SUP'R'TERMINAL

M & R Enterprises
Sunnyvale, California

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

M&R ENTERPRISES makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual; its quality, performance, merchantability, or fitness for any particular purpose. M&R ENTERPRISES' software is sold "as is". The entire risk as to its quality and performance is with the buyer. Should the software or hardware prove defective following their purchase, the buyer assumes the cost of all necessary servicing, repair or correction, and any incidental or consequential damages. In no event will M&R ENTERPRISES be liable for direct, indirect, incidental or consequential damages resulting from any defect in the software or hardware, even if M&R ENTERPRISES has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This manual is copyrighted and contains proprietary information. This manual may not, in whole or in part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without written authorization from M&R ENTERPRISES.

The information in this manual is believed to be correct at the time of publication, but M&R ENTERPRISES assumes no liability arising from the use of this manual.

NINETY-DAY WARRANTY

M&R ENTERPRISES warrants the products it manufactures against defects in material and workmanship for a period of ninety days from the date of purchase. During said warranty period, M&R ENTERPRISES will repair (or at its option replace) at NO-CHARGE, components that prove to be defective, provided the product is returned, shipping prepaid, to:

M&R ENTERPRISES
285 Sobrante, Suite E
Sunnyvale, California 94086

PROOF OF PURCHASE

Your sales receipt is your warranty validation. Dated proof of purchase (such as bill of sale or cancelled check) must be provided when requesting warranty work to be performed.

"NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. M&R ENTERPRISES IS NOT RESPONSIBLE FOR CONSEQUENTIAL DAMAGES." Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitation or exclusion may not apply to you.

NOTE

The terms APPLE, APPLE II, and APPLE COMPUTER CO. and the APPLE logo, as used in this manual, are registered trademarks of APPLE COMPUTER, Inc.

NOTICE

M&R ENTERPRISES reserves the right to make improvements to the product described in this manual at any time and without notice.

Documentation By

THE INNER LOOP
P.O. Box 31324
San Francisco, California 94131

Documentation Written By
Morgan P. Caffrey

Illustrations & Layout By
Douglas J. Platz

SUP'R'TERMINAL Design By
John R. Wilbur Jr.

Firmware By
Andy Hertzfeld

TABLE OF CONTENTS

INTRODUCTION

Welcome to the world of 80-column video display on the APPLE II computer!

What SUP'R'TERMINAL Is

The SUP'R'TERMINAL is an 80-character per line, 24-line per screen peripheral interface "card" (circuit board) used with a separate adaptor board. It requires a video "monitor" instead of a standard or color television set.

What SUP'R'TERMINAL Does

SUP'R'TERMINAL provides an independent video display storage area within the APPLE II. From this display storage area, a video signal is produced and routed to your video display monitor via a separate video cable. When SUP'R'TERMINAL is turned on, normal character output to the standard APPLE II screen is inhibited. The APPLE II graphics are not displayed on the SUP'R'TERMINAL monitor. Graphics features can still be used, but only when two monitors are connected simultaneously. The color signal associated with the graphics feature is considerably reduced.

What Is Expected Of The Reader

This manual is written with the expectation that the user is familiar with the APPLE II power-on sequence, normal keyboard functions, etc. Users who are exploring the APPLE II for the first time should become familiar with the available user's guides:

"APPLE BASIC PROGRAMMING MANUAL"
"APPLESOFT BASIC PROGRAMMING REFERENCE MANUAL"
"DOS 3.2 INSTRUCTIONAL AND REFERENCE MANUAL"
"APPLE REFERENCE MANUAL"
"APPLE PASCAL REFERENCE MANUAL"

NOTE

It is not necessary to read ALL of these manuals. They are available for different system configurations and all contain useful general information.

This manual consists of the following sections:

1. Installation Guide
2. Operator's Guide
3. Programmer's Guide
4. Sample Program Listings
5. Appendices

SECTION 1 - INSTALLATION GUIDE

What Happens During Installation

One integrated circuit (IC) is removed from its socket in the APPLE II main circuit board (''motherboard''). The IC chip is transferred (inserted) to a SUP'R'TERMINAL adaptor board.

The adaptor board is inserted in the socket from which the IC chip was removed.

The cable leading forward from the adaptor board is plugged into the main SUP'R'TERMINAL circuit card, completing the computer-to-SUP'R'TERMINAL connections.

The new video signal is routed to the video monitor from the video-out jack at the front of the main SUP'R'TERMINAL circuit card.

NOTE

This process requires a tool to remove and insert the IC chip. It is recommended that the novice *NOT* attempt this installation procedure. We suggest that dealer assistance be obtained for installation.

Parts Required:

1. SUP'R'TERMINAL main circuit card.
2. Adaptor plug-in board.
3. Video cable with connectors (The same kind as used with the APPLE II standard VIDEO OUT.)
4. Video monitor with 8+ megahertz bandwidth

Installation Steps

The installation steps described below void the APPLE II warranty.

1. TURN OFF THE APPLE II. REMOVE THE POWER CABLE.

2. Use an IC puller to remove IC ''C2'' from its socket in the ''motherboard''. See Illustration 1. Be careful not to bend the pins. This IC chip will be used in step 3.

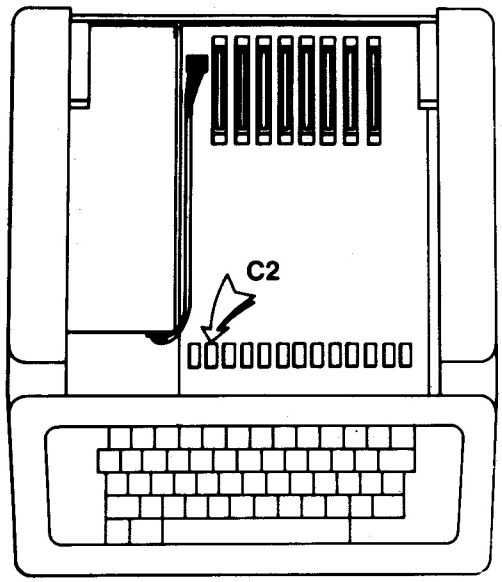


Illustration 1

3. Insert the IC chip into the empty socket of the adaptor board as shown in Illustration 2. Be sure to orient the front of the IC chip (with the half-moon or single dot) to the front of the adaptor board (with the wires). Use special care when inserting the IC chip not to bend the pins. Don't rush this. Most of us don't keep extra IC chips around.

WARNING

If you insert the IC chip backwards in the socket and apply power, *the chip will be DESTROYED.*

4. Insert the adaptor board pins (wires to the *front* of the computer - See Illustration 3.) into the empty "C2" socket on the "motherboard". Take care to hold the motherboard *firmly* to avoid excessive bending. Firm continuous pressure works best. Suggestion: Four pennies, wrapped once in tape and placed one-eighth inch under the motherboard make the board *extra* secure.

WARNING

Don't slide the pennies in more than one-eighth inch under the edge of the mother board. This could bend protruding leads under the motherboard, causing a short.

WARNING

If you insert the adaptor board backwards and apply power, damage to the Apple II is likely to occur. Be *sure* to orient the cable (4 wires) on the adaptor board towards the front of the computer (towards the keyboard) before installing the adaptor board.

5. Connect the cable leading from the adaptor board to the four-pin connector on the SUP'R'TERMINAL main circuit card (see Illustration 4). The black wire *must* be on top.

6. Insert the video cable into the SUP'R'TERMINAL's VIDEO-OUT jack. See Illustration 4. Route the cable forward, around the side and out the small vent in the back. Connect the cable to the video monitor.

7. Insert the SUP'R'TERMINAL main circuit board into SLOT 3.

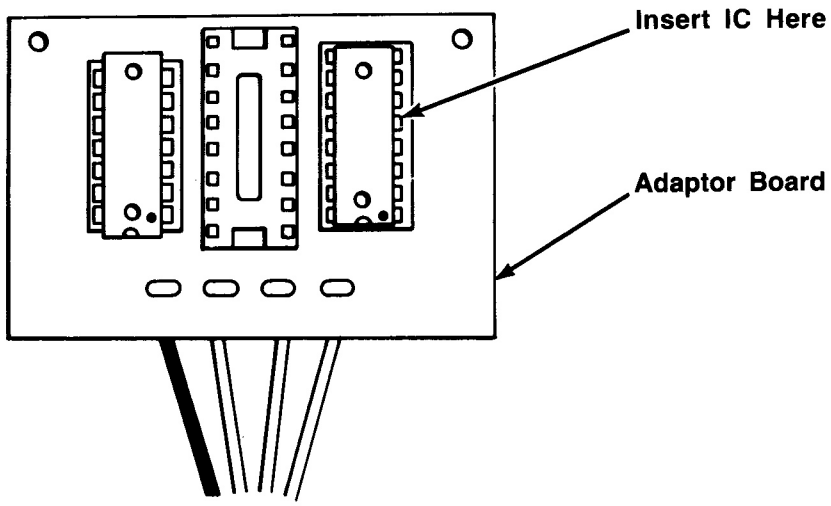


Illustration 2

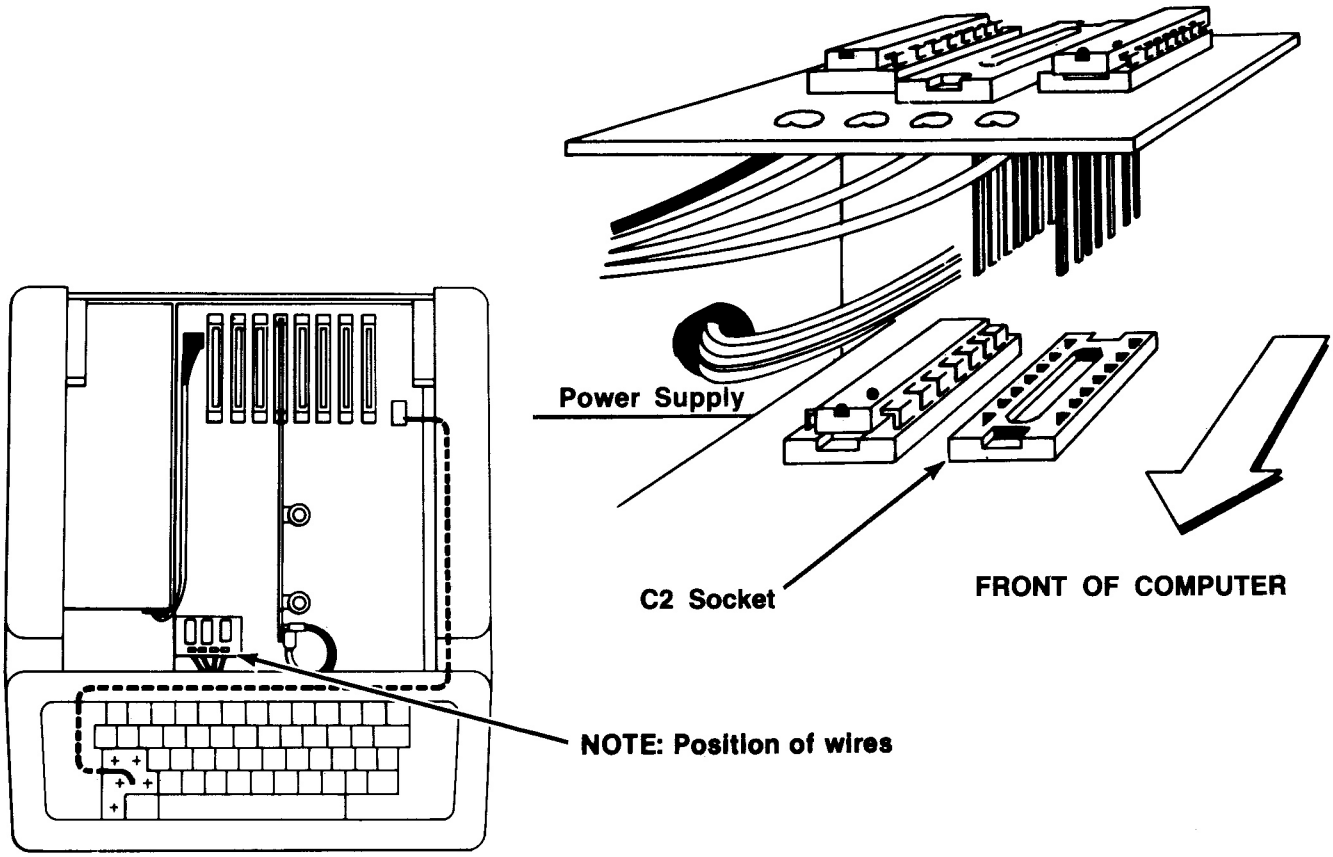


Illustration 3

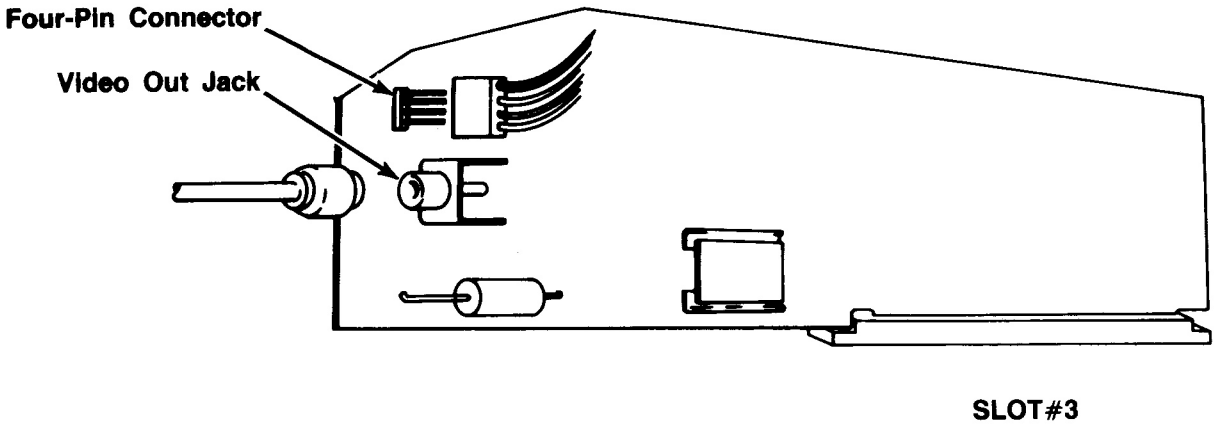


Illustration 4

ADJUSTMENTS

You may adjust the strength of the video signal (brightness) and the sharpness of the character image.

Read the Operator's Guide section on initialization and then, if required, return here to make any adjustments.

Video Signal Strength

Print a set of characters on the screen. Include both normal and inverse characters. Adjust the Video Strength Wheel until both kinds of character display are legible and comfortable for your eyes.

Video Signal Balance (Image Sharpness)

The horizontal bars on each character provide the image sharpness. Use the balance adjustment wheel to adjust for the sharpest possible character on your monitor (See Illustration 6).

This completes the installation process. The installed system will have no effect on APPLE II operation until intialized (this is automatic on PASCAL systems).

Optional SHIFT-Key Modification

NOTE

The hardware modification described below permanently voids the APPLE II warranty. It is HIGHLY recommended that only a qualified technician perform this procedure.

This modification permits the shift key to perform the actual upper case function normally associated with that key.

To make the modification: (see dotted line in Illustration 5)

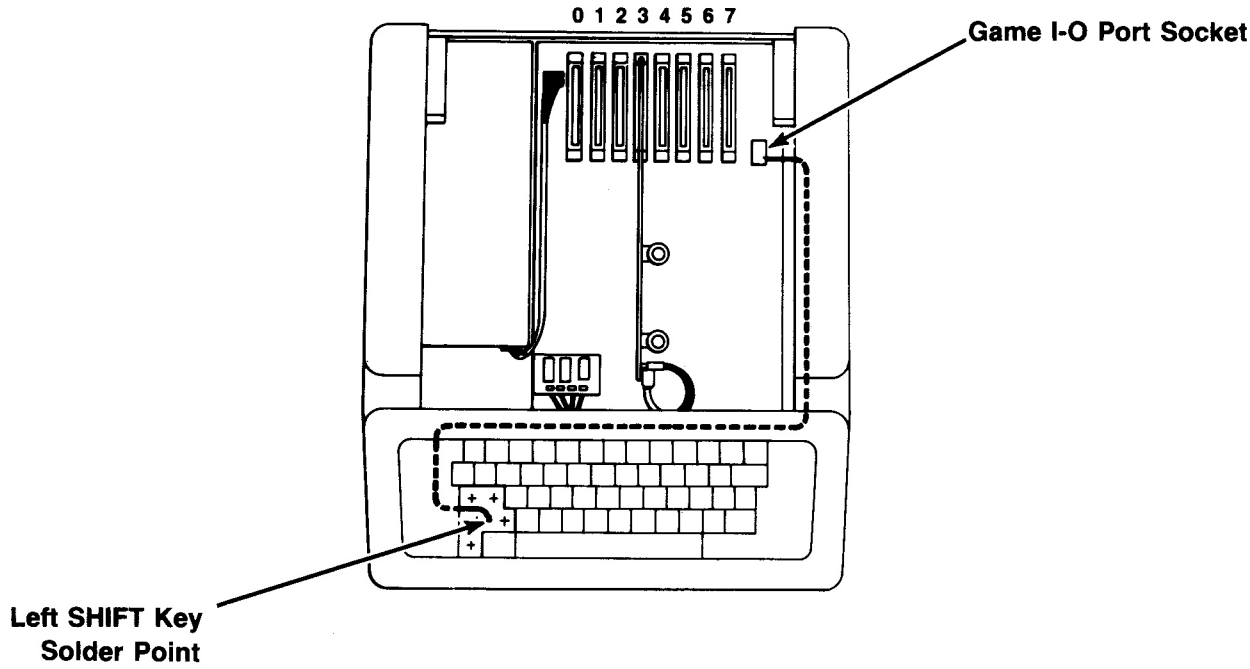


Illustration 5

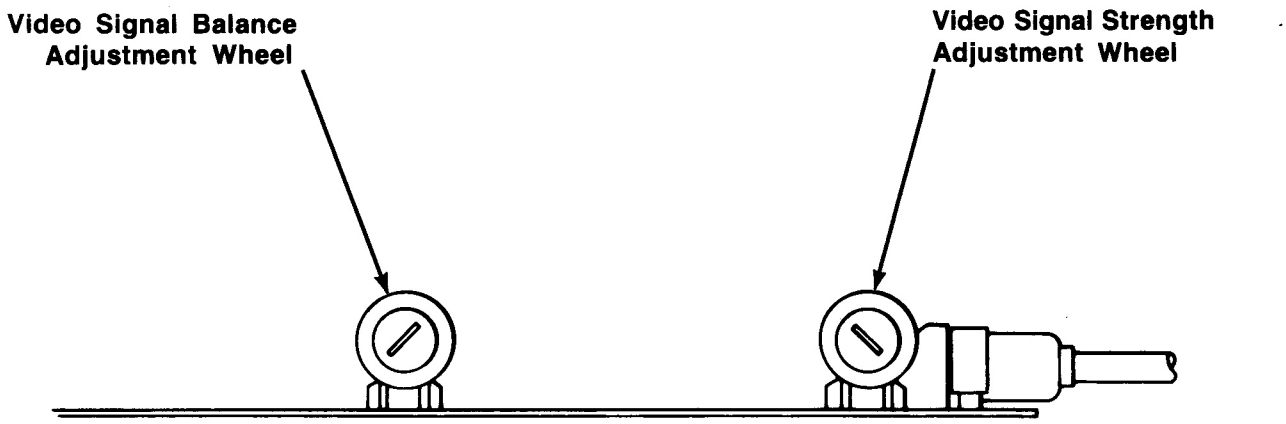


Illustration 6

1. Remove the left SHIFT key cap (and other key caps as necessary).

2. Insert a long (3 feet) wire through the opening to the left and route the wire to the 16-pin game I-O port area.

3. Strip the end of the wire near the SHIFT key and "tin" the end.

4. Locate the P. C. board feed-through pad directly to the right of the SHIFT key. Tin this pad.

5. Solder the wire to the tinned pad. Take care to avoid touching the plastic.

6. Neatly route the wire to the GAME I-O port.

7. Replace the key caps.

8. Remove excess wire, strip the end and insert the end in SW2 (pin 4) of the 16-pin GAME I-O port.

While this procedure is not directly involved with the operation of SUP'R'TERMINAL, the software to handle this modification has been included in the program ROM. See Section 2, under SHIFT KEY for directions on how to use the feature.

SECTION 2 - OPERATOR'S GUIDE

The operation of the APPLE II with the SUP'R'TERMINAL is, in most ways, unchanged. Few general program alterations are required.

Initialization

The SUP'R'TERMINAL card needs to be initialized in order for the hardware to produce the video image. There are two kinds of initializations, "cold start" and "warm start". Each consists of the identical procedure but produces different results.

COLD START

A cold start is performed only the first time SUP'R'TERMINAL is initialized after power up (or when forced by a CTRL-TR). A character set is transferred from the EPROM to the character storage RAM. The screen is blanked and the cursor displayed at top left.

WARM START

A warm start turns SUP'R'TERMINAL on, leaving the cursor at position 0 on the next line position. The next character set is not disturbed. The screen is not cleared.

FROM INTEGER & APPLESOFT

To initialize SUP'R'TERMINAL from either BASIC, type:

PR#3 RETURN

This activates the board and screen RAM and, if a cold start, "downloads" a character font from the program EPROM to the character RAM area.

Disk System HELLO Program

To simplify the process, enter the following statements for either BASIC and make it part of each diskette's HELLO program.

```
10 D$ = "":REM CTRL-D
20 PRINT D$;"PR #3"
30 END
```

These statements initialize SUP'R'TERMINAL. Any other HELLO program statements may be executed in their normal sequence.

Cassette System Initialization

Systems operating without the DISK II may also use SUP'R'TERMINAL. With a cassette-based system, just type "PR#3" at initialization.

FROM PASCAL SYSTEMS

Place any bootable PASCAL diskette in the drive and turn on power. SUP'R'TERMINAL comes up initialized and running.

PASCAL Notes:

Everything in the PASCAL system works exactly as it does in BASIC except, as noted above, the ESCAPE commands are not valid in the PASCAL system.

All default PASCAL control characters work. The shift key mod, toggled by CTRL-V, also works.

80 COLUMNS

One minor change can be made to take full advantage of SUP'R'TERMINAL capabilities. The "SETUP" program on the "APPLE 3" diskette can be used to change "screen width" from 79 to 80 characters. This will cause a full prompt line to be displayed.

Horizontal scrolling and screen switching are no longer needed so they are not implemented.

PASCAL COMMANDS

CTRL-T commands must be printed; just typing them has no effect. This can be achieved from the FILER by transferring keyboard: to screen:

CTRL-A and CTRL-V work all the time.

Reset Recovery Procedure

If you happen to press RESET, SUP'R'TERMINAL will be temporarily disconnected. This leaves the screen exactly as it was but keystrokes APPEAR to have no effect (unless you have a separate TV displaying the standard APPLE II VIDEO OUT). The video display may waver somewhat following a RESET. This is normal.

To recover from the accidental RESET:

STANDARD ROM SYSTEMS:

(RESET leaves you in Monitor Mode.)

TYPE: **3D0G RETURN** - Goes to BASIC
TYPE: **PR#3 RETURN** - Performs a warm start

AUTOSTART ROM SYSTEMS:

RESET leaves you in basic

TYPE: **PR#3 RETURN** - Performs a warm start

Initialized Character Mode

Since all BASIC and Disk Operating System commands must be in upper case, initialization sets the character mode to upper case only. Note that this affects alphabetic characters only. Number keys always print numbers unless the SHIFT key is pressed.

SUP'R'TERMINAL Functions

Once your SUP'R'TERMINAL board is operating, APPLE BASIC and PASCAL control keys perform standard functions as described in the respective manuals.

ESCAPE COMMANDS

The ESCAPE Key functions that work in BASIC or when in the monitor (but which do NOT work with PASCAL systems) are briefly listed here.

ESCAPE Commands

ESC @ Clear screen (HOME cursor)
ESC A Non-copy forward space
ESC B Cursor back
ESC C Cursor down
ESC D Cursor up
ESC E Clear to end of line
ESC F Clear to end of screen
ESC I cursor up - locks cursor-move mode
ESC J cursor left - locks cursor-move mode
ESC M cursor down - locks cursor-move mode
ESC K cursor right - locks cursor-move mode

The four keys (I, J, K, M) form a diamond which points in the direction of the cursor movement. Once any of the four ESC key combinations have been used, all four of the keys in the diamond assume cursor movement control functions. No further use of the ESC key is required until the cursor-move mode is released. To release the cursor-move mode, press any other key. It will not print. This feature is exactly the same as described in the manuals for the Autostart ROM and APPLE II PLUS systems as well as the new "APPLE II REFERENCE MANUAL".

ESC Q -- Clears the screen, cancels SUP'R'TERMINAL operation and enables standard APPLE 40 column video and color output.

Control Key Commands

Case-Mode Change

Initially the character case-mode is set to upper case so all commands will work normally. To display lower case letters type:

CTRL-A

This sets the character mode to lower case. To get single upper case letters from the lower case mode, press CTRL-A once, followed by the character to be capitalized. The next character you type will be capitalized. The character mode immediately returns to lower case. To get in to the upper case mode again type:

CTRL-A CTRL-A

(This can be done by holding down the CTRL key while pressing the "A" key twice.)

Stoplist:CTRL-S

CTRL-S halts temporarily, anything being printed out. Any key starts the printing again. This feature differs from the autostart ROM in that it takes effect at any time rather than just after carriage returns.

SHIFT-Key Modification CTRL-V

CTRL-V is a "switch" command for use by those who have had the optional SHIFT key modification done (described in Section 1). The first use switches the feature on, the next use switches it off, etc.

This modification allows the SHIFT key to perform its normal function of upper and lower case. The SHIFT key will function to provide upper case. To disable the feature or have access to the <@] ^ > characters, just repeat the command.

Left-Bracket "[": CTRL-K

CTRL-K prints the left bracket character. This means that no other function can be assigned to the CTRL-K key.

Cursor Column-Tab: CTRL-I

CTRL-I advances the cursor to the next multiple-of-8 column position, which is very useful for creating tables.

HOME: CTRL-L(ASCII FORMFEED)

positions the cursor to the top left position. It replaces the Applesoft "HOME" and Integer CALL -936 commands. The ESC SHIFT-P also provides the same function but only from the keyboard (on input).

NOTE

If CTRL-L is directly followed by a RETURN, a SYNTAX ERROR message will occur. To prevent this, always type CTRL-X after an immediate-mode CTRL-L. Also, an accidental CTRL-L in the middle of a program will clear the screen of any display. This function cannot be disabled.

*GOTO XY: CTRL-^*

CTRL-(SHIFT-N) Prepares SUP'R'TERMINAL to interpret the next two characters keyed as the absolute horizontal (x) and vertical (y) position of the cursor.

This is done by interpreting the ASCII (numeric) value of the key pressed. Normally this is done only under program control and program use is covered in Section 3.

It can also be used to replace the "immediate mode" VTAB function which does not work with SUP'R'TERMINAL. The sequence to get to any line at horizontal position zero is:

VTAB TABLE

CMND	K1	K2	C POS.
CTRL SHIFT-N	Space	sp.	LINE 0
CTRL SHIFT-N	Space	!	LINE 1
CTRL SHIFT-N	Space	"	LINE 2
CTRL SHIFT-N	Space	#	LINE 3
CTRL SHIFT-N	Space	$	LINE 4
CTRL SHIFT-N	Space	%	LINE 5
CTRL SHIFT-N	Space	&	LINE 6
CTRL SHIFT-N	Space	'	LINE 7
CTRL SHIFT-N	Space	(	LINE 8
CTRL SHIFT-N	Space	)	LINE 9
CTRL SHIFT-N	Space	*	LINE 10
CTRL SHIFT-N	Space	+	LINE 11
CTRL SHIFT-N	Space	,	LINE 12
CTRL SHIFT-N	Space	-	LINE 13
CTRL SHIFT-N	Space	.	LINE 14
CTRL SHIFT-N	Space	/	LINE 15
CTRL SHIFT-N	Space	0	LINE 16
CTRL SHIFT-N	Space	1	LINE 17
CTRL SHIFT-N	Space	2	LINE 18
CTRL SHIFT-N	Space	3	LINE 19
CTRL SHIFT-N	Space	4	LINE 20
CTRL SHIFT-N	Space	5	LINE 21
CTRL SHIFT-N	Space	6	LINE 22
CTRL SHIFT-N	Space	7	LINE 23

To move the cursor to an extended horizontal position, just change the KEY1 "space" character in the VTAB table above to a character higher up in the ASCII table. Use the table of ASCII values in Appendix A to learn which key(s) will move the cursor to the position you want. Experiment and see the effects you get.

Cursor Size and Flash Rate

You can control the cursor in ways not previously possible with the APPLE II 40-character screen. You may select from two cursor flash rates, two cursor sizes, a no-flash mode and a no-cursor mode.

To alter the cursor display:

Press CTRL-TC n (n = 0 - 6)

COMMAND	KEY	RESULT
CTRL-TC	0	underline flashes 4 per sec.
CTRL-TC	1	box flashes 4 per sec.
CTRL-TC	2	underline no flash.
CTRL-TC	3	box no flash.
CTRL-TC	4	underline flashes 2 per sec.
CTRL-TC	5	box flashes 2 per sec.
CTRL-TC	6	no cursor

ALTERING SCREEN DISPLAY "WINDOW"

The APPLE standard video display can be manipulated so that you control the amount of screen area being used for output. This is ordinarily done with "POKEs" to special locations. This feature is provided with SUP'R'TERMINAL using a special CTRL-KEY sequence.

(If you have trouble with this concept, enter and run the sample program "WINDOW MAKER" provided in the Section 4.)

The video display window is altered with the commands shown below:

CTRL-TT n - WINDOW TOP (n = 0 - 23)
CTRL-TB n - WINDOW BOTTOM (n = 1 - 24)
CTRL-TL n - WINDOW LEFT (n = 0 - 79)
CTRL-TW n - WINDOW WIDTH (n = 1 - 80)

NOTE

WINDOW LEFT + WIDTH MUST BE less than or equal to 80. When the window TOP is raised (moving the top of the window DOWN the screen), the window BOTTOM lower limit raises accordingly (the bottom may not be higher than the top). Also, when the window LEFT number increases, the maximum window WIDTH goes down accordingly.

CHARACTER DISPLAY

The normal character display will be white characters on a black background. SUP'R'TERMINAL provides for inverse but not flashing characters.

To get inverse characters:

From Integer BASIC: (› prompt)
For INVERSE display: POKE 50,63 RETURN
For NORMAL display: POKE 50,255 RETURN

From Applesoft BASIC: (] prompt)
For INVERSE display: INVERSE RETURN
FOR NORMAL DISPLAY: NORMAL RETURN

USER-DEFINED CHARACTER SETS

You may define and store different character fonts of your own design. The character font format is described in Section 3. Once a font is defined and saved to diskette, you simply load the character font from the diskette to the proper location, described below.

The character set being displayed at any time is stored in RAM locations on the SUP'R'TERMINAL main circuit board. To change fonts, the contents of this RAM area must be replaced with the new font data.

LOADING A CHARACTER SET

One character font resides on the SUP'R'TERMINAL program ROM. It is automatically loaded into the character RAM during cold-start initialization.

Once you have acquired additional fonts, and stored them on diskette(s), use one of the following procedures to load any font:

1. From Diskette to SUP'R'TERMINAL Character RAM

Type the following:

- **ESC-Q RETURN** (Turns SUP'R'TERMINAL OFF)
- **POKE -16202,0 (C0B6)**(Toggles master off/on switch for RAM banks)
- **POKE -12287,0 RETURN ($cfff)** (Turns all memory banks off)
- **POKE -16206,0 RETURN ($C0B2)** (Turns on the Character RAM bank)
- **BLOAD anyfilename,A$C800 RETURN**
- **PR#3 RETURN**(Performs warm start)

2. From Diskette to "Staging" Memory

The second method of loading character fonts is to pre-load them into alternate APPLE II RAM locations and load to the display area using special move commands described below. This area may be thought of as a "staging" area of memory. This method has the advantage of allowing much quicker font loading under software control.

Fonts loaded into the staging area must be of a special "compressed" format described in Section 3. Up to 10 fonts may be held in this staging area at one time. Each font may be loaded into the Character RAM area with only a few keystrokes.

To load the font into active use, type:

CTRL-TF n - Where n is a number key from 0 to 9.

The character keys in the ASCII table (Appendix A) from a "space" (160=0) to "*" (170=9) will also work.

The font is moved (from the addresses in the table below) into the active character set RAM. The font is translated during the move and the new font is immediately active.

These are the staging storage locations:

FONT	HEX	DEC.	CMND
0 -	$1000	4096	CTRL-TF 0
1 -	$1400	5120	CTRL-TF 1
2 -	$1800	6144	CTRL-TF 2
3 -	$1C00	7168	CTRL-TF 3
4 -	$2000	8192	CTRL-TF 4
5 -	$2400	9216	CTRL-TF 5
6 -	$2800	10240	CTRL-TF 6
7 -	$2C00	11264	CTRL-TF 7
8 -	$3000	12288	CTRL-TF 8
9 -	$3400	13312	CTRL-TF 9

To load any compressed character set to any of the locations, type:

BLOAD anyfilename,Adecimal address
or
BLOAD anyfilename,A$hex address

NOTE

LOMEM (a BASIC command which "tells" BASIC where to store program statements or variables) should be set so as to avoid character set interference. If interference does occur, it will garble the character display on the monitor. To recover, just BLOAD the font again and type **"CTRL-TF n"** again. If the program has not accounted for this problem, it may reoccur.

Control Character Function Table

KEYS		FUNCTION
CTRL-	A	Upper and lower case switch
"	B	Unassigned
"	C	Stops BASIC programs
"	D	Disk Operating System flag
"	E	Unassigned
"	F	Unassigned
"	G	BELL
"	H	Backspace
"	I	Cursor column tab (next multiple-of-8 column)
"	J	Line Feed
"	K	Left bracket "["
"	L	Form Feed (HOME and clear)
"	M	RETURN
"	N	Unassigned
"	O	Unassigned
"	P	Unassigned
"	Q	Unassigned
"	R	Unassigned
"	S	Stoplist
"	T	Special function flag
"	TCn	Alters cursor shape or flash rate
"	TFchar	Load compressed character font
"	TR	SUP'R'TERMINAL cold-start RESET
"	TTchar	Sets screen window TOP
"	TBchar	Sets screen window BOTTOM
"	TLchar	Sets screen window LEFT
"	TWch	Sets screen window WIDTH
"	U	Forward space (copy)
"	V	Switches SHIFT-key modification ON or OFF
"	W	Unassigned
"	X	Cancels input line
"	Y	HOME cursor (does NOT clear screen)
"	Z	Clear current line

SECTION 3 - PROGRAMMER'S GUIDE

For the most part you may program with SUP'R'TERMINAL without paying special attention to its features. Only those areas dealing with screen formatting are affected.

There are three main areas of SUP'R'TERMINAL to understand in order do applications programming.

1. PROGRAM Modifications

Program commands which are no longer effective and the commands or sequences to replace them with.

2. SUP'R'TERMINAL MEMORY MAP

Memory areas on the main circuit board including:
a) Character RAM
b) Screen RAM
c) Program EPROM
d) Cursor Control Registers

3. CHARACTER SETS
a) Format
b) Compression
c) Storage & retrieval

These subjects are covered in detail below.

1. Program Modifications

Command Replacement

The commands which need to be replaced in any program are those which make use of APPLE II monitor routines which specifically reference addresses in the range (hex $400 to $7FF). Since SUP'R'TERMINAL provides an entirely separate video display area, these commands do not work properly and must be replaced.

INTEGER BASIC	APPLESOFT BASIC
CALL -936	HOME
VTAB	VTAB
TAB	HTAB
	TAB
","	","

CALL -936 or HOME

The replacement command to blank the video screen and leave the cursor in the upper left corner of the screen is:

FROM THE KEYBOARD - CTRL-L

NOTE

This command causes a SYNTAX ERROR message unless followed by a CTRL-X or Backspace.

The standard alternative to this, ESC-@ also works in the immediate mode.

FROM PROGRAMS - PRINT CHR$(140)

TAB or HTAB

The BASIC commands TAB, and HTAB, used with VTAB, normally allow the programmer or operator to direct the cursor to a specific location.

The TAB command still works in the normal fashion up to column 40. Thereafter use:

POKE 36,X

Where X is a number greater than the current cursor position and less than or equal to 80.

Notice that this means that tabbing backward will not work. In any situation where the programmer does not know the present cursor position, forcefully return the cursor to the horizontal zero position before tabbing to the destination location.

VTAB

VTAB no longer works. To replace VTAB and TAB-VTAB combinations, SUP'R'TERMINAL provides a command sequence to position the cursor to any screen location. To direct the cursor to any absolute position (N):

PRINT CTRL-SHIFT-N HORIZ. VERT.
or
PRINT CHR$(158); CHR$(N+32); CHR$(N+32)

The CTRL-SHIFT-N prepares SUP'R'TERMINAL to interpret the ASCII value of the next two characters received or transmitted. The ASCII for a "space" (160) marks the zero point.

NOTE

An INTEGER BASIC routine to perform the CHR$ function is used in the programs in section 4. (GOTO XY - Lines 2025 - 2040).

"CTRL-^ space space" sends the cursor to horizontal position 0, line 0.
"CTRL-^ space !"
sends the cursor to horizontal position 1, line 1.

Below are further examples:

COMMAND	POSITION	
CTRL-^	(170) (160)	pos. 10, line 0.
"	(190) (161)	30, line 1.
"	(200) (162)	40, line 2.
"	(210) (163)	50, line 3.
"	(220) (164)	60, line 4.
"	(230) (165)	70, line 5.
"	(240) (166)	80, line 6.

TAB & "COMMA"

The Applesoft TAB function (equivalent to the CTRL-I function provided with SUP'R'TERMINAL) is correctly translated. The cursor is advanced to the next multiple-of-8 column position.

The PRINT COMMAND "," function of both INTEGER and APPLESOFT languages are correctly interpreted for the respective languages. The TAB will operate correctly to the 80-column maximum rather than the standard 40-column.

2. SUP'R'TERMINAL Memory Map

SUP'R'TERMINAL memory is divided into three "banks", all of which begin at the same base address ($C800). In order to access one of these memory banks, all banks must first be switched off. Any Read (PEEK) or Write (POKE) in the area of $CFE0 to $CFFF will turn all three banks OFF. Once all banks are turned off, the appropriate bank may be turned on.

The screen and character RAM banks are not accessible from the keyboard while using SUP'R'TERMINAL. You must revert to the normal Apple mode by typing ESC-Q RETURN. (You must now view what you type on the normal Apple monitor, connected to the standard Apple video out jack.) The address $C0B6 (-16202) is a master toggle switch which allows access to the three RAM banks. Read or Write $C0B6 from the monitor or BASIC (-16202) to toggle this switch on (the video signal stabilizes when this is done). The ESC-Q RETURN sequence always sets this switch to the off state. Always follow the $C0B6 (-16202) read or write with a read or write to $CFFF (-16289). This turns off all RAM banks.

This sequence is not required when access is done under program control. For safety, however, always read $CFFF before selecting any RAM bank. The

MEMORY areas are mapped as follows:

CHARACTER RAM

$C800 to $CBFF

Read or Write $CFFF (-12289) - Switches out all banks
Read or Write $C0B2 (-16206) - Switches in Character RAM bank

NOTE

This is "WRITE-ONLY" RAM. Character sets may be written into this area but may NOT be read back.

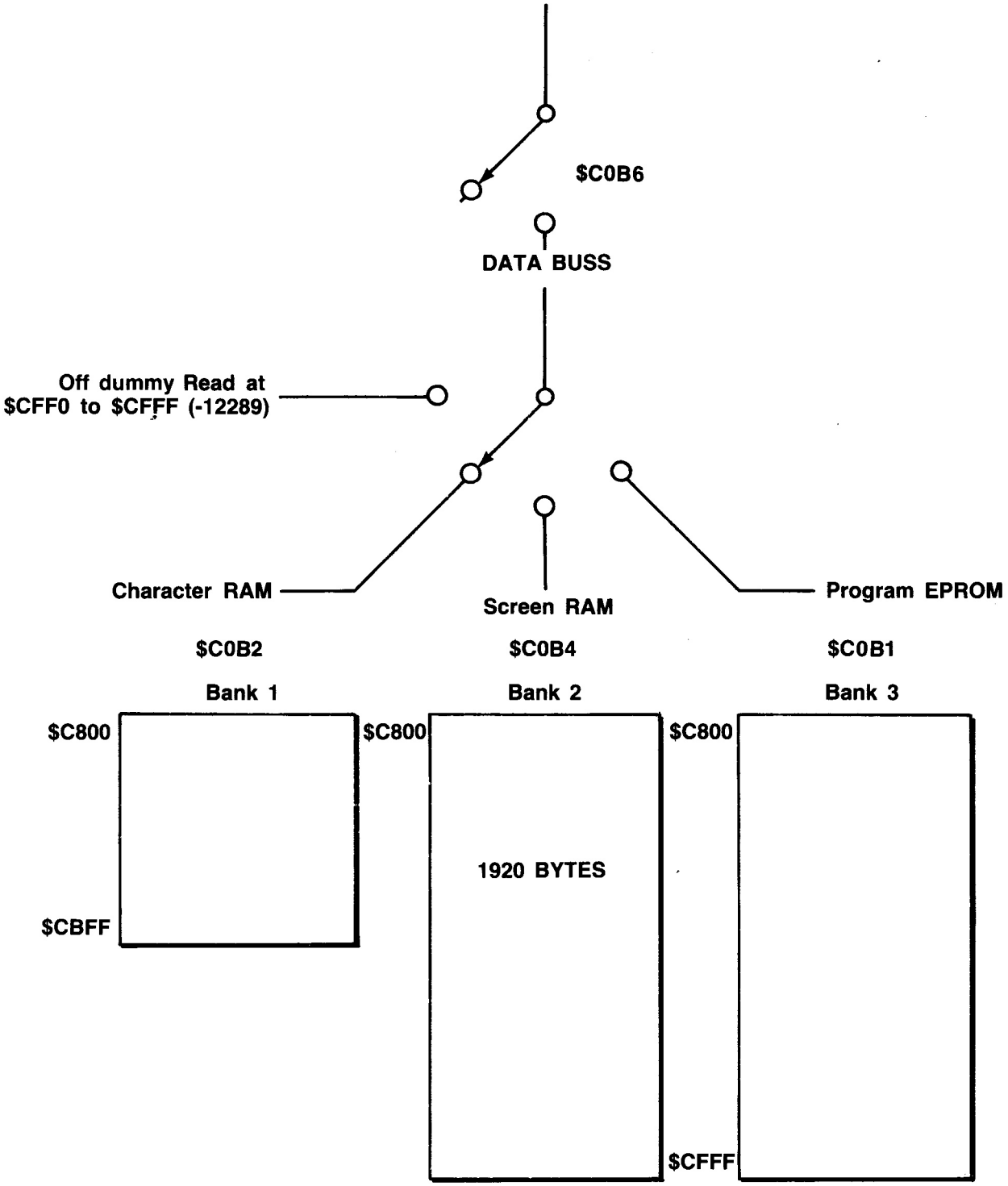
SCREEN RAM

$C800 TO $CF7F

Read or Write $CFFF (-12289) - Switches out all banks
Read or Write $C0B4 (-16204) - Switches in Screen RAM bank
Directly store to $C800 to $CFDF (-14336 to -12352)

The bytes from $CF80 to $CFDF are available to the users. The user is cautioned, however, that *many* of these bytes are used by the controlling firmware. Experiment before committing code or data to any of these locations.

The RAM dedicated to the screen image is **linearly** mapped (i. e. characters stored to seqential Screen RAM locations will appear on the screen in the proper visual sequence). See the program "SCREEN POKER" in Section 4 for an example.


$C0B6
DATA BUSS
Off dummy Read at
$CFF0 to $CFFF (-12289)
Character RAM
Screen RAM
Program EPROM
$C0B2
$C0B4
$C0B1
Bank 1
Bank 2
Bank 3
$C800
$C800
$C800
$CBFF
1920 BYTES
$CFFF

Memory Map

PROGRAM EPROM

$C800 TO $CFFF

Read or Write $CFFF (-12289)
Read or Write $C3XX will enable ROM
Read or Execute code in range $C800 - CFDF (-14336 to -12352)

The accompanying diagram shows the memory "map" of SUP'R'TERMINAL.

Cursor Control

CURSOR CONTROL REGISTERS

The cursor may be directly manipulated under program control.

Four registers are used to control screen character-print location, the absolute character position and the condition of the cursor.

In order to access a register, store the register to be accessed in $C0B8 (-16200). Then store the new contents of the register in $C0B9 (-16199).

The following registers are accessible:

REGISTER $0A (10) - Starting "RASTER" scan line for any character
BITS: 2-0 control start line number of 8-line character

BITS: 4 and 3 are not decoded
BITS: 6-5 are decoded as follows:
0 0 - no blink
0 1 - no cursor
1 0 - 4 blinks per second
1 1 - 2 blinks per second

REGISTER $0B (11) - Ending raster scan line for any character (always 7).
REGISTER $0E (14) - Cursor position high byte
REGISTER $0F (15)- Cursor position low byte

Registers 14 and 15 form a two-byte number which assigns the cursor position relative to address $C800 (-14336) - which is line zero, horizontal position zero. The "GOTO XY " command already described performs the function of updating this register.

The contents of the registers may NOT be read back. The IC chip programmed is a 6845. For more information, consult a specification sheet for that IC chip.

CURSOR DISPLAY MODE

The CTRL-TC commands provide easy cursor-mode control:

CTRL-TC 0 - An underline flashing 4 times a second.
CTRL-TC 1 - A box flashing 4 times a second.
CTRL-TC 2 - An underline with no flash.
CTRL-TC 3 - A box with no flash.
CTRL-TC 4 - An underline flashing 2 times a second.
CTRL-TC 5 - A box flashing 4 times a second.
CTRL-TC 6 - No cursor is displayed.

3. Character Sets

FORMAT

Character sets are made up of 128 (0-127) 8-byte groups. Each 8-byte group is a "bit-map" which is interpreted by the SUP'R'TERMINAL hardware into signals representing a character.

The character format is 5 by 8. Bits 0-4 of each byte form the basis of the character. Bit 5, when set to 1, causes inverse display. Bits 6 and 7 are not decoded.

CREATING A CHARACTER

To create the character "A" in the normal (white on black) mode:

BITS:	7	6	5	4	3	2	1	0
BYTE 0 -	0	0	0	0	0	**1**	0	0
BYTE 1 -	0	0	0	0	**1**	0	**1**	0
BYTE 2 -	0	0	0	**1**	0	0	0	**1**
BYTE 3 -	0	0	0	**1**	0	0	0	**1**
BYTE 4 -	0	0	0	**1**	**1**	**1**	**1**	**1**
BYTE 5 -	0	0	0	**1**	0	0	0	**1**
BYTE 6 -	0	0	0	**1**	0	0	0	**1**
BYTE 7 -	0	0	0	0	0	0	0	0

Integer BASIC "POKE 50,63 or 255" and Applesoft INVERSE and NORMAL commands work in the standard manner. FLASH does not work since SUP'R'TERMINAL does not provide that feature.

In the inverse mode the same character format is used but bit 5 is also set or reset by the INVERSE or NORMAL command. All "1s" are interpreted as black, all "0s" as white. It is possible to think of an extra bit 5, just to the right of bit 0, appearing when bit 5 is set. This extra bit fills the gap between letters:

BITS:	7	6	5	4	3	2	1	0	5
BYTE 0 -	0	0	1	1	1	**0**	1	1	1
BYTE 1 -	0	0	1	1	**0**	1	**0**	1	1
BYTE 2 -	0	0	1	**0**	1	1	1	**0**	1
BYTE 3 -	0	0	1	**0**	1	1	1	**0**	1
BYTE 4 -	0	0	1	**0**	**0**	**0**	**0**	**0**	1
BYTE 5 -	0	0	1	**0**	1	1	1	0	1
BYTE 6 -	0	0	1	**0**	1	1	1	**0**	1
BYTE 7 -	0	1	1	1	1	1	1	1	1

In the example above, the character would appear as inverse (black on a white background).

NOTE

The inverse mode is interpreted only for the alpha characters. All numbers, symbols and the space character appear in the normal mode only.

CHARACTER SET COMPRESSION

Character sets may be converted into a compressed mode. This compressed mode saves disk space and allows the programmer or user to quickly change display fonts. Loading may be done in the middle of a program under program control. With a little more difficulty, it may also be performed in the immediate mode from the keyboard.

Compression Technique

Each byte in a character definition is compared with the byte which follows. If a byte is found to be identical, a count is started. When the first different byte is encountered the count is encoded into the top three bits (5-7) as a binary count. This is, in effect, a pattern repetition count. Thus the letter "A" can be compressed as follows:

BITS:	7	6	5	4	3	2	1	0	
BYTE 0 -	0	0	0	0	0	**1**	0	0	- 0 repetitions.
BYTE 1 -	0	0	0	0	0	**1**	0	**1**	0 - 0 repetitions
BYTE 2 -	0	0	0	**1**	0	0	0	**1**	- initiate count
BYTE 3 -	0	0	0	**1**	0	0	0	**1**	- 1 repetition
BYTE 4 -	0	0	0	**1**	**1**	**1**	**1**	**1**	- 0 repetitions
BYTE 5 -	0	0	0	**1**	0	0	0	**1**	- initiate count
BYTE 6 -	0	0	0	**1**	0	0	0	**1**	- 1 repetition
BYTE 7 -	0	0	0	0	0	0	0	0	- 0 repetitions

Thus the 8-byte letter "A" can be stored as the following six bytes:

	count--character								
BITS:	7	6	5--4	3	2	1	0		
BYTE 1 -	0	0	0--0	0	**1**	0	0		
BYTE 2 -	0	0	0--0	**1**	0	**1**	0		
BYTE 3 -	0	0	**1**--**1**	0	0	0	**1**	-	repetition count = 1
BYTE 4 -	0	0	0--**1**	**1**	**1**	**1**	**1**		
BYTE 5 -	0	0	**1**--**1**	0	0	0	**1**	-	repetition count = 1
BYTE 6 -	0	0	0--0	0	0	0	0		

Thus The "space" character can be compressed to the single byte "E0".

For a program which compresses full fonts into the compressed mode, see FONT COMPRESSOR in Section 4.

CHARACTER SET STORAGE

Characters are stored sequentially from $C800 to $CBFF. Thus:

ASCII 0 - $C800-$C807 (-14336 to -14329)
" 1 - $C808-$C80F (-14328 to -14321)
.
.
.
" 128 - $CBF8-$CBFF (-12359 to -12351)

Compressed Character Set Storage

Compressed character fonts may not be directly loaded into the Character RAM bank. They must be loaded into one of 10 staging areas and subsequently loaded into Character RAM via the CTRL-TF routines.

The character set and storage areas are:

FONT	BASE ADDRESS
0	$1000
1	$1400
2	$1800
3	$1C00
4	$2000
5	$2400
6	$2800
7	$2C00
8	$3000
9	$3400

For easiest loading into the staging area, first move the compressed set into the area in which it will normally be staged (i. e. font 3 in $1C00, font 7 in $2C00, etc.)

Next, BSAVE the font from its intended location:

BSAVE FONT2,A$1800,L(number of bytes)
BSAVE FONT7,A$2C00,L(number of bytes)

Remember that the number of bytes in a compressed font is variable, depending on how many pattern repetitions there are.

Loading Fonts to Staging Area

If you have used the font-saving method described above, loading compressed character sets into a staging area will be done by typing:

BLOAD FONT0
BLOAD FONT2
BLOAD FONT7

But, if a font is to be loaded to a non-standard area then type:

BLOAD FONT0,A$2400 - Loads FONT0 to storage location for FONT5
BLOAD FONT7,A$1000 - Loads FONT7 to storage location for FONT0

Transferring Compressed Fonts Into Character RAM

This will usually be done by program command:

PRINT CHR$(148);"F";CHR$(160–) - FONT n
or
PRINT CHR$("CTRL-T");"F";CHR$(167) - FONT 7
From the keyboard:

CTRL-TF "0" - FONT 0
CTRL-TF "1" - FONT 1
CTRL-TF "2" - FONT 2
CTRL-TF "3" - FONT 3
.
.
.
CTRL-TF "9" - FONT 9

The key sequences above will transfer a compressed font from its staging area in the APPLE memory, into the SUP'R'TERMINAL Character RAM storage area. The compressed font is expanded to its full pattern during the transfer process. It immediately changes the displayed font.

LOADING AN UNCOMPRESSED FONT

It is also possible to load a non-compressed font directly into Character RAM.

Under Program Control

(SUP'R'TERMINAL must already have been initialized.)
1. Read or Write $CFFF (-12209) - Switches out all banks
2. Read or Write $C0B2 (-16206) - Switches in Character RAM
3. BLOAD CHARSET,A$C800

Remember that the character set must be in the non-compressed mode in order for this to work.

Direct From the Keyboard

From the keyboard, the method to use is slightly more complex. You must turn SUP'R'TERMINAL off.

1. Press ESC-Q RETURN to exit SUP'R'TERMINAL mode. This means you must have a separate TV monitor attached to the APPLE II standard VIDEO OUT jack or be able to perform this routine without seeing the characters.

2. Read or Write $C0B6 (-16202)(toggles master RAM switch on) 3. Read or Write $CFFF (-12287) RAM off
4. Read or Write $C0B2 (-16206) Char RAM on
5. BLOAD charsetname,A$C800
6. PR#3 turns SUP'R'TERMINAL on

SECTION 4 - SAMPLE PROGRAMS

WINDOW MAKER

```
 0 REM INTEGER VERSION: WINDOW MAKER
 1 REM WRITTEN BY MORGAN P. CAFFREY
 2 REM  SAN FRANCISCO, CALIFORNIA
 3 REM  JANUARY 23,1980 (V. 2)
 5 X$="X": REM REQUIRED TO ESTABLISH A "CHR$" ROUTINE
 6 CHR=2053: REM LOCATION OF VALUE OF X$
 40 POKE CHR,140: PRINT X$: PRINT "              DEMONSTRATION PROGRAM           ": PRINT
 41 PRINT "THIS PROGRAM ALLOWS YOU TO SET YOUR OWN VIDEO SCREEN LIMITS AND SEE THE RESULTS": PRINT : PRINT
 49 REM DEFINE VARIABLES AND ARRAYS
 50 DIM A(4): FOR I=1 TO 4:A(I)=0: NEXT I
 60 DIM B(4):B(1)=23:B(2)=23:B(3)=79:B(4)=79
 70 DIM C(4): FOR I=1 TO 4:C(I)=0: NEXT I
 80 DIM C$(4):C$="TBLW"
 85 BASE=160: REM EQUALS "1" AFTER A CTRL-T T HAS BEEN ENTERED OR PRINTED
 86 MAX=BASE+80: REM MAX POSSIBLE WINDOW WIDTH
 87 CTRLL=140:CTRLT=148
 90 DIM D$(255): REM  STRING TO PRINT FOR DEMONSTRATION
100 REM FIND OUT WHERE TO PUT THE PRINT OUTPUT
110 INPUT "WINDOW TOP (RANGE 0 - 23) ",A(1): IF A(1)<=B(1) AND A(1)>=0 THEN 120: PRINT "OUTSIDE THE RANGE - TRY
AGAIN": GOTO 110
120 PRINT "WINDOW BOTTOM (RANGE ";A(1);" - 23) ";: INPUT A(2): IF A(2)>=A(1) AND A(2)<=B(2) THEN 125: PRINT
"OOPS - TRY AGAIN": GOTO 120
125 PRINT
130 INPUT "WINDOW LEFT MARGIN (RANGE 0 - 79) ",A(3): IF A(3)<=B(3) AND A(3)>=0 THEN 140: PRINT "OUTSIDE OF
RANGE - TRY AGAIN": GOTO 130
140 PRINT "WINDOW WIDTH (RANGE 1 - ";79-A(3);") ";: INPUT A(4): IF A(4)>=1 AND A(4)<=79-A(3) THEN 150: PRINT "
RANGE ERROR": GOTO 140
150 REM
160 PRINT : PRINT : PRINT " WHAT IS TO BE PRINTED IN THE WINDOW?": PRINT "PRESS RETURN TO GET A STANDARD SAMPLE
DISPLAY": INPUT D$
170 IF D$="" THEN D$="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
180 INPUT "HOW MANY REPETITIONS",Z: IF Z>255 THEN Z=255
200 REM COMPUTE NEW WINDOW VALUES
210 FOR I=1 TO 4
220 C(I)=BASE+A(I)
230 NEXT I
300 REM SET WINDOW
305 POKE CHR,CTRLT: PRINT X$;"R": REM  RESET ALL
310 FOR I=1 TO 4
320 POKE CHR,CTRLT: PRINT X$;C$(I,I);: POKE CHR,C(I): PRINT X$: PRINT : PRINT
330 NEXT I
335 PRINT : PRINT
400 REM NOW PRINT IT
410 FOR I=1 TO 25: PRINT D$;: NEXT I: PRINT : PRINT
415 INPUT "PRESS RETURN TO CONTINUE",E$
420 REM NOW RESTORE TO NORMAL
450 POKE CHR,CTRLT: PRINT X$;"R": PRINT
460 INPUT "ANOTHER? (Y/N) ",E$: IF E$="Y" THEN 100
500 REM RESTORE EVERYTHING AND END
510 POKE CHR,CTRLT
520 PRINT X$;"T";: POKE CHR,BASE: PRINT X$: PRINT : POKE CHR,CTRLT
530 PRINT X$;"B";: POKE CHR,BASE+23: PRINT X$: PRINT : POKE CHR,CTRLT
540 PRINT X$;"L";: POKE CHR,BASE: PRINT X$: PRINT : POKE CHR,CTRLT
550 PRINT X$;"W";: POKE CHR,MAX: PRINT X$: PRINT : POKE CHR,CTRLT
600 PRINT X$;"R": PRINT : PRINT "BYE!": END
```

SCREEN "POKER" DEMONSTRATION

```
  0 GOTO 1000
  1 REM ROUTINE TO POKE CHARACTERS TO INDIVIDUAL SCREEN LOCATIONS
  3 REM GO TO INDIVIDUAL LOCATIONS, REMOVE LINES 23,24,50,55 AND MAKE(55 GOTO 10)
  5 A=0
  8 BASE=-14336:LAST=-14335:A=0
  10 POKE CHR,CTRLL: PRINT X$: PRINT
  15 INPUT "CHARACTER YOU WOULD LIKE TO SEE? ",A$
  23 FOR LINE=0 TO 23
  24 FOR POS=1 TO 80
  25 POKE -12287,0: REM  BANK SELECTS OFF
  26 POKE -16204,0: REM  SCREEN RAM ENABLE
  40 SPOT=BASE+(LINE*80)+POS: POKE SPOT, ASC(A$): POKE LAST,160:LAST=SPOT
  41 REM SPOT=LINE+POS. SET SPOT=CHAR. SET LAST = SPACE. SET LAST = SPOT.
  50 NEXT POS
  55 NEXT LINE
  60 FOR I=1 TO 200: NEXT I: GOTO 1050
 500 POKE CHR,CTRLL: PRINT X$: PRINT
 505 PRINT "NOTE: THIS SAME FUNCTION MAY BE DONE WITH THE 'GOTO X,Y' COMMAND": PRINT : PRINT
 510 INPUT "CHARACTER YOU WOULD LIKE TO SEE ",A$
 520 INPUT "IN WHICH LINE SHOULD IT APPEAR (0 - 23) ",LINE: IF LINE>=0 AND LINE<=23 THEN 530: PRINT "": GOTO 520
 530 INPUT "IN WHICH POSITION (0 - 79) ",POS: IF POS>=0 OR POS<=79 THEN 550: PRINT "": GOTO 530
 550 POKE -12287,0: REM    BANK SELECTS OFF
 560 POKE -16204,0: REM  SCREEN RAM ENABLE
 570 SPOT=BASE+(LINE*80)+POS: POKE SPOT, ASC(A$)
 580 REM  SPOT=LINE+POS
 590 INPUT "ANOTHER (Y/N)",A$: IF A$="Y" THEN 500: GOTO 1050
1000 REM WRITTEN BY MORGAN P. CAFFREY
1010 REM SAN FRANCISCO, CALIFORNIA
1020 REM JANUARY 20,1980
1040 X$="X":CTRLT=148:CTRLL=140:CHR=5+ PEEK (74)+ PEEK (75)*256: REM  LOMEM + 5 = LOCATION OF X$
1045 BASE=-14336:LAST=-14335:A=0
1050 POKE CHR,CTRLL: PRINT X$: PRINT
1060 PRINT "YOU MAY SEE SINGLE CHARACTERS MOVING TO EVERY SCREEN POSITION (ENTER 1)": PRINT
1070 PRINT "YOU MAY SEND THE CHARACTER OF YOUR CHOICE TO ANY SCREEN POSITION (ENTER 2)": PRINT
1075 PRINT "YOU MAY CHOOSE TO QUIT (ENTER 3)": PRINT
1080 INPUT "? ",A$:A= ASC(A$)-176
1100 IF A=1 THEN 10
1110 IF A=2 THEN 500
1120 IF A=3 THEN END
1130 IF A>3 OR A<1 THEN PRINT "OUTSIDE OF LEGAL RANGE - PLEASE TRY AGAIN ": FOR I=1 TO 450: NEXT I: GOTO 1050
```

GOTO XY DEMONSTRATION

```
  0 GOTO 2000
  10 REM GOTO X,Y DEMONSTRATION
  20 POKE CHR,CTRLL: PRINT X$: PRINT : REM  CLEAR SCREEN
  25 BASE=160
  30 INPUT "CHARACTER YOU WANT TO SEE? ",A$
  40 INPUT "HORIZONTAL POSITION ",HRIZ:HRIZ=HRIZ+BASE: REM  GET PROPER OFFSET
  50 INPUT "VERTICAL POSITION ",VERT:VERT=VERT+BASE: REM  GET PROPER OFFSET
  60 POKE CHR,CTRLL: PRINT X$: PRINT
  70 POKE CHR,GXY: PRINT X$;: POKE CHR,HRIZ: PRINT X$;: POKE CHR,VERT: PRINT X$;: PRINT A$: PRINT
  80 INPUT "ANOTHER (Y/N) ",A$
  90 IF A$="Y" THEN 20
  95 GOTO 2050
 500 REM  EVERY POSITION
 505 ZERO=160: REM  THE 'SPACE' CHARACTER REPRESENTS 0: '!' = 1: '"' = 2: ETC.
 510 POKE CHR,CTRLL: PRINT X$: PRINT
 515 INPUT "WHAT CHARACTER WOULD YOU LIKE TO SEE? ",A$
 520 PRINT X$: PRINT
 530 FOR VERT=0 TO 23
 531 C=VERT+ZERO
 532 FOR HRIZ=0 TO 79
 535 B=HRIZ+ZERO: REM  160+THE POSITION
```

```
  550 POKE CHR,GXY: PRINT X$;: POKE CHR,B: PRINT X$;: POKE CHR,C: PRINT X$;: PRINT A$;
  560 NEXT HRIZ
  570 NEXT VERT
  580 FOR N=1 TO 450: NEXT N: PRINT
  590 INPUT "ANOTHER (Y/N) ",A$: IF A$="Y" THEN 510: GOTO 2050
 1000 GOTO 2050
 2000 REM  WRITTEN BY MORGAN P. CAFFREY
 2010 REM  SAN FRANCISCO, CALIFORNIA
 2020 REM  JANUARY 24,1980
 2025 REM  CHR$ FUNCTION LOCATES X$ TO A FIXED POSITION - FIRST IN THE TABLE
 2026 X$="0": REM  THE '0' IS A 'DUMMY' WHOSE ABSOLUTE POSITION IS DECIMAL 2053
 2040 CHR=5+ PEEK (74)+ PEEK (75)*256: REM    LOMEM + 5 = LOCATION OF X$
 2042 CTRLT=148:CTRLL=140:GXY=158
 2045 BASE=-14336:LAST=-14335:A=0:GXY=158
 2050 POKE CHR,CTRLL: PRINT X$: PRINT
 2060 PRINT "THIS ROUTINE USES THE 'GOTO X,Y' FEATURE OF SUP'R'TERMINAL": PRINT
 2061 PRINT "NORMALLY THIS IS ONLY DONE FROM WITHIN A PROGRAM": PRINT
 2062 PRINT " TO POSITION THE CURSOR - PRINT CHR$(158);CHR$(HORIZ);CHR$(VERT)"
 2063 PRINT "LIST THIS PROGRAM TO SEE AN INTEGER 'CHR$' ROUTINE": PRINT : PRINT : PRINT
 2070 PRINT "YOU MAY SEND THE CHARACTER OF YOUR CHOICE TO ANY SCREEN POSITION (ENTER 1)": PRINT
 2071 PRINT "YOU MAY CHOOSE TO SEND A CHARACTER TO EVERY POSITION ON THE SCREEN (ENTER 2) ": PRINT
 2075 PRINT "YOU MAY CHOOSE TO QUIT (ENTER 3)": PRINT
 2080 INPUT "? ",A$: IF A$="" THEN 2050:A= ASC(A$)-176
 2100 IF A=1 THEN 10
 2110 IF A=2 THEN 500
 2120 IF A=3 THEN END
 2130 IF A>3 OR A<1 THEN PRINT "OUTSIDE OF LEGAL RANGE - PLEASE TRY AGAIN ": FOR I=1 TO 450: NEXT I: GOTO 2050
10000 END
```

FONT COMPRESSOR PROGRAM

```
  0 REM CHARACTER COMPRESSOR
  1 GOTO 2000: REM  INITS
100 REM  GET THE NEXT CHARACTER
105 CALL HOME
110 FOR I=I0 TO I7
120 OLD(I)= PEEK (OLDBASE+(CHAR*I8)+I):TEMP=OLD(I)
126 FOR J=I0 TO I7: REM GET BINARY IMAGE IN REVERSE
127 F=OLD(I) MOD I2:BW(J)=F
128 OLD(I)=OLD(I)/I2: NEXT J: PRINT
129 OLD(I)=TEMP: IF OLD(I)<I32 THEN 130: POKE I0,OLD(I): CALL I2:OLD(I)= PEEK (I1)
130 FOR J=I7 TO I0 STEP I1N: IF BW(J)=I1 THEN PRINT "O";: IF BW(J)=0 THEN PRINT ".";: NEXT J
133 NEXT I: PRINT
140 RETURN
199 REM COLLECT AND MOVE COMPRESSED CHARACTER
200 REM TOTAL
201 VTAB I1: PRINT "CHARACTER ";CHAR;"  COMPRESSION COUNT = ";CT+I1
202 Z=0: FOR X=0 TO CT: REM  DISPLAY STORAGE VERSION
205 VTAB Z+2: TAB 15
210 PRINT CMPR(X):Z=Z+ZZ(X)
220 NEXT X: VTAB 15
225 FOR N=0 TO CT:ZZ(N)=0: NEXT N
230 IF NO=I1 THEN INPUT "OK?",A$: IF A$="N" THEN END : REM  LAST REVIEW LINE
240 FOR X=I0 TO CT: REM  MOVE COMPRESSED CHAR
250 POKE NBASE+NPTR,CMPR(X)
255 NPTR=NPTR+I1
260 NEXT X
270 IF NO=0 THEN PRINT "CHARACTER ";CHAR;" AT ";NBASE+NPTR;" - ";CT+I1;" BYTES"
271 IF NO=I1 THEN 272: FOR AA=I1 TO 150: NEXT AA
272 CHAR=CHAR+I1
280 IF CHAR=E THEN GOTO FINISH: REM 20000
285 BYTE=0:ONEUP=BYTE+1:CT=0:RC=0: FOR I=I0 TO I7:CMPR(I)=I0: NEXT I
290 GOTO 1000: REM GET THE NEXT CHAR
399 REM COLLECT NEXT BYTE ROUTINE
400 REM FILL CMPR ARRAY WITH COMPRESSED CHARS
401 CMPR(CT)=OLD(BYTE)+O(RC): REM THE COMPRESSION STROKE!
410 CT=CT+I1:ZZ(CT-I1)=RC+I1:RC=0:FLAG=0
```

```
420 BYTE=BYTE+1:ONEUP=BYTE+1
430 IF BYTE=7 THEN FLAG=2
440 IF FLAG#I2 THEN GOTO 1010
445 REM LAST BYTE DIFFERENTLY
450 CMPR(CT)=OLD(BYTE)+O(I0): GOTO TOT: REM
999 REM MAIN PROGRAM ROUTINE
1000 GOSUB GETCHAR: REM  MAIN TEST - 128 TIMES FOR FULL CHARACTER SET
1010 IF OLD(BYTE)#OLD(ONEUP) THEN GOTO COLLECT: REM  400
1015 RC=RC+I1
1020 REM GOT A MATCH
1025 IF BYTE#6 THEN 1030: REM WHEN BYTE= 6 THEN LAST TWO ARE IDENTICAL
1026 CMPR(CT)=OLD(BYTE)+O(RC): GOTO TOT
1030 BYTE=BYTE+I1:ONEUP=BYTE+I1: IF BYTE=I8 THEN GOTO TOT: REM  200
1050 IF ONEUP<8 THEN GOTO 1010
1060 GOTO TOT: REM  200 SHOW RESULTS
2000 REM INITS
2001 DSP COUNT
2010 CHR$="X":I=0:I0=0:I7=7:I8=8:I2=2:I15=15:I1=1:HOME=-936:I1N=-1:I32=32
2011 POKE I2,169: POKE 3,31: POKE 4,37: POKE 5,0: POKE 6,133: POKE 7,1: POKE 8,96: REM MACH LANG ROUTINE
2019 REM SET ARRAY FOR TOP THREE BITS
2020 DIM O(7):O(0)=0:O(1)=32:O(2)=64
2021 O(3)=96:O(4)=128:O(5)=160:O(6)=192:O(7)=224
2024 REM  VARIOUS ARRAYS
2025 DIM A$(39),B$(39),C$(39),D$(39),E$(39)
2026 DIM BW(7),CMPR(7),ZZ(7),OLD(7)
2028 FOR I=I0 TO I7:BW(I)=I0:ZZ(I)=I0:CMPR(I)=I0:OLD(I)=I0: NEXT I
2040 REM CREDITS
2041 CALL HOME: GOSUB 30000
2048 REM  POINTERS
2050 OLDBASE=2048: REM  BASE OF OLD CHAR
2051 NBASE=16384: REM  BASE OF NEW CCHAR
2055 FOR I=0 TO 1024: POKE OLDBASE+I,I0: POKE NBASE+I,I0: NEXT I: REM BLANK CHAR AREAS
2066 NPTR=0: REM  POINTER INTO NEW OUTPUT FILE
2069 CHAR=0: REM  POINTER TO PRESENT CHAR
2070 CT=I0: REM  NUMBER OF BYTES IN NEW CHAR
2071 RC=I0: REM  RC= REPETITION COUNT
2072 BYTE=I0: REM  POINTER IN "OLD" ARRAY
2074 ONEUP=I1: REM  COMPARISON POINTER
2075 REM SUBROUTINE POINTERS
2076 GETCHAR=100:TOT=200:COLLECT=400:FINISH=20000
2078 NO=0: REM  DEFAULT= NO REVIEW
2079 FLAG=0: REM FIND OUT WHAT TO DO
2080 PRINT "WHAT CHARACTER SET SHALL I": PRINT "COMPRESS? (ENTER 'CAT' FOR CATALOG) ": INPUT A$
2082 IF A$#"CAT" THEN 2086: PRINT "CATALOG":PRINT: INPUT"(PRESS RETURN TO CONTINUE",A$:GOTO2080:
2086 INPUT "STOP TO REVIEW EACH CHARACTER ?",B$: IF B$="Y" THEN NO=1:B$=""
2087 PRINT "NUMBER OF CHARACTERS TO MOVE?": INPUT "DEFAULT (0) = 128",E: IF E=0 THEN E=128
2089 PRINT "BLOAD",A$
2099 GOTO 1000
19999 REM  LOG TO DISK
20000 INPUT "SAVE COMPRESSED FILE TO DISKETTE? ",A$: IF A$#"Y" THEN 20030
20010 INPUT "NAME OF FILE? ",A$
20020 PRINT "BSAVE";A$;".COMP,A16384,L";NPTR: REM  VARIABLE NUMBER OF CHARACTERS
20025 PRINT "CREATED FILE: ";A$;".COMP"
20030 PRINT "CATALOG": END
30000 REM CREDIT SCREEN
30001 CALL HOME
30002 A$="SUP'R'TERMINAL FONT COMPRESSOR"
30004 B$="BY MORGAN P. CAFFREY"
30010 C$="P.O. BOX 31324"
30011 D$="SAN FRANCISCO, CA. 94131":E$="JANUARY 22, 1980"
30020 VTAB 10: TAB (39- LEN(A$))/2: PRINT A$: PRINT : PRINT
30022 TAB (39- LEN(B$))/2: PRINT B$: TAB (39- LEN(C$))/2: PRINT C$: TAB (39- LEN(D$))/2: PRINT D$
30023 TAB (39- LEN(E$))/2: PRINT E$
30025 PRINT : PRINT : PRINT "CLEARING CHARACTER SET SPACE"
30040 RETURN
```

Last Minute Notes

When using SUP'R'TERMINAL with the D.C. Hayes Micromodem II, the following constitutes a workable terminal (although the upper-lower case command will not work):

1. Turn power on.
2. Boot the diskette.
3. Type: PR#3 RETURN (initializes SUP'R'TERMINAL in slot #3).
4. Type: IN#2 RETURN (initializes Micromodem in slot #2).
5. Perform normal CTRL-A commands to initiate dialing. (notice here that the prompts will not appear on the screen; they are sent directly to the Apple II screen RAM.)
6. press RETURN several times. if the connection is made, the prompt will appear on the screen (cursor will not be present).

To obtain lower case display, exit the terminal mode, type: POKE 1786,0 (slot 2)'RETURN and then reenter terminal mode.

Again, Micromodem now supersedes SUP'R'TERMINAL CTRL-A sequence.

The following is a bare bones terminal program for those who do not wish to tie themselves to the Micromodem II firmware.

```
TERMINAL        LDA #$03        ; TERMINAL IN SLOT 3
                JSR $FE95       ; SET HOOKS FOR SLOT #3
                LDA #$8D        ; CARRIAGE RETURN
                JSR $FDED       ; COUT ROUTINE
*
                LDA $00         ; MODEM SLOT MUST BE IN LOCATION $00
                ASL             ; MOVE SLOT# TO HIGH NYBBLE
                ASL
                ASL
                ASL
                TAX             ; AND PUT IT IN X-REG
                LDA #$EF        ; 6850 SET-UP
                STA $C085,X     ; CONTROL REG
                LDA #$23        ;
                STA $C085,X     ; STATUS REG
                LDA #$11        ; 300 BAUD
                STA $C086,X     ; READ 6850 LITERATURE TO UNDERSTAND
*
* HERE IS THE MAIN LOOP
*
POLLKBD         LDA$C000        ; KEYBOARD
                BPL POLLMODEM   ; IF NOTHING, THEN CHECK MODEM
*
* WE HAVE SOMETHING SO SEND IT
*
                STA $C010       ; RESET KEY STROBE
                PHA             ; SAVE COPY
CHKREADY        LDA $C086,X     ; CHECK FOR XMITTER READY
                AND #$02        ; BIT 1 IS SIGNIFICANT HERE
                BEQ CHKREADY    ; WAIT IF NOT READY
                PLA             ; GET COPY BACK
                STA $C087,X     ; PUT IN TRANSMITTER BUFFER
*
* NOW POLL THE MODEM
*
```

```
POLLMODEM       LDA $C086,X     ; GET STATUS AGAIN
                LSR             ; MOVE BIT 0 TO CARRY FLAG
                BCC POLLKBD     ; IF CLEAR THEN NOTHING IN
                LDA $C087,X     ; GOT SOMETHING
                ORA #$80        ; ADD HIGH BIT
                JSR $FDED       ; SEND IT TO SCREEN
                JMP POLLKBD     ; NOW DO IT ALL AGAIN
```

The same routine, modified, will work well with the Communications card. Just change the status and data addresses in accordance with the information presented in the Communications card manual. Notice that there is no Control register on the Communications card as there is on the Micromodem II card. This is just a difference in firmware approach.

NOTE

The terminal programs provided above are *not* exhaustive. They present a beginning point for those interested in writing more sophisticated terminal software for themselves.

APPENDIX A - APPLE ASCII CODES

The normal range of American Standard Code for Information Interchange (ASCII) codes is from 0 to 127. The APPLE ASCII is what is known as 'negative ASCII'. Negative ASCII begins at 128 and goes to 255. The high bit of each character is set and the numbers are considered to be negative. This slight difference in representation has no effect on the meaning. 128 characters are coded into numerical representations.

The following table gives the normal and APPLE ASCII code in decimal and hexidecimal representation and the character represented. It also provides a quick guide to APPLE & SUP'R'TERMINAL meanings.

NOTE

The control characters from 0 to 31 have character names acquired from their original use in telecommunication systems.

ASCII DEC	NEG. ASCII HEX	CHAR DEC	TYPE: HEX	MEAN-ING	KEY	
0	$00	128	$80	NULL	CTRL@	HOME & CLEAR
1	$01	129	$81	SOH	CTRL-A	CASE "SWITCH''
2	$02	130	$82	STX	CTRL-B	ENTER BASIC
3	$03	131	$83	ETX	CTRL-C	STOP BASIC PROGRAM
4	$04	132	$84	ET	CTRL-D	D.O.S. FLAG
5	$05	133	$85	ENQ	CTRL-E	UNASSIGNED
6	$06	134	$86	ACK	CTRL-F	UNASSIGNED
7	$07	135	$87	BEL	CTRL-G	BELL
8	$08	136	$88	BS	CTRL-H	BACKSPACE
9	$09	137	$89	HT	CTRL-I	TAB
10	$0A	138	$8A	LF	CTRL-J	LINEFEED
11	$0B	139	$8B	VT	CTRL-K	LEFT BRACKET
12	$0C	140	$8C	FF	CTRL-L	HOME & CLEAR
13	$0D	141	$8D	CR	CTRL-M	RETURN
14	$0E	142	$8E	SO	CTRL-N	UNASSIGNED
15	$0F	143	$8F	SI	CTRL-O	UNASSIGNED
16	$10	144	$90	DLE	CTRL-P	UNASSIGNED
17	$11	145	$91	DC1	CTRL-Q	UNASSIGNED
18	$12	146	$92	DC2	CTRL-R	UNASSIGNED
19	$13	147	$93	DC3	CTRL-S	STOPLIST
20	$14	148	$94	DC4	CTRL-T	SPECIAL FUNCTION KEY
21	$15	149	$95	NAK	CTRL-U	RIGHT ARROW
22	$16	150	$96	SYN	CTRL-V	SHIFT KEY MOD. SWITCH
23	$17	151	$97	ETB	CTRL-W	UNASSIGNED
24	$18	152	$98	CAN	CTRL-X	CANCEL LINE
25	$19	153	$99	EM	CTRL-Y	HOME CURSOR
26	$1A	154	$9A	SUB	CTRL-Z	CLEARS LINE
27	$1B	155	$9B	ESC	ESCAPE	
28	$1C	156	$9C	FS	*	NOT AVAILABLE ON APPLE KEYBOARD
29	$1D	157	$9D	GS		CTRL-]

DEC	HEX	DEC	HEX			
30	$1E	158	$9E	RS	CTRL-^	
31	$1F	159	$9F	US	*	NOT AVAILABLE ON APPLE KEYBOARD CTRL-T MEANING
32	$20	160	$A0	space	space	0
33	$21	161	$A1	!	!	1
34	$22	162	$A2	"	"	2
35	$23	163	$A3	#	#	3
36	$24	164	$A4	$	$	4
37	$25	165	$A5	%	%	5
38	$26	166	$A6	&	&	6
39	$27	167	$A7	'	'	7
40	$28	168	$A8	(	(	8
41	$29	169	$A9	)	)	9
42	$2A	170	$AA	*	*	10
43	$2B	171	$AB	+	+	11
44	$2C	172	$AC	,	,	12
45	$2D	173	$AD	-	-	13
46	$2E	174	$AE	.	.	14
47	$2F	175	$AF	/	/	15
48	$30	176	$B0	0	0	16
49	$31	177	$B1	1	1	17
50	$32	178	$B2	2	2	18
51	$33	179	$B3	3	3	19
52	$34	180	$B4	4	4	20
53	$35	181	$B5	5	5	21
54	$36	182	$B6	6	6	22
55	$37	183	$B7	7	7	23
56	$38	184	$B8	8	8	24
57	$39	185	$B9	9	9	25
58	$3A	186	$BA	:	:	26
59	$3B	187	$BB	;	;	27
60	$3C	188	$BC	<	<	28
61	$3D	189	$BD	=	=	29
62	$3E	190	$BE	>	>	30
63	$3F	191	$BF	?	?	31
64	$40	192	$C0	@	@	32
65	$41	193	$C1	A	A	33
66	$42	194	$C2	B	B	34
67	$43	195	$C3	C	C	35
68	$44	196	$C4	D	D	36
69	$45	197	$C5	E	E	37
70	$46	198	$C6	F	F	38
71	$47	199	$C7	G	G	39
72	$48	200	$C8	H	H	40
73	$49	201	$C9	I	I	41
74	$4A	202	$CA	J	J	42
75	$4B	203	$CB	K	K	43
76	$4C	204	$CC	L	L	44
77	$4D	205	$CD	M	M	45
78	$4E	206	$CE	N	N	46
79	$4F	207	$CF	O	O	47
80	$50	208	$D0	P	P	48
81	$51	209	$D1	Q	Q	49
82	$52	210	$D2	R	R	50
83	$53	211	$D3	S	S	51
84	$54	212	$D4	T	T	52
85	$55	213	$D5	U	U	53

DEC	HEX	DEC	HEX			
86	$56	214	$D6	V	V	54
87	$57	215	$D7	W	W	55
88	$58	216	$D8	X	X	56
89	$59	217	$D9	Y	Y	57
90	$5A	218	$DA	Z	Z	58
91	$5B	219	$DB	{	{	59
92	$5C	220	$DC	\	\	60
93	$5D	221	$DD	]	]	61
94	$5E	222	$DE	~	~	62
95	$5F	223	$DF	_	_	63
96	$60	224	$E0	sp.	sp.	64
97	$61	225	$E1	a	a	65
98	$62	226	$E2	b	b	66
99	$63	227	$E3	c	c	67
100	$64	228	$E4	d	d	68
101	$65	229	$E5	e	e	69
102	$66	230	$E6	f	f	70
103	$67	231	$E7	g	g	71
104	$68	232	$E8	h	h	72
105	$69	233	$E9	i	i	73
106	$6A	234	$EA	j	j	74
107	$6B	235	$EB	k	k	75
108	$6C	236	$EC	l	l	76
109	$6D	237	$ED	m	m	77
110	$6E	238	$EE	n	n	78
111	$6F	239	$EF	o	o	79
112	$70	240	$F0	p	p	80
113	$71	241	$F1	q	q	
114	$72	242	$F2	r	r	
115	$73	243	$F3	s	s	
116	$74	244	$F4	t	t	
117	$75	245	$F5	u	u	
118	$76	246	$F6	v	v	
119	$77	247	$F7	w	w	
120	$78	248	$F8	x	x	
121	$79	249	$F9	y	y	
122	$7A	250	$FA	z	z	
123	$7B	251	$FB	{	{	
124	$7C	252	$FC	|	|	
125	$7D	253	$FD	}	}	
126	$7E	254	$FE	~	~	
127	$7F	255	$FF	DEL	?	

CONTROL (''CTRL'') CHARACTERS

KEY	HEX	DECIMAL	FUNCTION
CTRL-A	$81	129	Upper-Lower case switch
CTRL-B	$82	130	Unassigned
CTRL-C	$83	131	Keyboard Program Interrupt
CTRL-D	$84	132	D.O.S Attention flag
CTRL-E	$85	133	Unassigned
CTRL-F	$86	134	Unassigned
CTRL-G	$87	135	Bell
CTRL-H	$88	136	Backspace
CTRL-I	$89	137	Tab to next multiple of 8
CTRL-J	$8A	138	Line feed
CTRL-K	$8B	139	Left bracket
CTRL-L	$8C	140	Form feed (home and clear)
CTRL-M	$8D	141	Carriage return (generates a line feed).
CTRL-N	$8E	142	Unassigned
CTRL-O	$8F	143	Unassigned
CTRL-P	$90	144	Unassigned
CTRL-Q	$91	145	Unassigned
CTRL-R	$92	146	Unassigned
CTRL-S	$93	147	STOPLIST
CTRL-T	$94	148	Special function mode
--CTRL-TF char			- ''load'' compressed font into character RAM.
	CHAR RANGE		
0	$B0	176	From $1000 - $13FF
1	$B1	177	From $1400 - $17FF
2	$B2	178	From $1800 - $1BFF
3	$B3	179	From $1C00 - $1FFF
4	$B4	180	From $2000 - $23FF
5	$B5	181	From $2400 - $27FF
6	$B6	182	From $2800 - $2BFF
7	$B7	183	From $2C00 - $2FFF
8	$B8	183	From $3000 - $33FF
9	$B9	184	From $3400 - $37FF
--CTRL-TR			Cold Start reset
--CTRL-TC n			Alter Cursor Size & Flash rate
	NUM-BER	RANGE	
0	$B0	176	Underline - 4 flashes per sec.
1	$B1	177	Box - 4 flash per sec.
2	$B2	178	Underline - no flash.
3	$B3	179	Box - no flash.
4	$B4	180	Underline - 2 flashes per sec.
5	$B5	181	Box - 2 flashes per sec.
6	$B6	182	No cursor displayed.
--CTRL-TT ch			Top of window (0 - 23)
--CTRL-TB ch			Bottom of window (0 - 23)
--CTRL-TW ch			Width of window (0 - 79)
--CTRL-TL ch			Absolute left margin (0 - 79)
CTRL-U	$95	149	Right arrow (BASIC ONLY)
CTRL-V	$96	150	Switch TO or FROM shift-key modification mode
CTRL-W	$97	151	Unassigned
CTRL-X	$98	152	Cancels Input line
CTRL-Y	$99	153	Home cursor (doesn't blank screen)

KEY	HEX	DECIMAL	FUNCTION
CTRL-Z	$9A	154	Clears line
*	$9C	156	(non-copy) forward space
CTRL-SHIFT-N	$9E	158	GOTO XY (X = HORIZONTAL, Y = VERTICAL) absolute cursor positioning)
*	$9F	159	Cursor up

* Not available from keyboard. Can be used only as part of a program.

ESCAPE ("ESC") COMMANDS (BASIC only)

ESC @	Clears screen and homes cursor -- immediate mode only.
ESC A	Non-copy forward space
ESC B	Backspace
ESC C	Line feed
ESC D	Reverse line feed
ESC E	Clear to end of line
ESC F	Clear to end of screen
ESC I	Moves cursor up one line and locks screen into cursor-move mode
ESC J	Moves cursor left one space and locks screen as does ESC I
ESC K	Moves cursor right one space and locks screen as does ESC I
ESC M	Moves cursor down one line and locks screen as does ESC I
ESC Q	Exits SUP'R'TERMINAL mode

NOTE

The I, J, K, and M keys form a diamond which points the direction of cursor movement:

I

J **K**

M

NOTE: THE BELOW LISTED PROGRAMS ARE AVAILABLE ON DISK FROM YOUR LOCAL COMPUTER STORE.

Keypress Note

Pascal users will soon see that KEYPRESS (a function used in applestuf) doesn't seem to work properly. Don't fret, here's the solution. The following is a listing of required additions to your Pascal program to allow KEYPRESS.

Lines 30 through 38 should be added to your GLOBAL VARIABLES.

Lines 41 through 70 go into your program before the KEYPRESS replacement function is used.

Replace all your existing **KEYPRESS** functions with the word **KEY.** If your not using applestuf, you can leave the function named KEYPRESS as is, although lines 30 through 38 and 41 through 70 must still be typed in with line 60 changed to **KEYPRESS** where it now says **KEY.**

```
 1   1    1:D  1 (*$LPRINTER:*)
 2   1    1:D  1 program KEYTEST;
 3   1    1:D  3
 4   1    1:D  3 (  ********************************************************************************************************************************
 5   1    1:D  3 *                                                                                                                              *
 6   1    1:D  3*  Example program for M&R enterprises.                                                                                         *
 7   1    1:D  3 *                                                                                                                              *
 8   1    1:D  3 *  It shows how one may replace the APPLESTUFF                                                                                 *
 9   1    1:D  3 *  function KEYPRESS when using an M&R                                                                                         *
10   1    1:D  3 *  Sup'r'terminal board.                                                                                                       *
11   1    1:D  3 *                                                                                                                              *
12   1    1:D  3 *                                                  By Dan Sokol—6 May 80                                                       *
13   1    1:D  3 *                                                                                                                              *
14   1    1:D  3 ********************************************************************************************************************************* )
15   1    1:D  3
16  22    1:D  3 [$  ]
17  22    1:D  3
18  22    1:D  3
19  22    2:D  3       FUNCTION PADDLE (SELECT: INTEGER) : INTEGER;
20  22    3:D  3       FUNCTION BUTTON (SELECT: INTEGER) : BOOLEAN;
21  22    4:D  1       PROCEDURE TTLOUT (SELECT: INTEGER: DATA: BOOLEAN);
22  22    5:D  3       FUNCTION KEYPRESS: BOOLEAN:
23  22    6:D  3       FUNCTION RANDOM: INTEGER;
24  22    7:D  1       PROCEDURE RANDOMIZE
25  22    8:D  1       PROCEDURE NOTE (PITCH,DURATION: INTEGER);
26  22    8:D  3
27  22    1:D  3       IMPLEMENTATION
28  22    1:D  1
29   1    1:D  1 uses applestuff;
30   1    1:D  3
31   1    1:D  3 type PA = packed array [0..1] of 0..255;
32   1    1:D  3      MAGIC = record case booolean of
33   1    1:D  3                  true : (INT: integer);
34   1    1:D  3                  false : (PTR: ^ PA) ;
35   1    1:D  3                  end;
36   1    1:D  3
37   1    1:D  3 var CHEAT : MAGIC;
38   1    1:D  4     PITCH,DURATION,DATA : integer;
39   1    1:D  7
40   1    2:D  1 procedure TEST;
41   1    2:0  0   begin
42   1    2:1  0    if DATA < 0 then DATA:=-DATA;
43   1    2:1  9    while DATA > 255 do DATA:=DATA-256;
44   1    2:0 25   end;
45   1    2:0 40
46   1    3:D  1 procedure POKE(ADDR,DATA : INTEGER);
47   1    3:0  0   begin
48   1    3:1  0    TEST;
49   1    3:1  2    CHEAT.INT:=ADDR;
50   1    3:1  5   CHEAT.PTR ^ [0]:=DATA;
```

```
51   1    3:0  17   end;
52   1    3:0  30
53   1    4:D   3 function PEEK (ADDR : integer) : integer;
54   1    4:0   0   begin
55   1    4:1   0     CHEAT.INT:=ADDR;
56   1    4:1   3     PEEK:=CHEAT.PTR  ^ [0];
57   1    4:0  11   end;
58   1    4:0  24
59   1    4:0  24 (*$P*)
60   1    5:D   3 function KEY : boolean; (* Don't call it KEYPRESS if your using applestuff
61   1    5:D   3   var CLEAR ,KEYBOARD,TEMP : integer;
62   1    5:0   0   begin
63   1    5:1   0     CLEAR:=-16368; KEYBOARD: =-16384; TEMP:=peek (KEYBOARD);
64   1    5:1  19     if TEMP > 128 then
65   1    5:2  26       begin
66   1    5:3  26       KEY : =true;
67   1    5:3  29       poke(CLEAR,TEMP);
68   1    5:2  33     end else
69   1    5:2  35     KEY:=false;
70   1    5:0  38   end;
71   1    5:0  50
72   1    5:0  50
73   1    5:0  50 (* main program—to test KEY function *)
74   1    1:0   0 begin
75   1    1:1   0 writeln ('Press any key to start');
76   1    1:1  44 repeat until KEY; (* loop here till key is pressed *)
77   1    1:1  50 writeln ('Press any key to stop.');
78   1    1:1  92 repeat
79   1    1:2  92   PITCH:=PITCH+1 mod 30;
80   1    1:2  99   DURATION:=DURATION+1 mod 10;
81   1    1:2 106   note(PITCH, DURATION);
82   1    1:1 111 until KEY;                    (* sound off till KEY is pressed *)
83   1    1:0 117 end.
```

Automatic Program Converter for Existing Applesoft Programs

To allow your SUP'R'TERMINAL to work with existing Applesoft Basic Programs, you must change VTAB statements, HOME, and CALL-936 statements as these cannot be interpreted by SUP'R'TERMINAL. The following programs will do this automatically, enabling you to make this as easy as possible.

Type in and save the following 5 short Applesoft programs. Make sure you name each one exactly, as it appears, as the programs must call each other up while running under program control. Once you have the programs on disc, you must put any program you wish to convert on the same disc.

To run the program conversion, just type **RUN VTAB** followed by a RETURN. The program will ask you for the name of the program you wish to convert. When it is through converting, the converted program is automatically stored in RAM as an Applesoft Basic Program. You may save it on disk as you do any other Basic program. Type **SAVE** followed by the name you wish to call it.

Please note that the programs you are converting were initially written with a 40 column display in mind. The display may not yet be optimized for 80 characters. Also remember that any direct references, within programs, to the Apple screen (other than VTAB, CALL-936, and HOME instructions) will still only reference the Apple screen rather than SUP'R'TERMINAL. These more complicated references may still be modified by direct reference to SUP'R'TERMINAL screen rams (see manual or call the factory for assistance if you have any difficulty).

VTAB

```
 10 REM DOCUMENTATION
 20 D$ = CHR$ (4)
 30 PRINT D$; "PR#3"
 35 PRINT D$; "MAXFILES 5": PRINT: PRINT D$
 40 PRINT CHR$ (140)
 50 PRINT "This program converts VTAB statements in APPLSOFT programs"
 60 PRINT "to the correct CTRL-SHIFT-N sequence for use with the M&R"
 70 PRINT "Enterprises SUP'R'TERMINAL board."
 80 PRINT
 90 PRINT "The following limitations must be observed!"
100 PRINT "1. The program to be converted MUST be an APPLESOFT (not INTEGER) program."
110 PRINT "2. The program must be on this disk."
120 PRINT "3. The program cannot have any line numbers above 32749."
130 PRINT "4. There must be room on this disk for 2 ascii copies of the program."
140 PRINT
150 PRINT "The conversion is not very fast . . . but it works!"
160 PRINT "After the conversion is complete the new program is in memory,"
170 PRINT "IT IS UP TO YOU TO SAVE IT!!!! Don't forget!!!."
180 PRINT
190 PRINT "Type return if you are ready to convert, space to exit."
200 GET A$
205 D$ = CHR$ (4): PRINT
210 IF A$ = CHR$ (13) THEN PRINT D$; "RUN FIXVTABS"
220 END
```

FIXVTABS

```
 10 D$ = CHR$ (4)
 20 PRINT D$; "OPEN TEMP"
 30 PRINT D$; "DELETE TEMP"
 40 PRINT "ENTER NAME OF PROGRAM TO BE CONVERTED .... ";: INPUT A$
 50 PRINT D$: "NOMON C,I,O"
 60 PRINT D$: "EXEC XFERTOTEXT"
 70 END
```

XFERTOTEXT

```
PRINT D$; "LOAD ";A$
32750 D$=CHR$ (4)
32751 PRINT D$;"OPEN TEMP"
32752 PRINT D$; "DELETE TEMP"
32753 PRINT D$; "OPEN TEMP"
32754 PRINT D$; "WRITE TEMP"
32755 LIST 0-32749
32756 PRINT CHR$(27)+CHR$(27)+CHR$(27)
32757 PRINT D$; "CLOSE TEMP"
RUN 32750
RUN CONVERT
```

LIST CONVERT

```
 10 D$ = CHR$ (4)
 20 PRINT D$;" MON C,I,O"
 30 PRINT D$;"OPEN NEWPROG"
 40 PRINT D$;"DELETE NEWPROG"
 50 PRINT D$;"OPEN NEWPROG"
 60 PRINT D$;"OPEN TEMP"
 70 PRINT D$;"READ TEMP"
 80 GET A$
 90 C$ = C$ + A$
100 IF MID$ (C$,2,3) = CHR$ (27) + CHR$ (27) + CHR$ (27) THEN GOTO 310
110 IF A$ < > CHR$ (13) THEN GOTO 80
120 FOR I = 1 TO LEN (C$)
130 IF MID$ (C$,I,4) = "VTAB" THEN GOSUB 180
140 B$ = B$ + MID$ (C$,I,1)
150 NEXT I
160 PRINT D$: PRINT D$; "WRITE NEWPROG": PRINT B$
```

```
170 B$ = "   ":C$ = "   ": GOTO 70
180 REM CONVERT VTABS HERE
190 P = 0
200 X$ = "PRINT CHR$(158)+CHR$(32+POS(0))+CHR$(32+"
210 I = I + 4
220 B$ = B$ + X$
230 FOR K = I TO LEN (C$)
235 IF MID$ (C$,K,1) = CHR$ (13) THEN GOTO 270
240 IF MID$ (C$,K,1) < > ":" THEN B$ = B$ + MID$ (C$,K,1):P = P + 1
250 IF MID$ (C$,K,1) = ":" THEN GOTO 270
260 NEXT K
270 I = I + P
280 B$ = B$ + ");"
290 PRINT D$: PRINT "****** FOUND ONE *****";
300 RETURN
310 PRINT D$;"CLOSE"
320 TEXT
330 PRINT D$;"EXEC TEXTOPROG"
340 END
```

TEXTOPROG

```
NEW
D$=CHR$(4)
PRINT D$;"EXEC NEWPROG"
```

Procedure For Keying In Programs To Run with D C Hayes

Type in and save **AUTODIAL** Program with your selected phone number in line 30. Type in and save the program **OBJECT MAKER.** Then run **OBJECT MAKER.** This program you just ran will create the Binary File called **TERMINAL.OBJ** which is needed for running **AUTO-DIAL** with the D C Hayes. To run the D C Hayes, put the D C Hayes card in slot 2 and type RUN AUTODIAL.

Procedure For Keying In Program To Run with the Comm. Card.

Type in and save the Program called **TERMINAL.** Put the Comm Card in slot 2 and type RUN TERMINAL.

Also there is a listing to give you an idea of the program flow and explain the options, if you want to put in the 1 wire Shift Key modification (see Manual). Without the wire modification for the Shift Key, you won't get lower case, special characters, or half duplex.

AUTODIAL

```
10 Q$ = CHR$ (17) :D$ = CHR$ (4)
15 PRINT D$;"BLOAD TERMINAL.OBJ"
20 PRINT D$;"NOMON C,I,O"
30 DATA XXX-XXXX : REM INSERT PHONE NUMBER IN PLACE OF X'S
40 PRINT D$;"PR#2"
45 READ NU$
50 PRINT Q$;"   "NU$
60 IF PEEK (1658) < 128 THEN GOTO 80
70 CALL 16384
80 END
```

TERMINAL

```
 10 LOMEM: 17000
 20 DATA 173,255,207,169,0,141,122,4,141
 30 DATA 122,5,141,250,4,141,122,6,141
 40 DATA 250,6,141,250,5,169,24,205,0
 50 DATA 194,240,12,169,3,141,174,192,169
 60 DATA 17,141,174,192,208,18,169,143,141
 70 DATA 165,192,169,3,141,166,192,169,17
 80 DATA 141,166,192,141,250,5,169,32,141
 90 DATA 122,7,169,3,32,149,254,173,0
100 DATA 192,16,91,141,16,192,201,159,144
110 DATA 123,201,192,144,22,201,218,176,18
120 DATA 72,173,122,6,240,11,173,99,192
130 DATA 240,6,104,24,105,32,144,1,104
140 DATA 72,173,250,4,208,94,104,41,127
150 DATA 72,173,250,5,240,14,173,166,192
160 DATA 41,2,240,249,104,141,167,192,76
170 DATA 148,64,173,174,192,41,2,240,249
180 DATA 104,141,175,192,72,173,122,4,240
190 DATA 11,104,32,47,65,9,128,32,237
200 DATA 253,48,1,104,32,78,65,173,250
210 DATA 5,240,11,173,166,192,74,144,146
220 DATA 173,167,192,176,9,173,174,192,74
230 DATA 144,135,173,175,192,32,47,65,9
240 DATA 128,32,237,253,76,70,64,76,236
250 DATA 64,104,201,222,240,11,201,192,240
260 DATA 11,201,221,240,11,76,115,64,169
270 DATA 206,208,249,169,208,208,245,169,205
280 DATA 208,241,72,173,99,192,240,10,104
290 DATA 201,139,208,2,169,219,76,108,64
300 DATA 104,201,150,208,11,169,255,77,250
310 DATA 4,141,250,4,76,70,64,201,148
320 DATA 208,11,169,255,77,122,4,141,122
330 DATA 4,76,70,64,201,129,208,11,169
340 DATA 255,77,122,6,141,122,6,76,70
350 DATA 64,201,145,208,203,96,72,41,127
360 DATA 201,32,144,2,104,96,173,122,5
370 DATA 240,249,169,160,32,237,253,169,136
380 DATA 32,237,253,169,0,141,122,5,240
390 DATA 232,72,206,250,6,208,226,206,122
400 DATA 7,208,221,169,32,141,122,7,173
410 DATA 122,5,73,255,141,122,5,240,7
420 DATA 169,223,32,237,253,208,5,169,160
430 DATA 32,237,253,169,136,32,237,253,104
440 DATA 96,0,0,0,0,0,0,0,0
450 DATA 0,132,1,0,
1000 START = 16384:SIZE = 390
1010 RESTORE
1020 FOR I = START TO START + SIZE
1030 READ D
1040 POKE I,D
1050 NEXT I
1055 HOME : PR# 3: PRINT CHR$ (140);
1060 CALL START
1070 END
```

OBJECTMAKER

```
 10 LOMEM: 17000
 20 DATA 173,255,207,169,0,141,122,4,141
 30 DATA 122,5,141,250,4,141,122,6,141
 40 DATA 250,6,141,250,5,169,24,205,0
 50 DATA 194,240,12,169,3,141,174,192,169
 60 DATA 17,141,174,192,208,18,169,143,141
 70 DATA 165,192,169,3,141,166,192,169,17
 80 DATA 141,166,192,141,250,5,169,32,141
 90 DATA 122,7,169,3,32,149,254,173,0
100 DATA 192,16,91,141,16,192,201,159,144
110 DATA 123,201,192,144,22,201,218,176,18
```

```
120 DATA 72,173,122,6,240,11,173,99,192
130 DATA 240,6,104,24,105,32,144,1,104
140 DATA 72,173,250,4,208,94,104,41,127
150 DATA 72,173,250,5,240,14,173,166,192
160 DATA 41,2,240,249,104,141,167,192,76
170 DATA 148,64,173,174,192,41,2,240,249
180 DATA 104,141,175,192,72,173,122,4,240
190 DATA 11,104,32,47,65,9,128,32,237
200 DATA 253,48,1,104,32,78,65,173,250
210 DATA 5,240,11,173,166,192,74,144,146
220 DATA 173,167,192,176,9,173,174,192,74
230 DATA 144,135,173,175,192,32,47,65,9
240 DATA 128,32,237,253,76,70,64,76,236
250 DATA 64,104,201,222,240,11,201,192,240
260 DATA 11,201,221,240,11,76,115,64,169
270 DATA 206,208,249,169,208,208,245,169,205
280 DATA 208,241,72,173,99,192,240,10,104
290 DATA 201,139,208,2,169,219,76,108,64
300 DATA 104,201,150,208,11,169,255,77,250
310 DATA 4,141,250,4,76,70,64,201,148
320 DATA 208,11,169,255,77,122,4,141,122
330 DATA 4,76,70,64,201,129,208,11,169
340 DATA 255,77,122,6,141,122,6,76,70
350 DATA 64,201,145,208,203,96,72,41,127
360 DATA 201,32,144,2,104,96,173,122,5
370 DATA 240,249,169,160,32,237,253,169,136
380 DATA 32,237,253,169,0,141,122,5,240
390 DATA 232,72,206,250,6,208,226,206,122
400 DATA 7,208,221,169,32,141,122,7,173
410 DATA 122,5,73,255,141,122,5,240,7
420 DATA 169,223,32,237,253,208,5,169,160
430 DATA 32,237,253,169,136,32,237,253,104
440 DATA 96,0,0,0,0,0,0,0,0
450 DATA 0,132,1,0,
1000 START = 16384:SIZE = 390
1010 RESTORE
1020 FOR I = START TO START + SIZE
1030 READ D
1040 POKE I,D
1050 NEXT I
1055 D$ = CHR$(4)
1060 PRINT D$;"BSAVE TERMINAL.OBJ, A$4000,L$1FF"
1070 END
```

Listing To Give You An Idea of the Program Flow

This program emulates a moderately dumb terminal using an M&R Enterprises Sup'r'terminal and an APPLE Communications Card (or Micromodem II). It has the following features:

1. 80 characters by 24 lines.
2. Upper and Lower case with caps lock.
3. Half or Full Duplex switch.
4. Special character switch.
5. It works with either a Comm Card or a D.C. Hayes Micromodem.

The program expects the Sup'r'terminal to be in slot #3 and the Comm Card (or modem) to be in slot #2. The program avoids using the X and Y registers to minimize interference with the Sup'r'terminal. The shift key modification described in the Sup'r'terminal manual must have been made. If it has not been made, the funcion keys will not work.

The program functions are accessed as follows:

1. CTRL—SHIFT—A ..... Shift lock (caps only). Default = upper case only.
2. CTRL—SHIFT—V ..... Special Characters ([ ] @). Default =+ off.
3. CTRL—SHIFT—T ..... Duplex switch. Default = Full Duplex (no local echo).
4. CTRL—SHIFT—Q ..... Quit program. Returns control to calling program.

```
0000 |  0002          SLOT      .EQU   2              ; THE LOCATION OF THE COMM CARD (OR MODEM)
0000 |  0020          SLOTCN    .EQU   SLOT * 10      ; RAM OFFSET CALCULATOR
0000 |  CFFF          CFFF      .EQU   0CFFF          ; C800 ROM DISABLE ADDRESS
0000 |  C000          KEYSTAT   .EQU   0C000          ; APPLE KEYBOARD AND STATUS PORT ADDRESS
0000 |  C010          KEYCLR    .EQU   0C010          ; CLEAR KEYBOARD STROBE
0000 |  FE95          SETUP     .EQU   0FE95          ; APPLE MONITOR PR#N ROUTINE (N IN A REGISTER)
0000 |  FDED          CHROUT    .EQU   0FDED          ; APPLE MONITOR CHARACTER OUT ROUTINE (CHARACTER IN A)
0000 |  C063          SWIT2     .EQU   0C063          ; SHIFT KEY STATUS (0 WHEN DOWN)
0000 |  047A          DUPLEX    .EQU   478+SLOT       ; DUPLEX STATUS FLAG (DEFAULT = FULL = 0)
0000 |  04FA          SPECIAL   .EQU   4F8+SLOT       ; SPECIAL CHARACTER FLAG (DEFAULT = OFF = 0)
0000 |  057A          CURSTAT   .EQU   578+SLOT       ; CURSOR STATUS (ON/OFF)
0000 |  05FA          CARD      .EQU   5F8+SLOT       ; CARD TYPE (0 = COMM CARD)
0000 |  067A          UPCASE    .EQU   678+SLOT       ; CAPS LOCK FLAG (DEFAULT = ON = 0)
0000 |  06FA          BLINK1    .EQU   6F8+SLOT       ; CURSOR BLINK DELAY
0000 |  077A          BLINK2    .EQU   778+SLOT       ; MORE DELAY
0000 |  0011          BAUDRTE   .EQU   11             ; 11 = 300 BAUD, 52 = 110 BAUD
0000 |  C0A5          MRESET    .EQU   0C085+SLOTCN   ; MODEM RESET PORT
0000 |  C0A6          MSTAT     .EQU   MRESET+1       ; MODEM STATUS PORT
0000 |  C0A7          MCTRL     .EQU   MSTAT          ; MODEM CONTROL PORT
0000 |  C0A7          CREAD     .EQU   MSTAT+1        ; MODEM DATA IN PORT
0000 |  C0AE          MWRITE    .EQU   MSTAT+1        ; MODEM DATA OUT PORT
0000 |  C0A3          CSTAT     .EQU   0C08E+SLOTCN   ; COMM CARD STATUS PORT
0000 |  C0AF          CCTRL     .EQU   CSTAT          ; COMM CARD CONTROL PORT
0000 |  C0AF          CREAD     .EQU   CSTAT+1        ; COMM CARD DATA IN PORT
0000 |  C0AF          CWRITE    .EQU   CSTAT+1        ; COMM CARD DATA OUT PORT
0000 |                          ; ACIA STATUS PORT — BIT DEFINITIONS
0000 |                          ; BIT — MEANING
0000 |                          ;  0   INPUT DATA READY (TRUE=1)
0000 |                          ;  1   TRANSMIT BUFFER EMPTY (TRUE=1)
0000 |                          ;  2   CARRIER DETECT FAILURE (WHO CARES)
0000 |                          ;  3   CLEAR TO SEND
0000 |                          ;  4   FRAMING ERROR
0000 |                          ;  5   RECEIVER OVERRUN
0000 |                          ;  6   PARITY ERROR
0000 |                          ;  7   IRQ
0000 |                          ;
0000 |                          ;  I'M NOT USING BITS 2 THRU 7 (THIS IS A VERY DUMB TERMINAL).
0000 |                          .ORG 4000
4000 |
4000 |  AD FFCF       START     LDA    CFFF           ; ALL C800 ROMS OFF
4003 |  A9 00                   LDA    #0             ; GET READY TO SETUP DEFAULTS
4005 |  8D 7A04                 STA    DUPLEX         ; FULL DUPLEX
4008 |  8D 7A05                 STA    CURSTAT        ;CURSOR OFF
400B |  8D FA04                 STA    SPECIAL        ; SPECIAL CHARACTERS OFF
400E |  8D 7A06                 STA    UPCASE         ; UPPER CASE ONLY
4011 |  8D FA06                 STA    BLINK1         ; CLEAR CURSOR DELAY COUNTER
4014 |  8D FA05                 STA    CARD           ; IF IT'S NOT A COMM CARD WE FIX LATER
4017 |  A9 18                   LDA    #18            ; WHAT'S ON SECOND?
4019 |  CD 00C2                 CMP    100*SLOT+0C000 ; LOCATION CX00 IS 18 FOR THE MODEM, 2C FOR THE COMM CARD
401C |  F00C                    BEQ    MINIT          ; INITIALIZE FOR THE MODEM
401E |  A9 03                   LDA    #03            ; ELSE, RESET THE COMM CARD
4020 |  8D AEC0                 STA    CCTRL          ; COMM CARD RESET
4023 |  A9 11                   LDA    #BAUDRTE       ; BAUD RATE
4025 |  8D AEC0                 STA    CCTRL          ; GETS SET HERE
4028 |  D012                    BNE    INIT           ; FORCED BRANCH AROUND MODEM INITIALIZATION
402A |  A9 8F         MINIT     LDA    #8F            ; MODEM HARDWARE RESET, OFF HOOK, ORIGINATE,
                                                       XMIT ON 300 BAUD
402C |  8D A5C0                 STA    MRESET         ; RESETS THE MODEM
402F |  A9 03                   LDA    #03            ; NEEDED TO RESET THE ACIA ON THE MODEM CARD
4031 |  8D A6C0                 STA    MCTRL
4034 |  A9 11                   LDA    #BAUDRTE       ; SAME AS COMM CARD
4036 |  8D A6C0                 STA    MCTRL          ; SETS CHARACTER LENGTH (8 BITS, NO PARITY, 2 STOP BITS)
4039 |  8D FA05                 STA    CARD           ; ANY NON-ZERO BYTE = MODEM
403C |  A9 20         INIT      LDA    #20            ; FOR CURSOR BLINK DELAY
403E |  8D 7A07                 STA    BLINK2
4041 |  A9 03                   LDA    #3             ; DO A PR#3 FOR SUP'R'TERMINAL
4043 |  20 95FE                 JSR    SETUP
4046 |
4046 |                  ; THE KEYBOARD INPUT ROUTINE STARTS HERE. IT IS INDEPENDENT OF THE TYPE OF
4046 |                  ; BOARD IN SLOT#2. THE PROGRAM SPENDS MOST OF IT'S TIME GOING BACK AND FORTH
4046 |                  ; BETWEEN HERE AND THE INCHAR ROUTINE.
4046 |
```

```
4046 | AD 00C0    KEY       LDA   KEYSTAT   ; KEYBOARD STATUS > 127 MEANS CHARACTER READY
4049 | 105B                 BPL   INCHAR    ; IF < 128 CHECK FOR INCOMING DATA
404B | 8D 10C0              STA   KEYCLR    ; GOT ONE! , CLEAR KEYBOARD STROBE
404E | C9 9F                CMP   #9F       ; CHECK FOR CTRL CHARACTERS
4050 | 907B                BCC   STEST     ; TEST FOR SPECIAL CTRL CHARACTERS
4052 | C9 C0                CMP   #0C0      ; IS IT < THEN CAP 'A'
4054 | 9016                BCC   NOFIX
4056 | C9 DA                CMP   #0DA      ; IS IT > THEN CAP 'Z'
4058 | B012                BCS   NOFIX
405A | 48        FIX       PHA             ; SAVE THE CHARACTER
405B | AD 7A06              LDA   UPCASE    ; DO WE ALLOW LOWER CASE?
405E | F00B                BEQ   FIXED     ; NO, BAIL OUT NOW
4060 | AD 63C0              LDA   SWIT2     ; IS SHIFT KEY DOWN? (VIA SUP'R'TERMINAL MOD)
4063 | F006                BEQ   FIXED     ; YES, BAIL OUT
4065 | 68                  PLA             ; GET THE CHARACTER BACK
4066 | 18                  CLC             ; PREPARE TO MAKE LOWER CASE
4067 | 69 20               ADC   #20       ; UPPER CASE + 20 = LOWER CASE
4069 | 9001                BCC   NOFIX     ; FORCED BRANCH (CARRY IS 0)
406B | 68        FIXED     PLA             ; RECOVER CHARACTER ITS FINE AS IS
406C | 48        NOFIX     PHA             ; SAVE WHILE WE TEST FOR SPECIAL CHARACTERS
406D | AD FA04              LDA   SPECIAL
4070 | D05E                BNE   SPECON    ; SPECIAL CHARACTERS ARE ON
4072 | 68                  PLA             ; RECOVER CHARACTER
4073 | 29 7F     SPRET     AND   #7F       ; REMOVE MSB
4075 | 48                  PHA             ; SAVE ON STACK
4076 | AD FA05   XMIT      LDA   CARD      ; WHO'S ON FIRST (NO, WHO'S ON SECOND)
4079 | F00E                BEQ   COMOUT    ; COMM CARD OUTPUT ROUTINE
407B | AD A6C0   MLOOP     LDA   MSTAT
407E | 29 02               AND   #02       ; XMIT BUFFER EMPTY?
4080 | F0F9                BEQ   MLOOP     ; WAIT TILL IT IS
4082 | 68                  PLA             ; RECOVER CHARACTER
4083 | 8D A7C0             STA   MWRITE    ; ZIP, IT'S GONE
4086 | 4C 9440             JMP   DLX       ; GO CHECK DUPLEX STATUS
4089 | AD AEC0   COMOUT    LDA   CSTAT
408C | 29 02               AND   #02       ; LOOKS FAMILIAR, HUH ...
408E | FoF9                BEQ   COMOUT    ; IT'S THE SAME AS MLOOP
4090 | 68                  PLA             ; WITH DIFFERENT ADDRESSES
4091 | 8D AFC0             STA   CWRITE
4094 | 48        DLX       PHA             ; SAVE THE CHARACTER (AGAIN)
4095 | AD 7A04             LDA   DUPLEX
4098 | F00B                BEQ   FULL      ; FULL DUPLEX , SKIP THE REST
409A | 68                  PLA
409B | 20 2F41             JSR   NOCTRL    ; REMOVES CURSOR IF CHAR IS CTRL.
409E | 09 80               ORA   #80       ; PUT MSB BACK FOR LOCAL USE
40A0 | 20 EDFD             JSR   CHROUT
40A3 | 3001                BMI   INCHAR    ; FORCED BRANCH (N=1)
40A5 | 68        FULL      PLA             ; FIX STACK
40A6 |
40A6 |            ; INCHAR GETS CHARACTER FROM SLOT #2 AND PUTS THEM ON THE
40A6 |            ; SCREEN. IF THERE'S NO CHARACTER IT GOES BACK TO KEY.
40A6   20 4E41    INCHAR    JSR   CBLINK    ; MAKE THE CURSOR BLINK
40A9   AD FA05              LDA   CARD      ; WHERE'S THE STATUS PORT?
40AC   F00B                 BEQ   COMIN     ; 0 = COMM CARD
40AE   AD A6C0    MODIN     LDA   MSTAT     ; MUST BE THE MODEM
40B1   4A                   LSR   A         ; MOVE INTO CARRY
40B2   9092                 BCC   KEY       ; NO CHARACTER, START OVER
40B4   AD A7C0              LDA   MREAD     ; GET THE CHARACTER
40B7   B009                 BCS   DISPLAY   ; FORCED BRANCH (C = 1)
40B9   AD AEC0    COMIN     LDA   CSTAT     ; COMM CARD DATA INPUT ROUTINE
40BC   4A                   LSR   A         ; WHICH IS THE SAME AS ABOVE
40BD   9087                 BCC   KEY
40BF   AD AFC0              LDA   CREAD
40C2 | 20 2F41    DISPLAY   JSR   NOCTRL    ; REMOVES CURSOR IF CTRL CHAR
40C5 | 09 80               ORA   #80       ; ADD SMB FOR APPLE
40C7 | 20 EDFD             JSR   CHROUT
40CA | 4C 4640             JMP   KEY       ; BACK TO THE BEGINNING
40CD |            ; THE SUBROUTINES FOLLOW .....
40CD |
40CD | 4C EC40    STEST     JMP   STEST1    ; EXTEND THE BRANCH
40D0 |
40D0 | 68         SPECON    PLA             ; RECOVER CHARACTER
40D1 | C9 DE                CMP   #0DE      ; SHIFT-N
40D3 | F00B                 BEQ   CAPN
40D5 | C9 C0                CMP   #0C0      ; SHIFT-P
40D7 | F00B                 BEQ   CAPP
40D9 | C9 DD                CMP   #0DD      ; SHIFT-M
40DB | F00B                 BEQ   CAPM
```

```
40DD | 4C 7340      SP1      JMP   SPRET      ; NO CHANGE
40E0 | A9 CE        CAPN     LDA   #0CE
40E2 | D0F9                  BNE   SP1
40E4 | A9 D0        CAPP     LDA   #0D0
40E6 | D05F                  BNE   SP1
40E8 | A9 CD        CAPM     LDA   #0CD
40EA | D0F1                  BNE   SP1
40EC |
40EC | 48          STEST1   PHA                ; SAVE THE CHARACTER
40ED | AD 63C0              LDA   SWIT2        ; SHIFT KEY DOWN?
40F0 | F00A                BEQ   CTRLV        ; CHECK FOR CTRL-V
40F2 | 68                  PLA
40F3 | C9 8B               CMP   #8B          ; CTRL-K ?
40F5 | D002                BNE   EXIT         ; NO, WE'RE DONE.
40F7 | AO DB               LDA   #0DB         ; CHANGE IT TO A '['
40F9 | 4C 6C40      EXIT     JMP   NOFIX        ; SO LONG FOLKS
40FC | 68          CTRLV    PLA
40FD | C9 96               CMP   #96          ; CTRL—SHIFT—V (SPECIAL CHARACTERS)
40FF | D00B                BNE   HORF         ; NO, TRY THE DUPLEX SWITCH
4101 | A9 FF               LDA   #0FF         ; YES, TOGGLE THE SPECIAL CHARACTER SWITCH
4103 | 4D FA04             EOR   SPECIAL
4106 | 8D FA04             STA   SPECIAL      ; FIXED
4109 | 4C 4640             JMP   KEY          ; START OVER ... DON'T XMIT THE CHARACTER
410C | C9 94        HORF     CMP   #94          ; CTRL-SHIFT-T (HALF/FULL DUPLEX)
410E | D00B                BNE   SLOCK        ; NO, TRY SHIFT LOCK
4110 | A9 FF               LDA   #0FF
4112 | 4D 7A04             EOR   DUPLEX
4115 | 8D 7A04             STA   DUPLEX
4118 | 4C 4640             JMP   KEY
411B | C9 81        SLOCK    CMP   #81          ; CTRL-SHIFT-A
411D | D00B                BNE   QUIT         ; LAST CHANCE
411F | A9 FF               LDA   #0FF
4121 | 4D 7A06             EOR   UPCASE
4124 | 8D 7A06             STA   UPCASE
4127 | 4C 4640             JMP   KEY
412A | C9 91        QUIT     CMP   #91
412C | D0CB                BNE   EXIT         ; NOT SPECIAL...
412E | 60                  RTS                ; BYE BYE
412F |
412F | 48          NOCTRL   PHA                ; BETTER SAVE THE CHARACTER
4130 | 29 7F               AND   #7F          ; REMOVE MSB (PARITY) ON INCOMING CHARACTERS
4132 | C9 20               CMP   #20          ; LESS THAN A SPACE MEANS IT'S A CTRL CHARACTER
4134 | 9002                BCC   SHUFFLE      ; GO FIX THE CURSOR
4136 | 68          AOK      PLA                ; RECOVER CHARACTER
4137 | 60                  RTS                ; BACK TO CALLER
4138 | AD 7A05     SHUFFLE  LDA   CURSTAT      ; CHECK FOR CURSOR STATUS
413B | F0F9                BEQ   AOK          ; CURSOR IS OFF
413D | A9 A0               LDA   #0A0         ; A SPACE
413F | 20 EDFD             JSR   CHROUT       ; REMOVE CURSOR
4142 | A9 88               LDA   #88          ; A BACKSPACE (FIX POSITION)
4144 | 20 EDFD             JSR   CHROUT
4147 | A9 00               LDA   #0           ; FIX THE CURSOR ON/OFF FLAG
4149 | 8D 7A05             STA   CURSTAT
414C | F0E8                BEQ   AOK          ; FORCED BRANCH (A = 0)
414E |
414E | 48          CBLINK   PHA                ; SAVE THE CHARACTER
414F | CE FA06             DEC   BLINK1       ; COUNT DOWN TO 0
4152 | D0E2                BNE   AOK          ; CURSOR STAYS AS IS
4154 | CE 7A07             DEC   BLINK2       ; SECOND TIMER
4157 | D0DD                BNE   AOK
4159 | A9 20               LDA   #20          ; TIMER #2 DELAY
415B | 8D 7A07             STA   BLINK2
415E | AD 7A05             LDA   CURSTAT      ; ON OR OFF?
4161 | 49 FF               EOR   #0FF         ; FLIP IT
4163 | 8D 7A05             STA   CURSTAT      ; PUT IT BACK
4166 | F007                BEQ   OFF
4168 | A9 DF               LDA   #0DF         ; UNDERLINE
416A | 20 EDFD             JSR   CHROUT
416D | D005                BNE   CDONE        ; FINISH UP HERE
416F | A9 A0        OFF      LDA   #0A0         ; A SPACE
4171 | 20 EDFD             JSR   CHROUT
4174 | A9 88        CDONE    LDA   #088         ; BACKSPACE
4176 | 20 EDFD             JSR   CHROUT
4179 | 68                  PLA                ; RECOVER CHARACTER
417A | 60                  RTS
417B |
417B |                     .END
```

PASCAL DEMONSTRATION PROGRAM SOURCE LISTING

The following is a source listing for a Pascal demonstration program that shows Sup'r'terminal text capabilities, including modifying a character on the fly, while also running a color graphic program for true z screen operation.

```
 1   1    1:D     1 {$LREMOUT:}
 2   1    1:D     1 program DEMO1;
 3   1    1:D     3 { Part of the M&R demo programs. Rev b - 21 Jun 80 - D. Sokol }
 4  26    1:D     3
 5  26    1:D     3
 6  26    2:D     1 PROCEDURE POKE(VAR ADDR,DATA:INTEGER);
 7  26    3:D     3 FUNCTION PEEK(VAR ADDR:INTEGER):INTEGER;
 8  26    3:D     4
 9  20    1:D     4
10  20    1:D     3     TYPE
11  20    1:D     1       SCREENCOLOR=(none,white,black,reverse,radar,
12  20    1:D     1                    black1,green,violet,white1,black2,orange,blue,white2);
13  20    1:D     1
14  20    2:D     1     PROCEDURE INITTURTLE;
15  20    3:D     1     PROCEDURE TURN(ANGLE: INTEGER);
16  20    4:D     1     PROCEDURE TURNTO(ANGLE: INTEGER);
17  20    5:D     1     PROCEDURE MOVE(DIST: INTEGER);
18  20    6:D     1     PROCEDURE MOVETO(X,Y: INTEGER);
19  20    7:D     1     PROCEDURE PENCOLOR(PENMODE: SCREENCOLOR);
20  20    8:D     1     PROCEDURE TEXTMODE;
21  20    9:D     1     PROCEDURE GRAFMODE;
22  20   10:D     1     PROCEDURE FILLSCREEN(FILLCOLOR: SCREENCOLOR);
23  20   11:D     1     PROCEDURE VIEWPORT(LEFT,RIGHT,BOTTOM,TOP: INTEGER);
24  20   12:D     3     FUNCTION  TURTLEX: INTEGER;
25  20   13:D     3     FUNCTION  TURTLEY: INTEGER;
26  20   14:D     3     FUNCTION  TURTLEANG: INTEGER;
27  20   15:D     3     FUNCTION  SCREENBIT(X,Y: INTEGER): BOOLEAN;
28  20   16:D     1     PROCEDURE DRAWBLOCK(VAR SOURCE; ROWSIZE,XSKIP,YSKIP,WIDTH,HEIGHT,
29  20   16:D     2                         XSCREEN,YSCREEN,MODE: INTEGER);
30  20   17:D     1     PROCEDURE WCHAR(CH: CHAR);
31  20   18:D     1     PROCEDURE WSTRING(S: STRING);
32  20   19:D     1     PROCEDURE CHARTYPE(MODE: INTEGER);
33  20   19:D     2
34   1    1:D     2 uses peekpoke,turtlegr;
35   1    1:D     3 {$R-,I-}
36   1    1:D     3 type BITS = packed array [0..7] of integer;
37   1    1:D     3      SHAPE = packed array [0..23,0..23] of boolean;
38   1    1:D     3
39   1    1:D     3 var S,U,P,R1,T,E,R2,M,I1,N,A,L : SHAPE;
40   1    1:D   579     DELAY,ADDR,LEN,I,DATA,X,Y : integer;
41   1    1:D   586     BANNERFILE : file of SHAPE;
42   1    1:D   934     CHARFILE : file of bits;
43   1    1:D  1242     Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9,Z10,Z11,Z12,Z13,Z14,Z15 : BITS;
44   1    1:D  1362     HELLFREEZESOVER : boolean;
45   1    1:D  1363
46   1    2:D     1 procedure BACKGROUND;
47   1    2:0     0  begin
48   1    2:1     0    textmode;
49   1    2:1     3    viewport(0,279,0,191); fillscreen(WHITE);
50   1    2:0    18  end;
51   1    2:0    30
52   1    3:D     1 procedure GETZ;
53   1    3:0     0  begin
54   1    3:1     0    reset(CHARFILE,'Z.CHARSET');
55   1    3:1    20    Z1:=CHARFILE^; get(CHARFILE);
56   1    3:1    34    Z2:=CHARFILE^; get(CHARFILE);
57   1    3:1    48    Z3:=CHARFILE^; get(CHARFILE);
58   1    3:1    62    Z4:=CHARFILE^; get(CHARFILE);
59   1    3:1    76    Z5:=CHARFILE^; get(CHARFILE);
60   1    3:1    90    Z6:=CHARFILE^; get(CHARFILE);
61   1    3:1   104    Z7:=CHARFILE^; get(CHARFILE);
62   1    3:1   118    Z8:=CHARFILE^; get(CHARFILE);
63   1    3:1   132    Z9:=CHARFILE^; get(CHARFILE);
64   1    3:1   146    Z10:=CHARFILE^; get(CHARFILE);
65   1    3:1   160    Z11:=CHARFILE^; get(CHARFILE);
66   1    3:1   174    Z12:=CHARFILE^; get(CHARFILE);
67   1    3:1   188    Z13:=CHARFILE^; get(CHARFILE);
68   1    3:1   202    Z14:=CHARFILE^; get(CHARFILE);
69   1    3:1   216    Z15:=CHARFILE^; get(CHARFILE);
70   1    3:1   230    close(CHARFILE);
71   1    3:0   237  end;
72   1    3:0   250
73   1    4:D     1 procedure HOME;
74   1    4:0     0  begin
75   1    4:1     0    write(chr(12));
76   1    4:0     8  end;
77   1    4:0    20
78   1    5:D     1 procedure CHANGEZ(FIGURE : BITS);
```

```
  1    5:D    10  var    ROMON,ROMOFF,TEMP,I,ADDR,DATA : integer;
  1    5:0     0  begin
  1    5:1     0   ROMON:=-16206; ROMOFF:=-12209; TEMP:=0;
  1    5:1    20   poke(ROMOFF,TEMP); poke(ROMON,TEMP);
  1    5:1    34   for I:=0 to 7 do
  1    5:2    45   begin
  1    5:3    45    ADDR:=-13360+I;
  1    5:3    53    DATA:=FIGURE[I];
  1    5:3    61    poke(ADDR,DATA);
  1    5:2    68   end;
  1    5:0    75  end;
  1    5:0    90
  1    6:D     1 procedure SETSCREEN;
  1    6:D     1  var I : integer;
  1    6:0     0  begin
  1    6:1     0   HOME; CHANGEZ(Z15);
  1    6:1     7   for I:=1 to 23 do writeln(
  1    6:2    18 'zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz');
  1    6:0   119  end;
  1    6:0   134
  1    6:0   134
  1    7:D     1 procedure MESSWITH;
  1    7:D     1  var I : integer;
  1    7:0     0  begin
  1    7:1     0   for I:=12 downto 1 do
  1    7:2    11   begin
  1    7:3    11     fillscreen(reverse);
  1    7:3    15     case I of
  1    7:3    18    1:CHANGEZ(Z1);
  1    7:3    25    2:CHANGEZ(Z2);
  1    7:3    32    3:CHANGEZ(Z3);
  1    7:3    39    4:CHANGEZ(Z4);
  1    7:3    46    5:CHANGEZ(Z5);
  1    7:3    53    6:CHANGEZ(Z6);
  1    7:3    60    7:CHANGEZ(Z7);
  1    7:3    67    8:CHANGEZ(Z8);
  1    7:3    74    9:CHANGEZ(Z9);
  1    7:3    81    10:CHANGEZ(Z10);
  1    7:3    88    11:CHANGEZ(Z11);
  1    7:3    95    12:CHANGEZ(Z12);
  1    7:3   102    end; end;
  1    7:0   141  end;
  1    7:0   158
  1    7:0   158
  1    8:D     1  procedure WRITEBANNER;
  1    8:0     0   begin
  1    8:1     0    X:=0; Y:=90; pencolor(none); moveto(75,120); chartype(6);
  1    8:1    21    pencolor(white); wstring('M & R Enterprises');
  1    8:1    48    viewport(0,279,90,114); fillscreen(blue);
  1    8:1    61    drawblock(S,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1    87    drawblock(U,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   113    drawblock(P,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   139    drawblock(R1,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   165    drawblock(T,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   191    drawblock(E,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   217    drawblock(R2,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   243    drawblock(M,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   269    drawblock(I1,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   295    drawblock(N,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   320    drawblock(A,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   345    drawblock(L,4,0,0,24,24,X,Y,6); X:=X+24;
  1    8:1   370    grafmode;
  1    8:0   373   end;
  1    8:0   386
  1    9:D     1  procedure GETBANNER;
  1    9:0     0   begin
  1    9:1     0    reset(BANNERFILE,'BANER.CHARSET');
  1    9:1    24    S:=BANNERFILE^; get(BANNERFILE);
  1    9:1    38    U:=BANNERFILE^; get(BANNERFILE);
  1    9:1    52    P:=BANNERFILE^; get(BANNERFILE);
  1    9:1    66    R1:=BANNERFILE^; get(BANNERFILE);
  1    9:1    80    T:=BANNERFILE^; get(BANNERFILE);
  1    9:1    94    E:=BANNERFILE^; get(BANNERFILE);
  1    9:1   108    R2:=BANNERFILE^; get(BANNERFILE);
  1    9:1   122    M:=BANNERFILE^; get(BANNERFILE);
  1    9:1   136    I1:=BANNERFILE^; get(BANNERFILE);
  1    9:1   150    N:=BANNERFILE^; get(BANNERFILE);
  1    9:1   163    A:=BANNERFILE^; get(BANNERFILE);
  1    9:1   176    L:=BANNERFILE^;
  1    9:1   183    close(BANNERFILE);
  1    9:0   190   end;
  1    9:0   202
  1   10:D     1 procedure INVERT(var ADR,LEN : integer);
  1   10:0     0  begin
  1   10:1     0   ADDR:=-12289; DATA:=0; poke(ADDR,DATA); {  CFFF rom off }
  1   10:1    20   ADDR:=-16204; poke(ADDR,DATA); { bank 2 on }
  1   10:1    36   ADDR:=-14336+ADR; { C800+ADR }
  1   10:1    46   for I:=ADDR to ADDR+LEN do
  1   10:2    67   begin
```

```
168   1   10:3    67    DATA:=peek(I)+64; poke(I,DATA);
169   1   10:2    89   end;
170   1   10:0    99  end;
171   1   10:0   114
172   1   11:D     1 procedure MROFF;
173   1   11:D     1  var ADDR,DATA : integer;
174   1   11:0     0  begin
175   1   11:1     0   ADDR:=-16202; DATA:=0;
176   1   11:1     9   poke(ADDR,DATA);
177   1   11:0    16  end;
178   1   11:0    28
179   1   12:D     1 procedure MRON;
180   1   12:D     1  var ADDR,DATA : integer;
181   1   12:0     0  begin
182   1   12:1     0   ADDR:=-12289; DATA:=0;
183   1   12:1     9   poke(ADDR,DATA);
184   1   12:1    16   ADDR:=-16202; poke(ADDR,DATA);
185   1   12:0    29  end;
186   1   12:0    42
187   1   13:D     1 procedure FEATURES;
188   1   13:0     0 begin
189   1   13:1     0 write(chr(12));
190   1   13:1     8 gotoxy(32,1); write('FEATURES');
191   1   13:1    31 gotoxy(5,3); write(
192   1   13:1    36 '1. 80 Character, 24 Line, Upper and Lower case characters.');
193   1   13:1   104 gotoxy(5,5); write(
194   1   13:1   109 '2. Fully PASCAL compatable - no software modifications needed.');
195   1   13:1   181 gotoxy(5,7); write(
196   1   13:1   186 '3. Allows two screen operation - SUP''R''TERMINAL and HIRES graphics.');
197   1   13:1   263 gotoxy(5,9); write(
198   1   13:1   268 '4. User programable character set - 128 characters available !');
199   1   13:1   340 gotoxy(5,11); write(
200   1   13:1   345 '5. User selectable cursor - including no cursor.');
201   1   13:1   403 gotoxy(5,13); write(
202   1   13:1   408 '6. Direct cursor addressing - From PASCAL use the GOTOXY intrinsic.');
203   1   13:1   485 gotoxy(35,14); write(
204   1   13:1   490 'From BASIC use CTRL-SHIFT-N.');
205   1   13:1   528 gotoxy(5,16); write(
206   1   13:1   533 '7. NORMAL and INVERTED characters are available');
207   1   13:1   590 X:=(16*80)+18; LEN:=8; INVERT(X,LEN);
208   1   13:1   610 gotoxy(5,18); write(
209   1   13:1   615 '8. Adds lower case capability to the APPLE keyboard.');
210   1   13:1   677 gotoxy(5,20); write(
211   1   13:1   682 '9. Low cost - $395 list.');
212   1   13:0   716 end;
213   1   13:0   728
214   1   14:D     3 function TIMEOUT : boolean;
215   1   14:0     0  begin
216   1   14:1     0   if DELAY=0 then TIMEOUT:=true else TIMEOUT:=false;
217   1   14:1    15   DELAY:=DELAY-1;
218   1   14:0    23  end;
219   1   14:0    36
220   1   15:D     1 procedure POSITION;
221   1   15:0     0  begin
222   1   15:1     0   ADDR:=-16200; DATA:=14; poke(ADDR,DATA);
223   1   15:1    20   ADDR:=-16199; DATA:=4; poke(ADDR,DATA);
224   1   15:1    40   ADDR:=-16200; DATA:=15; poke(ADDR,DATA);
225   1   15:1    60   ADDR:=-16199; DATA:=210; poke(ADDR,DATA);
226   1   15:0    82  end;
227   1   15:0    94
228   1   16:D     1 procedure CURSOR;
229   1   16:D     1  var C : integer;
230   1   16:0     0  begin
231   1   16:1     0   writeln;
232   1   16:1     6   for C:=0 to 6 do case C of
233   1   16:2    20   0:begin gotoxy(12,12); write(chr(26));
234   1   16:4    33      write('An underlining cursor flashing 4 times a second.');
235   1   16:4    91      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
236   1   16:4   113      ADDR:=-16199; DATA:=71; poke(ADDR,DATA);
237   1   16:4   133      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
238   1   16:3   147     end;
239   1   16:2   149   1:begin gotoxy(12,12); write(chr(26));
240   1   16:4   162      write('A box cursor flashing 4 times a second.');
241   1   16:4   211      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
242   1   16:4   233      ADDR:=-16199; DATA:=64; poke(ADDR,DATA);
243   1   16:4   253      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
244   1   16:3   267     end;
245   1   16:2   269   2:begin gotoxy(12,12); write(chr(26));
246   1   16:4   282      write('An underlining cursor with no flashing.');
247   1   16:4   331      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
248   1   16:4   353      ADDR:=-16199; DATA:=7; poke(ADDR,DATA);
249   1   16:4   373      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
250   1   16:3   387     end;
251   1   16:2   389   3:begin gotoxy(12,12); write(chr(26));
252   1   16:4   402      write('A box cursor with no flashing.');
253   1   16:4   442      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
254   1   16:4   464      ADDR:=-16199; DATA:=0; poke(ADDR,DATA);
255   1   16:4   484      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
256   1   16:3   498     end;
257   1   16:2   500   4:begin gotoxy(12,12); write(chr(26));
```

```
258   1   16:4   513      write('An underlining cursor flashing 2 times a second.');
259   1   16:4   571      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
260   1   16:4   593      ADDR:=-16199; DATA:=103; poke(ADDR,DATA);
261   1   16:4   613      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
262   1   16:3   627     end;
263   1   16:2   629   5:begin gotoxy(12,12); write(chr(26));
264   1   16:4   642      write('A box cursor flashing 2 times a second.');
265   1   16:4   691      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
266   1   16:4   713      ADDR:=-16199; DATA:=96; poke(ADDR,DATA);
267   1   16:4   733      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
268   1   16:3   747     end;
269   1   16:2   749   6:begin gotoxy(12,12); write(chr(26));
270   1   16:4   762      write('No cursor !');
271   1   16:4   783      POSITION; ADDR:=-16200; DATA:=10; poke(ADDR,DATA);
272   1   16:4   805      ADDR:=-16199; DATA:=32; poke(ADDR,DATA);
273   1   16:4   825      DELAY:=50; repeat fillscreen(reverse); until TIMEOUT;
274   1   16:3   839     end;
275   1   16:2   841   end; { case }
276   1   16:0   869   end; { procedure }
277   1   16:0   904
278   1   17:D     1 procedure ANYWHERE;
279   1   17:D     1  var ST : string;
280   1   17:D    42      J,K : integer;
281   1   17:0     0  begin
282   1   17:1     0   ST:=' * * * M & R Enterprises SUP''R''TERMINAL * * * ';
283   1   17:1    53   J:=length(ST);
284   1   17:1    59   for K:=1 to (J div 2) do
285   1   17:2    75   begin
286   1   17:3    75    gotoxy(14+K,9); write(ST[K]);
287   1   17:3    95    fillscreen(reverse); fillscreen(reverse);
288   1   17:3   103    gotoxy(14+J-K,9); write(ST[J-K]);
289   1   17:3   129    fillscreen(reverse); fillscreen(reverse);
290   1   17:2   137   end;
291   1   17:0   145  end;
292   1   17:0   160
293   1   18:D     1 procedure SCROLL;
294   1   18:D     1  var K,J,L : integer;
295   1   18:D     4      ST : string[96];
296   1   18:0     0  begin
297   1   18:1     0   write(chr(20),chr(84),chr(41)); writeln; { LINE 9 }
298   1   18:1    30   write(chr(20),chr(66),chr(50)); writeln;  { LINE 20 }
299   1   18:1    60   for K:=32 to 127 do
300   1   18:2    72   ST[K-31]:=chr(K);
301   1   18:1    86   L:=1; gotoxy(0,9);
302   1   18:1    94   repeat
303   1   18:2    94    J:=L; for K:=0 to 79 do
304   1   18:3   109    begin
305   1   18:4   109     write(ST[J]);
306   1   18:4   120     J:=(J MOD 94) + 1;
307   1   18:3   127     end;
308   1   18:2   134    L:=(L MOD 94) + 1; fillscreen(reverse)
309   1   18:1   142   until TIMEOUT;
310   1   18:0   151  end;
311   1   18:0   170
312   1   19:D     1 procedure ONEATATIME;
313   1   19:D     1  var DEMO : integer;
314   1   19:0     0  begin
315   1   19:1     0   for DEMO := 1 to 6 do begin
316   1   19:3    11   HOME;
317   1   19:3    13   gotoxy(5,5);
318   1   19:3    18   case DEMO of
319   1   19:3    21   1:begin write('1. 80 Character, 24 Line, Upper and Lower case characters.');
320   1   19:5    89           gotoxy(8,6); write('With user selectable scrolling window.');
321   1   19:5   142           writeln; writeln; DELAY:=100; SCROLL;
322   1   19:5   160           write(chr(20),chr(82)); { resets suprterminal }
323   1   19:4   176     end;
324   1   19:3   178   2:begin
325   1   19:5   178       writeln('2. Fully PASCAL compatable - no software modifications needed.');
326   1   19:5   256       writeln('        This demonstration program was written in PASCAL.');
327   1   19:5   329       MROFF; DELAY:=75; repeat fillscreen(reverse) until TIMEOUT; MRON;
328   1   19:4   347     end;
329   1   19:3   349   3:begin
330   1   19:5   349      write(
331   1   19:5   349      '3. Allows two screen operation - SUP''R''TERMINAL and HIRES graphics.');
332   1   19:5   426       MROFF; DELAY:=60; repeat fillscreen(reverse) until TIMEOUT; MRON;
333   1   19:4   444     end;
334   1   19:3   446   4:begin SETSCREEN; DELAY:=9; gotoxy(5,5); write(
335   1   19:5   457           ' 4. User programable character set - 128 characters available ! ');
336   1   19:5   531           repeat MESSWITH; until TIMEOUT;
337   1   19:5   539           write(chr(20),chr(82)); { resets suprterminal }
338   1   19:4   555     end;
339   1   19:3   557   5:begin write('5. User selectable cursor - including no cursor.');
340   1   19:5   615           CURSOR;
341   1   19:4   617     end;
342   1   19:3   619   6:begin
343   1   19:5   619    write('6. Direct cursor addressing - From PASCAL use the GOTOXY intrinsic.');
344   1   19:5   696     ANYWHERE; DELAY:=100; MROFF; repeat fillscreen(reverse); until TIMEOUT;
345   1   19:5   714     MRON;
346   1   19:4   716     end;
347   1   19:3   718   end; { case end }
```

```
348   1   19:3   738    fillscreen(reverse);
349   1   19:2   742    end; { for loop }
350   1   19:0   749   end; { procedure }
351   1   19:0   778
352   1   19:0   778   { main }
353   1    1:0     0  begin
354   1    1:1     0   HELLFREEZESOVER:=false;
355   1    1:1    26   GETBANNER;
356   1    1:1    28   GETZ;
357   1    1:1    30   INITTURTLE;
358   1    1:1    33   BACKGROUND;
359   1    1:1    35   WRITEBANNER;
360   1    1:1    37  repeat
361   1    1:2    37   HOME;
362   1    1:2    39   for I:=1 to 35 do begin DELAY:=40; MROFF; repeat until TIMEOUT; MRON; end;
363   1    1:2    80   FEATURES; X:=32+80; LEN:=8;
364   1    1:2    92   DELAY:=250;
365   1    1:2    98   repeat INVERT(X,LEN); fillscreen(reverse); until TIMEOUT;
366   1    1:2   116   ONEATATIME;
367   1    1:2   118   write(chr(20),chr(82)); { resets suprterminal }
368   1    1:1   134  until HELLFREEZESOVER;
369   1    1:0   139  end.

  1   1    1:D     1  (*$LREMOUT:*)
  2   1    1:D     1  (* THIS PROGRAM TAKES INPUT FROM THE
  3   1    1:D     1  KEYBOARD AND GENERATES THE FILE Z.CHARSET .
  4   1    1:D     1  THE FILE IS USED TO MODIFY THE CHARACTER 'z'
  5   1    1:D     1  TO MAKE IT APPEAR TO ROTATE *)
  6   1    1:D     1
  7   1    1:D     1  PROGRAM MAKEZFILE;
  8   1    1:D     3  TYPE BITS = PACKED ARRAY[0..7] of integer;
  9   1    1:D     3  var charfile : file of bits;
 10   1    1:D   311      Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9,Z10,Z11,Z12,Z13,Z14,Z15 : BITS;
 11   1    1:D   431      P,I,K : INTEGER;
 12   1    1:D   434      T : PACKED ARRAY [0..7] OF INTEGER;
 13   1    1:D   442
 14   1    2:D     1  PROCEDURE SAVEZ;
 15   1    2:0     0   BEGIN
 16   1    2:1     0    REWRITE(CHARFILE,'Z.CHARSET');
 17   1    2:1    21    CHARFILE^:=Z1; PUT(CHARFILE);
 18   1    2:1    34    CHARFILE^:=Z2; PUT(CHARFILE);
 19   1    2:1    47    CHARFILE^:=Z3; PUT(CHARFILE);
 20   1    2:1    60    CHARFILE^:=Z4; PUT(CHARFILE);
 21   1    2:1    73    CHARFILE^:=Z5; PUT(CHARFILE);
 22   1    2:1    86    CHARFILE^:=Z6; PUT(CHARFILE);
 23   1    2:1    99    CHARFILE^:=Z7; PUT(CHARFILE);
 24   1    2:1   112    CHARFILE^:=Z8; PUT(CHARFILE);
 25   1    2:1   125    CHARFILE^:=Z9; PUT(CHARFILE);
 26   1    2:1   138    CHARFILE^:=Z10; PUT(CHARFILE);
 27   1    2:1   151    CHARFILE^:=Z11; PUT(CHARFILE);
 28   1    2:1   164    CHARFILE^:=Z12; PUT(CHARFILE);
 29   1    2:1   177    CHARFILE^:=Z13; PUT(CHARFILE);
 30   1    2:1   190    CHARFILE^:=Z14; PUT(CHARFILE);
 31   1    2:1   203    CHARFILE^:=Z15; PUT(CHARFILE);
 32   1    2:1   216    CLOSE(CHARFILE,LOCK);
 33   1    2:0   224   END;
 34   1    2:0   236
 35   1    3:D     1  PROCEDURE READZ;
 36   1    3:0     0   BEGIN
 37   1    3:1     0    FOR I:=0 TO 7 DO BEGIN
 38   1    3:3    14     WRITE('CHAR #',K,' LINE ',I,' - '); READLN(T[I]);
 39   1    3:2   116    END;
 40   1    3:1   126    K:=K+1;
 41   1    3:0   134   END;
 42   1    3:0   148
 43   1    4:D     1  PROCEDURE XCNG;
 44   1    4:0     0   BEGIN
 45   1    4:1     0   CASE P OF
 46   1    4:1     5   1:Z1:=T;
 47   1    4:1    15   2:Z2:=T;
 48   1    4:1    25   3:Z3:=T;
 49   1    4:1    35   4:Z4:=T;
 50   1    4:1    45   5:Z5:=T;
 51   1    4:1    55   6:Z6:=T;
 52   1    4:1    65   7:Z7:=T;
 53   1    4:1    75   8:Z8:=T;
 54   1    4:1    85   9:Z9:=T;
 55   1    4:1    95   10:Z10:=T;
 56   1    4:1   105   11:Z11:=T;
 57   1    4:1   115   12:Z12:=T;
 58   1    4:1   125   13:Z13:=T;
 59   1    4:1   135   14:Z14:=T;
 60   1    4:1   145   15:Z15:=T;
 61   1    4:1   155   END;
 62   1    4:0   192   END;
 63   1    4:0   208
 64   1    5:D     1  PROCEDURE DOIT;
 65   1    5:0     0  BEGIN
 66   1    5:1     0   FOR P:=1 TO 15 DO BEGIN
```

```
67   1     5:3     14  WRITE(CHR(140));
68   1     5:3     26  READZ;
69   1     5:3     28  XCNG;
70   1     5:2     30  END;
71   1     5:0     40 END;
72   1     5:0     54
73   1     5:0     54  (* MAIN *)
74   1     1:0      0 BEGIN
75   1     1:1      0  K:=1;
76   1     1:1     15  DOIT;
77   1     1:1     17  SAVEZ;
78   1     1:0     19 END.


   1   1     1:D      1 (*$LREMOUT:*)
 2   1     1:D      1 program MAKEBANNER;
 3   1     1:D      3
 4   1     1:D      3 type SHAPE = packed array [0..23,0..23] of boolean;
 5   1     1:D      3
 6   1     1:D      3 var S,U,P,R1,T,E,R2,M,I1,N,A,L : SHAPE;
 7   1     1:D    579     I,J,ROW : integer;
 8   1     1:D    582     BIT : boolean;
 9   1     1:D    583     BANNERFILE : file of SHAPE;
10   1     1:D    931
11   1     2:D      1 procedure MAKESHAPES(var BITMAP:SHAPE; ST:string);
12   1     2:0      0  begin
13   1     2:1      0   for J:=1 to 24 do
14   1     2:2     20   begin
15   1     2:3     20    BIT:=(ST[J]<>' ');
16   1     2:3     32    BITMAP[ROW,J-1]:=BIT;
17   1     2:2     56   end;
18   1     2:1     66   ROW:=ROW-1;
19   1     2:0     74  end;
20   1     2:0     88
21   1     3:D      1  procedure PUTBANNER;
22   1     3:0      0   begin
23   1     3:1      0    rewrite(BANNERFILE,'BANER.CHARSET');
24   1     3:1     26    BANNERFILE^:=S; put(BANNERFILE);
25   1     3:1     42    BANNERFILE^:=U; put(BANNERFILE);
26   1     3:1     58    BANNERFILE^:=P; put(BANNERFILE);
27   1     3:1     74    BANNERFILE^:=R1; put(BANNERFILE);
28   1     3:1     90    BANNERFILE^:=T; put(BANNERFILE);
29   1     3:1    106    BANNERFILE^:=E; put(BANNERFILE);
30   1     3:1    122    BANNERFILE^:=R2; put(BANNERFILE);
31   1     3:1    138    BANNERFILE^:=M; put(BANNERFILE);
32   1     3:1    154    BANNERFILE^:=I1; put(BANNERFILE);
33   1     3:1    170    BANNERFILE^:=N; put(BANNERFILE);
34   1     3:1    185    BANNERFILE^:=A; put(BANNERFILE);
35   1     3:1    200    BANNERFILE^:=L; put(BANNERFILE);
36   1     3:1    215    close(BANNERFILE,lock);
37   1     3:0    224   end;
38   1     3:0    236
39   1     4:D      1  procedure INIT1;
40   1     4:0      0  begin
41   1     4:1      0   ROW:=23;   (* . . . . . . . . . . . . *)
42   1     4:1      4   MAKESHAPES(S,'      X X X                ');
43   1     4:1     36   MAKESHAPES(S,'    X X X X X              ');
44   1     4:1     68   MAKESHAPES(S,'  X X       X X            ');
45   1     4:1    100   MAKESHAPES(S,'  X           X            ');
46   1     4:1    132   MAKESHAPES(S,'X                          ');
47   1     4:1    164   MAKESHAPES(S,'X                          ');
48   1     4:1    196   MAKESHAPES(S,'X                          ');
49   1     4:1    228   MAKESHAPES(S,'X                          ');
50   1     4:1    260   MAKESHAPES(S,'  X                        ');
51   1     4:1    292   MAKESHAPES(S,'  X                        ');
52   1     4:1    324   MAKESHAPES(S,'    X X X X X              ');
53   1     4:1    356   MAKESHAPES(S,'    X X X X X              ');
54   1     4:1    388   MAKESHAPES(S,'              X            ');
55   1     4:1    420   MAKESHAPES(S,'              X            ');
56   1     4:1    452   MAKESHAPES(S,'                X          ');
57   1     4:1    484   MAKESHAPES(S,'                X          ');
58   1     4:1    516   MAKESHAPES(S,'                X          ');
59   1     4:1    548   MAKESHAPES(S,'                X          ');
60   1     4:1    580   MAKESHAPES(S,'  X           X            ');
61   1     4:1    612   MAKESHAPES(S,'  X X       X X            ');
62   1     4:1    644   MAKESHAPES(S,'    X X X X X              ');
63   1     4:1    676   MAKESHAPES(S,'      X X X                ');
64   1     4:1    708   MAKESHAPES(S,'                           ');
65   1     4:1    740   MAKESHAPES(S,'                           ');
66   1     4:1    772   ROW:=23;
67   1     4:1    776   for I := 1 to 16 do
68   1     4:2    790   MAKESHAPES(U,'X             X            ');
69   1     4:1    832   MAKESHAPES(U,'X X         X X            ');
70   1     4:1    864   MAKESHAPES(U,'  X         X              ');
71   1     4:1    896   MAKESHAPES(U,'  X X     X X              ');
72   1     4:1    928   MAKESHAPES(U,'  X X     X X              ');
73   1     4:1    960   MAKESHAPES(U,'    X X X X                ');
74   1     4:1    992   MAKESHAPES(U,'    X X X X                ');
75   1     4:1   1024   MAKESHAPES(U,'                           ');
76   1     4:1   1056   MAKESHAPES(U,'                           ');
```

```
 77   1    4:0  1088  end;
 78   1    4:0  1102
 79   1    5:D     1  procedure INIT2;
 80   1    5:0     0  begin
 81   1    5:1     0   ROW:=23;
 82   1    5:1     4   MAKESHAPES(P,'X X X X X          X      ');
 83   1    5:1    36   MAKESHAPES(P,'X X X X X X        X      ');
 84   1    5:1    68   MAKESHAPES(P,'X          X       X      ');
 85   1    5:1   100   MAKESHAPES(P,'X            X     X      ');
 86   1    5:1   132   MAKESHAPES(P,'X            X            ');
 87   1    5:1   164   for I:= 1 to 3 do
 88   1    5:2   178   MAKESHAPES(P,'X              X          ');
 89   1    5:1   220   MAKESHAPES(P,'X            X            ');
 90   1    5:1   252   MAKESHAPES(P,'X            X            ');
 91   1    5:1   284   MAKESHAPES(P,'X          X              ');
 92   1    5:1   316   MAKESHAPES(P,'X X X X X X               ');
 93   1    5:1   348   MAKESHAPES(P,'X X X X X                 ');
 94   1    5:1   380   for I:=1 to 9 do
 95   1    5:2   394   MAKESHAPES(P,'X                         ');
 96   1    5:1   436   MAKESHAPES(P,'                          ');
 97   1    5:1   468   MAKESHAPES(P,'                          ');
 98   1    5:1   500   ROW:=23;     (* . . . . . . . . . . . . *)
 99   1    5:1   504   MAKESHAPES(R1,'X X X X X          X     ');
100   1    5:1   536   MAKESHAPES(R1,'X X X X X X        X     ');
101   1    5:1   568   MAKESHAPES(R1,'X          X       X     ');
102   1    5:1   600   MAKESHAPES(R1,'X            X     X     ');
103   1    5:1   632   MAKESHAPES(R1,'X            X           ');
104   1    5:1   664   for I:= 1 to 3 do
105   1    5:2   678   MAKESHAPES(R1,'X              X         ');
106   1    5:1   720   MAKESHAPES(R1,'X            X           ');
107   1    5:1   752   MAKESHAPES(R1,'X            X           ');
108   1    5:1   784   MAKESHAPES(R1,'X          X             ');
109   1    5:1   816   MAKESHAPES(R1,'X X X X X X               ');
110   1    5:1   848   MAKESHAPES(R1,'X X X X X                 ');
111   1    5:1   880   MAKESHAPES(R1,'X X                       ');
112   1    5:1   912   MAKESHAPES(R1,'X   X                     ');
113   1    5:1   944   MAKESHAPES(R1,'X     X                   ');
114   1    5:1   976   MAKESHAPES(R1,'X       X                 ');
115   1    5:1  1008   MAKESHAPES(R1,'X         X               ');
116   1    5:1  1040   for I:=1 to 4 do
117   1    5:2  1054   MAKESHAPES(R1,'X            X           ');
118   1    5:1  1096   MAKESHAPES(R1,'                         ');
119   1    5:1  1128   MAKESHAPES(R1,'                         ');
120   1    5:0  1160  end;
121   1    5:0  1180
122   1    6:D     1  procedure INIT3;
123   1    6:0     0  begin
124   1    6:1     0   ROW:=23;   (* . . . . . . . . . . . . *)
125   1    6:1     4   MAKESHAPES(T,'X X X X X X X            ');
126   1    6:1    36   MAKESHAPES(T,'X X X X X X X            ');
127   1    6:1    68   for I:=1 to 20 do
128   1    6:2    82   MAKESHAPES(T,'       X                 ');
129   1    6:1   124   MAKESHAPES(T,'                         ');
130   1    6:1   156   MAKESHAPES(T,'                         ');
131   1    6:1   188   ROW:=23;
132   1    6:1   192   MAKESHAPES(E,'X X X X X X X X           ');
133   1    6:1   224   MAKESHAPES(E,'X X X X X X X X           ');
134   1    6:1   256   for I:=1 to 8 do
135   1    6:2   270   MAKESHAPES(E,'X                         ');
136   1    6:1   312   MAKESHAPES(E,'X X X X X X               ');
137   1    6:1   344   MAKESHAPES(E,'X X X X X X               ');
138   1    6:1   376   for I:=1 to 8 do
139   1    6:2   390   MAKESHAPES(E,'X                         ');
140   1    6:1   432   MAKESHAPES(E,'X X X X X X X X           ');
141   1    6:1   464   MAKESHAPES(E,'X X X X X X X X           ');
142   1    6:1   496   MAKESHAPES(E,'                          ');
143   1    6:1   528   MAKESHAPES(E,'                          ');
144   1    6:0   560  end;
145   1    6:0   578
146   1    7:D     1  procedure INIT4;
147   1    7:0     0  begin
148   1    7:1     0   ROW:=23;
149   1    7:1     4   MAKESHAPES(R2,'X X X X X                ');
150   1    7:1    36   MAKESHAPES(R2,'X X X X X X              ');
151   1    7:1    68   MAKESHAPES(R2,'X          X             ');
152   1    7:1   100   MAKESHAPES(R2,'X            X           ');
153   1    7:1   132   MAKESHAPES(R2,'X            X           ');
154   1    7:1   164   for I:= 1 to 3 do
155   1    7:2   178   MAKESHAPES(R2,'X              X         ');
156   1    7:1   220   MAKESHAPES(R2,'X            X           ');
157   1    7:1   252   MAKESHAPES(R2,'X            X           ');
158   1    7:1   284   MAKESHAPES(R2,'X          X             ');
159   1    7:1   316   MAKESHAPES(R2,'X X X X X X               ');
160   1    7:1   348   MAKESHAPES(R2,'X X X X X                 ');
161   1    7:1   380   MAKESHAPES(R2,'X X                       ');
162   1    7:1   412   MAKESHAPES(R2,'X   X                     ');
163   1    7:1   444   MAKESHAPES(R2,'X     X                   ');
164   1    7:1   476   MAKESHAPES(R2,'X       X                 ');
165   1    7:1   508   MAKESHAPES(R2,'X         X               ');
166   1    7:1   540   for I:=1 to 4 do
```

```
167   1    7:2   554   MAKESHAPES(R2,'X           X           ');
168   1    7:1   596   MAKESHAPES(R2,'                        ');
169   1    7:1   628   MAKESHAPES(R2,'                        ');
170   1    7:0   660  end;
171   1    7:0   676
172   1    8:D     1 procedure INIT5;
173   1    8:0     0   begin
174   1    8:1     0   ROW:=23;   (* . . . . . . . . . . . . *)
175   1    8:1     4   MAKESHAPES(M,'X             X         ');
176   1    8:1    36   MAKESHAPES(M,'X X         X X         ');
177   1    8:1    68   MAKESHAPES(M,'X X         X X         ');
178   1    8:1   100   MAKESHAPES(M,'X   X     X   X         ');
179   1    8:1   132   MAKESHAPES(M,'X   X     X   X         ');
180   1    8:1   164   MAKESHAPES(M,'X     X X     X         ');
181   1    8:1   196   MAKESHAPES(M,'X     X X     X         ');
182   1    8:1   228   for I:=1 to 15 do
183   1    8:2   242   MAKESHAPES(M,'X             X         ');
184   1    8:1   284   MAKESHAPES(M,'                        ');
185   1    8:1   316   MAKESHAPES(M,'                        ');
186   1    8:1   348   ROW:=23;    (* . . . . . . . . . . . . *)
187   1    8:1   352   MAKESHAPES(Il,'  X X X X X X X         ');
188   1    8:1   384   MAKESHAPES(Il,'  X X X X X X X         ');
189   1    8:1   416   for I:= 1 to 18 do
190   1    8:2   430   MAKESHAPES(Il,'        X               ');
191   1    8:1   472   MAKESHAPES(Il,'  X X X X X X X         ');
192   1    8:1   504   MAKESHAPES(Il,'  X X X X X X X         ');
193   1    8:1   536   MAKESHAPES(Il,'                        ');
194   1    8:1   568   MAKESHAPES(Il,'                        ');
195   1    8:0   600  end;
196   1    8:0   616
197   1    9:D     1 procedure INIT6;
198   1    9:0     0 begin
199   1    9:1     0  ROW:=23;   (* . . . . . . . . . . . . *)
200   1    9:1     4  for I:=1 to 3 do
201   1    9:2    18  MAKESHAPES(N,'X             X         ');
202   1    9:1    59  MAKESHAPES(N,'X X           X         ');
203   1    9:1    90  MAKESHAPES(N,'X X           X         ');
204   1    9:1   121  for I:=1 to 3 do
205   1    9:2   135  MAKESHAPES(N,'X   X         X         ');
206   1    9:1   176  for I:=1 to 4 do
207   1    9:2   190  MAKESHAPES(N,'X     X       X         ');
208   1    9:1   231  for I:=1 to 3 do
209   1    9:2   245  MAKESHAPES(N,'X       X     X         ');
210   1    9:1   286  MAKESHAPES(N,'X         X   X         ');
211   1    9:1   317  MAKESHAPES(N,'X         X   X         ');
212   1    9:1   348  MAKESHAPES(N,'X           X X         ');
213   1    9:1   379  MAKESHAPES(N,'X           X X         ');
214   1    9:1   410  for I:=1 to 3 do
215   1    9:2   424  MAKESHAPES(N,'X             X         ');
216   1    9:1   465  MAKESHAPES(N,'                        ');
217   1    9:1   496  MAKESHAPES(N,'                        ');
218   1    9:1   527  ROW:=23;   (* . . . . . : . . . . *)
219   1    9:1   531  MAKESHAPES(A,'      X X               ');
220   1    9:1   562  MAKESHAPES(A,'      X X               ');
221   1    9:1   593  MAKESHAPES(A,'    X     X             ');
222   1    9:1   624  MAKESHAPES(A,'    X     X             ');
223   1    9:1   655  MAKESHAPES(A,'  X         X           ');
224   1    9:1   686  MAKESHAPES(A,'  X         X           ');
225   1    9:1   717  MAKESHAPES(A,'X             X         ');
226   1    9:1   748  MAKESHAPES(A,'X             X         ');
227   1    9:1   779  MAKESHAPES(A,'X X X X X X X X         ');
228   1    9:1   810  MAKESHAPES(A,'X X X X X X X X         ');
229   1    9:1   841  for I:=1 to 12 do
230   1    9:2   855  MAKESHAPES(A,'X             X         ');
231   1    9:1   896  MAKESHAPES(A,'                        ');
232   1    9:1   927  MAKESHAPES(A,'                        ');
233   1    9:1   958  ROW:=23;
234   1    9:1   962  for I:=1 to 20 do
235   1    9:2   976  MAKESHAPES(L,'X                       ');
236   1    9:1  1017  MAKESHAPES(L,'X X X X X X X X         ');
237   1    9:1  1048  MAKESHAPES(L,'X X X X X X X X         ');
238   1    9:1  1079  MAKESHAPES(L,'                        ');
239   1    9:1  1110  MAKESHAPES(L,'                        ');
240   1    9:0  1141  end;
241   1    9:0  1168
242   1    9:0  1168  (* main *)
243   1    1:0     0  begin
244   1    1:1     0  INIT1;
245   1    1:1    14  INIT2;
246   1    1:1    16  INIT3;
247   1    1:1    18  INIT4;
248   1    1:1    20  INIT5;
249   1    1:1    22  INIT6;
250   1    1:1    24  PUTBANNER;
                 0  end.
```

MANUAL ADDENDUM

1. New Keyboard

The newer model keyboard requires a different method for performing the SHIFT KEY MOD. There is a wire wrap 25 pin connecter (P1) that connects an adapter board to the keyboard. Instead of soldering to the shift key pad, connect a wire from pin 24 of (P1) to the normal place on the game port (pin 4). This must still be done carefully and by a qualified tech., and may still void your warranty of the computer.

2. Z80 Board

Some of the commands such as the cursor control and window select won't work unless you type "escape" first, followed by the normal SUP'R'TERMINAL command. This is because the control of reading the keyboard is now done by the Z80 card, instead of SUP'R'TERMINAL.

3. PASCAL

To perform the normal SUP'R'TERMINAL commands, such as cursor control and window select etc. . . . from the keyboard:

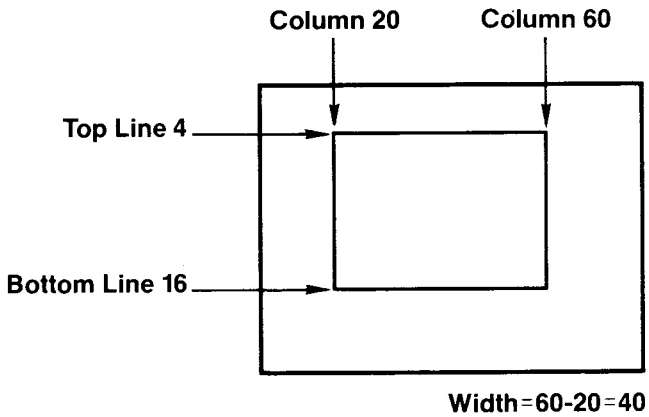
1. Get into the filer
2. Transfer console to system
3. Type in the commands you want
4. Get out of the filer with control C.

4. Cont.—TC

Cont.—TC 4 means
1. Hold the control key down, while you push the "T" key.
2. Release the "T" key, then release the control key.
3. Push and release the "C" key, then push and release the "4" key.

5. Screen Window Altering and Go to XY.

Some people have difficulty understanding how to select a row or column when trying to alter the screen window or going to an XY location from the keyboard or a program. Here is an example of how to alter the screen window



(1.) Set the top to line 4 by doing CTRL-T T $. Hold the control key down, while pushing and releasing the "T" key, then release the control key. Push and release the "T" key. Push and release the $ key. The decision to use the $ key was arrived at by looking at page 26 of the manual. Find the righthandmost column (labeled "CTRL T meaning"). Go down to the 4 in this column because we want to set the top to line 4. Look at the character just to the left of the 4 and notice that it is $. This means that to specify line 4, we must push the $ key. This same method holds true for selecting rows or columns.

(2.) Set window Bottom to line 16 by doing Ctrl-T B 0
zero means 16 from page 26

(3.) Set window left to column 20 by doing Ctrl-T L 4
4 means 20 from page 26

(4.) Set window width to 40 by doing Ctrl-T W H
H means 40 from page 26

The same method of using the righthand column of page 26 can be used from the keyboard or from within a program. This also works the same way for the go to XY function.

6. What is the difference between REV 1.0 and 2.2 of SUP'R'TERMINAL?

Rev 2.2 of the EPROM(U14) has the following features:

1. Interprets VTAB and HTAB Basic Commands. However, CALL-936 and HOME must still be modified in Basic programs. This can be done automatically by a program available on our dealer disk.

2. Control N now gives] regardless of whether you are in the control V mode or not.

3. Expanded keyboard characters are now available if you have performed the shift key modification and are in the control V mode. Control shift A through J gives you the ASCII characters you previously couldn't get.

Control Shift	A	\		F	|
	B	]		G	{
	C	^		H	~
	D			I	del
	E	{		K	@