



Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Programming the Display Terminal: Models D217, D413, and D463

014-002111-00

Programming the Display Terminal: Models D217, D413, and D463

014-002111-00

Ordering No. 014-002111
Copyright © Data General Corporation, 1991
All Rights Reserved
Printed in the United States of America
Rev. 00, October 1991

Notice

DATA GENERAL CORPORATION (DGC) HAS PREPARED THIS DOCUMENT FOR USE BY DGC PERSONNEL, LICENSEES, AND CUSTOMERS. THE INFORMATION CONTAINED HEREIN IS THE PROPERTY OF DGC, AND THE CONTENTS OF THIS MANUAL SHALL NOT BE REPRODUCED IN WHOLE OR IN PART NOR USED OTHER THAN AS ALLOWED IN THE DGC LICENSE AGREEMENT.

DGC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DGC to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DGC HARDWARE PRODUCTS AND THE LICENSING OF DGC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DGC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE, OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DGC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF DGC WHATSOEVER.

AViON, CEO, DASHER, DATAPREP, DESKTOP GENERATION, ECLIPSE, ECLIPSE MV/4000, ECLIPSE MV/6000, ECLIPSE MV/8000, GENAP, INFOS, microNOVA, NOVA, PRESENT, PROXI, SWAT, and TRENDAVIEW are U.S. registered trademarks of Data General Corporation; and AOSMAGIC, AOS/VSMAGIC, AROSE/PC, ArrayPlus, AV Object Office, AV Office, BaseLink, BusiGEN, BusiPEN, BusiTEXT, CEO Connection, CEO Connection/LAN, CEO Drawing Board, CEO DXA, CEO Light, CEO MAILL, CEO Object Office, CEO PXA, CEO Wordview, CEOwrite, COBOL/SMART, COMPUCALC, CSMAGIC, DASHER/One, DASHER/286, DASHER/286-12c, DASHER/286-12j, DASHER/386, DASHER/386-16c, DASHER/386-25, DASHER/386-25k, DASHER/386SX, DASHER/386SX-16, DASHER/386SX-20, DASHER/486-25, DASHER/LN, DATA GENERAL/One, DESKTOP/UX, DG/500, DG/AROSE, DGConnect, DG/DBUS, DG/Fontstyles, DG/GATE, DG/GEO, DG/HEO, DG/L, DG/LIBRARY, DG/UX, DG/XAP, ECLIPSE MV/1000, ECLIPSE MV/1400, ECLIPSE MV/2000, ECLIPSE MV/2500, ECLIPSE MV/3500, ECLIPSE MV/5000, ECLIPSE MV/5500, ECLIPSE MV/5600, ECLIPSE MV/7800, ECLIPSE MV/9300, ECLIPSE MV/9500, ECLIPSE MV/9600, ECLIPSE MV/10000, ECLIPSE MV/15000, ECLIPSE MV/18000, ECLIPSE MV/20000, ECLIPSE MV/30000, ECLIPSE MV/40000, FORMA-TEXT, GATEKEEPER, GDC/1000, GDC/2400, microECLIPSE, microMV, MV/UX, PC Liaison, RASS, REV-UP, SLATE, SPARE MAIL, SUPPORT MANAGER, TEO, TEO/3D, TEO/Electronics, TURBO/4, UNITE, WALKABOUT, WALKABOUT/SX, and XODIAC are trademarks of Data General Corporation.

UNIX is a U.S. registered trademark of American Telephone and Telegraph Company.

VP/ix is a trademark of Interactive Systems Corporation.

DEC is a U.S. registered trademark of Digital Equipment Corporation.

Tektronix is a U.S. registered trademark of Tektronix, Incorporated.

MS-DOS is a U.S. registered trademark of Microsoft Corporation.

AT is a U.S. registered trademark of International Business Machines Corporation.

Programming the Display Terminal:

Models D217, D413, and D463

014-002111-00

Revision History:

Original Release – October 1991

Preface

This manual provides information on the programming environment of the D217, D413, and D463 display terminals. Intended for persons programming host-resident software, this manual was designed as a reference tool and does not explain basic operating functions of these terminals. Refer to *Installing and Operating D216E+, D217, D413, and D463 Display Terminals* (014-001767) for information on operating one of these terminals.

This reference manual is organized as described below:

- Chapter 1 Characteristics of the programming environment that are common to all operating modes or emulations; *everyone should read this chapter.*
- Chapter 2 Data General native-mode operations and commands.
- Chapter 3 VT320, VT100, and VT52 emulations and control sequences.
- Chapter 4 Tektronix® 4010 emulation and commands.
- Chapter 5 PCTERM operation and commands. Although PCTERM is actually an operating mode of the VT320/100 emulation, it is covered within its own chapter to avoid confusion within Chapter 3.
- Appendix A Tables of all character sets used in any emulation or operating mode.
- Appendix B Keyboard layouts for all national-language keyboards supported on these terminals.
- Appendix C Sample code (in C and FORTRAN 77) that illustrates the programming environment within several emulations.

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone Assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service

options. If you are within the United States or Canada, contact the Data General Service Center by calling 1-800-DG-HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining Our Users Group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive FOCUS monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1-800-877-4787 or 1-512-345-5316.

End of Preface

Contents

Chapter 1 — Introduction

Terminal Features	1-2
Enhancements for the D413.D463	1-3
Supported Emulations and Modes	1-3
Summary of On-Line Operations	1-4
Communications Interface	1-4
Input Buffer	1-4
Flow Control	1-5
The Character Generator	1-5
25th Row Support	1-6
Term-Server Support	1-6

Chapter 2 — Data General Native-Mode

Data General Native-Mode — Summary of Operations	2-1
Data General Native-Mode Features	2-4
Keyboard Character Generation	2-4
Forming Command Arguments	2-7
Recovering a Decimal Value for Command Arguments	2-9
Forming Location Arguments	2-10
Recovering a Decimal Value for Location Arguments	2-12
Character Sets	2-13
Hard Character Sets	2-13
Soft Character Sets	2-15
Graphics	2-17
The Graphics Coordinate System	2-17
Graphics Cursor	2-18
Windows	2-19
Lead-In Codes	2-19
UNIX Support	2-20
Host-Programmable Function Keys	2-22

Debugging Support	2-23
Dual Emulation Support	2-23

Data General Native-Mode — Commands 2-25

Format of Command Listings in this Section	2-26
Command Syntax and Code Conventions	2-26
Character Set Commands	2-27
Select Character Set	2-28
Shift Out	2-28
Shift In	2-28
Deallocate Character Sets	2-28
Define Character	2-29
Reserve Character	2-29
Read Characters Remaining	2-30
Character Attribute Commands	2-31
Change Attributes	2-31
Dim On	2-32
Dim Off	2-32
Blink On	2-32
Blink Off	2-32
Blink Enable	2-32
Blink Disable	2-33
Underscore On	2-33
Underscore Off	2-33
Reverse Video On	2-33
Reverse Video Off	2-33
Protect On	2-34
Protect Off	2-34
Protect Enable	2-34
Protect Disable	2-34
Double High/Double Wide	2-35
Field Attributes	2-35
Page Attributes	2-36
Relative Cursor-Positioning Commands	2-37
Cursor Right	2-37
Cursor Left	2-37
Cursor Up	2-37
Cursor Down	2-38
New Line	2-38
Carriage Return	2-38
Margins Commands	2-39
Set Margins	2-39
Set Alternate Margins	2-39

Restore Normal Margins	2-40
Screen Management Commands	2-41
Write Window Address	2-41
Window Home	2-42
Set Windows	2-43
Set 25th Line Mode	2-44
Push	2-44
Pop	2-45
Write Screen Address	2-46
Select Compressed Spacing	2-47
Select Normal Spacing	2-48
Named Save/Restore Cursor	2-48
Save/Restore Screen Contents	2-49
Screen Home	2-49
Set Row Length	2-51
Scrolling Commands	2-51
Show Columns	2-51
Set Scroll Rate	2-51
Scroll Down	2-52
Scroll Up	2-52
Scroll Left	2-52
Scroll Right	2-53
Roll Enable	2-53
Roll Disable	2-54
Horizontal Scroll Disable	2-54
Horizontal Scroll Enable	2-54
Editing Commands	2-55
Erase Window	2-55
Erase Screen	2-55
Erase to End of Line	2-55
Insert Line	2-55
Delete Line	2-56
Insert Line Between Margins	2-56
Delete Line Between Margins	2-56
Erase Unprotected	2-57
Insert Character	2-57
Delete Character	2-57
Programmable Function Key Commands	2-58
Host Programmable Function Keys	2-58
Reporting Commands	2-61
Report Screen Size	2-61
Read Horizontal Scroll Offset	2-62
Read Window Address	2-62

Read Screen Address	2-63
Read Window Contents	2-63
Read Model ID	2-64
Read New Model ID	2-65
Dual-Emulation Support Commands	2-67
Hot Key Switch	2-67
Switch Emulation Mode	2-67
Set Split Screen Mode	2-68
Set First Row To Display	2-68
Set Device Options	2-69
Miscellaneous Commands	2-70
Set Cursor Type	2-70
Set Model ID	2-71
Set Clock Time	2-71
Bell	2-71
Reset	2-71
Select 7/8 Bit Operation	2-72
Set Keyboard Language	2-72
UNIX Mode	2-73
Drawing Commands	2-74
Line	2-74
Arc	2-75
Bar	2-75
Polygon Fill	2-76
Set Pattern	2-76
Set Foreground Color	2-77
Graphics Cursor Commands	2-78
Read Cursor Location	2-78
Cursor On	2-78
Cursor Off	2-78
Cursor Location	2-79
Cursor Track	2-79
Cursor Attributes	2-80
Cursor Reset	2-80
Printer Commands	2-81
Print Window	2-81
Print Form	2-81
Form Bit Dump	2-82
Window Bit Dump	2-83
Print Pass Through On	2-84
Print Pass Through Off	2-84
Printer Pass Back To Host	2-85
Simulprint On	2-85

Simulprint Off	2-86
VT-Style Autoprint On	2-86
VT-Style Autoprint Off	2-86
Select Printer National Character Set	2-87
Debugging Commands	2-88
Data Trap Mode	2-88
Diagnostic Commands	2-89
Read Cursor Contents	2-89
Read Bit Contents	2-90
Character Loopback	2-90
Fill Screen With Character	2-91
Fill Screen With Grid	2-91
Display Character Generator Contents	2-91
Perform UART Loopback Test	2-91

Chapter 3 — VT320/100/52 Emulations

VT320/100 Emulation — Summary of Operations 3-3

VT320/100 Emulation Features	3-4
Control Codes	3-5
Received Control Codes	3-5
Transmitted Control Codes	3-8
Using 8-bit Code in 7-bit Environments	3-12
Control Sequences	3-13
Escape Sequences	3-13
Device Control Strings	3-13
Generated Keyboard Codes	3-14
Character Sets	3-17
Hard Character Sets	3-17
Soft Character Sets	3-20
ANSI Standard Mode Switches	3-23
ANSI Private Mode Switches	3-25
User-Defined Keys	3-27
Character Attributes	3-27
Line Attributes	3-28

VT320/100 Emulation — Control Sequences 3-29

Format of Control Sequences in This Section 3-30

A Note on Syntax Conventions 3-30

Hard Character Set Control Sequences 3-31

 Sequences for Designating Character Sets 3-31

 Assign User-Preferred Supplemental Set (AUPSS) 3-32

 Sequences for Invoking Character Sets 3-32

 Shift In (SI) 3-32

 Shift Out (SO) 3-32

 Shift Lock Two (SL2) 3-33

 Shift Lock Three (SL3) 3-33

 Shift Lock G1 GR 3-33

 Shift Lock G2 GR 3-33

 Shift Lock G3 GR 3-33

 Single Shift Two (SS2) 3-34

 Single Shift Three (SS3) 3-34

Soft Character Set Control Sequences 3-35

 Sequences for Downloading Soft Characters (VT320) 3-35

 Sequences for Clearing Downloaded Soft Character Sets 3-36

Attribute Control Sequences 3-37

 Line Attribute Sequences 3-37

 Character Attribute Sequences 3-38

 Select Graphic Rendition (SGR) 3-38

 Select Character Attributes (SCA) 3-38

Cursor Positioning Control Sequences 3-39

 Cursor Up (CUU) 3-39

 Cursor Down (CUD) 3-39

 Cursor Forward (CUF) 3-40

 Cursor Backward (CUB) 3-40

 Cursor Position (CUP) 3-40

 Horizontal and Vertical Position (HVP) 3-41

 Index (IND) 3-41

 Reverse Index (RI) 3-41

 Next Line (NEL) 3-42

 Save Cursor (SC) 3-42

 Restore Cursor (RC) 3-42

Tabulation Control Sequences 3-43

 Set Horizontal Tab (HTS) 3-43

 Clear Tab Stops (TBC) 3-43

Screen Editing Control Sequences	3-44
Delete Character (DCH)	3-44
Insert Character (ICH)	3-44
Insert Line (IL)	3-45
Delete Line (DL)	3-45
Erase Character (ECH)	3-45
Erase In Line (EL)	3-46
Erase In Display (ED)	3-46
Selective Erase In Line (DECSEL)	3-47
Selective Erase In Display (DECSED)	3-47
Scroll Down (SD)T320	3-48
Scroll Up (SU)	3-48
ANSI Standard Mode Control Sequences	3-49
Set Mode (SM)	3-49
Reset Mode (RM)	3-49
ANSI Standard Mode Parameters	3-50
Keyboard Action Mode (KAM)	3-50
Insert/Replace Mode (IRM)	3-50
Send/Receive Mode (SRM)	3-51
Line Feed/New Line Mode (LNM)	3-51
ANSI Private Mode Control Sequences	3-52
Private Set Mode (EXSM)	3-52
Private Reset Mode (EXRM)	3-53
ANSI Private Operating Mode Parameters	3-54
Application/ANSI Cursor Keys Mode (ACKM)	3-54
Column Mode (COLM)	3-54
Scrolling Mode (SCRLM)	3-55
Screen Mode (SCRNM)	3-55
Cursor Origin Mode (COM)	3-55
Set to VT52 Mode	3-56
Set Limited Transmit	3-56
Auto Wrap Mode (AWM)	3-56
Auto Repeat Mode (ARM)	3-57
Print Form Feed Mode (PFF)	3-57
Print Extent Mode (PEXM)	3-58
Text Cursor Enable Mode (TCEM)	3-58
Multi/National Character Set Mode (MNCISM)	3-58
Numeric Keypad Mode (NKM)	3-59
Backarrow Key Mode (BKM)	3-59
Typewriter/Data Processing Keys Mode (KBUM)	3-59
PCTERM Mode	3-60

Transmission Control Sequences	3-61
Transmit 7-bit Controls	3-61
Transmit 8-bit Controls	3-61
User-Defined Key Control Sequences	3-62
User Defined Keys (UDK)	3-62
Miscellaneous Control Sequences	3-63
Soft Terminal Reset	3-61
Hard Terminal Reset	3-61
Alignment	3-61
Display Character Generator Contents	3-63
Set/Report Language	3-64
Set Clock Time	3-64
Hot Key Switch	3-64
Set Top and Bottom Margins (STBM)	3-65
Bit Dump Screen	3-65
Force Display	3-65
Data Trap Mode	3-66
Select Active Status Display (SASD)	3-66
Select Status Line Type (SSDT)	3-67
Set Conformance Level (SCL)	3-67
Set Device Options	3-68
Split Screen	3-69
Reporting Control Sequences	3-70
Terminal Identification (DECID)	3-70
Device Status Report (DSR)	3-70
Primary Device Attribute Request (DA)	3-71
Secondary Device Attribute Request (SDA)	3-72
Cursor Position Report (CPR)	3-72
User Defined Key Status	3-73
Keyboard Language	3-74
Answerback	3-75
Printer Port Status	3-75
Request Terminal State Report (RQTSR)	3-76
Restore Terminal State (RSTS)	3-76
Request Presentation State Report (RQPSR)	3-77
Restore Presentation State (RSPS)	3-79
Request Mode (RQM)	3-79
Request User-Preferred Supplemental Set (RQUPSS)	3-81
Read Cursor Content	3-81
Request Selection or Setting (RQSS)	3-82

Printing Control Sequences	3-84
Auto Print Mode	3-84
Print Screen	3-84
Print Cursor Line	3-84
Print Controller Mode	3-85
VT52 Emulation Operations and Escape Sequences	3-85
Character Sets and Graphics	3-86
Keyboard Generated Codes	3-87
VT52 Escape Sequences	3-89

Chapter 4 — Tektronix 4010 Emulation

Emulation Features	4-2
Overview of Operational Modes	4-2
Alphanumeric Mode	4-3
Margins	4-3
View and Hold Submodes	4-3
Graphic Plot Mode	4-4
Using Graphic Plot Mode	4-4
Graphics Input Mode	4-7
Alphanumeric Cursor	4-7
Graphics Cursor	4-8
Hard Copy Command	4-8
Hot-Key Switch	4-9
User Selectable Options	4-9
Graphic Input Terminators	4-9
Line/Local Operation	4-9
Data Communication Baud Rates	4-9
Control Codes	4-10

Chapter 5 — PCTERM Operations

Introduction	5-1
Inbound (Terminal to Host) Codes	5-2
Flow Control	5-2
Keyboard Generated Codes	5-2
Outbound (Host to Terminal) Codes	5-6
Cursor Addressing	5-6
Character Sets	5-6
VP/ix getty Setup	5-7
Sample terminfo File	5-7
Sample VP/ix term File	5-8

Appendix A — Character Sets

United States ASCII Character Set	A-2
NRC United Kingdom Character Set	A-3
NRC French Character Set	A-4
NRC German Character Set	A-5
NRC Swedish/Finnish Character Set	A-6
NRC Spanish Character Set	A-7
NRC Danish/Norwegian Character Set	A-8
NRC Swiss Character Set	A-9
NRC Katakana (G0) Character Set	A-10
Katakana (G1) Character Set	A-11
DG International Character Set	A-12
Word-Processing, Greek, and Math Character Set	A-13
DG Line Drawing Character Set	A-14
DG Special Graphics Character Set (PC Characters)	A-15
VT Multinational Character Set	A-16
VT Special Graphics Character Set (VT Line Drawing)	A-17
ISO 8859/1.2 Character Set	A-18
PCTERM Low Character Set (0 hex through 7F hex)	A-19
PCTERM High Character Set (80 hex through FF hex)	A-20

Appendix B — National Language Keyboards

Canadian/English 107-key Keyboard	B-2
Canadian/French 107-key Keyboard	B-2
Danish 107-key Keyboard	B-3
French 107-key Keyboard	B-3
German 107-key Keyboard	B-4
Italian 107-key Keyboard	B-4
Katakana 107-key Keyboard	B-5
Norwegian 107-key Keyboard	B-5
Spanish 107-key Keyboard	B-6
Swedish/Finnish 107-key Keyboard	B-6
Swiss/French 107-key Keyboard	B-7
Swiss/German 107-key Keyboard	B-7
United Kingdom 107-key Keyboard	B-8
United States 107-key Keyboard	B-8
Canadian/French 102-key Keyboard	B-9
Danish 102-key Keyboard	B-9
French 102-key Keyboard	B-10
German 102-key Keyboard	B-10
Italian 102-key Keyboard	B-11
Norwegian 102-key Keyboard	B-11
Spanish 102-key Keyboard	B-12
Swedish/Finnish 102-key Keyboard	B-12
Swiss 102-key Keyboard	B-13
United Kingdom 102-key Keyboard	B-13
United States 102-key Keyboard	B-14

Appendix C — Sample Programs

Notice	C-2
Data General Native-Mode "C"	C-3
Data General Native-Mode "Fortran 77"	C-10
VT320 Emulation "C"	C-20
VT320 Emulation "Fortran 77"	C-27
VT52 Emulation "C"	C-37
VT52 Emulation "Fortran 77"	C-44
Tektronix 4010 Emulation "C"	C-52
Tektronix 4010 Emulation "Fortran 77"	C-59

Figures

Figure

2-1	Combining LSBs to Create Decimal Values for Command Arguments	2-9
2-2	Combining LSBs to Create Decimal Values for Location Arguments	2-12
2-3	Designating Character Sets as GL and GR (Data General Native-Mode)	2-14
2-4	Defining a Soft Character Cell Row	2-16
2-5	Combining <dd> Pairs to Specify a Soft Character	2-16
2-6	The Graphics Coordinate System at Power Up or Reset	2-18
3-1	Invoking Character Sets into GL and GR (VT320/100)	3-19
3-2	Examples of Double-Height and Double-Width Lines	3-28
4-1	The Coordinate System of the Tektronix 4010 Emulation	4-4

Tables

Table

2-1	Keyboard Generated Codes — Function Keys (Data General Native-Mode)	2-5
2-2	Keyboard Generated Codes — Editing Keypad (Data General Native-Mode)	2-6
2-3	Keyboard Generated Codes — Cursor Keypad (Data General Native-Mode)	2-6
2-4	Keyboard Generated Codes — Numeric Keypad (Data General Native-Mode)	2-7
2-5	Command Argument Translation (Decimal to DG-Hex)	2-8
2-6	Location Argument Translation (Decimal into ASCII Characters)	2-11
2-7	Character Sets Available In Data General Native-Mode	2-13
2-8	Remapped Key Codes for UNIX Support	2-20
2-9	Altered Outbound Codes for UNIX Support	2-21
2-10	Altered Data General Native-Mode Commands for UNIX Support	2-21
2-11	Control Codes Altered for UNIX Support	2-22
3-1	All C0 and C1 Control Codes	3-5
3-2	Supported C0 Control Codes	3-6
3-3	Supported C1 Control Codes	3-7
3-4	Control Codes Generated from the Editing Keys (VT320/100)	3-8
3-5	Control Codes Generated from the Cursor Control Keys (VT320/100)	3-8
3-6	Control Codes Generated from the Auxiliary Keypad (VT320/100)	3-9
3-7	Control Codes Generated from the Function Keys (VT320/100)	3-10
3-8	C0 Control Codes Generated from the Main Keypad (VT320/100)	3-11

3-9	Keyboard Generated Codes — Function Keys (VT320)	3-15
3-10	Keyboard Generated Codes — Function Keys (VT100)	3-16
3-11	Keyboard Generated Codes — Editing Keypad (VT320)	3-16
3-12	Keyboard Generated Codes — Cursor Keypad (VT320 and VT100)	3-16
3-13	Keyboard Generated Codes — Numeric Keypad (VT320 and VT100)	3-17
3-14	Default ANSI Standard Mode Parameters	3-23
3-15	Default ANSI Private Mode Parameters	3-25
3-16	Keyboard Generated Codes — Function Keys (VT52)	3-86
3-17	Keyboard Generated Codes — Cursor Keys (VT52)	3-86
3-18	Keyboard Generated Codes — Numeric Keypad (VT52)	3-87
3-19	VT52 Escape Sequences	3-88
4-1	Format of the Alphanumeric Cursor Status Byte	4-7
4-2	Single Control Codes	4-10
4-3	Double Control Codes (Escape Sequences)	4-10
5-1	Keyboard Generated Codes — Function Keys (PCTERM)	5-3
5-2	Keyboard Generated Codes — Numeric Keypad (PCTERM)	5-3
5-3	Keyboard Generated Codes — Editing Keypad with Num Lock Off (PCTERM)	5-4
5-4	Keyboard Generated Codes — Editing Keypad with Num Lock On (PCTERM)	5-4
5-5	Keyboard Generated Codes — Main Keypad (PCTERM)	5-5

End of Contents



Chapter 1

Introduction

This chapter provides a brief introduction to the programming environment of the D217/ D413/D463 line of Data General terminals. This chapter has the major sections listed below.

Terminal Features

Enhancements for the D413/D463

Supported Emulations and Modes

Summary of On-Line Operations

Communications Interface

Input Buffer

Flow Control

The Character Generator

25th Row Support

Term-Server Support

Information on operating one of these terminals is contained within *Installing and Operating Your D216E+, D217, D413, and D463 Display Terminal* (014-001767).

Terminal Features

The D217/D413/D463 terminals provides maximum compatibility with both the DASHER/D200 and the expanded set of D460 primitives for the interactive applications programmer. Listed below are some of the main programming features:

- 25 lines of 80 characters displayed in a 10 x12 dot matrix on the D217
- 25 lines of 81 characters displayed in a 10 x 12 dot matrix in normal mode on the D413/D463, and 25 lines of 135 characters displayed in a 6 x 12 dot matrix in compressed mode on the D413/D463
- Bidirectional vertical scrolling on all models
- D200 upward compatibility on all models
- UNIX®-friendly protocol mode that simplifies terminfo file specifications on all models
- DEC® VT320, VT100, and VT52 emulations on the D413/D463
- PCTERM operating mode of VT100 emulation on all models
- Full-screen configuration set-up menus on all models
- Smooth vertical scrolling on the D413/D463
- Horizontal scrolling across 207 columns on the D413/D463
- Up to 25 scroll areas, or windows on the D413/D463
- Advanced editing features on the D413/D463
- Protected text on the D413/D463
- Tektronix® 4010 emulation on the D463
- Extended graphics with the Line, Arc, Bar, and Polygon Fill commands on the D463
- Graphics cursor for graphics input, controllable by keyboard or mouse on the D463
- Page and field attributes
- Double high and double wide rows selectable on all models.

Enhancements for the D413/D463

The D217/D413/D463 has significant improvements over the previous generation Data General terminals. These improvements allow wider system support and enhanced features:

- New diacritical (Spcl) key sequences allow access (using the ALT key) of new characters in the Data General International (DGI) character set on all models
- ISO 8859/1.2 Latin-1 compliant character set available in both Data General native-mode and VT emulations on all models
- Added bell to IBM PC AT style (101 key) keyboard
- No downloadable character sets on the D413
- Cmd-Cursor Uparrow/Downarrow enables/disables split screen on the D413/D463 and moves the split point up and down
- Cmd-Shift-Cursor Uparrow/Downarrow moves the active emulation viewing region up and down

Supported Emulations and Modes

The terminals support the following emulation or operational modes:

- Data General native-mode on all models
- VT100/VT52 on all models
- PCTERM (an operational mode of the VT320/100 emulation, PCTERM is treated as a separate emulation in this manual to avoid confusion within the VT320/100 sections) on all models
- VT320 on the D413/D463
- Tektronix 4010 on the D463

Summary of On-Line Operations

The terminal consists of two major units: the keyboard and the display unit. The keyboard is an input device that generates ASCII characters that are interpreted by the host computer. The display unit is an output device. It interprets commands from the host to control the screen image. The display unit also serves as a link between the host and any printer attached to the terminal.

Input from the keyboard consists of commands and characters. Characters entered at the keyboard for display must be forwarded to the display unit by the host (in full duplex mode only). Commands are either forwarded to the host or are used to invoke special functions in the host software.

The display unit, responding to the ASCII display and control characters, is primarily an output device for the host. A few of the ASCII character sequences or commands request status or configuration information from the display unit. In these cases, the display unit also functions as an input device.

Communications Interface

The terminal uses an asynchronous serial communication interface. The host computer and optional printer are connected to the display unit via serial interfaces. You can select the terminal transmit and receive baud rates, and other serial characteristics, through the Configuration Menu (accessed by simultaneously pressing the Cmd and N/C keys on the 107-key, Data General Proprietary keyboard or rightmost Ctrl and Scroll Lock keys on the 101-key IBM PC AT-style keyboard). Even though the bit transmission rate is set with the baud-rate settings, compatibility with Data General operating systems requires that the terminal transmit characters to the host at a maximum rate of 60 characters per second (paced transmissions). The actual transmission rate falls below this when baud rates below 600 are selected. The character transfer rate is menu-selectable so that other systems, such as UNIX, can take advantage of unpaced transmissions.

Input Buffer

The terminals process all display characters and most commands within the time it takes to receive them (within 2 ms) when the baud rate is 4800 or less. The terminal uses a 256-byte input buffer to accumulate characters the display unit is unable to interpret immediately. The characters are held in the buffer until the terminal is ready to accept them. This commonly occurs when the transmission rate is 9600 baud or higher, or when the terminal performs smooth-scrolling operations.

Flow Control

As the input buffer approaches its capacity, the terminal can automatically issue a Ctrl-S (DC3) to signal the host to stop transmitting characters. A Ctrl-S is sent automatically when the number of characters in the buffer reaches a menu-selected value of 64, 128, or 192. You can also choose in menus not to send any Ctrl-S, regardless of the state of the buffer. If the host does not immediately respond to the Ctrl-S by ceasing the flow of characters, the terminal sends another Ctrl-S after eight more characters have been received, and continues to send a Ctrl-S after each eight characters until either the host responds or the buffer fills. If the buffer fills before the host responds, the terminal sends a Ctrl-S back to the host for every character received. When the input buffer falls to 32 enqueued characters, the terminal sends a Ctrl-Q (DC1) to the host, signaling it to resume transmission. The host should respond before the input buffer empties, to avoid a stuttering effect on the display screen.

The Character Generator

All terminals are equipped with a 512-character Character Generator (CGEN), which resides in terminal ROM. All hard (ROM-resident) character sets in any mode or emulation are composed of a subset of the characters in the CGEN. Listed below are the sets of predefined characters that comprise the CGEN:

- U.S. ASCII characters
- Foreign language characters (National Replacement Characters)
- Word Processing with math, Greek, forms, superscripts and subscripts
- PC characters
- DEC VT characters
- ISO (international) characters
- Data General international characters
- Japanese Katakana (phonetic) characters
- Line drawing characters

25th Row Support

All terminals and emulations, except the Tektronix 4010, support a redefineable status line. This means that the status line can be treated as the 25th line on the screen. Thus, when this mode is enabled the screen rows run from 0 through 24, instead of the 0 through 23 range that was standard on previous terminals. Refer to the emulation chapters for additional information on the particular usage of this mode within that emulation.

Term-Server Support

The Cmd-C3 keystroke combination on 107-key proprietary Data General keyboards (or rightmost Ctrl and End keys on 101-key IBM PC AT-style keyboards) generates a special code (1C hex, 034 octal) that we provided for persons using term-servers. This code can be used to initiate a macro inside the term-server that switches to a new host computer and sends a code to the terminal to change to the appropriate emulation. Allowing the term-server to handle this process keeps the host computer and the emulation mode synchronized. The alternative is having the user manually tell the term-server to change to a new host and then manually change the emulation mode on the terminal. Full details on switching emulations is included within the appropriate emulation chapters.

End of Chapter

Chapter 2

Data General Native–Mode

This chapter provides the programming information for Data General native–mode operations on the D217/D413/D463 terminals. This chapter has two major sections:

Data General Native–Mode Summary of Operations

Data General Native–Mode Commands

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various Data General native–mode reference material are covered in appendices, located at the rear of this manual.

Command Syntax and Code Conventions

Throughout this manual, there are certain conventions used whenever Data General native-mode commands are explained or referenced. These conventions are:

- Any value enclosed within angle brackets (< >) is in octal, *except* command or location arguments.
- Values in the form of <n>, <nn>, and <nnn> are location arguments. These values are always (except in UNIX mode, see Table 2-8) expressed in DG-hex. For more information on command arguments (and DG-hex), see "Forming Command Arguments."
- Values in the form of <NNN> are location arguments. These values are always (except in UNIX mode) expressed with ASCII characters from "@" through "_". For more information on location arguments, see "Forming Location Arguments."
- Spaces are often included within a command to separate characters for clarity. When entering a command, *do not* enter these spaces. If a space is part of a command, it will be written as <space>.

Data General Native-Mode Summary of Operations

This section summarizes the operations information specific to Data General native-mode. This section does not cover format or usage of Data General native-mode commands, which are covered in "Data General Native-Mode Commands," later in this chapter.

This section covers the information listed below:

Data General Native-Mode Features

Command Syntax and Code Conventions

Keyboard Character Generation

Forming Command Arguments

Forming Location Arguments

Character Sets

Graphics

Windows

Lead-In Codes

UNIX Support

Host-Programmable Function Keys

Debugging Support

Dual Emulation Support

Information regarding functions of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various Data General native-mode reference material are contained within appendices, which are located at the rear of this manual.

Data General Native–Mode Features

All terminals provide maximum compatibility with the D200 and the expanded set of D400 primitives for the interactive applications programmer. Listed below are some of the main programming features of these terminals in Data General native–mode:

- 25 lines of 81 characters displayed in a 10 x 12 dot matrix in normal mode
- 25 lines of 135 characters displayed in a 6 x 12 dot matrix in compressed mode on the D413/D463
- 25 lines of 80 characters displayed in a 10 x 12 dot matrix on the D217
- D200 upwardly compatible with all models
- UNIX–friendly protocol mode that simplifies terminfo file specifications on all models
- 25th row can be configured as a status line, an extra screen line, a blank row, or as a host programmable row on all models
- Supports both IBM PC AT–style (101 key) and Data General proprietary (107–key) keyboards on all models
- IBM PC printer supported in character and graphics modes on all models
- IBM PC compatible character set available on all models
- ISO 8859/1.2 Latin–1 compliant character set available on all models
- Bidirectional vertical smooth scrolling on the D413/D463
- Horizontal scrolling across 207 columns on the D413/D463
- Up to 25 scroll areas, windows, on the D413/D463
- Enhanced editing commands on the D413/D463
- Protected text on the D413/D463
- Split–screen, dual–host mode on the D413/D463
- Hot–key switch between hosts, or between emulations on a single host, on the D413/D463
- Up to 37 sets of up to 94 characters on the D463, for a total of up to 3504 user–defined characters
- Extended graphics with the Line, Arc, Bar, and Polygon Fill commands on the D463
- Graphics cursor for graphics input, controllable by the keyboard or a mouse, on the D463
- Page and field attributes
- Double high and wide rows



Keyboard Character Generation

Each time you press a key, data is sent to the terminal. The terminal interprets this data as either a local key (such as a Shift key) or as a code generating key (such as the character "A"). If a character code is generated, it is either sent on to the host computer, if in on-line mode, or is taken as direct input by the terminal, if in off-line mode.

The terminal supports two keyboards, a 101-key keyboard, similar to an IBM-PC AT-style keyboard, and a 107-key Data General standard keyboard. Table 2-1, Table 2-2, Table 2-3, and Table 2-4 show the code generated by each key and recognized keystroke combination on 101-key and 107-key keyboards. We did not include the code generated by the main keypad because the generated code is simply the code of the character on the face of the key.

NOTE: The 107-key keyboard has a Cmd key that does not appear on the 101-key keyboard. To simulate the Cmd key, use the rightmost Ctrl key on 101-key keyboards.

Table 2-1 Keyboard Generated Codes — Function Keys (Data General Native-Mode)

107-key Keyboard	101-key Keyboard	Normal	Shift	Ctrl	Ctrl-Shift
F1	F1	<036> q	<036> a	<036> 1	<036> !
F2	F2	<036> r	<036> b	<036> 2	<036> "
F3	F3	<036> s	<036> c	<036> 3	<036> #
F4	F4	<036> t	<036> d	<036> 4	<036> \$
F5	F5	<036> u	<036> e	<036> 5	<036> %
F6	F6	<036> v	<036> f	<036> 6	<036> &
F7	F7	<036> w	<036> g	<036> 7	<036> '
F8	F8	<036> x	<036> h	<036> 8	<036> (
F9	F9	<036> y	<036> i	<036> 9	<036>)
F10	F10	<036> z	<036> j	<036> :	<036> *
F11	F11 or Alt-F1	<036> {	<036> k	<036> ;	<036> +
F12	F12 or Alt-F2	<036>	<036> l	<036> <	<036> ,
F13	Alt-F3	<036> }	<036> m	<036> =	<036> -
F14	Alt-F4	<036> ~	<036> n	<036> >	<036> .
F15	Alt-F5	<036> p	<036> '	<036> 0	<036> <SPACE>
Print	Print Screen	Local function	none	none	none
N/C	Scroll Lock	Local function	none	none	none
Hold	Pause/Break	Local function	none	none	none



Table 2-2 Keyboard Generated Codes — Editing Keypad (Data General Native-Mode)

107-key Keyboard	101-key Keyboard	Normal	Shift	UNIX Mode	
				Normal	Shift
Erase Page	Insert	<014>	<014>	<036> PH	<036> PH
C1	Home	<036> \	<036> X	<036> \	<036> X
C2	Page Up	<036>]	<036> Y	<036>]	<036> Y
Erase EOL	Delete	<013>	<013>	<036> PE	<036> PE
C3	End	<036> ^	<036> Z	<036> ^	<036> Z
C4	Page Down	<036> _	<036> [<036> _	<036> [
Print	Print Screen	none	none	none	none
Cmd-Print	R.Ctrl-Print Screen	<036><021>	<036><001>	<036>P0	<036>P1

Table 2-3 Keyboard Generated Codes — Cursor Keypad (Data General Native-Mode)

107-key Keyboard	101-key Keyboard	Normal	Shift	UNIX Mode	
				Normal	Shift
Uparrow	Uparrow	<027>	<036><027>	<036>PA	<036>Pa
Rightarrow	Rightarrow	<030>	<036><030>	<036>PC	<036>Pc
Leftarrow	Leftarrow	<031>	<036><031>	<036>PD	<036>Pd
Downarrow	Downarrow	<032>	<036><032>	<036>PB	<036>Pb
Home	n/a	<010>	<036><010>	<036>PF	<036>Pf



Table 2-4 Keyboard Generated Codes — Numeric Keypad (Data General Native-Mode)

107-key Keyboard ¹	101-key Keyboard	101-key Keyboard Mapping with Num Lock On	101-key Keyboard Mapping with Num Lock Off ²
On	Num Lock	On	Off
/	/	/	/
*	*	*	*
– (minus)	–	–	–
, (comma)	+	,	,
. (period)	./Delete	.	Erase EOL
0	0/Insert	0	Erase Page
1	1/End	1	C3
2	2/Downarrow	2	Downarrow
3	3/Pg Dn	3	C4
4	4/Leftarrow	4	Leftarrow
5	5	5	Home
6	6/rightarrow	6	rightarrow
7	7/Home	7	C1
8	8/uparrow	8	uparrow
9	9/Pg Up	9	C2
New Line	Enter	New Line	New Line

- 1 107-keyboard numeric keypads have no Num Lock On/Off mode.
- 2 Editing keys on the 107-key keyboard are mapped onto the numeric keypad on 101-key keyboards when Num Lock is Off. See Table 2-2 and Table 2-3 and for generated codes.

Forming Command Arguments

Command arguments, are ASCII characters. The majority of arguments in Data General native-mode commands are composed of either one, two, or three bytes. Each of these arguments are represented respectively by “n”, “nn”, or “nnn”. In command listings, the arguments are always enclosed by angle brackets (< >). However, these brackets simply separate the command argument from the surrounding characters. The brackets *do not* mean that the command arguments are in octal.

Command arguments are expressed as a version (known hereafter as DG-hex) of standard hex that replaces “A” with “:”, “B” with “;”, “C” with “<”, “D” with “=”, “E” with “>”, and replaces “F” with “?”. For example, in standard hex, the decimal value “15” is expressed as “F”. However, in DG-hex, the decimal value “15” is expressed as “?”.

CAUTION: *UNIX mode does not use DG-hex to express command arguments. For the format of command arguments in UNIX mode, see the section “UNIX Support,” later in this chapter.*



When command arguments are received by the terminal, the four least significant bits of each byte are concatenated. Thus, an <n> value has four bits; an <nn> value has eight bits; and an <nnn> value has twelve bits. The next section “Recovering a Decimal Value from DG–Hex,” has more details on this concatenation process.

Actually forming command arguments is a simple matter of expressing a decimal value in DG–hex. Table 2-5 shows how to change decimal values into DG–hex.

Table 2-5 Command Argument Translation (Decimal to DG–Hex)

ASCII Characters 0 to ?	Argument Forms and Decimal Values		
	<n	n	n>
		<n	n>
			<n>
0	0	0	0
1	256	16	1
2	512	32	2
3	768	48	3
4	1024	64	4
5	1280	80	5
6	1536	96	6
7	1792	112	7
8	2048	128	8
9	2304	144	9
:	2560	160	10
;	2816	176	11
<	3072	192	12
=	3328	208	13
>	3584	224	14
?	3840	240	15

Examples

For <n> = 12
 1st n = “<” from table
 <n> = < (DG–hex)

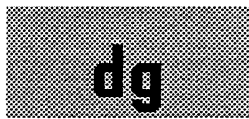
For <nn> = 135 = 128 + 7
 1st n = 128 = “8”
 2nd n = 7 = “7”
 <nn> = 87 (DG–hex)

For <nnn> = 3888 = 3840 + 48
 1st n = 3840 = “?”
 2nd n = 48 = “3”
 3rd n = 0 = “0”
 <nnn> = ?30 (DG–hex)

The largest possible decimal value for <nnn> is 4095. Every value from 0 to 4095 can be expressed as the sum of values that occur in the table, specifically, one value from each of the three columns. However, the decimal value *does not* determine the form of the argument. Rather, the *form* of the argument (n, nn, or nnn) determines the range. For example, <n> is always a single byte; <nn> is always two bytes; and <nnn> is always three bytes.

To express a decimal value in DG–hex, follow these steps.

1. Express the decimal value as the sum of up to three values, one from each column, in the table. An <n> uses only the rightmost column. An <nn> uses that column and the center one. An <nnn> uses all three columns. Locate the value in the table that is closest to, but not greater than, the decimal value you want to translate. Subtract the closest value from the original value. Take that difference and find the closest value, but not greater, to it in the table. Continue until you have expressed the original decimal value as the sum of three values from the table.

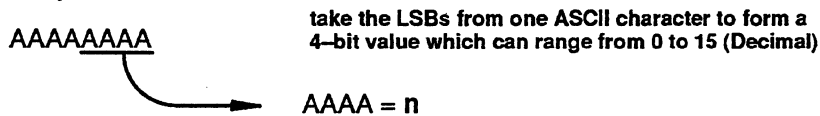


2. Locate each of the values found in the table and determine what ASCII character represents that value. Once you have determined the appropriate ASCII character for each column, string them together as shown in Table 2-5. The resulting code is in DG-hex.

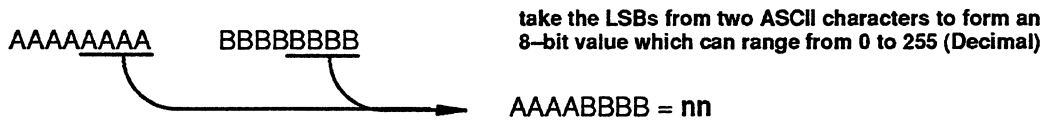
Recovering a Decimal Value from DG-hex

Some commands return parameter data to the host. In most cases, this data is in DG-hex form and will need to be converted back to decimal for application use. For the <n>, <nn>, and <nnn> argument forms, only the lower four bits of each argument byte are used. The lower four bits, known as Least Significant Bits or LSBs, of each argument byte are concatenated together to form a 4-bit value for <n>, an 8-bit value for <nn>, and a 12-bit value for <nnn>. Figure 2-1 shows how the argument bytes (8-bit ASCII characters) are evaluated.

to interpret "n" format



to interpret "nn" format



to interpret "nnn" format

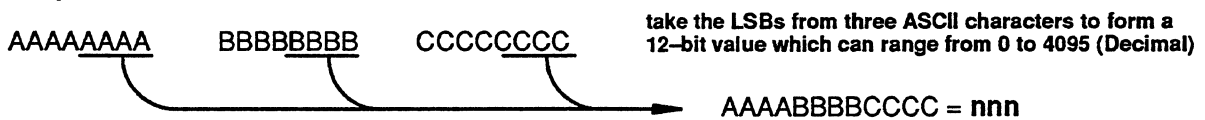


Figure 2-1 Combining LSBs to Create Decimal Values from DG-hex

Forming Location Arguments

Location arguments are used to specify x- and y-ordinates for graphics commands in Data General native-mode (not, however, in UNIX mode). They are always in the form of <NNN>, where NNN is three ASCII characters from “@” through “_”. In commands, the NNN values are enclosed within angle brackets. These brackets are only used in this manual to clearly separate the location argument from the surrounding codes; the brackets *do not* mean that location arguments are in octal. Forming location arguments is similar to forming command arguments. In both cases, you must express a decimal value as a value containing ASCII characters.

CAUTION: Location arguments in UNIX mode have a different format. Refer to the “UNIX Support” section later in this chapter for details on how UNIX mode uses location arguments.

Location arguments, which always have three bytes, are treated differently than command arguments by the terminal when they are transmitted. Each byte (8-bits) of the argument is truncated by the terminal into a 5-bit quantity by removing the three most significant bits of each byte. Then the remaining three 5-bit quantities are concatenated into a 15-bit value (rather than the 12-bit value in command arguments). For more information on how this concatenation occurs, see the next section “Generating a Decimal Value From Location Arguments.”

Forming location arguments is similar to forming command arguments. In both cases, you must express a decimal value (ranging from 0 through 3071) as a value containing one or more ASCII characters. Table 2-6 helps you transform decimal values into a form containing ASCII characters.



Table 2-6 Location Argument Translation Table (Decimal into ASCII Characters)

ASCII Characters @ to _	Argument Forms and Decimal Values		
	N	N	N
@	0	0	0
A	1024	32	1
B	2048	64	2
C	↑	96	3
D		128	4
E		160	5
F		192	6
G		224	7
H		256	8
I		288	9
J		320	10
K		352	11
L		384	12
M		416	13
N		448	14
O		480	15
P		512	16
Q		544	17
R		576	18
S		608	19
T	640	20	
U	672	21	
V	704	22	
W	736	23	
X	768	24	
Y	800	25	
Z	832	26	
[864	27	
\	896	28	
]	928	29	
^	960	30	
_	992	31	
	↓		

Examples

For <NNN> = 455
 = 0 + 448 + 7
 = @NG

For <NNN> = 1420
 = 1024 + 384 + 12
 = ALL

The largest possible decimal value for <NNN> is 3071. Every value from 0 through 3071 can be expressed as the sum of values that occur in the table, specifically, one value from each of the three columns. Remember, location arguments *always* have three bytes.



To express a location argument decimal value into a form containing ASCII characters, follow these steps:

1. Express the decimal value as the sum of three values, one from each column, in the table. Locate the value in the table that is closest to, but not greater than, the decimal value you want to translate. Then subtract the closest value from the original value. Now take that difference and find the closest value, but not greater, to it in the table. Continue until you have expressed the original decimal value as the sum of three values from the table.
2. Locate each of the values found in the table and determine what ASCII character represents that value. Once you have determined the appropriate ASCII character for each column, string them together as shown in Table 2-6.

Recovering a Decimal Value From Location Arguments

For location arguments, the terminal sends 15-bits to the host. This 15-bit quantity is interpreted by concatenating the five least significant bits from each of the three bytes that comprise the argument. Figure 2-1 shows how the argument bytes (8-bit ASCII characters) are evaluated.

location arguments are always in the <NNN> format

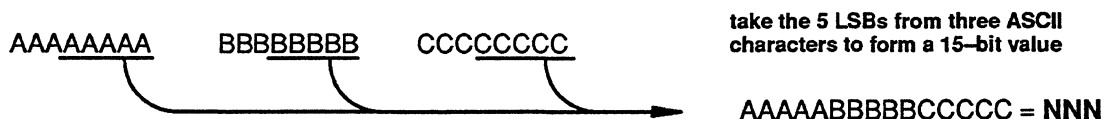


Figure 2-2 Combining LSBs to Create Decimal Values for Location Arguments

As is the case with command arguments, location arguments are also translated into a form containing only ASCII characters. Location arguments use the ASCII characters from “@” to “_” to express the 15-bit decimal value that results from concatenating the five LSBs of three bytes. Table 2-6 shows the translation values for location arguments.



Character Sets

All terminals are equipped with a 512-character Character Generator (CGEN), which is located within terminal ROM. All hard (ROM-resident) character sets are composed of characters contained within the CGEN. In addition to the predefined hard character sets, the D463 supports up to 37 soft character sets (each containing up to 94 characters), which reside in volatile RAM, that are composed of custom characters. Soft character sets store characters within the Dynamically Reconfigurable Character Buffer (DRCB), which contains up to 3504 soft characters.

Hard Character Sets

Hard character sets are composed of characters located within the CGEN. Two character sets may be selected for use at one time. They are designated as G0 (primary) and G1 (secondary). Table 2-7 shows all of the 7-bit and 8-bit character sets available in Data General native-mode.

Table 2-7 Character Sets Available In Data General Native-Mode

Keyboard language	VT Multinational (supplemental set)	
U.S. ASCII	VT Special Graphics (line drawing)	
NRC United Kingdom	Low PC Term (0 hex — 7F hex)	
NRC French	High PC Term (80 hex — FF hex)	
NRC German	ISO 8859/1.2 characters	
NRC Swedish/Finnish	DRCB Character Set 1	} D463
NRC Spanish	•	
NRC Danish/Norwegian	•	
NRC Swiss	•	
NRC Kata Kana (G0)	DRCB Character Set 38	
Data General International		
Kata Kana (G1)		
Word Processing, Greek, Math		
Line drawing		
Data General Special Graphics (PC characters)		

GL is used for 7-bit characters (20 hex through 7E hex). The GL character set is generally used with 7-bit sets such as U.S. ASCII or an appropriate National Replacement Character (NRC) set. The NRC sets are language-specific character sets that remap the least used characters from the ASCII set with those characters frequently used within each language. GL can point to G0 or G1.

The GR set is used for 8-bit character codes, which are characters A0 hex through FF hex. GR generally contains special graphics sets such as Data General International. GR is hard-wired to G1.

Powerup Character Sets

During powerup or reset, the terminal initializes the character sets with the default primary (G0) and secondary (G1) character sets. If the terminal is operating in 7-bit mode, the G0 set is the NRC set and the G1 set is the word processing set. If the terminal is operating in 8-bit mode, the G0 set is the U.S. ASCII set and the G1 set is the Data General International Set.

Designating Character Sets

Any two of the character sets listed in Table 2-7 may be designated as G0 and G1. The two sets chosen are then designated as either GL or GR. GR is always G1. GL can be either G0 or G1. When operating in 7-bit mode, you must send a Shift Out command to the terminal in order to switch GL from G0 to G1. All subsequent characters in GL are selected from G1. In order to return to G0, you must send a Shift In command to the terminal.

When operating in 8-bit mode, G1 is accessed directly by characters in the range A0 hex through FF hex. The Shift In and Shift Out commands select which character set is to be designated GL (used for characters in the range 20 hex through 7E). Figure 2-3 illustrates which sets can be designated GL or GR.

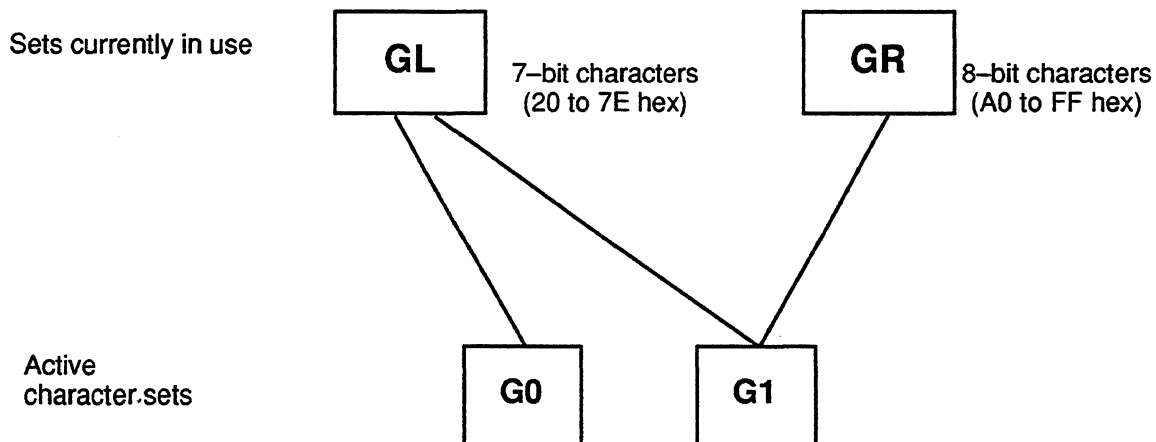


Figure 2-3 Designating Character Sets as GL and GR (Data General Native-Mode)

Soft Character Sets (D463 only)

CAUTION: Soft character bit patterns in UNIX mode have a different format. Refer to the Unix Support section for a description of how UNIX mode uses bit patterns.

Soft character sets, which contain custom characters, must be defined by the user before they can be used. Creating custom characters and generating character graphics are both functions of the Dynamically Reconfigurable Character Buffer (DRCB). This buffer, which contains up to 3504 characters, is the industry-standard term for the Down Line Loadable buffer (DLL) used in Data General terminals, such as the D450 and D460.

The D463 terminal has up to 37 soft character sets, each of 94 characters. The characters are in the range from 21 hex through 7E hex.

Custom character definitions are valid until the terminal is turned off or reset, or until the characters are redefined or deallocated. Save your custom character definitions on your host system so they can be easily transferred to the terminal, as needed.

The first step in defining one or more custom characters is to select a DRCB set number using the Select Character Set command. The DRCB set number identifies the character set to contain the custom characters. DRCB sets defined in this way can be designated as the G0 or G1 set, or as both. The D463 uses DRCB set numbers from 20 hex through 45 hex. Although this range of values is 38 decimal character sets, there is only enough RAM for 37 sets to be in use at one time.

Once you have selected the character set number, you must define the dot patterns for the characters with the Define Character command. Each character defined fits within a character cell that measures 10 by 12. The cell has 12 scan rows, each of which has 10 dots (columns) per row. Each character cell is a matrix of 120 dots, arranged in 12 horizontal rows by 10 vertical columns (under normal, noncompressed, spacing only). The Define Character command encodes the dot pattern of only one character cell. You must repeat the command for each character defined.



Defining Soft Characters

CAUTION: *Soft character bit patterns in Unix mode have a different format. Refer to the “Unix Support” section later in this chapter for details on how Unix mode uses bit patterns.*

Soft characters are defined by the Define Character command. This command contains up to twelve <dd> pairs, each representing the bit pattern of a character cell row. Each <dd> pair, which is a 10-bit quantity, is formed by concatenating the five least significant bits from each of two ASCII bytes. If you are defining a compressed character (character cell size of 6 by 12), only the six most significant bits of the 10-bit quantity are used. Figure 2-4 shows how the <dd> pair is created.

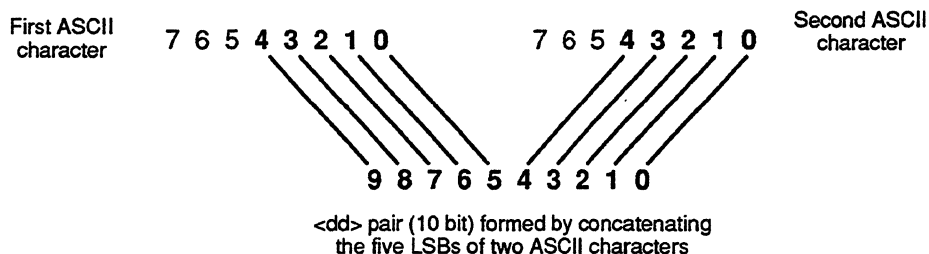


Figure 2-4 Defining a Soft Character Cell Row

Each <dd> pair specifies the bit pattern of each character cell row. In each row, the “1’s” in the <dd> pair turns screen pixels on and the “0’s” turn screen pixels off. For example, the 10-bit value (<dd> pair) 0100000001 turns on the second and last pixel in a character cell row. Figure 2-5 shows how the twelve <dd> pairs define a DRCB character, where each “d” value is a 5-bit binary quantity.

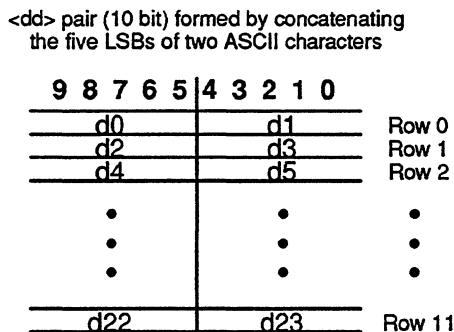


Figure 2-5 Combining <dd> Pairs to Specify a Soft Character

The <dd> pairs are sent to the host in a stream in the order: “d0, d1, ... , d23.” Remember, before a DRCB character may be defined, a DRCB character set must be selected as the current (active) character set.

Graphics

The DRCB provides the D463 user with character-graphics commands. In this case, the firmware defines DRCB characters to form the graphic image specified by a given graphics command. The D463 terminal creates graphics displays by combining line segments, arcs, filled polygons, and bars (filled in rectangles) into a composite screen image. The screen image is formed using DRCB characters defined by drawing algorithms within the terminal. When graphics images are no longer needed on the screen, the DRCB characters used to create them are automatically released and made available for custom characters (described in the previous section).

Character graphics commands supported by the D463 are listed below:

- **Deallocate Character Sets** — Lets the user free allocated DRCB character sets that were reserved by the Define Character command.
- **Bar** — Draws solid rectangles of any size, provided they fit entirely within the current window.
- **Line** — Draws lines from point to point within the current window.
- **Arc** — Draws arc with a specified radius, start angle, and end angle.
- **Polygon Fill** — Draws a filled polygon within the current window.
- **Set Pattern** — Defines the line style used in generating lines.
- **Read Characters Remaining** — Queries the terminal for a count of the DRCB characters remaining (but reports a maximum of only 1023 characters, even though more may remain).

The Graphics Coordinate System

The Line, Bar, Arc, and Polygon Fill commands are based upon an (x,y) coordinate system, where each point within the graphics coordinate system can be uniquely described by the combination of a horizontal component (the x ordinate) and a vertical component (the y ordinate).

The x-axis is defined by the bottom of the current window. The y-axis is defined by the left margin of the current window. The drawing origin (where $x=0$ and $y=0$) is, therefore, defined by the intersection of the left margin and the bottom of the last row in the current window.

Along the x-axis there are 10 x-ordinate units for each of the 81 columns, for a total of 810 units (or 2070 units for 207 columns). Along the y-axis, there are 24 y-ordinate units for each of the 24 rows, for a total of 576 units. If however, you define the status line as the 25th screen row, then there are a total of 600 units. These dimensions are derived from the 10 by 12 dot matrix in each character cell; one unit in the x-direction for each dot column (10), and two units in the y-direction for each scan row (12).

Figure 2-6 shows the margins and boundary values for the graphics coordinate system at power up or after reset. The margins and boundaries of the current window limit the available drawing area for character graphics commands.

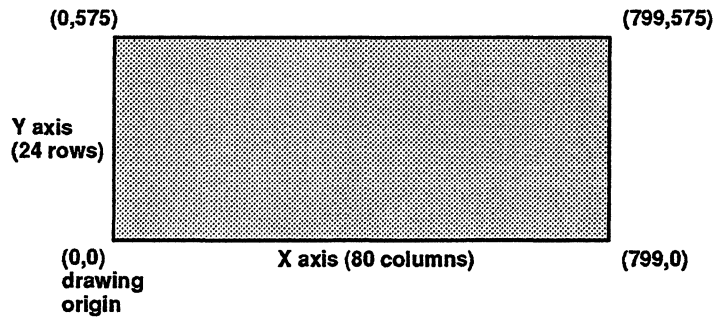


Figure 2-6 The Graphics Coordinate System at Power Up or Reset

The largest possible drawing area consists of the entire display–screen memory, with 24 rows and 207 columns. In this case, the x–ordinate can range from 0 to 2,069; the y–ordinate can range from 0 to 575. The smallest possible drawing area is one character cell (one row by one column). In this case, the x–ordinate can range from 0 to 9 and the y–ordinate can range from 0 to 23.

NOTE: If the arguments to the Line or Bar commands specify a location or dimension that extends out of the current drawing area, the command immediately aborts.

Graphics Cursor

The graphics cursor, which is available only on the D463, is used to indicate specific points (coordinates) on the screen by controlling the specified input device. This lets the user identify screen locations easily. The graphics cursor commands are similar to the G300 graphics cursor commands. The D463 does not support a blinking cursor (unlike the G300) and only supports a long crosshair cursor–type. Also, the format of data returned in response to a cursor command is different from the G300.

NOTE: The graphics cursor disappears during both vertical and horizontal scrolling. After the scrolling operation is complete, the cursor will reappear in the same location.

The Read Graphics Cursor command gives you the coordinates of the graphics cursor. Graphics Cursor On and Graphics Cursor Off cause the cursor to appear and disappear from the screen respectively. The Cursor Location command lets you move the graphics cursor to any position on the screen. Cursor Track lets you select what input device will control the graphics cursor. The Cursor Reset command sets the graphics cursor attributes to “off” and “no” tracking.



Windows

The D413/D463 terminals provide for up to 24 (25 if 25th Line Mode is set) scroll areas, called windows. Windows may have from 1 to a maximum of 25 screen rows. No overlap is permitted. A window does not provide extended or off-screen memory in the vertical direction. That is, as text in a window is vertically scrolled up or down, lines at the top or bottom of the window are lost. A window row, however, can contain more than the usual 81 visible columns; a window can be horizontally scrolled over 207 columns. The additional columns are stored within terminal memory.

A new feature of these terminals is an addressable status line. The 25th line of the screen can be blanked, used as an extra screen row, or reserved for the status line. The Set 25th Line Mode command controls the functions of the 25th line on the screen.

Each window is essentially a miniature DASHER D2/D200 screen. D2 commands work within and relative to the current window. This terminal, however, can display up to 81 columns on a screen row.

NOTE: To provide DASHER D2/D200 display compatibility, this terminal initializes the screen to consist of one window with 24 rows of 80 columns. Since all D2/D200 commands work relative to the current window, full compatibility is retained.

Lead-In Codes

Command sequences in Data General native-mode are composed of one or more ASCII characters. Commands composed of at least two characters always begin with 036 octal (1E hex), which is the 2-character command lead-in code. The remaining characters in the command are always printable ASCII characters from 041 octal (21 hex) through 176 octal (7E hex).

Invalid command sequences are ignored. For example, function-key sequences (all of which also begin with 036 octal) are not valid command sequences and are ignored if they are received by the terminal.



UNIX Support

All terminals have a UNIX mode which remaps troublesome Data General native-mode commands and keyboard codes to allow easier creation of UNIX terminfo files. This mode is entered and exited via the UNIX Mode command or the Configuration Menu. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Remapped keys that transmit inbound (terminal to host) codes are shown in Table 2-8.

Table 2-8 Remapped Key Codes for UNIX Support

Key	DG Native-mode (octal)	UNIX Mode (octal)	UNIX Mode (ASCII)
Erase Page	<014>	<036><120><110>	<036>PH
Cmd-Print	<036><021>	<036><120><060>	<036>P0
Shift Cmd-Print	<036><001>	<036><120><061>	<036>P1
Erase EOL	<013>	<036><120><105>	<036>PE
Uparrow	<027>	<036><120><101>	<036>PA
Rightarrow	<030>	<036><120><103>	<036>PC
Leftarrow	<031>	<036><120><104>	<036>PD
Downarrow	<032>	<036><120><102>	<036>PB
Home	<010>	<036><120><106>	<036>PF

Another change to support UNIX systems involves altered outbound (host to terminal) as well as inbound codes. In UNIX Mode, all Data General native-mode commands accept ASCII coded hex parameters ("0" through "9", "A" through "F", and "a" through "f"), instead of the Data General standard ASCII coded binary parameters and commands that return data will send hex data in upper case. These parameters are encoded as shown in Table 2-9.



Table 2-9 Altered Outbound Codes for UNIX Support

Data General Native-Mode	Unix Mode ¹	Bits	Range
<n>	<H>	4	0 through 15
<N>	<HH>	5	0 through 31
<nn>	<HH>	8	0 through 255
<dd>	<HHH>	10	0 through 1023
<nnn>	<HHH>	12	0 through 4095
<NNN>	<HHHH>	15	0 through 32767
<nnnn>	<HHHH>	16	0 through 65535

¹ H indicates a hex digit.

Certain Data General native-mode commands have different structures and codes in UNIX Mode. These changes, shown in Table 2-10, make the remapped keys (see Table 2-8) work in local mode and make some of the terminal descriptions easier.

Table 2-10 Altered Data General Native-Mode Commands for UNIX Support

Command	Native-Mode (octal)	UNIX Mode (octal)	UNIX Mode (ASCII)
Print Window	<021>	<036><106><077><071>	<036>F?9
Cursor Up	<027>	<036><120><101>	<036>PA
Cursor Down	<032>	<036><120><102>	<036>PB
Cursor Right	<030>	<036><120><103>	<036>PC
Cursor Left	<031>	<036><120><104>	<036>PD
Erase Field	<013>	<036><120><105>	<036>PE
Window Home	<010>	<036><120><106>	<036>PF
Roll Disable	<023>	<036><120><107>	<036>PG
Erase Window	<014>	<036><120><110>	<036>PH
Blink On	<016>	<036><120><111>	<036>PI
Blink Off	<017>	<036><120><112>	<036>PJ



Also, in UNIX Mode the control characters shown in Table 2-11 have altered meanings.

Table 2-11 Control Codes Altered for UNIX Support

Control Code (octal)	Command or Function
<005>	Ignored (Read Cursor Address uses binary codes)
<010>	Backspace (does not wrap on left margin)
<011>	Tab (fixed eight-column tab settings)
<012>	Line Feed (go to next line in same column)
<013>	Ignored
<014>	Ignored
<016>	Shift Out (still available with <036>N)
<017>	Shift In (still available with <036>O)
<021>	XON — restarts terminal transmission
<023>	XOFF — stops terminal transmission (with limited transmit <i>only</i>)
<027>	Ignored
<030>	Ignored
<031>	Ignored
<032>	Ignored

Host-Programmable Function Keys

Data General native-mode supports host-programmable function keys with the Programmable Function Keys command and through the Configuration Menu (only on D413/D463). For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

The Programmable Function Keys command allows one or more of the 60 predefined function key codes to be reprogrammed with user-defined sequences. Programmed function keys share 255 bytes of volatile (RAM) memory space. This memory can be allocated entirely to one key if desired. Keys that do not have a user definition send the default Data General sequences.

Debugging Support

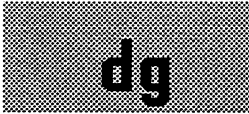
To aid in debugging operations, all terminals have a command called Data Trap Mode. This command lets the user view data from the host as a hex data-stream, similar to the AOS/VS X DISPLAY command. This command can be entered either on-line or off-line. Data Trap Mode has two operating states. One of them displays hex values and the other displays octal values.

Full data on the use of this command is in the section "Debugging Commands," later in this chapter.

Dual Emulation Support

The D413/D463 terminals can freely switch between two emulations. The manual *Installing and Operating the D216E+ / D413 / D463 Display Terminals* (014-001767) contains full information on this feature. Two commands that support dual emulations in Data General native-mode are Set Split Screen Mode and Hot Key Switch. The Set Split Screen Mode command lets you display portions of both emulations on the screen at one time. The Hot Key Switch command lets the user change from one emulation to the other.

End of Section



Data General Native–Mode Commands

This section describes the format and usage of Data General native–mode commands. This section does not explain conventions and practices of Data General native–mode operations, which are covered in the section “Data General Native–Mode Summary of Operations.”

The commands within this section are organized into the functional areas listed below:

Character Set Commands

Character Attribute Commands

Relative Cursor–Positioning Commands

Margins Commands

Screen Management Commands

Scrolling Commands

Editing Commands

Programmable Function Key Commands

Reporting Commands

Dual–Emulation Support Commands

Miscellaneous Commands

Drawing Commands

Graphics Cursor Commands

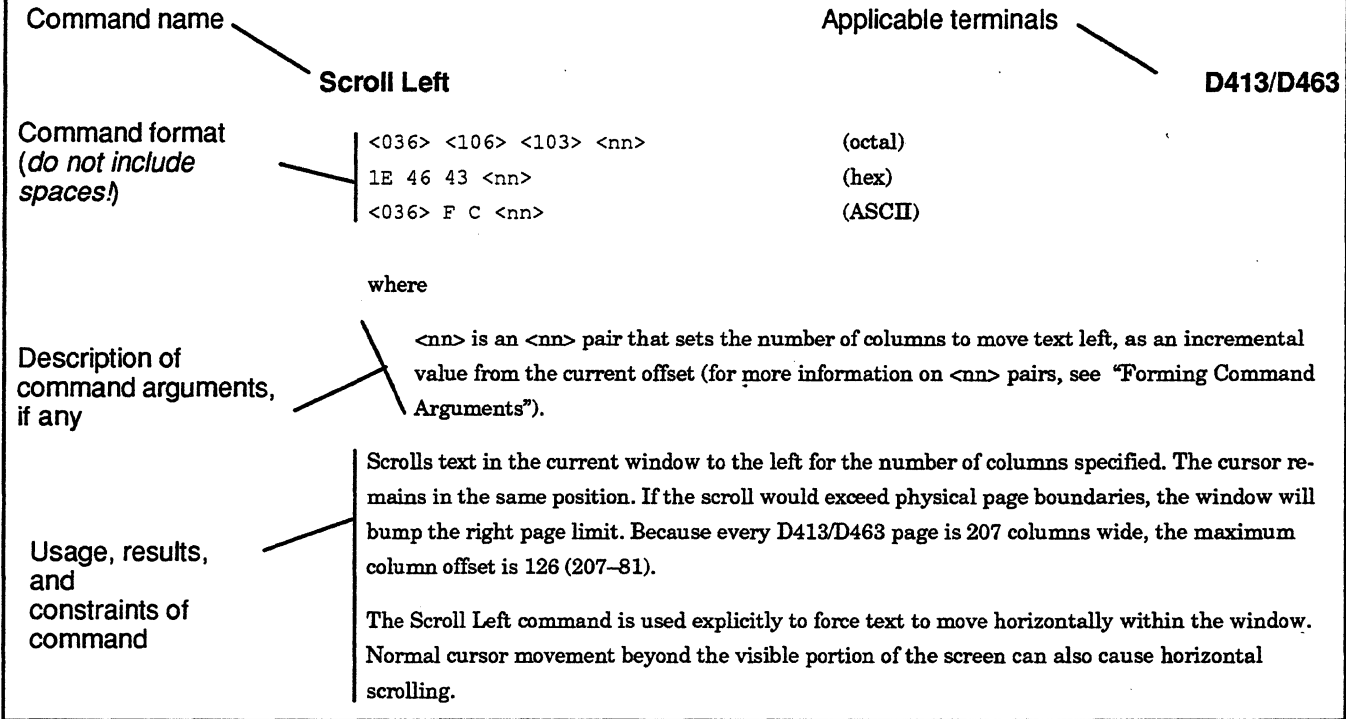
Printer Commands

Diagnostic Commands

Debugging Commands

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various Data General native–mode reference material is covered in appendices.

Format of Command Listings in this Section



Command Syntax and Code Conventions

Throughout this manual, there are certain conventions used whenever Data General native-mode commands are explained or referenced. These conventions are:

- Any value enclosed within angle brackets (< >) is in octal, *except* command or location arguments.
- Values in the form of <n>, <nn>, and <nnn> are command arguments. These values are expressed in DG-hex (except in UNIX mode). For more information on command arguments (and DG-hex), see "Forming Command Arguments."
- Values in the form of <NNN> are location arguments. These values are always expressed with ASCII characters from "@" through "_" (except in UNIX mode). For more information on location arguments, see "Forming Location Arguments."
- Spaces are often included within a command to separate characters for clarity. When entering a command, *do not* enter these spaces. If a space is part of a command, it will be written as <space>.



Character Set Commands

Not all of these character sets and commands are available on all terminals. Refer to the section “Character Sets” to get more details on character sets and their usage.

Select Character Set

all terminals

<036> <106> <123> <nn> (octal)
1E 46 53 <nn> (hex)
<036> F S <nn> (ASCII)

where

<nn> sets the character set that is being selected. The list below shows all possible character sets, which are in hex. See the “Forming Command Arguments” section for more information on <nn> pairs.

- | | |
|------------------------------------|--|
| 00 Keyboard language | 11 Line drawing |
| 01 U.S. ASCII (NRC North American) | 13 DG Special Graphics (PC characters) |
| 02 NRC United Kingdom | 14 VT Multinational |
| 03 NRC French | 15 VT Special Graphics (line drawing) |
| 04 NRC German | 1D Low PC Term (0–127) |
| 05 NRC Swedish/Finnish | 1E High PC Term (128–255) |
| 06 NRC Spanish | 1F ISO 8859/1.2 characters |
| 07 NRC Danish/Norwegian | 20 DLL Character Set 1 |
| 08 NRC Swiss | 21 DLL Character Set 2 |
| 09 NRC Kata Kana (G0) | 22 DLL Character Set 3 |
| 0E DG International character set | 23 DLL Character Set 4 |
| 0F Kata Kana (G1) | through |
| 10 Word Processing, Greek, Math | 45 DLL Character Set 38 |

This command selects the display character set for either the primary or secondary translation set (G0 or G1). Each character set consists of 94 characters in the range of 041 octal through 176 octal (or 241 octal through 376 octal for the secondary character set in 8 bit mode).



To select a character set as the primary translation set (G0), the terminal must be in the shift-out state. Conversely, to select a character set as the secondary translation set (G1), the terminal must be in the shift-in state.

Shift Out

all terminals

<036> <116> (octal)
 1E 4E (hex)
 <036> N (ASCII)

Changes to the terminal's alternate character set. This command allows access to the secondary character set with characters in the range of 041 octal through 176 octal.

Shift In

all terminals

<036> <117> (octal)
 1E 4F (hex)
 <036> O (ASCII)

Changes to the terminal's main character set. This command allows access to the primary character set with characters in the range of 041 octal through 176 octal.

Deallocate Character Sets

D463

<036> <106> <161> <nn> <nn> (octal)
 1E 46 71 <nn> <nn> (hex)
 <036> F q <nn> <nn> (ASCII)

where

the first <nn> pair sets the starting point of the range (from 00 hex to 25 hex) of character sets to deallocate.

the second <nn> pair sets the ending point of the range (from 00 hex to 25 hex) of character sets to deallocate.

See the "Forming Command Arguments" section for more information on <nn> pairs.

Lets the host return any or all of the defined character sets to the pool of free characters. This command scans a range of character set numbers and removes any of the included character sets from memory. The <nn> parameters are the character set hex codes (20 hex to 45 hex) minus 20 hex because you can deallocate only Dynamically Reconfigurable Character Buffer (DRCB) sets. For example, <036>Fq000? deallocates any used sets from 20 hex through 2F hex.



Define Character

D463

```
<036> <106> <122> <char> <dd>..<dd>      (octal)
1E 46 52 <char> <dd>..<dd>                  (hex)
<036> F R <char> <dd>..<dd>                  (ASCII)
```

where

<char> is the character to define. It is in the range from 041 octal to 176 octal.

<dd> (twelve pairs) are the bit values of the rows of the character cell. Each bit represents one dot in the row of the cell. Each bit that is set represents a lighted dot on the screen.

Allows you to define your own characters. The definition consists of a character to redefine and twelve <dd> pairs. Before a Dynamically Reconfigurable Character Buffer (DRCB) or soft character can be defined, a DRCB character set must be selected as the current (active) character set. The "Character Sets" section of this chapter has details of the process for defining soft characters.

Reserve Character

D463

```
<036> <106> <145> <n> <n>                    (octal)
1E 46 65 <n> <n>                              (hex)
<036> F e <n> <n>                              (ASCII)
```

where

the first <n> denotes the starting Dynamically Reconfigurable Character Buffer (DRCB) set (decimal value from 0 to 15). 0 decimal maps to DRCB set 20 hex and 15 decimal maps to DRCB set 2F hex.

the second <n> denotes the number of contiguous sets (decimal values from 1 to 15).

For more information on <n> values, see the "Forming Command Arguments" section of this chapter.

Deallocates DRCB character sets that were reserved by the Define Character command. The Reserve Character command can be issued prior to executing any of the drawing commands. It is no longer necessary to allocate characters for drawing.



Read Characters Remaining

D463

```
<036> <106> <144>          (octal)
1E 46 64                    (hex)
<036> F d                    (ASCII)
```

In response to this command, the terminal sends back the following five character sequence:

```
<036> <157> <071> <high> <low>      (octal)
1E 6F 39 <high> <low>              (hex)
<036> o 9 <high> <low>              (ASCII)
```

where

<high> is the upper 5 bits of a 10-bit count

<low> is the lower 5 bits of 10-bit count

Both of these parameters, when strung together, are similar to the <dd> pair used in the Define Character command. See the "Defining Soft Characters" section for more information on <dd> pairs.

NOTE: The count is returned to the user in the form of a base 32 numbering system where the ASCII characters "@", "A" through "O" and "_" represent the 32 digits. If more than 1023 characters remain, the terminal will respond with 1023.

Returns a count of the remaining characters (from the terminal to the host) that can be used to create a drawing or to allocate to character sets (the free list). If the user runs out of allocated characters, the terminal will cease processing the drawing commands.



Character Attribute Commands

Change Attributes

D413/D463

```
<036> <106> <116> <nnn> <n> <n>          (octal)
1E 46 4E <nnn> <n> <n>                    (hex)
<036> F N <nnn> <n> <n>                    (ASCII)
```

where

the first <nnn> group is the number of characters to change, starting at the cursor position. If you try to change characters past the end of the window, the command executes to the end of the window and then terminate.

the first <n> value is the attributes-to-set parameter. There are four attributes, each represented by a single bit:

bit	3	Dim
bit	2	Reverse video
bit	1	Underscore
bit	0	Blink

the last <n> is the attributes-to-reset parameter. If the same attribute is set in both the set and the reset, the attribute will be toggled. If the same bit in both the set parameter and the reset parameter is 0, the attribute will be left unchanged.

For more information on <nnn> and <n> values, see the "Forming Command Arguments" section of this chapter.

Sets, resets, or toggles visual attributes (dim, reverse video, underscore, and blink) of characters between the margins. It starts with the character at the cursor position and continues until the character count supplied is exhausted, wrapping to the next line if it encounters the right margin. However, this command executes within *only* the current window.

NOTE: The display attributes of protected text may be altered by this command since protection is not associated with a display attribute.



Dim On **all terminals**

<034>	(octal)
1C	(hex)
Ctrl-\	(ASCII)

Turns on the dim attribute for each subsequent character received by the terminal.

Dim Off **all terminals**

<035>	(octal)
1D	(hex)
Ctrl-]	(ASCII)

Turns off the dim attribute for each subsequent character received by the terminal.

Blink On **all terminals**

<016>	(octal)
0E	(hex)
Ctrl-N	(ASCII)

Turns on the blink attribute for each subsequent character received by the terminal. See the **Blink Enable** command.

Blink Off **all terminals**

<017>	(octal)
0F	(hex)
Ctrl-O	(ASCII)

Turns off the blink attribute for each subsequent character received by the terminal.

Blink Enable **all terminals**

<003>	(octal)
03	(hex)
Ctrl-C	(ASCII)

Allows blinking of characters which have the blink attribute turned on.



Blink Disable

all terminals

<004>	(octal)
04	(hex)
Ctrl-D	(ASCII)

Disables character blinking regardless of the state of the blink attributes assigned to displayed characters.

Underscore On

all terminals

<024>	(octal)
14	(hex)
Ctrl-T	(ASCII)

Turns on the underscore attribute for each subsequent character received by the terminal.

Underscore Off

all terminals

<025>	(octal)
15	(hex)
Ctrl-U	(ASCII)

Turns off the underscore attribute for each subsequent character received by the terminal.

Reverse Video On

all terminals

<036>	<104>	or	<026>	(octal)
1E	44	or	16	(hex)
<036>	D	or	Ctrl-V	(ASCII)

Turns on the reverse video attribute for each subsequent character received by the terminal.

Reverse Video Off

all terminals

<036>	<105>	or	<002>	(octal)
1E	45	or	02	(hex)
<036>	E	or	Ctrl-B	(ASCII)

Turns off the reverse video attribute for each subsequent character received by the terminal.



Protect On

D413/D463

<036> <106> <114>	(octal)
1E 46 4C	(hex)
<036> F L	(ASCII)

Sets the protect attribute for all subsequent characters that are received by the terminal. If protect mode is disabled, the characters will have the attribute set, but will not be protected until protect is enabled.

Protect Off

D413/D463

<036> <106> <115>	(octal)
1E 46 4D	(hex)
<036> F M	(ASCII)

Resets the protect attribute for each subsequent character that is received by the terminal.

Protect Enable

D413/D463

<036> <106> <126>	(octal)
1E 46 56	(hex)
<036> F V	(ASCII)

Enables all protected text so that the protected text cannot be changed with normal cursor-related commands. Protected text can be deleted only with an Erase Screen or Erase Window command.

Protect Disable

D413/D463

<036> <106> <127>	(octal)
1E 46 57	(hex)
<036> F W	(ASCII)

Allows the modification of protected text that is on the screen. Once the protect attribute is disabled, any command can modify the text.



Double High/Double Wide

D413/D463

<036> <122> <105>	(octal)
1E 52 45	(hex)
<036> R E	(ASCII)

Where the <n> value is one of the following row attributes parameters:

0	Normal Row
1	Double Wide
2	Double High Top
3	Double High Bottom
4	Double High Top/Double Wide
5	Double High Bottom/Double Wide

Selects double high and double wide screen row attributes. Double High Top selects one character row as the top half of the line. Double High Bottom is used to select the next lower character row as the bottom half of the line. Both rows should contain the same characters to form a complete display.

Field Attributes

D413/D463

<036> <122> <103> <s> <r>	(octal)
1E 52 43 <s> <r>	(hex)
<036> R C <s> <r>	(ASCII)

Where:

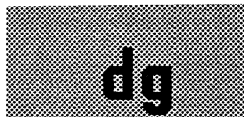
The <s> value is a <nn> pair that represents the field attribute to set and the <r> value is a <nn> pair that represents the field attribute to reset. There are four attributes, each represented by a single bit:

bit	4	Blank (invisible)
bit	3	Dim
bit	2	Reverse video
bit	1	Underscore
bit	0	Blink

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

NOTE: If the same bit is set in both <s> and <r>, then that bit will toggle. If the same bit in <s> and <r> is set to 0 that bit will not change.

Replaces the character at the cursor location with a field attribute marker. To remove a marker, position the cursor on the marker and type any character. The Field Attribute setting changes



when the next Field Attribute, End of Line, or Page is encountered in the display screen. The character, field, and page attribute settings determine the displayed attribute by exclusive OR as shown in the example below.

Character BURD	Field BURD	Page BURD	Display BURD
0101	1111	0000	1010
0111	1000	1111	0000

Page Attributes

D413/D463

<036> <122> <104> <s> <r> (octal)
1E 52 44 <s> <r> (hex)
<036> R D <s> <r> (ASCII)

Where:

The <s> value is a <nn> pair that sets page attributes and the <r> value is an <nn> pair that resets page attributes. There are four attributes, each represented by a single bit:

bit	3	Dim
bit	2	Reverse video
bit	1	Underscore
bit	0	Blink

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

The Field Attribute Extend bit determines whether the Field Attributes effect terminates at the end of the row or wraps until the next Field Attributes or end of the screen is encountered. Since the Field Attribute Extend and Blank bits are shared in dual emulations, changing these bits may affect the other emulation in split screen mode.

NOTE: If the same bit is set in both <s> and <r>, then that bit will toggle. If the same bit in <s> and <r> is set to 0, then that bit will not change.

Relative Cursor–Positioning Commands

Cursor Right

all terminals

<030>	(octal)
18	(hex)
Ctrl-X	(ASCII)

Moves the cursor one column to the right. If the cursor is at the right margin, the screen executes a New Line. If the command causes the cursor to move onto a protected character and protect mode is enabled, the command is repeated until the first unprotected character is encountered. If all character positions in the window are protected, the entire window is scanned for an unprotected character, beginning at the current cursor position. Upon reaching the character at which the Cursor Right was initiated, the screen executes a Cursor Right as if no characters were protected. However, no data is altered.

Cursor Left

all terminals

<031>	(octal)
19	(hex)
Ctrl-Y	(ASCII)

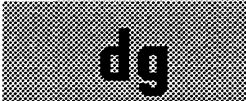
Moves the cursor one column position to the left. If the cursor is at the left margin of a line, it moves to the right margin and the screen executes a Cursor Up. If the command causes the cursor to move onto a protected character and protect mode is enabled, the command is repeated until the first unprotected character is encountered. If all character positions in the window are protected, the entire window is scanned for an unprotected character, beginning at the current cursor position. Upon reaching the character at which the Cursor Left was initiated, the screen executes a Cursor Left as if no characters were protected. However, no data is altered.

Cursor Up

all terminals

<027>	(octal)
17	(hex)
Ctrl-W	(ASCII)

Moves the cursor up one line while remaining in the same column. If the cursor is on the top row of a window, it moves to the bottom row of the window. If the cursor moves onto a protected character and protect mode is enabled, the screen executes a Cursor Left.



Cursor Down

all terminals

- <032> (octal)
- 1A (hex)
- Ctrl-Z (ASCII)

Moves the cursor down one line while remaining in the same column position. If the cursor is on the bottom row of the window, it moves to the top row of the same window. If the cursor moves onto a protected character and protect mode is enabled, the screen executes a Cursor Right.

New Line

all terminals

- <012> (octal)
- 0A (hex)
- Ctrl-J (ASCII)

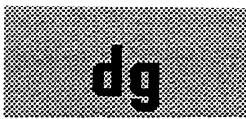
Moves the cursor to the left margin of the next row. If the cursor is on the last row of a window and Roll Mode is enabled, a Scroll Up is performed. If the cursor is on the last row of a window and Roll Mode is *not* enabled, a Home command is performed. If the command causes the cursor to move onto a protected character and protect mode is enabled, the screen executes a Cursor Right.

Carriage Return

all terminals

- <015> (octal)
- 0D (hex)
- Ctrl-M (ASCII)

Moves the cursor to the left margin of the same line. If the characters at the left margin are protected and protect mode is enabled, the screen executes a Cursor Right.



Margins Commands

Set Margins

D413/D463

```
<036> <106> <130> <nn> <nn>          (octal)
1E 46 58 <nn> <nn>                    (hex)
<036> F X <nn> <nn>                    (ASCII)
```

where

the first <nn> pair is the desired left margin. The left margin must be less than or equal to the right margin and in the range 0 to 206 decimal or the command is ignored.

the second <nn> pair is the desired right margin. The right margin must be greater than or equal to the left margin and less than or equal to 206 decimal. If out of this range, the command is ignored.

For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.

Resets the margins to the new values specified. The cursor is moved to the left margin.

Set Alternate Margins

D413/D463

```
<036> <106> <131> <nn> <nn> <nn>      (octal)
1E 46 59 <nn> <nn> <nn>                (hex)
<036> F Y <nn> <nn> <nn>                (ASCII)
```

where

the first <nn> pair is the row to place the cursor on, and must be in the range of decimal values from 0 to 24 decimal (for 25 line screens), or 255 decimal which means no row movement. If the row is out of range, the last row of the window is used.

the second <nn> pair is the column of the new left margin. This margin is relative to the previously set left margin. If the column is the decimal value 255, then the last value for the left margin is used. If the new left margin is out of range, the command is aborted.

the third <nn> pair is the column of the right margin, which must adhere to the rules defined by the Set Margins command. If the new right margin is greater than the previously set right margin, the previously set right margin is used.

For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.



Temporarily reassigns the margins to new margins while saving the previously set margins. The terminal then positions the cursor to the relative row and the relative left margin. If the row specified equals 255, the current window row is used (current row position of the cursor). Normal margins are saved and new margins are set to the left and right values specified.

NOTE: If Set Alternate Margins has already been issued, the previous alternate margins are *not* saved — only the original margins are saved. The previous alternate margins are lost in this case.

The alternate left and right margins are relative to the left margin set by the Set Margins command. Automatic horizontal scrolling of the view-port is disabled until the next Restore Normal Margins or Horizontal Scroll On command is encountered.

Restore Normal Margins

D413/D463

<036> <106> <132>	(octal)
1E 46 5A	(hex)
<036> F Z	(ASCII)

Restores the margins saved when the Set Alternate Margins command was executed. If horizontal scrolling is off as a result of a Set Alternate Margins command, it will be turned back on.



Screen Management Commands

Write Window Address

all terminals

<020> <col> <row> (octal)
10 <col> <row> (hex)
Ctrl-P <col> <row> (ASCII)

where

<col> sets the new column position of the cursor, relative to the left margin. If the parameter is equal to 177 octal, then the current value for the column is used.

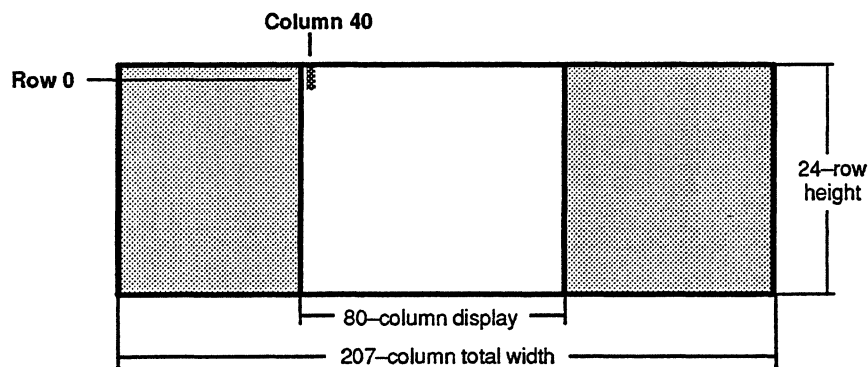
<row> sets the new row position of the cursor, relative to the top of the window. If the parameter is equal to 177 octal, then the current value for row is used.

Both values are entered in octal (raw, unencoded binary). Also, if the destination is in a protected area, the cursor is positioned to the specified position and the screen executes a Cursor Right. The cursor will be positioned at the address specified in the command after the Cursor Right sequence is completed.

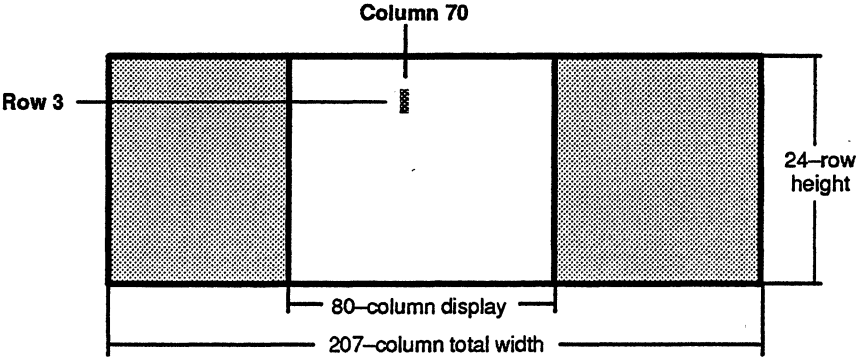
Positions the cursor to the requested row and column of the window. Window address coordinates are relative to the left margin and the top of the window. The left margin is position 0, with the maximum value being the right margin minus the left margin. The top row of the window is row 0. The last row of the window is equal to the length of the window minus one. If you try to move to a position past the last row of a window, the cursor pegs at the end of the window. If you try to move to a position beyond the right margin, the cursor pegs at the right margin.

If the command would place the cursor beyond the last column of the displayed area, the terminal horizontally scrolls to accommodate the cursor unless horizontal scrolling is disabled. If the value of an argument is 177 octal, the terminal interprets this to mean the current value.

In the following example, begin with an initial screen configuration as shown below.



When using the Write Window Address command, the ordinates are relative to the *current* left margin. For example, in the illustration above (initial configuration), the left margin is 40 and the right margin is 120. The cursor is at Row 0, Column 40. A Write Window Address to Row 3, Column 30 is issued, and the resulting position of the cursor is Row 3, Column 70, as shown below.



Window Home

all terminals

- <010> (octal)
- 08 (hex)
- Ctrl-H (ASCII)

Moves the cursor to the left margin of row 0 of the window. The command is automatically executed each time an Erase Window is issued. If the home position is protected and protect mode is enabled, the screen executes a Cursor Right.



Set Windows

D413/D463

<036>	<106>	<102>	<nn>	<n>	.	.	.	<nn>	<n>	(octal)
1E	46	42	<nn>	<n>	.	.	.	<nn>	<n>	(hex)
<036>	F	B	<nn>	<n>	.	.	.	<nn>	<n>	(ASCII)

where

<nn> sets the length of the window and is in the range of decimal values from 0 to 24.

<n> sets the spacing type:

- 0 normal (81 columns) character spacing
- 1 compressed (135 columns) character spacing.

For more information on <nn> pairs and <n> codes, see the "Forming Command Arguments" section of this chapter.

Lets the user specify the number of windows and number of rows associated with each window on the D413/D463 screen. It is possible to specify up to 24 windows or screen breaks with this command (or 25 windows if 25th Line Mode enabled).

Windows are specified by setting the number of rows associated with each window. For example, the sequence <036>FB0>0080020 designates three windows with lengths of 14, 8, and 2 rows respectively, each with 81 characters per row. The command can be terminated early by 000, which allocates the remaining rows to the last window. For example, the sequence 0:0080000 designates three windows with row lengths of 10, 8 and 6; the last window is created by default to hold the remaining rows of the screen ($24 - 10 - 8 = 6$). If the sum of the rows specified exceeds 24 (25 if the status line is configured as a screen row), the last window will contain the remaining portion of the screen. For example, the sequence <036>FB0:00:00:0 specifies three windows with row lengths of 10, 10 and 4.

When defining or redefining windows, current text on the screen is not lost. It is therefore possible to split an existing group of rows into multiple windows. Conversely, it is possible to join the text from two or more adjoining windows into one.

Since the text within windows may be horizontally scrolled independently of each other, the Set Windows command will align all text on the screen in accordance with the Left Margin. If horizontal scrolling is off, it is turned back on before the windows are set. For example, suppose two windows were currently defined on the D413/D463 screen, each with 12 rows. The top window is aligned in normal fashion (for example, window column 0 is aligned with screen column 0 and the left margin is zero). Text in the bottom window, however, has been horizontally scrolled to the left. Issuing a new Set Windows command causes the bottom 12 lines of the screen to align with the top window before the new screen breaks are enforced. A Screen Home command is executed as the last function of the Set Windows command.



Set 25th Line Mode

all terminals

<036> <106> <172> <n> (octal)
1E 46 7A <n> (hex)
<036> F z <n> (ASCII)

where

<n> sets the mode:

- 0 25th line displays a status line
- 1 displays a message on the status line. To enter the message, simply append an <nn> pair to the command (which specifies the number of characters to display) and string the specified number of characters together after the <nn> pair.
For example: <036>Fz104Text
- 2 25th line is used as an extra screen row. The last window will be lengthened by one to include this line (the 25th row of the screen).
- 3 25th line is blanked.

See the "Forming Command Arguments" section of this chapter for more information on <n> codes and <nn> pairs.

This command sets the usage of the status line.

Push

D413/D463

<036> <106> <150> (octal)
1E 46 68 (hex)
<036> F h (ASCII)

Saves only the character attributes, terminal modes, and screen boundaries described below:

Character attributes

Blink (on or off)
Underscore (on or off)
Reverse video (on or off)
Dim (on or off),
Protection (on or off).

Terminal modes

Blink mode (enabled or disabled)
Protect mode (enabled or disabled)
Scroll rate (jump, slow, fast)
7/8 bit mode (7 or 8 bit)
Cursor type (none, underscore, blinking underscore, reverse, or blinking reverse)
Roll mode (enabled or disabled).



Screen boundaries

Left margin

Right margin

Window definitions including the horizontal offset and current window.

After a Push–Pop sequence, the screen remains stationary unless a Horizontal Scroll Enable or margin control command follows a Push command.

Pop

D413/D463

<036> <106> <151>	(octal)
1E 46 69	(hex)
<036> F i	(ASCII)

Saves only the character attributes, terminal modes, and screen boundaries described below:

Character attributes

Blink (on or off)

Underscore (on or off)

Reverse video (on or off)

Dim (on or off)

Protection (on or off).

Terminal modes

Blink mode (enabled or disabled)

Protect mode (enabled or disabled)

Scroll rate (jump, slow, fast)

7/8 bit mode (7 or 8 bit)

Cursor type (none, underscore, blinking underscore, reverse, or blinking reverse)

Roll mode (enabled or disabled).

Screen boundaries

Left margin

Right margin

Window definitions including the horizontal offset and current window.

After a Push–Pop sequence, the screen remains stationary unless a Horizontal Scroll Enable or margin control command follows a Push command. After a Pop command, the terminal executes a Window Home command.



Write Screen Address

all terminals

```

<036> <106> <120> <nn> <nn>    (octal)
1E 46 50 <nn> <nn>              (hex)
<036> F P <nn> <nn>             (ASCII)

```

where

the first <nn> pair sets the absolute screen column within which to place the cursor. It must be in the range 0 to 206, or 255 decimal. If 255 is used, the terminal interprets the parameter to mean the current cursor column.

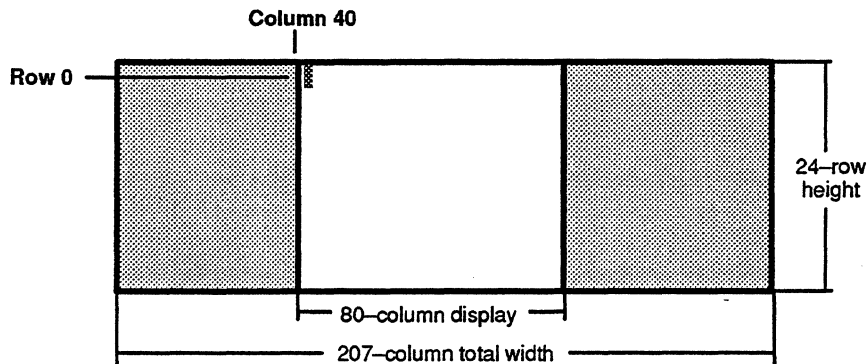
the second <nn> pair sets the absolute row within which to place the cursor. It must be in the range 0 to 23 decimal (0 to 24 if 25th Line Mode is set), or 255 decimal. If the 255 is used, the terminal interprets the parameter to mean the current cursor row.

Also, if either argument is out of range, the cursor pegs at the boundary encountered (at the margin if the column argument is outside the current margin and at the screen boundary if the row argument is out of range).

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

The write screen address command positions the cursor anywhere on the screen relative to the (0,0) position, *not* screen home (left margin,0). If the positioning of the cursor moves the cursor off the screen, the screen scrolls horizontally (unless horizontal scrolling is turned off). If the command causes the cursor to move onto a protected character and protect mode is enabled, the screen executes a Cursor Right. The cursor will be positioned at the address specified after the Cursor Right is completed. This may require a long time if the entire screen is filled with protected characters. When using Write Screen Address, the vertical range is 0 to 23 decimal (24 if 25th Line Mode is set) and the horizontal range is 0 to 206 decimal. If the range of the arguments takes the cursor outside the margins, the cursor pegs at the margin.

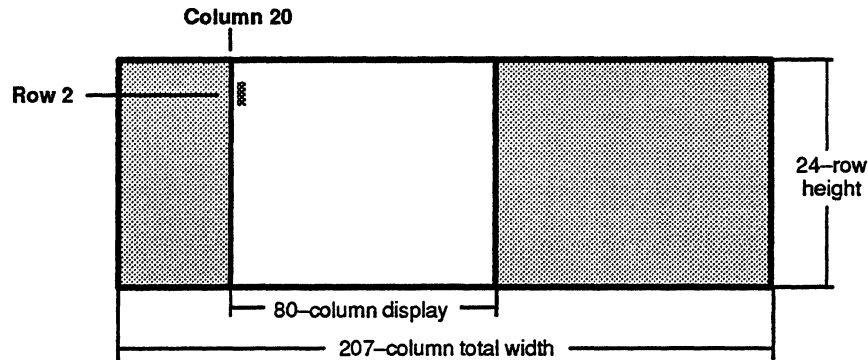
In the following example, begin with an initial screen configuration as shown below.



With the above configuration, a Write Screen Address to Column 20, Row 2 scrolls the window text twenty columns to the left (to Column 20) and places the cursor on the third row (Row 2) of



the screen, as shown in the next illustration. Remember that the first row of the screen is Row 0, so the third row of the screen is Row 2.



NOTE: This command may select a new window. If the new destination of the cursor is within a new window, the new window boundaries take effect immediately. Scrolling and cursor movement are restricted to the new window.

Select Compressed Spacing

D413/D463

<036> <106> <113>	(octal)
1E 46 48	(hex)
<036> F K	(ASCII)

Compresses character spacing for the current window to allow you to view all columns of a wide form (up to 135 columns) at one time. When you select compressed spacing, the columns displayed depend on the absolute column number of the leftmost column displayed with normal spacing. For example, when the leftmost absolute column number ranges from 0 to 72, compressed spacing displays the leftmost 135 columns. When the leftmost column number ranges from 72 to 126, compressed spacing displays the rightmost 135 columns. Margin settings are unchanged.



Select Normal Spacing

D413/D463

```
<036> <106> <112>      (octal)
1E 46 4A                (hex)
<036> F J                (ASCII)
```

Sets character spacing for the current window to allow you to view up to 81 columns at a time. If you select normal spacing and horizontal scrolling is disabled, 81 columns are displayed to the right of and including the leftmost column displayed with compressed spacing.

If you select normal spacing and horizontal scrolling is enabled, the columns displayed depend on the location of the cursor. The 81 columns will be displayed if the cursor is located in the leftmost 81 columns of the screen. If the cursor is not located in the first 81 columns of the screen, the column containing the cursor and the 80 columns to the immediate left of that column are displayed. The column containing the cursor becomes the rightmost column on the screen.

Margin settings are unchanged.

Named Save/Restore Cursor

D413/D463

```
<036> <106> <175> <n> <n>      (octal)
1E 46 7D <n> <n>                (hex)
<036> F } <n> <n>                (ASCII)
```

where

the first <n> sets the memory number to be used (0 through 15 decimal).

the second <n> has one of the following values:

- 0 saves the current position
- 1 restores the position from memory

See the "Forming Command Arguments" of this chapter for more information on <n> quantities.

Provides 16 absolute cursor position save areas that may be individually accessed. If a memory that has not yet been saved into is restored, the cursor is sent to location (0,0). Any location can be restored multiple times.



Save/Restore Screen Contents

D413/D463

<036> <106> <163> <n>	(octal)
1E 46 73 <n>	(hex)
<036> F s <n>	(ASCII)

where

<n> sets which command to enact:

- 0 command specified is Save Screen Contents
- 1 command specified is Restore Screen Contents

For more information on <n> values, see the “Forming Command Arguments” section of this chapter.

Saves the contents of the screen to a secondary save area or restores an old copy of the screen. The Save Screen Contents and Restore Screen Contents commands are only valid when they are enabled in the Terminal Configuration Menu. If the terminal is not in dual host mode (the emulation is set to “Device” in the Configuration Menu), then all of the display will be saved (using the inactive emulation’s memory). Otherwise, if another emulation is defined, the physical row length is cut in half (to 103 columns) and the screen is saved on undisplayed rows. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

When you issue a Save Screen Contents or Restore Screen Contents command, window definitions are pushed; then popped without interference with the operation of the Push/Pop command. Currently, the Save/Restore Screen Contents command does not support protected characters or graphics.

NOTE: Only the last save of the screen contents can be restored, and multiple restores can occur for each save. Graphics and Dynamically Reconfigurable Character Buffers (DRCB) are not saved.

Screen Home

D413/D463

<036> <106> <107>	(octal)
1E 46 47	(hex)
<036> F G	(ASCII)

Returns the cursor to the absolute 0 row, left margin of the screen. The current window is changed to the first window of the screen, and that window is scrolled to the left margin as necessary. If the home position is protected and protect mode is enabled, the screen executes a Cursor Right.



Set Row Length

D413/D463

<036>	<122>	<100>	<nn>	(octal)
1E	52	40	<nn>	(hex)
<036>	R	@	<nn>	(ASCII)

where

<nn> sets the row length. For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.

Alters the physical row length used for partitioning screen RAM. The number of available rows is recalculated based on the new row length and the data on the screen is preserved as much as possible (if the lines are lengthened, then rows will be removed and blanks added to remaining rows; if the lines are shortened, then blank rows will be added after the end of the screen and data in columns past the new maximum right margin are lost). If the row length is set to fewer than 135 columns, then compressed mode is disabled. This command is not currently useful since the Set Windows command cannot take advantage of these added lines.



Scrolling Commands

Show Columns

D413/D463

```
<036> <106> <137> <nn> <nn>      (octal)
1E 46 5F <nn> <nn>                  (hex)
<036> F _ <nn> <nn>                  (ASCII)
```

where

the first <nn> pair sets the left most column of the area to display, specified in absolute ordinates (0 to 206 decimal).

the second <nn> pair sets the right most column of the area to display, specified in absolute ordinates (0 to 206 decimal).

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

Scrolls text in the window horizontally the minimum amount necessary so that the columns in the specified range are visible. If these columns are visible in the viewing area, the window is not moved. If the distance between the left and right columns is greater than the width of the window (81 columns) the command performs a minimal horizontal scroll so the left column is on the left side of the screen.

Set Scroll Rate

D413/D463

```
<036> <106> <124> <n>                (octal)
1E 46 54 <n>                          (hex)
<036> F T <n>                          (ASCII)
```

where

<n> sets the scroll rate:

- 0 disables smooth scroll; makes all scrolling operations jump in the same fashion as the DASHER/D2 display.
- 1 enables smooth scroll at a maximum rate of 5 character rows per second. This rate permits easy reading of text as it is continuously displayed.
- 2 enables smooth scroll at a maximum rate of 10 character rows per second. This rate is useful for scanning of long documents.

Lets the host select one of three scroll rate options. If the argument specified does not fall within the inclusive range 0 to 2 (060 to 062 octal), the terminal defaults to jump scroll.



Scroll Down

D413/D463

```
<036> <111>          (octal)
1E 49                (hex)
<036> I              (ASCII)
```

Moves text in the window down one line and inserts a blank line at the top of the window. The cursor remains fixed on the screen. The line at the bottom of the window is lost. If a protected field is encountered, the screen executes a Cursor Right command. If horizontal scrolling is enabled and the cursor is off the screen and encounters a protected character, then the screen scrolls horizontally. If no such character is encountered, then no horizontal scrolling occurs. The command is valid even if the terminal is currently in Roll Disable Mode.

Scroll Up

D413/D463

```
<036> <110>          (octal)
1E 48                (hex)
<036> H              (ASCII)
```

Moves text in the window up one line and inserts a blank line at the bottom of the window. The cursor remains fixed on the screen. The line at the top of the window is lost. If a protected field is encountered, the screen executes a Cursor Right command. If horizontal scrolling is enabled and the cursor is off the screen and encounters a protected character, then the screen scrolls horizontally. If no such character is encountered, then no horizontal scrolling occurs. The command is valid even if the terminal is currently in Roll Disable Mode. New Line and Cursor Right commands do not scroll in Roll Disable Mode; however, a Scroll Up command always results in scrolling.

Scroll Left

D413/D463

```
<036> <106> <103> <nn>      (octal)
1E 46 43 <nn>              (hex)
<036> F C <nn>            (ASCII)
```

where

<nn> sets the number of columns to move text left, as an incremental value from the current offset. For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

Scrolls text in the current window to the left for the number of columns specified. The cursor remains in the same position. If the scroll exceeds physical page boundaries, the window bumps the right page limit. Because every D413/D463 page is 207 columns wide, the maximum column offset is 126 (207 - 81). If horizontal scrolling is disabled, then this command is ignored.



The Scroll Left command is used explicitly to force text to move horizontally within the window. Normal cursor movement beyond the visible portion of the screen can also cause horizontal scrolling.

Scroll Right

D413/D463

<036>	<106>	<104>	<nn>	(octal)
1E	46	44	<nn>	(hex)
<036>	F	D	<nn>	(ASCII)

where

<nn> sets the number of columns to incrementally offset the text. For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.

Moves text within the current window to the right. If the window movement violates physical page boundaries (for example, attempting to view something to the left of the first column), the window’s leftmost column in the window is column 0. If horizontal scrolling is disabled, then this command is ignored.

Horizontal offset is volatile after this command is executed if the cursor is moved off the screen. The next cursor movement or displayed character may cause an undesired horizontal scroll. To avoid this, use the Horizontal Scroll Disable command immediately after issuing a Scroll Right (or Left) command.

Roll Enable

all terminals

<022>	(octal)
12	(hex)
Ctrl-R	(ASCII)

Turns on Roll Mode. In Roll Mode, each time a command is issued that moves the cursor beyond the bottom of the window, the screen scrolls up one line. The cursor then moves to the new line. Information previously displayed on the top row is lost. When the terminal is powered up, Roll Mode is the default state. The Cursor Down command is an exception because it always wraps to the top line of the window.



Roll Disable

all terminals

<023>	(octal)
13	(hex)
Ctrl-S	(ASCII)

Turns off Roll Mode. With Roll Mode disabled, each time a command is issued that moves the cursor beyond the bottom of the window, the cursor wraps to the top line of the current window.

Horizontal Scroll Disable

D413/D463

<036> <106> <135>	(octal)
1E 46 5D	(hex)
<036> F]	(ASCII)

In normal operation, the terminal horizontally scrolls the window to keep the cursor within the displayed area of the screen. The Horizontal Scroll Disable command disables that (default) scrolling. This command can be used to paint an area of the screen without distracting the operator by scrolling the window. The Set Alternate Margins command also turns off horizontal scrolling.

Horizontal Scroll Enable

D413/D463

<036> <106> <136>	(octal)
1E 46 5E	(hex)
<036> F ^	(ASCII)

Restarts horizontal scrolling. If the cursor is off the screen, the screen is scrolled the minimum distance necessary to display the cursor. This is the default power-up setting.



Editing Commands

Erase Window

all terminals

<014>	(octal)
0C	(hex)
Ctrl-L	(ASCII)

Erases all characters in a window and performs a Window Home command . All character attributes, including protection, are cleared. If the left margin is off the screen, the window scrolls horizontally to accommodate the new cursor position.

Erase Screen

D413/D463

<036> <106> <105>	(octal)
1E 46 45	(hex)
<036> F E	(ASCII)

Erases all text in the terminal regardless of current window definitions or horizontal window alignments. The cursor is then moved to the first row of the screen, at the left margin. All window breaks are left unchanged. The window may scroll horizontally to accommodate the new cursor position. All character attributes, including protection, are cleared.

Erase to End of Line

all terminals

<013>	(octal)
0B	(hex)
Ctrl-K	(ASCII)

Erases all text from the cursor to the right margin. If protect mode is enabled and a protected character is encountered before the right margin is reached, that protected character is treated as the right margin. The cursor position is left unchanged.

Insert Line

D413/D463

<036> <106> <110>	(octal)
1E 46 48	(hex)
<036> F H	(ASCII)

Inserts a row in a window. Text on rows beneath the cursor, including the row on which the cursor resides, is moved down one row within the window. Protected fields are moved down and stay with the text rows they are on before the insert line command was issued. The entire cursor row is blanked and the cursor remains fixed on the screen.



Delete Line

D413/D463

```
<036> <106> <111>      (octal)
1E 46 49                (hex)
<036> F I               (ASCII)
```

Deletes the row the cursor is currently on. Text on rows beneath the cursor row is moved up one row within the window. Protected fields are moved up and stay with the text rows they were on before the Delete Line command was issued. The entire cursor row is lost and a blank row appears at the bottom of the window. The cursor remains fixed on the screen unless a protected field is encountered. If the cursor encounters a protected field, the screen executes a Cursor Right command.

Insert Line Between Margins

D413/D463

```
<036> <106> <133>      (octal)
1E 46 5B                (hex)
<036> F [               (ASCII)
```

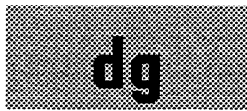
Inserts a line of text on the cursor row by copying rows down one row at a time. This command does not affect text outside of the margins. However, because it is a byte by byte copy, the execution is time consuming. The scrolling mode (smooth or jump scroll) is not enforced and the page/Roll Mode status is also ignored. Otherwise, operation of this command is the same as the Insert Line command.

Delete Line Between Margins

D413/D463

```
<036> <106> <134>      (octal)
1E 46 5C                (hex)
<036> F \               (ASCII)
```

Deletes a line of text on the cursor row by copying rows of the window up one row at a time. This command does not affect text outside of the margins. However, because it is a byte-by-byte copy, the execution is time consuming. The terminal does not enforce Scroll Mode (smooth or jump scroll), and the page/Roll Mode status is ignored. Otherwise, operation of this command is the same as the Delete Line command.



Erase Unprotected

D413/D463

<036> <106> <106>	(octal)
1E 46 46	(hex)
<036> F F	(ASCII)

If protect mode is enabled, this command erases all unprotected text from the cursor to the end of a window. If protect mode is disabled, all text from the cursor to the end of the window is blanked and protected fields are lost. Unlike other erase commands, this command operates between the left and right margins. The cursor position is left unchanged after the command.

Insert Character

D413/D463

<036> <112>	(octal)
1E 4A	(hex)
<036> J	(ASCII)

Inserts a blank character under the cursor by rippling characters to the right and putting a space under the cursor. The character at the right margin is lost. If protect mode is enabled and a protected character is encountered before the right margin is reached, that protected character is treated as the right margin.

Delete Character

D413/D463

<036> <113>	(octal)
1E 4B	(hex)
<036> K	(ASCII)

Deletes the character under the cursor by rippling characters to the left and bringing in a space at the right margin. If protect mode is enabled and a protected character is encountered before the right margin is reached, that protected character is treated as the right margin. The character under the cursor is lost.



Programmable Function Key Commands

Host Programmable Function Keys

D413/D463

```

<036> <106> <153> <mode> <key> <length> <string>      (octal)
1E 46 6B <mode> <key> <length> <string>                (hex)
<036> F k <mode> <key> <length> <string>                (ASCII)

```

where

<mode> is a one byte <n> parameter that defines erasure mode. Valid parameters are:

- 0 Clear to predefined defaults. This mode requires *no* arguments.
- 1 to 4 Reserved for D555 terminal. A value for mode between 1 and 4 will cause the sequence to fail.
- 5 Clear to predefined defaults and load new user key definitions.
- 6 Add to, or overwrite, previously defined key definitions.
- 7 Save up to 128 bytes out of the possible 255 bytes of the current key definitions in non-volatile RAM to be restored only on power up, or upon hard terminal reset (Cmd-Erase Page). This mode requires *no* additional arguments.
- 8 Read current user programmed key definitions (see also Read User Programmed Keys command, on next page). This mode requires *no* additional arguments.

<key> is a <nn> pair that sets the key to be defined. The range is 0 decimal to 59 decimal. Function keys F1 through F15 are broken into the ranges below:

Normal	0 – 14 decimal	00 – 0E hex
Shifted	15 – 29 decimal	0F – 1D hex
Ctrl	30 – 44 decimal	1E – 2C hex
Ctrl-Shift	45 – 59 decimal	2D – 3B hex

<length> is a <nn> pair that sets the length of the string being programmed. The range on this parameter is decimal values from 0 decimal to 255 decimal. A zero length will cause the key to be cleared to the predefined default. No string is required or expected with a zero length parameter.

<string> contains the actual ASCII key definition encoded as a string of <nn> pairs

For more information on <n> quantities and <nn> pairs, see the “Forming Command Arguments” section in this chapter.

Allows one or more of the 60 predefined function key codes to be reprogrammed with user defined sequences. Programmed function keys share 255 bytes of volatile memory space. This memory can be allocated entirely to one key if desired. Keys that do not have a user definition send the default function key sequences. When programming keys, this sequence will continue to



accept new key definitions until terminated with an out of range key number. You can also program function keys through the Configuration menus. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Programming Example

A user wants to clear any previous key definitions and make two of his own definitions: "hello" and "goodbye" to be on the Shift-F1 and Shift-F5 keys, respectively. The format for this command, as shown above is:

```
<036>Fk<mode><key><length><string>...<key><length><string><terminator>
```

The command sequence to perform our example is:

```
<036>Fk50?0568656<6<6?1307676?6?64627965?
```

where

<036>	Clear and load new keys
0?	Shift-F1
05	5 characters
68656<6<6?	ASCII hello in DG-hex
13	Shift-F5
07	7 characters
67?6?64627965	ASCII goodbye in DG-hex
??	Illegal key code terminates sequence.

Reading User Programmed Keys

The terminal will respond to this query from the host by sending information on current function key definitions in the following format.

```
<036>o;4<total_length><key><length><string>
```

where

<036>o;4 is the response header

<total_length> is the total number of bytes that will be sent in an <nnnn> format. For more information on <nnnn> quantities, see the "Forming Command Arguments" section of this chapter.

<key>, <length>, and <string> are the same as the parameters in the Host Programmable Keys command

This feature can be useful for applications that reprogram the function key definitions. It allows the application to read and save the current key definitions so that they may be restored on exit.



Programming Example

The sequence returned by <036>Fk8 after executing the above example (for the Host Programmable Keys command) would be:

```
<036>o;400200?0568656<6<6?1307676?6?64627965
```



Reporting Commands

Report Screen Size

all terminals

<code><036> <106> <164></code>	(octal)
<code>1E 46 74</code>	(hex)
<code><036> F t</code>	(ASCII)

Lets the host determine the maximum number of available rows and columns. The data returned is in the following format (for more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

<code><036> <157> <074> <nn> <nn> <nn> <nn> <status></code>	(octal)
<code>1E 6F 3C <nn> <nn> <nn> <nn> <status></code>	(hex)
<code><036> o < <nn> <nn> <nn> <nn> <status></code>	(ASCII)

where

the first <nn> pair indicates the number of available screen rows.

the second <nn> pair indicates the number of available screen columns (207 normally).

the third <nn> pair indicates the number of available rows in the current window.

the fourth <nn> pair indicates the number of available columns in the current window (right margin - left margin + 1).

<status> is an ASCII character composed of eight bits (representing the terminal state) in the format of 01HHMMSE,

HH is the ID code of the other emulator (00 hex Data General native-mode; 01 for VT320 mode; 10 for TEK4010; 11 for Printer or Mouse)

MM is the port mode of the emulators (00 for Both; 01 for Host; 10 for Auxiliary), and has no meaning if HH is set to 11.

S is 1 if the screen has been saved

E is 1 if screen save is enabled

For example, the host sends the code <036>Ft . The terminal responds with the code <036>o<18<?1050p. The interpretation of the returned data (in DG-hex) is the following:

18 means 24 rows in whole screen

<? means 207 columns in whole screen

10 means 16 rows in current window

50 means 80 columns in window (for example, left margin at 0 and right margin at 79)

p (01110000 in binary) means one host. Screen save is not enabled.



Read Horizontal Scroll Offset

D413/D463

```
<036> <106> <117>          (octal)
1E 46 4F                    (hex)
<036> F O                    (ASCII)
```

When this command is issued by the host, the terminal returns the distance the current window is horizontally scrolled over from the absolute zero (left most) column of the screen. The format of the data returned to the host is:

```
<036> <157> <072> <nn>      (octal)
1E 6F 3A <nn>              (hex)
<036> o : <nn>             (ASCII)
```

where

<nn> indicates the value of the horizontal offset. This value is in the range of decimal values from 0 to 126 with the lower four bits of two bytes specifying the eight-bit value of the scrolled offset. For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

Read Window Address

all terminals

```
<005>                        (octal)
05                             (hex)
Ctrl-E                         (ASCII)
```

The terminal sends the following three character sequence back to the computer.

```
<037> <col> <row>           (octal)
1F <col> <row>              (hex)
Ctrl- _ <col> <row>        (ASCII)
```

where

<col> is the window column the cursor is on (relative to the left margin mod 128)

<row> is the window row the cursor is on (relative to the top of the window)

Both <col> and <row> are two raw binary characters, and therefore are not encoded.



Read Screen Address

D413/D463

```
<036> <106> <142>          (octal)
1E 46 62                    (hex)
<036> F b                    (ASCII)
```

The terminal sends the following five character sequence back to the computer.

```
<036> <157> <070> <nn> <nn>  (octal)
1E 6F 38 <nn> <nn>          (hex)
<036> o 8 <nn> <nn>        (ASCII)
```

where

the first <nn> pair is the absolute screen column.

the second <nn> pair is the absolute screen row.

For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.

Read Window Contents

all terminals

```
<036> <106> <166> <r1> <c1> <r2> <c2>  (octal)
1E 46 76 <r1> <c1> <r2> <c2>          (hex)
<036> F v <r1> <c1> <r2> <c2>        (ASCII)
```

where

<r1> <c1> are two <nn> pairs that specify the bounded area’s upper left corner.

<r2> <c2> are two <nn> pairs that specify the bounded area’s lower right corner.

For more information on <nn> pairs, see the “Forming Command Arguments” section of this chapter.

Returns the contents of any bounded area of the screen to be transmitted back to the host. The rectangular area bounded by (r1,c1) and (r2,c2) is shipped to the host with <015> <012> (0D 0A hex) between each row and with trailing blanks on each row stripped. The data sent to the host is similar to what would be printed, were a printing command used. The characters on the screen are translated using the settings of G0 and G1. If a character on the screen does not belong to either assigned character set, then it will be translated as a space. If any of the given parameters are out of range, then nothing will be transmitted. Also, if the user presses the Local Print or Cmd-CR keys during this operation, it will be terminated. Like any of the printing commands, an <ACK> will be sent back to the host after the operation is complete *only* if this option has been selected on the Configuration Menus. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.



Read Model ID

all terminals

```

<036> <103>                (octal)
1E 43                       (hex)
<036> C                     (ASCII)

```

The terminal sends the following string back to the host computer:

```

<036> <157> <043> <m> <x> <y> (octal)
1E 6F 23 <m> <x> <y>         (hex)
<036> o # <m> <x> <y>       (ASCII)

```

where

<m> is a character that identifies the type of terminal. This character can be set in the menus or by the Set Model ID command. The table below lists the default settings.

| Terminal | Octal | Hexadecimal | ASCII |
|-----------|-------|-------------|-------|
| D217 | <065> | 35 | 5 |
| D413/D463 | <066> | 36 | 6 |

<x> is a character formed from bits defined as 01TC PRRR, where

- T 0 if the power on self test passed,
1 if errors are detected.
- C 0 if 7 bit communications mode
1 if 8 bit communications mode
- P 0 if printer not available
1 if printer is available
- R 3 bit firmware revision number 0-7.

<y> is a character formed from bits defined as 016K LLLL, where:

- G 0 if graphics are not available,
1 if D460 graphics available
- K 0 if keyboard missing
1 if keyboard installed
- L is from the following table:



| Keyboard Switches | Keyboard Nationality |
|-------------------|----------------------|
| 0000 | not used |
| 0001 | not used |
| 0010 | Norwegian |
| 0011 | Swiss/French |
| 0100 | Swiss/German |
| 1010 | Canadian/French |
| 0110 | Katakan |
| 0111 | Italian |
| 1000 | Canadian/French |
| 1001 | U.S. English |
| 1010 | United Kingdom |
| 1011 | French |
| 1100 | German |
| 1101 | Swedish/Finnish |
| 1110 | Spanish |
| 1111 | Danish/Norwegian |

Read New Model ID

all terminals

```

<036> <106> <167>          (octal)
1E 46 77                    (hex)
<036> F w                    (ASCII)

```

The terminal sends the following returned string back to the host computer:

```

<036><157><167><class><service-level><revision><name><reserved>  (octal)
1E 6F 77 <class><service-level><revision><name><reserved>    (hex)
<036> o w <class><service-level><revision><name><reserved>    (ASCII)

```

where

<class> is a single <n> digit specifying the general type of the terminal. The following classes are currently defined:

- 0 unintelligent terminal, handles no commands except <CR> and <LF>
- 1 D200 level data-entry terminal
- 3 D410 without Dynamically Reconfigurable Character Buffer (DRCB)
- 4 D410 level text-editing terminal
- 8 D460 level graphics terminal

<service-level> is an <nn> pair representing the specific terminal ID within the class. Any unknown code should be treated as the next lower known code. The D413/D463 returns 01.

<revision> is an <nn> pair encoding the firmware revision number.

<name> is an eight-character blank padded string with a human-readable terminal model-name. This is given for producing readable error logs for testing. The D217/D413/D463 returns "D217", "D413", or "D463".

CAUTION: Do not test or compare any part of this string!

<reserved> is a field of four spaces.

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

This command returns a more detailed description of the terminal type and capabilities. The original Model ID command (<036>C) reported a unique number for each terminal and there was no way to determine the capabilities of a new, unrecognized terminal.

It should not be necessary to reserve any new model IDs for the traditional Model ID command for future terminals.



Dual-Emulation Support Commands

Hot Key Switch

D413/D463

```
<036> <106> <155> <065>      (octal)
1E 46 6D 35                      (hex)
<036> F m 5                       (ASCII)
```

Changes the terminal from the active emulation to the currently inactive emulation. This command is only valid during a dual host session. It is mainly used with a diagnostic routine to check out the dual host functionality.

Switch Emulation Mode

all terminals

```
<036> <106> <176> <nn>        (octal)
1E 46 7E <nn>                  (hex)
<036> F ~ <nn>                 (ASCII)
```

where

<nn> has one of the following hex values:

| | |
|----|---|
| 00 | Data General native-mode (not generally used) |
| 08 | VT52 |
| 09 | VT100 |
| 0C | VT320 |
| 10 | Tektronix 4010 |

For more information on <nn> pairs, see the "Forming Command Arguments" section of this chapter.

This command returns the following string back to the host:

```
<036> <157> <176> <n>          (octal)
1E 6F 7E <n>                   (hex)
<036> o ~ <n>                  (ASCII)
```

where

<n> is 0 if the command failed; or 1 if it succeeded.

Changes the current emulation from Data General native-mode to any other emulation supported by the terminal. All volatile mode settings are lost.



Set Split Screen Mode

D413/D463

```
<036> <122> <101> <060> <nn> <n>      (octal)
1E 52 41 30 <nn> <n>                    (hex)
<036> R A 0 <nn> <n>                    (ASCII)
```

where

<nn> encodes the number of rows for the emulation to be displayed on the top. See the “Forming Command Arguments” section of this chapter for more information on <nn> pairs. If <nn> is 0, split screen is disabled. If <nn> is out of range, then the current split point is left unaltered.

<n> sets the emulation to be displayed in the top partition, with the following values:

```
0 host
1 auxiliary
```

Configures a terminal set for dual host to display portions of host and auxiliary emulations on the same screen, separated by a horizontal line. The status line is always displayed in Split Screen Mode. If the split point is changed by a valid parameter, both emulations’ first-row-to-display options are reset to zero.

NOTE: Cmd-Cursor Uparrow/Downarrow moves the split point up and down.

Set First Row To Display

D413/D463

```
<036> <122> <101> <061> <nn>      (octal)
1E 52 41 31 <nn>                    (hex)
<036> R A 1 <nn>                    (ASCII)
```

where

<nn> sets the first physical row of the emulation’s split-screen partition. See the “Forming Command Arguments” section in this chapter for more information on <nn> pairs. If this number is out of range, the maximum offset given the current setting of the status line is used (see Set 25th Line Mode command, earlier in this chapter).

Sets the first physical row to be displayed in the emulation’s split-screen partition.

NOTE: Cmd-Shift-Cursor Uparrow/Downarrow sets the first physical row of the active emulation.



Set Device Options

D413/D463

```
<036> <122> <102> <type> <ff> <cs> <graph>      (octal)
1E 52 42 <type> <ff> <cs> <graph>                (hex)
<036> R B <type> <ff> <cs> <graph>                (ASCII)
```

where

<type> is an <n> character with the following values:

- 0 printer
- 1 mouse (if mouse is selected, then the rest of the parameters are ignored).

<ff> is an <n> character with the following values:

- 0 Form Feed off
- 1 Form Feed before
- 2 Form Feed after
- 3 Form Feed before and after

<cs> is an <n> character with the following values:

- 0 ASCII and DGI
- 1 ASCII and VT Multinational
- 2 IBM PC
- 3 NRC Only
- 4 NRC and VT Line Drawing
- 5 Katakana

<graph> is an <n> character with the following values:

- 0 No graphics
- 1 DG graphics
- 2 IBM Pro-Printer Compatible graphics.

See the "Forming Command Arguments" section in this chapter for more information on <n> quantities.

Sets the device type connected to the other port if the other port is not used for an active emulation. This command is generally used for dual-host-on-one-port operations when some other device is attached to the other port.



Miscellaneous Commands

Set Cursor Type

D413/D463

<036> <106> <121> <n> (octal)
1E 46 51 <n> (hex)
<036> F Q <n> (ASCII)

where

<n> sets the cursor type:

- 0 disables the cursor from being displayed, while saving the current cursor attributes
- 1 selects a blinking underscore (like the D1/D2 cursor). If the cursor is currently off, turns the cursor on.
- 2 selects a reverse video block (like the D410/D460 cursor), which is the default cursor type. If the cursor is currently off, turns the cursor on.
- 3 selects a blinking reverse video block. If the cursor is off, turns the cursor on.
- 4 makes the cursor an underscore. If the cursor is off, turns it on.
- 5 displays cursor with the saved attributes.

Selects one of the five cursor types.

Set Model ID

all terminals

<036> <106> <173> <nn> <n> (octal)
1E 46 7B <nn> <n> (hex)
<036> F { <nn> <n> (ASCII)

where

<nn> sets the model ID (see the "Forming Command Arguments" section for more information on <nn> pairs). If this parameter is decimal value 0, then the default ID will be used.

<n> is 0 if graphics operations are not possible, or 1 if graphics are possible. This code will be ignored, but must be present, if <nn> is 0.

Allows the host computer to program the model ID response of the Read Model ID command so that older applications recognize newer terminals as older terminals.



Set Clock Time

all terminals

| | |
|------------------------------------|---------|
| <036> <106> <162> <n> <pos> <time> | (octal) |
| 1E 46 72 <n> <pos> <time> | (hex) |
| <036> F r <n> <pos> <time> | (ASCII) |

where

<n> defines the clock status (see the “Forming Command Arguments” section in this chapter for more information on <n> quantities). If 2, then the 12 hour clock is enabled; if 3, then the 24 hour clock is enabled.

<pos> = 0000. In D216/D412/D462 terminals, the clock was positionable within the screen boundaries. However, on all new terminals, the clock is always displayed within the status line. <pos> may be entered as 0000 and must be present.

<time> defines the clock time. The format of this parameter, which is in decimal values, is HH:MM where HH sets the hour and MM sets the minutes, such as “02:59” or “18:05”. You must enter the colon between HH and MM, and you must enter HH in a 24-hour (00–23) format.

NOTE: <pos> and <time> are specified only when <n> equals decimal values 2 or 3.

Lets the user set the clock to 12 or 24 hours, and set the time.

Bell

all terminals

| | |
|--------|---------|
| <007> | (octal) |
| 07 | (hex) |
| Ctrl-G | (ASCII) |

Rings the terminal bell once.

Reset

D413/D463

| | |
|-------------------|---------|
| <036> <106> <101> | (octal) |
| 1E 46 41 | (hex) |
| <036> F A | (ASCII) |

Restores the terminal to its initial power-on state, but does not run the powerup self-test. The same scrolling rate is in effect as was before. The status line is unaffected by this command, but any message displayed on the status line is lost.



Select 7/8 Bit Operation

all terminals

```
<036> <106> <125> <n>      (octal)
1E 46 55 <n>                (hex)
<036> F U <n>                (ASCII)
```

where

<n> sets the data bit syntax:

- 0 7-bit syntax is selected
- 1 8-bit syntax is selected.

Sets the 7/8 bit operations for the terminal. If 7-bit operations is selected, every inbound and outbound character has the most significant bit set to 0. For this reason, we recommend using the 8-bit mode only.

Set Keyboard Language (Keyboard Encoding Mode)

all terminals

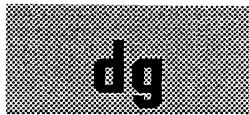
```
<036> <106> <146> <n>      (octal)
1E 46 66 <n>                (hex)
<036> F f <n>                (ASCII)
```

where

<n> sets either the default keyboard language or the U.S. ASCII language:

- 0 sets language to default (National)
- 1 sets language to U.S. ASCII and DG International
- 2 sets language to U.S. ASCII and ISO 8859.1 (8 bit characters)

Sets terminal keyboard translation mode. If <n> is set to 0, the keyboard transmits National Replacement Characters (NRC). NRC characters are 7-bit only and do not permit access to all ASCII characters. This command has no effect on characters received by the terminal. Only transmitted characters are encoded. Modes 1 and 2 select either DGI or ISO character codes for special (non-ASCII) characters.



UNIX Mode

all terminals

| | | | | |
|-------|-------|-------|-----|---------|
| <036> | <120> | <100> | <n> | (octal) |
| 1E | 50 | 40 | <n> | (hex) |
| <036> | P | @ | <n> | (ASCII) |

where

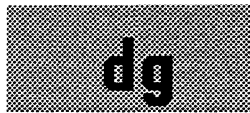
<n> has one of the following decimal values:

0 to exit UNIX mode

1 to enter.

See the “Forming Command Arguments” section in this chapter for more information on <n> quantities.

Remaps some troublesome commands and keyboard codes in traditional D200 terminals and allows easier creation of UNIX terminfo files to describe Data General terminals. For more information on UNIX mode, and the full listing of command and keyboard-code changes, see the “UNIX Support” section in this chapter.



Drawing Commands

Line

D463

```
<036> <114> <loc_list> <null>      (octal)
1E 4C <loc_list> <null>             (hex)
<036> L <loc_list> <null>           (ASCII)
```

or

```
<036> <107> <070> <loc_list> <null>  (octal)
1E 47 38 <loc_list> <null>           (hex)
<036> G 8 <loc_list> <null>          (ASCII)
```

where

<loc_list> = <location> *or* <location> <loc_list>.

In either case, <location> = <NNN> <NNN>

where

the first <NNN> quantity sets the X-ordinate of the location.

the second <NNN> quantity sets the Y-ordinate of the location.

For more information on location arguments and <NNN> quantities, see the "Forming Location Arguments" section in this chapter.

<null> is the ASCII NULL character (000 octal, 00 hex, or Ctrl-Shift-2).

Draws lines from point to point within the active window. If a single point is supplied in a line command, only that point is plotted. However, if multiple points are supplied, lines are drawn between the points in the order supplied. You can draw patterned lines by using the Set Pattern command before the Line command. If you enter an invalid parameter this command aborts.



Arc

D463

```

<036> <107> <060> <NNN> <NNN> <NNN> <NNN> <NNN>      (octal)
1E 47 30 <NNN> <NNN> <NNN> <NNN> <NNN>                (hex)
<036> G 0 <NNN> <NNN> <NNN> <NNN> <NNN>                (ASCII)

```

where

the first <NNN> sets the x-ordinate

the second <NNN> sets the y-ordinate

the third <NNN> defines the radius

the fourth <NNN> sets the starting angle

the fifth <NNN> sets the ending angle.

The <NNN> quantities are 15-bit location arguments (see the “Forming Location Arguments” section in this chapter).

Draws an arc within the current window with a given radius and start/end angles. The given angles are degrees. “Zero degrees” on the arc has the same x-ordinate as the center. The y-ordinate is equal to the y-ordinate of the center *plus* the radius. Thus “zero degrees” on the arc is directly above the center.

Bar

D463

```

<036> <107> <061> <NNN> <NNN> <NNN> <NNN> <n>          (octal)
1E 47 31 <NNN> <NNN> <NNN> <NNN> <n>                  (hex)
<036> G 1 <NNN> <NNN> <NNN> <NNN> <n>                  (ASCII)

```

where

the first <NNN> sets the x-ordinate of the lower left corner of the bar

the second <NNN> sets the y-ordinate of the lower left corner of the bar

the third <NNN> sets the size of the bar in the x-direction

the fourth <NNN> quantity defines the size of the bar in the y-direction

<n> sets the bar color:

0 makes the bar the same color as the background (Off Bar)

1 makes the bar the foreground color (On Bar)

The <NNN> quantities are 15-bit location arguments (see the “Forming Location Arguments” section in this chapter).

Draws a solid bar within the current window. Any characters that are completely filled in by this command are replaced with either a <space> or a reverse-video <space>, whichever is appropriate. An invalid parameter aborts this command



Polygon Fill

D463

```
<036> <107> <064> <loc> <null> (octal)
1E 47 3A <loc> <null> (hex)
<036> G : <loc> <null> (ASCII)
```

where

<loc> = <NNN> <NNN> ... <NNN> <NNN>. Each pair of values defines a vertex. This command supports from 3 through 255 vertices.

where

the first <NNN> defines the x-ordinate of a vertex.

the second <NNN> defines the y-ordinate of a vertex.

The <NNN> quantities are 15-bit location arguments (see the "Forming Location Arguments" section in this chapter).

<null> is the ASCII NULL character (00 hex, 000 octal, or Ctrl-Shift-2)

Draws a filled polygon within the current window. A filled polygon is defined by the vertices in the argument list. The polygon is automatically closed with a straight line between the first and last vertices. This polygon command does not support intersecting polygons. If filled polygons are defined so that they intersect other filled polygons, the polygons may or may not be filled correctly.

Set Pattern

D463

```
<036> <107> <160> <061> <offset> <pattern_definition> <null> (octal)
1E 47 70 31 <offset> <pattern_definition> <null> (hex)
<036> G p 1 <offset> <pattern_definition> <null> (ASCII)
```

where

<offset> specifies where in the pattern to begin drawing the dots, in the range of decimal values from 0 to 31. The value is formed from the 5 least significant bits (LSBs) of a single ASCII character.

<pattern_definition> = <pattern_character> or <pattern character> <pattern_definition>. If no pattern definition is specified, the terminal defaults to white lines. Otherwise, <pattern_character>, which is an <n> value, is 0 for off, and is 1 for on. Any other value results in transparent.

<null> is the ASCII NULL character (00 hex, 000 octal, or Ctrl-Shift-2)

Defines the line style used in generating lines. The terminal default is a solid line.



Set Foreground Color

D463 ■

| | |
|---------------------|---------|
| <036> <101> <color> | (octal) |
| 1E 41 <color> | (hex) |
| <036> A <color> | (ASCII) |

Sets the line foreground color for the Polygon Fill command to one of the following specified <color> parameters:

<060> black
<other> green or amber (depends on screen phosphor).



Graphics Cursor Commands

Read Cursor Location

D463

```
<036> <107> <077> <174>          (octal)
1E 37 3F 7C                        (hex)
<036> G ? |                          (ASCII)
```

The terminal responds to this command by returning the graphics cursor location to the host computer in the following form:

```
<036> <157> <174> <040> <nnnnn> <040> <nnnnn> <015>  (octal)
1E 6F 7C 20 <nnnnn> 20 <nnnnn> 0D                    (hex)
<036> 0 | <space> <nnnnn> <space> <nnnnn> <CR>      (ASCII)
```

where

the first <nnnnn> indicates the x-ordinate of the cursor

the second <nnnnn> indicates the y-ordinate of the cursor

Each <nnnnn> is a 5-digit ASCII coded decimal value. This encoding is not altered by UNIX mode.

Cursor On

D463

```
<036> <107> <102>          (octal)
1E 47 42                    (hex)
<036> G B                    (ASCII)
```

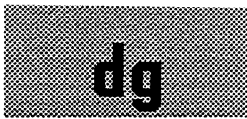
Turns on the graphics cursor (displays it on the screen) at the location specified in the graphics cursor location by the Cursor Location command.

Cursor Off

D463

```
<036> <107> <103>          (octal)
1E 47 43                    (hex)
<036> G C                    (ASCII)
```

Turns off the graphic cursor (erases it from the screen). The location and other attributes of the graphics cursor are not affected by this command.



Cursor Location

D463

```
<036> <107> <076> <174> <NNN> <NNN>      (octal)
1E 47 3E 7C <NNN> <NNN>                    (hex)
<036> G > | <NNN> <NNN>                    (ASCII)
```

where

the first <NNN> sets the x-ordinate of the graphics cursor location

the second <NNN> sets is the y-ordinate of the graphics cursor location

Each <NNN> value is a 15-bit location argument. See the "Forming Location Arguments" section for more information on location arguments and <NNN> values.

If the graphics cursor is turned on, this command moves the graphics cursor to the specified location. Ordinate arguments outside of the screen limits will be truncated to the nearest limits.

Cursor Track

D463

```
<036> <107> <110> <n>                      (octal)
1E 47 48 <n>                                (hex)
<036> G H <n>                               (ASCII)
```

where

<n> sets the input device, with the following values:

| | |
|--------|------------------------------|
| 0 | no tracking |
| 1 | reserved |
| 2 | track the keypad |
| 3 | reserved |
| 4 | track the mouse |
| 5 | reserved |
| 6 | track the mouse and keyboard |
| 7 – 14 | reserved |
| 15 | track all devices |

The graphics cursor location will be updated by the specified input device.



Cursor Attributes

D463

```
<036> <107> <100> (octal)
1E 47 40 (hex)
<036> G @ (ASCII)
```

The terminal responds to this command by returning the current value of the graphics cursor attributes in the following form:

```
<036> <157> <054> <v1> <v2> <v3> <v4> <015> (octal)
1E 6F 2C <v1> <v2> <v3> <v4> 0D (hex)
<036> o , <v1> <v2> <v3> <v4> <CR> (ASCII)
```

Where

<v1>, <v2>, <v3>, and <v4> are <n> quantities with the following values:

- <v1> = 0 cursor is off
- 1 cursor is on
- <V2> = 0 cursor is not blinking
- <V3> = 1 cursor is long
- <V4> = 0 no cursor tracking
- 1 reserved
- 2 cursor tracks the keypad
- 3 reserved
- 4 cursor tracks the mouse or bit pad
- 5 reserved
- 6 cursor tracks the mouse and keyboard
- 7-14 reserved
- 15 cursor tracks all devices

Cursor Reset

D463

```
<036> <107> <101> (octal)
1E 47 41 (hex)
<036> G A (ASCII)
```

Resets the graphics cursor attributes to off and no tracking.



Printer Commands

Print Window

all terminals

| | |
|--------|---------|
| <021> | (octal) |
| 11 | (hex) |
| Ctrl-Q | (ASCII) |

or

use of the Local Print key and the appropriate menu selection

Prints all characters (that reside in the U.S. ASCII or DGI character sets) between the margins of the current window, beginning with the row containing the cursor. Prints blank spaces rather than characters from other character sets such as user defined, line-drawing, Greek. No character attributes (underscore, reverse video, blink or dim) are printed.

If this command is initiated by the host, the terminal responds with a Ctrl-F (006 octal) when the printing operation is completed *only* if this option is enabled in the Configuration Menu. To abort this command press the Local Print or Cmd-CR keys.

Print Form

all terminals

| | |
|--------|---------|
| <001> | (octal) |
| 01 | (hex) |
| Ctrl-A | (ASCII) |

or

use of the Local Print key and the appropriate menu selection

When character protection is disabled, this command prints all full-intensity characters between the margins in the current window, beginning with the row containing the cursor. When character protection (D413/D463) is enabled, prints all unprotected text in the current window, beginning with the row containing the cursor. Prints dimmed and protected text as spaces. Transmits CR-LF sequence on initiation of print activity.

NOTE: In form operations if character protection is disabled the dim flag is used. If character protection is enabled the protection bit is used.

If this command is initiated by the host computer, the terminal responds with a Ctrl-F (006 octal) when the printing operation is completed *only* if this option is enabled in the Configuration Menu. To abort this command press the Local Print or Cmd-CR keys.



Print Screen

all terminals

```
<036> <106> <077> <072>      (octal)
1E 46 3F 3A                      (hex)
<036> F ? :                      (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Prints all characters (that reside in the U.S. ASCII or DGI character sets) between the margins in the current screen. Prints blank spaces rather than characters from other character sets such as user defined, line-drawing, Greek. No character attributes (underscore, reverse video, blink or dim) are printed. To abort this command press the Local Print or Cmd-CR keys.

If the Printer Acknowledge option is enabled in the Configuration Menu, the terminal sends a Ctrl-F (006 octal) to the host when the printing operation initiated by the host completes.

Form Bit Dump

D413/D463

```
<036> <106> <077> <066>      (octal)
1E 46 3F 36                      (hex)
<036> F ? 6                      (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Dumps a bit-image to an 8-bit graphics printer. When character protection is disabled, this command prints all full-intensity characters between the margins in the current window, beginning with the row containing the cursor. When character protection is enabled, prints all unprotected text in the current window, beginning with the row containing the cursor. Prints dimmed and protected text as spaces. Transmits CR-LF sequence on initiation of print activity. Reverse video spaces and underscores are printed as they appear on the screen.

If this command is initiated by the host computer, the terminal responds with a Ctrl-F (<006>) when the printing operation is completed *only* if this option is enabled in the Configuration Menu. To abort this command press the Local Print or Cmd-CR keys.



Window Bit Dump

D413/D463

```
<036> <106> <077> <065>      (octal)
1E 46 3F 35                      (hex)
<036> F ? 5                      (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Dumps a bit-image to a graphics slave printer. Prints the current window exactly as it appears to the user including reverse video and underscores. Disregards the blink and dim attributes.

If this command is initiated by the host computer, the terminal responds with a Ctrl-F (006 octal) when the printing operation is completed *only* if this option is enabled in the Configuration Menu. To abort this command press the Local Print or Cmd-CR keys.

Print Pass Through On

all terminals

```
<036> <106> <077> <063>      (octal)
1E 46 3F 33                      (hex)
<036> F ? 3                      (ASCII)
```

or

```
<036> <106> <140>            (octal)
1E 46 60                        (hex)
<036> F `                        (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Sends all subsequent characters from the host to the printer without affecting the display screen; the character flow from the host is interpreted by the printer and not by the display unit (with the only exception being the Print Pass Through Off command). To abort this command press the Local Print or Cmd-CR keys.



Print Pass Through Off

all terminals

| | |
|-------------------------|---------|
| <036> <106> <077> <062> | (octal) |
| 1E 46 3F 32 | (hex) |
| <036> F ? 2 | (ASCII) |

or

| | |
|-------------------|---------|
| <036> <106> <141> | (octal) |
| 1E 46 61 | (hex) |
| <036> F a | (ASCII) |

or

use of the Local Print key

Turns off Print Pass Through so that character flow from the host computer to the terminal is once again interpreted by the display unit. The display unit returns a Ctrl-F (006 octal) to the host to signal that the printing operation is completed *only* if this option is enabled in the Configuration Menu. This sequence is not copied to the printer.



Printer Pass Back To Host

all terminals

```
<036> <106> <170> <n>      (octal)
1E 46 78 <n>                (hex)
<036> F x <n>                (ASCII)
```

where

<n> determines the print mode:

- 0 turns this feature off
- 1 turns this feature on

The terminal responds to this command with the following string:

```
<036> <122> <170> <n>      (octal)
1E 52 78 <n>                (hex)
<036> R x <n>                (ASCII)
```

where

<n> reports the current information on this mode:

- 1 pass-back-to-host mode is now set
- 0 it is reset or cannot be set.

See the "Forming Command Arguments" section in this chapter for more information on <n> values.

Lets the host receive data directly from an attached device on the other port. This lets application software directly control and monitor a digitizer, scanner, bar code reader, or similar device. When this mode is set, all data coming from the printer port is sent to the host computer and the keyboard is disabled.

Simulprint On

D413/D463

```
<036> <106> <077> <061>    (octal)
1E 46 3F 31                (hex)
<036> F ? 1                (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Sends all characters received from the host to the attached printer while simultaneously displaying the characters on the screen. Once this command is issued, all subsequent print commands are ignored (with the exception of the Simulprint Off command). To abort this command press the Local Print or Cmd-CR keys.



Simulprint Off

D413/D463

```
<036> <106> <077> <060>      (octal)
1E 46 3F 30                    (hex)
<036> F ? 0                    (ASCII)
```

or

use of the Local Print key and the appropriate menu selection

Turns off Simulprint so that character flow from the host computer (to the terminal) is no longer copied to the printer. If this command is initiated by the host computer, the terminal responds with a Ctrl-F (006 octal) when the printing operation is completed *only* if this option is enabled in the Configuration Menu. To abort this command press the Local Print or Cmd-CR keys.

VT-Style Autoprint On

all terminals

```
<036> <106> <077> <070>      (octal)
1E 46 3F 38                    (hex)
<036> F ? 8                    (ASCII)
```

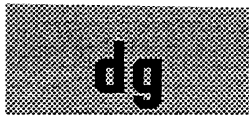
Turns on VT220-style autoprint mode. This command performs a line-buffered print operation. The line that the cursor is on will not be printed until a CR, LF, or VT is sent to the terminal *or* until autowrap has occurred.

VT-Style Autoprint Off

all terminals

```
<036> <106> <077> <067>      (octal)
1E 46 3F 37                    (hex)
<036> F ? 7                    (ASCII)
```

Turns off VT220-style autoprint mode.



Select Printer National Character Set

all terminals

```
<036> <106> <067> <nn>      (octal)
1E 46 37 <nn>                  (hex)
<036> F 7 <nn>                 (ASCII)
```

where

<nn> sets the printer language:

- 00 keyboard language
- 01 U.S. ASCII
- 02 United Kingdom
- 03 French
- 04 German
- 05 Swedish/Finnish
- 06 Spanish
- 07 Danish/Norwegian
- 08 Swiss
- 09 Katakana

See the "Forming Command Arguments" section in this chapter for more information on <n> values.



Debugging Commands

Data Trap Mode

all terminals

```

<036> <106> <073> <command>    (octal)
1E 46 3B <command>              (hex)
<036> F ; <command>             (ASCII)

```

where

<command> sets data trap mode:

- 124 octal (54 hex, T ASCII) data trap mode is turned off
- 110 octal (48 hex, H ASCII) data trap mode is turned on and the terminal displays hex values
- 117 or 177 octal (4F or 7F hex, O or DEL ASCII) data trap mode is turned on and the terminal displays octal values

Lets the user view data from the host as a hex or octal data-stream, similar to the AOS/VS X DISPLAY command. This command can be entered either on-line, off-line, or through the Configuration Menu. The data is displayed as 16 bytes per line on the left and the corresponding 8-bit characters on the right, as follows:

Hex Mode

```

40 30 31 32 33 34 35 36 37 38 39 21 23 24 2E 0A      @0123456789!#$,LF
41 42 43 44 45 44 47 48 49 4A 4B 4C 4D 4E 4F 50      ABCDEFGHIJKLMNOP
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70      abcdefghijklmnop
1E 46 3B 54                                           RS F;T

```

Octal Mode

```

100060 061062 063064 065066 067070 071041 043044 056012      @0123456789!#$,LF
101102 103104 105106 107110 111112 113114 115116 117120      ABCDEFGHIJKLMNOP
141142 143144 145146 147150 151152 153154 155156 157160      abcdefghijklmnop
036106 073124                                           RS F;T

```

In octal mode, the octal data is shown with every other space removed. This is done to make 16 bytes fit on one line, so that each group of three octal digits is one byte. This is different from AOS/VS's X DISPLAY command in that X DISPLAY shows octal data in six digit (16-bit) words, where the terminal shows individual (8-bit) bytes.

NOTE: The Index entries "DG native-mode commands by hex order" and "DG native-mode commands by octal order" list all commands by both hex and octal order. Refer to these entries to determine which commands are shown while using Data Trap mode.



Diagnostic Commands

Read Cursor Contents

all terminals

```
<036> <106> <155> <060>      (octal)
1E 46 6D 30                    (hex)
<036> F m 0                    (ASCII)
```

Lets the diagnostic routines more fully test the terminal. This command causes the terminal to respond to the host computer with the following 10 byte return sequence:

```
<036><157><073><060><attr><cs><fg><bg><charprt1><charprt2>      (octal)
1E 6F 3B 30 <attr><cs><fg><bg><charprt1><charprt2>          (hex)
<036> o ; 0 <attr><cs><fg><bg><charprt1><charprt2>          (ASCII)
```

where

<attr> and <cs> make one <nn> pair, where

<attr> are the character attributes at the cursor position in the form of BURD as used in the Change Attribute command (see the "Character Attribute Commands" section in this chapter). These are the upper four bits of the ATTR/CS byte of screen memory.

<cs> is the character set pointer, which is the lower four bits of the ATTR/CS byte of screen memory.

<fg> and <bg> are each <n> values, where

<fg> is the foreground color (always 1), which is the upper four bits of the COLOR byte of screen memory

<bg> is the background color (always 0), which is the lower four bits of the COLOR byte of screen memory

<charprt1> and <charprt2> make one <nn> pair, where

<charprt1> is the first half of the character pointer, which is the upper four bits of the CHAR/PTR byte of screen memory. This is an <n> value. For more information on <n> values, see the "Forming Command Arguments" section in this chapter.

<charprt2> is second half of the character pointer, which is the lower four bits of CHAR/PTR byte of screen memory. This is an <n> value.

CAUTION: The information returned for a given character is revision dependent.



Read Bit Contents

D413/D463

```
<036> <106> <155> <066>          (octal)
1E 46 6D 36                        (hex)
<036> F m 6                         (ASCII)
```

Causes the terminal to transmit graphics bit data in the current cursor cell back to the host in the following format:

```
<036> <157> <073> <061> <list>    (octal)
1E 6F 3B 31 <list>                (hex)
<036> o ; 1 <list>                (ASCII)
```

where

<list> is 12 <dd> pairs, with each pair containing 10 bits. The Define Character command explains <dd> pairs in more detail.

Each 5-bit sequence is encoded in the following format:

010BBBBB or @ through _ (40 hex through 5F hex)

Character Loopback

all terminals

```
<036> <106> <155> <064> <nn> <string> (octal)
1E 46 6D 34 <nn> <string>            (hex)
<036> F m 4 <nn> <string>           (ASCII)
```

where

<nn> sets the length of the string, from 0 through 255 characters. For more information on <nn> pairs, see the "Forming Command Arguments" section in this chapter.

<string> is the string of characters to echo back to the host.

Causes the terminal to echo the character string back to host without any translation of the 8-bit characters.



Fill Screen With Character

all terminals

```
<036> <106> <076> <char>      (octal)
1E 46 3E <char>                  (hex)
<036> F > <char>                 (ASCII)
```

where

<char> is the ASCII character that will fill the terminal screen.

Fills the display screen with the given character, using the current character attributes.

Fill Screen With Grid

all terminals

```
<036> <106> <071>                (octal)
1E 46 39                            (hex)
<036> F 9                            (ASCII)
```

Fills the display with a grid to facilitate screen alignments or measurements.

Display Character Generator Contents

all terminals

```
<036> <106> <070>                (octal)
1E 46 38                            (hex)
<036> F 8                            (ASCII)
```

Dumps the contents of the CGEN ROM to the display screen. The entire contents of the CGEN, a total of 512 characters, is displayed over and over, filling the entire screen between the current margins.

Perform UART Loopback Test

all terminals

```
<036> <106> <074>                (octal)
1E 46 3C                            (hex)
<036> F <                            (ASCII)
```

Performs loopback testing on host and printer UARTS. Error messages will be displayed on the screen. A loopback connector must be placed on the back of the terminal for the UART being tested for this command to work properly. This command causes a soft terminal reset. This command must be entered off-line.

End of Section

End of Chapter



Chapter 3

VT320/100/52 Emulations

This chapter provides the programming information for the VT320, VT100, or VT52 terminal emulations running on the D217/D413/D463 line of Data General terminals. This chapter has three major sections:

Summary of VT320/100 Operations

VT320/100 Emulation Control Sequences

VT52 Operations and Escape Sequences

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various VT320/100/52 emulator reference material is covered in related appendices.

A Note on VT Emulation Syntax Conventions

We established a set of general syntax rules that are used wherever references occur to VT320/100 control sequences. These rules, listed below, provide a uniform method of documenting VT320/100 control sequences.

- In control sequences, 7-bit and 8-bit control codes are represented by ANSI mnemonics.
- Remaining hex codes in control sequences are represented by printable ASCII characters.
- Nonprintable ASCII characters in control sequences are represented in capital letters and are enclosed within angle brackets (< >). For example, the space character is <SPACE>.
- Spaces between characters in control sequences are provided to improve clarity *only*. If a space character is part of a control sequence, it will be indicated as <SPACE>.
- Control sequence parameters, when possible, are illustrated with labels or abbreviations that indicate the nature or use of the parameter. These labels, always in lower case type, are enclosed within angle brackets to indicate that they are not to be taken literally. Examples of parameter forms are: <location>, <version>, or <set>.

VT320/100

Summary of Operations

This section provides a summary of the operations information specific to the VT320/100 emulation. This section *does not* cover specific format or usage information for VT320/100 control sequences, which are covered in “VT320/100 Control Sequences,” later in this chapter

In particular, this section explains the subjects covered in the following list:

VT320/100 Emulation Features

Control Codes

Control Sequences

Generated Keyboard Codes

Character Sets

ANSI Standard Mode Switches

ANSI Private Mode Switches

User-Defined Keys

Character Attributes

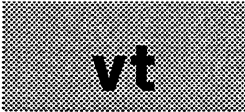
Line Attributes

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various VT320/100 emulator reference material is covered in related appendices, located at the back of this manual.

VT320/100 Emulation Features

The D217/D413/D463 terminals, in both 7-bit and 8-bit communications modes, operate in accordance with ANSI standards 3.4, 3.41, and 3.64. Due to hardware differences, there are some minor differences between our VT320/100 terminal emulations and the DEC VT320 and VT100 terminals. Major operating features of the VT320/100 emulations are listed below:

- Emulates both DEC VT320 and VT100 terminals
- Screen displays up to 25 lines of 80 characters displayed in a 10 x 12 dot matrix or up to 25 lines of 132 characters displayed in a 6 x 12 dot matrix
- Definable scrolling region
- Bidirectional (vertical) smooth scrolling
- Displays and prints double-height or double-width lines
- Text attribute for selective erasure (VT320 emulation only)
- Two downline loadable (DLL) character sets (VT320 emulation only on D463)



Control Codes

There are two groups of control codes, one 7-bit set and one 8-bit set. The set of 7-bit control codes is called C0 and the set of 8-bit codes is called C1. C0 runs from 0 hex through 1F hex. C1 runs from 80 hex through 9F hex.

The C0 set and the C1 set of codes are shown in Table 3-1.

Table 3-1 All C0 and C1 Control Codes

| Hex Codes | 0 | 1 | 2 | ... | 7 | 8 | 9 | A | ... | F |
|-----------|----------|-----|---------------|-----|---|----------|-----|---------------|-----|---|
| 0 | NUL | DLE | | | | | DCS | | | |
| 1 | SOH | DC1 | | | | | PU1 | | | |
| 2 | STX | DC2 | | | | | PU2 | | | |
| 3 | ETX | DC3 | | | | | STS | | | |
| 4 | EOT | DC4 | | | | IND | CCH | | | |
| 5 | ENQ | NAK | | | | NEL | MW | | | |
| 6 | ACK | SYN | | | | SSA | SPA | | | |
| 7 | BEL | ETB | | | | ESA | EPA | | | |
| 8 | BS | CAN | | | | HTS | | | | |
| 9 | HT | EM | | | | HTJ | | | | |
| A | LF | SUB | | | | VTS | DID | | | |
| B | VT | ESC | | | | PLD | CSI | | | |
| C | FF | FS | | | | PLU | ST | | | |
| D | CR | GS | | | | RI | OSC | | | |
| E | SO | RS | | | | SS2 | PM | | | |
| F | SI | US | | | | SS3 | APC | | | |
| | C0 Codes | | GL Characters | | | C1 Codes | | GR Characters | | |

Received Control Codes

Subsets of the entire range of C0 and C1 control codes are supported by the terminal and therefore cause a terminal-defined operation to occur upon receipt of the code from the host. Table 3-2 shows the supported control codes from the C0 set. Table 3-3 shows the supported control codes from the C1 set.

Table 3-2 Supported C0 Control Codes

| Mnemonic | Name | Hex Code | Ctrl Key Plus ... | Action |
|----------|------------------|----------|-------------------|---|
| ENQ | Enquiry | 05 | E | Transmits answerback message. |
| BEL | Bell | 07 | G | Generates bell tone if bell is enabled. |
| BS | Backspace | 08 | H | Moves cursor to the left one position, unless the cursor is at the left margin. |
| HT | Horizontal tab | 09 | I | Moves cursor to the next tab stop or right margin if no more tab stops in the line. |
| LF | Linefeed | 0A | J | Causes a linefeed or a new line, depending upon the state of new line mode (LNM). |
| VT | Vertical tab | 0B | K | Processed the same as a LF. |
| FF | Form feed | 0C | L | Processed the same as a LF. |
| CR | Carriage return | 0D | M | Moves cursor to left margin on current line. |
| SO | Shift out | 0E | N | Shifts G1 character set into GL. (locking shift) |
| SI | Shift in | 0F | O | Shifts G0 character set into GL. (locking shift) |
| DC1 | Device control 1 | 11 | Q | Also called XON. Allows transmission to resume if in limited transmit mode and XOFF was previously received; ignored otherwise. |
| DC2 | Device control 2 | 12 | R | Prints the next character literally. |
| DC3 | Device control 3 | 13 | S | Also called XOFF. Halts transmission of all codes except XON/XOFF if in limited transmit mode; ignored otherwise. |
| ESC | Escape | 1B | [| Processed as escape sequence introducer. Terminates any control or escape sequence in progress. |

NOTE: The Ctrl Key Plus (fourth column) combination is created by pressing the Ctrl key on the keyboard while simultaneously pressing the indicated character or symbol key.

Table 3-3 Supported C1 Control Codes

| Mnemonic | Name | Hex Code | 7-bit Code Equivalent | Action |
|----------|-----------------------------|----------|-----------------------|---|
| IND | Index | 84 | ESC D | Moves cursor down one line in current column. If the cursor is at the bottom margin, the screen scrolls up one line. |
| NEL | Next line | 85 | ESC E | Moves cursor to first position in the next line. If the cursor is at the bottom margin, the screen scrolls up one line. |
| HTS | Horizontal tab set | 88 | ESC H | Sets a tab stop at the current cursor column. |
| RI | Reverse index | 8D | ESC M | Moves cursor up one line in current column. If cursor is at the top margin, the screen scrolls down one line. |
| SS2 | Single shift G2 | 8E | ESC N | Shifts the G2 character set into GL for the next character only. |
| SS3 | Single shift G3 | 8F | ESC O | Shifts the G3 character set into GL for the next character only. |
| DCS | Device control string | 90 | ESC P | Marks the beginning of a device control string. Terminates any control or escape sequence in progress. |
| CSI | Control sequence introducer | 9B | ESC [| Marks the beginning of a control sequence. Terminates any control or escape sequence in progress. |
| ST | String terminator | 9C | ESC \ | Terminates the data stream in a device control string. |
| OSC | Operating system command | 9D | ESC] | Starts a string similar to a DCS string, but is ignored. Terminates any control or escape sequence in progress. |
| PM | Private message | 9E | ESC ^ | Starts a string similar to a DCS string, but is ignored. Terminates any control or escape sequence in progress. |
| APC | Application common | 9E | ESC _ | Starts a string similar to a DCS string, but is ignored. Terminates any control or escape sequence in progress. |

Transmitted Control Codes

Various keys on the keyboard, when pushed, send a control code to the host (if on-line). The code sent often depends upon operating mode (ANSI standard mode versus ANSI private mode) parameters. Transmitted codes are always 7-bit ANSI codes for the D217 and are either 7-bit or 8-bit for other terminals. Refer to "Transmission Control Sequences," later in this chapter, for more information on selecting 7-bit or 8-bit communications.

Table 3-4, Table 3-5, Table 3-6, Table 3-7, and Table 3-8 list the control codes that are transmitted by the terminal. These tables are for the 107-key standard Data General keyboard. The control codes transmitted by a 101-key keyboard may be similar; see "Generated Keyboard Code," later in this chapter, for more information on the 101-key keyboard. That section lists *all* codes, including many from different keystroke combinations, sent by the terminal to the host computer.

Table 3-4 Control Codes Generated from the Editing Keys (VT320/100)

| Keyboard | Function | Code | |
|--------------------|-----------------|--------|-------|
| | | VT320 | VT100 |
| Erase page | Find | CSI 1~ | none |
| Print | Insert here | CSI 2~ | none |
| Erase EOL | Remove | CSI 3~ | none |
| Shift - Erase page | Select | CSI 4~ | none |
| Shift - Print | Previous Screen | CSI 5~ | none |
| Shift - EOL | Next Screen | CSI 6~ | none |

Table 3-5 Control Codes Generated from the Cursor Control Keys (VT320/100)

| Key | Control Codes | |
|------------|---------------|------------------|
| | Normal mode | Application mode |
| Uparrow | CSI A | SS3 A |
| Downarrow | CSI B | SS3 B |
| Rightarrow | CSI C | SS3 C |
| Leftarrow | CSI D | SS3 D |

NOTE: "Normal mode" and "Application mode" refer to the reset and set state, respectively, of the Cursor Key Mode, which is an ANSI private operating mode parameter.



Table 3-6 Control Codes Generated from the Auxiliary Keypad (VT320/100)

| Key | Keypads | |
|------------|---------------------------|-------------------------------|
| | Numeric Mode ¹ | Application Mode ¹ |
| 0 | 0 | SS3 p |
| 1 | 1 | SS3 q |
| 2 | 2 | SS3 r |
| 3 | 3 | SS3 s |
| 4 | 4 | SS3 t |
| 5 | 5 | SS3 u |
| 6 | 6 | SS3 v |
| 7 | 7 | SS3 w |
| 8 | 8 | SS3 x |
| 9 | 9 | SS3 y |
| – (minus) | – | SS3 m |
| , (comma) | , | SS3 l |
| . (period) | . | SS3 n |
| Enter | CR | SS3 M |
| PF1 | SS3 P | SS3 P |
| PF2 | SS3 Q | SS3 Q |
| PF3 | SS3 R | SS3 R |
| PF4 | SS3 S | SS3 S |

¹ These modes refer to the set or reset state of the Keypad Numeric Mode and Keypad Application Mode, both of which are ANSI private operating mode parameters.

Table 3-7 Control Codes Generated from the Function Keys (VT320/100)

| 107-Key
Keyboard | VT320
Keyboard | Control Code | |
|---------------------|----------------------|--------------|-------|
| | | VT320 | VT100 |
| F20 | F1 (or Hold Screen) | none | none |
| F18 | F2 (or Print Screen) | none | none |
| Cmd/F17 | F3 (or Set-Up) | none | none |
| F19 | F4 (or Data/Talk) | none | none |
| F16 | F5 (or Break) | none | none |
| F1 | F6 | CSI 17~ | none |
| F2 | F7 | CSI 18~ | none |
| F3 | F8 | CSI 19~ | none |
| F4 | F9 | CSI 20~ | none |
| F5 | F10 | CSI 21~ | none |
| F6 | F11 | CSI 23~ | ESC |
| F7 | F12 | CSI 24~ | BS |
| F8 | F13 | CSI 25~ | LF |
| F9 | F14 | CSI 26~ | none |
| F10 | F15 | CSI 28~ | none |
| F11 | F16 | CSI 29~ | none |
| F12 | F17 | CSI 31~ | none |
| F13 | F18 | CSI 32~ | none |
| F14 | F19 | CSI 33~ | none |
| F15 | F20 | CSI 34~ | none |

NOTE: The D217 lets you select VT220 function keys (F1 through F15 on the DG 147 keyboard) in the Configuration menus. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.



Table 3-8 C0 Control Codes Generated from the Main Keypad (VT320/100)

| Ctrl Key Plus . . . | Mnemonic | Hex Code | Dedicated Function Keys |
|---------------------|----------|----------|-------------------------|
| 2 or @ | NUL | 00 | |
| A | SOH | 01 | |
| B | STX | 02 | |
| C | ETX | 03 | |
| D | EOT | 04 | |
| E | ENQ | 05 | |
| F | ACK | 06 | |
| G | BEL | 07 | |
| H | BS | 08 | F12 (BS) ¹ |
| I | HT | 09 | Tab |
| J | LF | 0A | F13 (LF) ¹ |
| K | VT | 0B | |
| L | FF | 0C | |
| M | CR | 0D | Return (CR) |
| N | SO | 0E | |
| O | SI | 0F | |
| P | DLE | 10 | |
| Q | DC1 | 11 | |
| R | DC2 | 12 | |
| S | DC3 | 13 | |
| T | DC4 | 14 | |
| U | NAK | 15 | |
| V | SYN | 16 | |
| W | ETB | 17 | |
| X | CAN | 18 | |
| Y | EM | 19 | |
| Z | SUB | 1A | |
| 3 or [| ESC | 1B | F11 (ESC) ¹ |
| 4 or \ | FS | 1C | |
| 5 or] | GS | 1D | |
| 6 or ~ | RS | 1E | |
| 7 or ? | US | 1F | |
| 8 | DEL | 7F | DEL |

¹ These keys are used in the VT100 to generate the control codes indicated. (D217 – VT220 function keys are disabled.)

Using 8-bit Code in 7-bit Environments

All C1 (8-bit) control codes may be expressed as C0 (7-bit) code extensions by using the ANSI defined conversion method outlined in ANSI document X3.41. The process of converting C1 code into C0 code is summarized below:

1. Make the first character of the sequence an escape character (1B hex, also known as the C0 code ESC).
2. Subtract 40 hex from the hexadecimal C1 control code, and make that the second character.

For example, consider the C1 control code CSI (9B hex). This is converted into a C0 code as shown below:

CSI = 1B [9B minus 40] = 1B 5B = ESC [

You should use 8-bit control codes in applications programs because there is one less byte to transmit and process. We also suggest that your application support both 7-bit and 8-bit modes because not all terminals can use 8-bit control codes. For example, the VT100 emulation running on a D217 uses 7-bit codes only. The VT100 running on a D217 lets you select 8-bit characters in the Configuration menu (keyboard mode – ISO.) For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Control Sequences

A control sequence is a string of ASCII characters that contains commands and/or arguments (parameters) that direct the terminal to perform specific functions. Most control sequences begin with the control sequence introducer, which is the 8-bit CSI control code. The introducer is then followed by one or more ASCII characters, which specify function types and parameters. CSI is an 8-bit code and can only be used in 8-bit environments. However, CSI can be expressed as a 7-bit code by replacing the CSI character with “ESC [”. Examples of a control sequence (and its 7-bit equivalent) are:

```
CSI 3 L (8-bit only)
or
ESC [ 3 L (7-bit or 8-bit environments)
```

where “L” specifies the Insert Line function, and “3” is a parameter indicating the number of lines to insert.

There are two other types of control sequences, both of which differ from the control sequence shown above because they begin with different ASCII characters (the escape character and a device control string) and have different purposes.

Escape Sequences

An escape sequence begins with the escape character (ESC), which is followed by one or more ASCII characters. ESC is a 7-bit control code, thus escape sequences can be used in both 7-bit and 8-bit environments. An example of an escape sequence is:

```
ESC H (7-bit or 8-bit environments)
```

Where “H” specifies a command that sets a tab stop at the current column.

Device Control Strings

Generally, a device control string starts with the 8-bit DCS control code, is followed by parameters and a command code and a data stream of ASCII characters, and is terminated by the 8-bit ST control code. A device control string can also begin with the C1 control codes OSC (“ESC]”), PM (“ESC ^”), or APC (“ESC _”). A device control string is functionally similar to a control sequence but contains variable length data; for this reason, it is delimited with a string terminator (ST), which tells the terminal when the last piece of data has been sent. As is the case with the CSI code, both the DCS and ST 8-bit control codes can be replaced with “ESC P” and “ESC \” for use in 7-bit environments. Examples of 7-bit and 8-bit device control strings are:

```
DCS 0;1 | 17/7F;18/08 ST (8-bit environments only)
or
ESC P 0;1 | 17/7F;18/08 ESC \ (7-bit or 8-bit environments)
```

where “0;1” are parameters, “|” is a character that specifies the User Define Keys operation, and “17/7F;18/08” is the data stream that contains information to map function keys F1 and F2 to the Delete and Backspace keys, respectively. The string terminator “ST” is used at the end to delimit data and to terminate the operation.

Generated Keyboard Codes

Each time you press a key, data is sent to the terminal. The terminal then interprets this data as either a local key (such as a Shift key) or as a code generating key (such as the character “A”). If a character code is generated, it is then either sent on to the host computer, if in on-line mode, or is taken as direct input by the terminal, if in off-line mode.

Transmitted codes are always 7-bit ANSI codes for the D217 and are either 7-bit or 8-bit for other terminals. Refer to the “Transmission Control Sequences” section of this chapter for more information on selecting 7-bit or 8-bit communications.

The terminal supports two keyboards, a 101-key keyboard, similar to an IBM-PC AT-style keyboard, and a 107-key Data General proprietary keyboard. Table 3-9 through Table 3-13 show the codes generated by each key or keystroke combination for the 101-key and 107-key keyboards, including the key mapping to a VT320 or VT100 keyboard. We did not include the code generated by the main keypad because the code is simply the code of the character on the keycap.

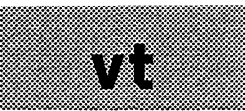


Table 3-9 Keyboard Generated Codes — Function Keys (VT320)

| 107-key Keyboard | 101-key Keyboard | VT320 Keyboard | Normal | Shift |
|------------------|--------------------|----------------|---------|-------|
| F1 | F1 | F6 | CSI 17~ | UDK1 |
| F2 | F2 | F7 | CSI 18~ | UDK1 |
| F3 | F3 | F8 | CSI 19~ | UDK1 |
| F4 | F4 | F9 | CSI 20~ | UDK1 |
| F5 | F5 | F10 | CSI 21~ | UDK1 |
| F6 | F6 | F11 | CSI 23~ | UDK1 |
| F7 | F7 | F12 | CSI 24~ | UDK1 |
| F8 | F8 | F13 | CSI 25~ | UDK1 |
| F9 | F9 | F14 | CSI 26~ | UDK1 |
| F10 | F10 | Help (F15) | CSI 28~ | UDK1 |
| F11 | F11 or Alt-F1 | Do (F16) | CSI 29~ | UDK1 |
| F12 | F12 or Alt-F2 | F17 | CSI 31~ | UDK1 |
| F13 | Alt-F3 | F18 | CSI 32~ | UDK1 |
| F14 | Alt-F4 | F19 | CSI 33~ | UDK1 |
| F15 | Alt-F5 | F20 | CSI 34~ | UDK1 |
| Cursor Type | Ctrl-Break | Break | Break | none |
| Cmd-N/C | R.Ctrl-Scroll Lock | Setup | none | none |
| Local Print | Print Screen | Print | none | none |
| Scroll Rate | n/a | n/a | n/a | n/a |
| Hold | Pause | Hold | none | none |
| C1 | Alt-F9 | PF1 | SS3 P | SS3 P |
| C2 | Alt-F10 | PF2 | SS3 Q | SS3 Q |
| C3 | Alt-F11 | PF3 | SS3 R | SS3 R |
| C4 | Alt-F12 | PF4 | SS3 S | SS3 S |

1 UDK means user defined key function.



Table 3-10 Keyboard Generated Codes — Function Keys (VT100)

| 107-key Keyboard | 101-key Keyboard | VT100 Keyboard | Code |
|------------------|--------------------|------------------------|--------|
| F6 | F6 | Escape ¹ | 1B hex |
| F7 | F7 | Backspace ¹ | 08 hex |
| F8 | F8 | Line Feed ¹ | 0A hex |
| C1 | Alt-F9 | PF1 | SS3 P |
| C2 | Alt-F10 | PF2 | SS3 Q |
| C3 | Alt-F11 | PF3 | SS3 R |
| C4 | Alt-F12 | PF4 | SS3 S |
| Cursor Type | Ctrl-Break | Break | n/a |
| Cmd-N/C | R.Ctrl-Scroll Lock | Setup | n/a |
| Local Print | Print Screen | Print | n/a |
| Scroll Rate | n/a | n/a | n/a |
| Hold | Pause | Hold | n/a |

¹ D217/VT100 with VT220 function keys disabled.

Table 3-11 Keyboard Generated Codes — Editing Keypad (VT320)

| 107-key Keyboard | 101-key Keyboard | VT320 Keyboard | Code |
|------------------|------------------|----------------|--------|
| Erase Page | Insert | Find | CSI 1~ |
| Print | Home | Insert Here | CSI 2~ |
| Erase EOL | Page Up | Remove | CSI 3~ |
| Shift-Erase | Delete | Select | CSI 4~ |
| Page | End | Previous | CSI 5~ |
| Shift-Print | Page Down | Screen | CSI 6~ |
| Shift-Erase EOL | | Next Screen | |

Table 3-12 Keyboard Generated Codes — Cursor Keypad (VT320 and VT100)

| 107-key Keyboard | 101-key Keyboard | VT320/100 Keyboard | Normal Cursor Keys | Application Cursor Keys |
|------------------|------------------|--------------------|--------------------|-------------------------|
| Uparrow | Uparrow | Uparrow | CSI A | SS3 A |
| Rightarrow | Rightarrow | Rightarrow | CSI C | SS3 C |
| Leftarrow | Leftarrow | Leftarrow | CSI D | SS3 D |
| Downarrow | Downarrow | Downarrow | CSI B | SS3 B |
| Home | n/a | n/a | CSI H | SS3 H |



Table 3-13 Keyboard Generated Codes — Numeric Keypad (VT320 and VT100)

| 107-key Keyboard ¹ | 101-key Keyboard | VT320/100 Keyboard and Numeric Mode with Num Lock On | Application Mode with Num Lock On | n/a ² Num Lock Off |
|-------------------------------|------------------|--|-----------------------------------|-------------------------------|
| n/a | Num Lock | On | On | Off |
| n/a | / | / | none | / |
| n/a | * | * | none | * |
| - (minus) | - | - (minus) | SS3 m | - |
| , (comma) | + | , (comma) | SS3 l | , |
| . (period) | ./Delete | . (period) | SS3 n | Select |
| 0 | 0/Insert | 0 | SS3 p | Find |
| 1 | 1/End | 1 | SS3 q | Previous Screen |
| 2 | 2/Downarrow | 2 | SS3 r | Cursor Down |
| 3 | 3/Pg Dn | 3 | SS3 s | Next Screen |
| 4 | 4/Leftarrow | 4 | SS3 t | Cursor Left |
| 5 | 5 | 5 | SS3 u | Home |
| 6 | 6/rightarrow | 6 | SS3 v | Cursor Right |
| 7 | 7/Home | 7 | SS3 w | Insert Here |
| 8 | 8/uparrow | 8 | SS3 x | Cursor Up |
| 9 | 9/Pg Up | 9 | SS3 y | Remove |
| Enter ³ | Enter | Enter ³ | SS3 M | Enter |

1 107-key keyboards numeric keypads have no Num Lock On/Off mode.

2 Neither Numeric Mode nor Application Mode have any bearing when Num Lock is Off.

3 Will send either a CR or a CR-LF depending upon the state of New Line Mode (LNM).

Character Sets

All terminals are equipped with a 512-character Character Generator (CGEN), which is located within terminal ROM. All hard character sets are composed of characters contained within the CGEN. In addition to the predefined hard character sets, the VT320 emulation on the D463 supports up to two soft character sets (each containing up to 96 characters), which reside in volatile RAM, that are composed of custom characters. Each of these two sets of custom characters are called downline loadable (DLL) character sets. The only terminal and emulation combination that supports soft character sets is the VT320 emulation on D463 terminals.

Hard Character Sets

Hard character sets are composed from characters located within the CGEN. Four character sets may be selected for use at one time. They are designated as G0, G1, G2, and G3. From these four, two character sets are *active*. These are designated as GL (Graphic Left) and GR (Graphic Right). The D217 has only two character sets available at one time. They are designated G0 and G1.

GL is generally used for 7-bit character codes (20 hex through 7F hex), character sets such as U.S. ASCII or an appropriate National Replacement Character (NRC) set. These NRC sets map national-language specific characters that are 8-bit codes onto the U.S. ASCII character set. This remapping process deletes least-used characters from the ASCII set and replaces those characters with frequently used 8-bit characters. Each NRC set is language-specific.

The GR set is used for 8-bit character codes, which are characters A0 hex through FF hex. GR generally contains special graphics sets such as Line Drawing or contains the Supplemental Graphics set, which is also known as the 8-bit character code portion of the VT Multinational Character Set.

Character sets are initialized on power up. In multinational mode, G0 and G1 are initialized with the standard ASCII character set, and G2 and G3 are initialized with VT Multinational. The G0 set is then invoked into GL, and the G2 set into GR (see ANSI document X3.41 for further information). On the D217, G0 is initialized with U.S. ASCII; G1 is initialized with VT Special Graphics; and G0 is invoked into GL.

Hard character sets are used in a two step process. They must first be designated and then invoked.



Designating Hard Character Sets

Character sets are designated as G0, G1, G2, or G3 by an escape sequence and a parameter, which are detailed in the “Hard Character Set Control Sequences” section later in this chapter.

Invoking Hard Character Sets

The GL character set is set to G0 and G1, respectively, with the Shift In and Shift Out commands for all the terminals. In addition to these two commands, because there are additional available sets on the D413/D463, GL is set to G2 and G3 by the Shift Lock Two and Shift Lock Three commands. Redefining the GR set is limited to the D413/D463; additional shift lock commands set the GR set to G1, G2, or G3. Full information on the use of these commands is located in within the “Hard Character Set Control Sequences” section later in this chapter.

Figure 3-1 illustrates which character sets can be invoked into G0 through G3.

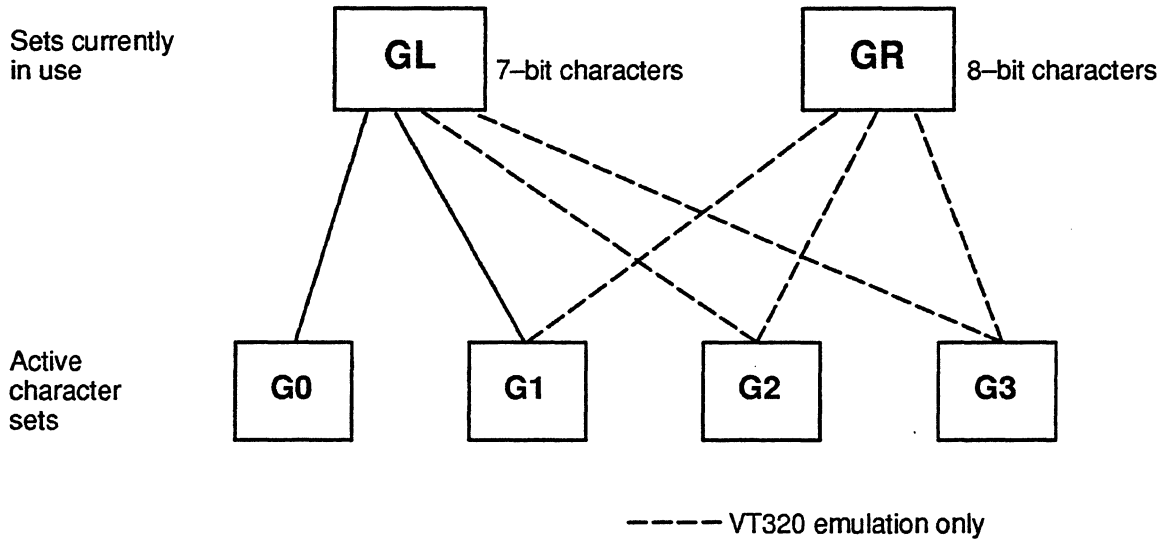


Figure 3-1 Invoking Character Sets into GL and GR (VT320/100)

Soft Character Sets

It is possible to define up to 96 characters that may be downloaded as a character set; the VT320 emulation on the D463 supports two soft character sets. A soft character set resides in volatile RAM, and is lost when the terminal is given the command to reset or the power is turned off. Soft character sets must be selected, defined (custom characters created), and then downloaded to the terminal. Also, to clear up terminal RAM space, previously downloaded soft character sets that are no longer in use can be cleared from the terminal.

Selecting Soft Character Sets

Before a soft character set can be used, it must first be assigned a name. This name, or designation, is used to refer to the soft character set within character set control sequences. A maximum of three characters are used to name a soft character set. The first two characters are optional, and can be any combination of characters in the range of 20 hex to 2F hex. The final character is required and must be in the range of 30 hex to 7E hex. An example of a legitimate name for a soft character set is:

```
<space>@
```

Where <space> is the space character (20 hex), and is in the range 20 hex to 2F hex, and “@” (40 hex) is the final character in the range 30 hex to 7E hex. This is the recommended default. Once a soft character set is named the characters within the set must then be defined.

Defining a Soft Character

Soft character cells must be created and defined as data before they can be downloaded as soft character sets to the terminal. The following steps outline the procedure for defining soft character fonts.

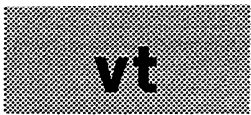
1. Define the character on a 10 x 12 matrix. In the following design example, the greek alpha symbol is being defined.

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 |
|-----|----|----|----|----|----|----|----|----|----|----|
| R0 | . | . | . | . | . | . | . | . | . | . |
| R1 | . | . | . | . | . | . | . | . | . | . |
| R2 | . | . | . | . | . | . | . | . | # | . |
| R3 | . | . | # | # | . | . | . | # | . | . |
| R4 | . | # | . | . | # | . | # | . | . | . |
| R5 | # | . | . | . | . | # | . | . | . | . |
| R6 | # | . | . | . | . | # | . | . | . | . |
| R7 | . | # | . | . | # | . | # | . | . | . |
| R8 | . | . | # | # | . | . | . | # | . | . |
| R9 | . | . | . | . | . | . | . | . | # | . |
| R10 | . | . | . | . | . | . | . | . | . | . |
| R11 | . | . | . | . | . | . | . | . | . | . |

2. Divide the character cell between the sixth and seventh row to create two sets of six (6 bit) patterns.

| | C0 | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 |
|-----|----|----|----|----|----|----|----|----|----|----|
| R0 | . | . | . | . | . | . | . | . | . | . |
| R1 | . | . | . | . | . | . | . | . | . | . |
| R2 | . | . | . | . | . | . | . | . | # | . |
| R3 | . | . | # | # | . | . | . | # | . | . |
| R4 | . | # | . | . | # | . | # | . | . | . |
| R5 | # | . | . | . | . | # | . | . | . | . |
| R6 | # | . | . | . | . | # | . | . | . | . |
| R7 | . | # | . | . | # | . | # | . | . | . |
| R8 | . | . | # | # | . | . | . | # | . | . |
| R9 | . | . | . | . | . | . | . | . | # | . |
| R10 | . | . | . | . | . | . | . | . | . | . |
| R11 | . | . | . | . | . | . | . | . | . | . |

3. Generate a string of binary numbers for the upper section of the character cell by reading each column from bottom to top, assigning a 1 if a pixel is on, and a 0 if the pixel is off. Start with column C0 and end with column C9. Please note, order must be preserved throughout the remaining steps. By applying this method to the design example, you generate the binary numbers 100000, 010000, 001000, 001000, 010000, 100000, 010000, 001000, 000100, and 000000.
4. Generate a string of binary numbers for the lower half of the character cell by following the process described in step 3. Remember that you begin at the bottom of the column. This yields the binary numbers 000001, 000010, 000100, 000100, 000010, 000001, 000010, 000100, 001000, and 000000.
5. These binary numbers must now be extended to form 8-bit binary numbers. This is done by prefixing all of the 6-bit numbers with 01.
6. Convert the 8-bit binary numbers to their hexadecimal equivalent values. This is done by dividing the 8-bit numbers into two groups of 4-bits each, and finding the hex equivalent for each of those 4-bit groups.
7. Subtract 1 (01 hex) from each hex number generated in Step 6.
8. Convert each hex number into its equivalent 7-bit ASCII character.



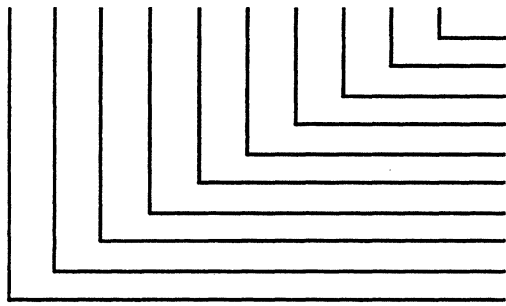
9. Arrange the hex data strings into a sixel bit pattern such that data for the top portion of the character cell is first, followed by the "p" character, and completed with data for the bottom portion of the cell. The pixel generated from the alpha character in our example is:

```
?CGO_OGGO_/?GCA@ACCA@
```

The example below illustrates Steps 6, 7, and 8.

```
.....
.....
.....#
..##...#
.#...#.#
#.....#
```

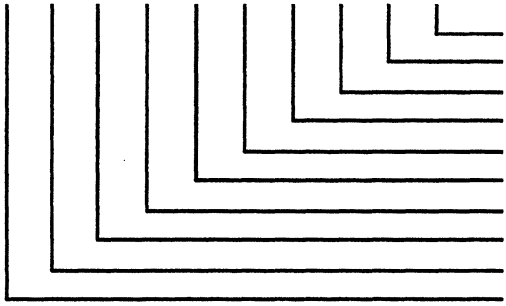
Where bits 7 and 6 are fixed as 01.



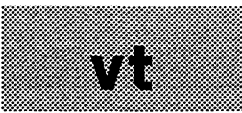
| <u>7654</u> | <u>3210</u> |
|-------------|------------------------------|
| 0100 | 0000 = 40 H - 1 H = 3F H = ? |
| 0100 | 0100 = 44 H - 1 H = 43 H = C |
| 0100 | 1000 = 48 H - 1 H = 47 H = G |
| 0101 | 0000 = 50 H - 1 H = 4F H = O |
| 0110 | 0000 = 60 H - 1 H = 5F H = _ |
| 0101 | 0000 = 50 H - 1 H = 4F H = O |
| 0100 | 1000 = 48 H - 1 H = 47 H = G |
| 0100 | 1000 = 48 H - 1 H = 47 H = G |
| 0101 | 0000 = 50 H - 1 H = 4F H = O |
| 0110 | 0000 = 60 H - 1 H = 5F H = _ |

```
#.....#
.#...#.#
..##...#
.....#
.....
.....
```

Where bits 7 and 6 are fixed as 01.



| <u>7654</u> | <u>3210</u> |
|-------------|------------------------------|
| 0100 | 0000 = 40 H - 1 H = 3F H = ? |
| 0100 | 1000 = 48 H - 1 H = 47 H = G |
| 0100 | 0100 = 44 H - 1 H = 43 H = C |
| 0100 | 0010 = 42 H - 1 H = 41 H = A |
| 0100 | 0001 = 41 H - 1 H = 40 H = @ |
| 0100 | 0010 = 42 H - 1 H = 41 H = A |
| 0100 | 0100 = 44 H - 1 H = 43 H = C |
| 0100 | 0100 = 44 H - 1 H = 43 H = C |
| 0100 | 0010 = 42 H - 1 H = 41 H = A |
| 0100 | 0010 = 42 H - 1 H = 41 H = A |
| 0100 | 0001 = 41 H - 1 H = 40 H = @ |



Downloading Soft Characters

Soft character sets are downloaded to the terminal by a device control string. Refer to the “Soft Character Set Control Sequences” section later in this chapter, for details on this process.

Clearing Downloaded Soft Character Sets

The control sequence for downloading a character set can also be used to clear a set that has been previously loaded. Refer to the “Soft Character Set Control Sequences” section for details on this process.

ANSI Standard Mode Switches

ANSI standard mode parameters are loosely defined by the appropriate ANSI document. The Set Mode and Reset Mode control sequences cause a set of operating parameters on the terminal to be set or reset. The “ANSI Standard Mode Control Sequences” section, later in this chapter, has full details of the parameters mentioned in the indented lists below. Table 3-14 shows the default powerup setting for ANSI standard mode parameters.

Table 3-14 Default ANSI Standard Mode Parameters

| ANSI Parameter | Default Setting |
|--------------------|-----------------|
| Keyboard action | Reset |
| Insert/Replace | Reset |
| Send/Receive | Set |
| Line feed/New line | Reset |

Any of the following ANSI standard mode operating parameters are set with the Set Mode command:

Keyboard Action — Disables the keyboard, turns on the “wait” LED, and prevents further data entry from the keyboard.

Insert/Replace — Causes new characters to be inserted at the active position and moves all following characters right one position.

Send/Receive — Transmits characters typed at the keyboard directly through the serial output port; the terminal displays only characters received from the serial port.

Line Feed/New Line — Lets the Line Feed command trigger a line feed (LF) operation followed by a carriage return (CR). Also causes the New Line key to send the sequence CR-LF.

Any of the following ANSI standard modes are selected with the Reset Mode control sequence:

Keyboard Action — Enables the keyboard as an input device.

Insert/Replace — Causes a new character received by the terminal to “overwrite” or replace an existing character at the active position.

Send/Receive — Displays characters typed at the keyboard on the screen and also transmits the characters through the serial port.

Line Feed/New Line — Limits a Line Feed command to vertical movement only, with respect to the current active position. Also causes the New Line key to send only a CR.

ANSI Private Mode Switches

The Private Set Mode and Private Reset Mode control sequences cause a set of operating parameters on the terminal to be set or reset. The “ANSI Private Mode Control Sequences” section, later in this chapter, has full details of the parameters mentioned in the indented lists below. Table 3-15 shows the powerup default states for the ANSI private mode parameters.

Table 3-15 Default ANSI Private Mode Parameters

| Private Parameter | Default Setting |
|---------------------------------|----------------------|
| Cursor key application | Reset |
| Column | Reset |
| Scrolling | Reset |
| Screen | Reset |
| Cursor origin | Reset |
| Auto wrap | Reset |
| Auto repeat | Reset |
| Print form feed | Reset |
| Print extent | Set |
| Text cursor | Set |
| Multi/national character set | Reset |
| Numeric keypad | Reset |
| Typewriter/data processing keys | Permanently
Reset |
| PC terminal | Reset |
| Backarrow key | Reset |

Any of the following ANSI private mode parameters are selected with the Private Set Mode control sequence:

Cursor Key Application — Causes the four cursor-control keys to transmit special predefined codes instead of the standard ANSI cursor control sequences. These special codes that are transmitted can then be used for application-specific tasks.

Column — Defines a maximum of 132 columns that can appear on the screen. When this function is used, the screen is erased and the cursor is moved to the home position.

Scrolling — Enables smooth scrolling at a maximum rate of 6 lines per second.

Screen — Displays characters in black against an illuminated background (reverse video).

Cursor Origin — Defines the origin or cursor-home to be the upper left character position of the current margin setting. Line numbers are relative to the current margin settings and the top line of the margin is referenced as line one. Cursor control sequences can not go beyond the current margin settings.

Auto Wrap — Characters received when the cursor is already at the right margin cause the cursor to move to the beginning of the next line. Cursor control sequences never wrap.

Auto Repeat — All normal typematic keys repeat when held down for more than one-half second. Repeating stops when the key is released.

Print Form Feed — Selects Form Feed to follow screen print commands.

Print Extent — Allows the entire screen to be printed during a print screen operation.

Text Cursor Enable — Displays the cursor.

Multi/National Character Set — Selects National mode, which allows the use of the 7-bit National Replacement Character Sets (NRC).

Keypad Application — Causes the numeric keypad to transmit alternate sequences.

Any of the following ANSI private mode operating parameters are selected with the Private Reset Mode control sequence:

Cursor Key Application — Causes the cursor-control keys to return standard ANSI cursor-control sequences.

Column — Allows a maximum of 80 columns to appear on the screen. When this function is used, the screen is erased and the cursor is moved to the home position.

Scrolling — Allows a scroll operation to jump in increments of one line.

Screen — Illuminates graphic characters against a black background.

Cursor Origin — Defines the origin or cursor-home to be at the upper left corner-position of the screen. When in the reset condition, the cursor addressing commands CUP and HVP can position the cursor anywhere on the screen regardless of the margins that have been set.

Auto Wrap — Any characters received while the cursor is at the right margin replace the existing character and the cursor does not move.

Auto Repeat — No keys will repeat without first pressing the Rept key.

Print Form Feed — No Form Feed is performed at the end of a print operation.

Print Extent — Allows printing of the current scrolling region only (defined by Set Margins).

Text Cursor Enable — Cursor is not displayed.

Multi/National Character Set — Selects Multinational mode for the keyboard, which allows the use of the Supplemental Character Set for 8-bit characters (GR), and the ASCII character set for 7-bit characters (GL).

Keypad Application — Causes the numeric keypad to operate in numeric mode.

User-Defined Keys

Function keys F6 to F20 may be programmed with user-definable key sequences. A total of 255 bytes of volatile RAM is available for key definitions and one key sequence may use up all or most of the available memory if desired. The control sequence that downloads user-defined keys is covered in more detail in the “User-Defined Key Control Sequences” section later in this chapter. You can also program keys using the Configuration Menu. For information on using the Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Character Attributes

Character attributes affect either the appearance of text on the screen or affect how text on the screen can be erased. Visual character attributes, set with the Select Graphic Rendition control sequence, include the following:

- Bold (or increased intensity) text
- Underscored text
- Blinking text
- Reverse video text

The only non-visual character attribute is text protection, which is set with the Select Character Attributes control sequence. Protected text is not erased by either of the Selective Erase control sequences.

Line Attributes

Examples of double-height and double-width characters are shown in Figure 3-2.

Single-height lines

B
B

Double-height lines

B

Single-width line

BBBB

Double-width line

B B B B

Figure 3-2 Examples of Double-Height and Double-Width Lines

A double-height line is created by designating one character row as the top half and the next lower row as the bottom of the line. Although there is no software requirement, to be readable both rows should contain the same character. Lines that have been expanded by either control sequence lose all characters to the right of the center of the screen. Both double-height and double-width line attributes affect only the current cursor row. There is no command to make characters double-high but not double-wide.

End of Section

VT320/100 Emulation Control Sequences

This section describes the format and usage of VT320/100 control sequences. Refer to the “Summary of VT320/100 Operations” section for an explanation of conventions and practices of the VT320/100 emulation.

The control sequences within this section are organized into the functional areas listed below:

Hard Character Set Control Sequences

Soft Character Set Control Sequences

Attribute Control Sequence

Cursor Positioning Control Sequences

Tabulation Control Sequences

Screen Editing Control Sequences

ANSI Standard Mode Control Sequences

ANSI Private Mode Control Sequences

Transmission Control Sequences

User-Defined Key Control Sequences

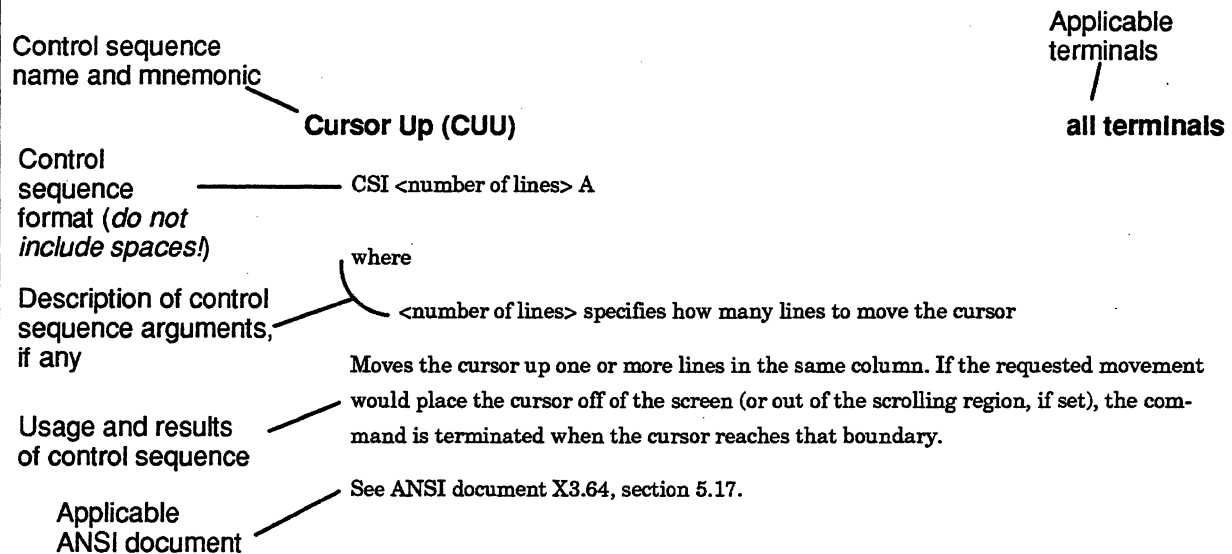
Miscellaneous Control Sequences

Reporting Control Sequences

Printing Control Sequences

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various VT320/100 emulator reference material is covered in appendices.

Format of Control Sequence Listings in this Section



A Note on Syntax Conventions

We established a set of general syntax rules that are used wherever references occur to VT320/100 control sequences. These rules, listed below, provide a uniform method of documenting VT320/100 control sequences.

- In control sequences, 7-bit and 8-bit control codes are represented by ANSI mnemonics.
- Remaining hex codes in control sequences are represented by printable ASCII characters.
- Nonprintable ASCII characters in control sequences are represented in capital letters and are enclosed within angle brackets (<>). For example, the space character is <SPACE>.
- Spaces between characters in control sequences are provided to improve clarity *only*. If a space character is part of a control sequence, it will be indicated as <SPACE>.
- Control sequence parameters, when possible, are illustrated with labels or abbreviations that indicate the nature or use of the parameter. These labels, always in lower case type, are enclosed within angle brackets to indicate that they are not to be taken literally. Examples of parameter forms are: <location>, <version>, or <set>.



Hard Character Set Control Sequences

Before character sets can be used, they must be designated into one of the four available character sets: G0, G1, G2, and G3. From these four available sets, two sets are active, that is have been invoked into the Graphic Left (GL) and Graphic Right (GR) sets. GL is for 7-bit characters and GR is for 8-bit characters.

Sequences for Designating Character Sets

Character sets are designated as G0, G1, G2, or G3 by an escape sequence and a parameter, as shown below:

| | | |
|-------------|-------------|--|
| ESC (<set> | (1B 28 hex) | designates a character set to be the G0 set |
| ESC) <set> | (1B 29 hex) | designates a 94-character set to be the G1 set |
| ESC - <set> | (1B 2D hex) | designates a 96-character set to be the G1 set |
| ESC * <set> | (1B 2A hex) | designates a 94-character set to be the G2 set (VT320) |
| ESC . <set> | (1B 2E hex) | designates a 96-character set to be the G2 set (VT320) |
| ESC + <set> | (1B 2B hex) | designates a 94-character set to be the G3 set (VT320) |
| ESC / <set> | (1B 2F hex) | designates a 96-character set to be the G3 set (VT320) |

where

<set> is a code that specifies a character set. The possible choices for hard character sets, and the code to designate them, are shown below.

| | | | |
|-------------|--|-------|--|
| B | U. S. ASCII (North American) | %5 | Supplemental graphics (from VT Multinational) |
| A | NRC United Kingdom | | |
| C or 5 | NRC Finnish | 0 | Special graphics |
| R or 9 | NRC French | 1 | DG International |
| Q | NRC French Canadian | 2 | DG Word Processing |
| K | NRC German | 3 | DG Line Drawing |
| E or 6 or ' | NRC Danish/Norwegian | 4 | DG Special Graphics |
| Z | NRC Spanish | l | Kata Kana G1 |
| H or 7 | NRC Swedish | <csd> | Downline Loadable soft character set. See the section "Soft Character Sets" in this chapter for more information. (VT320 emulation on D463 only) |
| = | NRC Swiss | | |
| J | NRC Kata Kana G0 | | |
| < | VT User-Preferred Supplemental Graphics (ISO or VT Supplemental selected by command) | | |

NOTE: The NRC selections will only give you your current keyboard language. For example, it is not possible to ask for a British NRC set if your keyboard language is North American. This is an emulation of a VT220 feature.

The G1, G2, and G3 set may contain either 94 or 96 characters. A new feature for all terminals is available whenever you select the Supplemental Graphics set (from VT/Multinational). You have the option of either leaving the Supplemental Graphics set exactly as it appears in the VT Multinational character set, or you can select the Supplemental Graphics set to be ISO Latin-1 (8859/1.2). The control sequence to change the Supplemental Graphics set to ISO Latin is described below.

Assign User-Preferred Supplemental Set (AUPSS)

D413/D463

```
DCS <Ps> ! u <Pss> ST
90 <Ps> 21 75 <Pss> 9C          (8-bit hex)
1B 50 <Ps> 21 75 <Pss> 1B 5C    (7-bit hex)
```

where <Ps> and <Pss> have the following values:

| <Ps> | <Pss> | |
|------|-------|---|
| 0 | %5 | supplemental graphics set is DEC Multinational |
| 1 | A | supplemental graphics set is ISO Latin-1 (8859/1.2) |

Sets the character set that will be used when Supplemental Graphics (from the VT/Multinational set) is designated.

Sequences for Invoking Character Sets

Shift In (SI)

all terminals

```
SI
0F          (7-bit hex)
```

Sets GL to point to G0 (default). See ANSI document X3.41, section 5.2.

Shift Out (SO)

all terminals

```
SO
0E          (7-bit hex)
```

Sets GL to point to G1. See ANSI document X3.41, section 5.2.

Shift Lock Two (SL2)**D413/D463**

ESC n
1B 6E (7-bit hex)

Sets GL to point to G2.

Shift Lock Three (SL3)**D413/D463**

ESC o
1B 6F (7-bit hex)

Sets GL to point to G3.

Shift Lock G1 GR**D413/D463**

ESC ~
1B 7E (7-bit hex)

Sets GR to point to G1.

Shift Lock G2 GR**D413/D463**

ESC }
1B 7D (7-bit hex)

Sets GR to point to G2.

Shift Lock G3 GR**D413/D463**

ESC |
1B 7C (7-bit hex)

Sets GR to point to G3.

Single Shift Two (SS2)**D413/D463**

SS2 or ESC N

8E (8-bit hex)

1B 4E (7-bit hex)

Displays the next character received from the G2 set, regardless of the state of GL and GR. See ANSI document X3.64, section 5.85.

Single Shift Three (SS3)**D413/D463**

SS3 or ESC O

8F (8-bit hex)

1B 4F (7-bit hex)

Displays the next character received from the G3 set, regardless of the state of GL and GR. See ANSI document X3.64, section 5.86.

Soft Character Set Control Sequences

Sequences for Downloading Soft Characters (VT320 on D463)

```
DCS <Pfn>;<Pcn>;<Pe>;<Pcmw>;<Pw>;<Pt>;<Pcmh>;<Pcss>
{ Dscs <Sxbp1>;<Sxbp2>;...;<Sxbpn> ST
```

```
90 <Pfn> 3B <Pcn> 3B <Pe> 3B <Pcmw> 3B <Pw> 3B <Pt> 3B <Pcmh> 3B <Pcss>
7B Dscs <Sxbp1> 3B <Sxbp2> 3B ... 3B <Sxbpn> 9C (8-bit hex)
```

```
1B 50 <Pfn> 3B <Pcn> 3B <Pe> 3B <Pcmw> 3B <Pw> 3B <Pt> 3B <Pcmh> 3B <Pcss>
7B Dscs <Sxbp1> 3B <Sxbp2> 3B ... 3B <Sxbpn> 1B 5C (7-bit hex)
```

where

- <Pfn> specifies which font buffer (character set) to load (0 or 1). The DEC VT320 terminal has only one set available, and both 0 and 1 refer to the same character set. The D463 terminal has two character sets available to load.
- <Pcn> specifies the starting location in the font buffer (character offset — from 0 to 95). A value of 0 begins the new character set at character code 20 hex.
- <Pe> specifies the erasure mode, where:
 - 0 erases all characters in the font buffer before loading in new ones (default)
 - 1 erases only character locations being loaded with new definitions
 - 2 erases all font buffers (soft character sets)
- <Pcmw> is the character cell size, where:
 - 0 15 pixels wide in 80 column rendition, 9 pixels wide in 132 column rendition
 - 1 illegal, and defaults to 0
 - 2 5x10 pixel cell
 - 3 6x10 pixel cell
 - 4 7x10 pixel cell
 - 5 5 pixels wide
 - .
 - .
 - .
 - 15 15 pixels wide

NOTE: Any cell width above 10 defaults to 10.

- <Pw> is the screen width, where:
 - 0 or 1 is 80 column (0 is default)
 - 2 132 column

- <Pt>** is the type of characters to be defined, where:
 0 or 1 is text (pixels are centered in cell)
 2 graphics (left aligned within cell)
- <Pcmh>** is the cell height, where:
 0 12 pixels high
 1 1 pixel high
 2 2 pixels high
 .
 .
 .
 12 12 pixels high
- <Pcss>** is the size of the character set
 0 94 characters (VT220 compatible)
 1 96 characters (allows redefinition of characters 20 hex and 7F hex)
- <Dscs>** is a character set name (same as in the VT320)
- <Sxbpn>** are DEC sixel bit-patterns (same as in the VT320)

DCS marks the beginning of the control sequence and ST terminates the control string. The new features in this control sequence are the expanded definition of <Pcmw> (<Pcms> in the VT320), <Pw>, and <Pt>, as well as the new parameters <Pcmh> and <Pcss>. The VT320 does not allow a character set defined in 80 column mode to be displayed in 132 column mode and vice versa because the font widths are different. Also, the VT220 would not automatically center characters in the cell. This feature allows future programs and terminals to be relatively independent of the character cell width for most non-graphics applications.

Refer to the "Character Sets" section earlier in this chapter for information on defining soft characters.

Sequences for Clearing Downloaded Soft Character Sets

```
DCS <Pfn>;<Pe> { <Dscs> ST
90 <Pfn> 3B <Pe> 7B <Dscs> 9C          (8-bit hex)
1B 50 <Pfn> 3B <Pe> 7B <Dscs> 1B 5C    (7-bit hex)
```

where

- <Pfn>** specifies which font buffer (character set) to clear (0 or 1). The DEC VT320 terminal has only one set available, and both 0 and 1 refer to the same character set.
- <Pe>** specifies the erasure mode, where:
 0 erases all characters in the font buffer
 2 erases all font buffers (soft character sets)

DCS marks the beginning of the control sequence and ST terminates the control string. The <Dscs> code is the designated name of the soft character set that you will clear by this control sequence.

Attribute Control Sequences

Line Attribute Sequences

Line Attribute control sequences apply to all terminals.

To designate the current cursor row as the top row of a double-height/double-width line, use the escape sequence:

ESC # 3
1B 23 33 (7-bit hex)

To designate the current cursor row as the bottom row of a double-height/double-width line, use the escape sequence:

ESC # 4
1B 23 34 (7-bit hex)

To designate the current cursor row as a single-height/double-width line, use the escape sequence:

ESC # 6
1B 23 36 (7-bit hex)

To return the current cursor row to a single-width/single-height line, use the escape sequence:

ESC # 5
1B 23 35 (7-bit hex)

Character Attribute Sequences

Select Graphic Rendition (SGR)

all terminals

```
CSI <parameter> ; ... ; <parameter> m
9B <parameter> 3B ... 3B <parameter> 6D      (8-bit hex)
1B 5B <parameter> 3B ... 3B <parameter> 6D  (7-bit hex)
```

where

<parameter> determines visual character attributes, as determined by the list below:

- 0 turns off all visual attributes
- 1 displays bold or at increased intensity
- 4 turns underscore on
- 5 turns blink on
- 7 turns reverse video on
- 22 displays at normal intensity (VT320 only)
- 24 turns underscore off (VT320 only)
- 25 turns blink off (VT320 only)
- 27 turns reverse video off (VT320 only)

Changes one or more visual attributes for successive characters in the data stream according to the parameter values. Visual attributes not specified in the parameter string are turned off. If no parameters are supplied with the control sequence, then *all* attributes for subsequent characters are turned off. See ANSI document X3.64, section 5.77.

Select Character Attributes (SCA)

D413/D463

```
CSI <parameter> " q
9B <parameter> 22 71      (8-bit hex)
1B 5B <parameter> 22 71  (7-bit hex)
```

where

<parameter> determines text erasure, according to the list below:

- 0 same as 2, below
- 1 subsequent characters are not erasable (protected)
- 2 subsequent characters are erasable (unprotected)

This control sequence selects all subsequent characters displayed to be erasable or not erasable with regard to Selective Erase In Line (SEL) and Selective Erase In Display (SED) control sequences.



Cursor Positioning Control Sequences

Parameters listed in this section are optional. If you do not specify a parameter or specify it as 0, then it will be treated as 1.

Cursor Up (CUU)

all terminals

```
CSI <lines> A
9B <lines> 41      (8-bit hex)
1B 5B <lines> 41  (7-bit hex)
```

where

<lines> is a whole number that specifies how many lines to move the cursor

Moves the cursor up one or more lines in the same column. If the requested movement would place the cursor off the screen (or out of the scrolling region, if set), the control sequence is terminated when the cursor reaches that boundary. See ANSI document X3.64, section 5.17.

Cursor Down (CUD)

all terminals

```
CSI <lines> B
9B <lines> 42      (8-bit hex)
1B 5B <lines> 42  (7-bit hex)
```

where

<lines> is a whole number that specifies how many lines to move the cursor

Moves the cursor down one or more lines in the same column. If the requested movement places the cursor off the screen (or out of the scrolling region, if set), the control sequence is terminated when the cursor reaches that boundary. See ANSI document X3.64, section 5.14.

Cursor Forward (CUF)**all terminals**

```

CSI <columns> C
9B <columns> 43          (8-bit hex)
1B 5B <columns> 43      (7-bit hex)

```

where

<columns> is a whole number that specifies how many columns to move the cursor

Moves the cursor to the right one or more columns in the same row. If the requested movement places the cursor off the screen, the control sequence is terminated when the cursor reaches the right margin. Auto-wrap mode is ignored. See ANSI document X3.64, section 5.15.

Cursor Backward (CUB)**all terminals**

```

CSI <columns> D
9B <columns> 44          (8-bit hex)
1B 5B <columns> 44      (7-bit hex)

```

where

<columns> is a whole number that specifies how many columns to move the cursor

Moves the cursor to the left one or more columns in the same row. If the requested movement places the cursor off the screen, the control sequence terminates when the cursor reaches the left margin. See ANSI document X3.64, section 5.13.

Cursor Position (CUP)**all terminals**

```

CSI <line> ; <column> H
9B <line> 3B <column> 48      (8-bit hex)
1B 5B <line> 3B <column> 48  (7-bit hex)

```

where

<line> is from 1 through 24, with the number 1 representing the top line of the screen

<column> is from 1 to 80 (or 1 through 132, depending upon whether normal or compressed characters) are displayed, with the number 1 representing the left most column on the screen.

Moves the cursor to the specified line and column. Parameter values greater than the ranges specified causes the cursor to peg at the nearest reasonable position. The parameters are relative to the upper left corner of the scrolling region if Cursor Origin mode is set. See ANSI document X3.64, section 5.16.



Horizontal and Vertical Position (HVP)

all terminals

```
CSI <vert> ; <horiz> f
9B <vert> 3B <horiz> 66      (8-bit hex)
1B 5B <vert> 3B <horiz> 66  (7-bit hex)
```

where

<vert> is a line number from 1 through 24, with the number 1 representing the top line of the screen

<horiz> is a column number from 1 to 80 (or 1 through 132, depending upon normal or compressed characters), with the number 1 representing the left most column on the screen.

Moves the cursor to the specified line (vertical position) and column (horizontal position). Parameter values greater than the ranges specified causes the cursor to peg at the nearest reasonable position. The parameters are relative to the upper left corner of the scrolling region if Cursor Origin mode is set. See ANSI document X3.64, section 5.47.

NOTE: This command is functionally the same as the Cursor Position command.

Index (IND)

all terminals

```
IND or ESC D
84      (8-bit hex)
1B 44   (7-bit hex)
```

Causes the cursor to move downward one line without changing the column position. If the cursor is at the bottom margin, the screen scrolls up. See ANSI document X3.64, section 5.50.

Tab

all terminals

09

Advances the cursor to the next tab stop to the right of the current cursor position. If no additional tab stops are defined to the right of the current cursor position, then the cursor advances to the last screen column. Use the Set Horizontal Tab (HTS) and Clear Tab Stops (TBC) commands to change the tab stops. You can also use the Answerback and Tabs Menu to change the tab stops as described in the *Installing and Operating Your D216E, D217, D413, and D463 Display Terminals* manual.

Reverse Index (RI)**all terminals**

RI or ESC M
8D (8-bit hex)
1B 4D (7-bit hex)

Moves the cursor up one line maintaining the same column position. If the cursor is at the top margin, the screen scrolls down. See ANSI document X3.64, section 5.71.

Next Line (NEL)**all terminals**

NEL or ESC E
85 (8-bit hex)
1B 45 (7-bit hex)

Moves the cursor to the first position on the next line down. If the cursor is already at the bottom margin, the screen scrolls up. See ANSI document X3.64, section 5.59.

Save Cursor (SC)**all terminals**

ESC 7
1B 37 (hex)

Saves the following information in terminal memory:

- Cursor position
- State of graphic rendition
- Character set shift state
- Wrap flag state
- State of origin mode
- State of selective erase attribute

Restore Cursor (RC)**all terminals**

ESC 8
1B 38 (hex)

Restores the information saved by the Save Cursor control sequence. If the Save Cursor control sequence was not used before this control sequence is issued, the terminal resets to its default condition, and the cursor moves to Home.

Tabulation Control Sequences

Set Horizontal Tab (HTS)

all terminals

HTS or ESC H
88 (8-bit hex)
1B 48 (7-bit hex)

Sets a horizontal tab at the current cursor column. See ANSI document X3.64, section 5.46.

Clear Tab Stops (TBC)

all terminals

CSI <parameter> g
9B <parameter> 67 (8-bit hex)
1B 5B <parameter> 67 (7-bit hex)

Where

<parameter> specifies which tab stops to clear, according to the following list:
0 clears a horizontal tab stop at the cursor position
3 clears all horizontal tab stops

Clears one or more tab stops. The parameter supplied with this control sequence determines the mode in which tabs are cleared. If no parameter is specified, a parameter value of 0 is used. See ANSI document X3.64, section 5.91.

Screen Editing Control Sequences

Delete Character (DCH)

D413/D463

```
CSI <# of chars> P
9B <# of chars> 50          (8-bit hex)
1B 5B <# of chars> 50      (7-bit hex)
```

where

<# of chars> is the number of characters to delete; the default value when no parameter is present is 1.

Deletes one or more characters starting at the cursor position. Characters to the right of the deleted area, if any, are moved left to the current cursor position to fill the void created by the deletion. Characters can be deleted from the current cursor position to the end of the current line. See ANSI document X3.64, section 5.21.

Insert Character (ICH)

D413/D463

```
CSI <# of chars> @
9B <# of chars> 40          (8-bit hex)
1B 5B <# of chars> 40      (7-bit hex)
```

where

<# of chars> is the number of character spaces to insert; the default value when no parameter is present is 1.

Inserts one or more erased characters (spaces) to the right, starting at the current cursor position. Characters at the cursor position and to the right of the cursor are scrolled right. Characters that are scrolled off the screen are lost. See ANSI document X3.64, section 5.48.



Insert Line (IL)

D413/D463

CSI <# of lines> L
9B <# of lines> 4C (8-bit hex)
1B 5B <# of lines> 4C (7-bit hex)

where

<# of lines> is the number of blank lines to insert; the default value when no parameter is present is 1.

Inserts one or more blank lines in the current scrolling region starting at the line the cursor is on. Lines at and below the cursor are shifted down. See ANSI document X3.64, section 5.49.

Delete Line (DL)

D413/D463

CSI <# of lines> M
9B <# of lines> 4D (8-bit hex)
1B 5B <# of lines> 4D (7-bit hex)

where

<# of lines> is the number of lines to delete; the default value when no parameter is present is 1.

Deletes one or more lines starting at the current line the cursor is on, and successive lines below. Lines are deleted only in the current scrolling region, and any lines left below the deleted rows are scrolled up; blank lines are inserted as required. See ANSI document X3.64, section 5.23.

Erase Character (ECH)

D413/D463

CSI <# of chars> X
9B <# of chars> 58 (8-bit hex)
1B 5B <# of chars> 58 (7-bit hex)

where

<# of chars> is the number of characters to erase; the default value when no parameter is present is 1.

Erases the specified number of characters on a line starting at, and including, the cursor position. See ANSI document X3.64, section 5.28.

Erase In Line (EL)**all terminals**

CSI <parameter> K
9B <parameter> 4B (8-bit hex)
1B 5B <parameter> 4B (7-bit hex)

where

- <parameter> specifies what portion of the line to erase, according to the following list:
- 0 erases from the cursor to the end of the current line. This is the default if no parameter is specified.
 - 1 erases from the first character on the line to the cursor position, inclusive.
 - 2 erases the entire line.

Erases displayed characters on the current line. The cursor position is not changed. See ANSI document X3.64, section 5.31.

Erase In Display (ED)**all terminals**

CSI <parameter> J
9B <parameter> 4A (8-bit hex)
1B 5B <parameter> 4A (7-bit hex)

where

- <parameter> specifies what portion of the screen to erase, according to the following list:
- 0 erases from the cursor to the end of the screen. This is the default if no parameter is specified.
 - 1 erases from the beginning of the screen to the cursor position, inclusive.
 - 2 erases the entire screen.

Erases displayed characters. The cursor position is not changed. See ANSI document X3.64, section 5.29.



Selective Erase In Line (DECSEL)

D413/D463

CSI ? <parameter> K
9B 3F <parameter> 4B (8-bit hex)
1B 5B 3F <parameter> 4B (7-bit hex)

where

- <parameter> specifies what portion of erasable characters to erase in a line:
- 0 erases all erasable characters from the cursor to the end of the current line. This is the default if no parameter is specified.
 - 1 erases all erasable characters from the beginning of the line to the current cursor position, inclusive.
 - 2 erases all erasable characters on the entire line.

Erases characters that have been designated as erasable with the Select Character Attributes control sequence.

Selective Erase In Display (DECSED)

D413/D463

CSI ? <parameter> J
9B 3F <parameter> 4A (8-bit hex)
1B 5B 3F <parameter> 4A (7-bit hex)

where

- <parameter> specifies what portion of erasable characters to erase on the screen:
- 0 erases all erasable characters from the current cursor position to the end of the screen. This is the default if no parameter is specified.
 - 1 erases all erasable characters beginning from the beginning of the screen to the cursor position inclusive.
 - 2 erases all erasable characters on the entire screen.

Erases characters that have been designated as erasable with the Select Character Attributes control sequence.



Scroll Down (SD)

all terminals

```
CSI <parameter> T
9B <parameter> 54      (8-bit hex)
1B 5B <parameter> 54  (7-bit hex)
```

where

<parameter> specifies the number of lines to scroll on the screen:

Scrolls the screen contents down.

Scroll Up (SU)

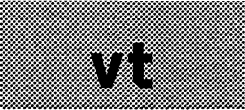
all terminals

```
CSI <parameter> S
9B <parameter> 53      (8-bit hex)
1B 5B <parameter> 53  (7-bit hex)
```

where

<parameter> specifies the number of lines to scroll on the screen:

Scrolls the screen contents up.



ANSI Standard Mode Control Sequences

Set Mode and Reset Mode are control sequences that cause one or more standard modes to be set or reset within the terminal. See "ANSI Standard Mode Parameters," for details on the mode parameters.

Set Mode (SM)

all terminals

```

CSI <parameter> h
9B <parameter> 68           (8-bit hex)
1B 5B <parameter> 68       (7-bit hex)

```

where

<parameter> determines which mode will be set:

| | | |
|----|-------------|------------------------------|
| 2 | (32 hex) | sets Keyboard Action mode |
| 4 | (34 hex) | sets Insert/Replace mode |
| 12 | (31 32 hex) | sets Send/Receive mode |
| 20 | (32 30 hex) | sets Line Feed/New Line mode |

Several mode parameters can be set in one operation by using a string in the following format:

CSI <parameter> ; <parameter> ; ... ; <parameter> h

See ANSI document X3.64, section 5.79.

Reset Mode (RM)

all terminals

```

CSI <parameter> l
9B <parameter> 6C           (8-bit hex)
1B 5B <parameter> 6C       (7-bit hex)

```

where

<parameter> determines which mode will be reset:

| | | |
|----|-------------|--------------------------------|
| 2 | (32 hex) | resets Keyboard Action mode |
| 4 | (34 hex) | resets Insert/Replace mode |
| 12 | (31 32 hex) | resets Send/Receive mode |
| 20 | (32 30 hex) | resets Line Feed/New Line mode |

Several mode parameters can be reset in one operation by using a string in the following format:

CSI <parameter> ; <parameter> ; ... ; <parameter> l



See ANSI document X3.64, section 5.73.

ANSI Standard Mode Parameters

Keyboard Action Mode (KAM)

all terminals

| | |
|-------------|--------------------|
| CSI 2 h | to set |
| 9B 32 68 | (8-bit hex) |
| 1B 5B 32 68 | (7-bit hex) |
| CSI 2 l | to reset (default) |
| 9B 32 6C | (8-bit hex) |
| 1B 5B 32 6C | (7-bit hex) |

The reset state of this parameter enables the keyboard as an input device. The set state disables the keyboard, turns on the wait LED, and prevents further data entry. See ANSI document X3.64, section 5.54.

Insert/Replace Mode (IRM)

D413/D463

| | |
|-------------|--------------------|
| CSI 4 h | to set |
| 9B 34 68 | (8-bit hex) |
| 1B 5B 34 68 | (7-bit hex) |
| CSI 4 l | to reset (default) |
| 9B 34 6C | (8-bit hex) |
| 1B 5B 34 6C | (7-bit hex) |

This parameter determines the insert or replace mode for new graphic characters placed on the video display. Reset Mode causes a new character to overwrite or replace an existing one at the active position. Set Mode causes new characters to be inserted at the active position and moves all following characters one position to the right. See ANSI document X3.64, section 5.52.

Send/Receive Mode (SRM)**all terminals**

| | |
|----------------|------------------|
| CSI 12 h | to set (default) |
| 9B 31 32 68 | (8-bit hex) |
| 1B 5B 31 32 68 | (7-bit hex) |
| CSI 12 l | to reset |
| 9B 31 32 6C | (8-bit hex) |
| 1B 5B 31 32 6C | (7-bit hex) |

The Send-Receive parameter controls the function of local echoing of characters to the video display. In Set Mode, characters typed at the keyboard are transmitted directly through the serial output port; only received characters from the serial port are made visible. In Reset Mode, characters typed at the keyboard are made visible on the screen and are also transmitted through the serial port. See ANSI document X3.64, section 5.83.

Line Feed/New Line Mode (LNM)**all terminals**

| | |
|----------------|--------------------|
| CSI 20 h | to set |
| 9B 32 30 68 | (8-bit hex) |
| 1B 5B 32 30 68 | (7-bit hex) |
| CSI 20 l | to reset (default) |
| 9B 32 30 6C | (8-bit hex) |
| 1B 5B 32 30 6C | (7-bit hex) |

This parameter controls the interpretation of the Line Feed control sequence and the keyboard. In the reset state, a Line Feed implies only vertical movement with respect to the current active position and the New Line key produces only a CR. The set state causes the Line Feed to trigger a Line Feed operation followed by a CR and the New Line key to send the sequence CR-LF. See ANSI document X3.64, section 5.55.

ANSI Private Mode Control Sequences

Private Set Mode and Private Reset Mode are ANSI private control sequences that set or reset one or more ANSI private operating modes within the terminal. See section “ANSI Private Operating Mode Parameters,” which follows, for details on the mode parameters.

Private Set Mode (EXSM)

all terminals

```
CSI ? <parameter> h
9B 3F <parameter> 68      (8-bit hex)
1B 5B 3F <parameter> 68  (7-bit hex)
```

where

| | |
|----------------|--|
| <parameter> | determines which private mode will be set: |
| 1 (31 hex) | sets Application/ANSI Cursor Keys mode |
| 2 (32 hex) | sets ANSI mode |
| 3 (33 hex) | sets Column mode |
| 4 (34 hex) | sets Scrolling mode |
| 5 (35 hex) | sets Screen mode |
| 6 (36 hex) | sets Cursor Origin mode |
| 7 (37 hex) | sets Auto Wrap mode |
| 8 (38 hex) | sets Auto Repeat mode |
| 18 (31 38 hex) | sets Print Form Feed mode |
| 19 (31 39 hex) | sets Print Extent mode |
| 25 (32 35 hex) | sets Text Cursor Enable mode |
| 42 (34 32 hex) | sets Multi/National Character Set mode |
| 66 (36 36 hex) | sets Numeric Keypad mode |
| 67 (36 37 hex) | sets Backarrow Key mode |
| 68 (36 38 hex) | sets Typewriter/Data Processing Keys mode |
| 99 39 39 hex) | sets PCTERM mode |

Several mode parameters can be set in one operation by using a string in the following format:

```
CSI ? <parameter> ; <parameter> ; ... ; <parameter> h
```

Private Reset Mode (EXRM)**all terminals**

```

CSI ? <parameter> l
9B 3F <parameter> 6C      (8-bit hex)
1B 5B 3F <parameter> 6C  (7-bit hex)

```

where

| | |
|-------------|---|
| <parameter> | determines which private mode will be reset: |
| 1 | (31 hex) resets Application/ANSI Cursor Keys mode |
| 2 | (32 hex) resets ANSI mode (VT52) |
| 3 | (33 hex) resets Column mode |
| 4 | (34 hex) resets Scrolling mode |
| 5 | (35 hex) resets Screen mode |
| 6 | (36 hex) resets Cursor Origin mode |
| 7 | (37 hex) resets Auto Wrap mode |
| 8 | (38 hex) resets Auto Repeat mode |
| 18 | (31 38 hex) resets Print Form Feed mode |
| 19 | (31 39 hex) resets Print Extent mode |
| 25 | (32 35 hex) resets Text Cursor Enable mode |
| 42 | (34 32 hex) resets Multi/National Character Set mode |
| 66 | (36 36 hex) resets Numeric Keypad mode |
| 67 | (36 37 hex) resets Backarrow Key mode |
| 68 | (36 38 hex) resets Typewriter/Data Processing Keys mode |
| 99 | 39 39 hex) resets PCTERM mode |

Several mode parameters can be reset in one operation by using a string in the following format:

```
CSI ? <parameter> ; <parameter> ; ... ; <parameter> l
```

ANSI Private Operating Mode Parameters

For additional information on ANSI private operating mode parameters, see ANSI document X3.64, section 3.5.6.

Application/ANSI Cursor Keys Mode (ACKM)

all terminals

| | |
|----------------|--------------------|
| CSI ? 1 h | to set |
| 9B 3F 31 68 | (8-bit hex) |
| 1B 5B 3F 31 68 | (7-bit hex) |
| CSI ? 1 l | to reset (default) |
| 9B 3F 31 6C | (8-bit hex) |
| 1B 5B 3F 31 6C | (7-bit hex) |

With Cursor Key Application mode set, the four cursor control keys no longer transmit standard ANSI control sequences for cursor movement. Instead, special predefined codes are sent that can be interpreted by the user for various applications. For more information on application key codes, refer to the “Generated Keyboard Characters” section of this chapter. Reset Mode returns the ANSI control sequences to the cursor keys.

Column Mode (COLM)

D413/D463

| | |
|----------------|--------------------|
| CSI ? 3 h | to set |
| 9B 3F 33 68 | (8-bit hex) |
| 1B 5B 3F 33 68 | (7-bit hex) |
| CSI ? 3 l | to reset (default) |
| 9B 3F 33 6C | (8-bit hex) |
| 1B 5B 3F 33 6C | (7-bit hex) |

The reset state allows a maximum of 80 columns to appear on the screen. The set state defines a maximum of 132 columns that can appear on the screen. When this function is used, the screen is erased and the cursor is moved to the Home position.

Scrolling Mode (SCRLM)**D413/D463**

| | |
|----------------|--------------------|
| CSI ? 4 h | to set |
| 9B 3F 34 68 | (8-bit hex) |
| 1B 5B 3F 34 68 | (7-bit hex) |
| CSI ? 4 l | to reset (default) |
| 9B 3F 34 6C | (8-bit hex) |
| 1B 5B 3F 34 6C | (7-bit hex) |

The reset state allows a scroll operation to jump in increments of one whole line. The set state enables smooth scrolling at a maximum rate of 6 lines per second.

Screen Mode (SCRNM)**all terminals**

| | |
|----------------|--------------------|
| CSI ? 5 h | to set |
| 9B 3F 35 68 | (8-bit hex) |
| 1B 5B 3F 35 68 | (7-bit hex) |
| CSI ? 5 l | to reset (default) |
| 9B 3F 35 6C | (8-bit hex) |
| 1B 5B 3F 35 6C | (7-bit hex) |

In the reset state, graphic characters are illuminated against a black background. When set, characters are displayed in black against an illuminated background (reverse video).

Cursor Origin Mode (COM)**all terminals**

| | |
|----------------|--------------------|
| CSI ? 6 h | to set |
| 9B 3F 36 68 | (8-bit hex) |
| 1B 5B 3F 36 68 | (7-bit hex) |
| CSI ? 6 l | to reset (default) |
| 9B 3F 36 6C | (8-bit hex) |
| 1B 5B 3F 36 6C | (7-bit hex) |

The reset state defines the origin, or Home, to be at the upper left corner position of the screen. When in the reset condition, the cursor-addressing control sequences Cursor Postition (CUP) and Horizontal and Vertical Position (HVP) can position the cursor anywhere on the screen regardless of the margins that have been set. The set state defines the origin, or Home, to be the upper left character position of the current margin setting. Line numbers in Set Mode are relative to the current margin settings; the top line of the margin is referenced as line 1. Cursor control sequences cannot go beyond the current margin settings.

Set to ANSI Mode**all terminals**

CSI ? 2 h to set
 9B 3F 32 68 (8-bit hex)
 1B 5B 3F 32 68 (7-bit hex)

CSI ? 2 l to reset
 9B 3F 32 6C (8-bit hex)
 1B 5B 3F 32 6C (7-bit hex)

Reverts terminal to VT52 mode and sets all other modes to their saved configuration values or defaults. The set command is ignored.

Auto Wrap Mode (AWM)**all terminals**

CSI ? 7 h to set
 9B 3F 37 68 (8-bit hex)
 1B 5B 3F 37 68 (7-bit hex)

CSI ? 7 l to reset (default)
 9B 3F 37 6C (8-bit hex)
 1B 5B 3F 37 6C (7-bit hex)

If Auto Wrap mode is set, any characters received when the cursor is already at the right margin cause the cursor to move to the beginning of the next line. In Reset Mode, any character received while the cursor is at the right margin replaces the existing character and the cursor does not move. Cursor control sequences never wrap.

Auto Repeat Mode (ARM)**all terminals**

| | |
|----------------|-------------|
| CSI ? 8 h | to set |
| 9B 3F 38 68 | (8-bit hex) |
| 1B 5B 3F 38 68 | (7-bit hex) |

| | |
|----------------|--------------------|
| CSI ? 8 l | to reset (default) |
| 9B 3F 38 6C | (8-bit hex) |
| 1B 5B 3F 38 6C | (7-bit hex) |

In the reset state, no keys repeat without first pressing the Rept key. If Auto Repeat mode is set, all normal typematic keys repeat when held down for more than one-half second and will stop when the key is released.

Print Form Feed Mode (PFF)**all terminals**

| | |
|-------------------|-------------|
| CSI ? 18 h | to set |
| 9B 3F 31 38 68 | (8-bit hex) |
| 1B 5B 3F 31 38 68 | (7-bit hex) |

| | |
|-------------------|--------------------|
| CSI ? 18 l | to reset (default) |
| 9B 3F 31 38 6C | (8-bit hex) |
| 1B 5B 3F 31 38 6C | (7-bit hex) |

This mode, when set, will terminate a print operation with a Form Feed (FF) character. In the reset state, no Form Feed is performed at the end of a print operation.

Print Extent Mode (PEXM)**all terminals**

| | |
|-------------------|------------------|
| CSI ? 19 h | to set (default) |
| 9B 3F 31 39 68 | (8-bit hex) |
| 1B 5B 3F 31 39 68 | (7-bit hex) |

| | |
|-------------------|-------------|
| CSI ? 19 l | to reset |
| 9B 3F 31 39 6C | (8-bit hex) |
| 1B 5B 3F 31 39 6C | (7-bit hex) |

Set Mode allows the entire screen to be printed during a print screen operation. Reset Mode allows printing of the current scrolling region only, as defined by Set Margins.

Text Cursor Enable Mode (TCEM)**D413/D463**

| | |
|-------------------|------------------|
| CSI ? 25 h | to set (default) |
| 9B 3F 32 35 68 | (8-bit hex) |
| 1B 5B 3F 32 35 68 | (7-bit hex) |
| CSI ? 25 l | to reset |
| 9B 3F 32 35 6C | (8-bit hex) |
| 1B 5B 3F 32 35 6C | (7-bit hex) |

If set, the cursor is visible. If reset, the cursor is not displayed.

Multi/National Character Set Mode (MNCSM)**D413/D463**

| | |
|-------------------|--------------------|
| CSI ? 42 h | to set |
| 9B 3F 34 32 68 | (8-bit hex) |
| 1B 5B 3F 34 32 68 | (7-bit hex) |
| CSI ? 42 l | to reset (default) |
| 9B 3F 34 32 6C | (8-bit hex) |
| 1B 5B 3F 34 32 6C | (7-bit hex) |

The set state selects the National mode which allows the use of the 7-bit National Replacement Character (NRC) Sets. The reset state selects the Multinational mode, which allows the use of the Supplemental Character Set for 8-bit characters (GR), and the U.S. ASCII character set for 7-bit characters (GL).

Numeric Keypad Mode (NKM)**D413/D463**

| | |
|-------------------|--------------------|
| CSI ? 66 h | to set |
| 9B 3F 36 36 68 | (8-bit hex) |
| 1B 5B 3F 36 36 68 | (7-bit hex) |
| CSI ? 66 l | to reset (default) |
| 9B 3F 36 36 6C | (8-bit hex) |
| 1B 5B 3F 36 36 6C | (7-bit hex) |

Operates the same as the older VT52 control sequences "ESC =" (1B 3D hex) and "ESC >" (1B 3E hex). This command extends the control sequence set to include the older VT52 control sequences. Refer to the "Generated Keyboard Codes" section of this chapter.



Backarrow Key Mode (BKM)

D413/D463

| | |
|-------------------|--------------------|
| CSI ? 67 h | to set |
| 9B 3F 36 37 68 | (8-bit hex) |
| 1B 5B 3F 36 37 68 | (7-bit hex) |
| CSI ? 67 1 | to reset (default) |
| 9B 3F 36 37 6C | (8-bit hex) |
| 1B 5B 3F 36 37 6C | (7-bit hex) |

Sets the code that the DEL key will send. If this mode is reset (the default condition), then the DEL key sends an ASCII DEL code (7F hex). If the mode is set, the DEL key sends an ASCII BS (Backspace) code (08 hex).

Typewriter/Data Processing Keys Mode (KBUM)

D413/D463

| | |
|-------------------|--------------------|
| CSI ? 68 h | to set |
| 9B 3F 36 38 68 | (8-bit hex) |
| 1B 5B 3F 36 38 68 | (7-bit hex) |
| CSI ? 68 1 | to reset (default) |
| 9B 3F 36 38 6C | (8-bit hex) |
| 1B 5B 3F 36 38 6C | (7-bit hex) |

This mode was available in DEC's original VT220 only through a menu selection. Although the command exists in the VT320 emulation, it does nothing (it is permanently reset) because of different keyboard implementation.

Set Limited Transmit

D413/D463

| | |
|-------------------|--------------------|
| CSI ? 7 3 h | to set |
| 9B 3F 37 33 68 | (8-bit hex) |
| 1B 5B 3F 37 33 68 | (7-bit hex) |
| CSI ? 7 3 1 | to reset (default) |
| 9B 3F 37 33 6C | (8-bit hex) |
| 1B 5B 3F 37 33 6C | (7-bit hex) |

Causes the terminal to respond to ^S (13 hex) sent by the host by stopping data transmission within 2 characters. The host can restart data transmission by sending ^Q (11 hex) to the terminal. When this mode is reset, the terminal treats ^S and ^Q as data.

You can also use the Communication Configuration menu to change the mode setting as described in the *Installing and Operating Your D216E+, D217, D413, and D463 Display Terminal* manual.

PCTERM Mode**all terminals**

| | |
|-------------------|--------------------|
| CSI ? 99 h | to set |
| 9B 3F 39 39 68 | (8-bit hex) |
| 1B 5B 3F 39 39 68 | (7-bit hex) |
| CSI ? 99 l | to reset (default) |
| 9B 3F 39 39 6C | (8-bit hex) |
| 1B 5B 3F 39 39 6C | (7-bit hex) |

PCTERM mode is a new (Data General-specific) function added to the VT320/100 emulation. This mode is designed to be used with the VP/ix environment running under INTERACTIVE UNIX. It allows a simple terminfo file to be devised for VP/ix that will emulate the MS-DOS® environment on a terminal connected to a multi-user 80386 PC running UNIX. See Chapter 5, "PCTERM," for complete details on this operating mode.

When the terminal enters PCTERM mode, autowrap is turned off, the keyboard is set to deliver IBM PC scan codes, G0 is set to PC Low characters, G1 is set to PC High characters, XON is set to "e", XOFF is set to "g", GL is set to G0, and GR is set to G1.

When the terminal exits PCTERM mode, autowrap is turned on, the keyboard is set back to ASCII codes, XON is set to "Ctrl-Q", XOFF is set to "Ctrl-S", and G0, G1, GL and GR are restored from a save area.

Transmission Control Sequences

These sequences select the host-preferred mode of control sequence transmission and are only used on the VT320 emulation. If the communications mode is set to 7-bits, then these control sequences have no affect. If the communications mode is set to 8-bits, then the host has the option of receiving 7-bit (C0) sequences or 8-bit (C1) codes. These control sequences do not affect the terminal's reception of control sequences sent by the host.

Transmit 7-bit Controls

D413/D463

ESC <space> F
1B 20 46 (hex)

Terminal transmits only 7-bit (C0) control sequences to the host (returns 8-bit (C1) codes as their 7-bit code equivalents).

Transmit 8-bit Controls

D413/D463

ESC <space> G
1B 20 47 (hex)

Lets the terminal transmit both 7-bit (C0) and 8-bit (C1) control sequences to the host.

User-Defined Key Control Sequences

User Defined Keys (UDK)

D413/D463

```
DCS <Pcl> ; <Plk> | <Kv> / <Stn> ; ... ; <Kv> / <Stn> ST
90 <Pcl> 3B <Plk> 7C <Kv> 2F <Stn> 3B ... 3B <Kv> 2F <Stn> 9C      (8-bit hex)
1B 50 <Pcl> 3B <Plk> 7C <Kv> 2F <Stn> 3B ... 3B <Kv> 2F <Stn> 1B 5C  (7-bit hex)
```

where

- <Pcl>** is the clear control parameter with one of the following values:
- none clear all keys before loading new values
 - 0 clear all keys before loading new values
 - 1 clear and load the key being defined with a new value.
- <Plk>** is the lock control parameter, with one of the values below. (If keys are locked, they may be unlocked only by using the General Set-Up menu.)
- none locks definable keys against redefinition
 - 0 locks definable keys against redefinition
 - 1 unlocks keys regarding redefinition.
- <Kv>** is the key value for a particular function key, with one of the following values.
- | | | | |
|-----|----|-----|-----------|
| F6 | 17 | F14 | 26 |
| F7 | 18 | F15 | 28 (Help) |
| F8 | 19 | F16 | 29 (Do) |
| F9 | 20 | F17 | 31 |
| F10 | 21 | F18 | 32 |
| F11 | 22 | F19 | 33 |
| F12 | 23 | F20 | 34 |
| F13 | 24 | | |
- <Stn>** is a string of hexadecimal bytes that defines the character sequence for the specified key.

Function keys F6 to F20 may be programmed with user-definable key sequences. A total of 255 bytes of volatile RAM is available for key definitions, and one key sequence may use up all or most of the available memory if desired.

For example, the control sequence "DCS 1;1|17/41424344454647 ST" defines function key F6 to "ABCDEFG". Where clear control <Pcl> is 1 to clear only the being defined, Lock control <Plk> is a 1 to prevent keys from being locked against future redefinition. The "|" character indicates the UDK control sequence, 17 the value for function key six F6, / indicates the start of data, and 41424344454647 are hex values for the characters "ABCDEFG".

Miscellaneous Control Sequences

Soft Terminal Reset

all terminals

```
CSI ! p
9B 21 70          (8-bit hex)
1B 5B 21 70      (7-bit hex)
```

Resets all operating modes (both ANSI standard and ANSI private) to defaults. Also resets the character sets to defaults. Refer to the following sections of this chapter: “ANSI Standard Mode”, “ANSI Private Mode”, and “Character Sets.”

Hard Terminal Reset

all terminals

```
ESC c
1B 63             (hex)
```

Causes the terminal to act as if the power had been shut off and turned back on again. We do not recommend using this control sequence since it will interfere with dual-host operation and may also cause the loss of characters from the host during the self-test sequence.

Alignment

all terminals

```
ESC # 8
1B 23 38         (hex)
```

Fills the screen with an alignment pattern of E's.

Display Character Generator Contents

all terminals

```
ESC # 9
1B 23 39         (hex)
```

Displays the contents of the character generator. This is a Data General extension to the Alignment command.

Set/Report Language**all terminals**

```

CSI ? <lang> E
9B 3F <lang> 45      (8-bit hex)
1B 5B 3F <lang> 45  (7-bit hex)

```

Sets the current keyboard language to the supplied language. If no language is supplied, then the terminal defaults to the current language. Reports the keyboard language as a single printable character formed by adding 1F hex to the language numbers as shown in the table below. This command is only used for diagnostic and testing purposes.

| <lang> | <char> | Language |
|--------|--------|------------------|
| 1 | | North American |
| 2 | ! | United Kingdom |
| 3 | " | French/Belgian |
| 4 | # | German/Dutch |
| 5 | \$ | Swedish/Finnish |
| 6 | % | Spanish |
| 7 | & | Danish |
| 8 | ' | Norwegian |
| 9 | (| Swiss (German) |
| 10 |) | Swiss (French) |
| 11 | * | Italian |
| 12 | + | Canadian English |
| 13 | , | Canadian French |
| 14 | - | Katakana |
| 15 | . | Portuguese |

Set Clock Time**all terminals**

```

CSI ? ; <mode> ; ; ; <hours> ;<minutes> F
9B 3F 3B <mode> 3B 3B 3B <hours> 3B <minutes> 46      (8-bit hex)
1B 5B 3F 3B <mode> 3B 3B 3B <hours> 3B <minutes> 46  (7-bit hex)

```

where

- <mode> sets the hour mode:
 - 1 sets 12 hour (AM/PM) mode
 - 2 sets 24 hour (military) mode
- <hours> is the current time, from 0 through 23 (0 is midnight)
- <minutes> is the current time from 0 through 59

If either <hours> or <minutes> are specified, then the seconds counter is reset to 0. This control sequence has been altered from its previous format used in the D216/D412/D462 terminals. The parameters for turning the clock on and off and for positioning the clock have been removed. Any data specified in the positions for these unused parameters will be ignored.

Hot Key Switch

D413/D463

```
CSI ? G
9B 3F 47          (8-bit hex)
1B 5B 3F 47      (7-bit hex)
```

Switches terminal operation to another emulation. This control sequence is ignored if no other emulation is specified in the Configuration Menu. For more information on the Configuration Menu, refer to the *Installing and Operating Your D216E, D217, D413, and D463 Display Terminals* manual.

Set Top and Bottom Margins (STBM)

all terminals

```
CSI <top> ; <bottom> r
9B <top> 3B <bottom> 72      (8-bit hex)
1B 5B <top> 3B <bottom> 72  (7-bit hex)
```

where

<top> and <bottom> are numeric parameters that define the first and last line of the scrolling region, respectively. Line number parameters are referenced from the top of the screen where the top line is always 1. If no parameters are supplied with this control sequence, then <top> and <bottom> are assumed to be the first and last line of the entire screen.

Sets the top and bottom margins to define a window or scrolling region.

Bit Dump Screen

D413/D463

```
CSI 6 i
9B 36 69          (8-bit hex)
1B 5B 36 69      (7-bit hex)
```

Dumps the contents of the screen graphically to a Data General or IBM Pro-Printer compatible printer. No action is taken if a graphics capable printer is not selected. This control sequence allows the printing of double-height and double-width characters, as well as DRCS (soft character set) characters, when the terminal is used with a graphics printer. This feature is not available in standard DEC VT320 terminals.

Force Display**all terminals**

```
DC2 <character>
12 <character>          (hex)
```

This is a new (Data General–specific) function that forces the terminal to display the next character after the control code literally. It is used as an escape code (primarily for VP/ix) to allow access to the 128–character character sets. Ordinarily, codes 00 hex through 1F hex, and 80 hex through 9F hex, are interpreted as control sequences. With this code given first, they will be forced into being displayed on the screen.

Data Trap Mode**all terminals**

This mode lets the user view data from the host as a hex data stream, similar to AOS/VS's X DISPLAY/HEX command. Data Trap mode is entered and exited via the Configuration Menu. For more information on the Configuration Menu, refer to the *Installing and Operating Your D216E, D217, D413, and D463 Display Terminals* manual.

Outbound (host to terminal) data is displayed as 16 hex bytes per line (on the left, below) and the corresponding 8–bit characters on the right, as follows:

```
40 30 31 32 33 34 35 36 37 38 39 21 23 24 2E 0A      @0123456789!#$..
41 42 43 44 45 44 47 48 49 4A 4B 4C 4D 4E 4F 50      ABCDEFGHIJKLMNOP
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70      abcdefghijklmnop
```

NOTE: The Index entry "VT320/100 mode control sequences by hex order" lists all VT320/100 control sequences by hex order. Refer to this entry to determine which control sequences are returned while using Data Trap mode.

Select Active Status Display (SASD)**D413/D463**

```
CSI <Ps> $ }
9B <Ps> 24 7D          (8–bit hex)
1B 5B <Ps> 24 7D      (7–bit hex)
```

where

- <Ps> determines where to place the cursor;
- 0 places the cursor in the active display area. This is the default selection if no parameter is specified.
 - 1 places the cursor on the status line. When the cursor is in the status line, the status line behaves as if it were a one–row window.

Sets the cursor to either the main display area or the status line. If the status line is not set to host–writable (SSDT), then this control sequence has no effect.

Select Status Line Type (SSDT)**D413/D463**

```

CSI <Ps> $ ~
9B <Ps> 24 7E          (8-bit hex)
1B 5B <Ps> 24 7E      (7-bit hex)

```

where

<Ps> determines the status line type:

- 0 makes the 25th line blank (VT320 compatible)
- 1 displays terminal status on the 25th line
- 2 makes the status line host-writable (for the Select Active Display (SASD) control sequence).
- 3 25th row

This control sequence selects the type of status line displayed on the 25th line of the screen.

Set Conformance Level (SCL)**all terminals**

```

CSI <level> ; <controls> " p
9B <level> 3B <controls> 22 70          (8-bit hex)
1B 5B <level> 3B <controls> 22 70      (7-bit hex)

```

where

<level> sets the terminal type:

- 61 VT100
- 62 VT220
- 63 VT320.

<controls> has one of the following values:

- 1 sets 7-bit controls
- 2 sets 8-bit controls

This control sequence sets the terminal's conformance, or compatibility, level.

Set Device Options**all terminals**

```

CSI ? <type> ; <ff> ; <cs> ; <graph> Q
9B 3F <type> 3B <ff> 3B <cs> 3B <graph> 57      (8-bit hex)
1B 5B 3F <type> 3B <ff> 3B <cs> 3B <graph> 51  (7-bit hex)

```

where

- <type> sets the device type:
- 1 printer
 - 2 mouse.
- <ff> specifies the printer form feed setting:
- 1 none (no form feed character sent to printer)
 - 2 form feed character sent before the page
 - 3 form feed character send after the page
 - 4 form feed characters are send both before and after the page prints
- <cs> specifies the printer character set:
- 1 U.S. ASCII and DGI
 - 2 U.S. ASCII and VT Multinational
 - 3 IBM PC
 - 4 NRC only
 - 5 NRC and VT Line Drawing (special graphics set)
 - 6 Katakana
- <graph> specifies the graphics capability of the printer:
- 1 none
 - 2 Data General
 - 3 IBM PC

Sets the type and preference options for the auxiliary device, if it is available; the auxiliary device in not available if the terminal is in Dual Host mode and the Port mode is set to "Both" (from Configuration Menu selections). Any parameter omitted defaults to the current setting. This control sequence is specific to the Data General VT-emulations, and is not available on DEC VT320 terminals.

Split Screen**D413/D463**

```

CSI ? <first> ; <top> ; <split> I
9B 3F <first> 3B <top> 3B <split> 49      (8-bit hex)
1B 5B 3F <first> 3B <top> 3B <split> 49  (7-bit hex)

```

where

- <first> specifies the first physical row of the current emulation to display in this partition, and is from 1 through 24. If the parameter given is large enough that lines past the end of the screen would be visible, this value is set to the maximum possible.
- <top> specifies the emulation to be shown in the top partition:
0 host
1 auxiliary
- <split> specifies the number of rows to allocate for the top partition, and is from 0 through 23. If you enter 0 here, split screen is disabled.

Sets or resets split screen. To enter split screen mode, you must first define two emulations in the Terminal Operations Menu. When split screen is set, one emulation will occupy the top *n* rows of the screen, and the other emulation will occupy the bottom “24 – *n*” rows. Because only a portion of each emulation’s screen is displayed, this control sequence lets you select the starting row in each partition.

Any parameter omitted defaults to the current setting. This control sequence is specific to the Data General VT-emulations, and is not available on DEC VT320 terminals.

NOTE: Press the CMD key and the Cursor Up/Down to move the split screen up or down. This command sequence also toggles the split screen setting. Press the CMD and SHIFT keys and Cursor up/down to move the first row of the current emulation up or down.

Reporting Control Sequences

A report is a character sequence that contains information about status, terminal identification, or current parameters. It is generated by the terminal in response to a request from the host computer.

Terminal Identification (DECID)

all terminals

ESC Z
1B 5A (hex)

Causes the terminal to send a Primary Device Attributes (DA) response sequence. Use this command to identify the terminal's VT52 mode; otherwise use the Primary Device Attribute Request (DA).

Device Status Report (DSR)

all terminals

CSI 5 n
9B 35 6E (8-bit hex)
1B 5B 35 6E (7-bit hex)

Reports the terminal operating status, where the terminal responds to the host with:

CSI 0 n no malfunction was detected during powerup self-test
9B 30 6E (8-bit hex)
1B 5B 30 6E (7-bit hex)

CSI 3 n malfunction was detected during powerup self-test
9B 33 6E (8-bit hex)
1B 5B 33 6E (7-bit hex)

Primary Device Attribute Request (DA)**all terminals**

```

CSI c
9B 63 (8-bit hex)
1B 5B 63 (7-bit hex)

```

Returns the current operating level and options of the terminal. The response string has the following format:

```

CSI ? <feature> ; ... ; <feature> c
9B 3F <feature> 3B ... 3B <feature> 63 (8-bit hex)
1B 5B 3F <feature> 3B ... 3B <feature> 63 (7-bit hex)

```

where

<level> indicates the operating level:

- 61 VT100
- 62 VT220
- 63 VT320

<feature> indicates the features in the following list:

- 1 132 column capability
- 2 printer port
- 6 selective erase
- 7 soft character sets (DRCS) (D463 only)
- 8 user defined keys (UDK)
- 9 National Replacement Sets (NRC)
- 11 25th row is the status line
- 14 8-bit communications interface
- 17 terminal state interrogation (TABSR, RQM, ...)

Response 63 for the <level> parameter and <feature> responses 11, 14, and 17 are new to the DEC VT320. Otherwise, this control sequence is identical to the DEC VT220 control sequence.

Secondary Device Attribute Request (SDA)**all terminals**

```
CSI > c
9B 3E 63          (8-bit hex)
1B 5B 3E 63      (7-bit hex)
```

This control sequence returns additional information specific to the VT320 emulation. The response string is:

```
CSI > <Pp> ; <Pv> ; <Po> c
9B 3E <Pp> 3B <Pv> 3B <Po> 63      (8-bit hex)
1B 5B 3E <Pp> 3B <Pv> 3B <Po> 63  (7-bit hex)
```

where

<Pp> is the identification code, where:

```
1   VT220
24  VT320
```

<Pv> is the firmware revision level, where 11 is Version 1.1

<Po> is the number of hardware options installed, which should always be 0

Response 24 for the <Pp> parameter and the <Pv> parameter are new to the DEC VT320. Otherwise, this control sequence is identical to the DEC VT220 control sequence.

Cursor Position Report (CPR)**all terminals**

```
CSI 6 n
9B 36 6E          (8-bit hex)
1B 5B 36 6E      (7-bit hex)
```

Reports the current cursor position, with a sequence in the format below:

```
CSI <vert> ; <horiz> R
9B <vert> 3B <horiz> 52      (8-bit hex)
1B 5B <vert> 3B <horiz> 52  (7-bit hex)
```

where

<vert> and <horiz> are numeric parameters that indicate the current line and column position of the cursor.

User Defined Key Status**D413/D463**

CSI ? 25 n
 9B 3F 32 35 6E (8-bit hex)
 1B 5B 3F 32 35 6E (7-bit hex)

Reports the status of the user defined keys, where the terminal's response is as follows:

CSI ? 20 n means user defined keys are unlocked
 9B 3F 32 30 6E (8-bit hex)
 1B 5B 3F 32 30 6E (7-bit hex)

CSI ? 21 n means user defined keys are locked
 9B 3F 32 31 6E (8-bit hex)
 1B 5B 3F 32 31 6E (7-bit hex)

Keyboard Language Report**D413/D463**

CSI ? 26 n
 9B 3F 32 36 6E (8-bit hex)
 1B 5B 3F 32 36 6E (7-bit hex)

Reports the current keyboard language in use, where the terminal's response is as follows:

CSI ? 27 ; <language> n
 9B 3F 32 37 3B <language> 6E (8-bit hex)
 1B 5B 3F 32 37 3B <language> 6E (7-bit hex)

where <language>

reports the current keyboard language, from the following list:

| | |
|----|---|
| 0 | Undetermined |
| 1 | North American |
| 2 | British |
| 3 | Flemish |
| 4 | French Canadian |
| 5 | Danish |
| 6 | Finnish |
| 7 | German |
| 8 | Dutch |
| 9 | Italian |
| 10 | Swiss/French |
| 11 | Swiss/German |
| 12 | Swedish |
| 13 | Norwegian |
| 14 | French/Belgian |
| 15 | Spanish |
| 16 | Portuguese (was Katakana on D216/D412/D462) |
| 99 | Katakana |

In addition, for testing purposes there is a Data General private control sequence to set the Keyboard Language, as shown below:

```
CSI ? <language> E
9B 3F 32 37 3B <language> 45          (8-bit hex)
1B 5B 3F 32 37 3B <language> 45      (7-bit hex)
```

where

<language> is a numeric parameter, from the list above, indicating the keyboard language type.

Answerback**D413/D463**

ENQ

05

(7-bit hex)

Transmits answerback message (set in the Configuration Menu) to the host if an answerback message is set and answerback is enabled.

Printer Port Status**all terminals**

CSI ? 15 n

9B 3F 31 35 6E

(8-bit hex)

1B 5B 3F 31 35 6E

(7-bit hex)

Reports the status of the terminal's printer port:

CSI ? 13 n

means DTR was never asserted; no
printer attached

9B 3F 31 33 6E

(8-bit hex)

1B 5B 3F 31 33 6E

(7-bit hex)

CSI ? 10 n

means DTR was asserted; printer is ready

9B 3F 31 30 6E

(8-bit hex)

1B 5B 3F 31 30 6E

(7-bit hex)

CSI ? 11 n

means DTR is not currently asserted;
printer not ready

9B 3F 31 31 6E

(8-bit hex)

1B 5B 3F 31 31 6E

(7-bit hex)

This status request checks the state of Data Terminal Ready (DTR) on the printer port.



Request Terminal State Report (RQTSR)

D413/D463

```
CSI 1 $ u
9B 31 24 75 (8-bit hex)
1B 5B 31 24 75 (7-bit hex)
```

This sequence returns a DCS string from the terminal to the host, as follows:

```
DCS 1 $ s <D1> ... <Dn> <Ck1> <Ck2> ST
90 31 24 73 <D1> ... <Dn> <Ck1> <Ck2> 9C (8-bit hex)
1B 50 31 24 73 <D1> ... <Dn> <Ck1> <Ck2> 1B 5C (7-bit hex)
```

where

<D1> ... <Dn> are characters in the range 40 to 4F hex, representing encoded state information.

<Ck1><Ck2> are a checksum equal to the 2's complement of all of the data above, such that the following statement holds:

$$\text{sum}(\langle D1 \rangle \dots \langle Dn \rangle) + ((\langle Ck2 \rangle \& 0xF) \ll 4) + (\langle Ck1 \rangle \& 0xF) = 0$$

The contents of the returned string are an encoded form of the internal state of the terminal. This control sequence allows an application to fetch the current state of the terminal, make changes, and then revert the state on exit. It is not intended that any application should examine the data returned to find state information.

This control sequence is used in conjunction with the Restore Terminal State (RSTS) control sequence. All ANSI standard modes (CSI h and CSI l) and ANSI private modes (CSI ? h and CSI ? l) are saved.

Restore Terminal State (RSTS)

D413/D463

```
DCS 1 $ p <D1> ... <Dn> <Ck1> <Ck2> ST
90 31 24 70 <D1> ... <Dn> <Ck1> <Ck2> 9C (8-bit hex)
1B 50 31 24 70 <D1> ... <Dn> <Ck1> <Ck2> 1B 5C (7-bit hex)
```

where

all parameters are in the same format as the response strings given by Request Terminal State Report (RQTSR). If any error is detected in the given data, the rest will be ignored. This may put the terminal into an indeterminate state.

Restores the terminal state from an encoded string produced by the Request Terminal State Report (RQTSR) control sequence. If any error is detected in the given data, the rest will be ignored. This may put the terminal into an indeterminate state.

Request Presentation State Report (RQPSR)**D413/D463**

CSI <Ps> \$ w
 9B <Ps> 24 77 (8-bit hex)
 1B 5B <Ps> 24 77 (7-bit hex)

For a cursor information report, set <Ps> to 1. The returned DCS string is as follows:

DCS 1 \$ u <Pr>;<Pc>;<Pp>;<Srend>;<Satt>;<Sflag>;<Pgl>;<Pgr>;<Scss>;<Sdesig> ST

90 31 24 75 <Pr>;<Pc>;<Pp>;<Srend>;<Satt>;<Sflag>;<Pgl>;<Pgr>;
 <Scss>;<Sdesig> 9C (8-bit hex)

1B 50 31 24 75 <Pr>;<Pc>;<Pp>;<Srend>;<Satt>;<Sflag>;<Pgl>;<Pgr>;
 <Scss>;<Sdesig> 1B 5C (7-bit hex)

where

- <Pr> is the line number the cursor is on (decimal)
- <Pc> is the column number the cursor is on (decimal)
- <Pp> is the page number, which is always 0 on a VT320
- <Srend> is the current rendition (SGR) encoded as a single character in the form 01E0 RBUB:
 - E indicates if another byte follows (extension), is currently unused, and should be set to 0
 - R is 1 for reverse video on
 - B is 1 for blink on
 - U is 1 for underscore on
 - b is 1 for bold on
- <Satt> is the selective erase attribute encoded as a single character in the format 01E0 000S:
 - E indicates if another byte follows (extension), is currently unused, and should be set to 0
 - S is 1 for protected, 0 for unprotected

- <Sflag> is certain saved flags and is in the format of 01E0 A320:
 E indicates if another byte follows (extension), is currently unused, and should be set to 0
 A is 1 for autowrap pending (not very useful, although it is in the DEC version of this sequence)
 3 is 1 for SS3 pending (cannot be set, although it is in the DEC version of this sequence)
 2 is 1 for SS2 pending (cannot be set, although it is in the DEC version of this sequence)
- <Pgl> is the Gx set mapped into GL (decimal).
- <Pgr> is the Gx set mapped into GR (decimal).
- <Scss> is the number and size of the character sets, encoded into a single character: 01E0 DCBA, where:
 E indicates if another byte follows (extension), is currently unused, and should be set to 0
 D indicates G3, and is set to 0 for 94 characters, set to 1 for 96 characters
 C indicates is G2, and is set to 0 for 94 characters, set to 1 for 96 characters
 B indicates is G1, and is set to 0 for 94 characters, set to 1 for 96 characters
 A indicates is G0, and is set to 0 for 94 characters, set to 1 for 96 characters
- <Sdesig> is four Dscs-style character set names for G0 through G3.

For a tab stop report, set <Ps> to 2. The returned DCS string is as follows:

```
DCS 2 $ u <Ts1> / <Ts2> / ... <Tsn> ST
90 32 24 77 <Ts1> 2F <Ts2> 2F ... <Tsn> 9C      (8-bit hex)
1B 50 32 24 77 <Ts1> 2F <Ts2> 2F ... <Tsn> 1B 5C  (7-bit hex)
```

where

<Ts1> ... <Tsn> are decimal column numbers representing set tabs. Note that the separators here are / (2F hex), and not ; (3B hex).

The contents of this string are either the cursor status (the same information as that saved by the Save Cursor and the Restore Cursor control sequences), or the current tab stops. Unlike Request Terminal State Report (RQTSR), the data returned by this control sequence may be examined and acted upon by the application program. This control sequence is often used in conjunction with the Restore Presentation State (RSPS) control sequence.



Restore Presentation State (RSPS)

D413/D463

```
DCS <Ps> $ t <D1> ... <Dn> ST
90 <Ps> 24 74 <D1> ... <Dn> 9C      (8-bit hex)
1B 50 <Ps> 24 74 <D1> ... <Dn> 1B 5C (7-bit hex)
```

where

<Ps> is 1 to restore the cursor state, and 2 to restore the tab settings

<D1> ... <Dn> are in the same format as the response strings given by Request Presentation State Report (RQPSR). If any error is detected in the given data, the rest will be ignored. This may put the terminal into an indeterminate state.

Restores either the current cursor state or the tab settings, as saved using Request Presentation State Report (RQPSR).

Request Mode (RQM)

D413/D463

```
CSI [?] <Pa> $ p
9B [3F] <Pa> 24 70      (8-bit hex)
1B 5B [3F] <Pa> 24 70 (7-bit hex)
```

where

? (3F hex) tells the terminal to return only ANSI private modes. Do not include this character if you want a report on ANSI standard modes.

<Pa> is the number of the mode (from 0 to 255) on which you want a report:

ANSI standard modes

| | |
|----------------|-------------------------|
| 2 (32 hex) | Keyboard Action mode |
| 4 (34 hex) | Insert/Replace mode |
| 12 (31 32 hex) | Send/Receive mode |
| 20 (32 30 hex) | Line Feed/New Line mode |

ANSI private modes

| | |
|----------------|--------------------------------------|
| 1 (31 hex) | Application/ANSI Cursor Keys mode |
| 3 (33 hex) | Column mode |
| 4 (34 hex) | Scrolling mode |
| 5 (35 hex) | Screen mode |
| 6 (36 hex) | Cursor Origin mode |
| 7 (37 hex) | Auto Wrap mode |
| 8 (38 hex) | Auto Repeat mode |
| 18 (31 38 hex) | Print Form Feed mode |
| 19 (31 39 hex) | Print Extent mode |
| 25 (32 35 hex) | Text Cursor Enable mode |
| 42 (34 32 hex) | Multi/National Character Set mode |
| 66 (36 36 hex) | Numeric Keypad mode |
| 67 (36 37 hex) | Backarrow Key mode |
| 68 (36 38 hex) | Typewriter/Data Processing Keys mode |
| 99 (39 39 hex) | PCTERM mode |

Returns the current value of any of the terminal's mode settings. If the "?" character is absent, ANSI standard modes will be reported. If it is present, ANSI private modes will be returned. The control sequence returned is in the following format:

```
CSI [?] <Pa> ; <Ps> $ y
9B [3F] <Pa> 3B <Ps> 24 79          (8-bit hex)
1B 5B [3F] <Pa> 3B <Ps> 24 79      (7-bit hex)
```

where

? (3F hex) is present if and only if this is a private mode.

<Pa> is the mode number that was requested.

<Ps> is the current setting of the mode:

- 1 set
- 2 reset
- 3 permanently set
- 4 permanently reset

This control sequence can return information on only one mode at a time.

Request User-Preferred Supplemental Set (RQUPSS)**D413/D463**

```

CSI & u
9B 26 75          (8-bit hex)
1B 5B 26 75      (7-bit hex)

```

Returns data on which set has been assigned as the user-preferred supplemental character set, in the following form:

```

DCS <Ps> ! <Pss> ST
90 <Ps> 21 <Pss> 9C          (8-bit hex)
1B 50 <Ps> 21 <Pss> 1B 5C    (7-bit hex)

```

where <Ps> and <Pss> have the following values:

| <Ps> | <Pss> | |
|------|-------|---|
| 0 | %5 | supplemental graphics set is DEC Multinational |
| 1 | A | supplemental graphics set is ISO Latin-1 (8859/1.2) |

These parameters are the same as those used in the Assign User Preferred Supplemental Set (AUPSS) control sequence.

Read Cursor Contents**all terminals**

```

CSI ? 17 h
9B 3F 31 37 68          (8-bit hex)
1B 5B 3F 31 37 68      (7-bit hex)

```

The read cursor content control sequence is intended for diagnostic purposes only. It returns five bytes from the terminal in the following formats:

Byte 1 is in the form 0C00 00NN, where C is the characters per line (0 for 80, 1 for 132) and NN is the line mode (00 for normal, 01 for double width, 10 for double height top line, and 11 for double height bottom line).

Byte 2 is in the form 0000 CCCC, where CCCC are the upper four bits of the character underneath the cursor.

Byte 3 is in the form 0000 CCCC, where CCCC are the lower four bits of the character underneath the cursor.

Byte 4 is in the form 00P0 BURD, where P is the protection attribute of the character underneath the cursor (0 for unprotected, 1 for protected) and BURD are the character attributes (blink, underscore, reverse, and dim).

Byte 5 is in the form 0000 BBBB, where BBBB is the character set selection (0 or 1 for the CGEN, and 4 through 15 for the DLL).

The data returned by this control sequence is not intended for user-applications because the returned data may vary with future terminals.

Request Selection or Setting (RQSS)

D413/D463

```
DCS $ q <Seqn> ST
90 34 71 <Seqn> 9C          (8-bit hex)
1B 50 34 71 <Seqn> 1B 5C    (7-bit hex)
```

where

<Seqn> is one of the following requests:

- \$ } (24 7D hex) cursor on/off status line, set by Select Active Display (SASD)
- \$ ~ (24 7E hex) status line type, set by Select Status Line Type (SSDT)
- " q (22 71 hex) character protect on/off, set by Select Character Attributes (SCA)
- " p (22 70 hex) operation mode, set by Set Conformance Level (SCL)
- r (72 hex) windows, set by Set Top and Bottom Margins (STBM)
- m (6D hex) character attributes, set by Select Graphic Rendition (SGR)

Returns the current status of several different terminal features with a string in the following format:

```
DCS <Ps> $ r <Resp> ST
90 <Ps> 24 72 <Resp> 9C          (8-bit hex)
1B 50 <Ps> 24 72 <Resp> 1B 5C    (7-bit hex)
```

where

<Ps> is 0 if the request was invalid, 1 if valid

<Resp> is in the form of the corresponding CSI control sequence

Some examples of these host sequences and the terminal's response are shown below:

```
Host:      DCS $ q $ } ST          get SASD
Terminal:  DCS 1 $ r 0 $ } ST      in scrolling area
```

```
Host:      1B 50 24 71 24 7D 1B 5C    get SASD (7-bit hex)
Terminal:  1B 50 31 24 72 30 24 7D 1B 5C in scrolling area (7-bit hex)
```

```
Host:      DCS $ q $ ~ ST          get SSDT
Terminal:  DCS 1 $ r 1 $ ~ ST      indicator status
```

```
Host:      1B 50 24 71 24 7E 1B 5C    get SSDT (7-bit hex)
Terminal:  1B 50 31 24 72 31 24 7E 1B 5C indicator status (7-bit hex)
```

Host: DCS \$ q m ST get SGR
Terminal: DCS 1 \$ r 0 ; 1 m ST bold is on

Host: 1B 50 24 71 6D 1B 5C get SGR (7-bit hex)
Terminal: 1B 50 31 24 72 30 3B 31 6D 1B 5C bold is on (7-bit hex)

In general, the <Resp> portion of the return string may be prefaced by CSI and used to set the indicated mode.

Printing Control Sequences

Auto Print Mode

all terminals

| | |
|----------------|---------------------|
| CSI ? 5 i | turns Auto Print on |
| 9B 3F 35 69 | (8-bit hex) |
| 1B 5B 3F 35 69 | (7-bit hex) |

| | |
|----------------|----------------------|
| CSI ? 4 i | turns Auto Print off |
| 9B 3F 34 69 | (8-bit hex) |
| 1B 5B 3F 34 69 | (7-bit hex) |

Auto Print mode causes the current cursor line to be printed when the terminal receives a Line Feed (LF), Form Feed (FF), or Vertical Tab (VT), or when the line automatically wraps. All keyboard printing functions such as Print Screen are operational in this mode.

Auto Print mode may be toggled on or off manually from the keyboard by pressing Local Print. The control sequences, above, may be used by the host to access Auto Print Mode. This mode may also be selected from the Print Option Menu. For information on using the Print Option Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Print Screen

all terminals

| | |
|----------|-------------|
| CSI i | |
| 9B 69 | (8-bit hex) |
| 1B 5B 69 | (7-bit hex) |

Prints a full screen of text. The host may also use the print screen function by sending the control sequence. This option may also be selected in the Print Option Menu. For information on using the Print Option Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*.

Print Cursor Line

all terminals

| | |
|----------------|-------------|
| CSI ? 1 i | |
| 9B 3F 31 69 | (8-bit hex) |
| 1B 5B 3F 31 69 | (7-bit hex) |

Prints the display line containing the cursor.

Print Controller Mode**all terminals**

| | |
|-------------|----------------------------|
| CSI 5 i | turns Print Controller on |
| 9B 35 69 | (8-bit hex) |
| 1B 5B 35 69 | (7-bit hex) |
| CSI 4 i | turns Print Controller off |
| 9B 34 69 | (8-bit hex) |
| 1B 5B 34 69 | (7-bit hex) |

The Print Controller mode lets the host computer have direct control of the printer. Characters received from the host go directly to the printer with the exception of XON, XOFF, CSI 5 i, and CSI 4 i. These sequences are used as control codes for the Print Controller mode. Characters from the keyboard are still transmitted to the host in this mode. This option may also be selected in the Print Option Menu. For information on using the Print Option Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*. The print controller mode can be entered from Auto Print mode, but does not restore Auto Print mode when turned off.

End of Section

VT52 Emulation Operations and Escape Sequences

This section provides a summary of the operations information specific to the VT52 emulation on the D217/D413/D463 terminals.

In particular, this section explains the subjects covered in the following list:

Character Sets and Graphics

Keyboard Generated Codes

VT52 Escape Sequences

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and various VT52 emulator reference material is covered in related appendices, located at the rear of this manual.

Character Sets and Graphics

The VT52 emulation supports only the National Replacement Character (NRC) sets, which you select from the Emulation Configuration Menu. For information on using the Emulation Configuration Menu, refer to the manual *Installing and Operating D216E+, D217, D413, and D463 Display Terminals*. You can access the Special Graphics character set by pressing the ESC key and typing F (1B 46 hex). You exit the graphics mode by pressing the ESC key and typing G (1B 47 hex).

Keyboard Generated Codes

The keyboard codes generated by the VT52 emulation are shown in Table 3-16, Table 3-17, and Table 3-18. The tables cross reference the VT52 keyboard with the 101-key keyboard (similar to an IBM-PC AT keyboard) and the 107-key Data General proprietary keyboard.

Table 3-16 Keyboard Generated Codes — Function Keys (VT52)

| 107-key Keyboard | 101-key Keyboard | VT52 Keyboard | Code |
|------------------|------------------|---------------|-------------------|
| F6 | F6 | Escape | 1B hex |
| F7 | F7 | Backspace | 08 hex |
| F8 | F8 | Line Feed | 0A hex |
| Local Print | Print Screen | Print | Print-Menu |
| Scroll Rate | Scroll Lock | Menu | none |
| Hold | Pause | Hold | Hold Screen |
| C1 | Alt-F9 | PF1 | ESC P (1B 50 hex) |
| C2 | Alt-F10 | PF2 | ESC Q (1B 51 hex) |
| C3 | Alt-F11 | PF3 | ESC R (1B 52 hex) |
| C4 | Alt-F12 | PF4 | ESC S (1B 53 hex) |

Table 3-17 Keyboard Generated Codes — Cursor Keys (VT52)

| 107-key Keyboard | 101-key Keyboard | VT320/100 Keyboard | Code |
|------------------|------------------|--------------------|-------------------|
| Uparrow | Uparrow | Uparrow | ESC A (1B 41 hex) |
| Rightarrow | Rightarrow | Rightarrow | ESC B (1B 42 hex) |
| Leftarrow | Leftarrow | Leftarrow | ESC C (1B 43 hex) |
| Downarrow | Downarrow | Downarrow | ESC D (1B 44 hex) |
| Home | n/a | n/a | ESC H (1B 48 hex) |

Table 3-18 Keyboard Generated Codes — Numeric Keypad (VT52)

| 107-key Keyboard ¹ | 101-key Keyboard | VT52 Keyboard with Num Lock On | VT52 Keyboard with Num Lock Off |
|-------------------------------|------------------|--------------------------------|---------------------------------|
| n/a | Num Lock | On | Off |
| n/a | / | / | none |
| n/a | * | * | none |
| – (minus) | – | – (minus) | ESC ? m (1B 3F 6D hex) |
| , (comma) | + | , (comma) | ESC ? l (1B 3F 6C hex) |
| . (period) | ./Delete | . (period) | ESC ? n (1B 3F 6E hex) |
| 0 | 0/Insert | 0 | ESC ? p (1B 3F 70 hex) |
| 1 | 1/End | 1 | ESC ? q (1B 3F 71 hex) |
| 2 | 2/Downarrow | 2 | ESC ? r (1B 3F 72 hex) |
| 3 | 3/Pg Dn | 3 | ESC ? s (1B 3F 73 hex) |
| 4 | 4/Leftarrow | 4 | ESC ? t (1B 3F 74 hex) |
| 5 | 5 | 5 | ESC ? u (1B 3F 75 hex) |
| 6 | 6/Rightarrow | 6 | ESC ? v (1B 3F 76 hex) |
| 7 | 7/Home | 7 | ESC ? w (1B 3F 77 hex) |
| 8 | 8/Uparrow | 8 | ESC ? x (1B 3F 78 hex) |
| 9 | 9Pg Up | 9 | ESC ? y (1B 3F 79 hex) |
| Enter ² | Enter | CR (0D hex) | ESC ? M (1B 3F 4D hex) |

¹ 107-key keyboards numeric keypads have no Num Lock On/Off mofe.

VT52 Escape Sequences

The escape sequences for the VT52 emulation, and their functions, are shown in Table 3-19.

Table 3-19 VT52 Escape Sequences

| Sequence | Action |
|---|-------------------------------|
| ESC = (1B 3D hex) | Enter alternate keypad mode |
| ESC > (1B 3E hex) | Exit alternate keypad mode |
| ESC < (1B 3C hex) | Enter ANSI mode (VT100 mode) |
| ESC A (1B 41 hex) | Cursor up |
| ESC B (1B 42 hex) | Cursor down |
| ESC C (1B 43 hex) | Cursor right |
| ESC D (1B 44 hex) | Cursor left |
| ESC F (1B 46 hex) | Enter graphics mode |
| ESC G (1B 47 hex) | Exit graphics mode |
| ESC H (1B 48 hex) | Move cursor to home |
| ESC I (1B 49 hex) | Reverse line feed |
| ESC J (1B 4A hex) | Erase to end of screen |
| ESC K (1B 4B hex) | Erase to end of line |
| ESC P (1B 50 hex) | Read cursor contents |
| ESC V (1B 56 hex) | Print cursor line |
| ESC W (1B 57 hex) | Enter printer controller mode |
| ESC X (1B 58 hex) | Exit printer controller mode |
| ESC Y <line><col> ¹
(1B 59 hex) | Position cursor ¹ |
| ESC Z (1B 5A hex) | Identify |
| ESC ^ (1B 5E hex) | Enter auto print mode |
| ESC _ (1B 5F hex) | Exit auto print mode |
| ESC] (1B 5D hex) | Print screen |

¹ <line> and <col> are encoded by taking the desired row (0 to 23 decimal) and column (0 to 79 decimal) and adding a 20 hex offset. Thus, the valid ranges (in ASCII form) for row and column are from <space> to "7", and from <space> to "o" respectively. In hex, row ranges from 20 to 37; column ranges from 20 to 4F.

End of Section

End of Chapter



Chapter 4

Tektronix 4010 Emulation

This chapter provides the information necessary to program host resident software that will be accessed through the Tektronix 4010 emulation running on the D463 Data General terminal. This chapter has the major sections listed below:

Emulation Features

Overview of Operational Modes

Alphanumeric Mode

Graphic Plot Mode

Graphics Input Mode

Hard Copy Command

Hot-Key Switch

User Selectable Options

Control Codes

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and reference material, which may apply to the Tektronix 4010 emulation, is covered in appendices, located at the rear of this manual.

Emulation Features

The Tektronix 4010 emulation provides maximum compatibility with Tektronix 4010 operational modes and implements 4010 vector graphics with a raster display. Some of the main programming features of the Tektronix 4010 emulation are listed below:

- 36 lines of 101 displayed characters in an 8 x 8 dot matrix
- 1024 x 780 graphic display window mapped into a hardware resolution of 810 x 288
- Graphic hard copy ability on both Data General printers and IBM Pro-Printers

The Tektronix 4010 emulation allows complete control of the terminal from both local and host sources in all Tektronix operational modes. Use of these features is explained fully in the following sections.

Overview of Operational Modes

The operational modes of the Tektronix 4010 emulation can be selected either by a control sequence generated from the host computer or by the keyboard. Data entry can be confined to the keyboard through the use of the Cmd-On Line key sequence. The operational modes are listed below:

- Alphanumeric Mode
- Graphic Plot Mode
- Graphic Input Mode

Alphanumeric Mode

While in the Alphanumeric Mode, the Tektronix 4010 emulation displays any of the 95 printable keyboard characters. In the original Tektronix 4010 terminal, lower-case letters were not displayed and could not be entered from the keyboard; if attempted, the terminal would display these characters in upper case. However, our emulation allows lower case letters to be generated from the keyboard and to be displayed as text. This makes it possible to test all graphics coordinates from the keyboard. A block cursor is displayed to indicate the position of the next displayed character. Our implementation of the Tektronix 4010 emulation has a screen size of 36 lines with a maximum of 101 characters per line. The cursor automatically wraps to the next line when it passes the right-hand screen margin. When the Tektronix 4010 emulation is selected, the terminal is reset to the Alphanumeric mode, and the cursor appears in the upper-left (Home) position.

Margins

The Tektronix 4010 emulation has two left margins on the alphanumeric screen. The first, called Margin 0, is the extreme left of the screen (at the first character location, i.e. column 1). The second is Margin 1. It is located at the 51st character position (i.e. column 51). When the cursor moves past the 36th line, Margin 1 is automatically set as the new left margin and the cursor returns to the first line. Margin 1 remains set until the cursor moves past the 36th line again. Similarly, cursor up movement wraps when it passes the 1st line and causes a margin switch, and clearing the screen resets to Margin 0. The advantage of using Margin 1 is that the display has two columns of output. The major disadvantage is that if there are any first-column lines that extend past the 51st column (the center of the screen), they will be overstruck by text in the second column and may become illegible.

View and Hold Submodes

The Alphanumeric screen is displayed at normal intensity. If no data is received from the host or the keyboard for 15 minutes, the screen blanks to prolong the life of the CRT and to prevent raster burn in. This dimming is referred to as the Hold submode, while normal Alphanumeric operation is referred to as View mode. Once in Hold submode, the screen stays blank until data is received from either the host or the keyboard.



Graphic Plot Mode

In Graphic Plot Mode, the Tektronix 4010 emulation addresses a screen of 1024 (x-axis) x 780 (y-axis) points, with the origin (0,0) in the lower left of the screen. These points are addressed by a series of variable-length codes (1 to 4 bytes long) that are generated by the host or the keyboard. Plot information is usually generated from the host source, but can be entered from the keyboard. Figure 4-1 shows the coordinate system used in the Tektronix 4010 emulation.

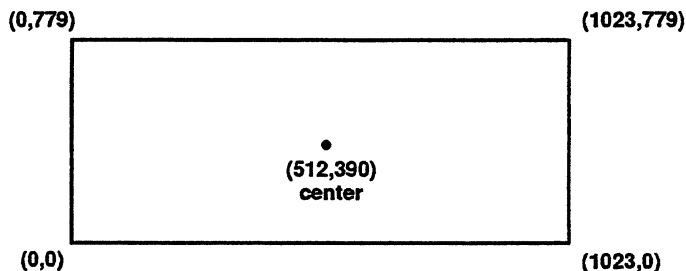


Figure 4-1 The Coordinate System of the Tektronix 4010 Emulation

Using Graphic Plot Mode

To enter Graphics Plot Mode an ASCII <GS> (“Ctrl-]” ASCII, 1D hex) must be sent from either the host or the keyboard. Once the terminal is in the Graphic Plot Mode, a series of coordinate locations are sent to draw the desired lines. Upon entering the Graphics Plot mode, the last entered graphics coordinate is automatically specified as the first point in the first vector. This last coordinate is initialized to (0,0) on power-up or reset. The first vector after a <GS> is always drawn as a dark (invisible) vector. Vectors are drawn from the previously addressed coordinate to the current specified point. Dark vectors can be specified at any time by placing a <GS> before the coordinate address of the terminating point of the specified vector. Plot information is sent in 1 to 4 byte sequences containing the high- and low-order bytes for the x-coordinate and the high- and low-order bytes for the y-coordinates. The next section describes how this information is encoded.

NOTE: In the original Tektronix 4010 terminal, lower-case characters could not be generated, hence a large portion of the coordinates could not be manually entered from the keyboard. The Tektronix 4010 emulation allows these characters to be entered manually from the keyboard.

Coordinate Conversion

Follow the steps in this section to convert (x,y) coordinates to the variable-length code format necessary for the Tektronix 4010 emulation. Remember that if no points have yet been plotted, the “previous” (x,y) referred to is (0,0).

1. **Determine an (x,y) coordinate.** Begin with an (x,y) pair with each ordinate in the range of [0,1023]. The displayable range of the y-ordinate is only [0,779]. If a y-ordinate in the range of [780,1023] is selected, the generated line will be clipped where it crosses the upper edge of the screen.
2. **Convert the (x,y) coordinate.** Convert the (x,y) coordinate to a pair of binary strings, each 10 bits long. Each 10-bit binary string has five Most Significant Bits (MSBs or “high-order bits”) and five Least Significant Bits (LSBs or “low-order bits”).
3. **Set the y high-order bits.** Compare the high-order bits of the current y and the previous y. If they are unequal (changed), add an offset of 20 hex (32 decimal, 40 octal) to the value of this five-bit string to yield a value between 20 hex and 3F hex, and transmit this character to the terminal.
4. **Set the y low-order bits.** Compare the low-order bits of the current y and the previous y and compare the high-order bits of the current x and the previous x. If the current low-order y does not equal the previous low-order y (changed) or if the current high-order x does not equal the previous high-order x (changed), then add an offset of 60 hex (96 decimal, 140 octal) to the value of the y low-order bits to yield a value between 60 hex and 7F hex, and transmit this character to the terminal.
5. **Set the x high-order bits.** Compare the high-order bits of the current x and the previous x. If they are unequal (changed), add an offset of 20 hex (32 decimal, 40 octal) to the value of this 5-bit binary string to yield a value between 20 hex and 3F hex, and transmit this character to the terminal.
6. **Set the x low-order bits.** Add an offset of 40 hex (64 decimal, 100 octal) to the value of the low-order bits of the current x to yield a value between 40 hex and 5F hex, and transmit this character to the terminal. This code signals the terminal to draw the line, thus you must always transmit this character.

Refer to Appendix C for an example of the code that performs this algorithm.



Example of Conversion Process

Let's draw a box around the perimeter of the screen.

From Step 1 — The vertices of this box are (0,0), (0,779), (1023,779), and (1023,0). Put the terminal into Graphic Plot Mode by sending it a GS character (1D hex, 35 octal). Because of the GS character, we are drawing an initial dark (or invisible) vector. This vector should go to (0,0). From Step 2 — Convert (0,0) to a pair of binary strings, each 10-bits long. The result is (0000000000,0000000000). From Step 3 — We want to begin drawing the box at (0,0), so assume that none of the binary strings match the previous strings. Because the strings do not match and the value of the string we just created is 0 hex, the 20 hex offset means we must transmit a <space> character (20 hex). From Step 4 — Because the strings do not match and the value of the string we just created is 0 hex, the 60 hex offset means we must transmit a “ ” character (60 hex). From Step 5 — Because the strings do not match and the value of the string we just created is 0 hex, the 20 hex offset means we must transmit a <space> character (20 hex). From Step 6 — Because the strings do not match and the value of the string we just created is 0 hex, the 40 hex offset means we must transmit an “@” character (40 hex). Thus the first four characters (the first vector) transmitted are: <space> ‘ <space> @ (20 60 20 40 hex).

Now we will draw from (0,0) to (0,779). The binary representation of (0,779) is (0000000000,1100001011). From Step 3, because the high-order bits of the current y (11000) do not equal the high-order bits of the previous y (00000), add 20 hex to 11000 binary to get 38 hex, the ASCII numeral “8”. From Step 4, because the low-order bits of the current y (01011) do not equal the low-order bits of the previous y (00000), add 60 hex to 01011 to get 6B hex. or ASCII “k”. From Step 5, because the x-ordinate does not change (the high-order bits of the current x equals those of the previous x), we do not transmit this character. From Step 6, add 40 hex to the low-order bits of the current x (00000) to get 40 hex, or the ASCII “@”. Thus the second vector (of only three characters) is: 8k@ (38 6B 40 hex).

Now we will draw the line across the top of the screen to (1023,779). From Step 2, in binary this coordinate is (1111111111,1100001011). From Step 3, because the upper-order y does not change, we do not transmit this character. From Step 4, even though the lower-order y did not change, the upper-order x did change, so we must transmit a character, which is the value of the lower-order y plus 60 hex, or an ASCII “k”. From Step 5, because the high-order x changed, we must transmit a character, which is the value of the upper-order x (11111 binary) plus 40 hex, which yields the ASCII “?”. From Step 6, the character transmitted is 5F hex or an ASCII “_”. Thus the third vector is: k?_ (6B 3F 5F hex).

Now we draw the line down the the right side of the screen. The coordinate is (1023,0) or (1111111111,0000000000) in binary. Both high-order and low-order y have changed, so we must send them. X has not changed, but low-order x must be sent anyway to tell the terminal that we are done with this vector. Thus the final vector is: <space>_ (20 60 5F hex).

The last vector returns us to (0,0). Its form is: ‘<space>@ (60 20 40 hex). The final action will be to return to Alphanumeric Mode by sending an ASCII US character (1F hex).



Graphics Input Mode

The Graphics Input Mode is initiated by the host and terminated by the user. The host program has the ability to turn on the graphics cursor in the Tektronix 4010 emulation. Once on, the operator can move the cursor to any point and transmit the coordinates of the point to the host program by pressing any alphanumeric key.

There are three protocols used to receive cursor information back from the 4010 emulation. The first transmits alphanumeric-cursor information and a status byte to the host. The second and third transmit graphics cursor information. This lets user programs interactively manipulate both the alphanumeric and graphics displays on the screen, while constantly updating cursor information.

Alphanumeric Cursor

To receive information on the current coordinates of the alphanumeric cursor on the screen, the host program sends an ESC ENQ (1B 05 hex) sequence to the terminal. In turn, the emulation transmits an 8-bit status byte, the coordinates of the alphanumeric cursor, and the data terminator (if any is selected). The format of the status byte is shown in Table 4-1.

Table 4-1 Format of the Alphanumeric Cursor Status Byte

| Bit | Function | Default | Comment |
|-----|----------|---------|--|
| 7 | none | 0 (Off) | Bits 765 must be 001 respectively to keep the status byte in the range of 20 hex to 3F hex. |
| 6 | none | 0 (Off) | |
| 5 | none | 1 (On) | |
| 4 | HCU | 0 (Off) | Indicates whether hard copy unit (printer) in use.
(0 printer not available/1 printer is available) |
| 3 | none | 0 (Off) | n/a |
| 2 | GRAPH | 0 (Off) | Indicates what mode emulation is in.
(0 for graphic/1 for alphanumeric) |
| 1 | MARGIN | 0 (Off) | Indicates what margin in effect.
(0 for margin 0/1 for margin 1) |
| 0 | none | 0 (Off) | n/a |

After the status byte is transmitted, the coordinates of the alphanumeric cursor are transmitted to the host. These bytes may or may not be followed by terminators, depending upon whether some form of terminator was selected from the Configuration Menu. The format of the cursor coordinates is a 10-bit binary value for the x-ordinate and a 10-bit binary value for the y-ordinate. Each of these 10-bit values is formed from two characters (one for the high-order bits and one for the

low-order bits), with each 5-bit value in the range from 20 hex to 3F hex. The alpha cursor is placed with its lower left-hand corner at the end of the last graphic vector drawn. This position may be anywhere on the screen, but is aligned with the 36x101 grid by the next cursor up or down movement. To position the cursor without drawing, send: <GS> (1D hex) <coordinates> <US> (1F hex).

The alpha cursor may or may not be aligned with the bottom left corner of a character cell. The emulation normally multiplies the current cursor location (row and column) by the size of the character cell (8x8). Multiplying each row and column by eight (the row and column dimensions of a character cell) positions the cursor at the bottom left of the character cell. However, the actual numeric value of this multiplication may be offset, because the cursor may be placed anywhere within the 64 points that define the character cell.

Graphics Cursor

There are two different methods for obtaining graphics cursor information. In the first procedure, the host sends an “<ESC> <SUB>” (1B 1A hex) sequence, waits 20 milliseconds, and then sends an “<ESC> <ENQ>” (1B 05 hex). The 20 millisecond delay is only for compatibility with the original Tektronix 4010, and is not necessary for operation of the Tektronix 4010 emulation. The emulation responds to this sequence by sending the cross-hair coordinates followed by terminators, if any were chosen. These sequences are in the format of <x><y><terminator>. Each <x> and <y> is composed of two characters from 20 hex to 3F hex, each of which represents the 5 high-order bits and 5 low-order bits of each ordinate location. The <terminator> sent depends upon the setting for this option in the Configuration Menu.

In the second procedure, the host computer enables the cross-hair cursor by sending an “<ESC> <SUB>” (1B 1A hex) to the terminal. The operator can then move the cross-hair cursor to a desired point by using the arrow keys or mouse, and strike a keyboard character. The emulation responds by transmitting the keyboard character that was struck, followed by the coordinate information, and any terminators that were selected. The format of the coordinate information and the terminator is identical to those in the paragraph above.

Hard Copy Command

The Tektronix 4010 emulation can produce a screen dump of the current screen to a printer attached to the printer port of the terminal. Pressing the Print key or entering the “ESC <ETB>” (1B 17 hex) control sequence dumps the contents of the screen to the terminal’s serial printer port.

Hot-Key Switch

This command toggles from the active emulation to the currently inactive emulation making it active. This command is only valid during a dual-host session.

| | |
|-------------|------------------|
| <033> <076> | (octal) |
| 1B | (hex) |
| ESC > | (ASCII/keyboard) |

This hot-key switch is used in a diagnostic routine to verify operation of the Dual Host mode.

User Selectable Options

Many of the configuration parameters on the original Tektronix 4010 were implemented through the use of jumpers inside the terminal. Our Tektronix 4010 emulation uses a software menu to configure the terminal. The menu options in the following sections are available to the terminal operator via the Configuration Menu.

Graphic Input Terminators

The data terminators in the Graphic Input Mode can be chosen, from the Configuration Menu, to be either: <CR> only (default); <CR> followed by <EOT>; or no <CR> and no <EOT>.

Line/Local Operation

Tektronix 4010 emulation line and local operation can be toggled by means of the Cmd-On Line key sequence or through the Configuration Menu. Line status is displayed by the On-Line status LED.

Data Communication Baud Rates

Various baud rates can be selected through the Configuration Menu: 150, 300, 600, 1200, 2400, 4800, 9600 and 19200. The terminal transmits only 7-bit codes to the host, but may be set to 8-bits per character for compatibility.



Control Codes

The single control codes used by the Tektronix 4010 emulation are shown in Table 4-2. Some of these single codes are also used in double control sequences (escape sequences), which are shown in Table 4-3.

Table 4-2 Single Control Codes

| ASCII | Decimal | Hex | Keyboard | Comment |
|-------|---------|-----|-------------------|-----------------------------------|
| BEL | 7 | 07 | Ctrl-G | Tone from Speaker |
| BS | 8 | 08 | Ctrl-H | Backspace |
| HT | 9 | 09 | Ctrl-I | Cursor Right |
| LF | 10 | 0A | Ctrl-J | Line Feed (move down one line) |
| VT | 11 | 0B | Ctrl-K | Move Up One Line |
| CR | 13 | 0D | Ctrl-M | Carriage Return |
| ESC | 27 | 1B | Ctrl-[or ESC key | For ESC Sequences (see Table 4-3) |
| GS | 29 | 1D | Ctrl-] | Enter Graphic Plot Mode |
| US | 31 | 1F | Ctrl-_ | Enter Alphanumeric Mode |

Table 4-3 Double Control Codes (Escape Sequences)



| Sequence | Decimal | Hex | Keyboard | Comment |
|----------|---------|------|------------|---|
| ESC ENQ | 27 5 | 1B05 | ESC Ctrl-E | Terminal Returns Status Byte and Alphanumeric Cursor Coordinates. |
| ESC SUB | 27 26 | 1B1A | ESC Ctrl-Z | Starts Cross-Hair Cursor
a. Followed by 20 ms delay then ESC ENQ, terminal returns position of the cross-hair cursor.
b. Followed by movement of cursor and striking of any key on keyboard, terminal returns the character struck, then the cross-hair position. |
| ESC ETB | 27 23 | 1B17 | ESC Ctrl-W | Send Screen Dump to Printer |
| ESC FF | 27 12 | 1B0C | ESC Ctrl-L | Erase Screen, Return to Alphanumeric Mode, and Home Cursor. |

End of Chapter

Chapter 5

PCTERM Operations

This chapter provides programming information for the PCTERM operating mode of the VT320/100 emulation running on the D217/D413/D463 line of Data General terminals. This chapter has the major sections listed below:

Introduction

Inbound (Terminal to Host) Codes

Outbound (Host to Terminal) Codes

VP/ix getty Setup

Sample terminfo File

Sample VP/ix term File

Information regarding functions and operations of the terminal that apply to all modes or emulations is covered in Chapter 1. Additional information on keyboard layouts and reference material, which may apply to the PCTERM operating mode is covered in appendices, located at the rear of this manual.

Introduction

PCTERM lets terminals emulate an IBM PC environment that can be connected to different types of systems. For example, PCTERM can be used with a VP/ix system running under INTERACTIVE UNIX.

PCTERM is accessed through either the VT320 or VT100 emulation. For more details on PCTERM as a VT320/100 operating mode, refer to the "PCTERM Mode" section in Chapter 3.

Inbound (Terminal to Host) Codes

Two sets of inbound codes were altered for the operation of PCTERM: flow control codes and keyboard generated codes.

Flow Control

Under normal operations, the flow control codes XOFF and XON are generated by Ctrl-S and Ctrl-Q (13 hex and 11 hex). However, in PCTERM, XOFF and XON are generated by the ASCII characters "g" (hex 67) and "e" (hex 65). Regardless of this change, flow control operations within PCTERM are identical to those of other terminal modes and emulations. For more information on terminal flow-control operations, see the "Flow Control" section in Chapter 1.

Keyboard Generated Codes

All terminals support two keyboards: a 101-key keyboard, similar to that of the IBM PC AT-style, and a 107-key Data General proprietary keyboard. With the exception of two keys, the Data General proprietary keyboard is fully mapped as an enhanced XT-style keyboard. The rightmost Alt key and the rightmost Ctrl key were omitted from the Data General proprietary keyboard.

Since a terminal operating in the PCTERM mode must send codes for almost every key, there are a limited number of local-functions available on the terminal while in this mode. However, the local functions Cursor Type, Cmd-Alpha Lock (hot-key switch between emulations), and Cmd-N/C (Configuration Menu) are supported and do not send codes to the host.

When using PCTERM, you will find it handy to refer to the red legends on the front of certain keycaps on the Data General proprietary keyboard; these legends are correct with the exception of the "+" sign in the numeric keypad. Even though they have no red legend, keys F14 and F15 substitute as numeric keypad keys "/" and "*" respectively.

In PCTERM mode, the state of Caps Lock, Scroll Lock, and Num Lock are correctly displayed on the appropriately titled LEDs. Table 5-1 through Table 5-5 show the keyboard codes generated.

Table 5-1 Keyboard Generated Codes — Function Keys (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------|----------------------|--------|
| | | Key Down | Key Up |
| F1 | F1 | 3B | BB |
| F2 | F2 | 3C | BC |
| F3 | F3 | 3D | BD |
| F4 | F4 | 3E | BE |
| F5 | F5 | 3F | BF |
| F6 | F6 | 40 | C0 |
| F7 | F7 | 41 | C1 |
| F8 | F8 | 42 | C2 |
| F9 | F9 | 43 | C3 |
| F10 | F10 | 44 | C4 |
| F11 | F11 | 57 | D7 |
| F12 | F12 | 58 | D8 |
| F13 | n/a | none | none |
| F14 | / (num kypd) | E0 35 | E0 B5 |
| F15 | * (num kypd) | 37 | B7 |
| Cursor Type (F16) | N/A | none | none |
| N/C (F17) | N/A | none | none |
| Local Print (F18) | Scroll Lock | 46 | C6 |
| Scroll Rate (F19) | Num Lock | 45 | C5 |
| Hold (F20) | Pause | E1 1D 45
E1 9D C5 | none |
| N/A | Ctrl-Pause (Break) | E0 46 | E0 C6 |

Table 5-2 Keyboard Generated Codes — Numeric Keypad (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------|--------------------|--------|
| | | Key Down | Key Up |
| 7 | 7 | 47 | C7 |
| 8 | 8 | 48 | C8 |
| 9 | 9 | 49 | C9 |
| - | - | 4A | CA |
| 4 | 4 | 4B | CB |
| 5 | 5 | 4C | CC |
| 6 | 6 | 4D | CD |
| , | + | 4E | CE |
| 1 | 1 | 4F | CF |
| 2 | 2 | 50 | D0 |
| 3 | 3 | 51 | D1 |
| 0 | 0 | 52 | D2 |
| . | . | 53 | D3 |
| New Line | Enter | E0 1C | E0 9C |

Table 5-3 Keyboard Generated Codes — Editing Keypad with Num Lock Off (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------------|--------------------|-------------|
| | | Key Down | Key Up |
| Erase Page | Insert | E0 52 | E0 D2 |
| Print | Print Screen | E0 2A E0 37 | E0 B7 E0 AA |
| Alt-Print | Alt-Print Screen (Sysreq) | 54 | D4 |
| Erase EOL | Delete | E0 53 | E0 D3 |
| C1 | Home | E0 47 | E0 C7 |
| (uparrow) | (uparrow) | E0 48 | E0 C8 |
| C2 | Page Up | E0 49 | E0 C9 |
| (leftarrow) | (leftarrow) | E0 4B | E0 CB |
| Home | n/a | none | none |
| (rightarrow) | (rightarrow) | E0 4D | E0 CD |
| C3 | End | E0 4F | E0 CF |
| (downarrow) | (downarrow) | E0 50 | E0 D0 |
| C4 | Page Down | E0 51 | E0 D1 |

Table 5-4 Keyboard Generated Codes — Editing Keypad with Num Lock On (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------------|--------------------|-------------|
| | | Key Down | Key Up |
| Erase Page | Insert | E0 2A E0 52 | E0 D2 E0 AA |
| Print | Print Screen | E0 2A E0 37 | E0 B7 E0 AA |
| Alt-Print | Alt-Print Screen (Sysreq) | 54 | D4 |
| Erase EOL | Delete | E0 2A E0 53 | E0 D3 E0 AA |
| C1 | Home | E0 2A E0 47 | E0 C7 E0 AA |
| (uparrow) | (uparrow) | E0 2A E0 48 | E0 C8 E0 AA |
| C2 | Page Up | E0 2A E0 49 | E0 C9 E0 AA |
| (leftarrow) | (leftarrow) | E0 2A E0 4B | E0 CB E0 AA |
| Home | n/a | none | none |
| (rightarrow) | (rightarrow) | E0 2A E0 4D | E0 CD E0 AA |
| C3 | End | E0 2A E0 4F | E0 CF E0 AA |
| (downarrow) | (downarrow) | E0 2A E0 50 | E0 D0 E0 AA |
| C4 | Page Down | E0 2A E0 51 | E0 D1 E0 AA |

Table 5-5 Keyboard Generated Codes — Main Keypad (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------|--------------------|--------|
| | | Key Down | Key Up |
| ESC | ESC | 01 | 81 |
| 1! | 1! | 02 | 82 |
| 2@ | 2@ | 03 | 83 |
| 3# | 3# | 04 | 84 |
| 4\$ | 4\$ | 05 | 85 |
| 5% | 5% | 06 | 86 |
| 6^ | 6^ | 07 | 87 |
| 7& | 7& | 08 | 88 |
| 8* | 8* | 09 | 89 |
| 9(| 9(| 0A | 8A |
| 0) | 0) | 0B | 8B |
| -_ | -_ | 0C | 8C |
| =+ | =+ | 0D | 8D |
| '~ | '~ | 29 | A9 |
| \ | \ | 2B | AB |
| Del | Backspace | 0E | 8E |
| Tab | Tab | 0F | 8F |
| Q | Q | 10 | 90 |
| W | W | 11 | 91 |
| E | E | 12 | 92 |
| R | R | 13 | 93 |
| T | T | 14 | 94 |
| Y | Y | 15 | 95 |
| U | U | 16 | 96 |
| I | I | 17 | 97 |
| O | O | 18 | 98 |
| P | P | 19 | 99 |
| [| [| 1A | 9A |
|] |] | 1B | 9B |
| Spcl | Spcl | none | none |
| CR | Enter | 1C | 9C |
| Ctrl | Ctrl | 1D | 9D |
| A | A | 1E | 9E |
| S | S | 1F | 9F |
| D | D | 20 | A0 |
| F | F | 21 | A1 |
| G | G | 22 | A2 |
| H | H | 23 | A3 |
| J | J | 24 | A4 |
| K | K | 25 | A5 |
| L | L | 26 | A6 |
| :: | :: | 27 | A7 |
| ;" | ;" | 28 | A8 |

continued

Table 5-5 Keyboard Generated Codes — Main Keypad, continued (PCTERM)

| 107-key
Keyboard | 101-key
Keyboard | Scan Code (in hex) | |
|---------------------|---------------------|--------------------|--------|
| | | Key Down | Key Up |
| New Line | Enter | 1C | 9C |
| Rept | n/a | none | none |
| Left Shift | Left Shift | 2A | AA |
| Z | Z | 2C | AC |
| X | X | 2D | AD |
| C | C | 2E | AE |
| V | V | 2F | AF |
| B | B | 30 | B0 |
| N | N | 31 | B1 |
| M | M | 32 | B2 |
| , < | , < | 33 | B3 |
| . > | . > | 34 | B4 |
| / ? | / ? | 35 | B5 |
| Right Shift | Right Shift | 36 | B6 |
| Cmd | n/a | none | none |
| On Line | Right Alt | 38 | B8 |
| Space | Space | 39 | B9 |
| Alpha Lock | Caps | 3A | BA |

Outbound (Host to Terminal) Codes

Command codes in PCTERM are largely identical to those of the VT320/100 emulation. The only changes are in the areas of cursor addressing and character sets.

Cursor Addressing

The 25th row status-line is not used on PCTERM. Thus, the 25th row is displayed as part of the active screen area, and all cursor-positioning commands may access it.

Character Sets

The GL and GR character sets are set to IBM PC characters (PCTERM Low and PCTERM High, respectively). Each of these character sets has 128 characters. The codes 00 to 1F hex and 80 to 9F hex may be accessed by prefixing the character to be displayed with 12 hex (Ctrl-R, from Force Display command). Shift In and Shift Out may still be used to access other characters as desired. The keyboard on the terminal will not operate in 7-bit mode because PCTERM is strictly an 8-bit operating environment due to the use of 8-bit scan codes on the keyboard.

NOTE: The D217 terminal does not have a GR character set so the application program will have to use Shift in and Shift out to appropriately set GL.

VP/ix getty Setup

This entry should go into `/etc/gettydefs`. It defines a reasonably useful communications setup for a VP/ix user.

```
vpixtty# B9600 HUPCL CS8 CREAD ICANON ECHO ECHOE ECHOK ECHONL OPOST ICRNL  
#B9600 CREAD CLOCAL ICRNL OPOST ICANON ECHO ECHOE ECHOK ECHONL CS8 IXON IXOFF  
ONLCR ISIG HUPCL #System-name login: #vpixtty
```

This should be entered all on one line and the case is significant. An easy way to do this is to type the following command:

```
cat >> /etc/gettydefs
```

Then enter the getty definition and press Newline twice. Then press Ctrl-D to finish this update.

Sample terminfo File

This terminfo source file should be compiled with the `tic` utility. This description gives the UNIX terminal drivers access to the VT320/100 emulation for operations outside of VP/ix, as well as providing background information inside VP/ix.

```
dg_pcterm|DG PC Terminal,  
use=vt100,
```

Sample VP/ix term File

This following lines should go in `/usr/vpix/term/dg_pcterm` to describe a Data General terminal running PCTERM for use within the VP/ix environment.

```
* @(#)dg_pcterm 1.0 -89/10/09
* Part I:  output definitions
output: \001 \022-\001 * smiling face, black-on-white
output: \002 \022-\002 * smiling face, white-on-black
output: \003 \022-\003 * heart
output: \004 \022-\004 * diamond
output: \005 \022-\005 * club
output: \006 \022-\006 * spade
output: \007 \022-\007 * centered dot, black-on-white
output: \010 \022-\010 * centered dot, white-on-black
output: \011 \022-\011 * circle, black-on-white
output: \012 \022-\012 * circle, white-on-black
output: \013 \022-\013 * male symbol
output: \014 \022-\014 * female symbol
output: \015 \022-\015 * musical note
output: \016 \022-\016 * musical double-note
output: \017 \022-\017 * sun
output: \020 \022-\020 * right-hand turn signal
output: \021 \022-\021 * left-hand turn signal
output: \022 \022-\022 * uparrow and downarrow
output: \023 \022-\023 * double-exclamation mark
output: \024 \022-\024 * paragraph mark
output: \025 \022-\025 * section mark
output: \026 \022-\026 * horizontal black rectangle
output: \027 \022-\027 * underlined uparrow and downarrow
output: \030 \022-\030 * uparrow
output: \031 \022-\031 * downarrow
output: \032 \022-\032 * rightrightarrow
output: \033 \022-\033 * leftarrow
output: \034 \022-\034 * centered lower-left of box
output: \035 \022-\035 * rightrightarrow and leftarrow
output: \036 \022-\036 * centered black up triangle
output: \037 \022-\037 * centered black down triangle
output: \177 \022-\177 * large centered empty triangle
output: \200 \022-\200 * C with cedilla
output: \201 \022-\201 * u with umlaut
output: \202 \022-\202 * e with accent-egu
output: \203 \022-\203 * a with circumflex
```



```

output: \204 \022-\204 * a with umlaut
output: \205 \022-\205 * a with accent-grave
output: \206 \022-\206 * a with little circle
output: \207 \022-\207 * c with cedilla
output: \210 \022-\210 * e with circumflex
output: \211 \022-\211 * e with umlaut
output: \212 \022-\212 * e with accent-grave
output: \213 \022-\213 * i with umlaut
output: \214 \022-\214 * i with circumflex
output: \215 \022-\215 * i with accent-grave
output: \216 \022-\216 * A with umlaut
output: \217 \022-\217 * A with little circle
output: \220 \022-\220 * E with accent-egu
output: \221 \022-\221 * joined ae
output: \222 \022-\222 * joined AE
output: \223 \022-\223 * o with circumflex
output: \224 \022-\224 * o with umlaut
output: \225 \022-\225 * o with accent-grave
output: \226 \022-\226 * u with circumflex
output: \227 \022-\227 * u with accent-grave
output: \230 \022-\230 * y with umlaut
output: \231 \022-\231 * O with umlaut
output: \232 \022-\232 * U with umlaut
output: \233 \022-\233 * cent symbol
output: \234 \022-\234 * English pound symbol
output: \235 \022-\235
output: \236 \022-\236
output: \237 \022-\237

```

[NOTE: These definitions can be used for all terminals.]

* Part II: terminfo-type definitions

```

clear:      \E[H\E[2J
csr:        \E[%i%p1%d;%p2%dr
ri:         \EM
svpix:      \E[?99h
fvpix:      \E[?99l
c_xon:      e
c_xoff:     g
pccompat:   y
ind:        \ED
cup:        \E[%i%p1%d;%p2%dH
cubl:       \b
cuf1:       \E[C
cuul:       \E[A

```

cucl: \n
ed: \E[J
el: \E[K

[NOTE: Use these definitions for D413/D463 only.]

* Part II: terminfo-type definitions

clear: \E[H\E[2J
ri: \EM
svpix: \E[?99h
fvpix: \E[?99l
c_xon: e
c_xoff: g
pccompat: y
ind: \ED
ill: \E[L
dll: \E[M
ichl: \E[@
dchl: \E[P
cup: \E[%i%p1%d;%p2%dH
cubl: \b
cuf1: \E[C
cuul: \E[A
cucl: \n
ed: \E[J
el: \E[K

* Part III: attribute definitions

sgr0: \E[0m
blink: \E[0;5m
bold: \E[0;1m
rev: \E[0;7m
smul: \E[0;4m
blbo: \E[0;1;5m
blr: \E[0;5;7m
blu: \E[0;4;5m
bor: \E[0;1;7m
bou: \E[0;1;4m
blbor: \E[0;1;5;7m
blbou: \E[0;1;4;5m

If you are using the `/bin/sh` (Bourne) shell, place the following lines in `/usr/"$LOGNAME"/.profile` to specify that a particular user will use this terminal type:

```
TERM=dg_pcterm
export TERM
```

If you are using the `/bin/csh` (C) shell, then use the following in the `/usr/$home/.login` file:

```
set term=dg_pcterm
```

To finish the setup for a VP/ix user, use the `addvpixuser` option in the `packagemgmt` selection in `sysadm`. Then go to the `ttymgmt` menu and select `modtty`. Use this function to set the user's login Line Setting to `vpixtty`.

End of Chapter



Appendix A

Character Sets

This appendix contains tables of each character set supported by the D217+/D413/D463:

- United States ASCII
- NRC United Kingdom
- NRC French
- NRC German
- NRC Swedish/Finnish
- NRC Spanish
- NRC Danish/Norwegian
- NRC Swiss
- NRC Katakana (G0 Set)
- Katakana (G1 Set)
- DG International
- Word-Processing, Greek, and Math Set
- DG Line Drawing
- DG Special Graphics (PC Characters)
- VT Multinational
- VT Special Graphics (VT Line Drawing)
- ISO 8859/1.2 Characters
- PCTERM Low Characters (0 hex through 7F hex)
- PCTERM High Characters (80 hex through FF hex)

Each of the tables displayed in this appendix show the character displayed on the screen and the decimal, octal, and hex value associated with that character. If no character is recorded within a space in a table, that code will produce a blank on the screen.

United States ASCII Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | @ | 64
100
40 | P | 80
120
50 | · | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | ' | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | [| 91
133
5B | k | 107
153
6B | { | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | \ | 92
134
5C | l | 108
154
6C | | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D |] | 93
135
5D | m | 109
155
6D | } | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ^ | 94
136
5E | n | 110
156
6E | ~ | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | _ | 95
137
5F | o | 111
157
6F | █ | 127
177
7F |

| | | | |
|------------------------|---|-----------|-------------------------|
| Displayed
Character | E | 27 | Decimal
Octal
Hex |
| | C | 033
1B | |

NRC United Kingdom Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | @ | 64
100
40 | P | 80
120
50 | ' | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | £ | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | , | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | [| 91
133
5B | k | 107
153
6B | { | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | \ | 92
134
5C | l | 108
154
6C | | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D |] | 93
135
5D | m | 109
155
6D | } | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ↑ | 94
136
5E | n | 110
156
6E | ~ | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | - | 95
137
5F | o | 111
157
6F | ▒ | 127
177
7F |

Displayed Character

| | |
|---|-----|
| E | 27 |
| C | 033 |
| | 1B |

 Decimal Octal Hex

NRC French Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | à | 64
100
40 | P | 80
120
50 | · | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | £ | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | ' | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | ° | 91
133
5B | k | 107
153
6B | é | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | ç | 92
134
5C | l | 108
154
6C | ù | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | § | 93
135
5D | m | 109
155
6D | è | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ^ | 94
136
5E | n | 110
156
6E | ~ | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | _ | 95
137
5F | o | 111
157
6F | ☐ | 127
177
7F |

Displayed Character

| | |
|---|---|
| E | C |
|---|---|

| | | |
|----|-----|----|
| 27 | 033 | 1B |
|----|-----|----|

 Decimal
Octal
Hex

NRC German Character Set

| | | | | | | | | | | | | | | | | |
|---|---|-----|----------------|-----|----|-----|---|-----|---|-----|---|-----|---|-----|---|-----|
| N | U | 0 | DL | 16 | | 32 | 0 | 48 | S | 64 | P | 80 | . | 96 | p | 112 |
| | | 000 | | 020 | | 040 | 0 | 060 | | 100 | | 120 | | 140 | | 160 |
| | | 00 | | 10 | | 20 | | 30 | | 40 | | 50 | | 60 | | 70 |
| S | H | 1 | D ₁ | 17 | ! | 33 | 1 | 49 | A | 65 | Q | 81 | a | 97 | q | 113 |
| | | 001 | | 021 | | 041 | | 061 | | 101 | | 121 | | 141 | | 161 |
| | | 01 | | 11 | | 21 | | 31 | | 41 | | 51 | | 61 | | 71 |
| S | X | 2 | D ₂ | 18 | " | 34 | 2 | 50 | B | 66 | R | 82 | b | 98 | r | 114 |
| | | 002 | | 022 | | 042 | | 062 | | 102 | | 122 | | 142 | | 162 |
| | | 02 | | 12 | | 22 | | 32 | | 42 | | 52 | | 62 | | 72 |
| E | X | 3 | D ₃ | 19 | # | 35 | 3 | 51 | C | 67 | S | 83 | c | 99 | s | 115 |
| | | 003 | | 023 | | 043 | | 063 | | 103 | | 123 | | 143 | | 163 |
| | | 03 | | 13 | | 23 | | 33 | | 43 | | 53 | | 63 | | 73 |
| E | T | 4 | D ₄ | 20 | \$ | 36 | 4 | 52 | D | 68 | T | 84 | d | 100 | t | 116 |
| | | 004 | | 024 | | 044 | | 064 | | 104 | | 124 | | 144 | | 164 |
| | | 04 | | 14 | | 24 | | 34 | | 44 | | 54 | | 64 | | 74 |
| E | Q | 5 | NK | 21 | % | 37 | 5 | 53 | E | 69 | U | 85 | e | 101 | u | 117 |
| | | 005 | | 025 | | 045 | | 065 | | 105 | | 125 | | 145 | | 165 |
| | | 05 | | 15 | | 25 | | 35 | | 45 | | 55 | | 65 | | 75 |
| A | K | 6 | S _Y | 22 | & | 38 | 6 | 54 | F | 70 | V | 86 | f | 102 | v | 118 |
| | | 006 | | 026 | | 046 | | 066 | | 106 | | 126 | | 146 | | 166 |
| | | 06 | | 16 | | 26 | | 36 | | 46 | | 56 | | 66 | | 76 |
| B | L | 7 | E _B | 23 | , | 39 | 7 | 55 | G | 71 | W | 87 | g | 103 | w | 119 |
| | | 007 | | 027 | | 047 | | 067 | | 107 | | 127 | | 147 | | 167 |
| | | 07 | | 17 | | 27 | | 37 | | 47 | | 57 | | 67 | | 77 |
| B | S | 8 | C _N | 24 | (| 40 | 8 | 56 | H | 72 | X | 88 | h | 104 | x | 120 |
| | | 010 | | 030 | | 050 | | 070 | | 110 | | 130 | | 150 | | 170 |
| | | 08 | | 18 | | 28 | | 38 | | 48 | | 58 | | 68 | | 78 |
| H | T | 9 | E _M | 25 |) | 41 | 9 | 57 | I | 73 | Y | 89 | i | 105 | y | 121 |
| | | 011 | | 031 | | 051 | | 071 | | 111 | | 131 | | 151 | | 171 |
| | | 09 | | 19 | | 29 | | 39 | | 49 | | 59 | | 69 | | 79 |
| L | F | 10 | S _B | 26 | * | 42 | : | 58 | J | 74 | Z | 90 | j | 106 | z | 122 |
| | | 012 | | 032 | | 052 | | 072 | | 112 | | 132 | | 152 | | 172 |
| | | 0A | | 1A | | 2A | | 3A | | 4A | | 5A | | 6A | | 7A |
| V | T | 11 | E _C | 27 | + | 43 | ; | 59 | K | 75 | Ä | 91 | k | 107 | ä | 123 |
| | | 013 | | 033 | | 053 | | 073 | | 113 | | 133 | | 153 | | 173 |
| | | 0B | | 1B | | 2B | | 3B | | 4B | | 5B | | 6B | | 7B |
| F | F | 12 | F _S | 28 | , | 44 | < | 60 | L | 76 | Ö | 92 | l | 108 | ö | 124 |
| | | 014 | | 034 | | 054 | | 074 | | 114 | | 134 | | 154 | | 174 |
| | | 0C | | 1C | | 2C | | 3C | | 4C | | 5C | | 6C | | 7C |
| C | R | 13 | G _S | 29 | - | 45 | = | 61 | M | 77 | Ü | 93 | m | 109 | ü | 125 |
| | | 015 | | 035 | | 055 | | 075 | | 115 | | 135 | | 155 | | 175 |
| | | 0D | | 1D | | 2D | | 3D | | 4D | | 5D | | 6D | | 7D |
| S | O | 14 | R _S | 30 | . | 46 | > | 62 | N | 78 | ^ | 94 | n | 110 | ß | 126 |
| | | 016 | | 036 | | 056 | | 076 | | 116 | | 136 | | 156 | | 176 |
| | | 0E | | 1E | | 2E | | 3E | | 4E | | 5E | | 6E | | 7E |
| S | I | 15 | U _S | 31 | / | 47 | ? | 63 | O | 79 | - | 95 | o | 111 | ■ | 127 |
| | | 017 | | 037 | | 057 | | 077 | | 117 | | 137 | | 157 | | 177 |
| | | 0F | | 1F | | 2F | | 3F | | 4F | | 5F | | 6F | | 7F |

Displayed Character

| | |
|---|-----|
| E | 27 |
| C | 033 |
| | 1B |

 Decimal Octal Hex

NRC Swedish/Finnish Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|---|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | É | 64
100
40 | P | 80
120
50 | é | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | ” | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | Å | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | , | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | Ä | 91
133
5B | k | 107
153
6B | ä | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | Ö | 92
134
5C | l | 108
154
6C | ö | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | Å | 93
135
5D | m | 109
155
6D | å | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | Ü | 94
136
5E | n | 110
156
6E | ü | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | - | 95
137
5F | o | 111
157
6F | o | 127
177
7F |

| | | | |
|------------------------|---|-----------|-------------------------|
| Displayed
Character | E | 27 | Decimal
Octal
Hex |
| | C | 033
1B | |

NRC Spanish Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | @ | 64
100
40 | P | 80
120
50 | · | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | ' | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | I | 91
133
5B | k | 107
153
6B | { | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | Ñ | 92
134
5C | l | 108
154
6C | ñ | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | l | 93
135
5D | m | 109
155
6D | } | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ^ | 94
136
5E | n | 110
156
6E | ~ | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | - | 95
137
5F | o | 111
157
6F | ☐ | 127
177
7F |

| | | | |
|------------------------|---|-----------|-------------------------|
| Displayed
Character | E | 27 | Decimal
Octal
Hex |
| | C | 033
1B | |

NRC Danish/Norwegian Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|---|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | Ä | 64
100
40 | P | 80
120
50 | ä | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | œ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | ' | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
010
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
011
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
012
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
013
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | Æ | 91
133
5B | k | 107
153
6B | æ | 123
173
7B |
| F _F | 12
014
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | Ø | 92
134
5C | l | 108
154
6C | ø | 124
174
7C |
| C _R | 13
015
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | Å | 93
135
5D | m | 109
155
6D | å | 125
175
7D |
| S _O | 14
016
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | Ü | 94
136
5E | n | 110
156
6E | ü | 126
176
7E |
| S _I | 15
017
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | - | 95
137
5F | o | 111
157
6F | ◻ | 127
177
7F |

Displayed Character

| | |
|---|-----|
| E | 27 |
| C | 033 |
| | 1B |

 Decimal
 Octal
 Hex

NRC Swiss Character Set

| | | | | | | | | | | | | | | | |
|----------------|-----------------|----------------|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| N _U | 0
000
00 | D _L | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | à | 64
100
40 | P | 80
120
50 | ô | 96
140
60 | p | 112
160
70 |
| S _H | 1
001
01 | D ₁ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| S _X | 2
002
02 | D ₂ | 18
022
12 | ” | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| E _X | 3
003
03 | D ₃ | 19
023
13 | ù | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| E _T | 4
004
04 | D ₄ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| E _Q | 5
005
05 | N _K | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| A _K | 6
006
06 | S _Y | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| B _L | 7
007
07 | E _B | 23
027
17 | , | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| B _S | 8
008
08 | C _N | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| H _T | 9
009
09 | E _M | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| L _F | 10
010
0A | S _B | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| V _T | 11
011
0B | E _C | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | ë | 91
133
5B | k | 107
153
6B | ä | 123
173
7B |
| F _F | 12
012
0C | F _S | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | ç | 92
134
5C | l | 108
154
6C | ö | 124
174
7C |
| C _R | 13
013
0D | G _S | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | è | 93
135
5D | m | 109
155
6D | ü | 125
175
7D |
| S _O | 14
014
0E | R _S | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ê | 94
136
5E | n | 110
156
6E | é | 126
176
7E |
| S _I | 15
015
0F | U _S | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | - | 95
137
5F | o | 111
157
6F | ■ | 127
177
7F |

| | | | |
|------------------------|---|-----------|-------------------------|
| Displayed
Character | E | 27 | Decimal
Octal
Hex |
| | C | 033
1B | |

NRC Katakana (G0) Character Set

| | | | | | | | | | | | | | | | |
|----------------|----------------------------|----------------|----------------------------|----|----------------------------|---|----------------------------|---|----------------------------|---|----------------------------|---|-----------------------------|---|-----------------------------|
| N _U | ⁰
000
00 | D _L | ¹⁶
020
10 | | ³²
040
20 | 0 | ⁴⁸
060
30 | @ | ⁶⁴
100
40 | P | ⁸⁰
120
50 | · | ⁹⁶
140
60 | p | ¹¹²
160
70 |
| S _H | ¹
001
01 | D ₁ | ¹⁷
021
11 | ! | ³³
041
21 | 1 | ⁴⁹
061
31 | A | ⁶⁵
101
41 | Q | ⁸¹
121
51 | a | ⁹⁷
141
61 | q | ¹¹³
161
71 |
| S _X | ²
002
02 | D ₂ | ¹⁸
022
12 | ” | ³⁴
042
22 | 2 | ⁵⁰
062
32 | B | ⁶⁶
102
42 | R | ⁸²
122
52 | b | ⁹⁸
142
62 | r | ¹¹⁴
162
72 |
| E _X | ³
003
03 | D ₃ | ¹⁹
023
13 | # | ³⁵
043
23 | 3 | ⁵¹
063
33 | C | ⁶⁷
103
43 | S | ⁸³
123
53 | c | ⁹⁹
143
63 | s | ¹¹⁵
163
73 |
| E _T | ⁴
004
04 | D ₄ | ²⁰
024
14 | \$ | ³⁶
044
24 | 4 | ⁵²
064
34 | D | ⁶⁸
104
44 | T | ⁸⁴
124
54 | d | ¹⁰⁰
144
64 | t | ¹¹⁶
164
74 |
| E _Q | ⁵
005
05 | N _K | ²¹
025
15 | % | ³⁷
045
25 | 5 | ⁵³
065
35 | E | ⁶⁹
105
45 | U | ⁸⁵
125
55 | e | ¹⁰¹
145
65 | u | ¹¹⁷
165
75 |
| A _K | ⁶
006
06 | S _Y | ²²
026
16 | & | ³⁸
046
26 | 6 | ⁵⁴
066
36 | F | ⁷⁰
106
46 | V | ⁸⁶
126
56 | f | ¹⁰²
146
66 | v | ¹¹⁸
166
76 |
| B _L | ⁷
007
07 | E _B | ²³
027
17 | , | ³⁹
047
27 | 7 | ⁵⁵
067
37 | G | ⁷¹
107
47 | W | ⁸⁷
127
57 | g | ¹⁰³
147
67 | w | ¹¹⁹
167
77 |
| B _S | ⁸
010
08 | C _N | ²⁴
030
18 | (| ⁴⁰
050
28 | 8 | ⁵⁶
070
38 | H | ⁷²
110
48 | X | ⁸⁸
130
58 | h | ¹⁰⁴
150
68 | x | ¹²⁰
170
78 |
| H _T | ⁹
011
09 | E _M | ²⁵
031
19 |) | ⁴¹
051
29 | 9 | ⁵⁷
071
39 | I | ⁷³
111
49 | Y | ⁸⁹
131
59 | i | ¹⁰⁵
151
69 | y | ¹²¹
171
79 |
| L _F | ¹⁰
012
0A | S _B | ²⁶
032
1A | * | ⁴²
052
2A | : | ⁵⁸
072
3A | J | ⁷⁴
112
4A | Z | ⁹⁰
132
5A | j | ¹⁰⁶
152
6A | z | ¹²²
172
7A |
| V _T | ¹¹
013
0B | E _C | ²⁷
033
1B | + | ⁴³
053
2B | ; | ⁵⁹
073
3B | K | ⁷⁵
113
4B | [| ⁹¹
133
5B | k | ¹⁰⁷
153
6B | { | ¹²³
173
7B |
| F _F | ¹²
014
0C | F _S | ²⁸
034
1C | , | ⁴⁴
054
2C | < | ⁶⁰
074
3C | L | ⁷⁶
114
4C | ¥ | ⁹²
134
5C | l | ¹⁰⁸
154
6C | | ¹²⁴
174
7C |
| C _R | ¹³
015
0D | G _S | ²⁹
035
1D | - | ⁴⁵
055
2D | = | ⁶¹
075
3D | M | ⁷⁷
115
4D |] | ⁹³
135
5D | m | ¹⁰⁹
155
6D | } | ¹²⁵
175
7D |
| S _O | ¹⁴
016
0E | R _S | ³⁰
036
1E | . | ⁴⁶
056
2E | > | ⁶²
076
3E | N | ⁷⁸
116
4E | ^ | ⁹⁴
136
5E | n | ¹¹⁰
156
6E | ~ | ¹²⁶
176
7E |
| S _I | ¹⁵
017
0F | U _S | ³¹
037
1F | / | ⁴⁷
057
2F | ? | ⁶³
077
3F | O | ⁷⁹
117
4F | _ | ⁹⁵
137
5F | o | ¹¹¹
157
6F | ▒ | ¹²⁷
177
7F |

| | | | |
|------------------------|---|-----------|-------------------------|
| Displayed
Character | E | 27 | Decimal
Octal
Hex |
| | C | 033
1B | |

Katakana (G1) Character Set

| | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|-----|----|----|-----|----|----|-----|----|---|----|-----|----|---|----|-----|----|---|----|-----|----|-----|-----|----|-----|-----|----|
| 0 | 000 | 00 | 16 | 020 | 10 | 32 | 040 | 20 | ー | 48 | 060 | 30 | 夕 | 64 | 100 | 40 | 三 | 80 | 120 | 50 | 96 | 140 | 60 | 112 | 160 | 70 |
| 1 | 001 | 01 | 17 | 021 | 11 | 33 | 041 | 21 | ア | 49 | 061 | 31 | チ | 65 | 101 | 41 | 厶 | 81 | 121 | 51 | 97 | 141 | 61 | 113 | 161 | 71 |
| 2 | 002 | 02 | 18 | 022 | 12 | 34 | 042 | 22 | イ | 50 | 062 | 32 | ツ | 66 | 102 | 42 | メ | 82 | 122 | 52 | 98 | 142 | 62 | 114 | 162 | 72 |
| 3 | 003 | 03 | 19 | 023 | 13 | 35 | 043 | 23 | ウ | 51 | 063 | 33 | テ | 67 | 103 | 43 | モ | 83 | 123 | 53 | 99 | 143 | 63 | 115 | 163 | 73 |
| 4 | 004 | 04 | 20 | 024 | 14 | 36 | 044 | 24 | エ | 52 | 064 | 34 | ト | 68 | 104 | 44 | パ | 84 | 124 | 54 | 100 | 144 | 64 | 116 | 164 | 74 |
| 5 | 005 | 05 | 21 | 025 | 15 | 37 | 045 | 25 | オ | 53 | 065 | 35 | ナ | 69 | 105 | 45 | ユ | 85 | 125 | 55 | 101 | 145 | 65 | 117 | 165 | 75 |
| 6 | 006 | 06 | 22 | 026 | 16 | 38 | 046 | 26 | カ | 54 | 066 | 36 | ニ | 70 | 106 | 46 | ヨ | 86 | 126 | 56 | 102 | 146 | 66 | 118 | 166 | 76 |
| 7 | 007 | 07 | 23 | 027 | 17 | 39 | 047 | 27 | キ | 55 | 067 | 37 | ヌ | 71 | 107 | 47 | ラ | 87 | 127 | 57 | 103 | 147 | 67 | 119 | 167 | 77 |
| 8 | 010 | 08 | 24 | 030 | 18 | 40 | 050 | 28 | ク | 56 | 070 | 38 | ネ | 72 | 110 | 48 | リ | 88 | 130 | 58 | 104 | 150 | 68 | 120 | 170 | 78 |
| 9 | 011 | 09 | 25 | 031 | 19 | 41 | 051 | 29 | ケ | 57 | 071 | 39 | ノ | 73 | 111 | 49 | ル | 89 | 131 | 59 | 105 | 151 | 69 | 121 | 171 | 79 |
| 10 | 012 | 0A | 26 | 032 | 1A | 42 | 052 | 2A | コ | 58 | 072 | 3A | ハ | 74 | 112 | 4A | レ | 90 | 132 | 5A | 106 | 152 | 6A | 122 | 172 | 7A |
| 11 | 013 | 0B | 27 | 033 | 1B | 43 | 053 | 2B | サ | 59 | 073 | 3B | ヒ | 75 | 113 | 4B | ロ | 91 | 133 | 5B | 107 | 153 | 6B | 123 | 173 | 7B |
| 12 | 014 | 0C | 28 | 034 | 1C | 44 | 054 | 2C | シ | 60 | 074 | 3C | フ | 76 | 114 | 4C | ワ | 92 | 134 | 5C | 108 | 154 | 6C | 124 | 174 | 7C |
| 13 | 015 | 0D | 29 | 035 | 1D | 45 | 055 | 2D | ス | 61 | 075 | 3D | ヘ | 77 | 115 | 4D | ニ | 93 | 135 | 5D | 109 | 155 | 6D | 125 | 175 | 7D |
| 14 | 016 | 0E | 30 | 036 | 1E | 46 | 056 | 2E | セ | 62 | 076 | 3E | ホ | 78 | 116 | 4E | 〃 | 94 | 136 | 5E | 110 | 156 | 6E | 126 | 176 | 7E |
| 15 | 017 | 0F | 31 | 037 | 1F | 47 | 057 | 2F | ソ | 63 | 077 | 3F | マ | 79 | 117 | 4F | 。 | 95 | 137 | 5F | 111 | 157 | 6F | 127 | 177 | 7F |

Displayed Character

| |
|---|
| ユ |
|---|

 Decimal 45 Octal 055 Hex 2D

DG International Character Set

| | | | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|------------------|------------------|
| 0
000
00 | 16
020
10 | 32
040
20 | 48
060
30 | 64
100
40 | 80
120
50 | 96
140
60 | 112
160
70 |
| 1
001
01 | 17
021
11 | 33
041
21 | 49
061
31 | 65
101
41 | 81
121
51 | 97
141
61 | 113
161
71 |
| 2
002
02 | 18
022
12 | 34
042
22 | 50
062
32 | 66
102
42 | 82
122
52 | 98
142
62 | 114
162
72 |
| 3
003
03 | 19
023
13 | 35
043
23 | 51
063
33 | 67
103
43 | 83
123
53 | 99
143
63 | 115
163
73 |
| 4
004
04 | 20
024
14 | 36
044
24 | 52
064
34 | 68
104
44 | 84
124
54 | 100
144
64 | 116
164
74 |
| 5
005
05 | 21
025
15 | 37
045
25 | 53
065
35 | 69
105
45 | 85
125
55 | 101
145
65 | 117
165
75 |
| 6
006
06 | 22
026
16 | 38
046
26 | 54
066
36 | 70
106
46 | 86
126
56 | 102
146
66 | 118
166
76 |
| 7
007
07 | 23
027
17 | 39
047
27 | 55
067
37 | 71
107
47 | 87
127
57 | 103
147
67 | 119
167
77 |
| 8
010
08 | 24
030
18 | 40
050
28 | 56
070
38 | 72
110
48 | 88
130
58 | 104
150
68 | 120
170
78 |
| 9
011
09 | 25
031
19 | 41
051
29 | 57
071
39 | 73
111
49 | 89
131
59 | 105
151
69 | 121
171
79 |
| 10
012
0A | 26
032
1A | 42
052
2A | 58
072
3A | 74
112
4A | 90
132
5A | 106
152
6A | 122
172
7A |
| 11
013
0B | 27
033
1B | 43
053
2B | 59
073
3B | 75
113
4B | 91
133
5B | 107
153
6B | 123
173
7B |
| 12
014
0C | 28
034
1C | 44
054
2C | 60
074
3C | 76
114
4C | 92
134
5C | 108
154
6C | 124
174
7C |
| 13
015
0D | 29
035
1D | 45
055
2D | 61
075
3D | 77
115
4D | 93
135
5D | 109
155
6D | 125
175
7D |
| 14
016
0E | 30
036
1E | 46
056
2E | 62
076
3E | 78
116
4E | 94
136
5E | 110
156
6E | 126
176
7E |
| 15
017
0F | 31
037
1F | 47
057
2F | 63
077
3F | 79
117
4F | 95
137
5F | 111
157
6F | 127
177
7F |

Displayed Character

| | |
|---|-----|
| ° | 69 |
| Å | 105 |
| | 45 |

 Decimal Octal Hex

Word-Processing, Greek, and Math Character Set

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----|-----|----|--|----|-----|----|------------|----|-----|----|----|----|-----|----|------------|----|-----|----|----------------|----|-----|----|---|-----|-----|----|----------------|-----|-----|----|
| 0 | 000 | 00 | | 16 | 020 | 10 | | 32 | 040 | 20 | 0 | 48 | 060 | 30 | !! | 64 | 100 | 40 | π | 80 | 120 | 50 | ┌ | 96 | 140 | 60 | 0 | 112 | 160 | 70 |
| 1 | 001 | 01 | | 17 | 021 | 11 | ┌ | 33 | 041 | 21 | 1 | 49 | 061 | 31 | α | 65 | 101 | 41 | ρ | 81 | 121 | 51 | ◆ | 97 | 141 | 61 | 1 | 113 | 161 | 71 |
| 2 | 002 | 02 | | 18 | 022 | 12 | L | 34 | 042 | 22 | 2 | 50 | 062 | 32 | β | 66 | 102 | 42 | σ | 82 | 122 | 52 | ▶ | 98 | 142 | 62 | 2 | 114 | 162 | 72 |
| 3 | 003 | 03 | | 19 | 023 | 13 | └ | 35 | 043 | 23 | 3 | 51 | 063 | 33 | γ | 67 | 103 | 43 | τ | 83 | 123 | 53 | ▷ | 99 | 143 | 63 | 3 | 115 | 163 | 73 |
| 4 | 004 | 04 | | 20 | 024 | 14 | ┘ | 36 | 044 | 24 | 4 | 52 | 064 | 34 | δ | 68 | 104 | 44 | υ | 84 | 124 | 54 | ◀ | 100 | 144 | 64 | 4 | 116 | 164 | 74 |
| 5 | 005 | 05 | | 21 | 025 | 15 | f | 37 | 045 | 25 | 5 | 53 | 065 | 35 | ϵ | 69 | 105 | 45 | ϕ | 85 | 125 | 55 | ▲ | 101 | 145 | 65 | 5 | 117 | 165 | 75 |
| 6 | 006 | 06 | | 22 | 026 | 16 | ~ | 38 | 046 | 26 | 6 | 54 | 066 | 36 | ζ | 70 | 106 | 46 | χ | 86 | 126 | 56 | ▼ | 102 | 146 | 66 | 6 | 118 | 166 | 76 |
| 7 | 007 | 07 | | 23 | 027 | 17 | ∂ | 39 | 047 | 27 | 7 | 55 | 067 | 37 | η | 71 | 107 | 47 | ψ | 87 | 127 | 57 | ⊠ | 103 | 147 | 67 | 7 | 119 | 167 | 77 |
| 8 | 010 | 08 | | 24 | 030 | 18 | ∇ | 40 | 050 | 28 | 8 | 56 | 070 | 38 | θ | 72 | 110 | 48 | ω | 88 | 130 | 58 | ⊞ | 104 | 150 | 68 | 8 | 120 | 170 | 78 |
| 9 | 011 | 09 | | 25 | 031 | 19 | ∠ | 41 | 051 | 29 | 9 | 57 | 071 | 39 | ι | 73 | 111 | 49 | Ω | 89 | 131 | 59 | ⊗ | 105 | 151 | 69 | 9 | 121 | 171 | 79 |
| 10 | 012 | 0A | | 26 | 032 | 1A | ƒ | 42 | 052 | 2A | ≠ | 58 | 072 | 3A | κ | 74 | 112 | 4A | Δ | 90 | 132 | 5A | ⊗ | 106 | 152 | 6A | H _F | 122 | 172 | 7A |
| 11 | 013 | 0B | | 27 | 033 | 1B | J | 43 | 053 | 2B | └┘ | 59 | 073 | 3B | λ | 75 | 113 | 4B | ¶ | 91 | 133 | 5B | ⊗ | 107 | 153 | 6B | ↑ | 123 | 173 | 7B |
| 12 | 014 | 0C | | 28 | 034 | 1C | √ | 44 | 054 | 2C | ↵ | 60 | 074 | 3C | μ | 76 | 114 | 4C | B _P | 92 | 134 | 5C | ▷ | 108 | 154 | 6C | → | 124 | 174 | 7C |
| 13 | 015 | 0D | | 29 | 035 | 1D | ✕ | 45 | 055 | 2D | ↯ | 61 | 075 | 3D | ν | 77 | 115 | 4D | B _E | 93 | 135 | 5D | ▷ | 109 | 155 | 6D | ← | 125 | 175 | 7D |
| 14 | 016 | 0E | | 30 | 036 | 1E | ∞ | 46 | 056 | 2E | → | 62 | 076 | 3E | ξ | 78 | 116 | 4E | F _N | 94 | 136 | 5E | ◄ | 110 | 156 | 6E | ↓ | 126 | 176 | 7E |
| 15 | 017 | 0F | | 31 | 037 | 1F | ∞ | 47 | 057 | 2F | . | 63 | 077 | 3F | o | 79 | 117 | 4F | F _E | 95 | 137 | 5F | ◄ | 111 | 157 | 6F | | 127 | 177 | 7F |

Displayed Character

| |
|-------|
| ξ |
|-------|

 Decimal Octal Hex

| |
|-----|
| 78 |
| 116 |
| 4E |

DG Line Drawing Character Set

| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|-----|----|----|-----|----|----|-----|----|----|-----|----|-----|----|-----|----|----|-----|----|-----|-----|----|-----|-----|----|
| 0 | 000 | 00 | 16 | 020 | 10 | 32 | 040 | 20 | 48 | 060 | 30 | Ⓡ | 64 | 100 | 40 | 80 | 120 | 50 | 96 | 140 | 60 | 112 | 160 | 70 |
| 1 | 001 | 01 | 17 | 021 | 11 | 33 | 041 | 21 | 49 | 061 | 31 | Ⓒ | 65 | 101 | 41 | 81 | 121 | 51 | 97 | 141 | 61 | 113 | 161 | 71 |
| 2 | 002 | 02 | 18 | 022 | 12 | 34 | 042 | 22 | 50 | 062 | 32 | ◐ | 66 | 102 | 42 | 82 | 122 | 52 | 98 | 142 | 62 | 114 | 162 | 72 |
| 3 | 003 | 03 | 19 | 023 | 13 | 35 | 043 | 23 | 51 | 063 | 33 | | 67 | 103 | 43 | 83 | 123 | 53 | 99 | 143 | 63 | 115 | 163 | 73 |
| 4 | 004 | 04 | 20 | 024 | 14 | 36 | 044 | 24 | 52 | 064 | 34 | | 68 | 104 | 44 | 84 | 124 | 54 | 100 | 144 | 64 | 116 | 164 | 74 |
| 5 | 005 | 05 | 21 | 025 | 15 | 37 | 045 | 25 | 53 | 065 | 35 | | 69 | 105 | 45 | 85 | 125 | 55 | 101 | 145 | 65 | 117 | 165 | 75 |
| 6 | 006 | 06 | 22 | 026 | 16 | 38 | 046 | 26 | 54 | 066 | 36 | | 70 | 106 | 46 | 86 | 126 | 56 | 102 | 146 | 66 | 118 | 166 | 76 |
| 7 | 007 | 07 | 23 | 027 | 17 | 39 | 047 | 27 | 55 | 067 | 37 | | 71 | 107 | 47 | 87 | 127 | 57 | 103 | 147 | 67 | 119 | 167 | 77 |
| 8 | 010 | 08 | 24 | 030 | 18 | 40 | 050 | 28 | 56 | 070 | 38 | | 72 | 110 | 48 | 88 | 130 | 58 | 104 | 150 | 68 | 120 | 170 | 78 |
| 9 | 011 | 09 | 25 | 031 | 19 | 41 | 051 | 29 | 57 | 071 | 39 | | 73 | 111 | 49 | 89 | 131 | 59 | 105 | 151 | 69 | 121 | 171 | 79 |
| 10 | 012 | 0A | 26 | 032 | 1A | 42 | 052 | 2A | 58 | 072 | 3A | | 74 | 112 | 4A | 90 | 132 | 5A | 106 | 152 | 6A | 122 | 172 | 7A |
| 11 | 013 | 0B | 27 | 033 | 1B | 43 | 053 | 2B | 59 | 073 | 3B | | 75 | 113 | 4B | 91 | 133 | 5B | 107 | 153 | 6B | 123 | 173 | 7B |
| 12 | 014 | 0C | 28 | 034 | 1C | 44 | 054 | 2C | 60 | 074 | 3C | --- | 76 | 114 | 4C | 92 | 134 | 5C | 108 | 154 | 6C | 124 | 174 | 7C |
| 13 | 015 | 0D | 29 | 035 | 1D | 45 | 055 | 2D | 61 | 075 | 3D | ÷ | 77 | 115 | 4D | 93 | 135 | 5D | 109 | 155 | 6D | 125 | 175 | 7D |
| 14 | 016 | 0E | 30 | 036 | 1E | 46 | 056 | 2E | 62 | 076 | 3E | ⊙ | 78 | 116 | 4E | 94 | 136 | 5E | 110 | 156 | 6E | 126 | 176 | 7E |
| 15 | 017 | 0F | 31 | 037 | 1F | 47 | 057 | 2F | 63 | 077 | 3F | ™ | 79 | 117 | 4F | 95 | 137 | 5F | 111 | 157 | 6F | 127 | 177 | 7F |

Displayed Character

| | | |
|--|--|-----|
| | | 41 |
| | | 051 |
| | | 29 |

 Decimal Octal Hex

DG Special Graphics Character Set (PC Characters)

| | | | | | | | | | | | | | | | |
|--|-----------------|--|-----------------|---|-----------------|---|-----------------|---|-----------------|--|-----------------|---|------------------|---|------------------|
| | 0
000
00 | | 16
020
10 | | 32
040
20 | ○ | 48
060
30 | Ω | 64
100
40 | | 80
120
50 | | 96
140
60 | ≡ | 112
160
70 |
| | 1
001
01 | | 17
021
11 | ? | 33
041
21 | ● | 49
061
31 | ┌ | 65
101
41 | | 81
121
51 | | 97
141
61 | · | 113
161
71 |
| | 2
002
02 | | 18
022
12 | Ⓟ | 34
042
22 | ♂ | 50
062
32 | └ | 66
102
42 | | 82
122
52 | | 98
142
62 | η | 114
162
72 |
| | 3
003
03 | | 19
023
13 | ℵ | 35
043
23 | ♀ | 51
063
33 | | 67
103
43 | | 83
123
53 | | 99
143
63 | ■ | 115
163
73 |
| | 4
004
04 | | 20
024
14 | — | 36
044
24 | ♪ | 52
064
34 | | 68
104
44 | | 84
124
54 | | 100
144
64 | ÿ | 116
164
74 |
| | 5
005
05 | | 21
025
15 | — | 37
045
25 | ♪ | 53
065
35 | | 69
105
45 | | 85
125
55 | | 101
145
65 | | 117
165
75 |
| | 6
006
06 | | 22
026
16 | — | 38
046
26 | ⚙ | 54
066
36 | | 70
106
46 | | 86
126
56 | | 102
146
66 | | 118
166
76 |
| | 7
007
07 | | 23
027
17 | — | 39
047
27 | ↕ | 55
067
37 | | 71
107
47 | | 87
127
57 | | 103
147
67 | | 119
167
77 |
| | 8
010
08 | | 24
030
18 | □ | 40
050
28 | ■ | 56
070
38 | | 72
110
48 | | 88
130
58 | Γ | 104
150
68 | | 120
170
78 |
| | 9
011
09 | | 25
031
19 | ☺ | 41
051
29 | ↕ | 57
071
39 | | 73
111
49 | | 89
131
59 | π | 105
151
69 | | 121
171
79 |
| | 10
012
0A | | 26
032
1A | ☹ | 42
052
2A | ┌ | 58
072
3A | | 74
112
4A | | 90
132
5A | Σ | 106
152
6A | | 122
172
7A |
| | 11
013
0B | | 27
033
1B | ♥ | 43
053
2B | ↔ | 59
073
3B | | 75
113
4B | | 91
133
5B | Φ | 107
153
6B | | 123
173
7B |
| | 12
014
0C | | 28
034
1C | ♣ | 44
054
2C | Δ | 60
074
3C | | 76
114
4C | | 92
134
5C | θ | 108
154
6C | | 124
174
7C |
| | 13
015
0D | | 29
035
1D | ♠ | 45
055
2D | ÿ | 61
075
3D | | 77
115
4D | | 93
135
5D | ϑ | 109
155
6D | | 125
175
7D |
| | 14
016
0E | | 30
036
1E | • | 46
056
2E | Ⓟ | 62
076
3E | | 78
116
4E | | 94
136
5E | ε | 110
156
6E | | 126
176
7E |
| | 15
017
0F | | 31
037
1F | • | 47
057
2F | Ⓜ | 63
077
3F | | 79
117
4F | | 95
137
5F | Π | 111
157
6F | | 127
177
7F |

Displayed Character  47
057
2F Decimal
Octal
Hex

VT Multinational Character Set

| | | | | | | | | | | | | | | | |
|--------|------------------|--------|------------------|----|------------------|----|------------------|---|------------------|---|------------------|---|------------------|---|------------------|
| | 128
200
80 | D
S | 144
220
90 | | 160
240
A0 | o | 176
260
B0 | À | 192
300
C0 | | 208
320
D0 | à | 224
340
E0 | | 240
360
F0 |
| | 129
201
81 | | 145
221
91 | ì | 161
241
A1 | ± | 177
261
B1 | Á | 193
301
C1 | Ñ | 209
321
D1 | á | 225
341
E1 | ñ | 241
361
F1 |
| | 130
202
82 | | 146
222
92 | ¢ | 162
242
A2 | 2 | 178
262
B2 | Â | 194
302
C2 | Ò | 210
322
D2 | â | 226
342
E2 | ò | 242
362
F2 |
| | 131
203
83 | | 147
223
93 | £ | 163
243
A3 | 3 | 179
263
B3 | Ã | 195
303
C3 | Ó | 211
323
D3 | ã | 227
343
E3 | ó | 243
363
F3 |
| I
D | 132
204
84 | | 148
224
94 | | 164
244
A4 | | 180
264
B4 | Ä | 196
304
C4 | Ô | 212
324
D4 | ä | 228
344
E4 | ô | 244
364
F4 |
| N
E | 133
205
85 | | 149
225
95 | ¥ | 165
245
A5 | µ | 181
265
B5 | Å | 197
305
C5 | Õ | 213
325
D5 | å | 229
345
E5 | õ | 245
365
F5 |
| | 134
206
86 | | 150
226
96 | | 166
246
A6 | ¶ | 182
266
B6 | Æ | 198
306
C6 | Ö | 214
326
D6 | æ | 230
346
E6 | ö | 246
366
F6 |
| | 135
207
87 | | 151
227
97 | § | 167
247
A7 | · | 183
267
B7 | Ç | 199
307
C7 | Æ | 215
327
D7 | ç | 231
347
E7 | œ | 247
367
F7 |
| H
S | 136
210
88 | | 152
230
98 | ¸ | 168
250
A8 | | 184
270
B8 | È | 200
310
C8 | Ø | 216
330
D8 | è | 232
350
E8 | ø | 248
370
F8 |
| | 137
211
89 | | 153
231
99 | © | 169
251
A9 | ı | 185
271
B9 | É | 201
311
C9 | Ù | 217
331
D9 | é | 233
351
E9 | ù | 249
371
F9 |
| | 138
212
8A | | 154
232
9A |  | 170
252
AA |  | 186
272
BA | Ê | 202
312
CA | Ú | 218
332
DA | ê | 234
352
EA | ú | 250
372
FA |
| | 139
213
8B | C
I | 155
233
9B | << | 171
253
AB | >> | 187
273
BB | Ë | 203
313
CB | Û | 219
333
DB | ë | 235
353
EB | û | 251
373
FB |
| | 140
214
8C | S
T | 156
234
9C | | 172
254
AC | ¼ | 188
274
BC | Ì | 204
314
CC | Ü | 220
334
DC | ì | 236
354
EC | ü | 252
374
FC |
| R
I | 141
215
8D | | 157
235
9D | | 173
255
AD | ½ | 189
275
BD | Í | 205
315
CD | ÿ | 221
335
DD | í | 237
355
ED | ÿ | 253
375
FD |
| S
2 | 142
216
8E | | 158
236
9E | | 174
256
AE | | 190
276
BE | Î | 206
316
CE | | 222
336
DE | î | 238
356
EE | | 254
376
FE |
| S
3 | 143
217
8F | | 159
237
9F | | 175
257
AF | ı | 191
277
BF | Ï | 207
317
CF | ß | 223
337
DF | ï | 239
357
EF | | 255
377
FF |

Displayed Character

| | |
|--------|------------------|
| C
I | 155
233
9B |
|--------|------------------|

 Decimal Octal Hex

VT Special Graphics Character Set (VT Line Drawing)

| | | | | | | | | | | | | | | | |
|--|-----------------|--|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|----------------|------------------|---|------------------|
| | 0
000
00 | | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | @ | 64
100
40 | P | 80
120
50 | ◆ | 96
140
60 | | 112
160
70 |
| | 1
001
01 | | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | ▒ | 97
141
61 | | 113
161
71 |
| | 2
002
02 | | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | H _T | 98
142
62 | | 114
162
72 |
| | 3
003
03 | | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | F _F | 99
143
63 | | 115
163
73 |
| | 4
004
04 | | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | C _R | 100
144
64 | | 116
164
74 |
| | 5
005
05 | | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | L _F | 101
145
65 | | 117
165
75 |
| | 6
006
06 | | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | ° | 102
146
66 | | 118
166
76 |
| | 7
007
07 | | 23
027
17 | ' | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | ± | 103
147
67 | | 119
167
77 |
| | 8
010
08 | | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | N _L | 104
150
68 | | 120
170
78 |
| | 9
011
09 | | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | V _T | 105
151
69 | ≤ | 121
171
79 |
| | 10
012
0A | | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | └ | 106
152
6A | ≥ | 122
172
7A |
| | 11
013
0B | | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | I | 91
133
5B | └ | 107
153
6B | π | 123
173
7B |
| | 12
014
0C | | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | \ | 92
134
5C | └ | 108
154
6C | ≠ | 124
174
7C |
| | 13
015
0D | | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D | | 93
135
5D | └ | 109
155
6D | ∞ | 125
175
7D |
| | 14
016
0E | | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ^ | 94
136
5E | └ | 110
156
6E | . | 126
176
7E |
| | 15
017
0F | | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | | 95
137
5F | └ | 111
157
6F | ▒ | 127
177
7F |

Displayed Character

| |
|---|
| % |
|---|

| |
|-----------------|
| 37
045
25 |
|-----------------|

 Decimal
Octal
Hex

ISO 8859/1.2 Character Set

| | | | | | | | | | | | | | | | |
|----------------|------------------|----------------|------------------|--------|------------------|---|------------------|---|------------------|---|------------------|---|------------------|---|------------------|
| | 128
200
80 | D _S | 144
220
90 | | 160
240
A0 | ° | 176
260
B0 | À | 192
300
C0 | Ɔ | 208
320
D0 | à | 224
340
E0 | ò | 240
360
F0 |
| | 129
201
81 | | 145
221
91 | ·
i | 161
241
A1 | ± | 177
261
B1 | Á | 193
301
C1 | Ñ | 209
321
D1 | á | 225
341
E1 | ñ | 241
361
F1 |
| | 130
202
82 | | 146
222
92 | ¢ | 162
242
A2 | ² | 178
262
B2 | Â | 194
302
C2 | Ò | 210
322
D2 | â | 226
342
E2 | ò | 242
362
F2 |
| | 131
203
83 | | 147
223
93 | £ | 163
243
A3 | ³ | 179
263
B3 | Ã | 195
303
C3 | Ó | 211
323
D3 | ã | 227
343
E3 | ó | 243
363
F3 |
| I _D | 132
204
84 | | 148
224
94 | ¤ | 164
244
A4 | ´ | 180
264
B4 | Ä | 196
304
C4 | Ô | 212
324
D4 | ä | 228
344
E4 | ô | 244
364
F4 |
| N _E | 133
205
85 | | 149
225
95 | ¥ | 165
245
A5 | µ | 181
265
B5 | Å | 197
305
C5 | Õ | 213
325
D5 | å | 229
345
E5 | õ | 245
365
F5 |
| | 134
206
86 | | 150
226
96 | ¦ | 166
246
A6 | ¶ | 182
266
B6 | Æ | 198
306
C6 | Ö | 214
326
D6 | æ | 230
346
E6 | ö | 246
366
F6 |
| | 135
207
87 | | 151
227
97 | § | 167
247
A7 | · | 183
267
B7 | Ç | 199
307
C7 | × | 215
327
D7 | ç | 231
347
E7 | ÷ | 247
367
F7 |
| H _S | 136
210
88 | | 152
230
98 | ¨ | 168
250
A8 | ¸ | 184
270
B8 | È | 200
310
C8 | Ø | 216
330
D8 | è | 232
350
E8 | ø | 248
370
F8 |
| | 137
211
89 | | 153
231
99 | © | 169
251
A9 | ¹ | 185
271
B9 | É | 201
311
C9 | Ù | 217
331
D9 | é | 233
351
E9 | ù | 249
371
F9 |
| | 138
212
8A | | 154
232
9A | ª | 170
252
AA | º | 186
272
BA | Ê | 202
312
CA | Ú | 218
332
DA | ê | 234
352
EA | ú | 250
372
FA |
| | 139
213
8B | C _I | 155
233
9B | « | 171
253
AB | » | 187
273
BB | Ë | 203
313
CB | Û | 219
333
DB | ë | 235
353
EB | û | 251
373
FB |
| | 140
214
8C | S _T | 156
234
9C | ¬ | 172
254
AC | ¼ | 188
274
BC | Ì | 204
314
CC | Ü | 220
334
DC | ì | 236
354
EC | ü | 252
374
FC |
| R _I | 141
215
8D | | 157
235
9D | — | 173
255
AD | ½ | 189
275
BD | Í | 205
315
CD | Ý | 221
335
DD | í | 237
355
ED | ý | 253
375
FD |
| S ₂ | 142
216
8E | | 158
236
9E | ® | 174
256
AE | ¾ | 190
276
BE | Î | 206
316
CE | Þ | 222
336
DE | î | 238
356
EE | þ | 254
376
FE |
| S ₃ | 143
217
8F | | 159
237
9F | — | 175
257
AF | ¿ | 191
277
BF | Ï | 207
317
CF | ß | 223
337
DF | ï | 239
357
EF | ÿ | 255
377
FF |

Displayed Character

| | |
|----------------|------------------|
| C _I | 155
233
9B |
|----------------|------------------|

















































 Decimal
 Octal
 Hex

PCTERM Low Character Set (0 hex through 7F hex)

| | | | | | | | | | | | | | | | |
|---|-----------------|----|-----------------|----|-----------------|---|-----------------|---|-----------------|---|-----------------|---|------------------|---|------------------|
| | 0
000
00 | ▶ | 16
020
10 | | 32
040
20 | 0 | 48
060
30 | @ | 64
100
40 | P | 80
120
50 | ' | 96
140
60 | p | 112
160
70 |
| ☺ | 1
001
01 | ◀ | 17
021
11 | ! | 33
041
21 | 1 | 49
061
31 | A | 65
101
41 | Q | 81
121
51 | a | 97
141
61 | q | 113
161
71 |
| ☹ | 2
002
02 | ↕ | 18
022
12 | " | 34
042
22 | 2 | 50
062
32 | B | 66
102
42 | R | 82
122
52 | b | 98
142
62 | r | 114
162
72 |
| ♥ | 3
003
03 | !! | 19
023
13 | # | 35
043
23 | 3 | 51
063
33 | C | 67
103
43 | S | 83
123
53 | c | 99
143
63 | s | 115
163
73 |
| ♦ | 4
004
04 | ¶ | 20
024
14 | \$ | 36
044
24 | 4 | 52
064
34 | D | 68
104
44 | T | 84
124
54 | d | 100
144
64 | t | 116
164
74 |
| ♣ | 5
005
05 | § | 21
025
15 | % | 37
045
25 | 5 | 53
065
35 | E | 69
105
45 | U | 85
125
55 | e | 101
145
65 | u | 117
165
75 |
| ♠ | 6
006
06 | ■ | 22
026
16 | & | 38
046
26 | 6 | 54
066
36 | F | 70
106
46 | V | 86
126
56 | f | 102
146
66 | v | 118
166
76 |
| • | 7
007
07 | ↕ | 23
027
17 | , | 39
047
27 | 7 | 55
067
37 | G | 71
107
47 | W | 87
127
57 | g | 103
147
67 | w | 119
167
77 |
| • | 8
010
08 | ↑ | 24
030
18 | (| 40
050
28 | 8 | 56
070
38 | H | 72
110
48 | X | 88
130
58 | h | 104
150
68 | x | 120
170
78 |
| ○ | 9
011
09 | ↓ | 25
031
19 |) | 41
051
29 | 9 | 57
071
39 | I | 73
111
49 | Y | 89
131
59 | i | 105
151
69 | y | 121
171
79 |
| ◐ | 10
012
0A | → | 26
032
1A | * | 42
052
2A | : | 58
072
3A | J | 74
112
4A | Z | 90
132
5A | j | 106
152
6A | z | 122
172
7A |
| ♂ | 11
013
0B | ← | 27
033
1B | + | 43
053
2B | ; | 59
073
3B | K | 75
113
4B | [| 91
133
5B | k | 107
153
6B | { | 123
173
7B |
| ♀ | 12
014
0C | └ | 28
034
1C | , | 44
054
2C | < | 60
074
3C | L | 76
114
4C | \ | 92
134
5C | l | 108
154
6C | | 124
174
7C |
| ♪ | 13
015
0D | ↔ | 29
035
1D | - | 45
055
2D | = | 61
075
3D | M | 77
115
4D |] | 93
135
5D | m | 109
155
6D | } | 125
175
7D |
| ♫ | 14
016
0E | ▶ | 30
036
1E | . | 46
056
2E | > | 62
076
3E | N | 78
116
4E | ^ | 94
136
5E | n | 110
156
6E | ~ | 126
176
7E |
| ⚙ | 15
017
0F | ▼ | 31
037
1F | / | 47
057
2F | ? | 63
077
3F | O | 79
117
4F | _ | 95
137
5F | o | 111
157
6F | Δ | 127
177
7F |

Displayed Character ← 27
033
1B Decimal
Octal
Hex

PCTERM High Character Set (80 hex through FF hex)

| | | | | | | | | | | | | | | | |
|---|------------------|----------------|------------------|----|------------------|---|------------------|---|------------------|--|------------------|---|------------------|---|------------------|
| Ç | 128
200
80 | É | 144
220
90 | á | 160
240
A0 |  | 176
260
B0 |  | 192
300
C0 |  | 208
320
D0 | α | 224
340
E0 | ≡ | 240
360
F0 |
| ü | 129
201
81 | æ | 145
221
91 | í | 161
241
A1 |  | 177
261
B1 |  | 193
301
C1 |  | 209
321
D1 | β | 225
341
E1 | ± | 241
361
F1 |
| é | 130
202
82 | Æ | 146
222
92 | ó | 162
242
A2 |  | 178
262
B2 |  | 194
302
C2 |  | 210
322
D2 | Γ | 226
342
E2 | ≥ | 242
362
F2 |
| â | 131
203
83 | ô | 147
223
93 | ú | 163
243
A3 |  | 179
263
B3 |  | 195
303
C3 |  | 211
323
D3 | π | 227
343
E3 | ≤ | 243
363
F3 |
| ä | 132
204
84 | ö | 148
224
94 | ñ | 164
244
A4 |  | 180
264
B4 |  | 196
304
C4 |  | 212
324
D4 | Σ | 228
344
E4 | ƒ | 244
364
F4 |
| à | 133
205
85 | ò | 149
225
95 | Ñ | 165
245
A5 |  | 181
265
B5 |  | 197
305
C5 |  | 213
325
D5 | σ | 229
345
E5 | J | 245
365
F5 |
| ã | 134
206
86 | û | 150
226
96 | ä | 166
246
A6 |  | 182
266
B6 |  | 198
306
C6 |  | 214
326
D6 | μ | 230
346
E6 | ÷ | 246
366
F6 |
| ç | 135
207
87 | ù | 151
227
97 | œ | 167
247
A7 |  | 183
267
B7 |  | 199
307
C7 |  | 215
327
D7 | τ | 231
347
E7 | ≈ | 247
367
F7 |
| ê | 136
210
88 | ÿ | 152
230
98 | ¿ | 168
250
A8 |  | 184
270
B8 |  | 200
310
C8 |  | 216
330
D8 | Φ | 232
350
E8 | ○ | 248
370
F8 |
| ë | 137
211
89 | ÿ | 153
231
99 | ┐ | 169
251
A9 |  | 185
271
B9 |  | 201
311
C9 |  | 217
331
D9 | θ | 233
351
E9 | ● | 249
371
F9 |
| è | 138
212
8A | ÿ | 154
232
9A | └ | 170
252
AA |  | 186
272
BA |  | 202
312
CA |  | 218
332
DA | Ω | 234
352
EA | • | 250
372
FA |
| ï | 139
213
8B | ¢ | 155
233
9B | ½ | 171
253
AB |  | 187
273
BB |  | 203
313
CB |  | 219
333
DB | δ | 235
353
EB | √ | 251
373
FB |
| î | 140
214
8C | £ | 156
234
9C | ¼ | 172
254
AC |  | 188
274
BC |  | 204
314
CC |  | 220
334
DC | ∞ | 236
354
EC | η | 252
374
FC |
| ì | 141
215
8D | ¥ | 157
235
9D | ı | 173
255
AD |  | 189
275
BD |  | 205
315
CD |  | 221
335
DD | φ | 237
355
ED | ² | 253
375
FD |
| ÿ | 142
216
8E | P _t | 158
236
9E | << | 174
256
AE |  | 190
276
BE |  | 206
316
CE |  | 222
336
DE | ε | 238
356
EE | ■ | 254
376
FE |
| ÿ | 143
217
8F | J | 159
237
9F | >> | 175
257
AF |  | 191
277
BF |  | 207
317
CF |  | 223
337
DF | ∩ | 239
357
EF | | 255
377
FF |

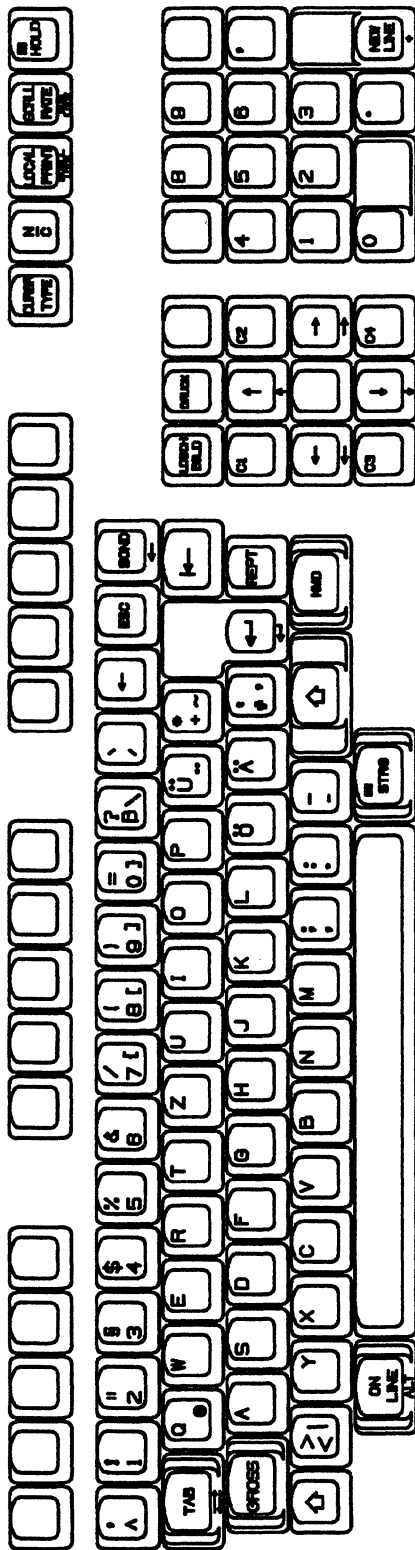
| | | | |
|---------------------|---|------------------|-------------------------|
| Displayed Character | ¢ | 155
233
9B | Decimal
Octal
Hex |
|---------------------|---|------------------|-------------------------|

End of Appendix

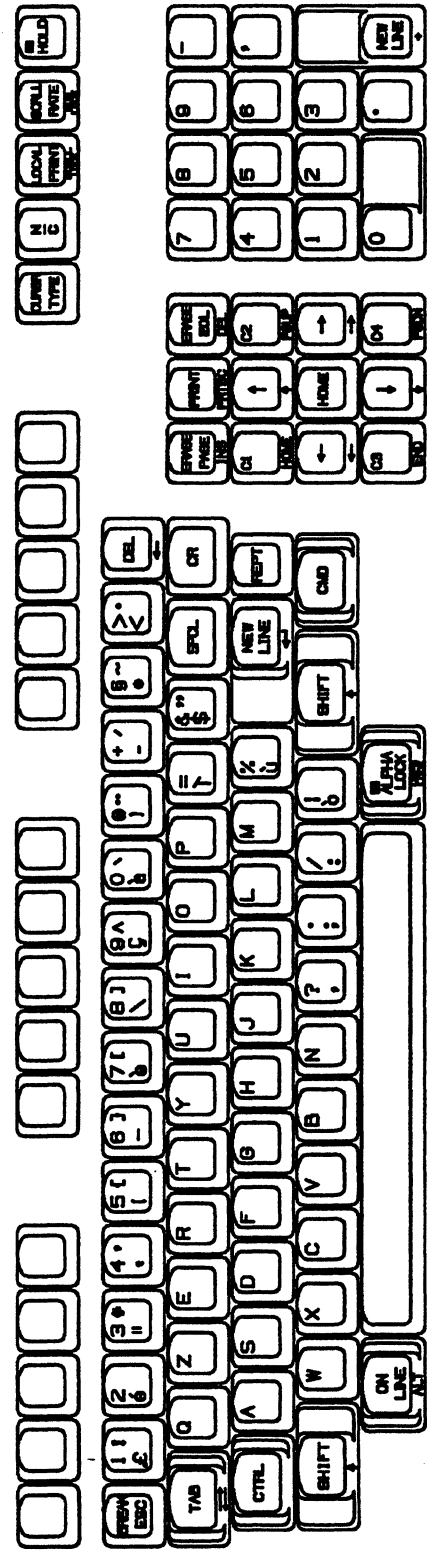
Appendix B

National Language Keyboards

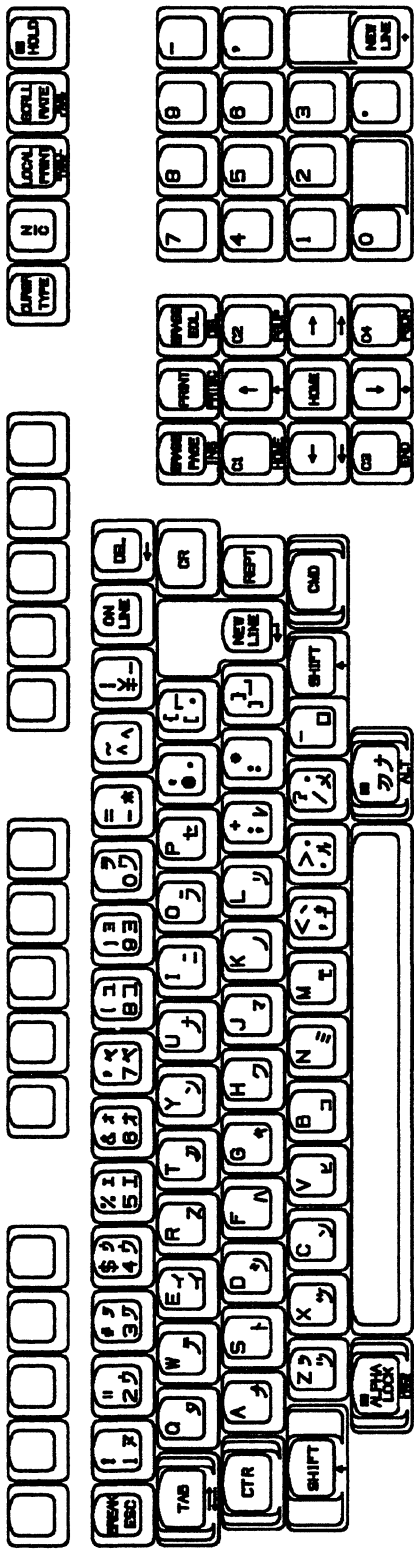
This appendix contains the national-language keycap mappings for both the 107-key Data General proprietary keyboard and the 101-key IBM PC AT-style keyboard.



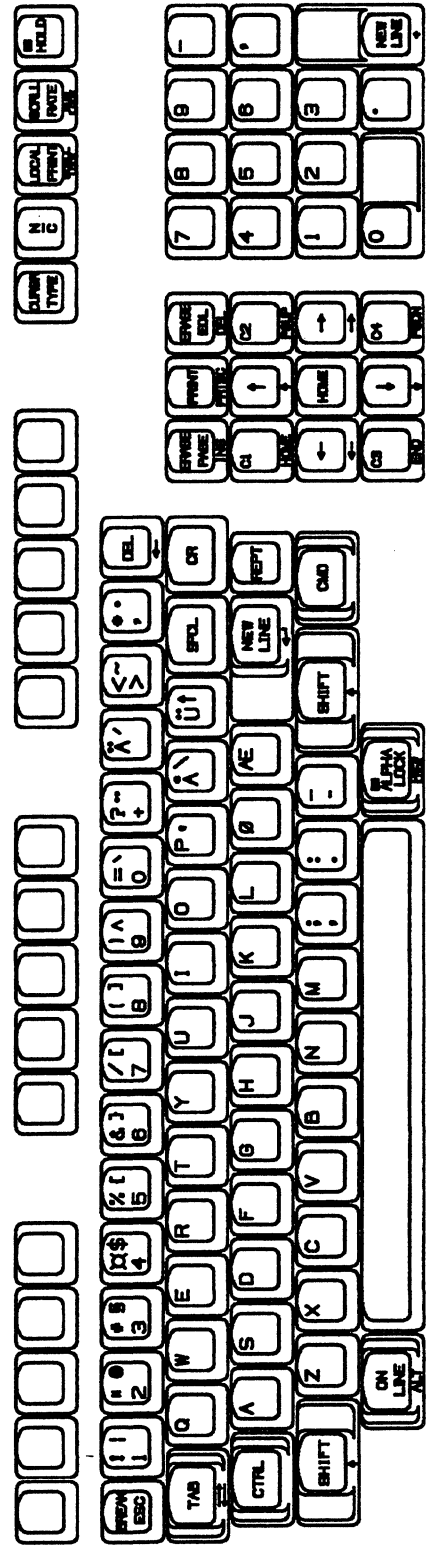
**German
107-key Keyboard**



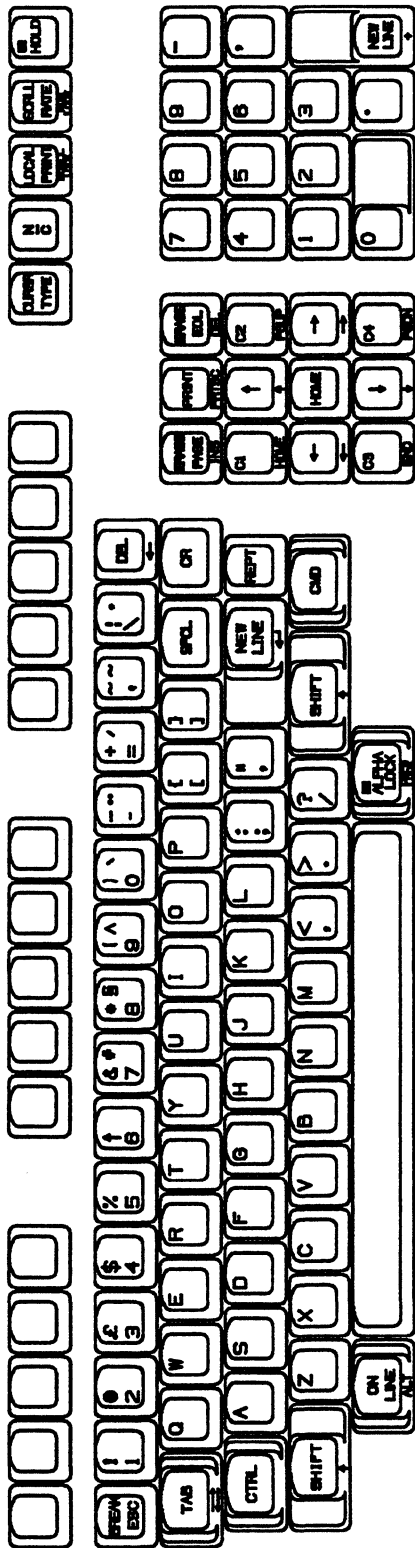
**Italian
107-key Keyboard**



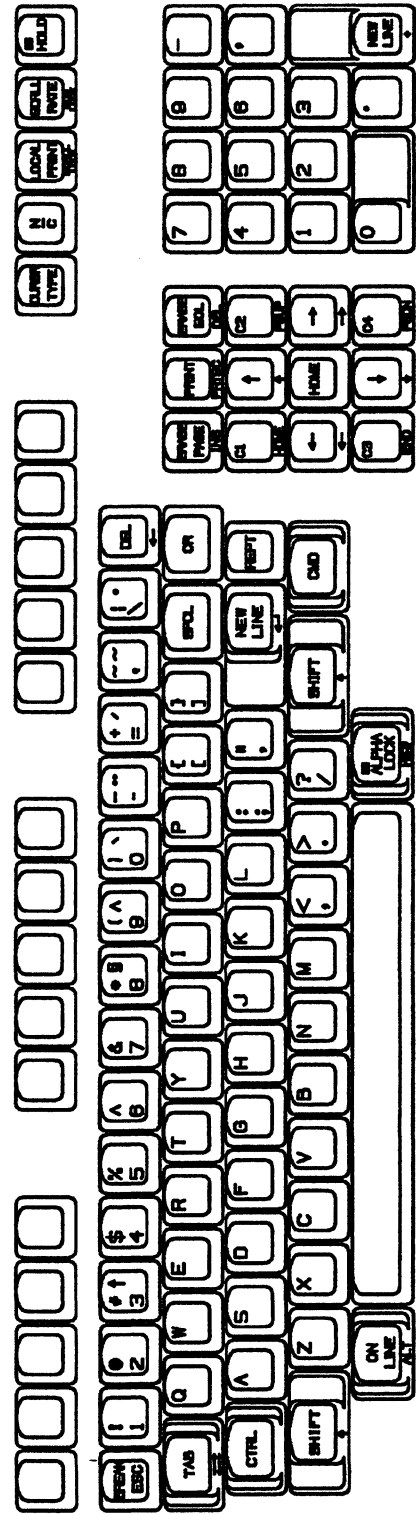
**Katakana
107-key Keyboard**



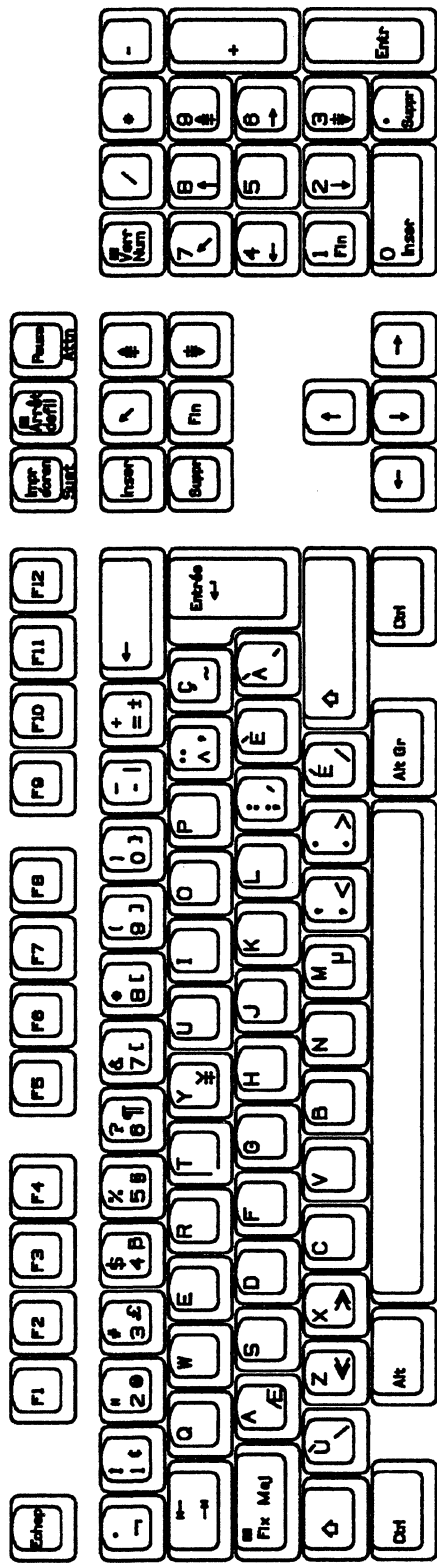
**Norwegian
107-key Keyboard**



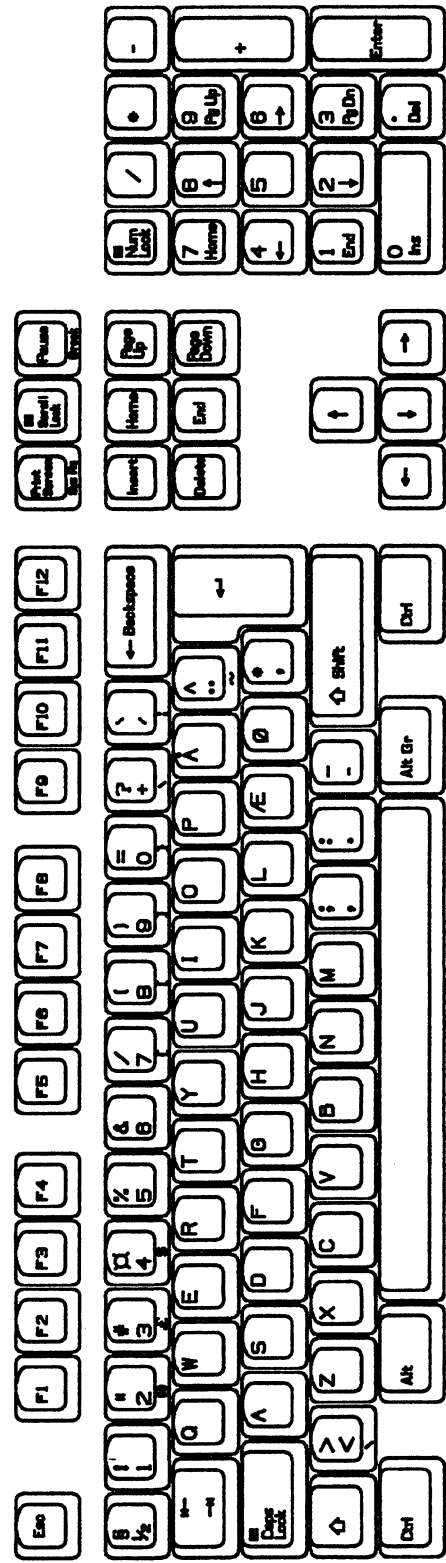
**United Kingdom
107-key Keyboard**



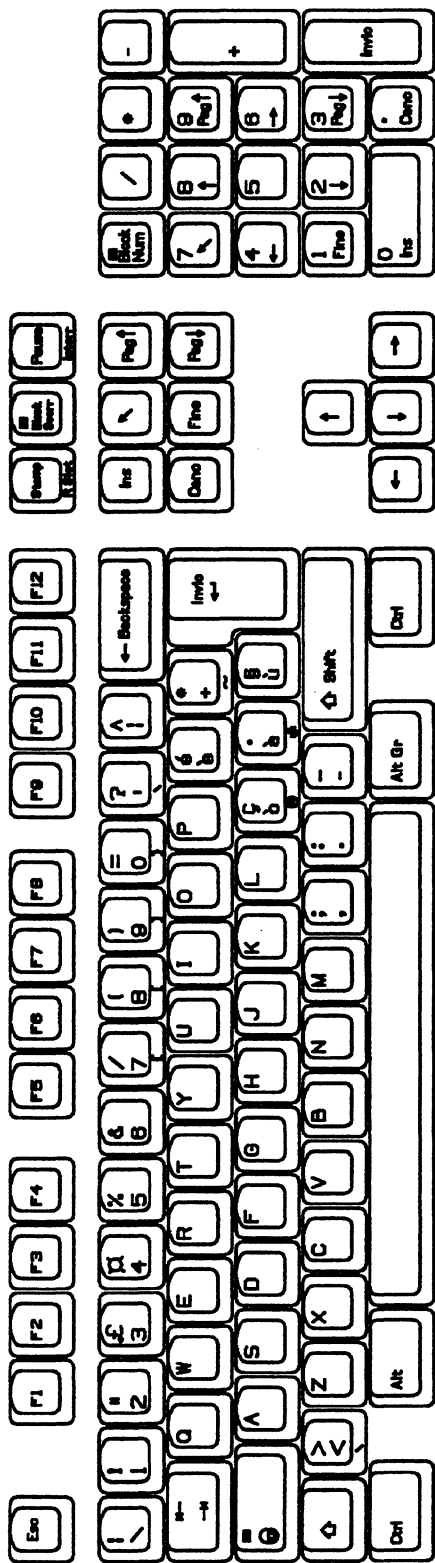
**United States
107-key Keyboard**



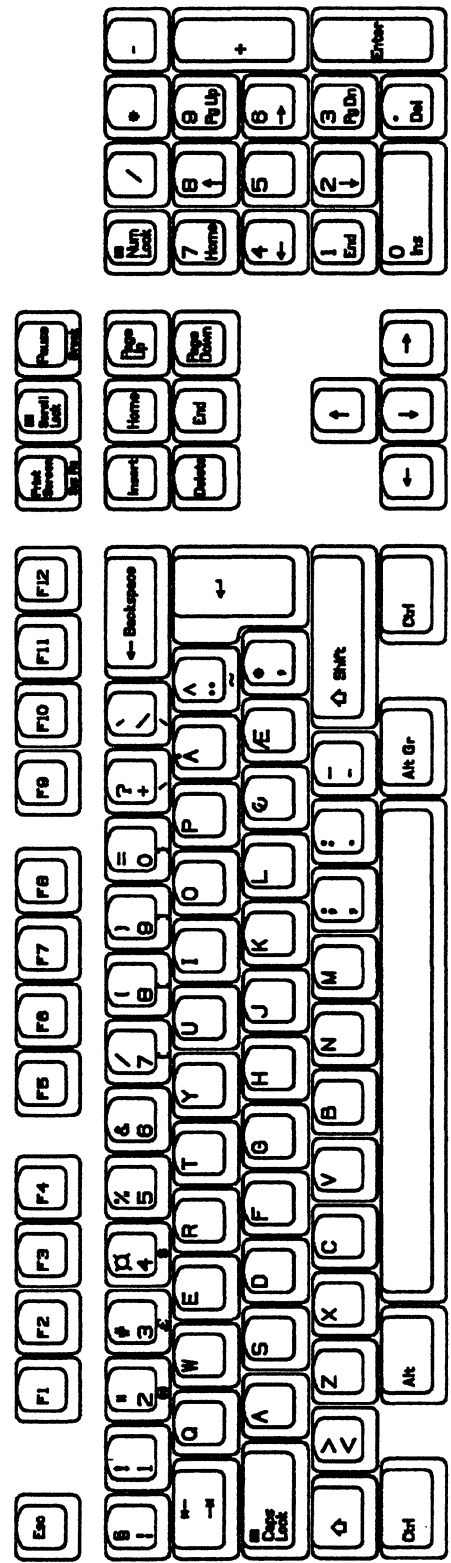
**Canadian/French
102-key Keyboard**



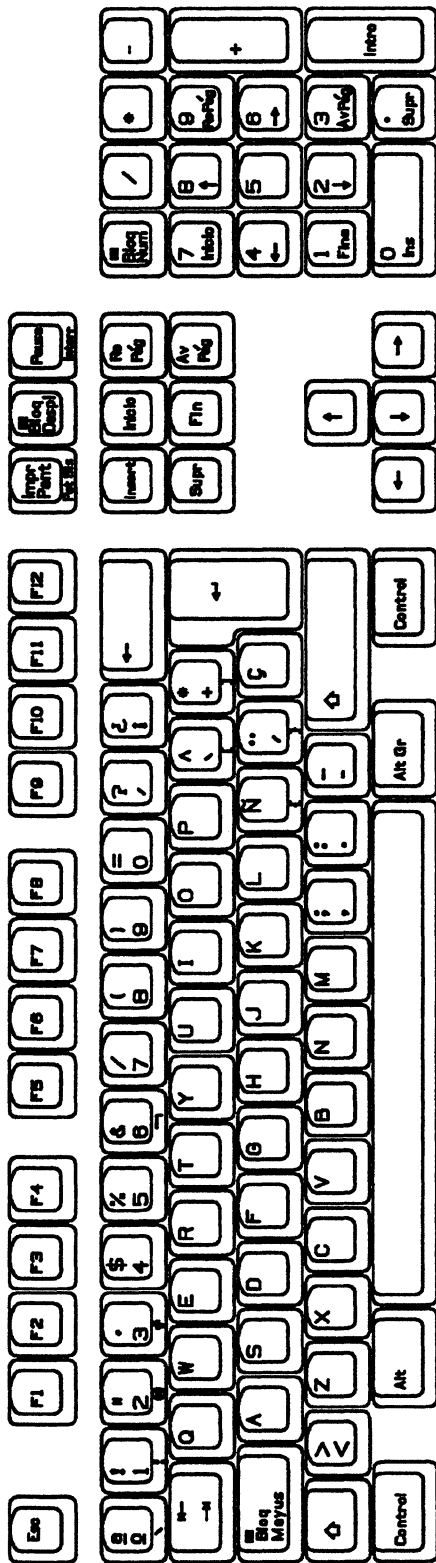
**Danish
102-key Keyboard**



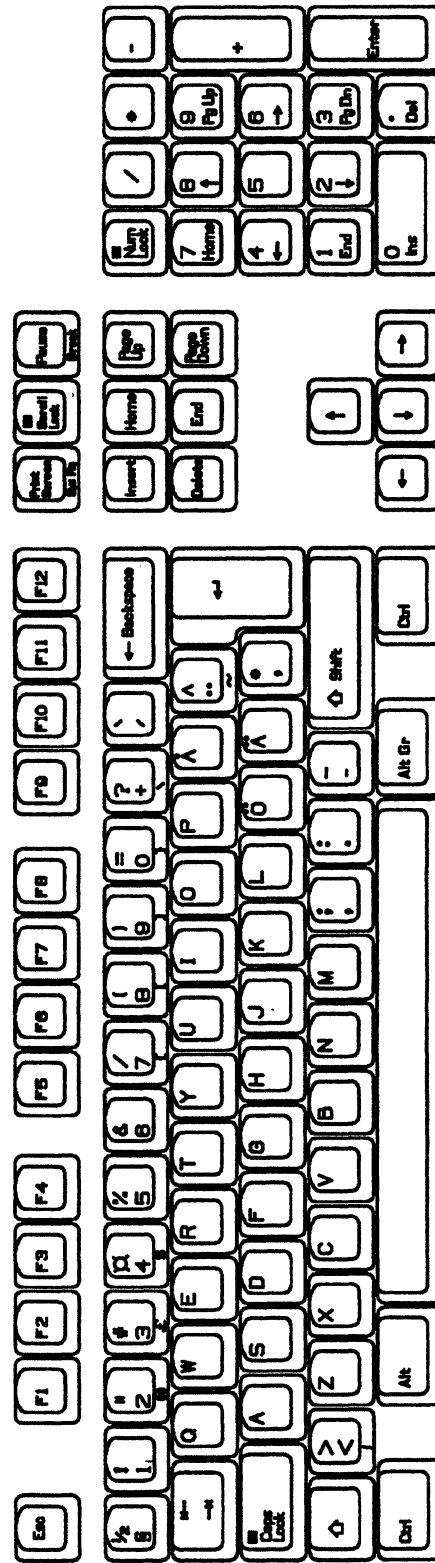
**Italian
102-key Keyboard**



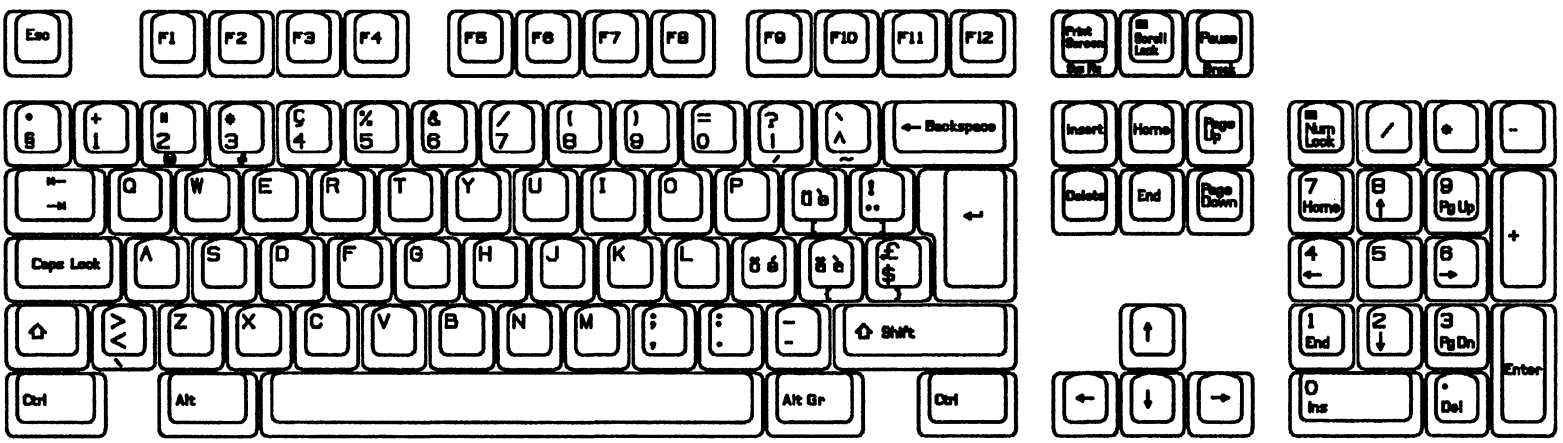
**Norwegian
102-key Keyboard**



**Spanish
102-key Keyboard**

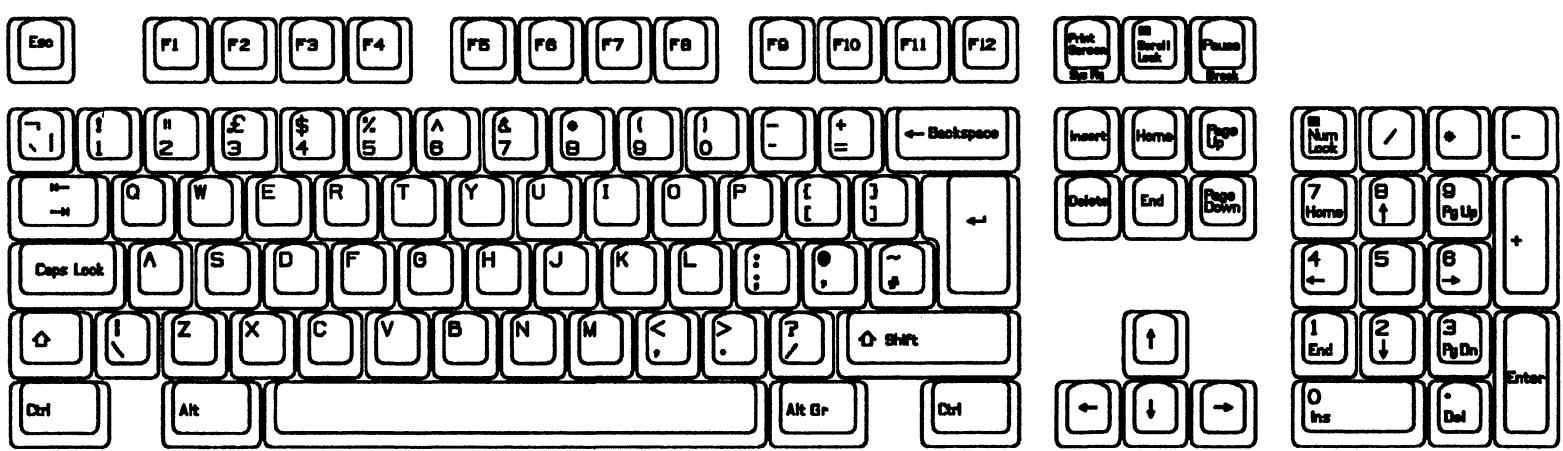


**Swedish/Finnish
102-key Keyboard**



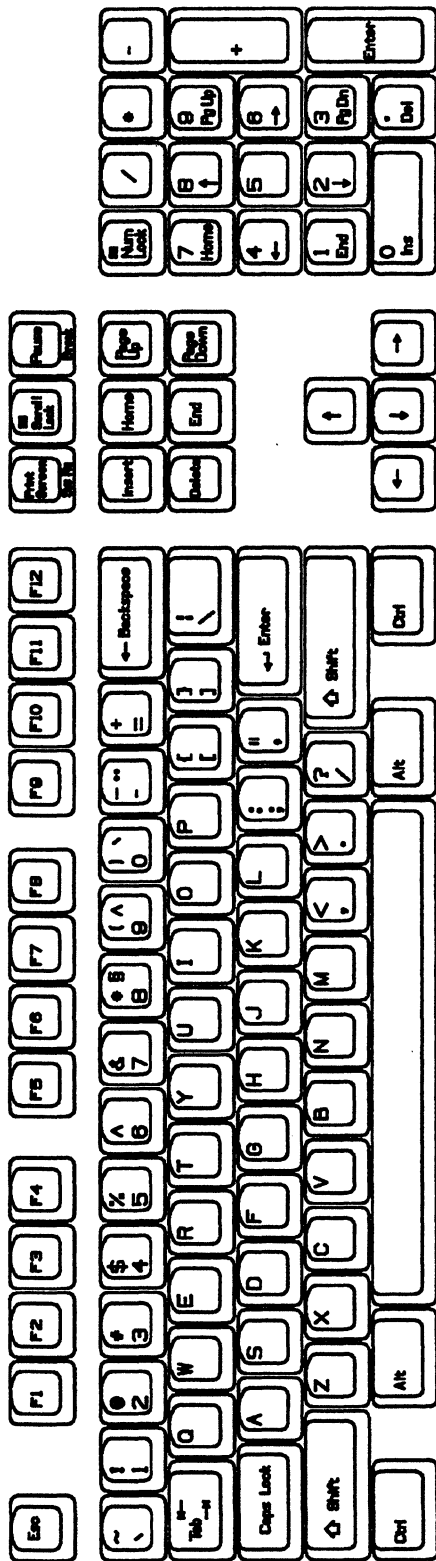
Swiss
102-key Keyboard

014-002111



United Kingdom
102-key Keyboard

B-13



**United States
102-key Keyboard**

Appendix C

Sample Programs

This appendix contains sample programs that illustrate programming methods for Data General D217, D413, and D463 display terminals. Where possible, we demonstrated functions that show a sampling of important terminal features and common applications. However, these programs are not meant to, and do not, show all possible command combinations, or program requirements. The modes and emulations covered in this appendix are listed below:

Data General Native-Mode "C"

Data General Native-Mode "Fortran 77"

VT320 Emulation "C"

VT320 Emulation "Fortran 77"

VT52 Emulation "C"

VT52 Emulation "Fortran 77"

Tektronix 4010 Emulation "C"

Tektronix 4010 Emulation "Fortran 77"

Notice

Initial version 1 June 1990.

Copyright (c) 1990 by Data General Corporation. Non-exclusive license to use, distribute, and modify this code is hereby granted without monetary consideration, provided all copyright messages are included intact in all such derivative works. Standard disclaimers regarding merchantability, suitability for a particular purpose, et cetera, all apply. This code is distributed with no warranty whatsoever.

These programs are examples of programming methods for Data General Dasher D216, D216E, D412, D462, D216+, D413, D413, D217, D463 and D462+ display terminals. Where possible, demonstration functions were chosen to show a sampling of important terminal features and common applications. However, these programs are not meant to, and do not, show all possible command combinations, or program requirements.

Send comments or questions to:

QUERIES@MRX.WEBO.DG.COM.

Source code for these programs is available from a mail server at:

SERVER@MRX.WEBO.DG.COM.

Data General Native-Mode "C"

```
/*
 * dg_mode.c -
 *   DG native mode demonstration.
 *
 * Initial version 01JUN90.
 *
 * Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
 * use, distribute, and modify this code is hereby granted without monetary
 * consideration, provided this copyright message is included in all such
 * derivative works. Standard disclaimers regarding merchantability,
 * suitability for a particular purpose, et cetera, all apply. This code is
 * distributed with no warranty whatsoever.
 */

#include <stdio.h>
#include <ctype.h>

/*
 * This routine can be used to condition a communication line for binary I/O
 * when writing a C program under DG AOS. It switches both input and output
 * to binary mode, which disables character translation, and unbuffers the
 * input.
 */
void
set_up_terminal()
{
    if (freopen("@input","j",stdin)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary input.\n");
        exit(1);
    }
    setbuf(stdin,NULL);
    if (freopen("@output","k",stdout)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary output.\n");
        exit(2);
    }
}

/*
 * These routines may be used on any system that uses ASCII to provide support
 * for a native-mode DG terminal in a C program.
 */

/* This flag is non-zero if the terminal is expecting real hex data */
static int in_DG_Unix_mode=0;

/* This variable saves the current attributes for comparison */
static int old_attr=0;

/* This is the failing character in the return sequence */
static int offending_character;

/* This macro insures that a given character is printable */
#define ccs(c) (((c) &0x7F) >= ' ' && ((c) &0x7F) < 0x7F) ? (c) &0x7F : '.'

```

```

/*
 * Send a DG <nn> pair. 'f' is the output device, and 'n' is between 0 and
 * 255.
 */
int
fputnn(n,f)
int n;
FILE *f;
{
    if (in_DG_Unix_mode)                /* if in Unix mode */
        return fprintf(f,"%02X",n);    /* then send real hex */
    else {
        fputc('0'+(n>>4),f);           /* otherwise generate DG hex */
        fputc('0'+n&0xF,f);
    }
    return 2;                            /* two characters shipped */
}

/*
 * Get a DG <n> digit. 'f' is the input device. Returns a number between 0
 * and 15 or EOF.
 */
int
fgetn(f)
FILE *f;
{
    int temp;

    offending_character=fgetc(f);        /* get character from host */
    if (offending_character==EOF) return EOF;
    if (in_DG_Unix_mode) {
        temp=toupper(offending_character); /* upcase alpha */
        if (temp<'0' || temp>'F' || temp<'A' && temp>'9') {
            fprintf(stderr,"\n\15Illegal <H> character '%c' (%02X)\n\15",
                ccs(temp),temp);
            return EOF;
        }
        return temp>'9' ? temp-'A'+10 : temp-'0'; /* decode hex */
    }
    temp=offending_character;
    if (temp<'@' || temp>'O') {
        fprintf(stderr,"\n\15Illegal <n> character '%c' (%02X)\n\15",
            ccs(temp),temp);
        return EOF;
    }
    return temp&0xF;                    /* decode DG hex */
}

/*
 * Decode a DG mode <nn> pair. Returns an integer in the range 0 to 255, or
 * EOF
 */
int
fgetnn(f)
FILE *f;
{
    int temp,t2;

    if ((temp=fgetn(f))==EOF) return EOF; /* get one <n> parm */
    if ((t2=fgetn(f))==EOF) return EOF; /* get the second */
    return (temp<<4) + t2;                /* combine them */
}

```

```

/*
 * Set DG Unix mode. Note that this will work whether or not DG Unix mode is
 * currently enabled.
 */
void
set_DG_unix(f)
FILE *f;
{
    in_DG_Unix_mode=1;      /* set the global flag */
    fputs("\036P@1",f);    /* send the enter-unix-mode sequence */
}

/*
 * Reset DG Unix mode. Note that this will work whether or not DG Unix mode is
 * currently enabled.
 */
void
reset_DG_unix(f)
FILE *f;
{
    in_DG_Unix_mode=0;     /* clear the global flag */
    fputs("\036P@0",f);    /* send the exit-unix-mode sequence */
}

/*
 * Example of DG mode Write Window Address command. Note that this is not
 * generally useful with DG Unix Mode.
 * 'l' is line number (0 to 23) and 'c' is column number (0 to 126).
 */
void
write_window_address(l,c,f)
int l,c;
FILE *f;
{
    fputc('\20',f);        /* Write ctrl-P header */
    fputc((char)c,f);      /* Write column */
    fputc((char)l,f);      /* Write row */
}

/*
 * Example of DG mode Write Screen Address implementation. This command will
 * work with DG unix mode.
 */
void
write_screen_address(l,c,f)
int l,c;
FILE *f;
{
    fputs("\36FP",f);      /* Write <036>FP header */
    fputnn(c,f);           /* Write <nn> column */
    fputnn(l,f);           /* Write <nn> row */
}

```

```

/*
 * Expect a certain fixed response from the terminal.
 * Returns 0 for success, 1 for failure.
 */

int
fexpect(s,f)
char *s;
FILE *f;
{
    int i;

    while (*s) /* while there are characters, check them */
        if (*s != (i=fgetc(f))) {
            fprintf(stderr, "\n\15Got '%c' (%02X) while expecting '%c'
(%02X).\n\15",
                    ccs(i),i,ccs(*s),*s);
            return 1;
        }
        else
            s++;
    return 0; /* everything compared -- return OK status */
}

/*
 * Example testing program. This will send the cursor to all D200 screen
 * addresses and make sure that the cursor actually gets there.
 * Returns 0 for success, 1 for failure.
 */

int
test(func,fin,fout)
void (*func)(/* int,int */); /* DG C doesn't like ANSI prototypes */
FILE *fin,*fout;
{
    int l,c,rc,rl;

    for (l=0;l<24;l++) /* loop over rows and columns */
        for (c=0;c<80;c++) {
            (*func)(l,c,fout); /* position the cursor */
            fputs("\36Fb",fout); /* read position */
            if (fexpect("\36o8",fin)) { /* expect header */
                printf("\n\15Bad Read Screen Address return
header.\n\15");
                return 1;
            }
            rc=fgetnn(fin); /* get column data */
            if (rc==EOF) {
                printf("\n\15Error in column data returned.\n\15");
                return 1;
            }
            rl=fgetnn(fin); /* get row data */
            if (rl==EOF) {
                printf("\n\15Error in row data returned.\n\15");
                return 1;
            }
            if (rc!=c || rl != l) {
                printf("\n\15Went to position %d,%d but read back
%d,%d\n\15",l,c,rl,rc);
                return 1;
            }
        }
    return 0;
}

```

```

/*
 * Driver for above testing routine. This will call the above test routine and
 * direct it to use Write Window Address and Write Screen Address.
 */

int
run_test(argc,argv)
int argc;
char **argv;
{
/* Go into binary mode */
    set_up_terminal();

/* Test the write-window-address command */
    fputs("\14Testing Write Window Address ...",stdout);
    if (test(write_window_address,stdin,stdout))
        return 1;

/* Test the write-screen-address command */
    fputs("\14Testing Write Screen Address ...",stdout);
    if (test(write_screen_address,stdin,stdout))
        return 1;

/* Successful! */
    fputs("\nDone",stdout);
    return 0;
}

/*
 * Turn off the indicated attributes. (Internal routine.)
 */

static void
reset_attrs(attrs,f)
int attrs;
FILE *f;
{
    int i,j;
    static char *dg_off[]={"\35","\36E","\25","\17"};

    if (attrs)
        for(i=1,j=0;i<16;i<=&1,j++)
            if (i&attrs) printf(dg_off[j]);
}

/*
 * Turn on the indicated attributes. (Internal routine.)
 */

static void
set_attrs(attrs,f)
int attrs;
FILE *f;
{
    int i,j;
    static char *dg_on[]={"\34","\36D","\24","\16"};

    if (attrs)
        for(i=1,j=0;i<16;i<=&1,j++)
            if (i&attrs) printf(dg_on[j]);
}

```

```

/*
 * Set the attributes to the given state with as few commands as possible.
 */
void
change_attributes(attr,f)
int attr;
FILE *f;
{
    reset_attrs((attr^old_attr)&old_attr,f);
    set_attrs((attr^old_attr)&attr,f);
    old_attr=attr;
}

/*
 * Constants for use with above routine.
 */
#define DIM      1
#define REVERSE  2
#define UNDER    4
#define BLINK    8

/*
 * Read binary information from 'fp' and output it in PINK style using D400
 * commands to 'f'.
 */

#define linend(chr)    { change_attributes(DIM,f); fputc(chr,f); ch='\n'; }

void
show_file(fp,f)
FILE *fp,*f;
{
    int ch,chlast=0,attrs;
    while ((ch=fgetc(fp))!=EOF) {
        attrs=0;
        if (ch > 127) {
            /* if high bit set */
            attrs = UNDER; /* then turn on underscore */
            ch -= 128;
            if (ch < 32) { /* convert if control code */
                attrs |= DIM;
                ch+='@';
            }
            if (ch == 127) {
                attrs |= DIM;
                ch = '~';
            }
        }
        else {
            if (ch == 127) { /* convert if control code */
                attrs = DIM;
                ch = '~';
            }
            if (ch < 32) {
                attrs = DIM;
                switch (ch) { /* special line-end symbols */
                    case '\n':
                        if (chlast==' ') linend('|');
                        break;
                    case '\r':
                        linend('<');
                        break;
                    case '\14':
                        linend('*');
                        break;
                }
            }
        }
        fputc(ch,f);
        chlast=ch;
    }
}

```

```

                                default:
                                ch+=64;
                                }
                                }
                                }
                                change_attributes(attrs,f); /* set new attributes */
                                fputc(ch,f); /* send out the character */
                                chlast=ch;
                                }
                                fclose(fp);
}

/*
 * Main routine for example code above. This program will read any number of
 * input files and show the output in PINK format.
 */
int
show(argc,argv)
int argc;
char **argv;
{
    FILE *fp;

    if (argc<2) {
        fputs("Usage:\n\tSHOW [file-name [file-name ...]]\n",stderr);
        return 0;
    }
    set_up_terminal();
    while (*++argv) /* for each given file */
        if (access(*argv,0)
            fprintf(stderr,"%s does not exist!\n",*argv);
            else if (access(*argv,4)
                fprintf(stderr,"Insufficient access rights on %s!\n",*argv);
            else if ((fp=fopen(*argv,"j"))==NULL)
                fprintf(stderr,"Cannot open %s!\n",*argv);
            else {
                if (argc>2) { /* if more than one file, give names */
                    change_attributes(0);
                    printf("\nFile: %s\n",*argv);
                }
                show_file(fp,stdout); /* display the file */
            }
        change_attributes(0,stdout);
        return 0;
}

/* Main program */
int
main(argc,argv)
int argc;
char **argv;
{
    char str[256];

    printf("Run (S)how or (T)est? ");
    if (gets(str)) switch(tolower(*str)) {
        case 's': return show(argc,argv);
        case 't': return run_test(argc,argv);
    }
    return 1;
}

```

Data General Native-Mode "Fortran 77"

```
C dg_mode.f77 -
C   DG Native mode demonstration.
C
C Initial version 01JUN90.
C
C Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
C use, distribute, and modify this code is hereby granted without monetary
C consideration, provided this copyright message is included in all such
C derivative works. Standard disclaimers regarding merchantability,
C suitability for a particular purpose, et cetera, all apply. This code is
C distributed with no warranty whatsoever.

C This routine can be used to condition a communication line for binary I/O
C when writing an F77 program under DG AOS. It switches both input and output
C to binary mode, which disables character translation, and unbuffers the
C input.

      subroutine set_up_terminal(iin,iout)
      integer iin,iout

      integer in_DG_Unix_mode,old_attr
      common /DGTERM/ in_DG_Unix_mode,old_attr

C Set I/O to binary.
      open(unit=iin,file='@input',screenedit='no',maxrecl=1,
1         delimiter='include',force='yes',iointent='input',
2         mode='binary',err=100)
      open(unit=iout,file='@output',carriagecontrol='none',
1         screenedit='no',force='yes',iointent='output',mode='binary',
2         err=100)

C Reset global variables.
      in_DG_Unix_mode=0
      old_attr=0
      return

100  continue
      write(*,1)
1     format(' Unable to open your terminal as a binary device. ')
      stop
      end

C These routines may be used on any system that uses ASCII to provide support
C for a native-mode DG terminal in an F77 program.

      subroutine sendnn(n,iout)
      integer n,iout
```


C Send a DG <nn> pair. iout is the output device, and 'n' is between 0 and C 255.

```
intrinsic mod

integer in_DG_Unix_mode,old_attr
common /DGTERM/ in_DG_Unix_mode,old_attr

integer*2 czero
data czero/2H0 /
integer*2 c1,c2

if (in_DG_Unix_mode.ne.0) then
  write(iout,1) n
else
  c1=czero + (n/16)*256
  c2=czero + mod(n,16)*256
  write(iout,2) c1,c2
endif
1 format(Z2.2)
2 format(2A1)
return
end
```

integer function getn(iin)
integer iin

C Get a DG <n> digit. iin is the input device. Returns a number between 0
C and 15 (or -1 for EOF).

```
intrinsic ichar
integer in_DG_Unix_mode,old_attr
common /DGTERM/ in_DG_Unix_mode,old_attr
character ch
integer ich
```

C Read in the character.
read(iin,1,err=100,end=100) ch
1 format(A1)

C Strip high bit.
ich=ichar(ch)
if (ich.ge.128) ich=ich-128

if (in_DG_unix_mode.ne.0) then

C In DG Unix mode; check character value.
if (ich.ge.ichar('a') .and. ich.le.ichar('f')) ich=ich-32
if (ich.lt.ichar('0') .or. ich.gt.ichar('F') .or.
1 (ich.lt.ichar('A') .and. ich.gt.ichar('9'))) then
 write(*,2) ich
 stop
endif

C Convert hex to decimal and return.
getn = ich-ichar('0')
if (ich.gt.ichar('9')) getn=ich-ichar('A')+10
else

C In DG native mode; check character value.
if (ich.lt.ichar('@') .or. ich.gt.ichar('O')) then
 write(*,2) ich
2 format(' Illegal "nn" character: ',Z2.2)
 stop
endif

```

C Convert DG hex to decimal and return.
    getn = ich-ichar('@')
    endif
    return

100  continue
    getn = -1
    return
    end

    integer function getnn(iin)
    integer iin

C Decode a DG mode <nn> pair. Returns an integer in the range 0 to 255, or
C EOF (-1)

    integer t1,t2
    external getn
    integer getn

C Default is error.
    getnn=-1

C Get first <n> character.
    t1=getn(iin)
    if (t1.lt.0) return

C Get second.
    t2=getn(iin)
    if (t2.lt.0) return

C Combine them and return.
    getnn=t1*16+t2
    return
    end

    subroutine set_DG_unix(iout)
    integer iout

C Set DG Unix mode. Note that this will work whether or not DG Unix mode is
C currently enabled.

    integer in_DG_Unix_mode,old_attr
    common /DGTERM/ in_DG_Unix_mode,old_attr

C Set the flag and put the terminal into DG Unix mode.
    in_DG_unix_mode=1
    write(iout,1)
1    format('<036>P@1')
    return
    end

    subroutine reset_DG_unix(iout)
    integer iout

C Reset DG Unix mode. Note that this will work whether or not DG Unix mode is
C currently enabled.

    integer in_DG_Unix_mode,old_attr
    common /DGTERM/ in_DG_Unix_mode,old_attr

```

```

C Clear the flag and take the terminal out of DG Unix mode.
  in_DG_unix_mode=0
  write(iout,1)
1   format('<036>P@0')
    return
    end

    subroutine write_window_address(l,c,iout)
      integer l,c,iout

      integer*2 l2,c2

C Example of DG mode Write Window Address command. Note that this is not
C generally useful with DG Unix Mode.
C 'l' is line number (0 to 23) and 'c' is column number (0 to 126).

C Convert line and column to character values and ship them.
  l2 = l*256+32
  c2 = c*256+32
  write(iout,1) c2,l2
1   format('<020>',2A1)
    return
    end

    subroutine write_screen_address(l,c,iout)
      integer l,c,iout

C Example of DG mode Write Screen Address implementation. This command will
C work with DG unix mode.

C Send <036>FP header.
  write(iout,1)
1   format('<036>FP')

C Send column and line numbers.
  call sendnn(c,iout)
  call sendnn(l,iout)
  return
  end

  character function ccs(c)
  character c

C Insure that the given character is printable.

  intrinsic ichar,char
  integer ich

C Strip high bit.
  ich=ichar(c)
  if (ich.ge.128) ich=ich-128

C If it's a control character, replace it with '.'.
  if (ich.lt.32.or.ich.eq.127) ich=ichar('.')

  ccs=char(ich)
  return
  end

  subroutine expect(str,iin)
  character*(*) str
  integer iin

```

C Expect a certain fixed response from the terminal

```
intrinsic ichar
external ccs
integer ic
character in,ccs

ic=0
10 continue
   ic=ic+1

C Blank terminates the string.
   if (str(ic:ic).eq.' ') return

C Get next character from host.
   read(iin,'(A1)') in
   if (str(ic:ic).ne.in) then
     write(*,1) ccs(in),ichar(in),ccs(str(ic:ic)),ichar(str(ic:ic))
     stop
   endif
1   format(' Got ',A1,' (',Z2.2,') while expecting ',A1,' (',
1     Z2.2,')')
   go to 10
end
```

```
subroutine run_test(func,iin,iout)
logical func
integer iin,iout
```

C Example testing program. This will send the cursor to all D200 screen C addresses and make sure that the cursor actually gets there.

```
external getnn
integer getnn
integer l,c,rc,rl

C Loop over all rows and columns.
do 10 l=0,23
do 10 c=0,79
  rl=l
  rc=c

C Call requested function.
  if (func) call write_screen_address(rl,rc,iout)
  if (.not.func) call write_window_address(rl,rc,iout)

C Send read-cursor-position command.
  write(iout,1)
1   format('<036>Fb')

C Expect return header and data.
  call expect('<036>o8 ',iin)
  rc=getnn(iin)
  rl=getnn(iin)
  if (rc.ne.c .or. rl.ne.l) then
    write(*,2) l,c,rl,rc
    stop
  endif
10  continue
2   format(' Went to ',i2,',',i2,' and got ',i2,',',i2,')
   return
end

subroutine test
```

C Driver for above testing routine. This will call the above test routine and
 C direct it to use Write Window Address and Write Screen Address.

```
call set_up_terminal(5,6)
```

C Test write-window-address functions.

```
write(*,1)
1 format(' <014>Testing Write Window Address ...')
  call run_test(.false.,5,6)
```

C Test write-screen-address functions.

```
write(*,2)
2 format(' <014>Testing Write Screen Address ...')
  call run_test(.true.,5,6)
  return
end
```

```
subroutine reset_attrs(attrs,iout)
integer attrs,iout
```

C Turn off the indicated attributes. (Internal routine.)

```
intrinsic iand
character*6 dg_off(4)
data dg_off/' ('<035>' )', ' ('<036>E' )', ' ('<025>' )',
1 ' ('<017>' )'/
integer i,j

if (attrs.eq.0) return
i=1
do 10 j=1,4
if (iand(i,attrs).ne.0) write(iout,dg_off(j))
i=i*2
10 continue
return
end
```

```
subroutine set_attrs(attrs,iout)
```

C Turn on the indicated attributes. (Internal routine.)

```
integer attrs,iout
intrinsic iand
character*6 dg_on(4)
data dg_on/' ('<034>' )', ' ('<036>D' )', ' ('<024>' )',
1 ' ('<016>' )'/
integer i,j

if (attrs.eq.0) return
i=1
do 10 j=1,4
if (iand(i,attrs).ne.0) write(iout,dg_on(j))
i=i*2
10 continue
return
end
```

```
subroutine change_attributes(attr,iout)
integer attr,iout
```

C Set the attributes to the given state with as few commands as possible.

```
intrinsic ieor,iand
integer in_DG Unix_mode,old_attr
common /DGTERM/ in_DG Unix_mode,old_attr
integer temp
```

```

C Get differences.
  temp=ieor(attr,old_attr)

C Clear those that need to be cleared.
  call reset_attrs(iand(temp,old_attr),iout)

C Set those that need to be set.
  call set_attrs(iand(temp,attr),iout)

  old_attr=attr
  return
end

  subroutine show_file(fin,istext,iout)
  integer fin,iout
  logical istext

C Read binary information from fin and output it in PINK style using D400
C commands to iout.
C This program runs very slowly since the output is unbuffered. This is done
C deliberately since this is a pedagogic example.

  intrinsic char,ichar
  integer dim,reverse,under,blink
  data dim,reverse,under,blink/1,2,4,8/
  integer attrs,ich,ipos
  character ch,chlast
  character*255 chstr

C Assume last character was a normal ASCII character.
  chlast='a'
  ipos=0
10  continue

C Get next character from file.
  if (istext) then
    if (ipos.eq.0) read(fin,3,end=100,err=200) chstr
3   format(A255)
    ipos = ipos+1
    ich = ichar(chstr(ipos:ipos))
    if (ich.eq.10.or.ipos.eq.255) ipos=0
  else
    read(fin,1,end=100,err=200) ch
1   format(A1)
    ich = ichar(ch)
  endif
  attrs=0
  if (ich .ge. 128) then

C   If high bit is set, then turn on underscore attribute.
    attrs=under
    ich=ich-128

C   If control character, then turn on dim attribute.
    if (ich .lt. 32) then
      attrs=attrs+dim
      ich=ich+64
    else if (ich .eq. 127) then
      attrs=attrs+dim
      ich=ichar('~')
    endif
  else
    if (ich .eq. 127) then

```

```

C      If control character, then turn on dim attribute.
      attrs=dim
      ich=ichar('~')
      else if (ich .lt. 32) then
        attrs=dim

C      Check for special line-end sequences.
      if (ich .eq. 10) then
        if (chlast .eq. ' ') then
          call change_attributes(dim,iout)
          write(iout,'''|''')
        endif
      else if (ich .eq. 13) then
        call change_attributes(dim,iout)
        write(iout,'''<074>''')
        ich=10
      else if (ich .eq. 12) then
        call change_attributes(dim,iout)
        write(iout,'''*''')
        ich=10
      else
        ich=ich+64
      endif
    endif
  endif

C Set the attributes and send out the adjusted character.
  call change_attributes(attrs,iout)
  ch=char(ich)
  write(iout,1) ch
  chlast=ch
  go to 10

200  continue
     write(*,2)
2    format(' Unexpected error while reading file!')
100  continue
     return
     end

     subroutine extract_next(inline,i,ilen,fname)
     character*80 inline
     integer i,ilen
     character*32 fname

C Internal routine -- extract next file name from input line.
     integer j
     fname=' '
     j=1
10   continue
     if (inline(i:i).gt.' ') go to 20
     i=i+1
     if (i.gt.ilen) return
     go to 10
20   continue
     fname(j:j)=inline(i:i)
     j=j+1
     i=i+1
     if (i.gt.ilen.or.j.gt.32) return
     if (inline(i:i).gt.' ') go to 20
     return
     end

     subroutine show

```

C Main routine for example code above. This program will read any number of C input files and show the output in PINK format.

```
integer nfiles,i
character*32 fname
character*80 inline
logical  istext
```

C Prompt for file names.

```
write(*,'(' Enter names of files to show: ' ')')
read(*,'(A80)') inline
```

C Count the number of files.

```
do 20 ilen=80,1,-1
  if (inline(ilen:ilen).ne.' ') go to 30
20  continue
30  continue
   nfiles=0
   i=1
40  continue
   call extract_next(inline,i,ilen,fname)
   if (fname(1:1).eq.' ') go to 50
   nfiles=nfiles+1
   go to 40

50  continue
   if (nfiles.le.0) then
     write(*,1)
1   format(' Usage: '/8X,'SHOW [file-name [file-name ...]]')
     stop
   endif
```

C Loop over files and send them to the screen.

```
call set_up_terminal(5,6)
i=1
10  continue
   call extract_next(inline,i,ilen,fname)
   if (fname(1:1).eq.' ') go to 60
   istext = .false.
   open(unit=20,file=fname,iointent='input',mode='binary',
2     status='old',maxrecl=1,delimiter='include',err=100)
102 continue

C   If there is more than one file requested, write out the names.
   if (nfiles.gt.1) then
     call change_attributes(0,6)
     write(6,2) fname
2   format('File: ',A32,'<012>')
   endif
```



```

C Send the file to the screen.
  call show_file(20,istext,6)
  go to 10

100  continue
     istext = .true.
     open(unit=20,file=fname,status='old',recfm='datasensitive',
1     maxrecl=255,delimiter='include',pad='yes',err=101)
     go to 102
101  continue
     write(*,3) fname
3     format('Unable to locate file:  ',A32,' '<012>')
     go to 10

60   continue
     call change_attributes(0,6)
     return
     end

     program which

C Main program.

     character ans

     write(*,'(' Run (S)how or (T)est?  ')')
     read(*,'(A1)') ans
     if (ans.eq.'s'.or.ans.eq.'S') call show
     if (ans.eq.'t'.or.ans.eq.'T') call test
     end

```

VT320 Emulation "C"

```
/*
 * vt320_mode.c -
 *   VT320/VT100 mode demonstration.
 *
 * Initial version 01JUN90.
 *
 * Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
 * use, distribute, and modify this code is hereby granted without monetary
 * consideration, provided this copyright message is included in all such
 * derivative works. Standard disclaimers regarding merchantability,
 * suitability for a particular purpose, et cetera, all apply. This code is
 * distributed with no warranty whatsoever.
 */

#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <fcntl.h>
#include <termio.h>

/*
 * These routines can be used to condition a communication line for binary I/O
 * when writing a C program under DG AOS. It switches both input and output
 * to binary mode, which disables character translation, and unbuffers the
 * input.
 */

static int UnbufferedMode = 0;

void
set_unbuffered()
{
    if (UnbufferedMode)
        return;
    UnbufferedMode=1;
    fflush(stdout);
    if (freopen("@input","j",stdin)==NULL) {
        fputs("Unable to open your terminal for binary input.\r\n",stderr);
        exit(1);
    }
    setbuf(stdin,NULL);
    if (freopen("@output","k",stdout)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary output.\r\n");
        exit(2);
    }
}

void
set_buffered()
{
    if (!UnbufferedMode)
        return;
    UnbufferedMode=0;
    if (freopen("@input","r",stdin)==NULL) {
        fputs("Unable to open your terminal for normal input.\r\n",stderr);
        exit(1);
    }
    if (freopen("@output","w",stdout)==NULL) {
        fputs("Unable to open your terminal for normal output.\r\n",stderr);
    }
}
```

```

        exit(2);
    }
}
/*
 * This routine is required for AOS because it mistakenly uses LF for
 * end-of-line, rather than CR, and it assumes DG mode cursor controls.
 */
char *
afgets(str, len, fp)
char *str;
int len;
FILE *fp;
{
    int chr, ilen;

    if (fp != stdin)
        return fgets(str, len, fp);
    set_unbuffered();
    chr = fgetc(fp);
    if (chr == EOF)
        return NULL;
    ilen = 0;
    for (;;) {chr=fgetc(fp)} switch (chr) {
        case EOF:
        case '\r':
        case '\n':
            if (ilen != 0)
                *str = '\0';
            fputs("\r\n", stdout);
            fflush(stdout);
            return str;
        case '\10':
        case '\177':
            if (ilen == 0)
                break;
            fputs("\10 \10", stdout);
            str--;
            ilen--;
            break;
        default:
            if (ilen+1 >= len)
                break;
            fputc(chr, stdout);
            *str++ = chr;
            ilen++;
    }
    /*NOTREACHED*/
}

```

```

/*
 * These routines may be used on any system that uses ASCII to provide support
 * for a VT320 terminal in a C program.
 */

/*
 * Constants for use with routine below.
 */

#define DIM      1
#define REVERSE  2
#define UNDER   4
#define BLINK    8

static int old_attr=DIM;

static void
do_vt_attrs(attrs, strings, fp, pfirst)
int attrs, *pfirst;
char *strings[];
FILE *fp;
{
    int i, j;
    for(i=1, j=0; i<16; i<=&1, j++)
        if (i&attrs) {
            if (!*pfirst) fputc(';', fp);
            *pfirst=0;
            fputs(strings[j], fp);
        }
}

/*
 * Set the attributes to the given state with as few commands as possible.
 */

void
change_attributes(attr, fp)
int attr;
FILE *fp;
{
    int change, first;
    static char *vt_off[]={"1", "27", "24", "25"};
    static char *vt_on[]={"22", "7", "4", "5"};

    change=attr^old_attr;          /* get changes from current */
    if (change==0)                 /* if none, exit */
        return;

    first=1;
    fputs("\33[", fp);
    if (change&old_attr)            /* if any need clearing, do them */
        do_vt_attrs(change&old_attr, vt_off, fp, &first);
    if (change&attr)               /* if any need setting, do them */
        do_vt_attrs(change&attr, vt_on, fp, &first);
    fputc('m', fp);
    old_attr=attr;
}

```

```

/*
 * Example of VT320 Cursor Position (CUP) command.
 * 'l' is line number (1 to 24) and 'c' is column number (1 to 80).
 */

void
cursor_position(l,c,f)
int l,c;
FILE *f;
{
    fprintf(f,"\33[%d;%dH",l,c);
}

/*
 * Example of VT320 Horizontal and Vertical position (HVP) command
 * implementation.
 */

void
horizontal_vertical(l,c,f)
int l,c;
FILE *f;
{
    fprintf(f,"\33[%d;%df",l,c);
}

/*
 * Decode a CSI sequence returned from the terminal.
 */

#define ccs(c) (((c)&0x7F)>' '&&((c)&0x7F)<0x7F)?(c)&0x7F:'.')

void
ansi_decode(fp,termc,bufp,maxp)
FILE *fp;
char termc;
int *bufp,*maxp;
{
    int ich;
    char tbuf[256],*tp;

/* Get a CSI header first */
    if ((ich=fgetc(fp))!=0x9B) {
        if (ich!=0x1B) {
            fprintf(stderr,"Error while waiting for CSI from terminal. Got
%c (%02X).\r\n",
                    ccs(ich),ich);
            exit(21);
        }
        if ((ich=fgetc(fp))!=0x5B) {
            fprintf(stderr,"Error while waiting for CSI from terminal. Got
<ESC>%c (1B %02X).\r\n",
                    ccs(ich),ich);
            exit(22);
        }
    }

/* Read in an ANSI sequence */
    tp=tbuf;
    while ((ich=fgetc(fp))!=EOF)
        if ((ich&0x60)==0x20) *tp++=ich;
        else break;
}

```

```

/* Check the terminating character */
    if (ich!=termc) {
        fprintf(stderr,"Error while parsing CSI string from terminal.
Unexpected terminator:  %c (%02X).\r\n",
                        ccs(ich),ich);
        exit(23);
    }

/* Parse decimal arguments in the string */
    *tp=0;
    tp=tbuf;
    for (ich=0;tp&&ich<*maxp;ich++) {
        *bufp+=atoi(tp);
        tp=strchr(tp,',');
        if (tp) tp++;
    }
    if (tp) fputs("Extra arguments returned from terminal ignored.\r\n",stderr);
    *maxp=ich;
}

/*
 * Example testing program. This will send the cursor to all VT320 screen
 * addresses and make sure that the cursor actually gets there.
 */

void
test(func,fin,fout)
void (*func)(/* int,int,FILE * */); /* DG C doesn't like ANSI prototypes */
FILE *fin,*fout;
{
    int l,c,lc[2],nc;

    for (l=1;l<25;l++) /* for all lines */
        for (c=1;c<81;c++) { /* and columns */
            (*func)(l,c,fout); /* go to position */
            fputs("\33[6n",fout); /* read cursor pos */
            nc=sizeof(lc)/sizeof(*lc);
            ansi_decode(fin,'R',lc,&nc); /* read in data */
            if (nc!=2) {
                fprintf(stderr,"Expected 2 parms returned, got
%d.\r\n",nc);
                exit(4);
            }
            if (lc[1]!=c || lc[0] != l) {
                fprintf(stderr,"Went to %d,%d got
%d,%d\r\n",l,c,lc[0],lc[1]);
                exit(3);
            }
        }
}

/*
 * Driver for above testing routine. This will call the above test routine and
 * direct it to use Cursor Position and Horz/Vert Position.
 */

int
run_test(argc,argv)
int argc;
char **argv;
{
    set_unbuffered();
}

```

```

/* Test with CUP command */
fputs("\33[2J\33[HTesting Cursor Position (CUP) ...\r\n",stdout);
test(cursor_position,stdin,stdout);

/* Test with HVP command */
fputs("\33[2J\33[HTesting Horizontal and Veritical Position (HVP)
...\r\n",stdout);
test(horizontal_vertical,stdin,stdout);

fputs("\r\nDone.",stdout);
return 0;
}

/*
 * Read binary information from 'fp' and output it in PINK style using VT320
 * commands to 'f'.
 */

#define linend(chr)      { change_attributes(DIM,f); fputc(chr,f); ch='\n'; }

void
show_file(fp,f)
FILE *fp,*f;
{
    int ch,chlast=0,attrs;

    while ((ch=fgetc(fp))!=EOF) {
        attrs=0;
        if (ch > 127) {
            attrs = UNDER;          /* if upper bit set */
            ch -= 128;              /* then underscore on */
            if (ch < 32) {
                attrs |= DIM;      /* if control character */
                ch+='@';          /* then character is dim */
            }
            if (ch == 127) {
                attrs |= DIM;
                ch = '~';
            }
        }
        else {
            if (ch == 127) {
                attrs = DIM;      /* if control character */
                ch = '~';        /* then character is dim */
            }
            if (ch < 32) {
                attrs = DIM;
                switch (ch) {      /* check for special end-of-line */
                    case '\n':
                        if (chlast==' ') linend('|');
                        break;
                    case '\r':
                        linend('<');
                        break;
                    case '\14':
                        linend('*');
                        break;
                    default:
                        ch+=64;
                }
            }
        }
        change_attributes(attrs,f); /* set new attributes */
        fputc(ch,f);                /* send out the character */
        if (ch=='\n') fputc('\r',f); /* if LF, do CR also */
    }
}

```

```

        chlast=ch;
    }
    fclose(fp);
}

/*
 * Main routine for example code above. This program will read any number of
 * input files and show the output in PINK format.
 */

int
show(argc,argv)
int argc;
char **argv;
{
    FILE *fp;

    if (argc<2) {
        fputs("Usage:\r\n\tSHOW [file-name [file-name ...]]\r\n",stderr);
        exit(0);
    }
    set_unbuffered();
    while (*++argv) /* for each file in list */
        if (access(*argv,0))
            fprintf(stderr,"%s does not exist!\r\n",*argv);
        else if (access(*argv,4))
            fprintf(stderr,"Insufficient access rights on %s!\r\n",*argv);
        else if ((fp=fopen(*argv,"r"))==NULL)
            fprintf(stderr,"Cannot open %s!\r\n",*argv);
        else {
            if (argc>2) { /* if 2 or more files, give names */
                change_attributes(0,stdout);
                printf("\r\nFile: %s\r\n",*argv);
            }
            show_file(fp,stdout);
        }
    change_attributes(DIM,stdout);
    return 0;
}

/* Main program */

int
main(argc,argv)
int argc;
char **argv;
{
    char str[256];

    printf("\r\nRun (S)how or (T)est? ");
    if (afgets(str,256,stdin)) switch(tolower(*str)) {
        case 's': return show(argc,argv);
        case 't': return run_test(argc,argv);
    }
    return 1;
}

```


VT320 Emulation "Fortran 77"

```
C vt320_mode.f77 -
C   VT320/VT100 mode demonstration.
C
C Initial version 01JUN90.
```

```
C
C Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
C use, distribute, and modify this code is hereby granted without monetary
C consideration, provided this copyright message is included in all such
C derivative works. Standard disclaimers regarding merchantability,
C suitability for a particular purpose, et cetera, all apply. This code is
C distributed with no warranty whatsoever.
```

```
C This routine can be used to condition a communication line for binary I/O
C when writing an F77 program under DG AOS. It switches both input and output
C to binary mode, which disables character translation, and unbuffers the
C input.
```

```
      subroutine set_up_terminal(iin,iout)
      integer iin,iout

      integer old_attr
      common /VTTERM/ old_attr

      open(unit=iin,file='@input',screenedit='no',maxrecl=1,
1       delimiter='include',force='yes',iointent='input',
2       mode='binary',err=100)
      open(unit=iout,file='@output',carriagecontrol='none',
1       screenedit='no',force='yes',iointent='output',mode='binary',
2       err=100)
      old_attr=1
      return

100   continue
      write(*,1)
1     format(' Unable to open your terminal as a binary device.')
      stop
      end

      subroutine wstr(str,iout)
      character*(*) str
      integer iout

      write(iout,'(A)') str
      return
      end
```

```
C These routines may be used on any system that uses ASCII to provide support
C for a VT320 or compatible terminal in an F77 program.
```

```
      subroutine write_decimal(num,iout)
      integer num,iout
```

```
C This routine will send out a generic decimal number without space padding.
```

```
      intrinsic char,ichar
      character*20 str
      integer a,i,nu
```

```

C If number is negative, send out a leading minus sign.
  nu = num
  if (num.lt.0) then
    nu = -nu
    write(iout,(''-'))
  endif

C Extract digits from number.
  i = 20
10  continue
    a = nu
    nu = nu/10
    str(i:i) = char(a-10*nu+ichar('0'))
    i=i-1
    if (i.gt.0 .and. nu.ne.0) go to 10

C Write out digit string.
  call wstr(str(i+1:20),iout)
  return
end

  subroutine cursor_position(l,c,iout)
  integer l,c,iout

C Example of VT320 Cursor Position (CUP) command.
C 'l' is line number (1 to 24) and 'c' is column number (1 to 80).

  write(iout,(' '<033>['])
  call write_decimal(l,iout)
  write(iout,(';'))
  call write_decimal(c,iout)
  write(iout,('H'))
  return
end

  subroutine horizontal_vertical(l,c,iout)
  integer l,c,iout

C Example of VT320 Horizontal and Vertical position (HVP) command
C implementation.

  write(iout,(' '<033>['])
  call write_decimal(l,iout)
  write(iout,(';'))
  call write_decimal(c,iout)
  write(iout,('f'))
  return
end

  integer function atoi(buf,start,end)
  character(*) buf
  integer start,end

```

C Convert character string to integer.

```
intrinsic ichar
character chr
integer i,val

i = start
val = 0
10 continue
chr = buf(i:i)
if (chr.lt.'0'.or.chr.gt.'9') go to 20
val = val*10
val = val+ichar(chr)-ichar('0')
i = i+1
20 if (i.le.end) go to 10
continue
atoi=val
return
end

character function ccs(c)
character c
```

C Insure that character given is printable and return.

```
intrinsic ichar, char
integer ich

ich=ichar(c)
if (ich.ge.128) ich=ich-128
if (ich.lt.32) ich=ichar('.')
ccs=char(ich)
return
end

subroutine ansi_decode(termc,buf,maxb,iin)
character termc
integer maxb,iin
integer buf(maxb)
```

C Decode a CSI sequence returned from the terminal.

```
intrinsic ichar
external ccs,atoi
character ccs
integer atoi
character in
character*128 tbuf
integer tp,ich,i,j
```

C Get a character.

```
read(iin,1,end=100,err=100) in
1 format(A1)
```

```

C Check for CSI header.
  if (in.ne.'<233>') then
    if (in.ne.'<033>') then
      write(*,2) ccs(in),ichar(in)
2      format('Error while waiting for CSI from terminal. Got ',
1        A1,' (' ,Z2.2,') .<015>')
      stop
    endif
    read(iin,1,end=100,err=100) in
    if (in.ne.'[') then
      write(*,3) ccs(in),ichar(in)
3      format('Error while waiting for CSI from terminal. Got '
1        5H<ESC>,A1,' (1B ',Z2.2,') .<012><015>')
      stop
    endif
  endif

C Read in the ANSI string.
  tp = 0
10  continue
  read(iin,1,end=200,err=200) in
  ich = ichar(in)
  if (ich.lt.32 .or. ich.gt.63) go to 20
  tp = tp+1
  tbuf(tp:tp) = in
  go to 10
20  continue

C Check terminator character.
  if (in.ne.termc) then
    write(*,4) ccs(in),ich
4    format('Error while parsing CSI from terminal. Unexpected ',
1      'terminator: ',A1,' (' ,Z2.2,') .<012><015>')
    stop
  endif

```

```

C Convert decimal arguments.
  i = 0
  j = 0
30  continue
    i = i+1
    if (i.gt.tp .or. j.ge.maxb) go to 50
    j = j+1
    buf(j) = atoi(tbuf,i,tp)
40  continue
    if (i.gt.tp) go to 50
    if (tbuf(i:i).eq.(';')) go to 30
    i = i+1
    go to 40
50  continue

    if (i.le.tp) write(*,5)
5   format('Extra arguments returned from terminal ignored.<012><015>')
    maxb = j
    return

100 continue
    write(*,6)
6   format('End of file on input while waiting for CSI.<012><015>')
    stop

200 continue
    write(*,7)
7   format('End of file on input while reading CSI string.<012><015>')
    stop
    end

    subroutine run_test(func,iin,iout)
    logical func
    integer iin,iout

C Example testing program. This will send the cursor to all VT320 screen
C addresses and make sure that the cursor actually gets there.

    integer l,c,rc,rl
    integer nc,lc(2)

C Loop over all rows and columns.
    do 10 l=1,24
    do 10 c=1,80

C Send cursor to position
    rl=l
    rc=c
    if (func) call cursor_position(rl,rc,iout)
    if (.not.func) call horizontal_vertical(rl,rc,iout)

```

```

C Read cursor position back
write(iout,1)
1   format('<033>[6n')
    nc=2
    call ansi_decode('R',lc,nc,iin)
    if (nc.ne.2) then
        write(*,2) nc
2   format('Expected 2 parms returned, got ',i2,'.<012><015>')
        stop
    endif
    if (lc(2).ne.c .or. lc(1).ne.1) then
        write(*,3) 1,c,lc(1),lc(2)
3   format('Went to ',i2,',',i2,' and got ',i2,',',i2,'.<012><015>')
        stop
    endif
10  continue
    return
    end

subroutine clear_screen(iout)
integer iout

write(iout,'(' '<033>[2J<033>[H''')
return
end

subroutine test

C Driver for above testing routine. This will call the above test routine and
C direct it to use Cursor Position and Horizontal/Vertical Position.

    call set_up_terminal(5,6)
    call clear_screen(6)
    write(*,1)
1   format('Testing Cursor Position (CUP) ...<015>')
    call run_test(.false.,5,6)
    call clear_screen(6)
    write(*,2)
2   format('Testing Horizontal/Veritical Position (HVP) ...<015>')
    call run_test(.true.,5,6)
    return
    end

subroutine send_attrs(attrs,aary,first,iout)
integer attrs,iout,aary(4)
logical first

C Send the indicated attributes. (Internal routine.)

intrinsic iand
integer i,j

i=1
do 10 j=1,4
if (iand(i,attrs).ne.0) then
    if (.not.first) write(iout,'('';''')
    call write_decimal(aary(j),iout)
    first=.false.
endif
i=i*2
10  continue
    return
    end

subroutine change_attributes(attr,iout)
integer attr,iout

```

C Set the attributes to the given state with as few commands as possible.

```
intrinsic ieor,iand
integer old_attr
common /VTTERM/ old_attr
integer vt_off(4),vt_on(4),temp
logical first
data vt_off/1,27,24,25/
data vt_on/22,7,4,5/
```

C Get attributes that differ. If none, then return.

```
temp=ieor(attr,old_attr)
if (temp.eq.0) return

first=.true.
write(iout,'('<033>[''')')
call send_attrs(iand(temp,old_attr),vt_off,first,iout)
call send_attrs(iand(temp,attr),vt_on,first,iout)
write(iout,'(''m'')')
old_attr=attr
return
end
```

```
subroutine show_file(fin,istext,iout)
integer fin,iout
logical istext
```

C Read binary information from fin and output it in PINK style using VT320
C commands to iout.

C This program runs very slowly since the output is unbuffered. This is done
C deliberately since this is a pedagogic example.

```
intrinsic char,ichar
integer dim,reverse,under,blink
data dim,reverse,under,blink/1,2,4,8/
integer attrs,ich,ipos
character ch,chlast
character*255 chstr

chlast='a'
ipos = 0
10 continue
if (istext) then
  if (ipos.eq.0) read(fin,3,end=100,err=200) chstr
3  format(A255)
  ipos = ipos+1
  ich = ichar(chstr(ipos:ipos))
  if (ich.eq.10.or.ipos.eq.255) ipos=0
else
  read(fin,1,end=100,err=200) ch
1  format(A1)
  ich = ichar(ch)
endif
attrs=0
if (ich .ge. 128) then
  attrs=under
  ich=ich-128
  if (ich .lt. 32) then
    attrs=attrs+dim
    ich=ich+64
  else if (ich .eq. 127) then
    attrs=attrs+dim
    ich=ichar('~')
  endif
else
  if (ich .eq. 127) then
```

```

    attrs=dim
    ich=ichar('~')
else if (ich .lt. 32) then
    attrs=dim
    if (ich .eq. 10) then
        if (chlast .eq. ' ') then
            call change_attributes(dim,iout)
            write(iout,7(''|'''))
        endif
    else if (ich .eq. 13) then
        call change_attributes(dim,iout)
        write(iout,7(' '<074>'))
        ich=10
    else if (ich .eq. 12) then
        call change_attributes(dim,iout)
        write(iout,7(' '*'))
        ich=10
    else
        ich=ich+64
    endif
endif
endif
call change_attributes(attrs,iout)
ch=char(ich)
write(iout,1) ch
if (ich.eq.10) write(iout,1) char(13)
chlast=ch
go to 10
200 continue
write(*,2)
2 format('Unexpected error while reading file!<012><015>')
100 continue
return
end

```

```

subroutine extract_next(inline,i,ilen,fname)
character*80 inline
integer i,ilen
character*32 fname

```

C Internal routine -- extract next file name from input line.

```

integer j
fname=' '
j=1
10 continue
if (inline(i:i).gt.' ') go to 20
i=i+1
if (i.gt.ilen) return
go to 10
20 continue
fname(j:j)=inline(i:i)
j=j+1
i=i+1
if (i.gt.ilen.or.j.gt.32) return
if (inline(i:i).gt.' ') go to 20
return
end

subroutine show

```


C Main routine for example code above. This program will read any number of C input files and show the output in PINK format.

```

integer nfiles,i
character*32 fname
character*80 inline
logical  istext

```

C Prompt for file names.

```

write(*,'('' <015>Enter names of files to show: ''')
read(*,'(A80)') inline
write(*,'('' <015>'')')

```

C Count the number of files.

```

do 20 ilen=80,1,-1
  if (inline(ilen:ilen).ne.' ') go to 30
20  continue
30  continue
   nfiles=0
   i=1
40  continue
   call extract_next(inline,i,ilen,fname)
   if (fname(1:1).eq.' ') go to 50
   nfiles=nfiles+1
   go to 40

50  continue
   if (nfiles.le.0) then
     write(*,1)
1   format(' Usage: /8X, 'SHOW [file-name [file-name ...]]')
     stop
   endif

```

C Loop over files and send them to the screen.

```

call set_up_terminal(5,6)
i=1
10  continue
   call extract_next(inline,i,ilen,fname)
   if (fname(1:1).eq.' ') go to 60
   istext = .false.
   open(unit=20,file=fname,iointent='input',mode='binary',
2    status='old',maxrecl=1,delimiter='include',err=100)
102 continue

```

C If there is more than one file requested, write out the names.

```

if (nfiles.gt.1) then
  call change_attributes(0,6)
  write(6,2) fname
2  format('<015>File: ',A32,'<012><015>')
endif

```

```

C Send the file to the screen.
  call show_file(20,istext,6)
  go to 10

100  continue
     istext = .true.
     open(unit=20,file=fname,status='old',recfm='datasensitive',
1     maxrecl=255,delimiter='include',pad='yes',err=101)
     go to 102
101  continue
     write(*,3) fname
3    format('<015>Unable to locate file:  ''',A32,'''<012><105>')
     go to 10

60   continue
     call change_attributes(0,6)
     return
     end

program which
character ans

write(*,('' <015>Run (S)how or (T)est?  ''')
read(*,('A1')) ans
if (ans.eq.'s'.or.ans.eq.'S') call show
if (ans.eq.'t'.or.ans.eq.'T') call test
end

```

VT52 Emulation "C"

```
/*
 * vt52_mode.c -
 * This program permits the user to draw lines using the VT52 graphics
 * mode (line drawing) characters. This program also demonstrates a
 * set of routines for use with the DEC VT52.
 *
 * Usage:
 * The following keys are valid (case is not significant):
 *
 *      Key      Description
 *      M        Set movement to non-destructive, non-drawing.
 *      D        Set movement to drawing.
 *      E        Set movement to erasing.
 *      S        Save screen data in a file.
 *      Q        Quit
 *
 * The arrow keys move the cursor about the screen, and optionally draw or
 * erase lines.
 *
 * Initial version 01JUN90 by James Carlson.
 *
 * Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
 * use, distribute, and modify this code is hereby granted without monetary
 * consideration, provided this copyright message is included in all such
 * derivative works. Standard disclaimers regarding merchantability,
 * suitability for a particular purpose, et cetera, all apply. This code is
 * distributed with no warranty whatsoever.
 */

#include <stdio.h>

/*
 * This routine can be used to condition a communication line for binary I/O
 * when writing a C program under DG AOS. It switches both input and output
 * to binary mode, which disables character translation, and unbuffers the
 * input.
 */

void
set_up_terminal()
{
    if (freopen("@input","j",stdin)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary input.\n");
        exit(1);
    }
    setbuf(stdin,NULL);
    if (freopen("@output","k",stdout)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary output.\n");
        exit(2);
    }
}
```

```

/*
 * This routine is required for AOS because it mistakenly uses LF for
 * end-of-line, rather than CR, and it assumes DG mode cursor controls.
 */
char *
afgets(str, len, fp)
char *str;
int len;
FILE *fp;
{
    int chr, ilen;

    if (fp != stdin)
        return fgets(str, len, fp);
    chr = fgetc(fp);
    if (chr == EOF)
        return NULL;
    ilen = 0;
    for (;;) chr=fgetc(fp) switch (chr) {
        case EOF:
        case '\r':
        case '\n':
            if (ilen != 0)
                *str = '\0';
            fputs("\r\n", stdout);
            fflush(stdout);
            return str;
        case '\10':
        case '\177':
            if (ilen == 0)
                break;
            fputs("\10 \10", stdout);
            str--;
            ilen--;
            break;
        default:
            if (ilen+1 >= len)
                break;
            fputc(chr, stdout);
            *str++ = chr;
            ilen++;
    }
    /*NOTREACHED*/
}

```

```

/*
 * The first section of this program provides support routines for moving
 * the cursor around the screen.
 */

int crow,ccol;

/* Go to a specific row and column */
void
VT52_set_position(row,col,fp)
int row,col;
FILE *fp;
{
    if (row==0 && col==0)
        fputs("\33H",fp);      /* special case -- home */
    else {
        int absflag=0;

        /* Set flag if column move is expensive */
        if (col!=ccol && col!=0)
            absflag=1;

        /* Check for inexpensive row movement */
        if (row==crow+1 && absflag==0)
            fputc('\12',fp);
        else if (row==crow-1 && col==ccol)
            fputs("\33A",fp);
        else if (row!=crow)
            absflag=2;

        /* If movement not expensive and column changes, do column move */
        if (absflag!=2 && col!=ccol)
            if (col==0)
                fputc('\15',fp);
            else if (col==ccol+1)
                fputs("\33C",fp);
            else if (col==ccol-1)
                fputc('\10',fp);
            else
                absflag=2;

        /* If movement is expensive, then do absolute row/col positioning */
        if (absflag==2)
            fprintf(fp,"\33Y%c%c",row+' ',col+' ');
    }
    crow = row;
    ccol = col;
}

/* Clear the screen and home the cursor */
void
VT52_clear(fp)
FILE *fp;
{
    VT52_set_position(0,0,fp);
    fputs("\33J",fp);
}

/* Put into graphics (line drawing) mode */
void
VT52_graphics_mode(fp)
FILE *fp;
{
    fputs("\33F",fp);
}

```

```

/* Exit graphics (line drawing) mode */
void
VT52_normal_mode(fp)
FILE *fp;
{
    fputs("\33G", fp);
}

/*
 * The second part of this program implements the line-drawing algorithm for
 * the special graphics characters.
 */

#define NROWS    24
#define NCOLS    80

/* The screen array contains the encoded line connection information */
char screen[NROWS][NCOLS];

/* These constants define bits used for connection information */
#define LEFTCON    1
#define RIGHTCON   2
#define UPCON      4
#define DOWNCON    8

/* This string defines the characters used to show connectivity above */
char lines[] = " qqxjmvxklwxutn";

/* This flag indicates whether drawing, moving or erasing */
int Drawing;

/* Put connected line drawing character on screen at current position */
void
show_connection(fp)
FILE *fp;
{
    fputc(lines[screen[crow][ccol]], fp);    /* put connected line char */
    if (ccol != NCOLS-1)                    /* VT52 has no autowrap */
        ccol++;
}

/*
 * Clear the screen, then loop through all of the screen data and produce
 * a minimized sequence to produce that image on the screen.
 */
void
redraw_screen(fp)
FILE *fp;
{
    int row, col;

    VT52_clear(fp);
    VT52_graphics_mode(fp);
    for (row=0; row<NROWS; row++)           /* loop over rows */
        for (col=0; col<NCOLS; col++)      /* ... and columns */
            if (screen[row][col]) {        /* if non-blank */
                VT52_set_position(row, col, fp); /* go there */
                show_connection(fp);        /* show data */
            }
}

```

```

/* Get a file name from the user and save the screen data there */
void
save_file()
{
    char temp[128];
    FILE *fp;
    int rowsave, colsave;

    rowsave=crow;
    colsave=ccol;
    VT52_clear(stdout);
    VT52_normal_mode(stdout);
    printf("File name for data: ");
    afgets(temp, 128, stdin);
    if (*temp=='\0')
        printf("No data saved.");
    else if ((fp=fopen(temp, "w"))==NULL)
        printf("Unable to open %s for output.", temp);
    else {
        redraw_screen(fp);
        VT52_normal_mode(fp);
        VT52_set_position(NROWS-2, 0, fp);
        close(fp);
        printf("Saved screen data in %s.", temp);
    }
    printf(" Press <CR> to continue.\r\n");
    afgets(temp, 128, stdin);
    redraw_screen(stdout);
    VT52_set_position(rowsave, colsave, stdout);
}

/* Set the connection type for the current cell and update the VT52 screen */
void
set_connection(conn)
char conn;
{
    int colsave;

    colsave=ccol;
    switch (Drawing) {
        case 1: /* Drawing */
            if (screen[crow][ccol] & conn)
                return; /* already set */
            screen[crow][ccol] |= conn;
            show_connection(stdout);
            break;
        case -1: /* Erasing */
            if ((screen[crow][ccol] & conn)==0)
                return; /* already clear */
            screen[crow][ccol] &= ~conn;
            show_connection(stdout);
            break;
    }
    VT52_set_position(crow, colsave, stdout); /* put cursor back */
}

```

```

/* Handle <ESC><char> keyboard sequences (see dispatch below) */
void
esc_dispatch()
{
    switch (fgetc(stdin)) {
        case 'A':
            set_connection(UPCON);           /* leave through top */
            if (crow==0) break;
            VT52_set_position(crow-1,ccol,stdout);
            set_connection(DOWNCON);        /* enter through bot */
            break;
        case 'B':
            set_connection(DOWNCON);        /* leave through bot */
            if (crow==NROWS-1) break;
            VT52_set_position(crow+1,ccol,stdout);
            set_connection(UPCON);         /* enter through top */
            break;
        case 'C':
            set_connection(RIGHTCON);       /* leave through right */
            if (ccol==NCOLS-1) break;
            VT52_set_position(crow,ccol+1,stdout);
            set_connection(LEFTCON);       /* enter through left */
            break;
        case 'D':
            set_connection(LEFTCON);        /* leave through left */
            if (ccol==0) break;
            VT52_set_position(crow,ccol-1,stdout);
            set_connection(RIGHTCON);      /* enter through right */
            break;
    }
}

/* Dispatch user's input */
void
dispatch()
{
    for (;;) switch (fgetc(stdin)) {
        case '\33':
            esc_dispatch();                 /* ESC sequence */
            break;
        case '\15':
            VT52_set_position(crow,0,stdout);
            break;
        case 'm': case 'M':
            Drawing = 0;                    /* Movement mode */
            break;
        case 'd': case 'D':
            Drawing = 1;                    /* Drawing mode */
            break;
        case 'e': case 'E':
            Drawing = -1;                   /* Erasing mode */
            break;
        case 's': case 'S':
            save_file();                    /* Save screen in file */
            break;
        case 'q': case 'Q':
            return;                          /* Quit */
    }
    /*NOTREACHED*/
}

```



```

/* Initialize internal data for program */
void
initialize()
{
    int i;
    char *sp;

    sp=screen;
    for (i=NROWS*NCOLS;i>0;i--) /* initialize connection array */
        *sp++ = 0;
    Drawing=1; /* set drawing mode */
    VT52_clear(stdout);
    VT52_graphics_mode(stdout);
}

int
main(argc,argv)
int argc;
char **argv;
{
    set_up_terminal(); /* set to unbuffered binary */
    initialize(); /* setup internal data */
    dispatch(); /* handle user's commands */
    VT52_clear(stdout); /* reset VT52 modes before exit */
    VT52_normal_mode(stdout);
    return 0;
}

```

VT52 Emulation "Fortran 77"

```
C vt52_mode.f77 -
C   This program permits the user to draw lines using the VT52 graphics
C   mode (line drawing) characters. This program also demonstrates a
C   set of routines for use with the DEC VT52.
C
C Usage:
C   The following keys are valid (case is not significant):
C       Key      Description
C       M       Set movement to non-destructive, non-drawing.
C       D       Set movement to drawing.
C       E       Set movement to erasing.
C       S       Save screen data in a file.
C       Q       Quit
C   The arrow keys move the cursor about the screen, and optionally draw or
C   erase lines.
C
C Initial version 01JUN90.
C
C Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
C use, distribute, and modify this code is hereby granted without monetary
C consideration, provided this copyright message is included in all such
C derivative works. Standard disclaimers regarding merchantability,
C suitability for a particular purpose, et cetera, all apply. This code is
C distributed with no warranty whatsoever.
C
C This routine can be used to condition a communication line for binary I/O
C when writing an F77 program under DG AOS. It switches both input and output
C to binary mode, which disables character translation, and unbuffers the
C input.
C
C   subroutine set_up_terminal
C
C   integer stdin,stdout
C   common /prog/ stdin,stdout
C
C   open(unit=stdin,file='@input',screenedit='no',maxrecl=1,
1   delimiter='include',force='yes',iointent='input',
2   mode='binary',err=100)
C   open(unit=stdout,file='@output',carriagecontrol='none',
1   screenedit='no',force='yes',iointent='output',mode='binary',
2   err=100)
C   return
100  continue
C   write(*,('Unable to open your terminal as a binary device.'))
C   stop
C   end
```

C
 C The first section of this program provides support routines for moving
 C the cursor around the screen.
 C

```

character function getc(iin)
integer iin

character ch

read(iin,1,err=100,end=100) ch
1 format(A1)
getc=ch
return
100 continue
getc=' '
return
end

```

```

subroutine VT52_set_position(row,col,iout)
integer row,col,iout

```

C Send cursor to an absolute row/column location.

```

integer stdin,stdout,output,crow,ccol
common /prog/ stdin,stdout,output,crow,ccol

integer absflag
integer*2 cblank
data cblank/2H /
integer*2 c1,c2

if (row.eq.0 .and. col.eq.0) then
C   Special case -- home
   write(iout,' (' '<033>H'' )')
else
   absflag=0
C   Set flag if column move is expensive.
   if (col.ne.ccol .and. col.ne.0) absflag=1
C   Check for inexpensive row movement.
   if (row.eq.crow+1 .and. absflag.eq.0) then
     write(iout,' (' '<012>'' )')
   else if (row.eq.crow-1 .and. col.eq.ccol) then
     write(iout,' (' '<033>A'' )')
   else
     if (row.ne.crow) absflag=2
   endif
C   If movement not expensive and column changes, do column move.
   if (absflag.ne.2 .and. col.ne.ccol) then
     if (col.eq.0) then
       write(iout,' (' '<015>'' )')
     else if (col.eq.ccol+1) then
       write(iout,' (' '<033>C'' )')
     else if (col.eq.ccol-1) then
       write(iout,' (' '<010>'' )')
     else
       absflag=2
     endif
   endif
endif
endif

```

```

C      If movement is expensive, then do absolute row/col positioning.
      if (absflag.eq.2) then
        c1 = cblank + row*256
        c2 = cblank + col*256
        write(iout,1) c1,c2
1      format('<033>Y',2A1)
      endif
      endif
      crow = row
      ccol = col
      return
      end

      subroutine VT52_clear(iout)
      integer iout

C Clear the screen and home the cursor.
      call VT52_set_position(0,0,iout)
      write(iout,' ('<033>J''')
      return
      end

      subroutine VT52_graphics_mode(iout)
      integer iout

C Put into graphics (line drawing) mode.
      write(iout,' ('<033>F''')
      return
      end

      subroutine VT52_normal_mode(iout)
      integer iout

C Exit graphics (line drawing) mode.
      write(iout,' ('<033>G''')
      return
      end

C
C The second part of this program implements the line-drawing algorithm for
C the special graphics characters.
C

      subroutine show_connection(iout)
      integer iout

C Put connected line drawing character on screen at current position.

      integer NROWS,NCOLS
      parameter (NROWS=24)
      parameter (NCOLS=80)

      integer stdin,stdout,output,crow,ccol,drawing,screen(NCOLS,NROWS)
      common /prog/ stdin,stdout,output,crow,ccol,drawing,screen

C This string defines the characters used to show connectivity.
      character*16 lines
      data lines/' qqxjmvxklwxutn'/

      integer i

C Put connected line character.
      i = screen(ccol+1,crow+1)+1
      write(iout,' (A1)') lines(i:i)

```

```

C VT52 has no autowrap.
  if (ccol .ne. NCOLS-1) ccol = ccol+1

  return
end

subroutine redraw_screen(iout)
integer iout

C Clear the screen, then loop through all of the screen data and produce
C a minimized sequence to produce that image on the screen.

  integer NROWS,NCOLS
  parameter (NROWS=24)
  parameter (NCOLS=80)

  integer stdin,stdout,output,crow,ccol,drawing,screen(NCOLS,NROWS)
  common /prog/ stdin,stdout,output,crow,ccol,drawing,screen

  integer row,col,r,c

C Reset the screen.
  call VT52_clear(iout)
  call VT52_graphics_mode(iout)

C Loop over rows and columns on screen.
  do 10 r=1,NROWS
  do 10 c=1,NCOLS

    row=r-1
    col=c-1
    if (screen(c,r).ne.0) then
C If non-blank in this position, then update data here
      call VT52_set_position(row,col,iout)
      call show_connection(iout)
    endif

10  continue
  return
end

subroutine save_file

C Get a file name from the user and save the screen data there.

  integer NROWS,NCOLS
  parameter (NROWS=24)
  parameter (NCOLS=80)

  integer stdin,stdout,output,crow,ccol,drawing,screen(NCOLS,NROWS)
  common /prog/ stdin,stdout,output,crow,ccol,drawing,screen

  character*64 temp
  integer rowsave,colsave

C Save current cursor position.
  rowsave=crow
  colsave=ccol

C Reset the screen and put back to normal text mode.
  call VT52_clear(stdout)
  call VT52_normal_mode(stdout)

```

```

C Prompt user for name of file for output data.
  write(*,'(''File name for data: ''')')
  read(11,1) temp
1   format(A64)
  write(*,'('<015>'')')

C Open file and save data.
  if (temp(1:1).eq.' ') then
    write(*,'(''No data saved. ''')')
  else
    open(unit=output,file=temp,carriagecontrol='none',
1     screenedit='no',force='yes',iointent='output',mode='binary',
2     err=100)

C   Send controls to file.
  call redraw_screen(output)

C   Put a mode exit at end of file.
  call VT52_normal_mode(output)
  call VT52_set_position(NROWS-2,0,output)
  close(iout)

  write(*,2) temp
2   format('Saved screen data in: ',A64,'<015><012>')
  endif
200  continue

  write(*,'(''Press CR to continue.<015><012>'')')
  read(11,1) temp

C Put user's screen back up.
  call set_up_terminal
  call redraw_screen(stdout)
  call VT52_set_position(rowsave,colsave,stdout)
  return

100  continue
  write(*,3) temp
3   format('Unable to open file for output: ',A64,'<015><012>')
  go to 200
  end

  subroutine set_connection(conn)
  integer conn

C Set the connection type for the current cell and update the VT52 screen

  intrinsic iand,ior,ixor

  integer NROWS,NCOLS
  parameter (NROWS=24)
  parameter (NCOLS=80)

  integer stdin,stdout,output,crow,ccol,drawing,screen(NCOLS,NROWS)
  common /prog/ stdin,stdout,output,crow,ccol,drawing,screen

  integer colsave

  colsave=ccol
  if (drawing.eq.1) then
C   Is the flag already set?
    if (iand(screen(ccol+1,crow+1),conn).ne.0) return
    screen(ccol+1,crow+1) = ior(screen(ccol+1,crow+1),conn)
    call show_connection(stdout)
  else if (drawing.eq.-1) then
C   Is the flag already clear?

```

```

        if (iand(screen(ccol+1,crow+1),conn).eq.0) return
        screen(ccol+1,crow+1) = ixor(screen(ccol+1,crow+1),conn)
        call show_connection(stdout)
    endif

C Put cursor back in place.
    call VT52_set_position(crow, colsave, stdout)

    return
end

    subroutine esc_dispatch

C Handle <ESC><char> keyboard sequences (see dispatch below).

    external getc
    character getc

C These constants define bits used for connection information.
    integer LEFTCON,RIGHTCON,UPCON,DOWNCON
    parameter (LEFTCON=1)
    parameter (RIGHTCON=2)
    parameter (UPCON=4)
    parameter (DOWNCON=8)

    integer NROWS,NCOLS
    parameter (NROWS=24)
    parameter (NCOLS=80)

    integer stdin,stdout,output,crow,ccol
    common /prog/ stdin,stdout,output,crow,ccol

    character chr

    chr=getc(stdin)
    if (chr.eq.'A') then

C        Leave through top, then enter through bottom.
        call set_connection(UPCON)
        if (crow.ne.0) then
            call VT52_set_position(crow-1,ccol,stdout)
            call set_connection(DOWNCON)
        endif
    else if (chr.eq.'B') then

C        Leave through bottom, then enter through top.
        call set_connection(DOWNCON)
        if (crow.ne.NROWS-1) then
            call VT52_set_position(crow+1,ccol,stdout)
            call set_connection(UPCON)
        endif
    else if (chr.eq.'C') then

C        Leave through right, then enter through left.
        call set_connection(RIGHTCON)
        if (ccol.ne.NCOLS-1) then
            call VT52_set_position(crow,ccol+1,stdout)
            call set_connection(LEFTCON)
        endif
    else if (chr.eq.'D') then

C        Leave through left, then enter through right.
        call set_connection(LEFTCON)
        if (ccol.ne.0) then
            call VT52_set_position(crow,ccol-1,stdout)
            call set_connection(RIGHTCON)
        endif
    endif

```

```

endif
return
end

subroutine dispatch
C Dispatch user's input.

external getc
character getc

integer stdin, stdout, output, crow, ccol, drawing
common /prog/ stdin, stdout, output, crow, ccol, drawing

character chr

10  continue
chr=getc(stdin)
if (chr.eq.'<033>') then
    call esc_dispatch
else if (chr.eq.'<015>') then
    call VT52_set_position(crow,0,stdout)
else if (chr.eq.'m' .or. chr.eq.'M') then
    Drawing = 0
else if (chr.eq.'d' .or. chr.eq.'D') then
    Drawing = 1
else if (chr.eq.'e' .or. chr.eq.'E') then
    Drawing = -1
else if (chr.eq.'s' .or. chr.eq.'S') then
    call save_file
else if (chr.eq.'q' .or. chr.eq.'Q') then
    return
endif
go to 10
end

subroutine initialize
C Initialize internal data for program.

integer NROWS, NCOLS
parameter (NROWS=24)
parameter (NCOLS=80)

integer stdin, stdout, output, crow, ccol, drawing, screen (NCOLS, NROWS)
common /prog/ stdin, stdout, output, crow, ccol, drawing, screen

integer i, j

C Initialize connection array.
C The screen array contains the encoded line connection information.
do 10 i=1, NROWS
do 10 j=1, NCOLS
screen(j,i) = 0
10  continue

C Set drawing mode.
drawing=1

C Clear screen and enter line drawing mode
call VT52_clear(stdout)
call VT52_graphics_mode(stdout)

return
end

```



```
      program drawing
C This is the main program.

      integer stdin, stdout, output
      common /prog/ stdin, stdout, output

      stdin = 5
      stdout = 6
      output = 7

C Set to unbuffered binary communication.
      call set_up_terminal

C Setup internal data
      call initialize

C Handle user's commands
      call dispatch

C Reset VT52 modes before exit
      call VT52_clear(stdout)
      call VT52_normal_mode(stdout)

      end
```

Tektronix 4010 Emulation "C"

```
/*
 * tek_mode.c -
 * This program permits the user to draw lines using the Tek4010 emulation
 * mode. This program also demonstrates a set of routines for use with
 * the Tektronix 4010.
 *
 * Usage:
 * Use the following keys to enter points -
 *
 * Key      Description
 * L        Begin a line at current location
 * B        Begin a box at current location
 * T        Begin entering text here
 * D        Delete last object
 * S        Save objects in file
 * Q        Quit
 * SPACE    Finish operation (line, box) here
 * NewLine  Finish operation (line, box, text) here
 *
 * Initial version 01JUN90 by James Carlson.
 *
 * Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
 * use, distribute, and modify this code is hereby granted without monetary
 * consideration, provided this copyright message is included in all such
 * derivative works. Standard disclaimers regarding merchantability,
 * suitability for a particular purpose, et cetera, all apply. This code is
 * distributed with no warranty whatsoever.
 */

#include <stdio.h>

/*
 * This routine can be used to condition a communication line for binary I/O
 * when writing a C program under DG AOS. It switches both input and output
 * to binary mode, which disables character translation, and unbuffers the
 * input.
 */

void
set_up_terminal()
{
    if (freopen("@input","j",stdin)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary input.\n");
        exit(1);
    }
    setbuf(stdin,NULL);
    if (freopen("@output","k",stdout)==NULL) {
        fprintf(stderr,"Unable to open your terminal for binary output.\n");
        exit(2);
    }
}
```

```

/*
 * This routine is required for AOS because it mistakenly uses LF for
 * end-of-line, rather than CR, and it assumes DG mode cursor controls.
 */

char *
afgets(str, len, fp)
char *str;
int len;
FILE *fp;
{
    int chr, ilen;
    if (fp != stdin)
        return fgets(str, len, fp);
    chr = fgetc(fp);
    if (chr == EOF)
        return NULL;
    ilen = 0;
    for (;;) {chr=fgetc(fp)} switch (chr) {
        case EOF:
        case '\r':
        case '\n':
            if (ilen != 0)
                *str = '\0';
            fputs("\r\n", stdout);
            fflush(stdout);
            return str;
        case '\10':
        case '\177':
            if (ilen == 0)
                break;
            fputs("\10 \10", stdout);
            str--;
            ilen--;
            break;
        default:
            if (ilen+1 >= len)
                break;
            fputc(chr, stdout);
            *str++ = chr;
            ilen++;
    }
    /*NOTREACHED*/
}

typedef struct {
    FILE *fp;
    int xbeam, ybeam;
    enum {
        Initial, Alpha, Drawing
    } state;
} Device;

/* Convert two characters from terminal into 10 bit number */
int
tenbit(high, low)
char high, low;
{
    return ( ((high & 0x1F) << 5) + (low & 0x1F) );
}

/* Get a point on the screen from the user. */
char
getpoint(xp, yp, outd)

```

```

int *xp,*yp;
Device *outd;
{
    char buffer[4];
    int c,i;
/* Start cross-hair */
    fputs("\037\033\032",stdout);
/* Wait for key press */
    c=fgetc(stdin)&0x7F;
    for(i=0;i<4;i++) buffer[i]=fgetc(stdin)&0x7F;
    (*xp) = tenbit(buffer[0],buffer[1]);
    (*yp) = tenbit(buffer[2],buffer[3]);
    outd->state = Alpha;
    return c;
}

void
optimizesend(x,y,outd)
int x,y;
Device *outd;
{
    int hox,hoy,hoxb,hoyb;
    int lox,loy,loxb,loyb;
/* Extract 5 bit portions of the data */
    hox = x>>5;
    hoy = y>>5;
    lox = x&0x1F;
    loy = y&0x1F;
    if (outd->state == Initial) {
        hoxb = hox+1;
        hoyb = hoy+1;
        loxb = lox+1;
        loyb = loy+1;
    }
    else {
        hoxb = outd->xbeam>>5;
        hoyb = outd->ybeam>>5;
        loxb = outd->xbeam&0x1F;
        loyb = outd->ybeam&0x1F;
    }
/* Do optimized Tek coordinate transmission */
    if (hoy != hoyb) putc(hoy+0x20,outd->fp);
    if ((loy != loyb) || (hox != hoxb)) putc(loy+0x60,outd->fp);
    if (hox != hoxb) putc(hox+0x20,outd->fp);
    putc(lox+0x40,outd->fp);
    outd->xbeam=x;
    outd->ybeam=y;
}

void
darkto(x,y,outd)
int x,y;
Device *outd;
{
    if (x!=outd->xbeam || y!=outd->ybeam || outd->state!=Drawing) {
        putc('\035',outd->fp);
        optimizesend(x,y,outd);
        outd->state = Drawing;
    }
}

void
gotoalpha(outd)

```

```

Device *outd;
{
    if (outd->state != Alpha) {
        putc('\037', outd->fp);
        outd->state = Alpha;
    }
}

void
startstring(x, y, outd)
int x, y;
Device *outd;
{
    darkto(x, y, outd);
    gotoalpha(outd);
}

void
drawvector(xfrom, yfrom, xto, yto, outd)
int xfrom, yfrom, xto, yto;
Device *outd;
{
    darkto(xfrom, yfrom, outd);
    optimizesend(xto, yto, outd);
}

void
clearscreen(outd)
Device *outd;
{
    fputs("\033\014", outd->fp);
    outd->state = Initial;
}

typedef struct _item {
    struct _item *next;
    enum {
        None, Line, Box, Text
    } type;
    int xloc, yloc;
    union {
        char *textp;
        struct {
            int x2, y2;
        } x;
    } Item;
} Item;

Item *list = NULL;

#define New(x) ((x *) malloc(sizeof(x)))

void
draw_object(it, outd)
Item *it;
Device *outd;
{
    switch (it->type) {
        case Line:
            drawvector(it->xloc, it->yloc, it->x.x.x2, it->x.x.y2, outd);
            break;
        case Box:
            drawvector(it->xloc, it->yloc, it->xloc, it->x.x.y2, outd);
            drawvector(it->xloc, it->x.x.y2, it->x.x.x2, it->x.x.y2, outd);
            drawvector(it->x.x.x2, it->x.x.y2, it->x.x.x2, it->yloc, outd);
            drawvector(it->x.x.x2, it->yloc, it->xloc, it->yloc, outd);
    }
}

```

```

                break;
        case Text:
            startstring(it->xloc, it->yloc, outd);
            fputs(it->x.textp, outd->fp);
            break;
    }
}

void
redraw_screen(outd)
Device *outd;
{
    Item *temp;

    clearscreen(outd);
    for (temp=list; temp=temp->next)
        draw_object(temp, outd);
}

/* Get a file name from the user and save the screen data there */

void
save_file(outd)
Device *outd;
{
    char temp[128];
    Device file;

    clearscreen(outd);
    printf("File name for data: ");
    fgets(temp, 128, stdin);
    if (*temp=='\0')
        printf("No data saved.");
    else if ((file.fp=fopen(temp, "w"))==NULL)
        printf("Unable to open %s for output.", temp);
    else {
        redraw_screen(&file);          /* send controls to file */
        gotoalpha(&file);             /* put a mode exit at end */
        close(file.fp);
        printf("Saved screen data in %s.", temp);
    }
    printf(" Press <CR> to continue.\r\n");
    fgets(temp, 128, stdin);
    redraw_screen(outd);              /* put user's screen back up */
}

void
commit_object(it)
Item *it;
{
    Item *temp;

    temp = New(Item);
    *temp = *it;
    temp->next = list;
    list = temp;
}

void
dellast(outd)
Device *outd;
{
    Item *temp;

    if (list==NULL)
        return;
}

```

```

    temp = list;
    list = temp->next;
    if (temp->type==Text) free(temp->x.textp);
    free(temp);
    redraw_screen(outd);
}

void
gettext(it,outd)
Item *it;
Device *outd;
{
    char temp[128];

    startstring(it->xloc,it->yloc,outd);
    if (afgets(temp,128,stdin)!=NULL && temp[0]!='\0') {
        it->x.textp = strsave(temp);
        commit_object(it);
    }
}

static void
ptcopy(it)
Item *it;
{
    it->xloc = it->x.x.x2;
    it->yloc = it->x.x.y2;
}

/* Dispatch user's input */

void
dispatch(outd)
Device *outd;
{
    Item temp;

    temp.type = None;
    for (;;) switch (getpoint(&temp.x.x.x2,&temp.x.x.y2,outd)) {
        case '\15': /* CR */
            case ' ':
                if (temp.type != None) {
                    draw_object(&temp,outd);
                    commit_object(&temp);
                }
                ptcopy(&temp);
                break;
        case 'l': case 'L':
            ptcopy(&temp);
            temp.type = Line;
            break;
        case 'b': case 'B':
            ptcopy(&temp);
            temp.type = Box;
            break;
        case 't': case 'T':
            ptcopy(&temp);
            temp.type = Text;
            gettext(&temp,outd);
            temp.type = None; /* Can't duplicate item */
            break;
        case 'd': case 'D':
            dellast(outd);
            break;
        case 's': case 'S':

```

```

                save_file(outd);          /* Save screen in file */
                break;
        case 'q': case 'Q':
                return;                    /* Quit */
        }
        /*NOTREACHED*/
}

void
initialize(outd)
Device *outd;
{
        outd->fp = stdout;
        clearscreen(outd);
}

/* Main program */

int
main(argc,argv)
int argc;
char **argv;
{
        Device terminal;

        set_up_terminal();                /* set to unbuffered binary */
        initialize(&terminal);            /* setup internal data */
        dispatch(&terminal);              /* handle user's commands */
        clearscreen(&terminal);
        return 0;
}

```


Tektronix 4010 Emulation "Fortran 77"

```
C tek_mode.f77 -
C   This program permits the user to draw lines using the Tek4010 emulation
C   mode. This program also demonstrates a set of routines for use with
C   the Tektronix 4010.
C
C Usage:
C   Use the following keys to enter points -
C       Key      Description
C       L       Begin a line at current location
C       B       Begin a box at current location
C       T       Begin entering text here
C       D       Delete last object
C       S       Save objects in file
C       Q       Quit
C       SPACE   Finish operation (line, box) here
C       NewLine Finish operation (line, box, text) here
C
C Initial version 01JUN90.
C
C Copyright (c) 1990 by Data General Corporation. Non-exclusive license to
C use, distribute, and modify this code is hereby granted without monetary
C consideration, provided this copyright message is included in all such
C derivative works. Standard disclaimers regarding merchantability,
C suitability for a particular purpose, et cetera, all apply. This code is
C distributed with no warranty whatsoever.
C
C This routine can be used to condition a communication line for binary I/O
C when writing an F77 program under DG AOS. It switches both input and output
C to binary mode, which disables character translation, and unbuffers the
C input.
C
C   subroutine set_up_terminal
C
C   integer stdin, stdout
C   common /units/ stdin, stdout
C
C   open (unit=stdin, file='@input', screenedit='no', maxrecl=1,
1     delimiter='include', force='yes', iointent='input',
2     mode='binary', err=100)
C   open (unit=stdout, file='@output', carriagecontrol='none',
1     screenedit='no', force='yes', iointent='output', mode='binary',
2     err=100)
C   return
100 continue
C   write(*, '( " Unable to open your terminal as a binary device. " )')
C   stop
C   end
```

```

C
C The first section of this program provides support routines for moving
C the cursor around the screen.
C

```

```

        character function getc(iin)
        integer iin

        character ch

        read(iin,1,err=100,end=100) ch
1       format(A1)
        getc=ch
        return
100    continue
        getc=' '
        return
        end

```

```

C Device structure:

```

```

C     1 - output file unit number
C     2 - X beam position
C     3 - Y beam position
C     4 - State (0 - initial, 1 - alpha, 2 - drawing)

```

```

        integer function tenbit(high,low)
        character high,low

```

```

C Convert two characters from terminal into 10 bit number.

```

```

        intrinsic ichar,iand

        tenbit = iand(ichar(high),31)*32 + iand(ichar(low),31)
        return
        end

```

```

        character function getpoint(x,y,outd)
        integer x,y,outd(4)

```

```

C Get a point on the screen from the user.

```

```

        external getc,tenbit
        character getc
        integer tenbit
        integer stdin,stdout
        common /units/ stdin,stdout
        character buffer(4)
        integer i

```

```

C Start cross-hair.

```

```

        write(*,'(''<037><033><032>'')')

```

```

C Wait for key press.

```

```

        getpoint = getc(stdin)
        do 10 i=1,4
        buffer(i) = getc(stdin)
10    continue

```

```

C Decode received point data.
  x = tenbit(buffer(1),buffer(2))
  y = tenbit(buffer(3),buffer(4))
  outd(4) = 1
  return
  end

  subroutine optimizesend(x,y,outd)
  integer x,y,outd(4)

C Send optimized point data in Tek 4010 format.

  intrinsic iand,char
  integer hox,hoy,hoxb,hoyb
  integer lox,loy,loxb,loyb

C Extract 5 bit portions of the data.
  hox = x/32
  hoy = y/32
  lox = iand(x,31)
  loy = iand(y,31)
  if (outd(4) .eq. 0) then
    hoxb = hox+1
    hoyb = hoy+1
    loxb = lox+1
    loyb = loy+1
  else
    hoxb = outd(2)/32
    hoyb = outd(3)/32
    loxb = iand(outd(2),31)
    loyb = iand(outd(3),31)
  endif

C Do optimized Tek coordinate transmission.
  if (hoy.ne.hoyb) write(outd(1),1) char(hoy+32)
  if (loy.ne.loyb .or. hox.ne.hoxb) write(outd(1),1) char(loy+96)
  if (hox.ne.hoxb) write(outd(1),1) char(hox+32)
  write(outd(1),1) char(lox+64)
1
  format(A1)
  outd(2) = x
  outd(3) = y
  return
  end

  subroutine darkto(x,y,outd)
  integer x,y,outd(4)

C Send the beam to a new position without drawing.

  if (x.ne.outd(2) .or. y.ne.outd(3) .or. outd(4).ne.2) then
    write(outd(1),' (' '<035>' ')')
    call optimizesend(x,y,outd)
    outd(4) = 2
  endif
  return
  end

  subroutine gotoalpha(outd)
  integer outd(4)

```

C Send device into alpha mode.

```
if (outd(4).ne.1) then
  write(outd(1), ' (' '<037>' ' ' )
  outd(4) = 1
endif
return
end
```

```
subroutine startstring(x,y,outd)
integer x,y,outd(4)
```

C Begin printing string at given location.

```
call darkto(x,y,outd)
call gotoalpha(outd)
return
end
```

```
subroutine drawvector(xfrom,yfrom,xto,yto,outd)
integer xfrom,yfrom,xto,yto,outd(4)
```

C Draw a vector between the given points.

```
call darkto(xfrom,yfrom,outd)
call optimizesend(xto,yto,outd)
return
end
```

```
subroutine clearscreen(outd)
integer outd(4)
```

C Initialize the screen of the given device.

```
write(outd(1), ' (' '<033><014>' ' ' )
outd(4) = 0
return
end
```

```
integer function lengthof(str,maxlen)
character*(*) str
integer maxlen

integer len
```

C Find actual string length.

```
do 10 len=maxlen,1,-1
if (str(len:len).ne.' ') go to 20
10 continue
len = 0
20 continue
lengthof = len
return
end
```

```
subroutine show_text(text,maxlen,unit)
character*(*) text
integer maxlen,unit
```

C Print a text string without printing trailing blanks.

```
external lengthof
integer lengthof
integer len
```

```
len = lengthof(text,maxlen)
if (len.gt.0) write(unit,' (A)' ) text(1:len)
return
end
```

```

C Item structure:
C   1 - type of item (0 - none, 1 - line, 2 - box, 3 - text)
C   2 - X location
C   3 - Y location
C   4 - X2 or text index
C   5 - Y2

```

```

subroutine draw_object(it,outd)
integer it(5),outd(4)

```

C Draw the object indicated by 'it' on device 'outd'.

```

integer NTEXT
parameter (NTEXT=200)
character*64 texts(NTEXT)
common /texts/ texts

if (it(1).eq.1) then
call drawvector(it(2),it(3),it(4),it(5),outd)
else if (it(1).eq.2) then
call drawvector(it(2),it(3),it(2),it(5),outd)
call drawvector(it(2),it(5),it(4),it(5),outd)
call drawvector(it(4),it(5),it(4),it(3),outd)
call drawvector(it(4),it(3),it(2),it(3),outd)
else if (it(1).eq.3) then
call startstring(it(2),it(3),outd)
call show_text(texts(it(4)),64,outd(1))
endif
return
end

```

```

subroutine redraw_screen(outd)
integer outd(4)

```

C Clear the screen and redraw all objects in the list.

```

integer NITEMS
parameter (NITEMS=10000)
integer items(5,NITEMS)
common /items/ items
integer nexti,nextt
common /pointers/ nexti,nextt
integer i

call clearscreen(outd)
if (nexti.eq.1) return
do 10 i = 1,nexti-1
call draw_object(items(1,i),outd)
10 continue
return
end

subroutine commit_object(it)
integer it(5)

```

C Add given object to database.

```
integer NITEMS
parameter (NITEMS=10000)
integer items(5,NITEMS)
common /items/ items
integer nexti,nextt
common /pointers/ nexti,nextt
integer i

if (nexti.gt.NITEMS) return
do 10 i=1,5
items(i,nexti) = it(i)
100 continue
nexti = nexti+1
return
end

subroutine dellast(outd)
integer outd(4)
```

C Delete last item added to database.

```
integer nexti,nextt
common /pointers/ nexti,nextt

if (nexti.le.1) return
nexti = nexti-1
if (items(1,nexti).eq.3) nextt = nextt-1
call redraw_screen(outd)
return
end

subroutine gettext(it,outd)
integer it(5),outd(4)
```

C Get a text string from the user.

```
external lengthof
integer lengthof
integer NTEXT
parameter (NTEXT=200)
character*64 texts(NTEXT)
common /texts/ texts
integer nexti,nextt
common /pointers/ nexti,nextt
character*64 temp

call startstring(it(2),it(3),outd)
read(11,'(A64)',end=100,err=100) temp
if (lengthof(temp,64).eq.0) go to 100
it(4) = nextt
texts(nextt) = temp
nextt = nextt+1
call commit_object(it)
100 continue
return
end

subroutine ptcopy(it)
integer it(5)
```

C Copy new X/Y to last X/Y

```
it(2) = it(4)
it(3) = it(5)
return
end
```

```
subroutine save_file(outd)
integer outd(4)
```

C Get a file name from the user and save the screen data in it.

```
external lengthof
integer lengthof
integer stdin, stdout, output
common /units/ stdin, stdout, output
character*64 temp
integer file(4), len
```

C Clear the screen and get the file name from the user.

```
call clearscreen(outd)
write(*, '(File name for data: ')
read(11,1) temp
1 format(A64)
write(*, ('<015>'))
```

C Open file and save data.

```
len = lengthof(temp,64)
if (len.eq.0) then
write(*, ('No data saved. '))
else
open(unit=output, file=temp, carriagecontrol='none',
1   screenedit='no', force='yes', iointent='output', mode='binary',
2   err=100)
file(1) = output
```

C Send controls to file.

```
call redraw_screen(file)
```

C Put a mode exit at end.

```
call gotoalpha(file)
close(output)
```

```
write(*,2) temp(1:len)
2 format('Saved screen data in: ',A,'<015><012>')
endif
200 continue
```

```
write(*, ('Press CR to continue.<015><012>'))
read(11,1) temp
```

C Put user's screen back up.

```
call redraw_screen(outd)
return
```

100 continue

```
write(*,3) temp(1:len)
```

```
3 format('Unable to open file for output: ',A,'<015><012>')
go to 200
end
```

```
subroutine dispatch(outd)
integer outd(4)
```

C Dispatch user's input.

```
external getpoint
character getpoint

integer stdin, stdout, output
common /units/ stdin, stdout, output

character chr
integer temp(5)
```

```
temp(1) = 0
10 continue
chr = getpoint(temp(4), temp(5), outd)
if (chr.eq.'<015>' .or. chr.eq.' ') then
  if (temp(1).ne.0) then
    call draw_object(temp, outd)
    call commit_object(temp)
  endif
  call ptcopy(temp)
else if (chr.eq.'l' .or. chr.eq.'L') then
  call ptcopy(temp)
  temp(1) = 1
else if (chr.eq.'b' .or. chr.eq.'B') then
  call ptcopy(temp)
  temp(1) = 2
else if (chr.eq.'t' .or. chr.eq.'T') then
  call ptcopy(temp)
  temp(1) = 3
  call gettext(temp, outd)
```

C Can't duplicate item.

```
temp(1) = 0
else if (chr.eq.'d' .or. chr.eq.'D') then
  call dellast(outd)
else if (chr.eq.'s' .or. chr.eq.'S') then
  call save_file(outd)
else if (chr.eq.'q' .or. chr.eq.'Q') then
  return
endif
go to 10
end
```

```
subroutine initialize(outd)
integer outd(4)
```

C Initialize internal data for program.

```
integer stdin, stdout, output
common /units/ stdin, stdout, output

integer nexti, nextt
common /pointers/ nexti, nextt
```

C Set next next pointers to beginning of array.

```
nexti = 1
nextt = 1
```

C Initialize the terminal screen.

```
outd(1) = stdout
call clearscreen(outd)
return
end
```



```
program drawing
C This is the main program.
    integer stdin, stdout, output
    common /units/ stdin, stdout, output
    data stdin, stdout, output/5, 6, 7/

    integer terminal(4)

C Set to unbuffered binary terminal I/O.
    call set_up_terminal

C Setup internal data.
    call initialize(terminal)

C Handle user's commands.
    call dispatch(terminal)

C Clear Tek 4010 screen before exiting.
    call clearscreen(terminal)

end
```

End of Appendix

Index

Numbers

4010. *See* Tektronix 4010 Mode

A

ACKM. *See* Application/ANSI Cursor Keys Mode

Alignment, VT320/100 mode, 3-63

Answerback, VT320/100 mode, 3-75

Application/ANSI Cursor Keys Mode, VT320/100 mode, 3-54

ARM. *See* Auto Repeat Mode

Arc, DG native-mode, 2-75

Assign User-Preferred Supplemental Set, VT320/100 mode, 3-32, 3-33

Assistance, telephone, iii

AUPSS. *See* Assign User-Preferred Supplemental Set

Auto Print Mode, VT320/100 mode, 3-84

Auto Repeat Mode, VT320/100 mode, 3-57

Auto Wrap Mode, VT320/100 mode, 3-56

AWM. *See* Auto Wrap Mode

B

Backspace, VT320/100 mode, 3-6

Bar, DG native-mode, 2-75

Bell

DG native-mode, 2-71

VT320/100 mode, 3-6

Bit Dump Screen, VT320/100 mode, 3-65

Blink Disable, DG native-mode, 2-33

Blink Enable, DG native-mode, 2-32

Blink Off, DG native-mode, 2-32

Blink On, DG native-mode, 2-32

C

Carriage Return

DG native-mode, 2-38

VT320/100 mode, 3-6

Change Attributes, DG native-mode, 2-31

Character Loopback, DG native-mode, 2-90

Clear Tab Stops, VT320/100 mode, 3-43

Clearing Downloaded Character Sets, VT320/100 mode, 3-36

COM. *See* Cursor Origin Mode

CPR. *See* Cursor Position Report

CUB. *See* Cursor Backward

CUD. *See* Cursor Down

CUF. *See* Cursor Forward

CUP. *See* Cursor Position

CUU. *See* Cursor Up

Cursor Attributes, DG native-mode, 2-80

Cursor Backward, VT320/100 mode, 3-40

Cursor Down

DG native-mode, 2-38

VT320/100 mode, 3-39

Cursor Forward, VT320/100 mode, 3-40

Cursor Left, DG native-mode, 2-37

Cursor Location, DG native-mode, 2-79

Cursor Off, DG native-mode, 2-78

Cursor On, DG native-mode, 2-78

Cursor Origin Mode, VT320/100 mode, 3-55

Cursor Position, VT320/100 mode, 3-40

Cursor Position Report, VT320/100 mode, 3-72

Cursor Reset, DG native-mode, 2-80

Cursor Right, DG native-mode, 2-37

Cursor Track, DG native-mode, 2-79

Cursor Up

DG native-mode, 2-37

VT320/100 mode, 3-39

D

DA. *See* Primary Device Attribute Request

Data General. *See* DG Native-Mode

Data Trap Mode, DG native-mode, 2-88

DCH. *See* Delete Character

DEC. *See* VT320/100 Mode or VT52 Mode

DECID. *See* Terminal Identification

DECSSED. *See* Selective Erase in Display

DECSSEL. *See* Selective Erase in Line

Deallocate Character Sets, DG native-mode, 2-28

Define Character, DG native-mode, 2-29

Delete Character

- DG native-mode, 2-57
- VT320/100 mode, 3-44

Delete Line, DG native-mode, 2-56

Delete Line Between Margins, DG native-mode, 2-56

Designating Character Sets, VT320/100 mode, 3-31

Device Status Report, VT320/100 mode, 3-70

DG native-mode

- Arc, 2-75
- Bar, 2-75
- Bell, 2-71
- Blink Disable, 2-33
- Blink Enable, 2-32
- Blink Off, 2-32
- Blink On, 2-32
- Carriage Return, 2-38
- Change Attributes, 2-31
- Character Loopback, 2-90
- Cursor Attributes, 2-80
- Cursor Down, 2-38
- Cursor Left, 2-37
- Cursor Location, 2-79
- Cursor Off, 2-78
- Cursor On, 2-78
- Cursor Reset, 2-80
- Cursor Right, 2-37
- Cursor Track, 2-79
- Cursor Up, 2-37
- Data Trap Mode, 2-88
- Deallocate Character Sets, 2-28
- Define Character, 2-29
- Delete Character, 2-57
- Delete Line, 2-56
- Delete Line Between Margins, 2-56
- Dim Off, 2-32
- Dim On, 2-32
- Display Character Generator Contents, 2-91
- Double High/Double Wide, 2-35
- Erase Screen, 2-55
- Erase to End of Line, 2-55
- Erase Unprotected, 2-57
- Erase Window, 2-55
- Field Attributes, 2-35
- Fill Screen With Character, 2-91
- Fill Screen With Grid, 2-91
- Form Bit Dump, 2-82
- Horizontal Scroll Disable, 2-54
- Horizontal Scroll Enable, 2-54
- Host Programmable Function Keys, 2-58
- Hot Key Switch, 2-67
- Insert Character, 2-57
- Insert Line, 2-55
- Insert Line Between Margins, 2-56
- Line, 2-74
- Named Save/Restore Cursor, 2-48
- New Line, 2-38
- Page Attributes, 2-36
- Perform UART Loopback Test, 2-91
- Polygon Fill, 2-76
- Pop, 2-45
- Print Form, 2-81
- Print Pass Through Off, 2-84
- Print Pass Through On, 2-83
- Print Screen, 2-82
- Print Window, 2-81
- Printer Pass Back to Host, 2-85
- Protect Disable, 2-34
- Protect Enable, 2-34
- Protect Off, 2-34
- Protect On, 2-34
- Push, 2-44
- Read Bit Contents, 2-90
- Read Characters Remaining, 2-30
- Read Cursor Contents, 2-89
- Read Cursor Location, 2-78
- Read Horizontal Scroll Offset, 2-62
- Read New Model ID, 2-64, 2-65
- Read Screen Address, 2-63
- Read Window Address, 2-62
- Read Window Contents, 2-63
- Report Screen Size, 2-61
- Reserve Character, 2-29
- Reset, 2-71
- Restore Normal Margins, 2-40
- Reverse Video Off, 2-33
- Reverse Video On, 2-33
- Roll Disable, 2-54
- Roll Enable, 2-53
- Save/Restore Screen Contents, 2-49
- Screen Home, 2-49
- Scroll Down, 2-52
- Scroll Left, 2-52
- Scroll Right, 2-53
- Scroll Up, 2-52
- Select 7/8 Bit Operation, 2-72
- Select Character Set, 2-27

Select Compressed Spacing, 2-47
 Select Normal Spacing, 2-48
 Select Printer National Character Set, 2-87
 Set 25th Line Mode, 2-44
 Set Alternate Margins, 2-39
 Set Clock Time, 2-71
 Set Cursor Type, 2-70
 Set Device Options, 2-69
 Set First Row to Display, 2-68
 Set Foreground Color, 2-77
 Set Keyboard Language, 2-72
 Set Margins, 2-39
 Set Model ID, 2-70
 Set Pattern, 2-76
 Set Row Length, 2-50
 Set Scroll Rate, 2-51
 Set Split Screen Mode, 2-68
 Set Windows, 2-43
 Shift In, 2-28
 Shift Out, 2-28
 Show Columns, 2-51
 Simulprint Off, 2-86
 Simulprint On, 2-85
 Switch Emulation Mode, 2-67
 UNIX Mode, 2-73
 Underscore Off, 2-33
 Underscore On, 2-33
 VT-Style Autoprint Off, 2-86
 VT-Style Autoprint On, 2-86
 Window Bit Dump, 2-83
 Window Home, 2-42
 Write Window Address, 2-41

DG native-mode by hex order

- 01, Print Form, 2-81
- 02, Reverse Video Off, 2-33
- 03, Blink Enable, 2-32
- 036 106 077 072, Print Screen, 2-82
- 04, Blink Disable, 2-33
- 05, Read Window Address, 2-62
- 07, Bell, 2-71
- 08, Window Home, 2-42
- 0A, New Line, 2-38
- 0B, Erase to End of Line, 2-55
- 0C, Erase Window, 2-55
- 0D, Carriage Return, 2-38
- 0E, Blink On, 2-32
- 0F, Blink Off, 2-32
- 10 . . . , Write Window Address, 2-41
- 11, Print Window, 2-81
- 12, Roll Enable, 2-53
- 13, Roll Disable, 2-54
- 14, Underscore On, 2-33
- 15, Underscore Off, 2-33
- 16, Reverse Video On, 2-33
- 17, Cursor Up, 2-37
- 18, Cursor Right, 2-37
- 19, Cursor Left, 2-37
- 1A, Cursor Down, 2-38
- 1C, Dim On, 2-32
- 1D, Dim Off, 2-32
- 1E 37 3F 7C, Read Cursor Location, 2-78
- 1E 41 color, Set Foreground Color, 2-77
- 1E 44, Reverse Video On, 2-33
- 1E 45, Reverse Video Off, 2-33
- 1E 46 37 . . . , Select Printer National Character Set, 2-87
- 1E 46 38, Display Character Generator Contents, 2-91
- 1E 46 39, Fill Screen With Grid, 2-91
- 1E 46 3B . . . , Data Trap Mode, 2-88
- 1E 46 3C, Perform UART Loopback Test, 2-91
- 1E 46 3E . . . , Fill Screen With Character, 2-91
- 1E 46 3F 30, Simulprint Off, 2-86
- 1E 46 3F 31, Simulprint On, 2-85
- 1E 46 3F 32, Print Pass Through Off, 2-84
- 1E 46 3F 33, Print Pass Through On, 2-83
- 1E 46 3F 35, Window Bit Dump, 2-83
- 1E 46 3F 36, Form Bit Dump, 2-82
- 1E 46 3F 37, VT-Style Autoprint Off, 2-86
- 1E 46 3F 38, VT-Style Autoprint On, 2-86
- 1E 46 41, Reset, 2-71
- 1E 46 42 . . . , Set Windows, 2-43
- 1E 46 43 . . . , Scroll Left, 2-52
- 1E 46 44 . . . , Scroll Right, 2-53
- 1E 46 45, Erase Screen, 2-55
- 1E 46 46, Erase Unprotected, 2-57
- 1E 46 47, Screen Home, 2-49
- 1E 46 48, Insert Line, 2-55
- 1E 46 48 . . .
 - Select Compressed Spacing, 2-47
 - Select Normal Spacing, 2-48
- 1E 46 49, Delete Line, 2-56
- 1E 46 4C, Protect On, 2-34
- 1E 46 4D, Protect Off, 2-34
- 1E 46 4E . . . , Change Attributes, 2-31
- 1E 46 4F, Read Horizontal Scroll Offset, 2-62
- 1E 46 51 . . . , Set Cursor Type, 2-70
- 1E 46 52 . . . , Define Character, 2-29
- 1E 46 53 . . . , Select Character Set, 2-27
- 1E 46 54 . . . , Set Scroll Rate, 2-51
- 1E 46 55 . . . , Select 7/8 Bit Operation, 2-72
- 1E 46 56, Protect Enable, 2-34
- 1E 46 57, Protect Disable, 2-34
- 1E 46 58 . . . , Set Margins, 2-39
- 1E 46 59 . . . , Set Alternate Margins, 2-39
- 1E 46 5A . . . , Restore Normal Margins, 2-40
- 1E 46 5B, Insert Line Between Margins, 2-56
- 1E 46 5C, Delete Line Between Margins, 2-56
- 1E 46 5D, Horizontal Scroll Disable, 2-54

- 1E 46 5E, Horizontal Scroll Enable, 2-54
- 1E 46 5F . . . , Show Columns, 2-51
- 1E 46 60, Print Pass Through On, 2-83
- 1E 46 61, Print Pass Through Off, 2-84
- 1E 46 62, Read Screen Address, 2-63
- 1E 46 64, Read Characters Remaining, 2-30
- 1E 46 65 . . . , Reserve Character, 2-29
- 1E 46 66 . . . , Set Keyboard Language, 2-72
- 1E 46 68, Push, 2-44
- 1E 46 69, Pop, 2-45
- 1E 46 6B . . . , Host Programmable Function Keys, 2-58
- 1E 46 6D 30, Read Cursor Contents, 2-89
- 1E 46 6D 34 . . . , Character Loopback, 2-90
- 1E 46 6D 35, Hot Key Switch, 2-67
- 1E 46 6D 36, Read Bit Contents, 2-90
- 1E 46 71 . . . , Deallocate Character Sets, 2-28
- 1E 46 72 . . . , Set Clock Time, 2-71
- 1E 46 73 . . . , Save/Restore Screen Contents, 2-49
- 1E 46 74, Report Screen Size, 2-61
- 1E 46 76 . . . , Read Window Contents, 2-63
- 1E 46 77, Read New Model ID, 2-64, 2-65
- 1E 46 78 . . . , Printer Pass Back to Host, 2-85
- 1E 46 7A . . . , Set 25th Line Mode, 2-44
- 1E 46 7B . . . , Set Model ID, 2-70
- 1E 46 7D . . . , Named Save/Restore Cursor, 2-48
- 1E 46 7E . . . , Switch Emulation Mode, 2-67
- 1E 47 30 . . . , Arc, 2-75
- 1E 47 31 . . . , Bar, 2-75
- 1E 47 38 . . . , Line, 2-74
- 1E 47 3A . . . , Polygon Fill, 2-76
- 1E 47 3E 7C, Cursor Location, 2-79
- 1E 47 40, Cursor Attributes, 2-80
- 1E 47 41, Cursor Reset, 2-80
- 1E 47 42, Cursor On, 2-78
- 1E 47 43, Cursor Off, 2-78
- 1E 47 48 . . . , Cursor Track, 2-79
- 1E 47 70 31 . . . , Set Pattern, 2-76
- 1E 48, Scroll Up, 2-52
- 1E 49, Scroll Down, 2-52
- 1E 4A, Insert Character, 2-57
- 1E 4B, Delete Character, 2-57
- 1E 4C . . . , Line, 2-74
- 1E 4E, Shift Out, 2-28
- 1E 4F, Shift In, 2-28
- 1E 50 40 . . . , UNIX Mode, 2-73
- 1E 52 40 . . . , Set Row Length, 2-50
- 1E 52 41 30 . . . , Set Split Screen Mode, 2-68
- 1E 52 41 31 . . . , Set First Row to Display, 2-68
- 1E 52 42 . . . , Set Device Options, 2-69
- 1E 52 43, Field Attributes, 2-35
- 1E 52 44, Page Attributes, 2-36
- 1E 52 45, Double High/Double Wide, 2-35
- 1E 52 78 . . . , Response to Printer Pass Back to Host, 2-85
- 1E 6F 2C . . . , Response to Cursor Attributes, 2-80
- 1E 6F 38 . . . , Response to Read Screen Address, 2-63
- 1E 6F 39 . . . , Response to Read Characters Remaining, 2-30
- 1E 6F 3A . . . , Response to Read Horizontal Scroll Offset, 2-62
- 1E 6F 3B 30 . . . , Response to Read Cursor Contents, 2-89
- 1E 6F 3B 31 . . . , Response to Read Bit Contents, 2-90
- 1E 6F 3C . . . , Response to Report Screen Size, 2-61
- 1E 6F 77 . . . , Response to Read New Model ID, 2-65
- 1E 6F 7C 20 . . . , Response to Read Cursor Location, 2-78
- 1F, Response to Read Window Address, 2-62
- DG native-mode by octal order
 - 001, Print Form, 2-81
 - 002, Reverse Video Off, 2-33
 - 003, Blink Enable, 2-32
 - 004, Blink Disable, 2-33
 - 005, Read Window Address, 2-62
 - 007, Bell, 2-71
 - 010, Window Home, 2-42
 - 012, New Line, 2-38
 - 013, Erase to End of Line, 2-55
 - 014, Erase Window, 2-55
 - 015, Carriage Return, 2-38
 - 016, Blink On, 2-32
 - 017, Blink Off, 2-32
 - 020 . . . , Write Window Address, 2-41
 - 021, Print Window, 2-81
 - 022, Roll Enable, 2-53
 - 023, Roll Disable, 2-54
 - 024, Underscore On, 2-33
 - 025, Underscore Off, 2-33
 - 026, Reverse Video On, 2-33
 - 027, Cursor Up, 2-37
 - 030, Cursor Right, 2-37
 - 031, Cursor Left, 2-37
 - 032, Cursor Down, 2-38
 - 034, Dim On, 2-32
 - 035, Dim Off, 2-32
 - 036 101 color, Set Foreground Color, 2-77
 - 036 104, Reverse Video On, 2-33
 - 036 105, Reverse Video Off, 2-33
 - 036 106 067 . . . , Select Printer National Character Set, 2-87
 - 036 106 070, Display Character Generator Contents, 2-91
 - 036 106 071, Fill Screen With Grid, 2-91
 - 036 106 073 . . . , Data Trap Mode, 2-88

- 036 106 074, Perform UART Loopback Test, 2-91
- 036 106 076 . . . , Fill Screen With Character, 2-91
- 036 106 077 060, Simulprint Off, 2-86
- 036 106 077 061, Simulprint On, 2-85
- 036 106 077 062, Print Pass Through Off, 2-84
- 036 106 077 063, Print Pass Through On, 2-83
- 036 106 077 065, Window Bit Dump, 2-83
- 036 106 077 066, Form Bit Dump, 2-82
- 036 106 077 067, VT-Style Autoprint Off, 2-86
- 036 106 077 070, VT-Style Autoprint On, 2-86
- 036 106 077 072, Print Screen, 2-82
- 036 106 101, Reset, 2-71
- 036 106 102 . . . , Set Windows, 2-43
- 036 106 103 . . . , Scroll Left, 2-52
- 036 106 104 . . . , Scroll Right, 2-53
- 036 106 105, Erase Screen, 2-55
- 036 106 106, Erase Unprotected, 2-57
- 036 106 107, Screen Home, 2-49
- 036 106 110, Insert Line, 2-55
- 036 106 111, Delete Line, 2-56
- 036 106 112 . . . , Select Normal Spacing, 2-48
- 036 106 113 . . . , Select Compressed Spacing, 2-47
- 036 106 114, Protect On, 2-34
- 036 106 115, Protect Off, 2-34
- 036 106 116 . . . , Change Attributes, 2-31
- 036 106 117, Read Horizontal Scroll Offset, 2-62
- 036 106 121 . . . , Set Cursor Type, 2-70
- 036 106 122 . . . , Define Character, 2-29
- 036 106 123 . . . , Select Character Set, 2-27
- 036 106 124 . . . , Set Scroll Rate, 2-51
- 036 106 125 . . . , Select 7/8 Bit Operation, 2-72
- 036 106 126, Protect Enable, 2-34
- 036 106 127, Protect Disable, 2-34
- 036 106 130 . . . , Set Margins, 2-39
- 036 106 131 . . . , Set Alternate Margins, 2-39
- 036 106 132 . . . , Restore Normal Margins, 2-40
- 036 106 133, Insert Line Between Margins, 2-56
- 036 106 134, Delete Line Between Margins, 2-56
- 036 106 135, Horizontal Scroll Disable, 2-54
- 036 106 136, Horizontal Scroll Enable, 2-54
- 036 106 137 . . . , Show Columns, 2-51
- 036 106 140, Print Pass Through On, 2-83
- 036 106 141, Print Pass Through Off, 2-84
- 036 106 142, Read Screen Address, 2-63
- 036 106 144, Read Characters Remaining, 2-30
- 036 106 145 . . . , Reserve Character, 2-29
- 036 106 146 . . . , Set Keyboard Language, 2-72
- 036 106 150, Push, 2-44
- 036 106 151, Pop, 2-45
- 036 106 153 . . . , Host Programmable Function Keys, 2-58
- 036 106 155 060, Read Cursor Contents, 2-89
- 036 106 155 064 . . . , Character Loopback, 2-90
- 036 106 155 065, Hot Key Switch, 2-67
- 036 106 155 066, Read Bit Contents, 2-90
- 036 106 161 . . . , Deallocate Character Sets, 2-28
- 036 106 162 . . . , Set Clock Time, 2-71
- 036 106 163 . . . , Save/Restore Screen Contents, 2-49
- 036 106 164, Report Screen Size, 2-61
- 036 106 166 . . . , Read Window Contents, 2-63
- 036 106 167, Read New Model ID, 2-64, 2-65
- 036 106 170 . . . , Printer Pass Back to Host, 2-85
- 036 106 172 . . . , Set 25th Line Mode, 2-44
- 036 106 173 . . . , Set Model ID, 2-70
- 036 106 175 . . . , Named Save/Restore Cursor, 2-48
- 036 106 176 . . . , Switch Emulation Mode, 2-67
- 036 107 060 . . . , Arc, 2-75
- 036 107 061 . . . , Bar, 2-75
- 036 107 064 . . . , Polygon Fill, 2-76
- 036 107 070 . . . , Line, 2-74
- 036 107 076 174, Cursor Location, 2-79
- 036 107 077 174, Read Cursor Location, 2-78
- 036 107 100, Cursor Attributes, 2-80
- 036 107 101, Cursor Reset, 2-80
- 036 107 102, Cursor On, 2-78
- 036 107 103, Cursor Off, 2-78
- 036 107 110 . . . , Cursor Track, 2-79
- 036 107 160 061 . . . , Set Pattern, 2-76
- 036 110, Scroll Up, 2-52
- 036 111, Scroll Down, 2-52
- 036 112, Insert Character, 2-57
- 036 113, Delete Character, 2-57
- 036 114 . . . , Line, 2-74
- 036 116, Shift Out, 2-28
- 036 117, Shift In, 2-28
- 036 120 100 . . . , UNIX Mode, 2-73
- 036 122 100 . . . , Set Row Length, 2-50
- 036 122 101 060 . . . , Set Split Screen Mode, 2-68
- 036 122 101 061 . . . , Set First Row to Display, 2-68
- 036 122 102 . . . , Set Device Options, 2-69
- 036 122 103, Field Attributes, 2-35
- 036 122 104, Page Attributes, 2-36
- 036 122 105, Double High/Double Wide, 2-35
- 036 122 170 . . . , Response to Printer Pass Back to Host, 2-85
- 036 157 054 . . . , Response to Cursor Attributes, 2-80
- 036 157 070 . . . , Response to Read Screen Address, 2-63
- 036 157 071 . . . , Response to Read Characters Remaining, 2-30
- 036 157 072 . . . , Response to Read Horizontal Scroll Offset, 2-62
- 036 157 073 060 . . . , Response to Read Cursor Contents, 2-89
- 036 157 073 061 . . . , Response to Read Bit Contents, 2-90

036 157 074 . . . , Response to Report Screen Size,
2-61
036 157 167 . . . , Response to Read New Model ID,
2-65
036 157 174 040 . . . , Response to Read Cursor Lo-
cation, 2-78
037, Response to Read Window Address, 2-62
Dim Off, DG native-mode, 2-32
Dim On, DG native-mode, 2-32
Display Character Generator Contents
 DG native-mode, 2-91
 VT320/100 mode, 3-63
Double High/Double Wide, DG native-mode, 2-35
Downloading Soft Characters, VT320/100 mode, 3-35
DSR. *See* Device Status Report

E

ECH. *See* Erase Character
ED. *See* Erase in Display
EL. *See* Erase in Line
Erase Character, VT320/100 mode, 3-45, 3-47
Erase in Display, VT320/100 mode, 3-46
Erase in Line, VT320/100 mode, 3-46
Erase Screen, DG native-mode, 2-55
Erase to End of Line, DG native-mode, 2-55
Erase Unprotected, DG native-mode, 2-57
Erase Window, DG native-mode, 2-55
Escape Sequences, list of
 Tektronix 4010 Mode, 4-10
 VT52 Mode, 3-89
EXRM. *See* Private Reset Mode
EXSM. *See* Private Set Mode

F

Field Attributes, DG native-mode, 2-35
Fill Screen With Character, DG native-mode, 2-91
Fill Screen With Grid, DG native-mode, 2-91
Force Display, VT320/100 mode, 3-66
Form Bit Dump, DG native-mode, 2-82
Form feed, VT320/100 mode, 3-6

H

Hard Terminal Reset, VT320/100 mode, 3-63
Help from Data General, iii
Horizontal and Vertical Position, VT320/100 mode,
3-41
Horizontal Scroll Disable, DG native-mode, 2-54
Horizontal Scroll Enable, DG native-mode, 2-54
Horizontal Tab, VT320/100 mode, 3-6
Host Programmable Function Keys, DG native-mode,
2-58
Hot Key Switch
 DG native-mode, 2-67
 VT320/100 mode, 3-65
HTS. *See* Set Horizontal Tab
HVP. *See* Horizontal and Vertical Position

I

ICH. *See* Insert Character
IL. *See* Insert Line
IND. *See* Index
Index, VT320/100 mode, 3-41
Insert Character
 DG native-mode, 2-57
 VT320/100 mode, 3-44
Insert Line
 DG native-mode, 2-55
 VT320/100 mode, 3-45, 3-47
Insert Line Between Margins, DG native-mode, 2-56

K

KAM. *See* Keyboard Action
KBUM. *See* Typewriter/Data Processing Keys Mode
Keyboard Action, VT320/100 mode, 3-50
Keyboard Language Report, VT320/100 mode, 3-73

L

Line, DG native-mode, 2-74
Line Attributes, VT320/100 mode, 3-37
Line Feed/New Line, VT320/100 mode, 3-51
Linefeed, VT320/100 mode, 3-6
LNM. *See* Line Feed/New Line

M

MNCSM. *See* Multi/National Character Set Mode
Multi/National Character Set Mode, VT320/100 mode, 3-58

N

Named Save/Restore Cursor, DG native-mode, 2-48
NEL. *See* Next Line
New Line, DG native-mode, 2-38
Next Line, VT320/100 mode, 3-42
NKM. *See* Numeric Keypad Mode
Numeric Keypad Mode, VT320/100 mode, 3-58

P

Page Attributes, DG native-mode, 2-36
PCTERM Mode, VT320/100 mode, 3-60
PEXM. *See* Print Extent Mode
Perform UART Loopback Test, DG native-mode, 2-91
PFF. *See* Print Form Feed Mode
Polygon Fill, DG native-mode, 2-76
Pop, DG native-mode, 2-45
Primary Device Attribute Request, VT320/100 mode, 3-71
Print Controller Mode, VT320/100 mode, 3-85
Print Cursor Line, VT320/100 mode, 3-84
Print Extent Mode, VT320/100 mode, 3-57
Print Form, DG native-mode, 2-81
Print Form Feed Mode, VT320/100 mode, 3-57
Print Pass Through Off, DG native-mode, 2-84
Print Pass Through On, DG native-mode, 2-83
Print Screen
 DG native-mode, 2-82
 VT320/100 mode, 3-84
Print Window, DG native-mode, 2-81
Printer Pass Back to Host, DG native-mode, 2-85
Printer Port Status, VT320/100 mode, 3-75
Private Reset Mode, VT320/100 mode, 3-53
Private Set Mode, VT320/100 mode, 3-52
Protect Disable, DG native-mode, 2-34

Protect Enable, DG native-mode, 2-34
Protect Off, DG native-mode, 2-34
Protect On, DG native-mode, 2-34
Push, DG native-mode, 2-44

R

RC. *See* Restore Cursor
Read Bit Contents, DG native-mode, 2-90
Read Characters Remaining, DG native-mode, 2-30
Read Cursor Contents
 DG native-mode, 2-89
 VT320/100 mode, 3-81
Read Cursor Location, DG native-mode, 2-78
Read Horizontal Scroll Offset, DG native-mode, 2-62
Read New Model ID, DG native-mode, 2-64, 2-65
Read Screen Address, DG native-mode, 2-63
Read Window Address, DG native-mode, 2-62
Read Window Contents, DG native-mode, 2-63
Report Screen Size, DG native-mode, 2-61
Request Mode, VT320/100 mode, 3-79
Request Presentation State Report, VT320/100 mode, 3-77
Request Selection or Setting, VT320/100 mode, 3-82
Request Terminal State Report, VT320/100 mode, 3-76
Request User-Preferred Supplemental Set, VT320/100 mode, 3-81
Reserve Character, DG native-mode, 2-29
Reset, DG native-mode, 2-71
Reset Mode, VT320/100 mode, 3-49
Restore Cursor, VT320/100 mode, 3-42
Restore Normal Margins, DG native-mode, 2-40
Restore Presentation State, VT320/100 mode, 3-79
Restore Terminal State, VT320/100 mode, 3-76
Reverse Index, VT320/100 mode, 3-42
Reverse Video Off, DG native-mode, 2-33
Reverse Video On, DG native-mode, 2-33
RI. *See* Reverse Index
RM. *See* Reset Mode
Roll Disable, DG native-mode, 2-54
Roll Enable, DG native-mode, 2-53

RQM. *See* Request Mode
RQPSR. *See* Request Presentation State Report
RQSS. *See* Request Selection or Setting
RQTSR. *See* Request Terminal State Report
RQUPSS. *See* Request User-Preferred Supplemental Set
RSPS. *See* Restore Presentation State
RSTS. *See* Restore Terminal State

S

SASD. *See* Select Active Status Display
Save Cursor, VT320/100 mode, 3-42
Save/Restore Screen Contents, DG native-mode, 2-49
SC. *See* Save Cursor
SCA. *See* Select Character Attributes
SCL. *See* Set Conformance Level
SCRLM. *See* Scrolling Mode
SCRNM. *See* Screen Mode
Screen Home, DG native-mode, 2-49
Screen Mode, VT320/100 mode, 3-55
Scroll Down
 DG native-mode, 2-52
 VT320/100 mode, 3-48
Scroll Left, DG native-mode, 2-52
Scroll Right, DG native-mode, 2-53
Scroll Up
 DG native-mode, 2-52
 VT320/100 mode, 3-48
Scrolling Mode, VT320/100 mode, 3-55
SD. *See* Scroll Down
SDA. *See* Secondary Device Attribute Request
Secondary Device Attribute Request, VT320/100 mode, 3-72
Select 7/8 Bit Operation, DG native-mode, 2-72
Select Active Status Display, VT320/100 mode, 3-66
Select Character Attributes, VT320/100 mode, 3-38, 3-45
Select Character Set, DG native-mode, 2-27
Select Compressed Spacing, DG native-mode, 2-47
Select Graphic Rendition, VT320/100 mode, 3-38
Select Normal Spacing, DG native-mode, 2-48

Select Printer National Character Set, DG native-mode, 2-87
Select Status Line Type, VT320/100 mode, 3-67
Selective Erase in Display, VT320/100 mode, 3-47
Selective Erase in Line, VT320/100 mode, 3-47
Set 25th Line Mode, DG native-mode, 2-44
Set Alternate Margins, DG native-mode, 2-39
Set Clock Time
 DG native-mode, 2-71
 VT320/100 mode, 3-64
Set Conformance Level, VT320/100 mode, 3-67
Set Cursor Type, DG native-mode, 2-70
Set Device Options
 DG native-mode, 2-69
 VT320/100 mode, 3-68
Set First Row to Display, DG native-mode, 2-68
Set Foreground Color, DG native-mode, 2-77
Set Horizontal Tab, VT320/100 mode, 3-43
Set Keyboard Language, DG native-mode, 2-72
Set Limited Transmit, VT320/100 mode, 3-59, 3-62
Set Margins, DG native-mode, 2-39
Set Model ID, DG native-mode, 2-70
Set Pattern, DG native-mode, 2-76
Set Row Length, DG native-mode, 2-50
Set Scroll Rate, DG native-mode, 2-51
Set Split Screen Mode, DG native-mode, 2-68
Set Top and Bottom Margins, VT320/100 mode, 3-65
Set to VT52 Mode, VT320/100 mode, 3-56
Set Windows, DG native-mode, 2-43
Set/Report Language, VT320/100 mode, 3-64
SGR. *See* Select Graphic Rendition
Shift In
 DG native-mode, 2-28
 VT320/100 mode, 3-32
Shift Lock G1 GR, VT320/100 mode, 3-33, 3-34, 3-38
Shift Lock G2 GR, VT320/100 mode, 3-33, 3-34, 3-38
Shift Lock Three, VT320/100 mode, 3-33
Shift Lock Two, VT320/100 mode, 3-33
Shift Out
 DG native-mode, 2-28
 VT320/100 mode, 3-32
Show Columns, DG native-mode, 2-51

SI. *See* Shift In

Simulprint Off, DG native-mode, 2-86

Simulprint On, DG native-mode, 2-85

Single Shift Three, VT320/100 mode, 3-34, 3-38, 3-45

Single Shift Two, VT320/100 mode, 3-34, 3-38

SL2. *See* Shift Lock Two

SL3. *See* Shift Lock Three

SO. *See* Shift Out

Soft Terminal Reset, VT320/100 mode, 3-63

Split Screen, VT320/100 mode, 3-69

SS2. *See* Single Shift Two

SS3. *See* Single Shift Three

SSDT. *See* Select Status Line Type

STBM. *See* Set Top and Bottom Margins

SU. *See* Scroll Up

Switch Emulation Mode, DG native-mode, 2-67

T

Tab, VT320/100 mode, 3-41

TBC. *See* Clear Tab Stops

TCEM. *See* Text Cursor Enable Mode

TEK. *See* Tektronix 4010 Mode

Tektronix 4010 Mode, Escape Sequences, list of, 4-10

Telephone assistance, iii

Terminal Identification, VT320/100 mode, 3-70

Text Cursor Enable Mode, VT320/100 mode, 3-58

Transmit 7-bit Controls, VT320/100 mode, 3-61

Transmit 8-bit Controls, VT320/100 mode, 3-61

Typewriter/Data Processing Keys Mode, VT320/100 mode, 3-59

U

UDK. *See* User Defined Keys

UNIX Mode, DG native-mode, 2-73

Underscore Off, DG native-mode, 2-33

Underscore On, DG native-mode, 2-33

User Defined Key Status, VT320/100 mode, 3-73

User Defined Keys, VT320/100 mode, 3-62

V

Vertical tab, VT320/100 mode, 3-6

VT-Style Autoprint Off, DG native-mode, 2-86

VT-Style Autoprint On, DG native-mode, 2-86

VT320/100 by hex order

05, Answerback, 3-75

07, Bell, 3-6

08, Backspace, 3-6

09

Horizontal Tab, 3-6

Tab, 3-41

0A, Linefeed, 3-6

0B, Vertical tab, 3-6

0C, Form feed, 3-6

0D, Carriage Return, 3-6

0E, Shift Out, 3-32

0F, Shift In, 3-32

12, Force Display (DG private), 3-66

1B 20 46, Transmit 7-bit Controls, 3-61

1B 20 47, Transmit 8-bit Controls, 3-61

1B 23 33, Single-width, 3-37

1B 23 34, Double-height top, 3-37

1B 23 35, Double-width, 3-37

1B 23 36, Double-height bottom, 3-37

1B 23 38, Alignment, 3-63

1B 23 39, Display Character Generator Contents (DG private), 3-63

1B 28 . . . , Designating Character Sets, 3-31

1B 29 . . . , Designating Character Sets, 3-31

1B 2A . . . , Designating Character Sets, 3-31

1B 2B . . . , Designating Character Sets, 3-31

1B 2D . . . , Designating Character Sets, 3-31

1B 2E . . . , Designating Character Sets, 3-31

1B 2F . . . , Designating Character Sets, 3-31

1B 37, Save Cursor, 3-42

1B 38, Restore Cursor, 3-42

1B 48, Set Horizontal Tab, 3-43

1B 4D, Reverse Index, 3-42

1B 4E, Single Shift Two, 3-34, 3-38

1B 4F, Single Shift Three, 3-34, 3-38, 3-45

1B 50 . . . 1B 5C, Clearing Downloaded Soft Character Sets, 3-36

1B 50 . . . 21 . . . 1B 5C, Response to Request User-Preferred Supplemental Set, 3-81

1B 50 . . . 24 70 . . . 1B 5C, Restore Terminal State, 3-76

1B 50 . . . 24 72 . . . 1B 5C, Response to Request Selection or Setting, 3-82

1B 50 . . . 24 74 . . . 1B5C, Restore Presentation State, 3-79

1B 50 . . . 7B . . . 1B5C

Assign User-Preferred Supplemental Set, 3-32, 3-33

- Downloading Soft Characters, 3-35
- 1B 50 . . . 7C . . . 1B5C, User Defined Keys, 3-62
- 1B 50 31 24 73 . . . 1B 5C, Response to Request Terminal State Report, 3-76
- 1B 50 31 24 75 . . . 1B 5C, Request Presentation State Report, 3-77
- 1B 50 34 71 . . . 1B 5C, Request Selection or Setting, 3-82
- 1B 5A, Terminal Identification, 3-70
- 1B 5B . . . 22 71, Select Character Attributes, 3-38, 3-45
- 1B 5B . . . 24 70, Request Mode, 3-79
- 1B 5B . . . 24 77, Request Presentation State Report, 3-77
- 1B 5B . . . 24 79, Response to Request Mode, 3-80
- 1B 5B . . . 24 7D, Select Active Status Display, 3-66
- 1B 5B . . . 24 7E, Select Status Line Type, 3-67
- 1B 5B . . . 40, Insert Character, 3-44
- 1B 5B . . . 41, Cursor Up, 3-39
- 1B 5B . . . 42, Cursor Down, 3-39
- 1B 5B . . . 43, Cursor Forward, 3-40
- 1B 5B . . . 44, Cursor Backward, 3-40
- 1B 5B . . . 48, Cursor Position, 3-40
- 1B 5B . . . 4A, Erase in Display, 3-46
- 1B 5B . . . 4B, Erase in Line, 3-46
- 1B 5B . . . 4C, Insert Line, 3-45, 3-47
- 1B 5B . . . 50, Delete Character, 3-44
- 1B 5B . . . 52, Response to Cursor Position Report, 3-72
- 1B 5B . . . 53, Scroll Up, 3-48
- 1B 5B . . . 54, Scroll Down, 3-48
- 1B 5B . . . 58, Erase Character, 3-45, 3-47
- 1B 5B . . . 63, Primary Device Attribute Request, 3-71
- 1B 5B . . . 66, Horizontal and Vertical Position, 3-41
- 1B 5B . . . 67, Clear Tab Stops, 3-43
- 1B 5B . . . 6C, Reset Mode, 3-49
- 1B 5B . . . 6D, Select Graphic Rendition, 3-38
- 1B 5B . . . 6E, Device Status Report, 3-70
- 1B 5B . . . 70, Set Conformance Level, 3-67
- 1B 5B . . . 72, Set Top and Bottom Margins, 3-65
- 1B 5B 21 70, Soft Terminal Reset, 3-63
- 1B 5B 26 75, Request User-Preferred Supplemental Set, 3-81
- 1B 5B 31 24 75, Request Terminal State Report, 3-76
- 1B 5B 32 30 68, Line Feed/New Line (set), 3-51
- 1B 5B 32 30 6C, Line Feed/New Line (reset), 3-51
- 1B 5B 32 68, Keyboard Action (set), 3-50
- 1B 5B 32 6C, Keyboard Action (reset), 3-50
- 1B 5B 34 69, Print Controller Mode (off), 3-85
- 1B 5B 35 69, Print Controller Mode (on), 3-85
- 1B 5B 36 69, Bit Dump Screen, 3-65
- 1B 5B 3E . . . 63, Response to Secondary Device Attribute Request, 3-72
- 1B 5B 3E 63, Secondary Device Attribute Request, 3-72
- 1B 5B 3F . . . 45, Set/Report Language (DG private), 3-64
- 1B 5B 3F . . . 46, Set Clock Time (DG private), 3-64
- 1B 5B 3F . . . 49, Split Screen (DG private), 3-69
- 1B 5B 3F . . . 4A, Selective Erase in Display, 3-47
- 1B 5B 3F . . . 4B, Selective Erase in Line, 3-47
- 1B 5B 3F . . . 51, Set Device Options, 3-68
- 1B 5B 3F . . . 63, Response to Primary Device Attribute Request, 3-71
- 1B 5B 3F . . . 68, Private Set Mode, 3-52
- 1B 5B 3F . . . 6C, Private Reset Mode, 3-53
- 1B 5B 3F 31 37 68, Read Cursor Contents, 3-81
- 1B 5B 3F 31 38 68, Print Form Feed Mode (set), 3-57
- 1B 5B 3F 31 38 6C, Print Form Feed Mode (reset), 3-57
- 1B 5B 3F 31 39 68, Print Extent Mode (set), 3-57
- 1B 5B 3F 31 39 6C, Print Extent Mode (reset), 3-57
- 1B 5B 3F 31 68, Application/ANSI Cursor Keys Mode (set), 3-54
- 1B 5B 3F 31 69, Print Cursor Line, 3-84
- 1B 5B 3F 31 6C, Application/ANSI Cursor Keys Mode (reset), 3-54
- 1B 5B 3F 32 35 68, Text Cursor Enable Mode (set), 3-58
- 1B 5B 3F 32 35 6C, Text Cursor Enable Mode (reset), 3-58
- 1B 5B 3F 34 32 68, Multi/National Character Set Mode (set), 3-58
- 1B 5B 3F 34 32 6C, Multi/National Character Set Mode (reset), 3-58
- 1B 5B 3F 34 68, Scrolling Mode (set), 3-55
- 1B 5B 3F 34 69, Auto Print Mode (off), 3-84
- 1B 5B 3F 34 6C, Scrolling Mode (reset), 3-55
- 1B 5B 3F 35 68, Screen Mode (set), 3-55
- 1B 5B 3F 35 69, Auto Print Mode (on), 3-84
- 1B 5B 3F 35 6C, Screen Mode (reset), 3-55
- 1B 5B 3F 36 36 68, Numeric Keypad Mode (set), 3-58
- 1B 5B 3F 36 36 6C, Numeric Keypad Mode (reset), 3-58
- 1B 5B 3F 36 38 68, Typewriter/Data Processing Keys Mode (set), 3-59
- 1B 5B 3F 36 38 6C, Typewriter/Data Processing Keys Mode (reset), 3-59
- 1B 5B 3F 36 68, Cursor Origin Mode (set), 3-55
- 1B 5B 3F 36 6C, Cursor Origin Mode (reset), 3-55
- 1B 5B 3F 37 68, Auto Wrap Mode (set), 3-56
- 1B 5B 3F 37 6C, Auto Wrap Mode (reset), 3-56
- 1B 5B 3F 38 68, Auto Repeat Mode (set), 3-57
- 1B 5B 3F 38 6C, Auto Repeat Mode (reset), 3-57

1B 5B 3F 39 39 68, PCTERM Mode (set), 3-60
 1B 5B 3F 39 39 6C, PCTERM Mode (reset), 3-60
 1B 5B 3F 47, Hot Key Switch (DG private), 3-65
 1B 5B 63, Primary Device Attribute Request, 3-71
 1B 5B 69, Print Screen, 3-84
 1B 5D . . . 1B 5C, Operating System Command, 3-7
 1B 5E . . . 1B 5C, Private Message, 3-7
 1B 5F . . . 5C, Application Command, 3-7
 1B 63, Hard Terminal Reset, 3-63
 1B 6E, Shift Lock Two, 3-33
 1B 6F, Shift Lock Three, 3-33
 1B 7D, Shift Lock G2 GR, 3-33, 3-34, 3-38
 1B 7E, Shift Lock G1 GR, 3-33, 3-34, 3-38
 80 through 9F. *See* 1B40 through 1B 5F

VT320/100 mode
 Alignment, 3-63
 Answerback, 3-75
 Application/ANSI Cursor Keys Mode, 3-54
 Assign User-Preferred Supplemental Set, 3-32, 3-33
 Auto Print Mode, 3-84
 Auto Repeat Mode, 3-57
 Auto Wrap Mode, 3-56
 Bit Dump Screen, 3-65
 Clear Tab Stops, 3-43
 Clearing Downloaded Character Sets, 3-36
 Cursor Backward, 3-40
 Cursor Down, 3-39
 Cursor Forward, 3-40
 Cursor Origin Mode, 3-55
 Cursor Position, 3-40
 Cursor Position Report, 3-72
 Cursor Up, 3-39
 Delete Character, 3-44
 Designating Character Sets, 3-31
 Device Status Report, 3-70
 Display Character Generator Contents, 3-63
 Downloading Soft Characters, 3-35
 Erase Character, 3-45, 3-47
 Erase in Display, 3-46
 Erase in Line, 3-46
 Force Display, 3-66
 Hard Terminal Reset, 3-63
 Horizontal and Vertical Position, 3-41
 Hot Key Switch, 3-65
 Index, 3-41
 Insert Character, 3-44
 Insert Line, 3-45, 3-47
 Keyboard Action, 3-50
 Keyboard Language Report, 3-73
 Line Attributes, 3-37
 Line Feed/New Line, 3-51
 Multi/National Character Set Mode, 3-58
 Next Line, 3-42
 Numeric Keypad Mode, 3-58
 PCTERM Mode, 3-60
 Primary Device Attribute Request, 3-71
 Print Controller Mode, 3-85
 Print Cursor Line, 3-84
 Print Extent Mode, 3-57
 Print Form Feed Mode, 3-57
 Print Screen, 3-84
 Printer Port Status, 3-75
 Private Reset Mode, 3-53
 Private Set Mode, 3-52
 Read Cursor Contents, 3-81
 Request Mode, 3-79
 Request Presentation State Report, 3-77
 Request Selection or Setting, 3-82
 Request Terminal State Report, 3-76
 Request User-Preferred Supplemental Set, 3-81
 Reset Mode, 3-49
 Restore Cursor, 3-42
 Restore Presentation State, 3-79
 Restore Terminal State, 3-76
 Reverse Index, 3-42
 Save Cursor, 3-42
 Screen Mode, 3-55
 Scroll Down, 3-48
 Scroll Up, 3-48
 Scrolling Mode, 3-55
 Secondary Device Attribute Request, 3-72
 Select Active Status Display, 3-66
 Select Character Attributes, 3-38, 3-45
 Select Graphic Rendition, 3-38
 Select Status Line Type, 3-67
 Selective Erase in Display, 3-47
 Selective Erase in Line, 3-47
 Set Clock Time, 3-64
 Set Conformance Level, 3-67
 Set Device Options, 3-68
 Set Horizontal Tab, 3-43
 Set Limited Transmit, 3-59, 3-62
 Set Top and Bottom Margins, 3-65
 Set to VT52 Mode, 3-56
 Set/Report Language, 3-64
 Shift In, 3-32
 Shift Lock G1 GR, 3-33, 3-34, 3-38
 Shift Lock G2 GR, 3-33, 3-34, 3-38
 Shift Lock Three, 3-33
 Shift Lock Two, 3-33
 Shift Out, 3-32
 Single Shift Three, 3-34, 3-38, 3-45
 Single Shift Two, 3-34, 3-38
 Soft Terminal Reset, 3-63
 Split Screen, 3-69
 Tab, 3-41
 Terminal Identification, 3-70
 Text Cursor Enable Mode, 3-58

Transmit 7-bit Controls, 3-61
Transmit 8-bit Controls, 3-61
Typewriter/Data Processing Keys Mode, 3-59
User Defined Key Status, 3-73
User Defined Keys, 3-62
VT52 Mode, Escape Sequences, list of, 3-89

W

Window Bit Dump, DG native-mode, 2-83
Window Home, DG native-mode, 2-42
Write Window Address, DG native-mode, 2-41

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
 - a) **MAIL ORDER** – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

Send your order form with payment to: Data General Corporation
 ATTN: Educational Services/TIPS G155
 4400 Computer Drive
 Westboro, MA 01581-9973

- b) **TELEPHONE** – Call TIPS at (508) 870-1600 for all orders that will be charged by credit card or paid for by purchase orders over \$50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
 - a) **Purchase Order** – Minimum of \$50. If ordering by mail, a hard copy of the purchase order must accompany order.
 - b) **Check or Money Order** – Make payable to Data General Corporation.
 - c) **Credit Card** – A minimum order of \$20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

| Total Quantity | Shipping & Handling Charge |
|-----------------------|---------------------------------------|
| 1-4 Items | \$5.00 |
| 5-10 Items | \$8.00 |
| 11-40 Items | \$10.00 |
| 41-200 Items | \$30.00 |
| Over 200 Items | \$100.00 |

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

| Order Amount | Discount |
|---------------------|-----------------|
| \$0-\$149.99 | 0% |
| \$150-\$499.99 | 10% |
| Over \$500 | 20% |

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870-1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

DATA GENERAL CORPORATION TECHNICAL INFORMATION AND PUBLICATIONS SERVICE TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION

Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub-licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES

Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS

Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY

DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY

EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY

A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL

A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)

Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision-locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.