View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by DigitalCommons@USU

Utah State University

DigitalCommons@USU

All Graduate Plan B and other Reports

Graduate Studies

5-1997

Terminal Emulation System

Chun-Fu Lee
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/gradreports

 Part of the Computer Sciences Commons

Recommended Citation

Lee, Chun-Fu, "Terminal Emulation System" (1997). *All Graduate Plan B and other Reports*. 1529.
https://digitalcommons.usu.edu/gradreports/1529

This Report is brought to you for free and open access by the Graduate Studies at DigitalCommons@USU. It has been accepted for inclusion in All Graduate Plan B and other Reports by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.

TERMINAL EMULATION SYSTEM

by

Chun-Fu Lee

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

(Plan B)

Approved:

Dr. Jianping Zhang Major Professor	Dr. Donald H. Cooley Committee Member
Dr. Daniel W. Watson Committee Member	

UTAH STATE UNIVERSITY
Logan, Utah

1997

Copyright © 1998 Chun-Fu Lee

All Rights Reserved

ACKNOWLEDGMENTS

Several people have helped me during the preparation of this report. I would like to particularly acknowledge and thank Dr. Jianping Zhang who assisted and provided me with the opportunity to pursue this fascinating topic. I would also like to thank Dr. Donald H. Cooley and Dr. Daniel W. Watson who carefully reviewed this report.

Thanks to my parents for their continuous support and encouragement. I also thank my mother-in-law who came to take care of my son and let me have more time in writing this report.

Finally and most importantly, I dedicate this report to my wife, Hsiu-Chun, and my son, Joseph. Without them, I can hardly complete this report.

Chun-Fu J. Lee

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Terminal Emulation System

by

Chun-Fu Lee, Master of Science

Utah State University, 1997

Major Professor: Jianping Zhang

Department: Computer Science

Terminal emulation is a personal computer application software which emulates the terminal's behavior to communicate with the mainframe/host computer. It can show the terminal screen on a personal computer screen and make the mainframe/host computer interact with it as it would with a real terminal without any distinction.

Personal computers are increasingly more powerful and efficient today. Their mobility, compatibility, flexibility, and extendibility are definitely superior to the 'dumb' traditional terminals. Terminal emulation is therefore devised to take advantage of these personal computers' redemptions to offset the traditional terminal's drawbacks and inconveniences.

A terminal emulation is actually a very complex system. It needs to take care of terminal data presentation, input device handling, output device manipulation, communication line control and handshaking, system configuration processing, and so forth. A fully functioning terminal emulation further provides users with additional convenient utilities. The purpose of this report is to develop a VT220 terminal emulation system, mainly focusing on the terminal function, in a Windows environment.

(66 pages)

CHAPTER 1

INTRODUCTION

1.1 Motivation

Before personal computers had been invented and were not so popular as they are now, the mainframe/host computer was the major computing machinery in the related areas. Most companies, institutes and organizations that would have liked to employ computers to perform their daily tasks had to make use of mainframe/host computers to serve their purposes. Since traditional terminals were the standard equipment to operate mainframe/host computers, it was essential for computer persons to work with them.

Little by little, the mainframe/host computer applications have become mature and refined in each respect after successive improvements for 20 years or more. They are actually the incarnation of integrated intelligence, especially the rules and information structures which manage the business, developed by generations of developers and participants. Even though the personal computer is so popular today, the mainframe/host computer still plays a very important role. In fact, there are still quite a few people who prefer to work with mainframe/host computer for its performance, reliability, and security.

In another aspect, the technology of computer networking [8] has made great progress in the past decades. All the networks with different architectures are no longer incompatible any more. Presently, almost all the mainframe computers in the world are connected to each other for the purpose of exchanging meaningful information, sharing resources, and supporting each other. In which case, the mainframe computer will be much more significant in the hereafter especially

while "the Information Superhighway is actually still under construction." [6]

To have access with the mainframe/host computer, the terminal is after all the traditional medium which people are used to working with. Now that the mainframe/host computer is going to be the substantial facility based upon these foregoing reasons, the solution of the terminal therefore is still a necessity in the future.

1.2 The Problem

In spite of demands and requirements, terminals, however, have many defects which limit themselves to function as a powerful mechanism. Following are the most frequent problems of a traditional terminal access.

First of all, a traditional terminal was generally located relatively close to a mainframe/host computer or a cluster controller [10] connecting to the host computer. The connection was always made by a specific point-to-point link to communicate with its host. As a result, long distance communications between host computers and terminals, especially when many networks were crossed (including telephone networks), required much more expense and were otherwise difficult. This caused many inconveniences to remote users to utilize mighty resources which were implemented to the mainframe/host computer.

Secondly, the complicated and difficult settings for a terminal were complained about by the user. To set up the terminal equipment, it was always the case that users could do nothing but totally depend on their terminal provider to configure their terminals. In addition, troubleshooting was almost impossible for users because the terminal equipment was just like a "black box" to them. The maintenance problem therefore made the user do business faithfully with their

terminal provider. This restricted those users who would have liked to try another product from a different provider at lower cost.

There was also a frequent reproach from users that the 'unalterable' terminal was commonly old-fashioned and worthless. To upgrade their terminal equipment, all they could do was purchase new models and throw their old terminals into a repository.

Moreover, there are users who may want to work with various host applications for different processes. Meanwhile, some users may also want to have several links to different host computers simultaneously to access diverse information. Nevertheless, a traditional terminal was usually a single-sessioned terminal that provided only one terminal session to the user. This forced those users to give up their ideal.

At last, a terminal possessed no secondary storage and was unable to attach other peripheral devices for different purposes or even to add a new feature interface card, including the software for hardware installation, whenever the user would like to have brought the user much more harassment.

1.3 The Solution

Due to the disadvantages of a traditional terminal as mentioned above, people might want a new solution with more flexibility and extendibility to cope with the forthcoming unanticipated challenges. Terminal emulation on a personal computer is now the best choice of substitution for traditional terminals.

A terminal emulation can have the terminal screen appear on a personal computer screen. When dealing with a terminal emulation, the mainframe/host computer always interacts with the

terminal emulator as it would with a real terminal without any discrimination.

Terminal emulation is much more useful and user-friendly than a terminal. A fully functioning terminal emulation further provides users with various advantageous utilities and functions such as the file transfer function [7], the keyboard remapping (redefinition) module, phonebook administration utility, automatic dial-up and the log-on procedure, diverse communication channels, application programming interfaces (APIs) [5], magnetic strip reader, bar code reader, and so on. Through terminal emulations, users are capable of accessing mainframe data while concurrently working with varied personal computer applications.

Furthermore, personal computers are increasingly more powerful and efficient in the present day. Their mobility, compatibility, flexibility, and extendibility exactly match the user's demands. Particularly, their low price and their ability to be upgraded have made personal computers the user's favorite.

CHAPTER 2

VT220 TERMINAL EMULATION SYSTEM TECHNICAL DESCRIPTION

In general, a terminal emulation system involves four major components--communication, keyboard, terminal protocol, and printer. Each topic has its own technical description or regulation. This chapter is to describe the VT220 terminal emulation system involving these subjects and their technical descriptions.

2.1 Communication Channels

There are two kinds of communication modes that a VT220 terminal emulation might employ. They are asynchronous serial communication and internetworking communication. Figure 2.1 illustrates how a VT220 terminal emulation would connect to the host computers.

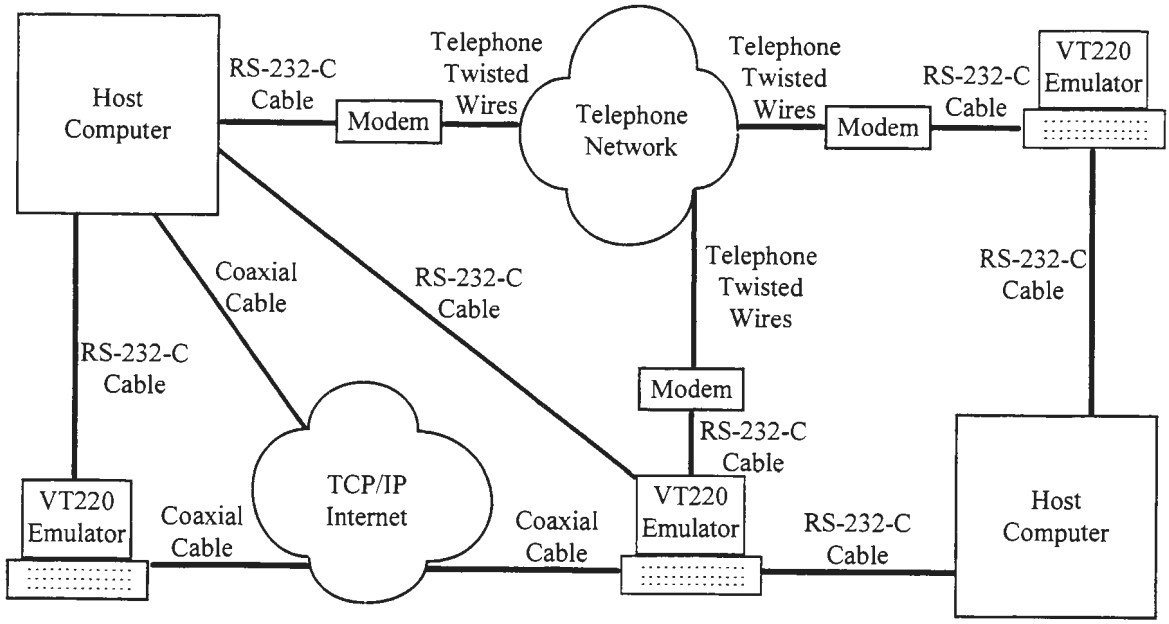

Figure 2.1 VT220 Terminal Emulation Communication Configuration

2.1.1 Asynchronous Serial Communication

The standard design of hardware communication channel for the VT220 terminal is RS-232-C [9] asynchronous serial communication. As in Figure 2.1, a VT220 emulator can connect directly to the host computer with an RS-232-C cable. It is actually the conventional manner for a VT220 terminal to communicate with its host. A terminal emulation may concurrently connect to more than one host computer if it uses both the personal computer's RS-232-C communication ports or a multiple-channel interface for serial communication. For the remote user, the emulator needs a Modem (MOdulator-DEModulator) [11] to connect to the host computer. In such case, it also needs a modem at the host end to communicate with the one at the terminal end. Figure 2.1 shows that the emulator connects itself to a host computer by way of modems and telephone network. The VT220 terminal emulation system proposed by this report adopts the RS-232-C serial port as its communication channel and the in-band method (XON/XOFF) for flow control.

2.1.2 Internetworking Communication

A VT220 terminal emulation may employ TELNET [12], a remote terminal service provided by the internetworking TCP/IP protocol standards [12], as the communication channel. Likewise, a VT220 terminal emulation can also make use of rlogin [12] as the communication channel. Rlogin is in fact a remote login service included in the operating system derived from BSD UNIX. The internetworking communication is more reliable and secure than serial communication. In contrast to the RS-232-C serial communication cable, internetworking communication generally utilizes the coaxial cable as the transmission medium. Figure 2.1 also shows how VT220 terminal emulators connect to hosts through the TCP/IP internet.

2.2 Terminal Presentation

The VT220 terminal employs the ASCII alphabet (see Appendix A.1) as its character set. Yet, it still has its own auxiliary codes to support device controls. When a host downloads its data for terminals, a terminal emulation has to present the data into the corresponding action, especially making the terminal screen data visible to the Video Display Unit (VDU) [10], generally the terminal screen. Other than printable characters, the host controls the VT220 terminal by control characters and control sequences.

2.2.1 Printable Characters

The printable character is the ASCII alphabet numbered 32 through 126 (see Appendix A.1) with a total 95 characters.

2.2.2 C0 and C1 Control Characters

There are two control character sets, C0 and C1 , for the VT220 terminal. The C0 control character set is actually the control characters of the ASCII alphabet--the first 32 (numbered 0 through 31) plus the last one (number 127). If the operation mode is VT200 7-bit or VT100, all control characters downloaded by the host computer are within the C0 control character set. Table 2.1 lists C0 control characters and their descriptions.

The codes of the C1 control character set are from 128 to 159. In a similar way, the C1 control character set is specifically for the VT200 8-bit mode. When the communication channel allows eight-bit data transmission, users may select VT200 8-bit operation mode. Under such a mode, host computers will use C1 control characters instead of certain control sequences or their

Table 2.1 C0 Control Characters

Control Char.		Code	Legend
NUL	(Ctrl @)	0	-
SOH	(Ctrl A)	1	-
STX	(Ctrl B)	2	-
ETX	(Ctrl C)	3	-
EOT	(Ctrl D)	4	-
ENG	(Ctrl E)	5	Enquiry; generates answerback message
ACK	(Ctrl F)	6	-
BEL	(Ctrl G)	7	Bell; sound a beep if warning bell is on
BS	(Ctrl H)	8	Backspace; moves cursor a character left, if the cursor is already at the left margin, no action occurs
HT	(Ctrl I)	9	Horizontal tab; moves cursor to the next tab stop, or to the right margin if no further tab is present on the current line
LF	(Ctrl J)	10	Line feed; line feed/new line operation
VT	(Ctrl K)	11	Vertical tab; interpreted as LF
FF	(Ctrl L)	12	Form feed; interpreted as LF
CR	(Ctrl M)	13	Carriage return; moves cursor to left margin of current line
SO	(Ctrl N)	14	Shift out; invokes G1 character set into GL
SI	(Ctrl O)	15	Shift in; invokes G0 character set into GL
DLE	(Ctrl P)	16	-
DC1/ XON	(Ctrl Q)	17	Device control 1 or XON; causes terminal to resume transmission
DC2	(Ctrl R)	18	-
DC3/ XOFF	(Ctrl S)	19	Device control 3 or XOFF; causes terminal to stop transmitting all codes except XON and XOFF
DC4	(Ctrl T)	20	-
NAK	(Ctrl U)	21	-
SYN	(Ctrl V)	22	-
ETB	(Ctrl W)	23	-
CAN	(Ctrl X)	24	Cancel; terminates the execution of a control sequence without displaying any error message
EM	(Ctrl Y)	25	-
SUB	(Ctrl Z)	26	Substitute; terminates the execution of a control sequence and displays a reverse question mark
ESC	(Ctrl [)	27	Escape; introduces a control sequence, terminate any escape control or device control sequence
FS	(Ctrl \)	28	-
GS	(Ctrl])	29	-
RS	(Ctrl ^)	30	-
US	(Ctrl _)	31	-
RUB		127	-

Table 2.2 C1 Control Characters

Control Char.	Code	7-bit Code	Legend
-	128	-	-
-	129	-	-
-	130	-	-
-	131	-	-
IND	132	ESC D	Moves cursor down one line
NEL	133	ESC E	Moves cursor to the first position on next line
SSA	134	-	-
ESA	135	-	-
HTS	136	ESC H	Sets one tab stop at current cursor position
HTJ	137	-	-
VTS	138	-	-
PLD	139	-	-
PLU	140	-	-
RI	141	ESC M	Moves cursor up one line in the same column position
SS2	142	ESC N	Temporarily invokes G2 character set into GL for the next character
SS3	143	ESC O	Temporarily invokes G3 character set into GL for the next character
DCS	144	ESC P	Opening delimiter of a device control string
PU1	145	-	-
PU2	146	-	-
STS	147	-	-
CCH	148	-	-
MW	149	-	-
SPA	150	-	-
EPA	151	-	-
-	152	-	-
-	153	-	-
-	154	-	-
CSI	155	ESC [	Control sequence introducer
ST	156	ESC \	Closing delimiter of a device control string
OSC	157	-	-
PM	158	-	-
APC	159	-	-

introducers. To cite an example, the host computer will spare "ESC [" two characters but outbound 'CSI' instead under a VT200 8-bit mode. Even under the VT200 8-bit mode, however,

host still can render terminals the control sequence to select seven-bit C0 control character transmissions. In this case, the terminal may not accept C1 control characters until the host computer notifies it for C1 control character transmissions. Table 2.2 lists C1 control characters with equivalent seven-bit control sequences or control sequence introducer and their actions.

2.2.3 Control Sequences

A host computer might outbound control sequences as well as control characters to control its terminal devices. Unlike control character, control sequence is a character sequence introduced by a control character followed by a couple of printable characters. Technically speaking, C1 control characters are designed for the consideration of efficiency. Following is the categorization of the VT220 terminal control sequences. For some control sequences, we may find that the VT220 terminal control sequences practically contain most of the ANSI control sequences.

- Operation Mode

These control sequences are for terminal operation mode selections. As is stated above, the host computer may render a specific control sequence to notify a terminal to enter VT200 8-bit, VT200 7-bit, or VT100 mode.

Table 2.3 Select Operation Mode (Compatibility Level) Control Sequences

Sequence	Legend
ESC [61 " p	Selects VT100 mode
ESC [62;Pn " p	Selects VT200 mode Pn = 0 -- VT200 mode, 8-bit controls = 1 -- VT200 mode, 7-bit controls = 2 -- VT200 mode, 8-bit controls

- Control Character Sets

These control sequences are for C0/C1 control character set selections. The default control character set is C0, although the host may still select C1 under VT200 8-bit mode.

Table 2.4 Control Character Set Selection Control Sequences

Sequence	Legend
ESC sp F	Selects 7-bit C0 control character set transmission
ESC sp G	Selects 8-bit C1 control character set transmission

- Scrolling Region

The terminal screen may be split by a scrolling region which is defined by the control sequence downloaded by host as shown in Table 2.5. If a scrolling region is defined, the terminal will focus on this region as if it is the whole terminal screen for data displaying and ignore the other divisions outside the scrolling region. The default scrolling region is within 1 and 24. That is, the default home position is at the first column of the first row--(1,1). If a scrolling region is defined within rows Pt and Pb, then the home position would become (Pt,1).

Table 2.5 Scrolling Region Definition Control Sequence

Sequence	Legend
ESC [Pt;Pb r	Region top Pt = 1-23 Region bottom Pb = 2-24 Pb > Pt

- Tabbing

The default tab stops are at columns 9, 17, 25, 33, 41, 49, 57, 65, 73 with an eight-space interval strategy starting from the first column of a row. However, the host may reset the existing tab stops and set new ones.

Table 2.6 Tabbing Control Sequences

Sequence	Legend
ESC H	Sets TAB at current cursor position
ESC [Ps g	Clears TAB Ps = 0 -- clears TAB at current cursor position = 3 -- clears all Tabs

- Cursor Moving

These sequences are used to control the position or for storing/restoring property of the cursor (in the Microsoft Windows environment, it is sometimes the synonym of the caret) on the terminal Video Display Unit.

Table 2.7 Cursor Moving Control Sequences

Sequence	Legend
ESC [Pn A	Cursor up; moves cursor up Pn lines Pn = 0-23
ESC [Pn B	Cursor down; moves cursor down Pn lines Pn = 0-23
ESC [Pn C	Cursor right; moves cursor right Pn columns Pn = 0-79
ESC [Pn D	Cursor left; moves cursor left Pn columns Pn = 0-79
ESC [H	Cursor home; sets cursor to the home position of the screen; if a scrolling is defined, set to the corresponding home position
ESC [Pl;Pc H	Cursor addressing; moves cursor to (Pl, Pc) Pl = 1-24 (line) Pc = 1-80 (column)
ESC [Pl;Pc f	Cursor addressing; moves cursor to (Pl, Pc) (not recommended control) Pl = 1-24 (line) Pc = 1-80 (column)
ESC E	Next line
ESC D	Index
ESC M	Reverses index
ESC 7	Saves cursor
ESC 8	Restores cursor

- Graphic Rendition

This control sequence is for displaying the terminal screen data graphic rendition. All characters to be shown on the terminal screen will need to refer to current attribute status which is constantly supplied by the host end and maintained by the terminal end. In Table 2.8, the color attributes (Ps numbered above 27) are the definition of ANSI control sequences other than the VT220 terminal standard specification. The VT220 terminal emulation system proposed by this report employs these control sequences because some BBS sites employ them for colorful effect.

Table 2.8 Graphic Rendition Selection Control Sequence

Sequence	Legend
ESC [Ps;...;Ps m	Ps = 0 -- all attributes off (normal display) = 1 -- high-intensity display = 2 -- non-display = 4 -- underscore display = 5 -- blinking display = 7 -- reverse display = 22-- normal intensity = 24-- underscore off = 25-- blinking off = 27-- reverse off = 30-- foreground black (ANSI) = 31-- foreground red (ANSI) = 32-- foreground green (ANSI) = 33-- foreground yellow (ANSI) = 34-- foreground blue (ANSI) = 35-- foreground magenta (ANSI) = 36-- foreground cyan (ANSI) = 37-- foreground white (ANSI) = 40-- background black (ANSI) = 41-- background red (ANSI) = 42-- background green (ANSI) = 43-- background yellow (ANSI) = 44-- background blue (ANSI) = 45-- background magenta (ANSI) = 46-- background cyan (ANSI) = 47-- background white (ANSI)

- Line Attribute

Besides character attributes, there are also control sequences for the line attribute. On the VT220 terminal Video Display Unit, a character may be enlarged into one of the following two options: double height and double width; or double width and single height.

Table 2.9 Line Attribute Control Sequences

Sequence	Legend
ESC # 3	Double height top-half
ESC # 4	Double height bottom-half
ESC # 5	Single width line
ESC # 6	Double width line

- Erasing

These control sequences are for erasing character(s), line(s), and even the whole screen. Yet, the sequence for erasing character(s) is for VT200-related modes only.

Table 2.10 Erasing Control Sequences

Sequence	Legend
ESC [Pn X (VT200 only)	Erase character(s); erases Pn characters from the cursor position
ESC [Ps K	Erase in line Ps = 0 -- erases from the current cursor position to the end of line = 1 -- erases from the beginning of the line to the current cursor position = 2 -- erase the entire line where the cursor locates at
ESC [Ps J	Erase in screen Ps = 0 -- erases from the current cursor position to the end of screen = 1 -- erases from the beginning of the screen to the current cursor position = 2 -- erase the entire screen without moving the cursor

- Select Character Attribute

These control sequences are for the VT200-related modes only. All these activities are to set the attribute of an individual character on the terminal screen without affecting the current graphic rendition status maintained by the terminal.

Table 2.11 Character Attribute Selection Control Sequences (VT200 only)

Sequence	Legend
ESC [Ps " q	Ps = 0 -- all attributes off = 1 -- designate character as "non-erasable" = 2 -- designate character as "erasable"

- Selective Erasing

These control sequences are also for VT200-related modes only. They are similar to the erasing control sequences introduced in Table 2.10. The unique difference is that selective erasing control sequences erase only the characters with an erasable attribute as introduced in Table 2.11 while erasing control sequences erase the characters whether it is with an erasable attribute or not.

Table 2.12 Selective Erasing Control Sequences (VT200 only)

Sequence	Legend
ESC [? Ps K	Selective erase in line (erasable data only) Ps = 0 -- erases from the current cursor position to the end of line = 1 -- erases from the beginning of the line to the current cursor position = 2 -- erase the entire line which the cursor locates at
ESC [? Ps J	Selective erase in screen (erasable data only) Ps = 0 -- erases from the current cursor position to the end of screen = 1 -- erases from the beginning of the screen to the current cursor position = 2 -- erase the entire screen without moving the cursor

- Insertion and Deletion

These control sequences are used to insert or delete character(s) or line(s) from the current cursor position on the terminal screen. Unlike plain erasing, deletion makes the deleting character(s) or line(s) disappear from the existing screen data and then shifts the remaining data to the current cursor position.

Table 2.13 Insertion/Deletion Control Sequences

Sequence	Legend
ESC [Pn @	Insert character(s); inserts Pn characters at current cursor position
ESC [Pn P	Delete character(s); deletes Pn characters from the current cursor position
ESC [Pn L	Insert line(s); inserts Pn lines at current cursor position
ESC [Pn M	Delete line(s); deletes Pn lines starting from the current line

- ANSI Terminal Mode

These control sequences belong to ANSI control sequences. They are for setting/resetting terminal affiliated statuses (modes) such as the locking keyboard, insertion/replacing screen data, local echoing, and interpretation for control character LF (ASCII code 10).

Table 2.14 ANSI Terminal Mode Control Sequences

Sequence	Legend
ESC [Ps;...;Ps h	ANSI set mode Ps = 2 -- keyboard lock = 4 -- insert mode = 12-- local echo off (default) = 20-- new line
ESC [Ps;...;Ps l	ANSI reset mode Ps = 2 -- keyboard unlock (default) = 4 -- replace mode (default) = 12-- local echo on = 20-- line feed (default)

- DEC Terminal Mode

These are DEC VT220 terminal private setting/resetting control sequences for other terminal affiliated modes over and above the sequences in Table 2.14. Parameters 4, 5, and 7 are for screen operation; 6 and 25 are for cursor processing; 18 and 19 are for local printer administration. Cursor origin (relative) mode means that the cursor position is relative to the home position of the scrolling region. On the contrary, cursor absolute mode means the cursor position is relative to the home position of the whole screen.

Sequences concerning keyboard handling will be portrayed in section 2.3. Smooth scrolling is not implemented in this report as it was usually supported by hardware. This report does not support VT52 terminal mode since it is outmoded.

Table 2.15 DEC Terminal Mode Control Sequences

Sequence	Legend
ESC [? Ps;...;Ps h	DEC private set mode Ps = 1 -- cursor keys send application control sequences = 4 -- smooth scrolling mode (not implemented) = 5 -- reverse screen = 6 -- origin (relative) mode = 7 -- auto wrap on = 18-- print form feed on = 19-- print full screen = 25-- text cursor enable
ESC [? Ps;...;Ps l	DEC private reset mode Ps = 1 -- cursor keys send normal control sequences = 2 -- enter VT52 mode (not implemented) = 4 -- jump scrolling mode (default) = 5 -- normal screen = 6 -- absolute mode = 7 -- auto wrap off = 18-- print form feed off = 19-- print scrolling screen = 25-- text cursor disable
ESC =	Keypad application
ESC >	Keypad numeric (default)

- User Defined Keys (UDKs)

This control sequence is for VT220-related operation modes only. It is regarding the definitions of VT220 terminal user defined keys (UDKs). For the user's convenience, the VT220 terminal provides 15 function keys (Shift-F6 through Shift-F20) to users who can define these keys by themselves with any character (even a control character) totaling up to 256 characters. Likewise, the host may define these keys via downloading this control sequence. The outbound strings would be encoded into an unpacked hexadecimal format in order not to conflict with real on-line controls. As an example, the host will download two characters "0A" whenever it is to download the control character LF (hexadecimal ASCII code is 0A). The definitions of these UDKs are maintained by terminal end to save host end's loading.

Table 2.16 User Defined Function Key Control Sequences (VT200 only)

Sequence	Legend
ESC P Pc;Pl | key1/str1;...; keyn/strn ESC \	Pc = 0 -- clears all function keys = 1 -- clears old only where defined Pl = 0 -- lock the function keys (cannot be redefined) = 1 -- unlock the function keys (can be redefined) keyn = 17 -- F6 = 18 -- F7 = 19 -- F8 = 20 -- F9 = 21 -- F10 = 23 -- F11 = 24 -- F12 = 25 -- F13 = 26 -- F14 = 28 -- F15 = 29 -- F16 = 31 -- F17 = 32 -- F18 = 33 -- F19 = 34 -- F20 strn = encoded contents of the specified function keys

- Character Set Designation

These control sequences are for designations of VT220 terminal character sets. There are four character sets G0, G1, G2, and G3 that a VT220 terminal can use. Before any one of them is invoked as the actual character set used, GL or GR (see Table 2.18), a real alphabet set, such as ASCII or DEC special graphics (see Appendix A), needs to be designated to it first. Startup defaults are ASCII character set in G0 and G1, DEC Supplemental character set in G2 and G3, GL points to G0, GR points to G2. Due to the operating system font restriction, this VT220 terminal emulation supports ASCII and most of the DEC special graphic characters only.

Table 2.17 Character Set Designation Control Sequences

Sequence	Legend
ESC (B	Sets ASCII to G0
ESC) B	Sets ASCII to G1
ESC * B	Sets ASCII to G2
ESC + B	Sets ASCII to G3
ESC (0	Sets DEC special graphics to G0
ESC) 0	Sets DEC special graphics to G1
ESC * 0	Sets DEC special graphics to G2
ESC + 0	Sets DEC special graphics to G3

- Character Set Selection

After any of the four character sets has been designated a real alphabet set, it can be selected as the character set, GLeft or GRight, for real usage. GL actually contains C0 control characters while GR contains C1 control characters. As C1 control characters are coded greater than seven bits, there is no GR for the VT100 operation mode. In Table 2.18, functions of C1 control characters SI and SO are also listed since there are no equivalent control sequences to invoke G0 and G1 as GL.

Table 2.18 Character Set Selection Control Sequences

Sequence	Legend
SI (0x0F)	Locks shift G0 character set to GL
SO (0x0E)	Locks shift G1 character set to GL
ESC n	Locks shift G2 character set to GL (VT200 only)
ESC o	Locks shift G3 character set to GL (VT200 only)
ESC ~	Locks shift G1 character set to GR (VT200 only)
ESC }	Locks shift G2 character set to GR (VT200 only)
ESC |	Locks shift G3 character set to GR (VT200 only)
ESC N	Single shift G2 character set to GL
ESC O	Single shift G3 character set to GL

- Device Status Report

To better control a terminal device, the host might want to know the current status of a terminal as well as its peripherals. A host can use the request control sequences to inquire about the terminal device related information. Table 2.19 lists these request control sequences and their corresponding response control sequences which a terminal ought to acknowledge the host.

Table 2.19 Device Status Report Control Sequences

Sequence	Legend
ESC [5 n	Status request ==> ESC [0 n status report--terminal ready ==> ESC [3 n status report--terminal not ready
ESC [6 n	Cursor position request ==> ESC [Pl;Pc R cursor position report Pl = 1-24, Pc = 1-80
ESC [? 15 n	Request for printer status ==> ESC [? 10 n printer ready ==> ESC [? 11 n printer not ready ==> ESC [? 13 n no printer connected
ESC [? 25 n	Request for UDK status ==> ESC [? 20 n user defined keys unlocked ==> ESC [? 21 n user defined keys locked
ESC [? 26 n	Request for keyboard status ==> ESC [? 27;1 n north American keyboard

- Device Attribute Request

For the reason stated above, sometimes the host may require the terminal device attributes (DA's). These device attributes include terminal identification, terminal firmware version, and installed option number. The terminal identification is connected with the terminal operation mode. For example, there are three terminal identifications for the VT100 mode: VT100, VT101, and VT102. VT200 mode, however, has only one terminal identification--VT220.

The host computer interacts differently with diverse terminal identification. But for similar terminal identifications of an identical terminal operation mode, the host computer generally interacts with few distinctions.

Table 2.20 lists the DA request control sequences and their corresponding report control sequences. A terminal device needs to report the associated device attributes when requested. The secondary DA response is not implemented in this report as it relates to the practical VT220 terminal hardware production.

Table 2.20 DA Request Control Sequences

Sequence	Legend
ESC [c	Primary DA request
ESC [0 c	Primary DA request ==> ESC [? 62;2;6;8 c for VT220 report ==> ESC [? 1;2 c for VT100/VT100 ID report ==> ESC [? 1;0 c for VT100/VT101 ID report ==> ESC [? 6 c . for VT100/VT102 ID report
ESC Z	Identification; causes the terminal to send a primary DA response (not recommended control)
ESC [> c	Secondary DA request (not implemented)
ESC [> 0 c	Secondary DA request (not implemented) ==> ESC [> 1;Pv;Po c firmware version and option report Pv = firmware version Po = installed option number

- Terminal Adjustment

This control sequence is for the terminal VDU adjustment. Users may check if there is any video displaying problem and adjust the screen when the screen is sufficiently filled with capital characters 'E'.

Table 2.21 Terminal Adjustment Control Sequences

Sequence	Legend
ESC # 8	Fill the screen with uppercase 'E'

2.3 Keyboard Handling

The VT220 terminal has its own standard keyboard layout as shown in Figure 2.2. Since terminal emulation is based on the PC platform, there are some differentiations between this VT220 terminal emulation keyboard and the VT220 terminal standard keyboard, especially in the function keys. Figure 2.3 depicts this VT220 terminal emulation's keyboard layout which is actually the PC keyboard layout. From Figure 2.2 and Figure 2.3, we can see the VT220 terminal keyboard is quite different from the PC keyboard. Following is the description of the keyboard specification of this VT220 terminal emulation in five parts.

2.3.1 Control Characters

The VT220 terminal keyboard can generate only C0 control characters. Basically, the VT220 terminal emulation keyboard physically provides Esc, Tab, and Backspace keys (see Figure 2.3), which will generate C0 control characters ESC, HT, and BS or RUB respectively. The Backspace's key code, BS or RUB (DEL), can be defined in the system setup function.

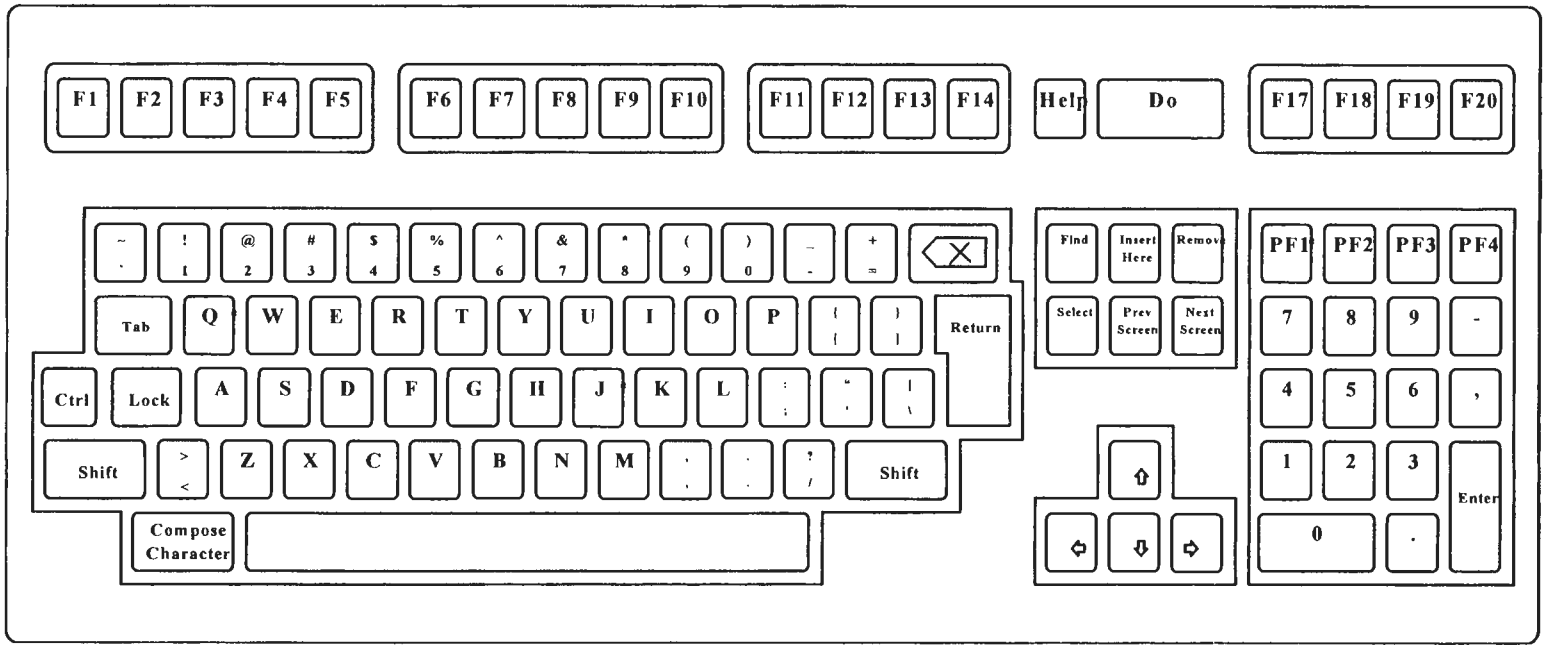

Figure 2.2 VT220 Terminal Standard Keyboard Layout

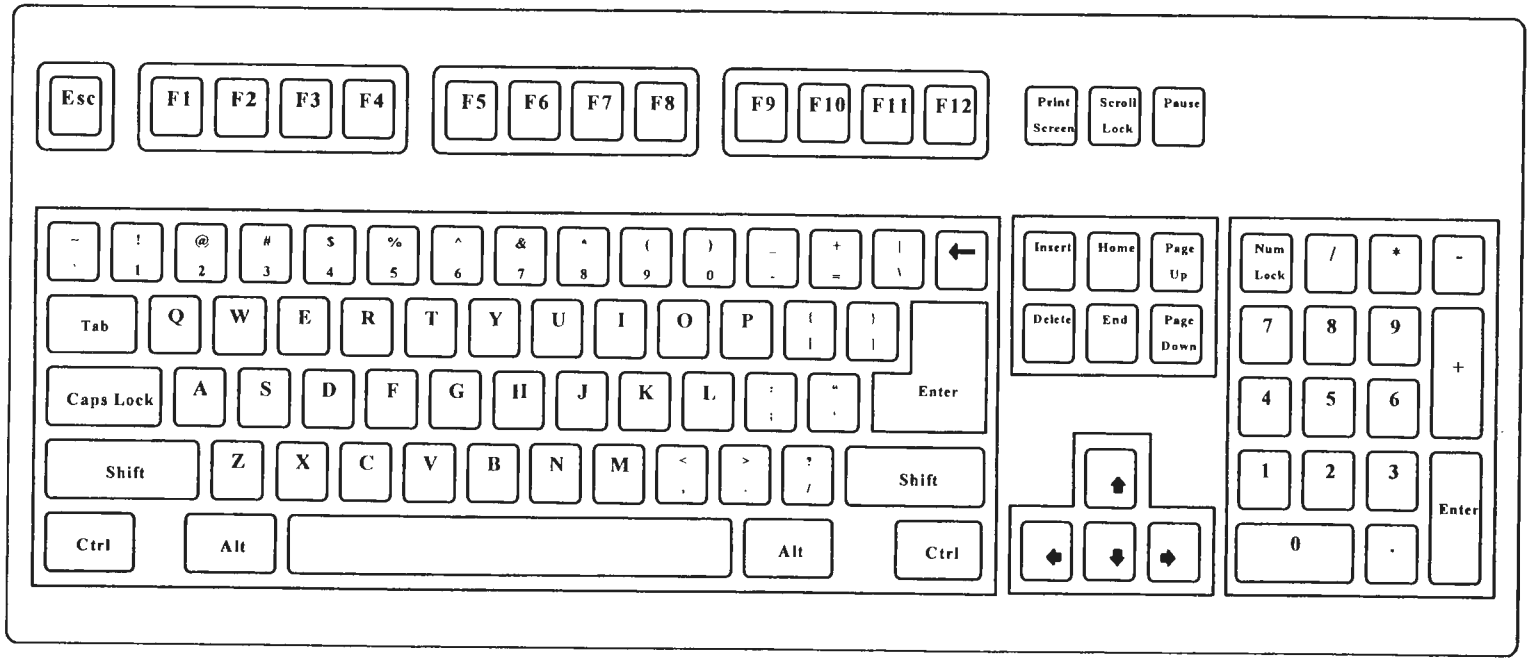

Figure 2.3 VT220 Terminal Emulation Keyboard Layout

In addition to the above four C0 control characters, there is another way to generate the C0 control characters from the keyboard: the user can press the compound key which is the combination of a certain printable character key and the 'Ctrl' key. In Table 2.1, the first column lists the corresponding printable characters of those compound keys for all C0 control character generations.

2.3.2 Printable Characters

The printable character key area is located in the lower left-hand part of both the VT220 terminal keyboard (see Figure 2.2) and the PC terminal emulation keyboard (see Figure 2.3). The VT220 terminal emulation keyboard can generate any individual printable character and inbound it to the host computer.

2.3.3 Function Keys

For the VT220 terminal keyboard, there are 20 function keys (F1 through F20; see Figure 2.2) located in the similar position as the location of the personal computer function keys (F1 through F12; see Figure 2.3). They are the greatest differences between the VT220 terminal keyboard and this VT220 terminal emulation keyboard. Since the 20 VT220 terminal function keys are not available on a personal computer keyboard, this VT220 terminal emulation uses 20 virtual keys on the screen to simulate them.

The following specifications of these VT220 terminal emulation function keys are itemized for both 20 screen virtual function keys (that is, 20 VT220 terminal function keys) and 12 physical emulation function keys.

- VT220 Terminal Function Keys F1-F5

For different operation modes, the first five function keys' actions will stay the same while the other 15 function keys are altered. Table 2.22 lists the first five VT220 terminal function keys and their processing descriptions. These keys are generally for hardware control of terminal screen, printer, and communication signal. Some of these keys might have other conducts when compounded with other keys.

Table 2.22 VT220 Terminal Function Keys F1-F5 Processing

Key	Legend
F1 (Hold Screen)	Hold current screen
F2 (Print Screen)	Have current screen data printed at local printer
* Ctrl F2	Set/reset auto print mode
F3 (SetUp)	Enter terminal system setup function (not implemented)
F4	-
F5 (Break)	Send 250-millisecond hardware Break signal from serial communication port (RS-232-C) to host
* Ctrl F5	Inbound Answerback Message (defined in system setup function) to host

* Compound key.

- VT220 Terminal Function Keys F6-F20

The last 15 VT220 terminal function keys (F6 through F20) have diversified operations which depend on the current terminal operation mode.

Table 2.23 lists these function keys and their behaviors under the VT100 operation mode. Some of them generate C0 control characters or control sequences while the others have no response. Table 2.24 lists these function keys for VT200-related operation modes. Under such modes, each of these keys sends a distinctive control sequence to the host.

Table 2.23 VT220 Terminal Function Keys F6-F20 Processing under VT100 Mode

Key	Legend
F6	-
F7	-
F8	Send control sequence ESC [J to host to erase screen data starting from current cursor position to screen bottom without changing the cursor position
F9	Send control sequence ESC [K to host to erase screen data starting from current cursor position to the end of line without changing the cursor position
F10	-
F11 (Esc)	Send C0 control character ESC (ASCII code 27) to host
F12 (Bs)	Send C0 control character BS (ASCII code 08) to host
F13 (Lf)	Send C0 control character LF (ASCII code 10) to host
F14	-
Help (F15)	-
Do (F16)	-
F17	-
F18	-
F19	-
F20	-

Table 2.24 VT220 Terminal Function Keys F6-F20 Processing under VT200-related Modes

Key	Legend
F6	Send control sequence ESC [17 ~ to host
F7	Send control sequence ESC [18 ~ to host
F8	Send control sequence ESC [19 ~ to host
F9	Send control sequence ESC [20 ~ to host
F10	Send control sequence ESC [21 ~ to host
F11 (Esc)	Send control sequence ESC [23 ~ to host
F12 (Bs)	Send control sequence ESC [24 ~ to host
F13 (Lf)	Send control sequence ESC [25 ~ to host
F14	Send control sequence ESC [26 ~ to host
Help (F15)	Send control sequence ESC [28 ~ to host
Do (F16)	Send control sequence ESC [29 ~ to host
F17	Send control sequence ESC [30 ~ to host
F18	Send control sequence ESC [32 ~ to host
F19	Send control sequence ESC [33 ~ to host
F20	Send control sequence ESC [34 ~ to host

Moreover, the user defined keys (UDKs; Shift-F6 to Shift-F20) could be defined either by the host as depicted in Table 2.16 or by users through the system setup function when they are at unlocked status. Yet, they can not be cleared or redefined by the host when they are locked. A host might therefore request the terminal for UDK status by applying a device status report control sequence (see Table 2.19) to see if it is able to change the UDKs' definitions. The UDK locked/unlocked status is maintained by the terminal end and can be altered at the user's convenience through the system setup function.

- VT220 Terminal Emulation Function Keys F1-F12

Because personal computer users customarily use the numeric keypad, this VT220 terminal emulation employs the 12 personal computer function keys to simulate the VT220 keypad with application mode as illustrated in Table 2.31 except the ',' and 'Enter' keys (see Figure 2.2 keypad portion). These function keys may also be redefined in the configuration function provided by this emulation system.

Table 2.25 VT220 Terminal Emulation Function Keys F1-F12 Processing

Key	VT220 Keypad	Legend
F1	1	Send control sequence ESC O q to host
F2	2	Send control sequence ESC O r to host
F3	3	Send control sequence ESC O s to host
F4	4	Send control sequence ESC O t to host
F5	5	Send control sequence ESC O u to host
F6	6	Send control sequence ESC O v to host
F7	7	Send control sequence ESC O w to host
F8	8	Send control sequence ESC O x to host
F9	9	Send control sequence ESC O y to host
F10	0	Send control sequence ESC O p to host
F11	.	Send control sequence ESC O m to host
F12	-	Send control sequence ESC O l to host

2.3.4 Editing Keys and Cursor Keys

In this VT220 terminal emulation, the editing keys and the cursor keys are located between the printable character key area and the keypad area on both the VT220 terminal keyboard and the personal computer keyboard. The cursor keys, in the lower area, are the four arrow keys which control the movement of screen cursor; and the editing keys, in the upper area, are the six keys that are located above the four cursor keys. Following are the itemized specifications of the editing keys and the cursor keys of a VT220 terminal keyboard.

- Editing Keys

From Figure 2.2 and Figure 2.3, we can see that the layouts of VT220 terminal editing keys (the six keys: Find, Insert Here, Remove, Select, Prev Screen, and Next Screen) and PC editing keys (Insert, Home, Page Up, Delete, End, and Page Down respective to VT220 terminal editing keys) are almost the same. Nevertheless their definitions are quite different from each other. In this VT220 terminal emulation, the editing keys follow the VT220 terminal definitions although it employs PC keyboard. Table 2.26 lists the corresponding behaviors of the six editing keys and their equivalent PC remapping keys. Since the editing keys have the remapping problem, this terminal emulation enables users to redefine them by way of system configuration.

Table 2.26 Editing Keys Processing

VT220 Key	PC Key	Legend
Find	Insert	Send control sequence ESC [1 ~ to host
Insert Here	Home	Send control sequence ESC [2 ~ to host
Remove	Page Up	Send control sequence ESC [3 ~ to host
Select	Delete	Send control sequence ESC [4 ~ to host
Prev Screen	End	Send control sequence ESC [5 ~ to host
Next Screen	Page Down	Send control sequence ESC [6 ~ to host

- Cursor Keys

The VT220 terminal cursor keys, without the remapping problem of the editing keys, are definitely the same as PC cursor keys. Since VT220 terminal screen cursor is controlled by certain control sequences (see Table 2.7), a keyboard cursor key has to inbound the corresponding control sequence for cursor domination when pressed. There are two modes for the cursor key processing. The control sequences depicted in Table 2.27, which are actually the first four control sequences without Pn value introduced in Table 2.7, are for the normal mode. In the contrast, Table 2.28 lists the cursor key processing for application mode.

Table 2.27 Normal Cursor Key Processing

Cursor Key	Legend
↑	Send control sequence ESC [A to host
↓	Send control sequence ESC [B to host
→	Send control sequence ESC [C to host
←	Send control sequence ESC [D to host

Table 2.28 Application Cursor Key Processing

Cursor Key	Legend
↑	Send control sequence ESC O A to host
↓	Send control sequence ESC O B to host
→	Send control sequence ESC O C to host
←	Send control sequence ESC O D to host

2.3.5 Keypad

The VT220 terminal keyboard keypad is different from the PC keypad in four PF keys, ',' key and '-' key. For a better explanation, the following illustrates the division of the VT220

keypad into two parts.

- PF Function Keys

For the VT220 keypad, there are four PF function keys at the top of the VT220 terminal keyboard keypad (see Figure 2.2) which are in fact the same locations as PC keypad keys Num Lock, '/', '*', and '-' (see Figure 2.3). Table 2.29 lists these four VT220 PF function keys with their equivalent PC keypad keys and descriptions. The behaviors of the PF keys are always the same, no matter what the keypad mode is.

Table 2.29 Keypad PF Keys Processing

VT220 Key	PC Key	Legend
PF1	Num Lock	Send control sequence ESC O P to host
PF2	/	Send control sequence ESC O Q to host
PF3	*	Send control sequence ESC O R to host
PF4	-	Send control sequence ESC O S to host

- Other Keypad Keys

For the other keypad keys, this VT220 terminal emulation uses PC keypad key '+' and a compound key Shift '+" to simulate VT220 keypad keys '-' and ',' correspondingly because these two keys together occupy the same location as the PC keypad key '+' (see Figure 2.2 and Figure 2.3). Furthermore, the VT220 keypad does not have the key '+', also the PC keypad does not have the key ','.

Like the VT220 cursor keys, there are two keypad modes, numeric and application, for a VT220 terminal keyboard. All the VT220 keypad keys except the PF function keys perform in a different way for the alternative keypad mode. Table 2.30 and Table 2.31 cite these VT220 keypad keys with their equivalent PC keypad keys and their carriages under numeric keypad

mode and application keypad mode respectively.

Table 2.30 Numeric Keypad Processing

VT220 Key	PC Key	Legend
0	0 (Ins)	Send character '0' to host
1	1 (End)	Send character '1' to host
2	2 (↓)	Send character '2' to host
3	3 (Pg Dn)	Send character '3' to host
4	4 (←)	Send character '4' to host
5	5	Send character '5' to host
6	6 (→)	Send character '6' to host
7	7 (Home)	Send character '7' to host
8	8 (↑)	Send character '8' to host
9	9 (Pg Up)	Send character '9' to host
.	. (Del)	Send character '.' to host
-	+	Send character '-' to host
,	* Shift +	Send character ',' to host
Enter	Enter	Send C0 control character CR (ASCII code 13) to host

* Compound key.

Table 2.31 Application Keypad Processing

VT220 Key	PC Key	Legend
0	0 (Ins)	Send control sequence ESC O p to host
1	1 (End)	Send control sequence ESC O q to host
2	2 (↓)	Send control sequence ESC O r to host
3	3 (Pg Dn)	Send control sequence ESC O s to host
4	4 (←)	Send control sequence ESC O t to host
5	5	Send control sequence ESC O u to host
6	6 (→)	Send control sequence ESC O v to host
7	7 (Home)	Send control sequence ESC O w to host
8	8 (↑)	Send control sequence ESC O x to host
9	9 (Pg Up)	Send control sequence ESC O y to host
.	. (Del)	Send control sequence ESC O m to host
-	+	Send control sequence ESC O l to host
,	* Shift +	Send control sequence ESC O n to host
Enter	Enter	Send control sequence ESC O M to host

* Compound key.

2.3.6 Other Key

In Figure 2.2, we can see there is a key named Compose Character located at the lower left-hand corner of the printable character key area. Like the name, Compose Character is composing with another key. For example, the control sequence ESC [L will be sent if an end user presses Compose Character then presses Insert Here. Other than Insert Here, the Compose Character key can also compound with Remove, and the keys '1' through '0' from left to right during the printable character key area.

Since it is rarely used, however, this VT220 terminal emulation does not provide this key for the user.

2.4 Local Printer

The standard local printer equipment for VT220 terminals is the serial printer. A serial printer is a printer that interacts with the terminal equipment through the (RS-232-C asynchronous) serial communication port. Since this VT220 terminal emulation is based upon the PC platform whose standard printer port is actually the parallel one, it supports the parallel printer function instead of the serial printer.

There are two printing modes for the VT220 local printer. When it is under printer-controller mode, all outbound data from host ought to be directly sent to the local printer device without terminal presentations. When under auto print mode, the outbound data for a line (row) ought to be displayed on the terminal VDU first and should not be sent to the local printer until the screen cursor is moved to another line.

Apart from printer-controller and auto print, there are also two printer functions for the

local printer that the host might outbound the corresponding control sequence to print the data of whole screen or just a line at which the cursor is located. Table 2.32 lists the related control sequences and their descriptions for these four printing functions.

Table 2.32 Printer Control Sequences

Sequence	Legend
ESC [i	Print screen
ESC [0 i	Print screen
ESC [5 i	Enter printer-controller mode
ESC [4 i	Exit printer-controller mode
ESC [? 1 i	Print cursor line
ESC [? 5 i	Enter auto print mode
ESC [? 4 i	Exit auto print mode

CHAPTER 3

PRELIMINARY ANALYSIS AND DESIGN

This chapter is to discuss the preliminary analysis and design for developing this VT220 terminal emulation system before practical software implementation. It employs some diagrams and structure charts for the structured analysis and software structure designing [13], [14], [15].

In this VT220 terminal emulation system, the module for RS-232-C asynchronous serial communication is separated and made into a Dynamic Linking Library (DLL) [2], [3], [4]. The following figures illustrate the preliminary analysis and design for this VT220 terminal emulation system with terminal emulation and RS-232-C communication DLL together or separated.

3.1 Memory Map

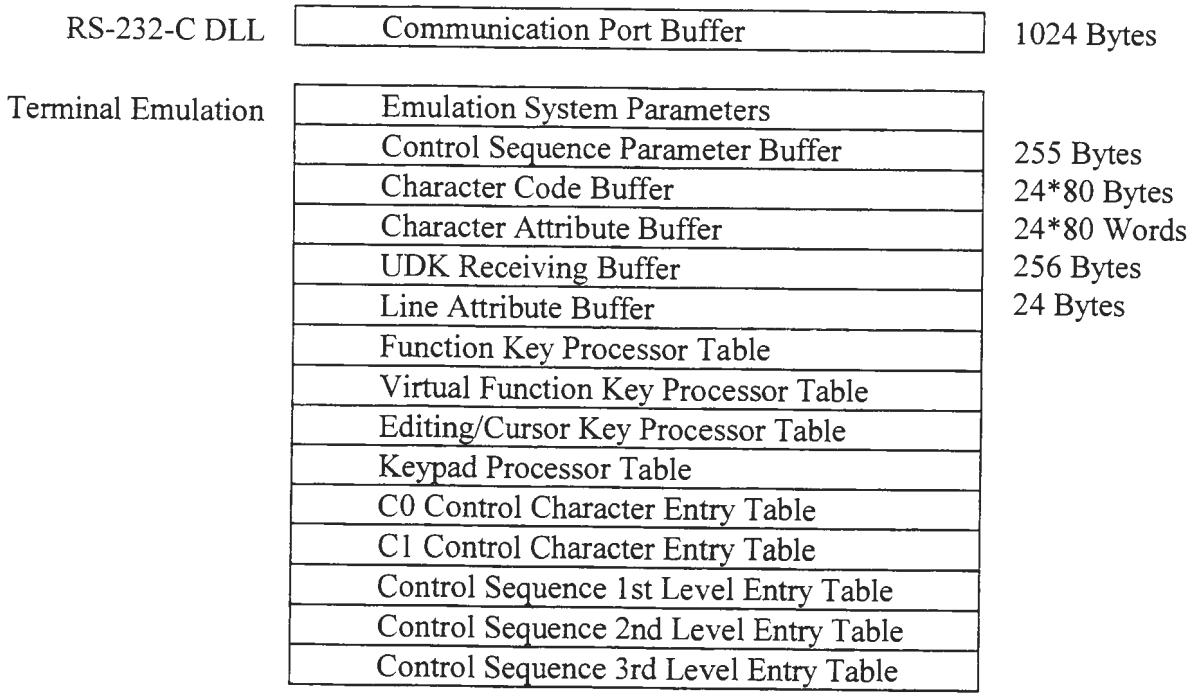

Figure 3.1 Memory Map for VT220 Terminal Emulation System

3.2 System Block Diagram

3.2.1 Terminal Emulation

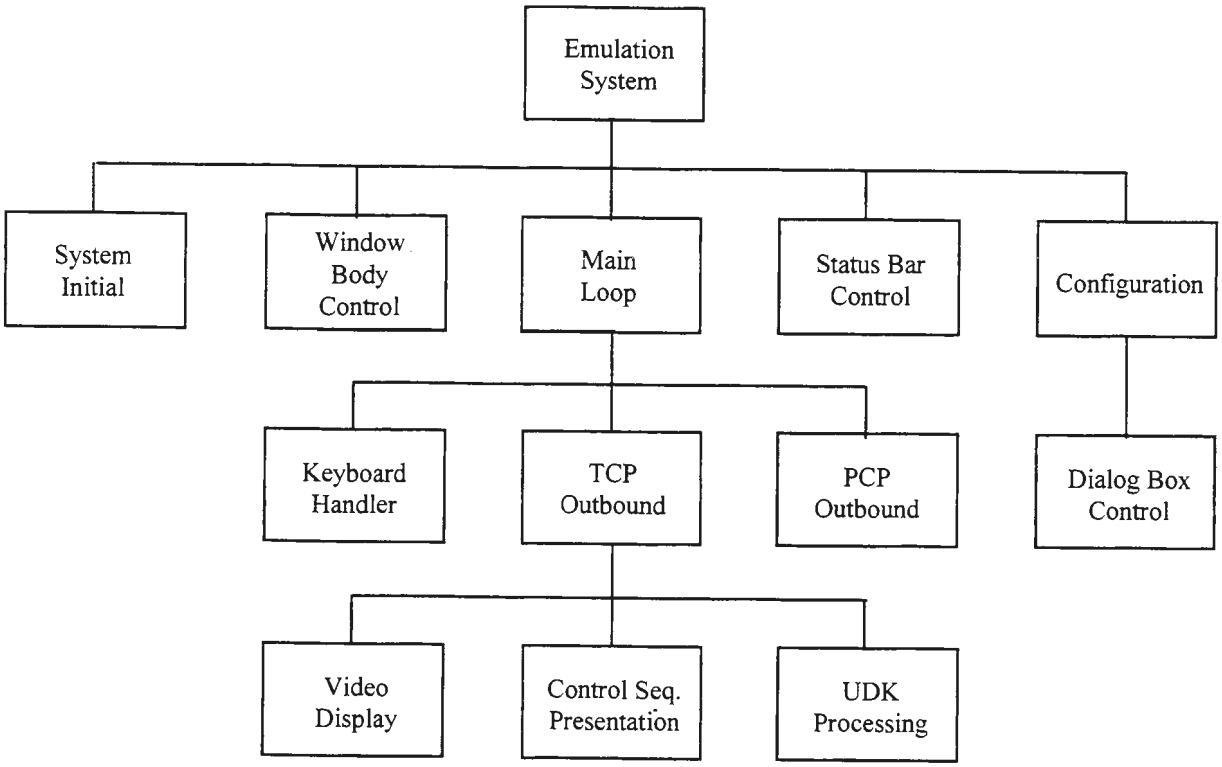

Figure 3.2 Terminal Emulation System Block Diagram

3.2.2 RS-232-C Communication DLL

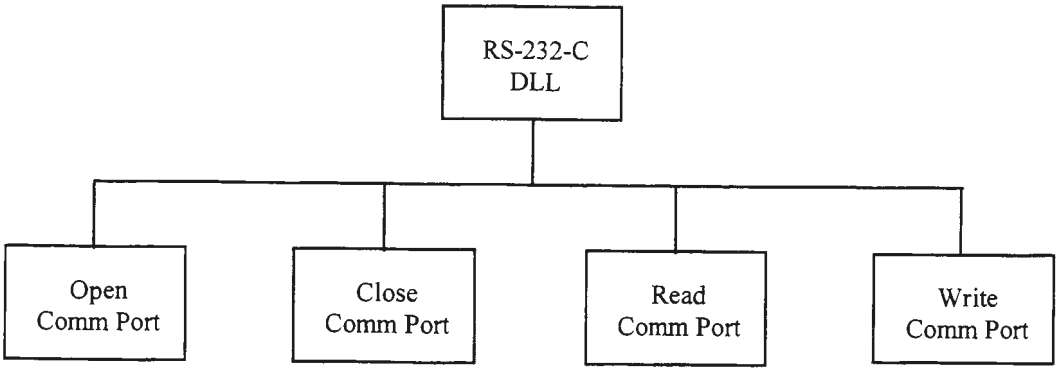

Figure 3.3 RS-232-C DLL System Block Diagram

3.3 Data Flow

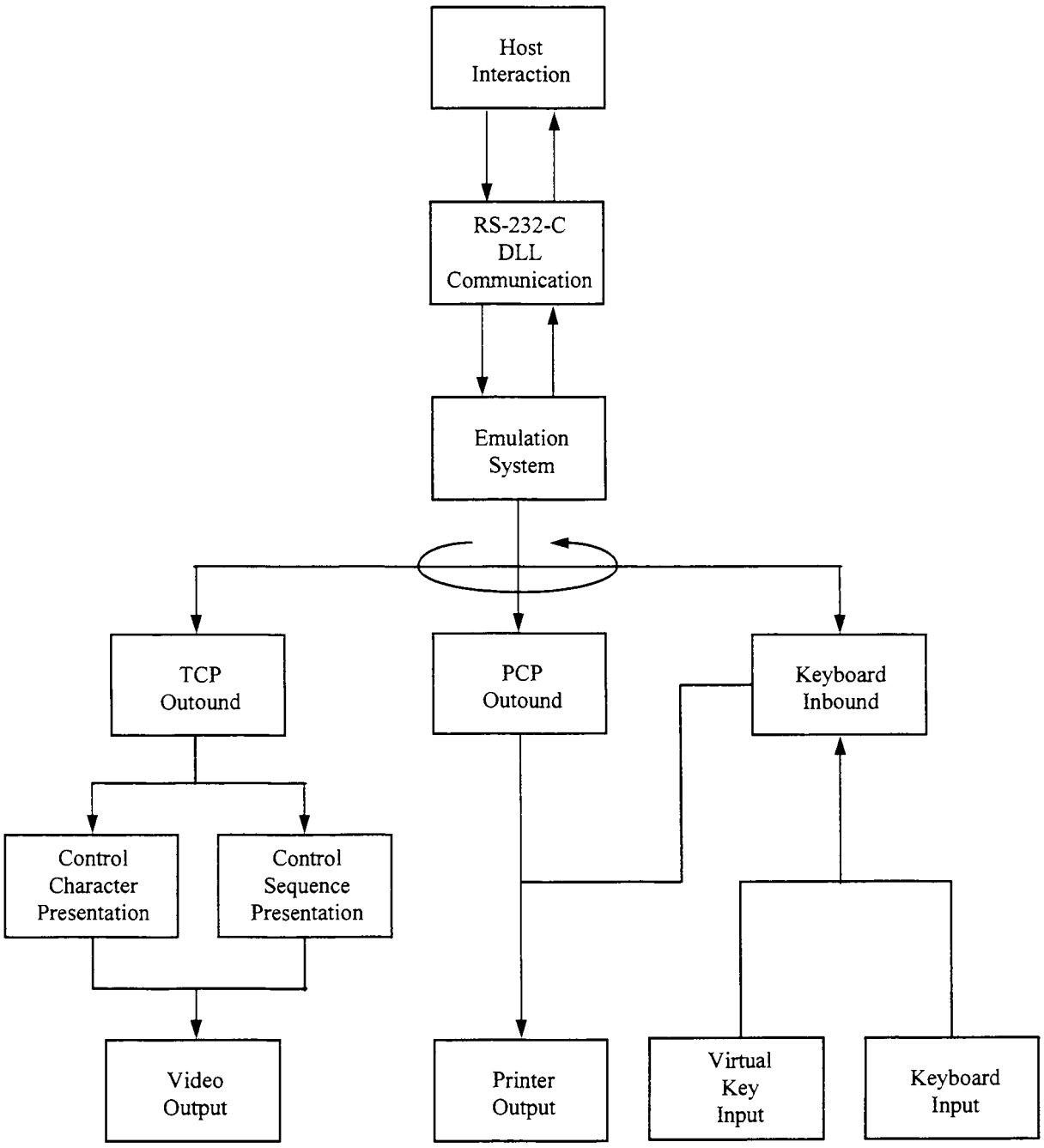

Figure 3.4 VT220 Terminal Emulation System Data Flow Diagram

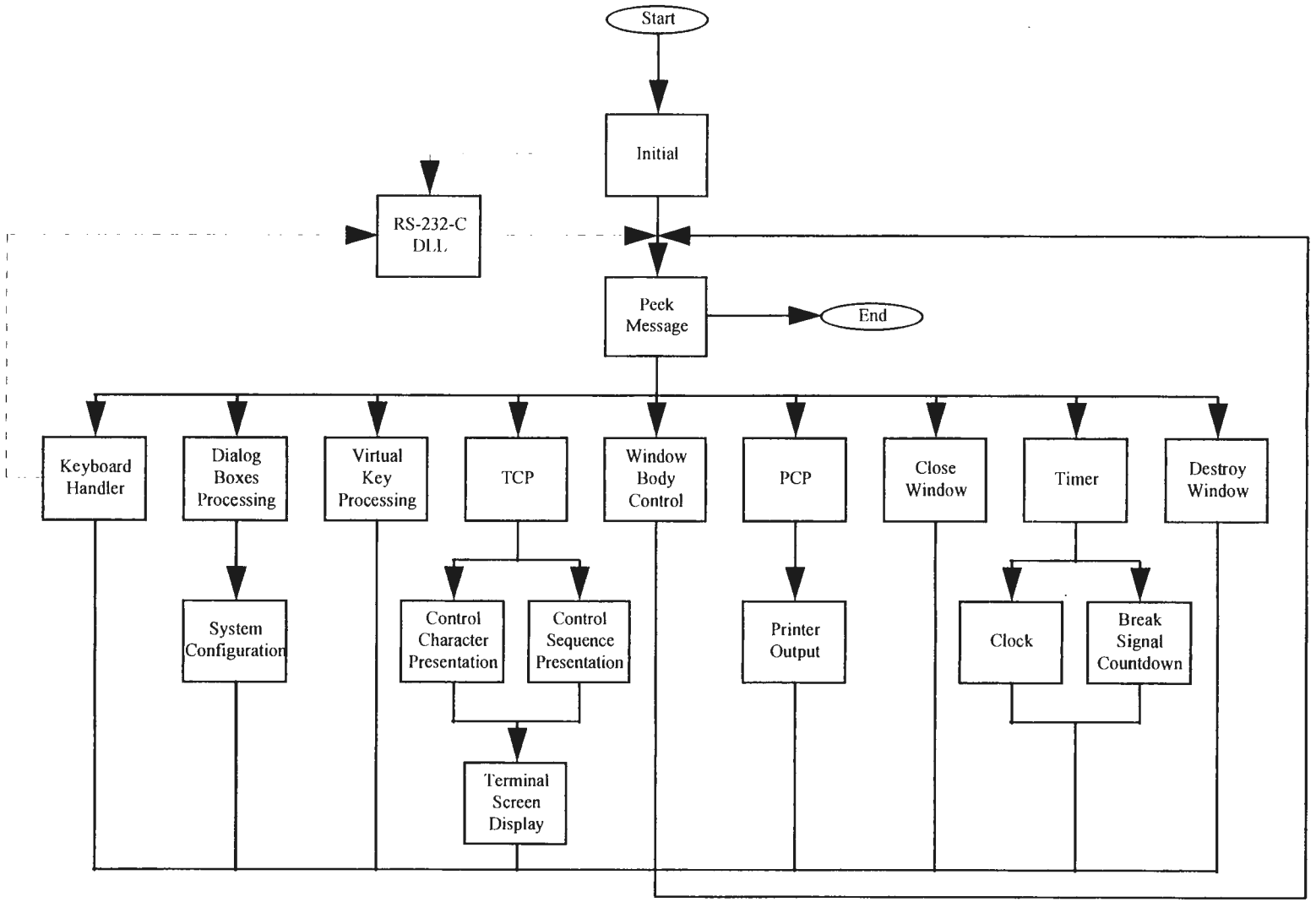

Figure 3.5 VT220 Terminal Emulation System Control Flow Diagram

3.5 Data Structures

The following figures are related to the most important data structures designed and used in this VT220 terminal emulation system. Each structure might have one or more substructures. The structures shown in these figures are declared in C programming language. There is also a comment for each statement to explain the designing consideration.

Figure 3.6 shows the primary and primitive structure for the most important parameters of this VT220 terminal emulation system. It contains all the other substructures shown in Figure 3.7 through Figure 3.12.

```
struct HostTag {                                    /* structure for emulator system parameters  */
        struct TermTag      Term;                   /* structure for terminal related parameters  */
        struct KeyBTag      Kbd;                    /* structure for keyboard related parameters */
        struct CommTag      Comm;                   /* structure for communication setting       */
        struct PrnTag       Prt;                    /* structure for printer related parameters  */
        struct CtrlTag      Ctrl;                   /* structure for miscellaneous control       */
};
```

Figure 3.6 Primary Structure for Emulation System

```
struct CtrlTag {                                    /* structure for miscellaneous control       */
        BYTE bStatusLine    : 1;                    /* 1-bit control for status line on/off      */
        BYTE bTimerMode     : 1;                    /* 1- bit control for timer/clock mode       */
        BYTE bUDKVisible    : 1;                    /* 1- bit control for UDK visible on screen  */
};
```

Figure 3.7 Structure for Window Miscellaneous Control

```
struct TermTag {                        /* structure for terminal related parameters  */
        BYTE  Type;                     /* terminal type                              */
        BYTE  TermId;                   /* terminal ID                                */
        BYTE  HostCode;                 /* host internal code                         */
        BYTE  BgColor;                  /* screen background color                    */
        struct TermPreferTag {          /* structure for terminal preferences         */
                BYTE bLineWrap   :1;    /* 1-bit control for auto wrap                */
                BYTE bNewLine    :1;    /* 1-bit control for LF interpretation        */
                BYTE bWarnBell   :1;    /* 1-bit control for warning bell             */
                BYTE bInverse    :1;    /* 1-bit control for screen inverse property  */
                BYTE bCurBlkUnd :1;     /* 1-bit control for cursor shape             */
                BYTE bCurBlink   :1;    /* 1-bit control for cursor blinking status   */
                BYTE bColumn     :1;    /* 1-bit control for 80/132 column modes      */
                BYTE bLocalEcho  :1;    /* 1-bit control for local echoing            */
        } Prefer;
        BYTE  TabSet[17];               /* bitwise tab stops control                  */
        struct AnsBTag {                /* structure for answerback message           */
                BYTE  Msg[21];          /* answerback message string                  */
                BYTE  AutoAnswer;       /* auto answerback control                    */
        } AnsBack;
        struct FontTag      FontType;   /* structure for screen character font        */
};
```

Figure 3.8 Structure for Terminal Related Parameters

```
struct CommTag {                        /* structure for communication setting        */
        BYTE   ComPort;                 /* COM port employed                          */
        BYTE   BaudRate;                /* baud rate setting                          */
        BYTE   Parity;                  /* parity check: none/even/odd/space/mark     */
        BYTE   DataBit;                 /* data bit selection: 5/6/7/8                */
        BYTE   StopBit;                 /* stop bit selection: 1/1.5/2                */
        BYTE   CD;                      /* carrier detect control                     */
};
```

Figure 3.9 Structure for Communication Setting

```
struct KeyBTag {                        /* structure for keyboard related parameters */
        struct KeyPreferTag {           /* structure for keyboard preferences        */
                BYTE bMarginBell   : 1; /* 1-bit control for margin bell             */
                BYTE bBkSpDel      : 1; /* 1-bit control for Backspace key code      */
                BYTE bKeypadMode   : 1; /* 1-bit control for keypad modes            */
                BYTE bCursorKey    : 1; /* 1-bit control for cursor key modes        */
        } Prefer;
        BYTE UDKLock;                   /* UDK locking status                        */
        int     UDKKeyLen[15];          /* UDK data length                           */
        BYTE UDKKey[15][257];           /* UDK data                                  */
        int     DefKeyLen[18];          /* keyboard F1-F12/editing key data length   */
        BYTE DefKey[18][257];           /* keyboard F1-F12/editing key definitions   */
};
```

Figure 3.10 Structure for Keyboard Related Parameters

```
struct PrnTag {                         /* structure for printer related parameters  */
        BYTE  PrtName[21];              /* printer information                       */
        BYTE  Mode;                     /* print mode: normal/controller/autoprint   */
        BYTE  Terminator;               /* printing terminator: none/formfeed (FF)   */
};
```

Figure 3.11 Structure for Printer Related Parameters

```
struct FontTag {                        /* structure for screen character font       */
        LOGFONT  LogFont;               /* font information                          */
        LONG     FontColor;             /* font color                                */
};
```

Figure 3.12 Structure for Screen Character Font

CHAPTER 4
SOFTWARE IMPLEMENTATION

This report employs the Microsoft Windows Software Development Kit (SDK) [2], [3], [4], which is actually based upon the C programming language [16], to develop the DEC VT220-type terminal emulation system in the Microsoft Windows environment.

There are many tasks for a terminal emulation system. Basically, they are terminal data presentation and protocol interpretation, input device handling and mapping, output device manipulations and outputting, communication lines flow control and handshaking, and so forth. Each part deals with its related tasks which might be very difficult or sophisticated.

The following modules are the software implementation for this VT220 terminal emulation system. These fall into two parts for briefing: the terminal emulation system and the RS-232-C DLL communication module.

4.1 Terminal Emulation System

4.1.1 Emulator Module

This module is the main module which contains the WinMain() function. The WinMain() function of SDK is like the main() function of the C programming language. The primary task of this function is to use a loop to:

(1) peek and dispatch diverse messages from system and application message queues; and

(2) poll the RS-232-C communication port if there is no message peeked.

It employs Windows API PeekMessage() function [3] in consideration of the non-preemptive multi-tasking feature of Windows 3.x environments.

The significant messages that need more additional efforts with which to process are:

(1) the message, notified by the communication port, accompanied with the outbound data which will actually be forwarded to the TCP and PCP modules; and

(2) the various key messages accompanied with ASCII and scan codes which will actually be forwarded to the keyboard module.

4.1.2 Initial Module

The initial module prepares for everything before processing. Following are the descriptions of the primary tasks of this module.

- Register for all window classes (including UDK virtual keys).
- Display all windows (including UDK virtual keys).
- Allocate for code buffer and attribute buffer.
- Initiate important global variables.
- Initiate RS-232-C communication port.
- Initiate windows for status line.

After initiation, the window for this VT220 terminal emulation would appear for users. The designed outlook for this window is shown as Figure 4.1.

Figure 4.2 through Figure 4.5 are for the diversified options of different menu items. As in Figure 4.2, there are two options, the Long Break and the Short Break with the shortcut keys Ctrl F9 and Shift F9 which respectively indicate 3.5 seconds and 250 milliseconds of RS-232-C hardware BREAK signal. The Short Break function is in fact the same as the F5 function key of the VT220 terminal(see Figure 2.2 and Table 2.22). The Long Break function is provided

because some users might need it for additional operation purposes.

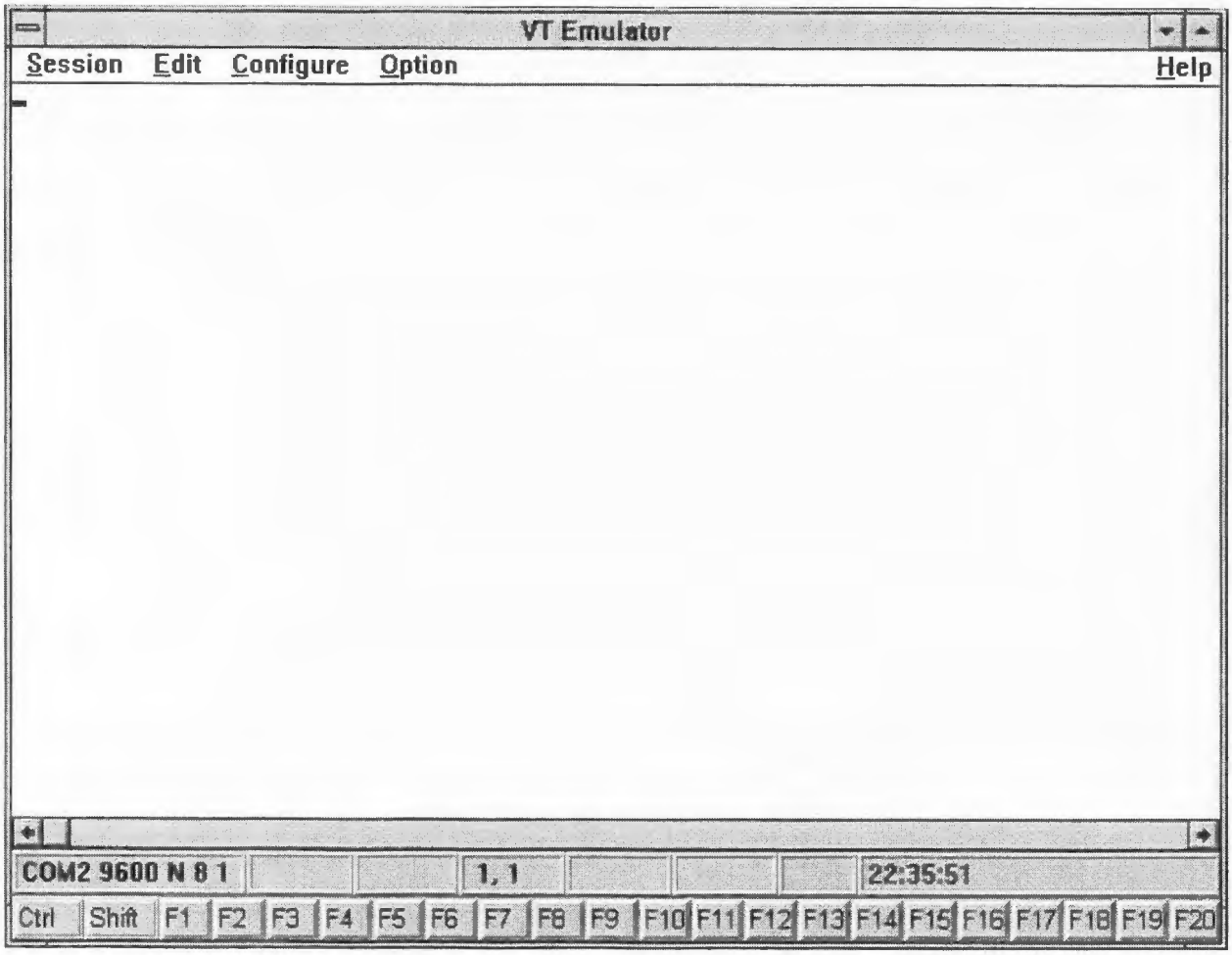

Figure 4.1 VT220 Terminal Emulation System Window

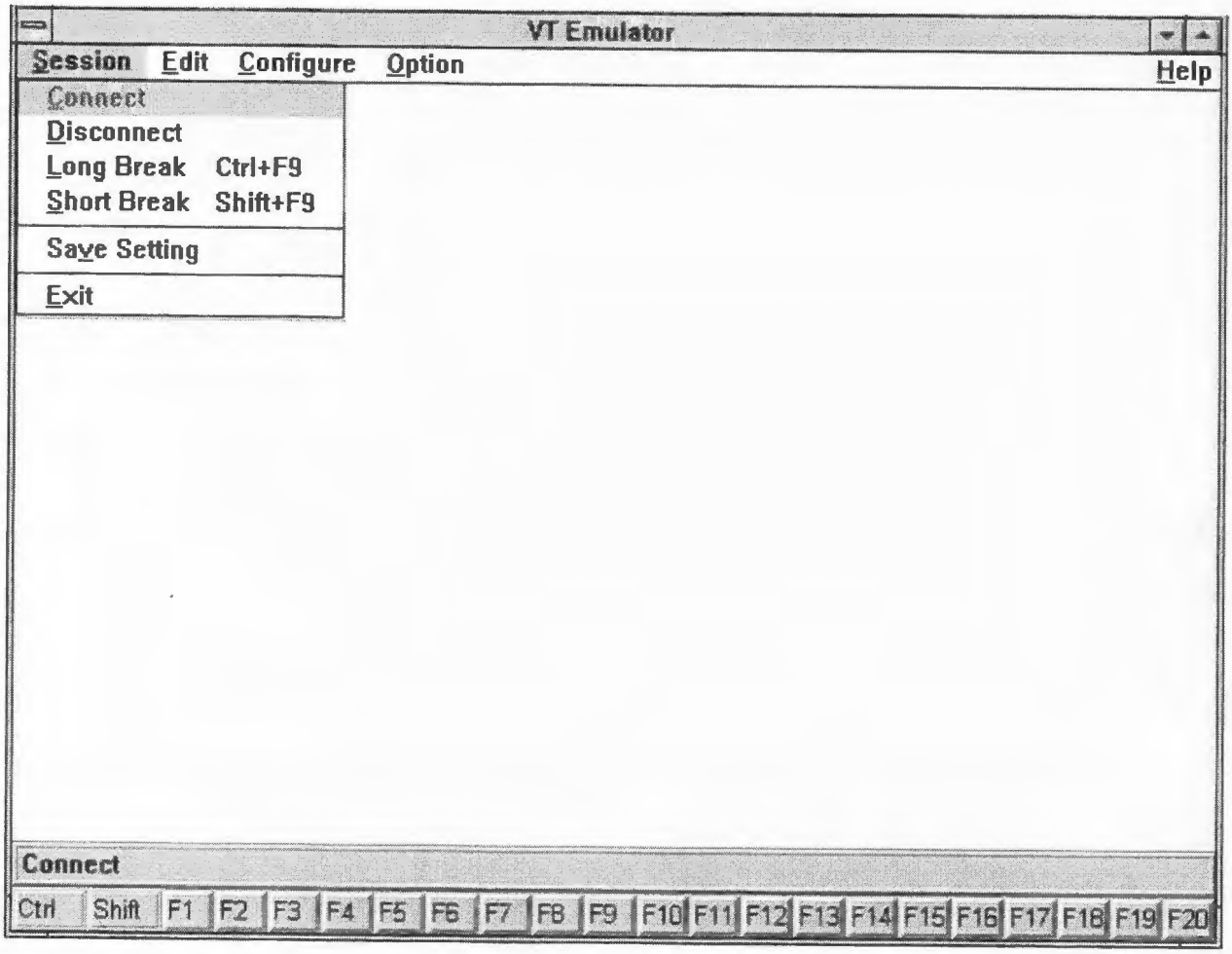

Figure 4.2 Designed Options of Session Menu Item

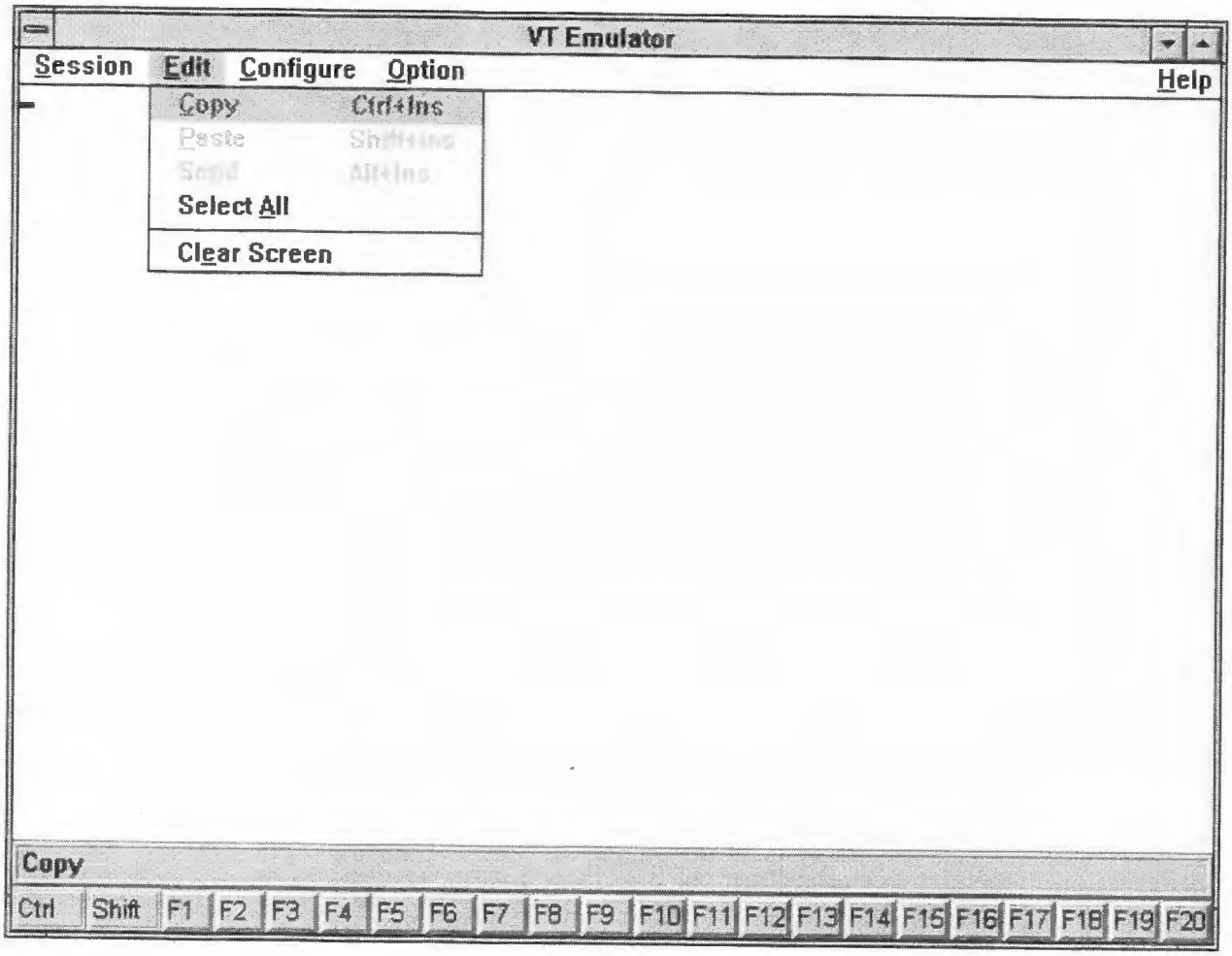

Figure 4.3 Designed Options of Edit Menu Item

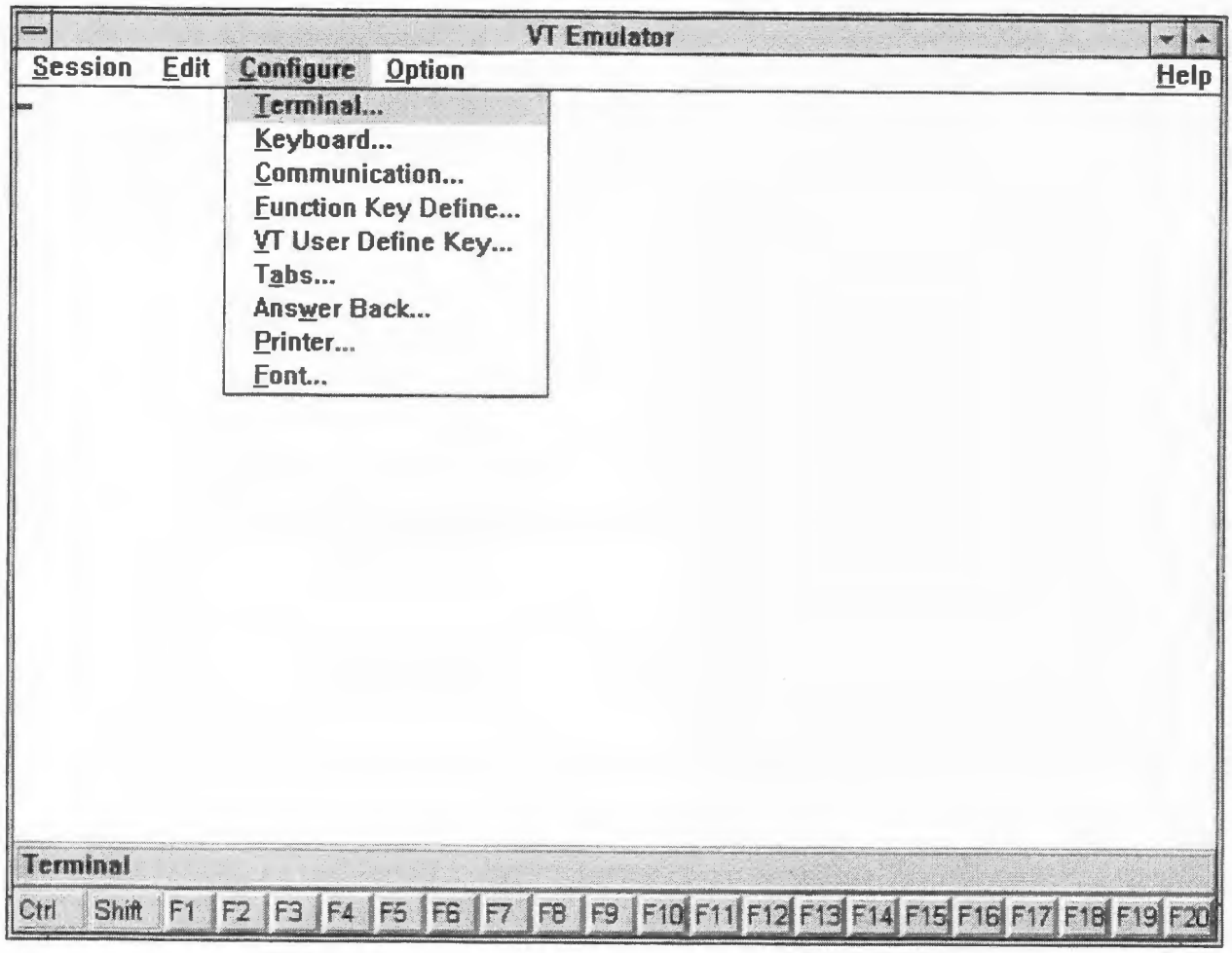

Figure 4.4 Designed Options of Configuration Menu Item

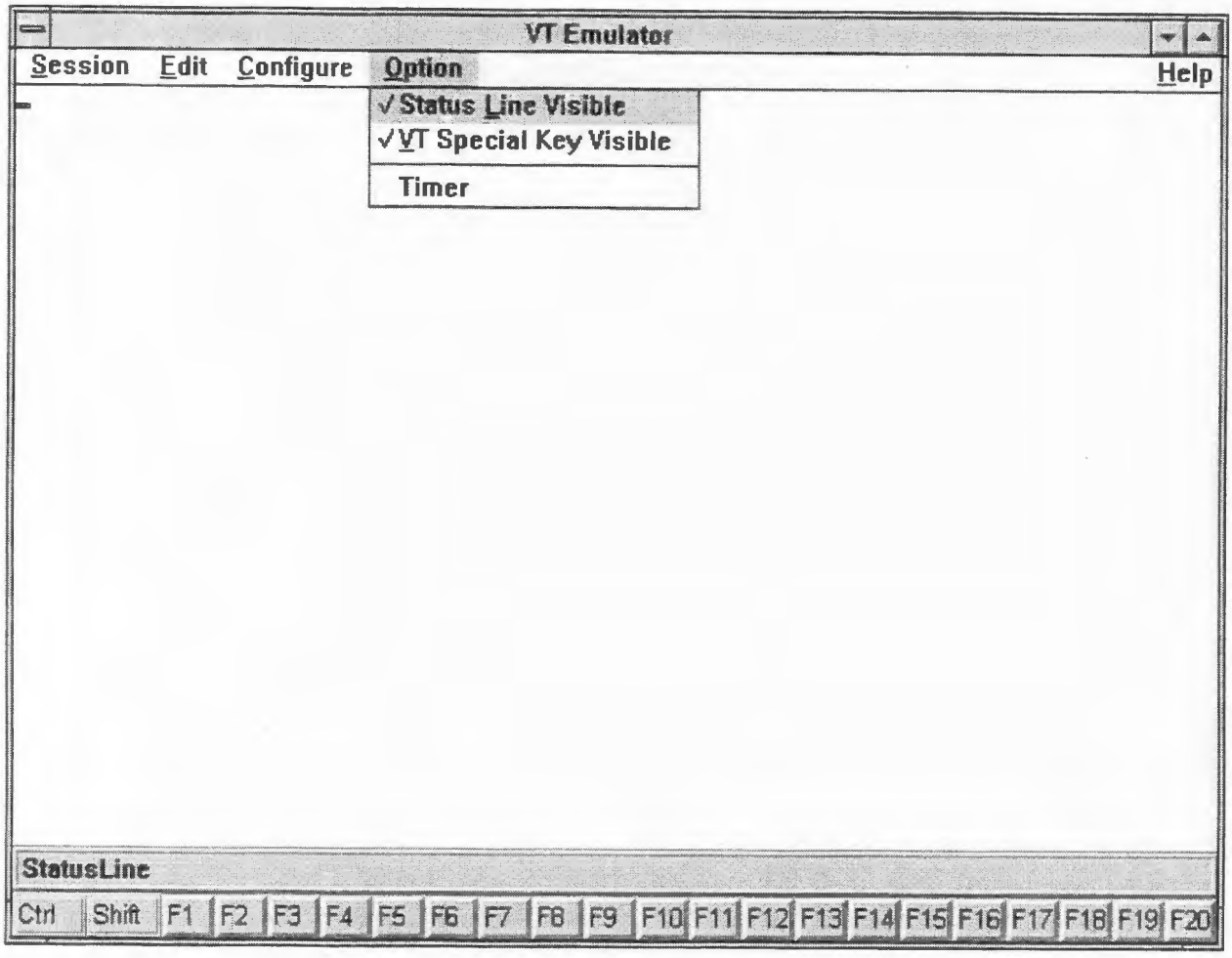

Figure 4.5 Designed Options of Option Menu Item

4.1.3 System Configuration Module

There are many complicated system settings for the VT220 terminal which always confuses the user. Some of the traditional terminals did not even provide a detailed user's manual and on-line help information for system setup. Therefore, a user who lacked the proper concept and knowledge of the terminal equipment was unable to configure the system.

This module which is based upon the Windows Graphic User Interface (GUI) characteristic provides several dialog boxes for the system configuration in brief and easily understood designs. The configuration contains terminal preferences, keyboard preferences and function key (including user defined keys) settings, communication port setting, printer setup, user custom preferences, and so on (see Figure 4.4). Through the interfaces provided by this module, users can complete the system configuration more easily than when using the traditional terminal.

Figure 4.6 through Figure 4.14 are the dialog boxes corresponding to each system setup.

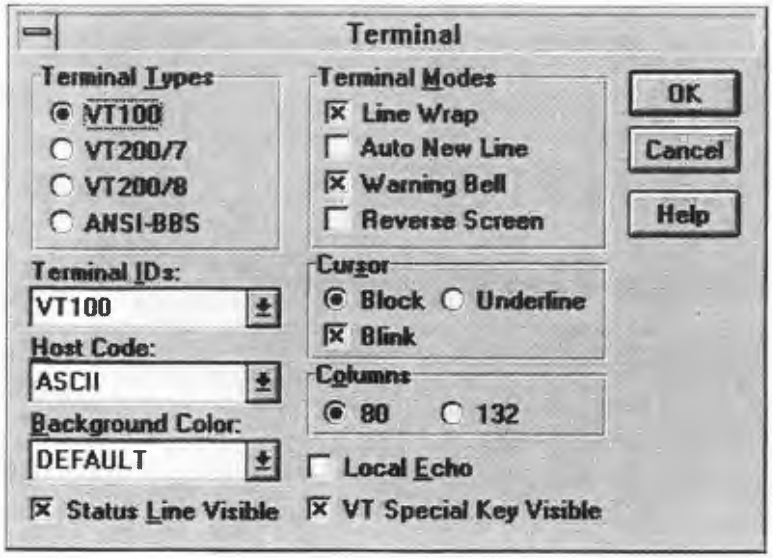

Figure 4.6 Designed Dialog Box for Terminal Preferences

Figure 4.7 Designed Dialog Box for Keyboard Preferences

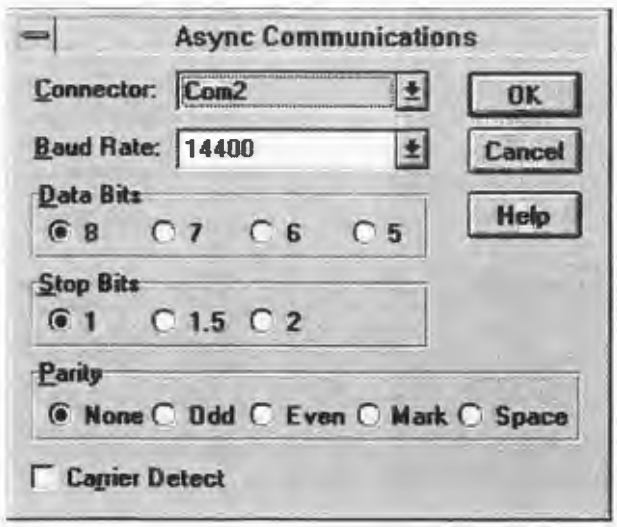

Figure 4.8 Designed Dialog Box for Communication Port Setting

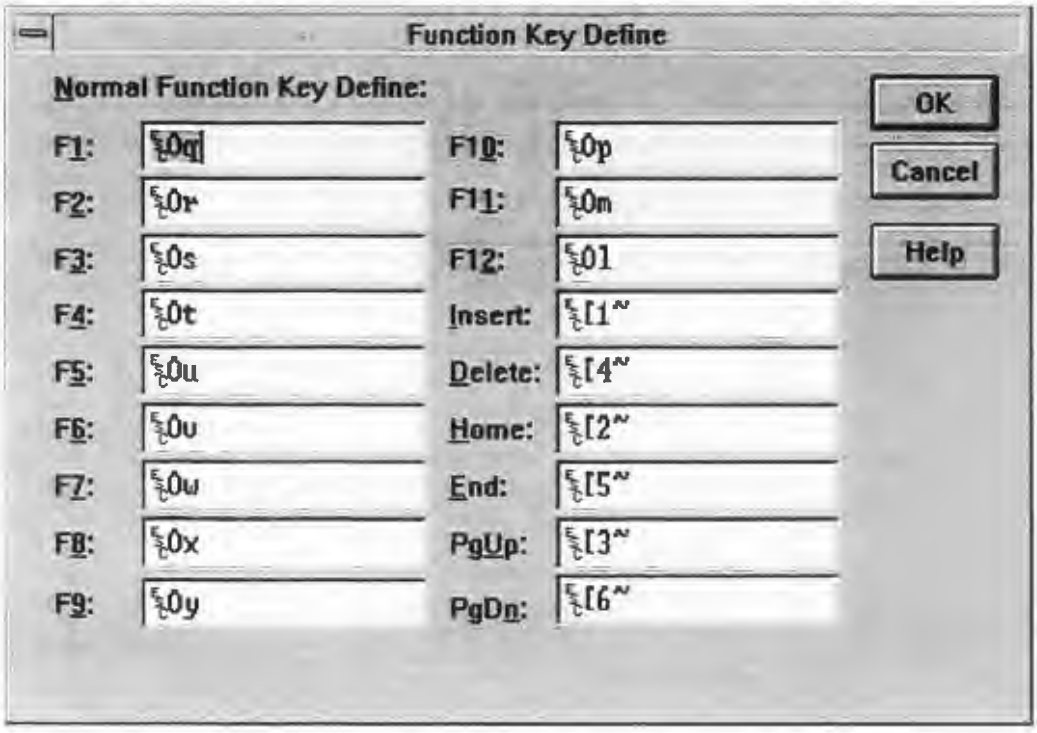

Figure 4.9 Designed Dialog Box for PC Function Key and Editing Key Definitions

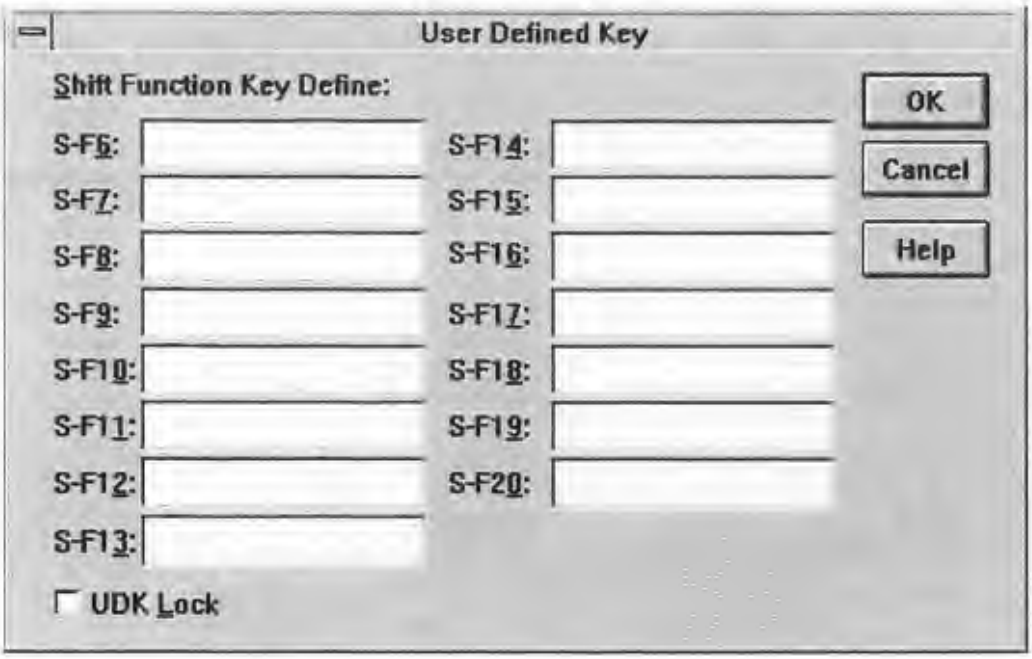

Figure 4.10 Designed Dialog Box for UDK Definitions

Figure 4.11 Designed Dialog Box for Tabulation Setting

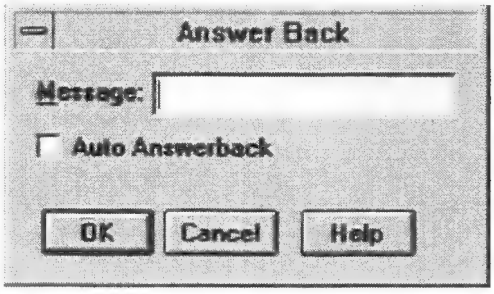

Figure 4.12 Designed Dialog Box for Answer Back Message

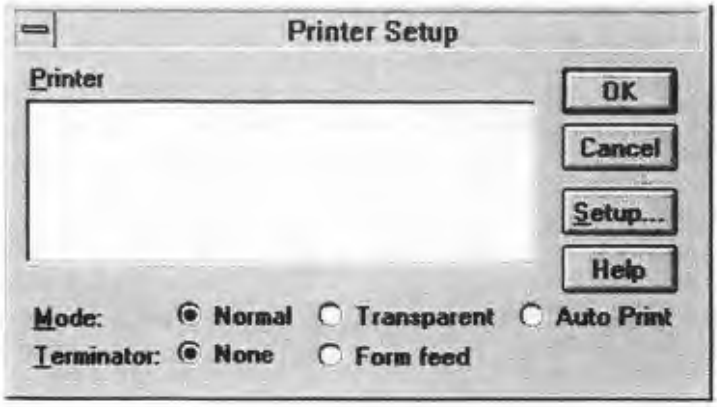

Figure 4.13 Designed Dialog Box for Printer Setup

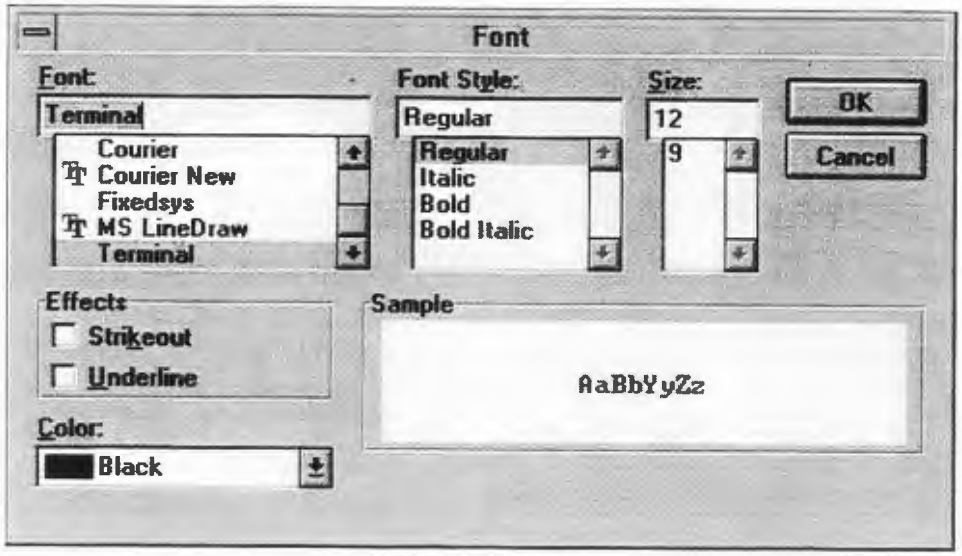

Figure 4.14 Designed Dialog Box for Font Preference

4.1.4 Terminal Control Protocol (TCP) Module

This module processes the terminal data presentation for the terminal protocol between the host computer and terminal emulation. For VT220-type terminals, there are several kinds of outbound data, which are sent from the host computer to the terminals. Based upon the ASCII codes, they are printable characters, C0/C1 control characters, and various control sequences as specified in section 2.2. After the translation is made for raw data received from communication line, the TCP module presents the data onto the terminal emulation screen.

There are two buffers maintained for the terminal screen data. One is the character code buffer (24*80 bytes) and the other is the character attribute buffer (24*80 words). The code buffer stores the (ASCII) codes and the attribute buffer stores the graphic renditions for characters displayed on terminal screen accordingly. Figure 4.15 shows the bitwise scheme of the graphic renditions for the character attributes stored in the attribute buffer.

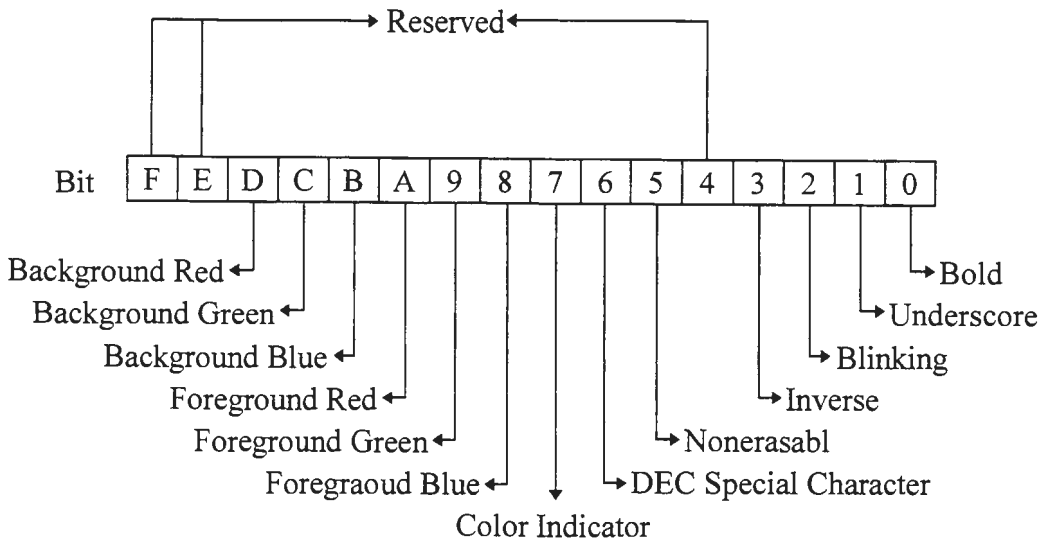

Figure 4.15 Bitwise Scheme of Character Attribute Buffer

The TCP module first checks if the outbound character belongs to C0 or C1 control characters. If it does, the corresponding handle procedure will be invoked via C0 or C1 control character entry table. The respective behaviors of C0 and C1 control characters are as described in Table 2.1 and Table 2.2. If the outbound character is a control sequence introducer, both the C0 and C1 entry tables will hook the control sequence module for the next processing state. Other cases will be treated as the printable character with the following procedures.

(1) Fills the printable character to the code buffer and the current attribute status (continuously maintained by emulation) to the attribute buffer; or

(2) inserts the printable character to the code buffer and the current attribute status to the attribute buffer if the insert mode is selected as 'on' by the host.

(3) Processes for margin bell if the margin bell and warning bell are coincidentally selected as 'on' by the host or system configuration.

(4) Processes for auto wrap around if it is at the last column of a line and auto wrap is selected as 'on' by the host or system configuration;

(5) Processes for auto print if it is at the last column of a line and auto print mode is selected as 'on' by the host or system configuration;

(6) Hooks the display module to show the character to the terminal screen.

(7) Sends the message for status bar subwindows to hook the status bar maintenance module to update for new information.

4.1.5 Control Sequence Module

This module employs the state machine scheme with several processing tables for final states to solve the different control sequences. At any state, it should reset to the start state while a new control sequence introducer shows up. If the control sequence is illegally terminated or canceled by C0 control character CAN or SUB, it should directly go to the end state. Figure 4.16 cites an example to portray how a state machine works in parsing a control sequence.

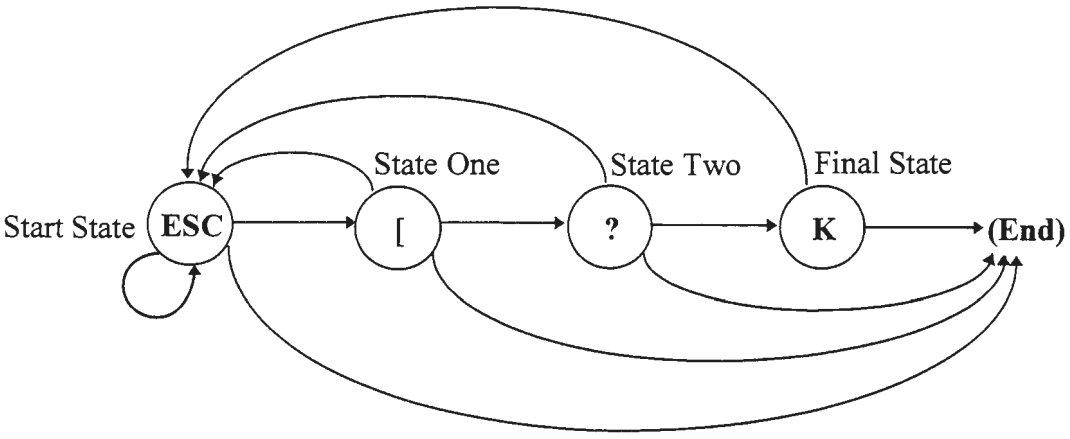

Figure 4.16 An Example for Control Sequence Parsing State Machine

4.1.6 Display Module

This module deals with the outputting of terminal data to the VDU. It displays the character string specified by the TCP module or control sequence module according to the code buffer and the attribute buffer. The following are the itemized primary tasks of this module.

- Displays the specified characters stored in code buffer with corresponding attribute indicated in attribute buffer and selected font.
- Processes for line attribute.
- Maintains cursor style and location.
- Processes for terminal screen scrolling up/down.
- Processes for window sizing and horizontal/ vertical scrolling.

4.1.7 Status Bar Maintenance Module

This module is regarding the status bar maintenance via window messages. The specific window message could be sent by the display module, TCP module, control sequence module, or system configuration module to update information for the followings:

- communication message including communication port, baud rate, parity check, data bit, and stop bit;
- keyboard locking status;
- terminal screen insert/replace mode;
- terminal screen hold message;
- cursor position with two-dimension message (row,column); and
- clock or timer.

4.1.8 Printer Control Protocol (PCP) Module

Apart from outbound data to terminal screen, the host computer might render data to the terminal emulation's local printer for the purpose of sharing the system printer's load. The control sequences for printer control protocol are explained in section 2.4.

The PCP module employs the same skill as the control sequence module, which is actually a submodule of the TCP module, to parse the control sequences. This module is similar to the TCP module in data presentation except for the outputting of the translated data to the local printer. The main processes in PCP module are:

- presentations for C0/C1 control characters;
- presentations for control sequences;
- presentations for printable characters; and
- printer manipulation functions.

4.1.9 Keyboard Handling Module

After the host computer has downloaded the screens to the terminal, users might follow the indications to input data to the host computer. The most important input interface is the keyboard, as well as the virtual keys appearing on the screen.

Like the outbound data, the inbound data, which is in contrast sent from the terminal to the host computer, also includes printable characters, control characters, and control sequences as the specification stated in section 2.3. This module maps the keys into relative key codes. After the corresponding key code is produced, this module invokes the communication module and sends the inbound data to the host. Since the PC keyboard is a little different from the VT220

terminal keyboard, this module makes some changes for certain keys (see Table 2.26, Table 2.29, and Table 2.30). The implementations in this module include:

- control character mapping;
- printable character mapping;
- editing key mapping, cursor key mapping;
- keypad mapping; and
- function key redefinition mapping.

This module employs some mapping tables for varied key codes. The key codes of PC function keys are stored in the data structure as illustrated in Figure 3.9. Other implementations are described as below in regard to the keyboard preferences of the system configuration.

- Sounds a beep when the cursor reaches the last eighth column on each line if the Margin Bell is selected as 'on'.
- Sends C0 control character BS if Backspace is selected for the Backspace Key, otherwise sends DEL (RUBOUT).

4.1.10 UDK Module

Because the UDK control sequence is longer and more complicated than any other control sequence, this module branches off to parse the control sequence and store the significant definition to the UDK buffer of the data structure as depicted in Figure 3.9. Another main effort is made in this module to handle and maintain the screen virtual keys. The specification of UDK control sequence is as characterized in Table 2.16.

4.2 Communication Module

The communication module is fulfilled as a Dynamic Linking Library (DLL). Most functions in this module are implemented through the Windows RS-232-C API functions. The major procedures in this module are:

- communication port initiation/termination function;
- communication speed getting/setting function;
- data-bit getting/setting function;
- stop-bit getting/setting function;
- parity-checking function;
- data input/output function; and
- BREAK signal function.

The communication flow control and handshaking employs the in-band method (XON/XOFF).

CHAPTER 5

CONCLUSION

In former times, when people had access to the mainframe/host computer, they completely depended on the traditional terminal. Nonetheless, traditional terminals had many problems such as immobility, inflexibility, unextendibility and incompatibility. For a long time, the user was constantly bothered by the 'dumb' terminals which had defects and inconveniences.

As personal computers are more efficient and powerful today, terminal emulation is thus devised to make up the disadvantages of the traditional terminal. With the advent of terminal emulation, people no longer feel so inconvenienced.

In this report, I employ the Microsoft SDK to develop the DEC VT220-type terminal emulation system in Microsoft Windows environment. This terminal emulation enables users to connect and log on to diverse systems, such as VAX VMS mainframes, UNIX-based machines, Bulletin Board System (BBS) services, various commercial sites, and so forth, to obtain the desired services, information, and resources.

5.1 Comparisons

Due to the hardware limitation or unnecessary implementation of rarely used functions, there are some distinctions between the standard VT220 terminal and the VT220 terminal emulation system proposed by this report. Table 5.1 inscribes their differentiation with comparisons.

Table 5.1 Differentiation Between Terminal Emulation System and VT220 Terminal

Function	Terminal Emulation	VT220 Terminal
4 LEDs display on keyboard	No	Yes
PC numeric keypad mode	Yes	No
Editing key redefine	Yes	No
Caps Lock and Shift Lock	Separate key	Same key
Keyclick option	No	Yes
Compose Character key	No	Yes
National keyboard & character set	North American	15 languages
VT52 terminal type	No	Yes
ANSI-BBS terminal type	Yes	No
ANSI color control sequence	Yes	No
Hardware terminal reset	No	Yes
Baud rate (transmit & receive)	No difference	Independent
Flow control	In-band (XON/XOFF)	Both in-band and out-of-band
Long Break signal (3.5 seconds)	Yes	No
Smooth scroll	No	Yes
Printer port	Parallel	Serial

5.2 Future Scope

The more functions added to personal computers, the more utilizations and values presented on each machine. Based upon the personal computer's outstanding virtues, a terminal emulation can well overcome the problems which had occurred with the traditional terminal.

Through terminal emulation, people now are able to connect their personal computers comfortably from their homes to any mainframe/host computer even across a long distance and to numerous networks in the world. The user can hook up to as many sessions as they want, particularly interacting with different hosts coincidentally.

A terminal emulation with full functionality further provides the user with various helpful functions and utilities which enable users to conveniently communicate with the host computer.

As a matter of fact, the ability and potential of a terminal emulation are unimaginably mighty especially in the event that the host computer applications mutually interact with the personal computer applications.

To conclude, by way of terminal emulation, people can gain profit with the host computer's resources via the useful functions provided by the terminal emulation. Also, the integration of their personal computers and the remote hosts into a unified powerful mechanism is a benefit of terminal emulation. As a result, the user may utilize the robust mechanism that the host computer combined with the personal computer by incorporating their effective devices, rich resources, and potent applications.

APPENDIX A

CHARACTER SETS

A.1 The ASCII Alphabet

Table A.1 The ASCII Alphabet

0	NUL	32	SPACE	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	63	?	95	_	127	RUBOUT

A.2 DEC Special Graphics Set

Most of the DEC Special Graphic characters are the same as the ASCII alphabet except for the graphics for ASCII codes 95 through 126. Due to the OS font restriction, this terminal emulation supports only those graphic characters depicted in the third column of Table A.2.

Table A.2 Differentiation Between DEC Special Graphics and ASCII Characters

Code	ASCII Char	DEC Special	Legend
95	_	(Blank)	Blank
96	\	◆	Diamond
97	a	▒	Checkerboard
98	b	(Blank)	Digraph: HT
99	c	(Blank)	Digraph: FF
100	d	(Blank)	Digraph: CR
101	e	(Blank)	Digraph: LF
102	f	°	Degree Symbol
103	g	±	+/- Symbol
104	h	(Blank)	Digraph: NL
105	i	(Blank)	Digraph: VT
106	j	┘	Lower-right corner
107	k	┐	Upper-right corner
108	l	┌	Upper-left corner
109	m	└	Lower-left corner
110	n	┼	Crossing lines
111	o	⎺	Horizontal Line - scan 1
112	p	⎻	Horizontal Line - scan 3
113	q	─	Horizontal Line - scan 5
114	r	⎼	Horizontal Line - scan 7
115	s	⎽	Horizontal Line - scan 9
116	t	├	Left "T" (|-)
117	u	┤	Right "T" (-|)
118	v	┴	Bottom "T" (|□_)
119	w	┬	Top "T" (T)
120	x	│	Vertical Bar (|)
121	y	≤	Less/Equal (<□_)
122	z	≥	Greater/Equal (>□_)
123	{	π	Pi symbol
124	|	(Blank)	Not equal (=□/)
125	}	£	UK pound symbol
126	~	•	Centered dot

References

1. *VT330/VT340 Programmer Reference Manual, Volume 1: Text Programming.* Digital Equipment Corporation, May 1988.
2. Petzold, Charles. *Programming Windows.* Microsoft Press, 1990.
3. Conger, James L. *Windows API Bible: The Definitive Programmer's Reference.* Waite Group Press, 100 Shoreline Highway, Suite A-285, Mill Valley, CA 94941, 1992.
4. Richter, Jeffrey M. *Windows 3: A Developer's Guide.* M&T Publishing, Inc., 501 Galveston Drive, Red Wood City, CA 94063, 1991.
5. Strauss, Paul. "Terminal Emulation for Client/Server." *Datamation* (April 1, 1994): 68.
6. Chom, Leslie H. "On ramps to the Information Superhighway." *Advanced Materials & Processes* (December 1994): 78-79.
7. da Cruz, Frank. *Kermit, A File Transfer Protocol.* Digital Press, One Burlington Woods Drive, Burlington, MA 01803, 1987.
8. Tanenbaum, Andrew S. *Computer Networks.* P T R Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1989.
9. *EIA Standard RS-232-C, Interface Between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange.* Electronic Industries Association, 2001 Eye Street N. W., Washington DC 20006, 1969.
10. Stamper, David A. *Business Data Communications.* The Benjamin/Cummings Publishing Company, Inc., 390 Bridge Parkway, Redwood City, CA 94065, 1994.
11. *Hayes Smartmodem 1200 User's Guide.* Hayes Microcomputer Products, Inc., 1985.
12. Comer, Douglas E. *Internetworking with TCP/IP, Volume I: Principles, Protocols, and*

Architecture. Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1995.

13. Gilbert, P. *Software Design and Development*. Science Research Associates, Inc., 1983.
14. Martin, J. and, C. McClure. *Diagramming Techniques for Analysts and Programmers*. Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1985.
15. Mynatt, Barbee T. *Software Engineering with Student Project Guidance*. Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1990.
16. Kernighan, B. W. and, D. M. Ritchie *Prentice-Hall Software Series: The C programming Language*. Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1978.