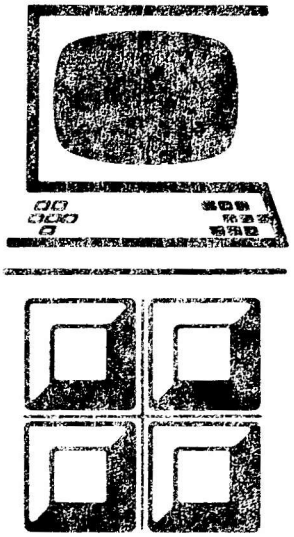
Series 6000 International Support Package Reference Manual

SOFTWARE RELEASE FE07



87601770A

MOTOROLA
Information Systems

List of Effective Pages

This publication contains 126 pages consisting of the following:

Page Number	Date	Page Number	Date	Page Number	Date
Title	30 Apr 85				
A	30 Apr 85				
i	30 Apr 85				
ii (Blank)	30 Apr 85				
iii and iv	30 Apr 85				
1-1 thru 1-3	30 Apr 85				
1-4 (Blank)	30 Apr 85				
2-1 thru 2-4	30 Apr 85				
3-1 thru 3-18	30 Apr 85				
4-1 thru 4-50	30 Apr 85				
A-1 thru A-6	30 Apr 85				
B-1 thru B-27	30 Apr 85				
B-28 (Blank)	30 Apr 85				
C-1	30 Apr 85				
C-2 (Blank)	30 Apr 85				
Index-1 thru Index-3	30 Apr 85				
Index-4 (Blank)	30 Apr 85				
User's Comments	– – –				
Reply Card	– – –				
Inside Back Cover (Blank)	– – –				
Back Cover	– – –				

*Asterisks indicate pages changed, added or deleted by the current change. The issue date for the change appears in place of the previous issue date at the bottom of each changed page included in the change package. Where a change involves a technical correction or the addition of new material, a vertical line appears at the appropriate place in the margin of the affected page. Deletions and editorial corrections are not specifically indicated.

Stock Number: 87601770A
Issue A: 30 April 1985

Specifications subject to change

Printed in U.S.A.

Preface

This manual describes the International Support Package (ISP) that accommodates national characters and other symbols on the System 6300. It details character translation for terminal I/O and printer output and the commands and routines available for programmers.

Readers should be familiar with UNIX or UNIX-derived operating systems and character codes. This manual is a companion volume to the Series 6000 Operating System Reference Manual and the Series 6000 Operating System Programmer's Guide. Readers should also be familiar with the Character Code Standard, XSIS 058404, IDENTITY XC1-1-1-0, by Xerox Corporation.

This issue covers release FE07 of the UNIX-derived operating system. The ISP consists of new and extended commands and routines.

Portions of this manual are excerpts of AT&T documents that describe the UNIX-derived operating system, reproduced by permission.

UNIX is a trademark of Bell Laboratories.

Motorola Inc.
10700 North DeAnza Boulevard
Cupertino, California 95014

Contents

Illustrations

Tables

Section 1
Introduction

INTRODUCTION

The International Support Package (ISP) provides software components for accommodating national characters in terminal I/O, data files, and printer output. These components, resident in the central processor and the terminal, handle device-dependent character-set aspects of terminals and printers and support for a comprehensive character set. This character set includes accented letters, currency symbols, stylized printing forms, and graphic characters. The ISP requires a TM31 Terminal, as do wm(1) and keyprompt(1).

This manual describes the commands, library routines, and character sets of the ISP that are available to the programmer. Character-code translation for terminal I/O and printer output is also described. For further information on this operating system, see the Series 6000 Operating System Reference Manual and the Series 6000 Operating System Programmer's Guide. See also the TM6000 Workstation Programmer's Guide for information on the TM31 Terminal.

Briefly, character translation works as follows: the terminal device handler performs I/O character translation for terminals. Keyboard input sequences that represent displayable characters are mapped from device-dependent codes into internal character-set codes (used by an application program). Outbound sequences (terminal screen- or printer-bound) that represent displayable characters are mapped from internal character-set codes to device-dependent codes for display. Files spooled to printers pass through a "filter" program that maps printable internal character-set codes to printer- and printwheel-dependent output codes. Printer output codes are device-dependent and are specified in the hardware description manual for each printer type.

ORGANIZATION OF THIS MANUAL

This manual consists of the following sections:

Section 1: Introduction to the manual. Gives an overview of the ISP, the layout of the manual, and terminology.

Section 2: User and Programmatic Interfaces. Describes briefly the commands and library routines in these interfaces. Describes terminal initialization, character-set translation, and download images.

Section 3: Translating character sets. Explains translation coding for terminal I/O and printer output.

Section 4: Commands and Files. Explains the ISP commands and files.

Appendix A: Character Set 040. Describes Motorola private character set 040.

Appendix B: TM31 Terminal Key Codes and Keyboards. Describes the keyboard input sequences for the TM31 Terminal, shows the international keyboards, and lists the international dead-key sequences.

Appendix C: Ordering the Character Code Standard. Tells how to order the Character Code Standard, XSIS 058404.

CONVENTIONS AND TERMINOLOGY

The following terms are used throughout this manual. Note that all references to character sets and character codes are in octal.

An internal character is one delivered by the terminal driver to an application program. Internal characters are defined by the Character Code Standard, XSIS 058404, Identity XC1-1-1-0. The Motorola private character set 040 follows this standard.

An inbound character is a terminal-dependent input sequence.

An outbound character is one destined for the screen or the printer.

A "dead" key is a key that does not produce a displayable character, but affects the keystroke that follows.

CSelect is the character-set select code, 377. This 8-bit byte may not be used for a character code or a character-set code.

CharSet8 specifies one of the legal character sets 0, 040-176, and 241-376. CharSet8 is any 8-bit number other than 377, the character-set select code.

Char8Code is the code of a character in its character set. This 8-bit number can have any value other than 377, the CSelect code. Legal values are in the ranges 040-176 and 241-376 and depend on the value of the associated character-set byte.

CS8Declaration names a new 188-character, character-code set. If this 2-byte sequence appears within a 2-byte encoded sequence, the encoding is changed to a 1-byte encoded sequence. If it occurs within a 1-byte encoded sequence, the character set is changed. Format: CSelect CharSet8.

CharSet16 represents a legal 16-bit character set. The only legal value of this 8-bit number is 0.

Char16Code is a 2-byte, 16-bit character code. The allowed values of Char8Code depend on the value of the associated CharSet8 byte.
Format: CharSet8 Char8Code.

CS16Declaration is a 3-byte sequence that precedes a sequence of 2-byte encoded characters. This declaration is always of the form 377 377 000, since the only legal value of CharSet16 is 0. Format: CSelect CSelect CharSet16.

A Stringlet is composed of a character-set declaration followed by a body of character codes. Stringlets make up Strings and are either one or two bytes.

Stringlet8 is a 1-byte Stringlet body that appears at the start of a String. The default interpretation is character codes within character set 0.
Format: 0 ¦ Stringlet8 Char8Code.

DeclaredStringlet8 is a 1-byte Stringlet. It declares the code space at its head and has a sequence of Char8Codes as a body.
Format: CS8Declaration Char8Code ¦ DeclaredStringlet8 Char8Code.

DeclaredStringlet16 is a 2-byte Stringlet. It has a precise CS16Declaration at its head and a sequence of Char16Codes for a body.
Format: CS16Declaration Char16Code ¦ DeclaredStringlet16 Char16Code.

String is a sequence of DeclaredStringlets. The first DeclaredStringlet must be either undeclared (null) or a Stringlet8. DeclaredStringlets can occur in any combination except for the first Stringlet8. Format:

Stringlet8 ¦ DeclaredStringlet8 ¦ DeclaredStringlet16 ¦
String DeclaredStringlet8 ¦ String DeclaredStringlet16

Section 2
User and Programmatic Interfaces

USER INTERFACE

The user-interface commands and operational procedures allow you to

- make translation source files for custom character sets, and
- select and initialize download operations for the TM31 Terminal.

These commands can be broken into two main groups:

1. Those that initialize, query, and control selection of character-set translation for terminals.

2. Those that actually perform the character-set translation of text and print files.

The commands use the programmatic interface components described under "Programmatic Interface."

User-Interface Commands

The commands and procedures in the user interface are as follows:

- Itt(1M)--terminal translation initialization. Itt constructs a character-set translation data structure and sends it to the terminal driver so that I/O translation can be selected. Itt enables but does not select terminal I/O translation.

- Cstty--terminal I/O, character-set query and control. Cstty sets character-set, I/O translation options. It can report the current settings of the character-set translation options and the currently-available translation tables.

- Cstrans(1)--character-set translation. Cstrans translates a file from one standard internal character representation to another, or translates a file from internal character representations to nonstandard or device-dependent codes.

- Csoffset(1)--print-file character-set translation. Csoffset is a filter program for processing files destined for printers. Csoffset translates incoming characters from internal character-set codes to printer device- and model-dependent codes for printing. It also provides print file formatting.

- Nw(1)--new window initialization. Nw initializes a new window, created by wm(1), with the existing environment and then executes a shell program in that window.

- Lpadmin(1M) and lp.cnfg(1M)--print-file character-translation administrative procedures. These two commands provide configuration control of print-file translation filters. Lp.cnfg(1M) is documented in the Software Release Guide.

These commands are described in detail in Section 4.

Initializing Terminals

You can initialize a TM31 Terminal with various language-specific download images. A System 6300 can provide multiple TM31 download images. You can select one of these download images when you power-on the terminal, or use the default download image, number 100. To select another download image,

a. Hold down the space bar.

b. Power-on the terminal.

c. Press T.

d. Enter the number of the download image.

e. Press B to boot the terminal.

PROGRAMMATIC INTERFACE

The programmatic interface performs character-set translation for terminal I/O and for text files, including files destined for printers. This interface includes both system and terminal software.

Programmatic-Interface Commands

The commands and procedures in the programmatic interface are as follows:

- Ioctl(2)—I/O control system call. Ioctl, as extended by the ISP, accommodates status requests and changes to data and option settings. See cstermio(7) for more information.

- Cstrans(3X)--character-set translation library routine. Cstrans translates character strings from one representation to another, based on character-set translation data structures.

- Csinit(3X)--character-set translation initialization library routine. Csinit constructs a character-set translation data structure from a character-set translation source file.

These commands are described in detail in Section 4.

Terminal I/O Character Translation Function

The terminal driver performs terminal I/O character-set translation. The central processor boot procedure initializes character-set translation. During initialization, itt(1M) commands executed in etc/ttrc provide character-set translation data from source files to the terminal driver. Character-set translation tables for the terminal driver reside in operating system kernel space. Character-set translation is activated or deactivated by the cstty(1) command.

The terminal driver has knowledge of internal character sets. Translation flows as follows:

- Terminal-dependent input sequences (inbound codes) are translated from device-dependent codes through character-set translation data structures to XSIS 058404 16-bit characters (CharSet8 Char8Code). Character sets 040, 360, and 361 are avoided wherever possible. The data structures are produced by csinit(3).

- From that representation, codes are further translated, depending on the cstty(1) options selected, and delivered to the user process that is reading from the terminal.

- Outbound codes are translated from internal character codes, as constrained by the cstty(1) options selected, to XSIS 058404 16-bit characters (CharSet8 Char8Code). Character sets 040, 360, and 361 are avoided wherever possible.

- From that character representation, codes are further translated through character-set translation data structures, produced by csinit(3), to terminal-dependent output sequences.

- For outbound characters that are not in character set 000 and that have no translation table entry, the terminal driver substitutes a question mark character (?).

When the terminal driver is optioned to echo input characters (stty echo), the terminal driver echoes those codes exactly as received from the terminal. Since input and output (translated) sequences can have different meanings, depending on the download image, a terminal must be able to display correctly its own input sequences that represent visible characters.

Terminal Download Images

The TM31 download image consists of functions and tables that are placed dynamically in terminal memory when you do any of the following:

- Power-on the terminal.

- Start a download program by making a selection from the terminal ROM program.

- Start a download program by executing the /usr/local/bin/tdl or /usr/local/bin/tmdl program.

TM31 Terminals with a boot ROM upgraded to at least version 2.0 require the tdl program, which downloads the terminal through an RS-232 connection. TM30 and TM31 Terminals that contain boot ROM version 1.0 require the tmdl download program. Tables in the download image determine

- the input sequences sent for each key press or combination of key presses
- what keys are "dead" keys and what combinations of "dead" key sequences are valid
- what international display font is selected
- the mapping for international character output sequences

The TM31 Terminal download image accommodates keyboard variants for various national languages. The input sequences transmitted by all keys not on the basic "typewriter" keypad, plus the TAB, BACKSPACE, and RETURN keys, are identical for all TM31 download variants. Appendix B lists the USA TM31 Terminal keyboard input sequences, displayable characters, ASCII control codes, and function codes for use by application programs.

Section 3
Translating Character Sets

TERMINAL INPUT AND OUTPUT CHARACTER-CODE TRANSLATIONS

The files /usr/lib/cs.term/$TERM.$LANG contain line-image entries that define terminal input and output character-code translations. Some of these files, or possibly all of them, are copied or linked into the directory /etc/cs.term. During the system boot procedure, the initialize terminal-translation command, itt(1M), reads this directory to obtain the source for character-set translation tables. The /usr/lib/cs.term directory contains all of the system's character-set translation source files for terminals, but only those that are to be accessed when the system is booted appear in /etc/cs.term.

The sections of these files that declare translations for inbound characters accommodate only those characters that can normally be entered from the keyboard. For the TM31 Terminal, such characters are those that can be transmitted by pressing a key, with or without the SHIFT or CTRL key, that yields a displayable character, or any "dead" key sequence. (A "dead" key sequence involves a key that does not produce a displayable character, but affects the keystroke that follows.) For example, typing CTRL . N x, which would give the input sequence ESC N x, is not considered to be a character that can normally be entered. The sections of these files that declare translations for outbound characters accommodate all XSIS 058404 characters that the terminal can display, including Motorola private character set 040. When there is no translation table entry for an outbound accented character, the given letter without the accent is substituted. Also, when there is no translation table entry for other outbound characters that are not in internal character set 0, the question mark character (?) is substituted.

These files declare values used in the translation of device-dependent input and output codes to and from 16-bit internal character codes. The 16-bit code is comprised of an 8-bit character set code (CharSet8) and an 8-bit character code (Char8Code) within the specified character set. The files also avoid, but don't necessarily exclude, use of internal character sets 040, 360, and 361. The terminal driver translation programs know about internal character sets and cstty(1) option settings, which determine the actual character codes delivered to or received from a user process doing terminal I/O.

Source files for character-set translation also occur for device types other than terminals, such as printers (see "Printer Output Character-Code Translations"). These files also specify translation mappings from one internal character set to another, such as from Motorola private character set 040 to XSIS 058404 non-040 character sets. The source file syntax described in this section covers all cases to which such files apply: terminal, printer, and internal character-set translations.

Format7 applies only to outbound characters and, if it is used, should follow immediately after an outbound statement. This keyword declares that for any outbound character whose value is greater than 177, that character will be represented by the output sequence

\016 x \017

where

\016 is the ASCII shift-out control code

x is the value of the outbound character byte minus 200

017 is the ASCII shift-in control code

The use of format7 is restricted to those device types that implement ASCII shift-out/shift-in, two-state character sets. Format7 makes coding character-set translation source files less laborious, and reduces the size of the resultant character-set translation data structure. The use of this keyword precludes use of primary and cselect statements.

Primary <pattern> applies only to outbound characters and device types that have state-switchable character sets. If it is used, this line must occur before any translate statements; it must be used if any cselect statements occur. This statement declares that the output sequence to select the device's

primary character set
(assumed to be character set 0)
is
<pattern>, a constant byte value pattern of one or more bytes

Cselect <cset_num> <pattern> applies only to outbound characters and device types that have state-switchable character sets. It must occur prior to any translate statements if it is used. This statement declares that the

cselect
(character-set selection operation to switch the device to its character-set number)
<cset_num>, the character-set number
(given by a constant byte value)
is to write
<pattern>, a constant byte value pattern of one or more bytes
to the device

Translate <pattern><x_variable> range <lo_value> <hi_value> applies to inbound, outbound, and internal characters. This statement declares that the translator program will

translate
<pattern>, a constant byte value pattern of zero or more bytes
(followed immediately by)

<x_variable>, a variable byte
(represented by the symbol \x, whose value will be other than the values of the accent characters in character set 0 and which falls within a range)
range, the range of <x_variable>, from <lo_value> to <hi_value>
<lo_value>, a constant byte value representing the lower bound
<hi_value>, a constant byte value representing the upper bound
(in accord with the value statement(s) on the following line(s) indexed by <x_variable> - <lo_value>)

Translate <pattern><a_variable> accent value applies only to outbound and internal characters. This statement declares that the translator program will

translate
<pattern>, a constant byte value pattern of zero or more bytes
(followed immediately by)
<a_variable>, a variable byte
(represented by the symbol \a, whose value will be among the values of the accent characters in character set 0 and is the accent)
accent, the accent
value, the constant byte value of the accent character
(in accord with the value statement(s) on the following line(s) located by the value of the next byte received)

[<Accent_flag>] <value_string> [<value_string>] declares an optional accent indicator. It also specifies one or more values to be associated with an <x_variable> or an <a_variable> in the previous translate statement.

For a translate ... range statement in an inbound or internal section, there must be a line containing two <value_string>'s, or two <value_string>'s and an <accent_flag>, for each value in the range. In either case,

- The first line declares the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <lo_value>.

- The second line gives the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <lo_value + 1>, and so on.

- The line that is the last noncomment line in the file, or is the last line before a keyword, gives the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <hi_value>.

When the value of the <x_variable> does not represent an accented letter, the value declaration line(s) for an inbound or internal translate ... range statement appears as

```
<CharSet8 Char8Code>
```

When the value of the <x_variable> does represent an accented letter, the value declaration line(s) for an inbound or internal translate ... range statement has the form

\a <accent_value> <letter_value>

where

\a indicates that this line is an accented character value declaration.

<accent_value> is the Char8Code of the appropriate accent in internal character set 000.

<letter_value> is the Char8Code of the appropriate letter in internal character set 000.

This value-statement format occurs only for translate ... range statements in inbound sections for devices that transmit a single <x_variable> to represent an accented character. Similarly, in internal sections, this statement appears only for translate ... range statements for single input characters that represent an accented character.

For a translate ... range statement in an outbound section, a line containing one <value_string> must occur for each value in the range.

- The first such line declares the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <lo_value>.

- The second line declares the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <lo_value + 1>, and so on.

- The line that is the last noncomment line in the file, or is the last line before a keyword, gives the byte sequence to be delivered as the translation result when the translator program receives the <x_variable>, whose value is <hi_value>.

Thus, the value declaration line(s) for a translate ... range statement appears in the one-value format

<device_output_sequence>

For a translate ... accent statement in an outbound section, a line containing two <value_string>'s must occur for each character to which the accent applies.

- The first <value_string> represents the character set 000 Char8Code value of a character that is valid in combination with the accent character.

- The second <value_string> declares the byte sequence to be delivered as the translation result when the translator program receives the accent character followed by the first <value_string> character.

Thus, the value-declarations line(s) for a translate ... accent statement appears in the two-value format

<accented_char> <device_output_sequence>

When one or more cselect statements occur in an outbound section and the output sequence is not in the device's primary character set, there is an additional <value_string>. This additional <value_string> declares the device character-set number for that output sequence and precedes the <value_string> that declares the device-output byte sequences following translate statements. Thus, for non-primary device character-set output sequences, the value declaration line(s) following a translate ... range statement appears in the two-value format

<cset_num> <device_output_sequence>

And, following a translate ... accent statement, the value declaration line(s) for these output sequences appears in the three-value format

<accented_char> <cset_num> <device_output_sequence>

If the <cset_num> character set is not already selected, the <cset_num> value identifies the selection sequence of the device character set to be sent to the device. This sequence is declared in the related cselect statement.

Figure 3-1 is an example of translation coding for inbound, outbound, and internal characters.

```
inbound
translate \EN\x range A C
\000 \243       # pound sign
\000 \373       # ess-zed
\a \310 U       # U dieresis
# When the input sequence ESC N is received from the terminal,
#   if the next character is A, then output \000 \243.
#   if the next character is B, then output \000 \373.
#   if the next character is C, then output \000 \310 \000 U.

outbound
primary \E(B
cselect \001 \E(K
translate \000\x range \247 \247
\001 @          # section
translate \000\a accent \310
a \001 {        # a dieresis
o \001 \174     # o dieresis
```

Figure 3-1. Translation Coding for Inbound, Outbound, and Internal Characters (Page 1 of 2)

```
# When the output sequence \000 \247 is destined for the device,
#   if the device is already in its character-set 1 state, then
#     output @
#   else
#     output \033 \050 \113 @.
# When the output sequence \000 \310 is destined for the device,
#   read the next character, called <letter>.
#   If the device is already in its character-set 1 state, then
#     if <letter> == a, then
#       output {
#     else if <letter> == o, then
#       output \174
#     else
#       output <letter>
#   else
#     if <letter> == a, then
#       output \033 ( K {
#     else if <letter> == o, then
#       output \033 ( K \174
#     else
#       output \033 ( K <letter>
#
# The code sequence <letter> is output if the translation table declares no
# matching value for the accent character followed by the <letter>
# character.

internal
translate \040\x range \044 \044
\000 \244       # dollar sign
translate \040\x range \174 \174
\357 \153       # broken vertical bar
translate \040\x range \241 \322
\a \310 A       # A dieresis
\a \312 A       # A ring
\a \304 A       # A tilde
\000 \341       # AE ligature
\a \313 C       # C cedilla
\a \302 E       # E acute
\a \310 O       # O dieresis
# fragment of the declarations for translating from Motorola private
# internal character-set 040 to XSIS 058404 non-040 character sets.
```

Figure 3-1. Translation Coding for Inbound, Outbound, and Internal Characters (Page 2 of 2)

Figure 3-2 is an example of a character-set translation source file for TM31 Terminal ASCII and German character sets.

```
# File Name:      tm31.deut
#                 "@(#) tm31.deut 1.0
#
# Description:    Character-set translation source file for TM31
#                 Terminal ASCII and German character sets.
#
# @(#) Copyright (C) 1985 by Information Systems Group of Motorola Inc.
#
# Content:        Character-set translation source statements.
#                 TERM=tm31
#                 LANG=deut           German
#
# Revision History:
#
#                 CBS       Cork      30 Jan 85      FE07 Release
#
#                 Initial version.
#


inbound               # device-independent codes translated into internal
                      # character codes

translate \EO\x range @ @             # ALT key translated to
\000   \E                             # ASCII ESC

translate \EN\x range @ L             # Characters entered through CHAR CODE
                                      # key
\a     \310 A                         # A dieresis
\a     \310 O                         # O dieresis
\a     \310 U                         # U dieresis
\a     \310 a                         # a dieresis
\a     \310 o                         # o dieresis
\a     \310 u                         # u dieresis
\000   \373                           # esset
\a     \301 a                         # a grave
\a     \301 e                         # e grave
\a     \301 u                         # u grave
\a     \302 e                         # e acute
\a     \303 a                         # a circumflex
\a     \303 e                         # e circumflex

translate \EN\x range N N
\a     \303 o                         # o circumflex

translate \EN\x range T W
\000   \322                           # registered
\000   \243                           # pound sign
\000   \254                           # west arrow
\000   \323                           # copyright
```

Figure 3-2. Terminal Input and Output Character-Code Translations (Page 1 of 6)

```
translate \EN\x range Z Z
\000    \257                          # south arrow

translate \EN\x range ^ _
\000    \255                          # north arrow
\000    \174                          # solid vertical bar

translate \EN\x range c c
\000    \270                          # divide

translate \EN\x range j k
\000    \266                          # paragraph
\000    \247                          # section

translate \EN\x range z {
\000    \275                          # 1/2
\000    \274                          # 1/4


outbound          # internal character codes translated into device-
                  # dependent codes

                  # The output sequences here represent TM31 Terminal G2 and
                  # G1 actual characters, even though the input sequences
                  # for some of these characters (those entered through the
                  # CHAR CODE key) are different.  This convention
                  # requires the TM31 Terminal memory program (download
                  # image) not to use TM31 G2 codes that represent XSIS
                  # 058404 characters.  The G2 codes available for national
                  # characters are \100 through \127, \132, \135, \136,
                  # \154 through \160, \163, \165 through \171, and \173.

primary \017       # ASCII shift-in code selects G0 character set
cselect \001 \016 # ASCII shift-out code selects G1 character set

translate \000\x range \174 \174              # XSIS 058404 character set 000
\EN_              # solid vertical bar

translate \000\x range \241 \377
?
?
\ENU              # pound sign
\044              # dollar sign
?
?
\ENk              #247 section
?
?
?
?
```

Figure 3-2. Terminal Input and Output Character-Code Translations (Page 2 of 6)

```
\EN>            #254 west arrow
\EN<            # north arrow
\EN?            # east arrow
\EN=            # south arrow
\ENq            # degree
\ENd            # plus or minus
\EN2            # superscript 2
\EN3            # superscript 3
\ENx            #264 multiply
?
\ENj            # paragraph
\EN[            # center
\ENc            #270 divide
?
?
?
\EN{            #274 1/4
\ENz            # 1/2
\EN'            #276 3/4
?
?               #300
\040\010        # SP BS for accent      \301
\040\010        # SP BS for accent      \302
\040\010        # SP BS for accent      \303
\040\010        # SP BS for accent      \304
\040\010        # SP BS for accent      \305
\040\010        # SP BS for accent      \306
\040\010        # SP BS for accent      \307
\040\010        # SP BS for accent      \310
\040\010        # SP BS for accent      \311 # Reserved, unassigned
\040\010        # SP BS for accent      \312
\040\010        # SP BS for accent      \313
\040\010        # SP BS for accent      \314
\040\010        # SP BS for accent      \315
\040\010        # SP BS for accent      \316
\040\010        # SP BS for accent      \317
?
\EN1            # superscript 1
\EN/            # registered
\EN.            # copyright
\EN:            #324 trademark
?
?
?
```

Figure 3-2. Terminal Input and Output Character-Code Translations (Page 3 of 6)

```
?                   #330
?
?
?
?
?
?
?
?                   #340
?
?
?
?
?
?
?
?                   #350
?
?
?
?
?
?
?
?                   #360
?
?
?
?
?
?
?
?                   #370
?
?
\ENF                #373 esset
?
?
?
?                   #377

translate \000\a accent \301
a \ENG              # a grave
e \ENH              # e grave
u \ENI              # u grave

translate \000\a accent \302
e \ENJ              # e acute
```

Figure 3-2. Terminal Input and Output Character-Code Translations (Page 4 of 6)

```
translate \000\a accent \303
a \ENK              # a circumflex
e \ENL              # e circumflex
o \ENN              # o circumflex

translate \000\a accent \310
A \EN@              # A dieresis
O \ENA              # O dieresis
U \ENB              # U dieresis
a \ENC              # a dieresis
o \END              # o dieresis
u \ENE              # u dieresis


translate \040\x range \323 \336     # character set 040 unique characters
\016_\017           # box: upper left corner
\016‾\017           # box: diagonal upper left corner (same as above)
\016'\017           # box: upper right corner
\016'\017           # box: diagonal upper right corner (same as above)
\016]\017           # box: lower left corner
\016]\017           # box: diagonal lower left corner (same as above)
\016^\017           # box: lower right corner
\016^\017           # box: diagonal lower right corner (same as above)
\016R\017           # box: right-pointing tee
\016S\017           # box: left-pointing tee
\016N\017           # box: up-pointing tee
\016M\017           # box: down-pointing tee

translate \040\x range \344 \345     # more character set 040 unique
                                     # characters
\016n\017           # Right sloping hatches (horizontal shading on TM31
                    # Terminals)
\016o\017           # Left sloping hatches (vertical shading on TM31 Terminals)

translate \041\x range \104 \104     # XSIS 058404 character set 041
\ENt                # leaders

translate \041\x range \142 \142
\ENh                # !=

translate \041\x range \145 \146
\ENf                # <=
\ENe                # >=

translate \042\x range \041 \043     # XSIS 058404 character set 042
\EN&                # diamond
\ENY                # open square
\ENX                # solid square
```

Figure 3-2. Terminal Input and Output Character-
Code Translations (Page 5 of 6)

```
translate \042\x range \045 \045
\EN%              # triangle

translate \357\x range \060 \061     # XSIS 058404 character set 357
\EN+              # dagger
\EN*              # double dagger

translate \357\x range \146 \146
\EN;              # bullet

translate \357\x range \152 \153
\ENi              # logical not
\174              # broken vertical bar

translate \357\x range \167 \167
\ENg              # approximately

translate \357\x range \270 \270
\EN\\             # QED

translate \357\x range \344 \346
\016 2 \017       # box: vertical line
\016 K \017       # box: horizontal line
\016 L \017       # box: crossing line

translate \357\x range \375 \376
\ENa              # 1/3
\ENb              # 2/3

# end of file
```

Figure 3-2. Terminal Input and Output Character-Code Translations (Page 6 of 6)

PRINTER OUTPUT CHARACTER-CODE TRANSLATIONS

The /usr/lib/cs.printer/<model>.<lang> files contain line-image entries that define character-code translations for printer output. After receiving the pathname of one of these files as an argument, the csoffset(1) command reads that file to obtain the source for a character-set translation table. The /usr/lib/cs.printer directory contains all of the system's source files for character-set translation for printers. These files declare translations for outbound characters and accommodate all XSIS 058404 characters that the printer can print, including Motorola private character set 040. For outbound characters that are not in internal character-set 0 and that have no entry in the translation table, the question mark character (?) is substituted.

These files declare values used in the translation of device-dependent output codes to and from 16-bit internal character codes. The 16-bit code is comprised of an 8-bit, character-set code (CharSet8) and an 8-bit character code (Char8Code) within the specified character set. The files also avoid, but don't necessarily exclude, use of internal character sets 040, 360, and 361.

The csoffset(1) translation command knows about internal character sets and accepts character-set selection arguments, which constrain the actual character codes encountered in a file to be printed.

The set-up of these files is as follows:

name /usr/lib/cs.printer/<model>.<lang>

where

<model> designates a printer model defined through the lp.cnfg(1M) command. (See the Software Release Guide for information on lp.cnfg(1M).)

<lang> is a user-language identifier associated with the printer model. For printer types that are not language-dependent, the .<lang> part of the file name is omitted. The values of <lang> are as follows:

Value	Language
cdn	Canadian
deut	German
engli	English (U.S.A.)
esp	Spanish
fra	French
hol	Dutch
sve	Swedish
uk	English (U.K.)

format This file is an ASCII line-image file. Its format is the same as for the character-set translation source-file format described in "File Structure," except that printer translation files have no inbound declaration.

Figure 3-3 is an example of a character-set translation source file for a PT34 Character Printer.

```
# File name:          pt34
#                     "@(#) pt34 1.0
#
# Description:  Character set translation source file for PT34 Character
#               Printer
#
# @(#) Copyright (C) 1985 by Four-Phase Systems (ISG) of Motorola, Inc.
#
# Content:       Character-set translation source statements.
#
```

Figure 3-3. Printer Output Character-Code Translations (Page 1 of 4)

```
# Revision History:
#
#       GOC.       ISDC, Cork, Ireland.          Feb. 13 1985.
#       Initial version.
#
#
primary             \E(B       # U.S.A.-ASCII
cselect \001        \E(A       # U.K.
cselect \002        \E(K       # German
cselect \003        \E(2       # Swedish
cselect \004        \E(R       # French
cselect \005        \E(4       # Spanish
cselect \006        \E(1       # Italian
cselect \007        \E(3       # Norwegian

outbound            # internal character codes translated into device-
                    # dependent codes

translate \000\x range \241 \376
\005 [              # \241     Inverted exclamation point
?                   # \242     Cent sign
\001 #              # \243     Pound (sterling) sign
\044                # \244     Dollar sign
?                   # \245     Yen sign
?                   # \246     Reserved
\002 @              # \247     Section mark
?                   # \250     Reserved
?                   # \251     Left single quotation mark
?                   # \252     Left double quotation mark
?                   # \253     Left double guillemet
?                   # \254     West arrow
?                   # \255     North arrow
?                   # \256     East arrow
?                   # \257     South arrow
\004 [              # \260     Degree sign
?                   # \261     Plus/minus sign
?                   # \262     Superscript 2
?                   # \263     Superscript 3
?                   # \264     Multiplication sign
?                   # \265     Micro sign
?                   # \266     Paragraph mark
?                   # \267     Centered dot
?                   # \270     Division sign
?                   # \271     Right single quotation mark
?                   # \272     Right double quotation mark
?                   # \273     Right double guillemet
?                   # \274     Fraction 1/4
?                   # \275     Fraction 1/2
?                   # \276     Fraction 3/4
\005 ]              # \277     Inverted question mark
?                   # \300     Reserved
```

Figure 3-3. Printer Output Character-Code Translations (Page 2 of 4)

```
'\010           # \301   Grave accent (non-spacing)
\040\010        # \302   Acute accent (non-spacing)
^\010           # \303   Circumflex (non-spacing)
~\010           # \304   Tilde accent (non-spacing)
\040\010        # \305   Macron (non-spacing)
\040\010        # \306   Breve accent (non-spacing)
\040\010        # \307   Over-dot accent (non-spacing)
\004 ~\010      # \310   Dieresis (non-spacing)
\040\010        # \311   Reserved (non-spacing)
\040\010        # \312   Ring (non-spacing)
,\010           # \313   Cedilla (non-spacing)
_\010           # \314   Underline (non-spacing)
\040\010        # \315   Double acute accent (non-spacing)
\040\010        # \316   Ogonek undermark (non-spacing)
\040\010        # \317   Hachek accent (non-spacing)
?               # \320   Horizontal bar
?               # \321   Superscript 1
?               # \322   Circle R (registered)
?               # \323   Circle C (copyright)
?               # \324   TM (trademark)
?               # \325   Music note
?               # \326   Reserved
?               # \327   Reserved
?               # \330   Reserved
?               # \331   Reserved
?               # \332   Reserved
?               # \333   Reserved
?               # \334   Fraction 1/8
?               # \335   Fraction 3/8
?               # \336   Fraction 5/8
?               # \337   Fraction 7/8
?               # \340   Ohm sign
\007 [          # \341   AE digraph
?               # \342   D stroke
?               # \343   a ordinal
?               # \344   H stroke
?               # \345   Reserved
?               # \346   IJ digraph
?               # \347   L middle dot
-\010L          # \350   L stroke
\007 ]          # \351   O slash
?               # \352   OE digraph
?               # \353   o ordinal
?               # \354   Uppercase thorn
-\010T          # \355   T stroke
?               # \356   Eng
?               # \357   N apostrophe
?               # \360   k (Greenlandic)
\007 {          # \361   ae digraph
?               # \362   d stroke
?               # \363   Eth
```

Figure 3-3. Printer Output Character-Code Translations (Page 3 of 4)

```
?                   # \364     h stroke
?                   # \365     Dotless i
?                   # \366     ij digraph
?                   # \367     l middle dot
-\0101              # \370     l stroke
\007 ¦              # \371     o slash
?                   # \372     oe digraph
\002 ~              # \373     ess-zed (German)
?                   # \374     Lowercase thorn
-\010t              # \375     t stroke
?                   # \376     Lowercase eng



translate \000\a accent \310
A \002 [            # A dieresis replaces [
O \002 \134         # O dieresis replaces backslash
U \002 ]            # U dieresis replaces ]
a \002 {            # a dieresis replaces {
o \002 ¦            # o dieresis replaces ¦
u \002 }            # u dieresis replaces }

translate \000\a accent \301
a \004 @            # grave
e \004 }
i \006 ~
o \006 ¦
u \004 ¦

translate \000\a accent \312
A \003 ]            # ring
a \003 }

translate \000\a accent \302
E \003 @            # acute
a \005 ^
e \004 {
i \005 {
o \005 }
u \005 ~

# end of file
```

Figure 3-3. Printer Output Character-Code Translations (Page 4 of 4)

Section 4
Commands and Files

COMMANDS

The commands described here are ISP user commands and are given in the same form as in the Series 6000 Operating System Reference Manual. The "Synopsis" subsections give summaries of the commands, the "Files" subsections list the files associated with the commands, and the "See Also" subsections give related information.

brc(1M)

brc, bcheckrc, rc, ttrc - system initialization shell scripts

SYNOPSIS

/etc/brc
/etc/bcheckrc
/etc/rc
/etc/ttrc

DESCRIPTION

Init(1M) executes these shell procedures through entries in inittab(4) files when the system changes out of SINGLE USER mode.

Brc clears the mounted file-system table, /etc/mnttab (see mnttab(4)), and loads any programmable microprocessors with appropriate scripts.

Bcheckrc performs all the necessary consistency checks before the system changes into multiuser mode. It actually contains two procedures: an interactive procedure that runs fsck(1M) and sets the time, and a noninteractive procedure that checks only the file system. The system administrator chooses the interactive or noninteractive procedure by modifying the line in bcheckrc that sets the variable CONSOLE. Use the value PRESENT to specify the interactive procedure and ABSENT to specify the noninteractive. If the ABSENT procedure fails because of file-system problems or because it was interrupted from the controlling terminal, bcheckrc switches the system to state 6, which is normally operating-system administrator mode.

Rc starts all system daemons before the terminal lines are enabled for multiuser mode. In addition, file systems are mounted, and accounting, error logging, and system activity logging are activated.

Ttrc uses the itt(1M) command to initialize character-set translation tables for the terminal device driver.

These shell procedures, in particular rc, can be used for several run-level states. Use the who(1) command to get the run-level information.

SEE ALSO

init(1M), shutdown(1M), who(1), inittab(4).

csoffset(1)

csoffset - print-file character-set translator and formatter

SYNOPSIS

csoffset [options]

DESCRIPTION

Csoffset(1) is a filter program for processing files destined for printers. Csoffset functions as follows:

1. Reads its standard input.

2. Translates incoming characters from internal character-set codes to printer device- and model-dependent codes for printing. It also provides print file formatting in accord with selected options.

3. Writes the resultant characters on its standard output.

Csoffset(1) is activated as a command in the /usr/spool/lp/model/ptnnx commands text file (see lpadmin(1M)). Knowledge of printer models, model availability, and access to the appropriate printer model commands text file resides in the lp(1) facility (lp, lp.cnfg, lpadmin, lpstat, lpsched, enable, disable, accept, reject. Note that lp.cnfg appears in the Software Release Guide.) The printer interface file built by lp.cnfg(1M) selects the appropriate character-set translation table. This file includes ptnnx, which receives arguments through the lp(1) command's -o option.

The options are as follows:

-tabs — Selects expansion of tab characters on output. Each tab (ASCII HT character) in the input stream is replaced by eight spaces (ASCII SP characters) on output. If you omit this option, tab expansion does not occur.

onlcr — Selects mapping of new-line (NL) characters to CR-NL (a carriage return followed by a new-line) on output. If you omit this option, CR characters are not inserted before the NL characters.

<nn> — Indicates the number of spaces (ASCII SP characters), in decimal, to add to the beginning of each line of the output stream. These spaces determine the width of the print image's left margin. If you omit this option, the value 0 is used.

-if7 — Declares that the input format is ASCII characters with SO (shift-out, 016) and SI (shift-in, 017) bracketing characters for which the leftmost bit is considered set.

For this format, SUB @ (032 100) represents the byte value 000 and SUB / (032 057) represents the byte value 377. The value 377 is used as a character-set selection (CSelect) byte to introduce standard internal character stringlets. The value 000 is used as a character-set number in standard internal character stringlets. Inbound SO and SI codes are treated as character-set selection codes and are not delivered as outbound codes.

t=<filename> Specifies the <filename> of a source file for character-set translation. This source file translates the input stream into device-dependent output sequences as required by the destination printer.

For outbound characters that are not in character set 000 and that have no translation table entry, csoffset substitutes the question mark character (?).

If you omit this option, the output stream is written in XSIS 058404 standard, 8-bit, stringlet representation without an initial CSelect byte.

DIAGNOSTICS

Csoffset writes error messages of the following form on its standard error file.

csoffset: unable to allocate space to build translation table
System error. Call your Customer Support Representative for assistance.

csoffset: array overflow while building translation table
System error. Call your Customer Support Representative for assistance.

csoffset: invalid option <option>
usage: csoffset [-tabs] [onclr] [<offset>] [-if7] [t=<filename>]
Change the command to include valid options only.

csoffset: can't access character set translation file <filename>
Change the command to include a valid translation source-file name or move the desired translation source file to <filename>.

csoffset: <filename> is not a valid character set translation file
Alter the command to include a valid translation source-file name, or move the desired translation source file to <filename>. Other diagnostics describing the problem(s) with the translation source file also appear.

FILES

/usr/lib/cs.printer/*	printer character-set translation source files

SEE ALSO

lp(1), lpadmin(1M), lpsched(1M), lpstat(1)

cstrans(1)

cstrans - text-file character-set translator

SYNOPSIS

cstrans [options] [t=<filename>]

DESCRIPTION

Cstrans reads its standard input, translates incoming characters either in accord with output format options or through a translation table, and writes the resultant characters on its standard output. Arguments passed to cstrans(1) specify the input file format and either the output format desired or the path name of a file that contains the character translation source file to be used.

Cstrans(1) can translate a file from one standard internal character representation to another, or translate a file from internal character representations to nonstandard or device-dependent codes. When output format options are specified, translation is from one internal character representation to another. When a source file t=<filename> for a character-set translation is specified, translation is from an internal character representation to code sequences declared in the translation source file. In the latter case, for outbound characters that are not in character set 000 and that have no translation table entry, cstrans substitutes a question mark (?).

The output format arguments -of7, -od040, -o8, and -o16, are mutually exclusive from the t=<filename> argument, and the output format arguments -od040, -o8, and -o16, are mutually exclusive from one another.

The options are as follows:

-id040 Declares that the input is XSIS 058404 standard stringlets, with Motorola private character-set 040 as the default. Thus, if the input stream does not contain an initial CSelect (377) byte, then each following byte represents a character in character-set 040 until a CSelect byte does occur to change the character set.

Without this option, the input is handled as XSIS 058404 standard stringlets with character set 000 as the default.

-if7 Declares that the input format is ASCII characters with SO (shift-out, 016) and SI (shift-in, 017) bracketing characters for which the leftmost bit is considered set. For this format, SUB @ (032 100) represents the byte value 000 and SUB / (032 057) represents the byte value 377. The value 377 is a character-set selection (CSelect) byte to introduce standard internal character

stringlets. The value 000 is a character-set number in standard internal-character stringlets. Inbound SO and SI codes are treated as character-set selection codes and are not delivered as outbound codes.

You can use this option along with the -id040 option.

-of7 Declares that the output format is ASCII characters with SO (shift-out, 016) and SI (shift-in, 017) bracketing characters for which the leftmost bit is considered set. For this format, SUB @ (032 100) represents the byte value 000 and SUB / (032 057) represents the byte value 377. The value 377 is a character-set selection (CSelect) byte to introduce standard internal character stringlets. The value 000 is a character-set number in standard internal character stringlets.

Use this option with or without any of the -od040, -o16, and -o8 options.

-od040 Declares that the output is XSIS 058404 standard stringlets with no CSelect bytes. Each byte represents a character in Motorola private character-set 040. Each input character that can't be represented in the output stream is replaced by a question mark (?).

-o16 Declares that the output format is 16-bit (two bytes per character), XSIS 058404 standard, stringlet representation.

-o8 Declares that the output format is 8-bit, XSIS 058404 standard, stringlet representation with an initial CSelect byte.

When none of the arguments -od040, -o16, or -o8 is selected, the output format is XSIS 058404 standard, 8-bit, stringlet representation without an initial CSelect byte.

DIAGNOSTICS

Cstrans writes error messages of the following form on its standard error file.

```
cstrans: invalid option
  usage: cstrans [-id040 -if7] [[-of7 [-od040 | -o8 | -o16]] |
         t=<filename>]
```

Change the command to include one or more valid options.

```
cstrans: can't access character set translation file <filename>
```

Change the command to include a valid translation source-file name or move the desired translation source-file to <filename>.

cstrans: <filename> is not a valid character set translation file
Alter the command to include a valid translation source-file name, or move the desired translation source file to <filename>. Other diagnostics describing the problem(s) with the translation source file also appear.

SEE ALSO

cstty(1).

cstty(1)

cstty - character-set control for a terminal

SYNOPSIS

cstty [options]

DESCRIPTION

The cstty(1) command provides character-translation status and selection control for terminal I/O. To select translation, the itt(1M) command must have been run for the terminal type that is the subject of the cstty(1) command, but does not necessarily have to be run for getting status through cstty(1).

Cstty sets character-set, I/O translation options for the device that is the current standard input. Used without arguments, it reports the current settings of the character-set translation options. With the -a argument, it also reports the currently-available translation tables, as established by the itt(1M) command.

Cstty(1) acts on the character special file /dev/ttynnn that is its standard input. The user upon whose behalf cstty(1) executes must have read-access permission for that file to get status information and read/write permission to set options. Cstty(1) uses ioctl(2) system calls, with ioctl command forms CSGETO, CSSETO, and so on, to communicate with the terminal driver.

Always select character translation for TM31 Terminals so the input sequence produced by the ALT key goes to user processes as an ESC. (See Appendix B for more information.) In the prototype .profile file (/etc/user.profile), cstty(1) is executed if $TERM = tm31. Do not select character translation for TM30 Terminals.

Character-set translation-control options are listed below. Note that the combination of cst16 and cs040 options is not valid.

cstrans (-cstrans) Select (deselect) character-set translation. When you select this option, the terminal driver translates inbound and outbound device-dependent characters to and from standard internal character representations. If none of the three options for representing internal character sets (cst16, csfmt7, and cs040) is selected, inbound characters go from the terminal driver in XSIS 058404 standard, 1-byte stringlets of the form

CS8Declaration Char8Code Char8Code ...

Also, outbound characters that the terminal driver receives are treated as being in XSIS 058404 standard, string-encoded

format, where the defaults are 1-byte stringlets and character set 000.

cst16 (-cst16) Select (deselect) 2-byte stringlet, internal-character representation of the form

CS16Declaration Char16Code Char1C6ode ...

as the translation result for inbound characters. Outbound characters that the terminal driver receives are treated as being in XSIS 058404 standard, string-encoded format, where the defaults are 1-byte stringlets and character set 000.

csfmt7 (-csfmt7) Select (deselect) representation of internal character codes as 7-bit values. All bytes with a value greater than 177 and less than 377 are represented by sequences of the form

SO <value minus 200> SI

where

SO represents the ASCII shift-out character.

SI represents the ASCII shift-in character.

Bytes with the values 000 and 377 are represented as follows:

Byte	Representation
000	SUB @
377	SUB /

where

SUB is the ASCII substitute code for 032.

@ is the ASCII code 100.

/ is the ASCII code 057.

Inbound characters are translated into this 7-bit encoded representation. The occurrence of the SO, SI, SUB @, and SUB / characters is significant for outbound characters.

cs040 (-cs040) Select (deselect) use of Motorola private character set 040. Inbound characters are delivered from the terminal driver in XSIS 058404 standard, 1-byte stringlets of the form

stringlet8 ...

in which the CS8Declaration does not occur. All bytes represent single characters in character set 040. Outbound characters that the terminal driver receives are treated as

being in XSIS 058404 standard, string-encoded format, with the defaults of 1-byte stringlets and character set 040.

t=<arg> Define the terminal type and the user language. You must define this option on the first cstty call for a terminal session. If you omit it thereafter, the terminal and language identifiers remain unchanged.

The archetype form of <arg> is

<term>.<lang>

where

<term> is the terminal type, indicated by an ASCII string of two to eight characters.

<lang> identifies the user language, as given by the value of a two- to five-character ASCII string.

Include the .<lang> part of <arg> only for device types whose character sets are language specific, such as the TM31 Terminal. Valid <arg> values appear in the directory /etc/cs.term.

If you specify only <term>, this option redefines the terminal type but does not redefine the language identifier. If you specify only .<lang>, the language identifier is redefined, but the terminal type is not. Note that this second form requires the leading period (.).

The value of <arg> is often derived from the user's execution environment, as in the example

t="$TERM.$LANG"

This assignment of the <arg> value assumes that the shell sh(1) program interprets what this value is.

DIAGNOSTICS

Cstty writes error messages of the following form on its standard error file.

cstty: unknown option <option>
Change the command to include valid options.

cstty: no terminal type defined
Change the command to include a t=<terminal_type> argument with a valid terminal type value.

cstty: invalid option combination cst16 and cst040
Both cst16 and cst040 options cannot be specified. Alter the command to remove one of them.

cstty: don't have write permission for <standard_input_file>
Change the command to access the correct device, or become superuser and proceed as before.

cstty: <standard_input_file> is not a terminal
Alter the command to access the correct device.

cstty: <term.lang> translation table is not installed
Change the command to contain a correct t=<arg> option or install the desired translation table through the itt(1M) command.

For the first two error messages listed above, the following message also appears:

```
usage:
        cstty               - print current settings
        cstty -a            - print current settings and translation tables
        cstty <options>     - alter settings
```

SEE ALSO

stty(1), itt(1M), cstrans(1), ioctl(2), termio(7).

itt(1M)

itt - initialize terminal I/O translation

SYNOPSIS

itt [-r] [-s] t=<filename>

DESCRIPTION

Itt constructs a character-set translation data structure (through the csinit(3X) routine), given a character-set translation source file called <filename>. Through an ioctl(2) system call (with ioctl command = CSSETTT), itt sends the data structure to the terminal driver so that I/O translation can be selected for the terminal type (and possibly user language) to which that structure relates. Note that itt does not select terminal I/O translation; it only enables translation when it is selected through the cstty(1) command.

The terminal-type identifier, and the user-language identifier, if required for the terminal type, are extracted from the t=<filename> argument. This argument must be of the form

$TERM.$LANG

with or without the preceding path-name components. $TERM is the terminal-type identifier and $LANG is the user-language identifier. $LANG can be null for terminal types that do not have language-dependent characteristics, in which case, omit the .$LANG portion.

Itt commands are placed in the /etc/ttrc(1M) command script for execution as part of the central processor boot procedure. Itt is accessible as a global system administration command to the operating system superuser. The superuser can use itt to dynamically control the terminal driver's translation-table configuration (while the operating system is executing) and to gain information for use during system generation.

The options, which are mutually exclusive, are as follows. Note that the -s option is required for system administration purposes during system generation, specifically configuration of the operating system kernel.

The -r argument indicates to the terminal driver that the translation-table space associated with the character-set translation source file named <filename> can be reused; that is, that the translation table is uninitialized.

The -s argument causes itt to write the size of the initialized character-set translation data structure on standard output and not to send that structure to the terminal driver. This option gains kernel size information during the operating system generation procedure. If you include a number with this argument, the size value reported is in terms of units of that number of bytes. For example, -s64 yields the number of

64-byte units required to contain the character-set translation data structure named <filename>. Itt writes one line of the form

<nnn>

where <nnn> is the size (in units) required for the data structure.

FILES

/etc/cs.term/$TERM.$LANG (during system boot procedure)
/usr/lib/cs.term/$TERM.$LANG

DIAGNOSTICS

Itt writes error and informational messages of the following form on its standard error file. Note that when itt is executed from within /etc/ttrc during the system boot procedure, any diagnostic output is written to the file /etc/log/confile and does not appear on the terminal screen.

itt: missing filename argument
Change the command to include the name of a character-set translation source file. The itt exit code is set to 1.

itt: can't access character set translation file <filename>
Change the command to include a valid translation source-file name, or move the desired translation source file to <filename>. The itt exit code is set to 2.

itt: <filename> is not a valid character set translation file
Alter the command to include a valid translation source-file name or move the desired translation source file to <filename>. The itt exit code is set to 3.

itt: Can't de-install <translate_table_name> because it's in use
Use cstty(1) to locate the device that is using the translation table, and proceed as needed. The itt exit code is set to 4.

itt: Invalid command option(s)
Usage: itt -r t=<filename> - remove translation table
itt -s t=<filename> - report size of translation table
itt t=<filename> - install translation table
Alter the command to include valid options. The itt exit code is set to 5.

itt: You must be superuser to execute this command
This command requires superuser status to be executed. Become superuser and proceed as before, or get help from the system administrator. The itt exit code is set to 6.

itt: <translate_table_name> is already installed
This message is informational only. The itt exit code is set to 7.

itt: Insufficient space to load <translation_table_name>
There is not enough kernel space to load the specified translation table. One solution is to deinstall a translation table to get space, then proceed as before. The itt exit code is set to 8.

itt: <translation_table_name> is not installed
This message is informational only. The itt exit code is set to 9.

SEE ALSO

cstty(1), cstrans(1), ioctl(2), termio(7).

keyprompt(1)

keyprompt - application shell

SYNOPSIS

/usr/local/bin/keyprompt [script]

DESCRIPTION

Keyprompt is a simplified shell for end users. It has a simple command structure and a built-in help facility.

Keyprompt requires a TM31 Terminal.

Script, an ASCII text file, defines the applications available to the keyprompt user. Someone familiar with keyprompt conventions and operating system commands must build the script file for the user; see the script file conventions below. If the script file parameter is missing, use /usr/local/bin/keypromptmenu.

Using Keyprompt

When keyprompt starts, there are four regions on the screen, going from top to bottom:

- A message line. This holds messages from keyprompt or the application being run.
- A tag line. This labels the particular application or keyprompt display that is active.
- The window. The main display area used by the application. Text input and output to keyprompt or an application program go here. The window is a window into a virtual display. An application uses the virtual display just as it would the screen on a dumb terminal. The virtual display is the same size as the terminal's physical screen, but the window never shows all of the virtual display because parts of the screen are used by other regions. The application can move the cursor randomly within the virtual display; if the application moves the cursor out of the window, wm scrolls the window until it has the cursor again.
- The function key line. This tells the user what the function keys do.

Use the following keys to command keyprompt:

Function Keys — Execute one of the commands on the function key line. This may run an application or it may provide a new keyprompt display.

If a function key runs an application program, follow the conventions of the application, which may differ from keyprompt's. When the application finishes, the user is returned to the last keyprompt display.

If a function key provides a new keyprompt display, you get a new function key line. Function keys may lead to applications or more keyprompt displays.

The terminal's number keys also execute the function key line commands.

CTRL EXIT — This key combination terminates keyprompt. If keyprompt is the main shell, use CTRL EXIT to log off.

HELP — Enter help mode. When keyprompt enters help mode, a help message for the particular display appears. If the user presses a function key while in help mode, keyprompt displays a help message for that function key. If the user presses EXIT while in help mode, keyprompt returns to normal (function) mode.

EXIT — In normal mode, returns to original keyprompt display.

If the user types the name of the command (as it's displayed on the function key line), keyprompt executes that command, even if it's not on the current function key line.

Script File Conventions

The script file consists of a series of directives. Each directive has one of two forms:

```
keyword = name
keyword = "string"
```

The keyword specifies the particular directive.

There can be any number of spaces before or after keyword or the equal sign (=). Each new directive must begin on a new line. Each directive must be all on one line, except that string can include new-lines. Name is one to six characters long and string is zero to 1024 characters long.

Any number of blank lines are allowed between directives. They are ignored.

Two directives define names: group and key. No name can be defined twice.

The script file begins with global definitions; the remainder of the file is a series of group definitions. The global definitions define things that apply to all keyprompt displays; each group definition defines a particular keyprompt display. The first group defines the original display.

Global Definitions

These directives provide global definitons. Either can be omitted.

prompt = "string"

Display string when the user types a command name.

error = "string"

Display string when the user types an undefined command name.

Group Definitions

Each group definition begins with the following directives. The group directive is mandatory and must be first; all other directives are optional and can appear in any order.

group = name

Begin a group definition and define its name.

tag = "string"

Print string on the tag line to identify the particular keyprompt display.

help = "string"

Print string upon entering help mode.

message = "string"

Print string on the message upon entering this display.

text = "string"

Print string in the window upon entering this display.

The remainder of the group definition consists of up to 10 key definitions.

Key Definitions

A key definition consists of the following directives. The key directive is mandatory and must be first. The other directives may be in any order; they are all optional, except that the definition must have a branch or command directive, but not both.

key = name

Begin and identify a key definition. Name appears as the function key label on the function key line. User can execute this command by typing name.

id = name

Specify which function key this definition defines. Name must be one of f1, f2, f3, f4, f5, f6, f7, f8, f9, or f10.

NOTE

Although name must be f10, you must press the RESET key to specify the f10 function.

A definition that lacks an id directive can be accessed only by name.

branch = name

This key switches to the keyprompt display defined by group name.

command = "string"

This key executes an application. When the user presses this key, keyprompt uses the Bourne shell to execute string.

tag = "string"

Print string on the tag line when the user invokes this command.

help = "string"

Print string when the user presses this key while in help mode.

message = "string"

Print string on the message line when the user invokes this command.

text = "string"

Print string in the window upon executing this key.

SEE ALSO

sh(1)

lpadmin(1M)

lpadmin - configure the LP spooling system

SYNOPSIS

```
/usr/lib/lpadmin -pprinter [options]
/usr/lib/lpadmin -xdest
/usr/lib/lpadmin -d[dest]
```

DESCRIPTION

Lpadmin configures LP spooling systems to describe printers, classes, and devices. Use lpadmin to

- Add and remove destinations.
- Change membership in classes.
- Change devices for printers.
- Change printer interface programs.
- Change the system default destination.

You may not use lpadmin when the LP scheduler, lpsched(1M), is running, except where noted below.

You must use exactly one of the -p, -d, or -x options for every legal invocation of lpadmin.

-d[dest] Makes dest, an existing destination, the new system default destination. If dest is not supplied, then there is no system default destination. This option can be used when lpsched(1M) is running. No other options are allowed with -d.

-xdest Removes destination dest from the LP system. If dest is a printer and is the only member of a class, then the class is deleted, too. No other options are allowed with -x.

-pprinter Names a printer to which all of the options below refer. If printer does not exist, then it will be created. The printer options are as follows and can appear in any order. The printer is refered to as P.

-cclass Inserts printer P into the specified class. The class is created if it doesn't already exist.

-eprinter Copies an existing printer's interface program to be the new interface program for P.

-h	Indicates that the device associated with P is hardwired. This option is assumed when creating a new printer unless the -l option is supplied.
-iinterface	Establishes a new interface program for P. Interface is the path name of the new program.
-l	Indicates that the device associated with P is a login terminal. The LP scheduler, lpsched, disables all login terminals automatically each time it is started. Before reenabling P, you should establish its current device using lpadmin.
-mmodel	Selects a model interface program for P. Model is one of the model interface names supplied with the LP software (see "Models," below).
-rclass	Removes printer P from the specified class. If P is the last member of the class, then the class is removed.
-vdevice	Associates a new device with printer P. Device is the path name of a file that is writable by the LP administrator, lp. Note that there is nothing to stop an administrator from associating the same device with one or more printers. If you supply only the -p and -v options, then you can use lpadmin while the scheduler is running.

Restrictions

When creating a new printer, you must use the -v option and exactly one of the -e, -i, or -m options. The -h and -l keyletters are mutually exclusive. Printer and class names can be no longer than 14 characters and must consist entirely of alphanumeric characters and underscore (_).

Models

The LP software supplies the model printer interface programs, which are shell procedures that interface between lpsched and devices. All models reside in the directory /usr/spool/lp/model and can be used as-is with lpadmin -m. Alternatively, LP administrators can modify copies of models and then use lpadmin -i to associate them with printers. The models and the options they can be given on the lp command line using the -o keyletter are as follows:

dumb Interface for a line printer without special functions and protocol. Form feeds are assumed. This model is a good one to copy and modify for printers that don't have models.

1640 Interface for a Diablo 1640 terminal running at 1200 baud, using XON/XOFF protocol. Options:

-12 12-pitch. The default is 10-pitch. This option should never be used in conjunction with nroff.

-f Don't use the 450(1) filter. The output has been preprocessed by either 450(1) or the nroff 450 driving table.

hp Interface for a Hewlett Packard 2631A line printer at 2400 baud. Options:

-c Compressed print

-e Expanded print

prx Interface for a Printronix P300 printer using XON/XOFF protocol at 1200 baud.

ptnnx Motorola System 5300 printers. Options:

B or -B	Suppress banner pages.
F or -F	Suppress form feed after each copy.
f or -f	Use "cat" as the filter program (effectively no filter).
g or -g	Use 450-style graphics (450 is the filter).
L* or -L*	List of file names for banner pages, where * is the list of file names.
o or -o	No offset from the left margin (selected by default).
o<n> or -o<n>	Offset left margin by n spaces.
p<n> or -p<n>	Set pitch to n characters per inch.
q* or -q*	Deselect mapping of new-line to carriage return (-onlcr) and suppress form feed between copies.
u?* or -u?*	User name for banner pages, where ?* is a string of at least one character.
xs or -xs or xs=y* or -xs=y*	Select XSIS 058404 standard, encoded strings as the input-file character-set representation.

xs=n* or -xs=n* Select Motorola private character-set 040 as the input-file character-set representation. Selected by default.

Examples

1. Assuming there is an existing Hewlett Packard 2631A line printer named hp2, it will use the hp model interface after the command

```
/usr/lib/lpadmin -php2 -mhp
```

2. To obtain compressed print on hp2, use the command

```
lp -dhp2 -o-c files
```

3. A Diablo 1640 printer called st1 can be added to the LP configuration with the command

```
/usr/lib/lpadmin -pst1 -v/dev/tty020 -m1640
```

4. An nroff document can be printed on lp in any of the following ways:

```
nroff -T450 files | lp -dst1 -of
nroff -T450 -12 files | lp -dst1 -of
nroff -T37 files | col | lp -dst1
```

5. The following command prints the password file on st1 in 12-pitch:

```
lp -dst1 -o12 /etc/passwd
```

FILES

/usr/spool/lp/*

SEE ALSO

accept(1M), enable(1), lp(1), lpsched(1M), lpstat(1), lp.cnfg(1M). Lp.cnfg(1M) is documented in the Software Release Guide.

nw(1)

nw - initialize new window

SYNOPSIS

nw

DESCRIPTION

Nw provides a mechanism for executing "set-up" commands in a new window created by the window manager wm(1), analogous to the execution of a .profile file by sh(1) at the outset of a terminal session. Setting the environmental variable SHELL=nw causes the window manager to execute the command nw (/usr/local/bin/nw) as the shell program in each window created, including the first window. The nw command uses the environment that was in effect at the time the window manager was initiated as the environment for the new window. If, however, the variable NWSHELL is defined in the original environment (as in NWSHELL=myshell), then in the new window, the value of SHELL is set to the value of NWSHELL (as in SHELL=myshell). Nw executes that SHELL ("myshell") as the shell program in the new window. If NWSHELL is undefined, nw executes the system default shell in the new window.

If the variable NWARGS is defined in the environment, nw submits the value of NWARGS as one or more commands for execution by the system default shell program. When embedded quotation marks occur in the value of NWARGS, nw submits the first substring for execution by sh(1), waits for execution to complete, then submits the next substring for execution by another instance of sh(1), and so on.

EXAMPLES

The line

```
NWARGS="cstty $CS"
```

causes the cstty command to establish the original cstty settings in the new window, assuming that these settings were saved in the environment variable CS.

The line

```
NWARGS="cstty $CS; echo Hello!; date"
```

causes execution of these commands in the new window: cstty; echo, which displays "Hello!"; and date, which displays the current date.

The line

```
NWARGS='"echo Command 1" "echo Command 2"'
```

causes nw to submit "echo Command 1" for execution by sh(1), wait for that shell to terminate, then submit "echo Command 2" for execution by sh(1).

SEE ALSO

wm(1), sh(1)

tdl(1)

tdl - download terminal through RS-232 connection

SYNOPSIS

tdl <filename>

DESCRIPTION

Tdl copies a terminal download file called <filename> to the terminal from which this command was entered. The download file is transmitted through the terminal's RS-232 connection. Tdl displays a message giving the <filename> of the download file it is trying to copy to the terminal. The <filename> argument is the path name of an RS-232 terminal download image, which is an executable file loaded into the processor in the terminal.

Tdl is used to download TM30 and TM31 Terminals that contain boot ROM version 2.0 and can be used at different baud rates. A sequence of hyphens (-) appears on the screen as the download operation proceeds. When the download operation is complete and the screen is cleared, you may need to press RETURN to continue operation of the terminal.

FILES

/usr/lib/iv/<term>.<protocol>.<lang>

where

<term> is the terminal-type identifier.

<protocol> indicates the kind of connection: 4 indicates RS-422, 2 means RS-232.

<lang> is a user-language identifier. If the terminal type has no language-dependent characteristics, omit the .<lang>.

Sample file names for tdl are

```
tm30.2
tm31.2.deut
tm31.2.engli
```

Note that download files present in the system's terminal download area (partition 0 of the system disk) also reside in the directory /usr/lib/iv/dl.term.

DIAGNOSTICS

Tdl writes error messages of the following form on its standard error file.

tdl: cannot open <filename>
Alter the command to include a valid terminal download file name.

tdl: conversion error
Change the command to include a valid terminal download file name.

SEE ALSO

tmdl(1)

BUGS

Tdl fails with a bus error, produces a core file, and leaves the terminal in an unusable state if it is executed with no <filename> argument. To regain use of the terminal, disconnect and reconnect its RS-232 cable from the processor (not the terminal) to get a new login prompt.

tmdl(1)

tmdl - download terminal through RS-232 connection

SYNOPSIS

tmdl <filename>

DESCRIPTION

Tmdl copies a terminal download file called <filename> to the terminal from which this command was entered. The download file is transmitted through the terminal's RS-232 connection. The <filename> argument is the path name of an RS-232 terminal download image, which is an executable file loaded into the processor in the terminal.

Tmdl is used to download TM30 and TM31 Terminals that contain boot ROM version 1.0 and can be used at 9600 baud only. A sequence of hyphens (-) appears on the screen as the download operation proceeds. When the download operation is complete and the screen is cleared, you may need to press RETURN to continue operation of the terminal.

FILES

/usr/lib/iv/<term>.<protocol>.<lang>

where

<term> is the terminal-type identifier.

<protocol> indicates the kind of connection: 4 indicates RS-422, 2 means RS-232.

<lang> is a user-language identifier. If the terminal type has no language-dependent characteristics, omit the .<lang>.

Sample file names for tmdl are

```
tm30.2
tm31.2.deut
tm31.2.engli
```

Note that download files present in the system's terminal download area (partition 0 of the system disk) also reside in the directory /usr/lib/iv/dl.term.

DIAGNOSTICS

Tmdl writes error messages of the following form on its standard error file. When the terminal is running in emulator mode (as when the terminal is not downloaded), such diagnostic messages appear somewhat garbled.

tdl: cannot open <filename>
Alter the command to include a valid terminal download file name.

tdl: <filename> is not a valid runfile
Change the command to include a valid terminal download file name.

SEE ALSO

tdl(1)

wm(1)

wm - window management

SYNOPSIS

exec /usr/local/bin/wm

DESCRIPTION

Wm is the window manager. It provides services to application programs running under its control and to users using terminals under its control. The window manager can divide the terminal screen into windows that the user can use like separate terminals. Other services include placement, size, scrolling, and synchronization of windows. Wm requires a TM31 Terminal on a cluster line. The window manager must be running for the window management library functions to work.

The window manager is normally executed in place of the user's login shell by the exec command in /etc/profile or by the user's own .profile. The window manager then executes the user's shell each time the user splits a window. The SHELL environment variable (normally set by login(1M) to /bin/sh) provides the full path name of the initial program run in the windows.

Setting SHELL=nw in the environment causes wm to execute the nw(1) command as the "shell" in each new window. Nw provides window initialization services and executes the shell named by the environment variable NWSHELL.

When wm starts, the user sees four regions on the screen, going from top to bottom:

- A message line. A single line, always at the top of the screen. It holds messages and prompts from application programs.
- A tag line. A single line, always above each window. It labels the particular application program or display that is active in the window.
- The window. The main display area used by programs. Text input and output to the shell or an application program goes here. The window is a window into a virtual display. An application program can use the virtual display as a 28-line screen, regardless of the size of the window. The virtual display is usually larger than the window. Normally, the window manager automatically positions the window over the part of the virtual display that contains the cursor. If the user program moves the cursor to a part of the virtual display not in the window, the window manager scrolls the window until the cursor is visible again. The user can also scroll the display (see below).

- The function key line. A single line, always at the bottom of the screen, that labels the functions keys for the currently active window.

Wm accepts user commands activated by the ACTION key; these commands are not seen by user program. Use the ACTION key like the CTRL or SHIFT keys: hold down the ACTION key and press the other key used with it. Holding down the ACTION key changes the function key line to show how ACTION changes the meanings of the function keys.

The valid wm user commands are described in Table 4-1.

Table 4-1. Wm Commands

Keys	Function
	Splitting the Active Window
ACTION- RESET (SPLIT)	Split the active window to create a new window. The new window and its tag line replace the bottom half of the window being split. Any program running in the old window is unaffected. The virtual display of the old window is unchanged, though less of it is visible. The user shell then starts up in the new window. The new window is active; all other windows are inactive. Programs running in inactive windows continue to run, but input calls will not return until you reactivate the window and type something. Keyboard input goes to the active window. Each window, whether active or inactive, has its own message line, function key line, and cursor, but the terminal displays them only if they belong to the active window. (Application programs can also make the cursor invisible.) If an application program in an inactive window writes to the message line, the message is not visible until you make that window active again. On TM31 Terminals, the active window's tag line is displayed full intensity with the other tag lines displayed half intensity. To get rid of a window, terminate all the programs running in it. If an application is running in the window under a user shell, you must first terminate the application (probably by pressing CTRL EXIT), then terminate the shell (again, probably by pressing CTRL EXIT). If the top window is removed, the window above it takes over its space.

Table 4-1. Wm Commands (Continued)

Keys	Function
	The SPLIT key becomes inoperative if the terminal already displays its maximum number of windows or if a user program has disabled window splitting.
Switching the Active Window	
ACTION-F9 (BELOW)	The window below the active window becomes the active window and the old active window becomes inactive. The new active window takes over the message line and the function key line, and its cursor becomes visible.
	ACTION-↓ is the same as ACTION-F9.
ACTION-F8 (ABOVE)	The window above the active window becomes the active window. ACTION-↑ is the same as ACTION-F8.
ACTION-n	Activate window n, where n is a number from one to four. A window's number is assigned when it is first created, with the new window getting the lowest unused number. Unless erased by a user program, the window number is displayed on the left end of the tag line.
Moving the Active Window	
ACTION-F7 (SWAP ↓)	The active window and the window below it trade places.
ACTION-F6 (SWAP ↑)	The active window and window above it trade places.
Shrinking the Active Window	
Shrinking the top window increases the size of the window below; shrinking any other window increases the size of the window above.	
ACTION-F5 (SHRINK)	The active window decreases in size by one line. Ignored if the window is already zero lines long (only the tag line is visible).
ACTION-SHIFT-F5	The active window decreases in size by four lines. If the window is already less than four lines long, it becomes zero lines long.
ACTION-CTRL-F5	The active window becomes zero lines long.

Table 4-1. Wm Commands (Continued)

Keys	Function
	Growing the Active Window
Growing the top window decreases the size of the window below; growing any other window decreases the size of the window above. If the window that would otherwise shrink is already zero lines long, the next window shrinks. If all the windows below the second or third window are zero lines long, space comes from the windows above.	
ACTION-F4 (GROW)	The active window increases in size by one line. Ignored if the other windows are all zero lines long.
ACTION-SHIFT-F4	The active window increases in size by four lines. If the other windows don't have four lines to spare, the active window increases until all other windows are zero lines long.
ACTION-CTRL-F4	The active window increases in size until all other windows are zero lines long.

EXAMPLE

The following text in the user's .profile provides a System 6300 user with window management, and keyprompt as a shell. The user's shell field in the password file must be blank or set to /bin/sh.

```
export SHELL
SHELL=/usr/local/bin/keyprompt
tty='tty|sed sX/dev/ttyXXX'
if [$tty -ge 020 -a $tty -le 027]
then
     exec wm
else
     exec $SHELL
fi
```

SEE ALSO

sh(1), nw(1)

DIAGNOSTICS

Wm(1) may display diagnostic messages on the terminal screen. If a message says to press the GO key to acknowledge an error condition, press CTRL D to exit the window.

WARNING

If a program quickly outputs two display lines at the virtual display's top and bottom, the user can easily miss one of them.

csinit(3X)

csinit - initialize a character-set translation table

SYNOPSIS

```
#define  CSMAXSIZ       1
#include <cs.h>
#include <ctype.h>
#include <stdio.h>

struct csttbl *
csinit (filename, silent, status)
register char *filename;
register int silent;
register int *status;
```

DESCRIPTION

Csinit(3X) constructs a character-set translation data structure from a character-set translation source file. Csinit reads the source file named by its <filename> argument, converts it to a csttbl character-set translation table, and returns a pointer to that structure. Validation of the character-set translation source file is performed. The RETURN value of csinit is NULL if the conversion operation was unsuccessful. The csttbl structure is shown in Figure 4-1.

```
/*
 * Character set translation table argument.
 * The user program should define CSMAXSIZ as the maximum translation
 * table size it is prepared to handle and set cs_tmax to that value.
 */
#ifdef CSMAXSIZ
struct  csttbl  {
        int             cs_tmax;        /* should be set to CSMAXSIZ */
        union   {
                struct cstthdr  cs_hdr;
                char            cs_tbl[CSMAXSIZ];
        }cs_u;
};
#endif
```

Figure 4-1. Csttbl Structure

Csinit arguments are:

<filename> Name of the character-set translation source file.

<silent> Flag to select or deselect printing of error messages. If the <silent> argument is FALSE, then diagnostics are written on the standard error file.

<status> Status word to reflect completion status. Values for completion status are defined in the cs.h header file.

This routine resides in the file /usr/lib/libcs.a. The program must be loaded with the object-file, access-routine, library libcs.a.

DIAGNOSTICS

When the <silent> argument is FALSE, csinit writes error messages of the following form on its standard error file. The %d represents the line number of the translation table at which the error occurred; %n represents the character-set number.

line %d - redeclaration of character set %n
line %d - undefined character set number %n
line %d - format7 statement unexpected
line %d - inbound statement unexpected
line %d - outbound statement unexpected
line %d - number of entries does not match defined range
line %d - translate statement missing accent value
line %d - translate statement missing character set number
line %d - translate statement missing high range value
line %d - translate statement missing input sequence
line %d - translate statement missing low range value
line %d - no primary character set defined
line %d - translate statement missing range keyword
line %d - syntax error

SEE ALSO

cstrans(3X), cstermio(7)

cstrans(3X)

cstrans - perform character-set translation

SYNOPSIS

```
#include <sys/csintern.h>
#include <cs.h>

cstrans (csdp)
    register CSDATP     csdp;
```

DESCRIPTION

Cstrans translates characters from one buffer to another through a translation table. It translates characters until either the output buffer becomes full or the input buffer is empty.

Its argument, <csdp>, is the address of a data structure that points to an input buffer, a translation table, and an output buffer, and contains information describing the current state of the translation.

This subroutine package handles translation of data that may be represented as XSIS 058404 strings, external device codes, or internal 16-bit characters. Input data is in a buffer of unsigned char or short. The output data is placed into a similar buffer. For outbound characters that are not in internal character-set 0 and that have no declared entry in the translation table, cstrans(3X) substitutes a question mark character (?).

There are five translation modes, all of which use an internal 16-bit character input or output buffer, with the other buffer being either 8-bit characters or internal 16-bit characters. Table 4-2 describes these modes.

Table 4-2. Character Translation Modes

Mode	Function
0	Translate from internal 16-bit to internal 16-bit using an internal translation table. This mode either enforces the Motorola private character-set or avoids the character sets for Motorola private, ligature, and accented characters.
1	Translate from external-device character code to internal 16-bit characters through an external-device translation table.
2	Translate from internal 16-bit characters to XSIS 058404 strings with options for 16-bit stringlets or for 7-bit representations.

Table 4-2. Character Translation Modes (Continued)

Mode	Function
3	Translate from XSIS 058404 strings to internal 16-bit characters.
4	Translate from internal 16-bit characters to external-device character codes through an external-device translation table.

As an output filter, three translations would be applied in sequence:

Mode 3 → Mode 0 using cs_tostd → Mode 4 using a device-specific translation table

To reformat XSIS 058404 strings, four translations could be applied. For example, to reformat them to Motorola private character-set 040 strings, use the following sequence:

Mode 3 → Mode 0 using cs_tostd → Mode 0 using cs_topri → Mode 2

Other combinations of translation modes can be used. The only requirement is that each output buffer must be in the form expected for the next translation's input buffer.

The external, device-translation input sections must provide characters in the standard internal character sets, avoiding sets 040, 360, and 361. They may assume that their input comes from that same standard form. The cs_tostd translation table is applied to input strings to ensure the standard input form.

This convention means that output translation tables do not have to handle all the different forms that are legal. For example, the A dieresis symbol (Ä) can be represented in three different internal forms:

```
<000><310> <000><101>     standard form: dieresis and "A"
<361><047>                the accented character rendering
<040><241>                the Motorola private form
```

Also, for devices that accept the ISO forms, no translation is required. For some hardcopy devices that don't accept the ISO form, the accents can still be mapped to <accent> and <backspace>.

This routine resides in the file /usr/lib/libcs.a. The program must be loaded with the object-file, access-routine library libcs.a.

SEE ALSO

csinit(3X)

cstermio(7)

cstermio - terminal I/O character set interface

DESCRIPTION

Terminal input-character sequences can be translated from device-dependent character codes to internal character-set representations (XSIS 058404 character-code standard, including Motorola private character-set 040) prior to receipt by a user process. Similarly, outbound characters destined for a terminal can be translated from internal character-set representations to device-dependent output sequences. Such character-set translations are handled by the terminal driver.

There are two aspects to terminal I/O character-set handling:

- Managing terminal I/O translation tables at the system-wide level.
- Controlling terminal I/O character-set translation options for each active terminal.

Managing terminal I/O translation tables is a superuser responsibility and can be done through the itt(1M) command. An interface to translation table management is also provided through the ioctl(2) system call. The ISP extends the interface to ioctl(2) to accommodate status requests and changes to data and option settings.

Csttbl Structure

The ioctl(2) system calls that apply to managing terminal I/O character-set translation tables use the structure shown in Figure 4-2 and the structures pointed to therein. All of these structures are defined in <cstty.h>. The fields in this structure are explained in Table 4-3.

```
/*
 * Character set translation table argument for CSGETTT and CSSETTT.
 * The user program should define CSMAXSIZ as the maximum translation
 * table size it is prepared to handle and set cs_tmax to that value.
 */
#ifdef CSMAXSIZ
struct  csttbl  {
        int         cs_tmax;        /* should be set to CSMAXSIZ */
        union   {
                struct cstthdr  cs_hdr;
                char            cs_tbl[CSMAXSIZ];
        }cs_u;
};
#endif
```

Figure 4-2. Csttbl Structure (Page 1 of 2)

```
/* description of a character set translation table header */
struct cstthdr {
   ushort   cs_tnum;      /* table number */
   ushort   cs_tlen;      /* length of the complete translation table
                             in bytes */
   csttname cs_tname;     /* name of the translation table */
   ushort   cs_nref;      /* number of open file references */
   ushort   cs_pesc;      /* position of the escape prefix index */
   ushort   cs_nesc;      /* number of escape sequence prefixes */
   ushort   cs_pchset;    /* position of the character set index */
   ushort   cs_nchset;    /* number of character sets */
   ushort   cs_ptrchar;   /* position of the table of 16-bit
                             translated characters */
   ushort   cs_nextcs;    /* number of external character sets */
   ushort   cs_pextcs;    /* position of the table of escape
                             sequences used to select the external
                             character sets */
   ushort   cs_ttflag;    /* translation table flag bits */
};

typedef struct {
   char     dev[9];       /* terminal or printer device name */
   char     lang[7];      /* name of the language */
} csttname;


#if !KERNEL || defined_io

/*
 * Character set option flag bits for cs_ttflag
 */
#define CSEXTSO      1         /* if the external device codes use SO
                                  and SI to indicate that bit 7 is on
                                */
#define CSINTERN     2         /* set for internal XSIS 058404 to
                                  XSIS 058404 translation tables */

#endif defined_io
```

Figure 4-2. Csttbl Structure (Page 2 of 2)

Table 4-3. Csttbl Structure Fields

Field	Description
cs_tnum	Number of this translation table. Identifies the translation table slot in kernel space for this translation table. The first slot is cs_tnum == 0. • A CSGETTT with a cs_tnum of a translation table that currently occupies a slot returns a copy of the translation table from that slot. • A CSGETTT with a cs_tnum of an empty slot sets errno == ENXIO. • A CSSETTT with a cs_tnum of an occupied slot writes over the translation table in that slot unless the table is currently in use, in which case errno == EBUSY is set. • A CSSETTT with a cs_tnum of an empty slot results in the translation table filling that slot. Unsuccessful ioctl CSGETTT and CSSETTT calls set the value of the pointer to the csttbl structure to -1, and errno is set appropriately.
cs_tlen	Size of the csttbl structure in bytes. See the CSMAXSIZ comment in Figure 4-2.
cs_tname	Name of the translation table (csttbl structure). The values for dev and lang in the structure type csttname are NULL-terminated strings representing TERM (terminal type) and LANG (user-language identifier), usually as contained in the execution environment.
cs_nref	Number of users of the translation table. Set by the terminal driver to indicate whether or not the translation table is in use.
cs_pesc	Offset of the first index structure of the csescix escape prefix in the csttbl.cs_tbl character array.
cs_nesc	Number of csescix escape prefix index structures in the csttbl.cs_tbl character array.
cs_pchset	Offset of the first cscsix character-set index structure in the csttbl.cs_tbl character array.
cs_nchset	Number of cscsix character-set index structures in the csttbl.cs_tbl character array.

Table 4-3. Csttbl Structure Fields (Continued)

Field	Description
cs_ptrchar	Offset of the Char16Code translated characters table in the csttbl.cs_tbl character array.
cs_nextcs	Number of character sets declared as supported by the device to which this translation table applies. Equal to the number of entries in the cs_pextcs ushort array.
cs_pextcs	Offset of a ushort array of indexes to NULL-terminated strings that contain output sequences for device character-set selection.
cs_ttflag	cs_ttflag == CSEXTSO indicates that the device uses 7-bit codes with the ASCII SO (016) code to designate that the logical 8th bit is set on ensuing bytes until an ASCII SI (017) code is received. cs_ttflag == CSINTERN indicates that the translation operation is between internal character-set representations, not device-dependent code sequences.

Csescix Structure

The csescix structure is used in mapping device-dependent inbound codes to Char16Code internal codes. This structure is shown in Figure 4-3. Descriptions of the fields in the csescix structure are given in Table 4-4.

```
/* description of an escape prefix index */
struct csescix {
   unsigned char  cs_esc[4];     /* escape sequence prefix */
   unsigned char  cs_esclo;      /* lowest valid character after
                                    prefix */
   unsigned char  cs_eschi;      /* highest valid character */
   ushort         cs_esctt;      /* position of the translation table
                                    table */
};
```

Figure 4-3. Csescix Structure

Table 4-4. Csescix Structure Fields

Field	Description
cs_esc[4]	NULL-terminated string of characters constituting a device-dependent escape sequence that precedes a character to be translated. An example of such a sequence is ESC N x, where ESC N is the escape sequence prefix,and x is a variable value.
cs_esclo	Lowest value of a range of characters subject to translation that follows the escape sequence prefix defined by cs_esc[4]. An example of this value is the lowest value for x in the sequence ESC N x.
cs_eschi	Highest value of a range of characters subject to translation that follows the escape sequence prefix defined by cs_esc[4]. An example of this value is the highest value for x in the sequence ESC N x.
cs_esctt	Offset of the csttent translation-table entry in the csttbl.cs_tbl character array associated with this escape sequence prefix.

Cscsix Structure

The cscsix structure, shown in Figure 4-4, is used in mapping Char16Code internal codes to device-dependent outbound codes. The fields in the cscsix structure are described in Table 4-5.

```
/* description of a character set index */
struct cscsix {
      unsigned char  cs_csnum;      /* character set number */
      unsigned char  cs_cspfx[2];   /* possible prefix character(s)
                                       (character set 000) for
                                       accented characters and
                                       ligatures */
      unsigned char  cs_nlist;      /* number of list entries */
      ushort         cs_plist;      /* position of list entries */
      ushort         cs_cstt;       /* position of the translation
                                       table */
};
```

Figure 4-4. Cscsix Stucture (Page 1 of 2)

```
#if !KERNEL || defined_io

/* used if cs_nlist == 0, so all values between cs_cslo and cs_cshi
   have translation tables.  Otherwise, cs_plist points to a list of
   the valid codes.
 */
#define cs_cslo(e)   (((e)->cs_plist)>>8)     /* lowest valid char. */
#define cs_cshi(e)   (((e)->cs_plist)&0xff)   /* highest valid char. */

#endif defined_io
```

Figure 4-4. Cscsix Stucture (Page 2 of 2)

Table 4-5. Cscsix Structure Fields

Field	Description
cs_csnum	CharSet8 value designating the character set to which this cscsix structure applies. Legal values defined by the character-code standard are 000, 040 through 176, and 241 through 376.
cs_cspfx[2]	Contains (in cs_cspfx[0]) a character-set 000 accent character.
cs_nlist	Number of entries in the list of code sequences for outbound characters in the csttbl.cs_tbl character array. See the above definitions of cs_cslo and cs_cshi for the special meaning of cs_nlist == 0.
cs_plist	Offset of the list of valid accented letters and code sequences for outbound characters in the csttbl.cs_tbl character array associated with this character-set index when cs_nlist != 0.
cs_cstt	Offset of the csttent translation table entry in the csttbl.cs_tbl character array associated with this character-set index.

Csttent Structure

The csttent structure and field descriptions are shown in Figure 4-5.

```
/* description of translation table entry */
struct csttent {
        ushort      cs_tttyp:4;         /* entry type code */
        ushort      cs_ttval:12;        /* low bits of accent
                                           character */
};

#if !KERNEL || defined_io

/* the cs_tttyp values are: */
#define CS_NOCHG     0    /* value unchanged by translation */
/*                   1-7     number of 16-bit characters in entry */
#define CS-CSO       8    /* cs_ttchar in character set 000+cs_ttnib
                             */
#define CS_CS40      9    /* cs_ttchar in character set 040+cs_ttnib
                             */
#define CS_ACC       14   /* C0+cs_ttnib plus cs_ttchar in character
                             set 000 */
#define CS_ERR       15   /* invalid character */

/* the cs_ttval field may be redefined as: */
#define cs_ttnib(e)      ((e)->cs_ttval >> 8)
#define cs_ttchar(e)     ((e)->cs_ttval & 0x0ff)

#endif defined_io

/* For cs_tttyp values of 1 through 7, the cs_ttval field contains a
   subscript into the array of ushort translated character
   sequences.  These 16-bit entries contain an 8-bit external
   character set number and an 8-bit character value (when cs_nextcs
   is greater than zero), an 8-bit character set number and an 8-bit
   character value (when CSINTERN is set in cs_ttflag), or an 8-bit
   escape sequence prefix number and an 8-bit char suffix.

   For cs_tttyp values of CS_CSO, CS_CS40 and CS_ACC, cs_ttchar
   contains a character value in the indicated character set.  For CS_
   ACC, cs_ttnib contains the low four bits of the accent character.
   For example, the csttent value 0xe841 represents a dieresis
   (0x00c8) followed by an uppercase A (0x0041).
 */
```

Figure 4-5. Csttent Structure

A complete character-set translation table is shown in Figure 4-6.

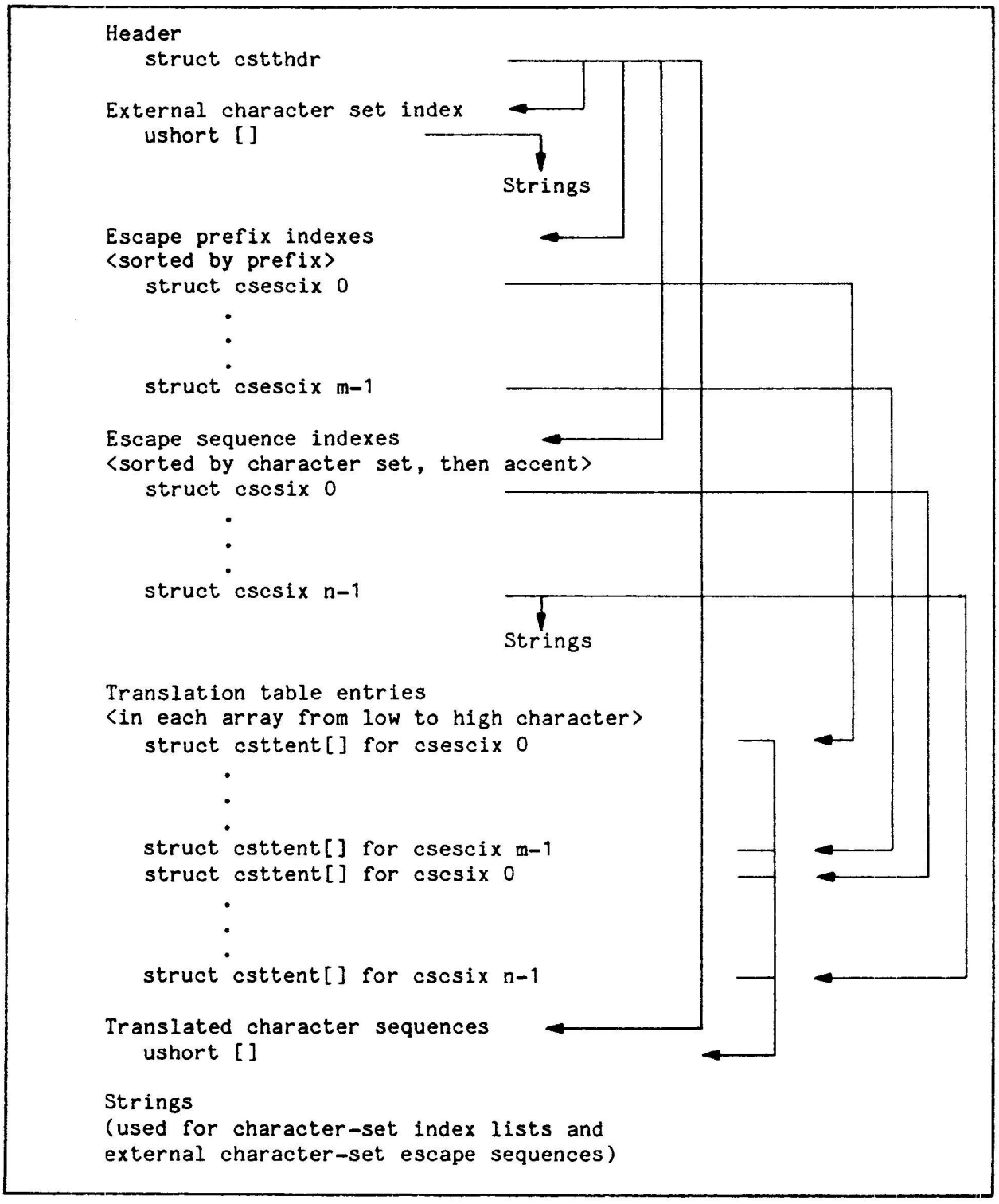


Figure 4-6. Character-Set Translation Table Structure

Control of Terminal Character-Set Options

There are two ways to control terminal I/O character-set translation options. You can use the cstty(1) command or the interface provided through the ioctl(2) system calls. Figure 4-7 shows the structure, defined in <cstty.h>, that these calls use.

```
/*
 * Character set option argument for CSGETO and CSSETO/OW/OF
 */
struct     csopt {
           ushort    cs_options;     /* option bits */
           csttname  cs_name;        /* name of the character set
                                        translation table */
};

typedef struct {
      char           dev[9];   /* terminal or printer device name */
      char           lang[7];  /* name of the language */
} csttname;
```

Figure 4-7. Csopt Structure

The cs_options values have the following meanings, which are defined in <cstty.h>. See the description of cstty(1) options for more information on these options.

```
#define CSTRANS        0001      /* Select translation */
#define CS040          0002      /* Select character set 040 */
#define CSFMT7         0004      /* Select 7-bit SO/SI+SUB codes */
#define CST16          0010      /* Select 16-bit defined strings */
```

The values for dev and lang in the structure type csttname are NULL-terminated strings representing TERM (terminal type) and LANG (user language identifier), usually as contained in the execution environment.

Ioctl(2) calls to manage terminal I/O character-set translation tables have the form:

```
ioctl (fildes, command, arg)
csttbl *arg;
```

The commands using this form are:

CSGETTT Get the character-set translation table parameters associated with the csttbl structure referenced by <arg>.

CSSETTT Set the character-set translation table parameters associated with the structure referenced by <arg>.

Ioctl(2) calls to control terminal I/O character-set translation options for each active terminal have the form:

```
ioctl (fildes, command, arg)
IS_xctl *arg;
```

The commands using this form are:

CSGETO Get the parameters associated with character-set translation for the terminal and store them in the IS_xctl structure referenced by <arg>.

CSSETO Set the character-set translation parameters for the terminal from the structure referenced by <arg>. The change is immediate.

CSSETOW Wait for the output to drain before setting the new parameters. Use this form when you change parameters that will affect output.

CSSETOF Wait for the output to drain, then flush the input queue, and set the new parameters.

FILES

/dev/tty*	terminal or terminal-like device character special file
/etc/cs.term/*	terminal character-set translation source files installed at boot time
/usr/lib/cs.term/*	terminal character-set translation source files

SEE ALSO

stty(1), itt(1M), cstty(1), ioctl(2), termio(7)

FILES

The ISP files listed below are C programming language header files and the file containing the international support library routines.

/usr/include/cs.h

This ISP file is a C programming language header file containing definitions of external symbols and data declarations. The ISP central processor software components and the user programs that interface with those components use this file.

NAME /usr/include/cs.h

FORMAT This is a C programming language header file.

/usr/include/sys/cstty.h

This ISP file is a C programming language header file containing definitions of external symbols and data declarations. The ISP central processor software components that deal with terminal I/O and the user programs that interface with those components use this file. As such, it is a logical extension of /usr/include/sys/termio.h.

NAME /usr/include/sys/cstty.h

FORMAT This is a C programming language header file.

/usr/include/sys/csintern.h

This ISP file is a C programming language header file that consolidates translation tables used for mapping one internal character-set representation to another.

NAME /usr/include/sys/csintern.h

FORMAT This is a C programming language header file.

EXAMPLE

```
/* @(#)csintern.h        1.7 */
/* "@(#) Copyright (C) 1985 by Four-Phase (ISG) of Motorola, Inc." */

#ifndef csintern_h
#define csintern_h

/* Define the standard internal character-set translation tables that
   convert to the Motorola private character set 040, or convert towards a
   standard representation avoiding character sets 040, 360, and 361.
```

```
    These are compiled by csinit(3) from the internal character-set
    translation tables "topri" and "tostd".
 */

#include <sys/cstty.h>

#include "cs_topri.h"

#include "cs_tostd.h"

#endif csintern_h
```

/usr/include/sys/cs_topri.h

This ISP file is a C programming language header file containing asm instructions to build a translation table for mapping XSIS 058404 standard character sets into Motorola private character set 040. This file is included by csintern.h.

NAME /usr/include/sys/cs_topri.h

FORMAT This is a C programming language header file.

/usr/include/sys/cs_tostd.h

This ISP file is a C programming language header file containing asm instructions to build a translation table for mapping Motorola private character set 040 into XSIS 058404 standard character sets. This file is included by csintern.h.

NAME /usr/include/sys/cs_tostd.h

FORMAT This is a C programming language header file.

/usr/lib/libcs.a

This ISP file contains the international-support library routines, which can be linked with other object files through the cc(1) command.

NAME /usr/lib/libcs.a

FORMAT This is an ar(1) format library file.

Appendix A
Character Set 040

The Motorola private character set 040 is provided for the System 6300 operating system and application programs, and is defined by the Character Code Standard, XSIS 058404 (IDENTITY XC1-1-1-0). Character set 0 of that standard is compatible with the ASCII/ISO/CCITT character-code standards. Character set 040 can be selected through a UNIX-derived operating system command or a system call to be the default character set.

Table A-1 defines character set 040: the numeric code for each character in octal, decimal, and hexadecimal, the character or ASCII code mnemonic, the XSIS 058404 non-040 character code (if it exists), and a description of the character. For accented characters, the XSIS 058404 code is shown as the CharSet8 code and Char8Code for the accent, followed by the character-set 000 letter to which the accent applies.

Table A-1. Motorola Private Character Set 040

Character Code			Character or	XSIS 058404 Code		
Oct	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
000	0	00	(NUL)	000	000	Null
001	1	01	(SOH)	000	001	Start of Heading
002	2	02	(STX)	000	002	Start of Text
003	3	03	(ETX)	000	003	End of Text
004	4	04	(EOT)	000	004	End of Transmission
005	5	05	(ENQ)	000	005	Enquiry
006	6	06	(ACK)	000	006	Acknowledge
007	7	07	(BEL)	000	007	Bell
010	8	08	(BS)	000	010	Backspace
011	9	09	(HT)	000	011	Horizontal Tabulation
012	10	0a	(LF)	000	012	Line Feed
013	11	0b	(VT)	000	013	Vertical Tabulation
014	12	0c	(FF)	000	014	Form Feed
015	13	0d	(CR)	000	015	Carriage Return
016	14	0e	(SO)	000	016	Shift Out
017	15	0f	(SI)	000	017	Shift In
020	16	10	(DLE)	000	020	Data Link Escape
021	17	11	(DC1)	000	021	Device Control 1
022	18	12	(DC2)	000	022	Device Control 2
023	19	13	(DC3)	000	023	Device Control 3
024	20	14	(DC4)	000	024	Device Control 4
025	21	15	(NAK)	000	025	Negative Acknowledge
026	22	16	(SYN)	000	026	Synchronous Idle
027	23	17	(ETB)	000	027	End of Transmission Block
030	24	18	(CAN)	000	030	Cancel
031	25	19	(EM)	000	031	End of Medium
032	26	1a	(SUB)	000	032	Substitute
033	27	1b	(ESC)	000	033	Escape
034	28	1c	(FS)	000	034	File Separator

Appendix A
Character Set 040

Table A-1. Motorola Private Character Set 040 (Continued)

Character Code			Character or (Mnemonic)	XSIS 058404 Code		Description
Oct	Dec	Hex		CharSet8	Char8Code	
035	29	1d	(GS)	000	035	Group Separator
036	30	1e	(RS)	000	036	Record Separator
037	31	1f	(US)	000	037	Unit Separator
040	32	20	(SP)	000	040	Space
041	33	21	!	000	041	Exclamation Point
042	34	22	"	000	042	Double Quotation Mark
043	35	23	#	000	043	Number or Pound Sign
044	36	24	$	000	244	Dollar Sign
045	37	25	%	000	045	Percent Sign
046	38	26	&	000	046	Ampersand
047	39	27	'	000	047	Apostrophe or Single Quot.
050	40	28	(	000	050	Left or Open Parenthesis
051	41	29	)	000	051	Right or Close Parenthesis
052	42	2a	*	000	052	Asterisk
053	43	2b	+	000	053	Plus Sign
054	44	2c	,	000	054	Comma
055	45	2d	-	000	055	Hyphen or Minus Sign
056	46	2e	.	000	056	Period
057	47	2f	/	000	057	Slash
060	48	30	0	000	060	0
061	49	31	1	000	061	1
062	50	32	2	000	062	2
063	51	33	3	000	063	3
064	52	34	4	000	064	4
065	53	35	5	000	065	5
066	54	36	6	000	066	6
067	55	37	7	000	067	7
070	56	38	8	000	070	8
071	57	39	9	000	071	9
072	58	3a	:	000	072	Colon
073	59	3b	;	000	073	Semi-colon
074	60	3c	<	000	074	Open Angle Bracket
075	61	3d	=	000	075	Equal Sign
076	62	3e	>	000	076	Close Angle Bracket
077	63	3f	?	000	077	Question Mark
100	64	40	@	000	100	At Sign
101	65	41	A	000	101	A
102	66	42	B	000	102	B
103	67	43	C	000	103	C
104	68	44	D	000	104	D
105	69	45	E	000	105	E
106	70	46	F	000	106	F
107	71	47	G	000	107	G
110	72	48	H	000	110	H
111	73	49	I	000	111	I
112	74	4a	J	000	112	J
113	75	4b	K	000	113	K
114	76	4c	L	000	114	L

Table A-1. Motorola Private Character Set 040 (Continued)

Character Code			Character or	XSIS 058404 Code		
Oct	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
115	77	4d	M	000	115	M
116	78	4e	N	000	116	N
117	79	4f	O	000	117	O
120	80	50	P	000	120	P
121	81	51	Q	000	121	Q
122	82	52	R	000	122	R
123	83	53	S	000	123	S
124	84	54	T	000	124	T
125	85	55	U	000	125	U
126	86	56	V	000	126	V
127	87	57	W	000	127	W
130	88	58	X	000	130	X
131	89	59	Y	000	131	Y
132	90	5a	Z	000	132	Z
133	91	5b	[	000	133	Open Bracket
134	92	5c	\	000	134	Backslash
135	93	5d	]	000	135	Close Bracket
136	94	5e	^	000	136	Circumflex
137	95	5f	_	000	137	Underscore
140	96	60	`	000	140	Left Single Quotation Mark
141	97	61	a	000	141	a
142	98	62	b	000	142	b
143	99	63	c	000	143	c
144	100	64	d	000	144	d
145	101	65	e	000	145	e
146	102	66	f	000	146	f
147	103	67	g	000	147	g
150	104	68	h	000	150	h
151	105	69	i	000	151	i
152	106	6a	j	000	152	j
153	107	6b	k	000	153	k
154	108	6c	l	000	154	l
155	109	6d	m	000	155	m
156	110	6e	n	000	156	n
157	111	6f	o	000	157	o
160	112	70	p	000	160	p
161	113	71	q	000	161	q
162	114	72	r	000	162	r
163	115	73	s	000	163	s
164	116	74	t	000	164	t
165	117	75	u	000	165	u
166	118	76	v	000	166	v
167	119	77	w	000	167	w
170	120	78	x	000	170	x
171	121	79	y	000	171	y
172	122	7a	z	000	172	z
173	123	7b	{	000	173	Open Brace
174	124	7c	¦	357	153	Broken Vertical Bar

Table A-1. Motorola Private Character Set 040 (Continued)

Character Code			Character or	XSIS 058404 Code		
Oct	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
175	125	7d	}	000	175	Close Brace
176	126	7e	~	000	176	Tilde
177	127	7f	(DEL)	000	177	Delete
200	128	80		000	200	Reserved for Control Code
201	129	81		000	201	Reserved for Control Code
202	130	82		000	202	Reserved for Control Code
203	131	83		000	203	Reserved for Control Code
204	132	84		000	204	Reserved for Control Code
205	133	85		000	205	Reserved for Control Code
206	134	86		000	206	Reserved for Control Code
207	135	87		000	207	Reserved for Control Code
210	136	88		000	210	Reserved for Control Code
211	137	89		000	201	Reserved for Control Code
212	138	8a		000	202	Reserved for Control Code
213	139	8b		000	203	Reserved for Control Code
214	140	8c		000	204	Reserved for Control Code
215	141	8d		000	205	Reserved for Control Code
216	142	8e		000	206	Reserved for Control Code
217	143	8f		000	207	Reserved for Control Code
220	144	90		000	220	Reserved for Control Code
221	145	91		000	201	Reserved for Control Code
222	146	92		000	202	Reserved for Control Code
223	147	93		000	203	Reserved for Control Code
224	148	94		000	204	Reserved for Control Code
225	149	95		000	205	Reserved for Control Code
226	150	96		000	206	Reserved for Control Code
227	151	97		000	207	Reserved for Control Code
230	152	98		000	230	Reserved for Control Code
231	153	99		000	201	Reserved for Control Code
232	154	9a		000	202	Reserved for Control Code
233	155	9b		000	203	Reserved for Control Code
234	156	9c		000	204	Reserved for Control Code
235	157	9d		000	205	Reserved for Control Code
236	158	9e		000	206	Reserved for Control Code
237	159	9f		000	207	Reserved for Control Code
240	160	a0		000	240	Reserved for Control Code
241	161	a1	Ä	000	310 A	A dieresis
242	162	a2	Å	000	312 A	A ring
243	163	a3	Ã	000	304 A	A tilde
244	164	a4	AE	000	341	AE ligature
245	165	a5	Ç	000	313 C	C cedilla
246	166	a6	É	000	302 E	E acute
247	167	a7	Ñ	000	304 N	N tilde
250	168	a8	Ö	000	310 O	O dieresis
251	169	a9	Õ	000	304 O	O tilde
252	170	aa	OE	000	352	OE ligature
253	171	ab	ø	000	351	O slash

Table A-1. Motorola Private Character Set 040 (Continued)

Character Code			Character or (Mnemonic)	XSIS 058404 Code		Description
Oct	Dec	Hex		CharSet8	Char8Code	
254	172	ac	Ü	000	310 U	U dieresis
255	173	ad	à	000	301 a	a grave
256	174	ae	á	000	302 a	a acute
257	175	af	â	000	303 a	a circumflex
260	176	b0	ã	000	304 a	a tilde
261	177	b1	ä	000	310 a	a dieresis
262	178	b2	å	000	312 a	a ring
263	179	b3	æ	000	361	ae ligature
264	180	b4	ç	000	313 c	c cedilla
265	181	b5	è	000	301 e	e grave
266	182	b6	é	000	302 e	e acute
267	183	b7	ê	000	303 e	e circumflex
270	184	b8	ë	000	310 e	e dieresis
271	185	b9	ì	000	301 i	i grave
272	186	ba	í	000	302 i	i acute
273	187	bb	î	000	303 i	i circumflex
274	188	bc	ï	000	310 i	i dieresis
275	189	bd	ñ	000	304 n	n tilde
276	190	be	ò	000	301 o	o grave
277	191	bf	ó	000	302 o	o acute
300	192	c0	ô	000	303 o	o circumflex
301	193	c1	õ	000	304 o	o tilde
302	194	c2	ö	000	310 o	o dieresis
303	195	c3	œ	000	372	oe ligature
304	196	c4	ø	000	371	o slash
305	197	c5	ß	000	373	ess-zed (German)
306	198	c6	ù	000	301 u	u grave
307	199	c7	ú	000	302 u	u acute
310	200	c8	û	000	303 u	u circumflex
311	201	c9	ü	000	310 u	u dieresis
312	202	ca	ů	000	312 u	u ring
313	203	cb	°	000	312	Ring
314	204	cc	¨	000	310	Dieresis
315	205	cd	£	000	243	Pound (Sterling) Sign
316	206	ce	§	000	247	Section Mark
317	207	cf	¤	000	044	Currency Symbol
320	208	d0	¢	000	242	Cent Sign
321	209	d1	¡	000	241	Inverted Exclamation Point
322	210	d2	¿	000	277	Inverted Question Mark
323	211	d3	┌			Box: upper left corner
324	212	d4	╲			Box: diag. upper left corner
325	213	d5	┐			Box: upper right corner
326	214	d6	╱			Box: diag. upper right corner
327	215	d7	└			Box: lower left corner
330	216	d8	╱			Box: diag. lower left corner
331	217	d9	┘			Box: lower right corner
332	218	da	╲			Box: diag. lower right corner
333	219	db	├			Box: right-pointing tee

Table A-1. Motorola Private Character Set 040 (Continued)

Character Code			Character or (Mnemonic)	XSIS 058404 Code		Description
Oct	Dec	Hex		CharSet8	Char8Code	
334	220	dc	┤			Box: left-pointing tee
335	221	dd	┴			Box: up-pointing tee
336	222	de	┬			Box: down-pointing tee
337	223	df	│	357	344	Box: vertical line
340	224	e0	─	357	345	Box: horizontal line
341	225	e1	┼	357	346	Box: crossing
342	226	e2	■	042	043	Small shaded block
343	227	e3	□	042	042	Open block
344	228	e4	▨			Right sloping hatches
345	229	e5	▨			Left sloping hatches
346	230	e6	¶	000	266	Paragraph Mark (Pilcrow)
347	231	e7	†	357	060	Dagger
350	232	e8	™	000	324	TM (trademark)
351	233	e9	©	000	323	Circle C (copyright)
352	234	ea	®	000	322	Circle R (registered)
353	235	eb	º	000	353	o ordinal
354	236	ec	ª	000	343	a ordinal
355	237	ed	ƒ	357	242	Script f (florin)
356	238	ee	₱	357	244	Peseta Sign
357	239	ef	¥	000	245	Yen Sign
360	240	f0	¬	357	152	Logical Not
361	241	f1	÷	000	270	Division Sign
362	242	f2	×	000	264	Multiplication Sign
363	243	f3	✓	357	317	Check Mark
364	244	f4	µ	000	265	Micro Symbol
365	245	f5	ŷ	000	303 y	y circumflex
366	246	f6	ý	000	302 y	y acute
367	247	f7	ÿ	000	310 y	y dieresis
370	248	f8	○	357	146	Middle Circle
371	249	f9	●	357	147	Middle Bullet
372	250	fa	¯	000	305	Macron
373	251	fb	|	000	174	Solid Vertical Bar
374	252	fc	¸	000	313	Cedilla
375	253	fd	´	000	302	Acute Accent
376	254	fe	π	046	163	Pi
377	355	ff	(CSelect)	(Cselect)		Character-Set Select Code

Appendix B
TM31 Terminal Key Codes and Keyboards

TM31 TERMINAL CHARACTER SET

The input-sequence codes a TM31 Terminal sends to the terminal driver are shown in Table B-1. This table is the specification of the download-image keyboard character set for the TM31 Terminal.

For each key or key sequence that transmits a character code for a printable character as a single byte, the value transmitted is shown as the graphic character plus its octal and hexadecimal values. For each nonprintable key or key sequence that yields a control code defined by the ANSI X3.64/3.4 standard, the ANSI code name is shown plus either the octal and hexadecimal values or the escape-code sequence. For each non-ANSI key or key sequence, the escape-code sequence is shown. Where no value appears in a column, no value is transmitted for the key sequence.

The codes produced by the following key sequences are identical:

- CTRL F1 through CTRL F5 and CTRL F7 through CTRL F9 are identical to those produced by CTRL Island 1 through CTRL Island 5 and CTRL Island 7 through CTRL Island 9, respectively.
- Codes produced by SHIFT F8 and SHIFT F9 are the same as those produced by SHIFT Island 8 and SHIFT Island 9, respectively.
- Codes produced by SHIFT SLCT, SHIFT CMD, and CTRL INS are the same as those produced by SHIFT F1, SHIFT F2, and SHIFT F3, respectively.
- Codes produced by the "typewriter" keypad numbers 1 through 9 (plain) are the same as those produced by (plain) Island 1 through Island 9, respectively.

Note that although the input sequence produced by the ALT key is ESC O @, the code that the terminal driver delivers to a user process is just one byte with the value \033 (ESC).

Table B-1. TM31 Terminal Key Codes

Key	Plain			SHIFT			CTRL			SHIFT+CTRL		
		Oct	Hex		Oct	Hex		Oct	Hex		Oct	Hex
A	a	141	61	A	101	41	SOH	001	01	SOH	001	01
B	b	142	62	B	102	42	STX	002	02	STX	002	02
C	c	143	63	C	103	43	ETX	003	03	ETX	003	03
D	d	144	64	D	104	44	EOT	004	04	EOT	004	04
E	e	145	65	E	105	45	ENQ	005	05	ENQ	005	05
F	f	146	66	F	106	46	ACK	006	06	ACK	006	06
G	g	147	67	G	107	47	BEL	007	07	BEL	007	07
H	h	150	68	H	110	48	BS	010	08	BS	010	08
I	i	151	69	I	111	49	HT	011	09	HT	011	09

Table B-1. TM31 Terminal Key Codes (Continued)

Key	Plain Oct	Plain Hex	SHIFT Oct	SHIFT Hex	CTRL Oct	CTRL Hex	SHIFT+CTRL Oct	SHIFT+CTRL Hex
J	j	152 6a	J	112 4a	LF	012 0a	LF	012 0a
K	k	153 6b	K	113 4b	VT	013 0b	VT	013 0b
L	l	154 6c	L	114 4c	FF	014 0c	FF	014 0c
M	m	155 6d	M	115 4d	CR	015 0d	CR	015 0d
N	n	156 6e	N	116 4e	SO	016 0e	SO	016 0e
O	o	157 6f	O	117 4f	SI	017 0f	SI	017 0f
P	p	160 70	P	120 50	DLE	020 10	DLE	020 10
Q	q	161 71	Q	121 51	DC1	021 11	DC1	021 11
R	r	162 72	R	122 52	DC2	022 12	DC2	022 12
S	s	163 73	S	123 53	DC3	023 13	DC3	023 13
T	t	164 74	T	124 54	DC4	024 14	DC4	024 14
U	u	165 75	U	125 55	NAK	025 15	NAK	025 15
V	v	166 76	V	126 56	SYN	026 16	SYN	026 16
W	w	167 77	W	127 57	ETB	027 17	ETB	027 17
X	x	170 78	X	130 58	CAN	030 18	CAN	030 18
Y	y	171 79	Y	131 59	EM	031 19	EM	031 19
Z	z	172 7a	Z	132 5a	SUB	032 1a	SUB	032 1a
^ ~	^	136 5e	~	176 7e	RS	036 1e		
1 !	1	061 31	!	041 21				
2 @	2	062 32	@	100 40				
3 #	3	063 33	#	043 23				
4 $	4	064 34	$	044 24				
5 %	5	065 35	%	045 25				
6 ¦	6	066 36	¦	174 7c				
7 &	7	067 37	&	046 26				
8 *	8	070 38	*	052 2a				
9 (	9	071 39	(	050 28				
0)	0	060 30	)	051 29				
- _	-	055 2d	_	137 5f	DEL	177 7f	US	037 1f
= +	=	075 3d	+	053 2b				
BACKSPACE	BS	010 08	BS	010 08	BS	010 08		
TAB	HT	011 09	ESC O G		ESC O E		ESC O E	
CHAR CODE	"dead key"		"dead key"		"dead key"		"dead key"	
[< {	[	133 5b	<	074 3c	{	173 7b		
] > }	]	135 5d	>	076 3e	}	175 7d	GS	035 1d
; :	;	073 3b	:	072 3a				
' " `	'	047 27	"	042 22	`	140 60		
RETURN	CR	015 0d	CR	015 0d	ESC O J		ESC O J	
, ,	,	054 2c	,	054 2c	FS	034 1c		
. .	.	056 2e	.	056 2e	ESC	033 1b		
/ ? \	/	057 2f	?	077 3f	\	134 5c		
SPACEBAR	SP	040 20	SP	040 20	SP	040 20		
0	0	060 30	ESC O F		ESC O H			
1	1	061 31	ESC O K		ESC O `			
2	2	062 32	ESC O ¦		ESC O a			
3	3	063 33	ESC O M		ESC O b			
4	4	064 34	ESC O i		ESC O c			
5	5	065 35	ESC O j		ESC O d			

Table B-1. TM31 Terminal Key Codes (Continued)

Key	Plain Oct	Plain Hex	SHIFT Oct	SHIFT Hex	CTRL Oct	CTRL Hex	SHIFT+CTRL Oct	SHIFT+CTRL Hex
6	6	066 36	ESC O o		ESC O z			
7	7	067 37	ESC O k		ESC O f			
8	8	070 38	ESC O w		ESC O g			
9	9	071 39	ESC O x		ESC O h			
ALT	ESC O @		ESC O N		ESC O O			
-	-	055 2d	ESC [2 Q		ESC [2 R			
.	.	056 2e	ESC [2 O		ESC [2 y			
ACCEPT	LF	012 0a	LF	012 0a	ESC O _			
CHAR ATTR	ESC O A		ESC [2 U		ESC [2 u			
UNDO	ESC O B		ESC [2 z		ESC [2 s			
WINDO	ESC O C		ESC [2 V		ESC [2 v			
PRINT	ESC O D		ESC [2 {		ESC [2 t			
SLCT	ESC [2 I		ESC O p		ESC O I			
MOVE	ESC O Z		ESC O e		ESC O t			
CMD	ESC O ^		ESC O q		ESC O 1			
DELETE	ESC [2 1		ESC [2 n		ESC [2 o			
EXIT	CAN 030	18	ESC [2 p		EOT 004	04		
ERASE	ESC [2 w		ESC [2 x		ESC [2 N			
INS	ESC [2 r		ESC 2 L		ESC O r			
COPY	ESC O [		ESC O {		ESC O v			
F1	ESC O P		ESC O p		ESC O `			
F2	ESC O Q		ESC O q		ESC O a			
F3	ESC O R		ESC O r		ESC O b			
F4	ESC O S		ESC O s		ESC O c			
F5	ESC O T		ESC O t		ESC O d			
F6	ESC O U		ESC O u		ESC O e			
F7	ESC O V		ESC O v		ESC O f			
F8	ESC O W		ESC O w		ESC O g			
F9	ESC O X		ESC O x		ESC O h			
RESET	ESC O Y		ESC O y		DEL 177	7f		
Up Arrow (↑)	CUU ESC [A		ESC [2 g		ESC [2 ¦			
Down Arrow (↓)	CUD ESC [B		ESC [2 h		ESC [2 b			
Right Arrow (→)	CUF ESC [C		ESC [2 i		ESC [2 c			
Left Arrow (←)	CUB ESC [D		ESC [2 j		ESC [2 }			
HOME	ESC O]		ESC O }		ESC O m			
HELP	ESC O \		ESC O ~		ESC O n			
ACTION	Enter manual mode							
CTRL	CTRL mode selector		no transmission to host					
SHIFT	SHIFT mode selector							
LOCK	SHIFT lock toggle							

Appendix B
TM31 Terminal Key Codes and Keyboards

TM31 TERMINAL KEYBOARDS

Figures B-1 through B-8 show the different national keyboards for the TM31 Terminal. The "dead" keys are shown shaded. The general "dead" key occurs in different aspects (plain or CTRL) on different keys on the various keyboards. For example, to access the general "dead" key on the U.K. keyboard, press CTRL @; on the Swedish keyboard, press the plain circumflex/dieresis "dead" key. Note that not all three aspects of a key are necessarily "dead." For example, on the Canadian keyboard, only the CTRL function of the quotation mark (') key is "dead."

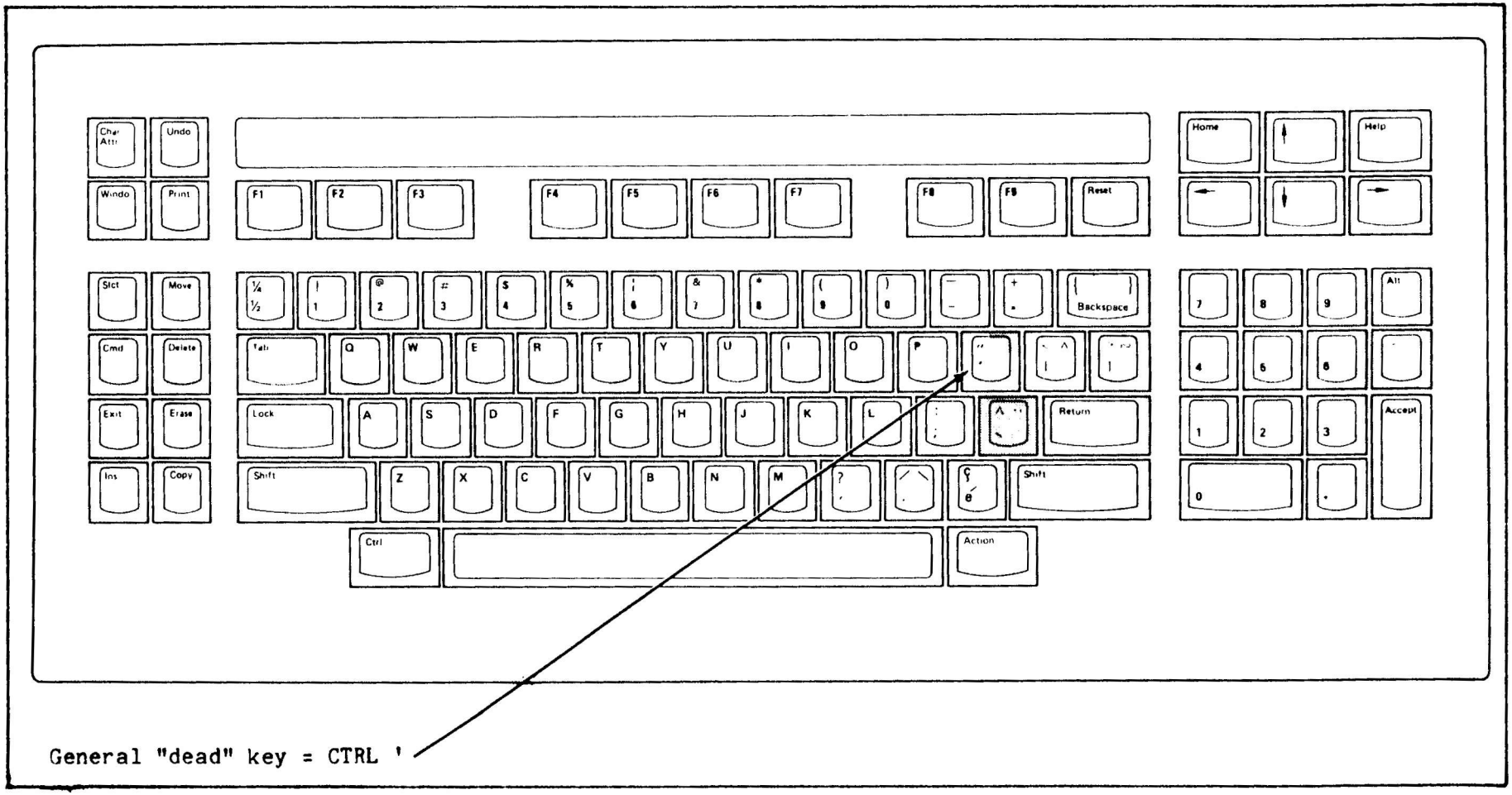


Figure B-1. TM31 Terminal Canadian Keyboard

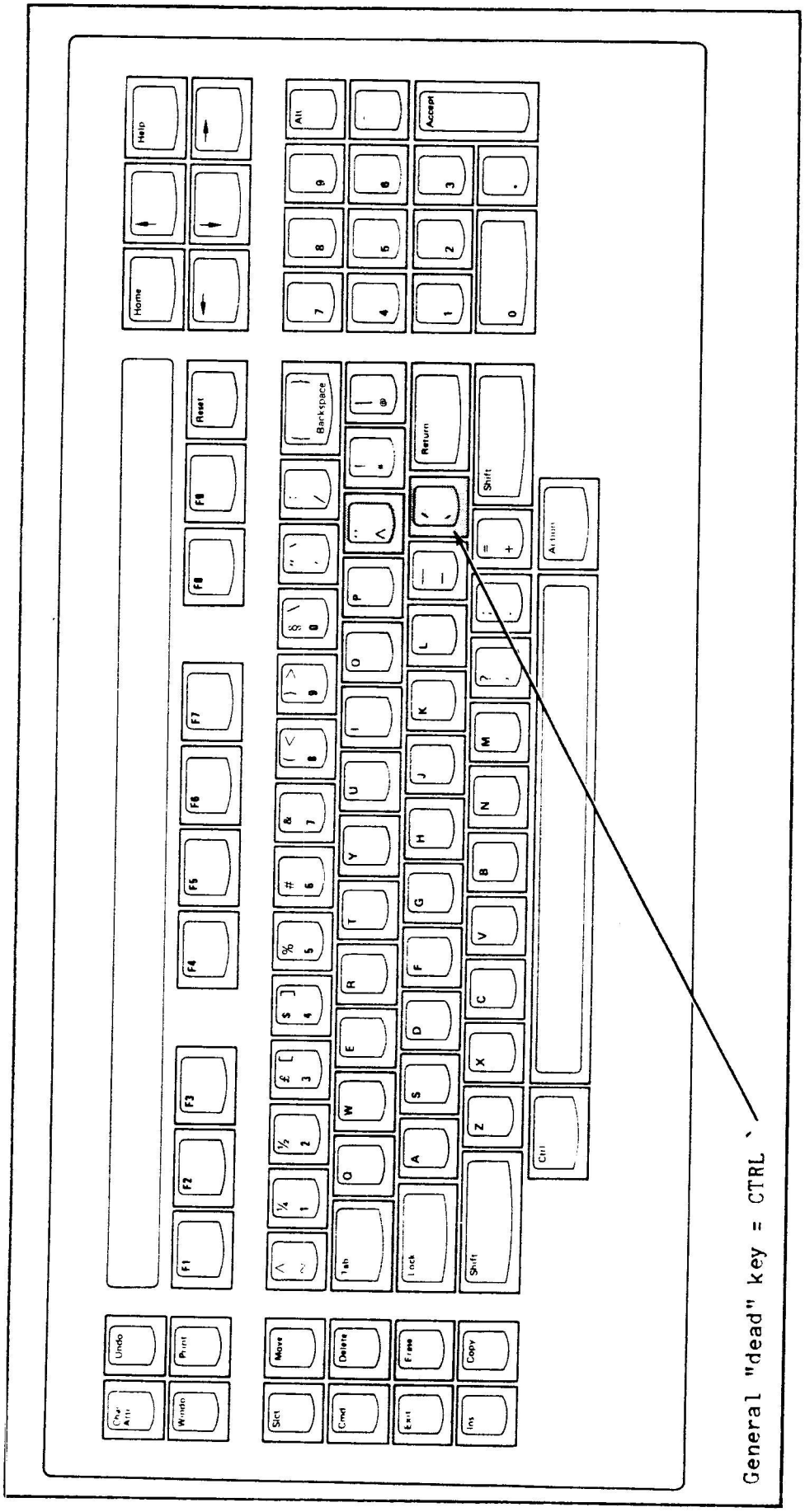


Figure B-2. TM31 Terminal Dutch Keyboard

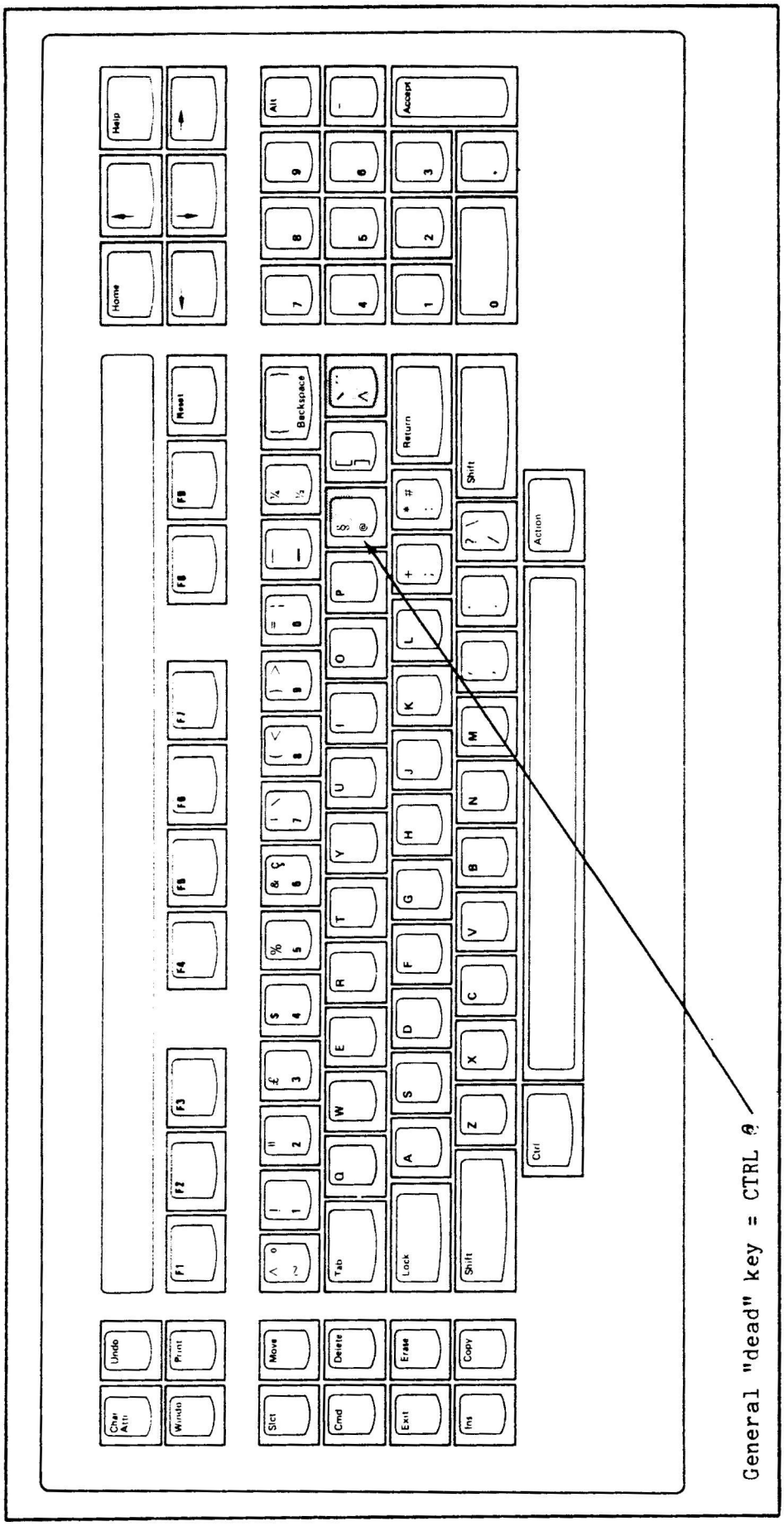


Figure B-3. TM31 Terminal English (U.K.) Keyboard

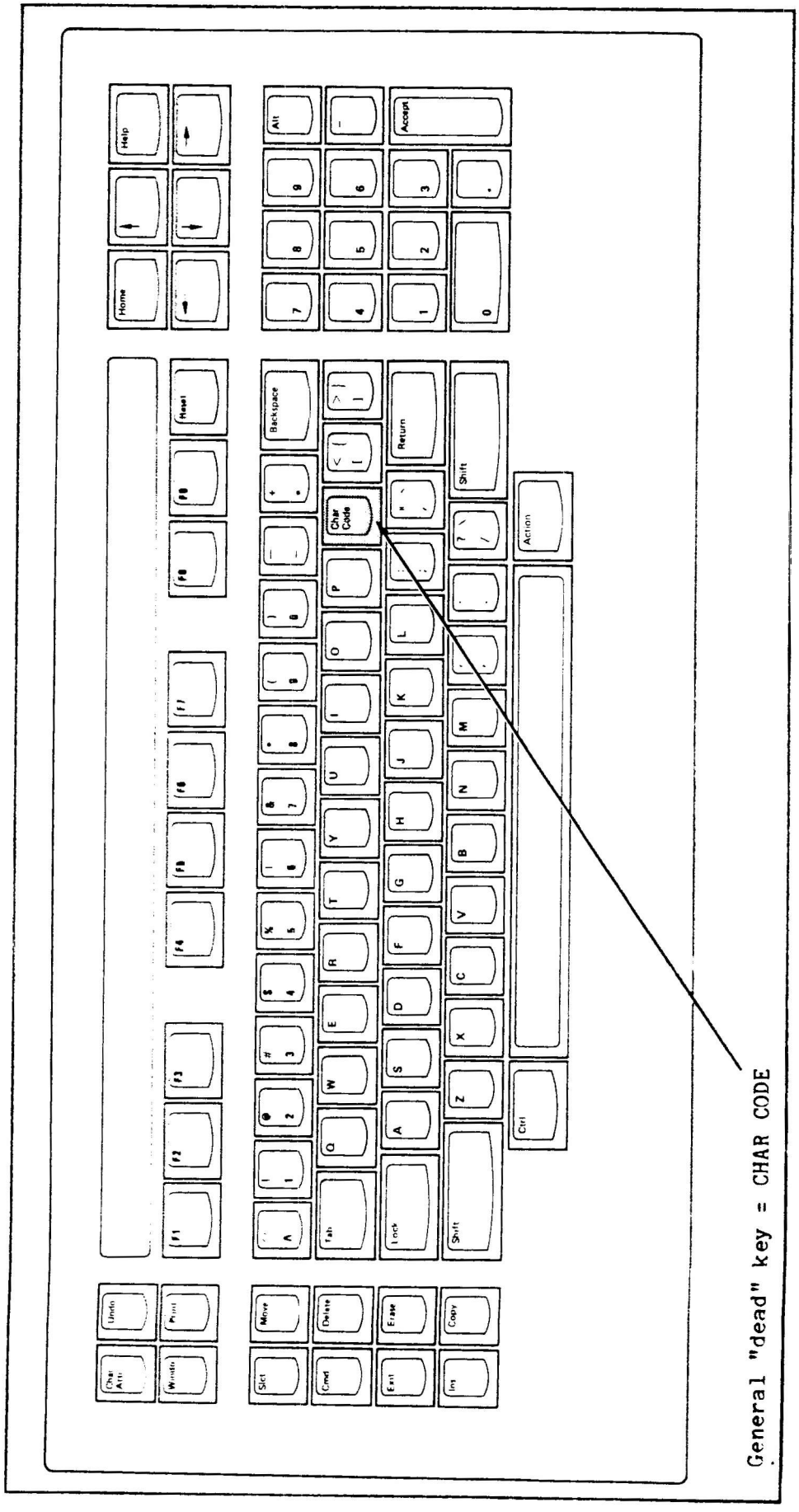


Figure B-4. TM31 Terminal English (U.S.A.) Keyboard

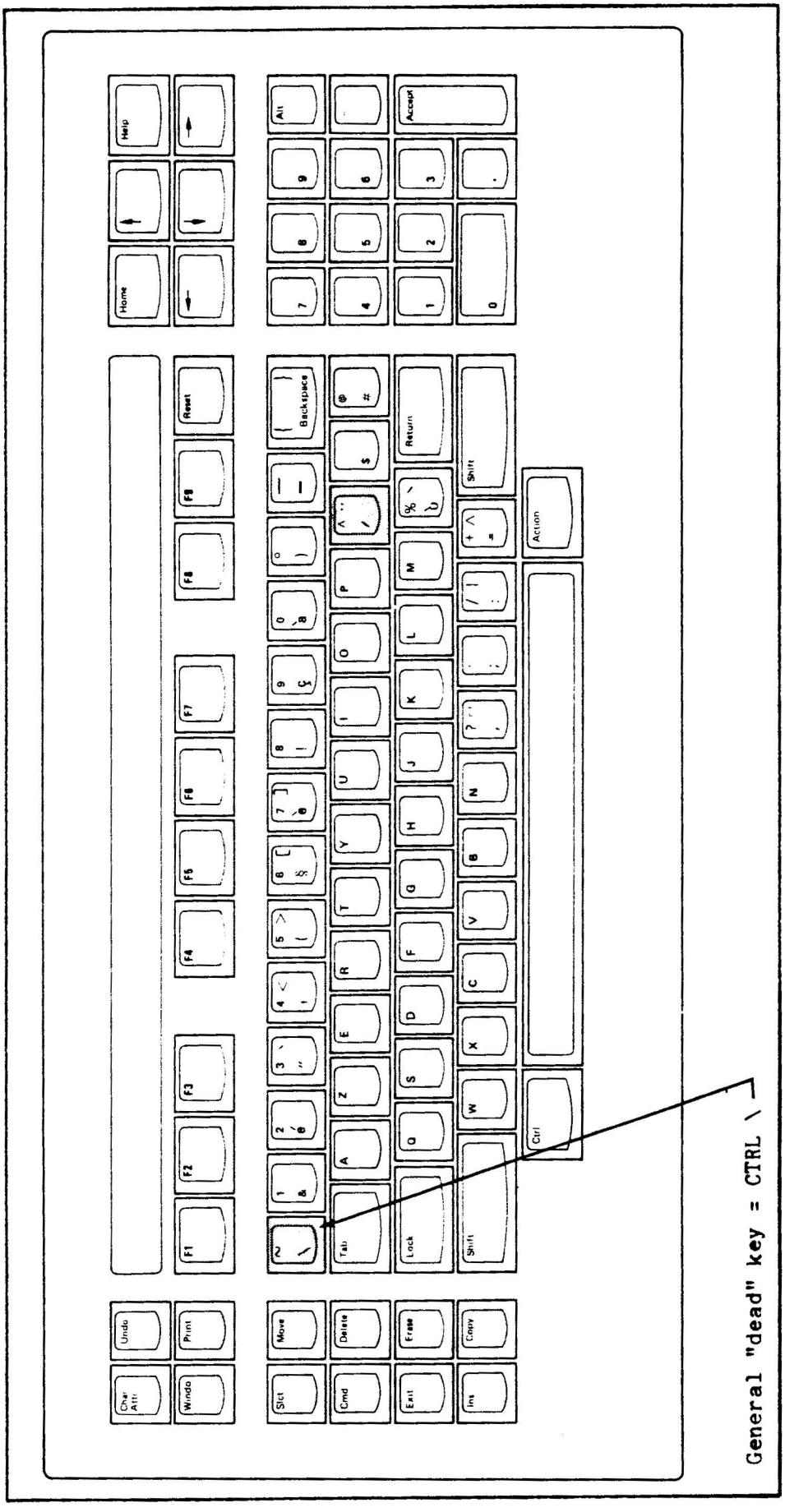


Figure B-5. TM31 Terminal French Keyboard

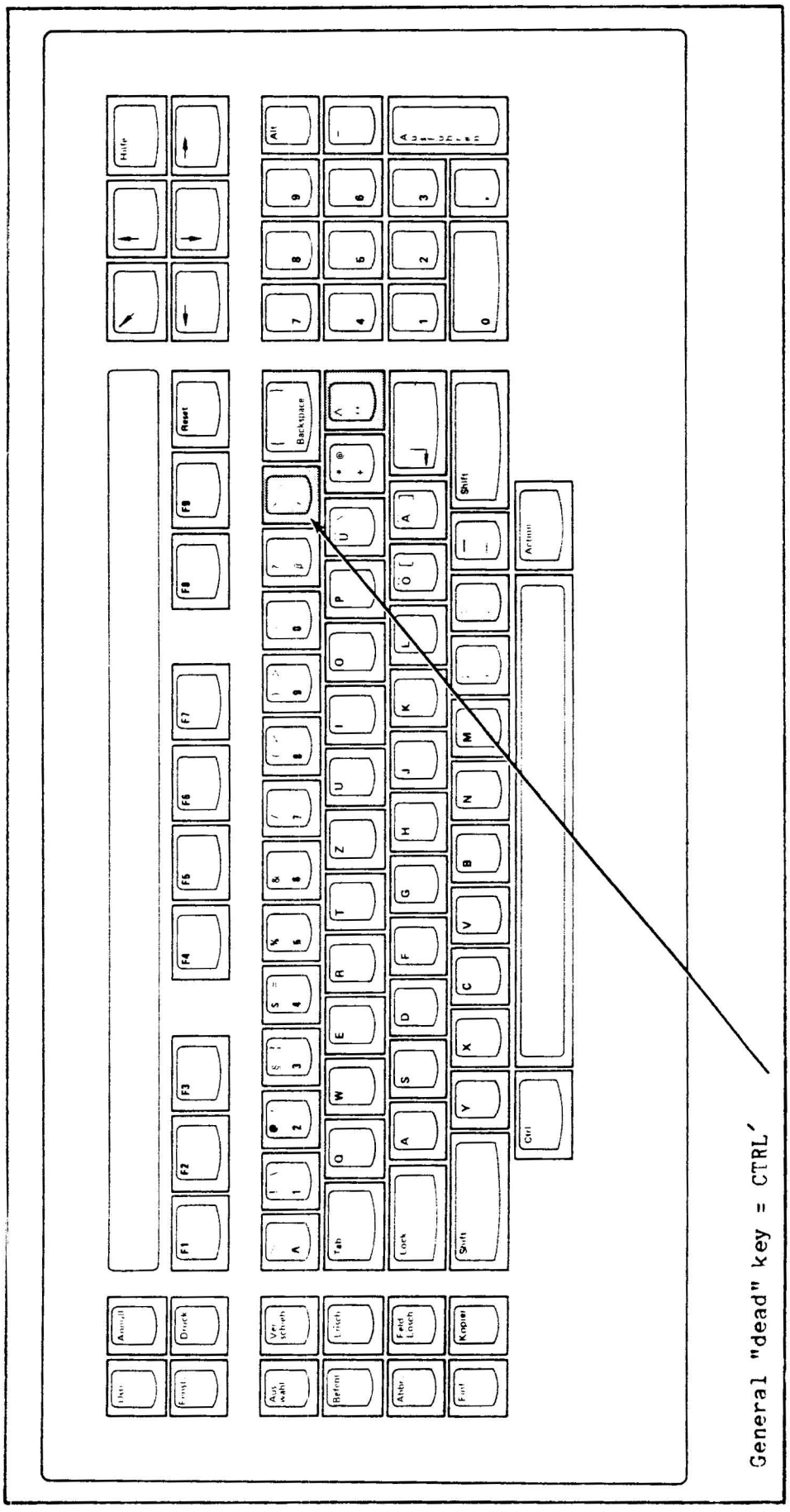


Figure B-6. TM31 Terminal German Keyboard

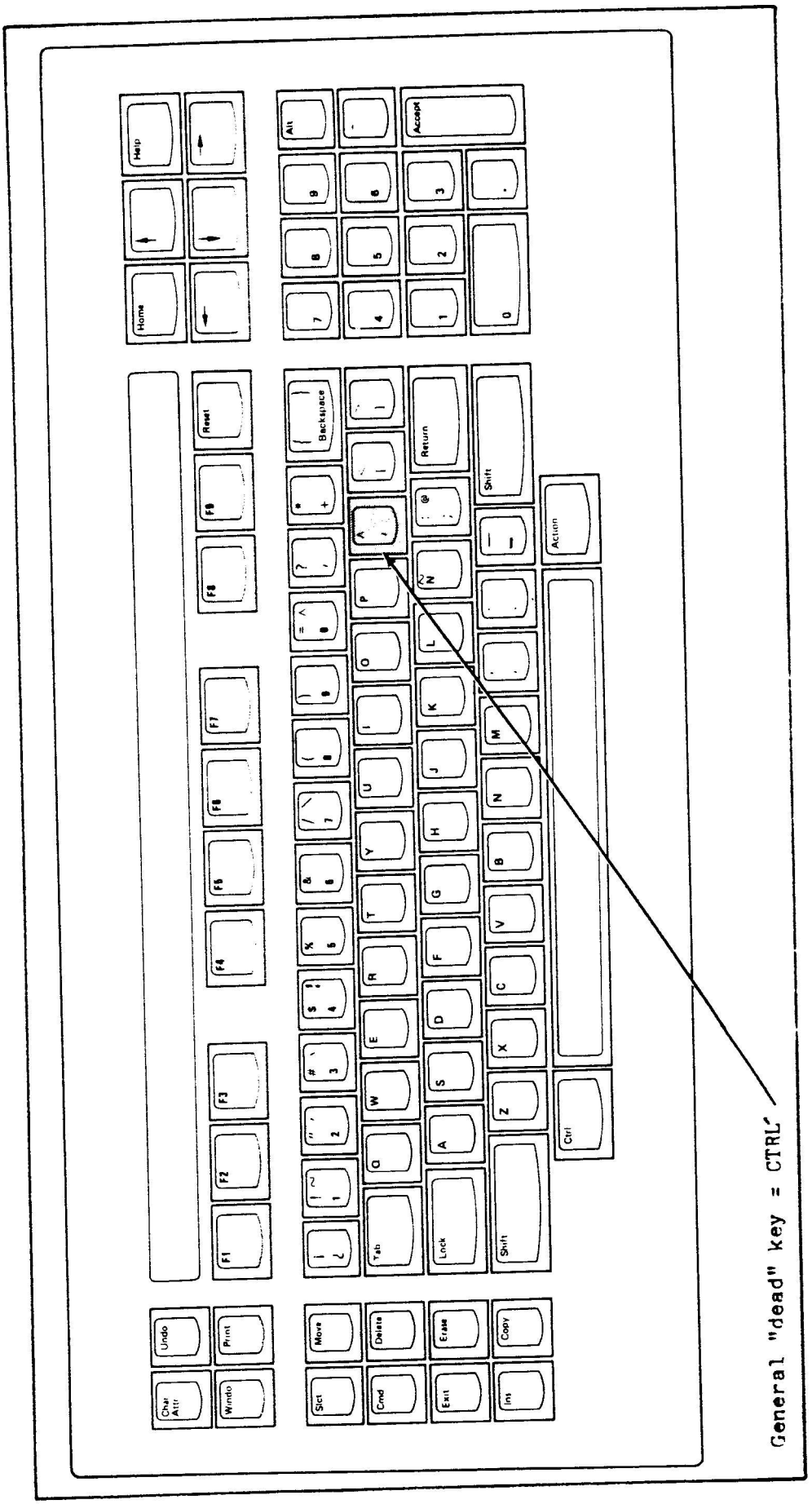


Figure B-7. TM31 Terminal Spanish Keyboard

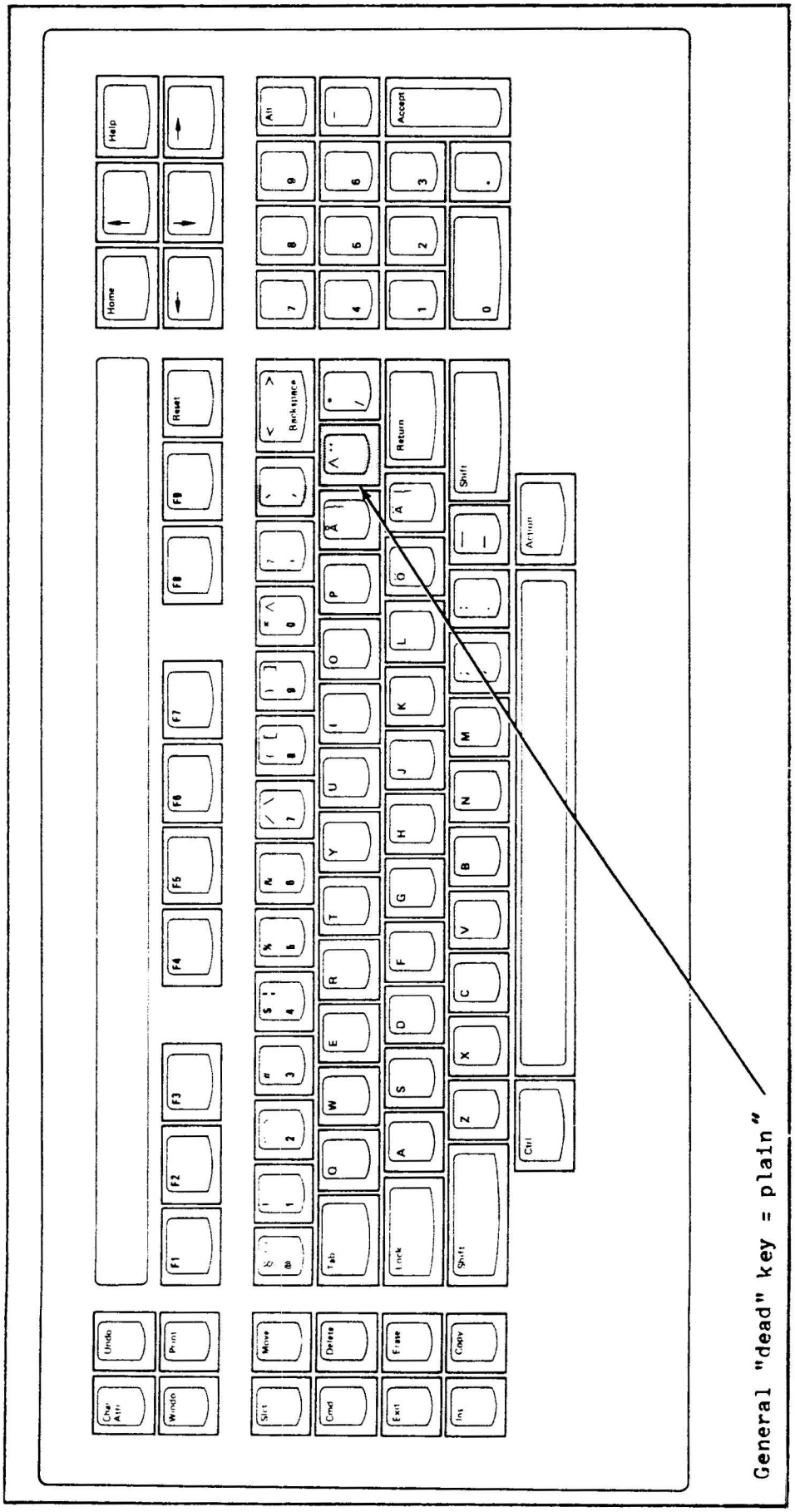


Figure B-8. TM31 Terminal Swedish Keyboard

"DEAD" KEY SEQUENCES

The key definitions in Tables B-2 through B-9 are the "dead" key sequences that yield ESC N x input codes for the TM31 Terminal download images. These codes access the TM31 Terminal G2 character set. The input sequence generated by the "dead" key sequence is translated by the terminal driver into a value in the internal character set, thereby yielding a code for a displayable character.

The Key columns indicate the keys that produce the "dead" key sequence. In the "Dead Key" column, "general" indicates the general "dead" key, as shown in Figures B-1 through B-8. The "Secondary Key(s)" column indicates the key(s) that is the second part of the key sequence. For example, to produce a pound sterling symbol with a German terminal download-image, press CTRL ' , then # (CTRL 4). The Character Set/Character Code columns show the octal codes representing the internal character-set value. Both the Motorola private character-set 040 code, when it exists for the character, and non-040 character-set codes are shown for the character. The displayable character is described in the rightmost column.

Table B-2. TM31 Terminal Canadian "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
General	A	040 000	244 341	AE ligature
`	a	040 000	255 301 a	a grave
^	a	040 000	257 303 a	a circumflex
..	a	040 000	261 310 a	a dieresis
General	a	040 000	263 361	ae ligature
`	e	040 000	265 301 e	e grave
^	e	040 000	267 303 e	e circumflex
^	o	040 000	300 303 o	o circumflex
..	o	040 000	302 310 o	o dieresis

Table B-2. TM31 Terminal Canadian "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
`	u	040 000	306 301 u	u grave
^	u	040 000	310 303 u	u circumflex
General	#	040 000	315 243	Pound (sterling) sign
General	s	040 000	316 247	Section mark
General	{	040 000	346 266	Paragraph mark
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	¦	040 000	373 174	Vertical bar
General	4	000	274	One-quarter
General	2	000	275	One-half
General	=	041	142	Not equal
General	←	000	254	Left arrow
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Table B-3. TM31 Terminal Dutch "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
`	a	040 000	255 301 a	a grave
^	a	040 000	257 303 a	a circumflex
`	e	040 000	265 301 e	e grave
'	e	040 000	266 302 e	e acute
^	e	040 000	267 303 e	e circumflex
..	e	040 000	270 310 e	e dieresis
`	o	040 000	276 301 o	o grave
..	o	040 000	302 310 o	o dieresis
`	u	040 000	306 301 u	u grave
..	u	040 000	311 310 u	u dieresis
General	{	040 000	346 266	Paragraph mark
General	- (Hyphen)	040 357	347 060	Dagger
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	/	040 000	361 270	Division sign
General	.	040 357	371 147	Middle bullet

Table B-3. TM31 Terminal Dutch "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
General	¦	040 000	373 174	Vertical bar
General	≠	041	142	Not equal
General	±	000	261	Plus or minus
General	←	000	254	Left arrow
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Table B-4. TM31 Terminal English (U.K.) "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
General	A	040 000	244 341	AE ligature
`	a	040 000	255 301 a	a grave
^	a	040 000	257 303 a	a circumflex
..	a	040 000	261 310 a	a dieresis
General	a	040 000	263 361	ae ligature
`	e	040 000	265 301 e	e grave
General	e	040 000	266 302 e	e acute
^	e	040 000	267 303 e	e circumflex
^	o	040 000	300 303 o	o circumflex
..	o	040 000	302 310 o	o dieresis
`	u	040 000	306 301 u	u grave
General	{	040 000	346 266	Paragraph mark
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	/	040 000	361 270	Division sign
General	¦	040 000	373 174	Vertical bar

Table B-4. TM31 Terminal English (U.K.) "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character		Character
		Set	Code	
General	=	041	142	Not equal
General	+	000	261	Plus or minus
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Table B-5. TM31 Terminal English (U.S.A.) "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
CHAR CODE	s	040 000	316 247	Section mark
CHAR CODE	$	040 000	320 242	Cent sign
CHAR CODE	{	040 000	346 266	Paragraph mark
CHAR CODE	- (Hyphen)	040 357	347 060	Dagger
CHAR CODE	t	040 000	350 324	Trademark
CHAR CODE	c	040 000	351 323	Copyright
CHAR CODE	r	040 000	352 322	Registered
CHAR CODE	_ (Underscore)	040 357	360 152	Logical not
CHAR CODE	/	040 000	361 270	Division sign
CHAR CODE	x	040 000	362 264	Multiplication sign
CHAR CODE	!	040 000	373 174	Vertical bar
CHAR CODE	4	000	274	One-quarter
CHAR CODE	3	357	375	One-third
CHAR CODE	2	000	275	One-half
CHAR CODE	5	357	376	Two-thirds
CHAR CODE	7	000	276	Three-quarters
CHAR CODE	=	041	142	Not equal
CHAR CODE	[	041	145	Less than or equal
CHAR CODE	]	041	146	Greater than or equal

Table B-5. TM31 Terminal English (U.S.A.) "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character		Character
		Set	Code	
CHAR CODE	+	000	261	Plus or minus
CHAR CODE	←	000	254	Left arrow
CHAR CODE	↑	000	255	Up arrow
CHAR CODE	↓	000	257	Down arrow

Table B-6. TM31 Terminal French "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
General	A	040 000	244 341	AE ligature
^	a	040 000	257 303 a	a circumflex
..	a	040 000	261 310 a	a dieresis
General	a	040 000	263 361	ae ligature
^	e	040 000	267 303 e	e circumflex
^	o	040 000	300 303 o	o circumflex
..	o	040 000	302 310 o	o dieresis
^	u	040 000	310 303 u	u circumflex
General	#	040 000	315 243	Pound (sterling) sign
General	{	040 000	346 266	Paragraph mark
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	/	040 000	361 270	Division sign
General	¦	040 000	373 174	Vertical bar
General	4	000	274	One-quarter
General	2	000	275	One-half
General	=	041	142	Not equal

Table B-6. TM31 Terminal French "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character		Character
		Set	Code	
General	←	000	254	Left arrow
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Table B-7. TM31 Terminal German "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
`	a	040 000	255 301 a	a grave
^	a	040 000	257 303 a	a circumflex
`	e	040 000	265 301 e	e grave
´	e	040 000	266 302 e	e acute
^	e	040 000	267 303 e	e circumflex
^	o	040 000	300 303 o	o circumflex
`	u	040 000	306 301 u	u grave
General	#	040 000	315 243	Pound (sterling) sign
General	{	040 000	346 266	Paragraph mark
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	/	040 000	361 270	Division sign
General	¦	040 000	373 174	Vertical bar
General	4	000	274	One-quarter
General	2	000	275	One-half
General	←	000	254	Left arrow
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Table B-8. TM31 Terminal Spanish "Dead" Key Sequences

Dead Key	Secondary Key(s)	Character Set	Character Code	Character
General	U	040 000	254 310 U	U dieresis
´	a	040 000	256 302 a	a acute
´	e	040 000	266 302 e	e acute
´	i	040 000	272 302 i	i acute
´	o	040 000	277 302 o	o acute
´	u	040 000	307 302 u	u acute
General	s	040 000	316 247	Section mark
General	{	040 000	346 266	Paragraph mark
General	- (Hyphen)	040 357	347 060	Dagger
General	c	040 000	351 323	Copyright
General	r	040 000	352 322	Registered
General	o	040 000	353 353	o ordinal
General	a	040 000	354 343	a ordinal
General	/	040 000	361 270	Division sign
General	.	040 357	371 147	Middle bullet
General	¦	040 000	373 174	Vertical bar

Table B-8. TM31 Terminal Spanish "Dead" Key Sequences (Continued)

Dead Key	Secondary Key(s)	Character		Character
		Set	Code	
General	=	041	142	Not equal
General	+	000	261	Plus or minus
General	←	000	254	Left arrow
General	↑	000	255	Up arrow
General	↓	000	257	Down arrow

Index

- What is your occupation?

☐ Programmer	☐ Operator	☐ Manager
☐ Systems Analyst	☐ Instructor	☐ Customer Engineer
☐ Engineer	☐ Student	☐ Other _____

- How do you use this manual?

☐ Reference Manual	☐ Introduction to the Subject
☐ In a Class	☐ Introduction to the System
☐ Self Study	☐ Other _____

fold fold

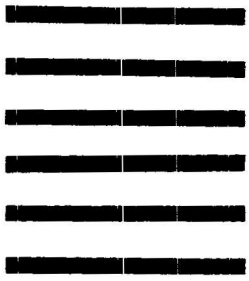
FIRST CLASS
Permit No. 194
Cupertino,
California

BUSINESS REPLY MAIL
No Postage Necessary if Mailed in the United States

Postage will be Paid by . . .

MOTOROLA INC.
10700 N. De Anza Blvd.
Cupertino, CA 95014

Attention: Technical Services, MS 32-2G2



fold fold

Cut Along Here

Staple Here

B1847A

Series 6000 International
Support Package Reference Manual

MOTOROLA
Information Systems

10700 North De Anza Blvd., Cupertino, California 95014 • (408) 255-0900