position.
- CBM-(1-8) [Top Row]:** Select current buffer to edit.

## **Range Operations** (Pressed while the CTRL key is held)

These options all function on a selected range only. The range is the group of buffer lines, set with the **CBM-R** command (described above).

- CTRL-C:** Copies the range to the line the cursor is on. This is an insert operation; nothing is overwritten. Ranges can even be copied from one buffer to another. Original text is always left unchanged.
- CTRL-D:** Deletes the entire range, after prompting for a confirmation.
- CTRL-E:** Erase range. Similar to Delete Range except the space held by the deleted text remains. Text is replaced with blank lines.
- CTRL-M:** Move range. This option deletes the range from its current location, and inserts it at the current cursor position.
- CTRL-S:** Saves just the text within a range to disk. Allows portions of buffer to be saved.

## **Other CTRL Functions**

- CTRL-I:** Toggle Insert/overwrite mode.
- CTRL-R:** Range select. Allows selection of a box (partial line) range; as opposed to **CBM-R** which selects full lines. Highlight the desired range using cursor keys or the mouse. GEOS-style users must exit with RUN/STOP or Return (in both range modes).
- CTRL-W:** Wordwrap. Toggle wordwrap on/off.

---

## The Macro Keys

### The Macro Definition Window **CBM** **M**

*Dialogue* treats 'macro keys' differently from the function keys. From the auto-dial menu, you were able to pre-program four function keys for each individual number in your directory. In addition to these keys, *Dialogue* supports ten other 'always active' macro keys. These keys are selected by pressing the ALT key, and any of the numerals on the numeric keypad. They are defined within the **Macro Definition** window accessed by pressing **CBM-M** in Terminal Mode, or by selecting **Macro keys** from the menus.

Macro refers to a single character that invokes a series of keystrokes. Macro sequences can be used to store frequently typed keystrokes for online systems. Also, they can emulate many of *Dialogue's* command sequences, allowing macro keys to be used for controlling many program operations.

While in the **Macro Definition** window, the highlight bar is active, and can be moved with the cursor keys, or the pointer. The following other keys are available while the macro window is displayed:

- Return:** Invoke highlighted macro. (The button can also be used.)
- RUN/STOP:** Exits the macro window, and returns to terminal mode.
- L:** Loads a new macro set from disk.
- S:** Saves the current set to disk. (Prompts if file exists).
- E:** Enter/edit. Define a macro definition for the highlighted key.

Each macro definition can have a maximum size of 60 characters. For longer macros, it is suggested you use the *Dialogue* auto-exec script language.

When defining macros, several special codes are available:

Pressing **CBM-C** will display a small underlined **c**. This signals the macro that the following text is to be sent as if the **CBM** key was pressed. In this way, you can emulate **CBM** key commands. Example: **c l "filename"** - This example would emulate pressing **CBM-L**, and load the file named **filename** into the capture buffer.

In the same way, pressing **CBM-A** will produce a small underlined **a**. This signifies that the following character is to be processed as if **ALT** was held. Example: **a l "rle.ext"** - This example would emulate pressing **ALT-L**, and load the extension file named **rle.ext** into the extension memory.

As well, all the **Transmit Modification Characters** (as described on page 29) apply.

### Executing Macros **Alt** <numeric keypad digit>

The macro definitions can be executed from Terminal Mode at any time by pressing the **ALT** key, and the corresponding digit on the numeric keypad to the right of the regular keys. While the macro window is displayed, pressing **Return** or the button will invoke the highlighted macro.

---

## Terminal Mode Commands

### Modem Hook Control      CBM **H**

This option has different functions, depending on the type of modem you are using. See Appendix A for more information on how this command works with your modem. For dumb modems, CBM-H acts as a manual hook control, and will toggle the modem on and offline. For smart modems that support hardware hangup, it causes the modem to disconnect from a remote system. If CBM-H does nothing with your modem, try setting some of the modem's switches differently. Look for the term DTR (Data Terminal Ready) in your modem manual. If a carrier is present when attempting to disconnect, a requester will appear, and you'll be asked if you're sure you want to disconnect.

### Handling Non-Visible Characters      Alt **N**

Pressing ALT-N causes the N on the status line to begin flashing, as *Dialogue* is put into a special mode in which characters normally not visible (those with an ASCII character value is less than 32) are displayed as a caret (^) followed by the CTRL character equivalent.

Examples:      CTRL-Q = chr\$(17). Appears as ^q  
                  ESC = chr\$(27). Appears as ^[

This mode can be used to determine or study the terminal emulation used by a system, or as a means of checking for control codes when creating an auto-exec script.

### Viewing the Function Key Definitions      CBM **F**

Since the four top function keys on the 128 are defined differently for each system you dial, it can be easy to forget what you've defined. Pressing CBM-F at any time during terminal mode will bring up a window showing the display of the key definitions. The highlight bar is active, and can be manipulated with the cursor keys or the pointer. Press RUN/STOP to exit. Pressing Return, or the button, when the highlight bar is over one of the options will transmit the selected data.

### Waiting for a Call (Host Mode)      CBM **W**

This enters *Dialogue's* Wait for call mode. While in this mode, the terminal will remain idle until it detects a carrier. It will then return to terminal mode. Using the auto-exec script language of *Dialogue*, it is possible to create a complete login mini-BBS system. Pressing RUN/STOP will abort the Wait for call mode, and returns to terminal mode.

*Note: Dialogue does not program the modem to auto-answer. You must do this before entering Wait for call mode. Smart modems must be set to auto-answer with ATSO=1, and dumb modems usually require a switch to be flipped for answer mode. More information on modem files is available in Appendix A.*

### Setting the *Dialogue* Clock      CBM **\***

This is where the real-time clock, that is displayed on the status line, can be set. When you press CBM-\*, you'll be prompted to enter the time. You *must* use the format: hh:mm:ss. The real time clock works on a 12-hour system, and you'll be prompted to enter AM or PM after you've entered the correct time.

---

*Note: If you're using a clock card with your C128, you can disable the initial zeroing of the TOD clock by changing the appropriate POKE (indicated in the REM statements) in the "dialogue" BASIC boot file. Following this line, you may add any code required to set up the TOD clock registers on CIA #2 (\$dd08-\$dd0b). Note also that the "intro" machine language boot file loads in the range \$0400-\$0a00. Thus, any machine language required to set up the clock cannot occupy this range of memory.*

### Split-Screen Conference Mode      CBM

Many systems have exciting conference chat capabilities online, but the control over text being entered is often very poor. Since these modes are 'live', incoming text can often interfere with your typing. Entering CBM= enters a special split-screen mode, occasionally referred to as **Chat Mode** on some terminals. It allows text to be pre-entered in a small window at the bottom of the screen, before being transmitted. This special window allows you to pre-type several lines at once, and even correct any typing errors, before the text is actually transmitted. It acts like a mini (5-line) screen editor with wordwrap, with all cursor keys active, including INS/DEL, HOME, and CLR. As well, the regular C128 escape codes ESC-I, ESC-D, ESC-P, ESC-Q and ESC-@ are supported. While you type within this window, all incoming text is still displayed above, without interruptions.

Pressing Return at any time will transmit the line the cursor is on, regardless of the cursor's position on the line. In this way, it's easy to resend a line, simply by cursoring up over it, and pressing Return again. Shift-Return will bring the cursor to the next line without transmitting the line. To exit the split-screen conference mode, press CBM= again.

Text is transmitted using the preset method and speed as defined within the **Configuration** menu (either **Stream** or **Reflect**). If you find text being garbled as it is transmitted, try a slower speed setting by increasing the **Transmit Delay** setting from the **Configuration** window (see page 15).

The terminal mode screen clears when enabling or disabling the conference mode. To re-display the last screen, use the **Scrollbar Buffer** option (described on page 30).

*Note: Conference Mode is not available during VT100, or VT52 emulation, because these modes rely on a full 24-line screen to operate correctly.*

### How to Send a 'Break' Interrupt      RUN/STOP

Some larger systems expect the terminal to be able to generate a 'break'. This is merely a short interruption in the normal handshake between modems. *Dialogue* can generate a true break of approximately 230 milliseconds by pressing ALT-RUN/STOP, or selecting **Transmit Break** from the menus. The break can be thought of as a hardware interrupt. While CTRL codes such as CTRL-C and CTRL-O are considered interrupts, they are software-based, and won't work if the system isn't accepting characters.

### Toggle Secondary Commodore Graphics Characters

When in the CBM terminal emulation mode, you may require the ability to transmit the CBM key character set, in addition to the upper/lower case characters. Since *Dialogue* uses the CBM key as a command key, you'll need to disable the CBM key commands.

When the CBM graphics mode is active, the S on the status line will flash, and the CBM key will produce the regular Commodore graphics, rather than the *Dialogue* commands. You'll

---

still be able to access all of the CBM key commands from the menus when you need them. To restore the CBM key commands, press ALT-S again.

*Note: This function is not available in terminal emulation modes other than CBM.*

### Clearing the Terminal Mode Screen      CBM **CLR**

Pressing CBM-CLR at any time clears the terminal mode screen.

### Quitting *Dialogue*      CBM **Q**

This function allows you to exit back to BASIC without having to turn the computer off. You'll be prompted to confirm you actually want to quit, and if you have information still stored in the buffer, a warning is displayed. If you are using a 1700 or 1750 REU, and quit *Dialogue* with text in the REU buffer, that text remains intact until the computer is shut off (or until another program uses the REU). When you reboot *Dialogue*, respond with No to the Clear REU question and your buffers will be just as you've left them.

### Print Screen      **P**

Pressing ALT-P at any time within terminal mode dumps the current screen, to the printer. Only the terminal mode text is printed, not the status line.

The print screen option is also available in several other circumstances:

- Viewing capture buffer with CBM-V command.
- Viewing review buffer with CBM-up arrow command.
- Viewing a disk file with ALT-V command.

### Release Cursor Mode      CBM **CRSR**

This mode, accessible by pressing the CBM key and the cursor up/down key, or by selecting **Release Cursor** from the menus, is one of *Dialogue's* best terminal mode features. It releases the cursor from its present position on the terminal screen, and allows you to move it around the screen freely (even in **Conference mode**), with the option to edit, and resend any on-screen text. Perhaps the best use of this function is with online message line editors when you discover a typo. Instead of having to retype the whole line in again, you merely cursor up to the old line, fix the mistake, and resend the entire line again. It's like using a full screen editor, even when the host doesn't support one!

When the release mode is invoked, the cursor's **Flash Mode** attribute (as set from the **Configuration** window) will be switched to its opposite, giving a visual indication of the mode change. Non-flashing cursors will flash, and flashing cursors will become solid.

#### Release Mode Options:

**RUN/STOP** Aborts release mode. The cursor returns to its original position with no change.

**Shift-Return** Generates an on-screen c/r symbol. This will stop the transmission of the line at this point, and transmit a carriage return.

**CBM- +** Pressing the CBM key and the plus key generates a small right arrow on the screen. It functions similar to the c/r symbol above, except that no carriage return is to be transmitted. The line is transmitted only up to the symbol, and the cursor remains at the end of the transmitted text.

---

**CBM-Return** This is the power key that sends the text. Text is transmitted starting at the current cursor position, and continues until it reaches either the end of the line, or one of the above special symbols.

**ESC Keys** Full support of Commodore's ESC-I, ESC-D, ESC-@, ESC-P and ESC-Q functions.

### Modem Initialization String

Most smart modems have powerful features that can be accessed by sending commands to the modem. These commands are described in your modem manual. When defining the modem file from the **Configuration** window (see page 18), you can use the **M** command to preset a string of commands to be transmitted to your modem each time *Dialogue* is loaded. Any time you reset your modem, you may be required to resend this initialization string to your modem. Pressing ALT-I works only when you are not connected to a system.

### Loading an Extension Module

Almost everyone can think of a special command or feature they'd like to see in their 'dream' terminal. Hopefully *Dialogue* is able to satisfy most of your wants by offering many features not found in other terminal programs - on any computer... Still, there is always room for more. Luckily, *Dialogue* has a built in a way of handling this need to expand. To provide special new 'un-thought of at release time' features, *Dialogue* has reserved an area dedicated to program extensions. Extensions are modules which are loaded in, and executed as part of the program. There are two types of extension modules. Those that merge into the code of *Dialogue* offering extra functions, and those which are actually separate programs, which take over control of the program.

On the *Dialogue* disk, several extension modules are provided. Appendix B contains a list, and description of the extension modules included on the *Dialogue* disk. Others will be available in the future, to provide many years of dream terminal features.

Pressing ALT-L, or selecting **Load Extension** from the menus will prompt for the filename of the extension module you wish to load. Each extension will function differently, and have its own set of instructions.

Not all extension files will be active at all times. Once an extension file has been loaded, it stays in memory until another is loaded. It can be re-activated at any time by pressing only Return at the extension filename prompt.

It is possible to set *Dialogue* to automatically load one extension regularly as part of its load sequence. In this way, the extension becomes part of *Dialogue* (see **Default files** - page 17).

### Commodore DOS Commands **CBM**

*Dialogue* supports a built in DOS wedge where all standard Commodore DOS commands can be entered, along with some additional features. The DOS wedge prompt is available by pressing **CBM- >** or by selecting **DOS Wedge** from the menus. Access to the DOS wedge is also available by entering > (the 'greater than' symbol) at any filename prompt.

Most of the DOS functions accept optional drive numbers or device numbers. A new device is selected by appending the new device to the end of the command with a comma (eg: **v0,9**) and doing so changes the default device to the new device. In this way, a device command need only be issued once. The active device number is always displayed in parenthesis within the wedge prompt.

---

Below is a brief list of available DOS commands. For full information on how to use them, consult your computer and/or disk drive manuals.

**Commands:**

**v** = Validate (collect) disk. (Do not use on GEOS disks!)  
**i** = Initialize drive  
**n** = New a disk (Format)  
**s** = Scratch disk files. (Supports pattern matches)  
**d** = Duplicate drive to drive. (Only works with dual drives)  
**c** = Copy Files drive to drive  
  
**#n** = Selects device n  
**#** = Enters device selection mode  
**\$** = Disk directory (supports pattern matches)  
**/** = 1581 partition support (see page 10)  
**//** = 1581 partition selection (see page 10)  
**@** = Display disk error channel on status line.

**Return** = Abort, and return to terminal mode or filename prompt.

Additionally, commands can be directed to the printer from this prompt, allowing special printer control codes to be sent if required. This function provides an effective method for turning on your printer's NLQ mode or other various printer options.

**p:** The **p:** prefix instructs *Dialogue* to output a command line to the printer. The various escape, text and control characters follow the same syntax as other *Dialogue* output strings, like those used with the function keys, or macro definitions (see **Transmit modification characters** - page 29).

Example 1: `p:<-4\007This is the equivalent to: ESC "4" CHR$(7)`

What the above example does:

**p:** Tells *Dialogue* to send the following data to the printer.  
**<-** The ESC character is the printer's attention getter. Displayed by pressing ESC.  
**4** The ASCII character 4. Sent following an ESC, it's used as a control code.  
**\007** Transmitted as CHR\$(7). (Equivalent to ^g.)

The above example sends an ESC 4 followed by a CHR\$(7). On most Epson printers, this would instruct the printer to first change its character set to italics, and then ring the printer bell.

Anything your printer is capable of understanding can be sent to it using this system. The example below will transmit a chr\$(14) and the text **Dialogue Printer Dump** to the printer. Since chr\$(14) is the code for expanded print on many printers, this provides an easy way to create titles for your printouts. When dumping straight text to the printer this way, you must remember to include a carriage return to complete the line. The text will not be printed without it.

Example 2: `p:;14Dialogue Printer Dump\r`



---

## The *Dialogue* Auto-Exec Scripts

One of the most powerful features of *Dialogue*, is the auto-exec script language, which allows the operation of the program to be completely, or partially automated.

Imagine the ability to log onto a group of BBS's, read and capture messages, send messages, log off, and save the messages on disk, all while you're asleep! With some imagination, almost anything is possible using the *Dialogue* auto-exec script capabilities. When using pay systems such as *CompuServe*, and *Genie*, the ability to have the terminal do everything automatically can shorten the time online, and thus save you money.

Some sample uses for auto-exec scripts:

- Automatic system logon.
- Loading in a specific macro key definition set, and/or an extension file for each system.
- Complete automation of an online session.
- Mini-BBS operation
- Intelligent automatic control of buffer captures.

### What are auto-execs?

An auto-exec (short for automatic execution) is a sequential file which contains a list of commands that instruct *Dialogue* to perform particular functions. These commands form a language which control everything, including making the program operate as if you had actually pressed any of the key sequences.

*Dialogue's* auto-exec script language uses powerful one or two character commands. Many of the commands functions are similar to BASIC. The language allows scripts to be as simple or as elaborate as you wish.

### Starting an Exec Script



There are two different kinds of auto-exec scripts. The 'auto-dial logon auto-exec' is linked to the phone number of each phone directory entry. These execs are executed when a connection is established.

It is also possible to invoke a generic auto-exec script from terminal mode at any time by pressing ALT-A, or by selecting the Execute Exec option from the menus. Unlike the auto-dial logon scripts, these auto-execs can have any filename. When the exec script is loaded, it will begin working automatically. Normal terminal operation is available while the auto-exec script is functioning. The script functions in the background. The only indication that a script is being executed is the flashing A on the status line. Executing scripts can be aborted at any time by pressing CBM-RUN/STOP.

### Viewing an Executing Auto-Exec Script

Although the auto-exec's normally work in the background, virtually invisible to the user, it is sometimes handy to see the auto-exec's progress. This is especially true when you are first creating the exec script. *Dialogue* has the provision to have the exec activity monitored in a small window just below the status line. The display can be toggled on or off by use of the 'I' (lower case L) command within the exec script. More on this will be discussed below, with the rest of the script language commands.

---

## Writing an Auto-Exec Script

Without a doubt, writing auto-execs isn't simple. It requires you to know *Dialogue* fairly well. If you're just beginning to use *Dialogue*, it is suggested that you hold off on programming an auto-exec right away. It is also helpful to have some understanding of programming BASIC, since many of the exec commands available are similar in meaning to BASIC commands.

Auto-execs can be used to automate practically any portion of your terminal use, from logging on and entering your name, to everything else. Once you have some idea about the scope of your auto-exec project, it is important to get on the host system and look around. Take note of the various prompts that you'll need to respond to, and the appropriate responses. Also note whether a session is always identical, or whether other prompts may come up when you have mail, or there is a new bulletin, etc. Since an auto-exec script only executes what you tell it to, you have to carefully prepare it for every possible online option.

Unlike the auto-dial logon scripts, which activate automatically when connected to a system, the regular auto-execs can have any filename, and be used for any purpose at any time. They are not specific to any one system or number. This allows you to write small exec scripts to handle a multitude of different functions online. These can be called using a macro key, or directly from the keyboard at any time with the ALT-A command (described above).

To create an auto-exec file, simply enter the editor with CBM-E. Clear the current buffer (or select an empty one) and enter the lines. Be sure to save your work. The remainder of this section of the manual (the last section) is devoted to the script language. First, the commands and functions are listed. This is followed by syntactical considerations of the coding and of the command descriptions. Finally, all commands are described fully with coding examples.

### The *Dialogue* Auto-Exec Script Language

Below is a quick reference of the available commands, and their meanings.

# = device selection	is =if status
\$ = temp string	l = (lower case 'L') toggle line display of exec activity on/off
; = comment line	on = incoming text trap
b = extract from buffer	p = print to screen
e = end auto-execute	r = return from subroutine
ei = endif	s = sleep (10ths of seconds)
f = file chain	t = transmit
g = goto	w = wait
gf= gosub file	wt = wait for time
gs = gosub	y = yes/no test
if = if conditional	

A = ALT key command (or **a** - generated by pressing CBM-A)

C = CBM key command (or **c** - generated by pressing CBM-C)

\ = non-printable character (see **Transmit Modification Characters**, page 29)

### Brief Glossary of Terms and Exec Script Functions

Colons : separate multiple commands on a single line.

Semi Colons ; indicates that following text is to be ignored. Used for comments.

Square brackets [ ] indicate optional information.

Angle brackets <> indicate mandatory information, to be entered as shown.

---

**Label #** a number at the beginning of a line in the auto-exec script which acts as a pointer to that particular line.

**PETSCII value** a code between 1 and 255 which the C128 uses to represent characters and/or keystrokes. Equivalent to `chr$(value)`.

**Quotes** "text" indicates any text, which must be enclosed in quotes.

### Script Language Programming Format

If you like, you may begin each line with a line number; however, it isn't necessary. *Dialogue* treats numbers at the beginning of lines as labels for use with the `goto (g)`, and `gosub (gs)` statements. When encountering either, *Dialogue* will search the exec script for the correct line label. Only those lines that are jump points *need* labels. The range of label numbers may only be within 1-255. Use each label only once. *Dialogue* does *not* check for re-use. As with BASIC, separate commands may exist on one line, and should be separated by colons (:).

Spaces within the auto-exec script are ignored, unless contained in quotes. A convenient way of writing exec scripts is to place label numbers within the first four spaces on a blank line, and indent the script code in four spaces. You can set a special tab setting in the editor to assist with this.

```
Example:      1          gs 200
                t "this is a test"
                g 1
                200       s 10:r
```

This example would transmit **this is a test** every second. It will all make more sense to you after reading the following documentation on the commands.

### Command Definitions

**Command: p**           **Title: print to screen**

- Prints the text within quotes to terminal screen. Text is *not* transmitted to host.

Syntax: `p <"text">`

Occasionally you may want to print a message to yourself, to keep track of what's going on in the exec script. This command will cause the text in quotes to be printed to the screen at the current cursor position.

Example: `p "No mail was waiting"` Prints the message on the terminal mode screen.

**Command: t**           **Title: transmit**

- Transmits text to host system

Syntax: `t <"text">`

This is the command you'll use the most. The characters or control codes in quotes are to be transmitted out over the RS-232 channel to the host system. (See Appendix C for information on valid control codes that can be transmitted)

When the closing quote is reached, an carriage return will automatically be added, unless a CTRL code was transmitted as the last data in the quotes. In that case, the carriage return is suppressed. It is also possible to manually suppress the transmitting of a carriage return by using the CBM+ character (see **Transmit Modification Characters**, page 29)

---

Examples: **t "quit"** Transmits the word quit, and a carriage return.

**t "y->"** Transmits "y", and *no* carriage return. The right arrow is produced by pressing CBM-+.  
**t ""** Transmits only a carriage return  
**t\$** Transmit temporary string  
**t "^c"** Transmits a CTRL-C code, and *no* carriage return.

The transmit speed can be altered in the Configuration window with the mode, and delay settings.

**Command: y** Title: yes/no test

- Prompts for yes/no keypress.

Syntax: **y** <"text">

This command will display a requester, prompting the exec user for an actual Yes or No selection. Execution of the exec script is halted till some input is received. If Yes is chosen, the command on the same line will be executed. If No is chosen, the command drops to the next line of code.

Example: **y "End auto-exec":e** If Y or Return is pressed, the end command will be executed. Otherwise, the exec will continue with the next line of code.

The text following the **y** command will appear within the requester. A maximum of 40 characters are allowed for the question text. The question mark is added to the end automatically.

**Command: w** Title: wait

- Wait for condition to occur before continuing exec.

Syntax: **w** <"text">

*Dialogue* is instructed to wait for the text inside the quotes to be received over the modem, before continuing. This is useful when waiting for a specific prompt. The prompt text would go between the quotes, and the exec will continually search all incoming text until the prompt appears, and then continue executing through the next command in the script.

Since Commodore DOS doesn't allow access to the RS-232 and serial port at the same time, you can use the **w** command to wait for a situation where the host stops sending characters, such as a prompt, before accessing the disk for any reason.

Example: **w "Command: "** Waits until the text prompt "Command: " is received.

**Command: g** Title: goto

- Jumps to the line which starts with the label number specified.

Syntax: **g** <label #>

Although line numbers are not required at the start of every line, the line numbers are used to 'point' to various places within the exec script. When a **goto** is encountered, the program jumps forward or back searching for the label numbers, and then continues executing the auto-exec from the command which begins with the given label number. These numbers need not be in any specific order. If no matching label number is found, an **Undefined label number error** message appears, and the exec is halted.

---

Example:            10 p "test" Prints "test" to the terminal mode screen, and  
                    g 10       jumps back to line 10 continuously.

Command: **gs**        Title: **gosub**

- Calls a subroutine within the current auto-exec file.

Syntax: **gs** <label #>

Jump to the line in the auto-exec which begins with the given label number, and execute the subroutine until a **return** command is issued.

You can also nest subroutine calls within other subroutines. This means a subroutine can jump to another subroutine. The limit of this type of nesting is 10 deep, after which an **Out of memory error** will be generated, and the exec will be aborted. When nesting subroutines in this way, it is handy to indent the commands deeper, to allow easier understanding.

If a specified label number doesn't exist, an **Undefined label number error** appears, and the exec aborts.

Example:            **gs 1: e**  
                    1 p "this is a test"  
                    **r**

The above example jumps to the routine at label #1, which prints **this is a test** to the screen, and then returns. The **e** command ends the exec.

Command: **r**           Title: **return from subroutine**

- Ends subroutine, and execution is transferred to the code following the most recent **gosub** command.

Syntax: **r**

Equivalent to a **RETURN** in BASIC, the **return** command terminates the current subroutine. Control is returned to the exec script following the most recent **gosub** (**gs**) command. Since nesting of subroutines is supported, it is important to remember to return from each subroutine individually. If a **return** command is encountered when no **gs** call had been issued, a **r without gs error** will be generated, and the auto-exec will be aborted.

Examples:           1 **gs 2** This will continuously loop through the subroutine  
                    **g 1**       which will print test on the screen.  
                    2 **p "test"**  
                    **r**        Return from subroutine.

Command: **e**           Title: **end auto-exec**

- Terminates the auto-exec currently running.

Syntax: **e**

When encountered, the auto-exec currently running is terminated. If the active script was part of a deeper nested exec script, control will return to the original. When all execs are complete, the **A** on the status line stops flashing. If no **e** command exists, the script ends when there is no more code to execute.

Examples:           **p "this is a test"** Prints "this is a test" on the screen.  
                    **e**        Exec ends, and program returns to normal terminal mode.

---

**Command:** gf      **Title:** gosub a file

- Calls another auto-exec file, as a subroutine.

**Syntax:** gf <"filename">

This command acts in a similar way to *gs*, except the subroutine is contained within a separate file. When encountered, the specified auto-exec script will be loaded into memory and executed from its beginning. **Only one file subroutine can be used at a time.** A call to a second, will end the first, etc.

A file subroutine ends, when the *e* command is executed, or the end of the auto-exec text is reached. At this point, the program returns to the original exec script, and continues executing normally.

**Example:**            gf "filename" Loads filename and executes it as a subroutine.

**Command:** f      **Title:** file chain

- Ends current script, and executes a second.

**Syntax:** f <"filename">

When encountered, the current exec script is cancelled, and the new script filename (in quotes) is then loaded in, and executed from its beginning.

**Example:** f "mail" Loads in the auto-exec script named mail, and executes it.

**Command:** s      **Title:** sleep

- Halts execution of the exec script for the defined amount of time.

**Syntax:** s <1-255> (time is given in 10ths of a second)

Generates a realtime delay in the execution of the exec code.

An example of its use is a settling time. Modems and/or systems can't always accept data *immediately* after a connection. Human beings typically aren't as fast as *Dialogue*. They'll delay for a while, before sending a character, but the exec script can be very fast. Often too fast for the modem or system to handle, so time delays are sometimes required.

The time of the delay is defined in tens of seconds between 1/10 and 255/10. This gives you a maximum delay of  $255/10 = 25.5$  seconds. For longer delays, either stack more than one *sleep* command, or use the *wait time* (*wt*) command.

**Examples:** s 10      Delays the script for 10/10ths = one second.  
              s 2      Delays the script for 2/10ths = 1/5 of a second.

**Command:** wt      **Title:** wait for time

- Halts execution of auto-exec until time (or time delay) occurs

**Syntax:** wt <"HH:MM:SS[a]/[p]">

This command has two uses. To wait until a specific time appears on the status line time of day clock, or to create a time delay which is longer than possible with the *s* (*sleep*) command.

The two uses are distinguished by the use of the *AM* or *PM* flag. If either the *a* or *p* is included, *Dialogue* assumes that the delay is based on the time of day clock. Otherwise it counts down the specified amount of time starting at the current time.

---

Examples:            **wt "12:30:00p"** Wait until 12:30pm before continuing exec  
                     **wt "00:30:00"** Wait 30 minutes before continuing

**Command:** ei            **Title:** endif

- End the if conditional structure

Informs *Dialogue* that it has reached the end of an **if** conditional structure. *Dialogue* will then re-execute the first **if** in that conditional structure. See instructions for **if conditional** below.

**Command:** if            **Title:** if conditional

- Tests incoming text for a specific string and executes the related exec command(s) if that text string is received

Syntax: **if** <"text">[:**command(s)**]

The **if** command would be used in groups of two or more, to check for various related incoming text strings. This makes it possible for the auto-exec to be able respond differently to variable host transmissions. For instance, you may wish to respond differently when you are greeted with "You have mail waiting" instead of "No mail today".

The group of 'if conditionals' constitutes the 'if conditional structure', which is terminated with the **endif** (ei) command. After each **if** command in the structure, you supply a text string for *Dialogue* to look for, followed optionally by exec commands to execute if that text string is received. If no commands follow the **if** command, then *Dialogue* will drop through the **endif** (ei) and continue executing on the exec line following the ei.

*Note: Care must be taken to choose strings that will be received, otherwise Dialogue will remain trapped executing the if conditional structure until you halt the auto-exec by pressing CBM-RUN/STOP.*

Examples:

```
if "You have mail!" :g 1 If "You have mail" is received, the g 1 is executed.
if "No mail waiting"     If this is received, the exec drops through to the
ei                             command following the endif, and quit is transmitted.
t "quit"
if "Command: "     When any one of the if conditionals is received the exec drops
if "Command> "     through to the command after the endif, and transmits read.
if "Commands: "
ei
t "read"
```

**Command:** \$            **Title:** exec temporary string

- A temporary string input from the buffer.

Syntax: depends on usage

The exec temporary string greatly extends the capabilities of an auto-exec by allowing you program it to handle only general actions, and have the specific actions stored in the buffer. The \$ is treated much like a general string variable in BASIC.

As an example, you may wish to download a large group of files. You could program the names into the auto-exec, but that would require you change the exec every time you wanted

---

to download a different set of files. Using \$, you can set up the exec for a general set of downloads, and get the specific filenames from a buffer file.

The buffer functions the same as it did with the ALT-T and ALT-R commands. ALT-R resets the pointers to beginning of the current active buffer, after which the current buffer can be changed for other purposes without affecting the \$ function. In this way, it is possible to capture incoming text in one of the buffers, and use another to transmit data for use with the \$ function. (See page 32, Transmit Single Line.)

Example:

```
ar          Reset $ buffer pointers
1 $ = B    Transfer line from the buffer to $
if $ = "end":e  If no more files, end auto-exec
t "d ->":t$   Transmit "load ", no carriage return, then $ (the buffer data)
cd$        Download the file with filename in $
g 1        Goto label 1 (to download next file)
```

The above loop would be a simple way of downloading multiple files from a PC PunterNet BBS. The filenames would need to be listed one by one in the current buffer, ending with the word END. This example also demonstrates all the syntax for \$ usage.

**Command:** if\$=      **Title:** if exec string conditional

- Tests the exec's temporary string, and executes the related exec command(s) if condition is true

**Syntax:** if \$ = <"text">[:command(s)]

Allows *Dialogue* to test the condition of the exec temporary string (see \$). If the characters of the test supplied match the beginning of \$, *Dialogue* executes the exec code following the command. Otherwise control is passed to the next line of the exec.

Examples:

```
$ = B          Get $ from buffer
t$            Transmit $
if $ = "quit":e  If $ starts with quit then end auto-exec
w "Command: "   Otherwise wait for "Command: " to be received
```

If \$ contains test12, the command if \$ = "test" will be true. No ei command is required with if \$ =.

**Command:** is      **Title:** if status equal

- Checks whether a particular *Dialogue* status is true.

**Syntax:** is <#>

Under certain circumstances it may be necessary to know the condition of the program, such as whether the buffer is full or not, etc. When an if status is encountered, *Dialogue* will check the condition related to the number given. If true, the exec performs the action following the is statement, otherwise the exec skips to the next line, and continues.

The following are the valid if status checks:

- 1 - buffer empty
- 2 - buffer full
- 3 - end of buffer
- 4 - carrier present



---

Example:            **gs 1**            Begin status subroutine.  
                  **1 is 3:r**        Is carrier present? If yes, return from subroutine.  
                  **e**                If no, then end exec. The stack is cleared for you.

**Command:** ;        **Title:** comment line

- Allows inclusion of comments in exec code

**Syntax:** ; [text]

When a ; (semi-colon) is encountered outside quotation marks, *Dialogue* immediately drops to the next line of auto-exec code. Nothing to the right of the ; will be executed. It can be issued on a line by itself, or following other valid commands. This provides a convenient means of commenting your auto-exec code. Comments are ignored by the program and take up no memory.

Example:            **gs 20 ; hangup modem**

**Command:** #        **Title:** change default device

- Changes default device number for subsequent disk operations

**Syntax:** #<8-15>

Example:            **#9** Sets *Dialogue* to use device 9 for subsequent disk operations.

**Command:** a        **Title:** ALT key command

- Invokes one of the ALT key commands available from terminal mode.

**Syntax:** a<ALT command character>["text"]    (a is generated by CBM-A)

Allows the auto-exec to directly control the operation of the program by simulating appropriate keypresses. When the auto-exec ALT key command is executed, one of *Dialogue*'s ALT key commands will be invoked.

Depending on the command, it may be necessary to provide some text in quotes. Usually this will be a filename, or other prompt response. This text in quotes is treated as if it had been typed in from the keyboard. A carriage return is always assumed.

If the ALT command doesn't leave terminal mode, or place a prompt on the status line, then the text in quotes is not needed, and will be ignored.

Examples:            **al "rle.ext"** Loads, and executes the RLE extension file.

**Command:** c        **Title:** CBM key command

- Invokes one of the CBM key commands available in the program.

**Syntax:** c<CBM command character>["text"]    (c is generated by CBM-C)

Allows the auto-exec to directly control the operation of the program as above, but with CBM commands instead of ALT commands.

As with the ALT functions, it may be occasionally necessary to provide some text in quotes. Usually this will be a filename, or other prompt response. This text in quotes is treated as if it was typed in from the keyboard.

If the command doesn't leave terminal mode, or place a prompt on the status line, then the text in quotes is not needed, and will be ignored.

- 
- Examples:
- `cl "message"` Loads the capture buffer, with the disk filename `message`.
  - `cs` Enters the buffer save command, and waits for typed filename.
  - `ca "list1"` Enters the auto-dial window, loads the dial list `list1` and then returns to terminal mode.

If you manually attempt to invoke a new auto-exec script while within another auto-exec, *Dialogue* will treat it equivalently to `gf "filename"`.

### Interactive Scripts

In some cases, the auto-exec may be used interactively to perform certain functions. If insufficient prompt response data is provide within a command, *Dialogue* will wait for you to enter an appropriate response, and then continue executing the script.

An example:

```
10 cl (lower case L) This simple example will activate the
      buffer load command, but since
      no filename has been provided, the exec will wait
      till a valid prompt has been entered
      before continuing.
```

---

## Appendix A

### *Dialogue* Modem Support Files

In the Commodore world, there are *many* wildly different modems available that can be used with the Commodore 128. These range from the early VIC-1600 modem which was incredibly priced for its day but very simple, up to today's 2400 baud MNP smart modems.

To allow *Dialogue* to support as many modems as possible, modem control is accomplished using individual machine language control programs which contain all the specific codes required to dial, hangup, etc.

The use of individual modem files, rather than merely packing the program with support for the most popular modems, allows *Dialogue* to support virtually every modem type, and the capability to allow even more modems to be supported in the future without needing to update the main program.

When you run the work disk creation program, it will prompt you for the filename of the modem support file you wish to use. Below is a listing of these files, and some of the modems that are compatible with each. Choose the file which lists your modem.

#### Modem Support Files

Modem support file: **1650.mod**

Compatible modems:

Commodore 1650	Sharedata
Pocket Modem 300/1200	Westridge
Total Telecommunications 64	Master Modem
MNP 1064	

**Notes:** Some of these modems, like the 1650 and Pocket modem have switches that turn the modem on, or choose between originate mode (dialing out) or answer mode (auto-answer). Make sure these switches are correctly set before using *Dialogue*. There is no way for the software to override a switch setting if they are not originally set. To dial, they should be set online and to originate mode.

The CBM-H hook control will toggle the modem on or offline. Note that this mode is software controlled, and if you reset the computer, your modem may remain 'off hook' thus leaving the phone line tied up unknowingly. Remember to shut off your modem when not using *Dialogue*.

Modem support file: **1660.mod**

Compatible modems:

Commodore 1660	Hes -II
Mitey Mo	Volks 6420

---

**Notes:** There are two versions of the 1660 modem. The early version (circa 1984) did not support carrier detect, and can not properly be used with *Dialogue*. The later version does have this feature, and works just fine.

Using the CBM-H hook control will toggle the modem on/offline.

Modem support file: **6480.mod**

Compatible modems:

Volks 6480 (not 6470)

Aprotek 12c (Not 1200c)

**Notes:** The CBM-H hook control will toggle the modem on/offline

Modem support file: **volks12.mod**

Compatible modems:

Volksmodem 12

**Notes:** This modem file will work with a Volksmodem 12 interfaced with a Volks J cable, or a Volks A-12 cable (through an RS-232 interface).

CBM-H hook control will hangup with A-12 cable. Has no effect with J cable.

*Both* DIP switches at back of unit *must be down* for *Dialogue* to operate correctly.

Modem support file: **1670.mod**

Compatible modems:

Commodore 1670 (old version - 3 DIP switches)

Aprotek 1200c (in 1670 mode)

Aprotek Minimodem (in 1670 mode)

**Notes:** The CBM-H sequence will send +++ and hangup the modem.

Modem support file: **1670n.mod**

Compatible modem: The *new* 1670 (4 DIP switches)

**Notes:** The CBM-H sequence will disconnect the modem instantly by dropping the DTR.

---

Modem support file: **hayes.mod**

Compatible modems:

All Hayes 300/1200/2400 modems	Supramodem 1200/2400
All US Robotics Password/Courier	Anchor 1200E/2400E
All GVC 1200/2400	Easydata 1200/2400
All Smarteam 1200/2400	Anchor Signalman 1200
Aprotek 1200 (in Hayes mode)	All Avatex 1200/1200hc/1200e/2400hc
Aprotek Minimodem (in Hayes mode)	

**Notes:** The CBM-H hook control will drop DTR momentarily and disconnect.

Although many Hayes compatible modems support the AT command set, not all modems support the switches the same way. Most popular Hayes compatible modems have eight switches. For *Dialogue* to work correctly, switches 1 and 6 should be up. These will provide true hardware carrier detect, and accept the true state of the DTR line. Check your modem manual for more details.

Modem support file: **mnp.mod**

Compatible modems:

ATI etc.	Practical Peripherals PM2400SA MNP/5
GVC 2400 MNP/5	SupraModem 2400 Plus

**Notes:** The CBM-H hook control will drop DTR momentarily and disconnect.

**Interfaces:** For full MNP/5, you require an RS-232 interface that supports at the CTS and RTS lines in addition to TxD, RxD, and DCD. Avoid the Peak Peripherals RS-232 interface, or the Aprotek Com-Modem Adapter (Aprotek's full RS-232 interface is fine).

MNP modems usually come factory preset to be merely regular Hayes compatible 2400 baud modems. It is necessary to turn on several modes to take advantage of all the power these modems offer.

It's impossible for us to describe specific commands, since these differ between modems. However, in general it can be accomplished using special AT commands. Here's what to look for in your modem manual:

**MNP negotiation:** MNP modems comes factory preset to not negotiate for an MNP connection (that is, connect only as regular 2400 baud modems). To take advantage of the error detection and correction offered by MNP, you must turn on the mode which allows MNP negotiation.

It's preferable to select the mode that turns on auto-negotiation, as this will allow the modem to connect properly to both MNP and non-MNP modems automatically.

The following are needed to take advantage of increased throughput offered by MNP modems (especially when using level 5).

---

**Baud rate adjust:** Traditionally the terminal communicates with your modem at the same baud rate (DCE/DTE rate) that your modem communicates with the remote modem (DCE/DCE rate). With MNP modems these two baud rates can be independent. This allows the terminal to communicate with the modem at a higher baud rate, allows for faster throughput (especially with MNP level 5).

Automatic DCE/DTE baud rate adjust should be turned off.

**Flow control:** Since the modem can receive data from the terminal at a faster rate than it gets sent over the phone line it is sometimes necessary to stop the terminal from transmitting. This is best accomplished using the RS-232 CTS line. Using the RTS line, *Dialogue* can stop the modem from transmitting to the terminal.

Turn on hardware flow control using CTS and RTS. This may be a single command, or two separate commands.

**Data compression:** Using special data compression techniques, modems with MNP level 5 capability can obtain effective throughput of up to 4800 baud, over a 2400 baud connection.

To allow for MNP level 5, turn on the data compression mode. MNP level 5 will automatically be selected if:

- (A) The terminal is set for 4800 baud *and*
- (B) The remote modem is capable of MNP level 5

**Note:** Be sure to save all your default settings in the modem's non-volatile memory so they'll be automatically installed when the modem powers up.

### **If your modem isn't listed**

Most of the modems in general use, are listed under the above modem files. If you don't see your modem listed, look for the words "Hayes compatible" on the modem box, or in the user's manual. If these words appear, use the file *hayes.mod*. If it says "1670 compatible", try the *1670.mod* file first.

Modems that directly connect to the Commodore 128 without any interface are likely to be either 1650 or 1660 compatible (300 baud) or 1670 compatible (1200 baud). Try one of those.

Some older modems, like acoustic coupled models, or the Commodore VIC modem (model 1600) don't support auto-dialing. For these 'dumb' modems, you'll have to operate all functions of the modem manually. Consult the modem's user guide for information.

If you have an unlisted modem, and find a file which works with it, please let us know! We'd like to include it in our list.

---

## Appendix B

### *Dialogue* Extension Files

Extensions are small programs which extend the capabilities of *Dialogue*. Even with a 128K computer, it's not possible to have an unlimited number of features. This is especially true when about 90K of the RAM is used by *Dialogue* for buffers.

Two types of extension files exist. There are those which take over control of *Dialogue*, and operate like a separate program, while keeping the phone line connection. These are handy for doing functions not normally associated with a terminal program without having to load another program. The other type of extension adds features to *Dialogue* internally - merging into the normal operation without a noticeable trace.

The extension programs allow new, and exciting features to be made available in the future, without having to continuously update the program.

Extension files are loaded by pressing ALT-L, or selecting **Load Extension** from the menus. Enter the filename when prompted, and the extension will be loaded and executed. If you decide to use one particular extension regularly, it can be defined as a default, and *Dialogue* will load it automatically each time the program is run (see **Set Defaults** - page 17).

On the *Dialogue* program disk, you'll find some extension files. The character editor, and translation table editor are both program extensions which disable the online activity of *Dialogue* while in use. The Dvorak utility is a module extension, and imbeds itself into the *Dialogue* code. Others will be introduced in the future.

#### Included *Dialogue* Extension Files:

<b>trans.ext</b>	A translation table editor
<b>chredit.ext</b>	Character set editor
<b>dvorak.ext</b>	Dvorak Keyboard Remap
<b>rlc.ext</b>	online RLE decoder for <i>CompuServe</i>
<b>copy.ext</b>	two drive copier

Filename: **dvorak.ext**

Use: Keyboard overlay for Dvorak layout.

This extension remaps the keyboard of the C128, and converts the standard 'qwerty' layout into the Dvorak Simplified layout.

Qwerty (so named because those are the first six letters on the standard typewriter/computer keyboard) was developed in 1872 by C.L. Sholes. The layout was introduced to reduce the jamming of early typewriter designs. Commonly used letter pairs (like Q and U) were purposely separated to actually slow down the typist, and thus reduce the chances of jamming.

In the early 1930s, a professor of statistics at the University of Washington named August Dvorak designed a new layout. His primary consideration was to improve the efficiency of speed and movement, with the goal of reducing typing errors, and fatigue.

---

Claims of the superiority of Dvorak have been a matter of some dispute. For instance, the much referred to ratio of the 'finger travel' between Qwerty to Dvorak is claimed to be as high as 16 to 1, however scientific tests have shown that the ratio is closer to 1.5 to 1. Recently, the Dvorak layout has received more press, and it may be becoming more popular.

#### DVORAK.EXT Options:

The standard ASCII Dvorak layout is provided. This extensively remaps the number keys and all the Commodore specific keys (see diagram).

All the CBM, ALT, ESC, and CTRL key command keys in *Dialogue* are converted to their Dvorak equivalent. The remapping will remain in effect until a new extension is installed.

The standard *Dialogue* QWERTY keyboard:

```
! " # $ % & ' ( ) { } `
1 2 3 4 5 6 7 8 9 0 + - | ~
Q W E R T Y U I O P @ * ^
A S D F G H J K L ; : =
Z X C V B N M , . /
```

Dvorak keyboard:

```
~ ! @ # $ % ^ & * ( ) . . .
' 1 2 3 4 5 6 7 8 9 0 . . .

' , . P Y F G C R L / .
A O E U I D H T N S - .
; Q J K X B M V Z .
```

Filename: **trans.ext**

Use: Translation Table Editor

This is a custom translation table editor for *Dialogue*. This allows you to modify *Dialogue* to handle unusual applications, such as accessing bulletin boards with strange non-standard ASCII tables (such as ATASCII used by some Atari 8-bit boards). It also offers users the capability to remap certain control code sequences to different keys on the keyboard.

The translation table consists of a file of 512 bytes. The first 256 are for conversion upon transmit, the second 256 for receive. In all emulation modes except CBM, *Dialogue* uses this



---

table when sending and receiving bytes from the modem. No translation is performed during a protocol transfer.

When a character is translated, its byte value (0-255) is used as an offset in the appropriate translation table. As an example, a (lower case A) has a value of 65 in PETASCII. In the translation table, its ASCII equivalent is located in position 65 (the beginning of transmit table + 65).

The default translation table loaded with *Dialogue* contains the codes necessary for translating to/from ASCII and PETASCII. *Dialogue* uses PETASCII for all its internal operations. The default translation table can be changed by creating a new default file as described on page 17.

Translation tables are saved to disk with a three character .trn extension.

### Translation Table Editor Options

<b>RUN/STOP</b>	Abort to terminal mode
<b>Return or Space</b>	Increment (or, with Shift, decrement) through byte values
<b>CRSR keys</b>	Move highlight bar
<b>(A)SCII toggle</b>	Toggles between ASCII->PETASCII and PETASCII->ASCII conversion.
<b>(P)age toggle</b>	Toggles the displayed page of the translation. (0-127 and 128-255).
<b>(D)ump table</b>	Transfer table from editor into <i>Dialogue</i> 's active translation table.
<b>(L)oad table</b>	Load translation table from disk.
<b>(S)ave table</b>	Save translation table from disk.

For large scale modifications of the translation table, you may want to generate the table directly, without using *Dialogue*, but for small changes, corrections, or just experimenting, this editor will come in handy.

Filename: **chredit.ext** Use: Character editor

**Chredit** is a full fledged character editor that will allow you create or edit entire new fonts for the *Dialogue* program to use, or for your own personal use in other applications.

*Note: The character editor does not automatically change the currently loaded character set. You may use the Dump option to view or use the edited character set, but the default set will return the next time you load Dialogue.*

To install your creation permanently, you must change the default definitions as set in the **Configuration** window (see **Set Defaults**, page 17).

---

## Character Editor Options:

<b>RUN/STOP</b>	Exits out of the character editor, and returns to terminal mode.
<b>HOME</b>	Places cursor at top left position in character editing window.
<b>SHIFT-CLR</b>	Places editing selection at top left character.
<b>CURSOR (bottom)</b>	Moves editing cursor around editing window.
<b>CURSOR (top)</b>	Selects character for editing.
<b>Return</b>	Toggles bit on/off.
<b>(B)uffer character</b>	Dump current character to copy buffer
<b>(C)opy character</b>	Copies character from buffer
<b>(D)ump set</b>	Dumps the displayed definitions into the 80-column screen for temporary use.
<b>(E)rase def</b>	Erases the current character definition.
<b>(L)oad defs</b>	Loads in a character definition file (filename prompt).
<b>(R)everse chr</b>	Reverses the bit pattern of the currently selected character.
<b>(S)ave defs</b>	Saves definitions in disk file (filename prompt).

All regular *Dialogue* functions are disabled while **chredit.ext** is active. Once you've exited the Character editor, it stays in memory until *Dialogue* is reset, or a new extension file is loaded. To re-activate the editor, press **ALT-L** again, and press **Return** at the filename prompt.

Filename: **rlc.ext** Use: Viewing *CompuServe* graphics online

This *Dialogue* extension module allows you to view the various **RLE** (Run Length Encoded) graphics available on *CompuServe*, while online. It also gives you the option of saving the graphic in **Doodle** format.

To install, use the **ALT-L** command, or choose **Load extension** from the menus. The filename can be entered as **rlc** or **rlc.ext** (*Dialogue* will add the **.ext** for you if it isn't present). When installed, the message **RLE enabled** will flash in the message area of the status line.

The terminal will operate normally until you view an RLE, by reading an RLE file in a forum library or choosing a section of *CompuServe* which displays an RLE file. Some places to find RLE's are:

<b>go qpics</b>	<b>go gallery</b>
<b>go cbmart</b>	<b>go bwmaps</b> (weather maps)

When *CompuServe* sends *Dialogue* the escape code to start an RLE viewing session. The screen will clear and turn black. Then you'll see the picture start building. You can abort the process by using the normal *CompuServe* interrupts (**CTRL-O** or **CTRL-P**). Then press **Return**.

Once the entire graphic is displayed, the bell will ring and *Dialogue* will then prompt for whether you want the graphic saved to disk. *Dialogue* will save in **Doodle** format, which is a popular format for Commodore bitmap graphics. Many utilities exist to view and convert **Doodle** pictures. The **CBMART** forum would be helpful in this regard.

After saving the graphic (or choosing not to) the screen will clear again and the normal terminal screen (with proper colour settings) will appear again. You're now back in normal terminal mode, where you can choose to view another RLE.

---

Filename: **copy.ext**      Use: A two drive RAMDOS-compatible file copier.

**Note:** This utility is particularly suited for moving files to and from a RAMdisk running under RAMDOS.

**Instructions:** Load the extension normally. The utility will prompt: **Source drive:**. It's expecting the device number (8-30) of the drive from which you're copying. Entering no device number is a signal to abort the copier and return to terminal mode. The next prompt is: **Destinaton drive:**. This is the device number of the drive to which you're copying. You must enter a device number and it must be different from the source drive number.

**Directory pattern:** This is a pattern matching string; as you would enter it if you were doing a directory:

test*	- all files beginning with test
test.?	- all files beginning with test and having a single character extension
test	- only the file test, if present

<Return>      - entering no pattern implies match all files (equivalent to \*)

**Selecting files:** After the directory pattern has been selected, you then begin the file selection process. The destination disk should be inserted in the destination drive before continuing.

The status line displays the query **Copy file?**. Each file that fits the selected directory pattern will be displayed one at a time, and you then choose Yes or No for whether or not to copy that particuplar file.

**ESC:** You've finished selecting files, begin copying. Copying will also commence once all filenames matching the directory pattern have been exhausted.

**RUN/STOP:** Abort the whole process and return to the **Source drive** prompt.

---

## Appendix C

### Control Codes

The following table lists the various control codes available from within terminal mode. To generate one, press and hold the Control key and then press one of the keys listed.

The usage of each one will vary greatly from system to system. Consult the instructions for the online system to determine what control codes are applicable.

Key Code	Value Name	Meaning (ASCII definition)
@	@	0 NUL Transmit a 'null' character
A	A	1 SOH Start of header
B	B	2 STX Start of text
C	C	3 ETX End of text
D	D	4 EOT End of transmission
E	E	5 ENQ Enquiry
F	F	6 ACK Acknowledge
G	G	7 BEL Bell
H	H	8 BS Backspace (Delete)
I	I	9 HT Horizontal tab
J	J	10 LF Line feed
K	K	11 VT Vertical tab
L	L	12 FF Form feed
M	M	13 CR Carriage return
N	N	14 SO Shift out
O	O	15 SI Shift in
P	P	16 DLE Data link escape
Q	Q	17 DC1 Device Control 1 (xon)
R	R	18 DC2 Device Control 2
S	S	19 DC3 Device Control 3 (xoff)
T	T	20 DC4 Device Control 4
U	U	21 NAK Negative Acknowledge
V	V	22 SYN Synchronous idle
W	W	23 ETB End of transmission block
X	X	24 CAN Cancel
Y	Y	25 EM End of medium
Z	Z	26 SUB Substitute
[	[	27 ESC Escape
£	\	28 FS File separator
]	]	29 GS Group separator
␣	^	30 RS Record separator
␣	_	31 US Unit separator



