DOCUMENT ROUTING FORM

Submitted To: Distribution Date Submitted: March 5, 1985

Document Title: System 6300 International Support Package Addendum

Please review the attached document for technical accuracy and completeness. Mark corrections as necessary, and return this document to me by March 12, 1985.

NOTE

Comments received after the due date may not be included in this issue of this document.

Distribution:

Steve Hammett	22-10F2	Jane Allen	32-5D5
Gail Porter	52-6A9	Ginger Ferguson	52-1E6
Bob Ney, Jr.	42-8G3	Sam Kiteley	32-2D1
Jim Breeze	Dept. 4170, Irving, TX	Allan Moluf	32-2D1
Terry Gunter	Dept. 7386, Dallas, TX	Conor Sexton	
Bernie Knost	Dept. 7380, Hillcrest, TX	Dave Tanner	
Barbara Schildt	32-5D5	Shay Barsabe	32-2G2
Donna Dulick	32-5D5	Eric Norris	32-2G2

Notes to Reviewers:

This review is for the first issue of the manual and applies to release FE07 of the software. The brc, keyprompt, and wm commands in Section 4, taken from the Series 6000 Operating System Reference Manual, have ISP additions, but information relating to System 6600's has been removed. Some of the terminology comes from Xerox Corp.'s Character Code Standard XSIS 058404, IDENTITY XC1-1-1-0, publication. It is presumed that the Xerox document will become available as a regular Four-Phase manual.

Reviewer Signature ______________________________ Date ______________

[ ] Approved [ ] Approved with changes [ ] Unread

Notes to Writer:

Please return to Gloria Grigsby Mail Stop 32-2G2 Due Date March 12, 1985

If you cannot review this document by the due date, or if you have any questions, please call me at x2408. Thank you.

Preface

This manual describes the International Support Package (ISP) for allowing national characters and other symbols on the System 6300. The process of translating characters for terminal I/O and printer output and commands and routines available for programmers are explained in detail.

Readers should be familiar with UNIX or UNIX-derived operating systems. See the Series 6000 Operating System Programmer's Guide and the Series 6000 Operating System Reference Manual for more information.

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.

Motorola Inc.
10700 North DeAnza Boulevard
Cupertino, California 95014

* * * * * * * * * * * * * * * *

System 6300 International
Language Support Addendum

Software Release: FE07
87601770

* * * * * * * * * * * * * * * *

****

PRELIMINARY INFORMATION

****

Four-Phase Systems
10700 N. De Anza Blvd.
Cupertino, CA 95014

Specifications Subject to Change

Contents

Illustrations

Tables

Section 2
User and Programmatic Interfaces

USER INTERFACE

The user interface consists of commands and operational procedures for making custom character set translation source files and a procedure selecting and initializing TM31 Terminal download operation. The commands can be broken into two main groups:

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

- o Those that actually perform the character set translation of text and print files, based on character set translation source files and internal character set representation options.

These commands are described below, and also given in "command" form in Section 4. The commands make use of the programmatic interface components described in the section "Programmatic Interface." Reading that section first may help in acquiring a detailed understanding.

Initialize Terminal Translation, itt(1M)

The initialize terminal translation command, itt(1M), converts a character set translation source file through the csinit(3) routine into a character translation data structure. Itt then passes that structure to the terminal driver through an ioctl(2) system call, with ioctl command = CSSETTT. Itt arguments are as follows:

- o A character set translation source file path name from which a terminal type identifier and, if present, a user language identifier can be determined.

- o An optional argument to indicate that translation associated with the path name argument is to be uninitialized.

- o An optional argument to indicate that the size of the initialized character set translation data structure is to be written on standard output.

The latter option is required for system administration purposes during system generation, specifically configuration of the operating system kernel, and it prevents passing of the data structure to the terminal driver. 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 control dynamically the terminal driver's translation table configuration while the operating system is executing and to gain information for use during system generation.

Terminal I/O Character Set Query and Control, cstty(1)

The cstty(1) command provides terminal I/O character translation status and selection control. For translation to be selected, 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(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.

Character translation must always be selected for TM31 Terminals so that the input sequence produced by the ALT key is delivered to user processes as an ESC. Character translation should not be selected for TM30 Terminals. In the archetype .profile file, cstty(1) is executed if $TERM = tm31.

Character Set Translation, cstrans(1)

The character set translation command, cstrans(1), reads its standard input, performs character translation optionally based on a character set translation source file converted into a translation data structure by csinit(3), and writes the translated result bytes 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 be used to translate a file from one standard internal character representation to another or to translate a file from internal character representations to nonstandard or device-dependent codes. When a character set translation source file argument is present, for outbound characters that are not in character set 000 and that have no translation table entry, cstrans(1) substitutes a question mark character, ?.

Print File Character Set Translation, csoffset(1)

Csoffset(1) handles print file character translation by mapping characters from internal character set codes to printer device- and model-dependent codes for printing. Csoffset(1) also provides optional expansion of tabs and control of the left margin width of the print image. When a character set translation source file argument is present, for outbound characters that are not in character set 000 and that have no translation table entry, csoffset(1) substitutes a question mark character, ?.

Csoffset(1) is activated as a command in the /usr/spool/lp/model/ptnnx commands text file (see lpadmin(1M) in the Series 6000 Operating System Reference Manual, Volume 1). Knowledge of printer models, model availability, and access to the appropriate printer model commands text file is embodied in the lp(1) facility (lp, lp.cnfg(1M), lpadmin, lpstat, lpsched, enable, disable, accept, reject. Note that lp.cnfg is given in the System Release Guide.) Selection of the appropriate character set translation table is a function of the particular printer interface file built by lp.cnfg(1M) that includes ptnnx, which receives arguments through the -o option of the lp(1) command.

Print File Character Translation Administrative Procedure

Configuration control of print file translation filters is done through the lpadmin(1M) command (see the Series 6000 Operating System Reference Manual, Volume 1) or the lp.cnfg(1M) command (see the System Release Guide). The ptrnx printer interface command script file containing the csoffset(1) command to provide outbound character translation for print files is located in the /usr/lib/lp/model directory and is accessed for particular printers or printer classes through printer driver scripts located in the /usr/spool/lp/interface directory. A given printer can be designated to be more than one printer model, where the model is dependent on the type of printwheel or printband currently mounted. Thus, a print request for a particular model will be queued until the printer model (printwheel) is declared available through the lp.cnfg(1M) command.

Initializing Terminals

A TM31 Terminal can be initialized with various language-specific download images, such as English (United Kingdom or USA), French, German, Spanish, and so on. A Series 6000 system can provide multiple TM31 download images, one of which you select when powering on the terminal. One of the download images, number 100, is selected by default. To select another download image, use the following procedure:

a. Hold down the space bar.

b. Power on the terminal.

c. Use the T=download_image_number option that is presented.

d. Use the B option to boot the terminal.

For TM31 Terminals on System 6300's, RS-422 download images are retrieved at terminal boot time from the download area in partition 0 of the system boot device. The download area must be configured to be large enough to contain all desired terminal download images when the system disk is formatted. The iv(1) command is used to initialize a system disk and to install the download images in the download area, based on information read from a disk description file. (See the Series 6000 Operating System Reference Manual, Volume 1, for more information on the iv(1) command.) Note that the size of the download area cannot be increased without destroying the contents of partitions numbered 1 and higher. Partition 1 is the root file system and partitions numbered 3 and higher probably contain users' files and other system software components, so enlarging the download area on a running system entails reinstallation of software. Thus, you should determine your download area size requirements before installing system software.

For TM31 Terminals on System 6600's, RS-422 terminal download images are retrieved at terminal boot time from files named

```
[!sys]<sys>wsnnn>sysimage.sys
```

where nnn is the download image number.

For TM31 Terminals on both System 6600's and System 6200's, RS-232 terminal download images are retrieved from an operating system file system.

PROGRAMMATIC INTERFACE

The programmatic interface consists of functions to perform character set translation for terminal I/O and for text files, which include files destined for printers. This interface includes both system and terminal software.

I/O Control System Call, ioctl(2)

The ISP extends the interface to the I/O control system call ioctl(2) to accommodate status requests and changes to data and option settings. The ioctl(2) command forms and their meanings are as follows:

CSGETTT	status request for translation table data
CSSETTT	changes to translation table data
CSGETO	status request for character set option settings
CSSETO	changes to character set option settings
CSSETOW	
CSSETOF	

See cstermio(7) in Section 4 for details of the extended ioctl(2) interface.

Character Set Translation Library Routine, cstrans(3)

The character set translation library routine cstrans(3) is a function that translates a character string from one representation to another, based on a character set translation data structure. A cstrans(3) parameter addresses 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. Cstrans(3) can be used to translate a sequence of bytes from one internal character set to another, for example from the Motorola private character set 040 to XSIS 058404 non-040 character sets, and from or to device-dependent character codes. For outbound characters that are not in internal character set 0 and that have no declared entry in the translation table, cstrans(3) substitutes a question mark character, ?. This routine resides in the file /usr/lib/libcs.a. See Section 4 for details on this command.

Character Set Translation Initialization Library Routine, csinit(3)

The character set translation initialization library routine csinit(3) is a function that constructs a character set translation data structure from a character set translation source file. Csinit(3) parameters are as follows:

o The file name of a character set translation source file.

- A flag to select or deselect printing of error messages.
- A status word to reflect completion status.

This routine resides in the file /usr/lib/libcs.s. See Section 4 for details on this command.

Terminal I/O Character Translation Function

The UNIX-derived operating system terminal driver performs terminal I/O character set translation. Initialization of character set translation occurs as a part of the central processor boot procedure, wherein itt(1) commands are executed to provide character set translation data from source files to the terminal driver. The terminal driver's character set translation tables 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. When translation is active, terminal-dependent input sequences (inbound codes) are translated from device-dependent codes through character set translation data structures produced by csinit(3) to XSIS 058404 16-bit characters (CharSet0 Char8Code), avoiding character sets 040, 360, and 361 wherever possible. From that character representation, codes are further translated, depending on 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 (CharSet0 Char8Code), avoiding character sets 040, 360, and 361 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 option to echo input characters is selected (stty echo), the codes echoed by the terminal driver are exactly as received from the terminal. Thus 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 the terminal is powered on, or by user activation of a download program by either making a selection offered by the terminal ROM program or by executing the /usr/local/bin/tdl program. The tdl program downloads a TM31 Terminal through an RS-232 connection and requires that the terminal contain a boot ROM upgraded to at least version 2.0. 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, and what the mapping is for international character output sequences.

The TM31 Terminal download image is somewhat configurable and accommodates keyboard variants for various national languages. The input sequences transmitted by all of the keys not on the basic "typewriter" keypad, plus the TAB, BACKSPACE, and RETURN keys, are identical for all TM31 download variants. Appendix A 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. The initialize terminal translation command, itt(1M), reads this directory to obtain the source for character set translation tables during the system boot procedure. The /usr/lib/cs.term directory contains all of the system's character set translation source files for terminals; 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 is one that involves a key that does not by itself 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, including Motorola private character set 040, that the terminal can display. When there is no translation table entry for an outbound accented character, the given letter without the accent is used in its place. Also, when there is no translation table entry for other outbound characters that are in internal character set 0, the question mark character, ?, is used in their place.

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.

Character set translation source files also occur for device types other than terminals, such as printers (see the Section on "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 character set translation source file syntax described in this section covers all cases to which such files apply: terminal, printer, and internal character set translations.

File Structure

The set-up of these files is as follows:

name /usr/lib/cs.term/$TERM.$LANG

where

$TERM is the terminal type identifier

$LANG is the user language identifier. $LANG can be null for terminal types that don't have language-dependent characteristics, in which case the period preceding $LANG is also omitted. The values of LANG are as follows:

Value	Language
cdn	Canadian
deut	German
engli	English (USA)
esp	Spanish
fra	French
hol	Dutch
sve	Swedish
uk	English (UK)

format These files are ASCII line-image files. Each line can contain comments, one or more keywords, and one or more values. Lines consisting only of white space (blank, tab, and new-line characters) can also occur.

A pound sign (#) introduces a comment, which occupies the remainder of the line. Comments are of no significance to the csinit(3) routine, which processes these files.

White space characters are separators and delimit keywords and values.

The keywords and their meanings are as follows:

inbound:	start of section for inbound characters
outbound:	start of section for outbound characters
internal:	start of section for internal characters
format7:	7-bit ASCII shift-out/shift-in notation for outbound characters
primary:	primary character set declaration
cselect:	non-primary character set declaration
translate:	translation pattern declaration
range:	translation range declaration
accent:	translation accent declaration

Values and their meanings are as follows:

\nnn nnn octal; nnn must be three digits
\E 033, the value of the ASCII ESC code
\x 1-byte variable value, other than an accent, in a translate statement
\a 1-byte variable value of an accent (XSIS 058404 character set 0 codes 301 through 317) in a translate statement. This value also indicates an accent in certain value statements.
\\ 134, the value of the ASCII backslash code
\# 043, the value of the ASCII pound sign code

In addition, ASCII-displayable characters other than space (040), dollar sign (044), and broken vertical bar (174) can be used to represent their ASCII 1-byte value.

Examples:

- o the value represented by the string \ENa is 033 116 141, which is three bytes.
- o the value represented by the string \174\\ is 174 134, which is two bytes.
- o the value represented by the string \ENa also represents one 3-byte value.
- o the value represented by the string \002 \141 represents two 1-byte values.

Keywords and Values

Keywords and values comprise statements of the following forms. Each statement as defined must occupy exactly one line.

```
inbound
outbound
internal
format7
primary <pattern>
cselect <cset_num> <pattern>
translate <pattern><x_variable> range <lo_value> <hi_value>
translate <pattern><a_variable> accent <value>
[<accent_flag>] <value_string> [<value_string> ... ]
```

Inbound, outbound, or internal must be the first non-comment line in the file. This keyword indicates the start of statements that apply to inbound (from the keyboard), outbound (to the screen or printer), or internal (XSIS 058404 character code standard) characters, respectively. The occurrence of inbound, outbound, or internal terminates any prior statement group. A terminal translation file must contain exactly one inbound declaration and exactly one outbound declaration. Printer translation files must contain exactly one outbound declaration. An internal character set translation file must contain exactly one internal declaration.

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-out 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)
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)
constant byte value
  (is to write)
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, and declares that the translator program will

```
translate
a constant byte value pattern of zero or more bytes
  (followed immediately by)
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
  (with a lower bound)
constant byte value
  (and the upper bound)
constant byte value
  (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 and declares that the translator program will

```
translate
a constant byte value pattern of zero or more bytes
  (followed immediately by)
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
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 would appear 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 would appear 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> that precedes the <value_string> that declares the device output byte sequences following translate statements. This extra <value_string> declares the device character set number for that output sequence. Thus, for non-primary device character set output sequences, the value declaration line(s) following a translate ... range statement would appear in the two-value format

```
    <cset_num> <device_output_sequence>
```

For a non-primary device character set output sequences, the value declaration line(s) following a translate ... accent statement, would appear in a three-value format

```
<accented_char> <cset_num> <device_output_sequence>
```

The <cset_num> value is used to identify the device character set selection sequence to be sent to the device if the <cset_num> character set is not already selected. This sequence is declared in the related cselect statement. Figure 3-1 is an example of how 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 diaeresis
# 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 diaeresis
o \001 \174      # o diaeresis
```

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 diaeresis
\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 diaeresis
# 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 a sample of terminal character code translations. This example shows 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 characters 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 diaeresis
\a     \310 O                       # O diaeresis
\a     \310 U                       # U diaeresis
\a     \310 a                       # a diaeresis
\a     \310 o                       # o diaeresis
\a     \310 u                       # u diaeresis
\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
```

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

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

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
```

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

```
translate \000\x range \241 \377
?
?
\ENU             # pound sign
\044             # dollar sign
?
?
\ENk             #247 section
?
?
?
?
\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              # ENx from ENz CS0131
?
\ENj             # paragraph
\EN[             # center
\ENc             #270 divide
?
?
?
\EN{             #274 1/4                   # EN{ from EN} CS0131
\ENz             # 1/2                 # ENz from EN¦ CS0131
\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
```

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

```
?
\EN1                    # superscript 1
\EN/                    # registered
\EN.                    # copyright
\EN:                    #324 trademark
?
?
?
?                       #330
?
?
?
?
?
?
?
?                       #340
?
?
?
?
?
?
?
?                       #350
?
?
?
?
?
?
?
?                       #360
?
?
?
?
?
?
?
?                       #370
?
?
\ENF                    #373 esset
?
?
?
?                       #377
```

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

```
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

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 diaeresis
O \ENA             # O diaeresis
U \ENB             # U diaeresis
a \ENC             # a diaeresis
o \END             # o diaeresis
u \ENE             # u diaeresis


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               # !=
```

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

```
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

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 printer output character code translations. 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 character set translation source files for printers. These files declare translations for outbound characters and accommodate all XSIS 058404 characters, including Motorola private character set 040, that the printer can print. 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 used in their place.

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 System Release Guide for information on this command.)

<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 (USA)
esp	Spanish
fra	French
hol	Dutch
sve	Swedish
uk	English (UK)

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

Figure 3-3 is sample of printer character code translations. This example shows a character set translation source file for PT34 Character Printer ASCII and German character sets.

```
# 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.
#                 LPDEST=pt34
#                 LANG=engli
#
# Revision History:
#
#      GOC.       ISDC, Cork, Ireland.          Feb. 13 1985.
#      Initial version.
#
#
primary           \E(B      #USA-ASCII
cselect \001      \E(A      #UK
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 quote
?                 # \252    Left double quote
?                 # \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
```

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

```
?                 # \263   Superscript 3
?                 # \264   Multiplication Sign
?                 # \265   Micro sign
?                 # \266   Paragraph Mark
?                 # \267   Centered dot
?                 # \270   Division Sign
?                 # \271   Right single quote
?                 # \272   Right double quote
?                 # \273   Right double guillemet
?                 # \274   Fraction 1/4
?                 # \275   Fraction 1/2
?                 # \276   Fraction 3/4
\005 ]            # \277   Inverted Question Mark
?                 # \300   Reserved
`\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   Diaeresis (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
```

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

```
?                  # \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
?                  # \364    h stroke
?                  # \365    Dotless i
?                  # \366    ij digraph
?                  # \367    l middle dot
-\010l             # \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-umlaut replaces [
O \002 \134 # O-umlaut replaces backslash
U \002 ]    # U-umlaut replaces ]
a \002 {    # a-umlaut replaces {
o \002 ¦    # o-umlaut replaces ¦
u \002 }    # u-umlaut replaces }

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

translate \000\a accent \312
A \003 ]
a \003 }
```

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

```
translate \000\a accent \302
E \003 @
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 form of the *Series 6000 Operating System Reference Manual*. The "Synopsis" subsections give summaries of the commands, the "Files" subsections list the files built into the programs, 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

These shell procedures are executed through entries in *inittab*(4) files by *init*(1M) when the system is changed out of SINGLE USER mode. The *brc* procedure clears the mounted file system table, */etc/mnttab* (see *mnttab*(4)), and loads any programmable micro-processors with their appropriate scripts.

The *bcheckrc* procedure performs all the necessary consistency checks to prepare the system to change into multi-user 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 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.

The *rc* procedure starts all system daemons before the terminal lines are enabled for multi-user mode. In addition, file systems are mounted and accounting, error logging, and system activity logging are activated in this procedure.

The *ttrc* procedure initializes character set translation tables for the terminal device driver through the itt(1M) command.

These shell procedures, in particular rc, can be used for several run-level states. The who(1) command can be used 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 reads its standard input, translates incoming characters provides print file formatting in accord with selected options, and writes the resultant characters on its standard output. Csoffset is used as a filter program for processing files destined for printers.

The options and their meanings are as follows:

-tabs Selects expansion of tab characters on output. Each tab (ASCII HT character) in the input stram is replaced by eight spaces (ASCII SP characters) on output. If this option is omitted, then 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 this option is omitted, then there is no insertion of CR characters 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 left margin of the print image. If this option is omitted, then the value 0 is used.

-t=<filename> Specifies the <filename> of a character set translation source file. This source file is to be used to translate 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 which have no translation table entry, csoffset substitutes the question mark character, ?.

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

DIAGNOSTICS

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

csoffset: unable to allocate space to build translation table
This message indicates a system error. Call your Customer Support Representative for assistance.

csoffset: array overflow while building translation table
This message indicates a system error. Call your Customer Support Representative for assistance.

csoffset: invalid option <option>
usage: csoffset [-tabs] [onclr] [<offset>] [-t=<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 will 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] [<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. When output format options are specified, translation is from one internal character representation to another. When a character set translation source file <filename> 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 (?) character.

The output format arguments -f7, -od040, -o8, and -o16, are mutually exclusive from the <filename> argument, and the output format arguments -od040, -o8, and -o16, are mutually exclusive from one another. The options and their meanings 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.

-f7 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 to be 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.

This option can be used with or without any of the -od040, -o16, and -o8 options. When none of the following options is selected, then the output format is XSIS 058404 standard 8-bit stringlet representation without an initial CSelect byte.

-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 (?) character.

-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.

DIAGNOSTICS

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

```
cstrans: invalid option
  usage: cstrans [-id040] [[-f7  [-od040 ¦ -o8 ¦ -o16]] ¦ <filename>]
   Change the command to include one or more valid options only.
```

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 will also appear.

SEE ALSO

cstty(1).

cstty(1)

cstty - character set control for a terminal

SYNOPSIS

cstty [options]

DESCRIPTION

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.

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 this option is selected, the terminal driver translates inbound and outbound device-dependent characters to and from standard internal character representations. If none of the three internal character set representation options, cst16, csfmt7, and cs040, is selected, then inbound characters are delivered 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 user language. This option must be present on the first cstty call for a terminal session.

If it is omitted 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.

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

If only <term> is specified, then this option redefines the terminal type but does not change the definition of the language identifier. If only .<lang> is given, then the language identifier is redefined but the terminal type definition remains unchanged. Note that this second form requires the leading period "." character.

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 can write 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] <filename>
```

DESCRIPTION

Itt constructs a character set translation data structure, given a character set translation source file called <filename>. 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 to occur 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 <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 the period "." is also omitted.

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, the translation table is uninitialized.

The -s argument causes itt to write the size of the character set translation data structure on standard output and not to send that structure to the terminal driver. This option is used to gain kernel size information during the operating system generation procedure. If a number is included with this argument, then the size value reported is in terms of units of that number of bytes. For example, -s64 would yield 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. The -r and -s arguments are mutually exclusive.

FILES

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

DIAGNOSTICS

Itt can write 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 a terminal screen.

itt: missing filename argument
Change the command to include a character set translation source file name. 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 <filename>     - remove translation table
           itt -s <filename>     - report size of translation table
           itt <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 <translate_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 Motorola TM31 Terminal.

Script, an ASCII text file, defines the applications available to the *keyprompt* user. Someone familiar with *keyprompt* conventions and operation 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:

- o A message line. This holds messages from keyprompt or the application being run.

- o A tag line. This labels the particular application or keyprompt display that is active.

- o 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.

- o 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, it displays a help message for the particular display. 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 directive 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 directive 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 directive 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 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 f10.

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)

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 Motorola TM31 or Graphics 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 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.

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 which 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 virutal display is usally 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).

o 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.

Here are the valid wm user commands:

ACTION-RESET (SPLIT)

Split the active window, creating 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 the user reactivates the window and types 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's, the active window's tag line is displayed full intensity with the other tag lines displayed half intensity. On Graphics Terminals, the active window's tag line is displayed in bold with the other tag lines displayed without bold.

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.

The SPLIT key becomes inoperative if the terminal already displays its maximum number of windows or if a user program has disabled window splitting.

ACTION-F9 (BELOW)
The window below the active window becomes the active window with the old active window becoming 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.

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.

ACTION-F5 (SHRINK)
The active window decrease 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-CODE-F5
The active window becomes zero lines long.

Shrinking the top window increases the size of the window below; shrinking any other window increases the size of the window 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-CODE-F5
The active window increases in size until all other windows are zero lines long.

Growing the top window decreases the size of the window below; growing any other window decreases the size of the window below. 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-SCROLL UP
The active window is scrolled up a line. Ignored if the window already shows the very bottom of the virtual display or if the cursor is on the window's top line.

ACTION-SCROLL DOWN
The active window is scrolled down a line. Ignored if the window already shows the very top of the virtual display or if the cursor is on the window's bottom line.

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/ttyXX'
if [$tty -ge 20 -a $tty -le 27]
then
     exec wm
else
     exec $SHELL
fi
```

SEE ALSO

sh(1).

WARNING

If a program quickly outputs two things 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 returns a pointer to a structure of the type shown in Figure 4-1.

```
/*
 * Character set translate 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 reads the character set translation source file named by its <filename> argument, converts it to a csttbl character set translation table, and returns a pointer to that structure. The RETURN value of csinit is NULL if the conversion operation was unsuccessful.

Validation of the character set translation source file is performed. If the <silent> argument is FALSE, then diagnostics are written on the standard error file.

Completion status is returned in the <status> argument. Values for completion status are defined in the cs.h header file.

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.

```
line %d - redeclaration of character set %d
line %d - undefined character set number %d
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)

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 using 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 are in a buffer of unsigned char or short. The output data are placed into a similar buffer.

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:

Mode	Function
0	Translate from internal 16-bit to internal 16-bit using an internal translation table. This mode is used to enforce the Motorola private character set or to avoid the character sets for Motorola private, ligature, and accented characters.
1	Translate from external device character code to internal 16-bit characters using 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.
3	Translate from XSIS 058404 strings to internal 16-bit characters.
4	Translate from internal 16-bit characters to external device character codes using 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 are expected to provide characters in the standard internal character sets, avoiding sets 040, 360, and 361. They may assume that their input is coming 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 diaeresis symbol can be represented in three different internal forms:

<000><310> <000><101>	standard form: diaeresis 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>.

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:

o Managing terminal I/O translation tables at the system-wide level.

o 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 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 below.

```
/*
 * 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 csthdr  cs_hdr;
charcs_tbl[CSMAXSIZ];
     }cs_u;
};
#endif

/* 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;
```

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

```
#if !KERNEL || define_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. Cstthdr Structure (Page 2 of 2)

```
ushort      cs_tnum;           /* table number */
```

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 will return a copy of the translation table from that slot. A CSGETTT with a cs_tnum of an empty slot will set errno == ENXIO. A CSSETTT with a cs_tnum of an occupied slot will write 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 will result 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.

```
ushort      cs_tlen;       /* length of the complete translation table
                              in bytes */
```

Size of the csttbl structure in bytes. See the CSMAXSIZ comment in Figure 4-2.

```
csttname    cs_tname;      /* name of the translation table */
```

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.

```
ushort      cs_nref;       /* number of open file references */
```

Number of users of the translation table. Set by the terminal driver to indicate whether or not the translation table is in use.

```
ushort      cs_pesc;       /* position of the escape prefix index */
```

Offset of the first csescix escape prefix index structure in the csttbl.cs_tbl character array.

```
ushort        cs_nesc;        /* number of escape sequence prefixes */
```

Number of csescix escape prefix index structures in the csttbl.cs_tbl character array.

```
ushort        cs_pchset;      /* position of the character set index */
```

Offset of the first cscsix character set index structure in the csttbl.cs_tbl character array.

```
ushort        cs_nchset;      /* number of character sets */
```

Number of cscsix character set index structures in the csttbl.cs_tbl character array.

```
ushort        cs_ptrchar;     /* position of the table of 16-bit
                                 translated characters */
```

Offset of the Char16Code translated characters table in the csttbl.cs_tbl character array.

```
ushort        cs_nextcs;      /* number of external character sets */
```

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.

```
ushort        cs_pextcs;      /* position of the table of escape
                                 sequences used to select the external
                                 character sets */
```

Offset of a ushort array of indexes to NULL-terminated strings that contain device character set selection output sequences.

```
ushort        cs_ttflag;      /* translation table flag bits */
```

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.

The csescix structure is used to map 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 below.

```
/* 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 */
};
```

Figure 4-3. Csescix Structure

```
unsigned char    cs_esc[4];      /* escape sequence prefix */
```

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 character.

```
unsigned char    cs_esclo;       /* lowest valid character after prefix */
```

Lowest value of a range of characters subject to translation that follows the above-defined escape sequence prefix. An example of this value is the lowest value for x in the sequence ESC N x.

```
unsigned char    cs_eschi;       /* highest valid character */
```

Highest value of a range of characters subject to translation that follows the above-defined escape sequence prefix. An example of this value is the highest value for x in the sequence ESC N x.

```
ushort           cs_esctt;       /* position of the translation table */
```

Offset of the csttent translation table entry in the csttbl.cs_tbl character array associated with this escape sequence prefix.

The cscsix structure, shown in Figure 4-4, is used to map Char16Code internal codes to device-dependent outbound codes. Descriptions of the fields in the cscsix structure follow.

```
/* 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 */
};

#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

```
unsigned char          cs_csnum;       /* character set number */
```

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.

```
unsigned char          cs_cspfx[2];    /* possible prefix character(s)
                                          (character set 000) for accented
                                          characters and ligatures */
```

Contains (in cs_cspfx[0]) a character set 000 accent character.

```
unsigned char          cs_nlist;       /* number of list entries */
```

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.

```
ushort                 cs_plist;       /* position of list entries */
```

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.

```
ushort                 cs_cstt;        /* position of the translation
                                          table */
```

Offset of the csttent translation table entry in the csttbl.cs_tbl character array associated with this character set index.

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   /* CO+cs_ttnib plus cs_ttchar in character
                              set 000 */
#define CS_ERR        15   /* invalid character */
```

Figure 4-5. Csttent Structure (Page 1 of 2)

```
/* 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_CS0, 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 diaeresis
   (0x00c8) followed by an uppercase A (0x0041).
 */
```

Figure 4-5. Csttent Structure (Page 2 of 2)

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

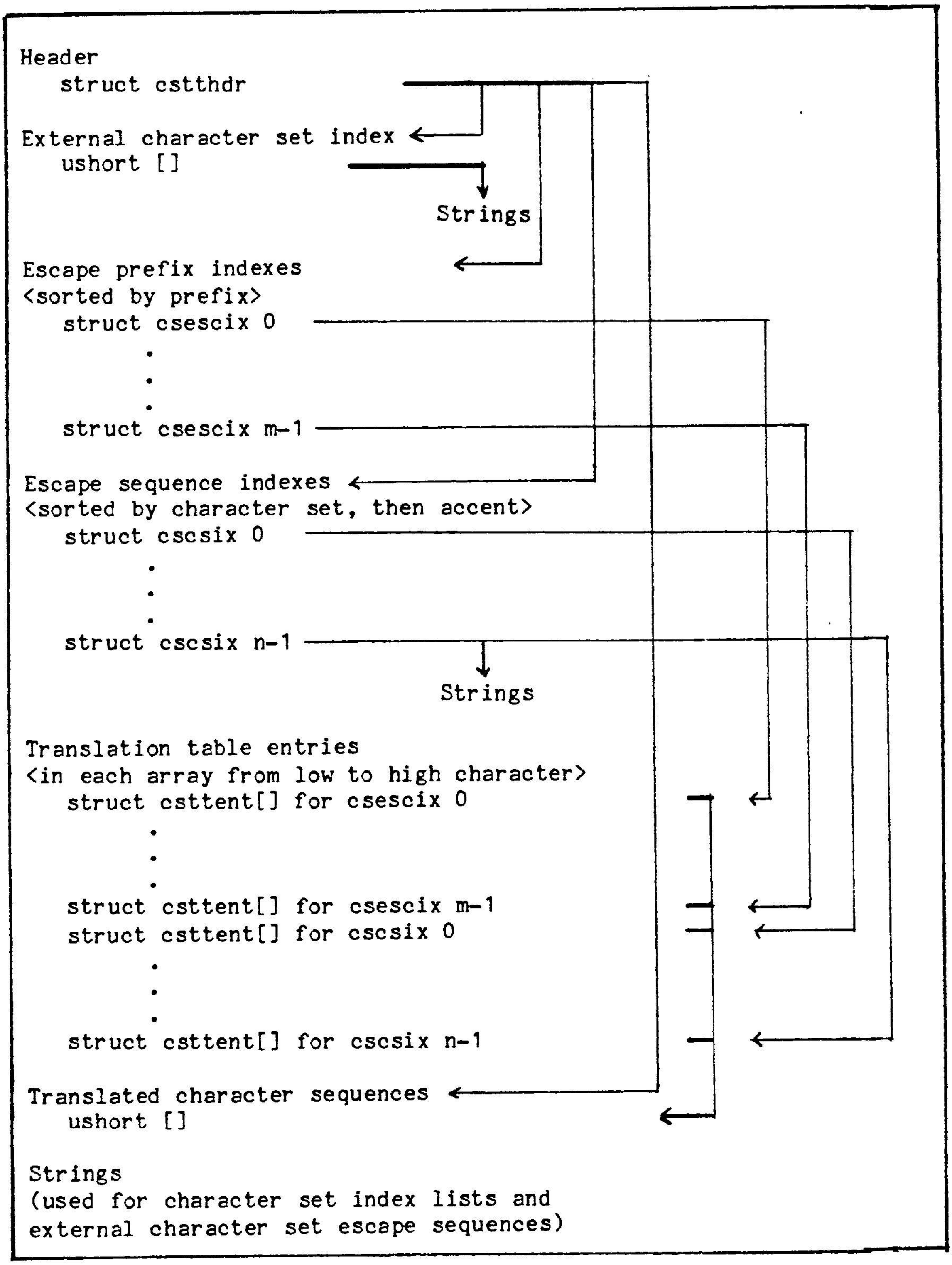
Header
struct cstthdr
External character set index
ushort []
Strings
Escape prefix indexes
<sorted by prefix>
struct csescix 0
struct csescix m-1
Escape sequence indexes
<sorted by character set, then accent>
struct cscsix 0
struct cscsix n-1
Strings
Translation table entries
<in each array from low to high character>
struct csttent[] for csescix 0
struct csttent[] for csescix m-1
struct csttent[] for cscsix 0
struct csttent[] for cscsix n-1
Translated character sequences
ushort []
Strings
(used for character set index lists and
external character set escape sequences)

Figure 4-6. Character Set Translation Table Structure

Use the cstty(1) command to control terminal I/O character set translation options. The ioctl(2) system calls, which use the structure shown in Figure 4-7, also provide an interface for the control of these options. This structure is defined in <cstty.h>.

```
/*
 * 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. Structure for Ioctl(2) Terminal
Character Set Translation Options

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. This form should be used when changing 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 are
     used to convert to the Motorola private character set 040, or to 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 used 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 used 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 character set provided for the Series 6000 operating system and application programs, Motorola private character set 040, is defined by the XSIS 058404 character code standard (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 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		
Octal	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

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

Character			Character	XSIS 058404		
Code			or	Code		
Octal	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
034	28	1c	(FS)	000	034	File Separator
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 Quote
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 Quote
050	40	28	(	000	050	Left or Open Paren
051	41	29	)	000	051	Right or Close Paren
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

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

Character			Character	XSIS 058404		
Code			or	Code		
Octal	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
114	76	4c	L	000	114	L
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 Quote
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

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

Character Code			Character or	XSIS 058404 Code		
Octal	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
174	124	7c	¦	357	153	Broken Vertical Bar
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 diaeresis
242	162	a2	Å	000	312 A	A ring
243	163	a3	Ã	000	304 A	A tilde
244	164	a4	Æ	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 diaeresis
251	169	a9	Õ	000	304 O	O tilde
252	170	aa	Œ	000	352	OE ligature

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

Character Code			Character or	XSIS 058404 Code		
Octal	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
253	171	ab	ø	000	351	O slash
254	172	ac	ü	000	310 U	U diaeresis
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 diaeresis
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 diaeresis
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 diaeresis
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 diaeresis
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 diaeresis
312	202	ca	ů	000	312 u	u ring
313	203	cb	˚	000	312	Ring
314	204	cc	¨	000	310	Diaeresis
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

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

Character			Character	XSIS 058404		
Code			or	Code		
Octal	Dec	Hex	(Mnemonic)	CharSet8	Char8Code	Description
333	219	db	├			Box: right pointing tee
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 diaeresis
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