TIC (Talk is Cheap)

3.1

By
Donald R. Elton

TIC (Talk is Cheap) 3.1 Manual

by

Donald R. Elton

Q Labs, Inc.
1066 Maryland • Detroit, MI 48230
(313) 331-0941
Technical Support: (313) 331-1115

SOFTWARE LICENSE AND COPYRIGHT

This software is copyrighted and all rights are reserved by Q Labs, Inc. Your use is subject to the limitations and restrictions of the copyright laws and of this license.

Permitted Uses
You MAY:

1. Use this software on a single computer.

2. Copy this software for the purpose of backup or convenient access, in the manner described in the manual, in support of your use of the software on a single computer.

Prohibited Uses
You MAY NOT:

1. Distribute, rent, sub-lease, lease, or otherwise make available to others, the software or documentation or copies thereof.

2. Use the software, or permit it to be used on more than one computer workstation at one time.

LIMITED WARRANTY

This software and documentation are sold "as is". The entire risk as to the quality and performance of the software is assumed by the user.

Q Labs, Inc. and Apple Computer, Inc. make no warranties, either expressed or implied, regarding the enclosed computer software package, its merchantability or its fitness for any particular purpose. In no event shall Q Labs, Inc. or anyone else who has been involved in the creation, production or delivery of this software be liable for any direct, incidental or consequential damages, such as, but not limited to, loss of anticipated profits, benefits, use, or data resulting from the use of this software, or arising out of any breach of any warranty. The exclusion of implied warranties or liability for incidental or consequential damages are not permitted by some states. The above exclusion may not apply to you. This warranty provides you with specific legal rights. There may be other rights that you may have which vary from state to state.

To the original purchaser only, Q Labs, Inc. warrants the magnetic diskette on which the software is recorded to be free from defects in materials and faulty workmanship for a period of five (5) years from the date the software is delivered. If during this period a defect in the diskette should occur, you may return the diskette to Q Labs, Inc. for a replacement without charge.

Table of Contents

Chapter 1

Introduction

Why TIC?

Why TIC? The MS-DOS (IBM) world has several full featured terminal programs that are well supported and that cost much less than Apple programs with less than half the capability. The existing Apple communications programs are typically high priced (over $100) and lack powerful scripting capabilities that have become the standard in the IBM and Macintosh arenas.

TIC "Talk is Cheap™ " is an attempt to remedy the above situation. The philosophy of TIC is that you shouldn't have to pay $100 for a communications package. TIC is well suited for users that want:

- The most powerful scripting capability available for an Apple II series computer where the scripting is done in plain-English.
- A program that fits into a single load file.
- A program that can emulate a dozen terminals with room for more emulations that the user can define as needed.
- A program with a no nonsense command structure where almost all features are available at a single keypress without having to wade through cute menus or mouse interfaces.
- A program that follows the rules for better compatibility with new hardware and system software.
- A program designed to be run from program selectors such as Finder, ECP, ProSEL or EasyDrive with full support for launching specific TIC scripts directly from the program selector.
- A program with a full complement of file transfer protocols including ascii capture and several binary protocols.
- A program where the bugs that users report actually get fixed and where enhancements suggested by users are frequently implemented in updates.

- A program with documentation that actually explains how Apple file transfers work so you can download programs and actually get them to run.

Updates

From time to time, new releases of TIC will be made available. Most of these updates will fix problems users have found in the software and others will add new features, usually suggested by users. Software updates may or may not require updates to the documentation depending on the reason for the update. When updates are available that are significant, we'll make it known on all of the major information services and for major updates mailings will be sent to registered users.

It is possible that an update to TIC has been released since this version of the documentation was printed. You will know that this is the case if the version number on your disk label is higher than the version number noted on this documentation. If this is the case with your software, there will be a READ.ME text file on your disk that describes any changes to the version of software you have that have been made since this manual was printed. Note that some version number changes only mean that a bug or two was fixed and that no features were changed. In these cases there may not be any mention of the change in the READ.ME file.

Important Notices

Versions of Talk is Cheap prior to version 3.00 were distributed as shareware. This means that people were free to give away copies to friends and associates with the understanding that the recipients would then send in their registration fees to receive legal copies of the software along with the complete documentation. Starting with version 3.00 of TIC, however, TIC became a commercial product and will no longer be distributed as shareware. This means that you cannot legally give copies of TIC to anyone unless you destroy all copies of the program that you own and completely transfer the program's ownership to another person. You are specifically prohibited from uploading TIC or its associated files and/or utilities to any bulletin board or information service or network for any reason. You may not use TIC on more than a single computer at a time without obtaining a license from Carolina System Software granting you permission to do so. You may not sell copies of TIC to anyone for any price and you may not bundle TIC in with other products or services without written permission from Carolina System Software and Q Labs.

ProDOS System Disk

APPLE COMPUTER, INC. AND THE AUTHOR MAKE NO WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING THE ENCLOSED COMPUTER SOFTWARE PACKAGE, ITS MERCHANTABILITY OR ITS FITNESS FOR ANY PARTICULAR PURPOSE. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME STATES. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY PROVIDES YOU WITH SPECIFIC LEGAL RIGHTS. THERE MAY BE OTHER RIGHTS THAT YOU MAY HAVE WHICH VARY FROM STATE TO STATE. ProDOS 8, FILER, CONVERT, and BASIC.SYSTEM are copyrighted programs of Apple Computer, Inc. licensed to Carolina System Software to distribute for use only in combination with Talk is Cheap™. Apple Software shall not be copied onto another diskette (except for archive purposes) or into memory unless as part of the execution of Talk is Cheap™. When Talk is Cheap™ has completed execution Apple Software shall not be used by any other program.

LTIC

The buffer scrolling capability in TIC was adapted from source code licensed from Stowe Keller. Stowe has also written a handy text file viewing utility which he markets as shareware. You can get in touch with Stowe at:

Stowe E. Keller
101 Viewmont Court
Charlottesville, VA 22901

EdIt ™

The Text Editor included with Talk is Cheap (TIC) has been licensed from Northeast Micro Systems for use with TIC. It is Copyright 1989 by Bill Tudor and may not be sold, altered, or distributed to others. Any questions regarding the editor can be directed to Northeast Micro Systems at the following address:

Northeast Micro Systems
1220 Gerling Street
Schenectady, NY 12308

ShrinkIT ™

The ShrinkIT™ file utility included with Talk is Cheap (TIC) is included for no charge with the permission of its author, Andy Nicholas. He sends the following message about it:

ShrinkIT is a freeware program. This means that I don't expect anyone to pay me for it, but retain the copyright. You may (please do!) distribute this program to whomever you please, but you may not sell it without my permission. I'm not getting rich from ShrinkIT, and neither should you. If you wish to bundle ShrinkIT with a commercial product, please contact me about including it with your software. I need to insure that the latest version is included with your product and that ShrinkIT remains unaltered.

If you feel that the program *IS* worth something to you, or simply want to send me something, please note the address below:

Paper Bag Productions
c/o Andy Nicholas
Box 435, Moravian College
Bethlehem, PA 18018

Chapter 2

Hardware

Requirements

You need to have one of the following computer set-ups to use TIC.

- Apple IIc or IIc Plus.

- Enhanced Apple IIe with 128K or more RAM, 80 column display, and Super serial card (or equivalent). Note that the Enhanced IIe is an Apple IIe sold after March 1985 or one in which the 4 chip Apple IIe Enhancement kit has been installed. If your Apple IIe displays "Apple IIe" when you start the system then you have the Enhanced IIe. Otherwise you can have it installed at any Apple dealership for $70 or less.

- Apple IIGS with Super Serial Card (or equivalent) or the built-in modem port.

- TIC supports a Pascal 1.1 or BASIC firmware protocol compatible printer interface card in slot 1 (or the built-in printer port for the IIc, IIc Plus, or IIGS). Note that you can select another slot for printing by using the SET PSLOT script command described in the script language reference section.

Note that if you are using an Applied Engineering Serial Pro card some versions of this card were shipped with two of the switches mislabeled. The two switches are the ones that control NMI and IRQ interrupts. TIC requires that the real NMI switch be in the OFF position and the real IRQ switch be in the ON position. If you have trouble getting TIC to work with this card, try reversing the setting of these two switches.

If you're using TIC with an EPIC Classic 2400 baud modem, (Epic Technologies), you'll need to change the DCD jumper such that carrier is always true.

Apple IIe

TIC requires a Super Serial Card compatible modem interface or internal modem compatible with the Serial Card to work with the Apple IIe. You can place the card in any slot other than the auxiliary memory slot. Note that TIC requires at least 128K of memory and that the IIe enhancement kit is strongly recommended. A

printer may be used with either a serial or parallel card placed in slot 1 (or the slot you specify using the SET PSLOT command in a script such as your TIC.STARTUP script.) If you have a choice, you should use a Pascal 1.1 firmware protocol compatible printer card for best results.

Apple IIc & IIc Plus

The Apple IIc & IIc Plus are ready for use with TIC with no special actions required by the user other than attaching the modem to the modem port. TIC will work with any printer you attach to the built-in printer port.

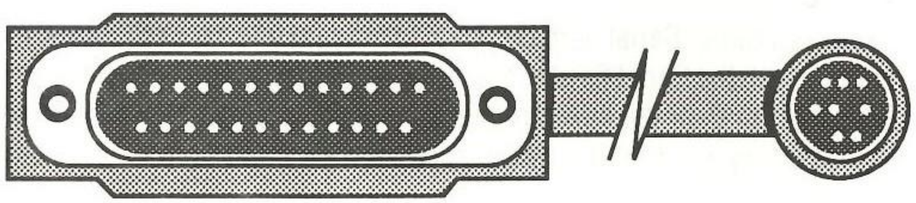
Apple IIGS

If you will be using the IIGS with a Super Serial card or compatible internal modem, you should enter the Control panel and select SLOTS. From there select "YOUR CARD" for slot 2 and proceed with the set-up instructions for the Apple IIe.

If you will be using the IIGS with the built-in serial port, then enter the Control panel and select SLOTS. From there select "MODEM PORT" for slot 2. If you use the printer port for your modem then you'll have to change the control panel settings of your IIgs accordingly. TIC will configure the other options automatically. If you have trouble getting TIC to use the IIgs built-in modem port, you should try changing either the DCD handshake setting or the DSR/DTR handshake setting. If this doesn't fix you up, then you may need to verify that you have the proper cable for the IIGS port. There has been a lot of confusion over what cable should be used with the IIGS, and many dealers have sold incorrect cables to unsuspecting customers. You may also connect a serial or parallel printer to slot 1 (or the slot you specify using the SET PSLOT command in a script such as your TIC.STARTUP script.).

Modem Cable for External Modems

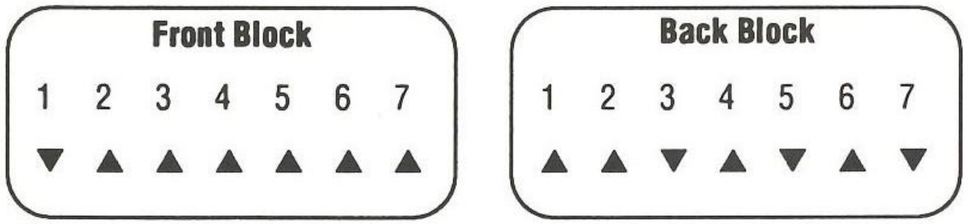
Most cables labeled for use with external modems on the IIGS or IIc or IIc Plus should work fine with TIC. Unfortunately there are lots of cables on the market that are either mislabeled or malconstructed. Because of this I'm including the following diagram of a cable design that is known to work with most modems and software (including TIC).



DB-25		MINI DIN 8	
2	<----------	< 3	TX-
3	>----------	> 5	RX-
6	>----------	> 2	DSR (handshake in)
20	<----------	< 1	DTR (handshake out)
8	>----------	> 7	DCD
7	----------	> 4	SIG GND
7	----------	> 8	RX+

- Ground RX+ to prevent noise pick-up
- RX+ is not needed for RS-232.
- 6 TX+ no connect.

Super Serial Card Switch Settings



Memory Cards

If you use an expanded memory card on the IIe or IIc that is compatible with the Applied Engineering RamWorks card then you will need to use the supplied SlotChanger software to set your ram disk memory up to use a slot/drive combination other than slot 3 drive 2 as TIC will assume that any slot 3 drive 2 based ram disk is the standard 64K ram disk set up by ProDOS and will disconnect the device, thus depriving you of your larger ram disk. Most users choose slot 3 drive 1 for these larger ram disks and TIC will work well with this setting.

Chapter 3

Getting Started

Now that you have TIC it is understandable that you will want to start using it as soon as possible. This chapter will explain how to set TIC up and get it running quickly so that you can try it out. Following chapters will give you the details about the many powerful features in TIC that will help you to save online time and communicate in a wide variety of circumstances.

Before doing anything else it is a wise idea to make a copy of your TIC disk. You may do this with the Apple File Utilities or any of the popular copy programs. TIC is not copy protected. After making the copy, put the original away and use the duplicate only.

TIC comes pre-configured so that most people will not have to change the configuration before logging onto their favorite communications services. This configuration includes:

1. Serial Port in slot 2
2. 8 data bits, No parity, 1 Stop Bit format
3. TTY terminal type

If your computer does not have an internal modem or serial port for a modem in slot 2, or you need a different data format or terminal type you will have to perform the following configuration process. Otherwise you may skip the following steps and continue with "Setting Up a Dialing Directory". If you don't know what the data format or terminal type are you probably will be able to skip ahead also.

Configuring TIC

1. Insert the TIC diskette and boot on it.

2. A menu will appear shortly. Select item 1 to "Run TIC" and press the RETURN key.

3. At the TIC copyright screen press RETURN, then press OPEN-APPLE-M for "Manual Settings". A new menu will appear.

4. If you need to change the Modem Slot press the '7' key. Type the number of the slot that you wish to use and press the RETURN key.

5. If you need to change the data format press the '8' key. Repeatedly pressing the 8 key will toggle through all of the possible combinations of data formats available.

6. If you need to change the terminal emulation press the '9' key. You will then be expected to type in the pathname to the emulation file and press RETURN when you are done. These are kept in the TERMCAPS subdirectory on the original TIC diskette, so you may type in something like "TERMCAPS/VT100". You must be sure that the emulation type you are selecting exists on the TIC disk. You can check the terminal types supported by TIC by getting a listing of the files in the TERMCAPS subdirectory.

Your TIC program will now be ready for use on your machine. You can now build a dialing directory of BBS numbers you will call most often.

1. If you have not already started TIC then do so (as in steps 1 and 2 above).

2. Press OPTION-D for the dialing directory. Note: This dialing directory is just a specially written macro included for your convenience.

3. A menu will appear. Select '3' for "Add a Service".

4. You will be prompted to enter a name. Type in a name that describes the service (ie. Genie, CIS, etc.)

5. You will be prompted to enter a number. Type in the phone number that you wish to have dialed. Spaces and dashes are not necessary, but can be included for readability.

6. You will be prompted to enter a baud rate. This should be the maximum baud rate of your modem or the telecom service you will be calling. (i.e. 2400, 1200)

7. You will be prompted to enter a login script name. Leave this blank for now and press the RETURN key.

8. You will be prompted to enter another name. Leave this prompt blank and press the RETURN key to return to the dialing directory menu.

9. Press '1' for "Dial a Service".

10. A list of available services will be displayed. Type the number of the one you wish to dial.

The dialing directory will make it easy for you to call your favorite telecommunications services. However, many other powerful features in TIC will help you to automate much of your online time. Be sure to read the rest of the manual in order to get the most out of TIC.

Chapter 4

Running TIC

Running the Program

TIC is now distributed with a program selector, The Extended Command Processor 8 (from Carolina System Software). ECP8 will startup with a menu batch file that will allow you to run TIC or any of several utilities. TIC can run without the program selector (ECP8.SYSTEM) and can be made the boot system program by deleting ECP8.SYSTEM and renaming TIC as TIC.SYSTEM.

TIC also runs directly under ProDOS and is a "system program." This means that you can run TIC from Basic, from autostart (booting), from another system program, or from a program selector. If you boot the distribution disk, TIC will be run automatically. If you are at the Applesoft Basic prompt "]" you can run TIC with the following command:

```
]-TIC
```

Note that if you get a NO BUFFERS AVAILABLE message, TIC is too large to be launched directly by your version of BASIC. If this happens, then you will need to type the BYE command and launch TIC from your program selector or the ProDOS default QUIT launcher that you may get when you type BYE if there's no program selector installed.

You can run the Editor included with TIC as a stand alone program as well by using:

```
]-TIC.EDITOR
```

How to Run TIC From a Program Selector

Program selectors vary in the way they allow you to start programs. From the Extended Command Processor (tm) From Carolina System Software you would just type the name of the program at the ":" prompt:

```
:TIC
```

With program selectors that allow it, you can specify a startup file with TIC. This

would be the name of the script you want TIC to execute when it is loaded. The default is that TIC will attempt to load and execute the file TIC.STARTUP but you may specify any TIC command file. For example:

```
:TIC COMPUSERVE
```

would execute TIC and make it execute the script named "COMPUSERVE".

When using a mouse-based program selector such as Catalyst™, MouseDesk™ or the Apple IIGS Finder, you would double-click the icon representing TIC. See the documentation for the appropriate program selector if you have trouble.

Some program selectors ask you to specify a startup prefix for the program you want to launch. You may set this prefix to whatever directory you want TIC to use at launch time. TIC does not depend on this prefix to find its own files.

The Editor included with TIC can also be run directly from a program selector and you can optionally specify the name of a file to edit as in:

```
:TIC.EDITOR filename
```

How to Set-Up an Autostart Disk

TIC comes distributed on an autostart disk. The only files required to create an autostart disk with TIC are PRODOS and TIC.SYSTEM. Note that if there are other files ending with ".SYSTEM," TIC.SYSTEM must be the first such file listed in the disk directory.

Saving Space on Your Disk

Note that the TIC distribution disk is almost completely full of files. If you are operating off of a 5.25 inch disk you will want to create a disk that has some room for downloaded files and such so you need to know what's essential to run TIC. Here's a list of the files you need to have to run TIC:

PRODOS	Needed if you want to be able to boot the disk
TIC.SYSTEM	Named TIC on the distribution disk
TIC.EDITOR	Needed if you want to use the Text Editor
TIC.MACROS	Needed to link scripts to macro keys
TIC.STARTUP	Needed if you want an auto-start script
TERMCAPS/	Only needed if you need emulations.

Chapter 5

Basic Operations

Basic Program Operation

TIC is, for the most part, a modeless program. This means that there is no special mode to get lost in while operating the program because there are no modes. All commands are typed with Apple keys. These are modifier keys like the shift or control key but they don't represent any character. With TIC, every key you type on the keyboard is transmitted out the serial port and every character received via the serial port is displayed except for a few display disrupting characters. To enter a command to TIC you must press either the ♁ key or the ♁ key. Note that the Apple IIgs substitutes the OPTION key for the ♁ key. The OPTION key has the same effect as the ♁ key. Pressing either the ♁ or the ♁ (OPTION) key with another character designates that character as a command key for TIC to invoke a particular defined program function.

In general the ♁ keys are commands while the ♁ keys execute scripts. The notation ♁—? means to press and hold the Open Apple key and then press "?" key. This particular function will display the help screen of available commands. Once in the help screen you can enter any of the commands that are displayed.

Standard Communications

Before you can communicate using TIC, you must set the baud rate in TIC to the proper speed for use with your modem. Once you have done this, you will be ready to initiate a connection with another computer.

You will need to dial a remote computer with your modem in originate mode (this is the standard mode for most modems). Once the connection is established everything you type will be transmitted and everything received will be displayed to the screen except for a few screen disrupting characters that are filtered out of the display stream.

If you are using a Hayes compatible modem, then you could enter a string at the keyboard to dial the phone and establish the connection. See your modem's manual for details, but, basically, to dial 555-1212 you would type the following at your keyboard from inside TIC:

```
ATDT 555-1212 [RETURN ]
```

"AT" gets the modem's attention, "D" is the Dial command, "T" is for tone dialing, and "555-1212" is the number to be dialed. Again, refer to your modem's documentation for more details on the features it supports. TIC is intended for use with Smart modems such as the Hayes that can respond to typed commands. Other modems may work with TIC provided that you can take manual control of them.

Once connected, there are a few other items you should know about. TIC assumes that the data format will always be 8 data bits, 1 stop bit, and no parity checking. This combination will work with the vast majority of systems regardless of what they tell you they use. If you have problems with this setting such as seeing nothing but garbage on your screen, then you may need to try another setting. These settings can be selected from the menu that appears when you type ♁—M. Select the Data Format item, and the data bits, stop bits, and parity will step through several different combinations. When you get the proper setting, you'll be able to read data from the host computer. Make a note of the setting you had to use so you can specify this setting in a script for use when you're calling this host in the future. Note that protocol (xmodem or ymodem) file transfers require by convention that the data format be 8/N/1 and that TIC will automatically select this format during file transfers using these protocols. When the file transfer terminates, TIC will automatically return you to the settings you've selected if necessary.

The other thing you will need to know is that TIC, like a standard TTY terminal, expects that the carriage return character and the line feed character represent separate functions. This means that the host should transmit a carriage return and a line feed as the new line sequence. Again, this is almost always the case without your asking for it, but some systems, such as TBBS for NEWDOS-80 and MS-DOS, will ask you if you need line feeds. If they should ask you this, you need to tell them that you do. If they don't send you line feeds then lines will all write over themselves on your screen.

Some systems will ask you if you need any nulls. A null is a zero byte transmitted to waste time. They are typically needed for people using slow printing terminals or slow scrolling display screens. Ordinarily, nulls should not be necessary since TIC uses hardware interrupts to keep up with high baud rates. You may need to use them though if you're doing online printing at a baud rate faster than your printer can keep up with.

Terminal Emulation

TIC supports screen emulation for several terminals including a standard TTY (teletype) mode for cases when terminal emulation is not needed or desired. You

select a terminal to emulate by specifying the pathname to an emulation definition file in the ♁—M manual settings menu or by using the EMULATE statement in a script (see the chapter on Script Documentation for details). While emulation is in effect, all screen output from the host, displayed from the recording buffer, or displayed from a text file, will be routed through the terminal emulator. There is a utility program supplied named DEFTERM that will allow knowledgeable users to create their own emulation definition files. There are several emulation definition files supplied in the /TIC/TERMCAPS subdirectory but these files can be kept in any directory so long as you specify the pathname to the specific emulation definition file you wish to use. Should you select TTY at the TERMINAL entry in the ♁—M menu or using the EMULATE command then no file will be loaded. The TTY emulation definition file is always available in memory.

Should the screen display become garbage perhaps due to line noise or because of a faulty emulation definition file, you can reinitialize the console driver by pressing the ♁—I (init) command at the keyboard.

Online Documentation

♁—? Use this command to display a one page help screen that will display all available commands for using TIC.

♁—? This command will display a list of defined script macro keys. See section 8 for details of how to set up script macro keys.

The Status Line Display

♁—ESC When you run TIC the first line of the display is the status line display while 23 lines of the display make up the communications window. The status line display can be turned off or turned on using this command. The status line displays the following information:

***** An asterisk in the left-most position of the status line indicates that a script is being executed. You may press the ESC key to terminate a script's execution so that TIC will again respond to commands from the keyboard.

Baud Rate 300, 1200, 2400, 4800, 9600, or 19,200 baud.

Duplex Full, half, or chat duplex.

[Z] Designates that zoom mode (control show mode) is enabled.

[P] Designates that online printing has been enabled with the ♁—O command.

[R] Designates that the recording buffer is actively recording modem input.

Buf: A percentage display of how much of the recording buffer is filled so far; range 0 - 99%. The recording buffer currently has a capacity of about 47K bytes of data.

Dir: The current ProDOS prefix directory is displayed.

Time The current ProDOS time is displayed.

Manual Settings

♁—M

This command brings up a menu of 10 items that can be set manually. Settings altered by this command will be remembered between runs of TIC as TIC will write a copy of these items into a disk file named TIC.CONFIG (so don't use this file for anything else). When TIC is run it will look for a TIC.CONFIG file and if it doesn't find one it will create one. The following options are available at the ♁—M menu:

[1] Work Directory: /TIC/

The work directory is where TIC will store any temporary files it needs to create. If you have one, you should specify a fast disk device such as a ram disk or hard disk directory for this entry. The default directory is the directory where TIC was loaded from. Note that if you're using a 128K computer that you cannot use the ProDOS/RAM disk as it is disconnected when TIC loads since TIC needs all 128K for its own use.

[2] Autosave File: TICTemp.1

This is the file name that TIC will save its recording buffer to when you use the ♁-S command or when the buffer becomes filled during a normal session. The default is to store the buffer to a file name in the root directory that begins with

TICTEMP. and ends with a number, (0 through 9). You may change this destination to any full pathname you desire. Many users find it convenient to have the recording buffer saved to a ram disk device as this is faster than storing to a physical disk drive. You must specify a full ProDOS pathname at this entry.

[3] Buffer: Manual

The two choices here are AUTO and MANUAL. The default is MANUAL which means that the recording buffer must be controlled manually with the ♁—R command. If this option is set to AUTO then the buffer will be turned on automatically when a Control-R is received from the host computer. The buffer will be turned off if a Control-T is received.

[4] Append: True

The two choices here are True and False. If True, then multiple saves of the recording buffer will go to a single large file with each buffer save appended to the end of the previously saved file. If you choose False here then each recording buffer will be saved to a different file whose name ends with a higher number than the previously saved file. You can have up to 9 recording files if Append is set to False.

[5] Debug mode: False

Debug mode is used to help you find mistakes in TIC scripts. If debug mode is set to True then script statements will be displayed in inverse video as they are executed. If set to false they will not be displayed as they execute.

[6] Printer Init: ^I80N

This is where you encode the string of characters that must be sent to your printer interface or printer to initialize it. You can specify control characters by using the "^" character. i.e. Control-I is represented as "^I" as shown above. See your printer manual if you need to change this setting. If you get double spacing or no spacing between lines then this is the item that probably needs changing.

[7] Modem Slot: 2

This is the slot where TIC will expect to find your modem. This will usually be slot 2 but it can be changed to any slot not being used by your printer.

[8] Data Format: 8N1

This is where you can set the number of data bits (7 or 8), the parity (None, Even, or Odd), and the number of stop bits (1 or 2). 99% of the time you'll want to leave this set to 8N1 (for 8 data bits, No parity, and 1 stop bit) but a few systems will require a different setting.

[9] Terminal: TTY
This is where you'll be able to select a terminal emulation table to be loaded by TIC. The pathname can be a complete pathname or the path needed to load the file from the root directory of TIC. Use TTY for no emulation. Note that emulation tables are stored in the /TIC/TERMCAPS subdirectory on your distribution disk.

[0] Time Constant: 10
This number is used by TIC to standardize the timing of the PAUSE and TIMER features of the script language and to standardize the timing of the file transfer protocols. Select a higher number if you have a speed-up card or speed-up chip (see the table on the next page).

SPEED UP DEVICE	TIME CONSTANT
Standard Apple IIe	10
Standard Apple IIc	10
Standard Apple IIc Plus	10
Standard Apple IIGS	10
Zip Chip 4 mhz	35
Accelerator IIe	35
TransWarp IIe	35
TransWarp GS 7 mhz	28

The above settings for speed-up cards and chips are only starting points. You may need to make adjustments for your particular hardware to make sure the timing statements in scripts take the correct amount of time to execute.

Once you've made changes using ⍁—M you can exit this screen with ESC or RETURN. If you made any changes then you will be asked if you want the changes to be permanent. If so, TIC will write a new TIC.CONFIG in the root directory so that your changes will remain in effect the next time you run TIC.

Set the Baud Rate
⍁—B
This command will flip TIC through all of its supported baud rates. Each time you press the ⍁—B keys the baud rate will be rotated in order from 300 —> 1200 —> 2400 —> 4800 —> 9600 —> 19,200 —> 300 etc. The status line is updated immediately to reflect the new baud rate.

Change Duplex Setting
⍁—E
This command toggles TIC from full to half duplex, chat duplex and back again. The status line is updated immediately to reflect the new duplex status. Chat duplex means that half duplex will be used and that incoming carriage returns will have a line feed automatically added so you can see what you're typing if you're calling another personal computer user. To use Chat duplex in talking to another personal computer user, the caller would dial the recipient as if he was dialing any other system. Both systems would need to be set at the same baud rate. The recipient would type the ATA command to tell his modem to answer the phone once the phone started ringing. From there, each user can type to the other or transfer files by mutual agreement.

Exiting the Program
⍁—Q
This command is used to return control back to ProDOS or to a calling program selector such as the Extended Command Processor or Finder if it is installed. There are three options presented when you use the ⍁—Q command:

Q Quit with port disabled - this option disables the communications port prior to leaving TIC. This will result in a dropped carrier with some modems if you are online when you issue this command. It will also prevent extraneous interrupts from resulting in later program crashes.

E Quit with port enabled - this option leaves the communications port intact when leaving TIC. This is the choice you would use if you wanted to leave TIC temporarily while still online with the intent of returning to TIC to finish up your session.

ESC Return to TIC - this option allows you to change your mind about leaving TIC and is useful if you accidentally pressed the ⍁—Q command when you really didn't want to leave TIC at all.

Once you exit TIC, follow the normal procedure for your program selector. If no program selector is installed then ProDOS will prompt for a new prefix directory and the name of a system program to run. You can use ⍁—Q to leave TIC for another utility and then return to TIC to continue communications. TIC will automatically save the contents of the recording buffer to the current autosave file if you quit with a non-empty buffer.

Print the Current Screen

Apple—P

This command will print the contents of the current screen to the device in slot 1 (or the slot specified by a SET PSLOT command from a script), usually a printer. Data that arrives over the serial port during the printing operation (up to 256 bytes) will be displayed on the screen after printing is completed.

Toggle Online Printing

Apple—O

This command will turn on or turn off online printing. In this mode, any text displayed to the screen either in terminal mode or as a result of viewing a text file will also be sent to the printer as it is displayed to the screen. A "P" on the status line indicates that online printing has been enabled.

Zoom and Show Control Characters

Apple—Z

This command toggles between Zoom and non-Zoom mode. Zoom mode means that control characters (other than carriage returns and line feeds) will be displayed as inverse video letters representing the particular control character received. i.e. the Bell character, ascii $07 will be displayed as an inverse video "G" for CONTROL-G. This feature is useful for determining exactly what data is being sent by a host computer during debugging operations with scripts.

Hang Up the Phone

Apple—H

This command will prompt you to confirm that you wish to disconnect the telephone connection and will hang up the phone if you press the Y key to confirm.

Chapter 6

Filing Commands

These commands are used to manipulate ProDOS in basic operations while your are using TIC. The descriptions that follow assume that you have at least a casual understanding of how ProDOS works with directories, files, and subdirectories. If you need more info on these areas then try the ProDOS users guides before proceeding.

Set a New Prefix

Apple—N

This command allows you to change the current prefix as displayed in the status line display at the top of the screen. A window will display on the screen showing you the current prefix. If you wish to change the current prefix then type in the name of the directory you wish to set it to and hit return. Just hitting return by itself will accept the current prefix as is.

Display a Disk Directory

Apple—D

This command will display a window showing the current prefix directory and will prompt for the name of the directory you wish to display. Pressing return will cause the prefix directory contents to be displayed on the screen, pausing after each screen-full waiting for a keypress. After the entire directory has been displayed you may press any key to restore the original communications window. You may type in the full or partial pathname of the directory you wish to display if it is different from the prefix directory.

Delete a File

Apple—K

This command prompts for the name of a file to delete. It will not delete a locked file (that's what lock is for!).

Volumes Online Display

⌘—V

This command will display a listing of the slot and drive assignments for all mounted ProDOS disk volumes including any installed ram disks or hard disks. This command is useful if you forget the volume names of mounted volumes since other commands require that pathnames be used instead of the slot and drive assignments.

Display a File

⌘—J

This command will prompt for a partial or full pathname to a text file you would like displayed to the screen. The file is displayed one screen-full at a time. Pressing any key causes the display to resume. The display will pause at the end of the file and pressing any key will restore the original communications window.

Chapter 7

File Transfer Operations

One of the nice things about modem communication is the ability it gives you to transfer files of data or programs across telephone lines. Unfortunately, just as human speech may be misunderstood by conversing humans, so can computer communications be faulty. When transferring text data, a data error caused by line noise may be easily recognized and cause no trouble but this is almost never the case when the file contains programs or data where a single byte error will cause the file to be worthless. To avoid problems like these, communications protocols were developed so communications programs like TIC would be able to detect transmission errors as they occur so they can direct the sending computer to re-send segments of the file that contained errors during the first transmission. One of the powerful features of TIC is that it supports several such file transfer protocols such that you can choose one best suited to the individual transfer you need to accomplish and with a variety of host systems that may not support but one or two protocols.

Choosing a Transfer Protocol

TIC supports many different file transfer protocols. They are ASCII text, Xmodem, Xmodem crc, 1K Xmodem (incorrectly called Ymodem in previous versions of TIC), 4K Xmodem, and ASCII Express (™USII) ProDOS protocols. TIC supports Ymodem batch mode downloads and supports downloading and uploading of Binary II formatted files with automatic extraction of member files and subdirectories. A special limited utility protocol called Turbo Xmodem is also available for downloads.

When to Use ASCII Text Transfers

ASCII text transfers can only be done when the file you want to send is made up of readable characters also known as 7 bit files. There is no error correction facility built into ASCII transfers but some systems can support no other protocol. If you want to upload a pre-prepared mail message to a bulletin board service then you would use a text editor or word processor to prepare your text, log onto the information service or bulletin board, and use TIC's ASCII transfer capability to upload the file into the editor of the information service just as if you had typed it. This has the advantage of letting you prepare the text on your own

time using whatever special formatting and spelling checking your text editor supports without tying up the phone line while creating the content of your message. Note that ASCII transfers work best where there is a fairly noise free telephone connection as there is no mechanism to detect or correct errors. When uploading a program source file or other long or important document, you should use one of the protocol transfers when you have that option as they can transmit or receive the file error-free. You can use the text downloading feature of TIC (also known as the recording buffer) to log or archive your online session for later review. You might also use this to download all of your mail at the start of a session at a high baud rate, then log off and read the mail on your own time.

TIC can also use the text transfer feature to upload AppleWorks word processing files directly. These files are stored as ProDOS type AWP. TIC will translate these files into plain readable text during the upload and will respect the formatting of the document as you typed it in AppleWorks. AppleWorks spread sheet and database files still have to be either uploaded with a protocol or must be printed to disk as a text file from AppleWorks before they can be uploaded as text by TIC.

When to Use Protocol Transfers

Use protocol transfers any time they are available as they are the best way of getting an error free transmission. Protocol transfers are the only way that you can transfer binary or program files as ASCII text transfers can only be used for readable ASCII text files.

Which Protocol Should I Use?

TIC supports 4 non-ASCII (binary) file transfer protocols and supports combinations of the starting 4. The basic considerations left to you are whether Xmodem, 1K Xmodem, or 4K Xmodem should be used. When you tell TIC that you want to transmit a file your choices are Xmodem, 1K Xmodem, 4K Xmodem, ProDOS, or Text. Text refers to ASCII text transfers and is described elsewhere. In addition to the above named full protocols, TIC supports Ymodem batch downloads. Each of the major protocols are discussed below.

Xmodem

Xmodem is a protocol where 128 byte blocks of data are transmitted from one computer to another with a single byte checksum that is calculated based on the values of the 128 data bytes. The receiving computer calculates its own checksum and compares what it calculates to the checksum sent by the other computer. If the two values match than it can be reasonably be assumed that the

128 byte data blocks were valid so the receiver tells the sender to send the next data block. If the checksums don't match then the receiver directs the sender to re-send the block. This continues until either the entire file is transmitted or the transfer aborts after 10 consecutive errors or a user abort from pressing CONTROL-X a few times at the other end of the transfer or the local user pressing the ESC key. The benefit of xmodem is that small data blocks are used so transfers can be successful even with relatively noisy lines since the odds are good that you can get 128 bytes to transfer error-free more times than not without too many repeated blocks. The bad effect is that there is a waiting period between every 128 bytes of the file for the handshake to occur between the sender and the receiver, a waiting period that prolongs the transfer process somewhat.

1K Xmodem

1K Xmodem is identical to Xmodem except that 1,024 byte blocks are used instead of 128 byte blocks. This works better on cleaner data lines since there are fewer handshaking pauses during the transfer. At 1200 baud, 1K Xmodem is as much as 10% faster than Xmodem for this reason. Of course, if errors do occur, then an entire 1,024 bytes must be retransmitted even if the error only involved one byte of data. Note that many hosts refer to 1K Xmodem (or Xmodem 1K) and Ymodem interchangebly. Even TIC, in previous versions, called its 1K Xmodem Ymodem. By the strict definitions, Ymodem means batch Ymodem while 1K Xmodem has the same 1024 byte blocks as Ymodem but has no batch capability.

4K Xmodem

4K Xmodem is similar to Xmodem and 1K Xmodem except 4,096 byte blocks are used. Again, cleaner data lines are needed to make larger block sizes efficient. 4K Xmodem is a new file transfer protocol currently supported by Talk is Cheap, DataTerm (™ AE), and MouseTalk (™ USII). A driver is available for the GBBS SuperTac file transfer module and drivers are being written for other BBS programs at the time of this writing. 4K Xmodem is also known as Xmodem 4K or 4K Xmodem.

ProDOS

This refers to ASCII Express (™ USII) protocol transfers. This is basically an extension to Xmodem where ProDOS directory information such as the exact length of a file, the ProDOS file type, the create date, the modification date, and the program load address are automatically transmitted along with the file. This is useful any time you're transferring a file from one Apple to another as it avoids

the problem of how this important file information should be transferred to the other system. This particular ProDOS protocol extension is compatible with Ascii Express: The Professional (™USII), ASCII Express MouseTalk (™USII), ProLine Message System (™ Morgan Davis Group), ModemWorks (™ Morgan Davis Group), and several other ProDOS based bulletin boards and message systems. It is not supported by any non-Apple system but the protocol allows for automatic detection and support for this protocol at the other end of a transfer. This means that you can select ProDOS transfers on a non-ProDOS host and the protocol in TIC is smart enough to determine whether the other end of the transmission can support the full protocol. If not, then TIC will drop into either Xmodem, 1K Xmodem, Ymodem or 4K Xmodem depending on the capabilities of the host.

Binary II

While ProDOS protocol works well with Apple systems, it is not supported by any information services or bulletin boards that do not run on Apple II's. On these systems many files will be found in what is known as Binary II format. In this format, files are stored with a 128 byte header that contains their ProDOS directory information. When you tell TIC you want to receive a file it will ask if you want it to automatically extract individual files from these Binary II format files when it detects that the incoming file is in this format. If you choose YES then TIC will automatically extract the member files from the Binary II file as it is downloading to your disk. If you choose NO then any incoming Binary II files will just be saved to disk in their Binary II format, just as they were on the host system. These Binary II files would then have to be unpacked by a separate program such as the Binary Library Utility (BLU) or ShrinkIT. Note that some Binary II files, prepared by the BLU program, will contained files that have been squeezed or compressed. Squeezing files makes them take up less space so you can save time during downloads. TIC can remove the members of these Binary II files during downloads but unsqueezing will have to be done with a separate program, either ShrinkIT or USQ. If you know a Binary II file contains squeezed members then it would probably be more convenient to disable TIC's automatic Binary II extraction process and just use the ShrinkIT program to both extract, and unsqueeze as a single step.

On protocol uploads, TIC asks whether you want Binary II headers added to the uploaded file. You should normally answer yes to this question unless perhaps, you know that the software receiving your upload supports ProDOS mode file transfers. TIC will only add the Binary II header to a file that doesn't already have the header so it's OK to answer yes to header-adding even if you're not sure whether the file already has a Binary II header.

Turbo Xmodem

Turbo Xmodem is a special limited use form of Xmodem for use with downloading. When you download a file from a network such as Compuserve, or GEnie the majority of protocol download time is spent between blocks waiting for verification that the previously transferred block is valid. This delay can be as long as a second or more between each block transferred. Turbo Xmodem pre-acknowledges blocks during downloads so the host will go ahead and start sending the next block. This can as much as double the speed with which a file is transferred. On the down side, however, if any errors existed in the block that was already acknowledged as good then the entire transfer fails and must be repeated. For this reason, Turbo Xmodem should only be used when you have a very clean data connection (no line noise). Additionally, because data is arriving almost continuously during a Turbo Xmodem download, the protocol can only be used when you are downloading to a ram disk or other interruptible device such as some hard disks or 3.5 inch floppy disks. Turbo Xmodem will definitely NOT work when downloading to a 5.25 disk drive since interrupts are turned off for long periods of time during disk writes to this type of device. As long as you understand these very significant limitations then Turbo Xmodem can be quite useful in selected circumstances. You should test Turbo Xmodem with your phone connections by first trying it on a small file. Most users have clean enough connections to use Turbo Xmodem if their connections are made via a local call.

CRC Checking

In the description of Xmodem, mention was made of checksum checking. Another error checking protocol is known as CRC for Cyclic Redundancy Check. This is a two byte value that is a more accurate means to detecting errors in transmitted blocks. TIC will use the CRC brand of error checking any time it can determine that the other end of the transfer supports it. You may, when given the choice, tell the host that you would prefer CRC checking as opposed to checksum checking and TIC will act accordingly with more accuracy.

Automatic Protocol Selection

TIC has several proprietary enhancements that allow it to automatically detect when another system supports a particular transfer protocol and will adjust itself automatically to support 1K Xmodem, ProDOS, or CRC checking protocols. When TIC is used to receive a file from a host, TIC sends a special code to the host that essentially says: "I can support 1K Xmodem or 4K Xmodem with CRC checking.. use it at your end if you can and I will reciprocate". TIC also will look out for this special code when it is sending a file to a host so it can respond accordingly. At the same time, when TIC transmits a file via the ProDOS option

it sends a code that translates to "I can support AE protocol, can you?". If so then TIC will use ProDOS protocol, if not it reverts to Xmodem, or 1K Xmodem or CRC checking if the receiver requests it. While this sounds complicated, the complication was in writing TIC, not in using it. All other things being equal, use ProDOS transfers when talking to an Apple based message system, otherwise use 1K Xmodem transfers. If there is a problem with line noise on your phone line then you might try Xmodem mode to force the smaller data blocks.

Transmitting Apple Files

As was alluded to in the description of ProDOS protocol file transfers above, there is more important information about ProDOS files than just what's included in the file. ProDOS maintains this important information in the file directory. If you can use the ProDOS protocol to transfer a file then everything is taken care of. Displaying the directory where you put a downloaded file will look the same on your computer as it did on the host computer right down to the date fields. If the file must be transmitted to a system that doesn't support ProDOS protocol transfers then you must choose a method of including that directory information. If the file you are transmitting or receiving is a text file then the information may not be important and you can just transmit the file. If the file is a binary or program file then your options are as follows:

1. Convert the file to a text file that will create the original file when Exec'ed from ProDOS BASIC. A utility to do this is available as freeware on Compuserve and elsewhere. It was written by Glen Bredon and is called "The Executioner".

2. Place the file into a Binary II file using the BLU utility included on the TIC distribution disk. The Apple ProDOS directory information will be included inside the Binary II file.

3. Place the file into a ShrinkIT archive using the ShrinkIT utility included with TIC. Note that some hosts require that you take advantage of ShrinkIT's ability to include a Binary II header on the file.

4. Let TIC automatically add a Binary II header to the file during the upload.

File Naming Conventions

Since Apple files uploaded to non-Apple hosts such as BBS systems or online information services can be stored in a variety of formats, people have devised file naming conventions to help give a clue to potential file downloaders as to the format the file is in. Below are some examples of how files might look in a typical download directory:

TIC.BNY This is a Binary II (BNY) file, use TIC's automatic Binary II extraction (Auto-BNY) capability or ShrinkIT to extract the members from this file. A binary transfer protocol such as Xmodem must be used to download this file.

TIC.BQY This is a Binary II file that contains squeezed members. Disable TIC's Auto-BNY extraction feature and use BLU to extract, and unsqueeze the members from this file. A binary file transfer protocol such as Xmodem must be used to download this file.

TIC.EXE This is a text EXEC file that will create TIC on your disk when you EXEC the file from Applesoft ProDOS BASIC. BASIC and machine language programs can be transferred in this format. This file can be downloaded via ASCII capture or via a binary transfer protocol such as Xmodem. Not all users own programs that support binary transfer protocols the way TIC does, so you will see some programs in text format to help these unfortunate people out. Text EXEC files can be created by Glen Bredon's utility The Executioner.

LOAN.FP This is another way of designating a text EXEC file.

INFO.QQ These designate squeezed files. Use ShrinkIT or USQ to unsqueeze these files. A binary protocol such as xmodem must be used to download these files.

TIC.PP
TIC.PBH
TIC.DDD This refers to a file that is an entire packed disk. There are several disk packing utilities available in the public domain but most are incompatible with each other and require that you have a supply of blank disks available. They're useful mostly for DOS 3.3 based programs.

FILE.SHK Denotes a file compressed and archived with the new ShrinkIT utility.

FILE.ACU This is a file archived using the ACU utility which is similar to BLU but used only by the America Online (formerly AppleLink Personal Edition) service. Unpack with ShrinkIT or ACU.

FILE.BXY This is a file packed with ShrinkIT like the SHK file above but this one also has a Binary II header added. TIC can remove the Binary II header during a download but Shrinkit will still be needed to extract the files from the resulting SHK file or ShrinkIT could be used both to remove the Binary II header and to unpack the SHK file in a single operation.

Text File Reception

Text file reception is accomplished via the TIC recording buffer. The recording buffer is a 47,000 byte area of memory where incoming data is stored. When this buffer is filled, the contents are automatically saved to disk in the current prefix directory in a file named TICTEMP.1. Subsequent buffer images are saved in TICTEMP.2, TICTEMP.3 and so on. You can specify another file name to use for autosave operations via the ♁—M (manual settings) command or with a script statement (SET AUTOSAVE). See section 8 for details. The following commands control the recording buffer:

♁—R This command toggles the recording buffer on and off. When the buffer is turned on the letter "R" is displayed near the center of the status line display. You should use this command to start TIC's recording of data received by your computer.

♁—S This command saves the current contents of the recording buffer into the appropriate TICTEMP.X file on your disk. The ♁—S command also clears the recording buffer after the buffer image has been saved to disk. Note that this function occurs automatically should the recording buffer fill up.

♁—W This command is like the ♁—S command in that it allows you to write the contents of the recording buffer to disk. Unlike ♁—S, however, the ♁—W allows you to specify the pathname of the file that will be used when the file is written to disk.

♁—C This command will erase the contents of the recording buffer.

♁—Y If you have the TIC.EDITOR file in the same directory TIC is run from, then this command will allow you to run the text editor. You are asked to press "B" to edit the recording buffer contents.

Note that the percentage of the recording buffer currently filled is displayed near the center of the status line display. During a typical session you would tell a host which file you wanted it to send you. You would tell the host that you desired an ASCII transfer. The host would tell you to press RETURN to start the transfer. You would then use the ♁—C to clear the buffer, then ♁—R to turn on the recording buffer. You then press RETURN to tell the host to send the file. Once the file has been received you would press ♁—R to turn off the recording buffer and then ♁—S to save the current buffer to disk.

Automatic Buffer Control

TIC also supports automatic or host control of the recording buffer. If you use a script command (SET BUFFER AUTO/MANUAL) or the ♁—M (manual settings) command to set the buffer mode to automatic then TIC will automatically turn on the recording buffer if a Control-R character is received from the host. A Control-T character will automatically turn the buffer back off again. This is useful on systems that send these codes to facilitate text downloading. This is helpful as it allows you to get a fairly clean copy of the downloaded file without the usual leading or trailing garbage characters that would normally have to be edited from a file. Unfortunately, sometimes line noise will trigger your recording buffer when you don't want it to so the default is for buffer mode to be manual.

Text File Transmission

♁—T

A text file is a file that contains words you can read. Text files are usually stored with the ProDOS file types TXT or SRC (for APW/ORCA source code files). TIC can also convert AWP files produced by the AppleWorks word processor into text during an upload as a convenience. This allows you to use the AppleWorks word processor to directly prepare files or messages for later uploading without having to first convert them into text files.

To transmit a text file to a host computer follow the steps below:

1. Select the File transfer dialogue box by pressing ♁—T.

2. When prompted to "Send or Receive" a file select "S" for "Send".

3. Enter the full or partial pathname of the text file you wish to send when prompted with "Sending File:". Optionally, you may enter the "/" character to specify that you want to upload the contents of the recording buffer to the host. Note that using this feature, you could record something from the host, edit it, perhaps adding your own comments, and then upload the buffer contents again to the host.

4. Select "Text transfer" by pressing the "T" key. If you specified the "/" character as the file name then TIC will know that you want a Text transfer since TIC doesn't support any other type of upload from the recording buffer.

5. Press the key that represents the "Prompt" character or press RETURN if no prompt is to be used. The prompt character is the character that TIC will wait for as a signal that the host is ready to accept the next line of data. This keeps TIC from sending the file too fast for the host to accept. A typical prompt might be the ":", or the ">" character. If you're uploading to a text editor on the host that does not display any prompt then you might try using the line feed (Down-Arrow key) as the prompt since the host will transmit a line feed prior to accepting each new line of text. Control characters such as the line feed key will be displayed as inverse video characters. When a prompt character has been specified, TIC will wait up to 20 seconds after sending a line for the prompt character to arrive and will transmit the next line anyway if the prompt does not arrive before this time limit expires. You can use the SET PTIMER command to vary the length of the wait for a prompt from 1 to 255 seconds or you can specify a value of 0 (zero) which will cause TIC to wait forever for a prompt to arrive.

6. The "Character delay" is an arbitrary time delay added between the sending of each character. Some host computers can only keep up with human typing and not with a fast program such as TIC so adding a delay between characters will help TIC slow down to near human speeds. Pressing RETURN or entering a "0" (zero) will mean no character delay while entering "9" would be the maximum inter-character delay.

7. The "Line delay" is an arbitrary time delay added between the sending of lines. This allows a host time to save each line of text uploaded prior to accepting the next line of text from TIC. Line delays are often needed when uploading text to a system that cannot send the sender a prompt character.

8. You are then asked if you want a space character added to blank lines that you upload. The default is YES. Some systems' message editors take a blank line to mean that you have finished uploading a message or file. If the file you are uploading contains blank lines then your transfer could be prematurely aborted. To avoid this, TIC adds a space character to lines that are otherwise blank. You can disable this feature by answering NO to this prompt. You can also change the default value using the Set PadCR ON/OFF command (described in the script language documentation).

The file specified will then be sent to the host until either the end of the file is reached or the user presses the ESC key to abort the transfer. Note that the file transfer window is not displayed while the file is being transferred. The display you see is that generated by the host computer.

Protocol File Reception

⌘—T

To receive a file via a file transfer protocol you should select the file transfer dialogue box with the ⌘—T command and specify "R" to receive a file. You then enter the name of the file you want to save the incoming data to and press RETURN. You then select whether you want TIC to automatically enable Binary II file downloading. You will usually want to say Yes to this prompt. You are next asked if you want to use Turbo Xmodem mode (note the discussion on page 36). The default is to not use Turbo Xmodem. The rest of the process is automated as TIC determines which file transfer protocol is being used by the host computer and acts accordingly. You must tell the host computer which protocol to use in sending you the file prior to initiating the transfer with TIC.

Protocol File Transmission

⌘—T

To send a file via a file transfer protocol you should select the file transfer dialogue box with the ⌘—T command and specify "S" to send a file. You should then enter the name of the file you want to transmit and select a protocol by letter. Note that you must first tell the host computer that you want to make a protocol transfer prior to initiating the transfer with TIC.

Chapter 8

Utilities

Text Editor

TIC includes a text editor stored in the file TIC.EDITOR. This editor can be used both as the editor for TIC or as a stand alone text editor from any ProDOS program selector. I licensed this editor from Bill Tudor so please don't give this editor away without Bill's permission. The editor is an external program for TIC that knows how to interact with TIC and TIC knows how to interact with the editor. Note that this text editor is very easy to use and its operations in most cases are a subset of AppleWorks™ functions. From within the editor you can press ♁–? to display a help screen containing all of the active commands. You can also use a Mouse, if you have one, to position the cursor or to select commands. From within the editor you use the ESC key to activate the menu bar and then use either the Mouse or the arrow keys to select commands you want to use. Most commands just use the ♁ key much the way TIC and AppleWorks do. When you use the ♁—Y command in TIC you have 4 options that determine how the editor is called:

B	Edit the buffer contents
F	Edit a file
RETURN	Edit a blank buffer
ESC	Return to TIC

If you choose "B" then TIC will write out the buffer contents to a temporary file in the Work directory specified by the ♁—M command. It will then load the text editor and tell the editor to edit the TIC.BUFFER File just written out. When you exit the editor, TIC will be restarted and will automatically reload the TIC.BUFFER file back into the recording buffer. Note that you will stay online during the time you are in the editor but it is possible that if you stay in the editor long enough, the host will hang up on you for lack of activity.

If you choose "F" then TIC will prompt for a file name. This file name will be passed on to the text editor after TIC autosaves the current buffer contents to disk to the current autosave file specified by the ♁—M command. As before, TIC will be reloaded once you finish editing the file specified.

If you simply press RETURN then the editor will be loaded with a blank buffer. The contents of the current buffer will be saved to the autosave file as above prior to loading the editor. If you add anything you can save it to disk from the editor by giving the editor's buffer a filename from within the editor.

If you decide that you didn't mean to call up the editor at all then you can press ESC to abort the effects of the ♁—Y command.

ShrinkIT

ShrinkIT is an utility program for archiving disks and files. With easy to use menus and powerful NuFX™ archiving features, anyone can use ShrinkIT in a matter of minutes. Dynamic LZW is used to achieve optimum compression in a relatively short period of time.

ShrinkIT can add files to archives, archive entire disks, mix disks with files, selectively extract the contents of archives, list the contents of archives, allow up to 60,000 files to be placed in an archive, and zero the unused blocks on disks so disks compress more efficiently. ShrinkIT also provides disk utility functions such as deleting files, cataloging, creating subdirectories, formatting and erasing disks, copying files, and typing the contents of text, AppleWorks and WordPerfect files.

ShrinkIT is COMPATIBLE with the past. ShrinkIT can extract and unSQueeze the contents of Binary II files (.BNY files), NuFX archives, ACU archives, .BQY files (Binary II files with SQueezed records), and .BXY files (a Binary II "wrapped" NuFX archive). It does this transparently. You never have to know the type of the file — just tell ShrinkIT to (U)nShrink and it will do the rest!

All menu selections in ShrinkIT are done by using the arrow keys to highlight a choice then pressing RETURN, or by typing special letters on the keyboard. Sometimes you must type the ♁ along with another key at the same time.

Many of the utilities use a standard file selection menu. When this menu is displayed you will see on the left of the screen the list of available files from your disk. The 'folder' character displayed next to a filename means that the file is a subdirectory. To scroll through the names use the up/down arrow keys. Above the list of file names is the current directory (prefix) that you are looking at.

Squeeze and Unsqueeze Utilities

In any computer system, efficient management of the file storage space is

important. The two programs SQ and USQ reduce the size of data files by using the Huffman Data Compression Algorithm.
The SQ program uses the Huffman coding technique to search for the frequency of use of each of the 256 possible byte patterns, and it then assigns a translation for each character to a bit string. All of these bit strings are placed end to end and written onto a disk file. The encoding information is also put on the file since the USQ program needs to know the character distribution of the original file.

The USQ program reads in the encoding information and then reads in the encoded file. It is a simple matter to scan the encoded file and produce an output file which is identical to the file that SQ started with.

General Hints:

1. Files must be above a threshold size, or else the output file will be longer than the input file because of the decoding information put at the beginning of the compressed data. We don't know the exact size of the threshold because the encoding binary tree information depends on the distribution of the characters in a file. At least we know to check the size of the encoded file after we run SQ to make sure our file didn't grow.

2. Some files will not compress well if they have a uniform distribution of byte values, for example, binary or system files. This is because of the way SQ builds the tree. Remember that bytes with the same frequency of occurrence are at the same depth (usually) in the tree. So if all of the bytes have the same depth, the output strings are all the same length.

3. SQ reads the input file twice. If you can, use a RAM disk at least for the input file and for both files if you have the room. The next best case is to use two floppy drives, one for input and one for output. This will cause a lot of disk starts and stops but not much head movement. Worst case is to use one floppy drive for both input and output. This will cause a lot of head movement as the programs alternate between the input and output files.

4. SQ and USQ are stand-alone ProDOS system programs. They can be run directly, or from BASIC. Each comes up with a title screen and a prompt asking for a source and destination file. At this prompt you should type the name of the file to be squeezed or unsqueezed, a comma, and the name of the file to be created. Note that these two names have to be different.

5. When you squeeze a file, the original file's name is stored with the squeezed file. If you want that name used when you unsqueeze the file then don't enter

a destination file. This will work fine if the file was squeezed with SQ and will probably work well with files squeezed under other operating systems. Note, however, that some operating systems such as CP/M and MS-DOS have different file naming rules from ProDOS. This may result in file names that contain characters that are illegal under ProDOS which would result in an error when the USQ program tries to create such a file. If this occurs then you must specify the destination file name rather than using the default name stored with the squeezed file. When you squeeze a file, you should name the destination file in such a way that other users will have a hint that this is a squeezed file that requires USQ to unsqueeze. The most common way of doing this is to add the letter "Q" to the end of the file. By convention, try to use the following:

FOO.LBR	→	FOO.LQR
FOO.DOC	→	FOO.DQC
FOO.PAS	→	FOO.PQS
FOO.TXT	→	FOO.TQT
MYFILE	→	MYFILE.QQ
MYFILE	→	MYFILE.SQ

These naming conventions are by no means mandatory but will be almost universally understood under other operating systems and hopefully under ProDOS as well as SQ and USQ become more widespread in use.

DefTerm

The DefTerm program is a simple Applesoft program that must be run from BASIC.SYSTEM (supplied with ProDOS 8). It allows you to create or modify terminal emulation definition files for use with TIC. With TIC's terminal emulation, incoming bytes of data are either translated to other bytes or they are translated into functions such as the Clear Screen function. You can also suppress an incoming data byte by specifying that it should be translated into a zero byte also known as a null. To create a terminal emulation definition file you would need a listing of the features that the desired terminal supports and what code bytes are used to bring about those functions. TIC can be used to emulate most single or double coded terminals. The current version of TIC cannot be used to emulate the DEC VT-100/200 series of terminals but will emulate the DEC VT-52 and about a dozen other terminals as shipped. Using the DefTerm program requires that you have at least a working knowledge of how terminal emulation works and is not for everybody. If you can't get DefTerm to work for you and you have documentation for a terminal that TIC can't currently emulate then send it to the address at the front of this manual and I'll see if it can't be added to a future version of TIC.

Chapter 9

About EdIt

EdIt is designed as a small but very functional text editor for the enhanced Apple IIe, IIc,IIc+, and IIGS computers. It is quite easy to learn how to use, and optionally supports the Apple Mouse™, Apple Extended keyboard, and all ProDOS™-compatible clocks. EdIt has been specially designed to work in conjunction with the Talk is Cheap™ communications package, although it functions equally well on its own. When called from TIC, EdIt will exit back to TIC rather than simply quitting to ProDOS.

Getting Started

To use EdIt, simply choose the ⍁-Y option (edit file) from TIC. Note that the editor can also be run by executing the TIC.EDITOR file directly. Everything that EdIt needs is contained in the TIC.EDITOR file. If the editor is launched from TIC, it will look in the directory that it was launched from for the file 'TIC.CONFIG'. If found, the printer slot and setup string are extracted from this file. Be sure to place the editor in the same directory as the TIC (and TIC.CONFIG) files. If the TIC.CONFIG file is not found (or the editor is run as a stand alone application), then a default printer slot and setup string is used. This can be changed when you select the print option in the editor.

Editing a File

From the edit screen, EdIt will display the cursor where text will be entered. The current column and row position of the cursor is displayed on the bottom line of the screen, along with some instructions. The top few lines of the screen contain EdIt's menus. The date and time will also be displayed on the menu bar, provided you have a clock installed in the system.

EdIt will allow you to enter text anywhere within the current document. However, if the cursor is positioned beyond the end of the current document, you will be unable to enter any text (simply move the cursor). A blinking underline ("_") cursor means that you are in insert mode, and text will be inserted into the document at the current cursor position. A blinking inverse block indicates you are in overstrike mode, and any text entered will overwrite the existing text. You can use the commands listed below to edit the document:

Moving around

Arrows	move cursor
⍁-Left arrow	Move left one word
⍁-Right arrow	Move right one word
⍁-Up arrow	Move up one page
⍁-Down arrow	Move down one page
⍁->	Move to end of current line
⍁-<	Move to beginning of current line
⍁-1	Move to beginning of document
⍁-9	Move to end of document
Tab	Move right to next tab stop
⍁-Tab	Move left to next tab stop

Deleting Text

⍁-Delete	Begin block delete process
⍁-F	Delete character to right of cursor
Delete	Delete character to left of cursor
⍁-X	Delete entire current line
Clear	Delete entire current line
⍁-Y	Clear from cursor to end of current line

Other Commands

Esc	Access menus
⍁-T	Set/remove new tab stops
Control-S	Search for a string
⍁-D	Directory of current prefix
⍁-N	Set new prefix
⍁-L	Load a file
⍁-S	QUICK save a file (no prompting)
⍁-A	EdIt! credit screen
⍁-M	Clear file in memory
⍁-?	Show help screen
⍁-P	Print the file
⍁-Q	Quit EdIt!
⍁-E	Toggle insert/overstrike modes
⍁-V	Display all volumes online (in a drive)
⍁-C	Copy to/from clipboard
⍁-Z	Toggle show/hide carriage returns

NOTE: A mouse movement will translate to an arrow keypress and a mouse button press will allow access to the menus. The PageUp, PageDown, Home, End, Help and Del keys on the extended keyboard will operate as expected.

Accessing the Menus

To access the top menus, press the ESCape key (or click on the mouse). The first menu (entitled "File") will be "pulled down". Now use the up/down arrow keys (or the mouse) to select the option you wish to use. Press the RETURN key to make the selection. To move over to the next menu, use the right/left arrow keys.

NOTE: Most menu commands can be performed by using an equivalent apple-keypress directly from edit mode.

Loading a File

To load in a file from the disk for editing, press ♁-L or select "Load File" from the "File" menu. If the current file in memory has not yet been saved, you will be warned and given the opportunity to save it. While loading, you will be presented with a dialogue box asking you to enter the pathname of the file you wish to edit. Note that the current prefix is displayed in the box. To change the prefix, enter oa-N for New prefix. Enter the filename in the edit box, or, as an alternative, you can press ♁-L (or click the mouse) to List files on the disk. From the list you can use the up/down arrow keys to highlight your choice, then press RETURN to select the file to load in. There will be a short pause as the file is loaded into memory.

NOTE: Using the ♁-L "List Files" option will clear any file currently in memory. The editor will warn you of this IF the file in memory has not yet been saved, or has been changed since it was loaded into memory.

NOTE: EdIt has a maximum capacity of about 1,111 lines of text. If a file is not able to fit into memory, you will be warned. As much of the file as possible will be loaded. You must be careful NOT to save the file back to the same filename, or the original file will be overwritten.

EdIt allows you to load ANY file type (except DIR), however, it can only properly edit a TEXT file. These include both TXT, SRC, and others. If a non-text file is loaded (such as AWP), the screen will look a bit messy, and the file will not Quick Save (see below) over the old version. BE SURE NOT to save the file on top of the version on the disk. The formatting, etc. of the file will be lost. Only valid text characters are loaded into memory.

Saving a File

To QUICK save a file, press ♁-S from edit mode. This causes the file to be saved under the same name from which it was loaded, overwriting the old file contents. If EdIt does not know what name to call the file (e.g., as in the case of an entirely

new file), you will be prompted to enter a filename. Enter the name in the edit box and press RETURN to save the file.

NOTE: If the file loaded was too large to fit into memory, EdIt will NOT overwrite the old contents of the file, but rather it will ask you for a filename under which to save this new and shorter file.

Another way to save a file is by selecting the "Save File.." option from the file menu. This method, unlike the ♁-S command, will ALWAYS ask you for a filename, and always ask you if you wish to replace the old file's contents if a file of the same name already exists. It will, however, provide a default filename for you in the edit box. In addition, the "Save file.." option gives you the opportunity to save the file with a carriage return character at the end of each line. This is done by using ♁-RETURN rather than just RETURN from the save box. The carriage returns are permanently added to the file. This is useful for uploading messages to online systems such as GEnie.

Printing a File

EdIt was primarily designed as a text file editor, but it does have printing capability. To print a file, select "Print File.." from the file menu or use ♁-P in edit mode. You will be asked for the slot number of your printer, left margin to use for printing, and for the setup initialization string to send to your printer. When entering this string, use the "^" character to signify "control", e.g., for "control-I 80N" you would use "^I80N".

Utilities

The utilities menu contains three options, directory, new prefix and volumes. Selecting Directory will catalog the current prefix directory, and selecting New prefix will bring up an edit box asking you to type in the new prefix. Note that both of these commands can be accessed from edit mode by pressing ♁-D and oa-N respectively. The "volumes" option will display the names of all disks currently online (in a drive).

Other Commands

To display the current version number of the editor, select ♁-A from edit mode or the "About.." option from the file menu.

The ♁-T command allows you to set the tab stops. Use the left/right arrows to position the cursor where you wish to set/remove a tab stop, then press 'T' to set/

remove a stop. Press RETURN or ESCape when you are finished setting the tabs.

To obtain a list of available commands, press ♁-? from edit mode. The screen will clear and a box showing all the available commands will be displayed.

To deleted an entire block of text, press ♁-Delete. The current line will be highlighted. Use the up/down arrow keys to highlight the block of text you wish to delete, then press RETURN. The ESCape key aborts the block delete mode.

You can search for a string in you document by pressing Control-S, then entering the string to search for. The editor will perform a case-insensitive search for the string starting from the current cursor position.

Options

The options menu allows you to customize EdIt a bit. You can changed the cursor blink rate to a value from 1-9, with 1 being the fastest and 9 the slowest blink rate. You may wish to do this, for example, if you have an accelerator card installed in your computer.

There is also an option that allows you to turn the mouse on and off (if you have a mouse installed).

The line length of the document can be changed from the options menu. In order to change the line length, there must NOT be a file in memory. Use oa-M to clear memory, or, if you have an important file there, save the file to disk, then clear memory with ♁-M, set the new line length, and finally load the file back into memory.

Macros

Another option is macro keys. EdIt now has limited macro capability. Nine macros of up to 78 characters each may be defined and saved. The macros are evoked by pressing ♁-1 through ♁-9. If you have an Apple Extended Keyboard, function keys F1-F9 will also call the macros 1-9 (no need to hold down the ♁ key). Note that the ♁ key is also the 'Option' key on the IIGS. Macros are defined by selecting 'Edit & Save Macros' from the Options menu. The current set of 9 macros will be displayed on the screen, numbered 1-9. Type the number of the macro you wish to define. When defining macros, any keypress will be placed into the macro. A ♁ keypress will have an apple character displayed just below the keypress. The Option key (♁) allows you to control the editing process. Option-Delete deletes characters to the left of the cursor, Option-Return accepts

the line as it is, and Option-Esc aborts the macro editing and restores the former macro.
When you are finished editing all the macros you wish to define, use ESC to return to edit mode WITHOUT saving the new macro set, RETURN to return to edit mode with the new macro set in place, and [S] to save the macros to disk. If you do not save the macros to disk, they will not be available the next time the program is run. Note that the disk containing the TIC.EDITOR file must be in a drive to save the macros.

NOTE: You CAN rename the TIC.EDITOR file if you like. The program will still be able to save the macros to disk, provided you launched the program with a program selector that follows the proper guidelines (Basic.System, ECP-8, ProSEL, FINDER, and Selector all work). Also note that if you rename the editor, TIC will not be able to launch it.

Quitting the Program

To quit the program, select "Quit" from the "File" menu, or press oa-Q from edit mode. You will then be presented will a dialogue box asking if you wish to exit without saving the file in memory, or save changes and quit. Press E to exit without a save.

Pressing 'Q' will cause the current file to be save IF it has been changed since the last save.

Notices

The software in this package has been tested thoroughly, however, Northeast Micro Systems and/or the author shall not be liable for any direct, indirect, incidental, or consequential damages resulting from the use of this diskette, manual, or software beyond the purchase price of the product.

Chapter 10

Script Documentation

TIC supports a very powerful scripting capability. A script is a list of commands for TIC to execute automatically. These scripts, if properly named, are automatically executed when you press the ♦ key (labeled the OPTION key on the Apple IIgs) together with a letter key or they may be specified manually using the ♦—X command described above. If the user presses the ♦—A combination then TIC will look for a file named "TIC.KEY.A" to execute as a script. Scripts that are executed as keyboard Macros are called Macro Scripts. In order for TIC to find its scripts, TIC keeps up with the concept of a Root directory. The Root directory is the directory that TIC resides in when it is run. All Macro Scripts must reside in the Root directory.

At startup time, if a file named "TIC.STARTUP" is found in the Root directory then TIC will execute it as a script automatically. If you are using a program selector such as the Extended Command Processor (ECP) then you may specify an alternate startup script at run-time by specifying the alternate file name after the TIC file name:

```
:tic <pathname>
```

The above would start TIC and force TIC to execute <pathname> as the startup script.

In order to invoke Macro scripts, you must have a text file named TIC.MACROS in TIC's Root directory. Each line of this file should begin with the letter of a valid Macro key. i.e. if you start a line in TIC.MACROS with "A" then you should have a file named "TIC.KEY.A" in the same Root directory with TIC. The remainder of the line should start immediately after the letter identifier and should be the name of the service associated with the particular Macro key. An example TIC.MACROS file might contain:

```
aApple BBS
bRadio Shack BBS
cMy BBS
fYour BBS
```


Most script operations can be aborted by pressing the ESC key. There may be a short delay in aborting certain script functions.

How Scripts Work

TIC scripts are simple text files. They are read into memory one line at a time. Before any parsing is done, variables are expanded if any are found in the line just read. Then the line is parsed for commands and the command found on the line is executed. Once execution of the line has completed, the next line is read in from disk and the process is repeated. More than one command can be placed on the same line by separating the commands with semi-colons:

```
set baud 2400;set duplex full;set buffer on
```

In some cases, commands will alter the flow of execution. A typical case is the GOTO command, that is used to transfer control to a user defined label within the script. When a GOTO command is encountered, the label is read into memory. TIC then starts reading the script file from the beginning, examining each line one at a time, until the line containing the label is found. If no such line is found then an error message is displayed and the script terminates. If the label is found, then execution continues as before with the command that follows the label. Because of the sequential nature of the search for a label, you will note that performance of scripts is improved if frequently called DO procedures or GOTO labels are found near the beginning of the file. Another side effect of the disk based nature of TIC scripts is the fact that some disk device drivers, most notably, 5.25 inch floppy disk drivers, turn off interrupts while the disk device is being accessed. This may result in loss of incoming modem data if a disk access needs to be made while data is coming in over the modem. ProDOS actually accesses a disk every 512 characters even though TIC is reading the lines one at a time. It is during these block reads by ProDOS that you may find you lose data if you are running a script from a 5.25 inch floppy disk. If you run into a problem with this, then consider running the script from a ram disk, hard disk, or 3.5 inch disk, or perhaps rearrange your script commands so a disk access doesn't occur at a critical part of your program.

Script Variables

TIC scripts can use two types of variables. The first type are User Defined Variables. There are eight of these numbered 1-8. User defined variables have no meaning when a script is started unless it is started by the CHAIN command (described below). The second type of variables are the Pre-Defined Variables. These can not be altered by the user but have a value set by TIC itself.

User Defined Variables

User defined variables can contain any string value. Values are assigned by the ASSIGN, GETLINE, and READFILE commands. Each of these three commands accepts a variable number between 1 and 8 inclusive to designate which variable gets the data. These variables are used in a script by coding them as "$" followed by the variable number. For example, the following script segment:

```
assign "Don " 1
assign "Elton" 2
display "$1$2 was here^M^J"
```

when executed would display the string: "Don Elton was here^M^J". Remember, that the variables are expanded before the line is even parsed. This is very important as it means that a variable can even contain a script command.

For example:

```
assign "display ^G" 1
$1
$1
```

would be converted to:

```
assign "display ^G" 1
display ^G
display ^G
```

before it was executed and would ring the bell twice when executed. This gives TIC scripts the ability to alter their commands by using variables to define the commands and their arguments.

Note that variables are expanded literally in place. Variables can be placed into a script line back-to-back with no blanks or separators between them as the variables are expanded before the line is parsed for commands and arguments.

Pre-Defined Variables

TIC has 8 pre-defined variables. These are defined as follows:

Variable	Value	Description
$date	19 Nov 89	current date
$emulation	TTY	current terminal
$file	TIC.KEY.A	name of the running script
$prefix	/a/files/	ProDOS Prefix Directory
$root	/a/tic/	Root Directory
$time	19:40	current time in 24 hour format
$version	TIC 3.10	software name and version
$work	/RAM6/	Work Directory

Working with Variables

TIC has several variants of IF commands to help you work with strings and variables. They are:

```
IF CONTAINS
IF EQUAL
IF NOTEQUAL
IF NULL
```

These are all documented in the script language reference section. Here's an example of how they might be used:

```
# wait for Friday
if equal $date "16 Mar 89" goto Friday
goto wait

# Friday
display "^G It's Friday!^M^J"
stop
```

Another example below shows you how to validate a user's input to a menu choice:

```
# menu
Display "^L"  clears screen
Display "G Call GEnie^M^J"
Display "C Call Compuserve^M^J"
Display "P Call ProLine^M^J"
Display "^J^M"
Display "Choose: "
GetLine 1
If Contains "GCP" "$1" Goto Good
Display "^GBad choice!^M^J"
Goto Menu

# good
Display "Valid choice!^M^J"
Stop
```

Script File I/O

TIC scripts also have the ability to work with text files, including other TIC scripts! These operations are accomplished by the Open, OpenA, ReadFile, WriteFile, and Close commands. A TIC script can work with up to two other files at once. To use a file, it must first be opened and it must be closed after you have finished working with the file. Note that when a script terminates, that all active user files are closed for you but it's better programming practice to close them yourself using the Close command. You can work with any number of user files but only two may be open at any given time. These user files are designated as file 0 and file 1. You assign a file number to a file when you open it. Open opens a file that exists or creates a file that doesn't exist (see syntax description below). OpenA works like Open except it also positions the read/write mark to the end of the file. You will usually use Open when you want to read or write at the start of a file and will use OpenA when you wish to write to the end of a file. ReadFile and WriteFile do what you think they'd do. You can read lines of text from a file into variables with ReadFile and you can write text to files using WriteFile. Note that variables and constants in any combination can be part of a WriteFile command and note that WriteFile doesn't write a carriage return unless you explicitly code one into the string using ^M.

Here's an example TIC Script that I use that updates a log file in TIC's Root directory every time TIC is executed. To make sure this script gets run each time TIC is run, I put this segment into my TIC.STARTUP file. Assume that I launch TIC from a subdirectory on my hard disk "/a/tic/".


```
opena 0 "$roottic.log"
writefile 0 "$date $time TIC booted^M"
close 0
```

If I run the above file as I write this, then the script, at run time, will look like:

```
opena 0 "/a/tic/tic.log"
writefile 0 "20 Nov 89 15:59 TIC booted^M"
close 0
```

and the line:

```
20 Nov 89 15:59 TIC booted
```

gets written to the /a/tic/tic.log file when I execute the script.

Script Syntax Reference

TIC scripts are text files created by any text editor. They consist of lines of text with one or more commands per line. A line may be as long as 127 characters. If lines are longer than this then subsequent characters are treated as if they were on the following line. This situation would normally result in an error. Blank lines are ignored as are leading and trailing spaces which may be included in the file to improve readability. Upper and lower case characters are allowed and will not affect parsing. TIC does not check the file type of scripts.

This is a Comment

This is the comment character. It is a command that is ignored. One or more blank characters must follow the # symbol. Labels are defined as the first word following the # symbol. Thus to encode the label 'start' in a script you would use the following:

```
# start <- this is a label.. "Start"
```

Assign "string" <Variable Number>

Assigns the value "string" to the variable designated by the number. The "string" may contain imbedded control characters or you may encode them by prefixing a letter key with the "^" symbol. Thus, "^C" is Control-C. Use "^^" to encode a single "^" character literally. Note that the characters displayed are sent to the CRT using the communications console driver so carriage returns and line feeds

must be encoded separately as "^M^J".

Example:

```
Assign "One" 1
Assign "Two" 2
goto $1

# One
goto $2

# this never gets executed

# Two
stop
```

The above script segment shows how you can use the ASSIGN command to assign a label name to a variable that is then used as the argument for another command, the GOTO command in this example.

Another example:

```
Assign "Don " 1
Assign "Elton" 2
Assign "$1$2" 3
Display $3
stop
```

In the above segment, $1 takes on the value "Don " while $2 takes on the value "Elton". The third command binds $1 and $2 together and stores the result in $3 such that the final DISPLAY command displays the string "Don Elton".

Break
Sends the BREAK (or Attention) signal to the host if your modem hardware supports this function.

Example:

```
break
```

The above script sends the BREAK signal (called Attention on some systems) to

the host computer. Note that some systems use the Control-C character for this function but the BREAK signal doesn't really correspond to any particular character but is a special modem signal.

Buffer On, Buffer Off
Turns the recording buffer on or off as indicated.

Example:

```
xmit "Read Mail^M"
Buffer Clear
Buffer On
waitfor string "Command: "
if failed goto abort

Buffer Off
Buffer Write MailFile
stop

# abort
display "^J^MScript aborted...^J^M^G"
hangup
stop
```

The above script segment sends a command to a host computer to tell it to send mail messages to TIC. The Buffer On command records the incoming data. The script pauses as data arrives waiting for the signal that the previous command has been completed. Once the string is received, TIC stops recording data, saves the buffer contents in the file "MailFile", and then the script terminates. The "Abort" segment gets executed if TIC fails to receive the string before the timeout limit is reached (See Set Timer).

Buffer Clear
Erases the recording buffer. See the example script above for Buffer On and Buffer Off.

Buffer Load <pathname>
Loads the recording buffer with the file specified by <pathname>. The buffer will hold about 47,000 characters of data. This command should only be used with text files.

Buffer Save
Saves the recording buffer to the autosave file.

Example:

```
Set Autosave /ram6/myfile
Buffer Save
stop
```

The above script segment sets the autosave file to "/ram6/myfile" and then uses Buffer Save to save the contents of the buffer to that file. If you want to save the buffer contents to a file other than the autosave file then use the Buffer Write command below. Also, see the example script segment for Buffer On and Buffer Off. above.

Buffer Write <pathname>
Saves the recording buffer to <pathname>. See the example script segment above for Buffer On and Buffer Off.

Chain <pathname>
Continues execution with the script file indicated by <pathname>. Variables retain their previous values at entry. There is no automatic return to the calling program.

Example:

This is the file Script1:

```
# Script1 here
Display "^M^JScript 1 running.^J^M"
Assign "Label" 1
Chain Script2
```


This is the file Script2:

```
# Script2 here
if null 1 goto Main
goto $1

# Main
Display "^J^MScript 2 not run by Script
   1^J^M"
stop

# label
Display "^J^MScript 2 run by Script 1^J^M"
stop
```

These two script files show you how you can branch a script's execution into another file. By using the script variables ($1 in this case) you can control what happens in the second script by setting a variable in the script you are chaining from that has some meaning in the script you are chaining to. You can even set a variable in the first script that the second script uses to know what file to run next, perhaps the first script for example. In the above example, the segment in script 2 that follows the label "Main" only gets run when Script 2 gets executed with $1 having a null value. This is the situation that would exist if Script 2 was run directly by the user. If Script 1 runs Script 2 then $1 will expand to "Label" and the "Main" segment will not get executed.

Close <file number>
Closes the file specified by <file number> which is an integer between 0 and 1 inclusive. If used without a file number then both user files are closed.

Example:

```
Open 0 MyFile
ReadFile 0 1
Close 0
Display "$1^J^M"
stop
```

The above script segment opens "MyFile", reads a single line from it and stores it in variable 1 and then closes the file using the CLOSE command. There can only be two files open at once so if you need to work with more than 2 files in a script then you have to close an earlier opened file with the CLOSE command before opening a third file. Once the file has been closed in the above example, the line read from the file gets displayed by the DISPLAY command.

DEC <Variable Number>
Decrements the unsigned integer value stored in the indicated variable. The range of legal integer values is 0..65535 inclusive. If you decrement a variable containing 0 then the value will not change.

Example:

```
assign "9" 1
dec 1
display "$1"
dec 1
display "$1"
dec 1
display "$1"
stop
```

The above script when executed would display "876". You can also use this command, together with DEC and IF EQUAL to control loops, generate unique file names, or count the lines in a file and so forth. You could also prompt a user for a list of file names to be uploaded and count them as you write them to a text file for later use.

Delete <pathname>
Deletes <pathname> from disk. This command will not work with a locked file.

Example:

```
# Loop
Display "^J^MDelete What file? "
getline 1
```



```
if null 1 stop
if exists $1 goto killit
Display "^J^M^GFile not found!^J^M"
goto Loop

# killit
delete $1
goto Loop
```

The above script segment prompts the user for a filename to delete and stores the response in variable 1. It then checks to see if anything was entered. If not, the script is terminated with the STOP command. Next, the script tests to see if the file exists (you can't delete a file that's not there to begin with). If the file exists, it is deleted. The script then branches back to the top and starts over again.

Display "string"
The "string" is displayed to the CRT.

Example:

```
Display "^I^IHello World^J^M"
stop
```

The above example displays two tab characters (^I) and then displays the string "Hello World" followed by a line feed (^J) and a carriage return (^M). Note that any control character besides the seldom used Control-^ character can be displayed to the CRT. Note that the Display command sends data to the standard Apple II screen firmware in ROM. The control codes available are documented below though codes that could disrupt the screen by changing it out of 80 column mode are filtered by the Display command. Note that the carriage return and line feed functions are separated in TIC's version of the console driver. This means that a carriage return (^M) just positions the cursor at the start of a line while the line feed (^J) advances the cursor down one position. You normally will want to send both when you're writing to the screen. Normally only carriage returns need to be sent to the host computer and you use the XMIT command to do that.

Table of Control characters used by Display

^G	Bell - beeps the speaker
^H	Backspace
^I	TAB
^J	Line feed
^K	Erase screen below cursor position
^L	Erase entire screen
^M	Carriage return
^N	Normal mode
^O	Inverse mode
^Y	Position cursor at top left corner

Do <label>
Calls a subroutine that begins with the line following <label>. See also, RETURN. Note that only one DO procedure may be active at one time.

Example:

```
Display "^J^MEnter your choice: "
waitfor keyboard
if failed stop

If keyboard 1 Do One
if keyboard 2 Do Two
if keyboard 3 Do Three
stop

# One
Display "One^M^J"
Return

# Two
Display "Two^M^J"
Return

# Three
Display "Three^M^J"
Return
```


The above script segment demonstrates using the DO command to call one of three simple subroutines based on a response typed at the keyboard to a prompt written to the screen using the DISPLAY command. DO commands are handy when you want to link several commands to a single command as shown above.

Emulate <pathname>
Loads the emulation definition file specified by <pathname>. Specify "TTY" if you do not want to use any disk based emulation file.

Example:

```
set baud 2400
set statline off
set duplex full
emulate termcaps/vt52
```

The above script segment is an example of how you might start off a script that sets TIC up for working with a full screen oriented host that expects you to be using a VT52 terminal.

Exit
Shuts down TIC and exits to the program selector while leaving the communications port active. This command is used when you plan to leave TIC while online and then return back online after using another program. See also QUIT.

Example:

```
Display "Going offline..."
exit
```

The above script segment shows how to use EXIT to leave TIC without disabling the modem port. You can do something offline and then re-run TIC and continue your session with the host computer. See also QUIT.

GetLine <Variable Number>
Reads a line of input from either the keyboard, the modem, or both, and stores it in the variable indicated by <Variable Number> which can have a range of 1-8. Note that input must be completed by the time specified by the TIMER (see SET TIMER).

Example:

```
Display "^J^MEnter file to display: "
GetLine 1

If Null 1 Stop
If Exists $1 View $1
stop
```

The above script segment asks for a filename to display. If no filename is entered (i.e. the user just presses RETURN) the script terminates with the stop command. Otherwise it tests to see if the file exists and if it does it displays the file to the screen. If the file doesn't exist it just terminates the script. Note that the GETLINE command can also be used to read data from the modem. If you wrote a script to handle a host mode for TIC whereby users could call your TIC equipped Apple then you could use GETLINE commands to read input from users. In the above example a few changes could make it work for remote users:

```
Display "^J^MEnter file to display: "
Xmit "^J^MEnter file to display: "
GetLine 1
If Failed Stop
If Null 1 Stop
If Exists $1 send $1 text
stop
```

Note that since we're dealing with a remote user that we need to XMIT the prompt string rather than just display it. We still display it in case the system operator (that's you) wants to help the remote user type in the answer. The IF FAILED command becomes true if the user presses the ESCape key before or during input.

Goto <label>
Transfers control to the command following <label>.

Example:

```
# Top
goto Bottom

Display "^J^MError if you see this!^J^M"
stop

# Bottom
Display "^J^MNo error here!^J^M"
stop
```

In the above example, the GOTO command skips the first display command and execution continues with the line that follows the "Bottom" label.

GotoXY <X Position> <Y Position>
Positions the cursor at the coordinates indicated. <X Position> can range from 0-79 while <Y Position> can range from 0-23.

Example:

```
GotoXY 0,0
Display "This is Top Left"
GotoXY 35,11
Display "This is the middle"
GotoXY 50,23
Display "This is Bottom Right"
stop
```

The above script segment displays the strings of text on three different parts of the screen as indicated.

Hangup
Hangs up the telephone connection.

Example:

```
Display "Hangup the phone? "

# TryKeyboard
waitfor keyboard
if keyboard Y goto Yes
if keyboard N goto No
goto TryKeyboard

# Yes
Hangup

# No
stop
```

In the above example, the user is asked if he/she wants to hangup the phone. If the reply is "Y" for Yes then we branch to "Yes" and hangup the phone. Otherwise we terminate the script.

If Contains "longstring" "shortstring" <command>
If the string "shortstring" is contained in the string "longstring" then <command> gets executed. Otherwise, execution continues with the next command.

Example:

```
Display "^J^MEnter Password: "
getline 1
if null 1 stop
if contains "Pass1Pass2Pass3Pass4" "$1"
    goto good
display "^G^J^MInvalid Password!^J^M"
stop

# Good
<do neat stuff here>
```


In the above example script segment, the user is prompted to enter a password. If the password entered is contained in the string "Pass1Pass2Pass3Pass4" then execution continues with the meat of the script (not shown). Otherwise, the user is denied access to the script. You could also use this command together with the file I/O commands to test a user's password against a list of passwords read from a password file on disk.

If Equal "string1" "string2" <command>
If "string1" and "string2" are identical (case ignored) then execution continues with <command>. Otherwise, execution continues with the next command.

Example:

```
# Loop
Display "^J^MEnter routine to run: "
GetLine 1
if null 1 goto bad
if equal "$1" "One" goto One
if equal "$1" "Two" goto Two
if equal "$1" "Three" goto Three

# bad
Display "^G^M^JNo such routine!^J^M"
goto Loop

# One
Display "Got One!^J^M"
stop

# Two
Display "Got Two!^J^M"
stop

# Three
Display "Got Three!^J^M"
stop
```

The above script uses IF EQUAL to test a user's input to be sure it matches one of several possibilities.

If Exists <pathname> <command>
If the file specified by <pathname> exists on an online disk device then <command> will be executed. Otherwise, control continues with the command that follows on the next line. This command clause is typically used so a script can decide if there is a message reply file to be uploaded or not so it can branch accordingly or to check to make sure a file exists before the script attempts to delete the file.

Example:

```
If Exists "email.replies" Send "email replies"
Text
If Exists "email.replies" Delete "email.replies"
stop
```

In the above example, IF EXISTS is used to see if a file called "email.replies" is on the disk. If so, then we transmit it to the host using Text mode and then delete the file once it's been sent. If IF EXISTS wasn't used then an error condition would abort the script if the file didn't exist on disk.

If Failed <command>
If the previous WAITFOR command failed (i.e. didn't find its target before the time limit expired) then <command> is executed. Otherwise, control passes to the next line and <command> is ignored. You can also use this command to test to see whether a protocol file transfer was successful or not and to test to see if a text upload was successful or not. IF FAILED can also be used to see if a GETLINE command was aborted by the user by pressing the ESCape key.

Example:

```
set timer 30

# Loop
xmit "AT DT 555-1212^M"
waitfor string "Connect"
if failed goto abort
stop

# abort
hangup
goto loop
```


In the above script segment, the IF FAILED command routes execution to the "Abort" routine that hangs up the phone and restarts the dialing process if the string "Connect" isn't received from the modem within 30 seconds of the time we started waiting for it.

If Found <command>
If the previous WAITFOR command found what it was waiting for then execution continues with <command>. Otherwise, control passes to the next line and <command> is ignored. IF FOUND is essentially the opposite of IF FAILED described above.

If Keyboard <character> <command>
If the key pressed in response to the most recent WAITFOR KEYBOARD command matches <character> then execution continues with <command>. Otherwise, execution continues with the next line and <command> is ignored. Note that this command can also test control characters designated by preceding a letter with the "^" character.

Example:

```
Display "Yes or No?"
waitfor keyboard

if keyboard Y goto Yes
if keyboard N goto No
stop

# Yes
# No
```

The above script segment demonstrates using IF KEYBOARD to test and branch based on a key the user supplied.

If NotEqual "string1" "string2" <command>
If "string1" is not identical to "string2" then execution continues with <command>. Otherwise, execution continues with the next command.

Example:

```
Display "^J^MEnter Password: "
GetLine 1
if null 1 goto abort
if notequal "$1" "password" goto abort
display "^J^MAccess granted!^J^M"
stop

# abort
display "^J^MAccess denied!^J^M"
stop
```

The IF NOTEQUAL has the opposite function of IF EQUAL as shown in the above example.

If Null <Variable Number> <Command>
If the indicated variable (1-8) contains the null string, i.e. is empty, then <command> is executed. If the indicated variable has a non-null value then the command following the If Null command gets executed. A variable could have a null value either because you tried to read past the end of a text file using the ReadFile command (described elsewhere) or because the variable never had a value assigned such as at the beginning of a script file or when a user presses RETURN before entering input in response to a GETLINE command.

Example:

```
# top
Display "^M^JEnter a string: "
GetLine 1
if null 1 goto error
Display "^J^M$1^J^M"
stop

# error
```



```
Display "^G You entered a blank bozo! Try
   again...^J^M"
goto top
```

The above shows how to use the IF NULL command to test for an empty variable.

INC <Variable Number>
Increments the unsigned integer value stored in the indicated variable. The range of legal integer values is 0..65535 inclusive. If you increment a variable containing 65535 then the value will not change.

Example:

```
assign "1" 1
inc 1
display "$1"
inc 1
display "$1"
inc 1
stop
```

The above script when executed would display "234". You can also use this command, together with DEC and IF EQUAL to control loops, generate unique file names, or count the lines in a file and so forth. You could also prompt a user for a list of file names to be uploaded and count them as you write them to a text file for later use.

Open <file number> <pathname>
Opens the indicated user file designated by <pathname> and assigns the <file number> specified.. If the file does not exist it is created. The read/write mark is set to the start of the file. <file number> is an integer value between 0 and 1 inclusive.

Example:

```
Open 0 MessageFile
WriteFile 0 "Hello there!^M"
Close 0
```

The above commands would open the file "MessageFile" and write the string "Hello There!" to the beginning of the file. You would also use the OPEN file to open a file that you intended to read from. Since the OPENA command positions the read/write marker at the end of the file, beyond all previously written data, you cannot use it on a file you intend to read from.

OpenA <file number> <pathname>
Opens the indicated user file designated by <pathname> and assigns <file number> to the file. <file number> is an integer between 0 and 1 inclusive. If the file does not exist it is created. The read/write mark is set to the end of the file for appending. You cannot read data from a file opened with OPENA.

Example:

```
OpenA 0 LogFile
WriteFile 0 "$date $time Compuserve^M"
Close 0
```

The above commands would open the file LogFile and would write the date, time, and the word "Compuserve" at the end of the file.

Pause <seconds>, Pause
Execution pauses for the duration specified in <seconds>. If <seconds> is omitted then execution will pause for approximately one second.

Example:

```
Display "Start clock!^M^J^G"
pause 1
display "1 "
pause 1
display "2 "
pause
display "3 ^M^J"
```

The above would display the string indicated and then display a number each second for 3 seconds.

Print Init "string"
Sets the printer initialization string as specified. Control characters can be encoded just like ASSIGN.

Example:

```
print init "^I80N"
```

The above would set the printer initialization string to "Control-I80N".

Print Off, Print On
Turns off online printing.

Example:

```
Print On
View TextFile
Print Off
```

Prints the file TextFile on your printer.

Print Screen
Prints the current screen.

Example:

```
print screen
```

The above would send the contents of all 24 lines of your screen to the printer.

Quit
The ProDOS QUIT MLI call will cause TIC to shutdown and exit to the selector program. The communications port is disabled by this command. See also EXIT.

Example:

```
Hangup
Quit
```

The above sequence would hang up the telephone connection and then exit TIC which would reload the program selector (if you were using one).

ReadFile <file number> <variable number>
Reads a line of text up until a carriage return or the end of the file specified by <file number> into the variable specified by <variable number>. You should use the <file number> assigned by the Open or OpenA command used to open the file. <Variable number> can range from 1-8. A null string will be read when the end of the file has been reached.

Example:

```
Open 0 "MyFile"

# ReadLoop
ReadFile 0 1
if null 1 goto exit
Display "$1^M"
goto ReadLoop

# Exit
close 0
stop
```

The above script would open the text file "MyFile" and read the lines of text from the file one at a time. It would continue until the variable 1 was null (i.e. the end of the file was reached) at which time the file would be closed.

Receive <pathname>
Receives <pathname> from a host computer using Xmodem, 1K Xmodem, 4K Xmodem or ProDOS xmodem protocol.

Examples:

```
Receive MyFile
```

Return
This command returns control back to the next line following the last executed DO command.

Example:

```
set baud 2400
Do Login
stop

# Login
<commands to log you in>
Return
```

The above script sequence sets the baud rate and then calls the LOGIN procedure as a subroutine. Once the LOGIN procedure terminates, control continues with the STOP command that follows the DO command.

Send <pathname> ProDOS
Transmits <pathname> to a host computer using ProDOS Xmodem protocol.

Example:

```
send binfile prodos
```

Send <pathname> Text
Transmits <pathname> to a host computer using Ascii protocol using the previously defined prompt, line delay, and character delay.

Example:

```
send MyTextFile Text
```

Send <pathname> Xmodem
Transmits <pathname> to a host computer using Xmodem protocol.

Example:

```
send gameprog xmodem
```

Send <pathname> 1kXmodem
Transmits <pathname> to a host computer using 1K Xmodem protocol.

Example:

```
send mydocument 1kXmodem
```

Send <pathname> 4kXmodem
Transmits <pathname> to a host computer using 4K Xmodem protocol.

Example:

```
send myprogram 4kXmodem
```

Set Append On/Set Append Off
This setting determines whether TIC will write all recording buffer disk saves into a single large file or not. The default is OFF meaning that each time a recording buffer is saved to disk, the file name is changed by incrementing a file number in the last character position of the name (i.e. TICTEMP.1, TICTEMP.2, TICTEMP.3 and so on). If you set this option ON then all subsequent automatic buffer saves will be appended to the end of the previous file. This option can be used to allow

you to download text files that are longer than the maximum capacity of the recording buffer (which is currently about 47,000 bytes long).

Example:

```
set append on
```

The above command would tell TIC to write all buffer saves to the same file, writing each buffer after the other to create a single large download file.

Set Autosave <pathname>
This sets a new autosave file. The <pathname> name should be at least 3 characters in length. Note that if a partial pathname is used, the root directory will be the default starting directory level for the partial pathname.

Example:

```
set autosave /ram5/messages
buffer on
stop
```

The above commands would tell TIC that automatic buffer saves should go to the filename indicated and turn on the recording buffer.

Set Baud 300, Set Baud 1200, Set Baud 2400, Set Baud 4800, Set Baud 9600, Set Baud 19200
Sets the Baud rate as indicated.

Example:

```
Set Baud 1200
Set Duplex Full
```

The above commands would set the baud rate to 1200 bits per second and then set the duplex mode to Full.

Set Binary2 Auto [or ON], Set Binary2 Manual [or OFF], Set Binary2 Uploads, Set Binary2 Downloads
Selects how TIC should handle Binary II extraction and header adding. Note that the arguments AUTO and ON are equivalent as are MANUAL and OFF. If you choose ON then TIC will always add Binary II headers during uploads and will always attempt to extract members of Binary II files during downloads. By choosing OFF TIC will ignore Binary II headers during downloads and won't add them during uploads. The options UPLOADS and DOWNLOADS let you specify that TIC should use Binary II for transfers in one direction but not for transfers in the other direction. While the default setting is ON, many users who prefer to use offline utilities such as Shrinkit will choose UPLOADS so that TIC will leave alone Binary II files as they're downloaded but will add Binary II headers to files that need them during uploads.

Example:

```
set binary2 off
```

The above command disables automatic binary II file extraction during protocol downloads and disables addition of Binary II headers during protocol uploads.

Set Buffer Auto, Set Buffer Manual
Sets whether TIC will recognize a ^R or ^T signal from the host to direct TIC to turn on or turn off the recording buffer under Host control. The default is Manual.

Example:

```
set autosave "/ram5/myfile"
set buffer manual
buffer on
stop
```

The above sequence sets the autosave file as indicated, disables automatic buffer control, turns on the recording buffer and terminates.

Set CDelay <0-9>, Set LDelay <0-9>
Sets the Character or Line delay timer for Text file uploads. These settings default to 0 if you do not set them and remain set until you reset them or re-run TIC.

Example:

```
Set CDelay 2
Set LDelay 1
Send MyTextFile Text
```

The above commands set up a text file transfer with a wait of 2 between each character, and a delay of 1 between each line. Note that line delays are longer than character delays. The SET CDELAY command also affects the speed of the XMIT command.

Set DFormat 8N1, Set DFormat 7E1, Set DFormat 701, Set DFormat 7E2, Set DFormat 702
Sets the serial port data format to the specified combination of data bits, parity, and stop bits. When TIC is first started it defaults to the 8N1 format (8 data bits, no parity, 1 stop bit). "O" means odd parity while "E" means even parity.

Example:

```
set baud 2400
set duplex half
set dformat 7E2
set statline off
emulate termcaps/vt52
```

The above commands set you up to communicate with a host computer that requires 2400 baud, half duplex, 7 data bits, even parity, 2 stop bits, a 24 line display, and DEC VT-52 emulation.

Set Duplex Full, Set Duplex Half, Set Duplex Chat
Sets Duplex to full, half, or chat for terminal mode.

Example:

```
Set Duplex Full
Set Baud 2400
Set Timer 30
```

The above commands could be added to the start of most of your scripts to set the duplex, baud rate, and timer limit the way you want them for the rest of the session started by the script to follow.

Set Debug On, Set Debug Off
Enables or disables the display of script commands as they execute. These commands are used primarily to find errors in scripts.

Example:

```
set debug on
<script commands that might contain errors>
set debug off
```

The above sequence shows how to have TIC display script commands as they execute on screen. Normally you only see the results of script command execution and not the commands themselves. The debug mode allows you to see the script command as it executes. This is useful to find mistakes in scripts or to just follow their progress as they execute.

Set Flush On, Set Flush Off
This setting tells TIC whether or not it should flush incoming data to the screen during script file execution between each command. The default situation is ON which means that between each script command execution, TIC flushes any waiting data to the screen. This keeps the display looking neat but can have the side effect that data your script is trying to read via a WAITFOR or GETLINE command might get lost if it gets flushed to the screen before the particular WAITFOR or GETLINE command executes. At the start of each script, FLUSH mode is set to ON.

Example:

```
set flush off
xmit "Go Apple2^M"
waitfor string "Forum"
set flush on
```

In the above example script, if the "Forum" string arrives between the time we transmit the "Go Apple2^M" command and the time we start the following WAITFOR command then TIC will not miss the "Forum" string as the incoming data buffer won't be flushed to the screen until after WAITFOR has read the data buffer.

Set PaDCR ON, Set PadCR Off
This sets your preference for whether TIC will add a space character to blank lines during text uploads. The default is ON.

Example:

```
set padcr on
send MyTextFile Text
```

The above sequence sends the file "MyTextFile" to the host using Text mode and adds a blank space to any line that consists only of a carriage return. This is used so the host won't think the end of the file has been reached when there's a blank line in the middle of a file you want to upload. Many, though not all, hosts take a blank line as a way to signal the end of a document.

Set Port SSC, Set Port IIGS
Sets the communications driver to Super Serial Card or IIGS internal port. This command overrides the initial card identification routine in TIC.

Example:

```
Set Slot 3
Set Port ssc
Set Baud 2400
```

The above script sets the modem slot to slot 3, tells TIC that a super serial card compatible card is there, and then sets the baud rate to 2400. These commands must be used in this order. Note that this command is ordinarily not needed as TIC will usually be able to identify the type of port in use once you set the slot (and you only need to set the slot when you're not using slot 2 where TIC always looks for a modem card or port.)

Set Prefix <pathname>
Sets the ProDOS prefix.

Example:

```
set prefix /ram5
```

The above script command sets the ProDOS prefix to the Apple IIGS RAM disk so that subsequent disk operations will assume that you want to work with files on the "/ram5" device unless you specify a full pathname to the contrary.

Set Prompt <character>
Sets the handshaking prompt for text uploads.

Example:

```
set prompt ":"
send MyTextFile Text
stop
```

The above script sequence sends the file "MyTextFile" to the host using the ASCII text mode and uses ":" as a hand shake prompt so each line of text will be sent followed by TIC waiting for the ":" character to arrive prior to sending the next line of text.

Set Pslot <0-7>
Tells TIC where to find your printer interface. This command would typically be placed in your TIC.STARTUP file so that it would be run at the start of a session.

It is not needed at all if your printer is attached as a slot 1 device (this is the usual situation on most Apples). You may choose any slot with the exception of slot 3 as its firmware is used by the display screen. If you specify slot 0 then no printing functions will be available. You should select slot 0 if you do not have a printer to prevent an accidental printer command from crashing the program.

Set PTimer <0-255>
This command allows you to tell TIC how long it should wait for a prompt character during an ASCII text mode file transfer. The default time limit is 20 seconds. If you specify a zero for this value then TIC will wait forever until the prompt arrives before it sends the next line of text in the upload. This setting remains in effect until you run TIC again at which time the setting with again be 20 seconds. You would usually not need to use this command unless you are uploading text to a networked information service that sometimes takes longer than 20 seconds to process uploaded text.

Example:

```
set ptimer 30
set prompt ":"
send MyFile text
```

Set Screen On, Set Screen Off
Either enables or disables the display of incoming data to the screen. Other commands are not affected by this command, therefore you can write a script that does a login handshake with the screen disabled so you can display your own screens instead of those from the host during the login process. The cursor is not displayed by TIC when the SCREEN is turned OFF.

Example:

```
display ^L              Clear the screen before
                        starting
set screen off          Turn off incoming data
                        display

# top
display "Dialing...^J^M"
```

```
xmit "776-3936^M"

waitfor string "connect"
if failed goto abort

display "Logging in...^J^M"
waitfor string "login:"
if failed goto abort
xmit "Delton^M"

waitfor string "password:"
if failed goto abort
xmit "MyPass^M"

Display ^L              Clear screen again
Set Screen On           Turn screen back on
                        now that we're logged
                        in.

stop

# Abort
hangup
goto top
```

The above is a complete login script that you can modify for use with ProLine BBS systems. Note that SET SCREEN is used to hide the login process so that your login and password are not displayed on screen as the script runs. This allows the script to send you status messages on what's going on behind the screen. This script also shows how to dial until a login is successful (Note that the Abort segment hangs up the phone and then restarts the dialing process after the label Top.)

Set Slot <1-7>
Sets the slot where TIC will look for a super serial card or clone if it is in use. The default is slot 2. This setting has no effect if you are using the built-in modem port on the Apple IIGS.

```
# start
set slot 3
set baud 2400
```



```
set timer 30
set duplex full
set debug off
```

The above is an example of a typical TIC.STARTUP file that gets executed each time TIC is run with no startup pathname specified. Note that the slot must be set before setting the baud rate so TIC can find the modem to actually change the baud rate. Note also that the slot can be set from the ♁—M manual settings menu.

Set StatLine On, Set StatLine Off
Turns on or turns off the TIC status line display at the top of the screen.

Example:

```
set statline off
emulate termcaps/vt52
stop
```

The above sequence sets the screen to a full 24 lines (by turning off the status line) so we can properly emulate a full screen terminal.

Set Timer <seconds>
Sets the time limit for WAITFOR searches and GETLINE commands. If TIMER is set to 0 then there will be no time limit. The maximum time limit is 255 seconds.

Example:

```
# top
set timer 5

# ask
Display "1 = 1200 baud, 2 = 2400 baud: "
waitfor keyboard
if failed goto 2400
if keyboard 1 goto 1200
if keyboard 2 goto 2400
goto ask
```

```
# 1200
set baud 1200
goto continue

# 2400
set baud 2400

# continue
set timer 30
stop
```

The above sequence shows how you can use the SET TIMER feature to allow for a default answer to a question. First we set the timer down to 5 seconds and ask a question. If we get an answer within 5 seconds we act on it. If the timer expires though we just go ahead and assume that the user wanted to use 2400 baud (if failed goto 2400). Once we get past setting the baud rate we set the timer back up to our usual timer setting for WAITFOR STRING commands for the rest of our script.

Set Turbo On, Set Turbo Off
Enables or disables Turbo Xmodem mode as the default for protocol file reception. Note the warnings and limitations regarding the use of Turbo Xmodem in the File transfer section of this documentation.

Example:

```
set turbo on
receive MyFile
```

The above script commands would tell TIC to download the file MyFile using Turbo Xmodem.

Stop
The script is terminated if this command is encountered. No error message is displayed.

Example:

```
Xmit "Go Apple2^M"
stop

#abort
hangup
stop
```

In the above script sequence, the STOP command appears twice. The first STOP command terminates the script after transmitting the string so that execution won't run into the Abort sequence that follows. The last STOP command is optional since scripts terminate automatically when the end of the script file is encountered.

View <pathname>
The file indicated by <pathname> is displayed to the CRT.

Example:

```
# Loop
Display "^J^MEnter name of file to view:"
GetLine 1
if null 1 stop
if exists goto GoodFile
Display "^J^M^GFile not found....^J^M"
goto Loop

# GoodFile
view "$1"
Display "^G^J^MEnd of file — Press a key..."
waitfor keyboard
goto Loop
```

The above script sequence asks the user to enter the name of a file to view. The response is stored in user variable 1. If variable 1 is null (i.e. the user just pressed RETURN) then the script terminates. The script then tests to make sure that the file exists before trying to view it (if you used VIEW on a file that didn't exist, the script would terminate with an error). If the file isn't found, the user is told as much and the script asks for another file name. If the file does exist then the script

branches to the VIEW command following the label "GoodFile" and variable 1 ($1) expands to equal the file name the user entered before. At the end of the file viewing, the script announces "End of file" to the user and waits for a key press before asking for another file name to view. This process repeats until the user just presses RETURN in response to the file name query.

Waitfor Keyboard
Execution will pause until a key is pressed at the keyboard or until the time-out limit expires (see SET TIMER).

Example:

```
# Loop
Display "^J^MStop or Go? "
waitfor keyboard
if failed goto Loop
if keyboard "S" Stop
if keyboard "G" goto Go
Display ^G^J^MInvalid Response!^J^M"
goto Loop

# Go
<rest of script>
```

The above script will display the string "Stop or Go?" to the screen and then it waits for the user to press a key. If the timeout limit expires then the question gets asked again. The If Keyboard commands then test for individual responses (note that upper and lower case don't matter here) and if there's no match the question gets asked again.

Waitfor String "string"
Execution will pause until "string" is received over the serial port or until the time limit expires. If "string" is omitted then execution pauses until any character is received over the serial port. Note that "string" may contain imbedded control characters just like the ASSIGN command.

Example:

```
Waitfor string "Login:"
if failed goto abort
xmit "MyID^M"
stop

# abort
hangup
stop
```

The above sequence waits for the string "Login:" to arrive over the modem port. If it doesn't arrive before the timeout timer expires then the wait will be considered to have failed so execution will continue after the "Abort" label by hanging up the phone and terminating the script. If the string "Login:" is received in time, then the string "MyID" followed by a carriage return will be transmitted to the host and the script will terminate.

Waitfor Time "19:50"
Execution will pause until the specified time matches the ProDOS system time. You should not use this command if you do not have a ProDOS compatible clock installed. Note that the time string must be encoded as shown in 24 hour format. You can also use the $time and $date pre-defined variables along with the If Equal command to control a script based on the time or date. Note that this command cannot time out (See SET TIMER).

Example:

```
Waitfor Time "22:00"
Display "It's 10 pm!^G^J^M"
```

The above commands would pause the execution of the script until the next time that the clock reads 10 pm and would then display the message to the screen followed by the bell signal "^G" and a line feed and carriage return.

Window Left X, Right X, Upper Y, Lower Y
Draws a window box on the current screen. The previous screen contents are overwritten. Note that the Right and Lower values cannot be smaller than or equal

to the Left or Upper values. The dimensions specify the outside borders of the window. The horizontal borders take up a single row and the vertical borders each take up 2 columns.

X = 0-79 = column number
Y = 0-23 = row number

Example:

```
Window 10,20,5,10
```

The above command would draw a box on the screen that was 10 spaces wide and 5 spaces tall. The space within the 'window' would actually be 6 spaces wide by 3 spaces tall accounting for the borders of the box.

WriteFile <File number> "string"
Writes "string" to the user file indicated by <file number>. use the <file number> specified when the file was originally opened by the Open or OpenA command. Note that "string" can contain imbedded control characters designated by "^" as in the ASSIGN command.

Example:

```
Open 0 MyFile
WriteFile 0 "Test line of text^M"
Close 0
```

The above commands open the file "MyFile" and write the line "Test line of text" followed by a carriage return to the file, and then close the file. You would leave the "^M" out of the string field for WriteFile if you wanted the next WriteFile command to continue writing to the same line of the file.

Xmit "string"
The "string" is transmitted out the serial port. Note that imbedded control characters may be encoded in "string" just like with the ASSIGN command. The SET CDELAY can be used to adjust the speed with which the XMIT command sends data. This is useful if TIC tends to send data too fast for a host computer to handle.

Example:

```
xmit "Password^M"
```

Example Login Script

Following is an example TIC script. It's purpose is to call up a bulletin board and login. To develop your own script you should login to the service of your choice by hand and make a note of what the system asks you and what you must type in reply to get yourself online. You will need this information so you can tell TIC how to do the job for you.

```
# script starts here with some init stuff
Set Timer 30
Set Baud 2400

# see if area code is needed?
Display "^L Calling from South Carolina? (Y/N): "
Waitfor Keyboard
If Keyboard "Y" Goto Dial
If Keyboard "N" Goto AreaCode
Goto See

# AreaCode to variable 1 (use your area code here
instead of 803)
Assign "1 803" 1

# dial - note that ^M is a carriage return and ^J is a
line feed
Display "^LDialing ProLine [pro-carolina]^M^J"

# Transmit the dialing command to the modem
Xmit "ATD $1 776-3936^M"

# look for the CONNECT string returned by the modem
if we get online
Waitfor String "CONNECT"
If Failed Goto Abort

# look for the login prompt
```

```
Waitfor String "login"
If Failed Goto Abort

# transmit the user id
Xmit "delton^M"

# look for the password prompt
Waitfor string "password"
If Failed Goto Abort

# transmit password and stop the script
Xmit "MYPASSWD^M"
Stop

# abort
Hangup
Goto Dial
```


Appendix A

Quick Reference Guide

⌘—?	Help screen Display
⌘—?	Display script macro key definitions
⌘—ESC	Hide/Reveal Status Line Display
⌘—A	Attention/Break signal
⌘—B	Change Baud Rate
⌘—C	Clear Recording Buffer
⌘—D	Show Disk Directory
⌘—E	Change Duplex Setting
⌘—G	Load a file into Recording Buffer
⌘—H	Hang up Phone
⌘—I	Initialize Console
⌘—J	Display File to screen
⌘—K	Delete File
⌘—L	List the Recording Buffer
⌘—M	Change Manual Settings
⌘—N	Set New Prefix
⌘—O	Toggle Online Printing
⌘—P	Print Screen to Printer
⌘—Q	Quit - Exit Program
⌘—R	Toggle Recording On/Off
⌘—S	Save Recording Buffer to Autosave file
⌘—T	Transfer a file to/from Host
⌘—V	Display Online Volumes
⌘—W	Write Recording Buffer to named file
⌘—X	Execute a named Script file
⌘—Y	Run the Text Editor
⌘—Z	Zoom - Show Control Characters

Appendix B

Troubleshooting

Symptom	Solution
While Offline:	
No response from modem	Check cable and IIGS control-panel for proper settings and make sure the modem is turned on.
Modem responds but no screen echo of commands	Check IIGS control-panel handshake settings. Make sure modem echo mode enabled (see modem docs)
While Online:	
Nothing you type is displayed	Change to half duplex.
You get garbage on-screen instead of proper text	Check data format (🍎—M menu), consider line noise and try a lower baud rate or cleaner phone line.
Printing doesn't work	Make sure proper slot is selected for printer, try using a Pascal 1.1 firmware protocol compatible printer interface since some older cards don't work well with TIC.
You can't get files you download to run correctly	Read the docs about file transfers, discuss the problem with the uploader as you might have the wrong hardware for the particular program.
You get lots of bad block errors on during upload/download	Try a lower baud rate, cleaner connection, or a protocolwith a smaller block size.

A text file you download is double spaced	You may need to remove line feeds from the file using a tility for this purpose.
Call Waiting knocks you offline	Add "1170" to the front of the numbers you dial to temporarily disable Call Waiting.
Text is displayed too fast to read	Use a slower baud rate or type Control-S to pause the host. Use Control-Q to resume.

File Transfer Error Messages

Message	Explanation
Prompt missing	During a text file upload using prompted mode, the prompt was not received before the PTIMER ran out. All is well though, the next line will just be sent anyway.
Bad ProDOS Info	In ProDOS mode, the ProDOS information sent by the host got messed up. The file will be written out as a TXT file. If the file has an SHK or BNY header then there's no problem. Otherwise, you may need to repeat the transfer if you can't restore the ProDOS information yourself.
Waiting for start	Every protocol starts with this message. It just means that the sender (you) is waiting for the receiver to start the transfer.
Error opening file	Something kept TIC from being able to open the file, i.e. damaged disk, file not found, file exists, etc.

Checksum Mode Enabled — Routine message to let you know that Checksums are being used to verify blocks on the current protocol transfer.

ProDOS transfer enabled — Either you or the user at the other end of a transfer have enabled ProDOS mode.

CRC Mode Enabled — Routine message to let you know that CRC's are being used to verify blocks on the current protocol transfer.

Transfer aborted — Either you aborted the transfer by pressing the ESC key or some serious error in the transfer caused it to be aborted. You must repeat the transfer.

Bad Block — The checksum or CRC calculated by the file receiver doesn't match the one sent by the senderso the block will be sent again.

Timed Out — The other end of the transfer took too long to send an expected reply to TIC. No harm done so long as there are not 10 of these in a row.

Synchronization error — This is a terminal error where the sender and receiver get hopelessly confused as to what block is being sent. You must repeat the transfer.

Sector number error — Similar to a Synchronization error but these can frequently be recovered from by TIC.

Waiting for file — Displayed by TIC in a Ymodem batch download between file arrivals.

ProDOS file write error — Some sort of operating system error is keeping TIC from being able to write a file. This is a terminal error.

Appendix C

Talk is Cheap News Feed

The following bulletin board systems make the Talk is Cheap news feed available to the general public for no fee. The TIC news feed is an interactive message thread available nationally where users can trade script files, ask technical questions, and receive update information in a timely fashion. Many other locations/computers are picking up this news feed including sites on the UUCP UNIX network. All of the systems listed below can be called at up to 2400 baud and most support 9600 baud calls:

City, State	System Name	Phone
Santee, CA	pro-ldm	619-448-1664
Hollywood, FL	pro-exchange	305-431-3203
Arlington Heights, IL	pro-harvest	708-253-8239
Maple Grove, MN	pro-hysteria	612-557-2811
Charlotte, NC	pro-charlotte	704-567-0029
Reading, PA	pro-palace	215-678-4438
Columbia, SC	pro-carolina	803-776-3936
Abilene, TX	pro-abilink	915-673-6856
San Antonio, TX	pro-party	512-829-1027
Falls Church, VA	pro-novapple	703-671-0416
Vancouver, WA	pro-freedom	206-253-9389

On any of the above systems you can post your questions or comments to the TIC news feed by addressing electronic mail to TIC @ pro-carolina. If you frequent a system with access to the proline or UUCP, UNIX, or BITNET networks and don't yet receive the TIC news feed, you can send a request to tic-request @ pro-carolina (see paths in the previous section) to receive the news feed on your system.

Appendix D

Glossary

Attention: Also known as Break. This is a special signal that is required by some mainframes and mini-computers as a sort of RESET command for remote operation. What this signal does varies from doing nothing to logging you off. You should not use the command (Apple—A) that sends this signal unless your host's documentation says it is required. You may see this signal referred to as the PA-1 or Program Attention - 1 signal.

Autostart: An autostart disk is one that will automatically load a program (TIC in this case) when you boot the disk or turn on the power switch with the autostart disk in the boot disk drive.

Baud: This is an acronym for Bits per second. There are 8 bits to a byte (or character) so 300 bits per second (i.e. 300 baud) is a data transfer rate of roughly 30 characters per second accounting for framing overhead. Both ends of any computer connection must be sending and receiving at the same baud rate. Telephone lines can typically support up to 9600 baud depending on line quality and modem in use. The serial port can handle up to 19,200 baud though the Apple II cannot handle sustained transfers of data at this high a baud rate without some loss of data once the internal buffers overflow.

Boot: This is the process of starting up a disk device. From BASIC you can initiate this process by entering the command PR# <slot num> where <slot num> is the number of the slot that contains the disk controller card of the device you wish to start up. On the Apple this will usually be slot 6. If you wish to startup a hard disk, consult the documentation for your particular hard disk for directions on how to startup from that drive. Some hard disk drives cannot be automatically started up. On the Apple IIe, IIc, and IIGS you can cause a disk to be booted (started up) by holding down the Control, Apple and RESET keys simultaneously.

Break: See ATTENTION.

Card: A card is a printed circuit card that is plugged into the inside of a computer to provide a feature, such as a serial port, that was not included with the original computer. See also Port.

Carrier: This is a constant tone broadcast over the telephone line by modems in order to act as a reference tone for data communications. When you dial another computer you (and many modems) can listen to the line for a whistle (the carrier signal) to recognize that another computer is on the line.

CRT: Cathode Ray Tube. This is your computer's display.

Control Panel: This is a special menu available on the Apple IIgs to allow you to set certain default conditions about the computer that will be remembered even after the power switch is cut off. You access the control panel by pressing the CONTROL, Apple and ESC keys simultaneously. The currently running program will be interrupted while you are in the control panel but you may return to the original application unchanged once you have finished with the control panel. See the Apple IIgs manual for more information about the control panel.

DCD: Data Carrier Detect - refers to the modem signal that tells whether or not a modem is online at the other end of the connection. For Hayes compatible modems that emulate communicating with an external computer all the time, the DCD signal must always be on. This is the case when you set up a serial card as indicated in the hardware set-up section of this manual. The EPIC internal modems assume that DCD will only be true when a carrier is actually detected so you have to alter a switch (see the modem's manual) to change the default setting.

DIP Switches: These are the tiny configuration switches on many modems and serial port cards.

Download: This is the process of transferring a file from a host computer to your computer. You download a file from another computer to your disk. See also upload.

DTR: Data Terminal Ready - This is the name of one of the control signals that the computer can send to many modems to cause the modem to hang up the phone under program control. TIC

uses the DTR signal to cause modems to hang up the phone via the HANG UP command described elsewhere. Your modem must support DTR and if applicable should be configured via a switch setting to act upon the advice of the DTR signal.

Duplex: Normally a host computer will transmit back to you every character you type. This is where the display of your typed characters comes from. This mode of operation is known as full duplex. Half duplex means that characters you type are not only transmitted to the host but are also displayed locally to the screen as opposed to the characters being sent from the host. If you find that you either get no characters displayed when you type or you get double characters when you type then change the duplex setting of TIC. "Chat duplex" is a specialized form of half duplex communication that is typically used when you are typing to a friend who is using another personal computer at another location

Emulation: This is a process by which software (TIC in this case) pretends to be a particular dedicated terminal that the host computer thinks it is sending data to. For example, a given host may use software that takes advantage of particular screen formatting features of the DEC VT—52 stand-alone terminal. To take advantage of these features you would need terminal software for your Apple II that could EMULATE the DEC VT—52. that is, pretend to be a VT—52. Emulation is done by translating some incoming characters into functions or other characters that are then sent to the standard Apple screen drivers.

Enhanced IIe: All Apple IIe computers sold after March 1985 are known as Enhanced Apple IIe's. The "Enhancement" consists of a new CPU, and 3 support chips that add a few new features but more importantly for TIC users, fix a few bugs in the way the screen control software is controlled. The enhancement kit is required if you use TIC and can be purchased (installed) from any Apple dealer for $70 or less. If your computer says "Apple IIe" when you boot it then you already have an Enhanced IIe. The unenhanced IIe boots with "Apple II".

Flow Control: The way communications software or hardware keeps data from being sent too fast for the receiver to digest it. The X-On (Control-S) X-Off (Control-Q) protocol whereby a Control-S stops data and Control-Q resumes it is an example of flow control.

Macro: A Macro is a function to save a user keypresses. Typically one talks about Macro keys meaning a keyboard sequence or character that will execute several functions automatically. An ex-ample macro might type a users name or password when he or she presses a single keyboard key. TIC implements Macros by allowing a single keypress to start a script that can then automate several functions chosen by the user.

Modem: A Modem is a computer accessory for converting digital data signals from a computer into tones that can be sent over telephone lines and back again.

Operating System Shell: An operating system shell program is a computer program that acts as an intermediary between the user and the regular operating system of the computer. The Extended Command Processor also from Carolina System Software is a shell that takes commands from you the user and passes them to ProDOS. A shell frequently adds functions not previously available directly from the operating system and makes it easier for the user to access operating system features that were not previously readily available.

Pathname: A pathname is a path of directories required to access a specific file on a disk device. To reference a file named "MYDATA" on volume "/MY.DISK" you would use a "path" to the file: "/MY.DISK/MYDATA". Some files may be additional directories and those directories may contain references to more files or more directories. Directories other than the main volume directory are called subdirectories. If you want to refer to a file contained in a subdirectory, you must include those intermediate subdirectory names in your full pathname. Say you have a volume called "/GAMES" and have a subdirectory called "ADVENTURE" and have a file called "WAR"; the complete pathname to access the file "WAR" would be: "/GAMES/ADVENTURE/WAR". See PREFIX and VOLUME.

Port: A port, a la portal, is a pathway for transmission or reception of data. In our usage, a port will represent a connector included on the back of a computer where you would plug in an RS-232 cable to attach the computer to a modem or printer. A port may be added to a computer via a plug-in card if desired.

Prefix: The ProDOS prefix is a standard default partial pathname that is added to any file name you specify if your pathname is not fully

qualified. A fully qualified pathname is one that starts with the volume name followed by any subdirectories that must be accessed to find the file in question. A prefix is used to shorten the amount of typing you must do to specify a pathname to a file. As an example, suppose you have a volume called "/MY.DISK" and that volume has a subdirectory called "GAMES" and you have a program file called "WAR" that is in the "GAMES" subdirectory. Without a prefix, you would have to use "/MY.DISK/GAMES/WAR" as the pathname to your file. You could, however, set the prefix to "/MY.DISK/GAMES" and then you would only have to specify "WAR" as your pathname as "/MY.DISK/GAMES" would be automatically attached to the pathname you specified.

Prefix Directory: The prefix directory is the directory specified by the current ProDOS prefix. If the prefix is set as "/MY.DISK/GAMES," then the prefix directory would be the subdirectory called "GAMES" found in the "/MY.DISK" volume directory.

Program Selector: A ProDOS program selector is a program that is used to switch from one ProDOS 8 system program to another. When you leave a ProDOS 8 system program via the QUIT or BYE option, a selector program, if available, is automatically loaded into memory. The Extended Command Processor from Carolina System Software is a combination operating system shell and program selector.

Reset: If you get yourself into a part of the program and can't leave, then as a last resort you can usually use a CONTROL-RESET command. You execute this feature by holding down the CONTROL and RESET keys simultaneously, followed by releasing the RESET key. Note that if you use this option while TIC is trying to write to disk, you may damage the data on your disk. A common reason you might have to use this option would be routing output to a printer that does not exist. TIC will wait for the printer to become ready and it never will, thus using CONTROL-RESET is the only way out of the situation short of buying a printer or turning off your computer. In the event of a serious software failure, a FATAL ERROR message will be displayed. You should record any screen message and if the reason for the error is not apparent, contact the program author. Pressing CONTROL-RESET will reset most fatal errors. Note that

pressing CONTROL-RESET will cause many modems to hang up the phone.

Root Directory: This is the directory from which TIC itself is launched. On the TIC distribution disk, the root directory happens to be the same thing as the volume directory. TIC can also be launched from a subdirectory; in this case, the root directory will be the name of that subdirectory that contains the TIC file. The root directory is where TIC will look for scripts and is where TIC will save its configuration file.

RS-232: This is the name of a cable standard for use in attaching serial ports to other serial ports or modems or printers.

RS-422: This is similar to RS-232 but also supports AppleTalk networking on the Apple IIGS.

Script: A script is a text file on a disk device that contains a list of TIC program commands to be executed automatically. See section 8 for a description of valid script commands and how to use them with TIC.

Serial Port: A serial port is an interface between a computer and a modem or the serial port of another device such as a computer. The Apple IIc and IIGS have built-in serial ports while the Apple IIe requires that you add a serial port on a card for this purpose. The Apple IIGS can optionally use a serial port card (such as the Apple Super Serial card) for this purpose.

Upload: This is the process of transferring a file from your computer to a host computer. You upload a file from your disk to another computer. See also download.

Volume: Refers to a logical disk storage unit. Usually a volume will be the same as a floppy disk. You may place any of several ProDOS volumes into a disk drive. Some disk devices such as hard disks may contain more than one volume online at any given time. Under ProDOS, volumes are referred to by name rather than physical location. Volume names may be up to 15 letters, numbers or periods long and must begin with a letter. Volume names are always preceded by a slash "/". An example volume name might be "/MY.DISK".

Appendix E

Product Support

If during your use of this program you note any discrepancy between the actual program action and this documentation or if you have suggestions for new features or improvement of old features, we would greatly appreciate your sending us a note describing your suggestions/findings. You may write to the following address:

Q Labs
1066 Maryland
Detroit, MI 48230
Tech Support: (313) 331-1115

GEnie:	Address: QC
CompuServe:	Address: 73477,3364
America OnLine:	Address: Walker A

To contact the author of the program directly, write to:

Carolina System Software
3207 Berkeley Forest Drive
Columbia, SC 29209-4111

ProLine [pro-carolina]:	delton (login as 'register') 803-776-3936 @ 300-2400 baud
UNIX:	crash!delton@pro-carolina.cts.com
ProLine:	delton@pro-carolina
America OnLine:	delton
GEnie:	delton
Compuserve:	72010,37
MCI Mail:	351-9930