Terminal Emulator

HP System 45 Desktop Computer

Hewlett-Packard Desktop Computer Division
3404 East Harmony Road, Fort Collins, Colorado 80525

Table of Contents

Introduction

About Terminals:

To better understand the use of the 9845B Terminal Emulators, one should be aware of some general information about terminals. There are several types of terminals now available. One type is the Remote–Job–Entry (RJE) terminal. The RJE terminal is used mainly for submitting batch jobs to a main computer, and receiving output. Examples of RJE terminals are card readers and line printers. Another type is the data collection terminal, which is used primarily for transfer of data to a mainframe for processing. A third type of terminal is the time–share terminal. The time–share terminal is used for interactive 'sessions' with a main processor. It is distinguished from the other two types of terminals because the interactive sessions involve a 'conversation' between the user and the main computer. That is, a time–share terminal receives and displays questions from the computer, and sends back answers typed in by the user, and vice versa. Time–share terminals also generally allow you to create, edit, and run programs, and to save or fetch them using files on the main computer system.

The 9845B Terminal Emulator programs are intended to emulate a time–share terminal; that is, they allow a user to log on to a time–share system, and to use whatever features that particular system has to offer, such as an editor for creating and editing programs, computer assisted instruction courses, problem solving programs, etc. The Terminal Emulator programs do not emulate RJE terminals or data collection terminals. However, using the tools provided by the I/O ROM, and the 98036A card, as well as some of the routines provided by the Terminal Emulator programs, the user may find it possible to write software to emulate these types of terminals also. Another characteristic of the emulators is that they only work with ASCII characters. However, if the need arises, the user may incorporate a conversion table into the terminal emulators which will convert whatever type of code his computer uses to ASCII.

The 9845B Terminal Emulators are capable of asynchronous data transfer only. This means that the data transfer is character oriented, and each character has a leading bit and trailing stop bits. The 'asychronous only' condition is a property of the 98036A interface, not the programs themselves.

The Terminal emulator programs can emulate an interactive graphics terminal as well. The emulated terminal has three distinct modes of operation. In Alpha mode the terminal will respond as a normal time-share terminal. This mode uses the alphanumeric area (raster) of the CRT display. The other two modes utilize the graphics raster. When the terminal is in Graphics mode it can draw vectors sent down the data link from the main processor. It can also send coordinates to the main processor by returning the current position of the graphics cursor. The cursor itself may be moved to any portion of the CRT.

The third mode is Graphics-alpha. In this mode the graphics raster is retained but the terminal reacts much as it would in alpha mode. Communications to and from the main processor are displayed on the CRT. These character strings will begin at the left side of the CRT with each new line printed below the previous line. Should the bottom of the screen be encountered a new margin is established at the center of the screen. The alpha cursor will be moved to the top of the screen at the new margin. When the bottom of the screen is encountered and the margin is set to the center of the CRT the new margin will be set to the extreme left of the CRT once more.

The alpha mode is entirely independent of the graphics and graphics-alpha modes. The graphics and alphanumeric rasters can be interchanged without loss of information. If a figure is drawn in graphics mode graphics-alpha information may be printed over it. However, if alpha mode is entered the graphics raster is replaced by the alphanumeric raster. The graphics raster will remain the same until graphics or graphics-alpha mode is again entered. The user can switch between the graphics raster and alpha raster by pressing SFK 14. The host computer controls the graphics and graphics-alpha modes when the graphics raster is present. It does this by sending control characters to the terminal emulator.

An alternative plotter may be defined in SETUP (UDK 10). This plotter may then be used by turning it on and off using the escape codes specified in the ESCAPE CODES section (Esc &p1P and Esc &pOP). When the alternative plotter is not on graphics output will go to the CRT. When the alternative plotter is on then entering Graphics mode means that all plotting and printed information will go to the plotter and NOT the graphics raster of the CRT.

When the terminal is in LOCAL mode and the SEND FILE key (UDK 12) is pressed, the data read from the defined file (see Shift UDK 11) will be acted upon as if it were being sent from the computer. Thus, if a series of graphics commands have been previously stored in a data file (using the record fle feature), they can later be reprocessed in LOCAL mode to create graphics on either the CRT or alternative plotter. This allows you to create plots off-line (not connected to the computer), often resulting in reduced time-share costs.

System Configuration

9845B desktop computer

I/O ROM (Option 312 or 98412A)

98036A interface (either the standard cable, or Opt. 001)

Graphics (Options 700, 311))

A hardcopy printer is optional, as the CRT is capable of displaying the computer's output. If the internal printer is used, or the 2631, or the 7245, or any printer which can print control characters (ASCII 0 through ASCII 31), it is possible to use the "display functions" feature of the emulators (see the sections marked "Key Definitions:"). However, for standard printed output, any printer that can be interfaced to the 9845A will suffice nicely (such as the 9866B, or the 9871A).

There are a set of escape sequences (explained further on in the manual) which allow the main computer to access local mass storage devices via the 9845B. If the built in tape cartridge(s) do not satisfy the user's particular data storage/retrieval needs, additional mass storage devices (and the Mass Storage ROM) may be added to the system with absolutely no required changes in the Terminal Emulator Programs.

NOTE

If you do not have graphics and desire to run the terminal emulator, it is possible to modify the program to emulate an alphanumeric terminal only. Instructions follow.

---

a) Follow the directions in the manual for loading the appropriate file.

b) Ignore error messages ("MISSING ROM" messages will replace statements containing graphics keywords in the program)

c) Change the default for power-on for compatability mode to "off" by:

1) For "KMGTRM"

a) Change line 360 to: 360 Handshake=1

b) Change line 3910 to: 3910 Comp_mode: GOTO Bad ! turn compatability mode on

2) For "LMGTRM"

a) Change line 340 to: 340 Handshake=1

b) Change line 3930 to: 3930 Comp_mode: GOTO Bad ! turn compatability mode on

d) Type: RE-STORE "KMGTRM" or RE-STORE "LMGTRM"
Press: EXECUTE

The default will now be compatability mode off and no graphics statements should be executed.)

Part Numbers

09845-10140 Complete Pack

The complete pack includes the following:

09845-10141 Manual (includes Source Documentation)
09845-10144 Program Cartridge

7120-7839 Special Function Key Overlay

9282-0689 Binder

Spine-(insert)

Source Documentation: Included in manual.

Special Considerations

Interfacing Requirements

The Terminal Emulator requires an HP 98036A Bit-Serial Interface to connect to the main computer. If you are using a modem, the interface will connect to the modem. If you have a direct line, connect the interface cable to that. Refer to the 98036A Serial I/O ROM manual for details on cable options and connections. The select code and baud rate are set with switches on the 98036A. Refer to the 98036A manual for details (Part No. 98036-90000). The select code on the printer interface (if applicable) and the alternative plotter (if applicable) must be different from the select code on the 98036A card.

Definitions

The following definitions will be used in describing the capabilities of the Terminal Emulators:

Echo

The computer sends back an exact copy (echo) of everything it receives from the terminal. This is a method of verifying what is sent from the terminal.

Full Duplex Terminal

Keys that are struck on the terminal keyboard are sent only to the computer. Characters received from the computer are displayed and printed. In this case, it is the responsibility of the computer to "echo back" the characters it receives so that the user can view what was entered. If the computer does not "echo back" to a full duplex terminal, the user will not be able to view what was entered.

Half Duplex Terminal

Keys that are struck on the terminal keyboard are sent to the display device as well as to the computer. Characters that are received from the computer are also displayed and/or printed. When a terminal operates in half duplex mode, it assumes the computer does not "echo-back" characters. If the computer does "echo back" to a half duplex terminal, every character will appear twice.

Full Duplex Line

Data may flow in both directions simultaneously on a full duplex line.

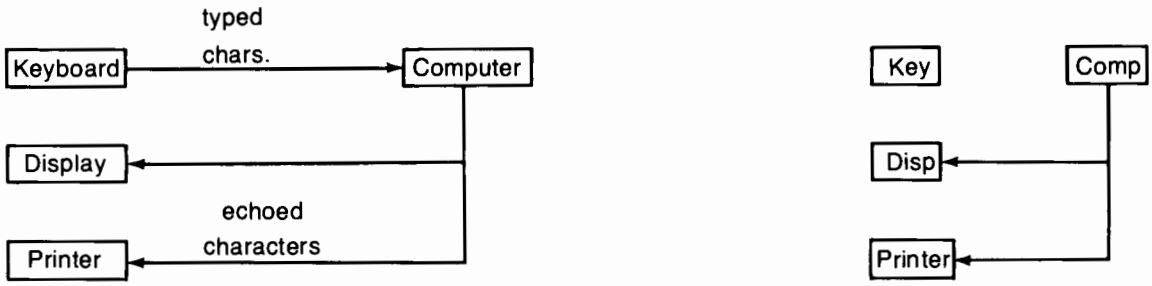
Half Duplex Line

Data may flow in only one direction at a time. The terminal and the computer must decide which direction is currently operative.

The following diagram illustrates the differences between Full and Half Duplex terminals:

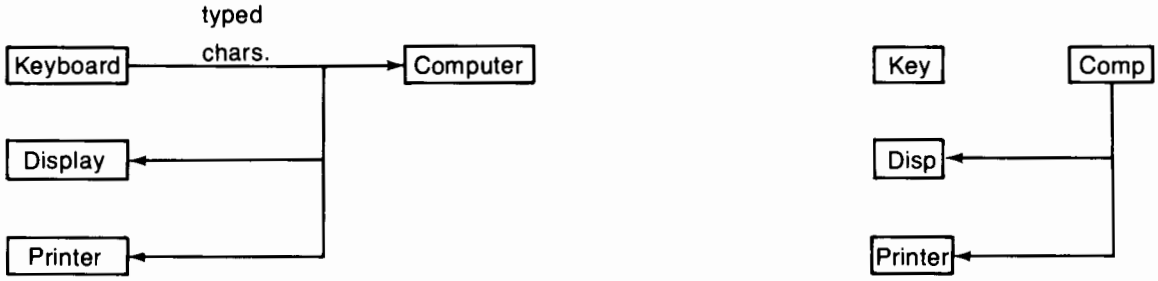
Full Duplex Terminal:

Input to Computer **Output from Computer**



Half Duplex Terminal:

Input to Computer **Output from Computer**



The terminal emulators can emulate either a full duplex terminal (if the computer "echos-back") or a half duplex terminal (if the computer doesn't "echo back"). However, the terminal emulator REQUIRES a full duplex line for data communications.

Time-Share System Requirements:

Before you can use the Terminal Emulator, you should know the following things about the time-share service you intend to use (all the items listed below will default to the given values):

1. The transmission speed or bit rate of your system. Some systems can use several different bit rates (default is 300 baud). The baud rate must be set on the back of the 98036A (refer to the 98036 card manual).
2. The parity requirements (odd,even,or no parity) for your system (default is no parity).
3. End-of-line protocol. Does the system require a carriage return and line feed at the end of each line, just a carriage return, or some other character sequence?(The default end-of-line sequence is a carriage return.)
4. Does the system "echo-back" (See Full Duplex vs. Half Duplex.) (The default is echo on.)
5. The number of bits per character that your time-share system accepts. This will normally be 8 when no parity is used, and 7 when even or odd parity is used (default is 8 bits per character).
6. The number of stop bits your system accepts. Stop bits are the "empty" bits sent between each character to separate the characters. Normal stop bit values will be 1,1.5, or 2 (default is 1).
7. The log-on procedure for the time-share system.

Which Program to Use:

Three programs are included in this Applications Package. All three are Terminal Emulators, but there are some differences between them.

One program is called "key mode". This program closely simulates a teletype in that each key is sent to the computer as soon as it is pressed. This does not allow for any keyboard editing other than that provided by the computer you are using. The backspace key and the "rubout" special function key (see key definitions) may be useful, depending on the particular system.

The second program is called "line mode", and allows you to type in an entire line (the user can decide how long the line will be), and edit the line locally, if required, before sending it to the computer. Most applications of the Terminal Emulator will be able to use the line mode program. However, if your computer requires a certain wait period between characters, or if you are more comfortable with a teletype, you should use the "key mode" program.

The third program is also a line mode program. However, none of the special function keys are enabled. The sophisticated features of the other two programs have been taken our, leaving a skeletal program which the user can use as an example on using the 98036A card. The program utilizes a set of modular subprograms which the user may use independently of the provided program if he has a specialized application.

Line Mode Program

Using the line mode program (LMGTRM), it is possible to type an entire line of characters, edit them locally if need be, and send the entire line to the computer (use the STORE or EXECUTE key to send lines). Lines sent from the computer to the terminal will appear on the CRT. Optionally, the lines may also be made to appear on a hardcopy printer also.

Key Definitions:

All the special function keys indicated on the overlay are defined in this program. In addition, some of the other keys on the 9845 keyboard have been redefined for special use with the Emulator program. Keys that are undefined will cause a beep, but will otherwise be ignored (such as the CONT key).

The following keys perform the functions indicated:

Typewriter (TYPWTR)	This key turns on/off typewriter mode. For an explanation of the typewriter key, refer to the 9845 Operating and Programming manual.
Recall	Recalls non-null entries, starting with the latest entry, and going back to the earliest entry. The recall buffer will hold up to 1000 characters.
Shift Recall	Behaves similarly to Recall, only in reverse. A pointer is maintained which allows the user to step through the recall buffer in both directions.
Delete char (DEL CHR)	Deletes one character in the entry line.
Insert char (INS CHR)	Enters/leaves "insert character mode" (acts as a toggle). "Insert character" allows characters to be inserted to the left of the cursor position. "Insert character " mode may be exited by pressing the INS CHR key again, or any other editing key or special function key.
Clear	Clears screen and entry line.
Clear line	Clears the entry line.
Clear to end (CLR-END)	Clears to end of entry line.
Home	Moves cursor to left end of entry line.
Backspace	Moves cursor one position left and clears remainder of line.
Left-arrow	Moves cursor one position left.
Right-arrow	Moves cursor one position right.
Shift Left-arrow	Deletes the first character of the prompt, if any.

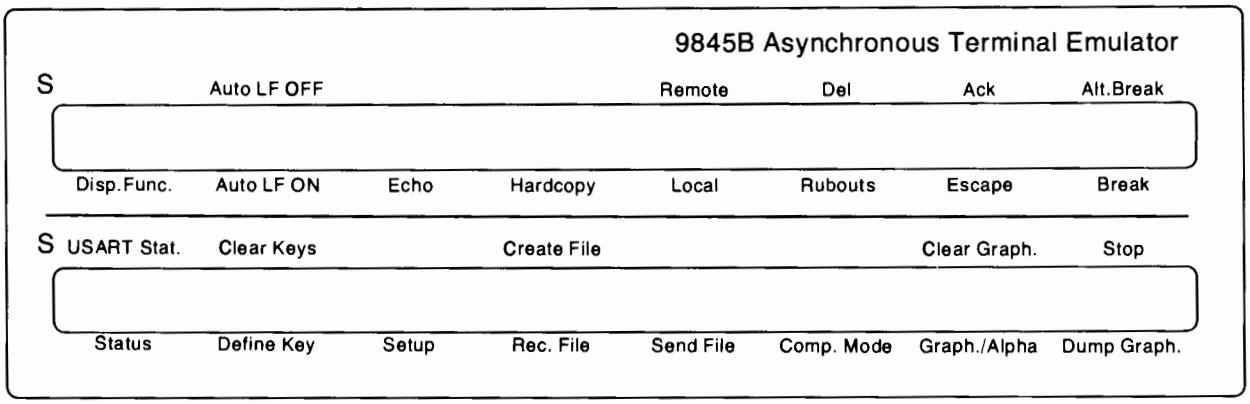
Shift Right-arrow	Inserts a space before the prompt.
Up-arrow	Moves screen up one line.
Down-arrow	Moves screen down one line.
Roll up	Moves screen up five lines.
Roll down	Moves screen down five lines.
Return (STORE, EXECUTE)	Sends the input buffer and a carriage return to the computer. The end-of-line characters may be changed by using the [Define Keys] special function key (i.e., UDK 10). The STORE key will not send the line unless a DC1 has been received (indicating a prompt); EXECUTE will send it anyway. To redefine the DC1 prompt character, see the section of the manual titled "Modifications".
STOP	Immediately stops program. This is alright to do, except it will not close the data comm channel or exit typewriter mode. (Note: Do not confuse the STOP key with UDK 31 (shift UDK 15).)
Tab	Advances the cursor to the next tab stop. Tab stops are at multiples of 8, and are not redefinable using the TAB SET and TAB CLEAR keys.

The following keys perform the functions specified in graphics mode:

Left-arrow	Moves cursor 14 units left.
Right arrow	Moves cursor 14 units right
Shift Left-arrow	Moves cursor 1 unit left.
Shift Right-arrow	Moves cursor 1 unit right.
Up-arrow	Moves cursor 10 units up.
Down-arrow	Moves cursor 10 units down.
Shift Up-arrow	Moves cursor 1 unit up.
Shift Down-arrow	Moves cursor 1 unit down.
Return (STORE,EXECUTE)	Sends the input buffer and the coordinates of the graphics cursor to the computer followed by the graphics input terminator. The graphics input terminator can be specified by using the Setup key (UDK 10). The STORE key will not send the line unless a DC1 prompt character has been received (indicating a prompt); EXECUTE will send it anyway. To redefine the DC1 prompt character, see the section titled "Modifications".
STOP	Same function as STOP in alpha mode.

The following keys perform the functions indicated in graphics-alpha mode:

Left-arrow	Moves cursor 1 character position (14 units) left.
Right-arrow	Moves cursor 1 character position (14 units) right.
Return (STORE,EXECUTE)	Same function as Return in alpha mode.
STOP	Same function as STOP in alpha mode.



The special function keys are defined as follows (refer to overlay):

Display functions (UDK 0) (toggle)	Turns on/off the display functions feature of the hardcopy printer. Default is display functions off.

NOTE

With the display functions feature turned on all ASCII control characters (decimal 0 through 31) are printed, rather than being executed). This feature will not work if the hardcopy printer does not have this capability.

Auto LF on (UDK 1)	Changes the end-of-line sequence to CR/LF.
Auto LF off (UDK 17 or Shift UDK 1)	Changes the end-of-line sequence to CR.

Echo (UDK 2) (toggle)	This key tells the terminal emulator that the computer does/does not "echo back" what the terminal sends it. If echo is off but the computer does "echo back", then everything you type in will appear twice. If the assumption that the computer does not have "echo back" is correct, however, then what you type will appear only once. If echo is on and the computer does not "echo back" then the commands the user sends will not appear at all. Default is echo on.
Hardcopy (UDK 3) (toggle)	Turns on/off the hardcopy printer. Default is hardcopy off.
Local (UDK 4)	Puts the terminal in local mode. Messages from the computer are ignored and messages which would be sent from the terminal are processed as if the computer had sent them to the terminal. Local mode is useful for giving escape sequence commands from the keyboard, or for ignoring large amounts of text sent from the computer. It can also be used for off-line processing of data, including graphics data.
Remote (UDK 20 or Shift UDK 4)	Resumes communication with the computer.
Rubouts (UDK 5) (toggle)	Disables/enables printing of rubouts. Default is rubouts not ignored. Rubouts may be sent by the computer as components of graphics coordinates. When rubouts are disabled they will be ignored. Instead <Esc ?> will be interpreted as ASCII 127 for graphics purposes. (Do NOT disable rubouts if the computer you are using uses DEL's in graphics commands).
Del (UDK 21 or Shift UDK 5)	Rubout or "Delete" character (ASCII 127).
Escape (UDK 6)	Escape character (may also be obtained by pressing CONTROL key and the "[" key simultaneously).
Ack (UDK 22 or Shift UDK 6)	Sends an ACK character (ASCII 6) to the computer as soon as it is pressed.
Break (UDK 7)	Sends BREAK signal to the computer as it is pressed.

Alt Break (UDK 23 or Shift UDK 7)	Sends a subsystem break (control Y) to the computer as soon as it is pressed. This break character may be changed by using the "Define Key" special function key (UDK 9).
Status (UDK 8)	Prints the status of the emulator program.

When the terminal is initially turned on this will be its status:

Status of Terminal Emulator

Echo on	Hardcopy on	Rubouts not ignored	Display funcs.off
Recording off	Local off	Cursor at 1	Background off
Prompt D_1(ASCII 17)	Keyboard lock off	Hard. printer at 0	
Autosend off	Baud rate 300	1 stop bit	Compatability on
8 bits per character		I/O card at 11	
Parity (even) disabled		Bit rate factor is 1/64	
Handshake on		Graphics input term.:	
No alternative plotter specified		[cr]	
File name: DCdata:T15		File size: 30	
Current line:			

Key Definitions

[sh UDK7]:	[em]
[sh UDK1]:	[UDK9][store] [cr] [UDK9]
[UDK6]:	[esc]
[sh UDK6]:	[ack]
[UDK1]:	[UDK9] [store] [cr] [lf] [UDK9]
[sh UDK5]:	▒
[end-of-line]:	[cr]

USART status control word (R4E):

Data set ready	Zero (Unused)	Framing error	Overrun error	Parity error	Transmtr empty	Receiver ready	Transmtr ready
0	0	0	0	0	1	0	1

USART
(UDK 24 or
Shift UDK 8)

Prints the status of the USART word in the 98036 card (register R4e) and tests the Data Set Ready line (given by the most significant bit in the status word) to see whether or not it is active. It should be noted that the DSR line is only used with Option 001 of the 90036 card. If you are using the standard option, the leading bit of the status register corresponds to the Request to Send line.

Define Key (UDK 9)

Enables any key except Break (UDK 7) or Define Key (UDK 9) to be redefined as any sequence of keystrokes (including other redefined keys). The definitions of STORE, EXECUTE, Ack (UDK 9), Stop (UDK 31) and Alt break (UDK 23) may be changed, but the essential function of these keys will not be affected. The "define key" mode is operated using the following sequence.

1. Press: UDK 9 (Define Key)
2. Press: The key you want to redefine
3. Type: The sequence of keystrokes you want to use to replace the given key
4. Press: UDK 9 (Define Key) to end the definition.

NOTE

The program will issue the message **Recursion level too great - infinite loop likely** if there are too many keystrokes (160) as a result of pressing a re-defined key.

To clear a single key's definition without affecting other redefined keys, use the following sequence:

1. Press: UDK 9 (Define Key)
2. Press: The key you want to clear
3. Press: The same key (again)
4. Press: UDK 9 (Define Key) to end the definition.

NOTE:

The keys Up-arrow, Down-arrow, Roll up, Roll down, Typewriter and STOP are executed by the system so they are not redefineable and cannot be used in definitions.

Clear Keys
(UDK 25 or
Shift UDK 9)

Clears all definitions produced by using the Define Key special function key and restores all default conditions.

Setup
(UDK10)

Allows the user to change the parameters of the datacomm line, as well as the hardcopy printer. This key will cause the live keyboard to be re-enabled while the user answers the questions the program will ask. When the operation is complete, the live keyboard will again be locked out.

1. When **Enter select code of I/O card:** appears in the display area:

   a. Enter: The select code of the 98036A card

   b. Press: CONT

2. When **What is the baud rate:** appears in the display area:

   a. Enter: The baud rate you are using

   b. Press: CONT

NOTE

If you wish to change baud rates, you must select the proper switch position on the 98036 card. This input is used only for error checking on the bit rate factor and will not cause the actual baud rate to be changed.

---

3. When **Bit rate factor (1,1/16,1/64):** appears in the display area:

   a. Type: 1,1/16, or 1/64 (for baud rates below 4800, use 1/64. For 4800 and 9600 (possible only with enq/ack handshaking) use 1/16).

   b. Press: CONT

4. When **Number of bits per character:** appears in the display area:

   a. Enter: The number of bits per character your computer recognizes

   b. Press: CONT

5. When **Is parity enabled?** appears in the display area:

   a. Type: Y or N (Y implies that parity is enabled, N implies that parity is disabled-the default is N)

   b. Press: CONT

6. If **Is parity even?** appears in the display area:

   a. Type: Y or N (Y implies even parity, N implies odd parity-the default is Y)

   b. Press: CONT

7. When **Enter alternative plotter description (such as 9872A or GRAPHICS):** appears, then:

   a. Type: <name of plotting device>(if no alternative plotter is to be specified, go to step b.)

   b. Press: CONT

8. When **Enter plotter select code (and HPIB address, if applicable):** appears:

   a. If there is no HPIB then Type: <selctcode> or

      If there is an HPIB then Type: <selectcode>, <HPIB address>

   b. Press: CONT

9. When **Graph. terminator: Enter (1) for** [cr] (2) for [cr][eot] (3) for none appears in the display area:

   a. Choose the desired graphics input terminator. The graphics terminator is sent following the graphics cursor coordinates when the coordinates are requested by the host computer. (see [illegible])

      1. Type 1 (for a carriage return only-this is the default) or
      2. Type 2 (for a carriage return and end of transmission) or
      3. Type 3 (for no graphics input terminator)

   b. Press: CONT

10. When **Is handshake for autosend enabled?** appears in the display area:

    a. Type: Y or N (Y implies that when a file is sent (UDK 11) only one line at a time will be sent and the terminal will wait for a prompt before sending the next line; N implies that the entire file should be sent without response from the computer; although the computer may temporarily halt the sending of the file by outputting an ASCII character 19 - this is necessary in some cases to prevent loss of information. Transmission is continued when the computer sends a DC1 (ASCII 17). This is sometimes called Xon and Xoff.) Default is handshake on.

    b. Press: CONT

11. When **Enter printer select code(and HPIB address, if applicable)** appears in the display area:

    a. If you are using a non-HPIB printer:

       1. Enter: The printer select code
       2. Press: CONT
       3. Wait for the **Terminal Ready...** message and then proceed normally.

    or

    b. If you are using an HPIB printer (i.e.9871A or 2631A):

1. Enter: The printer select code, followed by a comma, followed by the bus address (e.g.7,2)
2. Press: CONT
3. Wait for the **Terminal Ready...** message and then proceed normally.

NOTE:
The select code of the CRT is 16 and the select code of the internal printer is 0.

Record File
(UDK 11)
(toggle)

Opens data file storage and starts recording or, if in the process of recording, stops recording and closes file. This key has the effect of entering Local mode, executing the escape sequence (Esc &p1D or Esc &p0D) and entering Remote mode (if this was previously the state of the terminal). See the section titled "Escape Codes" for further details.

NOTE
Leaves the terminal in Local or Remote mode depending on state prior to pressing UDK 11). Default is file closed.

Define File
(UDK 27 or
Shift UDK 11)

Allows redefinition of the data file. The default data file is "DCdata:T15" with a size of 30 records. When this key is pressed the keyboard is re-enabled while the user answers the questions of the program. When the operation is complete, the live keyboard will be locked out.

1. When **Enter file name:** appears in the display area:
   a. Enter: The file name desired (may or may not include the mass storage device code). For example, FILE or FILE:T15 or
   b. Press: CONT (If the file already exists then that file becomes the new data file).
2. If the file does NOT exist then when **Enter size of file** appears in the display area:
   a. Enter: the number of records in the file desired.
   b. Press: CONT (The file is created and becomes the new data file)

NOTE

Data files should not cross the tape track boundary as the necessity of rewinding the tape at this point may cause loss of information.

Send File (UDK 12) (toggle)

Puts the terminal in an "Auto send" mode or, if the terminal IS in "Auto send" will exit that mode. This key has the effect of entering Local mode and executing the escape sequence (Esc &p1s or Esc &p3s). For more information, see the section entitled "Escape Codes". Default is not auto send. If Handshake is enabled (see Setup; UDK 10), then one line will be sent out each time a prompt is received from the computer. If Handshake has not been enabled, then the entire file will be sent without waiting for any response from the main processor unless an ASCII character 19 is received. If DC3 (ASCII 19) is received, then transmission of the file is temporarily halted until a DC1 (ASCII 17) is received. Transmission is then continued.

If the Send File key is pressed when the terminal is in LOCAL mode, then the information read from the data file is interpreted as if it had been sent from the computer. If commands from the computer have been recorded, then the terminal act as it did when it was receiving those commands.

Compatability mode (UDK 13) (toggle)

Turns on/off compatability mode. If compatability mode is on, then the graphics and graphics-alpha modes may be entered. If compatability mode is off, then the graphics raster is not available. All User Defined Keys and escape sequences involving graphics (for example, UDK 15, Dump graphics, and Esc [etb]) are disabled. The terminal will act as a normal time-share terminal and not as a graphics terminal. Default is compatability mode on.

Graphics/Alpha (UDK 13) (toggle)

If the terminal is in alpha-mode this key will cause it to enter graphics-alpha mode. If the terminal is in graphics or graphics-alpha mode this key causes it to enter alpha mode. Default is alpha mode. This key is disabled if compatability mode is off.

Clear Graphics (UDK 29 or Shift UDK 13)

This key causes the graphics raster to be cleared. This key is only enabled in graphics or graphics-alpha mode.

Graphics/Alpha (UDK 13) (toggle)	If the terminal is in alpha-mode this key will cause it to enter graphics-alpha mode. If the terminal is in graphics or graphics-alpha mode this key causes it to enter alpha mode. Default is alpha mode. This key is disabled if compatability mode is off.
Clear Graphics (UDK 29 or Shift UDK 13)	This key causes the graphics raster to be cleared. This key is only enabled in graphics or graphics-alpha mode.
Dump graphics (UDK 15)	This key causes the graphics raster to be dumped (printed by) the internal thermal printer. This key is enabled in graphics or graphics-alpha mode only.
Stop (UDK 31 or Shift UDK 15)	Stops the terminal emulator program, first sending any message that happened to have been designated using the DEFINE KEY operation.

Escape codes:

There are several escape code sequences which can produce various functions when sent from the host computer (or when typed in from the terminal in local mode). Several of the escape codes deal with opening, closing, purging, or linking of data files. The data file which is assumed to be referenced by these commands stays the same from one command to the next unless the data file name is redefined (using ESC &pG). The default file name is "DCdata:T15". The default file size assumed when opening a file is 30 physical records. The file size may be redefined using the ESC &pA sequence.

ESC &p0D	Closes the data file and stops recording.
ESC &p1D or ESC &p2D	Opens data file on mass storage and starts recording. The data file used will be either the default "DCdata:T15" or the most recent override of the default. The file size of the data file will either be the default (30 physical records) or the most recent override of the file size. Once this command is encountered, everything the computer sends will automatically be recorded on the mass storage file, as well as being printed on the screen file, as well as being printed.
ESC &p3D	Turns off the hardcopy printer and closes the data file.
ESC &p4D	Turns on the hardcopy printer.
ESC &p5D	Turns off the hardcopy printer.

⌐ &pG	Sets the data file name to the value of the current line being sent from the computer (or from the keyboard if the program is in local mode) and then clears the line. The default file name is "DCdata:T15". Notice that any mass storage device may be used, provided the Mass Storage ROM is plugged in, and that particular device is interfaced to the 9845 properly. Example: In order to get all Mass Storage escape sequences to refer to the file "DUMMY" on a floppy disk at select code 8, the following string would be sent from the computer (or from the keyboard if the emulator is in local mode): DUMMY: F8 ⌐&pG
⌐ &pA	Sets the file size of the data file (used when opening the file using ⌐ &p1D) to the value of the current line being sent from the computer, and then clears the line. Example: To set the file size to 50 physical records, the following string should be sent from the computer (or from the keyboard if the terminal emulator is in local mode): 50 ⌐&pA
⌐ &pN	Purges the data file.
⌐ &pE	LINKS the data file into memory after the terminal emulator program. The program which is linked in should not use arrays that aren't allocated in the terminal emulator program, because a LINK will not cause arrays (or strings) to be allocated. If arrays are needed, it will be necessary for the user modify the terminal emulator program so that it will allocate his arrays before the data file is linked into memory.
⌐ &cE	Transfers control to the program which has been previously linked via ⌐&pE. In order to return control to the terminal emulator when the program is done, the statement "GOTO Resume" should be used.
⌐ &p1S or ⌐ &p2S	Puts the terminal in an "Auto send" mode. Lines from the data file (defined using ⌐&pG) are sent to the computer automatically upon receiving a prompt character. (Refer to the section under "Modifications" for instructions on changing the prompt character.) This mode is exited upon three conditions: 1) End of file is reached, 2) Non-string data is encountered, or 3) An ⌐&p3S is received from the host computer.
⌐ &p3S	Takes the terminal out of "Auto-send" mode.

ESC &d@	Turns off IV, BL and UL.
ESC &dA	Turns on blinking (BL)
ESC &dB	Turns on inverse video (IV)
ESC &dC	Turns on IV and BL
ESC &dD	Turns on underline (UL)
ESC &dE	Turns on UL and BL
ESC &dF	Turns on UL and IV
ESC &dG	Turns on UL, IV and BL
ESC Y	Turns on the display functions feature of the hardcopy printer.
ESC Z	Turns off the display functions feature of the hardcopy printer.

Several of the above escape sequences have the side effect of automatically putting the emulator in REMOTE mode if it was in LOCAL mode to begin with. These sequences are: ESC&cE and ESC &pE

Graphics escape codes:

[ESC] &p1P	Turn on alternative plotter.
[ESC] &p0P	Turn off alternative plotter.
[ESC] &s1p0Q or [ESC] &s0p1Q	Turn compatability mode on.
[ESC] &s0p0Q	Turn compatability mode off.
[ESC][GS][ESC][ETB]	Send graphics cursor position to the computer.
[ESC]][GS]	Send graphics cursor position preceded by one ASCII character to the computer.
[ESC][GS]	Dump graphics.
[ESC][FF]	Clear screen, enter graphics alpha mode and home cursor.
[ESC][ETB]	Read status and cursor position. Status includes the status of the hard copy unit, the current left margin and whether the mode is graphics or graphics-alpha. The information sent to the computer is: <status byte><Hi X><Lo X><Hi Y><Lo Y><graph. terminator>.
	Status byte: 1 0 1 1/0 0 0/1 0/1 1
	Hard copy unit

	0 = not ready
	1 = ready
	Mode
	1 = Graphics
	1 = Graphics-alpha
	Margin
	1 = margin 1
	0 = margin 2
[Esc][GS]	Enter graphics mode.
[Esc]?	Character 127 (rubout) when plotting if rubouts disabled.
[GS]	(CONTROL/=) Enter graphics mode (same as Esc [GS]).
[US]	(CONTROL/?) Enter graphics-alpha mode.
[BS]	(CONTROL/H) Moves cursor one space left (14 units).
[HT]	(CONTROL/I) Moves the cursor one space right (14 units).
[CR]	Enter graphics-alpha mode and execute a carriage return.
[LF]	(CONTROL/J) Moves the cursor down 1 line (21 units).
[VT]	(CONTROL/K) Moves the cursor 1 line up (21 units).

Opening, closing, purging or chaining requires a certain amount of time to maneuver the tape drive, so it is necessary to implement a software handshake in order to keep the terminal emulator and the computer synchronized. The completion of an operation is signalled by either "S" (successful), or "F" (failure) followed by the end-of-line character(s). Closing (Esc &p0D) cannot be unsuccessful (i.e. an "S" will always be returned). It is up to the user to make the computer he is talking to cooperate with the 9845 terminal emulator when using the above escape sequences. The software handshake outlined above makes it easy to synchronize the two using a program which runs on the main computer. If the program were in BASIC, for example, an INPUT statement could be used to check whether a tape operation was successful or unsuccessful, since the terminal emulator will automatically send an "S" or an "F" upon completion of the commands given by the escape sequences.

The example program below shows how the escape sequences might be used from a computer which has a BASIC language compiler or interpreter.

```
10   REM FIRST RESERVE A STRING TO HOLD THE ESCAPE CHARACTER
20   E$=CHR$(27)
30   REM NEXT REDEFINE THE FILE NAME AND FILE SIZE
40   REM NEW FILE NAME IS "DUMMY:T15"
50   REM NEW FILE SIZE IS 1 PHYSICAL RECORD
60   PRINT "1";E$;"&pA"
70   PRINT "DUMMY:T15";E$;"&pG"
80   REM NOW OPEN THE DATA FILE AND START RECORDING
90   PRINT E$;"&p1D"
100  INPUT A$
110  IF A$="S" THEN 140
120  PRINT "PROGRAM FAILURE"
130  STOP
140  REM PRINT SOME NUMBERS ON THE TAPE
150  PRINT 1,2,3,4,5
160  REM NOW CLOSE THE DATA FILE AND STOP RECORDING ON IN IT
170  PRINT E$;"&p0D"
180  INPUT A$
190  IF A$="F" THEN 120
200  PRINT "PROGRAM SUCCESSFUL"
210  STOP
220  END
```

User Instructions

1. a. Make sure that the I/O ROM's are properly installed.
   b. Make sure the 98036A card is plugged into one of the I/O slots in the back of the machine.
   c. The select code of the 98036 card should be set to some value between 1 and 12, since select codes 0, 13, 14, 15, and 16 are reserved.
   d. Make sure the power is turned on.
   e. Insert the Terminal Emulator cartridge into the primary tape transport (i.e. the drive above the special function keys).

2. Load the line mode program into memory.
   a. Type: LOAD "LMGTRM"
   b. Press: EXECUTE

3. When the program is loaded (the busy light will go off and the tape will stop moving):
   a. Press: RUN

4. The program will begin initializing. The message **Please wait** will appear in the display area of the CRT. After a few seconds, the message **Terminal ready on 11** should appear in the display area if your system is correctly configured (11 will actually be replaced by whatever select code the 98036 card is set at). If the **Terminal Ready...** message appears, go to step 8. If the Terminal Ready... message does not appear, then an error has been detected. The error should result in a message outlining by one of the following three steps (5, 6, or 7).

5. If the message **What is the select code?** appears in the display area, the program has detected the presence of more than one 98036 card. It is only possible to open one data communication channel at a time.

   a. Enter: The select code of the card you wish to use

   b. Press: CONT

   c. Follow the instructions in step 4.

6. If the message **There are no 98036 cards present. Please insert one.** appears in the display area, make sure the card is plugged securely into one of the I/O slots in the back of the machine. Once the card is plugged in correctly, the program will detect its presence and you may proceed normally as outlined in step 4.

7. If the message **Hardcopy printer not operational** appears in the display area, the internal printer is either 1) missing, or 2) out of paper. If the printer is out of paper, put a new roll of paper in the printer before proceeding. On the other hand, if it happens to be the case that your machine does not have an internal printer, you will be asked to enter a different select code.

   a. When **Enter printer select code (and HPIB address, if applicable)** appears in the display area:

      1. Enter: The printer select code

      2. Press: CONT

NOTE 1:

If you are using an HPIB printer, it will be necessary to enter two numbers in part 7.a.1. above. The first number will be the select code, followed by a comma, followed by the HPIB address (example: 7,1).

NOTE 2:

The select code of the internal printer is 0. The select code of the CRT (if you do not desire to use a hardcopy printer at all) is 16.

---

   b. Once the new printer select code has been entered, the error checking routines outlined in step 4 will be repeated.

8. Once the message **Terminal Ready on 11** appears in the display area, the terminal emulator is ready to use. At this point, you will probably have to establish your communications linkage (see the section under Time Share Connection) and go through your system's log-on procedure.
9. For explanations on the functions and operation of the special function keys (as well as the other keyboard members), refer to the previous section entitled "Key Definitions".

Special Considerations

1. When using the Define Key operation, the user should be aware that if he has more than one EXECUTE or STORE key as part of the key's definition, and the terminal is in LOCAL mode, only the last command under the key will be processed.
2. It is possible to use the Define Key operation to make it possible to go into LOCAL mode, execute an escape sequence, and switch back to REMOTE mode under a single keystroke. Thus, if the user finds it cumbersome to press the LOCAL key, then type "Ec&p1D" and press EXECUTE to start recording information coming down from the mainframe, and then press the REMOTE key to resume communication with the computer again, this entire operation may be compacted into a single key definition. For more details, refer to the section under Special Function Keys.
3. This program will run at 300 baud with no overruns. The program will run at higher baud rates; however, if the main computer sends over a large amount of information (i.e. a program listing) information will periodically be lost, due to the fact that the data comm buffer will be filling up faster than the program can process the information being sent. If it is critical that the terminal emulator program be able to accept large volumes of information at higher baud rates, the following data comm service routine has been successfully tested at 1200 baud. This code can be inserted after line 8790 after first deleting lines 8800 through 13660. Note that no provisions have been made for stripping off the high order bit on each character, no remote escape sequences have been allowed for, and no logging of lines on mass storage has been included. Also, each line-feed is assumed to be preceeded by a carriage return. The variable Com$ will in general have to be dimensioned to some arbitrarily large number (like 1000).

```
8790 Dotacomservice: !
8800            Com$[LEN(Com$)+1]=TBUF$          ! Dump the buffer
8810 Lf1:       Lf=POS(Com$,CHR$(10))            ! Look for end-of-line
8820            IF NOT Lf THEN RETURN            ! Return to main program
8830            FOR I=1 TO Lf-2                  ! Put characters from line in the
8840                TDISP Com$[I;1]              !  entry area.
8850            NEXT I
8860            PRINT Com$[1,Lf-2]               ! Print the line.
8870                                             ! At this point the line could be
8880                                             !  logged on a mass storage device
8890                                             !  or sent to a hardcopy printer.
8900            Com$=Com$[Lf+1]                  ! Strip off used line.
8910            TDISP CHR$(12)                   ! Clear the entry line.
8920            GOTO Lf1
```

NOTE

If you are running on an HP 3000 Series II as terminal type 10, the Enq/Ack handshaking implemented in this program will allow the original program to run at 2400 baud with no overruns.

Time-Share Connection

Once the terminal emulator program is ready for use (the message **Terminal Ready on 11** will be in the display area of the CRT), it will be necessary to establish a communication linkage with the computer system. If you are using a hard-wired terminal port with the Opt. 001 cable, you will merely have to ensure that it is plugged into the cable leading from your computer. However, if you are using Opt. 001 with a modem or a data set, you should follow the sequence outlined below.

1. Turn on the modem.
2. If you are using:

   Acoustic Coupler

   a. Set the duplex switch to FULL.
   b. Dial the computer's number.
   c. When the computer answers with a high-pitched tone, place the handset in the coupler. Be sure the receiver and the transmitter on the handset are in their proper places (this should be marked on the modem). If the modem has a carrier indicator, it should light up, signifying an adequate connection.
   d. If the coupler has a line switch, set it on ON-LINE.

   or

   Data Set

   a. Press the TALK button on the data set.
   b. Dial the computer number.
   c. When the computer answers with a high-pitched tone, press the DATA button until the DATA light is on. Replace the handset.

3. Most computers require a carriage return to initiate a session. Press STORE or EXECUTE. The computer should respond with a prompt or command. At this point, follow the log-on procedure for your computer system. Be aware that if you fail to log within a certain time limit, some computers will drop the communications link, requiring you to dial the number and try again.

Once you are logged on to the system, most computers will not log you off until you give the log off command, or (maybe) if the communications link is broken. Just stopping the program will not necessarily cause you to be logged off the system. Indeed, it is possible to stop the terminal emulator program, make several modifications to it, and restart the program to resume interaction with the system, without requiring a logon procedure (assuming, of course, that you didn't hang the phone up).

Examples of log-on procedures:

1. Logging on to an HP 3000

   a. Press: STORE (or EXECUTE)

   b. When the **:** prompt appears in the input line:

      1. Type: HELLO NAME.ACCOUNT
      2. Press: STORE (or EXECUTE)

   c. If **USER PASSWORD** appears on the screen:

      1. Type: PASSWORD
      2. Press: STORE (or EXECUTE)

   d. The system will respond with a message similar to the following:
      SESSION NUMBER = #S135
      FRI, APR 14, 1978, 9:10 AM
      HP 32002A.01.01
      At this point, you are logged on to the 3000 system.

2. Logging on to a particular commercial time-share service

   a. Press: STORE (or EXECUTE)

   b. When the message..... **select desired service (cts, cts2, or tso)** appears on the screen:

      1. Type: CTS
      2. Press: STORE (or EXECUTE)

   c. The message **mainstream - cts online** will appear on the screen, and the prompt **CP>** will appear in the input line.

      1. Type: LOGON ACCOUNT PASSWORD
      2. Press: STORE (or EXECUTE)

   d. The system will respond with a message similar to the following:
      LOGON AT 12:14:23 EDT FRIDAY 04/14/78
      LINE 02F (CODE 2DB-1)
      CMS REL 3 06/10/77 V003
      R;
      At this point you are logged on.

Modifications

The terminal emulator programs have been designed in such a way as to make it possible for the user to make modifications fairly easily. There are several items which the user may want to change that are fairly simple to do. They are covered in the following order:

1. Changing the default "wake-up" parameters of the following items: 98036A interface select code, baud rate, number of stop bits, parity, bits per character, bitrate factor, the data file name (and mass storage device), data file size, hardcopy printer select code, prompt character, and remote flag character.
2. Changing the size of the recall buffer.
3. Eliminating prompt checking.
4. Changing the Handshake on/off wake up state.
5. Changing the compatibility mode on/off wake up state.
6. Changing the characters that get get sent to the computer as a result of any given keystroke.
7. Changing th echo on/off wake up state.
8. Changing the rubouts on/off wake up state.
9. Removing the Eng/Ack handshaking.
10. Changing the delay for the S/F handshake.

First, some general instructions on program editing.

a. Load the program into memory:

   1. Type: LOAD "LMGTRM"
   2. Press: EXECUTE

b. Edit the proper line.

   1. Type: EDIT LINE xxxx (where xxxx is either the number of the line you want to edit, or the label of the line)
   2. Press: EXECUTE
   3. The given line will appear in the middle of the screen. Use the editing keys (left and right arrow, insert and delete characters), and the typewriter keys to change the line to read like you want it.
   4. Press: STORE (This will cause the corrections you have made to the line to be entered into memory.

c. Repeat this step (b) for every line you wish to change in the given program. Once all the corrections for the program have been made, save the program on tape for future use.

   1. If you do not want to destroy the original program:

i. Type: STORE "filename" (where "filename" is some valid file name and mass storage device.)
ii. Press: EXECUTE
iii. In the future, to use the modified emulator, instead of the original emulator, use the filename selected above, instead of the filename "LMGTRM" when loading the emulator into memory.

2. If you do not wish to save the original program, but wish to save the modified version over the top of the original version:

   i. Type: RE-STORE "LMGTRM"
   ii. Press: EXECUTE

NOTE

The instructions STORE and RE-STORE are used to put the modified program onto a mass storage device instead of SAVE and RE-SAVE because the terminal emulator requires the use of a special binary program. The STORE and RE-STORE instructions handle the binary program as well as the BASIC language program, while the SAVE and RE-SAVE instructions will only save the BASIC language portion of the program.

Program Modifications:

CHANGING THE DEFAULT PARAMETERS OF THE EMULATOR:

Beginning at line 190, there are 3 READ statements which set, among other things, the interface select code, the baud rate, the number of stop bits, parity, the number of bits per character, bitrate factor, the data file name (and mass storage device), data file size, and hardcopy printer select code. The DATA statements which are accessed by these READ statements are located at the line labelled "Intialconds"

Line 13810

The DATA statement reads as follows:

change 2nd access on Interface to from #3 for 2400 Brate.

DATA 11,300,1,2,8,3,"off","on","DCdata:T15",30,17,1E99,0,999,21,"[illegible]","[illegible]"

2400

The first number (11) is the default 98036A interface select code. If the 9845 does not detect a 98036 card set to the default select code, it will scan the range of valid select codes until it finds a 98036A card. If there is no card plugged in, the program will issue a message to this effect.

The second number (300) is the baud rate.

NOTE

This is used merely for error checking to make sure an invalid bit rate factor isn't entered. The baud rate must still be set using the selector switch on the 98036A card.

The third number (1) is the number of stop bits desired. 1 means 1 stop bit, 2 means 1.5 stop bits, and 3 means two stop bits.

The fourth number (2) is the parity being used. 0 means odd parity (disabled), 1 means odd parity (enabled), 2 means even parity (disabled), and 3 means even parity (enabled).

The fifth number (8) is the number of bits per character. Any number from five through 8 may be used.

The sixth number (3) is the bit rate factor. 1 means 1 times the bit rate clock, 2 means 1/16 times the bit rate clock, and 3 means1/64 times the bit rate clock.

The string immediately following the "on" and "off" strings ("DCdata:T15") is the data file name and mass storage device. This may be changed to any valid file name and mass storage device. (For more information on file names and mass storage devices, refer to the Operating and Programming manual).

The number immediately following the data file name (30) is the file size. This number is used if the command to create the file is given (&p1D).

Next is a 17. This is the ASCII decimal code for the prompt character. If you want to change the character recognized by the terminal emulator as the prompt, change the 17 to the decimal ASCII equivalent of the character your computer uses for a prompt.

The next value in the DATA statement (1E99)initializes the variable Clear, which is used to set the time a message appears in the display.

The next two numbers in the DATA statement (0,999) are for the hardcopy printer's select code and HPIB address. If the printer is not an HPIB printer, the 999 is used as a bus address. If no hardcopy printer is available at all, use 16 (CRT) instead of 0 (the internal printer).

The next number in the DATA statement is set aside by the program to indicate that a key has been redefined to have both the Local key and the Remote key in its definition. The number is the ASCII decimal code of a character that has been set aside. Thus, if your particular data comm application involves sending a lot of ASCII 21's (no acknowledge), you should change this number to one whose character equivalent is not used. In general, those characters having ASCII values below 32 (space) are the best ones to use.

CHANGING THE SIZE OF THE RECALL BUFFER:

The recall buffer is set up to hold 1000 characters. This means that as you enter commands, they will be stored in the recall buffer, with the most recent commands being entered at the front of the buffer, while old commands are "pushed down" toward the end of the buffer. When 1000 characters have been entered in the buffer, the command at the rear of the buffer (i.e. the least recently entered command) will be eliminated to make room for the most recent command. If you want to expand the size of the recall buffer so that it will hold more (or less) than 1000 characters, you must change the following lines:

```
10 Dim: DIM Buffer$[164],Lastbuffer$[1000],Def$[300],Nam$[22],Prompt$[70]
380                    Rsize=1000
```

In the dimension statement, the size of the string Lastbuffer$ should be changed to whatever size you want it, and the variable Rsize in line 380 should be changed to the same value.

ELIMINATING PROMPT CHECKING:

There are two keys reserved for sending a line to the computer: EXECUTE and STORE. The only functional difference between these two keys is that when the STORE key is used, the terminal will check to see if it has received a prompt character from the computer. However, some computers do not send a prompt character at all. In this case, it will always be necessary to use the EXECUTE key to send commands to the computer, unless you eliminate the line of the program that checks for the prompt. The line in question immediately follows the line labelled "Store:". To eliminate the line entirely, you may either insert an exclamation point (!) in front of it, thus turning the line into a remark which the running program will ignore, or you may press the DEL LN key, removing the line from memory.

Changing The Handshake On/Off Wake Up State:

In the original emulator program handshake is set to "on" initially. If you want the emulator to wake up with handshake off, then the following line must be changed:

```
340          Handshake=Comp_mode=1
```

The value 1 means that Handshake is "on". In order to be "off", the variable "Handshake" must be set to 0. This can be done by changing the above line to:

```
340          Comp_mode=1
```

and adding the line:

```
345          Handshake=0
```

Changing The Compatability Mode On/Off Wake Up State:

In the original emulator program compatability mode is set to "on" initially. If you want the emulator to wake up with compatability mode off, then the following line must be changed:

```
340          Handshake=Comp_mode=1
```

The value 1 means that compatability is "on". In order to be "off", the variable "Comp_mode" must be set to 0. This can be done by changing the above line to:

```
340          Handshake=1
```

and adding the line:

```
345        Comp_mode=0
```

NOTE

If you would like both compatability mode and handshake to be off when the terminal emulator initially wakes up then just change line 340 to:

```
340        Handshake=Comp_mode=0
```

CHANGING KEY DEFINITIONS:

The Define Key operation accessed by SFK 9 (refer to the Special Funtion Keys section) may be used to redefine any key on the keyboard. However, if the user wants to define the keys to "wake up" in a certain way (other than that currently defined by the program), it is necessary to know how the program stores key definitions.

The string variable Def$ is used to store the key definitions. When the program goes through its initialization routines, it reads data defined by the four DATA statements immediately following the line labelled "Defs:" into Def$. The format of the data to be read into Def$ consists of two ASCII decimal codes (these codes tell which key is to be defined), followed by a string (which tells what the definition of the key is).

The two ASCII decimal codes which determine the key to be defined are found in the following way:

If the key to be re-defined is a standard ASCII key (having a decimal code less than 127), then the first code will be 0, and the second code will be the ASCII decimal code of the defined key (i.e. the two codes for the key A would be 0,65).

On the other hand, if the key to be re-defined is not a standard ASCII key (that is, any of the special function keys, any of the editing keys, etc.), then the first code will be 127, and the second code will be the alternative decimal code defined for that key. (To find the alternative codes returned for all non-ASCII keys, refer to the KBD$ section of the System 45B Operating and Programming Manual. For example, the two codes for the RESULT key would be 127,41, the CLEAR LINE key would be 127,43, and SFK 0 would be 127,0.

For the string following the two decimal codes, the actual definition, use normal alphanumeric characters where applicable. If you need as part of your definition characters not on the keyboard, there are two special characters set aside for altering the actual decimal code of a character. A | will cause 128 to be added to the decimal code of the character immediately following, and a ^ will cause 64 to be subtracted from the decimal code of the character immediately following. For example, suppose you wanted a line feed to be part of the definition. A line feed is ASCII 10. Since there is no key for the line feed character (other than pressing the CONTROL key and J at the same time), you could use the string ^J to signify a linefeed. J is ASCII 74. Subtracting 64 from this due to the ^ preceding the J leaves 10, which is the ASCII decimal code for a line feed.

In order to specify a non-ASCII key as part of a key definition, it is necessary to use an ASCII 127 as a delimiter, followed by the character corresponding to the alternate decimal code of the desired key. An ASCII 127 uses a combination of the | and ^ codes mentioned above, since there is no way to get a ▒ from the keyboard. The proper representation of an ASCII 127 is "^|?. The "?" is ASCII 63. The | causes 128 to be added, giving 191. The ^ causes 64 to be subtracted, resulting in the desired 127.

NOTE

A rubout (▒) is defined to be two 127's in a row (i.e.^|?^|?))

Thus, to add key definitions to the "wake up" or default conditions, it is only necessary to add to the cited DATA statements. The program will keep reading the data statements for key definitions until it hits the sequence of −1,0,"". It is important that this sequence always be the last sequence in the key definitions.

Examples

Suppose you want to define the RESULT key to perform the logon sequence for an HP 3000. The third DATA statement in the original program (see the line labelled "Defs:") reads as follows:

DATA 127,254,^M

To define the RESULT key, change the line to read:

DATA 127,254,^M,127,41,^|?^THELLO USER.ACCOUNT;TERM=10^|?^T

The 127, 41 identifies the key to be defined as the RESULT key. Note that the actual alternative decimal code of the RESULT key is 51. However, the terminal emulator program makes a special case out of the keys whose alternate decimal codes are over 49 (that is, 50, 51, 52, and 53). Whenever you want to define these four keys, always subtract ten (yielding 40, 41, 42, and 43). This affects the BACKSPACE, RESULT, STOP and CLEAR LINE keys. The ^|?^T part of the string indicates that the first key in the definition is the STORE key (the ^|? is ASCII 127, which indicates that the key is a non-ASCII key, and the ^T is ASCII 20, which indicates the STORE key, as shown in the section titled INFORMATION ON THE DATA COMM BINARY), which is used to gain the computer's attention. The HELLO USER.ACCOUNT;TERM=10 is simply the account number and terminal type, and the ^|?^T at the end is the STORE key again, which is used to send the message.

Suppose you want to define the TAB SET key to send the SHOWJOB command to an HP 3000, while retaining the definition of the RESULT key outlined above in the first example. Since the third DATA statement referred to above is now fairly cluttered, it might be wise to use a new DATA statement for the sake of clarity. Modify the fourth DATA statement to read thusly:

DATA 127,37,^|?+SHOWJOB^|?^U,−1,0,""

The 127,37 identifies the TAB SET key. The ^|?+ key means that the first keystroke of the definition is the CLEAR LINE key (^|? indicates a non-ASCII key, and + is decimal 41, which is the alternate decimal code for the CLEAR LINE key). SHOWJOB is the command for listing busy jobs and sessions on the 3000. ^|?^U corresponds to the EXECUTE key, which sends the command to the computer. The −1,0,"" at the end is the sequence which signifies the end of the key definitions.

Suppose you want to define the STEP key to open the data file (refer to the section under Escape Codes) and start recording. Modify the fourth DATA statement after the line labelled "Defs:" to read as follows:

DATA 127,16,~| ?~D~ [&p1D~| ?~U~| ?D,127,37,~| ?+SHOWJOB~ | ?~U, −1,0,""

The 127,16 identifies the STEP key. The ~| ?~D identifies SFK 4 (the LOCAL key for the terminal emulator program). ~[indicates the escape code "E" (ASCII 27), and &p1D is the sequence which tells the emulator to open the data file and start recording. ~| ?~U is the EXECUTE key, which causes the emulator program to execute the "E" command. The ~| ?D indicates shift SFK 4, which puts the emulator back in REMOTE mode. The remainder of the line is the definition for the second example, and the end of definition sequence.

If you have doubts that you have the correct definitions in your keys, remember that the Status Keys will list all key definitions in order.

CHANGING THE ECHO ON/OFF WAKE UP STATE:

In the original terminal emulator program, echo is set to "on" in the initial setup routines. If you want the emulator to wake up with echo "off", then the following lines must be changed.

400 Cursor=Notstopped=Locked=Echo=1
410 Definemode=Insertmode=Printall=Escseq=Recording=Locked=Gra_mode=0

The variable Echo tells whether the echo is "on" or "off". Echo=1 means that echo is "on", while Echo=0 means echo is "off". Thus, if you want the program to wake up with echo "off" rather than "on", remove the variable Echo from statement 400 and insert it in statement 410 to read:

400 Cursor=Notstopped=Locked=1
410 Definemode=Insertmode=Printall=Esceq=Recording=Locked=Gra_mode=Echo=0

CHANGING THE RUBOUTS ON/OFF WAKEUP STATE:

The terminal emulator will "wake up" printing any rubouts (▒) that the computer sends it. If you want the machine to ignore all rubouts without having to use special function key 12, change line 230. It originally reads:

230 Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&CHR$(13)&CHR$(7)&CHR$(19)

It should be changed to read:

**230 Select$**=CHR$(29) &CHR$(31) &CHR$(10) &CHR$(27) &CHR$(Prompt) &CHR$(13) &CHR$(7) &CHR$(19) &CHR$(127)

REMOVING THE ENQ/ACK HANDSHAKING:

The program will automatically respond with an "Ack" character (ASCII 6) whenever it receives an "Enq" character from the computer. If for some reason the user should desire to eliminate this feature, he should eliminate line 8820 and change line 10370 to: 10370 Trashit!

CHANGING THE DELAY FOR THE S/F HANDSHAKE:

There is a delay in the program to prevent the 9845 from sending an S or F handshake to the computer before it is ready to receive it. The delay is 500 ms (WAIT 500) and is located at lines 12600, 12680, 12940, 12990 . If you want to change the delay, edit the above lines and make the desired change.

Key Mode Program

The key mode terminal emulator sends each character to the computer as it is typed in. STORE and EXECUTE are used to send a carriage return character, signifying the end of the line. (One of the special function keys is defined so that STORE and EXECUTE will send a linefeed as well as a carriage return.) Local editing is suspended. The backspace key will give the "illusion" of local editing because it causes the last character in the entry line to be erased. However, the backspace key still causes a character to be sent to the computer. This character defaults to an ASCII backspace, but may be redefined using the Define Key operation.

Key Definitions

All the special function keys indicated on the overlay are defined in this program. In addition, some of the other keys on the 9845 keyboard have been redefined for special use with the Emulator program. Keys that are undefined will cause a beep, but will otherwise be ignored (such as the CONT key).

The following keys perform the functions indicated:

Typewriter (TYPWTR)	This key turns on/off typewriter mode. For an explanation of the typewriter key, refer to the 9845 Operating and Programming manual.
Backspace	Moves the cursor one position left and clears the previous character. The computer receives an ASCII 8 or whatever character the user replaces this with using the Define Key operation.
Left-arrow	Performs the same operation as the backspace key.
Up-arrow	Moves screen up one line.
Down-arrow	Moves screen down one line.
Roll up	Moves screen up five lines.
Roll down	Moves screen down five lines.
Return (STORE, EXECUTE)	Sends a carriage return to the computer. The end-of-line characters may be changed by using the [Define Keys] special function key (i.e. UDK 9).
STOP	Immediately stops program. This is alright to do, except it will not close the data comm channel or exit typewriter mode. Do not confuse the STOP key with UDK 31 (shift UDK 15).)

NOTE

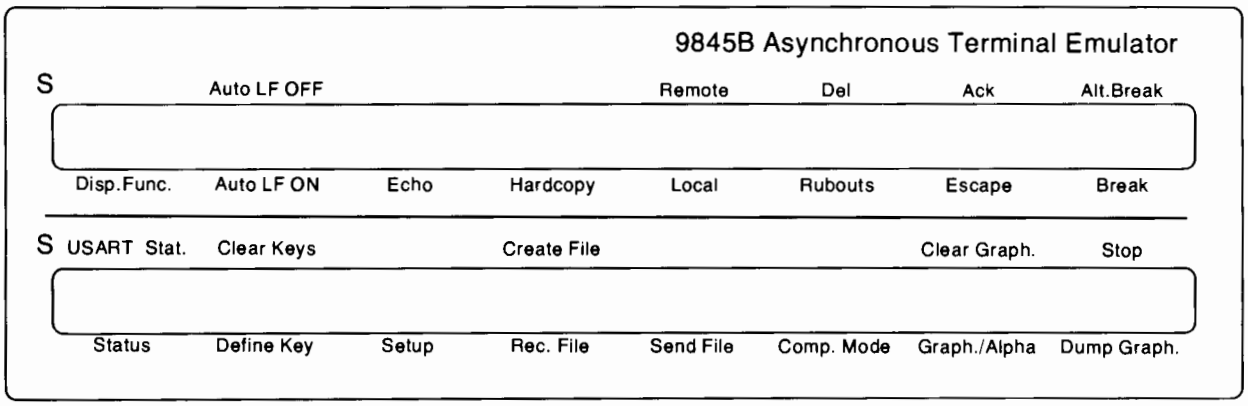
The keys Up-arrow, Down-arrow, Roll up, Roll down, Typewriter and STOP are executed by the system so they are not redefineable and cannot be used in definitions.

The following keys perform the functions specified in graphics mode:

Left-arrow	Moves cursor 14 units left.
Right arrow	Moves cursor 14 units right
Shift Left-arrow	Moves cursor 1 unit left.
Shift Right-arrow	Moves cursor 1 unit right.
Up-arrow	Moves cursor 10 units up.
Down-arrow	Moves cursor 10 units down.
Shift Up-arrow	Moves cursor 1 unit up.
Shift Down-arrow	Moves cursor 1 unit down.
Return (STORE,EXECUTE)	Sends the input buffer and the coordinates of the graphics cursor to the computer followed by the graphics input terminator. The graphics input terminator can be specified by using the Setup key (UDK 10). The STORE key will not send the line unless a DC1 prompt character has been received (indicating a prompt); EXECUTE will send it anyway. To redefine the DC1 prompt character, see the section titled "Modifications".
STOP	Same function as STOP in alpha mode.

The following keys perform the functions indicated in graphics-alpha mode:

Left-arrow	Moves cursor 1 character position (14 units) left.
Right-arrow	Moves cursor 1 character position (14 units) right.
Return (STORE,EXECUTE)	Same function as Return in alpha mode.
STOP	Same function as STOP in alpha mode.



The special function keys are defined as follows (refer to overlay):

Display functions (UDK 0) (toggle)	Turns on/off the display functions feature of the hardcopy printer. Default is display functions off.

NOTE

With the display functions feature turned on all ASCII control characters (decimal 0 through 31) are printed, rather than being executed). This feature will not work if the hardcopy printer does not have this capability.

Auto LF on (UDK 1)	Changes the end-of-line sequence to CR/LF.
Auto LF off (UDK 17 or Shift UDK 1)	Changes the end-of-line sequence to CR.
Echo (UDK 2) (toggle)	This key tells the terminal emulator that the computer does/does not "echo back" what the terminal sends it. If echo is off but the computer does "echo back", then everything you type in will appear twice. If the assumption that the computer does not have "echo back" is correct, however, then what you type will appear only once. If echo is on and the computer does not "echo back" then the commands the user sends will not appear at all. Default is echo on.
Hardcopy (UDK 3) (toggle)	Turns on/off the hardcopy printer. Default is hardcopy off.
Local (UDK 4)	Puts the terminal in local mode. Messages from the computer are ignored and messages which would be sent from the terminal are processed as if the computer had sent them to the terminal. Local mode is useful for giving escape sequence commands from the keyboard, or for ignoring large amounts of text sent from the computer. It can also be used for off-line processing of data, including graphics data.
Remote (UDK 20 or Shift UDK 4)	Resumes communication with the computer.

Rubouts (UDK 5) (toggle)	Disables/enables printing of rubouts. Default is rubouts not ignored. Rubouts may be sent by the computer as components of graphics coordinates. When rubouts are disabled they will be ignored. Instead <Esc ?> will be interpreted as ASCII 127 for graphics purposes. (Do NOT disable rubouts if the computer you are using uses DEL's in graphics commands).
Del (UDK 21 or Shift UDK 5)	Rubout or "Delete" character (ASCII 127).
Escape (UDK 6)	Escape character (may also be obtained by pressing CONTROL key and the "[" key simultaneously).
Ack (UDK 22 or Shift UDK 6)	Sends an ACK character (ASCII 6) to the computer as soon as it is pressed.
Break (UDK 7)	Sends BREAK signal to the computer as it is pressed.
Alt Break (UDK 23 or Shift UDK 7)	Sends a subsystem break (control Y) to the computer as soon as it is pressed. This break character may be changed by using the "Define Key" special function key (UDK 9).
Status (UDK 8)	Prints the status of the emulator program.

When the terminal is initially turned on this will be its status:

Status of Terminal Emulator

Echo on	Hardcopy on	Rubouts not ignored	Display funcs.off
Recording off	Local off	Cursor at 1	Background off
Prompt D_1(ASCII 17)	Keyboard lock off	Hard. printer at 0	
Autosend off	Baud rate 300	1 stop bit	Compatability on
8 bits per character		I/O card at 11	
Parity (even) disabled		Bit rate factor is 1/64	
Handshake on		Graphics input term.:	
No alternative plotter specified		[cr]	
File name: DCdata:T15		File size: 30	
Current line:			

Key Definitions

[sh UDK7]:	[em]
[sh UDK1]:	[UDK9][store] [cr] [UDK9]
[UDK6]:	[esc]
[sh UDK6]:	[ack]
[UDK1]:	[UDK9] [store] [cr] [lf] [UDK9]
[sh UDK5]:	⁘
[end-of-line]:	[cr]

USART status control word (R4E):

Data set ready	Zero (Unused)	Framing error	Overrun error	Parity error	Transmtr empty	Receiver ready	Transmtr ready
0	0	0	0	0	1	0	1

USART
(UDK 24 or
Shift UDK 8)

Prints the status of the USART word in the 98036 card (register R4e) and tests the Data Set Ready line (given by the most significant bit in the status word) to see whether or not it is active. It should be noted that the DSR line is only used with Option 001 of the 90036 card. If you are using the standard option, the leading bit of the status register corresponds to the Request to Send line.

Define Key
(UDK 9)

Enables any key except Break (UDK 7) or Define Key (UDK 9) to be redefined as any sequence of keystrokes (including other redefined keys). The definitions of STORE, EXECUTE, Ack (UDK22), Stop (UDK 31) and Alt break (UDK 23) may be changed, but the essential function of these keys will not be affected. The "define key" mode is operated using the following sequence:

1. Press: UDK 9 (Define Key)
2. Press: The key you want to redefine
3. Type: The sequence of keystrokes you want to use to replace the given key
4. Press: UDK 9 (Define Key) to end the definition.

NOTE

The program will issue the message **Recursion level too great - infinite loop likely** if there are too many keystrokes (160) as a result of pressing a re-defined key.

To clear a single key's definition without affecting other redefined keys, use the following sequence:

1. Press: UDK 10 (Define Key)
2. Press: The key you want to clear
3. Press: The same key (again)
4. Press: UDK 10 (Define Key) to end the definition.

NOTE

The keys Up-arrow, Down-arrow, Roll up, Roll down, Typewriter, and STOP are executed by the system so they are not redefinable and cannot be used in definitions.

Clear Keys (UDK 25 or Shift UDK 9)

Clears all definitions produced by using the Define Key special function key and restores all default conditions.

Setup (UDK10)

Allows the user to change the parameters of the datacomm line, as well as the hardcopy printer. This key will cause the live keyboard to be re-enabled while the user answers the questions the program will ask. When the operation is complete, the live keyboard will again be locked out.

1. When **Enter select code of I/O card:** appears in the display area:

   a. Enter: The select code of the 98036A card

   b. Press: CONT

2. When **What is the baud rate:** appears in the display area:

   a. Enter: The baud rate you are using

   b. Press: CONT

NOTE

If you wish to change baud rates, you must select the proper switch position on the 98036 card. This input is used only for error checking on the bit rate factor and will not cause the actual baud rate to be changed.

---

3. When **Bit rate factor (1,1/16, 1/64):** appears in the display area:

   a. Type: 1,1/16, or 1/64 (for baud rates below 4800, use 1/64. For 4800 and 9600 (possible only with enq/ack handshaking) use 1/16).

   b. Press: CONT

4. When **Number of bits per character:** appears in the display area:

   a. Enter: The number of bits per character your computer recognizes

   b. Press: CONT

5. When **Is parity enabled?** appears in the display area:

   a. Type: Y or N (Y implies that parity is enabled, N implies that parity is disabled-the default is N)

   b. Press: CONT

6. If **Is parity even?** appears in the display area:

   a. Type: Y or N (Y implies even parity, N implies odd parity-the default is Y)

   b. Press: CONT

7. When **Enter alternative plotter description (such as 9872A or GRAPHICS):** appears, then:

   a. Type: <name of plotting device>(if no alternative plotter is to be specified, go to step b.)

   b. Press: CONT

8. When **Enter plotter select code (and HPIB address, if applicable):** appears:

   a. If there is no HPIB then Type: <selctcode> or

      If there is an HPIB then Type: <selectcode>, <HPIB address>

   b. Press: CONT

9. When **Graph. terminator: Enter (1) for** [cr] (2) for [cr][eot] (3) for none appears in the display area:

   a. Choose the desired graphics input terminator. The graphics terminator is sent following the graphics cursor coordinates when the coordinates are requested by the host computer. (see [illegible])

      1. Type 1 (for a carriage return only-this is the default) or
      2. Type 2 (for a carriage return and end of transmission) or
      3. Type 3 (for no graphics input terminator)

   b. Press: CONT

10. When **Is handshake for autosend enabled?** appears in the display area:

    a. Type: Y or N (Y implies that when a file is sent (UDK 11) only one line at a time will be sent and the terminal will wait for a prompt before sending the next line; N implies that the entire file should be sent without response from the computer; although the computer may temporarily halt the sending of the file by outputting an ASCII character 19 - this is necessary in some cases to prevent loss of information. Transmission is continued when the computer sends a DC1 (ASCII 17). This is sometimes called Xon and Xoff.) Default is handshake on.

    b. Press: CONT

11. When **Enter printer select code(and HPIB address, if applicable)** appears in the display area:

    a. If you are using a non-HPIB printer:

        1. Enter: The printer select code
        2. Press: CONT
        3. Wait for the **Terminal Ready...** message and then proceed normally.

    or

    a. If you are using an HPIB printer (i.e.9871A or 2631A):

        1. Enter: The printer select code, followed by a comma, followed by the bus address (e.g.7,2)
        2. Press: CONT
        3. Wait for the **Terminal Ready...** message and then proceed normally.

NOTE:
The select code of the CRT is 16 and the select code of the internal printer is 0.

Record File
(UDK 11)
(toggle)

Opens data file storage and starts recording or, if in the process of recording, stops recording and closes file. This key has the effect of entering Local mode, executing the escape sequence (Esc &p1D or Esc &p0D) and entering Remote mode (if this was previously the state of the terminal). See the section titled "Escape Codes" for further details.

NOTE
Leaves the terminal in Local or Remote mode depending on state prior to pressing UDK 11). Default is file closed.

Define File
(UDK 27 or
Shift UDK 11)

Allows redefinition of the data file. The default data file is "DCdata:T15" with a size of 30 records. When this key is pressed the keyboard is re-enabled while the user answers the questions of the program. When the operation is complete, the live keyboard will be locked out.

1. When **Enter file name:** appears in the display area:

   a. Enter: The file name desired (may or may not include the mass storage device code). For example, FILE or FILE:T15 or

   b. Press: CONT (If the file already exists then that file becomes the new data file).

2. If the file does NOT exist then when **Enter size of file** appears in the display area:

   a. Enter: the number of records in the file desired.

   b. Press: CONT (The file is created and becomes the new data file)

NOTE

Data files should not cross the tape track boundary as the necessity of rewinding the tape at this point may cause loss of information.

Send File
(UDK 12)
(toggle)

Puts the terminal in an "Auto send" mode or, if the terminal IS in "Auto send" will exit that mode. This key has the effect of entering Local mode and executing the escape sequence (Esc &p1s or Esc &p3s). For more information, see the section entitled "Escape Codes". Default is not auto send. If Handshake is enabled (see Setup; UDK 10), then one line will be sent out each time a prompt is received from the computer. If Handshake has not been enabled, then the entire file will be sent without waiting for any response from the main processor unless an ASCII character 19 is received. If DC3 (ASCII 19) is received, then transmission of the file is temporarily halted until a DC1 (ASCII 17) is received. Transmission is then continued.

If the Send File key is pressed when the terminal is in LOCAL mode, then the information read from the data file is interpreted as if it had been sent from the computer. If commands from the computer have been recorded, then the terminal act as it did when it was receiving those commands.

Compatability mode (UDK 13) (toggle)	Turns on/off compatability mode. If compatability mode is on, then the graphics and graphics-alpha modes may be entered. If compatability mode is off, then the graphics raster is not available. All User Defined Keys and escape sequences involving graphics (for example, UDK 15, Dump graphics, and Esc [etb]) are disabled. The terminal will act as a normal time-share terminal and not as a graphics terminal. Default is compatability mode on.
Graphics/Alpha (UDK 13) (toggle)	If the terminal is in alpha-mode this key will cause it to enter graphics-alpha mode. If the terminal is in graphics or graphics-alpha mode this key causes it to enter alpha mode. Default is alpha mode. This key is disabled if compatability mode is off.
Clear Graphics (UDK 29 or Shift UDK 13)	This key causes the graphics raster to be cleared. This key is only enabled in graphics or graphics-alpha mode.
Dump graphics (UDK 15)	This key causes the graphics raster to be dumped (printed by) the internal thermal printer. This key is enabled in graphics or graphics-alpha mode only.
Stop (UDK 31 or Shift UDK 15)	Stops the terminal emulator program, first sending any message that happened to have been designated using the DEFINE KEY operation.

Escape Codes

There are several escape code sequences which can produce various functions when sent from the host computer (or when typed in from the terminal in local mode). Several of the escape codes deal with opening, closing, purging, or linking of data files. The data file which is assumed to be referenced by these commands stays the same from one command to the next unless the data file name is redefined (using Esc &pG). The default file name is "DCdata:T15". The default file size assumed when opening a file is 30 physical records. The file size may be redefined using the Esc &pA sequence.

Esc &p0D	Closes the data file and stops recording.

⌐ &p1D or ⌐ &p2D	Opens data file on mass storage and starts recording. The data file used will be either the default "DCdata:T15" or the most recent override of the default. The file size of the data file will either be the default (30 physical records) or the most recent override of the file size. Once this command is encountered, everything the computer sends will automatically be recorded on the mass storage file, as well as being printed on the screen and/or hardcopy printer.
⌐ &p3D	Turns off the hardcopy printer and closes the data file.
⌐ &p4D	Turns on the hardcopy printer.
⌐ &p5D	Turns off the hardcopy printer.
⌐ &pG	Sets the data file name to the value of the current line being sent from the computer (or from the keyboard if the program is in local mode) and then clears the line. The default file name is "DCdata:T15". Notice that any mass storage device may be used, provided the Mass Storage ROM is plugged in, and that particular device is interfaced to the 9845 properly. Example: In order to get all Mass Storage escape sequences to refer to the file "FILE" on a floppy disk at select code 8, the following string would be sent from the computer (or from the keyboard if the emulator is in local mode): FILE:F8⌐&pG
⌐ &pA	Sets the file size of the data file (used when opening the file using ⌐ &p1D) to the value of the current line being sent from the computer, and then clears the line. Example: To set the file size to 50 physical records, the following string should be sent from the computer (or from the keyboard if the terminal emulator is in local mode):50⌐&pA
⌐ &pN	Purges the data file.
⌐ &pE	LINKs the data file into memory after the terminal emulator program. The program which is linked in should not use arrays that aren't allocated in the terminal emulator program, because a LINK will not cause arrays (or strings) to be allocated. If arrays are needed, it will be necessary for the user to modify the terminal emulator program so that it will allocate his arrays before the data file is linked into memory.
⌐ &cE	Transfers control to the program which has been previously linked via ⌐ &pE. In order to return control to the terminal emulator when the program is done, the statement "GOTO Resume" should be used.

ESC &p1S or ESC &p2S	Puts the terminal in an "Auto send" mode. Lines from the data file (defined using ESC &pG) or UDK 27 (shift UDK 11) are sent to the computer automatically upon receiving a prompt character. (Refer to the section under "Modifications" for instructions on changing the prompt character.) This mode is exited upon three conditions: 1) End of file is reached, 2) Non-string data is encountered, or 3) An ESC&p3S is sent.
ESC &p3S	Takes the terminal out of "Auto send" mode.
ESC &d@	Turns off IV, BL and UL.
ESC &dA	Turns on blinking (BL)
ESC &dB	Turns on inverse video (IV)
ESC &dC	Turns on IV and BL
ESC &dD	Turns on underline (UL)
ESC &dE	Turns on UL and BL
ESC &dF	Turns on UL and IV
ESC &dG	Turns on UL, IV and BL
ESC Y	Turns on the display functions feature of the internal printer.
ESC Z	Turns off the display functions feature of the internal printer.

Several of the above escape sequences have the side effect of automatically putting the emulator in REMOTE mode if it was in LOCAL mode to begin with. These sequences are: ESC&cE and ESC &pE.

Graphics escape codes:

[ESC] &p1P	Turn on alternative plotter.
[ESC] &p0P	Turn off alternative plotter.
[ESC] &s1p0Q or [ESC] &s0p1Q	Turn compatability mode on.
[ESC] &s0p0Q	Turn compatability mode off.
[ESC][SB][ESC][SD]	Send graphics cursor position to the computer.
[ESC]][SB]	Send graphics cursor position preceded by one ASCII character to the computer.
[ESC][SB]	Dump graphics.
[ESC][SF]	Clear screen, enter graphics alpha mode and home cursor.

[Esc][Esc]	Read status and cursor position. Status includes the status of the hard copy unit, the current left margin and whether the mode is graphics or graphics-alpha. The information sent to the computer is: <status byte><Hi X><Lo X><Hi Y><Lo Y><graph. terminator>. Status byte: 1 0 1 1/0 0 0/1 0/1 1 Hard copy unit 0 = not ready 1 = ready Mode 1 = Graphics 1 = Graphics-alpha Margin 1 = margin 1 0 = margin 2
[Esc][GS]	Enter graphics mode.
[Esc]?	Character 127 (rubout) when plotting if rubouts disabled.
[GS]	(CONTROL/=) Enter graphics mode (same as Esc [GS]).
[US]	(CONTROL/?) Enter graphics-alpha mode.
[BS]	(CONTROL/H) Moves cursor one space left (14 units).
[HT]	(CONTROL/I) Moves the cursor one space right (14 units).
[CR]	Enter graphics-alpha mode and execute a carriage return.
[LF]	(CONTROL/J) Moves the cursor down 1 line (21 units).
[VT]	(CONTROL/K) Moves the cursor 1 line up (21 units).

Opening, closing, purging or chaining requires a certain amount of time to maneuver the tape drive, so it is necessary to implement a software handshake in order to keep the terminal emulator and the computer synchronized. The completion of an operation is signalled by either "F" (failure), or "S" (successful) followed by the end-of-line character(s). Closing (Esc &p0D) cannot be unsuccessful (i.e. an "S" will always be returned). It is up to you to make the computer you are talking to cooperate with the 9845 terminal emulator when using the above escape sequences. The software handshake outlined above makes it easy to synchronize the two using a program which runs on the main computer. If the program were in BASIC, for example, an INPUT statement could be used to check whether a tape operation was successful or unsuccessful, since the terminal emulator will automatically send an "S" or an "F" upon completion of the commands given by the escape sequences.

The example program below shows how the escape sequences might be used from a computer which has a BASIC language compiler or interpreter.

```
10   REM FIRST RESERVE A STRING TO HOLD THE ESCAPE CHARACTER
20   E$=CHR$(27)
30   REM NEXT REDEFINE THE FILE NAME AND FILE SIZE
40   REM NEW FILE NAME IS "DUMMY:T15"
50   REM NEW FILE SIZE IS 1 PHYSICAL RECORD
60   PRINT "1";E$;"&pA"
70   PRINT "DUMMY:T15";E$;"&pG"
80   REM NOW OPEN THE DATA FILE AND START RECORDING
90   PRINT E$;"&p1D"
100  INPUT A$
110  IF A$="S" THEN 140
120  PRINT "PROGRAM FAILURE"
130  STOP
140  REM PRINT SOME NUMBERS ON THE TAPE
150  PRINT 1,2,3,4,5
160  REM NOW CLOSE THE DATA FILE AND STOP RECORDING ON IN IT
170  PRINT E$;"&p0D"
180  INPUT A$
190  IF A$="F" THEN 120
200  PRINT "PROGRAM SUCCESSFUL"
210  STOP
220  END
```

User Instructions

1. a. Make sure that the I/O ROM's are properly installed.
   b. Make sure the 98036A card is plugged into one of the I/O slots in the back of the machine.
   c. The select code of the 98036 card should be set to some value between 1 and 12, since select codes 0, 13, 14, 15, and 16 are reserved.
   d. Make sure the power is turned on.
   e. Insert the Terminal Emulator cartridge into the primary tape transport (i.e. the drive above the special function keys).
2. Load the line mode program into memory:
   a. Type: LOAD "KMGTRM"
   b. Press: EXECUTE
3. When the program is loaded (the busy light will go off and the tape will stop moving):
   a. Press: RUN

4. The program will begin initializing. The message **Please wait** will appear in the display area of the CRT. After a few seconds, the message **Terminal ready on 11** should appear in the display area if your system is correctly configured (11 will actually be replaced by whatever select code the 98036 card is set at). If the **Terminal Ready...** message appears, go to step 8. If the **Terminal Ready...** message does not appear, then an error has been detected. The error should result in a message outlined by one of the following three steps (5, 6, or 7).

5. If the message **What is the select code?** appears in the display area, the program has detected the presence of more than one 98036 card. It is only possible to open one data communication channel at a time.

   a. Enter: The select code of the card you wish to use.

   b. Press: CONT

   c. Follow the instructions in step 4.

6. If the message **There are no 98036 cards present. Please insert one.** appears in the display area, make sure the card is plugged securely into one of the I/O slots in the back of the machine. Once the card is plugged in correctly, the program will detect its presence and you may proceed normally as outlined in step 4.

7. If the message **Hardcopy printer not operational** appears in the display area, the internal printer is either 1) missing or 2) out of paper. If the printer is out of paper, put a new roll of paper in the printer before proceeding. On the other hand, if it happens to be the case that your machine does not have an internal printer, you will be asked to enter a different select code.

   a. When **Enter printer select code (and HPIB address, if applicable)** appears in the display area:

      1. Enter: The printer select code

      2. Press: CONT

NOTE 1

If you are using an HPIB printer, it will be necessary to enter two numbers in part 7.a.1 above. The first number will be the select code, followed by a comma, followed by the HPIB address (example: 7,1).

NOTE 2

The select code of the internal printer is 0. The select code of the CRT (if you do not desire to use a hardcopy printer at all) is 16.

---

   b. Once the new printer select code has been entered, the error checking routines outlined in step 4 will be repeated.

8. Once the message **Terminal Ready on 11** appears in the display area, the terminal emulator is ready to use. At this point, you will probably have to establish your communications linkage (see the section under Time Share Connection) and go through your system's log-on procedure.

9. For explanations on the functions and operation of the special function keys (as well as the other keyboard members), refer to the previous section entitled "Key Definitions".

Special Considerations

1. When using the Define Key operation, the user should be aware that if he has more than one EXECUTE or STORE key as part of the key's definition, and the terminal is in LOCAL mode, only the last command under the key will be processed.
2. It is possible to use the Define Key operation to make it possible to go into LOCAL mode, execute an escape sequence, and switch back to REMOTE mode under a single keystroke. Thus, if the user finds it cumbersome to press the LOCAL key, then type ESC&p1D and press EXECUTE to start recording information coming down from the mainframe, and then press the REMOTE key to resume communication with the computer again, this entire operation may be compacted into a single key definition. For more details, refer to the section under Special Funtion Keys.
3. Care should be taken to set the Echo On/Off mode correctly. If the computer doesn't echo, and the emulator is set to Echo On, nothing will appear in the entry line while characters are being typed. On the other hand, if the computer does echo, but the terminal is set to Echo Off, two characters will appear in the entry line for every character that is entered.
4. The backspace key is originally defined to send an ASCII 8 to the computer every time it is pressed, while erasing the last character in the display line. If your computer doesn't understand ASCII 8 to be the backspace character, the Define Key operation may be used to send a different character to the computer, even though the backspace key will still cause the last character in the entry line to be arased.
5. This program will run at 300 baud with no overruns. The program will run at higher baud rates; however, if the main computer sends over a large amount of information (i.e. a program listing) information will periodically be lost, due to the fact that the data comm buffer will be filling up faster than the program can process the information being sent. If it is critical that the terminal emulator program be able to accept large volumes of information at higher baud rates, the following data comm service routine has been successfully tested at 1200 baud. This code can be inserted after line 8510 after first deleting lines 8520 through 13390. Note that no provisions have been made for stripping off the high order bit on each character, no remote escape sequences have been allowed for, and no logging of lines on mass storage has been included. Also, each line-feed is assumed to be preceeded by a carriage return. The variable Com$ will in general have to be dimensioned to some arbitrarily large number (like 1000).

```
8510 Datacomservice: !
8520            Com$[LEN(Com$)+1]=TBUF$          ! Dump the buffer
8530 Lf1:       Lf=POS(Com$,CHR$(10))            ! Look for end-of-line
8540            IF NOT Lf THEN RETURN            ! Return to main program
8550            FOR I=1 TO Lf-2                  ! Put characters from line in the
8560                TDISP Com$[I;1]              !  entry area.
8570            NEXT I
8580            PRINT Com$[1,Lf-2]               ! Print the line.
8590                                             ! At this point the line could be
8600                                             !  logged on a mass storage device
8610                                             !  or sent to a hardcopy printer.
8620            Com$=Com$[Lf+1]                  ! Strip off used line.
8630            TDISP CHR$(12)                   ! Clear the entry line.
8640            GOTO Lf1
```

NOTE

If you are running on an HP 3000 Series II as terminal type 10, the Enq/Ack handshaking implemented in this program will allow the original program to run at 2400 baud with no overruns.

Time-Share Connection

Once the terminal emulator program is ready for use (the message **Terminal Ready on 11** will be in the display area of the CRT), it will be necessary to establish a communication linkage with the computer system. If you are using a hard-wired terminal port with the Opt. 001 cable, you will merely have to ensure that it is plugged into the cable leading from your computer. However, if you are using Opt. 001 with a modem or a data set, you should follow the sequence outlined below.

1. Turn on the modem.
2. If you are using:

   Acoustic Coupler

   a. Set the duplex switch to FULL.
   b. Dial the computer's numbers.
   c. When the computer answers with a high-pitched tone, place the handset in the coupler. Be sure the receiver and the transmitter on the hand-set are in their proper places (this should be marked on the modem). If the modem has a carrier indicator, it should light up, signifying an adequate connection.
   d. If the coupler has a line switch, set it to ON-LINE.

   or

   Data Set

   a. Press the TALK button on the data set.
   b. Dial the computer number.
   c. When the computer answers with a high-pitched tone, press the DATA button until the DATA light is on. Replace the handset.

3. Most computers require a carriage return to initiate a session. Press STORE or EXECUTE. The computer should respond with a prompt or command. At this point, follow the log-on procedure for your computer system. Be aware that if you fail to log on within a certain time limit, some computers will drop the communications link, requiring to dial the number and try again.

Once you are logged on to the system, most computers will not log you off until you give the log off command, (or if the communications link is broken). Just stopping the program will not necessarily cause you to be logged off the system. Indeed, it is possible to stop the terminal emulator program, make several modifications to it, and restart the program to resume interaction with the system, without requiring a log-on procedure (assuming, of course, that you didn't hang the phone up).

Examples of log-on procedures:

1. Logging on to an HP 3000

   a. Press: STORE (or EXECUTE)

   b. When the **:** prompt appears in the input line:

      1. Type: HELLO NAME.ACCOUNT
      2. Press: STORE (or EXECUTE)

   c. If **USER PASSWORD** appears on the screen:

      1. Type: PASSWORD
      2. Press: STORE (or EXECUTE)

   d. The system will respond with a message similar to the following:
      SESSION NUMBER = #S135
      FRI< APR 14, 1978, 9:10 AM
      HP 3200A.01.01
      At this point, you are logged on to the 3000 system.

2. Logging on to a particular commerical time-share service

   a. Press: STORE (or EXECUTE)

   b. When the message **..... select desired service (cts, cts2, or tso)** appears on the screen:

      1. Type: CTS
      2. Press: STORE (or EXECUTE)

   c. The message **mainstream - cts online** will appear on the screen, and the prompt **CP>** will appear in the input line.

      1. Type: LOGON ACCOUNT PASSWORD
      2. Press: STORE (or EXECUTE)

   d. The system will respond with a message similar to the following:
      LOGON AT 12:14:23 EDT FRIDAY 04/14/78
      LINE 02F (CODE 2DB-1)
      CMS REL 3 06/10/77 V003
      R;
      At this point you are logged on.

Modifications

The terminal emulator programs have been designed in such a way as to make it possible for the user to make modifications fairly easily. There are several items which the user may want to change that are fairly simple to do. They are covered in the following order:

1. Changing the default "wake-up" parameters of the following items: 98036A interface select code, baud rate, number of stop bits, parity, bits per character, bitrate factor, the data file name (and mass storage device), data file size, hardcopy printer select code, prompt character, and remote flag character.
2. Changing the Handshake on/off wakeup state.
3. Changing the Compatibility mode on/off wakeup state.
4. Changing the characters that get sent to the computer as a result of any given keystroke.
5. Changing the echo on/off wake up state.
6. Changing the rubouts on/off wake up state.
7. Removing the Enq/Ack handshaking.
8. Changing the S/F delay.

First, some general instructions on program editing.

a. Load the program into memory:

   1. Type: LOAD "LMGTRM"
   2. Press: EXECUTE

b. Edit the proper line.

   1. Type: EDIT LINE xxxx (where xxxx is either the number of the line you want to edit, or the label of the line)
   2. Press: EXECUTE
   3. The given line will appear in the middle of the screen. Use the editing keys (left and right arrow, insert and delete characters), and the typewriter keys to change the line to read like you want it.
   4. Press: STORE (This will cause the corrections you have made to the line to be entered into memory.

c. Repeat this step (b) for every line you wish to change in the given program. Once all the corrections for the program have been made, save the program on tape for future use.

   1. If you do not want to destroy the original program:

      i. Type: STORE "filename" (where "filename" is some valid file name and mass storage device.)

ii. Press: EXECUTE
iii. In the future, to use the modified emulator, instead of the original emulator, use the filename selected above, instead of the filename "LMGTRM" when loading the emulator into memory.

2. If you do not wish to save the original program, but wish to save the modified version over the top of the original version:

   i. Type: RE-STORE "LMGTRM"
   ii. Press: EXECUTE

NOTE

The instructions STORE and RE-STORE are used to put the modified program onto a mass storage device instead of SAVE and RE-SAVE because the terminal emulator requires the use of a special binary program. The STORE and RE-STORE instructions handle the binary program as well as the BASIC language program, while the SAVE and RE-SAVE instructions will only save the BASIC language portion of the program.

Program Modifications:

CHANGING THE DEFAULT PARAMETERS OF THE EMULATOR:

Beginning at line 200, there are 3 READ statements which set, among other things, the interface select code, the baud rate, the number of stop bits, parity, the number of bits per character, bitrate factor, the data file name (and mass storage device), data file size, and hardcopy printer select code. The DATA statements which are accessed by these READ statements are located at the line labelled "Intialconds"

The DATA statement reads as follows:

DATA 11,300,1,2,8,3,"off","on","DCdata:T15",30,17,1E99,0,999,21,"",""

The first number (11) is the default 98036A interface select code. If the 9845 does not detect a 98036 card set to the default select code, it will scan the range of valid select codes until it finds a 98036A card. If there is no card plugged in, the program will issue a message to this effect.

The second number (300) is the baud rate.

NOTE

This is used merely for error checking to make sure an invalid bit rate factor isn't entered. The baud rate must still be set using the selector switch on the 98036A card.

The third number (1) is the number of stop bits desired. 1 means 1 stop bit, 2 means 1.5 stop bits, and 3 means two stop bits.

The fourth number (2) is the parity being used. 0 means odd parity (disabled), 1 means odd parity (enabled), 2 means even parity (disabled), and 3 means even parity (enabled).

The fifth number (8) is the number of bits per character. Any number from five through 8 may be used.

The sixth number (3) is the bit rate factor. 1 means 1 times the bit rate clock, 2 means 1/16 times the bit rate clock, and 3 means1/64 times the bit rate clock.

The string immediately following the "on" and "off" strings ("DCdata:T15") is the data file name and mass storage device. This may be changed to any valid file name and mass storage device. (For more information on file names and mass storage devices, refer to the Operating and Programming manual).

The number immediately following the data file name (30) is the file size. This number is used if the command to create the file is given (Ec &p1D).

Next is a 17. This is the ASCII decimal code for the prompt character. If you want to change the character recognized by the terminal emulator as the prompt, change the 17 to the decimal ASCII equivalent of the character your computer uses for a prompt.

The next value in the DATA statement (1E99)initializes the variable Clear, which is used to set the time a message appears in the display.

The next two numbers in the DATA statement (0,999) are for the hardcopy printer's select code and HPIB address. If the printer is not an HPIB printer, the 999 is used as a bus address. If no hardcopy printer is available at all, use 16 (CRT) instead of 0 (the internal printer).

The next number is the ASCII decimal code for the prompt character. If you want to change the character recognized by the terminal emulator as the prompt, change the 17 to the decimal ASCII equivalent of the character your computer uses for a prompt. The prompt character is only used to trigger the "auto send" feature (refer to Ec &p1S under the Escape Code section).

The next number in the DATA statement is set aside by the program to indicate that a key has been redefined to have both the Local key and the Remote key in its definition. The number is the ASCII decimal code of a character that has been set aside. Thus, if your particular data comm application involves sending a lot of ASCII 21's (no acknowledge), you should change this number to one whose character equivalent is not used. In general, those characters having ASCII values below 32 (space) are the best ones to use.

Changing The Handshake On/Off Wake Up State:

In the original emulator program handshake is set to "on" initially. If you want the emulator to wake up with handshake off, then the following line must be changed:

```
360        Handshake=Comp_mode=1
```

The value 1 means that Handshake is "on". In order to be "off", the variable "Handshake" must be set to 0. This can be done by changing the above line to:

```
360        Comp_mode=1
```

and adding the line:

```
365        Handshake=0
```

Changing The Compatability Mode On/Off Wake Up State:

In the original emulator program compatability mode is set to "on" initially. If you want the emulator to wake up with compatability mode off, then the following line must be changed:

```
360        Handshake=Comp_mode=1
```

The value 1 means that compatability is "on". In order to be "off", the variable "Comp_mode" must be set to 0. This can be done by changing the above line to:

```
360        Handshake=1
```

and adding the line:

```
365        Comp_mode=0
```

NOTE

If you would like both compatability mode and handshake to be off when the terminal emulator initially wakes up then just change line 340 to:

```
360        Handshake=Comp_mode=0
```

CHANGING KEY DEFINITIONS:

The Define Key operation accessed by SFK 9 (refer to the Special Function Keys section) may be used to redefine any key on the keyboard. However, if the user wants to define the keys to "wake up" in a certain way (other than that currently defined by the program), it is necessary to know how the program stores key definitions.

The string variable Def$ is used to store the key definitions. When the program goes through its initialization routines, it reads data defined by the four DATA statements immediately following the line labelled "Defs:" into Def$. The format of the data to be read into Def$ consists of two ASCII decimal codes (these codes tell which key is to be defined), followed by a string (which tells what the definition of the key is).

The two ASCII decimal codes which determine the key to be defined are found in the following way:

If the key to be re-defined is a standard ASCII key (decimal code less than 127), then the first code will be 0, and the second code will be the ASCII decimal code of the defined key (i.e. the two codes for the key A would be 0,65).

On the other hand, if the key to be re-defined is not a standard ASCII key (i.e. any of the special function keys, any of the editing keys, etc.), then the first code will be 127, and the second code will be the alternative decimal code defined for that key. (To find the alternative codes returned for all non-ASCII keys, refer to the KBD$ section of the System 45B Operating and Programming Manual.) For example, the two codes for the RESULT key would be 127,41, the CLEAR LINE key would be 127,43, and SFK 0 would be 127,0. You will notice that the table of alternate decimal codes in the section referred to above states that the alternate decimal codes for the RESULT and CLEAR LINE keys are actually 51 and 53, respectively. However, the program treats those codes between 50 and 53 as a special case and subtracts 10 from them before making use of them in the program. This affects the BACKSPACE, RESULT, STOP, and CLEAR LINE keys.

For the string following the two decimal codes, the actual definition, use normal alphanumeric characters where applicable. If you need as part of your definition characters not on the keyboard, there are two special characters set aside for altering the actual decimal code of a character. A "|" will cause 128 to be added to the decimal code of the character immediately following, and a ^ will cause 64 to be subtracted from the decimal code of the character immediately following. For example, suppose you wanted a line feed to be part of the definition. A line feed is ASCII 10. Since there is no key for the line feed character (other than pressing the CONTROL key and J at the same time), you could use the string ^J to signify a line feed. J is ASCII 74. Subtracting 64 from this due to the ^ preceding the J leaves 10, which is the ASCII decimal code for a line feed.

In order to specify a non-ASCII key as part of a key definition, it is necessary to use an ASCII 127 as a delimiter, followed by the character corresponding to the alternate decimal code of the desired key. An ASCII 127 uses a combination of the | and ^ codes mentioned above, since there is no way to get a ▒ from the keyboard. The proper representation of an ASCII 127 is ^|?. The "?" is ASCII 63. The | causes 128 to be added, giving 191. The ^ causes 64 to be subtracted, resulting in the desired 127.

NOTE

A rubout (▒) is defined to be two 127's in a row (i.e. ^|?^|?))

Thus, to add key definitions to the "wake up" or default conditions, it is only necessary to add to the cited DATA statements. The program will keep reading the data statements for key definitions until it hits the sequence of −1,0,"". It is important that this sequence always be the last sequence in the key definitions.

Examples

Suppose you want to define the RESULT key to perform the logon sequence for an HP 3000. The third DATA statement in the original program (see the line labelled "Defs:") reads as follows:

DATA 127,254,~M

To define the RESULT key, change the line to read:

DATA 127,254,~M,127,41,~| ?~THELLO USER.ACCOUNT;TERM=10~| ?~T

The 127,41 identifies the key to be defined as the RESULT key. The ~| ?~T part of the string indicates that the first key in the definition is the STORE key (the ~| ? is ASCII 127, which indicates that the key is a non-ASCII key, and the ~T is ASCII 20, which indicates the STORE key, which is used to gain the computer's attention. The HELLO USER.ACCOUNT;TERM=10 is simply the account number and terminal type, and the ~| ?~T at the end is the STORE key again, which is used to send the message.

Suppose you want to define the TAB SET key to send the SHOWJOB command to an HP 3000, while retaining the definition of the RESULT key outlined above in the first example. Since the third DATA statement referred to above is now fairly cluttered, it might be wise to use a new DATA statement for the sake of clarity. Modify the fourth DATA statement to read thusly:

DATA 127,37,~| ?+SHOWJOB~| ?~U,−1,0,""

The 127,37 identifies the TAB SET key. The ~| ?+ key means that the first keystroke of the definition is the CLEAR LINE key (~| ? indicates a non-ASCII key, and + is decimal 41, which is the alternate decimal code for the CLEAR LINE key). SHOWJOB is the command for listing busy jobs and sessions on the 3000. ~| ~U corresponds to the EXECUTE key, which sends the command to the computer. The −1,0,"" at the end is the sequence which signifies the end of the key definitions.

Suppose you want to define the STEP key to open the data file (refer to the section under Escape Codes) and start recording. Modify the fourth DATA statement after the line labelled "Defs:" to read as follows:

DATA 127,16,~| ?~D~[&p1D~| ?~U~| ?D,127,37,~| ?+SHOWJOB~| U,−1,0,""

The 127,16 identifies the STEP key. The ~| ?~D identifies SFK 4 (the LOCAL key for the terminal emulator program). ~[indicates the escape code "▒" (ASCII 27), and &p1D is the sequence which tells the emulator to open the data file and start recording. ~| ?~U is the EXECUTE key, which causes the emulator program to execute the "▒" command. The ~| ?D indicates shift SFK 4, which puts the emulator back in REMOTE mode. The remainder of the line is the definition for the second example, and the end of definition sequence.

Remember that the Status Key will list key re-definitions in order if you have doubts that you are defining the keys correctly.

CHANGING THE ECHO ON/OFF WAKE UP STATE:

In the original terminal emulator program, echo is set to "on" in the initial setup routines. If you want the emulator to wake up with echo "off", then the following lines must be changed.

420 Cursor=Notstopped=Locked=Echo=1
430 Definemode=Insertmode=Printall=Escseq=Recording=Locked=Gra_mode**=0**

The variable Echo tells whether the echo is "on" or "off". Echo=1 means that echo is "on", while Echo=0 means echo is "off". Thus, if you want the program to wake up with echo "off" rather than "on", remove the variable Echo from statement 730 and insert it in statement 740 to read:

420 Cursor=Notstopped=Locked=1
430 Definemode=Insertmode=Printall=Escseq=Recording=Locked=Gra_mode==Echo=0

CHANGING THE RUBOUTS ON/OFF WAKE UP STATE:

In the original program, all rubouts ▒ sent from the computer are printed. If you would like to eliminate this without having to access SFK 5, then line 240 should be changed. It originally reads as follows:

240 Select$= CHR$(29)& CHR$(31)& CHR$(10)&CHR$(27)& CHR$(Prompt)& CHR$(13)& CHR$(7)&CHR$(19)& CHR$(127)

It should be changed to read:

240 Select$= CHR#(29)& CHR$(31)&CHR$(10)& CHR$(27)& CHR$(Prompt)&CHR$(13)& CHR$(7)&CHR$(19)& CHR$(127)

REMOVING THE ENQ/ACK HANDSHAKING:

The program will automatically respond with an "Ack" character (ASCII 6) whenever it receives an "Enq" character from the computer. If, for some reason you should desired to eliminate this feature, you should eliminate line 8540 and change line 10100 to: 10100 Trashit!

CHANGING THE DELAY FOR THE S/F HANDSHAKE:

There is a delay in the program to prevent the 9845 from sending an S or F handshake to the computer before it is ready to receive it. The delay is 500 ms (WAIT 500) and is located at lines 12330, 12410, 12670, 12720. If you want to change the delay, edit the above lines and make the desired change.

Minimal Line Mode Program

The minimal line mode terminal emulator consists of a set of modular subprograms which allow the user to send lines to a computer and read lines from a computer. No extraordinary "whistles and bells" are included in this program. It is a skeletal, but well-structured, minimal program which may be studied as an example for users who need to develop a more specialized program.

The name of each subprogram in this package is given below, along with a brief description of its use. For more complete details, please refer to the annotations in the program listings.

Dcom_setup(Err)

This subprogram sets up the 98036 card on the select code passed in through the COM statement (refer to the listing). If the Err parameter is set to 1 upon exiting the subprogram, then the communications link was not established. To change the parameters of bit rate factor, parity, stop bits, and bits per character, refer to the 98036A manual concerning the R4C mode word.

Get_line(Line$)

This routine gets a line from the keyboard into Line$. The line can then be sent to the computer at will via an OUTPUT statement.

Kbd_isr

This subprogram is the keyboard interrupt service routine. It is responsible for acting upon keystrokes entered by the user. The routine **Get—line(Line$)** acts as a front-end to this routine.

NOTE

This routine defines the PAUSE key to send a BREAK to the computer.)

FNKbd_ready

This function returns a 1 if CONT, STORE, or EXECUTE was hit (indicating a SEND command). It is used by the **Get_line** subprogram.

Get_dcom(B$,Prompt)

This subprogram gets a line from the data comm channel. An end-of-line is signalled either by a carriage return (ASCII 13) or a DC1 (ASCII 17). The latter indicates that there was a prompt from the computer. The variable "Prompt" indicates which was the case.

Dcom_isr

This subprogram is the data comm interrupt service routine. It monitors the channel constantly, processing characters as they come in. **Get_dcom(B$,Prompt)** acts as a front-end for this routine.

FNDc_ready

This function returns a 1 if the data comm buffer contains a carriage return or a DC1. It is used by the **Get_dcom** subprogram.

User Instructions

1. a. Make sure the I/O ROM's are properly installed.
   b. Make sure the 98036A card is plugged into one of the I/O slots in the back of the machine.
   c. The select code of the 98036A card should be set to some value between 1 and 12, since select codes 0, 13, 14, 15, and 16 are reserved.
   d. Make sure the power is turned on.
   e. Insert the Terminal Emulator cartridge into the primary tape transport (i.e., the drive above the special function keys).
2. Load the minimal line mode program into memory:
   a. Type: LOAD "LTMIN"
   b. Press: EXECUTE
3. When the program is loaded (the busy light will go off, and the tape will stop moving):
   a. Press: RUN
4. When the message **RUNNING** appears in the display area of the CRT, the program is ready to use. At this point, you will probably want to log on to the computer, following the correct procedure for your particular system.

   If the **RUNNING** message does not appear and program execution halts, chances are that the select code of the 98036A card does not agree with the program. Change the fourth line of the program (Sel-code=11) so that the select code is set properly, and RUN the program again.

Special Considerations:

1. The PAUSE key is defined to be the BREAK key.
2. If the message **Datacomm overrun** appears in the display area, information has been lost because the internal buffer Dcom$ overflowed. There are two solutions to this problem: 1)Run at a lower baud rate, or 2) Increase the size of Dcom$. The second solution involves changing the COM statement at the beginning of the program, as well as changing line 2670 in the routine Dcom_isr.
3. This program will run at 300 baud with no overruns.

Modifications:

This section deals with program modifications. This includes such topics as changing the set-up parameters (stop bits, parity, bits per character, and bitrate factor), and adding key definitions to the keyboard interrupt service routine.

CHANGING THE SET-UP PARAMETERS

At line 3071 in the subprogram **Dcom_setup**, the parameters used for setting up the 98036 card are read from the DATA statement at line 3072. To change any or all of these parameters, refer to the 98036A manual for the proper codes of the parameters you wish to use, and change the DATA statement accordingly.

ADDING KEY DEFINITIONS

In the subprogram **Kbd_isr** starting at the line labelled **Sfk** is the section of the program which deals with non-ASCII keystrokes. Each each key can return four different codes, depending upon whether or not the control and/or shift keys were used in conjunction with the key itself. Any key which was used alone will return a code less than or equal to 63 (decimal). If the shift key was used, then the code will fall between 64 and 127. If the control key was used, the code will fall between 128 and 191. If both the control and shift keys were used, the code will fall between 192 and 255. The line which reads:

Shift=K DIV 64

determines which of these combinations was used. Shift=0 implies no shift, 1 implies shift, 2 implies control, and 3 implies control-shift. The variable K, which holds the actual code, is changed accordingly. Next, there is a series of ON... GOTO statements. The labels in the ON...GOTO statements are abbreviations of each key's title (Lf is left, Ri is right, Tbs is Tab Set, Step is step, Stop is stop, and so on). To enable any given key, all that is necessary is to insert a routine having the proper label as its entry point. Any unused labels in the ON... GOTO statements are ignored because of the ON ERROR statement at line 1870. If you want a shifted key to perform a different function than a non-shifted key, this can be easily built into the execution routine for that key because the Shift variable is already set.

Line Mode Program Listing

The terminal emulator program enables interrupt conditions at line 640 and 670 which cause branches to be taken whenever a key is pressed, or whenever any characters come in on the data comm channel. The interrupt on the Keyboard is set to a higher priority, so that interrupt will take precedence over the data comm service routine.

The main program does nothing but sit in an idle loop testing several flags (such as Local (for local mode), Reinit(for going through the setup routine), Sendflag(for the autosend feature), Gojump(for background program linkage), and Notstopped(for termination of the program)). If a key is pressed, a GOSUB is performed to the line labelled Keyservice. A RETURN encountered in the Keyservice routine will cause the program to resume where it left off. Similarily, characters coming in on the data comm channel will cause a GOSUB to Datacomservice. Again, because of the priority scheme, the Datacomservice routine can be interrupted by the data comm channel, but the Keyservice routine can not be interruted by the keyboard.

When a key is pressed causing a program branch to Keyservice, the keyboard buffer area KBD$ is immediately dumped, freeing it for further collection of key codes while the first batch are being processed. The string Key$ is then tested immediately to see if the break key (UDK 7) has been pressed, then to see if the alternate break key (Shift UDK 7) has been pressed, and finally to see if the Ack key (Shift UDK 6) has been pressed. Each of these three keys will be acted upon immediately, regardless of the order in which they were pressed relative to other members of the buffer.

Once all occurrences of the three keys listed above have been handled, the variable Key$ is processed one character at a time, starting at the line labelled More1. The ASCII character 127 is treated as a special case, since it is used as a delimiter for non-ASCII key codes inserted as a result of the key redefinition feature (255 delimits non-ASCII keys inserted from the keyboard). If the character itself is desired, two 127's will be in Key$ adjacent to each other. If neither a 127 nor a 255 is found, the ASCII key is tested to see if it has been redefined. If it has been redefined, the redefinition string is added to the front of Key$ and the loop is performed again. If the key wasn't redefined, then it is added to the end of the string Buffer$, which keeps a line until the STORE or EXECUTE key is pressed, at which time the line is sent to the computer and cleared. After the character is added to Buffer$, the next character the Key$ is processed, and so on until Key$ is empty. Then, a RETURN to the idle loop is executed.

On the other hand, if the character encountered in Key$ is a 127 or 255, indicating a non-ASCII key, the next character is picked up to tell which non-ASCII key was pressed. That key is then tested for redefinition. If the key has been redefined, the redefinition string is added to the front of Key$, and the loop is performed again. If the key was not redefined, then the program will branch (via a series of ON...GOTO statements) to the execution routine for that particular key. Upon completion of the key's execution routine, the next code in Key$ is processed as before until Key$ is empty.

Whenever information comes into the data comm channel, a GOSUB is performed to the line labelled Datacomservice. If the terminal emulator is in local mode, all information sent from the computer will be ignored, and the program will RETURN to wherever it branched from.

If the terminal is not in local mode, then the buffer TBUF$ is immediately dumped into the variable Com$. Com$ is first checked for the occurrence of an Enq character (ASCII 5), to which the program will respond with an Ack (ASCII 6).

Next, each character in Com$ is processed, one character at a time. First, the leading bit is stripped off of each character to allow for parity. Next, the incoming characters are tested against a string called Select$, which contains a unit specifier, a group separator, a line feed, an escape, a DC1, a carriage return, a bell, a DC3 and a ▒. If the character being tested corresponds to one of these characters, a branch is taken to a routine which is set up to handle the special character which was encountered. If none of the above characters was encountered, then a branch is taken to the line labelled Nctrl (no control). This routine first checks to see if an escape sequence is active. If an escape sequence is active, then the character is added to the sequence being decoded. If not, then the character is appended to the string Disp$, which keeps track of incoming characters.

Disp$ is printed and cleared whenever 1) a line feed comes in, or 2) when Disp$ reaches a length of 80 characters. The Print routine prints the string to the CRT always, as well as to the hardcopy printer and/or the mass storage file if either or both of these modes is currently active.

Carriage returns are ignored, while a DC1 (or whatever prompt character the user picks -- see the section entitled Modifications) causes a branch to the line labelled Cret. This routine will then set aside the current contents of Disp$ to be a prompt string which will subsequently appear in the keyboard entry line. An escape character will cause a flag to be set (Escseq) which indicates that subsequent characters should be routed to the escape sequence routine Eseq.

The Unit Specifier (ASCII 31) causes the terminal to enter graphics-alpha mode while the Group Specifier (ASCII 29) causes it to enter graphics mode.

The bell causes the terminal to beep.

The DC3 is used to indicate that Auto-send should be temporarily halted. A DC1 will cause Auto-send to be reactivated.

The ▒ is not necessarily always a member of Select$. It is included or not included as a result of pressing UDK 5. If it is included in Select$, then all rubouts will be ignored. If it is not included, then all rubouts will be treated as any other character.

When the terminal emulator is in local mode, lines typed by the user are not sent to the computer, but instead are sent to the data comm service routine, which processes them as if they had come from the computer. Thus, the user can send escape sequences locally to perform such tasks as turning on the auto send mode, or redefining the data file name, and so on.

LMGTRM

```
10 Dim:     DIM Buffer$[164],Lastbuffer$[1000],Def$[300],Nam$[22],
      Prompt$[70]
20          DIM Savebuffer$[160],Disp$[80],Key$[82],Com$[324],Name$[
               674],High$[16]
30          DIM Datafile$[10],Eol$[20],State$(0:1)[3],Select$[10],
               Parm$[4]
40          DIM Saveprompt$[70],Comment$[70],Tell$[60],Prev$[164],
               Send$[160]
50          DIM Eswitch$[5],W$[4],Comstr$[180]
60          INTEGER Bitsperchar,Parity,Key,Oldascii,Statascii,K1,K2,K3,
               K4,B2
70          INTEGER B3,Ascii,Selectcode,Baudrate,Rmchr,I,Char,Gra_al
80          INTEGER Retflag,Def_found,Rsize,Autosend,Sendflag,
               Character


Initialization routines

100 Begin:    Rub$=CHR$(127)
110           PLOTTER IS "GRAPHICS"
120           Parm$="dpcs"                  ! Used for parameterized
                 escapes
130           Eswitch$="EGNAQ"              ! Escape terminators
140           High$="@BACDFEGHJIKLNMO"      ! Highlighting codes
150           OVERLAP
160           Tell$="Please wait"
170           GOSUB Tell
180           RESTORE Initialconds
190           READ Selectcode,Baudrate,Stopbits,Parity,Bitsperchar,
                 Bitratefactor
200           READ State$(*),Datafile$,Filesize,Prompt,Clear
210           READ Hardprinter,Hpib,Rmchr,Enq$,Ack$
220           GOSUB Set_hard
230           Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
                 CHR$(13)&CHR$(7)&CHR$(19)
240           !          lf      escape          D1        cr      enq.
                 remote
250 Sfks:     READ B1                       ! Set up Name$ string
260           Name$=Name$&CHR$(B1)
270           IF B1=255 THEN Sfks
280           READ Name$[LEN(Name$)+1]
290           IF B1>-1 THEN Sfks
300           GOSUB Defsclear               ! Set up default key defs
310           PRINTER IS 16
320           PRINT LIN(20)
330           Buffer$=Lastbuffer$=Disp$=""    ! Initialize variables
340           Handshake=Comp_mode=1
350           Gt$=CHR$(13)
360           Marg=1
370           Marg2=512
380           Rsize=1000
390           Y=767
```

LMGTRM

```
400             Cursor=Notstopped=Locked=Echo=1
410             Definemode=Insertmode=Printall=Escseq=Recording=Locked=
                   Gra_mode=0
420             Gojump=Trace=Firstline=Recallptr=Local=Setup=Retflag=
                   Autosend=Not_draw=X=0
430             Character=Prompt

440 Initial:    IF NOT IOSTATUS(Selectcode) THEN Notoper
450             STATUS Selectcode;B1
460             IF BINAND(B1,48)<>16 THEN Notoper     ! --- 01- ---
470             WRITE IO Selectcode,5;1               ! 000 000 001 -> R5
480             WRITE IO Selectcode,4;64              ! 001 000 000 -> R4D
490             WRITE IO Selectcode,4;Stopbits*64+Parity*16+(Bitsperchar-
                   5)*4+Bitratefactor
500             WRITE IO Selectcode,4;39              ! 000 100 111 -> R4D
510             WRITE IO Selectcode,5;0               ! 000 000 000 -> R5
520             READ IO Selectcode,4;B1               ! cock i/f for
                   interrupts
530             WRITE IO Selectcode,7;0
540             WRITE IO Selectcode,5;132             ! 010 000 100
                   Input inter on
550             STATUS Selectcode;B1                  ! See if input
                   inter on
560             IF NOT BINAND(B1,128) THEN Notoper    ! Make sure it's on
570             IF Hardselect=16 THEN Topen
580             GOSUB Prtcheck
590             IF Printerokay THEN Topen
600             BEEP
610             DISP " Hardcopy printer not operational "
620             WAIT 500
630             GOTO Sp

640 Topen:      TOPEN Selectcode,2 GOSUB Datacomservice
650             Tell$="Terminal ready on "&VAL$(Selectcode)
660             GOSUB Telll
670             ON KBD 10 GOSUB Keyservice
680             TDISP CHR$(12)&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-Cursor+
                   1)
690             Local=0
```

Background loop

```
710 Wait:       IF Gojump THEN Leap                 ! Background loop
720             OVERLAP
730             IF Setup THEN Reinit
740             IF Define THEN Fil_def
750             IF Sendflag THEN GOSUB Send
760             IF Local THEN Loop
770             Clear=Clear-1
780             IF Clear<0 THEN DISP
790             IF Clear<0 THEN Clear=1E99
```

```
800            IF Notstopped THEN Wait
810            TYPEWRITER OFF
820            FOR B1=1 TO 10
830              WAIT 100
840            NEXT B1
850            TCLOSE
860            OFF KBD
870            Tell$="Terminal off"
880            GOSUB Tell
890            STOP

900 Resume:    Tell$="Background program surrendered"
910            GOSUB Tell
920            Gojump=Local=Locked=0
930            ON KBD 10 GOSUB Keyservice
940            GOTO Wait

950 Leap:      Tell$="Background program running"
960            GOSUB Tell
970            GOTO End
```

Local mode loop

```
990 Loop:      IF Keyrec OR Keysend OR Keycomp THEN Pre
1000           Disp$=""
1010 Loop1:    IF NOT Local OR Autosend THEN Wait
1020           IF Old THEN Loop1
1030 Pre:      K1=LEN(Prev$)
1040           IF NOT K1 THEN Endloop
1050           IF NUM(Prev$[K1])<>Rmchr THEN Com$=Prev$&CHR$(13)&CHR$(
                  10)
1060           IF NUM(Prev$[K1])=Rmchr THEN Com$=Prev$[1;K1-1]&CHR$(13)&
                  CHR$(10)&CHR$(Rmchr)
1070           Prev$=""
1080           GOSUB Com0
1090           Character=Prompt
1100           IF Autosend THEN GOSUB Send
1110 Endloop:Old=1
1120           Localflag=0
1130           GOTO Loop

1140 Reinit:   OFF KBD
1150           Local=1
1160 Sc:       INPUT "Enter select code of I/O card:     [Use CONT]",
         Selectcode
1170           IF (Selectcode<1) OR (Selectcode>15) THEN Sc
1180 Baud:     INPUT "What is the baud rate?             [Use CONT]",
         Baudrate
1190           RESTORE Baudrates
1200           FOR B1=1 TO 10
1210             READ B2
1220             IF B2=Baudrate THEN Brf
```

LMGTRM

```
1230          NEXT B1
1240          GOTO Baud
1250 Brf:     LINPUT "Bit rate factor (1, 1/16, 1/64): [Use CONT]",
        Savebuffer$
1260          IF Savebuffer$="" THEN Savebuffer$="1/64"
1270          Bitratefactor=0
1280          IF Savebuffer$="1/16" THEN Bitratefactor=2
1290          IF Savebuffer$="1" THEN Bitratefactor=1
1300          IF Savebuffer$="1/64" THEN Bitratefactor=3
1310          IF NOT Bitratefactor THEN Brf
1320          IF (Bitratefactor<>3) OR (Baudrate<4800) THEN Bpc
1330          BEEP
1340          DISP "Warning: 1/64 bit rate factor not recommended
                 at this baud rate!"
1350          WAIT 2000
1360 Bpc:     INPUT "Number of bits per character:     [Use CONT]",
        Bitsperchar
1370          IF (Bitsperchar<5) OR (Bitsperchar>8) THEN Bpc
1380          Parity=0
1390 Par2:    LINPUT "Is parity enabled?                [Use CONT]",
        Savebuffer$
1400          IF Savebuffer$="" THEN Savebuffer$="N"
1410          Savebuffer$=UPC$(Savebuffer$[1,1])
1420          IF Savebuffer$="Y" THEN Parity=Parity+1
1430          IF (Savebuffer$<>"N") AND (Savebuffer$<>"Y") THEN Par2
1440          IF Savebuffer$="N" THEN Sb
1450 Par:     LINPUT "Is parity even?                   [Use CONT]",
        Savebuffer$
1460          IF Savebuffer$="" THEN Savebuffer$="Y"
1470          Savebuffer$=UPC$(Savebuffer$[1,1])
1480          IF Savebuffer$="Y" THEN Parity=Parity+2
1490          IF (Savebuffer$<>"N") AND (Savebuffer$<>"Y") THEN Par
1500 Sb:      B1=1
1510          INPUT "Enter number stop bits (1,1.5,2): [Use CONT]",B1
1520          IF (B1<>1) AND (B1<>1.5) AND (B1<>2) THEN Sb
1530          Stopbits=B1*2-1
1540 Alt_plot: Plottercode=13
1550          INPUT "Enter alternate plotter discription (such as
                 9872A or GRAPHICS):  [Use CONT]",Plotter$
1560          IF (TRIM$(Plotter$)="GRAPHICS") OR (TRIM$(Plotter$)="")
                 THEN GOTO 1660
1570          LINPUT "Enter plotter select code (and HPIB address, if
                 applicable): [Use CONT]",Savebuffer$
1580          IF Savebuffer$="" THEN Savebuffer$="13"
1590          Phibflag=POS(Savebuffer$,",")
1600          IF NOT Phibflag THEN Nophb
1610          Plottercode=VAL(Savebuffer$[1,Phibflag-1])
1620          Phpib=VAL(Savebuffer$[Phibflag+1])
1630          GOTO 1660
1640 Nophb:   Plottercode=VAL(Savebuffer$)
1650          Phpib=999
1660          Gt=1
```

```
1670 Graterm:INPUT "Graph. terminator: Enter 1 for [cr] 2 for [cr]
        [eot] 3 for none    [CONT]",Gt
1680          IF (Gt=1) OR (Gt=2) OR (Gt=3) THEN Gtset
1690          BEEP
1700          GOTO Graterm
1710 Gtset: IF Gt=1 THEN Gt$=CHR$(13)
1720         IF Gt=2 THEN Gt$=CHR$(13)&CHR$(4)
1730         IF Gt=3 THEN Gt$=""
1740 Hand:Savebuffer$=""
1750      INPUT "Is handshake for autosend enabled?                [
             Use CONT]",Savebuffer$
1760      IF Savebuffer$="" THEN Handon
1770      IF UPC$(Savebuffer$[1,1])="Y" THEN Handon
1780      IF UPC$(Savebuffer$[1,1])="N" THEN Handoff
1790      BEEP
1800      GOTO Hand
1810 Handon:Handshake=1
1820         GOTO Sp
1830 Handoff:Handshake=0
1840          Xon=1
1850 Sp:      LINPUT "Enter printer select code (and HPIB adrress, if
        applicable)",Savebuffer$
1860          IF Savebuffer$="" THEN Savebuffer$="0"
1870          Hpibflag=POS(Savebuffer$,",")
1880          IF NOT Hpibflag THEN Nohpib
1890          Hardprinter=VAL(Savebuffer$[1,Hpibflag-1])
1900          Hpib=VAL(Savebuffer$[Hpibflag+1])
1910          GOTO Reset
1920 Nohpib:  Hardprinter=VAL(Savebuffer$)
1930          Hpib=999
1940 Reset:   Setup=0
1950          GOSUB Set_hard
1960          IF Hardselect=16 THEN Initial
1970          GOSUB Prtcheck
1980          IF Printerokay THEN Initial
1990          GOTO Sp
2000 Prtcheck:Printerokay=1
2010          IF Hpibflag THEN Chhpib
2020          IF IOSTATUS(Hardselect) THEN RETURN
2030          GOTO Buggyprinter
2040 Chhpib:  SET TIMEOUT Hardprinter;2000
2050          ON INT #Hardprinter GOTO Buggyprinter1
2060          STATUS Hardprinter;A
2070          IF NOT A THEN Buggyprinter1
2090          GOSUB Shutoff
2100          RETURN
2110 Shutoff:OFF INT #Hardprinter
2120          SET TIMEOUT Hardprinter;0
2130          RETURN
2140 Buggyprinter1: GOSUB Shutoff
2150 Buggyprinter: BEEP
2160          PRINT "PRINTER NOT OPERATIONAL ON ";Hardselect;""
```

LMGTRM

```
2170          Printerokay=0
2180          RETURN

2190 Notoper:Selectcode=0
2200          FOR B2=1 TO 12
2210            W1=PI
2220            STATUS B2;B3,W1
2230            IF W1<>PI THEN Notop1
2240            IF BINAND(48,B3)<>16 THEN Notop1
2250            IF Selectcode THEN Notop2
2260            Selectcode=B2
2270 Notop1: NEXT B2
2280          IF Selectcode THEN Initial
2290          BEEP
2300          DISP "There are no 98036 cards present.  Please insert
                one."
2310          WAIT 2000
2320          GOTO Notoper
2330 Notop2: BEEP
2340          OFF KBD
2350          INPUT "What is the select code?        [Use CONT]",
                Selectcode
2360          PRINT PAGE
2370          GOTO Initial

2380 Tell:    Clear=1000
2390 Tell1:   DISP TAB(74-LEN(Tell$));Rub$&" "&Tell$&" "&Rub$
2400          RETURN

2410 Send:    IF Autosend THEN K1=TYP(2)
2420          IF (K1=2) OR (K1>=8) AND (K1<=10) THEN Sendok
2430          BEEP
2440          Autosend=Sendflag=0
2450          ASSIGN #2 TO *
2460          Tell$="Autosend off -- "
2470          IF K1=3 THEN Tell$=Tell$&"End of file"
2480          IF K1<>3 THEN Tell$=Tell$&"Non-string data found"
2490          GOTO Tell
2500 Sendok: IF Autosend THEN READ #2;Send$    ! Send a line from the
        data file
2510          IF NOT Local THEN 2540
2520             Com$=Send$
2530             GOTO Com0
2540          Sendflag=0
2550          IF Local THEN Send1
2560          SET TIMEOUT Selectcode;2000
2570          SERIAL
2580          ON ERROR GOTO Send2
2590          EOL Selectcode;Eol$
2600          IF Autosend THEN OUTPUT Selectcode USING "#,K,L";Send$
2601          IF Echo THEN 2610
2602          Disp$=Disp$&Send$
```

```
2603          GOSUB Print
2610          OFF ERROR
2620 Send1:   Old=0
2630          IF Autosend AND NOT Handshake AND Xon AND NOT Keysend OR
                 Local THEN Send
2640          RETURN
2650 Send2:   Tell$="Card not operational"
2660          OFF ERROR
2670          BEEP
2680          GOSUB Tell
2690          RETURN
```

Keyboard interrupt service routine

```
2710 Keyservice:
2720          Key$=KBD$                ! Dump the keyboard buffer
2730 Init:    Recursionlevel=0

2740 Break:   IF NOT POS(Key$,CHR$(255)&CHR$(7)) THEN Albrk
2750          WRITE IO Selectcode,5;1                ! Break
2760          WRITE IO Selectcode,4;46
2770          WRITE IO Selectcode,5;132
2780          BEEP
2790          WAIT 100
2800          BEEP
2810          WAIT 100
2820          WRITE IO Selectcode,5;1
2830          WRITE IO Selectcode,4;39
2840          WRITE IO Selectcode,5;132
2850          BEEP
2860          K1=POS(Key$,CHR$(255)&CHR$(7))
2870          Key$[K1]=Key$[K1+2]              ! Strip off key codes and
                 loop back
2880          GOTO Break

2890 Albrk:   IF NOT POS(Key$,CHR$(255)&CHR$(7+64)) OR Definemode THEN
        Ack
2900          BEEP                             ! Alternate break
2910          Key=64+7
2920          GOSUB Checkdef
2930          IF Def_found THEN OUTPUT Selectcode;Def$[K2,K1];
2940          IF NOT Def_found THEN OUTPUT Selectcode;CHR$(31);
2950          WAIT 150
2960          BEEP
2970          K1=POS(Key$,CHR$(255)&CHR$(7+64))
2980          Key$[K1]=Key$[K1+2]              ! Strip off character and
                 loop back.
2990          GOTO Albrk

3000 Ack:     IF NOT POS(Key$,CHR$(255)&CHR$(70)) OR Definemode THEN
```

LMGTRM

```
        Recur
3010          Key=70
3020          GOSUB Checkdef
3030          IF Def_found THEN OUTPUT Selectcode;Def$[K2,K1];
3040          IF NOT Def_found THEN OUTPUT Selectcode;CHR$(6);
3050          BEEP
3060          WAIT 100
3070          K1=POS(Key$,CHR$(255)&CHR$(70))
3080          Key$[K1]=Key$[K1+2]          ! Strip off character and
                 loop back.
3090          BEEP
3100          GOTO Ack

3110 Checkdef: Def_found=0                 ! This routine checks to see
        if
3120          Retflag=1                    !  Alt Break or Ack have been
3130          GOSUB User2                  !  redefined.
3140          Retflag=0                    ! If they have, K2 and K1
                 mark the
3150          IF K2<>3 THEN Def_found=1    !  beginning and end in Def$
                 of the
3160          RETURN                       !  new defintion.


3170 Recur:   Recursionlevel=Recursionlevel+1
3180          IF Recursionlevel>160 THEN Abort

3190 More:    IF LEN(Key$) THEN More1      ! Check for null string.
3200          Localflag=0
3210          RETURN
3220 More1:   Ascii=1
3230          Key=NUM(Key$)                ! Strip off first character.
3240          IF Key<>127 THEN 3300        ! Allow for non-ascii key
                 codes
3250          Key$=Key$[2]                 !  inserted from user re-
                 definitions
3260          Key=NUM(Key$)                !  (which are flagged with
                 127 since
3270                                       !  255 is a delimiter in key
                                              defs.)
3280          IF Key<>127 THEN 3340        ! Rubouts are 127 127.
3290          GOTO Ascii                   ! 127 never produced in
                 definemode
3300          IF Key<255 THEN 3360         ! Check for non-ascii key code
3310          Key$=Key$[2]
3320          Key=NUM(Key$)                ! If non-ASCII, the next code
                 tells
3330                                       !  which key was pressed.
3331          IF (Key=1) OR (Key=65) THEN Lfkey=1
3340          Ascii=0
3350          IF (Key>=50) AND (Key<=53) OR (Key>=114) AND (Key<=117)
                 OR (Key>=178) AND (Key<=181) OR (Key>=242) AND (Key<=
```

```
                245) THEN Key=Key-10
3360            IF Definemode THEN Dmode
3370            IF Ascii THEN Asciiudk       ! Branch to 8-bit character
                section

3380 Doublechar: ! This section handles all keys returning two
        character codes
3390            ! (i.e. all non-ascii keys).
3400            IF NOT Insertmode THEN Ktest
3410            Insertmode=0                 ! Exit insert mode if in it
3420            Tell$="Insert mode off"
3430            GOSUB Tell
3440            ! Next, check if Insert Character button was toggled.
3450            IF (Key=32) OR (Key=32+64) THEN Strip

3460 Ktest:  IF (Key=20) OR (Key=21) OR (Key=84) OR (Key=85) THEN
        Store
3470                                         ! 20 is STORE, and 21 is
                                               EXECUTE
3480            IF Key=15+64 THEN Stop
3490            IF (Key=9) AND NOT Gra_mode THEN Tset    ! UDK 9  is
                Define Key
3500            ! Check to see if key has been user-defined (keys tested
                above this
3510            !  point can not be redefined).
3520            IF POS(Def$,CHR$(255)&CHR$(127)&CHR$(Key)) AND NOT
                Gra_mode THEN User2

3530    ! The rest of the program-defined keys (which can be
           overridden by
3540    ! the Define Key operation) are:
3550    ! UDK 0 is trace on/off toggle
3560    ! UDK 2 is echo on/off toggle
3570    ! UDK 3 is hardcopy on/off toggle
3580    ! UDK 4 is Local                                  Shift UDK 4 is
           Remote
3590    ! UDK 5 is rubouts ignored/not ignored toggle   Shift UDK 5 is
           DEL
3600    ! UDK 6 is escape character     Shift UDK 6 is Acknowledge
3610    ! UDK 7 is Break                Shift UDK 7 is alternate break
3620    ! UDK 8 is status               Shift UDK 8 is USART status
3630    !                               Shift UDK 9 is clear key
           definitions
3640    ! UDK 10 is setup
3650    ! UDK 11 is record file on/off Shift UDK 11 is define file
3660    ! UDK 12 is send file
3670    ! UDK 13 is compatability mode on/off toggle
3680    ! UDK 14 is graphics/alpha toggle
3690    !                               Shift UDK 14 is clear graphics
3700    ! UDK 15 is dump graphics
3710    ! Left arrow moves cursor left Shift left arrow moves prompt
           left
```

LMGTRM

```
3720    ! Rightarrow moves cursor right Shift rightarrow moves prompt
          right
3730    ! Home and Shift Home put cursor at left of input line
3740    ! Clear and Shift Clear will clear the screen and input line
3750    ! Clrem and Shift Clrem will clear the input line to the right
          of
3760    !  the cursor
3770    ! Delete Char and Shift Delete Char will delete a character
3780    ! Insert Char and Shift Insert Char toggle the insert mode
3790    ! Recall will bring back old commands (up to 1000 characters)
3800    ! Shift Recall works in the reverse of Recall
3810    ! Tab and Shift Tab move the cursor in increments of 8 spaces
3820    ! Backspace shifts the cursor left and clears to the end of the
3830    !  line
3840    ! Clear Line will clear the input line
3850          ON ERROR GOSUB Nada
3860 Test0: ON Key+1 GOTO Trace0,Bad,Echo0,Hard0,Local,Rubs0,Bad,Bad,
        Status,Bad,Setup,Rec_file,Send_file,Comp_mode,Ex_gra,Dump_gra
3870 Test1:  ON Key-63 GOTO Bad,Bad,Bad,Bad,Remote,Bad
3880 Test2:  ON Key-71 GOTO Dsr,Kclr,Bad,Deffil,Bad,Bad,Cgraph
3890 Test3:  ON Key-21 GOTO Left,Right,Gup,Gdown,Bad,Bad,Home,Clear,
        Clrem,Delet,Imode,Bad,Bad,Recal,Tab,Bad,Bad,Bad,Back,Bad,Bad,
        Cline
3900 Test4:  ON Key-85 GOTO Shleft,Shright,Sgup,Sgdown,Bad,Bad,Home,
        Clear,Clrem,Delet,Imode,Bad,Bad,Unrecal,Tab,Bad,Bad,Bad,Back,
        Bad,Bad,Cline


3910          OFF ERROR
3911          IF Key=19 THEN GOTO Noprmpt
3920          GOTO Bad


3930 Comp_mode: IF Gra_mode THEN Bad        ! turn compatability mode
        on
3940             IF Comp_mode THEN Compoff
3950            Prev$=CHR$(27)&"&s1p0Q"
3960            Local=Keycomp=1
3970            Escseq=Old=0
3990            RETURN
4000 Compoff: Prev$=CHR$(27)&"&s0p0Q"        ! turn compat. mode off
4010           Local=Keycomp=1
4020           Old=0
4030           RETURN
4040 Ex_gra: IF Gra_mode=0 THEN Graphic      ! graphics on/off
4050          Gra_mode=0
4060          ON KBD 10 GOSUB Keyservice
4070          EXIT GRAPHICS
4080          Tell$="Exit Graphics"
4090          GOSUB Tell
4100          GOTO Next
```

```
4110 Graphic:IF NOT Comp_mode THEN Bad
4120         IF NOT Altplot THEN 4160
4130            IF Phpib=999 THEN PLOTTER IS Plottercode,Plotter$
4140            IF Phpib<>999 THEN PLOTTER IS Plottercode,Phpib,
                   Plotter$
4150            GOTO 4170
4160         GRAPHICS                          ! graphics on
4170         Gra_mode=1
4180         IF NOT Gra_al THEN ON KBD 10 GOSUB Keyservice ,ALL
4190         Tell$="Graphics"
4200         GOSUB Tell
4210         GOTO Next
4220 Cgraph: IF NOT Comp_mode THEN Bad
4230         GCLEAR                            ! Clear graphics
4240         Gra_al=Marg=1
4250         X=0
4260         Y=767
4270         GOTO Next

4280 Dump_gra:  IF NOT Gra_mode THEN Bad
4290         OUTPUT 0;CHR$(27)&"Z"&CHR$(8)
4300         DUMP GRAPHICS                    ! Dump Graphics
4310         IF Trace THEN OUTPUT 0;CHR$(27)&"Y"
4320         GOTO Next


4330 Nada:   RETURN
4340 Tab:    IF Cursor>159 THEN Bad
4350         Cursor=8*(Cursor DIV 8)+8
4360         Buffer$[1,Cursor-1]=Buffer$
4370         GOTO Next
4380 Shleft: IF Gra_mode THEN S1_left
4390         IF LEN(Prompt$) THEN Prompt$=Prompt$[2]
4400         GOTO Next
4410 Shright:IF Gra_mode THEN S1_right
4420         Prompt$=" "&Prompt$[1,69]
4430         IF LEN(Prompt$)+LEN(Buffer$)<160 THEN Next
4440         Buffer$[LEN(Buffer$)]=""
4450         Cursor=MIN(Cursor,LEN(Buffer$)+1)
4460         GOTO Next

4470 Imode:  Insertmode=1
4480         Tell$="Insert mode on"
4490         GOSUB Tell
4500         Clear=1E99
4510         GOTO Strip

4520 Store:  IF Locked THEN Holng
4530         IF Gra_mode THEN POINTER -1,-1,2       ! Delete cursor
4540         IF (Character<>Prompt) AND (Key=20) AND NOT Local THEN
                Noprmpt
4550         Prompt$=""
```

LMGTRM

```
4560          IF NOT LEN(Buffer$) THEN Store1
4570 Store3:  IF LEN(Buffer$)+LEN(Lastbuffer$)<Rsize-2 THEN Store2
4580          FOR K1=MIN(LEN(Lastbuffer$),Rsize-2-LEN(Buffer$)) TO 1
                 STEP -1
4590          IF Lastbuffer$[K1,K1]=CHR$(255) THEN Store4
4600          NEXT K1
4610 Store4:  Lastbuffer$[K1]=""
4620          GOTO Store3
4630 Store2:  Lastbuffer$=Buffer$&CHR$(255)&Lastbuffer$

4640     ! The recall buffer is kept with
4650     !  the most recent commands at the
4660     !  front, and the least recent ones
4670     !  at the end. 255 is the delimiter.

4680          Recallptr=Unrecallptr=0
4690 Store1:  IF NOT Echo AND NOT Local AND NOT Recording THEN Disp$=
         Disp$&Buffer$[1,80-LEN(Disp$)]
         !***WW
4700          IF Recording THEN GOSUB Record
4730          Old=0
4731          IF NOT Gra_al THEN 4740
4733          Y=Y-21
4734          IF Y<0 THEN Y=767                          !***WW
4735          IF Y=767 THEN Marg=3-Marg                  !***WW
4736          X=(Marg-1)*Marg2
4740          IF Local THEN Prev$=Buffer$
4750          IF Local THEN Cline
4760          SET TIMEOUT Selectcode;2000     ! Timeout condition if
                 card hangs.
4770          SERIAL
4780          ON ERROR GOTO Store5
4790          EOL Selectcode;Eol$
4800          IF NOT Esc_sub THEN Outp_n     ! Jump if normal output
4810          IF Buffer$<>"" THEN Outp_co
4820          POINTER X,Y,1
4830          RETURN
4840 Outp_co:OUTPUT Selectcode USING "#,K";Buffer$&W$&Gt$; !Gin-mode,
         send coord
4850          POINTER -1,-1,2
4860          Esc_sub=0                                  !***WW
4870          GOTO Outp_s
4880 Outp_n:  OUTPUT Selectcode USING "#,K,L";Buffer$
4890 Outp_s:  OFF ERROR
4900          GOTO Cline
4910 Store5:  Tell$="Card not operational"
4920          OFF ERROR
4930          GOTO Telbeep
4940 Fmt:     IMAGE #,L
4950 Fmt1:    IMAGE #,A
4960 Holng:   Tell$="Locked out"
4970          GOTO Telbeep
4980 Noprmpt:Tell$="No prompt - use EXECUTE"
4990 Telbeep:BEEP
```

```
5000          GOTO Nextell

5010 Stop:                                  ! STOP
5020          IF Locked THEN Holng
5030          Notstopped=Local=0             !  to the mainframe
                 before turning
5040          K1=POS(Def$,CHR$(255)&CHR$(127)&CHR$(64+15))+3 ! off the
                 program
5050          IF K1=3 THEN RETURN            !  and closing the
                 datacomm channel.
5060          K2=POS(Def$[K1],CHR$(255))+K1-2
5070          IF K2<K1-1 THEN K2=LEN(Def$)
5080          Key$=Def$[K1,K2]
5090          GOTO More

5100 User1:   ! Process keys that have been redefined.
5110          K2=POS(Def$,CHR$(255)&CHR$(Key))+2 ! Find start of
                 definition
5120          K1=POS(Def$[K2],CHR$(255))+K2-2    ! Find end of
                 definition
5130          IF K1<K2-1 THEN K1=LEN(Def$)       ! Allow for last def.
                 in string
5140          IF LEN(Key$)+K1-K2>80 THEN Abort   ! Check for string
                 too long
5150          Key$=Def$[K2,K1]&Key$[2]           ! Append definition
5160          GOTO Recur

5170 User2:   K2=POS(Def$,CHR$(255)&CHR$(127)&CHR$(Key))+3
5180          K1=POS(Def$[K2],CHR$(255))+K2-2
5190          IF K1<K2-1 THEN K1=LEN(Def$)
5200          IF Retflag THEN RETURN
5210          IF LEN(Key$)+K1-K2>80 THEN Abort
5220          Key$=Def$[K2,K1]&Key$[2]
5230          GOTO Recur

5240 Abort:   Tell$="Recursion level too great - infinite loop likely"
5250          Key$=""
5260 Nextell:GOSUB Tell
5270          GOTO Next

5280 Back:    Cursor=MAX(1,Cursor-1)            ! Backspace
5290 Clrem:   Buffer$[Cursor]=""                ! Clear to end
5300          GOTO Next
5310 Clear:   PRINT PAGE,LIN(-19)
5320          DISP
5330 Cline:   Buffer$=""                        ! Clear line
5340 Home:    Cursor=1
5350          GOTO Next
5360 Fa_left:X=X-14                              ! Fast left
5370 Left1:   IF X<0 THEN X=0
5380          GOTO Esc_sub                       ! Goto set cursor
5390 Sl_left:IF Gra_al THEN Bad                  ! Slow left
```

LMGTRM

```
5400          X=X-1
5410          GOTO Left1
5420 Left:    IF Cursor>1 THEN TDISP CHR$(8)
5430          Cursor=MAX(1,Cursor-1)          ! Left-arrow
5440          IF Gra_mode THEN Fa_left
5450          GOTO Strip
5460 Fa_right:X=X+14                          ! Fast right
5470 Right1:  IF X>1023 THEN X=1023
5480          GOTO Esc_sub                    ! Goto set cursor
5490 Sl_right:IF Gra_al THEN Bad              ! Slow right
5500           X=X+1
5510          GOTO Right1
5520 Right:   IF Gra_mode THEN Fa_right
5530          Cursor=MIN(Cursor,LEN(Buffer$))+1
5540          GOTO Next

5550 Sgup:    IF NOT Gra_mode OR Gra_al THEN Bad     ! Slow up
5560          Y=Y+1
5570          GOTO Gu1
5580 Gup:     IF NOT Gra_mode OR Gra_al THEN Bad     ! Fast up
5590          Y=Y+10
5600 Gu1:     IF Y>780 THEN Y=780
5610          GOTO Esc_sub                      ! Goto set cursor
5620 Gdown:   IF NOT Gra_mode OR Gra_al THEN Bad ! Fast down
5630          Y=Y-10
5640 Gd1:     IF Y<1 THEN Y=1
5650          GOTO Esc_sub                      ! Goto set cursor
5660 Sgdown:  IF NOT Gra_mode OR Gra_al THEN Bad  ! Slow down
5670          Y=Y-1
5680          GOTO Gd1
5690 Delet:   Buffer$[Cursor]=Buffer$[MIN(LEN(Buffer$),Cursor)+1]
5700          GOTO Next

5710 Recal:   K1=Recallptr+POS(Lastbuffer$[Recallptr+1],CHR$(255))
5720          IF K1=Recallptr THEN Next
5730          Buffer$=Lastbuffer$[Recallptr+1,K1-1]
5740          Unrecallptr=Recallptr
5750          Recallptr=K1
5760 Recal1:  Cursor=LEN(Buffer$)+1
5770          GOTO Next

5780 Unrecal:IF NOT Unrecallptr THEN Next    ! Reverse recall
5790          K1=Unrecallptr-POS(REV$(Lastbuffer$[1,Unrecallptr-1]),
                 CHR$(255))
5800          IF K1=Unrecallptr THEN K1=0
5810          Buffer$=Lastbuffer$[K1+1,Unrecallptr-1]
5820          Recallptr=Unrecallptr
5830          Unrecallptr=K1
5840          GOTO Recal1

5850 Tset:    IF Gra_mode THEN Bad
5860          Savecursor=Cursor               ! Define key
```

```
5870          IF LEN(Def$)>297 THEN Derr
5880          Saveprompt$=Prompt$
5890          Savebuffer$=Buffer$
5900          Definemode=Cursor=1
5910          Buffer$=Prompt$=""
5920          Tell$="Enter character to define"
5930          GOSUB Tell
5940          Clear=1E99
5950          GOTO Next

5960 Kclr:    IF Gra_mode THEN Bad
5970          GOSUB Defsclear                  ! Clear keys
5980          Tell$="Definitions cleared"
5990          GOSUB Tell
6000          GOTO Next

6010 Hard0:   IF Printall=0 THEN Hard1         ! Hardcopy on/off
6020          Printall=0                       ! Hardcopy off
6030          GOTO Hard2
6040 Hard1:   Printall=1
6050          Trace=0
6060 Hard2:   Tell$="Hardcopy "&State$(Printall)
6070          GOTO Nextell

6080 Echo0:   IF Echo=0 THEN Echo1             ! Echo on/off
6090          Echo=0                           ! Echo     off
6100          GOTO Echo2
6110 Echo1:   Echo=1
6120 Echo2:   Tell$="Echo "&State$(Echo)
6130          GOTO Nextell

6140 Rubs1:   Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
         CHR$(13)&CHR$(7)&CHR$(19)                                    !***
6150          Tell$="Rubouts not ignored"
6160          GOTO Nextell
6170 Rubs0:   IF Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(
         Prompt)&CHR$(13)&CHR$(7)&CHR$(19)&CHR$(127) THEN Rubs1
         !***
6180          Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
                 CHR$(13)&CHR$(7)&CHR$(19)&CHR$(127)
                 !***
6190          Tell$="Rubouts ignored"
6200          GOTO Nextell

6210 Trace0:  IF Trace=0 THEN Trace1           ! Trace on/off
6220          Trace=0                          ! Trace on/off
6230          GOTO Trace2
6240 Trace1:  IF Hardselect()16 THEN Traceon
6250 Traceig: Tell$="Command ignored -- Hardcopy printer required"
6260          GOTO Nextell
6270 Traceon: Trace=1
6280 Trace2:  Tell$="Display functions "&State$(Trace)
```

LMGTRM

```
6290          OUTPUT Hardselect;CHR$(27);CHR$(90-Trace)
6300          Printall=0
6310          GOTO Nextell

6320 Setup:   IF Gra_mode THEN Bad
6330          Setup=1
6340          IF Define THEN Setup=0
6350          IF NOT Local THEN Strip
6360          BEEP
6370          DISP Rub$&" Local mode exited "&Rub$
6380          WAIT 2500
6390          GOTO Remote

6400 Status:  IF Printall THEN GOSUB Setprinter
6410          Statascii=Ascii
6420          Ascii=1
6430          PRINT RPT$("-",26);"Status of terminal emulator";RPT$("-
                ",27)
6440          PRINT "Echo ";State$(Echo),"Hardcopy ";State$(Printall),
6450          ON (LEN(Select$)<>8)+1 GOTO 6460,6480
6460          PRINT "Rubouts";State$&" not ignored";TAB(60);"  Display
                funcs. ";State$(Trace)
6470          GOTO 6490
6480          PRINT "Rubouts ";State$&" ignored";TAB(60);"  Display
                funcs. ";State$(Trace)
6490          PRINT "Recording ";State$(Recording),
6500          PRINT "Local ";State$(Local),"Cursor at";
6510          PRINT Cursor;TAB(60);"  Background ";State$(Gojump)
6520          IF Character=17 THEN PRINT "P";
6530          IF Character<>17 THEN PRINT "No p";
6540          PRINT "rompt";TAB(21);"Keyboard lock ";State$(Locked);
6550          PRINT TAB(41);"Hard. printer at "&VAL$(Hardselect)
6560          PRINT "Autosend ";State$(Autosend);TAB(21);
6570          PRINT "Baud rate";Baudrate;TAB(40);Stopbits/2+.5;"stop
                bit";
6580          PRINT RPT$("s",Stopbits<>1);TAB(60);"  Compatability ";
6590          PRINT State$(Comp_mode);LIN(1);VAL$(Bitsperchar);
6600          PRINT " bits per character";TAB(41);"I/O card at";
                Selectcode
6610          PRINT "Parity ";
6620          IF (Parity=0) OR (Parity=1) THEN PRINT "(odd) ";
6630          IF (Parity=2) OR (Parity=3) THEN PRINT "(even) ";
6640          IF (Parity=1) OR (Parity=3) THEN PRINT "enabled";
6650          IF (Parity=0) OR (Parity=2) THEN PRINT "disabled";
6660          PRINT TAB(41);"Bit rate factor is 1";
6670          IF Bitratefactor=2 THEN PRINT "/16";
6680          IF Bitratefactor=3 THEN PRINT "/64";
6690          PRINT LIN(1);"Handshake "&State$(Handshake);
6700          PRINT TAB(41);"Graphics input term.: ";
6710          IF Gt$=CHR$(13) THEN PRINT "[cr]";
6720          IF Gt$=CHR$(13)&CHR$(4) THEN PRINT "[cr][eot]";
6730          IF (Plottercode=0) OR (Plottercode=13) THEN 6780
```

```
6740          PRINT LIN(1);"Alternative plotter: ";
6750          IF Phpib=999 THEN PRINT Plotter$&" at";Plottercode;
                 State$(Altplot)
6760          IF Phpib<>999 THEN PRINT Plotter$&" at";Plottercode;",";
                 Phpib;State$(Altplot)
6770          GOTO 6790
6780          PRINT LIN(1);"No alternative plotter specified"
6790          PRINT "File name: "&Datafile$;TAB(41);"Default file
                 size:";Filesize
6800          PRINT "Current line:"
6810          K2=0
6820          Ktab=99
6830 Stat5:   K2=K2+1
6840          IF K2>LEN(Buffer$) THEN Stat6
6850          Key=NUM(Buffer$[K2])
6860          GOSUB Knam
6870          IF LEN(Nam$)+Ktab<79 THEN Stat7
6880          PRINT LIN(1);TAB(20);
6890          Ktab=20
6900 Stat7:   PRINT Nam$&" ";
6910          Ktab=Ktab+LEN(Nam$)+1
6920          GOTO Stat5
6930 Stat6:   PRINT LIN(1);"Key definitions:";
6940          K2=0
6950 Stat1:   K2=K2+1
6960          K$=""
6970          IF K2>LEN(Def$) THEN Statusart
6980          IF Def$[K2;1]<>CHR$(255) THEN Stat3
6990          Ktab=0
7000          K$=":"
7010          K2=K2+1
7020          PRINT LIN(1);TAB(5);
7030 Stat3:   Key=NUM(Def$[K2])
7040          Ascii=1
7050          IF Key<>127 THEN 7100
7060          K2=K2+1
7070          Key=NUM(Def$[K2])
7080          IF Key=127 THEN 7100
7090          Ascii=0
7100          GOSUB Knam
7110          Ktab=Ktab+LEN(Nam$&K$)+1
7120          IF Ktab<76 THEN Stat4
7130          PRINT LIN(1);TAB(20);
7140          Ktab=21+LEN(Nam$&K$)
7150 Stat4:   PRINT Nam$&K$&" ";
7160          IF NOT LEN(K$) THEN Stat1
7170          PRINT TAB(20);
7180          Ktab=20
7190          GOTO Stat1
7200 Dsr:     IF Printall THEN GOSUB Setprinter
7210          PRINT RPT$("-",80);
7220          Retflag=1
```

LMGTRM

```
7230          GOSUB Statusart
7240          Retflag=0
7250          Tell$="Data set ready"
7260          IF NOT BINAND(K1,128) THEN Tell$="Data set not ready"
7270          GOTO Nextell
7280 Statusart: WRITE IO Selectcode,5;1
7290          READ IO Selectcode,4;K1      ! Read status word
7300          WRITE IO Selectcode,4;55     ! Reset error bits (if any)
7310          WRITE IO Selectcode,5;0
7320          READ IO Selectcode,4;K2
7330          WRITE IO Selectcode,7;0      ! Cock the card for
                 interrupts
7340          WRITE IO Selectcode,5;132
7350          PRINT LIN(2),"USART status control word (R4E):",LIN(1)
7360          PRINT "Data set    Zero    Framing   Overrun   Parity
                 Transmtr   Receiver  Transmtr"
7370          PRINT " ready    (Unused)   error      error      error
                 empty      ready      ready"
7380          IMAGE #,3X,D,6X
7390          FOR K2=7 TO 0 STEP -1
7400             K3=BIT(K1,K2)
7410             PRINT USING 7380;K3
7420          NEXT K2
7430          PRINT
7440 Stat2:   PRINT LIN(1),RPT$("-",80)
7450          PRINTER IS 16
7460          IF Retflag THEN RETURN
7470          Ascii=Statascii
7480          GOTO Strip

7490 Local:   !
7510          Local=Localflag=1
7520          Old=0
7530          Tell$="Local"
7540          GOTO Nextell
7550 Remote:  IF NOT Localflag THEN Remote1
7560          Prev$=Prev$&CHR$(Rmchr)
7570          GOTO Nextell
7580 Remote1:Local=0
7590          Tell$="Remote"
7600          GOTO Nextell

7610 Asciiudk: IF POS(Def$,CHR$(255)&CHR$(Key)) THEN User1 ! Check
        for User Def.
7620 Ascii:   IF Cursor(160-LEN(Prompt$) THEN Addon
7630 Bad:     BEEP
7640          GOTO Strip
7650 Addon:   IF Insertmode THEN Buffer$[Cursor]=" "&Buffer$[Cursor,
        158]
7660          Buffer$[Cursor,Cursor]=CHR$(Key)
7670          Cursor=Cursor+1
7680          IF NOT Gra_mode OR Esc_sub THEN Dis_key
```

```
7690          X1=X*(560/1024)                                        LMGTRM
7700          Y1=Y*(455/788)
7710          IF NOT Altplot THEN 7750
7720             MOVE X,Y
7721             CSIZE 2
7730             LABEL CHR$(Key)
7740             GOTO 7760
7750          GPRINT X1,Y1,CHR$(Key)
7760          X=X+14
7770          IF X<1011 THEN Poi
7780          X=(Marg-1)*Marg2
7790          Y=Y-21
7800          IF Y<20 THEN Y=767                          !***WW
7810 Poi:     POINTER X+7,Y+12,2
7820          GOTO Strip
7830 Dis_key:TDISP CHR$(Key)
7840 Next:    TDISP CHR$(12)&Prompt$&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-
        Cursor+1)
7850 Strip:   Key$=Key$[2]              ! Next character from KBD$
7860          GOTO More


Character definition mode

7880 Dmode:   IF NOT Ascii AND (Key=9) AND (Cursor<>1) THEN Dleav
7890          IF LEN(Def$)<299 THEN Dent
7900          Tell$="Buffer full - press UDK 9"
7910          GOSUB Tell
7920          Clear=1E99
7930          BEEP
7940          GOTO Next
7950 Dent:    Tell$="Enter definition characters or UDK 9"
7960          GOSUB Tell
7970          Clear=1E99
7980          IF Cursor<>1 THEN Dmore
7990          IF NOT Ascii AND (Key=20) THEN Key=254 ! Store key
                 redefines EOL
8000          GOSUB Knam
8010          IF Ascii THEN K1=POS(Def$,CHR$(255)&CHR$(Key))
8020          IF NOT Ascii THEN K1=POS(Def$,CHR$(255)&CHR$(127)&CHR$(
                 Key))
8030          IF NOT K1 THEN Dfrst            ! Check for previous
                 defintion
8040 D_let:   Def$[K1]=Def$[K1+1]             ! Weed out the old
        definition
8050          IF (Def$[K1;1]<>CHR$(255)) AND (K1<=LEN(Def$)) THEN D_let
8060 Dfrst:   Savelen=LEN(Def$)               ! Save old definition
        length
8070          Oldascii=Ascii
8080          Def$=Def$&CHR$(255)             ! Append new key
8090          IF NOT Ascii THEN Def$=Def$&CHR$(127)
```

LMGTRM

```
8100          Def$=Def$&CHR$(Key)
8110          Buffer$=Nam$&": "
8120          GOTO Drest
8130 Dmore:   IF NOT Ascii THEN Def$=Def$&CHR$(127)
8140          Def$=Def$&CHR$(Key)              ! Append next part of
                definition
8150          GOSUB Knam                       ! Find name of key
8160          IF LEN(Buffer$&Nam$)>158 THEN Buffer$=Buffer$[LEN(Nam$)+
                3]
8170          Buffer$=Buffer$&Nam$&" "
8180 Drest:   Cursor=LEN(Buffer$)+1            ! Step cursor counter
8190          GOTO Next
8200 Derr:    Tell$="No more room for definitions"
8210          BEEP
8220 Dbyebye:GOSUB Tell
8230          GOTO Dbye
8240 Dleav:   IF Oldascii AND (Def$[Savelen+2;1]=Def$[Savelen+3]) THEN
        Dclr
8250          IF NOT Oldascii AND (Def$[Savelen+2;2]=Def$[Savelen+4])
                THEN Dclr
8260          IF NOT Oldascii AND (NUM(Def$[Savelen+3])=254) THEN Eol$=
                Def$[Savelen+4,MIN(300,Savelen+23)]
8261          DISP
8270          IF Lfkey AND (Eol$=CHR$(13)) THEN Tell$="Auto linefeed
                off"
8280          IF Lfkey AND (Eol$=CHR$(13)&CHR$(10)) THEN Tell$="Auto
                linefeed on"
8290          IF Lfkey THEN GOSUB Tell
8291          Lfkey=0
8300 Dbye:    Definemode=0                     ! Exit define mode
8310          Prompt$=Saveprompt$
8320          Buffer$=Savebuffer$
8330          Cursor=Savecursor
8340          GOTO Next
8350 Dclr:    Def$=Def$[1,Savelen]
8360          Tell$=Nam$&" key cleared"
8370          GOTO Dbyebye


8380 Knam:    Nam$=CHR$(Key)                   ! Check for case of name
        being char.
8390          IF Ascii AND (Key>32) AND (Key<254) THEN RETURN
8400          K3=1
8410          Shift=0
8420          IF Ascii THEN 8490
8430          Shift=Key DIV 64
8440          IF Key=254 THEN Shift=0
8450          K3=POS(Name$,CHR$(255))+1
8460                                           ! 255 separates Ascii 0
                                                 through 32
8470                                           !  from non-ascii keys.
8480          Nam$=CHR$(Key-Shift*64)
```

```
8490          K1=POS(Name$[K3],Nam$)
8500          Nam$=Name$[K1+K3,MIN(LEN(Name$),K1+K3+17)]
8510          FOR K1=1 TO LEN(Nam$)
8520            IF (Nam$[K1]="~") OR (Nam$[K1]<",") THEN Dnam
8530          NEXT K1
8540 Dnam:    IF NOT Shift THEN Nam$="["&Nam$[1,K1-1]&"]"
8550          IF NOT (Shift-1) THEN Nam$="[sh "&Nam$[1,K1-1]&"]"
8560          IF NOT (Shift-2) THEN Nam$="[cntl "&Nam$[1,K1-1]&"]"
8570          IF NOT (Shift-3) THEN Nam$="[sh cntl "&Nam$[1,K1-1]&"]"
8580          RETURN
```

Clear definitions of characters

```
8600 Defsclear:
8610          Eol$=CHR$(13)
8620          Def$=""
8630          RESTORE Defs
8640 Read:    READ D1,D2,Savebuffer$
8650          IF D1=-1 THEN Clline
8660          Def$=Def$&CHR$(255)
8670          IF D1=127 THEN Def$=Def$&CHR$(D1)
8680          Def$=Def$&CHR$(D2)&Savebuffer$
8690          GOTO Read
8700 Clline:  D1=POS(Def$,"|")                ! Add 128 to codes
        following |.
8710          IF NOT D1 THEN Stores
8720          Def$[D1]=CHR$(NUM(Def$[D1+1])+128)&Def$[D1+2]
8730          GOTO Clline
8740 Stores:  D1=POS(Def$,"~")                ! Subtract 64 from codes
        after ~.
8750          IF NOT D1 THEN RETURN
8760          Def$[D1]=CHR$(NUM(Def$[D1+1])-64)&Def$[D1+2]
8770          GOTO Stores
```

Data comm link service routine

```
8790 Datacomservice:
8800          IF Local THEN Trashit
8810          Com$=TBUF$                      ! Dump input buffer
8820          IF POS(Com$,Enq$) AND NOT Gra_mode THEN OUTPUT
                Selectcode;Ack$;
8830 Com0:    FOR I=1 TO LEN(Com$)
8840          GOTO Com2
8850 Com:     NEXT I
8860          RETURN
8870 Com2:    Char=NUM(Com$[I])
8880          Character=BINAND(127,Char)
8890          Char$=CHR$(Character)
```

LMGTRM

```
8900          IF (Character=22) OR NOT Character THEN Com! Skip out
                 syn-char;null
8901          IF (Character=5) AND NOT Gra_mode THEN Com      !***WW
8910          IF Trace THEN OUTPUT Hardselect USING "#,A";Char$
8915          IF POS(Select$,CHR$(127)) AND (Character=127) AND NOT
                 Gra_mode THEN Com!
                 !***WW
8920          IF NOT Recording THEN Com1
8930              Comstr$=Comstr$&Char$
8940              IF (LEN(Comstr$)(80) AND ((Char$()CHR$(10)) AND (
                     Char$()CHR$(Prompt))) THEN Com1
8950              PRINT #1;Comstr$
8960              Comstr$=""
8970 Com1:    ON 1+POS(Select$,Char$) GOTO Nctrl,Gs,Us,Lf,Esc,Cret,Cr,
        Bell,Dc3,Check_rubouts,Dcrmt,Com
        !***WW


8971 Check_rubouts: IF Gra_al THEN Com                    !***WW
8980 Nctrl:   IF NOT Gra_mode OR Escseq THEN Nctrl1 ! Jump if normal
        mode
8990          IF NOT Gra_al THEN Coor_rec           ! Jump if plot
                 coord
9000          IF Pointer THEN POINTER -1,-1,2       ! Clear cursor set
                 after D2
9010          Pointer=0
9020          IF Char$(CHR$(32) THEN Com
9030          X1=X*(560/1024)                       ! Calc. Gprint X
9040          Y1=Y*(455/788)                        ! Calc. Gprint Y
9050          IF NOT Altplot THEN 9090
9060             MOVE X,Y
9070             LABEL Char$
9080             GOTO 9100
9090          IF X1(1011 THEN GPRINT X1,Y1,Char$    ! Plot alpha char.
9100          X=X+14
9110          GOTO Com
9120 Coor_rec:IF NOT Comp_mode THEN Nctrl1
9130          V=NUM(Char$)                          ! get coord. char

9140          IF (W=1) AND (V(64) THEN Hy
9150          IF V>95 THEN Ly
9160          IF V(64 THEN Hx
9170 Lx:      W(4)=V                                ! Low X, 4.Byte
9180          W=1
9190          GOTO Draw
9200 Hy:      W=2                                   ! High Y, 1.Byte
9210          W(1)=V
9220          GOTO Com
9230 Ly:      W=3                                   ! Low Y, 2.Byte
9240          W(2)=V
9250          GOTO Com
9260 Hx:      W(3)=V                                ! High X, 3.Byte
```

```
9270          GOTO Com


9280 Esc_qu:                                          ! ***WW
9290          Char$=CHR$(127)
9300          Escseq=0
9310          GOTO Nctrl
9320 Nctrl1:  IF Escseq THEN Eseq
9321          IF Char$<>CHR$(Rmchr) THEN 9330
9322          Local=0
9323          Tell$="Remote"
9324          GOSUB Tell
9325          GOTO Com
9330          IF LEN(Disp$)>79 THEN GOSUB Print
9340          Disp$[LEN(Disp$)+1]=Char$
9350          IF (Character>31) AND (Character<>8) THEN TDISP Char$
9360          GOTO Com


9370 Gs:     IF NOT Comp_mode THEN Nctrl1
9380          Not_draw=1                      ! GS=Switch graphic; Not
                 graph.-alpha
9390          Gra_al=0
9400          Escseq=Esc_sub=0
9410          W=1                             ! Init char counter
9420          IF Gra_mode THEN Com            ! Jump if already gra-mode
9430 Gs1:     IF NOT Comp_mode THEN Nctrl1
9440          Gra_mode=1                      ! Set gra-mode
9450          IF Altplot AND (Phpib<>999) THEN PLOTTER IS Plottercode,
                 Phpib,Plotter$
9460          IF Altplot AND (Phpib=999) THEN PLOTTER IS Plottercode,
                 Plotter$
9470          IF NOT Altplot THEN PLOTTER IS "GRAPHICS"
9480          IF NOT Altplot THEN GRAPHICS
9490          ON KBD 10 GOSUB Keyservice ,ALL
9500          SCALE 0,1024,0,788              ! Scale
9510          GCLEAR
9520          W(1)=55                         ! Init. the 4 bytes of
                 coords.
9530          W(2)=96
9540          W(3)=32
9550          W(4)=64
9560          X=0
9570          Marg=1
9580          Y=767
9590          GOTO Com

9600 Cr:      IF NOT Gra_mode THEN Com          ! CR in normal mode
9610          Gra_al=1                          ! Graphics-alpha
9620          X=(Marg-1)*Marg2                  ! X-pos to 0
```

LMGTRM

```
9630          GOTO Com
9640 Draw:    X=W(4)-64+(W(3)-32)*32            ! Calc X-coord
9650          Y=W(2)-96+(W(1)-32)*32            ! Calc Y-coord
9660          W=1                               ! Reset Char-counter
9670          IF NOT Not_draw AND NOT Esc_sub THEN Draw1       ! Jump
                if plotting
9680          MOVE X,Y
9690          Not_draw=0
9700          GOTO Com

9710 Draw1:   DRAW X,Y                      ! Draw graphic
9720          GOTO Com

9730 Us:      IF NOT Comp_mode THEN Com
9740          Gra_al=1                      ! US=alpha mode
9741          Esc_sub=0
9750          IF NOT Gra_mode THEN Gs1
9760          GOTO Com

9770 Esc_ff:  IF NOT Comp_mode THEN Com
9780          Gra_al=1                  ! ESC-FF=end graphics; clear CRT;
                g-alpha
9790          GCLEAR
9800          Escseq=0
9810          GOTO Gs1

9820 Esc_etb: IF NOT Comp_mode THEN Com    ! Esc [etb] - DUMP GRAPHICS
9830         Escseq=0
9840         OUTPUT 0;CHR$(27)&"Z"&CHR$(8)
9850         DUMP GRAPHICS
9860         IF Trace THEN OUTPUT 0;CHR$(27)&"Y"
9870         GOTO Com

9880 Esc_sub1:Esc_sub=1
9881          Gra_al=0                                  !***WW
9890 Esc_sub:Escseq=0
9900          IF NOT Esc_sub THEN POINTER X+7,Y+12,2      ! pos. cursor
9910          IF Esc_sub THEN POINTER X,Y,1
9920 Cal_xy:  R=ABS((X/32-INT(X/32))*32)                ! Calc HX, HY
9930          Z=INT(X/32)
9940          W(4)=32+R                                 !***WW
9950          W(3)=32+Z
9960          R=ABS((Y/32-INT(Y/32))*32)                ! Calc. HY,LY
9970          Z=INT(Y/32)
9980          W(2)=32+R                                 !***WW
9990          W(1)=32+Z
10000         W$=CHR$(W(3))&CHR$(W(4))&CHR$(W(1))&CHR$(W(2))
10010         IF Es_enq AND NOT Esc_sub THEN Tra_xy ! Jump if only ESC-
                ENQ
10020         IF Es_enq AND Esc_sub THEN Only_co    ! Jump if ESC-SUB,
                ESC-ENQ
10030         GOTO Com
```

```
10040 Only_co:POINTER -1,-1,2                          ! Send coords
10050         OUTPUT Selectcode USING "#,K";W$&Gt$;
10060         Es_enq=Escseq=Esc_sub=0                  ! After ESC-SUB,
                 ESC-ENQ
10070         GOTO Com

10080 Esc_enq:Es_enq=1
10090         GOTO Cal_xy
10100 Tra_xy:Stb=BINIOR(St,128+32+16*(Hardprinter()16)+4*Gra_al+2*(
         Marg=1))    !CALC STATUS BYTE
10110         OUTPUT Selectcode USING "#,K";CHR$(Stb)&W$&Gt$;! Send X-
                 Y coords.
10120         POINTER -1,-1,2                          ! Clear cursor
10130         Es_enq=Escseq=Esc_sub=0
10140         GOTO Com

10150 Gr_lf:  Gra_al=1                        ! LF in gra. mode
10160         IF NOT Escseq THEN No_esc       ! for ESC/LF
10170         Escseq=0
10180         X=(Marg-1)*Marg2                ! ESC/LF as CR
10190         GOTO Com

10200 No_esc:IF Y>21 THEN Pos_c               ! if last line
10210         Marg=3-Marg
10220         Y=788                           ! Top of CRT  ***WW
10230         X=(Marg-1)*Marg2
10240 Pos_c:  Y=Y-21                          ! next line
10250         GOTO Com

10260 Lf:     IF Gra_mode THEN Gr_lf
10270         GOSUB Print
10280         GOTO Com
10290 Dcrmt:  Local=0
10300         Tell$="Remote"
10310         GOSUB Tell
10320         GOTO Com

10330 Bell:   BEEP
10340         GOTO Com
10350 Dc3:    IF NOT Handshake THEN Xon=0
10360         GOTO Com
10370 Trashit:IF POS(TBUF$,Enq$) THEN OUTPUT Selectcode;Ack$;
10380         RETURN

10390 Esc:    Escseq=1
10391         Eseq$=""                                 !***WW
10400         GOTO Com

10410 Eseq:   IF (Character=13) OR (Character=10) THEN Com!***
10420         IF NOT Comp_mode THEN Norm_esc
10430         IF Character=63 THEN Esc_qu
10440         IF Character=12 THEN Esc_ff
```

LMGTRM

```
10450          IF Character=26 THEN Esc_sub1
10460          IF Character=5 THEN Esc_enq
10470          IF Character=23 THEN Esc_etb
10480          IF Character=29 THEN Gs
10490          IF NOT Gra_mode THEN Norm_esc
10500 Norm_esc:IF LEN(Eseq$)<18 THEN Escokay  ! Check for sequence
         too long
10510 Termesc:GOSUB Termes
10520          GOTO Com
10530 Termes:Escseq=0                         ! Terminate escape
         processing
10540          Eseq$=""
10550          RETURN
10560 Escokay:Eseq$=Eseq$&Char$               ! Tack on new character
10570          IF (Character>63) AND (Character<91) THEN Endseq
10580                                         ! Check for end of
                                                 sequence
10590          GOTO Com

10600 Endseq:SERIAL
10610          IF LEN(Eseq$)=1 THEN Oneseq
10620          IF (Eseq$[1;1]<>"&") OR (LEN(Eseq$)<3) THEN Termesc
10630          ON 1+POS(Parm$,Eseq$[2;1]) GOTO Termesc,Dparm,Pparm,
                  Cparm,Sparm

10640 Dparm:  K4=POS(High$,Eseq$[3;1])
10650          IF NOT K4 THEN Termesc
10660          Disp$[LEN(Disp$)+1]=CHR$(127+K4)
10670          IF LEN(Disp$)=80 THEN GOSUB Print
10680          GOTO Termesc

10690 Pparm:  IF LEN(Eseq$)>3 THEN Pcheck
10700          K4=POS(Eswitch$,Eseq$[3;1])
10710          GOSUB Termes
10720          ON K4+1 GOTO Termesc,Chain,Fdfn,Purge,Setsize
10730 Sparm:IF Eseq$[6;1]<>"Q" THEN Termesc
10740         IF (Eseq$[3;3]="1p0") OR (Eseq$[3;3]="0p1") THEN Comp_on
10750         IF Eseq$[3;3]="0p0" THEN Nocomp
10760         GOSUB Termes
10770         GOTO Com
10780 Pcheck:IF Eseq$[4;1]<>"P" THEN Dcheck
10790          IF (Eseq$[3;1]<>"0") OR (Eseq$[3;1]<>"1") THEN Com  !***
                  WW
10800          K4=VAL(Eseq$[3;1])
10810          GOSUB Termes
10820          ON K4+1 GOTO Altplot_off,Altplot_on
10830 Dcheck:IF Eseq$[4;1]<>"D" THEN Scheck  ! Destination commands
10840          IF (Eseq$[3;1]>"5") OR (Eseq$[3;1]<"0") THEN Com
10850          K4=VAL(Eseq$[3;1])
10860          GOSUB Termes
10870          ON K4+1 GOTO Toff,Ton,Ton,Toffpall1,Pall,Pall1
```

```
10880 Scheck:IF Eseq$[4;1]<>"S" THEN Com      ! Source commands
10890        IF (Eseq$[3;1]>"3") OR (Eseq$[3;1]<"1") THEN Com
10900        K4=VAL(Eseq$[3;1])
10910        GOSUB Termes
10920        ON K4 GOTO Autosend_on,Autosend_on,Autosend_off
10930 Comp_on: IF Gra_mode THEN Bad
10940          Comp_mode=1                !Esc&s0p1Q
10950          Comment$="Compatability mode on"        !***WW
10960          Escseq=Local=Keycomp=0
10970          Eseq$=""
10990          GOTO Comcom                             !***WW
11000 Nocomp: Comp_mode=0        !     Esc&s0p0Q
11010         Local=Keycomp=Escseq=0
11020         Comment$="Compatability mode off"        !***WW
11030          Eseq$=""
11050          GOTO Comcom                             !***WW
11060 Oneseq:! Single character escapes
11070        GOSUB Termes
11080        IF (Character=89) OR (Character=90) THEN Tron
11090        GOTO Com

11100 Altplot_on:IF (Plottercode<>13) AND (Plottercode<>0) THEN
         11150!Esc&p1P
11110        Comment$="No alternate plotter specified"
11120        GOTO Comcom
11130        IF Phpib=999 THEN PLOTTER IS Plottercode,Plotter$
11140        IF Phpib<>999 THEN PLOTTER IS Plottercode,Phpib,Plotter$
11150        CSIZE 2
11160        GOSUB Testalt
11170        GOTO Comcom

11180 Altplot_off:Altplot=0
11190         Comment$="Alternate plotter off"
11200         PLOTTER IS "GRAPHICS"
11210         GOTO Comcom

11220 Rec_file: IF Autosend THEN Bad
11225         IF Local THEN Locflag=1
11230          IF Recflag THEN Endrec
11240         Prev$=CHR$(27)&"&p1D"
11250         Local=Keyrec=1
11260         Old=0
11270         RETURN

11280 Endrec: Prev$=CHR$(27)&"&p0D"
11290         Old=0
11295         IF Local THEN Locflag=1
11300         Local=Keyrec=1
11310         RETURN

11320 Ton:    Comment$="Tape recording start" ! Esc&p1D or Esc&p2D
11321        IF Keyrec AND NOT Locflag THEN Local=0
```

LMGTRM

```
11322          Locflag=0
11330          Disp$=""
11340          Recflag=1
11350          Locked=Firstline=1
11360          ON ERROR GOTO Eerr
11370 Tonloop:ASSIGN #1 TO Datafile$,T1
11380          IF T1>1 THEN Eerr
11390          Recording=1
11400          OFF ERROR
11410          IF NOT T1 THEN Pdone
11420          CREATE Datafile$,Filesize
11430          GOTO Tonloop
11440 Eerr:    ASSIGN #1 TO *
11450          Recording=0
11460          GOTO Xerr
11470 Toff:    K4=POS(Comstr$,CHR$(27)&"&p0D")
11480          IF K4<>0 THEN Comstr$=Comstr$[1,K4-1]
11490          PRINT #1;Comstr$,END
11500          ASSIGN #1 TO *
11510          Comstr$=""
11511          IF Keyrec AND NOT Locflag THEN Local=0
11512          Locflag=0
11520          Recflag=Recording=0
11530          Firstline=1
11540          Comment$="Tape file close"
11550          GOTO Pdone
11560 Toffpall1:ASSIGN #1 TO *                  ! Esc&p3D
11570          Printall=Recording=0
11580          Firstline=1
11590          Comment$="Tape file close and hardcopy off"
11600          GOTO Pdone
11610 Deffil:IF Gra_mode THEN Bad
11620          Define=1
11630         GOTO Setup
11640 Fil_wt:WAIT 1000
11650 Fil_def:OFF KBD
11660           Define=0
11670           INPUT "Enter file name:  [Use CONT]",Datafile$
11680           ON ERROR GOTO Crerror
11690           ASSIGN #9 TO Datafile$,T1
11700           ASSIGN #9 TO *
11710           OFF ERROR
11720           IF T1=1 THEN Crerror
11730 F_ok:     Tell$="File name redefined to "&Datafile$
11740           GOSUB Tell
11750           ON KBD 10 GOSUB Keyservice
11760           IF Gra_mode AND NOT Gra_al THEN ON KBD 10 GOSUB
                   Keyservice ,ALL
11770           GOTO Wait
11780 Crerror:OFF ERROR
11790          IF T1<>2 THEN Crfil
11800           BEEP
11810           Tell$=Datafile$&" not found -- error "&VAL$(ERRN)
11820           GOSUB Tell
```

```
11830          GOTO Fil_wt
11840 Crfil:INPUT "Enter size of file:  [Use CONT]",Filesize
11850        CREATE Datafile$,Filesize
11860        GOTO F_ok
11870 Testalt:ON INT #Plottercode GOTO Ploterr
11880          ON ERROR GOTO Ploterr
11890          SET TIMEOUT Plottercode;1000
11900          STATUS Plottercode*(1+99*(Phpib<>999))+Phpib*(Phpib<>
                 999);St
11910          Comment$="Alternate plotter on"
11920          Altplot=1
11930          GOTO 11950
11940 Ploterr:Comment$="Alternate plotter not found"
11950          St=TIME OUT(Plottercode)
11960          OFF ERROR
11970          OFF INT #Plottercode
11980          RETURN
11990 Chain: Comment$="Background program link" ! Esc &pE
12000        Locked=Firstline=1
12010        ON ERROR GOTO Xerr
12020        ASSIGN Datafile$ TO #1,T1
12030        IF T1 THEN Xerr
12040        LINK Datafile$,13970,Gstrt
12050 Gstrt: IF NOT Local THEN OUTPUT Selectcode USING Pfmt;"S"
12060        Comment$=Comment$&" successful"
12070        GOTO Oferr

12080 Cparm: K4=1
12090        IF Eseq$[3;1]<>"E" THEN K4=0
12100        GOSUB Termes
12110        IF NOT K4 THEN Com
12120 Gstrt1:Gojump=1                        ! Esc +
12130 Gstrt2:IF NOT Local THEN Oferr
12140        BEEP
12150        DISP Rub$&" Local mode exited "&Rub$
12160        WAIT 2500
12170        Local=0
12180        GOTO Oferr

12190 Fdfn:  T1=0                            ! Esc &pG
12200 Fdfn0: T1=T1+1
12210 Fdfn2: IF T1>LEN(Disp$) THEN Fdfn1
12220        IF Disp$[T1;1]>" " THEN Fdfn0
12230        Disp$[T1]=Disp$[T1+1]
12240        GOTO Fdfn2
12250 Fdfn1: ON ERROR GOTO Illegalfile
12260        Datafile$=Disp$
12270        ASSIGN #9 TO Datafile$,X
12280        ASSIGN #9 TO *
12290        OFF ERROR
12300        Disp$=""
12310        Comment$="File name redefined to "&Datafile$
```

LMGTRM

```
12320          GOTO Comcom
12330 Illegalfile:OFF ERROR
12340          BEEP
12350          Comment$=Disp$&" not found -- error "&VAL$(ERRN)
12360          Disp$=""
12370          GOTO Comcom

12380 Pall:    Comment$="Hardcopy on"          ! Esc&p4D
12390          Trace=0
12400          Printall=1
12410          GOTO Comcom
12420 Palll:   Comment$="Hardcopy off"         ! Esc&p5D
12430          Printall=0
12440          GOTO Comcom

12450 Setsize:ON ERROR GOTO Default            ! Esq &pA
12460          Filesize=VAL(Disp$)
12470 Size1:   OFF ERROR
12480          Comment$="Filesize set to "&VAL$(Filesize)
12490          Disp$=""
12500          GOTO Comcom
12510 Default:Filesize=30
12520          GOTO Size1
12530 Purge:   Comment$="File purge"           ! Esc &pN
12540          Locked=Firstline=1
12550          ON ERROR GOTO Xerr
12560          ASSIGN #1 TO Datafile$,T1
12570          IF T1=1 THEN Pdone
12580          ASSIGN #1 TO *
12590          PURGE Datafile$
12600 Pdone:   WAIT 500
12610          IF NOT Local AND NOT Keyrec THEN OUTPUT Selectcode USING
                  Pfmt;"S"
12620          Comment$=Comment$&" successful"
12630          IF NOT Keyrec THEN Oferr
12640          Keyrec=0
12650 Oferr:   OFF ERROR
12660          Locked=0
12670          GOTO Comcom
12680 Xerr:    WAIT 500
12690          IF NOT Local AND NOT Keysend AND NOT Keyrec THEN OUTPUT
                  Selectcode USING Pfmt;"F"
12700          Sendflag=Autosend=Keysend=Keyrec=Keycomp=0
12710          Comment$=Comment$&" unsuccessful"
12720 Pfmt:    IMAGE #,A,L
12730          GOTO Oferr

12740 Send_file: IF Gra_mode OR Recording THEN Bad
12750             IF Local THEN Locflag=1
12760             IF Autosend THEN Endsend
12770             Prev$=CHR$(27)&"&p1S"
12780             Local=Keysend=1
```

```
12790              Old=0
12800              RETURN
12810 Endsend:  Prev$=CHR$(27)&"&p3S"
12811           IF Local THEN Locflag=1
12820           Local=Keysend=1
12830           Old=0
12840           RETURN

12850 Autosend_on:Autosend=Sendflag=1       ! Esc&p1S OR Esc&p2S
12860        Comment$="Autosend "
12870        IF Keysend AND NOT Locflag THEN Local=0
12880        Locflag=0
12890        ON ERROR GOTO Xerr
12900        ASSIGN #2 TO Datafile$,T1
12910        IF T1 THEN Xerr
12920        OFF ERROR
12930        GOSUB Autocomment
12940        WAIT 500
12950        IF NOT Local AND NOT Keysend THEN OUTPUT Selectcode
                USING Pfmt;"S"
12960        Keysend=0
12970        GOTO Comcom
12980 Autosend_off:Autosend=Sendflag=0      ! Esc&p3S
12990        WAIT 500
13000        IF NOT Local AND NOT Keysend THEN OUTPUT Selectcode
                USING Pfmt;"S"
13001        IF Keysend AND NOT Locflag THEN Local=0
13010        Keysend=0
13020        ASSIGN #2 TO *
13030        GOSUB Autocomment
13040        GOTO Comcom
13050 Autocomment:Comment$="Auto send "&State$(Autosend)
13060        RETURN

13070 Tron:  IF Hardselect()16 THEN Tron1  ! Esc Y and Z turns trace
        on/off.
13080        BEEP
13090        Comment$="Command ignored -- Hardcopy printer required"
13100        GOTO Comcom
13110 Tron1: Trace=(Character(90)
13120        GOSUB Setprinter
13130        PRINT CHR$(27);CHR$(Character)
13140        PRINTER IS 16
13150        Comment$="Display functions "&State$(Trace)
13160        Printall=0
13170 Comcom:Clear=1000
13180 Comcom1:DISP TAB(74-LEN(Comment$));Rub$&" "&Comment$&" "&Rub$
13190        IF NOT Keysend THEN Com
13200           Keysend=0
13210           RETURN
13220        GOTO Com
```

LMGTRM

```
13230 Cret:   IF NOT Gra_mode THEN Cret1
13240         Pointer=1
13250         IF NOT Esc_sub THEN POINTER X+7,Y+12,2
13260         IF Esc_sub THEN POINTER X,Y,1
13270         IF Esc_sub THEN Gra_al=0
13280         GOTO Com
13290 Cret1:  IF Firstline THEN Com
13300         IF Autosend THEN Sendflag=1
13310         IF NOT Handshake THEN Xon=1
13320         Prompt$=Disp$[1,70]
13330         Buffer$=Buffer$[1,159-LEN(Prompt$)]
13340         TDISP CHR$(12)&Prompt$&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-
                 Cursor+1)
13350         GOTO Com

13360 Record:T1=0
13370 Rec2:   T1=T1+1
13380 Rec1:   IF T1>LEN(Disp$) THEN Rec3
13390         IF Disp$[T1;1]<>CHR$(5) THEN Rec2
13400         Disp$[T1]=Disp$[T1+1]          ! Strip off Enq's
13410         GOTO Rec1
13420 Rec3:   ON ERROR GOTO Msfail
13430         IF LEN(Disp$) AND (Disp$<>CHR$(10)) THEN PRINT #1;Disp$
13440         Comstr$=""
13450         OFF ERROR
13460         RETURN
13470 Msfail:BEEP
13480         OFF ERROR
13490         DISP " Mass storage failure -- file closed "
13500         Recording=0
13510         WAIT 1000
13520         RETURN
13530 Print:  SERIAL
13540         IF Firstline THEN PRINT
13550         IF LEN(Disp$)<80 THEN PRINT Disp$&CHR$(128)
13555         IF LEN(Disp$)>79 THEN PRINT Disp$
13560 Print2:IF NOT Printall AND Hardselect-16 THEN Print1
13570 Print3:T1=POS(Disp$,CHR$(12))
13580         IF NOT T1 THEN Print4
13590         OUTPUT Hardselect;Disp$[1,T1-1]&RPT$(CHR$(10),6)
13600         Disp$=Disp$[T1+1]              ! Replace formfeed with 6
                 line feeds
13610         GOTO Print3
13620 Print4:OUTPUT Hardselect;Disp$&CHR$(128)
13630 Print1: Disp$=""
13640         TDISP CHR$(12)
13650         Firstline=0
13660         RETURN

13670 Set_hard:Hpibflag=(Hpib<>999)
13680         Hardselect=Hardprinter*(NOT Hpibflag+100*Hpibflag)+Hpib*
                 Hpibflag
```

```
13690         RETURN

13700 Setprinter:IF Hpibflag THEN PRINTER IS Hardprinter,Hpib
13710         IF NOT Hpibflag THEN PRINTER IS Hardprinter
13720         RETURN




!Data for initial definitions of characters

13740 Defs:
13750 DATA 127,71,^Y,127,65,^!?^I^!?^T^M^!?^I,127,6,^I
13760 DATA 127,70,^F,127,1,^!?^I^!?^T^M^J^!?^I,127,69,^!?^!?
13770 DATA 127,254,^M
13780 DATA -1,0,""
!

 Data for names of total initialization

13800 Initialconds:
13810 DATA 11,300,1,2,8,3,"off","on","DCdata:T15",30,17,1E99,0,999,21,
         "","",""
!
13830 DATA 0,nul,1,soh,2,stx,3,etx,4,eot,5,enq,6,ack,7,bel,8,bs,9,ht,
         10,lf
13840 DATA 11,vt,12,ff,13,cr,14,so,15,si,16,dle,17,dc1,18,dc2,19,dc3,
         20,dc4
13850 DATA 21,nak,22,syn,23,etb,24,can,25,em,26,sub,27,esc,28,fs,29,
         gs,30,rs
13860 DATA 31,us,32,space,255,0,UDK0,1,UDK1,2,UDK2,3,UDK3,4,UDK4,5,
         UDK5,6,UDK6
13870 DATA 7,UDK7,8,UDK8,9,UDK9,10,UDK10,11,UDK11,12,UDK12,13,UDK13,
         14,UDK14
13880 DATA 15,UDK15,16,step,17,pause,18,run,19,cont,20,store,21,
         execute,22,left
13890 DATA 23,right,24,up,25,down,26,roll-up,27,roll-down,28,home,29,
         clear
13900 DATA 30,clear-to-end,31,delete-char,32,insert-char,33,delete-
         line
13910 DATA 34,insert-line,35,recall,36,tab,37,tab-set,38,tab-clear,39,
         typewriter
13920 DATA 40,backspace,41,result,42,stop,43,clear-line,254,end-of-
         line,-1,""
!

 Data for 98036 card baud rates

13940 Baudrates:
13950 DATA 75,110,150,300,600,1200,1800,2400,4800,9600
```

LMGTRM

```
13960 End:! End of terminal emulator program
13970 END
```

Key Mode Program Listing

The terminal emulator program enables interrupt conditions at line 940 and 970 which cause branches to be taken whenever a key is pressed, or whenever any characters come in on the data comm channel. The interrupt on the keyboard is set to a higher priority, so that interrupt will take precendence over the data comm service routine.

The main program does nothing but sit in an idle loop testing several flags (such as Local(for local mode), Reinit(for going through the setup routine), Sendflag(for the autosend feature), Gojump(for background program linkage), and Notstopped(for termination of the program)). If a key is pressed, a GOSUB is performed to the line labelled Keyservice. A RETURN encountered in the Keyservice routine will cause the program to resume where it left off. Similarily, characters coming in on the data comm channel will cause a GOSUB to Datacomservice. Again, because of the priority scheme, the Datacomservice routine can be interrupted by the data comm channel, but the Keyservice routine can not be interrupted by the keyboard.

When a key is pressed causing a program branch to Keyservice, the keyboard buffer area KBD$ is immediately dumped, freeing it for further collection of key codes while the first batch are being processed. The string Key$ is then tested immediately to see if the break key (UDK 7) has been pressed, then to see if the alternative break key (Shift UDK 7) has been pressed, and finally to see if the Ack key (Shift UDK 6) has been pressed. Each of these three keys will be acted upon immediately, regardless of the order in which they were pressed relative to other members of the buffer.

Once all occurrences of the three keys listed above have been handled, the variable Key$ is processed one character at a time, starting at the line labelled More1. The ASCII character 127 is treated as a special case, since it is used as a delimiter for non-ASCII key codes inserted as a result of the key redefinition feature (255 delimits non-ASCII keys inserted from the keyboard). If the character itself is desired, two 127's will be in Key$ adjacent to each other. If neither a 127 nor a 255 is found, the ASCII key is tested to see if it has been redefined. If it has been redefined, the redefinition string is added to the front of Key$ and the loop is performed again. If the key wasn't redefined, then it is sent immediately to the computer, which saves up the characters sent until the STORE or EXECUTE key is pressed, which sends a carriage return (or other redefinable end-of-line sequence) to the computer, at which time the computer will interpret the line. If the Echo mode is off, each character is also saved in Disp$ for future viewing, and is also sent to the display line via a TDISP statement. When this sequence is completed, the next character in Key$ is processed, and so on until Key$ is empty. Then, a RETURN to the idle loop is executed.

On the other hand, if the character encountered in Key$ is 127 or 155, indicating a non-ASCII key, the next character is picked up to tell which non-ASCII key was pressed. That key is then tested for redefinition. If the key has been redefined, the redefinition string is added to the front of Key$, and the loop is performed again. If the key was not redefined, then the program will branch (via a series of ON...GOTO statements) to the execution routine for that particular key. Upon completion of the key's execution routine, the next code in Key$ is processed as before until Key$ is empty.

Whenever information comes into the data comm channel, a GOSUB is performed to the line labelled Datacomservice. If the terminal emulator is in local mode, all information sent from the computer will be ignored, and the program will RETURN to wherever it branched from.

If the terminal is not in local mode, then the buffer TBUF$ is immediately dumped into the variable Com$. Com$ is first checked for the occurrence of an Enq character (ASCII 5), to which the program will respond with an Ack (ASCII 6).

Next, each character in Com$ is processed, one character at a time. First, the leading bit is stripped off of each character to allow for parity. Next, the incoming characters are tested against a string called Select$, which contains a unit specifier, a group separator, a line feed, an escape, a DC1, a carriage return, a bell, a DC3 and a ▒. If the character being tested corresponds to one of these characters, a branch is taken to a routine which is set up to handle the special character which was encountered. If none of the above characters was encountered, then a branch is taken to the line labelled Nctrl (no control). This routine first checks to see if an escape sequence is active. If an escape sequence is active, then the character is added to the sequence being decoded. If not, then the character is appended to the string Disp$, which keeps track of incoming characters.

Disp$ is printed and cleared whenever 1) a line feed comes in, or 2) when Disp$ reaches a length of 80 characters. The Print routine prints the string to the CRT always, as well as to the hardcopy printer and/or the mass storage file if either or both of these modes is currently active.

Carriage returns are ignored, while a DC1 (or whatever prompt character the user picks -- see the section entitled Modifications) causes a branch to the line labelled Cret. This routine will then set aside the current contents of Disp$ to be a prompt string which will subsequently appear in the keyboard entry line. An escape character will cause a flag to be set (Escseq) which indicates that subsequent characters should be routed to the escape sequence routine Eseq.

The Unit Specifier (ASCII 31) causes the terminal to enter graphics-alpha mode while the Group Specifier (ASCII 29) causes it to enter graphics mode.

The bell causes the terminal to beep.

The DC3 is used to indicate that Auto-send should be temporarily halted. A DC1 will cause Auto-send to be reactivated.

The ▒ is not necessarily always a member of Select$. It is included or not included as a result of pressing UDK 5. If it is included in Select$, then all rubouts will be ignored. If it is not included, then all rubouts will be treated as any other character.

When the terminal emulator is in local mode, lines typed by the user are not sent to the computer, but instead are sent to the data comm service routine, which processes them as if they had come from the computer. Thus, the user can send escape sequences locally to perform such tasks as turning on the auto send mode, or redefining the data file name, and so on.

KMGTRM

```
10 Dim:     DIM Buffer$[164],Def$[300],Nam$[22],Prompt$[70],Erase$[3],
      Back$[20]
20          DIM Savebuffer$[160],Disp$[80],Key$[82],Com$[324],Name$[
               674],Hich$[16]
30          DIM Datafile$[10],Eol$[20],State$(0:1)[3],Select$[10],X$[1]
               ,Parm$[4]!***
40          DIM Saveprompt$[70],Comment$[70],Tell$[60],Prev$[164],
               Send$[160],Null$[1]
50          DIM Eswitch$[5],W$[4],Comstr$[164]
60          INTEGER Bitsperchar,Parity,Key,Oldascii,Statascii,K1,K2,K3,
               K4,B2
70          INTEGER B3,Ascii,Selectcode,Baudrate,Rmchr,I,Char,Localflag
80          INTEGER Retflag,Def_found,Rsize,Autosend,Sendflag,
               Character


Initialization routines

100 Begin:    Rub$=CHR$(127)
110           PLOTTER IS "GRAPHICS"
120           Parm$="dpcs"                  ! Used for parameterized
                 escapes
130           Eswitch$="EGNAQ"              ! Escape terminators
140           Hich$="@BACDFEGHJIKLNMO"      ! Highlighting codes
150           Erase$=CHR$(8)&" "&CHR$(8)
160           OVERLAP
170           Tell$="Please wait"
180           GOSUB Tell
190           RESTORE Initialconds
200           READ Selectcode,Baudrate,Stopbits,Parity,Bitsperchar,
                 Bitratefactor
210           READ State$(*),Datafile$,Filesize,Prompt,Clear
220           READ Hardprinter,Hpib,Rmchr,Enq$,Ack$
230           GOSUB Set_hard
240           Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
                 CHR$(13)&CHR$(7)&CHR$(19)                    !***ADDED:4=
                 ET,29=GS,31=US,7=BELL,19=DC3
250           !          gs      us          lf       escape          D1
                 cr      enq.       bell      D3
260           INTEGER Gra_al
270 Sfks:     READ B1                  ! Set up Name$ string
280           Name$=Name$&CHR$(B1)
290           IF B1=255 THEN Sfks
300           READ Name$[LEN(Name$)+1]
310           IF B1>-1 THEN Sfks
320           GOSUB Defsclear          ! Set up default key defs
330           PRINTER IS 16
340           PRINT LIN(20)
350           Buffer$=Disp$=""         ! Initialize variables
360           Handshake=Comp_mode=1
370           Gt$=CHR$(13)             ! Init. graphics terminator
380           Marg=1
```

KMGTRM

```
390            Marg2=512
400            Rsize=1000
410            Y=767
420            Cursor=Notstopped=Locked=Echo=1
430            Definemode=Printall=Escseq=Recording=Locked=Gra_mode=0
440            Gojump=Trace=Firstline=Local=Setup=Retflag=Autosend=
                  Not_draw=X=0

450 Initial:   IF NOT IOSTATUS(Selectcode) THEN Notoper
460            STATUS Selectcode;B1
470            IF BINAND(B1,48)<>16 THEN Notoper     ! --- 01- ---
480            WRITE IO Selectcode,5;1               ! 000 000 001 -> R5
490            WRITE IO Selectcode,4;64              ! 001 000 000 -> R4D
500            WRITE IO Selectcode,4;Stopbits*64+Parity*16+(Bitsperchar-
                  5)*4+Bitratefactor
510            WRITE IO Selectcode,4;39              ! 000 100 111 -> R4D
520            WRITE IO Selectcode,5;0               ! 000 000 000 -> R5
530            READ IO Selectcode,4;B1               ! cock i/f for
                  interrupts
540            WRITE IO Selectcode,7;0
550            WRITE IO Selectcode,5;132             ! 010 000 100
                  Input inter on
560            STATUS Selectcode;B1                  ! See if input
                  inter on
570            IF NOT BINAND(B1,128) THEN Notoper    ! Make sure it's on
580            IF Hardselect=16 THEN Topen
590            GOSUB Prtcheck
600            IF Printerokay THEN Topen
610            BEEP
620            DISP " Hardcopy printer not operational "
630            WAIT 500
640            GOTO Sp

650 Topen:     TOPEN Selectcode,2 GOSUB Datacomservice
660            Tell$="Terminal ready on "&VAL$(Selectcode)
670            GOSUB Tell1
680            ON KBD 10 GOSUB Keyservice
690            TDISP CHR$(12)&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-Cursor+
                  1)
700            Local=0
```

Background loop

```
720 Wait:      IF Gojump THEN Leap      ! Background loop
730            OVERLAP
740            IF Setup THEN Reinit
750            IF Define THEN Fil_def
760            IF Sendflag THEN GOSUB Send
770            IF Local THEN Loop
780            Clear=Clear-1
790            IF Clear<0 THEN DISP
```

```
800            IF Clear<0 THEN Clear=1E99
810            IF Notstopped THEN Wait
820            TYPEWRITER OFF
830            FOR B1=1 TO 10
840               WAIT 100
850            NEXT B1
860            TCLOSE
870            OFF KBD
880            Tell$="Terminal off"
890            GOSUB Tell
900            STOP

910 Resume:    Tell$="Background program surrendered"
920            GOSUB Tell
930            Gojump=Local=Locked=0
940            ON KBD 10 GOSUB Keyservice
950            GOTO Wait

960 Leap:      Tell$="Background program running"
970            GOSUB Tell
980            GOTO End

Local mode loop

1000 Loop:     IF Keyrec OR Keysend OR Keycomp THEN Pre
1010           Disp$=""
1020 Loop1:    IF NOT Local OR Autosend THEN Wait
1030           IF Old THEN Loop1
1040 Pre:      K1=LEN(Prev$)
1050           IF NOT K1 THEN Endloop
1060           IF NUM(Prev$[K1])<>Rmchr THEN Com$=Prev$&CHR$(13)&CHR$(
                  10)
1070           IF NUM(Prev$[K1])=Rmchr THEN Com$=Prev$[1;K1-1]&CHR$(13)&
                  CHR$(10)&CHR$(Rmchr)
1080           Prev$=""
1090           GOSUB Com0
1100           Character=Prompt
1110           IF Autosend THEN GOSUB Send
1120 Endloop:Old=1
1130           Localflag=0
1140           GOTO Loop

1150 Reinit:   OFF KBD
1160           Local=1
1170 Sc:       INPUT "Enter select code of I/O card:     [Use CONT]",
        Selectcode
1180           IF (Selectcode<1) OR (Selectcode>15) THEN Sc
1190 Baud:     INPUT "What is the baud rate?             [Use CONT]",
        Baudrate
1200           RESTORE Baudrates
1210           FOR B1=1 TO 10
1220             READ B2
```

KMGTRM

```
1230            IF B2=Baudrate THEN Brf
1240          NEXT B1
1250          GOTO Baud
1260 Brf:     LINPUT "Bit rate factor (1, 1/16, 1/64): [Use CONT]",
        Savebuffer$
1270          IF Savebuffer$="" THEN Savebuffer$="1/64"
1280          Bitratefactor=0
1290          IF Savebuffer$="1/16" THEN Bitratefactor=2
1300          IF Savebuffer$="1" THEN Bitratefactor=1
1310          IF Savebuffer$="1/64" THEN Bitratefactor=3
1320          IF NOT Bitratefactor THEN Brf
1330          IF (Bitratefactor<>3) OR (Baudrate<4800) THEN Bpc
1340          BEEP
1350          DISP "Warning: 1/64 bit rate factor not recommended
                at this baud rate!"
1360          WAIT 2000
1370 Bpc:     INPUT "Number of bits per character:      [Use CONT]",
        Bitsperchar
1380          IF (Bitsperchar<5) OR (Bitsperchar>8) THEN Bpc
1390          Parity=0
1400 Par2:    LINPUT "Is parity enabled?                 [Use CONT]",
        Savebuffer$
1410          IF Savebuffer$="" THEN Savebuffer$="N"
1420          Savebuffer$=UPC$(Savebuffer$[1,1])
1430          IF Savebuffer$="Y" THEN Parity=Parity+1
1440          IF (Savebuffer$<>"N") AND (Savebuffer$<>"Y") THEN Par2
1450          IF Savebuffer$="N" THEN Sb
1460 Par:     LINPUT "Is parity even?                    [Use CONT]",
        Savebuffer$
1470          IF Savebuffer$="" THEN Savebuffer$="Y"
1480          Savebuffer$=UPC$(Savebuffer$[1,1])
1490          IF Savebuffer$="Y" THEN Parity=Parity+2
1500          IF (Savebuffer$<>"N") AND (Savebuffer$<>"Y") THEN Par
1510 Sb:      B1=1
1520          INPUT "Enter number stop bits (1,1.5,2): [Use CONT]",B1
1530          IF (B1<>1) AND (B1<>1.5) AND (B1<>2) THEN Sb
1540          Stopbits=B1*2-1
1550 Alt_plot:Plottercode=13
1560          INPUT "Enter alternate plotter discription  (such as
                9872A or GRAPHICS):[Use CONT]",Plotter$
1570          IF (TRIM$(Plotter$)="GRAPHICS") OR (TRIM$(Plotter$)="")
                THEN GOTO 1670
1580          LINPUT "Enter plotter select code (and HPIB address, if
                applicable):   [Use CONT]",Savebuffer$
1590          IF Savebuffer$="" THEN Savebuffer$="13"
1600          Phpibflag=POS(Savebuffer$,",")
1610          IF NOT Phpibflag THEN Nophb
1620          Plottercode=VAL(Savebuffer$[1,Phpibflag-1])
1630          Phpib=VAL(Savebuffer$[Phpibflag+1])
1640          GOTO 1670
1650 Nophb:   Plottercode=VAL(Savebuffer$)
1660          Phpib=999
```

```
1670          Gt=1
1680 Graterm:INPUT "Graph. terminator: Enter 1 for [cr] 2 for [cr]
        [eot] 3 for none     [CONT]",Gt
1690          IF (Gt=1) OR (Gt=2) OR (Gt=3) THEN Gtset
1700          BEEP
1710          GOTO Graterm
1720 Gtset:  IF Gt=1 THEN Gt$=CHR$(13)
1730         IF Gt=2 THEN Gt$=CHR$(13)&CHR$(4)
1740         IF Gt=3 THEN Gt$=""
1750 Hand:Savebuffer$=""
1760       LINPUT "Is handshake for autosend enabled?                [
             Use CONT]",Savebuffer$
1770       IF Savebuffer$="" THEN Handon
1780       IF UPC$(Savebuffer$[1,1])="Y" THEN Handon
1790       IF UPC$(Savebuffer$[1,1])="N" THEN Handoff
1800       BEEP
1810       GOTO 1760
1820 Handon:Handshake=1
1830         GOTO Sp
1840 Handoff:Handshake=0
1850          Xon=1

1860 Sp:      LINPUT "Enter printer select code (and HPIB adrress, if
        applicable)",Savebuffer$
1870          IF Savebuffer$="" THEN Savebuffer$="0"
1880          Hpibflag=POS(Savebuffer$,",")
1890          IF NOT Hpibflag THEN Nohpib
1900          Hardprinter=VAL(Savebuffer$[1,Hpibflag-1])
1910          Hpib=VAL(Savebuffer$[Hpibflag+1])
1920          GOTO Reset
1930 Nohpib:  Hardprinter=VAL(Savebuffer$)
1940          Hpib=999
1950 Reset:   Setup=0
1960          GOSUB Set_hard
1970          IF Hardselect=16 THEN Initial
1980          GOSUB Prtcheck
1990          IF Printerokay THEN Initial
2000          GOTO Sp
2010 Prtcheck:Printerokay=1
2020          IF Hpibflag THEN Chhpib
2030          IF IOSTATUS(Hardselect) THEN RETURN
2040          GOTO Buggyprinter
2050 Chhpib:  SET TIMEOUT Hardprinter;2000
2060          ON INT #Hardprinter GOTO Buggyprinter1
2070          STATUS Hardprinter;A
2080          IF NOT A THEN Buggyprinter1
2100          GOSUB Shutoff
2110          RETURN
2120 Shutoff:OFF INT #Hardprinter
2130          SET TIMEOUT Hardprinter;0
2140          RETURN
2150 Buggyprinter1: GOSUB Shutoff
```

KMGTRM

```
2160 Buggyprinter: !
2170          BEEP
2180          PRINT "PRINTER NOT OPERATIONAL ON ";Hardselect;""
2190          Printerokay=0
2200          RETURN

2210 Notoper:Selectcode=0
2220          FOR B2=1 TO 12
2230            W1=PI
2240            STATUS B2;B3,W1
2250            IF W1<>PI THEN Notop1
2260            IF BINAND(48,B3)<>16 THEN Notop1
2270            IF Selectcode THEN Notop2
2280            Selectcode=B2
2290 Notop1: NEXT B2
2300          IF Selectcode THEN Initial
2310          BEEP
2320          DISP "There are no 98036 cards present.  Please insert
                one."
2330          WAIT 2000
2340          GOTO Notoper
2350 Notop2: BEEP
2360          OFF KBD
2370          INPUT "What is the select code?        [Use CONT]",
                Selectcode
2380          PRINT PAGE
2390          GOTO Initial

2400 Tell:    Clear=1000
2410 Tell1:   DISP TAB(74-LEN(Tell$));Rub$&" "&Tell$&" "&Rub$
2420          RETURN

2430 Send:    IF Autosend THEN K1=TYP(2)
2440          IF (K1=2) OR (K1>=8) AND (K1<=10) THEN Sendok
2450          BEEP
2460          Autosend=Sendflag=0
2470          ASSIGN #2 TO *
2480          Tell$="Autosend off -- "
2490          IF K1=3 THEN Tell$=Tell$&"End of file"
2500          IF K1<>3 THEN Tell$=Tell$&"Non-string data found"
2510          GOTO Tell
2520 Sendok: IF Autosend THEN READ #2;Send$          ! Send a line
        from the data file
2530          IF NOT Local THEN 2560
2540            Com$=Send$
2550            GOTO Com0
2560          Sendflag=0
2570          IF Local THEN Send1
2580          SET TIMEOUT Selectcode;2000
2590          SERIAL
2600          ON ERROR GOTO Send2
2610          EOL Selectcode;Eol$
```

```
2620          IF Autosend THEN OUTPUT Selectcode USING "#,K,L";Send$
2630          OFF ERROR
2640 Send1:   Old=0
2650          IF Autosend AND NOT Handshake AND Xon AND NOT Keysend OR
                 Local THEN Send
2660          RETURN
2670 Send2:   Tell$="Card not operational"
2680          OFF ERROR
2690          BEEP
2700          GOSUB Tell
2710          RETURN


Keyboard interrupt service routine

2730 Keyservice:
2740          Key$=KBD$        ! Dump the keyboard buffer
2750 Init:    Recursionlevel=0

2760 Break:   IF NOT POS(Key$,CHR$(255)&CHR$(7)) THEN Albrk
2770          WRITE IO Selectcode,5;1     ! Break
2780          WRITE IO Selectcode,4;46
2790          WRITE IO Selectcode,5;132
2800          BEEP
2810          WAIT 100
2820          BEEP
2830          WAIT 100
2840          WRITE IO Selectcode,5;1
2850          WRITE IO Selectcode,4;39
2860          WRITE IO Selectcode,5;132
2870          BEEP
2880          K1=POS(Key$,CHR$(255)&CHR$(7))
2890          Key$[K1]=Key$[K1+2]      ! Strip off key codes and loop
                 back
2900          GOTO Break

2910 Albrk:   IF NOT POS(Key$,CHR$(255)&CHR$(7+64)) OR Definemode THEN
        Ack
2920          BEEP                     ! Alternate break
2930          Key=64+7
2940          GOSUB Checkdef
2950          IF Def_found THEN OUTPUT Selectcode;Def$[K2,K1];
2960          IF NOT Def_found THEN OUTPUT Selectcode;CHR$(31);
2970          WAIT 150
2980          BEEP
2990          K1=POS(Key$,CHR$(255)&CHR$(7+64))
3000          Key$[K1]=Key$[K1+2]              ! Strip off character and
                 loop back.
3010          GOTO Albrk

3020 Ack:     IF NOT POS(Key$,CHR$(255)&CHR$(70)) OR Definemode THEN
```

KMGTRM

```
        Recur
3030          Key=70
3040          GOSUB Checkdef
3050          IF Def_found THEN OUTPUT Selectcode;Def$[K2,K1];
3060          IF NOT Def_found THEN OUTPUT Selectcode;CHR$(6);
3070          BEEP
3080          WAIT 100
3090          K1=POS(Key$,CHR$(255)&CHR$(70))
3100          Key$[K1]=Key$[K1+2]        ! Strip off character and loop
                 back.
3110          BEEP
3120          GOTO Ack

3130 Checkdef: Def_found=0               ! This routine checks to see if
3140          Retflag=1                  !  Alt Break or Ack have been
3150          GOSUB User2                !  redefined.
3160          Retflag=0                  ! If they have, K2 and K1 mark
                 the
3170          IF K2<>3 THEN Def_found=1  !  beginning and end in Def$
                 of the
3180          RETURN                     !  new defintion.


3190 Recur:   Recursionlevel=Recursionlevel+1
3200          IF Recursionlevel>160 THEN Abort

3210 More:    IF LEN(Key$) THEN More1    ! Check for null string.
3220          Localflag=0
3230          RETURN
3240 More1:   Ascii=1
3250          Key=NUM(Key$)              ! Strip off first character.
3260          IF Key<>127 THEN 3320      ! Allow for non-ascii key codes
3270          Key$=Key$[2]               !  inserted from user re-
                 definitions
3280          Key=NUM(Key$)              !  (which are flagged with 127
                 since
3290                                     !  255 is a delimiter in key
                                            defs.)
3300          IF Key<>127 THEN 3360      ! Rubouts are 127 127.
3310          GOTO Ascii                 ! 127 never produced in
                 definemode
3320          IF Key<255 THEN 3380       ! Check for non-ascii key code
3330          Key$=Key$[2]
3340          Key=NUM(Key$)              ! if Non-ASCII, the next code
                 tells
3350                                     !  which key was pressed.
3360          Ascii=0
3361          IF (Key=1) OR (Key=65) THEN Lfkey=1
3370          IF (Key>=50) AND (Key<=53) OR (Key>=114) AND (Key<=117)
                 OR (Key>=178) AND (Key<=181) OR (Key>=242) AND (Key<=
                 245) THEN Key=Key-10
3380          IF Definemode THEN Dmode
```

```
3390          IF Ascii THEN Asciiudk     ! Branch to 8-bit character
                 section

3400 Doublechar: ! This section handles all keys returning two
        character codes
3410          ! (i.e. all non-ascii keys).
3420 Ktest:  IF (Key=20) OR (Key=21) OR (Key=84) OR (Key=85) THEN
        Store
3430                                          ! 20 is STORE, and 21 is
                                                 EXECUTE
3440          IF Key=15+64 THEN Stop
3450          IF (Key=9) AND NOT Gra_mode THEN Tset    ! UDK 9  is
                 Define Key
3460          IF Key=40 THEN Back                      ! 40 is backspace
3470          ! Check to see if key has been user-defined (keys tested
                 above this
3480          !  point can not be redefined).
3490          IF POS(Def$,CHR$(255)&CHR$(127)&CHR$(Key)) AND NOT
                 Gra_mode THEN User2

3500  ! The rest of the program-defined keys (which can be overridden
        by
3510  ! the Define Key operation) are:
3520  ! UDK 0 is trace on/off toggle
3530  ! UDK 2 is echo on/off toggle
3540  ! UDK 3 is hardcopy on/off toggle
3550  ! UDK 4 is Local                Shift UDK 4 is Remote
3560  ! UDK 5 is rubouts ignored/not ignored toggle
3570  !                               Shift UDK 5 is DEL
3580  ! UDK 6 is escape character    Shift UDK 6 is Acknowledge
3590  ! UDK 7 is Break                Shift UDK 7 is alternate break
3600  ! UDK 8 is status               Shift UDK 8 is USART status
3610  !                               Shift UDK 9 is clear key
        definitions
3620  ! UDK 10 is setup
3630  ! UDK 11 is record file on/off Shift UDK 11 is define file
3640  ! UDK 12 is send file
3650  ! UDK 13 is compatability mode on/off toggle
3660  ! UDK 14 is graphics/alpha toggle
3670  !                               Shift UDK 14 is clear graphics
3680  ! UDK 15 is dump graphics
3690  ! Left arrow moves cursor left Shift left arrow moves prompt
        left
3700  ! Rightarrow moves cursor right Shift rightarrow moves prompt
        right
3710  ! Home and Shift Home put cursor at left of input line
3720  ! Clear and Shift Clear will clear the screen and input line
3730  ! Clrem and Shift Clrem will clear the input line to the right
        of
3740  !  the cursor
3750  ! Delete Char and Shift Delete Char will delete a character
3760  ! Insert Char and Shift Insert Char toggle the insert mode
```

KMGTRM

```
3770  ! Recall will bring back old commands (up to 1000 characters)
3780  ! Shift Recall works in the reverse of Recall
3790  ! Tab and Shift Tab move the cursor in increments of 8 spaces
3800  ! Backspace shifts the cursor left and clears to the end of the
3810  !  line
3820  ! Clear Line will clear the input line
3830          ON ERROR GOSUB Nada
3840 Test0: ON Key+1 GOTO Trace0,Bad,Echo0,Hard0,Local,Rubs0,Bad,Bad,
        Status,Bad,Setup,Rec_file,Send_file,Comp_mode,Ex_gra,Dump_gra
3850 Test1:  ON Key-63 GOTO Bad,Bad,Bad,Bad,Remote,Bad
3860 Test2:  ON Key-71 GOTO Dsr,Kclr,Bad,Deffil,Bad,Bad,Cgraph
3870 Test3:  ON Key-21 GOTO Left,Right,Gup,Gdown,Bad,Bad,Bad,Clear
3880 Test4:  ON Key-85 GOTO Shleft,Shright,Sgup,Sgdown,Bad,Bad,Bad,
        Clear
3881          IF Key=19 THEN GOTO Noprmpt
3890          OFF ERROR
3900          GOTO Bad


3910 Comp_mode: IF Gra_mode THEN Bad     ! turn compatability mode on
3920            IF Comp_mode THEN Compoff
3930            Prev$=CHR$(27)&"&s1p0Q"
3940            Local=Keycomp=1
3950            Escseq=Old=0
3970            RETURN
3980 Compoff: Prev$=CHR$(27)&"&s0p0Q"     ! turn compat. mode off
3990          Local=Keycomp=1
4000          Old=0
4010          RETURN

4020 Ex_gra: IF Gra_mode=0 THEN Graphic   ! graphics off
4030         Gra_mode=Gra_al=0
4040         ON KBD 10 GOSUB Keyservice
4050         EXIT GRAPHICS
4060         Tell$="Exit Graphics"
4070         GOSUB Tell
4080         GOTO Next

4090 Graphic:IF NOT Comp_mode THEN Bad
4100         IF NOT Altplot THEN 4140     ! Graphics on
4110         IF Altplot AND (Phpib<>999) THEN PLOTTER IS Plottercode,
                Phpib,Plotter$
4120         IF Altplot AND (Phpib=999) THEN PLOTTER IS Plottercode,
                Plotter$
4130         GOTO 4150
4140         GRAPHICS
4150         Gra_mode=1
4160         IF NOT Gra_al THEN ON KBD 10 GOSUB Keyservice ,ALL
4170         Tell$="Graphics"
4180         GOSUB Tell
4190         GOTO Next
```

```
4200 Cgraph: IF NOT Comp_mode THEN Bad
4210         GCLEAR      ! clear graphics
4220         Gra_al=Marg=1
4230         X=0
4240         Y=767
4250         GOTO Next

4260 Dump_gra:  IF NOT Gra_mode THEN Bad    ! dump graphics
4270         OUTPUT 0;CHR$(27)&"Z"&CHR$(8)
4280         DUMP GRAPHICS
4290         IF Trace THEN OUTPUT 0;CHR$(27)&"Y"
4300         GOTO Next

4310 Nada:   RETURN
4320 Shleft: IF Gra_mode THEN Sl_left
4330         GOTO Bad
4340 Shright:IF Gra_mode THEN Sl_right
4350         GOTO Bad
4360 Store:  IF Locked THEN Holng
4370         IF Gra_mode THEN POINTER -1,-1,2
4380         Prompt$=""
4390 Store1:!
4400        IF NOT Recording THEN 4430
4420           GOSUB Record
4430         Old=0
4440         IF Local THEN Prev$=Buffer$
4450         IF Local THEN Cline
4460         SET TIMEOUT Selectcode;2000    ! Timeout condition if
                card hangs.
4470         SERIAL
4480         ON ERROR GOTO Store5
4490         EOL Selectcode;Eol$
4500         IF NOT Esc_sub THEN Outp_n
4510 Out_e_s:IF X$<>"" THEN Outp_co                          !***WW
4520         POINTER X,Y,1
4530         RETURN

4540 Outp_co:OUTPUT Selectcode USING "#,K";X$&W$&Gt$;          ! send
        coord & msg.
        !***WW
4550         POINTER -1,-1,2
4551         X$=""                                           !***WW
4560         Esc_sub=0
4570         GOTO Outp_s
4580 Outp_n:  OUTPUT Selectcode USING "#,K,L";Null$
4590 Outp_s: OFF ERROR
4600         GOTO Cline
4610 Store5: Tell$="Card not operational"
4620         OFF ERROR
4630         GOTO Telbeep
4640 Fmt:    IMAGE #,L
4650 Fmt1:   IMAGE #,A
```

KMGTRM

```
4660 Holng:   Tell$="Locked out"
4670          GOTO Telbeep
4680 Noprmpt:Tell$="No prompt - use EXECUTE"
4690 Telbeep:BEEP
4700          GOTO Nextell

4710 Stop:                                ! STOP
4720          IF Locked THEN Holng
4730          Notstopped=Local=0          !  to the mainframe before
                 turning
4740          K1=POS(Def$,CHR$(255)&CHR$(127)&CHR$(64+15))+3 ! off the
                 program
4750          IF K1=3 THEN RETURN         !  and closing the datacomm
                 channel.
4760          K2=POS(Def$[K1],CHR$(255))+K1-2
4770          IF K2<K1-1 THEN K2=LEN(Def$)
4780          Key$=Def$[K1,K2]
4790          GOTO More

4800 User1:   ! Process keys that have been redefined.
4810          K2=POS(Def$,CHR$(255)&CHR$(Key))+2 ! Find start of
                 definition
4820          K1=POS(Def$[K2],CHR$(255))+K2-2     ! Find end of
                 definition
4830          IF K1<K2-1 THEN K1=LEN(Def$)        ! Allow for last def.
                 in string
4840          IF LEN(Key$)+K1-K2>80 THEN Abort    ! Check for string
                 too long
4850          Key$=Def$[K2,K1]&Key$[2]            ! Append definition
4860          GOTO Recur

4870 User2:   K2=POS(Def$,CHR$(255)&CHR$(127)&CHR$(Key))+3
4880          K1=POS(Def$[K2],CHR$(255))+K2-2
4890          IF K1<K2-1 THEN K1=LEN(Def$)
4900          IF Retflag THEN RETURN
4910          IF LEN(Key$)+K1-K2>80 THEN Abort
4920          Key$=Def$[K2,K1]&Key$[2]
4930          GOTO Recur
4940 Abort:   Tell$="Recursion level too great - infinite loop likely"
4950          Key$=""
4960 Nextell:GOSUB Tell
4970          GOTO Strip
4980 Back:    Cursor=MAX(1,Cursor-1)      ! This section erases the
        previous
4990          TDISP Erase$[1,3]           ! character in the entry line,
                 and
5000          IF NOT Gra_al THEN 5050
5010             X=MAX((Marg-1)*Marg2,X-14)
5020             X1=X*(560/1024)
5030             GPRINT X1,Y1," "
5040             POINTER X+7,Y+12,2
5050          X$=Back$[1,1]               ! sends Back$ to the computer
```

```
5060          Buffer$[Cursor]=""
5070          IF Local THEN Strip
5080          GOTO Savechar
5090 Clear:   PRINT PAGE,LIN(-19)
5100          DISP
5110 Cline:   Buffer$=""                    ! Clear line
5120          Cursor=1
5130          TDISP CHR$(12)
5140          GOTO Strip
5150 Fa_left:X=X-14                 ! fast left
5160 Left1:   IF X<0 THEN X=0
5170          GOTO Esc_sub
5180 Sl_left:IF Gra_al THEN Bad  ! slow left
5190          X=X-1
5200          GOTO Left1
5210 Left:    IF Gra_mode AND NOT Gra_al THEN GOTO Fa_left
5220          GOTO Bad
5230 Fa_right:X=X+14                ! fast right
5240 Right1:  IF X>1023 THEN X=1023
5250          GOTO Esc_sub
5260 Sl_right:IF Gra_al THEN Bad ! slow right
5270           X=X+1
5280          GOTO Right1
5290 Right:   IF Gra_mode AND NOT Gra_al THEN Fa_right
5300          GOTO Bad
5310 Sgup:    IF NOT Gra_mode OR Gra_al THEN Bad  ! slow up
5320          Y=Y+1
5330          GOTO Gu1
5340 Gup:     IF NOT Gra_mode OR Gra_al THEN Bad  ! fast up
5350          Y=Y+10
5360 Gu1:     IF Y>780 THEN Y=780
5370          GOTO Esc_sub
5380 Gdown:   IF NOT Gra_mode OR Gra_al THEN Bad  ! fast down
5390          Y=Y-10
5400 Gd1:     IF Y<1 THEN Y=1
5410          GOTO Esc_sub
5420 Sgdown:  IF NOT Gra_mode OR Gra_al THEN Bad  ! slow down
5430          Y=Y-1
5440          GOTO Gd1
5450 Tset:    IF Gra_mode THEN Bad
5460          Savecursor=Cursor          ! Define key
5470          IF LEN(Def$)>297 THEN Derr
5480          Saveprompt$=Prompt$
5490          Savebuffer$=Buffer$
5500          Definemode=Cursor=1
5510          Buffer$=Prompt$=""
5520          Tell$="Enter character to define"
5530          GOSUB Tell
5540          Clear=1E99
5550          GOTO Next

5560 Kclr:    IF Gra_mode THEN Bad
```

KMGTRM

```
5570          GOSUB Defsclear            ! Clear keys
5580          Tell$="Definitions cleared"
5590          GOSUB Tell
5600          GOTO Next

5610 Hard0:   IF Printall=0 THEN Hard1
5620          Printall=0                 ! Hardcopy off
5630          GOTO Hard2
5640 Hard1:   Printall=1                 ! Hardcopy on
5650          Trace=0
5660 Hard2:   Tell$="Hardcopy "&State$(Printall)
5670          GOTO Nextell

5680 Echo0:   IF Echo=0 THEN Echo1       ! Echo on/off
5690          Echo=0                     ! Echo    off
5700          GOTO Echo2
5710 Echo1:   Echo=1
5720 Echo2:   Tell$="Echo "&State$(Echo)
5730          GOTO Nextell

5740 Rubs1:   Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
        CHR$(13)&CHR$(7)&CHR$(19)
5750          Tell$="Rubouts not ignored"
5760          GOTO Nextell
5770 Rubs0:   !
5780          IF Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(
                Prompt)&CHR$(13)&CHR$(7)&CHR$(19)&CHR$(127) THEN Rubs1
5790          Select$=CHR$(29)&CHR$(31)&CHR$(10)&CHR$(27)&CHR$(Prompt)&
                CHR$(13)&CHR$(7)&CHR$(19)&CHR$(127)
5800          Tell$="Rubouts ignored"
5810          GOTO Nextell

5820 Trace0:  IF Trace=0 THEN Trace1     ! Trace on/off
5830          Trace=0
5840          GOTO Trace2
5850 Trace1:  IF Hardselect<>16 THEN Traceon
5860 Traceig: Tell$="Command ignored -- Hardcopy printer required"
5870          GOTO Nextell
5880 Traceon: Trace=1
5890 Trace2:  Tell$="Display functions "&State$(Trace)
5900          OUTPUT Hardselect;CHR$(27);CHR$(90-Trace)
5910          Printall=0
5920          GOTO Nextell

5930 Setup:   IF Gra_mode THEN Bad
5940          Setup=1
5950          IF Define THEN Setup=0
5960          IF NOT Local THEN Stri
5970          BEEP
5980          DISP Rub$&" Local mode exited "&Rub$
5990          WAIT 2500
6000          GOTO Remote
```

```
6010 Status: IF Printall THEN GOSUB Setprinter
6020         Statascii=Ascii
6030         Ascii=1
6040         PRINT RPT$("-",26);"Status of terminal emulator";RPT$("-
                ",27)
6050         PRINT "Echo ";State$(Echo),"Hardcopy ";State$(Printall),
6060         IF POS(Select$,CHR$(127)) THEN PRINT "Rubouts ignored";
                TAB(62);"Display funcs. ";State$(Trace)
6061         IF NOT POS(Select$,CHR$(127)) THEN PRINT "Rubouts not
                ignored";TAB(62);"Display funcs. ";State$(Trace)
6070         PRINT "Recording ";State$(Recording),
6080         PRINT "Local ";State$(Local),"Cursor at";
6090         PRINT Cursor;TAB(62);"Background ";State$(Gojump)
6100         IF Character=17 THEN PRINT "P";
6110         IF Character<>17 THEN PRINT "No p";
6120         PRINT "rompt";TAB(21);"Keyboard lock ";State$(Locked);
6130         PRINT TAB(41);"Hard. printer at "&VAL$(Hardselect)
6140         PRINT "Autosend ";State$(Autosend);TAB(21);
6150         PRINT "Baud rate";Baudrate;TAB(40);Stopbits/2+.5;"stop
                bit";
6160         PRINT RPT$("s",Stopbits<>1);TAB(62);"Compatability ";
6170         PRINT State$(Comp_mode);LIN(1);VAL$(Bitsperchar);
6180         PRINT " bits per character";TAB(41);"I/O card at";
                Selectcode
6190         PRINT "Parity ";
6200         IF (Parity=0) OR (Parity=1) THEN PRINT "(odd) ";
6210         IF (Parity=2) OR (Parity=3) THEN PRINT "(even) ";
6220         IF (Parity=1) OR (Parity=3) THEN PRINT "enabled";
6230         IF (Parity=0) OR (Parity=2) THEN PRINT "disabled";
6240         PRINT TAB(41);"Bit rate factor is 1";
6250         IF Bitratefactor=2 THEN PRINT "/16";
6260         IF Bitratefactor=3 THEN PRINT "/64";
6270         PRINT LIN(1);"Handshake "&State$(Handshake);
6280         PRINT TAB(41);"Graphics input term.: ";
6290         IF Gt$=CHR$(13) THEN PRINT "[crl";
6300         IF Gt$=CHR$(13)&CHR$(4) THEN PRINT "[crl[eotl";
6310         PRINT LIN(1);"File name: "&Datafile$;TAB(41);"Default
                file size:";Filesize
6320         IF (Plottercode=0) OR (Plottercode=13) THEN 6360
6330         IF Phpib=999 THEN PRINT "Alternate plotter: ";Plotter$;"
                at";Plottercode;"    ";State$(Altplot)
6340         IF Phpib<>999 THEN PRINT "Alternate plotter: ";Plotter$;
                " at";Plottercode;",";Phpib;"     ";State$(Altplot)
6350         GOTO 6370
6360         PRINT "No alternate plotter specified"
6370         PRINT "Current line:"
6380         K2=0
6390         Ktab=99
6400 Stat5:  K2=K2+1
6410         IF K2>LEN(Buffer$) THEN Stat6
6420         Key=NUM(Buffer$[K2])
6430         GOSUB Knam
```

KMGTRM

```
6440          IF LEN(Nam$)+Ktab<79 THEN Stat7
6450          PRINT LIN(1);TAB(20);
6460          Ktab=20
6470 Stat7:   PRINT Nam$&" ";
6480          Ktab=Ktab+LEN(Nam$)+1
6490          GOTO Stat5
6500 Stat6:   PRINT LIN(1);"Key definitions:";
6510          K2=0
6520 Stat1:   K2=K2+1
6530          K$=""
6540          IF K2>LEN(Def$) THEN Statusart
6550          IF Def$[K2;1]<>CHR$(255) THEN Stat3
6560          Ktab=0
6570          K$=":"
6580          K2=K2+1
6590          PRINT LIN(1);TAB(5);
6600 Stat3:   Key=NUM(Def$[K2])
6610          Ascii=1
6620          IF Key<>127 THEN 6670
6630          K2=K2+1
6640          Key=NUM(Def$[K2])
6650          IF Key=127 THEN 6670
6660          Ascii=0
6670          GOSUB Knam
6680          Ktab=Ktab+LEN(Nam$&K$)+1
6690          IF Ktab<76 THEN Stat4
6700          PRINT LIN(1);TAB(20);
6710          Ktab=21+LEN(Nam$&K$)
6720 Stat4:   PRINT Nam$&K$&" ";
6730          IF NOT LEN(K$) THEN Stat1
6740          PRINT TAB(20);
6750          Ktab=20
6760          GOTO Stat1
6770 Dsr:     IF Printall THEN GOSUB Setprinter
6780          PRINT RPT$("-",80);
6790          Retflag=1
6800          GOSUB Statusart
6810          Retflag=0
6820          Tell$="Data set ready"
6830          IF NOT BINAND(K1,128) THEN Tell$="Data set not ready"
6840          GOTO Nextell

6850 Statusart: WRITE IO Selectcode,5;1
6860          READ IO Selectcode,4;K1    ! Read status word
6870          WRITE IO Selectcode,4;55   ! Reset error bits (if any)
6880          WRITE IO Selectcode,5;0
6890          READ IO Selectcode,4;K2
6900          WRITE IO Selectcode,7;0    ! Cock the card for interrupts
6910          WRITE IO Selectcode,5;132
6920          PRINT LIN(2),"USART status control word (R4E):",LIN(1)
```

```
6930          PRINT "Data set    Zero    Framing   Overrun   Parity
                 Transmtr   Receiver  Transmtr"
6940          PRINT " ready    (Unused)   error     error     error
                 empty     ready     ready"
6950          IMAGE #,3X,D,6X
6960          FOR K2=7 TO 0 STEP -1
6970             K3=BIT(K1,K2)
6980             PRINT USING 6950;K3
6990          NEXT K2
7000          PRINT
7010 Stat2:   PRINT LIN(1),RPT$("-",80)
7020          PRINTER IS 16
7030          IF Retflag THEN RETURN
7040          Ascii=Statascii
7050          GOTO Strip

7060 Local:   !
7080          Local=Localflag=1
7090          Old=0
7100          Tell$="Local"
7110          GOTO Nextell
7120 Remote:  IF NOT Localflag THEN Remote1
7130          Prev$=Prev$&CHR$(Rmchr)
7140          GOTO Nextell
7150 Remote1:Local=0
7160          Tell$="Remote"
7170          GOTO Nextell

7180 Asciiudk: IF POS(Def$,CHR$(255)&CHR$(Key)) THEN User1 ! Check
        for User Def.
7190 Ascii:   IF Cursor<160-LEN(Prompt$) THEN Addon
7200 Bad:     BEEP
7210          GOTO Strip
7220 Addon:   X$=CHR$(Key)
7230          IF Echo THEN Dis_key
7240          Buffer$[Cursor,Cursor]=X$
7250          Cursor=Cursor+1
7260          IF NOT Gra_mode THEN Dis_key
7270          IF Esc_sub THEN Dis_key
7280          X1=X*(560/1024)
7290          Y1=Y*(455/788)
7300          IF NOT Altplot THEN 7350
7310             MOVE X,Y
7320             CSIZE 2
7330             LABEL CHR$(Key)
7340             GOTO 7360
7350          GPRINT X1,Y1,CHR$(Key)
7360          X=X+14
7370          IF X<1011 THEN Po1
7380          X=(Marg-1)*Marg2
7390          Y=Y-21
7400          IF Y<20 THEN Y=767                          !***WW
```

KMGTRM

```
7410 Poi:     POINTER X+7,Y+12,2
7420 Dis_key:   IF NOT Echo THEN TDISP CHR$(Key)
7430          IF Local THEN Strip
7440          IF Esc_sub THEN Out_e_s                    !***WW
7450 Savechar: IF NOT Echo THEN Disp$=Disp$&X$
7460          IF LEN(Disp$)<80 THEN Sendchar
7470          IF Recording THEN GOSUB Record
7480          GOSUB Print
7490 Sendchar: SET TIMEOUT Selectcode;2000
7500            SERIAL
7510            ON ERROR GOTO Store5
7520            OUTPUT Selectcode USING "#,A";X$
7530            OFF ERROR
7540            GOTO Strip
7550 Next:    TDISP CHR$(12)&Prompt$&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-
        Cursor+1)
7560 Strip:   Key$=Key$[2]    ! Next character from KBD$
7570          GOTO More
```

Character definition mode

```
7590 Dmode:   IF NOT Ascii AND (Key=9) AND (Cursor<>1) THEN Dleav
7600          IF LEN(Def$)<299 THEN Dent
7610          Tell$="Buffer full - press UDK 9"
7620          GOSUB Tell
7630          Clear=1E99
7640          BEEP
7650          GOTO Next
7660 Dent:    Tell$="Enter definition characters or UDK 9"
7670          GOSUB Tell
7680          Clear=1E99
7690          IF Cursor<>1 THEN Dmore
7700          IF NOT Ascii AND (Key=20) THEN Key=254 ! Store key
                redefines EOL
7710          GOSUB Knam
7720          IF Ascii THEN K1=POS(Def$,CHR$(255)&CHR$(Key))
7730          IF NOT Ascii THEN K1=POS(Def$,CHR$(255)&CHR$(127)&CHR$(
                Key))
7740          IF NOT K1 THEN Dfrst            ! Check for previous
                defintion
7750 D_let:   Def$[K1]=Def$[K1+1]             ! Weed out the old
        definition
7760          IF (Def$[K1;1]<>CHR$(255)) AND (K1<=LEN(Def$)) THEN D_let
7770 Dfrst:   Savelen=LEN(Def$)               ! Save old definition
        length
7780          Oldascii=Ascii
7790          Def$=Def$&CHR$(255)             ! Append new key
                definition ID
7800          IF NOT Ascii THEN Def$=Def$&CHR$(127)
7810          Def$=Def$&CHR$(Key)
7820          Buffer$=Nam$&": "
```

```
7830            GOTO Drest
7840 Dmore:     IF NOT Ascii THEN Def$=Def$&CHR$(127)
7850            Def$=Def$&CHR$(Key)              ! Append next part of
                   definition
7860            GOSUB Knam                       ! Find name of key
7870            IF LEN(Buffer$&Nam$)>158 THEN Buffer$=Buffer$[LEN(Nam$)+
                   3]
7880            Buffer$=Buffer$&Nam$&" "
7890 Drest:     Cursor=LEN(Buffer$)+1            ! Step cursor counter
7900            GOTO Next
7910 Derr:      Tell$="No more room for definitions"
7920            BEEP
7930 Dbyebye:GOSUB Tell
7940            GOTO Dbye
7950 Dleav:     IF Oldascii AND (Def$[Savelen+2;1]=Def$[Savelen+3]) THEN
        Dclr
7960            IF NOT Oldascii AND (Def$[Savelen+2;2]=Def$[Savelen+4])
                   THEN Dclr
7970            IF NOT Oldascii AND (NUM(Def$[Savelen+3])=254) THEN Eol$=
                   Def$[Savelen+4,MIN(300,Savelen+23)]
7971            DISP
7980            IF Lfkey AND (Eol$=CHR$(13)) THEN Tell$="Auto linefeed
                   off"
7990            IF Lfkey AND (Eol$=CHR$(13)&CHR$(10)) THEN Tell$="Auto
                   linefeed on"
8000            IF Lfkey THEN GOSUB Tell
8001            Lfkey=0
8010 Dbye:      Definemode=0        ! Exit define mode
8020            Prompt$=Saveprompt$
8030            Buffer$=Savebuffer$
8040            Cursor=Savecursor
8050            GOTO Next
8060 Dclr:      Def$=Def$[1,Savelen]
8070            Tell$=Nam$&" key cleared"
8080            GOTO Dbyebye
8090 Knam:      Nam$=CHR$(Key)      ! Check for case of name being char.
8100            IF Ascii AND (Key>32) AND (Key<254) THEN RETURN
8110            K3=1
8120            Shift=0
8130            IF Ascii THEN 8200
8140            Shift=Key DIV 64
8150            IF Key=254 THEN Shift=0
8160            K3=POS(Name$,CHR$(255))+1
8170                                ! 255 separates Ascii 0 through 32
8180                                !  from non-ascii keys.
8190            Nam$=CHR$(Key-Shift*64)
8200            K1=POS(Name$[K3],Nam$)
8210            Nam$=Name$[K1+K3,MIN(LEN(Name$),K1+K3+17)]
8220            FOR K1=1 TO LEN(Nam$)
8230              IF (Nam$[K1]>"~") OR (Nam$[K1]<",") THEN Dnam
8240            NEXT K1
8250 Dnam:      IF NOT Shift THEN Nam$="["&Nam$[1,K1-1]&"]"
```

KMGTRM

```
8260          IF NOT (Shift=1) THEN Nam$="[sh "&Nam$[1,K1-1]&"]"
8270          IF NOT (Shift=2) THEN Nam$="[cntl "&Nam$[1,K1-1]&"]"
8280          IF NOT (Shift=3) THEN Nam$="[sh cntl "&Nam$[1,K1-1]&"]"
8290          RETURN
```

Clear definitions of characters

```
8310 Defsclear:
8320          Eol$=CHR$(13)
8330          Back$=CHR$(8)
8340          Def$=""
8350          RESTORE Defs
8360 Read:    READ D1,D2,Savebuffer$
8370          IF D1=-1 THEN Clline
8380          Def$=Def$&CHR$(255)
8390          IF D1=127 THEN Def$=Def$&CHR$(D1)
8400          Def$=Def$&CHR$(D2)&Savebuffer$
8410          GOTO Read
8420 Clline:  D1=POS(Def$,"!")      ! Add 128 to codes following !.
8430          IF NOT D1 THEN Stores
8440          Def$[D1]=CHR$(NUM(Def$[D1+1])+128)&Def$[D1+2]
8450          GOTO Clline
8460 Stores:  D1=POS(Def$,"~")      ! Subtract 64 from codes after ~.
8470          IF NOT D1 THEN RETURN
8480          Def$[D1]=CHR$(NUM(Def$[D1+1])-64)&Def$[D1+2]
8490          GOTO Stores
```

Data comm link service routine

```
8510 Datacomservice:
8520          IF Local THEN Trashit
8530          Com$=TBUF$            ! Dump input buffer
8540          IF POS(Com$,Enq$) AND NOT Gra_mode THEN OUTPUT
                 Selectcode;Ack$;
8550 Com0:    FOR I=1 TO LEN(Com$)
8560          GOTO Com2
8570 Com:     NEXT I
8580          RETURN
8590 Com2:    Char=NUM(Com$[I])
8600          Character=BINAND(127,Char)
8610          Char$=CHR$(Character)
8620          IF Trace THEN OUTPUT Hardselect USING "#,A";Char$
8630          IF (Character=22) OR (Character=0) THEN Com
8631          IF (Character=5) AND NOT Gra_mode THEN Com!***WW
8635          IF POS(Select$,CHR$(127)) AND (Character=127) AND NOT
                 Gra_mode THEN Com
                 !***WW
8640          IF NOT Recording THEN Com1
```

```
8650          Comstr$=Comstr$&Char$
8660          IF (LEN(Comstr$)<80) AND (Char$<>CHR$(10)) AND (Char$<>
                 CHR$(Prompt)) THEN Com1
8675          PRINT #1;Comstr$
8680          Comstr$=""
8685 Com1:    ON 1+POS(Select$,Char$) GOTO Nctrl,Gs,Us,Lf,Esc,Cret,Cr,
        Bell,Dc3,Check_rubouts,Dcrmt,Com

        !***WW
8690 Check_rubouts:IF Gra_al THEN Com                    !***WW
8691 Nctrl:   IF Char$<>CHR$(Rmchr) THEN 8700
8692          Local=0
8693          Tell$="Remote"
8694          GOSUB Tell
8695          GOTO Com
8700          IF NOT Gra_mode OR Escseq THEN Nctrl1
8710          IF NOT Gra_al THEN Coor_rec
8720          IF Pointer THEN POINTER -1,-1,2
8730          Pointer=0
8740          IF Char$<CHR$(32) THEN Com
8750          X1=X*(560/1024)          ! calculate X,Y for GPRINT
8760          Y1=Y*(455/788)
8770          IF NOT Altplot THEN 8820
8780             CSIZE 2
8790             MOVE X,Y
8800             LABEL Char$           ! alpha for alt. plotter
8810             GOTO 8830
8820          IF X<1011 THEN GPRINT X1,Y1,Char$  ! plot alpha on CRT
8830          X=MAX(X+14,14)
8840          GOTO Com

8850 Coor_rec:IF NOT Comp_mode THEN Nctrl1
8860          V=NUM(Char$)                ! get coord.-char
8870          IF (W=1) AND (V<64) THEN Hy
8880          IF V>95 THEN Ly
8890          IF V<64 THEN Hx
8900 Lx:      W(4)=V          ! Low X, 4.BYTE
8910          W=1
8920          GOTO Draw
8930 Hy:      W=2             ! High Y, 1.BYTE
8940          W(1)=V
8950          GOTO Com
8960 Ly:      W=3             ! Low Y, 2.BYTE
8970          W(2)=V
8980          GOTO Com
8990 Hx:      W(3)=V          ! High X, 3.BYTE
9000          GOTO Com

9010 Esc_qu:                  ! ***WW
9020          Char$=CHR$(127)
9030          Escseq=0
9040          GOTO Nctrl
```

KMGTRM

```
9050 Nctrl1:  IF Escseq THEN Eseq
9060          IF LEN(Disp$)>79 THEN GOSUB Print
9070          Disp$[LEN(Disp$)+1]=Char$
9080          IF (Character>31) AND (Character<>8) THEN TDISP Char$
9090          GOTO Com


9100 Gs:     IF NOT Comp_mode THEN Nctrl1
9110          Not_draw=1        ! GS; switch on graphics/do not draw
9120          Gra_al=0
9130          Escseq=0
9140          W=1
9150          IF Gra_mode THEN Com
9160 Gs1:     IF NOT Comp_mode THEN Nctrl1
9170          Gra_mode=1
9180          IF Altplot AND (Phpib=999) THEN PLOTTER IS Plottercode,
                 Plotter$
9190          IF Altplot AND (Phpib<>999) THEN PLOTTER IS Plottercode,
                 Phpib,Plotter$
9200          IF NOT Altplot THEN GRAPHICS
9210          ON KBD 10 GOSUB Keyservice ,ALL
9220          SCALE 0,1024,0,788     ! scale graphics
9230          GCLEAR
9240          W(1)=55                ! initialize 4 bytes of coord.
9250          W(2)=96
9260          W(3)=32
9270          W(4)=64
9280          X=0
9290          Marg=1
9300          Y=767
9310          GOTO Com

9320 Cr:      IF NOT Gra_mode THEN Com   ! cariage return/ alpha mode
9330          Gra_al=1                   ! cr/ graphics mode
9340          X=(Marg-1)*Marg2           ! reposition cursor
9350          GOTO Com

9360 Draw:    X=W(4)-64+(W(3)-32)*32     ! calc. X-coord
9370          Y=W(2)-96+(W(1)-32)*32     ! calc Y-coord
9380          W=1
9390          IF NOT Not_draw THEN Draw1
9400          MOVE X,Y
9410          Not_draw=0
9420          GOTO Com

9430 Draw1:                              ! draw graphics
9440          IF NOT Esc_sub THEN DRAW X,Y
9450          GOTO Com

9460 Us:      IF NOT Comp_mode THEN Com  ! grpahics-alpha
9470          Gra_al=1
9471          Esc_sub=0
```

```
9480          IF NOT Gra_mode THEN Gs1
9490          GOTO Com

9500 Esc_ff:  IF NOT Comp_mode THEN Com  ! end graphics/clear grpahics
9510          Gra_al=1
9520          GCLEAR
9530          Escseq=0
9531          Esc_sub=0
9540          GOTO Gs1

9550 Esc_etb: IF NOT Comp_mode THEN Com ! Esc [etb] - DUMP GRAPHICS
9560         Escseq=0
9570         OUTPUT 0;CHR$(27)&"Z"&CHR$(8)
9580         DUMP GRAPHICS
9590         IF Trace THEN OUTPUT 0;CHR$(27)&"Y"
9600         GOTO Com
9610 Esc_sub1:Esc_sub=1
9611          Gra_al=0                                        !***WW
9620 Esc_sub:Escseq=0
9630          IF NOT Esc_sub THEN POINTER X+7,Y+12,2
9640          IF Esc_sub THEN POINTER X,Y,1
9650 Cal_xy:  R=ABS((X/32-INT(X/32))*32) ! Calc. HX,LX
9660          Z=INT(X/32)
9670          W(4)=32+R                                       !***WW
9680          W(3)=32+Z
9690          R=ABS((Y/32-INT(Y/32))*32) ! Calc. HY,LY
9700          Z=INT(Y/32)
9710          W(2)=32+R                                       !***WW
9720          W(1)=32+Z
9730          W$=CHR$(W(3))&CHR$(W(4))&CHR$(W(1))&CHR$(W(2))
9740          IF Es_enq AND NOT Esc_sub THEN Tra_xy
9750          IF Es_enq AND Esc_sub THEN Only_co
9760          GOTO Com
9770 Only_co:POINTER -1,-1,2   ! send coords. only /esc-sub,esc-enq
9780          OUTPUT Selectcode USING "#,K";W$&Gt$;
9790          Es_enq=Escseq=Esc_sub=0
9800          GOTO Com
9810 Esc_enq:Es_enq=1
9820          GOTO Cal_xy
9830 Tra_xy:  Stb=BINIOR(St,128+32+16*(Hardprinter()16)+4*Gra_al+2*(
        Marg=1))   !calc status byte
9840          OUTPUT Selectcode USING "#,K";CHR$(Stb)&W$&Gt$; ! send
                 X_Y coord.
9850          POINTER -1,-1,2
9860          Es_enq=Escseq=Esc_sub=0
9870          GOTO Com
9880 Gr_lf:   Gra_al=1         ! LF in gra. mode
9890          IF NOT Escseq THEN No_esc
9900          Escseq=0
9910          X=(Marg-1)*Marg2
9920          GOTO Com

9930 No_esc:  IF Y>21 THEN Pos_c ! top of screen
```

KMGTRM

```
9940          Marg=3-Marg
9950          Y=788                                    !***WW
9960          X=(Marg-1)*Marg2
9970 Pos_c:   Y=Y-21                 ! next line
9980          GOTO Com

9990 Lf:      IF Gra_mode THEN Gr_lf
10000         GOSUB Print
10010         GOTO Com

10020 Dcrmt:  Local=0
10030         Tell$="Remote"
10040         GOSUB Tell
10050         GOTO Com

10060 Bell:   BEEP
10070         GOTO Com

10080 Dc3:    IF NOT Handshake THEN Xon=0
10090         GOTO Com

10100 Trashit:IF POS(TBUF$,Enq$) THEN OUTPUT Selectcode;Ack$;
10110         RETURN

10120 Esc:    Escseq=1
10121         Eseq$=""                                 !***WW
10130         GOTO Com

10140 Eseq:   IF (Character=13) OR (Character=10) THEN Com
10150         IF NOT Comp_mode THEN Norm_esc
10160         IF Character=63 THEN Esc_qu
10170         IF Character=12 THEN Esc_ff
10180         IF Character=26 THEN Esc_sub1
10190         IF Character=5 THEN Esc_enq
10200         IF Character=23 THEN Esc_etb
10210         IF Character=29 THEN Gs
10220         IF NOT Gra_mode THEN Norm_esc
10230    ! Strip out Enq's, LF's, or CR's after an Esc
10240 Norm_esc:IF LEN(Aseq$)<18 THEN Escokay  ! Check for sequence
          too long
10250 Termesc:GOSUB Termes
10260         GOTO Com
10270 Termes:Escseq=0                ! Terminate escape processing
10280         Aseq$=""
10290         RETURN
10300 Escokay:Aseq$=Aseq$&Char$      ! Tack on new character
10310         IF (Character>63) AND (Character<91) THEN Endseq
10320                                ! Check for end of sequence
10330         GOTO Com

10340 Endseq:SERIAL
10350         IF LEN(Aseq$)=1 THEN Oneseq
10360         IF (Aseq$[1;1]<>"&") OR (LEN(Aseq$)<3) THEN Termesc
```

```
10370          ON 1+POS(Parm$,Aseq$[2;1]) GOTO Termesc,Dparm,Pparm,
                  Cparm,Sparm

10380 Dparm:  K4=POS(Hich$,Aseq$[3;1])
10390          IF NOT K4 THEN Termesc
10400          Disp$[LEN(Disp$)+1]=CHR$(127+K4)
10410          IF LEN(Disp$)=80 THEN GOSUB Print
10420          GOTO Termesc

10430 Pparm:  IF LEN(Aseq$)>3 THEN Pcheck
10440          K4=POS(Eswitch$,Aseq$[3;1])
10450          GOSUB Termes
10460          ON K4+1 GOTO Termesc,Chain,Fdfn,Purge,Setsize
10470 Sparm:IF Aseq$[6;1]<>"Q" THEN Termesc
10480         IF (Aseq$[3;3]="1p0") OR (Aseq$[3;3]="0p1") THEN Comp_on
10490         IF Aseq$[3;3]="0p0" THEN Nocomp
10500         GOSUB Termes
10510         GOTO Com
10520 Pcheck:IF Aseq$[4;1]<>"P" THEN Dcheck
10530          IF (Aseq$[3;1]<>"0") AND (Aseq$[3;1]<>"1") THEN Com
10540          K4=VAL(Aseq$[3;1])
10550          GOSUB Termes
10560          ON K4+1 GOTO Altplot_off,Altplot_on
10570 Dcheck:IF Aseq$[4;1]<>"D" THEN Scheck  ! Destination commands
10580          IF (Aseq$[3;1]>"5") OR (Aseq$[3;1]<"0") THEN Com
10590          K4=VAL(Aseq$[3;1])
10600          GOSUB Termes
10610          ON K4+1 GOTO Toff,Ton,Ton,Toffpall1,Pall,Pall1

10620 Scheck:IF Aseq$[4;1]<>"S" THEN Com      ! Source commands
10630          IF (Aseq$[3;1]>"3") OR (Aseq$[3;1]<"1") THEN Com
10640          K4=VAL(Aseq$[3;1])
10650          GOSUB Termes
10660          ON K4 GOTO Autosend_on,Autosend_on,Autosend_off
10670 Comp_on:  IF Gra_mode THEN Bad
10680            Comp_mode=1                !Esc&s0p1Q
10690            Tell$="Compatability mode on"
10700            Escseq=Local=Keycomp=0
10710            Aseq$=""
10720            GOSUB Tell
10730            GOTO Next
10740 Nocomp:  Comp_mode=0        !    Esc&s0p0Q
10750           Local=Keycomp=Escseq=0
10760            Tell$="Compatability mode off"
10770            Aseq$=""
10780            GOSUB Tell
10790            GOTO Next
10800 Oneseq:       ! single char. escapes
10810          GOSUB Termes
10820          IF (Character=89) OR (Character=90) THEN Tron
10830          GOTO Com
```

KMGTRM

```
10840 Altplot_on: IF (Plottercode<>13) AND (Plottercode<>0) THEN
        10870 ! Esc&p1P
10850       Comment$="No alternate plotter specified"
10860       GOTO Comcom
10870       IF Phpib<>999 THEN PLOTTER IS Plottercode,Phpib,Plotter$
10880       IF Phpib=999 THEN PLOTTER IS Plottercode,Plotter$
10890       CSIZE 2
10900       GOSUB Testalt
10910       GOTO Comcom
10920 Altplot_off: Altplot=0        ! Esc&p0P
10930        Comment$="Alternate plotter off"
10940        PLOTTER IS "GRAPHICS"
10950        GOTO Comcom
10960 Rec_file:IF Autosend THEN Bad
10970          IF Recflag THEN Endrec
10980          Prev$=CHR$(27)&"&p1D"
10985          IF Local THEN Locflag=1
10990          Local=Keyrec=1
11000          Old=0
11010          RETURN
11020 Endrec:  Prev$=CHR$(27)&"&p0D"
11030          Old=0
11031          IF Local THEN Locflag=1
11040          Local=Keyrec=1
11050          RETURN
11060 Ton:    Comment$="Tape recording start" ! Esc&p1D or Esc&p2D
11061         IF Keyrec AND NOT Locflag THEN Local=0
11062         Locflag=0
11070         Disp$=""
11080         Recflag=1
11090         Locked=Firstline=1
11100         ON ERROR GOTO Eerr
11110 Tonloop:ASSIGN #1 TO Datafile$,T1
11120         IF T1>1 THEN Eerr
11130         Recording=1
11140         OFF ERROR
11150         IF NOT T1 THEN Pdone
11160         CREATE Datafile$,Filesize
11170         GOTO Tonloop
11180 Eerr:   ASSIGN #1 TO *
11190         Recording=0
11200         GOTO Xerr

11210 Toff:   K4=POS(Comstr$,CHR$(27)&"&p0D")
11220         IF K4 THEN Comstr$=Comstr$[1,K4-1]
11230         PRINT #1;Comstr$,END              ! Esc&p0D
11231         IF Keyrec AND NOT Locflag THEN Local=0
11232         Locflag=0
11240         ASSIGN #1 TO *
11250         Recflag=Recording=0
11260         Firstline=1
11270         Comment$="Tape file close"
```

```
11280          GOTO Pdone

11290 Toffpall1:ASSIGN #1 TO *                    ! Esc&p3D
11300          Printall=Recording=0
11310          Firstline=1
11320          Comment$="Tape file close and hardcopy off"
11330          GOTO Pdone

11340 Deffil:IF Gra_mode THEN Bad
11350          Define=1
11360         GOTO Setup
11370 Fil_wt:WAIT 1000
11380 Fil_def:OFF KBD
11390           Define=0
11400           INPUT "Enter file name:  [Use CONT]",Datafile$
11410           ON ERROR GOTO Crerror
11420           ASSIGN #9 TO Datafile$,T1
11430           ASSIGN #9 TO *
11440           OFF ERROR
11450           IF T1=1 THEN Crerror
11460 F_ok:     Tell$="File name redefined to "&Datafile$
11470           GOSUB Tell
11480           ON KBD 10 GOSUB Keyservice
11490           IF Gra_mode AND NOT Gra_al THEN ON KBD 10 GOSUB
                   Keyservice ,ALL
11500           GOTO Wait
11510 Crerror:OFF ERROR
11520           IF T1<>2 THEN Crfil
11530           BEEP
11540           Tell$=Datafile$&" not found -- error "&VAL$(ERRN)
11550           GOSUB Tell
11560           GOTO Fil_wt
11570 Crfil:INPUT "Enter size of file:  [Use CONT]",Filesize
11580         CREATE Datafile$,Filesize
11590         GOTO F_ok

11600 Testalt:ON ERROR GOTO Alterr
11610         ON INT #Plottercode GOTO Alterr
11620         SET TIMEOUT Plottercode;1000
11630         STATUS Plottercode*(1+99*(Phpib<>999))+Phpib*(Phpib<>999);
                St
11640         Altplot=1
11650         Comment$="Alternate plotter on"
11660         GOTO 11690
11670 Alterr: Comment$="Alternate plotter not found"
11680           St=TIME OUT(Plottercode)
11690           OFF ERROR
11700           OFF INT #Plottercode
11710           RETURN

11720 Chain: Comment$="Background program link" ! Esc &pE
11730          Locked=Firstline=1
```

KMGTRM

```
11740          ON ERROR GOTO Xerr
11750          ASSIGN Datafile$ TO #1,T1
11760          IF T1 THEN Xerr
11770          LINK Datafile$,13700,Gstrt
11780 Gstrt:   IF NOT Local THEN OUTPUT Selectcode USING Pfmt;"S"
11790          Comment$=Comment$&" successful"
11800          GOTO Oferr

11810 Cparm:   K4=1
11820          IF Aseq$[3;1]<>"E" THEN K4=0
11830          GOSUB Termes
11840          IF NOT K4 THEN Com
11850 Gstrt1:Gojump=1                          ! Esc +
11860 Gstrt2:IF NOT Local THEN Oferr
11870          BEEP
11880          DISP Rub$&" Local mode exited "&Rub$
11890          WAIT 2500
11900          Local=0
11910          GOTO Oferr

11920 Fdfn:    T1=0                            ! Esc &pG
11930 Fdfn0:   T1=T1+1
11940 Fdfn2:   IF T1>LEN(Disp$) THEN Fdfn1
11950          IF Disp$[T1;1]>" " THEN Fdfn0
11960          Disp$[T1]=Disp$[T1+1]
11970          GOTO Fdfn2
11980 Fdfn1:   ON ERROR GOTO Illegalfile
11990          Datafile$=Disp$
12000          ASSIGN #9 TO Datafile$,X
12010          ASSIGN #9 TO *
12020          OFF ERROR
12030          Disp$=""
12040          Comment$="File name redefined to "&Datafile$
12050          GOTO Comcom
12060 Illegalfile:OFF ERROR
12070          BEEP
12080          Comment$=Disp$&" not found -- error "&VAL$(ERRN)
12090          Disp$=""
12100          GOTO Comcom

12110 Pall:    Comment$="Hardcopy on"          ! Esc&p4D
12120          Trace=0
12130          Printall=1
12140          GOTO Comcom
12150 Pall1:   Comment$="Hardcopy off"         ! Esc&p5D
12160          Printall=0
12170          GOTO Comcom

12180 Setsize:ON ERROR GOTO Default            ! Esq &pA
12190          Filesize=VAL(Disp$)
12200 Size1:   OFF ERROR
12210          Comment$="Filesize set to "&VAL$(Filesize)
```

```
12220          Disp$=""
12230          GOTO Comcom
12240 Default:Filesize=30
12250          GOTO Size1
12260 Purge:   Comment$="File purge"          ! Esc &pN
12270          Locked=Firstline=1
12280          ON ERROR GOTO Xerr
12290          ASSIGN #1 TO Datafile$,T1
12300          IF T1=1 THEN Pdone
12310          ASSIGN #1 TO *
12320          PURGE Datafile$
12330 Pdone:   WAIT 500
12340          IF NOT Local AND NOT Keyrec THEN OUTPUT Selectcode USING
                  Pfmt;"S"
12350          Comment$=Comment$&" successful"
12370          Keyrec=0
12380 Oferr:   OFF ERROR
12390          Locked=0
12400          GOTO Comcom
12410 Xerr:    WAIT 500
12420          Sendflag=Autosend=Keysend=Keyrec=Keycomp=0
12430          IF NOT Local THEN OUTPUT Selectcode USING Pfmt;"F"
12440          Comment$=Comment$&" unsuccessful"
12450 Pfmt:    IMAGE #,A,L
12460          GOTO Oferr

12470 Send_file: IF Gra_mode OR Recording THEN Bad
12480             IF Local THEN Locflag=1
12490             IF Autosend THEN Endsend
12500             Prev$=CHR$(27)&"&p1S"
12510             Local=Keysend=1
12520             Old=0
12530             RETURN
12540 Endsend:   Prev$=CHR$(27)&"&p3S"
12545            IF Local THEN Locflag=1
12550            Local=Keysend=1
12560            Old=0
12570            RETURN
12580 Autosend_on:Autosend=Sendflag=1       ! Esc&p1S OR Esc&p2S
12590          Comment$="Autosend "
12600          IF NOT Locflag AND Keysend THEN Local=0
12610          Locflag=0
12620          ON ERROR GOTO Xerr
12630          ASSIGN #2 TO Datafile$,T1
12640          IF T1 THEN Xerr
12650          OFF ERROR
12660          GOSUB Autocomment
12670          WAIT 500
12680          IF NOT Local AND NOT Keysend THEN OUTPUT Selectcode
                  USING Pfmt;"S"
12690          Keysend=0
12700          GOTO Comcom
```

KMGTRM

```
12710 Autosend_off:Autosend=Sendflag=0      ! Esc&p3S
12711        IF Keysend AND NOT Locflag THEN Local=0
12720         WAIT 500
12730         IF NOT Local AND NOT Keysend THEN OUTPUT Selectcode
                 USING Pfmt;"S"
12740         Keysend=0
12750         ASSIGN #2 TO *
12760         GOSUB Autocomment
12770         GOTO Comcom
12780 Autocomment:Comment$="Auto send "&State$(Autosend)
12790         RETURN

12800 Tron:   IF Hardselect()16 THEN Tron1  ! Esc Y and Z turns trace
         on/off.
12810         BEEP
12820         Comment$="Command ignored -- Hardcopy printer required"
12830         GOTO Comcom
12840 Tron1:  Trace=(Character(90)
12850         GOSUB Setprinter
12860         PRINT CHR$(27);CHR$(Character)
12870         PRINTER IS 16
12880         Comment$="Display functions "&State$(Trace)
12890         Printall=0
12900 Comcom:Clear=1000
12910 Comcom1:DISP TAB(74-LEN(Comment$));Rub$&" "&Comment$&" "&Rub$
12920         IF NOT Keysend THEN Com
12930            Keysend=0
12940            RETURN
12950         GOTO Com

12960 Cret:   IF NOT Gra_mode THEN Cret1
12970         Pointer=1
12980         IF NOT Esc_sub THEN POINTER X+7,Y+12,2
12990         IF Esc_sub THEN POINTER X,Y,1
13000         IF Esc_sub THEN Gra_al=0
13010         GOTO Com
13020 Cret1:  IF Firstline THEN Com
13030         IF Autosend THEN Sendflag=1
13040         IF NOT Handshake THEN Xon=1
13050         Prompt$=Disp$[1,70]
13060         Buffer$=Buffer$[1,159-LEN(Prompt$)]
13070         TDISP CHR$(12)&Prompt$&Buffer$&RPT$(CHR$(8),LEN(Buffer$)-
                 Cursor+1)
13080         GOTO Com

13090 Record:T1=0
13100 Rec2:   T1=T1+1
13110 Rec1:   IF T1>LEN(Disp$) THEN Rec3
13120         IF Disp$[T1;1]<>CHR$(5) THEN Rec2
13130         Disp$[T1]=Disp$[T1+1]          ! Strip off Enq's
13140         GOTO Rec1
13150 Rec3:   ON ERROR GOTO Msfail
```

```
13160          IF (Disp$<>CHR$(10)) AND LEN(Disp$) THEN PRINT #1;Disp$,
                 END
13170          Comstr$=""
13180          OFF ERROR
13190          RETURN
13200 Msfail:BEEP
13210          OFF ERROR
13220          DISP " Mass storage failure -- file closed "
13230          Recording=0
13240          WAIT 1000
13250          RETURN
13260 Print:  SERIAL
13270          IF Firstline THEN Print2
13280          IF LEN(Disp$)<80 THEN PRINT Disp$&CHR$(128)
13285          IF LEN(Disp$)>79 THEN PRINT Disp$
13290 Print2:IF NOT Printall AND Hardselect=16 THEN Print1
13300 Print3:T1=POS(Disp$,CHR$(12))
13310          IF NOT T1 THEN Print4
13320          OUTPUT Hardselect;Disp$[1,T1-1]&RPT$(CHR$(10),6)
13330          Disp$=Disp$[T1+1]              ! Replace formfeed with 6
                 line feeds
13340          GOTO Print3
13350 Print4:OUTPUT Hardselect;Disp$&CHR$(128)
13360 Print1:Disp$=""
13370          TDISP CHR$(12)
13380          Firstline=0
13390          RETURN

13400 Set_hard:Hpibflag=(Hpib<>999)
13410          Hardselect=Hardprinter*(NOT Hpibflag+100*Hpibflag)+Hpib*
                 Hpibflag
13420          RETURN

13430 Setprinter:IF Hpibflag THEN PRINTER IS Hardprinter,Hpib
13440          IF NOT Hpibflag THEN PRINTER IS Hardprinter
13450          RETURN




!Data for initial definitions of characters

13470 Defs:
13480 DATA 127,71,^Y,127,65,^!?^I^!?^T^M^!?^I,127,6,^I
13490 DATA 127,70,^F,127,1,^!?^I^!?^T^M^J^!?^I,127,69,^!?^!?
13500 DATA 127,254,^M
13510 DATA -1,0,""
!

 Data for names of total initialization
```

KMGTRM

```
13530 Initialconds:
13540 DATA 11,300,1,2,8,3,"off","on","DCdata:T15",30,17,1E99,0,999,21,
        "","
1
13560 DATA 0,nul,1,soh,2,stx,3,etx,4,eot,5,enq,6,ack,7,bel,8,bs,9,ht,
        10,lf
13570 DATA 11,vt,12,ff,13,cr,14,so,15,si,16,dle,17,dc1,18,dc2,19,dc3,
        20,dc4
13580 DATA 21,nak,22,syn,23,etb,24,can,25,em,26,sub,27,esc,28,fs,29,
        gs,30,rs
13590 DATA 31,us,32,space,255,0,UDK0,1,UDK1,2,UDK2,3,UDK3,4,UDK4,5,
        UDK5,6,UDK6
13600 DATA 7,UDK7,8,UDK8,9,UDK9,10,UDK10,11,UDK11,12,UDK12,13,UDK13,
        14,UDK14
13610 DATA 15,UDK15,16,step,17,pause,18,run,19,cont,20,store,21,
        execute,22,left
13620 DATA 23,right,24,up,25,down,26,roll-up,27,roll-down,28,home,29,
        clear
13630 DATA 30,clear-to-end,31,delete-char,32,insert-char,33,delete-
        line
13640 DATA 34,insert-line,35,recall,36,tab,37,tab-set,38,tab-clear,39,
        typewriter
13650 DATA 40,backspace,41,result,42,stop,43,clear-line,254,end-of-
        line,-1,""
1


 Data for 98036 card baud rates

13670 Baudrates:
13680 DATA 75,110,150,300,600,1200,1800,2400,4800,9600
13690 End:! End of terminal emulator program
13700 END
```

Minimal Line Mode Program Listing

```
1000 COM Buf$[159],Cursor,Dcom$[500],Sel_code
1010 DIM A$[200],B$[500],C$[100]
1020 Cursor=1
1030 Sel_code=10
1040 ON KBD CALL Kbd_isr
1050 CALL Dcom_setup(Err)
1060 IF Err THEN STOP
1070 TOPEN Sel_code CALL Dcom_isr
1080 DISP "RUNNING"
1090 IF Dcom$[MAX(LEN(Dcom$),1)]=CHR$(5) THEN OUTPUT Sel_code;CHR$(6);
1100 IF NOT FNKbd_ready THEN 1130
1110 CALL Get_line(A$)
1120 OUTPUT Sel_code USING "#,K";A$&CHR$(13)
1130 IF NOT FNDc_ready THEN 1090
1140 CALL Get_dcom(B$,Prompt)
1150 IF Prompt<0 THEN STOP
1160 IF Prompt THEN 1210
1170 PRINT C$&B$
1180 C$=""
1190 DISP
1200 GOTO 1090
1210 DISP B$
1220 C$=B$
1230 GOTO 1090
1240 !
1250 !
1260 ! ***********************************************************
**********
1290 ! Package for use of datacomm channel using 98036 interface card
and
1300 ! ON KBD facilities.  Following are all subroutines and multi-lin
e
1310 ! which may be appended to any program.  Instructions precede eac
h routine.
1320 ! ***********************************************************
**********
1330 ! Kbd_isr: Keyboard entry interrupt service routine.  This routin
e handles
1340 !      taking in keystrokes into a buffer, editing the buffer, and
 then
1350 !      signalling completion when STORE, CONT, or EXECUTE is hit.
1360 ! Method of use:
1370 !      10        COM Kbuf$[159],Cursor,Dcbuf$[500],Sel_code
1380 !      20        DIM A$[159]
1390 !
1400 !      1000      ON KBD CALL Kbd_isr            [,ALL]
1410 !
1420 !      2000      IF NOT FNKbd_ready THEN Somewhere_else
1430 !      2010      CALL Get_line(A$)
1440 !
1450 !      Line 2000 is not strictly necessary, as the routine Get_lin
e will
1460 !      wait for FNKbd_ready; however this enables the user to do o
ther
1470 !      things while waiting for a keyboard input.
1480 ! Modifications:
1490 !      Special function key (anything non-ASCII) definitions may b
e deleted
1500 !      by deleting the label for that function.  For example, to d
elete
1510 !      the function of the "->" key, just delete where the label "
Rj:"
1520 !      appears on the left.  To install a function, just add a new
 label
```

```
1530 !      for that function (see names in multiple ON..GOTO's) possib
ly
1540 !      followed by code.  "Pall:" will re-display the keyboard lin
e with
1550 !      any modifications, "Nextkey:" goes on to next keystroke.  N
ote:
1560 !      Cursor must be in the range 0 < Cursor <= LEN(Kbd$)+1.
1570 ! ***********************************************************************
**********
1580 !
1590       SUB Kbd_isr
1600 Kbd_isr: COM Kbd$[159],Cursor,Dcom$[500],Sel_code
1610          DIM K$[10]
1620          ON ERROR GOTO Keyofl
1630          K$=KBD$
1640          OFF ERROR
1650 Nextkey: IF NOT LEN(K$) THEN Goway       ! Strip off keystrokes on
e by one
1660          IF NOT Cursor THEN Beep         ! Cursor=0 => already ent
ered line
1670          K=NUM(K$)
1680          IF K=255 THEN Second            ! implies special functio
n key
1690          K$=K$[2]
1700          IF (Cursor=160) OR (K=12) THEN Beep
1710          IF K=8 THEN Lf                  ! treat control-H like "<
-" key
1720          Kbd$[Cursor;1]=CHR$(K)          ! put keystroke in buffer
1730 Ri:      IF LEN(Kbd$)<Cursor THEN Nextkey  ! *** entry for "->" k
ey ***
1740          TDISP Kbd$[Cursor;1]            ! display character in cu
rsor pos
1750          Cursor=Cursor+1                 ! advance cursor
1760          GOTO Nextkey
1770 Keyofl:  DISP "Keyboard buffer overflow."
1780          BEEP
1790 Goway:   SUBEXIT
1800 Second:  IF LEN(K$)>1 THEN Sfk           ! wait until second half
appears
1810          K$=K$&KBD$                      ! in keystroke buffer (if
 not
1820          GOTO Second                     ! already there.)
1830 Sfk:     K=NUM(K$[2])
1840          K$=K$[3]                        ! throw out first two cha
racters
1850          Shift=K DIV 64                  ! store cntrl and shift b
its
1860          K=BINAND(K,63)                  ! and take off of key ind
icator
1870          ON ERROR GOSUB Nada             ! Do the various function
s
1880          ON K+1 GOTO U0,U1,U2,U3,U4,U5,U6,U7,U8,U9,Ua,Ub,Uc,Ud,Ue
,Uf
1890          ON K-15 GOTO Step,Pa,Ru,Co,St,Ex,Lf,Ri,Up,Dn,Rlu,Rld,Ho,
Clr,C2e
1900          ON K-30 GOTO Dlc,Inc,Dll,Inl,Rcl,Tab,Tbs,Tbc,Typ
1901          ON K-49 GOTO Bs,Res,Stop,Cll
1910          OFF ERROR
1920 Beep:    BEEP
1930          GOTO Nextkey
1940 Nada:    RETURN
1950 Pa:      WRITE IO Sel_code,5;1           ! *** entry for PAUSE (br
eak) ***
1960          WRITE IO Sel_code,4;46
1970          WRITE IO Sel_code,5;132
1980          BEEP
1990          WAIT 200
2000          BEEP
2010          WRITE IO Sel_code,5;1
2020          WRITE IO Sel_code,4;7
2030          WRITE IO Sel_code,5;132
```

```
2040            GOTO Nextkey
2050 Bs:        !                                   ! *** entry for BACKSPACE
 key ***
2060 Lf:        IF Cursor<2 THEN Nextkey            ! *** entry for "<-" key
***
2070            Cursor=Cursor-1
2080            TDISP CHR$(8)
2090            GOTO Nextkey
2100 Clr:       PRINT PAGE;                         ! *** entry for CLEAR key
 ***
2110            DISP
2120 Cll:       Kbd$=""                             ! *** entry for CLEAR LIN
E key ***
2130 Ho:        Cursor=1                            ! *** entry for HOME key
***
2140            GOTO Pall
2150 C2e:       Kbd$[Cursor]=""                     ! *** entry for CLEAR->EN
D key ***
2160 Pall:      TDISP CHR$(12)&Kbd$&RPT$(CHR$(8),LEN(Kbd$)-Cursor+1)
2170            GOTO Nextkey
2180 Co:                                            ! *** entry for CONT key
***
2190 St:                                            ! *** entry for STORE key
 ***
2200 Ex:        IF (K<19) OR (K>21) THEN Beep       ! *** entry for EXECUTE k
ey ***
2210            Cursor=0
2220        SUBEND
2230 !
2240 ! ***************************************************************
**********
2250 ! Get_line(Line$): Retrieves a line from the keyboard when comple
tion is
2260 !      signalled via Cursor=0 (implies STORE, CONT, or EXECUTE.)
This will
2270 !      wait until the line is completed.  There are no error condi
tions.
2280 ! See Kbd_isr for example of usage.
2290 ! ***************************************************************
**********
2300 !
2310        SUB Get_line(Line$)
2320 Get_line:COM Kbd$[159],Cursor,Dcom$[500],Sel_code
2330 Loop:      IF NOT FNKbd_ready THEN Loop        ! Wait until keyboard rea
dy
2340            Line$=Kbd$                          ! Store keyboard buffer
2350            TDISP CHR$(12)                      ! And clear entry area
2360            Kbd$=""
2370            Cursor=1                            ! Re-enable key entry
2380        SUBEND
2390 !
2400 ! ***************************************************************
**********
2410 ! Dcom_isr: Datacomm interrupt service routine.  This routine tak
es care of
2420 !      buffering input from the datacomm line, 500 characters maxi
mum.
2430 ! Example of usage:
2440 !      10      COM Kbuf$[159],Cursor,Dcbuf$[500],Sel_code
2450 !      20      DIM B$[500]
2460 !
2470 !      990     Sel_code=11
2480 !      1000    CALL Dcom_setup(Err)        ! I/O card is select code
11
2490 !      1010    IF Err THEN STOP
2500 !      1020    TOPEN 11 CALL Dcom_isr
2510 !
2520 !      2000    IF NOT FNDc_ready THEN Elsewhere
2530 !      2010    CALL Get_dcom(B$,Prompt)
2540 !      2020    IF Prompt<0 THEN STOP
```

```
2550 !
2560 !      As before, line 2000 is not strictly necessary since the ro
utine
2570 !      Get_dcom will wait for a completed line or prompt.
2580 ! ***********************************************************
**********
2590 !
2600        SUB Dcom_isr
2610 Dcom_isr:COM Kbd$[159],Cursor,Dcom$[500],Sel_code
2620          DIM D$[330]
2630          D$=TBUF$                          ! Get datacomm buffer
2640          FOR I=1 TO LEN(D$)                ! Strip parity bit off
2650            D$[I,I]=CHR$(BINAND(NUM(D$[I]),127))
2660          NEXT I
2670          IF LEN(D$)+LEN(Dcom$)>500 THEN Error
2680          Dcom$[LEN(Dcom$)+1]=D$            ! Add onto internal buffe
r
2690          SUBEXIT
2700 Error:   DISP "Datacomm overrun."
2710          BEEP
2720        SUBEND
2730 !
2740 ! ***********************************************************
**********
2750 ! Get_dcom(Dc$,Prompt): Subroutine to get a line from the datacom
m channel.
2760 !      An end-of-line is signalled by either a carriage return (AS
CII 13) or
2770 !      DC1 (ASCII 17).  The latter indicates that there is a promp
t from the
2780 !      computer.  The variable "Prompt" indicates which was the ca
se.
2790 ! ***********************************************************
**********
2800 !
2810        SUB Get_dcom(Dc$,Prompt)
2820 Get_dcom:COM Kbd$[159],Cursor,Dcom$[500],Sel_code
2830 Loop:    IF NOT FNDc_ready THEN Loop
2840          Prompt=0                          ! Prompt=0 => ended with
CR
2850          N1=POS(Dcom$,CHR$(13))            ! Position of CR
2860          N2=POS(Dcom$,CHR$(17))            ! Position of DC1
2870          N=N1*NOT N2+N2*NOT N1+MIN(N1,N2)*(N1 AND N2)
2880          ON ERROR GOTO Err                 ! Pick N=minimum
2890          Prompt=(N=N2)                     ! Prompt=1 => ended with
DC1
2900          Dc$=Dcom$[1,N-1]                  ! Get characters precedin
g
2910          Dcom$=Dcom$[N+1]                  ! and take out of buffer
2920          IF Dc$[1,1]=CHR$(10) THEN Dc$=Dc$[2] ! strip off leading
 LF's
2930          SUBEXIT
2940 Err:     DISP "String overflow from datacomm channel."
2950          Prompt=-1                         ! signals an error
2960          BEEP
2970        SUBEND
2980 !
2990 ! ***********************************************************
**********
3000 ! Dcom_setup(Err): Sets up the 98036 card on the select code indi
cated in
3010 !      the COM statement.  Err returns zero or one, indicating whe
ther card
3020 !      was successfully set up.  Err=1 usually implies wrong selec
t code.
3030 ! See Dcom_isr for example of use.
3040 ! ***********************************************************
**********
3050 !
3060        SUB Dcom_setup(Err)
3070 Dcom_set:COM Kbd$[159],Cursor,Dcom$[500],Sel_code
```

```
3071            READ Stopbits,Parity,Bitsperchar,Bitratefactor
3072            DATA 1,2,8,3
3080            Err=0
3090            IF NOT IOSTATUS(Sel_code) THEN Error ! check STS line of
 card
3100            STATUS Sel_code;S                   ! read R5, check card typ
e
3110            IF BINAND(S,48)<>16 THEN Error
3120            WRITE IO Sel_code,5;1               ! Set R4 mode to ctrl/sta
tus
3130            WRITE IO Sel_code,4;64              ! USART reset
3139                                                ! set R4 mode word (R4C)
3140            WRITE IO Sel_code,4;Stopbits*64+Parity*16+(Bitsperchar-5
)*4+Bitratefactor
3150            WRITE IO Sel_code,4;39              ! 00 100 111 -> R4D (ctrl
 word)
3160            WRITE IO Sel_code,5;0               ! set R4 mode to data in/
out
3170             READ IO Sel_code,4;S
3180            WRITE IO Sel_code,7;0               ! set up to generate inte
rrupts
3190            WRITE IO Sel_code,5;132
3200            STATUS Sel_code;S                   ! check if interrupts are
 on
3210            IF BINAND(S,128) THEN SUBEXIT
3220 Error:     DISP "Select code not operational."
3230            BEEP
3240            Err=1
3250        SUBEND
3260 !
3270 ! ***********************************************************
**********
3280 ! FNDc_ready: This function returns a 1 if the datacomm buffer co
ntains a
3290 !      Carriage return (ASCII 13) or DC1 (ASCII 17) indicating pro
mpt.
3300 ! ***********************************************************
**********
3310 !
3320        DEF FNDc_ready
3330 Dc_rdy:    COM Kbd$[159],Cursor,Dcom$[500],Sel_code
3340            RETURN POS(Dcom$,CHR$(13)) OR POS(Dcom$,CHR$(17))
3350        FNEND
3360 !
3370 ! ***********************************************************
**********
3380 ! FNKbd_ready:  This function returns a 1 if Cursor=0.  This usua
lly
3390 !      implies that CONT, STORE, or EXECUTE was hit.  If Cursor=0
the
3400 !      keyboard interrupt service routine is locked out from chang
ing Kbd$.
3410 ! ***********************************************************
**********
3420 !
3430        DEF FNKbd_ready
3440 Kbd_rdy: COM Kbd$[159],Cursor,Dcom$[500],Sel_code
3450            RETURN NOT Cursor
3460        FNEND
```

HEWLETT hp PACKARD

Part No. 09845-10141

Printed in U.S.A.
September 1, 1979