SAS Publishing

SAS® 9.1 National Language Support (NLS)

User's Guide

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2004. *SAS® 9.1 National Language Support (NLS): User's Guide*. Cary, NC: SAS Institute Inc.

SAS® 9.1 National Language Support (NLS): User's Guide

ISBN 1-59047-194-6

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, January 2004

Contents

What's New

Overview

- All information that is related to NLS has been consolidated into a single document for your convenience.
- The LOCALE= option supports a new set of locale values in the form of Portable Operating System Interface (POSIX) names.
- The LOCALE= option supports new values that identify unique language and country combinations.
- The LIBNAME statement for Base SAS supports three new options for NLS: CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER=.
- The LIBNAME statement for the XML engine supports three new options for NLS: ODSCHARSET=, ODSTRANTAB=, and XMLENCODING=.
- The LIBNAME statement in SAS/SHARE supports the RENCODING= option for NLS.
- Numerous NLS formats, informats, and functions are new. These new language elements are in the following categories: Bi-directional text handling, Date/Time, Monetary, and Unicode.

Note: z/OS is the successor to the OS/390 operating system. SAS 9.1 is supported on both OS/390 and z/OS operating systems and, throughout this document, any reference to z/OS also applies to OS/390, unless otherwise stated. △

Details

New Document Consolidates Information about NLS

SAS 9.1 introduces the *SAS National Language Support (NLS): User's Guide*, which consolidates all information about NLS that was previously contained in multiple SAS documents. The *SAS National Language Support (NLS): User's Guide* provides comprehensive conceptual information and detailed syntax for all SAS language elements that contain NLS properties.

Expanded Values for the LOCALE= System Option

- Locale can be specified by using Portable Operating System Interface (POSIX) naming standards. For example, the en_US value is the POSIX equivalent for the SAS value English_UnitedStates.
- Previous releases of SAS software specified some LOCALE= values in the form of *language*. The LOCALE= option supports new values that identify unique language and country combinations that are specified in the form *language_country*. Some single LOCALE= values have been replaced by more granular values. Some new values have been added, and some values have been deleted. Here is a summary of the changes to LOCALE= values:

  Arabic
  : The single LOCALE= value for Arabic has been deleted. The following new values have been added: Arabic_Algeria, Arabic_Bahrain, Arabic_Egypt, Arabic_Jordan, Arabic_Kuwait, Arabic_Lebanon, Arabic_Morocco, Arabic_Oman, Arabic_Qatar, Arabic_SaudiArabia, Arabic_Tunisia, and Arabic_UnitedArabEmirates.

  Chinese
  : The single LOCALE= value for Chinese has been deleted. The values Chinese_Simplified and Chinese_Traditional have also been deleted.

  Dutch
  : The single LOCALE= value for Dutch has been deleted. The following new values have been added: Dutch_Belgium and Dutch_Netherlands.

  English
  : The single LOCALE= value for English has been deleted. The following new values have been added: English_HongKong, English_India, and English_Singapore. The English_Britain value has been changed to English_UnitedKingdom.

  Estonian_Estonia
  : The LOCALE= value for Estonian_Estonia is new.

  French
  : The single LOCALE= value for French has been deleted. A new value, French_Luxembourg, has been added.

  German
  : The single LOCALE= value for German has been deleted. The following new values have been added: German_Liechtenstein and German_Luxembourg.

  Spanish
  : The LOCALE= values for Spain and Spanish_LatinAmerica have been deleted. The single LOCALE= value for Spanish_LatinAmerica has been replaced by the following new values: Spanish_Argentina, Spanish_Bolivia, Spanish_Chile, Spanish_Columbia, Spanish_CostaRica, Spanish_DominicanRepublic, Spanish_Ecuador, Spanish_ElSalvador, Spanish_Guatemala, Spanish_Honduras, Spanish_Mexico, Spanish_Nicaragua, Spanish_Panama, Spanish_Paraguay, Spanish_Peru, Spanish_PuertoRico, Spanish_UnitedStates, Spanish_Uruguay, and Spanish_Venezuela.

  For a comprehensive list, see Chapter 21, "Values for the LOCALE= System Option," on page 397.

New Options in LIBNAME Statements for NLS

- The LIBNAME statement in Base SAS supports the following new options for NLS:

  CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER=
  : specify the attributes for character variables that are needed in order to process (or transcode) a SAS file.

- The LIBNAME statement for the XML engine supports the following new options for NLS:

  ODSCHARSET=
  : specifies the character set to be generated in the META declaration for the output.

  ODSTRANTAB=
  : specifies the translation table to use when transcoding an XML document for an output file.

  XMLENCODING=
  : specifies the encoding to use when reading, writing, copying, or saving an external file.

- The LIBNAME statement in SAS/SHARE supports the following new option for NLS:

  RENCODING=
  : specifies that the ASCII-based or EBCDIC-based encoding be used when transcoding data for a SAS/SHARE server session that is using an ASCIIANY or an EBCDICANY session encoding.

Formats for NLS

The following formats for NLS are new:

$CPTDWw.
: writes a character string in Hebrew text that is encoded in IBM-PC (cp862) to Windows Hebrew encoding (cp1255).

$CPTWDw.
: writes a character string that is encoded in Windows (cp1255) to Hebrew DOS (cp862) encoding.

HDATEw.
: writes date values in the form *yyyy mmmmm dd* where *yyyy* is the year, *mmmmm* represents the month's name in Hebrew, and *dd* is the day of month.

HEBDATEw.
: writes date values according to the Jewish calendar.

$LOGVSw.
: writes a character string that is in left-to-right logical order to visual order.

$LOGVSRw.
: writes a character string that is in right-to-left logical order to visual order.

NLDATEw.
: converts a SAS date value to the date value of the specified locale, and then writes the value in the format of the date value.

NLDATEMNw.
: converts a SAS date value to the date value of the specified locale, and then writes the date value in the format of the name of the month.

NLDATEWw.
: converts a SAS date value to the date value of the specified locale, and then writes the date value in the format of the date and the day of the week.

NLDATEWNw.
: converts the SAS date value to the date value of the specified locale, and then writes the date value in the format of the name of the day of the week.

NLDATMw.
: converts a SAS datetime value to the datetime value of the specified locale, and then writes the value in the format of the datetime.

NLDATMAPw.
: converts a SAS datetime value to the datetime value of the specified locale, and then writes the value in the format of the datetime with a.m. or p.m.

NLDATMTMw.
: converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value in the format of the time of the day.

NLDATMWw.
: converts a SAS date value to a datetime value of the specified locale, and then writes the value in the format of day of the week and the datetime.

NLMNYw.d
: writes the monetary format of the local expression in the specified locale using local currency.

NLMNYIw.d
: writes the monetary format of the international expression in the specified locale.

NLNUMw.d
: writes the numeric format of the local expression in the specified locale.

NLNUMIw.d
: writes the numeric format of the international expression in the specified locale.

NLPCTw.d
: writes percentage data of the local expression in the specified locale.

NLPCTIw.d
: writes percentage data of the international expression in the specified locale.

NLTIMAPw.
: converts a SAS time value to the time value of a specified locale, and then writes the value in the format of the time with a.m. or p.m.

NLTIMEw.
: converts a SAS time value to the time value of the specified locale, and then writes the value in the format of the time.

$UCS2Bw.
: writes a character string in big-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS2BEw.
: writes a big-endian, 16-bit, universal character set code in 2 octets (UCS2), character string in the encoding of the current SAS session.

$UCS2Lw.
: writes data in little-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS2LEw.
: writes a character string that is encoded in little-endian, 16-bit, universal character set code in 2 octets (UCS2), in the encoding of the current SAS session.

$UCS2Xw.
: writes a character string in native-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS2XEw.
: writes a native-endian, universal character set code in 2 octets (UCS2), character string in the encoding of the current SAS session.

$UCS4Bw.
: writes a character string in big-endian, 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding.

$UCS4BEw.
: writes a big-endian, 32-bit, universal character set code in 4 octets (UCS4), character string in the encoding of the current SAS session.

$UCS4Lw.
: writes a character string in little-endian, 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding.

$UCS4LEw.
: writes a little-endian, 32-bit, universal character set code in 4 octets (UCS4), character string in the encoding of the current SAS session.

$UCS4Xw.
: writes a character string in native-endian, 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding.

$UCS4XEw.
: writes a native-endian, 32-bit, universal character set code in 4 octets (UCS4), character string in the encoding of the current SAS session.

$UESCw.
: writes a character string that is encoded in the current SAS session in Unicode escape (UESC) representation.

$UESCEw.
: writes a Unicode escape (UESC) representation character string in the encoding of the current SAS session.

$UNCRw.
: writes a character string that is encoded in the current SAS session in numeric character representation (NCR).

$UNCREw.
: writes the numeric character representation (NCR) character string in the encoding of the current SAS session.

$UPARENw.
: writes a character string that is encoded in the current SAS session in Unicode parenthesis (UPAREN) representation.

$UPARENEw.
: writes a Unicode parenthesis (UPAREN) character string in the encoding of the current SAS session.

$UTF8Xw.
: writes a character string in universal transformation format (UTF-8) encoding.

$VSLOGw.
: writes a character string that is in visual order to left-to-right logical order.

$VSLOGRw.
: writes a character string that is in visual order to right-to-left logical order.

WEEKUw.
: writes a week number in decimal format by using the U algorithm.

WEEKVw.
: writes a week number in decimal format by using the V algorithm.

WEEKWw.
: writes a week number in decimal format by using the W algorithm.

Informats for NLS

The following informats for NLS are new:

$CPTDWw.
: reads a character string that is encoded in Hebrew DOS (cp862) and then converts the character string to Windows (cp1255) encoding.

$CPTWDw.
: reads a character string that is encoded in Windows (cp1255) and then converts the character string to Hebrew DOS (cp862) encoding.

$LOGVSw.
: reads a character string that is in left-to-right logical order and then converts the character string to visual order.

$LOGVSRw.
: reads a character string that is in right-to-left logical order and then converts the character string to visual order.

NLMNYw.d
: reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value.

NLMNYIw.d
: reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value.

NLNUMw.d
: reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

NLNUMIw.d
: reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

NLPCTw.d
: reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

NLPCTIw.d
: reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value.

$UCS2Bw.
: reads a character string that is encoded in big-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS2BEw.
: reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS2Lw.
: reads a character string that is encoded in little-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS2LEw.
: reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS2Xw.
: reads a character string that is encoded in 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS2XEw.
: reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, universal character set code in 2 octets (UCS2), Unicode encoding.

$UCS4Bw.
: reads a character string that is encoded in big-endian, 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS4Lw
: reads a character string that is encoded in little-endian, 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS4Xw.
: reads a character string that is encoded in 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding, and then converts the character string to the encoding of the current SAS session.

$UCS4XEw.
: reads a character string that is in the encoding of the current SAS session and then converts the character string to 32-bit, universal character set code in 4 octets (UCS4), Unicode encoding.

$UESCw.
: reads a character string that is encoded in Unicode escape (UESC) representation, and then converts the character string to the encoding of the current SAS session.

$UESCEw.
: reads a character string that is encoded in the current SAS session, and then converts the character string to Unicode escape (UESC) representation.

$UNCRw.
: reads the numeric character representation (NCR) character string, and then converts the character string to the encoding of the current SAS session.

$UNCREw.
: reads a character string in the encoding of the current SAS session, and then converts the character string to session-encoded numeric character representation (NCR).

$UPARENw.
: reads a character string that is encoded in Unicode parenthesis (UPAREN) representation, and then converts the character string to the encoding of the current SAS session.

$UPARENEw.
: reads a character string that is encoded in the current SAS session, and then converts the character string to the encoding of the Unicode parenthesis (UPAREN) representation.

$UPARENPw.
: reads a character string that is encoded in Unicode parenthesis (UPAREN) representation, and then converts the character string to the encoding of the current SAS session with national characters remaining in the encoding of the UPAREN representation.

$UTF8Xw.
: reads a character string that is encoded in Unicode transformation format (UTF-8), and then converts the character string to the encoding of the current SAS session.

$VSLOGw.
: reads a character string that is in visual order and then converts the character string to left-to-right logical order.

$VSLOGRw.
: reads a character string that is in visual order and then converts the character string to right-to-left logical order.

WEEKUw.
: reads the format of the number-of-week value within the year and returns a SAS date value by using the U algorithm.

WEEKVw.
: reads the format of the number-of-week value within the year and returns a SAS date value using the V algorithm.

WEEKWw.
: reads the format of the number-of-week value within the year and returns a SAS date value using the W algorithm.

Functions for NLS

The following functions for NLS are new:

NLDATE
: converts the SAS date value to the date value of the specified locale by using the date-format modifiers.

NLDATM
: converts the SAS datetime values to the time value of the specified locale using the datetime-format modifiers.

NLTIME
: converts the SAS time or datetime value to the time value of the specified locale using the time-format modifiers.

TRANTAB
: transcodes a data string by using a translation table.

VARTRANSCODE
: returns the transcode attribute of a SAS data set variable.

VTRANSCODE
: returns a value that indicates whether transcoding is enabled for the specified character variable.

VTRANSCODEX
: returns a value that indicates whether transcoding is enabled for the specified argument.

WEEK
: returns the week-number value.

PART 1

NLS Concepts

CHAPTER 1

National Language Support (NLS)

Overview to National Language Support

National Language Support (NLS) is a set of features that enable a software product to function properly in every global market for which the product is targeted. The SAS System contains NLS features to ensure that SAS applications can be written so that they conform to local language conventions. Typically, software that is written in the English language works well for users who use the English language and use data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work well for users in other regions of the world. NLS in SAS enables users in regions such as Asia and Europe to process data successfully in their native languages and environments.

SAS provides NLS for data as well as for code under all operating environments and hardware, from the mainframe to the personal computer. This support is especially important to international users who are running applications in a client/server environment. SAS provides NLS for mainframes while maintaining consistency with applications that were developed with previous versions of SAS.

NLS is applied to data that is moved between machines; for example, NLS ensures that the data is converted to the correct format for use on the target machine.

Text-string operations are sensitive to SAS settings for language and region. This enables correct results for such operations as uppercasing and lowercasing characters, classifying characters, and scanning data. SAS provides features to ensure that national characters, which are characters specific to a particular nation or group of nations, display and print properly.

Software applications that incorporate NLS can avoid dependencies on language-specific or cultural-specific conventions for software features such as:

- character classifications
- character comparison rules
- code sets
- date and time formatting
- interface
- message-text language
- numeric and monetary formatting
- sort order.

Definition of Localization and Internationalization

Localization is the process of adapting a product to meet the language, cultural, and other requirements of a specific target environment or market so that users can employ their own languages and conventions when using the product. Translation of the user interface, system messages, and documentation is part of localization.

Internationalization is the process of designing a software application without making assumptions that are based on a single language or locale. One goal of internationalization is to ensure that international conventions, including rules for sorting strings and for formatting dates, times, numbers, and currencies, are supported. Another goal is to design the product to have a consistent look, feel, and functionality across different language editions.

Although the application logic might support cultural conventions (for example, the monetary and numeric formats of a particular region), only a localized version of the software presents user interfaces and system messages in the local language.

SAS NLS features are available for localizing and internationalizing your SAS applications.

CHAPTER 2

Locale for NLS

Overview of Locale Concepts for NLS

A *locale* reflects the language, local conventions such as data formatting, and culture for a geographical region. Local conventions may include specific formatting rules for dates, times, and numbers and a currency symbol for the country or region. Collating sequence, paper size, postal addresses, and telephone numbers can also be included in locale.

Dates have many representations, depending on the conventions that are accepted in a culture. The month may be represented as a number or as a name. The name may be fully spelled or abbreviated. The order of the month, day, and year may differ according to locale.

For example, “the third day of October in the year 2002” would be displayed in a different way for each of these locales:

Bulgaria	2002–X-3
Canada	02–10–03
Germany	3.10.2002
Italy	3/10/02
United States	10/03/02

Time can be represented in one English-speaking country or region by using the 12-hour notation, while other English speakers expect time values to be formatted using the 24-hour notation.

Language is part of a locale, but is not unique to any one locale. For example, Portuguese is spoken in Brazil as well as in Portugal, but the cultures are different. In Brazil and in Portugal, there are similarities in the formatting of data. Numbers are formatted using a comma (,) to separate integers from fractional values and a dot (.) to separate groups of digits to the left of the radix character. However, there are important differences, such as the currency symbols that are used in the two different locales. Portugal uses the Euro and requires the Euro symbol (€), while Brazil uses the Real which is represented by the two-character currency symbol R$.

Additionally, a country may have more than one official language. Canada has two official languages: English and French; two values can be specified for the LOCALE= system option: English_Canada and French_Canada.

Numbers, including currency, can have different representations. For example, the decimal separator, or radix character, is a dot (.) in some regions and a comma (,) in others, while the thousands separator can be a dot, comma, or even a space. Monetary conventions likewise vary between locales; for example, a dollar sign or a yen sign might be attached to a monetary value.

Paper size and measurement are also locale considerations. Standard paper sizes include letter (8-1/2-by-11-inch paper) and A4 (210-by-297-millimeter paper). The letter paper size is mainly used by some English-speaking countries; A4 is used by most other locales. While most locales use centimeters, some locales use inches.

Specifying a Locale

How Locale Is Specified at SAS Invocation

You can use the LOCALE= system option to specify the locale of the SAS session at SAS invocation. LOCALE= also implicitly sets the following SAS system options:

- DATESTYLE=
- DFLANG=
- ENCODING=
- PAPERSIZE=
- TRANTAB=

Windows example:

```
sas9 -locale English_UnitedStates
```

Note: Locale can also be specified using POSIX naming standards. For example, en_US is the POSIX equivalent for the SAS value English_UnitedStates. △

Default values for the LOCALE= option are the same under each operating environment. For details, see Chapter 21, "Values for the LOCALE= System Option," on page 397.

The English_UnitedStates value for LOCALE= causes the following options to be implicitly set to the specified default values SAS invocation:

- DATESTYLE=MDY
- DFLANG=English
- ENCODING=wlatin1
- PAPERSIZE=Letter
- TRANTAB=(lat1lat1, lat1lat1,wlt1_ucs,wlt1_lcs,wlt1_ccl,,,)

At invocation, an explicitly set system option will override any implicitly set option. Windows example:

```
options papersize=A4;
```

At invocation, the explicit setting PAPERSIZE=A4 will override an implicit setting of the PAPERSIZE= option via the LOCALE= option. For details, see "DATESTYLE= System Option" on page 349.

How Locale Is Specified During a SAS Session

You can use the LOCALE= system option to specify the locale of the SAS session during the SAS session. However, only the values for these system options will change implicitly to reflect the changed value of LOCALE=:

- DATESTYLE=
- DFLANG=
- PAPERSIZE=

The values for these system options will not change implicitly to reflect the changed value of LOCALE=:

- ENCODING=
- TRANTAB=

Note: ENCODING= cannot be reset during a SAS session. It can be set only at invocation. △

Windows example:

```
options locale=Italian_Ialy;
```

The Italian_Italy value that is assigned to the LOCALE= option causes the following options to be implicitly reset during the SAS session to reflect the changed value of the LOCALE= system option:

- DATESTYLE=DMY
- DFLANG=Italian
- PAPERSIZE=A4

The values for the ENCODING= and TRANTAB= options will not be reset; their former values will be retained.

For details about these system options, see "System Options for NLS by Category" on page 347.

Interaction between the LOCALE= and ENCODING= System Options

Most users will implicitly set encoding by using the LOCALE= system option.

Here is how LOCALE= and ENCODING= interact:

- Setting the LOCALE= option implicitly sets the value for the ENCODING= option only at SAS invocation.

  Note: The LOCALE= setting can be changed during a SAS session, but ENCODING= cannot be changed during a SAS session. If LOCALE= is changed during a session, ENCODING= is not affected. △

- Setting the LOCALE= option implicitly assigns a default value to each of the following options, unless an explicit value is set for a specific option:
  - DATESTYLE=
  - DFLANG=
  - PAPERSIZE=
  - ENCODING=
  - TRANTAB=

Note: Values for ENCODING= and TRANTAB= can be reset only at SAS invocation. △

- If LOCALE= and ENCODING= are both set, ENCODING= will override and set the session encoding.
- If DBCS (which specifies that SAS process DBCS encodings) is set, the following options to identify locale and session encoding are also implicitly set:
  - DBCSLANG=
  - DBCSTYPE=

  The DBCS option settings would override LOCALE=.

Example 1:
When the Spanish_Spain locale is specified under windows, the implicit default encoding value is Windows Latin1 (wlatin1).

```
sas9 -locale spanish_spain
```

Example 2:
The explicit encoding value of Pcoem850 overrides the implicit default encoding value of Windows Latin1 (wlatin1) for the Spanish_Spain locale.

```
sas9 -locale spanish_spain -encoding pcoem850;
```

For details about these system options, see “System Options for NLS by Category” on page 347.

CHAPTER

3

Encoding for NLS

Overview of Encoding for NLS

An encoding maps each character in a character set to a unique numeric representation, which results in a table of all code points. This table is referred to as a *code page*, which is an ordered set of characters in which a numeric index (code point value) is associated with each character. The position of a character on the code page determines its two-digit hexadecimal number.

For example, the following is the code page for the Windows Latin1 encoding. In the following example, the row determines the first digit and the column determines the second digit. The numeric representation for the uppercase A is the hexadecimal number 41, and the numeric representation for the equal sign (=) is the hexadecimal number 3D.

Figure 3.1 Windows Latin1 Code Page

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

A *character set* is the set of characters and symbols that are used by a language or group of languages. A character set includes national characters (which are characters specific to a particular nation or group of nations), special characters (such as punctuation marks), the unaccented Latin characters A-Z, the digits 0-9, and control characters that are needed by the computer.

An *encoding method* is a set of rules that assign the numeric representations to the set of characters. These rules govern the size of the encoding (number of bits used to store the numeric representation of the character) and the ranges in the code page where characters appear. The encoding methods result from the adherence to standards that have been developed in the computing industry. An encoding method is often specific to the computer hardware vendor.

An *encoding* results from applying an encoding method to a character set.

An individual character can occupy a different position in a code page, depending on the code page used. For example, the German uppercase letter Ä:

- □ is represented as the hexadecimal number C4 in the Windows Latin1 code page (1252)
- □ is represented as the hexadecimal number 4A in the German EBCDIC code page (273).

In the following code page example, German is the character set and EBCDIC is the encoding method.

In the following example, the column determines the first digit and the row determines the second digit.

Figure 3.2 German EBCDIC Code Page

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	ü LU170000	Ö LO180000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	ß LS610000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	[SM060000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	@ SM050000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	~ SD190000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	Ä LA180000	Ü LU180000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	¹ ND011000	² ND021000	³ ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	¦ SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	§ SM240000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	| SM650000	} SM140000	\ SM070000	] SM080000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

Each SAS session is set to a default encoding, which can be specified by using various SAS language elements.

Difference between Encoding and Transcoding

Encoding establishes the default working environment for your SAS session. For example, the Windows Latin1 encoding is the default encoding for a SAS session under Windows in a Western European locale. As an example, the Windows Latin1 code point for the uppercase letter Ä is C4 hexadecimal.

Note: The default encoding varies according to the operating environment and the locale. △

However, if you are working in an international environment (for example, you access SAS data that is encoded in German EBCDIC), the German EBCDIC code point for the uppercase letter Ä is 4A hexadecimal. In order for a version of SAS that normally uses Windows Latin1 to properly interpret a data set that is encoded in German EBCDIC, the data must be transcoded. *Transcoding* is the process of converting data from one encoding to another. When SAS transcodes the Windows Latin1 uppercase letter Ä to the German EBCDIC uppercase letter Ä, the hexadecimal representation for the character is converted from the value C4 to a 4A. For conceptual information, see Chapter 4, "Transcoding for NLS," on page 21.

Character Sets for Encoding in NLS

Encodings are available to address the requirements of the character set (few languages use the same 26 characters, A through Z as English). All languages are represented using either of the following classes of character sets:

SBCS (Single-Byte Character Set)
: represents each character in a single (one) byte. A single-byte character set can be either 7 bits (providing up to 128 characters) or 8 bits (providing up to 256 characters). An example of an 8-bit SBCS is the ISO 8859-5 (Cyrillic) character set (represents the Russian characters).

  For details, see "Locale Values and Encoding Values for SBCS, DBCS, and Unicode" on page 400.

DBCS (Double-Byte Character Set)
: refers to the East Asian character sets (Japanese, Korean, Simplified Chinese, and Traditional Chinese), which require a mixed-width encoding because most characters consist of more than one byte. Although the term DBCS (Double-Byte Character Set) is more commonly used than MBCS (Multi-Byte Character Set), MBCS is more accurate. Some, but not all characters in an East Asian character set do require more than one byte.

  For details, see Chapter 22, "SAS System Options for Processing DBCS Data," on page 405.

MBCS (Multi-Byte Character Set)
: is used as a synonym for DBCS.

Common Encoding Methods

The encoding methods result from standards developed by various computer hardware manufacturers and standards organizations. For more information, see "Standards Organizations for NLS Encodings" on page 14. The common encoding methods are listed here:

ASCII (American Standard Code for Information Interchange)
is a 7-bit encoding for the United States that provides 128 character combinations. The encoding contains characters for uppercase and lowercase English, American English punctuation, base 10 numbers, and a few control characters. This set of 128 characters is common to most other encodings. ASCII is used by personal computers.

EBCDIC (Extended Binary Coded Decimal Interchange Code) family
is an 8-bit encoding that provides 256 character combinations. There are multiple EBCDIC-based encodings. EBCDIC is used on IBM mainframes and most IBM mid-range computers. EBCDIC follows ISO 646 conventions to facilitate translations between EBCDIC encodings and 7-bit (and 8-bit) ASCII-based encodings. The 95 EBCDIC graphical characters include 82 invariant characters (including a black space), which occupy the same code positions across most EBCDIC single-byte code pages, and also includes 13 variant graphic characters, which occupy varying code positions across most EBCDIC single-byte code pages. For details about variant characters, see "Code Point Discrepancies among EBCDIC Encodings" on page 15.

ISO (International Organization for Standardization) 646 family
is a 7-bit encoding that is an international standard and provides 128 character combinations. The ISO 646 family of encodings is similar to ASCII except that it has 12 code points for national variants. The 12 national variants represent specific characters that are needed for a particular language.

ISO 8859 family and Windows family
is an 8-bit extension of ASCII that supports all of the ASCII code points and adds 12 more, providing 256 character combinations. Latin1, which is officially named ISO-8859-1, is the most frequently used member of the ISO 8859 family of encodings. In addition to the ASCII characters, Latin1 contains accented characters, other letters needed for languages of Western Europe, and some special characters. HTTP and HTML protocols are based on ISO Latin1.

Unicode
provides up to 65,536 character combinations. Unicode can accommodate basically all of the world's languages.

There are three Unicode encoding forms:

UTF-8
is an MBCS encoding that contains the Latin-script languages, Greek, Cyrillic, Arabic, and Hebrew, and East Asian languages such as Japanese, Chinese and Korean. The characters in UTF-8 are of varying width, from one to four bytes. UTF-8 maintains ASCII compatibility by preserving the ASCII characters in code positions 1 through 128.

UTF-16
is a 16-bit form that contains all of the most common characters in all modern writing systems. Most of the characters are uniformly represented with two bytes, although there is extended space, called surrogate space, for additional characters that require four bytes.

UTF-32
is a 32-bit form whose characters each occupy four bytes.

Other encodings
The ISO 8859 family has other members that are designed for other languages. The following table describes the other encodings that are approved by ISO.

Table 3.1 Other Encodings Approved by ISO

ISO Standard	Name of Encoding	Description
ISO 8859-1	Latin 1	US and West European
ISO 8859-2	Latin 2	Central and East European
ISO 8859-3	Latin 3	South European, Maltese and Esperanto
ISO 8859-4	Baltic	North European
ISO 8859-5	Cyrillic	Slavic languages
ISO 8859-6	Arabic	Arabic
ISO 8859-7	Greek	Modern Greek
ISO 8859-8	Hebrew	Hebrew and Yiddish
ISO 8859-9	Turkish	Turkish
ISO 8859-10	Latin 6	Nordic (Inuit, Sámi, Icelandic)
ISO 8859-11	Latin/Thai	Thai
ISO 8859-12		undefined
ISO 8859-13	Latin 7	Baltic Rim
ISO 8859-14	Latin 8	Celtic
ISO 8859-15	Latin 9	West European and Albanian

Additionally, a number of encoding standards have been developed for East Asian languages, some of which are listed in the following table.

Table 3.2 Some East Asian Language Encodings Approved by ISO

Standard	Name of Encoding	Description
GB 2312-80	Simplified Chinese	People's Republic of China
CNS 11643	Traditional Chinese	Taiwan
Big-5	Traditional Chinese	Taiwan
KS C 5601	Korean National Standard	Korea
JIS	Japan Industry Standard	Japan
Shift-JIS	Japan Industry Standard multibyte encoding	Japan

There are other encodings in the standards for EBCDIC and Windows that support different languages and locales.

Standards Organizations for NLS Encodings

Encodings that are supported by SAS are defined by the following standards organizations:

International Organization for Standardization (ISO) promotes the development of standardization and related activities to facilitate the free flow of goods and services between nations and to advocate for the exchange of intellectual, scientific, and technological information. ISO also establishes standards for encodings.

American National Standards Institute (ANSI)
coordinates voluntary standards and conformity to those standards in the United States. ANSI works with ISO to establish global standards.

Unicode Consortium
that develops and promotes the Unicode standard, which provides a unique number for every character.

Code Point Discrepancies among EBCDIC Encodings

Selected characters do not occupy the same code point locations in code maps for all EBCDIC encoding methods. For example, the following characters occupy different code point locations in the respective EBCDIC code maps for U.S. English and German.

Table 3.3 EBCDIC Code Point Discrepancies for Selected Languages

EBCDIC Code Points	U.S. English	Finnish	Spanish	German
4A	¢	§	[	Ä
4F	|	!	|	!
5A	!	¤	]	Ü
5B	$	Å	$	$
5F	¬	^	¬	^
6A	¦	ö	ñ	ö
79	‘	é	‘	‘
7B	#	Ä	Ñ	#
7C	@	Ö	@	§
A1	~	ü	¨	ß
C0	{	ä	{	ä
D0	}	å	}	ü
E0	\	É	\	Ö

These characters are known as *variant characters*. For example, if a German mainframe user entered an ä, which occupies code point C0, an American compiler would interpret code point C0 as a {.

Especially important are characters that are commonly used in programming languages, for example, { and $.

Collation Sequence

A major effect of the session encoding is the collation sequence (or *sorting sequence*) that is used to perform alphanumeric sorting operations (such as the SORT procedure). Sort order corresponds directly to the arrangement of the code points in the code page. The two single-byte character-encoding systems that are most widely used in data processing are ASCII and EBCDIC. OpenVMS, UNIX, and Windows operating environments use ASCII, and IBM mainframe computers use EBCDIC.

The collation sequence that you use corresponds to your session encoding, by default. However, when using the SORT procedure, you can override your session's default encoding collation sequence and specify an explicit collation sequence.

The following SAS language elements support a collation sequence:

- SORT statement in the SORT procedure (see "Collation Sequence Option" on page 370)
- SORTSEQ= data set option (see "SORTSEQ= Data Set Option" on page 42)
- SORTSEQ= system option (see "SORTSEQ= Data Set Option" on page 42)

You can also select sort sequences by using the VIEWTABLE Window. In the VIEWTABLE Window, you can select from the sort sequences that are listed under the Advanced tab of the Sort dialog box. For details about viewing and editing SAS data sets, see *SAS Language Reference: Concepts*.

The collation sequence that you use corresponds to your session encoding, by default. However, when using the SORT procedure, you can override your session's default encoding collation sequence and specify an explicit collation sequence. Standard collation sequences include:

- ASCII
- EBCDIC
- Danish
- Finnish
- Italian
- Norwegian
- Polish
- Spanish
- Swedish

By viewing the contents of a sort translation table, you can determine the collation sequence because the sort trantabs contain the weight that is assigned to each character.

You can use the following statement to view the trantab contents:

```
proc trantab table=table-name;
   list;
run;
```

The contents of the collation sequence are displayed in the SAS log.

Determining the Encoding of a SAS Session and a Data Set

Encoding of a SAS Session

To determine your current SAS session encoding, which is the value assigned to the ENCODING= system option, you can use the OPTIONS procedure or the OPTIONS window. For example, the following PROC OPTIONS statement displays the session encoding value:

```
proc options option=encoding;
run;
```

The SAS log displays the following information:

```
ENCODING=WLATIN1 Specifies default encoding for processing external data.
```

You can display the encoding of any SAS 9 data set by using the CONTENTS procedure or the Properties window in the SAS windowing environment.

An example follows of output that is reported from the CONTENT procedure in the SAS log. The encoding is Western latin1.

Output 3.1 Encoding Reported in the SAS Log

```
The SAS System       10:15 Friday, June 06, 2003   1

                                   The CONTENTS Procedure

     Data Set Name     WORK.GRADES                         Observations          1
     Member Type       DATA                                Variables             4
     Engine            V9                                  Indexes               0
     Created           11:03 Friday, June 06 2003     Observation Length    32
     Last Modified     11:03 Friday, June 06, 2003     Deleted Observations  0
     Protection                                            Compressed           NO
     Data Set Type                                         Sorted               NO
     Label
     Data Representation  HP_UX_64, RS_6000_AIX_64, SOLARIS_64, HP_IA64
     Encoding          latin1  Western (ISO)


                             Engine/Host Dependent Information

       Data Set Page Size          4096
       Number of Data Set Pages    1
       First Data Page             1
       Max Obs per Page            126
       Obs in First Data Page      1
       Number of Data Set Repairs  0
       File Name                   C:\TEMP\SAS Temporary Files\_TD228\grades.sas7bdat
       Release Created             9.0000M0
       Host Created                WIN_NT


                        Alphabetic List of Variables and Attributes

                               #    Variable    Type    Len

                               4    final       Num       8
                               1    student     Char      8
                               2    test1       Num       8
                               3    test2       Num       8
```

Encoding of a SAS Data Set

To determine the encoding of a specific SAS data set, follow these steps:

1 Locate the data set using SAS Explorer.

2 Right-click the data set.

3 Select Properties from the menu.
4 Click the Details tab.
5 The encoding of the data set is listed, along with other information.

Default SAS Session Encoding

The ENCODING= option is used to specify the SAS session encoding, which establishes the environment to process SAS syntax and SAS data sets, and to read and write external files. If neither the LOCALE= nor ENCODING= options is set, a default value is set.

Table 3.4 Default SAS Session Encoding Values

Operating Environment	Default ENCODING= Value	Description
OpenVMS Alpha	Latin1	Western (ISO)
z/OS	OPEN_ED_1047	OpenEdition EBCDIC cp1047-Latin1
UNIX	Latin1	Western (ISO)
Windows	WLatin1	Western (Windows)

For a complete list of supported encoding values for a SAS session, see “Locale Values and Encoding Values for SBCS, DBCS, and Unicode” on page 400.

Setting the Encoding of a SAS Session

You can set the session encoding by using the ENCODING= system option or the LOCALE= system option.

Note: Valid values only for ENCODING= are dependent on the operating environment used. △

- If ENCODING= is specified, the TRANTAB= option is implicitly set.
- If both LOCALE= and ENCODING= are set, ENCODING= will set the session encoding.
- If DBCS (which specifies that SAS process DBCS encodings) is set, the following options to identify locale and session encoding are also implicitly set:
  - DBCSLANG=
  - DBCSTYPE=.

  These options are used for East Asian languages or for English with DBCS extensions.

The following example shows that for the Spanish_Spain locale encoding is explicitly set by default.

```
sas9 -locale Spanish_Spain
```

The wlatin1 encoding is the default encoding for the Spanish_Spain locale.

The following example shows that wlatin2 encoding is set explicitly at SAS invocation.

```
sas9 -encoding wlatin2
```

Note: Changing the encoding for a SAS session does not affect SAS keywords, or SAS log output, which remain in English. △

Encoding Behavior in a SAS Session

Encoding Support for Data Sets by SAS Release

For Base SAS files, there are three categories of encoding support, which is based on the version of SAS that created the file:

- □ Data sets that are created in SAS 9 automatically have an encoding attribute, which is specified in the descriptor portion of the file.
- □ Data sets that are created in SAS 7 and SAS 8 not have an encoding value that is specified in the file. It is assumed that SAS 7 and SAS 8 data sets were created in the SAS session encoding of the operating environment. However, the descriptor portion of the file does support an encoding value. When you replace or update a SAS 7 or SAS 8 file in a SAS 9 session, SAS specifies the current session encoding in the descriptor portion of the file, by default.
- □ Data sets created in SAS 6 do not have an encoding value that is associated with the file and cannot have an encoding value specified in the file.

z/OS: Ensuring Compatibility with Previous SAS Releases

Setting the NLSCOMPATMODE system option ensures compatibility with previous releases of SAS.

Note: NLSCOMPATMODE is supported under the z/OS operating environment only. △

Programs that were run in previous releases of SAS will continue to work when NLSCOMPATMODE is specified.

The NONLSCOMPATMODE system option specifies that data is to be processed in the encoding that is set by the ENCODING= option or the LOCALE= option, including reading and writing external data and processing SAS syntax and user data.

Some existing programs that ran in previous releases of SAS will no longer run when NONLSCOMPATMODE is in effect. If you have made character substitutions in SAS syntax statements, you must modify your programs to use national characters. For example, a Finnish customer who has substituted the Å character for the $ character in existing SAS syntax will have to update the program to use the $ in the Finnish environment.

For details, see “NLSCOMPATMODE System Option: z/OS” on page 360.

Output Processing

When you create a data set in SAS 9, encoding is determined as follows:

- □ If a new output file is created, the data is written to the file using the current session encoding.

- If a new output file is created using the OUTREP= option, which specifies a data representation that is different from the current session, the data is written to the file using the default session encoding for the operating system that is specified by the OUTREP= value.
- If a new output file replaces an existing file, the new file inherits the encoding of the existing file. For output processing that replaces an existing file that is from another operating environment or if the existing file has no encoding that is specified in it, then the current session encoding is used.

Input Processing

For input (read) processing in SAS 9, encoding behavior is as follows:

- If the session encoding and the encoding that is specified in the file are incompatible, the data is transcoded to the session encoding. For example, if the current session encoding is ASCII and the encoding that is specified in the file is EBCDIC, SAS transcodes the data from EBCDIC to ASCII.
- If a file does not have an encoding specified in it, SAS transcodes the data only if the file's data representation is different from the current session.

Reading and Writing External Files

SAS reads and writes external files using the current session encoding. SAS assumes that the external file has the same encoding as the session encoding. For example, if you are creating a new SAS data set by reading an external file, SAS assumes that the encoding of the external file and the current session are the same. If the encodings are not the same, the external data could be written incorrectly to the new SAS data set. For details about the syntax for the SAS statements that perform input and output processing, see "SAS Options That Transcode SAS Data" on page 23.

CHAPTER

4

Transcoding for NLS

Overview to Transcoding

Transcoding is the process of converting a SAS file (its data) from one encoding to another encoding. Transcoding is necessary when the session encoding and the file encoding are different. Transcoding is often necessary when you move data between operating environments that use different locales.

For example, consider a file that was created under a UNIX operating environment that uses the Latin1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. When the file is processed on the IBM mainframe, the data is remapped from the Latin1 encoding to the German EBCDIC encoding. If the data contains an uppercase letter Ä, the hexadecimal number is converted from C4 to 4A.

Transcoding does not translate between languages; transcoding remaps characters.

In order to dynamically transcode data between operating environments that use different encodings, an explicit encoding value must be specified. For details, see Chapter 23, "Encoding Values in SAS Language Elements," on page 407.

Common Reasons for Transcoding

Some situations where data might commonly be transcoded are:

- when you share data between two different SAS sessions that are running in different locales or in different operating environments,
- when you perform text-string operations, such as converting to uppercase or lowercase,
- when you display or print characters from another language,

- when you copy and paste data between SAS sessions running in different locales

.

Transcoding and Translation Tables

Specifying LOCALE= or ENCODING= indirectly sets the appropriate trantab values in the TRANTAB= option. Trantabs are used for transcoding one SBCS encoding to another and back again. For example, there is a specific trantab that maps Windows Latin2 to ISO Latin2.

The following figure shows a translation table. The area of a trantab for mapping from Windows Latin 2 (wlt2) to ISO Latin 2 (lat2) is named "table 1," and the area for mapping characters from ISO Latin 2 to Windows Latin 2 is named "table 2."

Figure 4.1 SAS Windows Latin 2 to ISO Latin 2 Translation Table

```
WLT2LAT2 table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '606162636465666768696A6B6C6D6E6F'x
     70 '707172737475767778797A7B7C7D7E7F'x
     80 '808182838485868788898A98AA6ABAEAC'x
     90 '908B8C8D8E8F91929893B994B6BBBEBC'x
     A0 'A0B7A2A3A4A195A7A896AA9799AD9AAF'x
     B0 'B09BB2B3B49C9D9EB8B1BA9FA5BDB5BF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x


WLT2LAT2 table 2:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '606162636465666768696A6B6C6D6E6F'x
     70 '707172737475767778797A7B7C7D7E7F'x
     80 '808182838485868788898B9192939495'x
     90 '909697999BA6A9AB98ACAEB1B5B6B7BB'x
     A0 'A0A5A2A3A4BC8CA7A88AAA8D8FAD8EAF'x
     B0 'B0B9B2B3B4BE9CA1B89ABA9D9FBD9EBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

The LOCALE= or ENCODING= system option and other encoding options (to statements, commands, or procedures) eliminates the need to directly create or manage translation tables.

CAUTION:

Do not change a translation table unless you are familiar with its purpose. Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do not change the specifications without proper technical advice. △

The TRANTAB= option specifies the translation table to be used in the SAS session. For details, see "TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362. The TRANTAB procedure is used to create, edit, and display customized translation tables. For details, see Chapter 15, "The TRANTAB Procedure," on page 319.

SAS Options That Transcode SAS Data

The following SAS options for various language elements enable you to transcode, or to override the default encoding behavior. These elements enable you to specify a different encoding for a SAS file or a SAS application or to suppress transcoding.

Table 4.1 SAS Options That Transcode SAS Data

Option	Where Used
CHARSET=	ODS MARKUP statement
CORRECTENCODING=	MODIFY statement of the DATASETS procedure
ENCODING=	%INCLUDE, FILE, FILENAME, INFILE, ODS statements; FILE and INCLUDE commands
ENCODING=	in a DATA step
INENCODING=	LIBNAME statement
ODSCHARSET=	LIBNAME statement for XML
ODSTRANTAB=	LIBNAME statement for XML
OUTENCODING=	LIBNAME statement
XMLENCODING=	LIBNAME statement for XML

For complete details about these language elements, see Chapter 18, "Overview to NLS Options Used in Commands, Statements, and Procedures," on page 367. For a list of supported encoding values to use for these options, see "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 407.

Transcoding between Operating Environments

Transcoding occurs automatically when SAS files are moved or accessed across operating environments. Common SAS transcoding activities include:

CPORT and CIMPORT procedures
: To create a transport file, SAS automatically uses translation tables to transcode one encoding to another and back again. First, the data is converted from the source encoding to transport format, then the data is converted from the transport format to the target encoding. For details, see *Base SAS Procedures Guide*.

CEDA (cross environment data access) feature of SAS
: when you process a SAS data set that has an encoding that is different from the current session encoding, SAS automatically uses CEDA software to transcode data. (CEDA also converts a SAS file to the correct data representation when you move a file between operating environments.) For details, see *SAS Language Reference: Concepts* and *SAS Language Reference: Dictionary*.

SAS/CONNECT Data Transfer Services (UPLOAD and DOWNLOAD procedures)
: For details, see *SAS/CONNECT User's Guide*.

SAS/CONNECT Compute Services (RSUBMIT statement)
: identifies a block of statements that a client session submits to server session for processing. For details, see *SAS/CONNECT User's Guide*.

SAS/CONNECT and SAS/SHARE Remote Library Services (LIBNAME)
: References a library on a remote machine for client access. For details, see *SAS/CONNECT User's Guide* and *SAS/SHARE User's Guide*.

Transcoding Considerations

Although transcoding usually occurs with no problems, there are situations that can affect your data and produce unsatisfactory results. For example:

- Encodings can conflict with another. That is, two encodings can use different code points for the same character, or use the same code points for two different characters.
- Characters in one encoding might not be present in another encoding. For example, a specific encoding might not possess a character for the dollar sign ($). Transcoding the data to an encoding that does not support the dollar sign would result in the character not printing or displaying.
- The number of bytes for a character in one encoding can be different from the number of bytes for the same character in another encoding; for example, transcoding from a DBCS to an SBCS. Therefore, transcoding can result in character value truncation.
- If an error occurs during transcoding such that the data cannot be transcoded back to its original encoding, data can be lost. That is, if you open a data set for update processing, the observation might not be updated. However, if you open the data set for input (read) processing and no output data set is open, SAS issues a warning that can be printed. Processing proceeds and allows a PRINT procedure or other read operation to show the data that does not transcode.
- CEDA has some processing limitations. For example, CEDA does not support update processing.
- Incorrect encoding can be stamped on a SAS 7 or SAS 8 data set if it is copied or replaced in a SAS 9 session with a different session encoding from the data. If a character variable contains binary data, transcoding might corrupt the data.

Compatible and Incompatible Encodings

Overview to Compatible and Incompatible Encodings

ASCII is the foundation for most encodings, and is used by most personal computers, minicomputers, and workstations. However, the IBM mainframe uses an EBCDIC encoding. Therefore, ASCII and EBCDIC machines and data are incompatible. Transcoding is necessary if some or all characters in one encoding are different from the characters in the other encoding.

However, to avoid transcoding, you can create a data set and specify an encoding value that SAS will not transcode. For example, if you use the following values in either the ENCODING= data set option, or the INENCODING=, or the OUTENCODING= option in the LIBNAME statement, transcoding is not performed:

- ANY specifies that no transcoding is desired, even between EBCDIC and ASCII encodings.

  Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant. △

- ASCIIANY enables you to create a data set that is compatible with all ASCII-based encodings.
- EBCDICANY enables you to create a data set that is compatible with all EBCDIC-based encodings.

You might want to create a SAS data set that contains mixed encodings; for example, both Latin1 and Latin2. You do not want the data transcoded for either input or output processing. By default, data is transcoded to the current session encoding.

Data must be transcoded when the SAS file and the SAS session use *incompatible encodings*; for example, ASCII and EBCDIC.

In some cases, transcoding is not required because the SAS file and the SAS session have *compatible encodings*.

For a list of the encodings, by operating environment, see Chapter 24, "Encoding Values for a SAS Session," on page 413.

Line-feed Characters and Transferring Data between EBCDIC and ASCII

Software that runs under ASCII operating environments requires the end of the line be specified by the line-feed character. When data is transferred from z/OS to a machine that supports ASCII encodings, formatting problems can occur, particularly in HTML output, because the EBCDIC new-line character is not recognized. SAS supports two sets of EBCDIC-based encodings for z/OS:

- The encodings that have EBCDIC in their names use the traditional mapping of EBCDIC line-feed to ASCII line-feed character, which can cause data to appear as one stream.
- The encodings that have Open Edition in their names use the line-feed character as the end-of-line character. When the data is transferred to an operating environment that uses ASCII, the EBCDIC new-line character maps to an ASCII line-feed character. This mapping enables ASCII applications to interpret the end-of-line correctly, resulting in better formatting.

For a list of the encodings, by operating environment, see Chapter 24, "Encoding Values for a SAS Session," on page 413.

EBCDIC and OpenEdition Encodings Are Compatible

EBCDIC and OpenEdition are compatible encodings.

Encodings that contain EBCDIC in their names use the traditional mapping of EBCDIC line-feed (0x25) and new-line (0x15) characters.

Encodings that contain OPEN_ED in their names and OpenEdition in their descriptions switch the mapping of the new-line and line-feed characters. That is, they use the line-feed character as the end-of-line character.

If the two encodings use the same code page number but one is EBCDIC and the other is Open Edition, no transcoding is necessary.

Example:

If the data is encoded in EBCDIC1143 and the SAS session is encoded in OPEN_ED-1143, no transcoding is necessary because they use the same 1143 code page.

In order to transfer data between ASCII and EBCDIC, you can specify Open Edition encodings from the list of compatible encodings.

Note: Open Edition encodings are used by default in NONLSCOMPATMODE. △

Some East Asian MBCS Encodings Are Compatible

Some East Asian double-byte (DBCS) are compatible encodings. Each line in the list contains compatible encodings:

- SHIFT-JIS, MS-932, IBM-942, MACOS-1
- MS-949, MACOS-3, EUC-KR
- EUC-CN, MS-936, MACOS-25, DEC-CN
- EUC-TW, DEC-TW
- MS-950, MACOS-2, BIG5

If the SAS session is encoded in one of the encodings in the group and the data set is encoded in another encoding, but in the same group, then no transcoding occurs.

Example:

If the session encoding is SHIFT-JIS and the data set encoding is IBM-942, then no transcoding occurs.

Preventing Transcoding

Some encoding values enable you to create a data set that SAS does not transcode. You might not want to transcode data for input or output processing but rather you might want to create a SAS library that contains data in mixed encodings; for example, both Latin1 and Latin2.

For example, you can avoid transcoding if you use the following values in either the ENCODING= data set option or the INENCODING= or OUTENCODING= options in the LIBNAME statement:

- ANY specifies that no transcoding is desired, even between EBCDIC and ASCII encodings.

  Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant. △
- ASCIIANY specifies that no transcoding is required between any ASCII-based encodings.

- EBCDICANY specifies that no transcoding is required between any EBCDIC-based encodings.

For details, see “ENCODING= Data Set Option” on page 39 and “INENCODING= and OUTENCODING= Options” on page 381.

CHAPTER 5

Double-Byte Character Sets (DBCS)

Overview to Double-Byte Character Sets (DBCS)

Because East Asian languages have thousands of characters, double (two) bytes of information are needed to represent each character.

Each East Asian language usually has more than one DBCS encoding system, due to nonstandardization among computer manufacturers. SAS processes the DBCS encoding information that is unique to each manufacturer for the major East Asian languages.

With the proper software extensions, you can use SAS for the following functions:

- display any of the major East Asian languages in the DBCS version of the SAS System
- import data from East Asian language computers and move the data from one application or operating environment to another (which may require SAS ACCESS or other SAS products)
- convert standard East Asian date and time notation to SAS date values, SAS time values, and SAS datetime values
- create data sets and various types of output (such as reports and graphs) that contain East Asian language characters.

East Asian Languages

East Asian languages include:

- Chinese, which is written in Simplified Chinese script, and is used in the People's Republic of China and Singapore
- Chinese, which is written in Traditional Chinese script, and is used in Hong Kong SAR, Macau SAR, and Taiwan
- Japanese
- Korean.

Specifying DBCS

To specify DBCS, use the following SAS system options:

DBCS recognizes DBCS characters

DBCSLANG= specifies the language

DBCSTYPE= specifies the DBCS encoding method type

Example of a SAS configuration file for Windows:

```
/*basic DBCS options */

-dbms              /*Recognizes DBCS*/
-dbcstype PCMS     /*Specifies the PCMS encoding method*/

-dbcslang JAPANESE; /*specifies the Japanese language */
```

DBCSTYPE= and DBCSLANG= were introduced in Version 6.12. As an alternative, setting ENCODING= implicitly sets the DBCSTYPE= and DBCSLANG= options. For details, see "Locale Values and Encoding Values for SBCS, DBCS, and Unicode" on page 400.

Requirements for Displaying DBCS Character Sets

In order to display data sets that contain DBCS characters, you must have the following resources:

- system support for multiple code pages
- DBCS fonts that correspond to the language that you intend to use.

If you need to create a user-defined character for use with SAS software, your computer must support DBCS. These computers have a limited availability in the U.S. and Europe. These East Asian language computer systems use various methods of creating the characters. In one popular method, the user types the phonetic pronunciation of the character, often using Latin characters. The computer presents a menu of characters whose sounds are similar to the phonetic pronunciation and prompts the user to select one of them.

When You Can Use DBCS Features

After you have set up your SAS session to recognize a specific DBCS language and operating environment, you can work with your specified language in these general areas:

- the DATA step and batch-oriented procedures
- windowing and interactive capabilities
- cross-system connectivity and compatibility
- access to databases
- graphics.

In a DATA step and in batch-oriented procedures, you can use DBCS wherever a text string within quotation marks is allowed. Variable values, variable labels, and data set labels can all be in DBCS. DBCS can also be used as input data and with range and label specifications in the FORMAT procedure. In WHERE expression processing, you can search for embedded DBCS text.

DBCS and SAS on a Mainframe

Another type of DBCS encoding exists on mainframe systems, which combine DBCS support with the 3270-style data stream. Each DBCS character string is surrounded by escape codes called *shift out/shift in*, or SO/SI. These codes originated from the need for the old-style printers to shift out from the EBCDIC character set, to the DBCS character set. The major manufacturers have different encodings for SO/SI; some manufacturers pad DBCS code with one byte of shift code information while others pad the DBCS code with two bytes of shift code information. These differences can cause problems in reading DBCS information about mainframes.

PCs, minicomputers, and workstations do not have SO/SI but have their own types of DBCS encodings that differ from manufacturer to manufacturer. SAS has several formats and informats that can read DBCS on SO/SI systems:

Table 5.1 SAS Formats and Informats That Support DBCS on SO/SI Systems

Keyword	Language Element	Description
$KANJI	informat	Removes SO/SI from Japanese Kanji DBCS
$KANJIX	informat	Adds SO/SI to Japanese Kanji DBCS
$KANJI	format	Adds SO/SI to Japanese Kanji DBCS
$KANJIX	format	Removes SO/SI from Japanese Kanji DBCS

SAS Data Conversion between DBCS Encodings

Normally, DBCS data that is generated on one computer system is incompatible with data generated on another computer system. SAS has features that allow conversion from one DBCS source to another, as shown in the following table.

Language Element	Type	Use	See
KCVT	function	Converts DBCS data from one operating environment to another	"KCVT Function" on page 214
CPORT	procedure	Moves files from one environment to another	*Base SAS Procedures Guide*
CIMPORT	procedure	Imports a transport file created by CPORT	*Base SAS Procedures Guide*

Avoiding Problems with Split DBCS Character Strings

- When working with DBCS characters, review your data to make sure that SAS recognizes the entire character string when data is imported or converted or used in a DATA or a PROC step.
- On mainframe systems that employ shift out/shift in escape codes, DBCS character strings can become truncated during conversion across operating environments.
- There is a possibility that DBCS character strings can be split when working with the PRINT, REPORT, TABULATE, and FREQ procedures. If undesirable splitting occurs, you might have to add spaces on either side of your DBCS character string to force the split to occur in a better place. The SPLIT= option can also be used with PROC REPORT and PROC PRINT to force string splitting in a better location.

Avoiding Character Data Truncation by Using the CVP Engine

When you specify the ENCODING= data set option, the encoding for the output data set might require more space than the original data set. For example, when writing DBCS data in a Windows environment using the UTF8 encoding, each DBCS character may require three bytes. To avoid data truncation, each variable must have a width that is 1.5 times greater than the width of the original data.

When you process a SAS data file that requires transcoding, you can request that the CVP (character variable padding) engine expand character variable lengths so that character data truncation does not occur. (A variable's length is the number of bytes used to store each of the variable's values.)

Character data truncation can occur when the number of bytes for a character in one encoding is different from the number of bytes for the same character in another encoding, such as when a single-byte character set (SBCS) is transcoded to a double-byte character set (DBCS) or to a multi-byte character set (MBCS). A SBCS represents each character in one byte, and a DBCS represents each character in two bytes. An MBCS represents characters in a varying length from one to four bytes. For example, when transcoding from Wlatin2 to a Unicode encoding, such as UTF-8, the variable lengths (in bytes) might not be sufficient to hold the values, and the result is character data truncation.

Using the CVP engine, you specify an expansion amount so that variable lengths are expanded prior to transcoding, then the data is processed. Think of the CVP engine as an intermediate engine that is used to prepare the data for transcoding. After the lengths are increased, then the primary engine, such as the default base engine, is used to do the actual file processing.

The CVP engine is a read-only engine for SAS data files only. You can request character variable expansion (for example with the LIBNAME statement) in either of the following ways:

- explicitly specify the CVP engine and using the default expansion of 1.5 times the variable lengths.
- implicitly specifying the CVP engine with the LIBNAME options CVPBYTES= or CVPMULTIPLIER=. The options specify the expansion amount. In addition, you can use the CVPENGINE= option to specify the primary engine to use for processing the SAS file; the default is the default SAS engine.

For example, the following LIBNAME statement explicitly assigns the CVP engine. Character variable lengths are increased using the default expansion, which multiples the lengths by 1.5. For example, a character variable with a length of 10 will have a new length of 15, and a character variable with a length of 100 will have a new length of 150:

```
libname expand cvp 'SAS data-library';
```

Note: The expansion amount must be large enough to accommodate any expansion; otherwise, truncation will still occur. △

Note: For processing that conditionally selects a subset of observations by using a WHERE expression, using the CVP engine might affect performance. Processing the file without using the CVP engine might be faster than processing the file using the CVP engine. For example, if the data set has indexes, the indexes will not be used in order to optimize the WHERE expression if you use the CVP engine. △

For more information and examples, see the CVP options in the LIBNAME Statement in *SAS Language Reference: Dictionary*.

PART 2

Data Set Options for NLS

CHAPTER 6

Overview to Data Set Options for NLS

Data Set Options for NLS by Category

NLS affects the data set control category of options for selected data set options. The following table provides brief descriptions of the data set options. For more detailed descriptions, see the dictionary entry for each data set option:

Table 6.1 Summary of Data Set Options for NLS

Category	Data Set Options	Description
Data Set Control	"ENCODING= Data Set Option" on page 39	Overrides the encoding to use for reading or writing a SAS data set
	"OUTREP= Data Set Option" on page 41	Specifies the data representation for the output SAS data set
	"SORTSEQ= Data Set Option" on page 42	Specifies a language-specific collation sequence for the SORT procedure to use for the specified SAS data set

CHAPTER 7

Data Set Options for NLS

ENCODING= Data Set Option

Overrides the encoding to use for reading or writing a SAS data set

Valid in: DATA step and PROC steps

Category: Data Set Control

Syntax

ENCODING= ANY | ASCIIANY | EBCDICANY | *encoding-value*

Syntax Description

ANY
specifies that no transcoding occurs.

Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant. △

ASCIIANY
specifies that no transcoding occurs when the mixed encodings are ASCII encodings.

EBCDICANY
specifies that no transcoding occurs when the mixed encodings are EBCDIC encodings.

encoding-value
specifies an encoding value. For details, see "Locale Values and Encoding Values for SBCS, DBCS, and Unicode" on page 400.

Details

The value for ENCODING= indicates that the SAS data set has a different encoding from the current session encoding. When you read data from a data set, SAS transcodes the data from the specified encoding to the session encoding. When you write data to a data set, SAS transcodes the data from the session encoding to the specified encoding.

Input Processing

By default, encoding for input processing is determined as follows:

- If the session encoding and the encoding that is specified in the file are different, SAS transcodes the data to the session encoding.
- If a file has no encoding specified, but the file's data representation is different from the encoding of the current session, then SAS transcodes the data to the current session.

Output Processing

By default, encoding for output processing is determined as follows:

- Data is written to a file using the encoding of the current session, except when a different output representation is specified using the OUTREP= data set option, the OUTENCODING= option in the LIBNAME statement, or the ENCODING= data set option.
- If a new file replaces an existing file, then the new file will inherit the encoding of the existing file.
- If an existing file is replaced by a new file that was created under a different operating environment or that has no encoding specified, the new file will use the encoding of the current session.

Comparisons

- Session encoding is specified using the ENCODING= system option or the LOCALE= system option, with each operating environment having a default encoding.
- You can specify encoding for a SAS data library by using the LIBNAME statement's INENCODING= option (for input files) and the OUTENCODING= option (for output files). If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.

Examples

Example 1: Creating a SAS Data Set with Mixed Encodings and with Transcoding Suppressed By specifying the data set option ENCODING=ANY, you can create a SAS data set that contains mixed encodings, and suppress transcoding for either input or output processing.

In this example, the new data set MYFILES.MIXED contains some data that uses the Latin1 encoding, and some data that uses the Latin2 encoding. When the data set is processed, no transcoding occurs. For example, you will see correct Latin1 characters in a Latin1 session encoding and correct Latin2 characters in a Latin2 session encoding.

```
libname myfiles 'SAS data-library';

data myfiles.mixed (encoding=any);
   set work.latin1;
   set work.latin2;
run;
```

Example 2: Creating a SAS Data Set with a Particular Encoding For output processing, you can override the current session encoding. This might be necessary, for example, if the normal access to the file will use a different session encoding.

For example, if the current session encoding is Wlatin1, you can specify ENCODING=WLATIN2 in order to create the data set that uses the encoding Wlatin2. The following statements tell SAS to write the data to the new data set using the Wlatin2 encoding instead of the session encoding. The encoding is also specified in the descriptor portion of the file.

```
libname myfiles 'SAS data-library';

data myfiles.difencoding (encoding=wlatin2);
      .
      .
      .
run;
```

Example 3: Overriding Encoding for Input Processing For input processing, you can override the encoding that is specified in the file, and specify a different encoding.

For this example, the current session encoding is EBCDIC-870, but the file has the encoding value EBCDIC-1047 in the descriptor information. By specifying ENCODING=EBCDIC-870, SAS does not transcode the data, but instead displays the data using EBCDIC-870 encoding.

```
proc print data=myfiles.mixed (encoding=ebcdic870);
run;
```

See Also

Conceptual discussion in Chapter 3, “Encoding for NLS,” on page 9

Data Set Options:

“SORTSEQ= Data Set Option” on page 42

Options in Statements and Commands:

“ENCODING= Option” on page 378

“INENCODING= and OUTENCODING= Options” on page 381

System Options:

“ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS” on page 354

“LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS” on page 358

OUTREP= Data Set Option

Specifies the data representation for the output SAS data set

Valid in: DATA step and PROC steps

Category: Data Set Control

See: OUTREP= Data Set Option in *SAS Language Reference: Dictionary*

SORTSEQ= Data Set Option

Specifies a language-specific collation sequence that the SORT procedure uses for the specified SAS data set

Valid in: DATA step and PROC steps

Category: Data Set Control

Syntax

SORTSEQ=*collation-sequence*

Syntax Description

collation-sequence

specifies the collation sequence that the SORT procedure uses for the specified SAS data set. Valid values can be user-supplied, or they can be one of the following:

- ASCII
- DANISH (alias NORWEGIAN)
- EBCDIC
- FINNISH
- ITALIAN
- NATIONAL
- POLISH
- REVERSE
- SPANISH
- SWEDISH

Details

If you want to create or change a collation sequence, use the TRANTAB procedure to create or modify translation tables. When you create your own translation tables, they are stored in your PROFILE catalog, and they override any translation tables with the same name that are stored in the HOST catalog.

Note: System managers can modify the HOST catalog by copying newly created tables from the PROFILE catalog to the HOST catalog. All users can access the new or modified translation tables. △

If you are in a windowing environment, use the Explorer window to display the SASHELP.HOST catalog. In the HOST catalog, entries of type TRANTAB contain collation sequences that are identified by the entry name.

If you are not in a windowing environment, issue the following statements to generate a list of the contents of the HOST catalog. Collation sequences are entries of the type TRANTAB.

```
proc catalog catalog=sashelp.host;
   contents;
run;
```

To see the contents of a particular translation table, use these statements:

```
proc trantab table=translation-table-name;
   list;
run;
```

The contents of collation sequences are displayed in the SAS log.

See Also

System Options:

"SORTSEQ= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 361
"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

PART 3

Formats for NLS

CHAPTER 8

Overview to Formats for NLS

International Date and Datetime Formats

SAS supports international formats that are equivalent to some of the most commonly used English-language date formats. In each case the format works like the corresponding English-language format. Only the maximum, minimum, and default widths are different.

Table 8.1 International Date and Datetime Formats

Language	English Format	International Format	Min	Max	Default
Afrikaans (AFR)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWK.	2	38	28
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFDE.	3	37	29
Catalan (CAT)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	8
	MONYY.	EURDFMY.	5	32	5

Language	English Format	International Format	Min	Max	Default
	WEEKDATX.	EURDFWKX.	2	40	27
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EUDFWDX.	3	40	16
Croatian (CRO)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	10
	MONNAME.	EURDFMN.	1	32	8
	MONYY.	EURDFMY.	5	32	5
	WEEKDATX.	EURDFWKX.	3	40	27
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	40	16
Czech (CSY)	DATE.	EURDFDE.	10	14	12
	DATETIME.	EURDFDT.	12	40	21
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFWN.	1	32	7
	MONNAME.	EURDFMN.	1	32	8
	MONYY.	EURDFMY.	10	32	10
	WEEKDATX.	EURDFWKX.	2	40	25
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	8	40	16
Danish (DAN)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	7
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	2	31	31
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	18	18
Dutch (NLD)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	2	38	28

Language	English Format	International Format	Min	Max	Default
	WORDDATX.	EURDFWDX.	3	37	29
	WEEKDAY.	EURDFDN.	1	32	1
Finnish (FIN)	DATE.	EURDFDE.	9	10	9
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	10
	DOWNAME.	EURDFDWN.	1	32	11
	MONNAME.	EURDFMN.	1	32	11
	MONYY.	EURDFMY.	8	8	8
	WEEKDATX.	EURDFWKX.	2	37	37
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	20	20
French (FRA)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	8
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	27	27
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	18	18
German (DEU)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	10
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	30	30
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	18	18
Hungarian (HUN)	DATE.	EURDFDE.	8	12	10
	DATETIME.	EURDFDT.	0	40	19
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	10
	MONYY.	EURDFMY.	8	32	8
	WEEKDATX.	EURDFWKX.	3	40	28
	WEEKDAY.	EURDFDN.	1	32	1

Language	English Format	International Format	Min	Max	Default
	WORDDATX.	EURDFWDX.	6	40	18
Italian (ITA)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	28	28
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	17	17
Macedonian (MAC)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	10
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	32	5
	WEEKDATX.	EURDFWKX.	3	40	29
	WEEKDDATX.	EURDFWDX.	1	32	1
	WORDDATX.	EURDFDN.	3	40	17
Norwegian (NOR)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	7
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	26	26
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	17	17
Polish (POL)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	12
	MONNAME.	EURDFMN.	1	32	12
	MONYY.	EURDFMY.	5	32	5
	WEEKDATX.	EURDFWKX.	2	40	34
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	40	17

Language	English Format	International Format	Min	Max	Default
Portuguese (PTG)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	13
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	38	38
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	37	23
Russian (RUS)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	11
	MONNAME.	EURDFMN.	1	32	8
	MONYY.	EURDFMY.	5	32	5
	WEEKDATX.	EURDFWKX.	2	40	29
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	40	16
Spanish (ESP)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	9
	MONNAME.	EURDFMN.	1	32	10
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	1	35	35
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	24	24
Slovenian (SLO)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	10
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	32	5
	WEEKDATX.	EURDFWKX.	3	40	29
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	40	17
Swedish (SVE)	DATE.	EURDFDE.	5	9	7

Language	English Format	International Format	Min	Max	Default
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	7
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	26	26
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	17	17
Swiss_French (FRS)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	8
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	26	26
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	17	17
Swiss_German (DES)	DATE.	EURDFDE.	5	9	7
	DATETIME.	EURDFDT.	7	40	16
	DDMMYY.	EURDFDD.	2	10	8
	DOWNAME.	EURDFDWN.	1	32	10
	MONNAME.	EURDFMN.	1	32	9
	MONYY.	EURDFMY.	5	7	5
	WEEKDATX.	EURDFWKX.	3	30	30
	WEEKDAY.	EURDFDN.	1	32	1
	WORDDATX.	EURDFWDX.	3	18	18

European Currency Conversion

Overview to European Currency Conversion

SAS enables you to convert European currency from one country’s currency to an equivalent amount in another country’s currency. You can also convert a country’s currency to euros, and you can convert euros to a specific country’s currency.

SAS provides a group of formats, informats, and a function to use for currency conversion. The set of formats that begin with EURFR converts specific European currencies to an amount in euros. The set of formats that begin with EURTO converts an amount in euros to an amount in a specific European currency. The EUROCURR function also converts one European currency to an amount in another currency.

The default value of the euro symbol is €. The euro symbol precedes the amount with the EURO*w.d*, and EUROX*w.d* formats.

Conversion Rate Tables

The conversion rates for the first eleven countries to agree to euro conversion were established on January 1, 1999. Greece joined the EMU (European Monetary Union) on January 1, 2001 and the conversion of Greek drachmas to euros was established at that time. These rates are fixed, and are incorporated into the EURFR and EURTO formats, and into the EUROCURR function. The following table lists the currency codes and conversion rates for the specific currencies whose rates are fixed.

Note: Add the currency code to EURFR and EURTO to create the format that you need to use. For example, the EURFRATS format converts an amount from Austrian schillings to euros. △

Table 8.2 Fixed Currency Conversion Rates

Currency code	Conversion rate	Currency
ATS	13.7603	Austrian schilling
BEF	40.3399	Belgian franc
DEM	1.95583	Deutsche mark
ESP	166.386	Spanish peseta
EUR	1	Euro
FIM	5.94573	Finnish markka
FRF	6.55957	French franc
GRD	340.750	Greek drachma
IEP	0.787564	Irish pound
ITL	1936.27	Italian lira
LUF	40.3399	Luxembourg franc
NLG	2.20371	Dutch guilder
PTE	200.482	Portuguese escudo

For other countries, currency conversion rates can fluctuate. The conversion rates for these countries are stored in an ASCII text file that you reference with the EURFRTBL fileref.

The following table lists the currency codes and conversion rates for the specific currencies whose rates are changeable.

Table 8.3 Changeable Currency Conversion Rates

Currency code	Conversion rate	Currency
CHF	1.60430	Swiss franc
CZK	34.8563	Czech koruna
DKK	7.49009	Danish krone
GBP	0.700132	British pound

Currency code	Conversion rate	Currency
HUF	260.325	Hungarian forint
NOK	9.19770	Norwegian krone
PLZ	4.2	Polish zloty
ROL	13.71	Romanian leu
RUR	19.7680	Russian ruble
SEK	9.36591	Swedish krona
SIT	191	Slovenian tolar
TRL	336.912	Turkish lira
YUD	13.0644	Yugoslavian dinar

Methods for Converting from One European Currency to Another European Currency

The EUROCURR function uses the conversion rate tables to convert between currencies. If you are converting from one country's currency to euros, SAS divides the country's currency amount by that country's rate from one of the conversion rate tables. If you are converting from euros to a country's currency, SAS multiplies the country's currency amount by that country's rate from one of the conversion rate tables. If you are converting one country's currency to another country's currency, SAS first converts the amount you want to convert to euros. SAS stores the intermediate value as precisely as your operating environment allows, and does not round the value. SAS then converts the amount in euros to an amount in the currency you are converting to.

Formats for NLS by Category

The following categories relate to NLS issues:

Table 8.4 Categories of NLS Formats

Category	**Description**
BIDI text handling	Instructs SAS to write bidirectional data values from data variables.
Character	Instructs SAS to write character data values from character variables.
Currency Conversion	Instructs SAS to convert an amount from one currency to another currency.
DBCS	Instructs SAS to translate double–byte-character sets that are used in Asian languages.
Hebrew text handling	Instructs SAS to read Hebrew data from data variables.
International Date and Time	Instructs SAS to write data values from variables that represent dates, times, and datetimes.
Numeric	Instructs SAS to write numeric data values from numeric variables.

The following table provides brief descriptions of the SAS formats that are related to NLS. For more detailed descriptions, see the NLS entry for each format.

Table 8.5 Summary of NLS Formats by Category

Category	Formats for NLS	Description
BIDI text handling	"$BIDI*w*. Format" on page 65	Converts a logically ordered string to a visually ordered string, and vice versa by reversing the order of Hebrew characters while preserving the order of Latin words and numbers
	"$LOGVS*w*. Format" on page 151	Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order
	"$LOGVSR*w*. Format" on page 152	Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order
	"$VSLOG*w*. Format" on page 196	Processes a character string that is in visual order, and then writes the character string in left-to-right logical order
	"$VSLOGR*w*. Format" on page 197	Processes a character string that is in visual order, and then writes the character string in right-to-left logical order
Character	"$UCS2B*w*. Format" on page 174	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding
	"$UCS2BE*w*. Format" on page 175	Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UCS2L*w*. Format" on page 176	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding
	"$UCS2LE*w*. Format" on page 177	Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UCS2X*w*. Format" on page 178	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding
	"$UCS2XE*w*. Format" on page 180	Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UCS4B*w*. Format" on page 181	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding
	"$UCS4BE*w*. Format" on page 182	Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UCS4L*w*. Format" on page 183	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding

Category	Formats for NLS	Description
	"$UCS4LE*w*. Format" on page 184	Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UCS4X*w*. Format" on page 186	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 32-bit, UCS4, Unicode encoding
	"$UCS4XE*w*. Format" on page 187	Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session
	"$UESC*w*. Format" on page 188	Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation
	"$UESCE*w*. Format" on page 189	Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session
	"$UNCR*w*. Format" on page 190	Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR)
	"$UNCRE*w*. Format" on page 191	Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session
	"$UPAREN*w*. Format" on page 192	Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation
	"$UPARENE*w*. Format" on page 194	Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session
	"$UTF8X*w*. Format" on page 195	Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding
Currency Conversion	"EURFRATS*w.d* Format" on page 86	Converts an amount from Austrian schillings to euros
	"EURFRBEF*w.d* Format" on page 87	Converts an amount from Belgian francs to euros
	"EURFRCHF*w.d* Format" on page 88	Converts an amount from Swiss francs to euros
	"EURFRCZK*w.d* Format" on page 89	Converts an amount from Czech koruny to euros
	"EURFRDEM*w.d* Format" on page 90	Converts an amount from Deutsche marks to euros
	"EURFRDKK*w.d* Format" on page 91	Converts an amount from Danish kroner to euros
	"EURFRESP*w.d* Format" on page 93	Converts an amount from Spanish pesetas to euros

Category	Formats for NLS	Description
	"EURFRFIM*w.d* Format" on page 94	Converts an amount from Finnish markkaa to euros
	"EURFRFRF*w.d* Format" on page 95	Converts an amount from French francs to euros
	"EURFRGBP*w.d* Format" on page 96	Converts an amount from British pounds to euros
	"EURFRGRD*w.d* Format" on page 97	Converts an amount from Greek drachmas to euros
	"EURFRHUF*w.d* Format" on page 98	Converts an amount from Hungarian forints to euros
	"EURFRIEP*w.d* Format" on page 100	Converts an amount from Irish pounds to euros
	"EURFRITL*w.d* Format" on page 101	Converts an amount from Italian lire to euros
	"EURFRLUF*w.d* Format" on page 102	Converts an amount from Luxembourg francs to euros
	"EURFRNLG*w.d* Format" on page 103	Converts an amount from Dutch guilders to euros
	"EURFRNOK*w.d* Format" on page 104	Converts an amount from Norwegian krone to euros
	"EURFRPLZ*w.d* Format" on page 105	Converts an amount from Polish zlotys to euros
	"EURFRPTE*w.d* Format" on page 107	Converts an amount from Portuguese escudos to euros
	"EURFRROL*w.d* Format" on page 108	Converts an amount from Romanian lei to euros
	"EURFRRUR*w.d* Format" on page 109	Converts an amount from Russian rubles to euros
	"EURFRSEK*w.d* Format" on page 110	Converts an amount from Swedish kronor to euros
	"EURFRSIT*w.d* Format" on page 111	Converts an amount from Slovenian tolars to euros
	"EURFRTRL*w.d* Format" on page 113	Converts an amount from Turkish liras to euros
	"EURFRYUD*w.d* Format" on page 114	Converts an amount from Yugoslavian dinars to euros
	"EURTOATS*w.d* Format" on page 118	Converts an amount from euros to Austrian schillings
	"EURTOBEF*w.d* Format" on page 119	Converts an amount from euros to Belgian francs
	"EURTOCHF*w.d* Format" on page 120	Converts an amount from euros to Swiss francs
	"EURTOCZK*w.d* Format" on page 121	Converts an amount from euros to Czech koruny

Category	Formats for NLS	Description
	"EURTODEM*w.d* Format" on page 122	Converts an amount from euros to Deutsche marks
	"EURTODKK*w.d* Format" on page 124	Converts an amount from euros to Danish kroner
	"EURTOESP*w.d* Format" on page 125	Converts an amount from euros to Spanish pesetas
	"EURTOFIM*w.d* Format" on page 126	Converts an amount from euros to Finnish markkaa
	"EURTOFRF*w.d* Format" on page 127	Converts an amount from euros to French francs
	"EURTOGBP*w.d* Format" on page 128	Converts an amount from euros to British pounds
	"EURTOGRD*w.d* Format" on page 130	Converts an amount from euros to Greek drachmas
	"EURTOHUF*w.d* Format" on page 131	Converts an amount from euros to Hungarian forints
	"EURTOIEP*w.d* Format" on page 132	Converts an amount from euros to Irish pounds
	"EURTOITL*w.d* Format" on page 133	Converts an amount from euros to Italian lire
	"EURTOLUF*w.d* Format" on page 134	Converts an amount from euros to Luxembourg francs
	"EURTONLG*w.d* Format" on page 135	Converts an amount from euros to Dutch guilders
	"EURTONOK*w.d* Format" on page 137	Converts an amount from euros to Norwegian krone
	"EURTOPLZ*w.d* Format" on page 138	Converts an amount from euros to Polish zlotys
	"EURTOPTE*w.d* Format" on page 139	Converts an amount from euros to Portuguese escudos
	"EURTOROL*w.d* Format" on page 140	Converts an amount from euros to Romanian lei
	"EURTORUR*w.d* Format" on page 141	Converts an amount from euros to Russian rubles
	"EURTOSEK*w.d* Format" on page 142	Converts an amount from euros to Swedish kronor
	"EURTOSIT*w.d* Format" on page 144	Converts an amount from euros to Slovenian tolars
	"EURTOTRL*w.d* Format" on page 145	Converts an amount from euros to Turkish liras
	"EURTOYUD*w.d* Format" on page 146	Converts an amount from euros to Yugoslavian dinars
DBCS	"$KANJI*w.* Format" on page 149	Adds shift-code data to DBCS data

Category	Formats for NLS	Description
	"$KANJIX*w*. Format" on page 150	Removes shift-code data from DBCS data
Date and Time	"EURDFDD*w*. Format" on page 68	Writes international date values in the form *dd.mm.yy* or *dd.mm.yyyy*
	"EURDFDE*w*. Format" on page 70	Writes international date values in the form *ddmmmyy* or *ddmmmyyyy*
	"EURDFDN*w*. Format" on page 72	Writes international date values as the day of the week
	"EURDFDT*w.d* Format" on page 73	Writes international datetime values in the form *ddmmmyy:hh:mm:ss.ss* or *ddmmmyyyy hh:mm:ss.ss*
	"EURDFDWN*w*. Format" on page 75	Writes international date values as the name of the day
	"EURDFMN*w*. Format" on page 77	Writes international date values as the name of the month
	"EURDFMY*w*. Format" on page 79	Writes international date values in the form *mmmyy* or *mmmyyyy*
	"EURDFWDX*w*. Format" on page 81	Writes international date values as the name of the month, the day, and the year in the form *dd month-name yy* (or *yyyy*)
	"EURDFWKX*w*. Format" on page 83	Writes international date values as the name of the day and date in the form *day-of-week*, *dd month-name yy* (or *yyyy*)
	"HDATE*w*. Format" on page 147	Writes date values in the form *yyyy mmmmm dd* where *dd* is the day-of-the-month, *mmmmm* represents the month's name in Hebrew, and *yyyy* is the year
	"HEBDATE*w*. Format" on page 148	Writes date values according to the Jewish calendar
	"MINGUO*w*. Format" on page 153	Writes date values as Taiwanese dates in the form *yyymmdd*
	"NENGO*w*. Format" on page 154	Writes date values as Japanese dates in the form *e.yymmdd*
	"NLDATE*w*. Format" on page 155	Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date
	"NLDATEMN*w*. Format" on page 156	Converts a SAS date value to the date value of the specified locale, and then writes the value as the name-of-the-month
	"NLDATEW*w*. Format" on page 157	Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day-of-the-week
	"NLDATEWN*w*. Format" on page 158	Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day-of-the-week
	"NLDATM*w*. Format" on page 159	Converts a SAS date-time value to the date-time value of the specified locale, and then writes the value as a date-time

Category	Formats for NLS	Description
	"NLDATMAP*w*. Format" on page 160	Converts a SAS date-time value to the date-time value of the specified locale, and then writes the value as a date-time with a.m. or p.m.
	"NLDATMTM*w*. Format" on page 161	Converts the time portion of a SAS date-time value to the time-of-day value of the specified locale, and then writes the value as a time-of-day
	"NLDATMW*w*. Format" on page 162	Converts a SAS date value to a date-time value of the specified locale, and then writes the value a day-of-week and date-time
	"NLTIME*w*. Format" on page 172	Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value
	"NLTIMAP*w*. Format" on page 173	Converts a SAS time value to the time value of a specified locale, and then writes the value as a time-value with a.m. or p.m.
	"WEEKU*w*. Format" on page 198	Writes a week number in decimal format by using the U algorithm
	"WEEKV*w*. Format" on page 199	Writes a week number in decimal format by using the V algorithm
	"WEEKW*w*. Format" on page 201	Writes a week number in decimal format by using the W algorithm
Hebrew text handling	"$CPTDW*w*. Format" on page 66	Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255)
	"$CPTWD*w*. Format" on page 67	Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding
Numeric	"EURO*w.d* Format" on page 115	Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction
	"EUROX*w.d* Format" on page 116	Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction
	"NLMNY*w.d* Format" on page 163	Writes the monetary format of the local expression in the specified locale using local currency
	"NLMNYI*w.d* Format" on page 165	Writes the monetary format of the international expression in the specified locale
	"NLNUM*w.d* Format" on page 166	Writes the numeric format of the local expression in the specified locale
	"NLNUMI*w.d* Format" on page 168	Writes the numeric format of the international expression in the specified locale
	"NLPCT*w.d* Format" on page 169	Writes percentage data of the local expression in the specified locale

Category	Formats for NLS	Description
	"NLPCTI*w.d* Format" on page 170	Writes percentage data of the international expression in the specified locale
	"YEN*w.d* Format" on page 203	Writes numeric values with yen signs, commas, and decimal points

CHAPTER 9

Formats for NLS

$BIDI*w*. Format

Converts a logically-ordered string to a visually-ordered string, and vice versa, by reversing the order of Hebrew characters while preserving the order of Latin words and numbers

Category: BIDI text handling

Alignment: left

Syntax

$BIDI*w*.

Syntax Description

w

specifies the width of the output field.

Default: 1 if *w* is not specified

Range: 1–32767

Details

In the Windows operating environment, Hebrew text is stored in logical order. This means that the text is stored in the order that it is written and not necessarily as it is displayed. However, in other operating environments, Hebrew text is stored in the same order it is displayed. This can cause SAS users to encounter Hebrew text that is reversed. Such situations can occur when you use SAS/CONNECT or other software to transfer SAS data sets or reports with Hebrew text from a visual operating environment to a logical one. The $BIDI format is a format that reverses Hebrew text while maintaining the order of numbers and Latin-1 words.

Operating Environment Information: In mainframe operating environments, this format is designed to work with NewCode Hebrew. Some mainframe operating environments might experience unsatisfactory results, because they use the OldCode Hebrew encoding. There is a hotfix for this encoding on SAS Institute's Web site: **`http://support.sas.com/`**.

△

Comparisons

The $BIDI*w*. format performs a reversing function similar to the $REVERJ*w*. format, which writes character data in reverse order and preserves blanks. $BIDI*w*. behaves in the following way:

- $BIDI*w*. reverses the order of words and numbers in a specified string, preserving blanks. Latin-1 words and numbers themselves are not reversed, only their order in the string.
- When $BIDI encounters a word consisting of Hebrew characters in the text string, the characters in the Hebrew word are reversed and the position of the Hebrew word is reversed in the string.

Examples

This example demonstrates how $BIDI*w.* reverses Hebrew characters. The Hebrew is reversed in the string. The Hebrew characters in the words are also reversed.

```
data;
   a='א התיכ םולש abc 123';
   b1 = put (a,$bidi20.);
   put b=;
   b2 = put (b,$bidi20.);
   put b=;
run;
```

The following lines are written to the SAS log:

```
b1=123 abc שלום כיתה א
b2=א התיכ םולש abc 123
```

$CPTDW*w*. Format

Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255)

Category: Hebrew text handling

Alignment: left

Syntax

$CPTDW*w*.

Syntax Description

w

specifies the width of the output field.

Default: 200

Range: 1–32000

Comparisons

The $CPTDW*w*. format performs processing that is the opposite of the $CPTWD*w*. format.

Examples

The following example uses the input value of “808182x.”

Statement	Result
	`----+----1----+`
`put text $cptdw3.;`	אבג

See Also

Formats:

“$CPTWD*w*. Format” on page 67

Informats:

“$CPTDW*w*. Informat” on page 250

“$CPTWD*w*. Informat” on page 251

$CPTWD*w*. Format

Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding

Category: Hebrew text handling

Alignment: left

Syntax

$CPTWD*w*.

Syntax Description

w
specifies the width of the output field.

Default: 200

Range: 1–32000

Comparisons

The $CPTWD*w*. format performs processing that is the opposite of the $CPTDW*w*. format.

Examples

The following example uses the input value of “אבג”.

Statement	Result
	`----+----1----+----2----+`
`put text $cptwd3.;`	€□,

See Also

Formats:

“$CPTDW*w*. Format” on page 66

Informats:

“$CPTDW*w*. Informat” on page 250
“$CPTWD*w*. Informat” on page 251

EURDFDD*w*. Format

Writes international date values in the form *dd.mm.yy* or *dd.mm.yyyy*

Category: Date and Time

Alignment: right

Syntax

EURDFDD*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8 (except Finnish, which is 10)

Range: 2–10

Tip: When *w* is from 2 to 5, SAS prints as much of the month and day as possible. When *w* is 7, the date appears as a two-digit year without slashes, and the value is right aligned in the output field.

Details

The EURDFDD*w*. format writes SAS date values in the form *dd.mm.yy* or *dd.mm.yyyy*, where

dd
: is the two-digit integer that represents the day of the month.

mm
: is the two-digit integer that represents the month.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= system option.

Examples

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the international date value. The third PUT statement uses the French language prefix in the format to write the international date value. Therefore, the value of the DFLANG= option is ignored.

Statement	Result
	----+----1
put date eurdfdd8.;	**02.01.02**
put date espdfdd8.;	**02.01.02**
put date fradfdd8.;	**02/01/02**

See Also

Formats:
DATEw. in *SAS Language Reference: Dictionary*
DDMMYYw. in *SAS Language Reference: Dictionary*
MMDDYYw. in *SAS Language Reference: Dictionary*
YYMMDDw. in *SAS Language Reference: Dictionary*
Functions:
MDY in *SAS Language Reference: Dictionary*
Informats:
DATEw. in *SAS Language Reference: Dictionary*
DDMMYYw. in *SAS Language Reference: Dictionary*
MMDDYYw. in *SAS Language Reference: Dictionary*
YYMMDDw. in *SAS Language Reference: Dictionary*
System Options:
"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFDE*w*. Format

Writes international date values in the form *ddmmmyy* or *ddmmmyyyy*

Category: Date and Time

Alignment: right

Syntax

EURDFDE*w*.

Syntax Description

w
specifies the width of the output field.

Default: 7 (except Finnish)

Range: 5–9 (except Finnish)

Note: If you use the Finnish (FIN) language prefix, the *w* range is 9–10 and the default is 9. △

Details

The EURDFDE*w*. format writes SAS date values in the form *ddmmmyy* or *ddmmmyyyy*:

dd
is an integer that represents the day of the month.

mmm

is the first three letters of the month name.

yy or *yyyy*
is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set, such as UTF-8. △

Examples

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes the DFLANG= system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the international date value in Spanish. The third PUT statement uses the French language prefix in the format to write the international date value in French. Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1`
`put date eurdfde9.;`	`02ene2002`
`put date espdfde9.;`	`02ene2002`
`put date fradfde9.;`	`02jan2002`

See Also

Formats:

DATEw. in *SAS Language Reference: Dictionary*

Functions:

DATE in *SAS Language Reference: Dictionary*

Informats:

"EURDFDE*w*. Informat" on page 252

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFDN*w*. Format

Writes international date values as the day of the week

Category: Date and Time

Alignment: right

Syntax

EURDFDN*w*.

Syntax Description

w
specifies the width of the output field.

Default: 1

Range: 1–32

Details

The EURDFDN*w*. format writes SAS date values in the form *day-of-the-week*:

day-of-the-week
is represented as 1=Monday, 2=Tuesday, and so forth.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Examples

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes that the DFLANG= system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the day of the week in Spanish. The third PUT statement uses the Italian language prefix in the format to write the day of the week in Italian . Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1`
`put day eurdfdn.;`	`3`
`put day espdfdn.;`	`3`
`put day itadfdn.;`	`3`

See Also

Formats:
  DOWNAMEw. in *SAS Language Reference: Dictionary*
  WEEKDAYw. in *SAS Language Reference: Dictionary*
System Options:
  "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFDT*w.d* Format

Writes international datetime values in the form *ddmmmyy:hh:mm:ss.ss* or *ddmmmyyyy hh:mm:ss.ss*

Category: Date and Time

Alignment: right

Syntax

EURDFDT*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 16

Range: 7–40

Tip: If you want to write a SAS datetime value with the date, hour, and seconds, the width (*w*) must be at least 16. Add an additional two places to the width if you want to return values with optional decimal fractions of seconds.

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Range: 1–39

Restriction: must be less than *w*

Restriction: If $w - d < 17$, SAS truncates the decimal values.

Details

The EURDFDT*w.d* format writes SAS datetime values in the form *ddmmmyy:hh:mm:ss.ss*:

dd
: is an integer that represents the day of the month.

mmm
: is the first three letters of the month name.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

hh
: is the number of hours that range from 00 through 23.

mm
: is the number of minutes that range from 00 through 59.

ss.ss
: is the number of seconds that range from 00 through 59 with the fraction of a second following the decimal point.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Examples

The example table uses the input value of 1347453583, which is the SAS datetime value that corresponds to September 12, 2002, at 12:39:43 PM. The first PUT statement assumes the DFLANG= system option is set to German.

```
options dflang=german;
```

The second PUT statement uses the German language prefix in the format to write the international datetime value in German. The third PUT statement uses the Italian language prefix in the format to write the international datetime value in Italian. The value of the DFLANG= option, therefore, is ignored.

Statements	Results
	`----+----1----+----2`
`put date eurdfdt20.;`	`12Sep2002:12:39:43`
`put date deudfdt20.;`	`12Sep2002:12:39:43`
`put date itadfdt20.;`	`12Set2002:12:39:43`

See Also

Formats:

DATEw. in *SAS Language Reference: Dictionary*

DATETIMEw.d in *SAS Language Reference: Dictionary*

TIMEw.d in *SAS Language Reference: Dictionary*

Functions:

DATETIME in *SAS Language Reference: Dictionary*

Informats:

DATEw. in *SAS Language Reference: Dictionary*

DATETIMEw.d in *SAS Language Reference: Dictionary*

"EURDFDT*w*. Informat" on page 253

TIMEw.d in *SAS Language Reference: Dictionary*

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFDWN*w*. Format

Writes international date values as the name of the day

Category: Date and Time

Alignment: right

Syntax

EURDFDWN*w*.

Syntax Description

w

specifies the width of the output field.

Default: depends on the language prefix you use. The following table shows the default value for each language:

Language	Default
Afrikaans (AFR)	9
Catalan (CAT)	9
Croatian (CRO)	10
Czech (CSY)	7
Danish (DAN)	7
Dutch (NLD)	9
Finnish (FIN)	11
French (FRA)	8
German (DEU)	10
Hungarian (HUN)	9
Italian (ITA)	9
Macedonian (MAC)	10
Norwegian (NOR)	7
Polish (POL)	12
Portuguese (PTG)	13
Russian (RUS)	11
Slovenian (SLO)	10
Spanish (ESP)	9
Swedish (SVE)	7
Swiss-French (FRS)	8
Swiss-German (DES)	10

Range: 1–32

Tip: If you omit *w*, SAS prints the entire name of the day.

Details

If necessary, SAS truncates the name of the day to fit the format width. The EURDFDWN*w*. format writes SAS date values in the form *day-name*:

day-name
is the name of the day.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Examples

The following example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes the DFLANG= system option is set to French.

```
options dflang=french;

put day eurdfdwn8.;
```

The second PUT statement uses the French language prefix in the format to write the day of the week in French. The third PUT statement uses the Spanish language prefix in the format to write the day of the week in Spanish.Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1`
`put day eurdfdwn8.;`	`Vendredi`
`put day fradfdwn8.;`	`Vendredi`
`put day espdfdwn8.;`	`viernes`

See Also

Formats:
- DOWNAMEw. in *SAS Language Reference: Dictionary*
- WEEKDAYw. in *SAS Language Reference: Dictionary*

Informats:
- DATEw. in *SAS Language Reference: Dictionary*
- DATETIMEw.d in *SAS Language Reference: Dictionary*
- "EURDFDT*w*. Informat" on page 253
- TIMEw.d in *SAS Language Reference: Dictionary*

System Options:
- "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFMN*w*. Format

Writes international date values as the name of the month

Category: Date and Time

Alignment: right

Syntax

EURDFMN*w*.

Syntax Description

w
specifies the width of the output field.

Default: 9 (except for Finnish and Spanish)

Range: 1–32

Note: If you use the Finnish (FIN) language prefix, the default value for *w* is 11. If you use the Spanish (ESP) language prefix, the default value for *w* is 10. △

Details

If necessary, SAS truncates the name of the month to fit the format width. The EURDFMN*w*. format writes SAS date values in the form *month-name*:

month-name
is the name of the month.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Examples

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes the DFLANG= system option is set to Italian.

```
options dflang=ita;
```

The second PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. The third PUT statement uses German language prefix in the format to write the name of the month in German. Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1`
`put date eurdfmn10.;`	`janvier`
`put date itadfmn10.;`	`Gennaio`
`put date deudfmn10.;`	`Januar`

See Also

Formats:

MONNAMEw. in *SAS Language Reference: Dictionary*

Functions:

DATE in *SAS Language Reference: Dictionary*

Informats:

"EURDFDE*w*. Informat" on page 252

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFMY*w*. Format

Writes international date values in the form *mmmyy* or *mmmyyyy*

Category: Date and Time

Alignment: right

Syntax

EURDFMY*w*.

Syntax Description

w

specifies the width of the output field.

Default: 5 (excepteu for Finnish)

Range: 5–7

Note: If you use the Finnish (FIN) language prefix, the value for *w* must be 8, which is the default value. △

Details

The EURDFMY*w*. format writes SAS date values in the form *mmmyy*, where

mmm
: is the first three letters of the month name.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Examples

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes the DFLANG= system option is set to Spanish.

```
options dflang=spanish;
```

The second PUT statement uses the Spanish language prefix in the format to write the name of the month in Spanish. The third PUT statement uses the French language prefix in the format to write the name of the month in French. Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1`
`put date eurdfmy7.;`	**`ene2002`**
`put date espdfmy7.;`	**`ene2002`**
`put date fradfmy7.;`	**`jan2002`**

See Also

Formats:

DDMMYYw. in *SAS Language Reference: Dictionary*

MMDDYYw. in *SAS Language Reference: Dictionary*

MONYYw. in *SAS Language Reference: Dictionary*

YYMMDDw. in *SAS Language Reference: Dictionary*

Functions:

MONTH in *SAS Language Reference: Dictionary*

YEAR in *SAS Language Reference: Dictionary*

Informats:

"EURDFMY*w*. Informat" on page 255

MONYYw. in *SAS Language Reference: Dictionary*

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFWDX*w*. Format

Writes international date values as the name of the month, the day, and the year in the form *dd month-name yy* (or *yyyy*)

Category: Date and Time

Alignment: right

Syntax

EURDFWDX*w*.

Syntax Description

w

specifies the width of the output field.

Default: depends on the language prefix you use. The following table shows the default value for each language:

Language	Maximum	Default
Afrikaans (AFR)	37	29
Catalan (CAT)	40	16
Croatian (CRO)	40	16
Czech (CSY)	40	16
Danish (DAN)	18	18
Dutch (NLD)	37	29
Finnish (FIN)	20	20
French (FRA)	18	18
German (DEU)	18	18
Hungarian (HUN)	40	18
Italian (ITA)	17	17
Macedonian (MAC)	40	17
Norwegian (NOR)	17	17

Language	Maximum	Default
Polish (POL)	40	20
Portuguese (PTG)	37	23
Russian (RUS)	40	16
Slovenian (SLO)	40	17
Spanish (ESP)	24	24
Swedish (SVE)	17	17
Swiss-French (FRS)	17	17
Swiss-German (DES)	18	18

Range: 3–(maximum width)

Tip: If the value for *w* is too small to include the complete day of the week and the month, SAS abbreviates as necessary.

Details

The EURDFWDX*w*. format writes SAS date values in the form *dd month-name yy* or *dd month-name yyyy*:

dd
: is an integer that represents the day of the month.

month-name
: is the name of the month.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Comparisons

The EURDFWKX*w*. format is the same as the EURDFWDX*w*. format except that EURDFWKX *w*. format adds the day-of-week in front of *dd*.

Examples

The example table uses the input value 15342, which is the SAS date value that corresponds to January 2, 2002. The first PUT statement assumes the DFLANG= system option is set to Dutch.

```
options dflang=dutch;
```

The second PUT statement uses the Dutch language prefix in the format to write the name of the month in Dutch. The third PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1----+----2----+----3`
`put day eurdfwdx29.;`	**`2 januari 2002`**
`put day nlddfwdx29.;`	**`2 januari 2002`**
`put day itadfwdx17.;`	**`02 Gennaio 1998`**

See Also

Formats:

WORDDATXw. in *SAS Language Reference: Dictionary*

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURDFWKX*w*. Format

Writes international date values as the name of the day and date in the form *day-of-week*, *dd month-name yy* (or *yyyy*)

Category: Date and Time

Alignment: right

Syntax

EURDFWKX*w*.

Syntax Description

w

specifies the width of the output field.

Default: depends on the language prefix you use. The following table shows the default value for each language:

Language	Minimum	Maximum	Default
Afrikaans (AFR)	2	38	28
Catalan (CAT)	2	40	27
Croatian (CRO)	3	40	27
Czech (CSY)	2	40	25
Danish (DAN)	2	31	31
Dutch (NLD)	2	38	28
Finnish (FIN)	2	37	37
French (FRA)	3	27	27
German (DEU)	3	30	30
Hungarian (HUN)	3	40	28
Italian (ITA)	3	28	28
Macedonian (MAC)	3	40	29
Norwegian (NOR)	3	26	26
Polish (POL)	2	40	34
Portuguese (PTG)	3	38	38
Russian (RUS)	2	40	29
Slovenian (SLO)	3	40	29
Spanish (ESP)	1	35	35
Swedish (SVE)	3	26	26
Swiss-French (FRS)	3	26	26
Swiss-German (DES)	3	30	30

Tip: If the value for *w* is too small to include the complete day of the week and the month, SAS abbreviates as necessary.

Details

The EURDFWKX*w*. format writes SAS date values in the form *day-of-week*, *dd month-name yy* (or *yyyy*):

day-of-week
: is the name of day.

dd
: is an integer that represents the day of the month.

month-name
: is the name of the month.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 353 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

Note: The EUR-date formats require European character sets and encodings. Some formats will not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width will be larger than in the single byte system to allow formats to use a double byte representation of certain characters. However, you must use a session encoding that supports the European characters set like UTF-8. △

Comparisons

The EURDFWKX*w*. format is the same as the EURDFWDX*w*. format except that EURDFWKX*w*. format adds day-of-week in front of *dd*.

Examples

The example table uses the input value 15344, which is the SAS date value that corresponds to January 4, 2002. The first PUT statement assumes the DFLANG= system option is set to German.

```
options dflang=German;
```

The second PUT statement uses the German language prefix in the format to write the name of the month in German. The third PUT statement uses the Italian language prefix in the format to write the name of the month in Italian. Therefore, the value of the DFLANG= option is ignored.

Statements	Results
	`----+----1----+----2----+----3`
`put date eurdfwkx30.;`	**`       Freitag, 4. Januar 2002`**
`put date deudfwkx30.;`	**`       Freitag, 4. Januar 2002`**
`put date itadfwkx17.;`	**` Ven, 04 Gen 2002`**

See Also

Formats:
- DATEw. in *SAS Language Reference: Dictionary*
- DDMMYYw. in *SAS Language Reference: Dictionary*
- MMDDYYw. in *SAS Language Reference: Dictionary*
- TODw. in *SAS Language Reference: Dictionary*
- WEEKDATXw. in *SAS Language Reference: Dictionary*
- YYMMDDw. in *SAS Language Reference: Dictionary*

Functions:
- JULDATE in *SAS Language Reference: Dictionary*

MDY in *SAS Language Reference: Dictionary*
WEEKDAY in *SAS Language Reference: Dictionary*

Informats:

DATEw. in *SAS Language Reference: Dictionary*
DDMMYYw. in *SAS Language Reference: Dictionary*
MMDDYYw. in *SAS Language Reference: Dictionary*
YYMMDDw. in *SAS Language Reference: Dictionary*

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

EURFRATS*w.d* Format

Converts an amount from Austrian schillings to euros

Category: Currency Conversion
Alignment: right

Syntax

EURFRATS*w.d*

Syntax Description

w
specifies the width of the output field.
Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRATS *w.d* format converts an amount from Austrian schillings to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRATS*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Austrian schillings, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrats5.;`	`   E4`
	`put amount eurfrats9.2;`	`    E3,63`

Amounts	Statements	Results
5234.56	put amount eurfrats5.;	E380
	put amount eurfrats9.2;	E380,41
52345	put amount eurfrats5.;	3.804
	put amount eurfrats9.2;	E3.804,06

See Also

Formats:

"EURTOATS*w.d* Format" on page 118

Functions:

"EUROCURR Function" on page 209

EURFRBEF*w.d* Format

Converts an amount from Belgian francs to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRBEF*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRBEF*w.d* format converts an amount from Belgian francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRBEF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Belgian francs, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
50	`put amount eurfrbef5.;`	`   E1`
	`put amount eurfrbef9.2;`	`    E1,24`
5234.56	`put amount eurfrbef5.;`	` E130`
	`put amount eurfrbef9.2;`	`  E129,76`
52345	`put amount eurfrbef5.;`	`1.298`
	`put amount eurfrbef9.2;`	`E1.297,60`

See Also

Formats:

"EURTOBEF*w.d* Format" on page 119

Functions:

"EUROCURR Function" on page 209

EURFRCHF*w.d* Format

Converts an amount from Swiss francs to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRCHF*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRCHF*w.d* format converts an amount from Swiss francs to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRCHF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Swiss francs, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
50	`put amount eurfrchf5.;`	`  E31`
	`put amount eurfrchf9.2;`	`   E31,17`
1234.56	`put amount eurfrchf5.;`	` E770`
	`put amount eurfrchf9.2;`	`  E769,53`
12345	`put amount eurfrchf5.;`	`7.695`
	`put amount eurfrchf9.2;`	`E7.694,94`

See Also

Formats:
"EURTOCHF*w.d* Format" on page 120

Functions:
"EUROCURR Function" on page 209

EURFRCZK*w.d* Format

Converts an amount from Czech koruny to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRCZK*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRCZK*w.d* format converts an amount from Czech koruny to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRCZK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Czech koruny, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrczk5.;`	`   E1`
	`put amount eurfrczk9.2;`	`    E1,43`
`5234.56`	`put amount eurfrczk5.;`	` E150`
	`put amount eurfrczk9.2;`	`  E150,18`
`52345`	`put amount eurfrczk5.;`	`1.502`
	`put amount eurfrczk9.2;`	`E1.501,74`

See Also

Formats:
"EURTOCZK*w.d* Format" on page 121
Functions:
"EUROCURR Function" on page 209

EURFRDEM*w.d* Format

Converts an amount from Deutsche marks to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRDEM*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRDEM*w.d* format converts an amount from Deutsche marks to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRDEM*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Deutsche marks, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrdem5.;`	`   E26`
	`put amount eurfrdem9.2;`	`    E25,56`
`1234.56`	`put amount eurfrdem5.;`	`  E631`
	`put amount eurfrdem9.2;`	`   E631,22`
`12345`	`put amount eurfrdem5.;`	`6.312`
	`put amount eurfrdem9.2;`	`E6.311,90`

See Also

Formats:
"EURTODEM*w.d* Format" on page 122
Functions:
"EUROCURR Function" on page 209

EURFRDKK*w.d* Format

Converts an amount from Danish kroner to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRDKK*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRDKK*w.d* format converts an amount from Danish kroner to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRDKK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Danish kroner, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrdkk5.;`	`   E7`
	`put amount eurfrdkk9.2;`	`    E6,68`
`1234.56`	`put amount eurfrdkk5.;`	` E165`
	`put amount eurfrdkk9.2;`	`  E164,83`
`12345`	`put amount eurfrdkk5.;`	`1.648`
	`put amount eurfrdkk9.2;`	`E1.648,18`

See Also

Formats:

"EURTODKK*w.d* Format" on page 124

Functions:

"EUROCURR Function" on page 209

EURFRESP*w.d* Format

Converts an amount from Spanish pesetas to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRESP*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRESP*w.d* format converts an amount from Spanish pesetas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRESP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Spanish pesetas, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`200`	`put amount eurfresp5.;`	`   E1`
	`put amount eurfresp9.2;`	`    E1,20`
`20234.56`	`put amount eurfresp5.;`	`  E122`
	`put amount eurfresp9.2;`	`  E121,61`
`202345`	`put amount eurfresp5.;`	`1.216`
	`put amount eurfresp9.2;`	`E1.216,12`

See Also

Formats:
  "EURTOESP*w.d* Format" on page 125
Functions:
  "EUROCURR Function" on page 209

EURFRFIM*w.d* Format

Converts an amount from Finnish markkaa to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRFIM*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRFIM*w.d* format converts an amount from Finnish markkaa to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFIM*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Finnish markkaa, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrfim5.;`	`E8`
	`put amount eurfrfim9.2;`	`E8,41`
`1234.56`	`put amount eurfrfim5.;`	`E208`
	`put amount eurfrfim9.2;`	`E207,64`
`12345`	`put amount eurfrfim5.;`	`2.076`
	`put amount eurfrfim9.2;`	`E2.076,28`

See Also

Formats:
"EURTOFIM*w.d* Format" on page 126
Functions:
"EUROCURR Function" on page 209

EURFRFRF*w.d* Format

Converts an amount from French francs to euros

Category: Currency Conversion
Alignment: right

Syntax

EURFRFRF*w.d*

Syntax Description

w
specifies the width of the output field.
Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRFRF*w.d* format converts an amount from French francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRFRF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in French francs, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
50	`put amount eurfrfrf5.;`	`   E8`
	`put amount eurfrfrf9.2;`	`    E7,62`
1234.56	`put amount eurfrfrf5.;`	` E188`
	`put amount eurfrfrf9.2;`	`  E188,21`
12345	`put amount eurfrfrf5.;`	`1.882`
	`put amount eurfrfrf9.2;`	`E1.881,98`

See Also

Formats:
: “EURTOFRF*w.d* Format” on page 127

Functions:
: “EUROCURR Function” on page 209

EURFRGBP*w.d* Format

Converts an amount from British pounds to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRGBP*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRGBP*w.d* format converts an amount from British pounds to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRGBP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in British pounds, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrgbp5.;`	`   E71`
	`put amount eurfrgbp9.2;`	`    E71.42`
`1234.56`	`put amount eurfrgbp5.;`	`1,763`
	`put amount eurfrgbp9.2;`	`E1,763.32`
`12345`	`put amount eurfrgbp5.;`	`17632`
	`put amount eurfrgbp9.2;`	`17,632.39`

See Also

Formats:
“EURTOGBP*w.d* Format” on page 128
Functions:
“EUROCURR Function” on page 209

EURFRGRD*w.d* Format

Converts an amount from Greek drachmas to euros

Category: Currency Conversion
Alignment: right

Syntax

EURFRGRD*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRGRD*w.d* format converts an amount from Greek drachmas to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRGRD*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Greek drachmas, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`400`	`put amount eurfrgrd5.;`	`   E1`
	`put amount eurfrgrd9.2;`	`    E1,17`
`40234.56`	`put amount eurfrgrd5.;`	` E118`
	`put amount eurfrgrd9.2;`	`  E118,03`
`402345`	`put amount eurfrgrd5.;`	`1.180`
	`put amount eurfrgrd9.2;`	`E1.180,30`

See Also

Formats:

"EURTOGRD*w.d* Format" on page 130

Functions:

"EUROCURR Function" on page 209

EURFRHUF*w.d* Format

Converts an amount from Hungarian forints to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRHUF*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRHUF*w.d* format converts an amount from Hungarian forints to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRHUF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Hungarian forints, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`300`	`put amount eurfrhuf5.;`	`   E1`
	`put amount eurfrhuf9.2;`	`    E1,15`
`30234.56`	`put amount eurfrhuf5.;`	` E116`
	`put amount eurfrhuf9.2;`	`  E116,14`
`302345`	`put amount eurfrhuf5.;`	`1.161`
	`put amount eurfrhuf9.2;`	`E1.161,41`

See Also

Formats:

"EURTOHUF*w.d* Format" on page 131

Functions:

"EUROCURR Function" on page 209

EURFRIEP*w.d* Format

Converts an amount from Irish pounds to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRIEP*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRIEP*w.d* format converts an amount from Irish pounds to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRIEP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Irish pounds, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurfriep5.;`	`   E1`
	`put amount eurfriep9.2;`	`    E1.27`
`1234.56`	`put amount eurfriep5.;`	`1,568`
	`put amount eurfriep9.2;`	`E1,567.57`
`12345`	`put amount eurfriep5.;`	`15675`
	`put amount eurfriep9.2;`	`15,674.92`

See Also

Formats:
: "EURTOIEP*w.d* Format" on page 132

Functions:
: "EUROCURR Function" on page 209

EURFRITL*w.d* Format

Converts an amount from Italian lire to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRITL*w.d*

Syntax Description

w
: specifies the width of the output field.

  Default: 6

d
: optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRITL*w.d* format converts an amount from Italian lire to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRITL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Italian lire, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`2000`	`put amount eurfritl5.;`	`E1`
	`put amount eurfritl9.2;`	`E1,03`
`7234.56`	`put amount eurfritl5.;`	`E4`
	`put amount eurfritl9.2;`	`E3,74`
`72345`	`put amount eurfritl5.;`	`E37`
	`put amount eurfritl9.2;`	`E37,36`

See Also

Formats:

“EURTOITL*w.d* Format” on page 133

Functions:

“EUROCURR Function” on page 209

EURFRLUF*w.d* Format

Converts an amount from Luxembourg francs to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRLUF*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRLUF*w.d* format converts an amount from Luxembourg francs to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRLUF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in Luxembourg francs, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrluf5.;`	`   E1`
	`put amount eurfrluf9.2;`	`    E1,24`
`1234.56`	`put amount eurfrluf5.;`	`  E31`
	`put amount eurfrluf9.2;`	`   E30,60`
`12345`	`put amount eurfrluf5.;`	` E306`
	`put amount eurfrluf9.2;`	`  E306,02`

See Also

Formats:
“EURTOLUF*w.d* Format” on page 134
Functions:
“EUROCURR Function” on page 209

EURFRNLG*w.d* Format

Converts an amount from Dutch guilders to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRNLG*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRNLG*w.d* format converts an amount from Dutch guilders to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRNLG*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Dutch guilders, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrnlg5.;`	`  E23`
	`put amount eurfrnlg9.2;`	`   E22,69`
`1234.56`	`put amount eurfrnlg5.;`	` E560`
	`put amount eurfrnlg9.2;`	`  E560,22`
`12345`	`put amount eurfrnlg5.;`	`5.602`
	`put amount eurfrnlg9.2;`	`E5.601,92`

See Also

Formats:
 "EURTONLG*w.d* Format" on page 135
Functions:
 "EUROCURR Function" on page 209

EURFRNOK*w.d* Format

Converts an amount from Norwegian krone to euros

Category: Currency Conversion
Alignment: right

Syntax

EURFRNOK*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRNOK*w.d* format converts an amount from Norwegian krone to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRNOK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in Norwegian krone, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrnok5.;`	`   E5`
	`put amount eurfrnok9.2;`	`    E5,44`
`1234.56`	`put amount eurfrnok5.;`	` E134`
	`put amount eurfrnok9.2;`	`  E134,22`
`12345`	`put amount eurfrnok5.;`	`1.342`
	`put amount eurfrnok9.2;`	`E1.342,18`

See Also

Formats:

“EURTONOK*w.d* Format” on page 137

Functions:

“EUROCURR Function” on page 209

EURFRPLZ*w.d* Format

Converts an amount from Polish zlotys to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRPLZ*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRPLZ*w.d* format converts an amount from Polish zlotys to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRPLZ*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Polish zlotys, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrplz5.;`	`  E12`
	`put amount eurfrplz9.2;`	`   E11,90`
`1234.56`	`put amount eurfrplz5.;`	` E294`
	`put amount eurfrplz9.2;`	`  E293,94`
`12345`	`put amount eurfrplz5.;`	`2.939`
	`put amount eurfrplz9.2;`	`E2.939,29`

See Also

Formats:

"EURTOPLZ*w.d* Format" on page 138

Functions:

"EUROCURR Function" on page 209

EURFRPTE*w.d* Format

Converts an amount from Portuguese escudos to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRPTE*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRPTE*w.d* format converts an amount from Portuguese escudos to an amount in euros and produces a formatted euro value. The conversion rate is a fixed rate that is incorporated into the EURFRPTE*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Portuguese escudos, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`300`	`put amount eurfrpte5.;`	`   E1`
	`put amount eurfrpte9.2;`	`    E1,50`
`30234.56`	`put amount eurfrpte5.;`	`  E151`
	`put amount eurfrpte9.2;`	`  E150,81`
`302345`	`put amount eurfrpte5.;`	`1.508`
	`put amount eurfrpte9.2;`	`E1.508,09`

See Also

Formats:
    "EURTOPTE*w.d* Format" on page 139
Functions:
    "EUROCURR Function" on page 209

EURFRROL*w.d* Format

Converts an amount from Romanian lei to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRROL*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRROL*w.d* format converts an amount from Romanian lei to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRROL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Romanian lei, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
50	**put amount eurfrrol5.;**	**E4**
	put amount eurfrrol9.2;	**E3,65**
5234.56	**put amount eurfrrol5.;**	**E382**
	put amount eurfrrol9.2;	**E381,81**
52345	**put amount eurfrrol5.;**	**3.818**
	put amount eurfrrol9.2;	**E3.818,02**

See Also

Formats:
"EURTOROL*w.d* Format" on page 140
Functions:
"EUROCURR Function" on page 209

EURFRRUR*w.d* Format

Converts an amount from Russian rubles to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRRUR*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRRUR*w.d* format converts an amount from Russian rubles to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRRUR*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in Russian rubles, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrrur5.;`	`   E3`
	`put amount eurfrrur9.2;`	`    E2,53`
`5234.56`	`put amount eurfrrur5.;`	` E265`
	`put amount eurfrrur9.2;`	`  E264,80`
`52345`	`put amount eurfrrur5.;`	`2.648`
	`put amount eurfrrur9.2;`	`E2.647,97`

See Also

Formats:

“EURTORUR*w.d* Format” on page 141

Functions:

“EUROCURR Function” on page 209

EURFRSEK*w.d* Format

Converts an amount from Swedish kronor to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRSEK*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSEK*w.d* format converts an amount from Swedish kronor to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSEK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Swedish kronor, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfrsek5.;`	`   E5`
	`put amount eurfrsek9.2;`	`    E5,34`
`1234.56`	`put amount eurfrsek5.;`	` E132`
	`put amount eurfrsek9.2;`	`  E131,81`
`12345`	`put amount eurfrsek5.;`	`1.318`
	`put amount eurfrsek9.2;`	`E1.318,08`

See Also

Formats:
    "EURTOSEK*w.d* Format" on page 142
Functions:
    "EUROCURR Function" on page 209

EURFRSIT*w.d* Format

Converts an amount from Slovenian tolars to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRSIT*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRSIT*w.d* format converts an amount from Slovenian tolars to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSIT*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Slovenian tolars, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`200`	`put amount eurfrsit5.;`	`   E1`
	`put amount eurfrsit9.2;`	`    E1,05`
`20234.56`	`put amount eurfrsit5.;`	` E106`
	`put amount eurfrsit9.2;`	`  E105,94`
`202345`	`put amount eurfrsit5.;`	`1.059`
	`put amount eurfrsit9.2;`	`E1.059,40`

See Also

Formats:

"EURTOSIT*w.d* Format" on page 144

Functions:

"EUROCURR Function" on page 209

EURFRTRL*w.d* Format

Converts an amount from Turkish liras to euros

Category: Currency Conversion

Alignment: right

Syntax

EURFRTRL*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRTRL*w.d* format converts an amount from Turkish liras to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRTRL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Turkish liras, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`400`	`put amount eurfrtrl5.;`	`   E1`
	`put amount eurfrtrl9.2;`	`    E1,19`
`40234.56`	`put amount eurfrtrl5.;`	` E119`
	`put amount eurfrtrl9.2;`	`  E119,42`
`402345`	`put amount eurfrtrl5.;`	`1.194`
	`put amount eurfrtrl9.2;`	`E1.194,21`

See Also

Formats:
"EURTOTRL*w.d* Format" on page 145
Functions:
"EUROCURR Function" on page 209

EURFRYUD*w.d* Format

Converts an amount from Yugoslavian dinars to euros

Category: Currency Conversion
Alignment: right

Syntax

EURFRYUD*w.d*

Syntax Description

w
specifies the width of the output field.
Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRYUD*w.d* format converts an amount from Yugoslavian dinars to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRYUD*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in Yugoslavian dinars, SAS statements, and the conversion results in euros.

Amounts	Statements	Results
		`----+----1----2`
`50`	`put amount eurfryud5.;`	`E4`
	`put amount eurfryud9.2;`	`E3,83`
`5234.56`	`put amount eurfryud5.;`	`E401`
	`put amount eurfryud9.2;`	`E400,67`
`52345`	`put amount eurfryud5.;`	`4.007`
	`put amount eurfryud9.2;`	`E4.006,69`

See Also

Formats:
"EURTOYUD*w.d* Format" on page 146

Functions:
"EUROCURR Function" on page 209

EURO*w.d* Format

Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction

Category: Numeric

Alignment: right

Syntax

EURO*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

Range: 1-32

Tip: If you want the euro symbol to be part of the output, be sure to choose an adequate width. See "Examples" on page 116.

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Default: 0

Range: 0-31

Requirement: must be less than *w*

Comparisons

- The EURO*w.d* format is similar to the EUROX*w.d* format, but EUROX*w.d* format reverses the roles of the decimal point and the comma. This convention is common in European countries.
- The EURO*w.d* format is similar to the DOLLAR*w.d* format, except that DOLLAR*w.d* format writes a leading dollar sign instead of the euro symbol.

Examples

These examples use 1254.71 as the value of amount.

Statements	Results
	`----+----1----+----2----+----3`
`put amount euro10.2;`	` E1,254.71`
`put amount euro5.;`	`1,255`
`put amount euro9.2;`	`E1,254.71`
`put amount euro15.3;`	`     E1,254.710`

See Also

Formats:

"EUROX*w.d* Format" on page 116

Informats:

"EURO*w.d* Informat" on page 257
"EUROX*w.d* Informat" on page 258

EUROX*w.d* Format

Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction

Category: Numeric

Alignment: right

Syntax

EUROX*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

Range: 1-32

Tip: If you want the euro symbol to be part of the output, be sure to choose an adequate width. See "Examples" on page 117.

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Default: 0

Range: 0-31

Requirement: must be less than *w*

Comparisons

- The EUROX*w.d* format is similar to the EURO*w.d* format, but EURO*w.d* format reverses the roles of the comma and the decimal point. This convention is common in English–speaking countries.
- The EUROX*w.d* format is similar to the DOLLARX*w.d* format, except that DOLLARX*w.d* format writes a leading dollar sign instead of the euro symbol.

Examples

These examples use 1254.71 as the value of amount.

Statements	Results
	`----+----1----+----2----+----3`
`put amount eurox10.2;`	` E1.254,71`
`put amount eurox5.;`	`1.255`
`put amount eurox9.2;`	`E1.254,71`
`put amount eurox15.3;`	`     E1.254,710`

See Also

Formats:

"EURO*w.d* Format" on page 115

Informats:

"EURO*w.d* Informat" on page 257

"EUROX*w.d* Informat" on page 258

EURTOATS*w.d* Format

Converts an amount from euros to Austrian schillings

Category: Currency Conversion

Alignment: right

Syntax

EURTOATS*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOATS*w.d* format converts an amount in euros to an amount in Austrian schillings. The conversion rate is a fixed rate that is incorporated into the EURTOATS*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Austrian schillings.

Amounts	Statements	Results
		`----+----1----2`
1	`put amount eurtoats6.;`	`    14`
	`put amount eurtoats12.2;`	`       13.76`
1234.56	`put amount eurtoats6.;`	` 16988`
	`put amount eurtoats12.2;`	`    16987.92`
12345	`put amount eurtoats6.;`	`169871`
	`put amount eurtoats12.2;`	`   169870.90`

See Also

Formats:
"EURFRATS*w.d* Format" on page 86
Functions:
"EUROCURR Function" on page 209

EURTOBEF*w.d* Format

Converts an amount from euros to Belgian francs

Category: Currency Conversion

Alignment: right

Syntax

EURTOBEF*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOBEF*w.d* format converts an amount in euros to an amount in Belgian francs. The conversion rate is a fixed rate that is incorporated into the EURTOBEF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Belgian francs.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtobef6.;`	`    40`
	`put amount eurtobef12.2;`	`       40.34`
`1234.56`	`put amount eurtobef6.;`	` 49802`
	`put amount eurtobef12.2;`	`    49802.03`
`12345`	`put amount eurtobef6.;`	`497996`
	`put amount eurtobef12.2;`	`   497996.07`

See Also

Formats:
"EURFRBEF*w.d* Format" on page 87
Functions:
"EUROCURR Function" on page 209

EURTOCHF*w.d* Format

Converts an amount from euros to Swiss francs

Category: Currency Conversion

Alignment: right

Syntax

EURTOCHF*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOCHF*w.d* format converts an amount in euros to an amount in Swiss francs. The conversion rate is a changeable rate that is incorporated into the EURTOCHF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Swiss francs.

Amounts	Statements	Results
		`----+----1----2`
1	`put amount eurtochf6.;`	`     2`
	`put amount eurtochf12.2;`	`        1.60`
1234.56	`put amount eurtochf6.;`	`  1981`
	`put amount eurtochf12.2;`	`     1980.60`
12345	`put amount eurtochf6.;`	` 19805`
	`put amount eurtochf12.2;`	`    19805.08`

See Also

Formats:

“EURFRCHF*w.d* Format” on page 88

Functions:

“EUROCURR Function” on page 209

EURTOCZK*w.d* Format

Converts an amount from euros to Czech koruny

Category: Currency Conversion

Alignment: right

Syntax

EURTOCZK*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOCZK*w.d* format converts an amount in euros to an amount in Czech koruny. The conversion rate is a changeable rate that is incorporated into the EURTOCZK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Czech koruny.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoczk6.;`	`35`
	`put amount eurtoczk12.2;`	`34.86`
`1234.56`	`put amount eurtoczk6.;`	`43032`
	`put amount eurtoczk12.2;`	`43032.19`
`12345`	`put amount eurtoczk6.;`	`430301`
	`put amount eurtoczk12.2;`	`430301.02`

See Also

Formats:
"EURFRCZK*w.d* Format" on page 89
Functions:
"EUROCURR Function" on page 209

EURTODEM*w.d* Format

Converts an amount from euros to Deutsche marks

Category: Currency Conversion

Alignment: right

Syntax

EURTODEM*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTODEM*w.d* format converts an amount in euros to an amount in Deutsche marks. The conversion rate is a fixed rate that is incorporated into the EURTODEM*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Deutsche marks.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtodem6.;`	`     2`
	`put amount eurtodem12.2;`	`        1.96`
`1234.56`	`put amount eurtodem6.;`	`  2415`
	`put amount eurtodem12.2;`	`     2414.59`
`12345`	`put amount eurtodem6.;`	` 24145`
	`put amount eurtodem12.2;`	`    24144.72`

See Also

Formats:

"EURFRDEM*w.d* Format" on page 90

Functions:

"EUROCURR Function" on page 209

EURTODKK*w.d* Format

Converts an amount from euros to Danish kroner

Category: Currency Conversion

Alignment: right

Syntax

EURTODKK*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTODKK*w.d* format converts an amount in euros to an amount in Danish kroner. The conversion rate is a changeable rate that is incorporated into the EURTODKK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Danish kroner.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtodkk6.;`	`     7`
	`put amount eurtodkk12.2;`	`        7.49`
`1234.56`	`put amount eurtodkk6.;`	`  9247`
	`put amount eurtodkk12.2;`	`     9246.97`
`12345`	`put amount eurtodkk6.;`	` 92465`
	`put amount eurtodkk12.2;`	`    92465.16`

See Also

Formats:
    "EURFRDKK*w.d* Format" on page 91
Functions:
    "EUROCURR Function" on page 209

EURTOESP*w.d* Format

Converts an amount from euros to Spanish pesetas

Category: Currency Conversion

Alignment: right

Syntax

EURTOESP*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOESP*w.d* format converts an amount in euros to an amount in Spanish pesetas. The conversion rate is a fixed rate that is incorporated into the EURTOESP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Spanish pesetas.

Amounts	Statements	Results
		----+----1----2
1	put amount eurtoesp8.;	166
	put amount eurtoesp12.2;	**166.39**
1234.56	put amount eurtoesp8.;	205414
	put amount eurtoesp12.2;	205413.50
12345	put amount eurtoesp8.;	2054035
	put amount eurtoesp12.2;	2054035.17

See Also

Formats:
“EURFRESP*w.d* Format” on page 93
Functions:
“EUROCURR Function” on page 209

EURTOFIM*w.d* Format

Converts an amount from euros to Finnish markkaa

Category: Currency Conversion

Alignment: right

Syntax

EURTOFIM*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOFIM*w.d* format converts an amount in euros to an amount in Finnish markkaa. The conversion rate is a fixed rate that is incorporated into the EURTOFIM*w.d* format and the EUROCURR function. For more information about

European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Finnish markkaa.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtofim6.;`	`     6`
	`put amount eurtofim12.2;`	`        5.95`
`1234.56`	`put amount eurtofim6.;`	`  7340`
	`put amount eurtofim12.2;`	`     7340.36`
`12345`	`put amount eurtofim6.;`	` 73400`
	`put amount eurtofim12.2;`	`    73400.04`

See Also

Formats:
"EURFRFIM*w.d* Format" on page 94

Functions:
"EUROCURR Function" on page 209

EURTOFRF*w.d* Format

Converts an amount from euros to French francs

Category: Currency Conversion

Alignment: right

Syntax

EURTOFRF*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOFRF*w.d* format converts an amount in euros to an amount in French francs. The conversion rate is a fixed rate that is incorporated into the EURTOFRF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in French francs.

Amounts	Statements	Results
		`----+----1----2`
1	**put amount eurtofrf6.;**	`     7`
	put amount eurtofrf12.2;	`        6.56`
1234.56	**put amount eurtofrf6.;**	`  8098`
	put amount eurtofrf12.2;	`     8098.18`
12345	**put amount eurtofrf6.;**	` 80978`
	put amount eurtofrf12.2;	`    80977.89`

See Also

Formats:
“EURFRFRF*w.d* Format” on page 95
Functions:
“EUROCURR Function” on page 209

EURTOGBP*w.d* Format

Converts an amount from euros to British pounds

Category: Currency Conversion

Alignment: right

Syntax

EURTOGBP*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOGBP*w.d* format converts an amount in euros to an amount in British pounds. The conversion rate is a changeable rate that is incorporated into the EURTOGBP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in British pounds.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtogbp6.;`	`     1`
	`put amount eurtogbp12.2;`	`        0.70`
`1234.56`	`put amount eurtogbp6.;`	`   864`
	`put amount eurtogbp12.2;`	`      864.35`
`12345`	`put amount eurtogbp6.;`	`  8643`
	`put amount eurtogbp12.2;`	`     8643.13`

See Also

Formats:
"EURFRGBP*w.d* Format" on page 96
Functions:
"EUROCURR Function" on page 209

EURTOGRD*w.d* Format

Converts an amount from euros to Greek drachmas

Category: Currency Conversion

Alignment: right

Syntax

EURTOGRD*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOGRD*w.d* format converts an amount in euros to an amount in Greek drachmas. The conversion rate is a fixed rate that is incorporated into the EURTOGRD*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Greek drachmas.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtogrd8.;`	`     341`
	`put amount eurtogrd16.2;`	`          340.89`
`1234.56`	`put amount eurtogrd8.;`	`  420843`
	`put amount eurtogrd16.2;`	`       420842.99`
`12345`	`put amount eurtogrd8.;`	` 4208225`
	`put amount eurtogrd16.2;`	`      4208225.33`

See Also

Formats:

"EURFRGRD*w.d* Format" on page 97

Functions:

"EUROCURR Function" on page 209

EURTOHUF*w.d* Format

Converts an amount from euros to Hungarian forints

Category: Currency Conversion

Alignment: right

Syntax

EURTOHUF*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOHUF*w.d* format converts an amount in euros to an amount in Hungarian forints. The conversion rate is a changeable rate that is incorporated into the EURTOHUF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Hungarian forints.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtohuf8.;`	`     260`
	`put amount eurtohuf14.2;`	`        260.33`
`1234.56`	`put amount eurtohuf8.;`	`  321387`
	`put amount eurtohuf14.2;`	`     321386.83`
`12345`	`put amount eurtohuf8.;`	` 3213712`
	`put amount eurtohuf14.2;`	`    3213712.13`

See Also

Formats:
 "EURFRHUF*w.d* Format" on page 98
Functions:
 "EUROCURR Function" on page 209

EURTOIEP*w.d* Format

Converts an amount from euros to Irish pounds

Category: Currency Conversion

Alignment: right

Syntax

EURTOIEP*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOIEP*w.d* format converts an amount in euros to an amount in Irish pounds. The conversion rate is a fixed rate that is incorporated into the EURTOIEP*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Irish pounds.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoiep6.;`	`     1`
	`put amount eurtoiep12.2;`	`        0.79`
`1234.56`	`put amount eurtoiep6.;`	`   972`
	`put amount eurtoiep12.2;`	`      972.30`
`12345`	`put amount eurtoiep6.;`	`  9722`
	`put amount eurtoiep12.2;`	`     9722.48`

See Also

Formats:
   “EURFRIEP*w.d* Format” on page 100
Functions:
   “EUROCURR Function” on page 209

EURTOITL*w.d* Format

Converts an amount from euros to Italian lire

Category: Currency Conversion

Alignment: right

Syntax

EURTOITL*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOITL*w.d* format converts an amount in euros to an amount in Italian lire. The conversion rate is a fixed rate that is incorporated into the EURTOITL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Italian lire.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoitl8.;`	`    1936`
	`put amount eurtoitl12.2;`	`     1936.27`
`1234.56`	`put amount eurtoitl8.;`	` 2390441`
	`put amount eurtoitl12.2;`	`  2390441.49`
`12345`	`put amount eurtoitl8.;`	`23903253`
	`put amount eurtoitl12.2;`	` 23903253.15`

See Also

Formats:
 “EURFRITL*w.d* Format” on page 101
Functions:
 “EUROCURR Function” on page 209

EURTOLUF*w.d* Format

Converts an amount from euros to Luxembourg francs

Category: Currency Conversion

Alignment: right

Syntax

EURTOLUF*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOLUF*w.d* format converts an amount in euros to an amount in Luxembourg francs. The conversion rate is a fixed rate that is incorporated into the EURTOLUF*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Luxembourg francs.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoluf6.;`	`    40`
	`put amount eurtoluf12.2;`	`       40.34`
`1234.56`	`put amount eurtoluf6.;`	` 49802`
	`put amount eurtoluf12.2;`	`    49802.03`
`12345`	`put amount eurtoluf6.;`	`497996`
	`put amount eurtoluf12.2;`	`   497996.07`

See Also

Formats:
"EURFRLUF*w.d* Format" on page 102
Functions:
"EUROCURR Function" on page 209

EURTONLG*w.d* Format

Converts an amount from euros to Dutch guilders

Category: Currency Conversion

Alignment: right

Syntax

EURTONLG*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTONLG*w.d* format converts an amount in euros to an amount in Dutch guilders. The conversion rate is a fixed rate that is incorporated into the EURTONLG*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Dutch guilders.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtonlg6.;`	`     2`
	`put amount eurtonlg12.2;`	`        2.20`
`1234.56`	`put amount eurtonlg6.;`	`  2721`
	`put amount eurtonlg12.2;`	`     2720.61`
`12345`	`put amount eurtonlg6.;`	` 27205`
	`put amount eurtonlg12.2;`	`    27204.80`

See Also

Formats:

"EURFRNLG*w.d* Format" on page 103

Functions:

"EUROCURR Function" on page 209

EURTONOK*w.d* Format

Converts an amount from euros to Norwegian krone

Category: Currency Conversion

Alignment: right

Syntax

EURTONOK*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTONOK*w.d* format converts an amount in euros to an amount in Norwegian krone. The conversion rate is a changeable rate that is incorporated into the EURTONOK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Norwegian krone.

Amounts	Statements	Results
		`----+----1----2`
1	`put amount eurtonok6.;`	`     9`
	`put amount eurtonok12.2;`	`        9.20`
1234.56	`put amount eurtonok6.;`	` 11355`
	`put amount eurtonok12.2;`	`    11355.11`
12345	`put amount eurtonok6.;`	`113546`
	`put amount eurtonok12.2;`	`   113545.61`

See Also

Formats:
  "EURFRNOK*w.d* Format" on page 104
Functions:
  "EUROCURR Function" on page 209

EURTOPLZ*w.d* Format

Converts an amount from euros to Polish zlotys

Category: Currency Conversion

Alignment: right

Syntax

EURTOPLZ*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOPLZ*w.d* format converts an amount in euros to an amount in Polish zlotys. The conversion rate is a changeable rate that is incorporated into the EURTOPLZ*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Polish zlotys.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoplz6.;`	`4`
	`put amount eurtoplz12.2;`	`4.20`
`1234.56`	`put amount eurtoplz6.;`	`5185`
	`put amount eurtoplz12.2;`	`5185.15`
`12345`	`put amount eurtoplz6.;`	`51849`
	`put amount eurtoplz12.2;`	`51849.00`

See Also

Formats:
"EURFRPLZ*w.d* Format" on page 105

Functions:
"EUROCURR Function" on page 209

EURTOPTE*w.d* Format

Converts an amount from euros to Portuguese escudos

Category: Currency Conversion

Alignment: right

Syntax

EURTOPTE*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOPTE*w.d* format converts an amount in euros to an amount in Portuguese escudos. The conversion rate is a fixed rate that is incorporated into the EURTOPTE*w.d* format and the EUROCURR function. For more information about

European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Portuguese escudos.

Amounts	Statements	Results
		`----+----1----2`
1	`put amount eurtopte8.;`	`     200`
	`put amount eurtopte12.2;`	`      200.48`
1234.56	`put amount eurtopte8.;`	`  247507`
	`put amount eurtopte12.2;`	`   247507.06`
12345	`put amount eurtopte8.;`	` 2474950`
	`put amount eurtopte12.2;`	`  2474950.29`

See Also

Formats:

"EURFRPTE*w.d* Format" on page 107

Functions:

"EUROCURR Function" on page 209

EURTOROL*w.d* Format

Converts an amount from euros to Romanian lei

Category: Currency Conversion

Alignment: right

Syntax

EURTOROL*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOROL*w.d* format converts an amount in euros to an amount in Romanian lei. The conversion rate is a changeable rate that is incorporated into the EURTOROL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Romanian lei.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtorol6.;`	`    14`
	`put amount eurtorol12.2;`	`       13.71`
`1234.56`	`put amount eurtorol6.;`	` 16926`
	`put amount eurtorol12.2;`	`    16925.82`
`12345`	`put amount eurtorol6.;`	`169250`
	`put amount eurtorol12.2;`	`   169249.95`

See Also

Formats:

"EURFRROL*w.d* Format" on page 108

EURTORUR*w.d* Format

Converts an amount from euros to Russian rubles

Category: Currency Conversion

Alignment: right

Syntax

EURTORUR*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTORUR*w.d* format converts an amount in euros to an amount in Russian rubles. The conversion rate is a changeable rate that is incorporated into the EURTORUR*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Russian rubles.

Amounts	Statements	Results
		`----+----1----2`
1	`put amount eurtorur6.;`	`    20`
	`put amount eurtorur12.2;`	`       19.77`
1234.56	`put amount eurtorur6.;`	` 24405`
	`put amount eurtorur12.2;`	`    24404.78`
12345	`put amount eurtorur6.;`	`244036`
	`put amount eurtorur12.2;`	`   244035.96`

See Also

Formats:
“EURFRRUR*w.d* Format” on page 109
Functions:
“EUROCURR Function” on page 209

EURTOSEK*w.d* Format

Converts an amount from euros to Swedish kronor

Category: Currency Conversion

Alignment: right

Syntax

EURTOSEK*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOSEK*w.d* format converts an amount in euros to an amount in Swedish kronor. The conversion rate is a changeable rate that is incorporated into the EURTOSEK*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Swedish kronor.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtosek6.;`	`     9`
	`put amount eurtosek12.2;`	`        9.37`
`1234.56`	`put amount eurtosek6.;`	` 11563`
	`put amount eurtosek12.2;`	`    11562.78`
`12345`	`put amount eurtosek6.;`	`115622`
	`put amount eurtosek12.2;`	`   115622.16`

See Also

Formats:

"EURFRSEK*w.d* Format" on page 110

Functions:

"EUROCURR Function" on page 209

EURTOSIT*w.d* Format

Converts an amount from euros to Slovenian tolars

Category: Currency Conversion

Alignment: right

Syntax

EURTOSIT*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOSIT*w.d* format converts an amount in euros to an amount in Slovenian tolars. The conversion rate is a changeable rate that is incorporated into the EURTOSIT*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Slovenian tolars.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtosit8.;`	`     191`
	`put amount eurtosit14.2;`	`        191.00`
`1234.56`	`put amount eurtosit8.;`	`  235801`
	`put amount eurtosit14.2;`	`     235800.96`
`12345`	`put amount eurtosit8.;`	` 2357895`
	`put amount eurtosit14.2;`	`    2357895.00`

See Also

Formats:
"EURFRSIT*w.d* Format" on page 111
Functions:
"EUROCURR Function" on page 209

EURTOTRL*w.d* Format

Converts an amount from euros to Turkish liras

Category: Currency Conversion

Alignment: right

Syntax

EURTOTRL*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOTRL*w.d* format converts an amount in euros to an amount in Turkish liras. The conversion rate is a changeable rate that is incorporated into the EURTOTRL*w.d* format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see "European Currency Conversion" on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Turkish liras.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtotrl8.;`	`337`
	`put amount eurtotrl14.2;`	`336.91`
`1234.56`	`put amount eurtotrl8.;`	`415938`
	`put amount eurtotrl14.2;`	`415938.08`
`12345`	`put amount eurtotrl8.;`	`4159179`
	`put amount eurtotrl14.2;`	`4159178.64`

See Also

Formats:
"EURFRTRL*w.d* Format" on page 113

Functions:
"EUROCURR Function" on page 209

EURTOYUD*w.d* Format

Converts an amount from euros to Yugoslavian dinars

Category: Currency Conversion

Alignment: right

Syntax

EURTOYUD*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

d
optionally specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURTOYUD*w.d* format converts an amount in euros to an amount in Yugoslavian dinars. The conversion rate is a changeable rate that is incorporated into the EURTOYUD*w.d* format and the EUROCURR function. For more information about

European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 52.

Examples

The following table shows input values in euros, SAS statements, and the conversion results in Yugoslavian dinars.

Amounts	Statements	Results
		`----+----1----2`
`1`	`put amount eurtoyud6.;`	`    13`
	`put amount eurtoyud12.2;`	`       13.06`
`1234.56`	`put amount eurtoyud6.;`	` 16129`
	`put amount eurtoyud12.2;`	`    16128.79`
`12345`	`put amount eurtoyud6.;`	`161280`
	`put amount eurtoyud12.2;`	`   161280.02`

See Also

Formats:
: “EURFRYUD*w.d* Format” on page 114

Functions:
: “EUROCURR Function” on page 209

HDATE*w*. Format

Writes date values in the form *yyyy mmmmm dd* where *dd* is the day-of-the-month, *mmmmm* represents the month’s name in Hebrew, and *yyyy* is the year

Category: Date and Time

Alignment: right

Syntax

HDATE*w*.

Syntax Description

w
: specifies the width of the output field.

Note: Use widths 9, 11, 15, or 17 for the best view. △

Default: 17

Range: 9–17

Details

The HDATE*w*. format writes the SAS date value in the form *yyyy mmmmm dd*:

yyyy
: is the year

mmmmm
: is the Hebrew name of the month

dd
: is the day-of-the-month

Examples

The following example uses the input value of 15780, which is the SAS date of March 16, 2003.

Statements	Results
	`----+----1----+----2----+`
`put day hdate9.;`	`03 מרץ 16`
`put day hdate11.;`	`2003 מרץ 16`
`put day hdate17.;`	`2003 מרץ 16`

See Also

Formats:

"HEBDATE*w*. Format" on page 148

HEBDATE*w*. Format

Writes date values according to the Jewish calendar

Category: Date and Time

Alignment: right

Syntax

HEBDATE*w*.

Syntax Description

w

specifies the width of the output field.

Default: 16

Range: 7–24

Details

The HEBDATE*w*. format writes the SAS date value according to the Jewish calendar. The date is written in one of the following formats:

long	ראשון י' אדר ה'תשס"ג
default	י' אדר תשס"ג
short	י'/ו'/תשס"ג

Examples

The following example uses the input value of 15780, which is the SAS date of March 16, 2003.

Statements	Results
	`----+----1----+`
`put day hebdate13.;`	י"ב/ו2'/תשס"ג
`put day hebdate16.;`	י"ב אדר-ב' תשס"ג
`put day hebdate24.;`	ראשון י"ב אדר-ב' ה'תשס"ג

See Also

Informats:

"HDATE*w*. Format" on page 147

$KANJI*w*. Format

Adds shift-code data to DBCS data

Category: DBCS

Alignment: left

Syntax

$KANJI*w*.

Syntax Description

w

specifies the width of the output field.

Restriction: The width must be an even number. If it is an odd number, it is truncated.

Range: The minimum width of the format is **`2 + (length of shift code used on the current DBCSTYPE= setting)*2.`**

See Also

Formats:

"$KANJIX*w*. Format" on page 150

Informats:

"$KANJI*w*. Informat" on page 263

"$KANJIX*w*. Informat" on page 263

System Options:

"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351

$KANJIX*w*. Format

Removes shift-code data from DBCS data

Category: DBCS

Alignment: left

Syntax

$KANJIX*w*.

Syntax Description

w

specifies the width of the output field.

Restriction: The width must be an even number. If it is an odd number, it is truncated.

Range: The minimum width of the format is 2.

Details

The input data length must be **`2 + (SO/SI length)*2`**. The data must start with SO and end with SI, unless single-byte data is returned. This format always returns a blank for DBCSTYPE data that does not use a shift-code mechanism.

See Also

Formats:

"$KANJI*w*. Format" on page 149

Informats:

"$KANJI*w*. Informat" on page 263

"$KANJIX*w*. Informat" on page 263

System Options:

"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351

$LOGVS*w*. Format

Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order

Category: BIDI text handling

Alignment: left

Syntax

$LOGVS*w*.

Syntax Description

w

specifies the width of the output field.

Default: 200

Range: 1–32000

Details

The $LOGVS*w*. format is used when you store logical-ordered text on a visual server.

Note: If the $LOGVS*w*. format is not accessible, then the Hebrew portion of the data will be reversed. △

Comparisons

The $LOGVS*w*. format performs processing that is the opposite of the $LOGVSR*w*. format.

Examples

The following example uses the input value of "טיסה flight".

Statements	Results
	`----+----1----+----2----+`
`put text $logvs12.;`	הסיט **flight**

See Also

Formats:

“$LOGVSR*w*. Format” on page 152

Informats:

“$LOGVSR*w*. Informat” on page 265
“$LOGVS*w*. Informat” on page 264

$LOGVSR*w*. Format

Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order

Category: BIDI text handling

Alignment: left

Syntax

$LOGVSR*w*.

Syntax Description

w

specifies the width of the output field.

Default: 200

Range: 1–32000

Details

The $LOGVSR*w*. format is used when you store logical-ordered text on a visual server. The Hebrew portion of the text will be reversed if the $LOGVS*w*. format is not on the server.

Comparisons

The $LOGVSR*w*. format performs processing that is opposite of the $LOGVS*w*. format.

Examples

The following example uses the input value of “טיסה flight”.

Statements	Results
	`----+----1----+`
`put text $logvsr12.;`	**`flight`** טיסה

See Also

Formats:
: "$LOGVS*w*. Format" on page 151

Informats:
: "$LOGVS*w*. Informat" on page 264
: "$LOGVSR*w*. Informat" on page 265

MINGUO*w*. Format

Writes date values as Taiwanese dates in the form *yyymmdd*

Category: Date and Time

Alignment: left

Syntax

MINGUO*w*.

Syntax Description

w
specifies the width of the output field.

Default: 8

Range: 1–10

Details

The MINGUO*w*. format writes SAS date values in the form *yyyymmdd*, where

yyyy
is an integer that represents the year.

mm
is an integer that represents the month.

dd
is an integer that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates prior to 1912 appear as a series of asterisks. Year values do not roll around after 100 years; instead, they continue to increase.

Examples

The example table uses the following input values:

1 12054 is the SAS date value that corresponds to January 1, 1993.

2 18993 is the SAS date value that corresponds to January 1, 2012.

3 -20088 is the SAS date value that corresponds to January 1, 1905.

Statements	Results
	`----+----1`
`put date minguo10.;`	`0082/01/01`
	`0101/01/01`
	`**********`

See Also

Informats:

"MINGUO*w*. Informat" on page 266

NENGO*w*. Format

Writes date values as Japanese dates in the form *e.yymmdd*

Category: Date and Time

Alignment: left

Syntax

NENGO*w*.

Syntax Description

w

specifies the width of the output field.

Default: 10

Range: 2–10

Details

The NENGO*w*. format writes SAS date values in the form *e.yymmdd*, where

e
: is the first letter of the name of the emperor (Meiji, Taisho, Showa, or Heisei).

yy
: is an integer that represents the year.

mm
: is an integer that represents the month.

dd
: is an integer that represents the day of the month.

If the width is too small, SAS omits the period.

Examples

The example table uses the input value of 15342, which is the SAS date value that corresponds to January 2, 2002.

Statements	Results
	`----+----1`
`put date nengo3.;`	`H14`
`put date nengo6.;`	`H14/01`
`put date nengo8.;`	`H.140102`
`put date nengo9.;`	`H14/01/02`
`put date nengo10.;`	`H.14/01/02`

See Also

Informats:
"NENGO*w*. Informat" on page 267

NLDATE*w*. Format

Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date

Category: Date and Time

Alignment: left

Syntax

NLDATE*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date to fit the format width.

Default: 20

Range: 10–200

Comparisons

NLDATE*w*. is similar to DATE*w*. and WORDDATE*w*. except that NLDATE*w*. is locale specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

Statements	Results
	`----+----1----+----2`
`options locale=English_UnitedStates;`	
`put day nldate.;`	`February 24, 2003`
`options locale=German_Germany;`	
`put day nldate.;`	`24. Februar 2003`

See Also

Formats:

"NLDATEMN*w*. Format" on page 156
"NLDATEW*w*. Format" on page 157
"NLDATEWN*w*. Format" on page 158

NLDATEMN*w*. Format

Converts a SAS date value to the date value of the specified locale, and then writes the value as the name-of-the-month

Category: Date and Time

Alignment: left

Syntax

NLDATEMN*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the name-of-the-month to fit the format width.

Default: 10

Range: 4–200

Comparisons

NLDATEMN*w*. is similar to MONNAME*w*. except that NLDATEMN*w*. is locale specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

Statements	Results
	`----+----1`
`options locale=English_UnitedStates;`	
`put month nldatemn.;`	`February`
`options locale=German_Germany;`	
`put month nldatemn.;`	`Februar`

See Also

Formats:

"NLDATE*w*. Format" on page 155
"NLDATEW*w*. Format" on page 157
"NLDATEWN*w*. Format" on page 158

NLDATEW*w*. Format

Converts a SAS date value to the date value of the specified locale, and then writes the value as the date and the day-of-the-week

Category: Date and Time

Alignment: left

Syntax

NLDATEW*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date and the day-of-the-week to fit the format width.

Default: 20

Range: 10–200

Comparisons

NLDATEW*w*. is similar to WEEKDATE*w*. except that NLDATEW*w*. is locale specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

Statements	Results
	`----+----1----+----2`
`options locale=English_UnitedStates;`	
`put date nldatew.;`	`Sun, Feb 24, 03`
`options locale=German_Germany;`	
`put date nldatew.;`	`So, 24. Feb 03`

See Also

Formats:

"NLDATE*w*. Format" on page 155
"NLDATEMN*w*. Format" on page 156
"NLDATEWN*w*. Format" on page 158

NLDATEWN*w*. Format

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the day-of-the-week

Category: Date and Time

Alignment: left

Syntax

NLDATEWN*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the day-of-the-week to fit the format width.

Default: 10

Range: 4–200

Comparisons

NLDATEWN*w*. is similar to DOWNAME*w*. except that NLDATEWN*w*. is locale specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

Statements	Results
	`----+----1`
`options locale=English_UnitedStates;`	
`put date nldatewn.;`	`Sunday`
`options locale=German_Germany;`	
`put date nldatewn.;`	`Sonntag`

See Also

Formats:

"NLDATE*w*. Format" on page 155

"NLDATEMN*w*. Format" on page 156

"NLDATEW*w*. Format" on page 157

NLDATM*w*. Format

Converts a SAS date-time value to the date-time value of the specified locale, and then writes the value as a date-time

Category: Date and Time

Alignment: left

Syntax

NLDATM*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date-time value to fit the format width.

Default: 30

Range: 10–200

Comparisons

The NLDATM*w*. format is similar to the DATETIME*w*. format except that the NLDATM*w*. format is locale specific.

Examples

These examples use the input value of 1361709583, which is the SAS datetime value that corresponds to 12:39:43 p.m. on February 24, 2003.

Statements	Results
	`----+----1----+----2----+----3`
`options locale=English_UnitedStates;` `put day nldatm.;`	`24Feb03:12:39:43`
`options locale=German_Germany;` `put day nldatm.;`	`24. Februar 2003 12.39 Uhr`

See Also

Formats:

"NLDATMAP*w*. Format" on page 160
"NLDATMTM*w*. Format" on page 161
"NLDATMW*w*. Format" on page 162

NLDATMAP*w*. Format

Converts a SAS date-time value to the date-time value of the specified locale, and then writes the value as a date-time with a.m. or p.m.

Category: Date and Time
Alignment: left

Syntax

NLDATMAP*w*.

Syntax Description

w

specifies the width of the output field. If necessary, SAS abbreviates the date-time value to fit the format width.

Default: 32

Range: 16–200

Comparisons

The NLDATMAP*w*. format is similar to DATEAMPM*w*. except that the NLDATMAP*w*. format is locale specific.

Examples

These examples use the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003.

Statements	Results
	`----+----1----+----2----+----3`
`options locale=English_UnitedStates;`	
`put event nldatmap.;`	`February 24, 2003 12:39:43 PM`
`options locale=Spanish_Mexico;`	
`put event nldatmap.;`	`01 de enero de 2003 01:24:35 PM`

See Also

Formats:

“NLDATM*w*. Format” on page 159
“NLDATMTM*w*. Format” on page 161
“NLDATMW*w*. Format” on page 162

NLDATMTM*w*. Format

Converts the time portion of a SAS date-time value to the time-of-day value of the specified locale, and then writes the value as a time-of-day

Category: Date and Time

Alignment: left

Syntax

NLDATMTM*w*.

Syntax Description

w

specifies the width of the output field.

Default: 16

Range: 16–200

Comparisons

The NLDATMTM*w*. format is similar to the TOD*w*. format except that the NLDATMTM*w*. format is locale specific.

Examples

These examples use the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003.

Statements	Results
	`----+----1`
`options locale=English_UnitedStates;`	
`put event nldatmtm.;`	`12:39:43`
`options locale=German_Germany;`	
`put event nldatmtm.;`	`12.39 Uhr`

See Also

Formats:

"NLDATM*w*. Format" on page 159
"NLDATMAP*w*. Format" on page 160
"NLDATMW*w*. Format" on page 162

NLDATMW*w*. Format

Converts a SAS date value to a date-time value of the specified locale, and then writes the value a day-of-week and date-time

Category: Date and Time
Alignment: left

Syntax

NLDATMW*w*.

Syntax Description

w
specifies the width of the output field. If necessary, SAS abbreviates the day-of-week and date-time to fit the format width.
Default: 30
Range: 16–200

Comparisons

The NLDATMW*w*. format is similar to the TWMDY*w*. format except that the NLDATMW*w*. format is locale specific.

Examples

These examples use the input value of 1361709583, which is the SAS date-time value that corresponds to 12:39:43 p.m. on February 24, 2003.

Statements	Results
	`----+----1----+----2----+----3`
`options locale=English_UnitedStates;`	
`put event nldatmw.;`	`Sun, Feb 24, 2003 12:39:43`
`options locale=German_Germany;`	
`put event nldatmw.;`	`So, 24. Feb 2003 12.39 Uhr`

See Also

Formats:
"NLDATM*w*. Format" on page 159
"NLDATMAP*w*. Format" on page 160
"NLDATMTM*w*. Format" on page 161

NLMNY*w.d* Format

Writes the monetary format of the local expression in the specified locale using local currency

Category: Numeric

Alignment: left

Syntax

NLMNY*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 9

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal points, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMNY*w.d* format reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNY*w.d* format writes numeric values by using the currency symbol, the thousands separator, and the decimal separator that is used by the locale.

Note: The NLMNY*w.d* format does not convert currency format, therefore, the value of the formatted number should equal the currency of the current locale value. △

Comparisons

The NLMNY*w.d* format writes the monetary format of the local expression in the specified locale using local currency. The NLMNYI*w.d* format writes the monetary format of the international expression in the specified locale. Typically the NLMNY*w.d* format and the NLMNYI*w.d* format return the same results, however some of the locales produce different results for the local and international expressions.

The NLMNY*w.d* format is similar to the DOLLAR*w.d* format except that the NLMNY*w.d* format is locale-specific.

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmny32.2);
y=put(-1234.56789,dollar32.2);
```

Statements	Results
	`----+----1----+`
`put x=;`	`($1,234.57)`
`put y=;`	`$-1,234.57`

See Also

Formats:
"NLMNYI*w.d* Format" on page 165
Informats:
"NLMNYI*w.d* Informat" on page 270
"NLMNYI*w.d* Informat" on page 272

NLMNYI*w.d* Format

Writes the monetary format of the international expression in the specified locale

Category: Numeric

Alignment: left

Syntax

NLMNYI*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 9

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMNYI*w.d* format reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLMNYI*w.d* format writes numeric values by using the currency symbol, the thousands separator, and the decimal separator that is used by the locale.

Note: The NLMNYI*w.d* format does not convert currency format, therefore, the value of the formatted number should equal the currency of the current locale value. △

Comparisons

The NLMNY*w.d* format writes the monetary format of the local expression in the specified locale using local currency. The NLMNYI*w.d* format writes the monetary format of the international expression in the specified locale. Typically the NLMNY*w.d* format and the NLMNYI*w.d* format return the same results, however, some of the locales produce different results for the local and international expressions.

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnyi32.2);
y=put(-1234.56789,nlmny32.2);
z=put(-1234.56789,dollar32.2);
```

Statements	Results
	`----+----1----+`
`put x=;`	**`(USD1,234.57)`**
`put y=;`	**`($1,234.57)`**
`put z=;`	**`$-1,234.57`**

See Also

Formats:

“NLMNY*w.d* Format” on page 163

Informats:

“NLMNY*w.d* Informat” on page 270

“NLMNYI*w.d* Informat” on page 272

NLNUM*w.d* Format

Writes the numeric format of the local expression in the specified locale

Category: Numeric

Alignment: left

Syntax

NLNUM*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMUM*w.d* format reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUM*w.d* format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

Comparisons

The NLNUM*w.d* format writes the numeric format of the local expression in the specified locale. The NLNUMI*w.d* format writes the numeric format of the international expression in the specified locale. Typically the NLNUM*w.d* format and the NLNUMI*w.d* format return the same results, however some of the locales produce different results for the local and international expressions.

The NLNUM*w.d* format is similar to the COMMA*w.d* format except that the NLNUM*w.d* format is locale specific.

Examples

```
x=put(-1234356.7891,nlnum32.2);
```

Statements	Results
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`put x=;`	`-1,234,356.79`
`options LOCALE=German_Germany;`	
`put x=;`	`-1.234.356,79`

See Also

Formats:

"NLNUMI*w.d* Format" on page 168

Informats:

"NLNUM*w.d* Informat" on page 273

"NLNUMI*w.d* Informat" on page 274

NLNUMI*w.d* Format

Writes the numeric format of the international expression in the specified locale

Category: Numeric

Alignment: left

Syntax

NLNUMI*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal points, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMUMI*w.d* format reads integer binary (fixed-point) values, including negative values that are represented in two's-complement notation. The NLNUMI*w.d* format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

Comparisons

The NLNUM*w.d* format writes the numeric format of the local expression in the specified locale. The NLNUMI*w.d* format writes the numeric format of the international expression in the specified locale. Typically the NLNUM*w.d* format and the NLNUMI*w.d* format return the same results, however, some of the locales produce different results for the local and international expressions.

The NLNUMI*w.d* format is similar to the COMMA*w.d* format except that the NLNUMI*w.d* format is locale specific.

Examples

```
x=put(-1234356.7891,nlnumi32.2);
```

Statements	Results
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`put x=;`	**`-1,234,356.79`**
`options LOCALE=German_Germany;`	
`put x=;`	**`-1.234.356,79`**

See Also

Formats:
  "NLNUM*w.d* Format" on page 166
Informats:
  "NLNUM*w.d* Informat" on page 273
  "NLNUMI*w.d* Informat" on page 274

NLPCT*w.d* Format

Writes percentage data of the local expression in the specified locale

Category: Numeric

Alignment: left

Syntax

NLPCT*w.d*

Syntax Description

w
specifies the width of the output field.

Default: 6

Range: 1–32

d
optionally specifies to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Comparisons

The NLPCT*w.d* format writes percentage data of the local expression in the specified locale. The NLPCTI*w.d* format writes percentage data of the international expression in the specified locale. Typically the NLPCT*w.d* format and the NLPCTI*w.d* format return the same results, however some of the locales produce different results for the local and international expressions.

The NLPCT*w.d* format is similar to the PERCENT*w.d* format except the NLPCT*w.d* format is locale specific.

Examples

```
x=put(-12.3456789,nlpct32.2);
y=put(-12.3456789,nlpcti32.2);
z=put(-12.3456789,percent32.2);
```

Statements	Results
	`----+----1`
`options LOCALE=English_UnitedStates;`	
`put x=;`	`-1,234.57%`
`put y=;`	`-1,234.57%`
`put z=;`	`(1234.57%)`
`options LOCALE=German_Germany;`	
`put x=;`	`-1.234,57%`
`put y=;`	`-1,234.57%`
`put z=;`	`(1234.57%)`

See Also

Formats:

“NLPCTI*w.d* Format” on page 170

Informats:

“NLPCT*w.d* Informat” on page 275
“NLPCTI*w.d* Informat” on page 277

NLPCTI*w.d* Format

Writes percentage data of the international expression in the specified locale

Category: Numeric

Alignment: left

Syntax

NLPCTI*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 6

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Comparisons

The NLPCT*w.d* format writes percentage data of the local expression in the specified locale. The NLPCTI*w.d* format writes percentage data of the international expression in the specified locale. Typically the NLPCT*w.d* format and the NLPCTI*w.d* format return the same results, however some of the locales produce different results for the local and international expressions.

The NLPCT*w.d* format is similar to the PERCENT*w.d* format except the NLPCT*w.d* format is locale specific.

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-12.3456789,nlpcti32.2);
y=put(-12.3456789,percent32.2);
```

Statements	Results
	`----+----1`
`put x=;`	`-1,234.57%`
`put y=;`	`(1234.57)`

See Also

Formats:

"NLPCT*w.d* Format" on page 169

Informats:

"NLPCT*w.d* Informat" on page 275

"NLPCTI*w.d* Informat" on page 277

NLTIME*w*. Format

Converts a SAS time value to the time value of the specified locale, and then writes the value as a time value

Category: Date and Time

Alignment: left

Syntax

NLTIME*w*.

Syntax Description

w
specifies the width of the input field.

Default: 20

Range: 10–200

Comparisons

The NLTIME*w*. format is similar to the TIME*w*. format except that the NLTIME*w*. format is locale specific.

Examples

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`put time nltime.;`	`4:24:43`
`options locale=German_Germany;`	
`put time nltime.;`	`16.24`

See Also

Formats:

"NLTIMAP*w*. Format" on page 173

NLTIMAP*w*. Format

Converts a SAS time value to the time value of a specified locale, and then writes the value as a time value with a.m. or p.m.

Category: Date and Time

Alignment: left

Syntax

NLTIMAP*w*.

Syntax Description

w
specifies the width of the output field.

Default: 10

Range: 4–200

Comparisons

The NLTIMAP*w*. format is similar to the TIMEAMPM*w*. format except that the NLTIMAP*w*. format is locale specific.

Examples

These examples use the input value of 59083, which is the SAS date-time value that corresponds to 4:24:43 p.m.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`put time nltimap.;`	`4:24:43 PM`
`options locale=German_Germany;`	
`put time nltimap.;`	`16.24 Uhr`

See Also

Formats:
"NLTIME*w*. Format" on page 172

$UCS2B*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 16-bit, UCS2, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS2B*w*.

Syntax Description

w
specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32767

Details

The $UCS2B*w*. format writes a character string in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparison

The $UCS2B*w*. format performs processing that is the opposite of the $UCS2BE*w*. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Results
	`----+----1`
x = 大;	
`put x $ucs2b2.;`	**`'5927'x (binary)`**

See Also

Formats:
- “$UCS2L*w*. Format” on page 176
- “$UCS2X*w*. Format” on page 178
- “$UTF8X*w*. Format” on page 195
- “$UCS2BE*w*. Format” on page 175

Informats:
- “$UCS2B*w*. Informat” on page 282
- “$UCS2BE*w*. Informat” on page 283
- “$UCS2L*w*. Informat” on page 284
- “$UCS2X*w*. Informat” on page 286
- “$UTF8X*w*. Informat” on page 300

$UCS2BE*w*. Format

Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS2BE*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS2BE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparison

The $UCS2BE*w*. format performs processing that is the opposite of the $UCS2B*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Results
	`----+----1`
`x ='592700410042'x;`	
`put x $ucs2be4.;`	大AB

See Also

Formats:
- "$UCS2B*w*. Format" on page 174

Informats:
- "$UCS2B*w*. Informat" on page 282
- "$UCS2BE*w*. Informat" on page 283

$UCS2L*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 16-bit, UCS2, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS2L*w*.

Syntax Description

w
: specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

  Default: 8

  Range: 2–32767

Details

The $UCS2L*w*. format writes a character string in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparison

The $UCS2L*w*. format performs processing that is the opposite of the $UCS2LE*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x = 大;`	
`put x $ucs2l2.;`	`'2759'x (binary)`

See Also

Formats:

- "$UCS2B*w*. Format" on page 174
- "$UCS2LE*w*. Format" on page 177
- "$UCS2X*w*. Format" on page 178
- "$UTF8X*w*. Format" on page 195

Informats:

- "$UCS2B*w*. Informat" on page 282
- "$UCS2L*w*. Informat" on page 284
- "$UCS2LE*w*. Informat" on page 285
- "$UCS2X*w*. Informat" on page 286
- "$UTF8X*w*. Informat" on page 300

$UCS2LE*w*. Format

Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS2LE*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS2LE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparison

The $UCS2LE*w*. format performs processing that is the opposite of the $UCS2L*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='275941004200'x;`	
`put x $ucs2le4.;`	大AB

See Also

Formats:

"$UCS2L*w*. Format" on page 176

Informats:

"$UCS2L*w*. Informat" on page 284

"$UCS2LE*w*. Informat" on page 285

$UCS2X*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 16-bit, UCS2, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS2X*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32767

Details

The $UCS2X*w*. format writes a character string in 16-bit, UCS2 (universal character set code in two octets), Unicode encoding, by using byte order that is native to the operating environment.

Comparison

The $UCS2X*w*. format performs processing that is the opposite of the $UCS2XE*w*. format. If you are exchanging data within the same operating environment, use the $UCS2X*w*. format. If you are exchanging data with a different operating environment, use the $UCS2B*w*. format or $UCS2L*w*. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x = 大;`	
`put x $ucs2x2.;`	`'5927'x (binary) or '2759'x (little endian)`

See Also

Formats:

“$UCS2B*w*. Format” on page 174
“$UCS2XE*w*. Format” on page 180
“$UCS2L*w*. Format” on page 176
“$UTF8X*w*. Format” on page 195

Informats:

“$UCS2B*w*. Informat” on page 282
“$UCS2L*w*. Informat” on page 284
“$UCS2X*w*. Informat” on page 286
“$UCS2XE*w*. Informat” on page 287
“$UTF8X*w*. Informat” on page 300

$UCS2XE*w*. Format

Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS2XE*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS2XE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 16-bit, UCS2 (universal character set code in two octets), Unicode encoding.

Comparison

The $UCS2XE*w*. format performs processing that is the opposite of the $UCS2X*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='e5a4a7'x; /* Japanese '大' in UTF8 */;`	
`put x $utf8xe10.;`	大

See Also

Formats:

"$UCS2X*w*. Format" on page 178

Informats:

"$UCS2X*w*. Informat" on page 286
"$UCS2XE*w*. Informat" on page 287

$UCS4B*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in big-endian, 32-bit, UCS4, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS4B*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32767

Details

The $UCS4B*w*. format writes a character string in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparison

The $UCS4B*w*. format performs processing that is the opposite of the $UCS4BE*w*. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='太';`	
`put x $ucs4b4.;`	`'00005927'x (binary)`

See Also

Formats:

"$UCS2L*w*. Format" on page 176
"$UCS2X*w*. Format" on page 178
"$UCS4BE*w*. Format" on page 182
"$UCS4L*w*. Format" on page 183
"$UCS4X*w*. Format" on page 186
"$UTF8X*w*. Format" on page 195

Informats:

"$UCS2B*w*. Informat" on page 282
"$UCS2L*w*. Informat" on page 284
"$UCS2X*w*. Informat" on page 286
"$UCS4B*w*. Informat" on page 288
"$UCS4L*w*. Informat" on page 289
"$UCS4X*w*. Informat" on page 290
"$UTF8X*w*. Informat" on page 300

$UCS4BE*w*. Format

Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS4BE*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS4BE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in big-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparison

The $UCS4BE*w*. format performs processing that is the opposite of the $UCS4B*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='0000592700000041000000042'x;`	
`put x $ucs4be4.;`	大AB

See Also

Formats:

“$UCS4B*w*. Format” on page 181

Informats:

“$UCS4B*w*. Informat” on page 288

$UCS4L*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in little-endian, 32-bit, UCS4, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS4L*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32767

Details

The $UCS4L*w*. format writes a character string in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding. It processes character strings that are in the encoding of the current SAS session.

Comparisons

The $UCS4L*w*. format performs processing that is the opposite of the $UCS4LE*w*. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x = '大';`	
`put x $ucs4l4.;`	`'27590000'x (binary)`

See Also

Formats:

Informats:

$UCS4LE*w*. Format

Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS4LE*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS4LE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in little-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparison

The $UCS4LE*w*. format performs processing that is the opposite of the $UCS4L*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='275900004100000042000000'x;`	
`put x $ucs4le4.;`	`大AB`

See Also

Formats:

"$UCS4L*w*. Format" on page 183

Informats:

"$UCS4L*w*. Informat" on page 289

$UCS4X*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in native-endian, 32-bit, UCS4, Unicode encoding

Category: Character

Alignment: left

Syntax

$UCS4X*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32767

Details

The $UCS4X*w*. format writes a character string in 32-bit, UCS4 (universal character set code in two octets), Unicode encoding, by using byte order that is native to the operating environment.

Comparisons

The $UCS4X*w*. format performs processing that is the opposite of the $UCS4XE*w*. format. If you are exchanging data within the same operating environment, use the $UCS4X*w*. format. If you are exchanging data with a different operating environment, use the $UCS4B*w*. format or $UCS4L*w*. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

Statements	Results
	`----+----1`
`x = '大';`	
`put x $ucs4x4.;`	`'00005927'x (binary) or '27590000'x (little endian)`

See Also

Formats:
- "$UCS2L*w*. Format" on page 176
- "$UCS4XE*w*. Format" on page 187
- "$UCS2X*w*. Format" on page 178
- "$UCS4B*w*. Format" on page 181
- "$UCS4L*w*. Format" on page 183
- "$UTF8X*w*. Format" on page 195

Informats:
- "$UCS2B*w*. Informat" on page 282
- "$UCS2L*w*. Informat" on page 284
- "$UCS2X*w*. Informat" on page 286
- "$UCS4B*w*. Informat" on page 288
- "$UCS4L*w*. Informat" on page 289
- "$UCS4X*w*. Informat" on page 290
- "$UTF8X*w*. Informat" on page 300

$UCS4XE*w*. Format

Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UCS4XE*w*.

Syntax Description

w
specifies the width of the output field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Details

The $UCS4XE*w*. format writes a character string in the encoding of the current SAS session. It processes character strings that are in native-endian, 32-bit, UCS4 (universal character set code in four octets), Unicode encoding.

Comparison

The $UCS4XE*w*. format performs processing that is the opposite of the $UCS4X*w*. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1`
`x ='275900004100000042000000'x;`	
`put x $ucs4be4.;`	`大AB (little endian)`

See Also

Formats:
"$UCS4X*w*. Format" on page 186
Informats:
"$UCS4X*w*. Informat" on page 290

$UESC*w*. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation

Category: Character
Alignment: left

Syntax

$UESC*w*.

Syntax Description

w
specifies the width of the input field.
Default: 8
Range: 1–32000

Details

If the characters are not available on all operating environments, for example, 0–9, a–z, A–Z, they must be represented in UESC. $UESC*w*. can be nested.

Comparisons

The $UESC*w*. format performs processing that is opposite of the $UESCE*w*. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+----2`
`x='大' ;`	
`y='u5927'`	
`z='uu5927';`	
`put x = $uesc10. ;`	`¥u5927`
`put y = $uesc10. ;`	`¥uu5927`
`put z = $uesc10. ;`	`¥uuu5927`

See Also

Formats:
"$UESCE*w*. Format" on page 189

Informats:
"$UESC*w*. Informat" on page 292
"$UESCE*w*. Informat" on page 294

$UESCE*w*. Format

Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UESCE*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8

Range: 1–32000

Details

If the data is not supported by the encoding of the current SAS session, the data will remain in UESC.

Comparisons

The $UESCE*w*. format performs processing that is the opposite of the $UESC*w*. format.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+----2`
`x=put('¥u5927',$uesce10.) ;`	x=大
`x=put('¥uu5927',$uesce10.) ;`	`x=¥u5927`
`x=put('¥uuu5927',$uesce10.) ;`	`x=¥uu5927`

See Also

Formats:

“$UESC*w*. Format” on page 188

Informats:

“$UESC*w*. Informat” on page 292

“$UESCE*w*. Informat” on page 294

$UNCR*w*. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR)

Category: Character

Alignment: left

Syntax

$UNCR*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8

Range: 1–32000

Comparison

The $UNCR*w*. format performs processing that is the opposite of the $UNCRE*w*. format.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+----2`
`x='91E5'x ; /* Japanese '大' in Shift-JIS */`	
`y='abc';`	
`put x $uncr10.;`	`&#22823`
`put y $uncr10.;`	`abc`

See Also

Formats:

“$UNCRE*w*. Format” on page 191

Informats:

“$UNCR*w*. Informat” on page 295

“$UNCRE*w*. Informat” on page 296

$UNCRE*w*. Format

Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UNCRE*w*.

Syntax Description

w
specifies the width of the output field.

Default: 8

Range: 1–32000

Details

National characters should be represented in NCR.

Comparison

The $UNCRE*w*. format performs processing that is the opposite of the $UNCR*w*. format.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1`
`x='大abc';`	
`put x $uncr10.;`	`大abc`

See Also

Formats:
"$UNCR*w*. Format" on page 190
Informats:
"$UNCR*w*. Informat" on page 295
"$UNCRE*w*. Informat" on page 296

$UPAREN*w*. Format

Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation

Category: Character

Alignment: left

Syntax

$UPAREN*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8

Range: 27–32000

Details

The character string will be encoded with parenthesis and Unicode hex representation.

Comparisons

The $UPAREN*w*. format performs processing that is the opposite of the $UPARENE*w*. format.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+----2----+----3----+`
`x='大';`	
`y='abc3';`	
`put x $uparen7.;`	`<u5927>`
`put y $uparen28.;`	`<u0061> <u0062> <u0063> <u0033>`

See Also

Formats:

“$UPARENE*w*. Format” on page 194

Informats:

“$UPAREN*w*. Informat” on page 297

“$UPARENE*w*. Informat” on page 298

$UPARENE*w*. Format

Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session

Category: Character

Alignment: left

Syntax

$UPARENE*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8

Range: 1–32000

Comparisons

The $UPARENE*w*. format performs processing that is the opposite of the $UPAREN*w*. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+`
`x='<u0061><u0062><u0063><u0033>';`	
`put x $uparene4.;`	`abc3`

See Also

Formats:

"$UPAREN*w*. Format" on page 192

Informats:

"$UPAREN*w*. Informat" on page 297

"$UPARENE*w*. Informat" on page 298

$UTF8X*w*. Format

Processes a character string that is in the encoding of the current SAS session, and then writes the character string in universal transformation format (UTF-8) encoding

Category: Character

Alignment: left

Syntax

$UTF8X*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to include all of the characters in the variable. The width of the characters will be dependent on the code point value of the individual characters.

Default: 8

Range: 2–32767

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

Statements	Results
	`----+----1`
`x ='91E5'x; ; /* Japanese '大' in Shift-JIS */`	
`put x $utf8x10.;`	`x='e5a4a7'x`

See Also

Formats:

"$UCS2B*w*. Format" on page 174

"$UCS2L*w*. Format" on page 176

"$UCS2X*w*. Format" on page 178

Informats:

"$UCS2B*w*. Informat" on page 282

"$UCS2L*w*. Informat" on page 284

"$UCS2X*w*. Informat" on page 286

$VSLOG*w*. Format

Processes a character string that is in visual order, and then writes the character string in left-to-right logical order

Category: BIDI text handling

Alignment: left

Syntax

$VSLOG*w*.

Syntax Description

w

specifies the width of the output field.

Default: 200

Range: 1–32000

Details

The $VSLOG*w*. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

Note: The $VSLOG*w*. format does not correctly process all combinations of data strings. △

Comparisons

The $VSLOG*w*. format performs processing that is opposite to the $VSLOGR*w*. format.

Examples

The following example used the input value of “ה ס י ט flight”.

Statements	Results
	`----+----1----+----2----+`
`put text $vslog12.;`	`טיסה flight`

See Also

Formats:

“$VSLOGR*w*. Format” on page 197

Informats:

"$VSLOG*w*. Informat" on page 301

"$VSLOGR*w*. Informat" on page 302

$VSLOGR*w*. Format

Processes a character string that is in visual order, and then writes the character string in right-to-left logical order

Category: BIDI text handling

Alignment: left

Syntax

$VSLOGR*w*.

Syntax Description

w

specifies the width of the output field.

Default: 200

Range: 1–32000

Details

The $VSLOGR*w*. format is used when transferring data that is stored in visual order. An example is transferring data from a UNIX server to a Windows client.

Note: The $VSLOGR*w*. format does not correctly process all combinations of data strings. △

Comparisons

The $VSLOGR*w*. format performs processing that is opposite to the $VSLOG*w*. format.

Examples

The following example uses the input value of "הסיט flight."

Statements	Results
	`----+----1----+`
`put text $logvs12;`	`flight` טיסה

See Also

Formats:
  $VSLOGw.
Informats:
  "$VSLOG*w*. Informat" on page 301
  "$VSLOGR*w*. Informat" on page 302

WEEKU*w*. Format

Writes a week number in decimal format by using the U algorithm

Category: Date and Time
Alignment: left

Syntax

WEEKU*w*.

Syntax Description

w
specifies the width of the output field.
Default: 11
Range: 3–200

Details

The WEEKU*w*. format writes a week-number format. The WEEKU*w*. format writes the various formats depending on the specified width. Algorithm U calculates the SAS date value by using the number of the week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

Refer to the following table for widths, formats, and examples:

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKV*w*. format writes the week number as a decimal number in the range 01–53, with weeks beginning on a Monday and week 1 of the year including both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKU*w*. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Examples

```
sasdate = '01JAN2003'd;
```

Statements	Results
	----+----1----+
v=put(sasdate,weeku3.);	
w=put(sasdate,weeku5.);	
x=put(sasdate,weeku7.);	
y=put(sasdate,weeku9.);	
z=put(sasdate,weeku11.);	
put v;	W00
put w;	03W00
put x;	03W0004
put y;	2003W0004
put z;	2003-W00-04

See Also

Formats:
- "WEEKV*w*. Format" on page 199
- "WEEKW*w*. Format" on page 201

Functions:
- "WEEK Function" on page 239

Informats:
- "WEEKU*w*. Informat" on page 303
- "WEEKV*w*. Informat" on page 305
- "WEEKW*w*. Informat" on page 307

WEEKV*w*. Format

Writes a week number in decimal format by using the V algorithm

Category: Date and Time

Alignment: left

Syntax

WEEKV*w*.

Syntax Description

w
specifies the width of the output field.

Default: 11

Range: 3–200

Details

The WEEKV*w*. format writes the various formats depending on the specified width. Algorithm V calculates the SAS date value, with the number-of-the-week value represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

Refer to the following table for widths, formats, and examples:

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKV*w*. format writes the week number as a decimal number in the range 01–53, with weeks beginning on a Monday and week 1 of the year including both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKU*w*. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Examples

```
sasdate='01JAN2003'd;
```

Statements	Results
	`----+----1----+`
`v=put(sasdate,weekv3.);`	
`w=put(sasdate,weekv5.);`	
`x=put(sasdate,weekv7.);`	
`y=put(sasdate,weekv9.);`	
`z=put(sasdate,weekv11.);`	
`put v;`	`W01`
`put w;`	`03W01`
`put x;`	`03W0103`
`put y;`	`2003W0103`
`put z;`	`2003-W01-03`

See Also

Formats:

“WEEKU*w*. Format” on page 198
“WEEKW*w*. Format” on page 201

Functions:

“WEEK Function” on page 239

Informats:

“WEEKU*w*. Informat” on page 303
“WEEKV*w*. Informat” on page 305
“WEEKW*w*. Informat” on page 307

WEEKW*w*. Format

Writes a week number in decimal format by using the W algorithm

Category: Date and Time

Alignment: left

Syntax

WEEKW*w*.

Syntax Description

w

specifies the width of the output field.

Default: 11

Range: 3–200

Details

The WEEKW*w*. format writes the various formats depending on the specified width. Algorithm W calculates the SAS date value using the number of the week within the year (Monday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

Refer to the following table for widths, formats, and examples:

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKV*w*. format writes the week number as a decimal number in the range 01–53. Weeks beginning on a Monday and on week 1 of the year include both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. format writes the week number of the year as a decimal number in the range 00–53, with Monday as the first day of week 1. The WEEKU*w*. format writes the week number of the year (Sunday as the first day of the week) as a decimal number in the range 0–53, with a leading zero.

Examples

```
sasdate = '01JAN2003'd;
```

Statements	Results
	`----+----1----+`
`v=put(sasdate,weekw3.);`	
`w=put(sasdate,weekw5.);`	
`x=put(sasdate,weekw7.);`	
`y=put(sasdate,weekw9.);`	
`z=put(sasdate,weekw11.);`	
`put v;`	`W03`
`put w;`	`03W03`
`put x;`	`03W0003`
`put y;`	`2003W0003`
`put z;`	`2003-W00-03`

See Also

Formats:
"WEEKU*w*. Format" on page 198
"WEEKV*w*. Format" on page 199
Functions:
"WEEK Function" on page 239
Informats:
"WEEKU*w*. Informat" on page 303
"WEEKV*w*. Informat" on page 305
"WEEKW*w*. Informat" on page 307

YEN*w.d* Format

Writes numeric values with yen signs, commas, and decimal points

Category: Numeric

Alignment: right

Syntax

YEN*w.d*

Syntax Description

w

specifies the width of the output field.

Default: 1

Range: 1–32

d

optionally specifies the number of digits to the right of the decimal point in the numeric value.

Restriction: must be either 0 or 2

Tip: If *d* is 2, then YEN*w.d* writes a decimal point and two decimal digits. If *d* is 0, then YEN*w.d* does not write a decimal point or decimal digits.

Details

The YEN*w.d* format writes numeric values with a leading yen sign and with a comma that separates every three digits of each value.

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character these codes represent may be different in other countries.

Examples

```
put cost yen10.2;
```

Cost	Result
	----+----1
1254.71	¥1,254.71

See Also

Informats:

"YEN*w.d* Informat" on page 309

PART 4

Functions for NLS

CHAPTER 10

Overview to Functions for NLS

Functions for NLS by Category

The following table provides brief descriptions of the SAS functions. For more detailed descriptions, see the NLS entry for each function.

Table 10.1 NLS Functions Summary

Category	Functions for NLS	Description
Character	"KCVT Function" on page 214	Converts data from an encoding code to another encoding code
	"TRANTAB Function" on page 235	Transcodes a data string by using a translation table
Currency Conversion	"EUROCURR Function" on page 209	Converts one European currency to another
DBCS	"KCOMPARE Function" on page 212	Returns the result of a comparison of character strings
	"KCOMPRESS Function" on page 213	Removes specific characters from a character string
	"KCOUNT Function" on page 214	Returns the number of double-byte characters in a string
	"KINDEX Function" on page 216	Searches a character expression for a string of characters
	"KINDEXC Function" on page 216	Searches a character expression for specific characters
	"KLEFT Function" on page 217	Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI
	"KLENGTH Function" on page 218	Returns the length of an argument
	"KLOWCASE Function" on page 218	Converts all letters in an argument to lowercase
	"KREVERSE Function" on page 219	Reverses a character expression

Category	Functions for NLS	Description
	"KRIGHT Function" on page 219	Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI
	"KSCAN Function" on page 220	Selects a specific word from a character expression
	"KSTRCAT Function" on page 221	Concatenates two or more character strings
	"KSUBSTR Function" on page 221	Extracts a substring from an argument
	"KSUBSTRB Function" on page 222	Extracts a substring from an argument according to the byte position of the substring in the argument
	"KTRANSLATE Function" on page 223	Replaces specific characters in a character expression
	"KTRIM Function" on page 224	Removes trailing DBCS blanks and SO/SI from character expressions
	"KTRUNCATE Function" on page 224	Truncates a numeric value to a specified length
	"KUPCASE Function" on page 225	Converts all single-byte letters in an argument to uppercase
	"KUPDATE Function" on page 225	Inserts, deletes, and replaces character value contents
	"KUPDATEB Function" on page 226	Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument
	"KVERIFY Function" on page 227	Returns the position of the first character that is unique to an expression
Date and Time	"NLDATE Function" on page 228	Converts the SAS-date value to the date value of the specified locale by using the date format modifiers
	"NLDATM Function" on page 231	Converts the SAS-datetime value to the time value of the specified locale by using the datetime- format modifiers
	"NLTIME Function" on page 233	Converts the SAS time or the datetime value to the time value of the specified locale by using the time-format modifiers
	"WEEK Function" on page 239	Returns the week-number value
Variable Information	"VARTRANSCODE Function" on page 236	Returns the transcode attribute of a SAS-data set variable
	"VTRANSCODE Function" on page 237	Returns a value that indicates whether transcoding is on or off for the specified character variable
	"VTRANSCODEX Function" on page 238	Returns a value that indicates whether transcoding is on or off for the specified argument

CHAPTER
11

Functions for NLS

EUROCURR Function

Converts one European currency to another

Category: Currency Conversion

Syntax

EUROCURR(*from-currency-amount*, *from-currency-code*, *to-currency-code*)

Arguments

from-currency-amount
is a numeric value that specifies the amount to convert.

from-currency-code
specifies a three-character currency code that identifies the currency that you are converting from. (See European Currency and Currency CodesTable 11.1 on page 210.)

Tip: If *from-currency-code* has a blank value, EUROCURR converts currency values from euros to the currency of the European country that you specify.

Featured in: Example 4 on page 212

to-currency-code
specifies a three-character currency code that identifies the currency that you are converting to. (See European Currency and Currency CodesTable 11.1 on page 210.)

Tip: If *to-currency-code* has a blank value, EUROCURR converts values from the currency of the European country that you specify to euros.

Details

The following table lists European currencies and the associated currency codes. Use the currency codes to identify the type of currency that you are converting to or converting from.

Table 11.1 European Currency and Currency Codes

Currency	Currency code
Austrian schilling	ATS
Belgian franc	BEF
British pound sterling	GBP
Czech koruna	CZK
Danish krone	DKK
Deutsche mark	DEM
Dutch guilder	NLG
Euro	EUR
Finnish markka	FIM
French franc	FRF
Greek drachma	GRD
Hungarian forint	HUF
Irish pound	IEP

Currency	Currency code
Italian lira	ITL
Luxembourg franc	LUF
Norwegian krone	NOK
Polish zloty	PLZ
Portuguese escudo	PTE
Romanian leu	ROL
Russian ruble	RUR
Slovenian tolar	SIT
Spanish peseta	ESP
Swedish krona	SEK
Swiss franc	CHF
Turkish lira	TRL
Yugoslavian dinar	YUD

The EUROCURR function converts a specific country's currency to an equivalent amount in another country's currency. It also can convert a specific country's currency to euros. EUROCURR uses the values in either the fixed currency conversion rate table or the changeable currency conversion rate table to convert currency.

If you are converting from one country's currency to euros, SAS divides the *from-currency-amount* by that country's rate from one of the conversion rate tables. See Example 1 on page 211. If you are converting from euros to a country's currency, SAS multiplies the *from-currency-amount* by that country's rate from one of the conversion rate tables. See Example 2 on page 211. If you are converting one country's currency to another country's currency, SAS first converts the *from-currency-amount* to euros. SAS stores the intermediate value in as much precision as your operating environment allows, and does not round the value. SAS then converts the amount in euros to an amount in the currency you are converting to. See Example 3 on page 212.

Examples

Example 1: Converting from Deutsche Marks to Euros The following example converts 1 Deutsche mark to an equivalent amount of euros.

```
data _null_;
   amount=eurocurr(50,'dem','eur');
   put amount= ;
run;
```

The value in the SAS log is: **amount=25.56459406**.

Example 2: Converting from Euros to Deutsche Marks The following example converts 1 euro to an equivalent amount of Deutsche marks.

```
data _null_;
   amount=eurocurr(25,'eur','dem');
   put amount= ;
run;
```

The value in the SAS log is: **amount=48.89575**.

Example 3: Converting from French Francs to Deutsche Marks The following example converts 50 French francs to an equivalent amount of Deutsche marks.

```
data _null_;
   x=50;
   amount=eurocurr(x,'frf','dem');
   put amount=;
run;
```

The value in the SAS log is: **amount=14.908218069**.

Example 4: Converting Currency When One Variable is Blank The following example converts 50 euros to Deutsche marks.

```
data _null_;
   x=50;
   amount=eurocurr(x,' ','dem');
   put amount=;
run;
```

The value in the SAS log is: **amount=97.7915**.

See Also

Formats:
- "EURO*w.d* Format" on page 115
- "EUROX*w.d* Format" on page 116

Informats:
- "EURO*w.d* Informat" on page 257
- "EUROX*w.d* Informat" on page 258

KCOMPARE Function

Returns the result of a comparison of character strings

Category: DBCS

Syntax

KCOMPARE(*source*,<*pos*, <*count*,>>*findstr*)

Arguments

source
specifies the character string to be compared.

pos
specifies the starting position in *source* to begin the comparison. If *pos* is omitted, the entire *source* is compared. If *pos* is less than 0, *source* is assumed as extended DBCS data that does not contain any SO/SI characters.

count
specifies the number of bytes to compare. If *count* is omitted, all of *source* that follows *pos* is compared, except for any trailing blanks.

findstr
specifies the character string to compare to *source*.

Details

KCOMPARE returns values as follows:

- a negative value if *source* is less than *findstr*
- 0 if *source* is equal to *findstr*
- a positive value if *source* is greater than *findstr*.

KCOMPRESS Function

Removes specified characters from a character string

Category: DBCS

Syntax

KCOMPRESS(*source*,<*characters-to-remove*>)

Arguments

source
specifies a character string that contains the characters to be removed. When only *source* is specified, KCOMPRESS returns this string with all of the single and double-byte blanks removed.

characters-to-remove
specifies the character or characters that KCOMPRESS removes from the character string.

Note: If characters-to-remove is omitted, KCOMPRESS removes all blanks. △

Tip: Enclose a literal string of characters in quotation marks.

See Also

Functions:

"KLEFT Function" on page 217
"KTRIM Function" on page 224

KCOUNT Function

Returns the number of double-byte characters in a string

Category: DBCS

Syntax

KCOUNT(*source*)

Arguments

source
specifies the character string to count.

KCVT Function

Converts data from one type of encoding data to another encoding data

Category: Character

Syntax

KCVT(*text, intype, outtype*, <*options*,...>)

Arguments

text
specifies the character variable to be converted.

intype
specifies the encoding of the data. The encoding of the text must match the input data's encoding. For valid values, see "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 407.

Note: ASCIIANY and EBCIDICANY are invalid encoding values. △

outtype
specifies the encoding to be converted into character data. For valid values see"SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 407.

Note: ASCIIANY and EBCIDICANY are invalid encoding values. △

options
specifies character data options. Following are the available options:

NOSOSI | NOSHIFT	No shift code or Hankaku characters

INPLACE	Replaces character data by conversion. The INPLACE option is specified to secure the same location between different hosts whose lengths of character data are not identical. For example, the INPLACE option converts data from the host which requires Shift-Codes, into the other host, which does not require shift codes. Truncation occurs when the length of the character data that is converted into *outtype* for Shift-Codes is longer than the length that is specified in *intype*.
KANA	Includes Hankaku katakana characters in columns of character data.
UPCASE	Converts 2-byte alphabet to uppercase characters.
LOWCASE	Converts 2-byte alphabet to lowercase characters.
KATA2HIRA	Converts Katakana data to Hiragana.
HIRA2KATA	Converts Hiragana data to Katakana.

Details

The KCVT function converts SBCS, DBCS, and MBCS character strings into encoding data. For example, the KCVT function can convert: ASCII code data to UCS2 encoding data, Greek code data to UTF-8, and Japanese SJIS code data to another Japanese code data. You can specify the following types for Intype and Outtype options: UCS2, UCS2L, UCS2B, and UTF8. To enable the DBCS mode, specify the following SAS options in the configuration file or in the command line.

- DBCS
- DBCSLANG Japanese or Korean or Chinese or Taiwanese
- DBCSTYPE dbcstype value

Example

The following code converts IBM PC codes into DEC codes for the external text file specified as *my-input-file*, and writes in OUTDD.

```
data _null_;
   infile 'my-input-file';
   file outdd noprint;
   input @1 text $char80.;
   text = kcvt(text, 'pcibm', 'dec');
   put @1 text $char80.;
run;
```

See Also

System options:

"DBCS System Option: UNIX, Windows, and z/OS" on page 349

"DBCSLANG System Option: UNIX, Windows, and z/OS" on page 350

"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351

Procedure:

Chapter 14, "The DBCSTAB Procedure," on page 313

KINDEX Function

Searches a character expression for a string of characters

Category: DBCS

Syntax

KINDEX(*source, excerpt*)

Arguments

source
specifies the character expression to search.

excerpt
specifies the string of characters to search for in the character expression.

Tip: Enclose a literal string of characters in quotation marks.

Details

The KINDEX function searches *source*, from left to right, for the first occurrence of the string that is specified in *excerpt*, and returns the position in *source* of the string's first character. If the string is not found in *source*, KINDEX returns a value of 0. If there are multiple occurrences of the string, KINDEX returns only the position of the first occurrence.

See Also

Functions:

"KINDEXC Function" on page 216

KINDEXC Function

Searches a character expression for specified characters

Category: DBCS

Syntax

KINDEXC(*source,excerpt-1*<,... *excerpt-n*>)

Arguments

source
specifies the character expression to search.

excerpt
specifies the characters to search for in the character expression.

Tip: If you specify more than one excerpt, separate them with a comma.

Details

The KINDEXC function searches *source*, from left to right, for the first occurrence of any character present in the excerpts and returns the position in *source* of that character. If none of the characters in *excerpt-1* through *excerpt-n* in *source* are found, KINDEXC returns a value of 0.

Comparisons

The KINDEXC function searches for the first occurrence of any individual character that is present within the character string, whereas the KINDEX function searches for the first occurrence of the character string as a pattern.

See Also

Function:
"KINDEX Function" on page 216

KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI

Category: DBCS

Syntax

KLEFT(*argument*)

Arguments

argument
specifies any SAS character expression.

Details

KLEFT returns an argument and removes the leading blanks.

See Also

Functions:

"KCOMPRESS Function" on page 213

"KRIGHT Function" on page 219

"KTRIM Function" on page 224

KLENGTH Function

Returns the length of an argument

Category: DBCS

Syntax

KLENGTH(*argument*)

Arguments

argument
specifies any SAS expression.

Details

The KLENGTH function returns an integer that represents the position of the rightmost non-blank character in the argument. If the value of the argument is missing, KLENGTH returns a value of 1. If the argument is an uninitialized numeric variable, KLENGTH returns a value of 12 and prints a note in the SAS log that the numeric values have been converted to character values.

KLOWCASE Function

Converts all letters in an argument to lowercase

Category: DBCS

Syntax

KLOWCASE(*argument*)

Arguments

argument
specifies any SAS character expression.

Details

The KLOWCASE function copies a character argument, converts all uppercase letters to lowercase letters, and returns the altered value as a result.

KREVERSE Function

Reverses a character expression

Category: DBCS

Syntax

KREVERSE(*argument*)

Arguments

argument
specifies any SAS character expression.

KRIGHT Function

Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI

Category: DBCS

Syntax

KRIGHT(*argument*)

Arguments

argument
specifies any SAS character expression.

Details

The KRIGHT function returns an argument with trailing blanks moved to the start of the value. The argument's length does not change.

See Also

Functions:

"KCOMPRESS Function" on page 213
"KLEFT Function" on page 217
"KTRIM Function" on page 224

KSCAN Function

Selects a specified word from a character expression

Category: DBCS

Syntax

KSCAN(*argument,n*<, *delimiters*>)

Arguments

argument
specifies any character expression.

n
specifies a numeric expression that produces the number of the word in the character string you want KSCAN to select.

Tip: If *n* is negative, KSCAN selects the word in the character string starting from the end of the string. If |*n*| is greater than the number of words in the character string, KSCAN returns a blank value.

delimiters
specifies a character expression that produces characters that you want KSCAN to use as word separators in the character string.

Default: If you omit *delimiters* in an ASCII environment, SAS uses the following characters:

blank . < (+ & ! $ *); ^ – / , % |

In ASCII environments without the ^ character, KSCAN uses the ~ character instead.

If you omit *delimiters* on an EBCDIC environment, SAS uses the following characters:

blank . < (+ | & ! $ *); ¬ – / , % | ¢

Tip: If you represent *delimiters* as a constant, enclose *delimiters* in quotation marks.

Details

Leading delimiters before the first word in the character string do not effect KSCAN. If there are two or more contiguous delimiters, KSCAN treats them as one.

KSTRCAT Function

Concatenates two or more character strings

Category: DBCS

Syntax

KSTRCAT(*argument-1*, *argument-2*<, ... *argument-n*>)

Arguments

argument
specifies any single-byte or double-byte character string.

Details

KSTRCAT concatenates two or more single-byte or double-byte character strings. It also removes unnecessary SO/SI pairs between the strings.

KSUBSTR Function

Extracts a substring from an argument

Category: DBCS

Syntax

KSUBSTR(*argument,position*<,*n*>)

Arguments

argument
specifies any SAS character expression.

position
specifies a numeric expression that is the beginning character position.

n
specifies a numeric expression that is the length of the substring to extract.

Interaction: If *n* is larger than the length of the expression that remains in *argument* after *position*, SAS extracts the remainder of the expression.

Tip: If you omit *n*, SAS extracts the remainder of the expression.

Details

The KSUBSTR function returns a portion of an expression that you specify in *argument*. The portion begins with the character specified by *position* and is the number of characters specified by *n*.

A variable that is created by KSUBSTR obtains its length from the length of *argument*.

See Also

Functions:

"KSUBSTRB Function" on page 222

KSUBSTRB Function

Extracts a substring from an argument according to the byte position of the substring in the argument

Category: DBCS

Syntax

KSUBSTRB(*argument,position*<,*n*>)

Arguments

argument
specifies any SAS character expression.

position
specifies the beginning character position in byte units.

n
specifies the length of the substring to extract in byte units.

Interaction: If *n* is larger than the length (in byte units) of the expression that remains in *argument* after *position*, SAS extracts the remainder of the expression.

Tip: If you omit *n*, SAS extracts the remainder of the expression.

Details

The KSUBSTRB function returns a portion of an expression that you specify in *argument*. The portion begins with the byte unit specified by *position* and is the number of byte units specified by *n*.

A variable that is created by KSUBSTRB obtains its length from the length of *argument*.

See Also

Functions:

"KSUBSTR Function" on page 221

KTRANSLATE Function

Replaces specific characters in a character expression

Category: DBCS

See: KTRANSLATE Function in the documentation for your operating environment.

Syntax

KTRANSLATE(*source,to-1,from-1*<,...*to-n,from-n*>)

Arguments

source
specifies the SAS expression that contains the original character value.

to
specifies the characters that you want KTRANSLATE to use as substitutes.

from
specifies the characters that you want KTRANSLATE to replace.

Interaction: Values of *to* and *from* correspond on a character-by-character basis; KTRANSLATE changes character one of *from* to character one of *to*, and so on. If *to* has fewer characters than *from*, KTRANSLATE changes the extra *from* characters to blanks. If *to* has more characters than *from*, KTRANSLATE ignores the extra *to* characters.

Operating Environment Information: You must have pairs of *to* and *from* arguments on some operating environments. On other operating environments, a segment of the collating sequence replaces null *from* arguments. See the SAS documentation for your operating environment for more information. △

Details

You can use KTRANSLATE to translate a single-byte character expression to a double-byte character expression, or translate a double-byte character expression to a single-byte character expression.

The maximum number of pairs of *to* and *from* arguments that KTRANSLATE accepts depends on the operating environment you use to run SAS. There is no functional difference between using several pairs of short arguments, or fewer pairs of longer arguments.

KTRIM Function

Removes trailing DBCS blanks and SO/SI from character expressions

Category: DBCS

Syntax

KTRIM(*argument*)

Arguments

argument
specifies any SAS character expression.

Details

KTRIM copies a character argument, removes all trailing blanks, and returns the trimmed argument as a result. If the argument is blank, KTRIM returns one blank. KTRIM is useful for concatenating because concatenation does not remove trailing blanks.

Assigning the results of KTRIM to a variable does not affect the length of the receiving variable. If the trimmed value is shorter than the length of the receiving variable, SAS pads the value with new blanks as it assigns it to the variable.

See Also

Functions:

"KCOMPRESS Function" on page 213

"KLEFT Function" on page 217

"KRIGHT Function" on page 219

KTRUNCATE Function

Truncates a numeric value to a specified length

Category: DBCS

Syntax

KTRUNCATE(*argument*, *number*,*length*)

Arguments

argument
specifies any SAS character expression.

number
is numeric.

length
is an integer.

Details

The KTRUNCATE function truncates a full-length *number* (stored as a double) to a smaller number of bytes, as specified in *length* and pads the truncated bytes with 0s. The truncation and subsequent expansion duplicate the effect of storing numbers in less than full length and then reading them.

KUPCASE Function

Converts all single-byte letters in an argument to uppercase

Category: DBCS

Syntax

KUPCASE(*argument*)

Arguments

argument
specifies any SAS character expression.

Details

The KUPCASE function copies a character argument, converts all single-byte lowercase letters to uppercase letters, and returns the altered value as a result.

KUPDATE Function

Inserts, deletes, and replaces character value contents

Category: DBCS

Syntax

KUPDATE(*argument,position,n*<, *characters-to-replace*>)

KUPDATE(*argument,position*<,*n*>, *characters-to-replace*)

Arguments

argument
: specifies a character variable.

position
: specifies a numeric expression that is the beginning character position.

n
: specifies a numeric expression that is the length of the substring to be replaced.

Restriction: *n* can not be larger than the length of the expression that remains in *argument* after *position*.

Restriction: *n* is optional, but you cannot omit both *n* and *characters-to-replace* from the function.

Tip: If you omit *n*, SAS uses all of the characters in *characters-to-replace* to replace the values of *argument*.

characters-to-replace
: specifies a character expression that will replace the contents of *argument*.

Restriction: *characters-to-replace* is optional, but you cannot omit both *characters-to-replace* and *n* from the function.

Tip: Enclose a literal string of characters in quotation marks.

Details

The KUPDATE function replaces the value of *argument* with the expression in *characters-to-replace*. KUPDATE replaces *n* characters starting at the character you specify in *position*.

See Also

Functions:

"KUPDATEB Function" on page 226

KUPDATEB Function

Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument

Category: DBCS

Syntax

KUPDATEB(*argument,position,n*<,*characters-to-replace*>)

KUPDATEB(*argument,position* <, *n*>, *characters-to-replace*)

Arguments

argument
specifies a character variable.

position
specifies the beginning character position in byte units.

n
specifies the length of the substring to be replaced in byte units.

Restriction: *n* can not be larger than the length (in bytes) of the expression that remains in *argument* after *position*.

Restriction: *n* is optional, but you cannot omit both *n* and *characters-to-replace* from the function.

Tip: If you omit *n*, SAS uses all of the characters in *characters-to-replace* to replace the values of *argument*.

characters-to-replace
specifies a character expression to replace the contents of *argument*.

Restriction: *characters-to-replace* is optional, but you cannot omit both *characters-to-replace* and *n* from the function.

Tip: Enclose a literal string of characters in quotation marks.

Details

The KUPDATEB function replaces the value of *argument* with the expression in *characters-to-replace*. KUPDATEB replaces *n* byte units starting at the byte unit that you specify in *position*.

See Also

Functions:

"KUPDATE Function" on page 225

KVERIFY Function

Returns the position of the first character that is unique to an expression

Category: DBCS

Syntax

KVERIFY(*source,excerpt-1*<,*...excerpt-n*>)

Arguments

source
specifies any SAS character expression.

excerpt
specifies any SAS character expression. If you specify more than one excerpt, separate them with a comma.

Details

The KVERIFY function returns the position of the first character in *source* that is not present in any *excerpt*. If KVERIFY finds every character in *source* in at least one *excerpt*, it returns a 0.

NLDATE Function

Converts the SAS date value to the date value of the specified locale by using the date format descriptors

Category: Date and Time

Syntax

NLDATE(*date,descriptor*)

Arguments

date
specifies a SAS date value.

descriptor
is a variable or expression that specifies how dates and times will be formatted in output. The following descriptors are case sensitive:

%%
specifies the % character.

%a
specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.

%A
specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.

%b
specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.

%B
: specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%C
: specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%d
: specifies the day descriptor and uses 0 padding. The range for the day modifier is 01–31.

%e
: specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F
: specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%j
: specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%m
: specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%o
: specifies the month descriptor. The range for the month descriptor is 1–12 with blank padding.

%u
: specifies the weekday descriptor as a number in the range 1–7 that represents Monday–Sunday.

%U
: specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value using the number of week within the year (Sunday is considered the first day of the week). The number-of-the-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

%V
: specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53 and uses a leading zero and a maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

%w
: specifies the weekday descriptor as a number in the range 0–6 that represents Sunday–Saturday.

%W
: specifies the week-number-of-year descriptor by calculating the descriptor value as SAS date value by using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

%y
: specifies the year (2-digit) modifier. The range for the year descriptor is 00–99.

%Y
: specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details

The NLDATE function converts the SAS date value to the date value of the specified locale by using the date descriptors.

Examples

The following example shows a log file name that is created from a SAS date value.

Statements	Results
`options locale=English_unitedstates;`	
`logfile=nldate('24Feb2003'd,'%B-%d.log');`	
`put logfile;`	`February-24.log`
`options locale=German_Germany;`	
`logfile=nldate('24Feb2003'd,'%B-%d.log');`	
`put logfile;`	`Februar-24.log`

The following example shows a weekday name that is created from a SAS date value.

Statements	Results
	`----+----1----+`
`options locale=English_unitedstates;`	
`weekname=nldate('24Feb2003'd,'%A');`	
`put weekname;`	`Monday`
`options locale=German_Germany;`	
`weekname=nldate('24Feb2003'd,'%A');`	
`put weekname;`	`Montag`

See Also

Format:

"NLDATE*w*. Format" on page 155

NLDATM Function

Converts the SAS datetime value to the time value of the specified locale by using the datetime-format descriptors

Category: Date and Time

Syntax

NLDATM(*datetime,descriptor*)

Arguments

datetime
specifies a SAS datetime value.

descriptor
is a variable or expression that specifies how dates and times will be formatted in output. The following descriptors are case sensitive:

%%
specifies the % character.

%a
specifies the short-weekday descriptor. The range for the day descriptor is Mon–Sun.

%A
specifies the long-weekday descriptor. The range for the long-weekday descriptor is Monday–Sunday.

%b
specifies the short-month descriptor. The range for the short-month descriptor is Jan–Dec.

%B
specifies the long-month descriptor. The range for the long-month descriptor is January–December.

%c
specifies the long-month descriptor and uses blank padding. The range for the long-month descriptor is January–December.

%d
specifies the day descriptor and uses 0 padding. The range for the day descriptor is 01–31.

%e
specifies the day descriptor and uses blank padding. The range for the day descriptor is 01–31.

%F
specifies the long-weekday descriptor and uses blank padding. The range for the day descriptor is Monday–Sunday.

%H
: specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.

%I
: specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.

%j
: specifies the day-of-year descriptor as a decimal number and uses a leading zero. The range for the day-of-year descriptor is 1–366.

%m
: specifies the month descriptor and uses 0 padding. The range for the month descriptor is 01–12.

%M
: specifies the minute descriptor. The range for the minute descriptor is 00–59.

%o
: specifies the month descriptor and uses blank padding. The range for the month descriptor is 1–12.

%p
: specifies a.m. or p.m. descriptor.

%S
: specifies the second descriptor. The range for the second descriptor is 00–59.

%u
: specifies the weekday descriptor as a number in the range of 1–7 that represents Monday–Sunday.

%U
: specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value and uses the number-of-week value within the year (Sunday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53. A leading zero and a maximum value of 53 is used.

%V
: specifies the week-number-of-year descriptor by calculating the descriptor value as the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53. A leading zero and a maximum value of 53 is used. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

%w
: specifies the weekday descriptor as a number in the range of 0–6 that represents Sunday–Saturday.

%W
: specifies the week-number-of-year descriptor by calculating the descriptor value as SAS date value using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range of 0–53. A leading zero and a maximum value of 53 are used.

%y
: specifies the year (2-digit) descriptor. The range for the year descriptor is 00–99.

%Y
specifies the year (4-digit) descriptor. The range for the year descriptor is 1970–2069.

Details

The NLDATM function converts the SAS datetime value to the datetime value of the specified locale by using the datetime descriptors.

Examples

The following example shows a time (a.m or p.m.) that is created from a SAS datetime value.

Statements	Results
	`----+----1----+`
`options locale=English;`	
`time_ampm=nldatm('24Feb2003:12:39:43'dt,'%I%p');`	
`put time_ampm;`	`00 PM`
`options locale=German;`	
`time_ampm=nldatm(''24Feb2003:12:39:43'dt,'%I%p');`	
`put time_ampm;`	`00 nachm`

See Also

Format:
"NLDATM*w*. Format" on page 159

NLTIME Function

Converts the SAS time or the datetime value to the time value of the specified locale by using the NLTIME descriptors

Category: Date and Time

Syntax

NLTIME(*time* | *datetime,descriptor,startpos*)

Arguments

time
specifies a SAS time value.

datetime
specifies a SAS datetime value.

descriptor
is a variable, or expression, that specifies the value of a descriptor. You can enter the following descriptors in uppercase or lowercase:

%%
specifies the % character.

%H
specifies the hour descriptor that is based on a 24-hour clock. The range for the hour descriptor is 00–23.

%I
specifies the hour descriptor that is based on a 12-hour clock. The range for the hour descriptor is 01–12.

%M
specifies the minute modifier. The range for the minute descriptor is 00–59.

%P
specifies the a.m. or p.m. descriptor.

%S
specifies the second descriptor. The range for the second descriptor is 00–59.

startpos
is an integer that specifies the position at which the search should start and that specifies the direction of the search.

Details

The NLTIME function converts a SAS time or datetime value to the time value of the specified locale by using the time descriptors.

Examples

The following example shows an AM or PM time that is created from a SAS time.

Statements	Results
	`----+----1----+`
`options locale=English;`	
`time_ampm=nltime('12:39:43't,'%i%p');`	
`put time_ampm;`	`00 PM`
`options locale=German;`	
`time_ampm=nltime('12:39:43't,'%i%p');`	
`put time_ampm;`	`00 nachm`

See Also

Formats:
"NLTIME*w*. Format" on page 172

TRANTAB Function

Transcodes a data string by using the specified translation table

Category: Character

Syntax

TRANTAB(*string,trantab_name*)

Note: Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.1 supports the TRANTAB function for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. △

Arguments

string
input string that is transcoded.

trantab_name
translation table.

Details

The TRANTAB function transcodes a data string by using a translation table to remap the characters from one internal representation to another. The encoding of the data in the input string must match the encoding of table 1 in the translation table. The TRANTAB function remaps the data from the encoding using table 1.

CAUTION:
Only experienced SAS users should use the TRANTAB function. △

Examples

The following example uses a translation table that transcodes data that is encoded in Latin2 to an uppercase Latin2 encoding:

Statements	Result
`teststrg=trantab('testing','lat2_ucs');` `put teststrg;`	TESTING

See Also

Procedures:
  Chapter 15, "The TRANTAB Procedure," on page 319

VARTRANSCODE Function

Returns the transcode attribute of a SAS data set variable

Category: Variable Information

Syntax

VARTRANSCODE(*data-set-id*, *var-num*)

Arguments

data-set-id
specifies the data set identifier that the OPEN function returns.

var-num
specifies the position of the variable in the SAS data set.

Tip: The VARNUM function returns this value.

Details

Transcoding is the process of converting data from one encoding to another. The VARTRANSCODE function returns 0 if the *var-num* variable does not transcode its value, or 1 if the *var-num* variable transcodes its value.

For more information about transcoding variables, see "Transcoding" in *SAS National Language Support (NLS): User's Guide*. For information about encoding values and transcoding data, see "SBCS, DBCS, and Unicode Encoding Values When Transcoding SAS Data" in *SAS National Language Support (NLS): User's Guide*.

Examples

The following example shows how to determine whether a character variable is transcoded:

```
data a;
   attrib x length=$3. transcode=no;
   attrib y length=$3. transcode=yes;
   x='abc';
   y='xyz';
run;

data _null_;
   dsid=open('work.a','i');
```

```
      nobs=attrn(dsid,"nobs");
      nvars=attrn(dsid,"nvars");
      do i=1 to nobs;
         xrc=fetch(dsid,1);
         do j=1 to nvars;
            transcode = vartranscode(dsid,j);
            put transcode=;
         end;
      end;
   run;
```

SAS writes the following output to the log:

```
transcode=0
transcode=1
```

See Also

Functions:

- ATTRN in *SAS Language Reference: Dictionary*
- OPEN in *SAS Language Reference: Dictionary*
- VARNUM in *SAS Language Reference: Dictionary*
- "VTRANSCODE Function" on page 237
- "VTRANSCODEX Function" on page 238

VTRANSCODE Function

Returns a value that indicates whether transcoding is enabled for the specified character variable

Category: Variable Information

Syntax

VTRANSCODE (*var*)

Arguments

var
specifies a character variable that is expressed as a scalar or as an array reference.

Restriction: You cannot use an expression as an argument.

Details

The VTRANSCODE function returns 0 if transcoding is off, and 1 if transcoding is on.

By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

Comparisons

- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.
- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.
- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the "Variable Information" functions in *SAS Language Reference: Dictionary*.

Example

Statements	Result
	`----+----1----+`
`attrib x transcode = yes;`	
`attrib y transcode = no;`	
`rc1 = vtranscode(y);`	
`put rc1=;`	`rc1=0`

See Also

Functions:

"VTRANSCODEX Function" on page 238

Statements:

ATTRIB in *SAS Language Reference: Dictionary*

VTRANSCODEX Function

Returns a value that indicates whether transcoding is enabled for the specified argument

Category: Variable Information

Syntax

VTRANSCODEX (*var*)

Arguments

var

specifies any SAS character expression that evaluates to a character variable name.

Restriction: The value of the specified expression cannot denote an array reference.

Details

The VTRANSCODEX function returns 0 if transcoding is off, and 1 if transcoding is on.

By default, all character variables in the DATA step are transcoded. You can use the TRANSCODE= attribute of the ATTRIB statement to turn transcoding off.

Comparisons

- The VTRANSCODE function returns a value that indicates whether transcoding is enabled for the specified variable. The VTRANSCODEX function, however, evaluates the argument to determine the variable name. The function then returns the transcoding status (on or off) that is associated with that variable name.
- The VTRANSCODE function does not accept an expression as an argument. The VTRANSCODEX function accepts expressions, but the value of the specified expression cannot denote an array reference.
- Related functions return the value of other variable attributes, such as the variable name, type, format, and length. For a list of the variable attributes, see the "Variable Information" functions in *SAS Language Reference: Dictionary*.

Examples

Statements	Result
	`----+----1----+`

`attrib x transcode = yes;`
`attrib y transcode = no;`
`rc1 = vtranscodex('y');`
`put rc1=;`
	`rc1=0`

See Also

Functions:
    "VTRANSCODE Function" on page 237
Statements:
    ATTRIB in *SAS Language Reference: Dictionary*

WEEK Function

Returns the week-number value

Category: Date and Time

Syntax

WEEK(<*sas_date*>, <*descriptor*>)

Arguments

sas_date
specifies the SAS date value. If the SAS date argument is not specified, the WEEK function returns the week-number value of the current date.

descriptor
specifies the value of the descriptor. The following descriptors can be specified in uppercase or lowercase characters.

U
specifies the SAS date value by using the number-of-week within the year (Sunday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

V
specifies the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53 and uses a leading zero and a maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year.

W
calculates the SAS date value by using the number of week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53 and uses a leading zero and a maximum value of 53.

Details

The WEEK function reads a SAS date value and returns the week number.

Examples

The following example shows a week number that is created from a SAS date value.

```
week('01FEB2003'd,modifier);
```

Descriptors	Results
	----+----1----+
U	4
V	5
W	4

See Also

Formats:
"WEEKU*w*. Format" on page 198

"WEEKV*w*. Format" on page 199
"WEEKW*w*. Format" on page 201

Informats:

"WEEKU*w*. Informat" on page 303
"WEEKV*w*. Format" on page 199
"WEEKW*w*. Informat" on page 307

PART 5

Informats for NLS

CHAPTER 12

Overview to Informats for NLS

Informats for NLS by Category

There are six categories of SAS informats that support NLS:

Table 12.1 Categories of Informats for NLS

Category	Description
BIDI text handling	Instructs SAS to read bidirectional data values from data variables.
Character	Instructs SAS to read character data values into character variables.
DBCS	Instructs SAS to manage various Asian languages.
Date and Time	Instructs SAS to read data values into variables that represent dates, times, and datetimes.
Hebrew text handling	Instructs SAS to read Hebrew data from data variables.
Numeric	Instructs SAS to read numeric data values into numeric variables.

The following table provides brief descriptions of the SAS informats. For more detailed descriptions, see the NLS entry for each informat.

Table 12.2 Summary of NLS Informats by Category

Category	Informats for NLS	Description
BIDI text handling	"$LOGVS*w*. Informat" on page 264	Reads a character string that is in left-to-right logical order, and then converts the character string to visual order
	"$LOGVSR*w*. Informat" on page 265	Reads a character string that is in right-to-left logical order, and then converts the character string to visual order
	"$VSLOG*w*. Informat" on page 301	Reads a character string that is in visual order, and then converts the character string to left-to-right logical order
	"$VSLOGR*w*. Informat" on page 302	Reads a character string that is in visual order, and then converts the character string to right-to-left logical order
Character	"$REVERJ*w*. Informat" on page 280	Reads character data from right to left and preserves blanks

Category	Informats for NLS	Description
	"$REVERS*w*. Informat" on page 281	Reads character data from right to left, and then left aligns the text
	"$UCS2B*w*. Informat" on page 282	Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS2BE*w*. Informat" on page 283	Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding
	"$UCS2L*w*. Informat" on page 284	Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS2LE*w*. Informat" on page 285	Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding
	"$UCS2X*w*. Informat" on page 286	Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS2XE*w*. Informat" on page 287	Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding
	"$UCS4B*w*. Informat" on page 288	Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS4L*w*. Informat" on page 289	Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS4X*w*. Informat" on page 290	Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session
	"$UCS4XE*w*. Informat" on page 291	Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding
	"$UESC*w*. Informat" on page 292	Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session
	"$UESCE*w*. Informat" on page 294	Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation
	"$UNCR*w*. Informat" on page 295	Reads an NCR character string, and then converts the character string to the encoding of the current SAS session

Category	Informats for NLS	Description
	"$UNCRE*w*. Informat" on page 296	Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR
	"$UPAREN*w*. Informat" on page 297	Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session
	"$UPARENE*w*. Informat" on page 298	Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation
	"$UPARENP*w*. Informat" on page 299	Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session, with national characters remaining in the encoding of the UPAREN representation
	"$UTF8X*w*. Informat" on page 300	Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session
DBCS	"$KANJI*w*. Informat" on page 263	Removes shift code data from DBCS data
	"$KANJIX*w*. Informat" on page 263	Adds shift-code data to DBCS data
Date and Time	"EURDFDE*w*. Informat" on page 252	Reads international date values
	"EURDFDT*w*. Informat" on page 253	Reads international datetime values in the form *ddmmmyy hh:mm:ss.ss* or *ddmmmyyyy hh:mm:ss.ss*
	"EURDFMY*w*. Informat" on page 255	Reads month and year date values in the form *mmmyy* or *mmmyyyy*
	"JDATEYMD*w*. Informat" on page 260	Reads Japanese Kanji date values in the format *yymmmdd* or *yyyymmmdd*
	"JNENGO*w*. Informat" on page 261	Reads Japanese Kanji date values in the form *yymmdd*
	"MINGUO*w*. Informat" on page 266	Reads dates in Taiwanese format
	"NENGO*w*. Informat" on page 267	Reads Japanese date values in the form *eyymmdd*
	"NLDATE*w*. Informat" on page 269	Reads the date value in the specified locale, and then converts the date value to the local SAS date value
	"NLDATM*w*. Informat" on page 269	Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value
	"NLTIMAP*w*. Informat" on page 278	Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value
	"NLTIME*w*. Informat" on page 279	Reads the time value in the specified locale, and then converts the time value to the local SAS time value

Category	Informats for NLS	Description
	"WEEKU*w*. Informat" on page 303	Reads the format of the number-of-week value within the year and returns a SAS date value by using the U algorithm
	"WEEKV*w*. Informat" on page 305	Reads the format of the number-of-week value within the year and returns a SAS date value using the V algorithm
	"WEEKW*w*. Informat" on page 307	Reads the format of the number-of-week value within the year and returns a SAS date value using the W algorithm
Hebrew text handling	"$CPTDW*w*. Informat" on page 250	Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding
	"$CPTWD*w*. Informat" on page 251	Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding
Numeric	"EURO*w.d* Informat" on page 257	Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point
	"EUROX*w.d* Informat" on page 258	Reads numeric values and removes embedded characters in European currency
	"NLMNY*w.d* Informat" on page 270	Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value
	"NLMNYI*w.d* Informat" on page 272	Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value
	"NLNUM*w.d* Informat" on page 273	Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value
	"NLNUMI*w.d* Informat" on page 274	Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value
	"NLPCT*w.d* Informat" on page 275	Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value
	"NLPCTI*w.d* Informat" on page 277	Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value
	"YEN*w.d* Informat" on page 309	Removes embedded yen signs, commas, and decimal points

CHAPTER

13

Informats for NLS

$CPTDW*w*. Informat

Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding

Category: Hebrew text handling

Syntax

$CPTDW*w*.

Syntax Description

w
specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $CPTDW*w*. informat performs processing that is opposite of the $CPTWD*w*. informat.

Examples

The following example uses the input value of 808182.

Statements	Result
	`----+----1----+`
`x=input('808182',$cptdw6.);` `put x;`	אבג

See Also

Formats:
"$CPTDW*w*. Format" on page 66
"$CPTWD*w*. Format" on page 67
Informats:
"$CPTWD*w*. Informat" on page 251

$CPTWD*w*. Informat

Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding

Category: Hebrew text handling

Syntax

$CPTWD*w*.

Syntax Description

w
specifies the width of the input field.
Default: 200
Range: 1–32000

Comparisons

The $CPTWD*w*. informat performs processing that is opposite of the $CPTDW*w*. informat.

Examples

The following example uses the input value of אבג.

Statements	Result
	`----+----1----+`
`x=input ('אבג',$cptwd6.);` `put x;`	
	`€□,`

See Also

Formats:
    "$CPTWD*w*. Format" on page 67
    "$CPTDW*w*. Format" on page 66
Informats:
    "$CPTDW*w*. Informat" on page 250

EURDFDE*w*. Informat

Reads international date values

Category: Date and Time

Syntax

EURDFDE*w*.

w
specifies the width of the input field.

Default: 7 (except Finnish)

Range: 7–32 (except Finnish)

Note: If you use the Finnish (FIN) language prefix, the *w* range is 10–32 and the default *w* is 10. △

Details

The date values must be in the form *ddmmmyy* or *ddmmmyyyy*:

dd
is an integer from 01–31 that represents the day of the month.

mmm
is the first three letters of the month name.

yy or *yyyy*
is a two-digit or four-digit integer that represents the year.

You can place blanks and other special characters between day, month, and year values.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option. △

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes . When you specify the language prefix in the informat, SAS ignores the DFLANG= system option.

Examples

This INPUT statement uses the value of the DFLANG= system option to read the international date values in Spanish.

```
options dflang=spanish;
input day eurdfde10.;
```

This INPUT statement uses the Spanish language prefix in the informat to read the international date values in Spanish. The value of the DFLANG= option, therefore, is ignored.

```
input day espdfde10.;
```

Values	Results
	----+----1
01abr1999	**14335**
01-abr-99	**14335**

See Also

Formats:

"EURDFDE*w*. Format" on page 70

Informats:

DATEw. in *SAS Language Reference: Dictionary*

"EURDFDT*w*. Informat" on page 253

"EURDFMY*w*. Informat" on page 255

System Options:

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

YEARCUTOFF= in *SAS Language Reference: Dictionary*

EURDFDT*w*. Informat

Reads international datetime values in the form *ddmmmyy hh:mm:ss.ss* or *ddmmmyyyy hh:mm:ss.ss*

Category: Date and Time

Syntax

EURDFDT*w*.

Syntax Description

w

specifies the width of the input field.

Default: 18

Range: 13–40

Details

The date values must be in the form *ddmmmyy* or *ddmmmyyyy*, followed by a blank or special character, and then the time values as *hh:mm:ss.ss*. The syntax for the date is represented as follows:

dd
: is an integer from 01–31 that represents the day of the month.

mmm
: is the first three letters of the month name.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

The syntax for time is represented as follows:

hh
: is the number of hours ranging from 00–23,

mm
: is the number of minutes ranging from 00–59,

ss.ss
: is the number of seconds ranging from 00–59 with the fraction of a second following the decimal point.

The EURDFDT*w*. informat requires values for both the date and the time; however, the *ss.ss* portion is optional.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option. △

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353 for the list of language prefixes . When you specify the language prefix in the informat, SAS ignores the DFLANG= system option.

Examples

This INPUT statement uses the value of the DFLANG= system option to read the international datetime values in German.

```
options dflang=german;
input date eurdfdt20.;
```

This INPUT statement uses the German language prefix to read the international datetime values in German. The value of the DFLANG= option, therefore, is ignored.

```
input date deudfdt20.;
```

Values	Results
	`----+----1----+----2`
`23dez99:10:03:17.2`	`1261562597.2`
`23dez1999:10:03:17.2`	`1261562597.2`

See Also

Formats:
DATEw. in *SAS Language Reference: Dictionary*
DATETIMEw.d in *SAS Language Reference: Dictionary*
"EURDFDT*w.d* Format" on page 73
TIMEw.d in *SAS Language Reference: Dictionary*
Functions:
DATETIME in *SAS Language Reference: Dictionary*
Informats:
DATETIMEw. in *SAS Language Reference: Dictionary*
"EURDFDE*w*. Informat" on page 252
"EURDFMY*w*. Informat" on page 255
System Options:
"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353
YEARCUTOFF= in *SAS Language Reference: Dictionary*

EURDFMY*w*. Informat

Reads month and year date values in the form *mmmyy* or *mmmyyyy*

Category: Date and Time

Syntax

EURDFMY*w*.

Syntax Description

w
specifies the width of the input field.

Default: 5 (except Finnish)

Range: 5–32 (except Finnish)

Note: If you use the Finnish (FIN) language prefix, the *w* range is 7–32 and the default value for *w* is 7. △

Details

The date values must be in the form *mmmyy* or *mmmyyyy*:

mmm
: is the first three letters of the month name.

yy or *yyyy*
: is a two-digit or four-digit integer that represents the year.

You can place blanks and other special characters between day, month, and year values. A value that is read with EURDFMY*w*. results in a SAS date value that corresponds to the first day of the specified month.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option. △

You can set the language for the SAS session with the DFLANG= system option. (Because the SAS Installation Representative usually sets a default language for the site, you might be able to skip this step.) If you work with dates in multiple languages, you can replace the EUR prefix with a language prefix. See “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 353 for the list of language prefixes . When you specify the language prefix in the informat, SAS ignores the DFLANG= option.

Examples

This INPUT statement uses the value of DFLANG= system option to read the international date values in French.

```
options dflang=french;
input month eurdfmy7.;
```

The second INPUT statement uses the French language prefix, and DFLANG is not specified.

```
input month fradfmy7.;
```

Values	Results
	`----+----1`
avr1999	**14335**
avr 99	**14335**

See Also

Formats:

DDMMYYw. in *SAS Language Reference: Dictionary*

“EURDFMY*w*. Format” on page 79

MMDDYYw. in *SAS Language Reference: Dictionary*

MONYYw. in *SAS Language Reference: Dictionary*

YYMMDDw. in *SAS Language Reference: Dictionary*

Functions:

MONTH in *SAS Language Reference: Dictionary*

YEAR in *SAS Language Reference: Dictionary*

Informats:

“EURDFDE*w*. Informat” on page 252

“EURDFDT*w*. Informat” on page 253

MONYYw. in *SAS Language Reference: Dictionary*

System Options:

“DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 353

YEARCUTOFF= in *SAS Language Reference: Dictionary*

EURO*w.d* Informat

Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point

Category: Numeric

Syntax

EURO*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 6

Range: 1–32

d

optionally specifies the power of 10 by which to divide the value. If the data contains decimal points, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The EURO*w.d* informat reads numeric values and removes embedded euro symbols (E), commas, blanks, percent signs, dashes, and right parentheses from the input data. A decimal point is assumed to be a separator between the whole number and the decimal portion. The EURO*w.d* informat converts a left parenthesis at the beginning of a field to a minus sign.

Comparisons

- The EURO*w.d* informat is similar to the EUROX*w.d* informat, but EUROX*w.d* reverses the roles of the decimal point and the comma. This convention is common in European countries.
- If no commas or periods appear in the input, then the EURO*w.d* and the EUROX*w.d* informats are interchangeable.

Examples

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

Values	Statements	Results
		`----+----1----2`
`E1`	`input x euro10.;` `put x;`	`1`
`E1.23`	`input x euro10.;` `put x;`	`1.23`
`1.23`	`input x euro10.;` `put x;`	`1.23`
`1,234.56`	`input x euro10.;` `put x;`	`1234.56`

See Also

Formats:

"EURO*w.d* Format" on page 115
"EUROX*w.d* Format" on page 116

Informats:

"EUROX*w.d* Informat" on page 258

EUROX*w.d* Informat

Reads numeric values and removes embedded characters in European currency

Category: Numeric

Syntax

EUROX*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 6

Range: 1–32

d

optionally specifies the power of 10 by which to divide the value. If the data contains a comma, which represents a decimal point, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The EUROX*w.d* informat reads numeric values and removes embedded euro symbols (E), periods, blanks, percent signs, dashes, and right parentheses from the input data. A comma is assumed to be a separator between the whole number and the decimal portion. The EUROX*w.d* informat converts a left parenthesis at the beginning of a field to a minus sign.

Comparisons

- The EUROX*w.d* informat is similar to the EURO*w.d* informat, but EURO*w.d* reverses the roles of the comma and the decimal point. This convention is common in English–speaking countries.
- If no commas or periods appear in the input, the EUROX*w.d* and the EURO*w.d* informats are interchangeable.

Examples

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

Values	Statements	Results
		`----+----1----2`
`E1`	`input x eurox10.;` `put x;`	`1`
`E1.23`	`input x eurox10.;` `put x;`	`123`
`1.23`	`input x eurox10.;` `put x;`	`123`
`1,234.56`	`input x eurox10.;` `put x;`	`1.23456`

See Also

Formats:
  "EURO*w.d* Format" on page 115
  "EUROX*w.d* Format" on page 116
Informats:
  "EURO*w.d* Informat" on page 257

JDATEYMD*w*. Informat

Reads Japanese Kanji date values in the format *yymmmdd* or *yyyymmmdd*

Category: Date and Time

Syntax

JDATEYMD*w*.

Syntax Description

w
specifies the width of the input field.
Default: 12
Range: 12–32

Details

The date values must be in the form *yymmmdd* or *yyyymmmdd*.

You can separate the year, month, and day values by blanks or by special characters. Note that in the example, the date values in the datalines are separated by special characters.

When you use this informat, ensure that the width of the input field includes space for blanks and special characters.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option. △

Examples

The following examples show how to use the JDATEYMD informat to convert Kanji values to SAS date values.

```
data _null_;
   input x jdateymd14.;
   put x=;
   put x= jdateymd14.;
   datalines;
1582年1月1日
1980年12月31日
2000年 1月 1日
2100年11月30日
;
data _null_;
  input x jdateymd.;
  put x=;
  put x= jdateymd14.;
  datalines;
1年1月1日
12年12月31日
99年 1月 1日
;
```

See Also

Informats:

"JNENGO*w*. Informat" on page 261

System Options:

YEARCUTOFF= in *SAS Language Reference: Dictionary*

JNENGO*w*. Informat

Reads Japanese Kanji date values in the form *yymmdd*

Category: Date and Time

Alignment: left

Syntax

JNENGO*w*.

Syntax Description

w
specifies the width of the output field.

Default: 16

Range: 16–32

Details

The JNENGO*w*. informat reads Japanese Kanji values in the form *yymmdd*.

You can separate the year, month, and day values by blanks or by special characters. Note that in the example, the date values in the datalines are separated by special characters.

When you use this informat, ensure that the width of the input field includes space for blanks and special characters.

Note: SAS interprets a two-digit year as belonging to the 100-year span that is defined by the YEARCUTOFF= system option. △

Examples

The following examples show how to use the JNENGO informat to convert Kanji values to SAS date values.

```
data _null_;
   input x jnengo.;
   datalines;
明治1年4月6日
明治45年7月29日
大正1年7月30日
大正15年12月24日
昭和1年12月25日
昭和64年 1月 7日
平成1年1月8日
平成10年12月 8日
;
```

See Also

Informats:

“JDATEYMD*w*. Informat” on page 260

System Options:

YEARCUTOFF= in *SAS Language Reference: Dictionary*

$KANJI*w*. Informat

Removes shift code data from DBCS data

Category: DBCS

Syntax

$KANJI*w*.

Syntax Description

w

specifies the width of the input field.

Restriction: The width must be an even number. If it is an odd number, it is truncated.

Range: The minimum width for the informat is 2.

Details

The data must start with SO and end with SI, unless single-byte blank data are returned. This informat always returns a blank for the DBCSTYPE data that does not use a shift-code mechanism. The input data length must be `2 + (SO/SI length)*2`.

See Also

Formats:

"$KANJI*w*. Format" on page 149

"$KANJIX*w*. Format" on page 150

Informats:

"$KANJIX*w*. Informat" on page 263

$KANJIX*w*. Informat

Adds shift-code data to DBCS data

Category: DBCS

Syntax

$KANJIX*w*.

Syntax Description

w

specifies the width of the input field.

Restriction: The width must be an even number. If it is an odd number, it is truncated.

Range: The minimum width for the informat is **`2 + (length of shift code used on the current DBCSTYPE= setting)*2.`**

See Also

Formats:

"$KANJI*w*. Format" on page 149

"$KANJIX*w*. Format" on page 150

Informats:

"$KANJI*w*. Informat" on page 263

$LOGVS*w*. Informat

Reads a character string that is in left-to-right logical order, and then converts the character string to visual order

Category: BIDI text handling

Syntax

$LOGVS*w*.

Syntax Description

w

specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $LOGVS*w*. informat performs processing that is opposite to the LOGVSR*w*. informat.

Examples

The following example uses the input value of “טיסה flight.”

Statements	Result
	`----+----1----+`
x=input ('טיסה flight',$logvs12.);	
`put x;`	הסיט flight

See Also

Formats:
- “$LOGVSR*w*. Format” on page 152
- “$LOGVS*w*. Format” on page 151

Informats:
- “$LOGVSR*w*. Informat” on page 265

$LOGVSR*w*. Informat

Reads a character string that is in right-to-left logical order, and then converts the character string to visual order

Category: BIDI text handling

Syntax

$LOGVSR*w*.

Syntax Description

w
specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $LOGVSR*w*. informat performs processing that is opposite to the $LOGVS*w*. informat.

Examples

The following example uses the input value of “טיסה flight.”

Statements	Results
	`----+----1----+`
x=input ('טיסה flight',$logvsr12.);	
put x;	flight הסיט

See Also

Formats:
- "$LOGVS*w*. Format" on page 151
- "$LOGVSR*w*. Format" on page 152

Informats:
- "$LOGVS*w*. Informat" on page 264

MINGUO*w*. Informat

Reads dates in Taiwanese format

Category: Date and Time

Syntax

MINGUO*w.*

Syntax Description

w
specifies the width of the input field.

Default: 6

Range: 6–10

Details

The general form of a Taiwanese date is *yyyymmdd*:

yyyy
is an integer that represents the year.

mm
is an integer from 01 through 12 that represents the month.

dd
is an integer from 01 through 31 that represents the day of the month.

The Taiwanese calendar uses 1912 as the base year (01/01/01 is January 1, 1912). Dates prior to 1912 are not valid. Year values do not roll over after 100 years; instead, they continue to increase.

You can separate the year, month, and day values with any delimiters, such as blanks, slashes, or dashes, that are permitted by the YYMMDD*w*. informat. If delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that have a value less than 10.

Examples

The following examples use different dates for input values.

```
input date minguo10.;
put date date9.;
```

Values	Results
	`----+----1----+`
49/01/01	**01JAN1960**
891215	**15DEC2000**
103-01-01	**01JAN2014**

See Also

Formats:

"MINGUO*w*. Format" on page 153

Informats:

YYMMDDw. in *SAS Language Reference: Dictionary*

NENGO*w*. Informat

Reads Japanese date values in the form *eyymmdd*

Category: Date and Time

Syntax

NENGO*w*.

Syntax Description

w

specifies the width of the input field.

Default: 10

Range: 7–32

Details

The general form of a Japanese date is *eyymmdd*:

e
: is the first letter of the name of the imperial era(Meiji, Taisho, Showa, or Heisei).

yy
: is an integer that represents the year.

mm
: is an integer from 01 through 12 that represents the month.

dd
: is an integer from 01 through 31 that represents the day of the month.

The *e* value can be separated from the integers by a period. If you omit *e*, SAS uses the current imperial era. You can separate the year, month, and day values by blanks or any nonnumeric character. However; if delimiters are used, place them between all the values. If you omit delimiters, be sure to use a leading zero for days or months that are values less than 10.

Examples

The following examples use different input values.

```
input nengo_date nengo8.;
put nengo_date date9.;
```

Values	Results
	`----+----1----+`
h11108	**08OCT1999**
h.11108	**08OCT1999**
11/10/08	**08OCT1999**

See Also

Formats:

"NENGO*w*. Format" on page 154

NLDATE*w*. Informat

Reads the date value in the specified locale, and then converts the date value to the local SAS date value

Category: Date and Time

Syntax

NLDATE*w*.

Syntax Description

w
specifies the width of the input field.

Default: 20

Range: 10–200

Examples

The following examples use the input February 24, 2003.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`y=input('February 24, 2003,nldate17.);`	
`put y=nldate.;`	`y=February 24, 2003`
`options locale=German_Germany;`	
`y=input('24. Februar 2003',nldate16.);`	
`put y=nldate;`	`y=24. Februar 2003`

See Also

Formats:

"NLDATE*w*. Format" on page 155

NLDATM*w*. Informat

Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value

Category: Date and Time

Syntax

NLDATM*w*.

Syntax Description

w
specifies the width of the input field.
Default: 30
Range: 10–200

Examples

The following examples use the input value of February 24, 2003 12:39:43.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`y=input('24.Feb03:12:39:43' nldatm.);`	
`put y=;`	`1361709583`
`options locale=German_Germany;`	
`y=input('24. Februar 2003 12.39 Uhr;, nldatm.);`	
`put y=;`	`1330171200`

See Also

Formats:
“NLDATM*w*. Format” on page 159

NLMNY*w.d* Informat

Reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value

Category: Numeric

Syntax

NLMNY*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 9

Range: 1–32

d

optionally specifies whether to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMNY*w.d* informat reads monetary data in the specified locale for the local expression, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the right parenthesis from the input data.

Comparisons

The NLMNY*w.d* informat performs processing that is the opposite of the NLMNYI*w.d* informat.

The NLMNY*w.d* informat is similar to the DOLLAR*w.d* informat except that the NLMNY*w.d* informat is locale specific.

Examples

The following examples use the input value of $12,345.67.

Statements	Results
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`x=input('($12,345.67)',nlmny32.2);`	
`y=input('($12,345.67)',dollar32.2);`	
`put x=;`	
	`-12345.67`
`put y=;`	`-12345.67`

See Also

Formats:

"NLMNY*w.d* Format" on page 163

"NLMNYI*w.d* Format" on page 165

Informats:

"NLMNYI*w.d* Informat" on page 272

NLMNYI*w.d* Informat

Reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value

Category: Numeric

Syntax

NLMNYI*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 9

Range: 1–32

d

optionally specifies whether to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLMNYI*w.d* informat reads monetary data in the specified locale for the international expression, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the right parenthesis from the input data.

Comparisons

The NLMNYI*w.d* informat performs processing that is the opposite of the NLMNY*w.d* informat.

Examples

The following examples use the input value of 12,345.67.

Statements	Results
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`x=input('(USD12,345.67)',nlmnyi32.2);`	
`y=input('$-12,345.67)',dollar32.2);`	
`put x=;`	`-12345.67`
`put y=;`	`-12345.67`

See Also

Formats:

"NLMNY*w.d* Format" on page 163

"NLMNYI*w.d* Format" on page 165

Informats:

"NLMNY*w.d* Informat" on page 270

NLNUM*w.d* Informat

Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value

Category: Numeric

Syntax

NLNUM*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 6

Range: 1–32

d

optionally specifies whether to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLNUM*w.d*) informat reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the right parenthesis from the input data.

Comparisons

The NLNUM*w.d* informat performs processing that is opposite to the NLNUMI*w.d* informat.

Examples

The following example uses -1234356.78 as the input value.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`x=input('-1,234,356.78',nlnum32.2);`	
`put x=;`	`-1234356.78`

See Also

Formats:
"NLNUM*w.d* Format" on page 166
"NLMNYI*w.d* Format" on page 165
Informats:
"NLNUMI*w.d* Informat" on page 274

NLNUMI*w.d* Informat

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value

Category: Numeric

Syntax

NLNUMI*w.d*

Syntax Description

w
specifies the width of the input field.
Default: 6

Range: 1–32

d

optionally specifies to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLNUMI*w.d* informat reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value. It removes any thousands separators, decimal separators, blanks, the currency symbol, and the right parenthesis from the input data.

Comparisons

The NLNUMI*w.d* informat performs processing that is opposite to the NLNUM*w.d* informat.

Examples

The following example uses -1,234,356.78 as the input value.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`x=input('-1,234,356.78', nlnumi32.2);`	
`put x=;`	`-1234356.78`

See Also

Formats:

“NLNUM*w.d* Format” on page 166

“NLNUMI*w.d* Format” on page 168

Informats:

“NLNUM*w.d* Informat” on page 273

NLPCT*w.d* Informat

Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value

Category: Numeric

Syntax

NLPCT*w.d*

Syntax Description

w

specifies the width of the input field.

Default: 6

Range: 1–32

d

optionally specifies whether to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLPCT*w.d* informat reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the right parenthesis from the input data.

Comparisons

The NLPCT*w.d* informat performs processing that is opposite of the NLPCTI*w.d* informat. The NLPCT*w.d* informat is similar to the PERCENT*w.d* informat except that the NLPCT*w.d* informat is locale specific.

Examples

The following example uses -12,345.67% as the input value.

Statements	Result
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`x=input('-12,345.67%',nlpct32.2);`	
`y=input('(12,345.67%)',percent32.2);`	
`put x=;`	`-123.4567`
`put y=;`	`-123.4567`

See Also

Formats:
  "NLPCT*w.d* Format" on page 169
  "NLPCTI*w.d* Format" on page 170
Informats:
  "NLPCTI*w.d* Informat" on page 277

NLPCTI*w.d* Informat

Reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value

Category: Numeric

Syntax

NLPCTI*w.d*

Syntax Description

w
specifies the width of the input field.

Default: 6

Range: 1–32

d
optionally specifies whether to divide the number by 10^d. If the data contains decimal separators, the *d* value is ignored.

Default: 0

Range: 0–31

Details

The NLPCTI*w.d* informat reads percentage data in the specified locale for international expressions, and then converts the data to a numeric value. It divides the value by 100 and removes any thousands separators, decimal separators, blanks, the percent sign, and the right parentheses from the input data.

Comparisons

The NLPCTI*w.d* informat performs processing that is opposite of the NLPCT*w.d* informat.

Examples

The following example uses -12,345.67% as the input value.

Statements	Results
	`----+----1----+`
`options LOCALE=English_UnitedStates;`	
`x=input('-12,345.67%',nlpct32.2);`	
`y=input('(12,345.67%)',percent32.2);`	
`put x=;`	`-123.4567`
`put y=;`	`-123.4567`

See Also

Formats:

"NLPCT*w.d* Format" on page 169

"NLPCTI*w.d* Format" on page 170

Informats:

"NLPCT*w.d* Informat" on page 275

NLTIMAP*w.* Informat

Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value

Category: Date and Time

Syntax

NLTIMAP*w.*

Syntax Description

w

specifies the width of the input field.

Default: 10

Range: 4–200

Examples

The following example uses 04:24:43 p.m. as the input value.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`y=input('04:24:43 PM',nltimap11.);`	
`put y time.;`	`16:24:43`
`options locale=German_Germany;`	
`y=input('16.24 Uhr',nltimap11.);`	
`put y time.;`	`16:24:43`

See Also

Formats:
"NLTIMAP*w*. Format" on page 173

NLTIME*w*. Informat

Reads the time value in the specified locale, and then converts the time value to the local SAS time value

Category: Date and Time

Syntax

NLTIME*w*.

Syntax Description

w
specifies the width of the input field.

Default: 20

Range: 10–200

Examples

The following example uses 16:24:43 as the input value.

Statements	Results
	`----+----1----+`
`options locale=English_UnitedStates;`	
`y=input('16:24:43',nltime.);`	
`put y time.;`	`16:24:43`
`options locale=German_Germany;`	
`y=input('16.24 Uhr',nltime.);`	
`put y time;`	`16:24:00`

See Also

Formats:
"NLTIME*w*. Format" on page 172

$REVERJ*w*. Informat

Reads character data from right to left and preserves blanks

Category: Character

Syntax

$REVERJ*w*.

Syntax Description

w
specifies the width of the input field.
Default: 1 if *w* is not specified
Range: 1–32767

Comparisons

The $REVERJ*w*. informat is similar to the $REVERS*w*. informat except that $REVERS*w*. informat left aligns the result by removing all leading blanks.

Examples

The following example uses ABCD as the input value.

```
input @1 name $reverj7.;
```

Values	Results
	`----+----1`
`ABCD`	`###DCBA`
`   ABCD`	`DCBA###`

* The character # represents a blank space.

See Also

Informats:

"$REVERS*w*. Informat" on page 281

$REVERS*w*. Informat

Reads character data from right to left, and then left aligns the text

Category: Character

Syntax

$REVERS*w*.

Syntax Description

w
specifies the width of the input field.

Default: 1 if *w* is not specified

Range: 1–32767

Comparisons

The $REVERS*w*. informat is similar to the $REVERJ*w*. informat except that $REVERJ*w*. informat preserves all leading and trailing blanks.

Examples

The following example uses ABCD as the input value.

```
input @1 name $revers7.;
```

Values	Results
	`----+----1`
ABCD	**DCBA###**
ABCD	**DCBA###**

* The # character represents a blank space.

See Also

Informats:

"$REVERJ*w*. Informat" on page 280

$UCS2B*w*. Informat

Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS2B*w*.

Syntax Description

w
specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32000

Comparisons

The $UCS2B*w*. informat performs processing that is opposite of the $UCS2BE*w*. informat. If you are processing data within the same operating environment, then use the $UCS2X*w*. informat. If you are processing data from different operating environments, then use the $UCS2B*w*. and $UCS2L*w*. informats.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`x=input('5927'x,ucs2b.);`	
`put x=$hex4.;`	`x=91e5`

See Also

Formats:

“$UCS2B*w*. Format” on page 174

“$UCS2L*w*. Format” on page 176

“$UCS2X*w*. Format” on page 178

“$UTF8X*w*. Format” on page 195

Informats:

“$UCS2L*w*. Informat” on page 284

“$UCS2X*w*. Informat” on page 286

“$UTF8X*w*. Informat” on page 300

$UCS2BE*w*. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to big-endian, 16-bit, UCS2, Unicode encoding

Category: Character

Syntax

$UCS2BE*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Comparisons

The $UCS2BE*w*. informat performs processing that is opposite of the $UCS2B*w*. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`ucs2str=input ('大', $ucs2be2.);`	
`put ucs2str=$hex4;`	`ucs2str=5927`

See Also

Formats:

- "$UCS2B*w*. Format" on page 174
- "$UCS2BE*w*. Format" on page 175

Informats:

- "$UCS2B*w*. Informat" on page 282

$UCS2L*w*. Informat

Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS2L*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32000

Comparisons

The $UCS2L*w*. informat performs processing that is opposite of the $UCS2LE*w*. informat. If you are processing data within the same operating environment, then use the $UCS2X*w*.informat. If you are processing data from different operating environments, then use the $UCS2B*w*. and $UCS2L*w*. informats.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`x=input('2759'x,ucs2l.);`	
`put x=$hex4.;`	`x=91e5`

See Also

Formats:

"$UCS2B*w*. Format" on page 174
"$UCS2L*w*. Format" on page 176
"$UCS2X*w*. Format" on page 178
"$UTF8X*w*. Format" on page 195

Informats:

"$UCS2B*w*. Informat" on page 282
"$UCS2X*w*. Informat" on page 286
"$UTF8X*w*. Informat" on page 300

$UCS2LE*w*. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to little-endian, 16-bit, UCS2, Unicode encoding

Category: Character

Syntax

$UCS2LE*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Comparisons

The $UCS2LE*w*. informat performs processing that is opposite of the $UCS2L*w*. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`ucs2str=input ('大', $ ucs2le2.);`	
`put ucs2str=$hex4;`	`ucs2str=2759`

See Also

Formats:

“$UCS2L*w*. Format” on page 176

“$UCS2LE*w*. Format” on page 177

Informats:

“$UCS2L*w*. Informat” on page 284

$UCS2X*w*. Informat

Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS2X*w*.

Syntax Description

w

specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 2–32000

Comparisons

The $UCS2X*w*. informat performs processing that is the opposite of the $UCS2XE*w*. informat. If you are processing data within the same operating environment, then use the $UCS2X*w*. informat. If you are processing data from different operating environments, then use the $UCS2B*w*. and $UCS2L*w*. informats.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

Statements	Result
	`----+----1----+`
`x=input('5927'x,ucs2x.);`	
`put x=$hex4.;`	`x=91e5`

See Also

Formats:

“$UCS2B*w*. Format” on page 174
“$UCS2L*w*. Format” on page 176
“$UCS2X*w*. Format” on page 178
“$UTF8X*w*. Format” on page 195

Informats:

“$UCS2B*w*. Informat” on page 282
“$UCS2L*w*. Informat” on page 284
“$UTF8X*w*. Informat” on page 300

$UCS2XE*w*. Informat

Reads a character string that is in the encoding of the current SAS session and then converts the character string to 16-bit, UCS2, Unicode encoding

Category: Character

Syntax

$UCS2XE*w*.

Syntax Description

w
specifies the width of the input field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

Default: 8

Range: 1-32000

Comparisons

The $UCS2XE*w*. informat performs processing that is opposite of the $UCS2X*w*. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`ucs2str=input ('大', $ ucs2xe2.);`	
`put ucs2str=$hex6;`	`ucs2str=5927`

See Also

Formats:
- "$UCS2X*w*. Format" on page 178
- "$UCS2XE*w*. Format" on page 180

Informats:
- "$UCS2X*w*. Informat" on page 286

$UCS4B*w*. Informat

Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS4B*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32000

Comparison

If you are processing data within the same operating environment, then use the $UCS4X*w*. informat. If you are processing data from different operating environments, then use the $UCS4B*w*. and $UCS4L*w*. informats.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`z=put('Zero1',$UCS4B20.);`	
`x=input(z,$UCS4B20.);`	
`put x;`	`Zero1`

See Also

Formats:

"$UCS4B*w*. Format" on page 181

Informats:

"$UCS4L*w*. Informat" on page 289
"$UCS4X*w*. Informat" on page 290

$UCS4L*w*. Informat

Reads a character string that is encoded in little-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS4L*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32000

Comparison

If you are processing data within the same operating environment, then use the $UCS4X*w*. informat. If you are processing data from different operating environments, then use the $UCS4B*w*. and $UCS4L*w*. informats.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+----2----+----3----+`
`z=put('.com',$UCS4L16.);`	
`put z $hex32.;`	`2E000000630000006F0000006D000000`

See Also

Formats:

"$UCS4L*w*. Format" on page 183

Informats:

"$UCS4B*w*. Informat" on page 288

"$UCS4X*w*. Informat" on page 290

$UCS4X*w*. Informat

Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UCS4X*w*.

Syntax Description

w

specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 4

Range: 4–32000

Comparisons

The $UCS4X*w*. informat performs processing that is the opposite of the $UCS4XE*w*. informat. Use the $UCS4X*w*. informat when you are processing data within the same operating environment. Use the $UCS4B*w*. and $UCS4L*w*. informats when you are processing data from different operating environments.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

Statements	Results
	`----+----1----+`
`ucs4=put('91e5'x,$ucs4x.);`	
`sjis=input(ucs4,$ucs4x.);`	`ucs4=27590000`
`put ucs4=$hex8. sjis=$hex8.;`	`sjis=91E52020`
`run;`	

See Also

Formats:

"$UCS2X*w*. Format" on page 178
"$UCS2B*w*. Format" on page 174
"$UCS2L*w*. Format" on page 176
"$UCS4X*w*. Format" on page 186
"$UTF8X*w*. Format" on page 195

Informats:

"$UCS2B*w*. Informat" on page 282
"$UCS2L*w*. Informat" on page 284
"$UTF8X*w*. Informat" on page 300

$UCS4XE*w*. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to 32-bit, UCS4, Unicode encoding

Category: Character

Syntax

$UCS4XE*w*.

Syntax Description

w
specifies the width of the input field. Specify enough width to accommodate the 32-bit size of the Unicode characters.

Default: 8

Range: 1–32000

Comparisons

The $UCS4XE*w*. informat performs processing that is the opposite of the $UCS4X*w*. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`ucs4str=input ('大', $ ucs4xe2.);`	
`put ucs4str=$hex8;`	`ucs4str=00005927`

See Also

Formats:
- "$UCS4X*w*. Format" on page 186
- "$UCS4XE*w*. Format" on page 187

Informats:
- "$UCS4X*w*. Informat" on page 290

$UESC*w*. Informat

Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UESC*w*.

Syntax Description

w

specifies the width of the output field.

Default: 8

Range: 1–32000

Details

If the characters are not available on all operating environments, for example, 0–9, a–z, A–Z, they must be represented in UESC representation. The $UESC*w*. informat can be nested.

Comparisons

The $UESC*w*. informat performs processing that is the opposite of the $UESCE*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+`
`x=input('¥u5927', $uesc10.);`	
`y=input('¥uu5927', $uesc10.);`	
`z=input('¥uuu5927', $uesc10.);`	
`put x;`	大
`put y;`	`¥u5927`
`put z;`	`¥uu5927`

See Also

Formats:

“$UESC*w*. Format” on page 188

“$UESCE*w*. Format” on page 189

Informats:

“$UESCE*w*. Informat” on page 294

$UESCE*w*. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation

Category: Character

Syntax

$UESCE*w*.

Syntax Description

w
specifies the width of the input field.

Default: 8

Range: 1–32000

Details

The $UESCE*w*. informat can be nested.

Comparisons

The $UESCE*w*. informat performs processing that is opposite of the $UESC*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+`
`x=input('大', $uesc10.);`	
`y=input('¥u5927',$uesc10.);`	`¥u5927`
`z=input('¥uu5927',$uesc10.);`	`¥uu5927`
`put x y z;`	`¥uuu5927`

See Also

Formats:

“$UESC*w*. Format” on page 188
“$UESCE*w*. Format” on page 189

Informat:

“$UESC*w*. Informat” on page 292

$UNCR*w*. Informat

Reads an NCR character string, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UNCR*w*.

Syntax Description

w

specifies the width of the input field.

Default: 8

Range: 1–32000

Details

The input string must contain only characters and NCR. Any national characters must be represented in NCR.

Comparison

The $UNCR*w*. informat performs processing that is opposite of the $UNCRE*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Result
	`----+----1----+`
`x=input (’大’, $uncr10.);`	
`y=input(’abc’, $uncr10);`	
`put X;`	大
`put Y;`	abc

See Also

Formats:
  "$UNCR*w*. Format" on page 190
  "$UNCRE*w*. Format" on page 191
Informats:
  "$UNCRE*w*. Informat" on page 296

$UNCRE*w*. Informat

Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR

Category: Character

Syntax

$UNCRE*w*.

Syntax Description

w
specifies the width of the input field.
Default: 8
Range: 1–32000

Details

The output string will be converted to plain characters and NCR. Any national characters will be converted to NCR.

Comparison

The $UNCRE*w*. informat performs processing that is the opposite of the $UNCR*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Result
	----+----1----+
x=input ('大abc', $uncre12.);	
put x;	大abc

See Also

Formats:
    "$UNCR*w*. Format" on page 190
    "$UNCRE*w*. Format" on page 191
Informats:
    "$UNCR*w*. Informat" on page 295

$UPAREN*w*. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UPAREN*w*.

Syntax Description

w
specifies the width of the input field.

Default: 8

Range: 1–32000

Details

If the SAS session encoding does not have a corresponding Unicode expression, the expression will remain in encoding of the current SAS session.

Comparisons

The $UPAREN*w*. informat performs processing that is opposite of the $UPARENE*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
`v=input('<u0061>',$uparen10.);`	
`w=input('<u0062>',$uparen10.);`	
`x=input('<u0063>',$uparen10.);`	
`y=input('<u0033>',$uparen10.);`	
`z=input('<u5927>',$uparen10.);`	
`put v;`	`a`
`put w;`	`b`
`put x;`	`c`
`put y;`	`3`
`put z;`	大

See Also

Formats:
- "$UPAREN*w*. Format" on page 192
- "$UPARENE*w*. Format" on page 194

Informats:
- "$UPARENE*w*. Informat" on page 298
- "$UPARENP*w*. Informat" on page 299

$UPARENE*w*. Informat

Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation

Category: Character

Syntax

$UPARENE*w*.

Syntax Description

w
specifies the width of the input field.

Default: 8

Range: 1–32000

Comparisons

The $UPARENE*w*. informat performs processing that is opposite of the $UPAREN*w*. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+`
`v=input('a',$uparen10.);`	
`w=input('b',$uparen10.);`	
`x=input('c',$uparen10.);`	
`y=input('3',$uparen10.);`	
`z=input('大',$uparen10.);`	
`put v;`	`<u0061>`
`put w;`	`<u0062>`
`put x;`	`<u0063>`
`put y;`	`<u0033>`
`put z;`	`<u5927>`

See Also

Formats:
- "$UPAREN*w*. Format" on page 192
- "$UPARENE*w*. Format" on page 194

Informats:
- "$UPAREN*w*. Informat" on page 297
- "$UPARENP*w*. Informat" on page 299

$UPARENP*w*. Informat

Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session, with national characters remaining in the encoding of the UPAREN representation

Category: Character

Syntax

$UPARENP*w*.

Syntax Description

w
specifies the width of the input field.

Default: 8

Range: 1–32000

Details

If the UPAREN expression contains a national character, whose value is greater than Unicode 0x00ff, the expression will remain as a UPAREN expression.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

Statements	Results
	`----+----1----+`
`v=input('<u0061>',$uparen10.);`	
`w=input('<u0062>',$uparen10.);`	
`x=input('<u0063>',$uparen10.);`	
`y=input('<u0033>',$uparen10.);`	
`z=input('<u5927>',$uparen10.);`	
`put v;`	`a`
`put w;`	`b`
`put x;`	`c`
`put y;`	`3`
`put z;`	`<u5927>`

See Also

Formats:

“$UPAREN*w*. Format” on page 192

“$UPARENE*w*. Format” on page 194

Informats:

“$UPAREN*w*. Informat” on page 297

“$UPARENE*w*. Informat” on page 298

$UTF8X*w*. Informat

Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session

Category: Character

Syntax

$UTF8X*w*.

Syntax Description

w

specifies the width of the input field.

Default: 8

Range: 1–32000

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

Statements	Result
	`----+----1----+`
`x=input (' e5a4a7' x, utf8x3.);` `put x;`	大

See Also

Formats:

“$UCS2B*w*. Format” on page 174

“$UCS2L*w*. Format” on page 176

“$UCS2X*w*. Format” on page 178

“$UTF8X*w*. Format” on page 195

Informats:

“$UCS2B*w*. Informat” on page 282

“$UCS2L*w*. Informat” on page 284

“$UCS2X*w*. Informat” on page 286

$VSLOG*w*. Informat

Reads a character string that is in visual order, and then converts the character string to left-to-right logical order

Category: BIDI text handling

Syntax

$VSLOG*w*.

Syntax Description

w
specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $VSLOG*w*. informat performs processing that is opposite of the $VSLOGR*w*. informat.

Examples

The following example used the input value of “ט י ס ה flight”.

Statements	Result
	`----+----1----+`
x=input (’ט י ס ה’,$vslog12.);	
`put x;`	טיסה`flight`

See Also

Formats:

“$VSLOGR*w*. Format” on page 197

“$VSLOG*w*. Format” on page 196

Informats:

“$VSLOGR*w*. Informat” on page 302

$VSLOGR*w*. Informat

Reads a character string that is in visual order, and then converts the character string to right-to-left logical order

Category: BIDI text handling

Syntax

$VSLOGR*w*.

Syntax Description

w

specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $VSLOGR*w*. informat performs processing that is opposite of the $VSLOG*w*. informat.

Examples

The following example used the input value of “טיסה flight.”

Statements	Result
	`----+----1----+`
x=input ('טיסה',$vslogr12.);	
`put x;`	**`flight`**טיסה

See Also

Formats:

“$VSLOG*w*. Format” on page 196

“$VSLOGR*w*. Format” on page 197

Informats:

“$VSLOG*w*. Informat” on page 301

WEEKU*w*. Informat

Reads the format of the number-of-week value within the year and returns a SAS date value by using the U algorithm

Category: Date and Time

Syntax

WEEKU*w*.

Syntax Description

w

specifies the width of the input field.

Default: 11

Range: 3–200

Details

The WEEKU*w*. informat reads the format of the number-of-week within the year, and then returns a SAS date value by using the U algorithm. If the input does not contain a year expression, then WEEKU*w*. uses the current year as the year expression, which is the default. If the input does not contain a day expression, then WEEKU*w*. uses the first day of the week as the day expression, which is the default.

The U Algorithm calculates the SAS date value using the number-of-week value within the year (Sunday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

The inputs to the WEEKU*w*. informat are the same date for the following example. The current year is 2003.

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKU*w*. informat reads the number-of-week value within the year. Sunday is the first day of the week, as a decimal number in the range 0–53, with a leading zero. The WEEKV*w*. informat reads the number-of-week value as a decimal number in the range 01–53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. informat reads the week-number-of-year value as a decimal number in the range 00–53, with Monday as the first day of week 1.

Examples

The current year is 2003 in the following examples.

Statements	Results
	`----+----1----+`
`v=input('W01',weeku3.);`	
`w=input('03W01',weeku5.);`	
`x=input('03W0101',weeku7.);`	
`y=input('2003W0101',weeku9.);`	
`z=input('2003-W01-01',weeku11.);`	
`put v;`	`15710`
`put w;`	`15710`
`put x;`	`15710`
`put y;`	`15710`
`put z;`	`15710`

See Also

Formats:

"WEEKU*w*. Format" on page 198
"WEEKV*w*. Format" on page 199
"WEEKW*w*. Format" on page 201

Functions:

"WEEK Function" on page 239

Informats:

"WEEKV*w*. Informat" on page 305
"WEEKW*w*. Informat" on page 307

WEEKV*w*. Informat

Reads the format of the number-of-week value within the year and returns a SAS date value using the V algorithm

Category: Date and Time

Syntax

WEEKV*w*.

Syntax Description

w

specifies the width of the input field.

Default: 11

Range: 3–200

Details

The WEEKV*w*. informat reads a format of the number-of-week value. If the input does not contain a year expression, WEEKV*w*. uses the current year as the year expression, which is the default. If the input does not contain a day expression, WEEKV*w*. uses the first day of the week as the day expression, which is the default.

The V algorithm calculates the SAS date value. The number-of-week value is represented as a decimal number in the range 01–53, with a leading zero and maximum value of 53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. For example, the fifth week of the year would be represented as 06.

The inputs to the WEEKV*w*. informat are the same date for the following example. The current year is 2003.

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKU*w*. informat reads the number-of-week value within the year. Sunday is the first day of the week, as a decimal number in the range 0–53, with a leading zero. The WEEKV*w*. informat reads the number-of-week value as a decimal number in the range 01–53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. informat reads the week-number-of-year value as a decimal number in the range 00–53, with Monday as the first day of week 1.

Examples

The current year is 2003 in the following examples.

Statements	Results
	`----+----1----+`
`v=input('W01',weekv3.);`	
`w=input('03W01',weekv5.);`	
`x=input('03W0101',weekv7.);`	
`y=input('2003W0101',weekv9.);`	
`z=input('2003-W01-01',weekv11.);`	
`put v;`	`15704`
`put w;`	`15704`
`put x;`	`15704`
`put y;`	`15704`
`put z;`	`15704`

See Also

Formats:
- "WEEKU*w*. Format" on page 198
- "WEEKV*w*. Format" on page 199
- "WEEKW*w*. Format" on page 201

Functions:
- "WEEK Function" on page 239

Informats:
- "WEEKU*w*. Informat" on page 303
- "WEEKW*w*. Informat" on page 307

WEEKW*w*. Informat

Reads the format of the number-of-week value within the year and returns a SAS date value using the W algorithm

Category: Date and Time

Syntax

WEEKW*w*.

Syntax Description

w

specifies the width of the input field.

Default: 11

Range: 3–200

Details

The WEEKW*w*. informat reads a format of the number-of-week value. If the input does not contain a year expression, the WEEKW*w*. informat uses the current year as the year expression, which is the default. If the input does not contain a day expression, the WEEKW*w*. informat uses the first day of the week as the day expression, which is the default. Algorithm W calculates the SAS date value using the number of the week within the year (Monday is considered the first day of the week). The number-of-week value is represented as a decimal number in the range 0–53, with a leading zero and maximum value of 53. For example, the fifth week of the year would be represented as 05.

The inputs to the WEEKW*w*. informat are the same date for the following example. The current year is 2003.

Widths	Formats	Examples
3-4	Www	w01
5-6	yyWww	03W01
7-8	yyWwwdd	03W0101
9-10	yyyyWwwdd	2003W0101
11-200	yyyy-Www-dd	2003-W01-01

Comparisons

The WEEKU*w*. informat reads the number-of-week value within the year. Sunday is the first day of the week, as a decimal number in the range 0–53, with a leading zero. The WEEKV*w*. informat reads the number-of-week value as a decimal number in the range 01–53. Weeks begin on a Monday and week 1 of the year is the week that includes both January 4th and the first Thursday of the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year. The WEEKW*w*. informat reads the week-number-of-year value as a decimal number in the range 00–53, with Monday as the first day of week 1.

Examples

The current year is 2003 in the following examples.

Statements	Results
	`----+----1----+`
`v=input('W01',weekw3.);`	
`w=input('03W01',weekw5.);`	
`x=input('03W0101',weekw7.);`	
`y=input('2003W0101',weekw9.);`	
`z=input('2003-W01-01',weekw11.);`	
`put v;`	`15711`
`put w;`	`15711`
`put x;`	`15711`
`put y;`	`15711`
`put z;`	`15711`

See Also

Formats:
"WEEKU*w*. Format" on page 198
"WEEKV*w*. Format" on page 199
"WEEKW*w*. Format" on page 201
Function:
"WEEK Function" on page 239
Informats:
"WEEKU*w*. Informat" on page 303
"WEEKV*w*. Informat" on page 305

YEN*w.d* Informat

Removes embedded yen signs, commas, and decimal points

Category: Numeric

Syntax

YEN*w.d*

Syntax Description

w
specifies the width of the input field.
Default: 1
Range: 1–32

d
optionally specifies the power of 10 by which to divide the value.

Requirement: *d* must be 0 or 2

Tip: If the *d* is 2, then YEN*w.d* reads a decimal point and two decimal digits. If *d* is 0, YEN*w.d* reads the value without a decimal point.

Details

The hexadecimal representation of the code for the yen sign character is 5B on EBCDIC systems and 5C on ASCII systems. The monetary character that these codes represent might be different in other countries.

Examples

The following example uses yen as the input.

```
input value yen10.2;
```

Value	Result
	`----+----1----+`
`¥1254.71`	`1254.71`

See Also

Formats:

"YEN*w.d* Format" on page 203

PART 6

Procedures for NLS

CHAPTER

14

The DBCSTAB Procedure

Overview: DBCSTAB Procedure

The DBCSTAB procedure produces conversion tables for the double-byte character sets that SAS supports.

Syntax: DBCSTAB Procedure

PROC DBCSTAB TABLE=*table-name*
<BASETYPE=*base-type*> <CATALOG=<*libref.*>*catalog-name*>
<DATA=<*libref.*>*table-name* > <DBCSLANG=*language*>
<DESC='*description*'> <FORCE> <VERIFY> <VERBOSE>;

PROC DBCSTAB Statement

PROC DBCSTAB TABLE=*table-name*
<*option(s)*>;

Required Arguments

TABLE=*table-name*
specifies the name of the double-byte code table to produce. This table name becomes an entry of type DBCSTAB in the catalog that is specified with the CATALOG= option. By default, the catalog name is SASUSER.DBCS.

Alias: NAME=, N=

Options

BASETYPE=*base-type*
specifies a base type for the double-byte code table conversion. If you use this option, you reduce the number of tables that are produced.
If you specify BASETYPE=, then all double-byte codes are first converted to the base code, and then converted to the required code. If you have *n* codes, then there are *n*(*n*-1) conversions that must be made.

Alias: BTYPE=

CATALOG=<*libref.*>*catalog-name*
specifies the name of the catalog in which the table is to be stored. If the catalog does not exist, it is created.

Default: SASUSER.DBCS

DATA=<*libref.*>*table-name*
specifies the data for producing the double-byte code table. Several double-byte character variables are required to produce the table. Use variable names that are equivalent to the value of the DBCSTYPE system option and are recognized by the KCVT function.

DBCSLANG=*language*
specifies the language that the double-byte code table uses. The value of this option should match the value of the DBCSLANG system option.

Alias: DBLANG

DESC='*description*'
specifies a text string to put in the DESCRIPTION field for the entry.

FORCE
produces the conversion tables even if errors are present.

VERIFY
checks the data range of the input table per code. This option is used to check for invalid double-byte code.

VERBOSE
causes the statistics detail to be printed when building DBCS tables.

When to Use the DBCSTAB Procedure

Use the DBCSTAB procedure to modify an existing DBCS table when

- the DBCS encoding system that you are using is not supported by SAS
- the DBCS encoding system that you are using has a nonstandard translation table.

A situation where you would be likely to use the DBCSTAB procedure is when a valid DBCSTYPE= value is not available. These values are operating environment dependent. In such cases, you can use the DBCSTAB procedure to modify a similar translation table, then specify the use of the new table with the TRANTAB option.

Examples: DBCSTAB Procedure

Example 1: Creating a Conversion Table with the DBCSTAB Procedure

Procedure features:

PROC DBCSTAB statement options:

CATALOG=
DBLANG=
BASETYPE=
VERIFY

The following example creates a Japanese translation table called CUSTAB and demonstrates how the TRANTAB option can be used to specify this new translation table.

Note: The DBCS, DBCSLANG, and DBCSTYPE options are specified at startup. △

The TRANTAB data set is created as follows:

```
data trantab;
      pcms='8342'x; dec='b9b3'x;
run;

proc dbcstab
      /* name of the new translate table */
          name=custtab
      /* based on pcibm encoding */
          basetype=pcms
      /* data to create the new table */
          data=trantab
      /* japanese language */
          dbcslang=japanese
      /* catalog descriptor */
          desc='Modified Japanese Trantab'
      /* where the table is stored */
          catalog=sasuser.dbcs
      /* checks for invalid DBCS in the new data */
          verify;
run;
```

To specify the translate table, use the TRANTAB option:

```
options trantab=(,,,,,,,,custtab);
```

Translate tables are generally used for DBCS conversion with SAS/CONNECT software, PROC CPORT and PROC CIMPORT, and the DATA step function, KCVT.

The TRANTAB= option may be used to specify DBCS translate tables. For SAS release 8.2 and earlier versions, the ninth argument was formerly used to specify the DBCS system table. However, for SAS 9 and later versions, instead of using the ninth argument, the SAS system uses a system table that is contained in a loadable module.

```
options trantab=(,,,,,,,,systab); /* ninth argument */
```

Japanese, Korean, Chinese, and Taiwanese are acceptable for the systab name.

The tenth argument specifies the DBCS user table:

```
options trantab=(,,,,,,,,,usrtab); /* tenth argument */
```

Example 2: Producing Japanese Conversion Tables with the DBCSTAB Procedure

Procedure features:

PROC DBCSTAB statement options:

- TABLE=
- DATA=
- DBLANG=
- BASETYPE=
- VERIFY

Program

```
data ja_jpn;
   length ibm jis euc pcibm $2.;
   ibm='4040'x;
   jis='2121'x;
   euc='a1a1'x;
   pcibm='8140'x;
run;

proc dbcstab
   table=japanese
   data=ja_jpn
   dblang=japanese
   basetype=jis
   verify;
run;
```

Log

```
1   proc dbcstab
2   table=ja_jpn
3   data=work.ja_jpn
4   dblang=japanese
5   basetype=jis
6   verify;
7   run;

NOTE: Base table for JIS created.
NOTE: IBM table for JIS created.
NOTE: PCIBM table for JIS created.
NOTE: EUC table for JIS created.
NOTE: Base table for IBM created.
NOTE: JIS table for IBM created.
NOTE: Base table for PCIBM created.
NOTE: JIS table for PCIBM created.
NOTE: Base table for EUC created.
NOTE: JIS table for EUC created.
NOTE: 10 DBCS tables are generated. Each table has 1 DBCS characters.
NOTE: Each table is 2 bytes in size.
NOTE: Required table memory size is 612.
NOTE: There were 1 observations read from the dataset WORK.JA_JPN.
```

See Also

Functions:

“KCVT Function” on page 214

Procedures:

Chapter 15, “The TRANTAB Procedure,” on page 319

System Options:

“TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 362

“DBCS System Option: UNIX, Windows, and z/OS” on page 349

“DBCSLANG System Option: UNIX, Windows, and z/OS” on page 350

“DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 351

CHAPTER

15

The TRANTAB Procedure

Overview: TRANTAB Procedure

The TRANTAB procedure creates, edits, and displays customized translation tables. In addition, you can use PROC TRANTAB to view and modify translation tables that are supplied by SAS. These SAS supplied tables are stored in the SASHELP.HOST catalog. Any translation table that you create or customize is stored in your SASUSER.PROFILE catalog. Translation tables have an entry type of TRANTAB.

Translation tables are operating environment-specific SAS catalog entries that are used to translate the values of one (coded) character set to another. A translation table has two halves: table one provides a translation, such as ASCII to EBCDIC; table two provides the inverse (or reverse) translation, such as EBCDIC to ASCII. Each half of a translation table is an array of 256 two-digit *positions*, each of which contains a one-byte unsigned number that corresponds to a coded character.

The SAS System uses translation tables for the following purposes:

- determining the collating sequence in the SORT procedure
- performing transport-format translations when you transfer files with the CPORT and CIMPORT procedures
- performing translations between operating environments when you access remote data in SAS/CONNECT or SAS/SHARE software
- facilitating data communications between the operating environment and a graphics device when you run SAS/GRAPH software in an IBM environment
- accommodating national language character sets other than U.S. English.

PROC TRANTAB produces no output. It can display translation tables and notes in the SAS log.

Concepts: TRANTAB Procedure

Understanding Translation Tables and Character Sets for PROC TRANTAB

The *k*th element in a translation table corresponds to the *k*th element of an ordered character set. For example, position 00 (which is byte 1) in a translation table contains a coded value that corresponds to the first element of the ordered character set. To determine the position of a character in your operating environment's character set, use the SAS function RANK. The following example shows how to use RANK:

```
data _null_;
   x=rank('a');
   put "The position of a is " x ".";
```

The SAS log prints the following message: **`The position of a is 97 .`**

Each position in a translation table contains a hexadecimal number that is within the range of 0 ('00'x) to 255 ('FF'x). Hexadecimal values always end with an x. You can represent one or more consecutive hexadecimal values within quotation marks followed by a single x. For example, a string of three consecutive hexadecimal values can be written as '08090A'x. The SAS log displays each row of a translation table as 16 hexadecimal values enclosed in quotes followed by an x. The SAS log also lists reference numbers in the vertical and horizontal margins that correspond to the positions in the table. Example 1 on page 329 shows how the SAS log displays a translation table.

Storing Translation Tables with PROC TRANTAB

When you use PROC TRANTAB to create a customized translation table, the procedure automatically stores the table in your SASUSER.PROFILE catalog. This enables you to use customized translation tables without affecting other users. When you specify the translation table in the SORT procedure or in a GOPTIONS statement, the software first looks in your SASUSER.PROFILE catalog to find the table. If the specified translation table is not in your SASUSER.PROFILE catalog, the software looks in the SASHELP.HOST catalog.

If you want the translation table you create to be globally accessed, have your SAS Installation Coordinator copy the table from your SASUSER.PROFILE catalog (using the CATALOG procedure) to the SASHELP.HOST catalog. If the table is not found there, the software will continue to search in SASHELP.LOCALE for the table.

Modifying SAS Translation Tables with PROC TRANTAB

If a translation table that is provided by SAS does not meet your needs, you can use PROC TRANTAB to edit it and create a new table. That is, you can issue the PROC TRANTAB statement that specifies the SAS table, edit the table, and then save the table using the SAVE statement. The modified translation table is saved in your SASUSER.PROFILE catalog. If you are a SAS Installation Coordinator, you can modify a translation table with PROC TRANTAB and then use the CATALOG procedure to copy the modified table from your SASUSER.PROFILE catalog to the SASHELP.HOST catalog, as shown in the following example:

```
proc catalog c=sasuser.profile;
   copy out=sashelp.host entrytype=trantab;
run;
```

You can use PROC TRANTAB to modify translation tables stored in the SASHELP.HOST catalog only if you have update (or write) access to that data library and catalog.

Using Translation Tables Outside PROC TRANTAB

Using Translation Tables in the SORT Procedure

PROC SORT uses translation tables to determine the collating sequence to be used by the sort. You can specify an alternative translation table with the SORTSEQ= option of PROC SORT. For example, if your operating environment sorts with the EBCDIC sequence by default, and you want to sort with the ASCII sequence, you can issue the following statement to specify the ASCII translation table:

```
proc sort sortseq=ascii;
```

You can also create a customized translation table with PROC TRANTAB and specify the new table with PROC SORT. This is useful when you want to specify sorting sequences for languages other than U.S. English.

See Example 6 on page 339 for an example that uses translation tables to sort data in different ways. For information on the tables available for sorting and the SORTSEQ= option, see “SORTSEQ= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 361.

Using Translation Tables with the CPORT and CIMPORT Procedures

The CPORT and CIMPORT procedures use translation tables to translate characters in catalog entries that you export from one operating environment and import on another operating environment. You may specify the name of a supplied translation table or a customized translation table in the TRANTAB statement of PROC CPORT. See “TRANTAB Statement” on page 391 in the CPORT Procedure for more information.

Using Translation Tables with Remote Library Services

Remote Library Services (RLS) uses translation tables to translate characters when you access SAS 8 remote data. SAS/CONNECT and SAS/SHARE software use translation tables to translate characters when you transfer or share files between two operating environments that use different encoding standards.

Using Translation Tables in SAS/GRAPH Software

In SAS/GRAPH software, translation tables are most commonly used on an IBM operating environment where tables are necessary because graphics commands must leave IBM operating environments in EBCDIC representation but must reach asynchronous graphics devices in ASCII representation. Specifically, SAS/GRAPH software builds the command stream for these devices internally in ASCII representation but must convert the commands to EBCDIC representation before they can be given to the communications software for transmission to the device. SAS/GRAPH software uses a translation table internally to make the initial conversion from ASCII to EBCDIC. The communications software then translates the command stream back to ASCII representation before it reaches the graphics device.

Translation tables are operating environment-specific. In most cases, you can simply use the default translation table, SASGTAB0, or one of the SAS supplied graphics translation tables. However, if these tables are not able to do all of the translation correctly, you can create your own translation table with PROC TRANTAB. The SASGTAB0 table may fail to do the translation correctly when it encounters characters from languages other than U.S. English.

To specify an alternative translation table for SAS/GRAPH software, you can either use the TRANTAB= option in a GOPTIONS statement or modify the TRANTAB device parameter in the device entry. For example, the following GOPTIONS statement specifies the GTABTCAM graphics translation table:

```
goptions trantab=gtabtcam;
```

Translation tables used in SAS/GRAPH software perform both *device-to-operating environment* translation and *operating environment-to-device* translation. Therefore, a translation table is made up of 512 bytes, with the first 256 bytes used to perform device-to-operating environment translation (ASCII to EBCDIC on IBM mainframes) and the second 256 bytes used to perform operating environment-to-device translation (EBCDIC to ASCII on IBM mainframes). For PROC TRANTAB, the area of a translation table for device-to-operating environment translation is considered to be *table one*, and the area for operating environment-to-device translation is considered to be *table two*. See Example 1 on page 329 for a listing of the ASCII translation table (a SAS provided translation table), which shows both areas of the table.

On operating environments other than IBM mainframes, translation tables can be used to translate specific characters in the data stream that are created by the driver. For example, if the driver normally generates a vertical bar in the data stream, but you want another character to be generated in place of the vertical bar, you can create a translation table that translates the vertical bar to an alternate character.

For details on how to specify translation tables with the TRANTAB= option in SAS/GRAPH software, see *SAS/GRAPH Software: Reference, Version 6, First Edition, Volume 1* and *Volume 2*.

SAS/GRAPH software also uses key maps and device maps to map codes generated by the keyboard to specified characters and to map character codes to codes required by the graphics output device. These maps are specific to SAS/GRAPH software and are discussed in "The GKEYMAP Procedure" in *SAS/GRAPH Software: Reference*.

Syntax: TRANTAB Procedure

Tip: Supports RUN-group processing

PROC TRANTAB TABLE=*table-name* <NLS>;
 CLEAR <ONE|TWO|BOTH>;
 INVERSE;
 LIST <ONE|TWO|BOTH>;
 LOAD TABLE=*table-name* <NLS>;
 REPLACE *position value-1*<*…value-n*>;
 SAVE <TABLE=*table-name*> <ONE|TWO|BOTH>;
 SWAP;

Task	Use this statement
Set all positions in the translation table to zero	CLEAR
Create an inverse of table 1	INVERSE
Display a translation table in hexadecimal representation	LIST
Load a translation table into memory for editing	LOAD
Replace the characters in a translation table with specified values	REPLACE
Save the translation table in your SASUSER.PROFILE catalog	SAVE
Exchange table 1 with table 2	SWAP

Note: Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.1 supports the TRANTAB procedure for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases.

PROC TRANTAB is an interactive procedure. Once you submit a PROC TRANTAB statement, you can continue to enter and execute statements without repeating the PROC TRANTAB statement. To terminate the procedure, submit a QUIT statement or submit another DATA or PROC statement. △

PROC TRANTAB Statement

Tip: If there is an incorrect table name in the PROC TRANTAB statement, use the LOAD statement to load the correct table. You do not need to reinvoke PROC TRANTAB. New tables are not stored in the catalog until you issue the SAVE statement, so you will not have unwanted tables in your catalog.

PROC TRANTAB TABLE=*table-name* <NLS>;

Required Arguments

TABLE=*table-name*
specifies the translation table to create, edit, or display. The specified table name must be a valid one-level SAS name with no more than 8 characters.

Options

NLS
specifies that the table you listed in the TABLE= argument is one of five special internal translation tables provided with every copy of the SAS System. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

SASXPT
the local-to-transport format translation table (used by the CPORT procedure)

SASLCL
the transport-to-local format translation table (used by the CIMPORT procedure)

SASUCS
the lowercase-to-uppercase translation table (used by the UPCASE function)

SASLCS
the uppercase-to-lowercase translation table (used by the LOWCASE macro)

SASCCL
the character classification table (used internally), which contains flag bytes that correspond to each character position that indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to translate characters that exist in languages other than English. To make SAS use the modified NLS table, specify its name in the SAS system option TRANTAB= .

Note: When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all zeros. PROC TRANTAB does not use table 2 at all for translation in these special cases, so you do not need to be concerned about this note. △

CLEAR Statement

Sets all positions in the translation table to zero; used when you create a new table

CLEAR <ONE|TWO|BOTH>;

Options

ONE | TWO | BOTH

ONE
clears table 1.

TWO
clears table 2.

BOTH
clears both table 1 and table 2.

Default: ONE

INVERSE Statement

Creates an inverse of table 1 in a translation table; that is, it creates table 2.

Featured in: Example 5 on page 337

INVERSE;

Details

INVERSE does not preserve multiple translations. Suppose table 1 has two (or more) different characters translated to the same value; for example, "A" and "B" are both translated to "1". For table 2, INVERSE uses the last translated character for the value; that is, "1" is always translated to "B" and not "A", assuming that "A" appears before "B" in the first table.

Sort programs in SAS require an inverse table for proper operation.

LIST Statement

Displays in the SAS log a translation table in hexadecimal representation

Featured in: All examples

LIST <ONE|TWO|BOTH>;

Options

ONE | TWO | BOTH

ONE
displays table 1.

TWO
displays table 2.

BOTH
displays both table 1 and table 2.

Default: ONE

LOAD Statement

Loads a translation table into memory for editing

Tip: Use LOAD when you specify an incorrect table name in the PROC TRANTAB statement. You can specify the correct name without reinvoking the procedure.

Tip: Use LOAD to edit multiple translation tables in a single PROC TRANTAB step. (Be sure to save the first table before you load another one.)

Featured in: Example 4 on page 335

LOAD TABLE=*table-name* <NLS>;

Required Arguments

TABLE=table-name
specifies the name of an existing translation table to be edited. The specified table name must be a valid one-level SAS name.

Option

NLS
specifies that the table you listed in the TABLE= argument is one of five special internal translation tables that are provided with SAS. You must use the NLS option when you specify one of the five special tables in the TABLE= argument:

SASXPT
is the local-to-transport format translation table

SASLCL
is the transport-to-local format translation table

SASUCS
is the lowercase-to-uppercase translation table

SASLCS
: is the uppercase-to-lowercase translation table

SASCCL
: is the character classification table, which contains flag bytes that correspond to each character position, these indicate the class or classes to which each character belongs.

NLS stands for National Language Support. This option and the associated translation tables provide a method to map characters from languages other than English to programs, displays, and files.

Note: When you load one of these special translation tables, the SAS log displays a note that states that table 2 is uninitialized. That is, table 2 is an empty table that contains all zeros. PROC TRANTAB does not use table 2 for translation in these special cases. △

REPLACE Statement

Replaces characters in a translation table with the specified values, starting at the specified position

Alias: REP

Tip: To save edits, you must issue the SAVE statement.

Featured in: Example 2 on page 330, Example 3 on page 332, and Example 4 on page 335

REPLACE *position value-1*<*...value-n*>;

Required Arguments

position

specifies the position in a translation table where the replacement is to begin. The editable positions in a translation table begin at position decimal 0 and end at decimal 255. To specify the position, you can do either of the following:

- Use a decimal or hexadecimal value to specify an actual location. If you specify a decimal value, for example, 20, PROC TRANTAB locates position 20 in the table, which is byte 21. If you specify a hexadecimal value, for example, ’14’x, PROC TRANTAB locates the decimal position that is equivalent to the specified hexadecimal value, which in this case is position 20 (or byte 21) in the table.
- Use a quoted character. PROC TRANTAB locates the quoted character in the table (that is, the quoted character’s hexadecimal value) and uses that character’s position as the starting position. For example, if you specify the following REPLACE statement, the statement replaces the first occurrence of the hexadecimal value for "a" and the next two hexadecimal values with the hexadecimal equivalent of "ABC":

```
replace 'a' 'ABC';
```

  This is useful when you want to locate alphabetic and numerical characters but you do not know their actual location. If the quoted character is not found, PROC TRANTAB displays an error message and ignores the statement.

To edit positions 256 through 511 (table two), follow this procedure:

1 Issue the SWAP statement.

2 Issue the appropriate REPLACE statement.

3 Issue the SWAP statement again to reposition the table.

value-1 <...value-n>
is one or more decimal, hexadecimal, or character constants that give the actual value to be put into the table, starting at *position*. You can also use a mixture of the types of values. That is, you can specify a decimal, a hexadecimal, and a character value in one REPLACE statement. Example 3 on page 332 shows a mixture of all three types of values in the REPLACE statement.

SAVE Statement

Saves the translation table in your SASUSER.PROFILE catalog

Featured in: Example 2 on page 330 and Example 4 on page 335

SAVE <TABLE=*table-name*> <ONE|TWO|BOTH>;

Options

TABLE=*table-name*
specifies the name under which the current table is to be saved. The name must be a valid one-level SAS name.

Default: If you omit the TABLE= option, the current table is saved under the name you specify in the PROC TRANTAB statement or the LOAD statement.

ONE | TWO | BOTH

ONE
saves table one.

TWO
saves table two.

BOTH
saves both table one and table two.

Default: BOTH

SWAP Statement

Exchanges table 1 with table 2 to enable you to edit positions 256 through 511

Tip: After you edit the table, you must the issue SWAP statement again to reposition the table.

Featured in: Example 7 on page 341

SWAP;

Examples: TRANTAB Procedure

Note: All examples were produced in the UNIX environment. △

Example 1: Viewing a Translation Table

Procedure features:
LIST statement

This example uses PROC TRANTAB to display the SAS supplied ASCII translation table.

Program

Set the options and specify a translation table.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
```

Display both halves of the translation table. The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

```
   list both;
```

SAS Log

```
NOTE: Table specified is ASCII.
ASCII table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '606162636465666768696A6B6C6D6E6F'x
     70 '707172737475767778797A7B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x


ASCII table 2:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '606162636465666768696A6B6C6D6E6F'x
     70 '707172737475767778797A7B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

Example 2: Creating a Translation Table

Procedures features:

LIST statement
REPLACE statement
SAVE statement

This example uses PROC TRANTAB to create a customized translation table.

Program

Set the system options and specify the translation table to edit.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=newtable;
```

Replace characters in the translation table starting at a specified position. The REPLACE statement places the values in the table starting at position 0. You can use hexadecimal strings of any length in the REPLACE statement. This example uses strings of length 16 to match the way that translation tables appear in the SAS log.

```
replace 0
'00010203a309e57ff9ecc40b0c0d0e0f'x
'10111213a5e008e71819c6c51c1d1e1f'x
'c7fce9e2e40a171beaebe8efee050607'x
'c9e616f4f6f2fb04ffd6dca2b6a7501a'x
'20e1edf3faf1d1aababfa22e3c282b7c'x
'265facbdbca1abbb5f5f21242a293bac'x
'2d2f5fa6a6a6a62b2ba6a62c255f3e3f'x
'a62b2b2b2b2b2d2d603a2340273d22'x
'2b616263646566676869 2d2ba6a62b2b'x
'2d6a6b6c6d6e6f7071722da62d2b2d2d'x
'2d7e737475767778787a2d2b2b2b2b2b'x
'2b2b2b5f5fa65f5f5fdf5fb65f5fb55f'x
'7b414243444546474849 5f5f5f5f5f5f'x
'7d4a4b4c4d4e4f5051525f5f5fb15f5f'x
'5c835354555657585 95a5f5ff75f5fb0'x
'30313233343536373839b75f6eb25f5f'x
;
```

Save the table. The SAVE statement saves the table under the name that is specified in the PROC TRANTAB statement. By default, the table is saved in your SASUSER.PROFILE catalog.

```
save;
```

Display both halves of the translation table in the SAS log. The LIST BOTH statement displays both the table that provides the translation and the table that provides the inverse translation.

```
list both;
```

SAS Log

Create and edit table 2. Table 2 is empty; that is, it consists entirely of 0s. To create table 2, you can use the INVERSE statement. (See Example 5 on page 337 .) To edit table 2, you can use the SWAP statement with the REPLACE statement. (See Example 7 on page 341.)

```
   NOTE: Table specified is NEWTABLE.
WARNING: Table NEWTABLE not found! New table is assumed.
NOTE: NEWTABLE table 1 is uninitialized.
NOTE: NEWTABLE table 2 is uninitialized.

NOTE: Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.
NEWTABLE table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '00010203A309E57FF9ECC40B0C0D0E0F'x
     10 '10111213A5E008E71819C6C51C1D1E1F'x
     20 'C7FCE9E2E40A171BEAEBE8EFEE050607'x
     30 'C9E616F4F6F2FB04FFD6DCA2B6A7501A'x
     40 '20E1EDF3FAF1D1AABABFA22E3C282B7C'x
     50 '265FACBDBCA1ABBB5F5F21242A293BAC'x
     60 '2D2F5FA6A6A6A62B2BA6A62C255F3E3F'x
     70 'A62B2B2B2B2B2B2D2D603A2340273D22'x
     80 '2B6162636465666768692D2BA6A62B2B'x
     90 '2D6A6B6C6D6E6F7071722DA62D2B2D2D'x
     A0 '2D7E737475767778787A2D2B2B2B2B2B'x
     B0 '2B2B2B5F5FA65F5F5FDF5FB65F5FB55F'x
     C0 '7B414243444546474849 5F5F5F5F5F5F'x
     D0 '7D4A4B4C4D4E4F5051525F5F5FB15F5F'x
     E0 '5C835354555657585 95A5F5FF75F5FB0'x
     F0 '30313233343536373839B75F6EB25F5F'x



NOTE: NEWTABLE table 2 is uninitialized.
NEWTABLE table 2:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '00000000000000000000000000000000'x
     10 '00000000000000000000000000000000'x
     20 '00000000000000000000000000000000'x
     30 '00000000000000000000000000000000'x
     40 '00000000000000000000000000000000'x
     50 '00000000000000000000000000000000'x
     60 '00000000000000000000000000000000'x
     70 '00000000000000000000000000000000'x
     80 '00000000000000000000000000000000'x
     90 '00000000000000000000000000000000'x
     A0 '00000000000000000000000000000000'x
     B0 '00000000000000000000000000000000'x
     C0 '00000000000000000000000000000000'x
     D0 '00000000000000000000000000000000'x
     E0 '00000000000000000000000000000000'x
     F0 '00000000000000000000000000000000'x
```

Example 3: Editing by Specifying a Decimal Value for Starting Position

Procedure features:

LIST statement

REPLACE statement

SAVE statement

This example edits the translation table that was created in Example 2 on page 330. The decimal value specified in the REPLACE statement marks the starting position for the changes to the table.

The vertical arrow in both SAS logs marks the point at which the changes begin.

Program 1: Display the Original Table

Set the system options and specify the translation table to edit.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=newtable;
```

Display the original table. This LIST statement displays the original NEWTABLE translation table.

```
   list one;
```

SAS Log

The Original NEWTABLE Translation Table

```
NOTE: Table specified is NEWTABLE.
NOTE: NEWTABLE table 2 is uninitialized.
NEWTABLE table 1:
                             ↓
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '00010203A309E57FF9ECC40B0C0D0E0F'x
     10 '10111213A5E008E71819C6C51C1D1E1F'x
     20 'C7FCE9E2E40A171BEAEBE8EFEE050607'x
     30 'C9E616F4F6F2FB04FFD6DCA2B6A7501A'x
     40 '20E1EDF3FAF1D1AABABFA22E3C282B7C'x
     50 '265FACBDBCA1ABBB5F5F21242A293BAC'x
     60 '2D2F5FA6A6A6A62B2BA6A62C255F3E3F'x
     70 'A62B2B2B2B2B2B2D2D603A2340273D22'x
     80 '2B616263646566676869 2D2BA6A62B2B'x
     90 '2D6A6B6C6D6E6F7071722DA62D2B2D2D'x
     A0 '2D7E737475767778787A2D2B2B2B2B2B'x
     B0 '2B2B2B5F5FA65F5F5FDF5FB65F5FB55F'x
     C0 '7B414243444546474849 5F5F5F5F5F5F'x
     D0 '7D4A4B4C4D4E4F5051525F5F5FB15F5F'x
     E0 '5C835354555657585 95A5F5FF75F5FB0'x
     F0 '303132333435363738 39B75F6EB25F5F'x
```

Program 2: Edit the Table

Replace characters in the translation table, starting at a specified position. The REPLACE statement starts at position decimal 10, which is byte 11 in the original table, and performs a byte-to-byte replacement with the given values.

```
replace 10
20 10 200 ’x’ ’ux’ ’092040’x;
```

Save the changes. The SAVE statement saves the changes that you made to the NEWTABLE translation table.

```
save;
```

Display the new table. The second LIST statement displays the edited NEWTABLE translation table.

```
list one;
```

SAS Log

The Edited NEWTABLE Translation Table

```
NOTE: Saving table NEWTABLE.
NOTE: NEWTABLE table 2 will not be saved because it is uninitialized.
NEWTABLE table 1:
                            ↓
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 ’00010203A309E57FF9EC140AC8787578’x
     10 ’09204013A5E008E71819C6C51C1D1E1F’x
     20 ’C7FCE9E2E40A171BEAEBE8EFEE050607’x
     30 ’C9E616F4F6F2FB04FFD6DCA2B6A7501A’x
     40 ’20E1EDF3FAF1D1AABABFA22E3C282B7C’x
     50 ’265FACBDBCA1ABBB5F5F21242A293BAC’x
     60 ’2D2F5FA6A6A6A62B2BA6A62C255F3E3F’x
     70 ’A62B2B2B2B2B2B2D2D603A2340273D22’x
     80 ’2B6162636465666768692D2BA6A62B2B’x
     90 ’2D6A6B6C6D6E6F7071722DA62D2B2D2D’x
     A0 ’2D7E737475767778787A2D2B2B2B2B2B’x
     B0 ’2B2B2B5F5FA65F5F5FDF5FB65F5FB55F’x
     C0 ’7B414243444546474849 5F5F5F5F5F5F’x
     D0 ’7D4A4B4C4D4E4F5051525F5F5FB15F5F’x
     E0 ’5C8353545556575858595A5F5FF75F5FB0’x
     F0 ’30313233343536373839B75F6EB25F5F’x
```

At position 10 (which is byte 11), a vertical arrow denotes the starting point for the changes to the translation table.

- □ At byte 11, decimal 20 (which is hexadecimal 14) replaces hexadecimal C4.

- At byte 12, decimal 10 (which is hexadecimal 0A) replaces hexadecimal 0B.
- At byte 13, decimal 200 (which is hexadecimal C8) replaces hexadecimal 0C.
- At byte 14, character 'x' (which is hexadecimal 78) replaces hexadecimal 0D.
- At bytes 15 and 16, characters 'ux' (which are hexadecimal 75 and 78, respectively) replace hexadecimal 0E and 0F.
- At bytes 17, 18, and 19, hexadecimal 092040 replaces hexadecimal 101112.

Example 4: Editing by Using a Quoted Character for Starting Position

Procedure features:

LIST statement
LOAD statement
REPLACE statement
SAVE statement

This example creates a new translation table by editing the already fixed ASCII translation table. The first occurrence of the hexadecimal equivalent of the quoted character that was specified in the REPLACE statement is the starting position for the changes to the table. This differs from Example 3 on page 332 in that you do not need to know the exact position at which to start the changes to the table. PROC TRANTAB finds the correct position for you.

The edited table is saved under a new name. Horizontal arrows in both SAS logs denote the edited rows in the translation table.

Program 1: Display the Original Table

Set the system options and specify which translation table to edit.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=ascii;
```

Display the translation table. The LIST statement displays the original translation table in the SAS log.

```
   list one;
```

SAS Log

```
NOTE: Table specified is ASCII.
ASCII table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '606162636465666768696A6B6C6D6E6F'x   ←
     70 '707172737475767778797A7B7C7D7E7F'x   ←
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

Program 2: Edit the Table

Replace characters in the translation table, starting at a specified position. The REPLACE statement finds the first occurrence of the hexadecimal "a" (which is 61) and replaces it, and the next 25 hexadecimal values, with the hexadecimal values for uppercase "A" through "Z."

```
replace 'a' 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
```

Save your changes. The SAVE statement saves the changes made to the ASCII translation table under the new table name UPPER. The stored contents of the ASCII translation table remain unchanged.

```
save table=upper;
```

Load and display the translation table. The LOAD statement loads the edited translation table UPPER. The LIST statement displays the translation table UPPER in the SAS log.

```
load table=upper;
list one;
```

SAS Log

The UPPER Translation Table

The horizontal arrows in the SAS log denote the rows in which the changes are made.

```
NOTE: Table UPPER being loaded.
UPPER table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '604142434445464748494A4B4C4D4E4F'x   ←
     70 '505152535455565758595A7B7C7D7E7F'x   ←
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

Example 5: Creating the Inverse of a Table

Procedure features:

INVERSE statement

LIST statement

SAVE statement

This example creates the inverse of the translation table that was created in Example 4 on page 335. The new translation table that is created in this example is the operating environment-to-device translation for use in data communications.

Program

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

Create the inverse translation table, save the tables, and display the tables. The INVERSE statement creates table 2 by inverting the original table 1 (called UPPER). The SAVE statement saves the translation tables. The LIST BOTH statement displays both the original translation table and its inverse.

```
inverse;
save;
list both;
```

SAS Log

The UPPER Translation Table and Its Inverse

The SAS log lists all the duplicate values that it encounters as it creates the inverse of table one. To conserve space, most of these messages are deleted in this example.

```
NOTE: Table specified is UPPER.
NOTE: This table cannot be mapped one to one.
 duplicate of '41'x found at '61'x in table one.
 duplicate of '42'x found at '62'x in table one.
 duplicate of '43'x found at '63'x in table one.
   .
   .
   .
 duplicate of '58'x found at '78'x in table one.
 duplicate of '59'x found at '79'x in table one.
 duplicate of '5A'x found at '7A'x in table one.
NOTE: Saving table UPPER.
UPPER table 1:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '604142434445464748494A4B4C4D4E4F'x
     70 '505152535455565758595A7B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x

UPPER table 2:
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '60000000000000000000000000000000'x
     70 '000000000000000000000007B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

The INVERSE statement lists in the SAS log all of the multiple translations that it encounters as it inverts the translation table. In Example 4 on page 335, all the lowercase letters were converted to uppercase in the translation table UPPER, which means that there are two sets of uppercase letters in UPPER. When INVERSE cannot make a translation, PROC TRANTAB fills the value with 00. Note that the inverse of the translation table UPPER has numerous 00 values.

Example 6: Using Different Translation Tables for Sorting

Procedure features:
PROC SORT statement option:
SORTSEQ=

Other features:
PRINT procedure

This example shows how to specify a different translation table to sort data in an order that is different from the default sort order. Characters that are written in a language other than U.S. English may require a sort order that is different from the default order.

Note: You can use the TRABASE program in the SAS Sample Library to create translation tables for several different languages. △

Program

Set the SAS system options.

```
options nodate pageno=1 linesize=80 pagesize=60;
```

Create the TESTSORT data set. The DATA step creates a SAS data set with four pairs of words, each pair differing only in the case of the first letter.

```
data testsort;
   input Values $10.;
   datalines;
Always
always
Forever
forever
Later
later
Yesterday
yesterday
;
```

Sort the data in an order that is different from the default sort order. PROC SORT sorts the data by using the default translation table, which sorts all lowercase words first, then all uppercase words.

```
proc sort;
   by values;
run;
```

Print the data set. PROC PRINT prints the sorted data set.

```
proc print noobs;
   title 'Default Sort Sequence';
run;
```

SAS Output

Output from Sorting Values with Default Translation Table

The default sort sequence sorts all the capitalized words in alphabetical order before it sorts any lowercase words.

```
                     Default Sort Sequence                         1

                           Values

                           Always
                           Forever
                           Later
                           Yesterday
                           always
                           forever
                           later
                           yesterday
```

Sort the data according to the translation table UPPER and print the new data set. The SORTSEQ= option specifies that PROC SORT sort the data according to the customized translation table UPPER, which treats lowercase and uppercase letters alike. This is useful for sorting without regard for case. PROC PRINT prints the sorted data set.

```
proc sort sortseq=upper;
   by values;
run;
proc print noobs;
   title 'Customized Sort Sequence';
run;
```

SAS Output

Output from Sorting Values with Customized Translation Table

The customized sort sequence sorts all the words in alphabetical order, without regard for the case of the first letters.

```
                        Customized Sort Sequence                        2

                               Values

                               Always
                               always
                               Forever
                               forever
                               Later
                               later
                               Yesterday
                               yesterday
```

Example 7: Editing Table 1 and Table 2

Procedure features:

LIST statement

REPLACE statement

SAVE statement

SWAP statement

This example shows how to edit both areas of a translation table. To edit positions 256 through 511 (table 2), you must

1 Issue the SWAP statement to have table 2 change places with table 1.

2 Issue an appropriate REPLACE statement to make changes to table two.

3 Issue the SWAP statement again to reposition the table.

Arrows in the SAS logs mark the rows and columns that are changed.

Program

Set the SAS system options and specify the translation table.

```
options nodate pageno=1 linesize=80 pagesize=60;
proc trantab table=upper;
```

Display the original translation table. The LIST statement displays the original UPPER translation table.

```
list both;
```

SAS Log

The Original UPPER Translation Table

```
NOTE: Table specified is UPPER.
UPPER table 1:
          ↓
         0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x   ←
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '604142434445464748494A4B4C4D4E4F'x
     70 '505152535455565758595A7B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x

UPPER table 2:
          ↓
         0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000102030405060708090A0B0C0D0E0F'x   ←
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '60000000000000000000000000000000'x
     70 '00000000000000000000007B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

Replace characters in the translation table starting at a specified position. The REPLACE statement starts at position 1 and replaces the current value of 01 with '0A'.

```
replace 1 '0A'x;
```

Prepare table 2 to be edited. The first SWAP statement positions table 2 so that it can be edited. The second REPLACE statement makes the same change in table 2 that was made in table 1.

```
swap;
replace 1 '0A'x;
```

Save and display the tables in their original positions. The second SWAP statement restores tables 1 and table 2 to their original positions. The SAVE statement saves both areas of the translation table by default. The LIST statement displays both areas of the table.

```
swap;
save;
list both;
```

SAS Log

The Edited UPPER Translation Table In byte 2, in both areas of the translation table, hexadecimal value '0A' replaces hexadecimal value 01. Arrows mark the rows and columns of the table in which this change is made.

```
NOTE: Table specified is UPPER.
UPPER table 1:
           ↓
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000A02030405060708090A0B0C0D0E0F'x   ←
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '604142434445464748494A4B4C4D4E4F'x
     70 '505152535455565758595A7B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x

UPPER table 2:
           ↓
          0 1 2 3 4 5 6 7 8 9 A B C D E F
     00 '000A02030405060708090A0B0C0D0E0F'x   ←
     10 '101112131415161718191A1B1C1D1E1F'x
     20 '202122232425262728292A2B2C2D2E2F'x
     30 '303132333435363738393A3B3C3D3E3F'x
     40 '404142434445464748494A4B4C4D4E4F'x
     50 '505152535455565758595A5B5C5D5E5F'x
     60 '60000000000000000000000000000000'x
     70 '00000000000000000000007B7C7D7E7F'x
     80 '808182838485868788898A8B8C8D8E8F'x
     90 '909192939495969798999A9B9C9D9E9F'x
     A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
     B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
     C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x
     D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x
     E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'x
     F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'x
```

See Also

Conceptual discussion about "Transcoding and Translation Tables" on page 22

System Options:

- "TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

NLS Options for Commands, Statements, and Procedures:

- "TRANTAB Statement" on page 391

PART 7

System Options for NLS

CHAPTER 16

Overview to SAS System Options for NLS

System Options for NLS by Category

The language control category of SAS system options are affected by NLS. The following table provides brief descriptions of the SAS system options. For more detailed descriptions, see the dictionary entry for each SAS system option:

Table 16.1 Summary of Categories of System Options for NLS

Category	System Options for NLS	Description
Environment control: Language control	"DATESTYLE= System Option" on page 349	Identifies the sequence of month, day, and year when the ANYDTDTM, ANYDTDTE, or ANYDTTME informats encounter input where the year, month, and day determination is ambiguous
	"DBCS System Option: UNIX, Windows, and z/OS" on page 349	Recognizes double-byte character sets (DBCS)
	"DBCSLANG System Option: UNIX, Windows, and z/OS" on page 350	Specifies a double-byte character set (DBCS) language
	"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351	Specifies the encoding method to use for a double-byte character set (DBCS)
	"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353	Specifies the language for international date informats and formats
	"ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS" on page 354	Specifies the default character-set encoding for the SAS session
	"FSDBTYPE System Option: UNIX" on page 356	Specifies a full-screen double-byte character set (DBCS) encoding method

Category	System Options for NLS	Description
	"FSIMM System Option: UNIX" on page 357	Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS)
	"FSIMMOPT System Option: UNIX" on page 357	Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS)
	"LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS" on page 358	Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region
	"NLSCOMPATMODE System Option: z/OS" on page 360	Provides national language compatibility with previous releases of SAS
	"PAPERSIZE= System Option: OpenVMS, UNIX, Window, and z/OS" on page 361	Specifies the paper size for the printer to use
	"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362	Specifies the translation tables that are used by various parts of SAS
Sort: Procedure options	"SORTSEQ= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 361	Specifies a language-specific collation sequence for the SORT procedure to use in the current SAS session

CHAPTER 17

System Options for NLS

DATESTYLE= System Option

Identifies the sequence of month, day, and year when the ANYDTDTM, ANYDTDTE, or ANYDTTME informats encounter input where the year, month, and day determination is ambiguous

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Language control

Input control: Data processing

PROC OPTIONS GROUP= INPUTCONTROL, LANGUAGECONTROL

See: DATESTYLE= system option in *SAS Language Reference: Dictionary*

DBCS System Option: UNIX, Windows, and z/OS

Recognizes double-byte character sets (DBCS)

Default: NODBCS

Valid in: configuration file, SAS invocation

UNIX specifics: Also valid in SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP= LANGUAGECONTROL

Syntax

-DBCS | -NODBCS (UNIX and Windows)

DBCS | NODBCS (z/OS)

DBCS
recognizes double-byte character sets (DBCS) for encoding values. DBCS encodings are used to support East Asian languages.

NODBCS
does not recognize a DBCS for encoding values. Instead, a single-byte character set (SBCS) is used for encoding values. A single byte is used to represent each character in the character set.

Details

The DBCS system option is used for supporting languages from East Asian countries such as Chinese, Japanese, Korean, and Taiwanese.

See Also

Conceptual Information:
Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29
"DBCS Values for a SAS Session" on page 405
Chapter 23, "Encoding Values in SAS Language Elements," on page 407
System Options:
"DBCSLANG System Option: UNIX, Windows, and z/OS" on page 350
"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351

DBCSLANG System Option: UNIX, Windows, and z/OS

Specifies a double-byte character set (DBCS) language

Default: none

Valid in: configuration file, SAS invocation

Category: Environment control: Language control

UNIX specifics: Also valid in SASV9_OPTIONS environment variable

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-DBCSLANG *language* (UNIX and Windows)

DBCSLANG = *language* (z/OS)

language
depends on the operating environment. The following table contains valid language values:

Table 17.1 Supported DBCS Languages According to Operating Environment

Language	z/OS	UNIX	Windows
CHINESE (simplified)	yes*	yes	yes
JAPANESE	yes	yes	yes
KOREAN	yes	yes	yes
TAIWANESE (traditional)	yes	yes	yes
NONE	yes	no	yes
UNKNOWN	yes	no	no

* For z/OS only, HANGUL and HANZI are valid aliases for CHINESE.

Details

The proper setting for the DBCSLANG system option depends on which setting is used for the DBCSTYPE system option. Some of the settings of DBCSTYPE support all of the DBCSLANG languages, while other settings of DBCSTYPE support only Japanese.

CHINESE specifies the language used in the People's Republic of China, which is known as simplified Chinese. TAIWANESE specifies the Chinese language used in Taiwan, which is known as traditional Chinese.

See Also

Conceptual discussion about Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29

"DBCS Values for a SAS Session" on page 405

Chapter 23, "Encoding Values in SAS Language Elements," on page 407

System Options:

"DBCS System Option: UNIX, Windows, and z/OS" on page 349

"DBCSTYPE System Option: UNIX, Windows, and z/OS" on page 351

DBCSTYPE System Option: UNIX, Windows, and z/OS

Specifies the encoding method to use for a double-byte character set (DBCS)

z/OS Default: IBM

UNIX Default: Depends on the specific machine

Windows Default: PCMS

Valid in: configuration file, SAS invocation

Category: Environment control: Language control

UNIX specifics: Also valid in SASV9_OPTIONS environment variable

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-DBCSTYPE *encoding-method* (UNIX and Windows)

DBCSTYPE = *encoding-method* (z/OS)

encoding-method
specifies the method that is used to encode a double-byte character set (DBCS). Valid values for *encoding-method* depend on the standard that the computer hardware manufacturer applies to the operating environment.

Details

DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.

The DBCSLANG= system option specifies the language that the encoding method is applied to. You should specify DBCSTYPE= only if you also specify the DBCS and DBCSLANG= system options.

z/OS DBCSTYPE= supports the DBCSTYPE= value of IBM.

Operating Environment-Specific DBCSTYPE= Values

Table 17.2 DBCS Encoding Methods for z/OS

DBCSTYPE= Value	Description
IBM	IBM PC encoding method

Table 17.3 DBCS Encoding Methods for UNIX

DBCSTYPE= Value	Description
DEC	DEC encoding method
EUC	Extended UNIX Code encoding method
HP15	Hewlett Packard encoding method
PCIBM	IBM PC encoding method
PCMS	Microsoft PC encoding method

DBCSTYPE= Value	Description
SJIS	Shift-JIS encoding method for the Japanese language only
NONE	Disables DBCS processing

Table 17.4 DBCS Encoding Methods for Windows

DBCSTYPE= Value	Description
PCMS	Microsoft PC encoding method
WINDOWS	Alias for PCMS
SJIS	Shift-JIS encoding method for the Japanese language only

See Also

Conceptual Information:
Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29
"DBCS Values for a SAS Session" on page 405
Chapter 23, "Encoding Values in SAS Language Elements," on page 407
System Options:
"DBCS System Option: UNIX, Windows, and z/OS" on page 349
"DBCSLANG System Option: UNIX, Windows, and z/OS" on page 350

DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS

Specifies the language for international date informats and formats

Default: English

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

DFLANG='*language*'

Syntax Description

'language'
specifies the language that is used for international date informats and formats.
These are valid values for *language*:

- Afrikaans
- Catalan
- Croatian
- Czech
- Danish
- Dutch
- English
- Finnish
- French
- German
- Hungarian
- Italian
- Japanese
- Macedonian
- Norwegian
- Polish
- Portuguese
- Russian
- Slovenian
- Spanish
- Swedish
- Swiss_French
- Swiss_German

Details

You can change the value during a SAS session, but you can use only one language at a time. The values for *language* are not case-sensitive.

See Also

Chapter 8, "Overview to Formats for NLS," on page 47

ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS

Specifies the default character-set encoding for the SAS session

OpenVMS and UNIX Default: latin1

z/OS Default: OPEN_ED-1047

Windows Default: wlatin1

Valid in: configuration file, SAS invocation

Category: Environment control: Language control

OpenVMS specifics: Also valid in VMS_SAS_OPTIONS DCL symbol

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-ENCODING= ASCIIANY | EBCDICANY | *encoding-value* (UNIX and Windows)

ENCODING= *encoding-value* (OpenVMS, UNIX, Windows, and z/OS)

ASCIIANY

is valid only for OpenVMS, UNIX, and OpenVMS. Transcoding normally occurs when SAS detects that the session encoding and data set encoding are different. ASCIIANY enables you to create a data set that SAS will not transcode if the SAS session that accesses the data set has a session that encoding value of ASCII. If you transfer the data set to a machine that uses EBCDIC encoding, transcoding occurs.

Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant. △

EBCDICANY

is valid only for z/OS. Transcoding normally occurs when SAS detects that the session encoding and the data set encoding are different. EBCDICANY enables you to create a data set that SAS will not transcode if the SAS session accessing the data set has a session encoding value of EBCDIC. If you transfer the data set to a machine that uses ASCII encoding, transcoding occurs.

encoding-value

For valid values for all operating environments, see Chapter 24, "Encoding Values for a SAS Session," on page 413.

Details

A character-set encoding is a set of characters that have been mapped to numeric values called *code points*.

The ENCODING= system option is valid only when the NONLSCOMPATMODE system option is set.

The encoding for a SAS session is determined by the values of the ENCODING=, LOCALE=, DBCSTYPE=, and DBCSLANG= system options as follows:

- If the ENCODING= and LOCALE= system options are not specified, the default value is ENCODING=. For OpenVMS and UNIX, the default value is `latin1`; for Windows, the default value is wlatin1; for z/OS, the default is OPEN_ED-1047.
- If both LOCALE= and ENCODING= are specified, the session encoding is the value that is specified by the ENCODING= option.
- If LOCALE= is specified and ENCODING= is not specified, SAS infers the appropriate encoding value from the LOCALE= value.
- If the DBCS option is set, the values for the DBCSLANG= and DBCSTYPE= system options determine the ENCODING= and LOCALE= values.

See Also

Conceptual Information:

"Overview of Locale Concepts for NLS" on page 5

Conceptual discussion about "Overview of Encoding for NLS" on page 9

Conceptual discussion about "Overview to Transcoding" on page 21
Chapter 21, "Values for the LOCALE= System Option," on page 397
Chapter 22, "SAS System Options for Processing DBCS Data," on page 405
Chapter 23, "Encoding Values in SAS Language Elements," on page 407

FSDBTYPE System Option: UNIX

Specifies a full-screen double-byte character set (DBCS) encoding method

Default: DEFAULT

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

UNIX specifics: all

Syntax

-FSDBTYPE *encoding-method*

Details

The FSDBTYPE= system option specifies the encoding method that is appropriate for a full-screen DBCS enabling method. Full-screen DBCS encoding methods vary according to the computer hardware manufacturer and the standards organization.

Table 17.5 Full-Screen DBCS Encoding Methods

FSDBTYPE= Encoding Method	Description
dec	Digital Equipment Corporation encoding method
euc	Extended UNIX encoding method
hp15	HP-UX encoding method
jis7	7-bit Shift-JIS encoding method used in an X windows environment for the Japanese language only
pcibm	IBM PC encoding method
sjis	Shift-JIS encoding method for the Japanese language only
default	default method that is used by the specific host

See Also

Conceptual Information:

Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29

"DBCS Values for a SAS Session" on page 405
Chapter 23, "Encoding Values in SAS Language Elements," on page 407

FSIMM System Option: UNIX

Specifies input method modules (IMMs) for full-screen double-byte character set (DBCS)

Default: none
Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable
Category: Environment control: Language control
PROC OPTIONS GROUP: LANGUAGECONTROL
UNIX specifics: all

Syntax

-FSIMM *fsdevice_name=IMM-name1*<, *fsdevice_name=IMM-name2*>...

Details

You can specify the following values for *IMM-name*:

TTY | SASWUJT
: provides an interface for **/dev/tty**. This IMM enables you to enter DBCS strings through a terminal emulator that has DBCS input capability.

PIPE | SASWUJP
: provides a pipe interface. This interface forks the DBCS input server process. The default server name is **saswujms**, which uses the vendor-supplied MOTIF toolkit.

For example, to use the PIPE input method module for X11.motif drivers, you would specify:

```
-FSIMM X11.motif=PIPE
```

Note: The server is specified by using the FSIMMOPT option. △

See Also

Conceptual Information:
Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29
System Option:
"FSIMMOPT System Option: UNIX" on page 357

FSIMMOPT System Option: UNIX

Specifies options for input method modules (IMMs) that are used with a full-screen double-byte character set (DBCS)

Default: none

Valid in: configuration file, SAS invocation, SASV9_OPTIONS environment variable

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

UNIX specifics: all

Syntax

-FSIMMOPT *fullscreen-IMM*:*IMM-option*

Details

The FSIMMOPT system option specifies an option for each full-screen IMM (input method module). You can specify only one FSIMMOPT option for each IMM. If you specify multiple FSIMMOPT options for the same IMM, only the last specification is used.

For option values for each IMM, see SAS Technical Report J-121, *DBCS Support Usage Guide* (in Japanese).

For example, you can use the FSIMMOPT option to specify the name of the server, MOTIF, to be used for the PIPE IMM:

```
-fsimmopt PIPE:MOTIF
```

See Also

Conceptual Information:

Chapter 5, "Double-Byte Character Sets (DBCS)," on page 29

System Option:

"FSIMM System Option: UNIX" on page 357

LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS

Specifies a set of attributes in a SAS session that reflect the language, local conventions, and culture for a geographical region

Default: English_UnitedStates

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Language control

OpenVMS specifics: Also valid in VMS_SAS_OPTIONS DCL symbol

UNIX specifics: Also valid in SASV9_OPTIONS environment variable

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-LOCALE *locale-name* (UNIX and Windows)

LOCALE=*locale-name* (OpenVMS, UNIX, Windows, and z/OS)

locale-name
For a complete list of locale values (SAS names and POSIX names), see Chapter 21, "Values for the LOCALE= System Option," on page 397.

Details

The LOCALE= system option is used to specify the locale, which reflects the local conventions, language, and culture a geographical region.

If the value of the LOCALE= system option is not compatible with the value of the ENCODING= system option, the character-set encoding is determined by the value of the ENCODING= system option.

If the DBCS= system option is active, the values of the DBCSTYPE= and DBCSLANG= system options determine the locale and character-set encoding.

When you set a value for LOCALE=, the value of the following system options are modified unless explicit values have been specified:

ENCODING=
The locale that you set has a common encoding value that is used most often in the operating environment where SAS runs. If you start SAS with the LOCALE= system option and you do not specify the ENCODING= system option, SAS compares the default value for ENCODING= and the most common locale encoding value. If the two encoding values are not the same, the ENCODING= system option is set to the LOCALE= encoding value. When the ENCODING= system option is set, the TRANTAB= system option is also set.

DATESTYLE=
When LOCALE= is set, the DATESTYLE= system option uses the value that corresponds to the chosen locale.

DFLANG=
When LOCALE= is set, the DFLANG= system option is set to a value that corresponds to the chosen locale.

PAPERSIZE=
When LOCALE= is set, the PAPERSIZE= system option is set to a value that corresponds to the chosen locale and the ODS printer is set to the preferred unit of measurement, inches or centimeters, for that locale.

CAUTION:
Under the Windows operating systems only: The LOCALE= option can be used to specify PAPERSIZE= only if the UNIVERSALPRINT and UPRINTMENUSWITCH system options are also specified. For details about the UNIVERSALPRINT system option, see *SAS Language Reference: Dictionary*. For details about the UPRINTMENUSWITCH system option, see *SAS Companion for Windows*. △

See Also

Conceptual Information:

Chapter 2, "Locale for NLS," on page 5

Chapter 21, "Values for the LOCALE= System Option," on page 397

System Options:

"ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS" on page 354

DATESTYLE in *SAS Language Reference: Dictionary*

"DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 353

PAPERSIZE in *SAS Language Reference: Dictionary*

"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

NLSCOMPATMODE System Option: z/OS

Provides national language compatibility with previous releases of SAS

Default: NONLSCOMPATMODE

Valid in: configuration file, SAS invocation

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

NLSCOMPATMODE | NONLSCOMPATMODE

NLSCOMPATMODE
provides compatibility with previous releases of SAS in order to process data in languages other than English, which is the default language. Programs that ran in previous releases of SAS will continue to work when NLSCOMPATMODE is set.

Note: NLSCOMPATMODE might affect the format of outputs that are produced using ODS. If you are using ODS, set the option value to NONLSCOMPATMODE. △

NONLSCOMPATMODE
provides support for data processing using native characters for languages other than English. When NONLSCOMPATMODE is set, character data is processed using the encoding that is specified for the SAS session.

When NONLSCOMPATMODE is in effect, SAS does not support substitution characters in SAS syntax. If you run SAS with NONLSCOMPATMODE, you must update existing programs to use national characters instead of substitution characters. For example, Danish customers who have substituted the 'Å' for the '$' character in existing SAS programs will have to update the SAS syntax to use the '$' in their environments.

Details

The NONLSCOMPATMODE system option is provided for international customers who use non-English encodings and who want to take advantage of emerging industry standards when they are coding new applications.

The NLSCOMPATMODE or NONLSCOMPATMODE settings do not change the value of the LOCALE or ENCODING system options. If the ENCODING option is set, you will see the encoding values that you specified when you display the option, even though parts of SAS will not use the encoding value for processing when NLSCOMPATMODE is in effect.

PAPERSIZE= System Option: OpenVMS, UNIX, Window, and z/OS

Specifies the paper size for the printer to use

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

See: PAPERSIZE= System Option in *SAS Language Reference: Dictionary*

SORTSEQ= System Option: OpenVMS, UNIX, Windows, and z/OS

Specifies a language-specific collation sequence for the SORT procedure to use in the current SAS session

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Sort: Procedure options

PROC OPTIONS GROUP: SORT

Syntax

SORTSEQ=*collation-sequence*

Syntax Description

collation-sequence

specifies the collation sequence that the SORT procedure is to use in the current SAS session. Valid values can be user-supplied, or they can be one of the following:

- ASCII
- DANISH (alias NORWEGIAN)
- EBCDIC
- FINNISH
- ITALIAN
- NATIONAL
- POLISH
- REVERSE
- SPANISH
- SWEDISH

Details

To create or change a collation sequence, use the TRANTAB procedure to create or modify translation tables. When you create your own translation tables, they are stored in your PROFILE catalog, and they override any translation tables with the same name that are stored in the HOST catalog.

Note: System managers can modify the HOST catalog by copying newly created tables from the PROFILE catalog to the HOST catalog. All users can access the new or modified translation tables. △

If you are in a windowing environment, use the Explorer window to display the SASHELP HOST catalog. In the HOST catalog, entries of type TRANTAB contain collation sequences that are identified by the entry name.

If you are not in a windowing environment, issue the following statements to generate a list of the contents of the HOST catalog. Collation sequences are entries of the type TRANTAB.

```
proc catalog catalog=sashelp.host;
   contents;
run;
```

To see the contents of a particular translation table, use these statements:

```
proc trantab table=translation-table-name;
   list;
run;
```

The contents of collation sequences are displayed in the SAS log.

See Also

"Collation Sequence" on page 16

Data Set Options:

"SORTSEQ= Data Set Option" on page 42

System Options:

"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS

Specifies the translation tables that are used by various parts of SAS

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Environment control: Language control

PROC OPTIONS GROUP: LANGUAGECONTROL

Interaction: The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers. The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred.

Syntax

TRANTAB=(*catalog-entries*)

Note: TRANTAB= was introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on the features of TRANTAB=. SAS 9.1 supports TRANTAB= for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. △

Syntax Description

catalog-entries
specifies SAS catalog entries that contain translation tables. If you specify *entry-name.type*, SAS searches SASUSER.PROFILE first and then SASUSER.HOST.

Details

Translation tables are specified in a parenthesized list that has ten positions. The position in which a table appears in the list determines the type of translation table that is specified. Individual entries in the list are separated by commas. See the list of positions and types that follows:

Position	Type of Translation Table
1st	local-to-transport-format
2nd	transport-to-local-format
3rd	lowercase-to-uppercase
4th	uppercase-to-lowercase
5th	character classification
6th	scanner translation
7th	delta characters
8th	scanner character classification
9th	not used
10th	DBCS user table

CAUTION:
Do not change a translation table unless you are familiar with its purpose. Translation tables are used internally by the SAS supervisor to implement NLS. If you are unfamiliar with the purpose of translation tables, do *not* change the specifications without proper technical advice. △

To change one table, specify null entries for the other tables. For example, to change the lowercase-to-uppercase table, which is third in the list, specify uppercase as follows:

```
options trantab = ( , , new-uppercase-table);
```

The other tables remain unchanged. The output from the OPTIONS procedure reflects the last specification for the TRANTAB= option and not the composite specification. Here is an example:

```
options trantab = ( , , new-uppercase-table);
options trantab = ( , , , new-lowercase-table);
```

PROC OPTIONS shows that the value for TRANTAB= is (, , , *new-lowercase-table*), but both the *new-uppercase* and *new-lowercase* tables are in effect.

See Also

Chapter 15, "The TRANTAB Procedure," on page 319

PART 8

Other Commands, Statements, and Procedure Statements for NLS

CHAPTER 18

Overview to NLS Options Used in Commands, Statements, and Procedures

Commands, Statements, and Procedures for NLS by Category

The data set control and data access categories of options for selected SAS statements are affected by NLS. The following table provides brief descriptions of the statement options. For more detailed descriptions, see the dictionary entry for each statement option:

Table 18.1 Summary of NLS Statements by Category

Category	Statements for NLS	Description
Data Access	"CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options" on page 373	Specifies attributes for character variables that are needed in order to transcode a SAS file
	"ENCODING= Option" on page 378	Overrides and transcodes the encoding for input or output processing of external files
	"INENCODING= and OUTENCODING= Options" on page 381	Overrides and changes the encoding when reading or writing SAS data sets in the SAS data library
	"ODSCHARSET= Option" on page 382	Specifies the character set to be generated in the META declaration for the output
	"ODSTRANTAB = Option" on page 383	Specifies the translation table to use when transcoding an XML document for an output file
	"RENCODING= Option" on page 385	Specifies the ASCII-based or EBCDIC-based encoding to use for transcoding data for a SAS/SHARE server session that is using an EBCDICANY or ASCIIANY session encoding
	"XMLENCODING= Option" on page 390	Overrides the encoding of an XML document to import or export an external document
Information	"TRANSCODE= Option" on page 387	Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and/or length with one or more variables) that indicates whether character variables are to be transcoded

Category	Statements for NLS	Description
ODS: Third-Party Formatted	"CHARSET= Option" on page 369	Specifies the character set to be generated in the META declaration for the output
	"TRANTAB= Option" on page 389	Specifies the translation table to use when you are transcoding character data in a SAS file for the appropriate output file

CHAPTER 19

Options for Commands, Statements, and Procedures for NLS

CHARSET= Option

Specifies the character set to be generated in the META declaration for the output

Valid in: LIBNAME statement for the ODS MARKUP and ODS HTML statements

Category: ODS: Third-Party Formatted

Syntax

CHARSET=*character-set* ;

Arguments

character-set
Specifies the character set to use in the META tag for HTML output.

An example of an encoding is ISO-8859-1. Official character sets for use on the Internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, and options, codes and types. For a complete list of character-set values, visit **www.unicode.org/reports/tr22/index.html** and **www.iana.org/assignments/character-sets**.

Note: A *character set* is like an *encoding-value* in this context. However, *character set* is the term that is used to identify an encoding that is suitable for use on the Internet. △

Examples

Example 1: Generated Output in a META Declaration for an ODS MARKUP Statement

```
<META http-equiv="Content-Type" content="text/html; charset=iso-8858-1">
```

See Also

Conceptual Information:
  Chapter 3, "Encoding for NLS," on page 9
Statements:
  ODS MARKUP in *SAS Output Delivery System: User's Guide*
  ODS HTML in *SAS Output Delivery System: User's Guide*

Collation Sequence Option

Specifies the collation sequence for PROC SORT

Valid in: PROC SORT statement

PROC SORT statement: Sorts observations in a SAS data set by one or more characters or numeric variables

Syntax

PROC SORT *collation-sequence-option*;

Options

Specify the Collation Sequence	Use this option
Specify ASCII	ASCII
Specify EBCDIC	EBCDIC
Specify Danish	DANISH
Specify Finnish	FINNISH
Specify Norwegian	NORWEGIAN
Specify Polish	POLISH
Specify Swedish	SWEDISH
Specify a customized sequence	NATIONAL
Specify any of these collating sequences: ASCII, EBCDIC, DANISH, FINNISH, ITALIAN, NORWEGIAN, POLISH, SPANISH, SWEDISH	SORTSEQ=

Explanations for the options follow:

ASCII
: sorts character variables using the ASCII collation sequence. You need this option only when you sort by ASCII on a system where EBCDIC is the native collation sequence.

DANISH
NORWEGIAN
: sorts characters according to the Danish and Norwegian national standard.

EBCDIC
: sorts character variables using the EBCDIC collation sequence. You need this option only when you sort by EBCDIC on a system where ASCII is the native collation sequence.

FINNISH
SWEDISH
: sorts characters according to the Finnish and Swedish national standard.

NATIONAL
: sorts character variables using an alternate collation sequence, as defined by your installation, to reflect a country's National Use Differences. To use this option, your site must have a customized national sort sequence defined. Check with the SAS Installation Representative at your site to determine if a customized national sort sequence is available.

NORWEGIAN
: See DANISH.

POLISH
: sorts characters according to the Polish national standard.

SWEDISH
: See FINNISH.

SORTSEQ=*collation-sequence*
: specifies the collation sequence. The value of *collation-sequence* can be any one of the *collation-sequence-options* in the PROC SORT statement, or the value can be the name of a translation table, either a default translation table or one that you have created in the TRANTAB procedure. For an example of using PROC TRANTAB and PROC SORT with SORTSEQ=, see Example 6 on page 339. These are the available translation tables:

Danish

Finnish

Italian

Norwegian

Polish

Spanish

Swedish

CAUTION:
If you use a host sort utility to sort your data, then specifying the SORTSEQ= option might corrupt the character BY variables. For more information, see the PROC SORT documentation for your operating environment. △

See Also

"Collation Sequence" on page 16
System Options:
"SORTSEQ= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 361
"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362
Data Set Option:
"SORTSEQ= Data Set Option" on page 42

CORRECTENCODING = Option

Explicitly changes the encoding attribute of a SAS file to match the encoding of the data in the SAS file

Valid in: MODIFY statement of the DATASETS procedure

Syntax

MODIFY *SAS file* </<CORRECTENCODING=*encoding-value*>> ;

Options

</ <CORRECTENCODING=*encoding-value*> >
enables you to change the encoding indicator, which is recorded in the file's descriptor information, in order to match the actual encoding of the file's data. You cannot use this option in parenthesis after the name of each SAS file; you must specify CORRECTENCODING= after the forward slash. For example:

```
modify mydata / correctencoding=latin2;
```

For a list of valid encoding values for transcoding, see "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 407.

Restriction: CORRECTENCODING= can be used only when the SAS file uses the default base engine, which is V9 in SAS 9.

Examples

Example 1: Using the CORRECTENCODING= Option to Resolve a SAS Session Encoding and a SAS File Encoding A file's encoding indicator can be different from the data's encoding. For example, a SAS file that was created prior to SAS 9 has no encoding indicator stored on the file. If such a SAS file that has no recorded encoding is opened in a SAS 9 session, SAS assigns the encoding of the current session. For example, if the encoding of the data is Danish EBCDIC, but the encoding for the current session is Western Wlatin1, then the actual encoding of the file's data and the encoding indicator that is stored in the file's descriptor information do not match. When this occurs, the data does not transcode correctly and could result in unreadable output. The following MODIFY statement would resolve the problem by explicitly assigning an EDCDIC encoding:

```
proc datasets library=myfiles;
   modify olddata / correctencoding=ebcdic1142;
quit;
```

CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options

Specifies attributes for character variables that are needed in order to transcode a SAS file

Valid in: LIBNAME statement

Category: Data Access

PROC OPTIONS GROUP: LIBNAME statement in the documentation for your operating environment

See Also: LIBNAME, SAS/ACCESS

Syntax

LIBNAME *libref* <CVPBYTES=*bytes*> <CVPENGINE=*engine*> <CVPMULTIPLIER=*multiplier*> '*SAS data-library*';

Options

CVPBYTES=*bytes*

specifies the number of bytes by which to expand character variable lengths when processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by adding the specified value to the current length. You can specify a value from 0 to 32766.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPBYTES= option.

```
libname expand 'SAS data-library' cvpbytes=5;
```

Character variable lengths are increased by adding 5 bytes. A character variable with a length of 10 is increased to 15, and a character variable with a length of 100 is increased to 105.

Default: If you specify CVPBYTES=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to your specification. If you explicitly assign the CVP engine but do not specify either CVPBYTES= or CVPMULTIPLIER=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths of the character variables.

Requirement: The number of bytes that you specify must be large enough to accommodate any expansion; otherwise, truncation will still occur, which results in an error message in the SAS log.

Restriction: The CVP engine supports SAS data files only; that is, no SAS views, catalogs, item stores, and so on.

Restriction: The CVP engine is available for input (read) processing only.

Limitation: For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

Interaction: You cannot specify both CVPBYTES= and CVPMULTIPLIER=. Specify one of these options.

Featured in: Example 1 on page 374

See also: "Avoiding Character Data Truncation by Using the CVP Engine" on page 32

CVPENGINE=*engine*

specifies the engine to use in order to process the SAS file. The CVP engine expands the character variable lengths prior to transcoding so that character data truncation does not occur. Then the specified engine does the actual file processing.

Alias: CVPENG

Default: SAS uses the default SAS engine.

See also: "Avoiding Character Data Truncation by Using the CVP Engine" on page 32

CVPMULTIPLIER=*multiplier*

specifies a multiplier value in order to expand character variable lengths when you are processing a SAS data file that requires transcoding. The CVP engine expands the lengths so that character data truncation does not occur. The lengths for character variables are increased by multiplying the current length by the specified value. You can specify a multiplier value from 1 to 5.

For example, the following LIBNAME statement implicitly assigns the CVP engine by specifying the CVPMULTIPLIER= option.

```
libname expand 'SAS data-library' cvpmultiplier=2.5;
```

Character variable lengths are increased by multiplying the lengths by 2.5. A character variable with a length of 10 is increased to 25, and a character variable with a length of 100 is increased to 250.

Alias: CVPMULT

Default: If you specify CVPMULTIPLIER=, SAS automatically uses the CVP engine in order to expand the character variable lengths according to your specification. If you explicitly specify the CVP engine but do not specify either CVPMULTIPLIER= or CVPBYTES=, then SAS uses CVPMULTIPLIER=1.5 to increase the lengths.

Requirement: The number of bytes that you specify must be large enough to accommodate any expansion; otherwise, truncation will still occur, which results in an error in the SAS log.

Restriction: The CVP engine supports SAS data files only; that is, no SAS views, catalogs, item stores, and so on.

Restriction: The CVP engine is available for input (read) processing only.

Limitation: For library concatenation with mixed engines that include the CVP engine, only SAS data files are processed. For example, if you execute the COPY procedure, only SAS data files are copied.

Interaction: You cannot specify both CVPMULTIPLIER= and CVPBYTES=. Specify one of these options.

See also: "Avoiding Character Data Truncation by Using the CVP Engine" on page 32

Examples

Example 1: Using the CVP (Character Variable Padding) Engine The following example illustrates how to avoid character data truncation by using the CVP engine. The example uses a SAS data set named MYFILES.WLATIN2, which contains some national characters in Wlatin2 encoding.

Output 19.1 PROC PRINT Output for MYFILES.WLATIN2

```
                    The SAS System                              1

          Obs     var1     var2     var3     var4

           1       A        |        ð         3
```

Here is PROC CONTENTS output for MYFILES.WLATIN2, which shows that the encoding is Wlatin2 and that the length for each character variable is 1 byte:

Output 19.2 PROC CONTENTS Output for MYFILES.WLATIN2

```
                                   The SAS System                                        1

                                  The CONTENTS Procedure

       Data Set Name        MYFILES.WLATIN2                          Observations          1
       Member Type          DATA                                     Variables             4
       Engine               V9                                       Indexes               0
       Created              Thursday, November 07, 2003 02:02:36     Observation Length    4
       Last Modified        Thursday, November 07, 2003 02:02:36     Deleted Observations  0
       Protection                                                    Compressed            NO
       Data Set Type                                                 Sorted                NO
       Label
       Data Representation  WINDOWS_32
       Encoding             wlatin2  Central Europe (Windows)


                             Engine/Host Dependent Information

       Data Set Page Size          4096
       Number of Data Set Pages    1
       First Data Page             1
       Max Obs per Page            987
       Obs in First Data Page      1
       Number of Data Set Repairs  0
       File Name                   C:\Documents and Settings\xxxxxx\My
                                   Documents\myfiles\wlatin2.sas7bdat
       Release Created             9.0100A0
       Host Created                XP_PRO


                        Alphabetic List of Variables and Attributes

                               #    Variable    Type    Len

                               1    Var1        Char      1
                               2    Var2        Char      1
                               3    Var3        Char      1
                               4    Var4        Char      1
```

The following code is executed with the session encoding Wlatin2.

```
options msglevel=i;
libname myfiles 'SAS data-library';

data myfiles.utf8 (encoding="utf-8");
```

```
   set myfiles.wlatin2;
run;
```

The DATA step requests a new data set named MYFILES.UTF8, and requests that the data be read into the new data set in UTF-8 encoding, which means that the data must be transcoded from Wlatin2 to UTF-8. The request results in errors due to character data truncation that occurs from the transcoding. The new data set MYFILES.UTF8 is created but does not contain any data.

Output 19.3 SAS Log with Transcoding Error

```
   1    options msglevel=i;
   2    libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
   NOTE: Libref MYFILES was successfully assigned as follows:
         Engine:        V9
         Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
   3    data myfiles.utf8 (encoding="utf-8");
   4       set myfiles.wlatin2;
   5    run;

INFO: Data file MYFILES.UTF8.DATA is in a format native to another
host or the file encoding does not match the session encoding.
Cross Environment Data Access will be used, which may require additional
CPU resources and reduce performance.
   ERROR: Some character data was lost during transcoding in the dataset MYFILES.UTF8.
   NOTE: The data step has been abnormally terminated.
   NOTE: The SAS System stopped processing this step because of errors.
   NOTE: There were 1 observations read from the data set MYFILES.WLATIN2.
   WARNING: The data set MYFILES.UTF8 may be incomplete.  When this step was stopped there were 0
            observations and 4 variables.
```

The following code is executed again with the session encoding Wlatin2.

```
options msglevel=i;
libname myfiles 'SAS data-library';

libname expand cvp 'SAS data-library' cvpbytes=2;

data myfiles.utf8 (encoding="utf-8");
   set expand.wlatin2;
run;
```

In this example, the CVP engine is used to expand character variable lengths by adding 2 bytes to each length. The data is read into the new file in UTF-8 encoding by transcoding from Wlatin2 to UTF-8. There is no data truncation due to the expanded character variable lengths, and the new data set is successfully created:

Output 19.4 SAS Log Output for MYFILES.UTF8

```
   12   options msglevel=i;
   13   libname myfiles 'C:\Documents and Settings\xxxxxx\My Documents\myfiles';
   NOTE: Directory for library MYFILES contains files of mixed engine types.
   NOTE: Libref MYFILES was successfully assigned as follows:
         Engine:        V9
         Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
   14   libname expand cvp 'C:\Documents and Settings\xxxxxx\My Documents\myfiles' cvpbytes=2;
   WARNING: Libname EXPAND refers to the same physical library as MYFILES.
   NOTE: Libref EXPAND was successfully assigned as follows:
         Engine:        CVP
         Physical Name: C:\Documents and Settings\xxxxxx\My Documents\myfiles
   15   data myfiles.utf8 (encoding="utf-8");
   16      set expand.wlatin2;
   17   run;

INFO: Data file MYFILES.UTF8.DATA is in a format native to another
host or the file encoding does not match the session encoding.
Cross Environment Data Access will be used, which may require additional
CPU resources and reduce performance.
   NOTE: There were 1 observations read from the data set EXPAND.WLATIN2.
   NOTE: The data set MYFILES.UTF8 has 1 observations and 4 variables.
```

Finally, here is PROC CONTENTS output for MYFILES.UTF8 showing that it is in UTF-8 encoding and that the length of each character variable is 3:

Output 19.5 PROC CONTENTS Output for MYFILES.UTF8

```
                                    The SAS System                                        1

                                The CONTENTS Procedure

     Data Set Name        MYFILES.UTF8                            Observations          1
     Member Type          DATA                                    Variables             4
     Engine               V9                                      Indexes               0
     Created              Thursday, November 07, 2003 02:40:34    Observation Length    12
     Last Modified        Thursday, November 07, 2003 02:40:34    Deleted Observations  0
     Protection                                                   Compressed            NO
     Data Set Type                                                Sorted                NO
     Label
     Data Representation  WINDOWS_32
     Encoding             utf-8  Unicode (UTF-8)


                           Engine/Host Dependent Information

     Data Set Page Size          4096
     Number of Data Set Pages    1
     First Data Page             1
     Max Obs per Page            335
     Obs in First Data Page      1
     Number of Data Set Repairs  0
     File Name                   C:\Documents and Settings\xxxxxx\My Documents\myfiles\utf8.sas7bdat
     Release Created             9.0100A0
     Host Created                XP_PRO


                        Alphabetic List of Variables and Attributes

                               #    Variable    Type    Len

                               1    Var1        Char      3
                               2    Var2        Char      3
                               3    Var3        Char      3
                               4    Var4        Char      3
```

ENCODING= Option

Overrides and transcodes the encoding for input or output processing of external files

Valid in: %INCLUDE statement; FILE statement; FILENAME statement; FILENAME statement, EMAIL (SMTP) Access Method; INFILE statement; ODS statements; FILE command; INCLUDE command

%INCLUDE statement: Reads SAS statements and data lines from the specified source file

Category: Data Access

%INCLUDE statement-specific: Is not supported under z/OS

FILE statement: Writes to an external file

FILENAME statement: Reads from or writes to an external file

FILENAME statement, EMAIL (SMTP) Access Method: Sends electronic mail programmatically from SAS using the SMTP (Simple Mail Transfer Protocol)

INFILE statement: Reads from an external file

ODS statements: Controls features of the Output Delivery System that are used to generate, store, or reproduce SAS procedure and DATA step output

FILE command: Saves the contents of a window to an external file

INCLUDE command: Copies an external file into the current window

Syntax

ENCODING= ’*encoding-value*’

Options

ENCODING= ’*encoding-value*’

specifies the encoding to use for reading, writing, copying, or saving an external file. The value for ENCODING= indicates that the external file has a different encoding from the current session encoding.

When you read, write, copy, or save data using an external file, SAS transcodes the data from the session encoding to the specified encoding.

For details, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 407.

Default: SAS uses the current session encoding.

Examples

Example 1: Using the FILE Statement to Specify an Encoding for Writing to an External File

This example creates an external file from a SAS data set. The current session encoding is Wlatin1, but the external file’s encoding needs to be UTF-8. By default, SAS writes the external file using the current session encoding.

To specify what encoding to use for writing data to the external file, specify the ENCODING= option:

```
libname myfiles 'SAS data-library';

filename outfile 'external-file';
```

```
data _null_;
   set myfiles.cars;
   file outfile encoding="utf-8";
   put Make Model Year;
run;
```

When you tell SAS that the external file is to be in UTF-8 encoding, SAS then transcodes the data from Wlatin1 to the specified UTF-8 encoding.

Example 2: Using the FILENAME Statement to Specify an Encoding for Reading an External File This example creates a SAS data set from an external file. The external file is in UTF-8 character-set encoding, and the current SAS session is in the Wlatin1 encoding. By default, SAS assumes that an external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.

To specify which encoding to use when reading the external file, specify the ENCODING= option:

```
libname myfiles  'SAS data-library';

filename extfile 'external-file' encoding="utf-8";

data myfiles.unicode;
   infile extfile;
   input Make $ Model $ Year;
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in Wlatin1.

Example 3: Using the FILENAME Statement to Specify an Encoding for Writing to an External File This example creates an external file from a SAS data set. By default, SAS writes the external file using the current session encoding. The current session encoding is Wlatin1, but the external file's encoding needs to be UTF-8.

To specify which encoding to use when writing data to the external file, specify the ENCODING= option:

```
libname myfiles  'SAS data-library';

filename outfile 'external-file' encoding="utf-8";

data _null_;
   set myfiles.cars;
   file outfile;
   put Make Model Year;
run;
```

When you specify that the external file is to be in UTF-8 encoding, SAS then transcodes the data from Wlatin1 to the specified UTF-8 encoding when writing to the external file.

Example 4: Changing Encoding for Message Body and Attachment This example illustrates how to change text encoding for the message body as well as for the attachment.

```
filename mymail email 'Joe.Developer@sas.com';

data _null_;
   file mymail
      subject='Text Encoding'
      encoding=greek ❶
      attach=('C:\My Files\Test.out' ❷
         content_type='text/plain'
         encoding='ebcdic1047'
         outencoding='latin1'); ❸
run;
```

In the program, the following occurs:

1 The ENCODING= e-mail option specifies that the message body will be encoded to Greek (ISO) before being sent.

2 For the ATTACH= e-mail option, the attachment option ENCODING= specifies the encoding of the attachment that is read into SAS, which is Western (EBCDIC).

3 Because SMTP and other e-mail interfaces do not support EBCDIC, the attachment option OUTENCODING= converts the attachment to Western (ISO) before sending it.

Example 5: Using the INFILE= Statement to Specify an Encoding for Reading from an External File This example creates a SAS data set from an external file. The external file's encoding is in UTF-8, and the current SAS session encoding is Wlatin1. By default, SAS assumes that the external file is in the same encoding as the session encoding, which causes the character data to be written to the new SAS data set incorrectly.

To specify which encoding to use when reading the external file, specify the ENCODING= option:

```
libname myfiles 'SAS data-library';

filename extfile 'external-file';

data myfiles.unicode;
   infile extfile encoding="utf-8";
   input Make $ Model $ Year;
run;
```

When you specify that the external file is in UTF-8, SAS then transcodes the external file from UTF-8 to the current session encoding when writing to the new SAS data set. Therefore, the data is written to the new data set correctly in Wlatin1.

See Also

Statements:

%INCLUDE in *SAS Companion for OpenVMS Alpha*

%INCLUDE in *SAS Companion for UNIX Environments*

%INCLUDE in *SAS Companion for Windows*

FILE in *SAS Language Reference: Dictionary*

FILENAME in *SAS Language Reference: Dictionary*

INFILE in *SAS Language Reference: Dictionary*

ODS statements that use encoding options in *SAS Output Delivery System: User's Guide*

Commands:
FILE in *SAS Companion for OpenVMS Alpha*
FILE in *SAS Companion for z/OS*
FILE in *SAS Companion for UNIX Environments*
FILE in *SAS Companion for Windows*
INCLUDE in *SAS Companion for OpenVMS Alpha*
INCLUDE in *SAS Companion for z/OS*
INCLUDE in *SAS Companion for UNIX Environments*
INCLUDE in *SAS Companion for Windows*

INENCODING= and OUTENCODING= Options

Overrides and changes the encoding when reading or writing SAS data sets in the SAS data library

Valid in: LIBNAME statement
Category: Data Access

Syntax

INENCODING=

INENCODING= ANY | ASCIIANY | EBCDICANY | *encoding-value*
OUTENCODING=

OUTENCODING= ANY | ASCIIANY | EBCDICANY | *encoding-value*

Syntax Description

ANY
specifies no transcoding between ASCII and EBCDIC encodings.

Note: ANY is a synonym for binary. Because the data is binary, the actual encoding is irrelevant. △

ASCIIANY
specifies that no transcoding occurs, assuming that the mixed encodings are ASCII encodings.

EBCDICANY
specifies that no transcoding occurs, assuming that the mixed encodings are EBCDIC encodings.

encoding-value
specifies an encoding value. For a list of encoding values, see "Locale Values and Encoding Values for SBCS, DBCS, and Unicode" on page 400.

Details

The INENCODING= option is used to read SAS data sets in the SAS data library. The OUTENCODING= option is used to write SAS data sets in the SAS data library.

The INENCODING= or the OUTENCODING= value is written to the SAS log when you use the LIST argument.

INENCODING= and OUTENCODING= are most appropriate when using an existing library that contains mixed encodings. To read a library that contains mixed encodings, you can set INENCODING= to ASCIIANY or EBCDICANY. To write a separate data set, you can use OUTENCODING= to specify a specific encoding, which is applied to the data set when it is created.

Comparisons

- Session encoding is specified using the ENCODING= system option or the LOCALE= system option. Each operating environment has a default encoding.
- You can specify the encoding for reading data sets in a SAS data library by using the LIBNAME statement INENCODING= option for input files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.
- You can specify the encoding for writing data sets to a SAS data library by using the LIBNAME statement OUTENCODING= option for output files. If both the LIBNAME statement option and the ENCODING= data set option are specified, SAS uses the data set option.

See Also

"Overview of Encoding for NLS" on page 9

Statements:

LIBNAME in *SAS Language Reference: Dictionary*

System Options:

"ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS" on page 354

"LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS" on page 358

Data Set Options:

"ENCODING= Data Set Option" on page 39

ODSCHARSET= Option

Specifies the character set to be generated in the META declaration for the output

Valid in: LIBNAME statement for the XML engine

Category: Data Access

LIBNAME statement for the XML engine: Specifies the character set to use for generating an output XML document

Syntax

ODSCHARSET=*character-set* ;

Arguments

character-set

For the LIBNAME statement for the XML engine, specifies the character set to use in the ENCODING= attribute.

An example of an encoding is ISO-8859-1. Official character sets for use on the Internet are registered by IANA (Internet Assigned Numbers Authority). IANA is the central registry for various Internet protocol parameters, such as port, protocol and enterprise numbers, options, codes and types. For a complete list of character-set values, visit **`www.unicode.org/reports/tr22/index.html`** and **`www.iana.org/assignments/character-sets`**.

Note: A *character set* is like an *encoding-value* in this context. However, *character set* is the term that is used to identify an encoding that is suitable for use on the Internet. △

Details

An XML declaration is not required in all XML documents. Such a declaration is required only when the character encoding of the document is other than the default UTF-8 or UTF-16 and no encoding was determined by a higher-level protocol.

See Also

Conceptual Information:Chapter 3, "Encoding for NLS," on page 9

Statements:

LIBNAME XML in *SAS XML LIBNAME Engine User's Guide*

ODSTRANTAB = Option

Specifies the translation table to use when transcoding an XML document for an output file

Valid in: the LIBNAME statement for the XML engine

Category: Data Access

Syntax

TRANTAB ='*translation-table*'

Options

translation-table

specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The *table-name* can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

For SAS 9.1, using the ODSTRANTAB= option in the LIBNAME statement for the XML Engine is supported for backward compatibility. The preferred method for specifying an encoding is to use the LOCALE= system option.

See Also

Conceptual Information:

"Transcoding and Translation Tables" on page 22

Conceptual discussion of Chapter 2, "Locale for NLS," on page 5

System Options:

"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

"LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS" on page 358

Procedures:

Chapter 15, "The TRANTAB Procedure," on page 319

Statements:

LIBNAME XML in *SAS XML LIBNAME Engine User's Guide*

TRANSCODE= Column Modifier on PROC SQL

Specifies whether values can be transcoded for character columns

Valid in: Column modifier component in the SQL Procedure

Syntax

TRANSCODE=YES|NO

Arguments

TRANSCODE=YES|NO

for character columns, specifies whether values can be transcoded. Use TRANSCODE=NO to suppress transcoding. Note that when you create a table using the CREATE TABLE AS statement, the transcoding attribute for a particular character column in the created table is the same as it is in the source table unless you change it with the TRANSCODE= column modifier.

Default: YES

Restriction: Suppression of transcoding is not supported for the V6TAPE engine.

See Also

Conceptual Information:

Chapter 4, "Transcoding for NLS," on page 21

The SQL Procedure in *Base SAS Procedures Guide*

RENCODING= Option

Specifies the ASCII-based or EBCDIC-based encoding to use for transcoding data for a SAS/SHARE server session that is using an EBCDICANY or ASCIIANY session encoding

Valid in: LIBNAME statement for SAS/SHARE only

Category: Data Access

Important: The RENCODING= option in the LIBNAME statement is relevant only if using a SAS/SHARE server that has a session encoding set to EBCDICANY or ASCIIANY to preserve a mixed-encoding computing environment, which was more common prior to SAS 9.

See Also: LIBNAME statement in *SAS/SHARE User's Guide*

Syntax

RENCODING=*ASCII-encoding-value* | *EBCDIC-encoding-value*

Syntax Description

ASCII-encoding-value
For a list of valid values for ASCII encodings for OpenVMS, UNIX, and Windows, see Chapter 24, "Encoding Values for a SAS Session," on page 413.

EBCDIC-encoding-value
For a list of valid values for EBCDIC encodings for z/OS, see Chapter 24, "Encoding Values for a SAS Session," on page 413.

Details

If you use SAS/SHARE in a mixed-encoding environment (for example, SAS/SHARE client sessions using incompatible encodings such as Latin1 and Latin2), you can set the following options:

- in the SAS/SHARE server session, set the SAS system option ENCODING=EBCDICANY or ENCODING=ASCIIANY
- in the SAS/SHARE client session, set the RENCODING= option in the LIBNAME statement(s) under these conditions:
  - a client session that uses an ASCII-based encoding accesses an EBCDICANY server
  - a client session that uses an EBCDIC- based encoding accesses an ASCIIANY server.

  The RENCODING= option enables SAS/SHARE clients to specify which encoding to assume the server's data is in when transcoding to or from the client session encoding.

For SAS 9 and 9.1, if you are processing data in a SAS/SHARE client/server session from more than one SBCS or DBCS encoding, you are advised to use the UTF8 encoding. For more information about Unicode servers that run the UTF8 session encoding, visit **http://support.sas.com** and search for “Unicode Server”. Read the article *SUGI 28: Multi-Lingual Computing with the 9.1 SAS® Unicode Server*.

Background

In SAS 9 and 9.1, you can maintain multilingual data that contains characters from more than one traditional SBCS or DBCS encoding in a SAS data set by using a UTF8 encoding. To share update access to that data using SAS/SHARE, you must also run the SAS/SHARE server using a session encoding of UTF8. SAS will transcode the data to the client encoding if necessary.

Prior to SAS 9, if a SAS/SHARE client and a SAS/SHARE server ran on common architectures (for example, the client and server ran on UNIX machines), there was no automatic transcoding of character data. It was possible to build applications that accessed data sets in different EBCDIC or ASCII encodings within a single SAS/SHARE server, or that accessed data sets in mixed different encodings within a single data set. This was very uncommon and required careful programming to set up transcoding tables from clients that ran in different operating environments.

The following steps describe how you can maintain mixed encoding in SAS 9, if necessary.

- The SAS/SHARE server must run by using a session encoding of EBCDICANY for mixed-EBCDIC encodings or ASCIIANY for mixed-ASCII encodings.

  This will restore the behavior of Version 8 and earlier releases and prevent the automatic character transcoding between different client and server encodings in the same EBCDIC or ASCII family. That is, no transcoding will occur under these circumstances:

  - if the client session encoding is an EBCDIC encoding and the server session encoding is EBCDICANY
  - if the client session encoding is an ASCII encoding and the server session encoding is ASCIIANY.

- A SAS/SHARE client that does not share the same encoding family as an ASCIIANY or EBCDICANY server can control the necessary transcoding by using an RENCODING= option on the first LIBNAME statement that accesses the server.

  For example, an ASCII client that runs in a Polish locale could access a z/OS EBCDICANY server and specify RENCODING=EBCDIC870 to access data that the client knows contains Polish-encoded data. Another ASCII client that runs in a German locale could access the same z/OS EBCDICANY server and specify RENCODING=EBCDIC1141 to access data that the client knows contains German data. Similarly, EBCDIC clients that access an ASCIIANY server can specify the precise ASCII encoding of the data they are accessing by using the RENCODING= option in the LIBNAME statement.

See Also

Conceptual information:

Chapter 4, “Transcoding for NLS,” on page 21

Statements:

LIBNAME in *SAS/SHARE User’s Guide*

TRANSCODE= Option

Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and/or length with one or more variables) that indicates whether character variables are to be transcoded

Valid in: the ATTRIB statement in a DATA step

Category: Information

Type: Declarative

See: ATTRIB Statement in the documentation for your operating environment.

Syntax

ATTRIB *variable-list(s) attribute-list(s)* ;

Arguments

variable-list
names the variables that you want to associate with the attributes.

Tip: List the variables in any form that SAS allows.

attribute-list
specifies one or more attributes to assign to *variable-list*. Multiple attributes can be specified in the ATTRIB statement. For a complete list of attributes, see the ATTRIB Statement in *SAS Language Reference: Dictionary*.

TRANSCODE= YES | NO
Specifies whether to transcode character variables. Use TRANSCODE=NO to suppress transcoding. For more information, see "Overview to Transcoding" on page 21.

Default: YES

Restriction: Prior releases of SAS cannot access a SAS 9.1 data set that contains a variable with a TRANSCODE=NO attribute.

Interaction: You can use the VTRANSCODE and VTRANSCODEX functions to return whether transcoding is on or off for a character variable.

Interaction: If the TRANSCODE= attribute is set to NO for any character variable in a data set, PROC CONTENTS will print a transcode column that contains the TRANSCODE= value for each variable in the data set. If all variables in the data set are set to the default TRANSCODE= value (YES), no transcode column will be printed.

Examples

Example 1: Using the TRANSCODE= Option With the SET Statement When you use the SET statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is NO because B is the first data set and Z's TRANSCODE= attribute in data set B is NO.

```
data b;
   length z $4;
   z = 'ice';
   attrib z transcode = NO;
data c;
   length z $4;
   z = 'snow';
   attrib z transcode = YES;
data a;
   set b;
   set c;
   /* Check transcode setting for variable Z */
   rc1 = vtranscode(z);
   put rc1=;
run;
```

Example 2: Using the TRANSCODE= Option With the MERGE Statement When you use the MERGE statement to create a data set from several data sets, SAS makes the TRANSCODE= attribute of the variable in the output data set equal to the TRANSCODE= value of the variable in the first data set. In this example, the variable Z's TRANSCODE= attribute in data set A is YES because C is the first data set and Z's TRANSCODE= attribute in data set C is YES.

```
data b;
   length z $4;
   z = 'ice';
   attrib z transcode = NO;
data c;
   length z $4;
   z = 'snow';
   attrib z transcode = YES;
data a;
   merge c b;
   /* Check transcode setting for variable Z */
   rc1 = vtranscode(z);
   put rc1=;
run;
```

Note: The TRANSCODE= attribute is set when the variable is first seen on an input data set or in an ATTRIB TRANSCODE= statement. If a SET or MERGE statement comes before an ATTRIB TRANSCODE= statement and the TRANSCODE= attribute contradicts the SET statement, an error message will occur. △

See Also

Functions:

"VTRANSCODE Function" on page 237
"VTRANSCODEX Function" on page 238

TRANTAB= Option

Specifies the translation table to use when you are transcoding character data in a SAS file for the appropriate output file

Valid in: ODS MARKUP statement and ODS RTF statement

Category: ODS: Third-Party Formatted

Syntax

TRANTAB = (*translation-table*)

Note: Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.1 supports the the TRANTAB= option for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. △

Options

translation-table
specifies the translation table to use for the output file. The translation table is an encoding method that maps characters (letters, logograms, digits, punctuation, symbols, control characters, and so on) in the character set to numeric values. An example of a translation table is one that converts characters from EBCDIC to ASCII-ISO. The *table-name* can be any translation table that SAS provides, or any user-defined translation table. The value must be the name of a SAS catalog entry in either the SASUSER.PROFILE catalog or the SASHELP.HOST catalog.

Details

Note: For SAS 9.1, using the TRANTAB = option in the ODS MARKUP is supported for backward compatibility. For specifying encoding, the LOCALE= system option is preferred. △

See Also

Conceptual Information:

"Transcoding and Translation Tables" on page 22

Chapter 2, "Locale for NLS," on page 5

System Options:

"TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

"LOCALE System Option: OpenVMS, UNIX, Windows, and z/OS" on page 358

Procedures:

Chapter 15, "The TRANTAB Procedure," on page 319

Statements:

ODS MARKUP in *SAS Output Delivery System: User's Guide*

ODS RTF in *SAS Output Delivery System: User's Guide*

XMLENCODING= Option

Overrides the encoding of an XML document to import or export an external document

Valid in: LIBNAME statement for the XML engine

Category: Data Access

LIBNAME statement for the XML engine: Associates a SAS libref with an XML document to import or export an external document

Syntax

XMLENCODING= '*encoding-value*'

Options

encoding-value
specifies the encoding to use when you read, write, copy, or save an external file. The value for XMLENCODING= indicates that the external file has a different encoding from the current session encoding.

For details, see "SBCS, DBCS, and Unicode Encoding Values for Transcoding Data" on page 407.

Default: SAS uses the current session encoding.

See Also

Statements:

LIBNAME XML in *SAS XML LIBNAME Engine User's Guide*

CHAPTER

20

The TRANTAB Statement Used with Procedures

TRANTAB Statement

Specifies the translation table to use when you transcode character data in order to export or transfer a SAS file

Valid in: CPORT Procedure, UPLOAD procedure, DOWNLOAD procedure

PROC CPORT: Used when you export a SAS file across a network

PROC UPLOAD and PROC DOWNLOAD: Used when you transfer a SAS file across a network

Requirements for UPLOAD and DOWNLOAD: To use the TRANTAB statement, you must specify the INCAT= and OUTCAT= options in the PROC UPLOAD or PROC DOWNLOAD statement.

Restrictions: You can specify only one translation table per TRANTAB statement. To specify additional translation tables, use additional TRANTAB statements.

Interaction: The TRANTAB statement specifies a customized translation table (for example, to map an EBCDIC character to an ASCII character) to apply to the character set in the SAS file that is being exported or transferred. The TRANTAB= system option specifies a translation table to use for the SAS session, including file transfers.

Syntax

TRANTAB NAME=*translation-table-name* <TYPE=(*etype-list*) <OPT=DISP | SRC | (DISP SRC)>>;

Note: Translation tables were introduced in SAS 6 to support the requirements of national languages. SAS 8.2 introduced the LOCALE= system option as an improvement on direct use of translation tables. SAS 9.1 supports the TRANTAB statement for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases. △

Arguments

NAME=*translation-table-name*
specifies the name of the translation table to apply to the SAS catalog that you want to export (PROC CPORT) or transfer (PROC UPLOAD or PROC DOWNLOAD). The

translation-table-name that you specify as the name of a catalog entry in either your SASUSER.PROFILE catalog or the SASHELP.HOST catalog. The SASUSER.PROFILE catalog is searched first, and then the SASHELP.HOST catalog is searched.

In most cases, the default translation table is the correct one to use, but you might need to apply additional translation tables if, for example, your application requires different national language characters.

You can specify a translation table other than the default in two ways:

- To specify a translation table for an invocation of the procedure, use the TRANTAB statement in the procedure, as appropriate.
- To specify a translation table for your entire SAS session or job (including all file exports or transfers), use the TRANTAB= system option.

Options

TYPE=(*etype-list*)

applies the translation table only to the entries with the type or types that you specify. The *etype-list* can be one or more entry types. Examples of catalog entry types include DATA and FORMAT. If *etype-list* is a simple entry type, omit the parentheses.

By default, the UPLOAD, DOWNLOAD, and CPORT procedures apply the translation table to all specified catalog entries.

OPT=DISP | SRC | (DISP SRC)

OPT=DISP	applies the translation table only to the specified catalog entries, which produce window displays.
OPT=SRC	applies the translation table only to the specified catalog entries that are of the type SOURCE.
OPT=(DISP SRC)	applies the translation table only to the specified catalog entries that either produce window displays or are of type SOURCE.

If you do not specify the OPT= option, the UPLOAD or DOWNLOAD procedure applies the translation table to all of the entries in the catalog that you specify.

Default: PROC CPORT, PROC UPLOAD, and PROC DOWNLOAD apply the translation table to all entries and data sets in the specified catalog.

Examples

Procedure features:

PROC CPORT statement option: FILE=

TRANTAB statement option: TYPE=

This example shows how to apply a customized translation table to the transport file before PROC CPORT exports it. For this example, assume that you have already created a customized translation table called TTABLE1.

Example 1: Program

Assign library references. The LIBNAME and FILENAME statements assign a libref for the source library and a fileref for the transport file, respectively.

```
libname source 'SAS data-library';
filename tranfile 'transport-file'
                   host-option(s)-for-file-characteristics;
```

Apply the translation specifics. The TRANTAB statement applies the translation that you specify with the customized translation table TTABLE1. TYPE= limits the translation to FORMAT entries.

```
proc cport catalog=source.formats file=tranfile;
   trantab name=ttable1 type=(format);
run;
```

Example 2: SAS Log

```
NOTE: Proc CPORT begins to transport catalog SOURCE.FORMATS
NOTE: The catalog has 2 entries and its maximum logical record length is 104.
NOTE: Entry REVENUE.FORMAT has been transported.
NOTE: Entry DEPT.FORMATC has been transported.
```

See Also

Conceptual Information:

  Chapter 4, "Transcoding for NLS," on page 21

System Options:

  "TRANTAB= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 362

Procedures:

  Chapter 15, "The TRANTAB Procedure," on page 319

  CPORT in *Base SAS Procedures Guide*

  UPLOAD in *SAS/CONNECT User's Guide*

  DOWNLOAD in *SAS/CONNECT User's Guide*

PART 9

Values for Locale, Encoding, and Transcoding

CHAPTER 21

Values for the LOCALE= System Option

LOCALE= and Default Values for DFLANG, DATESTYLE, and PAPERSIZE Options

The valid LOCALE= values are specified by using either the SAS name or the POSIX name. The settings for the DFLANG=, DATESTYLE=, and PAPERSIZE= system options are set automatically.

Example:

```
sas9 -locale arabic_algeria
```

When the arabic_algeria locale value is specified, corresponding default settings for the system options are as follows:

```
DFLANG=English
DATESTYLE=DMY
PAPERSIZE=A4
```

Table 21.1 LOCALE= Values and Default Values Assigned to DFLANG=, DATESTYLE=, and PAPERSIZE=

LOCALE=		DFLANG=	DATESTYLE=	PAPERSIZE=
SAS Name	**POSIX Name**	**Default Values**		
Arabic_Algeria	ar_DZ	English	DMY	A4
Arabic_Bahrain	ar_BH	English	DMY	A4
Arabic_Egypt	ar_EG	English	DMY	A4
Arabic_Jordan	ar_JO	English	DMY	A4
Arabic_Kuwait	ar_KW	English	DMY	A4
Arabic_Lebanon	ar_LB	English	DMY	A4
Arabic_Morocco	ar_MA	English	DMY	A4
Arabic_Oman	ar_OM	English	DMY	A4
Arabic_Qatar	ar_QA	English	DMY	A4
Arabic_SaudiArabia	ar_SA	English	DMY	A4

LOCALE=		DFLANG=	DATESTYLE=	PAPERSIZE=
SAS Name	**POSIX Name**	**Default Values**		
Arabic_Tunisia	ar_TN	English	DMY	A4
Arabic_UnitedArabEmirates	ar_AE	English	DMY	A4
Bulgarian_Bulgaria	bg_BG	English	YMD	A4
Byelorussian_Belarus	be_BY	English	DMY	A4
Chinese_China	zh_CN	Locale	YMD	A4
Chinese_HongKong	zh_HK	Locale	YMD	A4
Chinese_Macau	zh_MO	Locale	YMD	A4
Chinese_Singapore	zh_SG	Locale	DMY	A4
Chinese_Taiwan	zh_TW	Locale	YMD	A4
Croatian_Croatia	hr_HR	Croatian	YMD	A4
Czech_CzechRepublic	cs_CZ	Czech	DMY	A4
Danish_Denmark	da_DK	Danish	DMY	A4
Dutch_Belgium	nl_BE	Dutch	DMY	A4
Dutch_Netherlands	nl_NL	Dutch	DMY	A4
English_Australia	en_AU	English	DMY	A4
English_Canada	en_CA	English	DMY	letter
English_HongKong	en_HK	English	DMY	A4
English_India	en_IN	English	DMY	A4
English_Ireland	en_IE	English	DMY	A4
English_Jamaica	en_JM	English	DMY	letter
English_NewZealand	en_NZ	English	DMY	A4
English_Singapore	en_SG	English	DMY	A4
English_SouthAfrica	en_ZA	English	DMY	A4
English_UnitedKingdom	en_GB	English	DMY	A4
English_UnitedStates	en_US	English	MDY	letter
Estonian_Estonia	et_EE	English	DMY	A4
Finnish_Finland	fi_FI	Finnish	DMY	A4
French_Belgium	fr_BE	French	DMY	A4
French_Canada	fr_CA	French	DMY	letter
French_France	fr_FR	French	DMY	A4
French_Luxembourg	fr_LU	French	DMY	A4
French_Switzerland	fr_CH	Swiss_French	DMY	A4
German_Austria	de_AT	German	DMY	A4
German_Germany	de_DE	German	DMY	A4
German_Liechtenstein	de_LI	German	DMY	A4
German_Luxembourg	de_LU	German	DMY	A4

LOCALE=		DFLANG=	DATESTYLE=	PAPERSIZE=
SAS Name	**POSIX Name**	**Default Values**		
German_Switzerland	de_CH	Swiss_German	DMY	A4
Greek_Greece	el_GR	English	DMY	A4
Hebrew_Israel	he_IL	English	DMY	A4
Hungarian_Hungary	hu_HU	Hungarian	YMD	A4
Icelandic_Iceland	is_IS	English	DMY	A4
Italian_Italy	it_IT	Italian	DMY	A4
Italian_Switzerland	it_CH	Italian	DMY	A4
Japanese_Japan	ja_JP	Japanese	YMD	A4
Korean_Korea	ko_KR	Locale	YMD	A4
Latvian_Latvia	lv_LV	English	YMD	A4
Lithuanian_Lithuania	lt_LT	English	YMD	A4
Norwegian_Norway	no_NO	Norwegian	DMY	A4
Polish_Poland	pl_PL	Polish	YMD	A4
Portuguese_Brazil	pt_BR	Portuguese	DMY	letter
Portuguese_Portugal	pt_PT	Portuguese	DMY	A4
Romanian_Romania	ro_RO	English	DMY	A4
Russian_Russia	ru_RU	Russian	DMY	A4
Serbian_Yugoslavia	sr_YU	English	DMY	A4
Slovak_Slovakia	sk_SK	English	DMY	A4
Slovenian_Slovenia	sl_SL	Slovenian	YMD	A4
Spanish_Argentina	es_AR	Spanish	DMY	letter
Spanish_Bolivia	es_BO	Spanish	DMY	letter
Spanish_Chile	es_CL	Spanish	DMY	letter
Spanish_Columbia	es_CO	Spanish	DMY	letter
Spanish_CostaRica	es_CR	Spanish	DMY	letter
Spanish_DominicanRepublic	es_DO	Spanish	DMY	letter
Spanish_Ecuador	es_EC	Spanish	DMY	letter
Spanish_ElSalvador	es_ES	Spanish	MDY	letter
Spanish_Guatemala	es_GT	Spanish	DMY	letter
Spanish_Honduras	es_HN	Spanish	MDY	letter
Spanish_Mexico	es_MX	Spanish	DMY	letter
Spanish_Nicaragua	es_NI	Spanish	MDY	letter
Spanish_Panama	es_PA	Spanish	MDY	letter
Spanish_Paraguay	es_PY	Spanish	DMY	letter
Spanish_Peru	es_PE	Spanish	DMY	letter
Spanish_PuertoRico	es_PR	Spanish	MDY	letter

LOCALE=		DFLANG=	DATESTYLE=	PAPERSIZE=
SAS Name	POSIX Name	Default Values		
Spanish_Spain	es_ES	Spanish	DMY	A4
Spanish_UnitedStates	es_US	Spanish	DMY	letter
Spanish_Uruguay	es_UY	Spanish	DMY	A4
Spanish_Venezuela	es_VE	Spanish	DMY	letter
Swedish_Sweden	sv_SE	Swedish	YMD	A4
Thai_Thailand	th_TH	English	DMY	A4
Turkish_Turkey	tr_TR	English	DMY	A4
Ukrainian_Ukraine	uk_UA	English	DMY	A4
Vietnamese_Vietnam	vi_VN	English	DMY	A4

Locale Values and Encoding Values for SBCS, DBCS, and Unicode

The following table lists the valid LOCALE= values (specified by using either the SAS name or the POSIX name) and the default settings for the ENCODING= option, by operating environment:

Table 21.2 LOCALE= and Default ENCODING= Values

LOCALE=		ENCODING=			
SAS Name	POSIX Name	OpenVMS and UNIX	Windows	z/OS	
				NLSCOMPATMODE	NONLSCOMPATMODE
Arabic_Algeria	ar_DZ	arabic	warabic	ebcdic425	open_ed-425
Arabic_Bahrain	ar_BH	arabic	warabic	ebcdic425	open_ed-425
Arabic_Egypt	ar_EG	arabic	warabic	ebcdic425	open_ed-425
Arabic_Jordan	ar_JO	arabic	warabic	ebcdic425	open_ed-425
Arabic_Kuwait	ar_KW	arabic	warabic	ebcdic425	open_ed-425
Arabic_Lebanon	ar_LB	arabic	warabic	ebcdic425	open_ed-425
Arabic_Morocco	ar_MA	arabic	warabic	ebcdic425	open_ed-425
Arabic_Oman	ar_OM	arabic	warabic	ebcdic425	open_ed-425
Arabic_Qatar	ar_QA	arabic	warabic	ebcdic425	open_ed-425
Arabic_SaudiArabia	ar_SA	arabic	warabic	ebcdic425	open_ed-425
Arabic_Tunisia	ar_TN	arabic	warabic	ebcdic425	open_ed-425
Arabic_UnitedArabEmirates	ar_AE	arabic	warabic	ebcdic425	open_ed-425
Bulgarian_Bulgaria	bg_BG	cyrillic	wcyrillic	ebcdic1025	open_ed-1025
Byelorussian_Belarus	be_BY	cyrillic	wcyrillic	ebcdic1025	open_ed-1025
Chinese_China	zh_CN	n/a	n/a	n/a	n/a
Chinese_HongKong	zh_HK	n/a	n/a	n/a	n/a

LOCALE=		ENCODING=			
SAS Name	POSIX Name	OpenVMS and UNIX	Windows	z/OS	
				NLSCOMPATMODE	NONLSCOMPATMODE
Chinese_Macau	zh_MO	n/a	n/a	n/a	n/a
Chinese_Singapore	zh_SG	n/a	n/a	n/a	n/a
Chinese_Taiwan	zh_TW	n/a	n/a	n/a	n/a
Croatian_Croatia	hr_HR	latin2	wlatin2	ebcdic870	open_ed-870
Czech_CzechRepublic	cs_CZ	latin2	wlatin2	ebcdic870	open_ed-870
Danish_Denmark	da_DK	latin9	wlatin1	ebcdic1142	open_ed-1142
Dutch_Belgium	nl_BE	latin1	wlatin1	ebcdic1148	open_ed-1148
Dutch_Netherlands	nl_NL	latin1	wlatin1	ebcdic1140	open_ed-1140
English_Australia	en_AU	latin1	wlatin1	ebcdic1047	open_ed-1047
English_Canada	en_CA	latin1	wlatin1	ebcdic1047	open_ed-1047
English_HongKong	en_HK	latin9	wlatin1	ebcdic1146	open_ed-1146
English_India	en_IN	latin9	wlatin1	ebcdic1146	open_ed-1146
English_Ireland	en_IE	latin9	wlatin1	ebcdic1146	open_ed-1146
English_Jamaica	en_JM	latin1	wlatin1	ebcdic1047	open_ed-1047
English_NewZealand	en_NZ	latin1	wlatin1	ebcdic1047	open_ed-1047
English_Singapore	en_SG	latin9	wlatin1	ebcdic1146	open_ed-1146
English_SouthAfrica	en_ZA	latin1	wlatin1	ebcdic1047	open_ed-1047
English_UnitedKingdom	en_GB	latin9	wlatin1	ebcdic1146	open_ed-1146
English_UnitedStates	en_US	latin1	wlatin1	ebcdic1047	open_ed-1047
Estonian_Estonia	et_EE	latin6	wbaltic	ebcdic1122	open_ed-1122
Finnish_Finland	fi_FI	latin9	wlatin1	ebcdic1143	open_ed-1143
French_Belgium	fr_BE	latin9	wlatin1	ebcdic1148	open_ed-1148
French_Canada	fr_CA	latin1	wlatin1	ebcdic1047	open_ed-1047
French_France	fr_FR	latin9	wlatin1	ebcdic1147	open_ed-1147
French_Luxembourg	fr_LU	latin9	wlatin1	ebcdic1147	open_ed-1147
French_Switzerland	fr_CH	latin9	wlatin1	ebcdic1148	open_ed-1148
German_Austria	de_AT	latin9	wlatin1	ebcdic1141	open_ed-1141
German_Germany	de_DE	latin9	wlatin1	ebcdic1141	open_ed-1141
German_Liechtenstein	de_LI	latin9	wlatin1	ebcdic1141	open_ed-1141
German_Luxembourg	de_LU	latin9	wlatin1	ebcdic1141	open_ed-1141
German_Switzerland	de_CH	latin9	wlatin1	ebcdic1148	open_ed-1148
Greek_Greece	el_GR	greek	wgreek	ebcdic875	open_ed-875
Hebrew_Israel	he_IL	hebrew	whebrew	ebcdic424	open_ed-424
Hungarian_Hungary	hu_HU	latin2	wlatin2	ebcdic870	open_ed-870

LOCALE=		ENCODING=			
SAS Name	POSIX Name	OpenVMS and UNIX	Windows	z/OS	
				NLSCOMPATMODE	NONLSCOMPATMODE
Icelandic_Iceland	is_IS	latin1	wlatin1	ebcdic1047	open_ed-1047
Italian_Italy	it_IT	latin9	wlatin1	ebcdic1144	open_ed-1144
Italian_Switzerland	it_CH	latin9	wlatin1	ebcdic1148	open_ed-1148
Japanese_Japan	ja_JP	n/a	n/a	n/a	n/a
Korean_Korea	ko_KR	n/a	n/a	n/a	n/a
Latvian_Latvia	lv_LV	latin6	wbaltic	ebcdic1112	open_ed-1112
Lithuanian_Lithuania	lt_LT	latin6	wbaltic	ebcdic1112	open_ed-1112
Norwegian_Norway	no_NO	latin9	wlatin1	ebcdic1142	open_ed-1142
Polish_Poland	pl_PL	latin2	wlatin2	ebcdic870	open_ed-870
Portuguese_Brazil	pt_BR	latin1	wlatin1	ebcdic275	open_ed-275
Portuguese_Portugal	pt_PT	latin1	wlatin1	ebcdic1140	open_ed-1140
Romanian_Romania	ro_RO	latin2	wlatin2	ebcdic870	open_ed-870
Russian_Russia	ru_RU	cyrillic	wcyrillic	ebcdic1025	open_ed-1025
Serbian_Yugoslavia	sr_YU	cyrillic	wcyrillic	ebcdic1025	open_ed-1025
Slovak_Slovakia	sk_SK	latin2	wlatin2	ebcdic870	open_ed-870
Slovenian_Slovenia	sl_SL	latin2	wlatin2	ebcdic870	open_ed-870
Spanish_Argentina	es_AR	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Bolivia	es_BO	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Chile	es_CL	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Colombia	es_CO	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_CostaRica	es_CR	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_DominicanRepublic	es_DO	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Ecuador	es_EC	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_ElSalvador	es_SV	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Guatemala	es_GT	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Honduras	es_HN	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Mexico	es_MX	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Nicaragua	es_NI	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Panama	es_PA	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Paraguay	es_PY	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Peru	es_PE	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_PuertoRico	es_PR	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Spain	es_ES	latin9	wlatin1	ebcdic1145	open_ed-1145
Spanish_UnitedStates	es_US	latin1	wlatin1	ebcdic1047	open_ed-1047

LOCALE=		ENCODING=			
SAS Name	**POSIX Name**	**OpenVMS and UNIX**	**Windows**	**z/OS**	
				NLSCOMPATMODE	**NONLSCOMPATMODE**
Spanish_Uruguay	es_UY	latin1	wlatin1	ebcdic1047	open_ed-1047
Spanish_Venezuela	es_VE	latin1	wlatin1	ebcdic1047	open_ed-1047
Swedish_Sweden	sv_SE	latin9	wlatin1	ebcdic1143	open_ed-1143
Thai_Thailand	th_TH	thai	pcoem874	ebcdic838	open_ed-838
Turkish_Turkey	tr_TR	latin5	wturkish	ebcdic1026	open_ed-1026
Ukrainian_Ukraine	uk_UA	cyrillic	wcyrillic	ebcdic1025	open_ed-1025
Vietnamese_Vietnam	vi_VN	latin1	wvietnamese	ebcdic1130	open_ed-1130

CHAPTER
22

SAS System Options for Processing DBCS Data

Overview to System Options Used in a SAS Session for DBCS

You use the DBCSLANG= and DBCSTYPE= system options to specify the DBCS encoding values for a SAS session. You do not directly use the ENCODING= system option when you are using DBCS.

DBCS Values for a SAS Session

Operating Environment Information: The following table shows the supported values for the DBCSLANG= and DBCSTYPE= system options under the z/OS, UNIX, and Windows operating environments. DBCS is not supported under the OpenVMS operating environment. △

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 22.1 DBCS Supported Values for the DBCSLANG= and DBCSTYPE= System Options

DBCSLANG=	z/OS DBCSTYPE=	UNIX DBCSTYPE=	Windows DBCSTYPE=
Chinese	ibm	dec	pcms
Chinese	n/a	hp15	n/a
Chinese	n/a	euc	n/a
Chinese	n/a	pcms	n/a
Japanese	ibm	dec	pcms
Japanese	pcibm	euc	sjis
Japanese	n/a	hp15	n/a
Japanese	n/a	sjis	n/a
Korean	ibm	pcibm	pcms

DBCSLANG=	z/OS DBCSTYPE=	UNIX DBCSTYPE=	Windows DBCSTYPE=
Korean	n/a	pcms	n/a
Korean	n/a	dec	n/a
Korean	n/a	euc	n/a
Korean	n/a	hp15	n/a
Taiwanese	ibm	dec	pcms
Taiwanese	pcibm	euc	n/a
Taiwanese	n/a	hp15	n/a
Taiwanese	n/a	pcms	n/a

CHAPTER 23

Encoding Values in SAS Language Elements

Overview to SAS Language Elements That Use Encoding Values

When the encoding of the SAS session is different from the encoding of the SAS file or from the data that resides in the SAS file, transcoding must occur. Consider a SAS file that was created in the Western Latin1 encoding, then moved to an IBM mainframe that uses the German EBCDIC encoding. In order for the IBM mainframe to successfully access the file, the SAS data file must be transcoded from the Western Latin1 encoding to the German EBCDIC encoding. For information about transcoding concepts, including SAS language elements that contain options for transcoding, see Chapter 4, "Transcoding for NLS," on page 21.

SBCS, DBCS, and Unicode Encoding Values for Transcoding Data

The following table presents a list of SBCS, DBCS, and Unicode encoding values for transcoding data for all operating environments: The encoding values in the following table are valid for SAS language elements that contain options for transcoding.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 23.1 SBCS, DBCS, and Unicode Encoding Values Used to Transcode Data

Encoding Value	Description
aarabic	Arabic Macintosh
agreek	Greek Macintosh
ahebrew	Hebrew Macintosh
aiceland	Icelandic Macintosh
any	no transcoding is specified
arabic	Arabic ISO
aroman	Roman Macintosh

Encoding Value	Description
asciiany	enables you to create a data set that is compatible with all ASCII encodings
aturkish	Turkish Macintosh
aukrainian	Ukrainian Macintosh
big5	Traditional Chinese Big5
cyrillic	Cyrillic ISO
dec-cn	Simplified Chinese DEC
dec-jp	Japanese DEC
dec-tw	Traditional Chinese DEC
ebcdic037	North America EBCDIC
ebcdic1025	Cyrillic EBCDIC
ebcdic1026	Turkish EBCDIC
ebcdic1047	Western EBCDIC
ebcdic1112	Baltic EBCDIC
ebcdic1122	Estonian EBCDIC
ebcdic275	Brazil EBCDIC
ebcdic424	Hebrew EBCDIC
ebcdic425	Arabic EBCDIC
ebcdic500	International EBCDIC
ebcdic838	Thai EBCDIC
ebcdic870	Central Europe EBCDIC
ebcdic875	Greek EBCDIC
ebcdic924	European EBCDIC
ebcdic1130	Vietnamese EBCDIC
ebcdic1140	North America EBCDIC
ebcdic1141	Austria/Germany EBCDIC
ebcdic1142	Denmark/Norway EBCDIC
ebcdic1143	Finland/Sweden EBCDIC
ebcdic1144	Italy EBCDIC
ebcdic1145	Spain EBCDIC
ebcdic1146	United Kingdom EBCDIC
ebcdic1147	France EBCDIC
ebcdic1148	International EBCDIC
ebcdicany	enables you to create a data set that is compatible with all EBCDIC encodings
euc-cn	Simplified Chinese EUC
euc-jp	Japanese EUC
euc-kr	Korean EUC

Encoding Value	Description
euc-tw	Traditional Chinese EUC
fujitsu-cn	Simplified Chinese FACOM
fujitsu-jp	Japanese FACOM
fujitsu-ko	Korean FACOM
fujitsu-tw	Traditional Chinese FACOM
greek	Greek ISO
hebrew	Hebrew ISO
hitachi-cn	Simplified Chinese HITAC
hitachi-jn	Japanese HITAC
hitachi-ko	Korean HITAC
hitachi-tw	Traditional Chinese HITAC
hitsas-jp	Japanese XHITAC
hitsas-ko	Korean XHITAC
hitsas-tw	Traditional Chinese XHITAC
hp15-tw	Traditional Chinese HP15
ibm-1381	Simplified Chinese PCIBM
ibm-933	Korean IBM
ibm-935	Simplified Chinese IBM
ibm-937	Traditional Chinese IBM
ibm-939	Japanese IBM
ibm-942	Japanese PCIBM
ibm-949	Korean PCIBM
latin1	Western ISO
latin2	Central Europe ISO
latin5	Turkish ISO
latin6	Baltic ISO
latin9	European ISO
macos-1	Japanese PCMAC
macos-2	Traditional Chinese PCMAC
macos-25	Simplified Chinese PCMAC
macos-3	Korean PCMAC
ms-932	Japanese PCMS
ms-936	Simplified Chinese PCMS
ms-949	Korean PCMS
ms-950	Traditional Chinese PCMS
msdos720	Arabic MS-DOS
msdos737	Greek MS-DOS

Encoding Value	Description
msdos775	Baltic MS-DOS
open_ed-275	Brazil OpenEdition
open_ed-425	Arabic OpenEdition
open_ed-838	Thai OpenEdition
open_ed-924	European OpenEdition
open_ed-1025	Cyrillic OpenEdition
open_ed-1026	Turkish OpenEdition
open_ed-1047	Western OpenEdition
open_ed-1112	Baltic OpenEdition
open_ed-1122	Estonian OpenEdition
open_ed-1130	Vietnamese OpenEdition
open_ed-1140	North American OpenEdition
open_ed-1141	Austria/Germany OpenEdition
open_ed-1142	Denmark/Norway OpenEdition
open_ed-1143	Finland/Sweden OpenEdition
open_ed-1144	Italy OpenEdition
open_ed-1145	Spain OpenEdition
open_ed-1146	United Kingdom OpenEdition
open_ed-1147	France OpenEdition
open_ed-1148	International OpenEdition
open_ed-424	Hebrew OpenEdition
open_ed-870	Central Europe OpenEdition
open_ed-875	Greek OpenEdition
pcoem437	USA IBM-PC
pcoem850	Western IBM-PC
pcoem852	Central Europe IBM-PC
pcoem857	Turkish IBM-PC
pcoem858	European IBM-PC
pcoem860	Portuguese MS-DOS
pcoem862	Hebrew IBM-PC
pcoem863	French Canadian IBM-PC
pcoem864	Arabic IBM-PC
pcoem865	Nordic IBM-PC
pcoem866	Cyrillic IBM-PC
pcoem869	Greek IBM-PC
pcoem874	Thai IBM-PC
pcoem921	Baltic IBM-PC

Encoding Value	Description
pcoem922	Estonia IBM-PC
pcoem1129	Vietnamese IBM-PC
shift-jis	Japanese SJIS
thai	Thai ISO
utf-8	Unicode (UTF-8)
utf-16	Unicode (UTF-16)
utf-32	Unicode (UTF-32)
warabic	Arabic Windows
wbaltic	Baltic Windows
wcyrillic	Cyrillic Windows
wgreek	Greek Windows
whebrew	Hebrew Windows
wlatin1	Western Windows
wlatin2	Central Europe Windows
wturkish	Turkish Windows
wvietnamese	Vietnamese Windows

CHAPTER 24 Encoding Values for a SAS Session

OpenVMS Encoding Values

The encodings in the following tables are valid in the OpenVMS operating environment.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 24.1 Single-Byte Encodings for OpenVMS

ENCODING= Value	Description
arabic	Arabic (ISO)
cyrillic	Cyrillic (ISO)
greek	Greek (ISO)
hebrew	Hebrew (ISO)
latin1	Western (ISO)
latin2	Central Europe (ISO)
latin5	Turkish (ISO)
latin6	Baltic (ISO)
latin9	European (ISO)
thai	Thai (ISO)

Table 24.2 Double-Byte Encodings for OpenVMS

ENCODING= Value	Description
big5	Traditional Chinese (Big5)
dec-cn	Simplified Chinese (DEC)

ENCODING= Value	Description
dec-jp	Japanese (DEC)
dec-tw	Traditional Chinese (DEC)
euc-cn	Simplified Chinese (EUC)
euc-jp	Japanese (EUC)
euc-kr	Korean (EUC)
euc-tw	Traditional Chinese (EUC)
ms-936	Simplified Chinese (PCMS)
ms-949	Korean (PCMS)
shift-jis	Japanese (SJIS)

UNIX Encoding Values

The encodings in the following tables are valid in UNIX environments.

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 24.3 Single-Byte Encodings for UNIX

ENCODING= Value	Description
arabic	Arabic (ISO)
cyrillic	Cyrillic (ISO)
greek	Greek (ISO)
hebrew	Hebrew (ISO)
latin1	Western (ISO)
latin2	Central Europe (ISO)
latin5	Turkish (ISO)
latin6	Baltic (ISO)
latin9	European (ISO)
thai	Thai (ISO)

Table 24.4 Double-Byte Encodings for UNIX

ENCODING= Value	Description
big5	Traditional Chinese (Big5)
euc-cn	Simplified Chinese (EUC)
euc-jp	Japanese (EUC)
euc-kr	Korean (EUC)
euc-tw	Traditional Chinese (EUC)

ENCODING= Value	Description
hp15-tw	Traditional Chinese (HP15)
ms-936	Simplified Chinese (PCMS)
ms-949	Korean (PCMS)
shift-jis	Japanese (SJIS)

UNIX also supports the utf-8 Unicode encoding.

Windows Encoding Values

The encodings in the following tables are valid in the Windows operating environment.

Note: If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 24.5 Single-Byte Encodings for Windows

Description	Windows ENCODING= Value	MS-DOS ENCODING= Value	IBM-PC ENCODING= Value
Arabic	warabic	msdos720	pcoem864
Baltic	wbaltic	msdos775	pcoem921
Central Europe	wlatin2	n/a	pcoem852
Cyrillic	wcyrillic	n/a	pcoem866 pcoem855
Central Europe	n/a	n/a	pcoem852
Estonia	n/a	n/a	pcoem922
European	n/a	n/a	pcoem858
French Canadian	n/a	n/a	pcoem863
Greek	wgreek	msdos737	n/a
Hebrew	whebrew	n/a	pcoem862
Nordic	n/a	n/a	pcoem865
Portuguese	n/a	pcoem860	n/a

Description	Windows ENCODING= Value	MS-DOS ENCODING= Value	IBM-PC ENCODING= Value
Thai	n/a	n/a	pcoem874
Turkish	wturkish	n/a	pcoem857
USA	n/a	n/a	pcoem437
Vietnamese	wvietnamese	n/a	n/a
Western	wlatin1	n/a	pcoem850

Table 24.6 Windows Double-Byte Encodings

Description	PCMS ENCODING= Value	No Vendor ENCODING= Value
Traditional Chinese	n/a	big5
Simplified Chinese	ms-936	n/a
Japanese	ms-932	shift-jis
Korean	ms-949	n/a

Note: Windows also supports the utf-8 Unicode encoding. △

z/OS Encoding Values

The encodings in the following tables are valid in the z/OS operating environment.

Note: If an encoding-value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 24.7 Single-Byte Encodings for z/OS

Encoding ENCODING= Value	Description
EBCDIC037	Old North America (EBCDIC)
EBCDIC275	EBCDIC cp275-Brazil
EBCDIC425	EBCDIC cp425-Arabic
EBCDIC838	EBCDIC cp838-Thai
EBCDIC870	EBCDIC cp870-Central Europe
EBCDIC875	EBCDIC cp875-Greek
EBCDIC924	EBCDIC cp924-Western Europe
EBCDIC1025	EBCDIC cp1025-Cyrillic
EBCDIC1026	EBCDIC cp1026-Turkish
EBCDIC1047	EBCDIC cp1047-Latin1
EBCDIC1112	EBCDIC cp1112-Baltic
EBCDIC1122	EBCDIC cp1122-Estonian
EBCDIC1130	EBCDIC cp1130-Vietnamese

Encoding ENCODING= Value	Description
EBCDIC1140	EBCDIC cp1140-North America
EBCDIC1141	EBCDIC cp1141-German/Austrian
EBCDIC1142	EBCDIC cp1142-Danish/Norwegian
EBCDIC1143	EBCDIC cp1143-Finnish/Swedish
EBCDIC1144	EBCDIC cp1144-Italian
EBCDIC1145	EBCDIC cp1145-Spanish
EBCDIC1146	EBCDIC cp1146-English (UK)
EBCDIC1147	EBCDIC cp1147-French
EBCDIC1148	EBCDIC cp1148-International
OPEN_ED-037	OpenEdition EBCDIC
OPEN_ED-275	OpenEdition EBCDIC cp275-Brazil
OPEN_ED-425	OpenEdition EBCDIC cp425-Arabic
OPEN_ED-838	OpenEdition EBCDIC cp838-Thai
OPEN_ED-870	OpenEdition EBCDIC cp870-Central Europe
OPEN_ED-875	OpenEdition EBCDIC cp875-Greek
OPEN_ED-924	OpenEdition EBCDIC cp924-Western Europe
OPEN_ED-1025	OpenEdition EBCDIC cp1025-Cyrillic
OPEN_ED-1026	OpenEdition EBCDIC cp1026-Turkish
OPEN_ED-1047	OpenEdition EBCDIC cp1047-Latin1
OPEN_ED-1112	OpenEdition EBCDIC cp1112-Baltic
OPEN_ED-1122	OpenEdition EBCDIC cp1122-Estonian
OPEN_ED-1130	OpenEdition EBCDIC cp1130-Vietnamese
OPEN_ED-1140	OpenEdition EBCDIC cp1140-North America
OPEN_ED-1141	OpenEdition EBCDIC cp1141-German/Austrian
OPEN_ED-1142	OpenEdition EBCDIC cp1142-Danish/Norwegian
OPEN_ED-1143	OpenEdition EBCDIC cp1143-Finnish/Swedish
OPEN_ED-1144	OpenEdition EBCDIC cp1144-Italian
OPEN_ED-1145	OpenEdition EBCDIC cp1145-Spanish
OPEN_ED-1146	OpenEdition EBCDIC cp1146-English (UK)
OPEN_ED-1147	OpenEdition EBCDIC cp1147-French
OPEN_ED-1148	OpenEdition EBCDIC cp1148-International

Table 24.8 Double-Byte Encodings for z/OS

Description	ENCODING= Value
Japanese	IBM-939
Korean	IBM-933
Simplified Chinese	IBM-935
Traditional Chinese	IBM-937

PART 10

APPENDIX 1

Recommended Reading

Recommended Reading

Here is the recommended reading list for this title:

- *SAS Language Reference: Concepts*
- *SAS Language Reference: Dictionary*
- *SAS Output Delivery System: User's Guide*
- *Base SAS Procedures Guide*
- *SAS/CONNECT User's Guide*
- *SAS/GRAPH Reference, Volumes 1 and 2*
- SAS® Companion for your operating environment

For a complete list of SAS publications, see the current *SAS Publishing Catalog*. To order the most current publications or to receive a free copy of the catalog, contact a SAS representative at

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: (800) 727-3228*
Fax: (919) 677-8166
E-mail: **sasbook@sas.com**
Web address: **support.sas.com/pubs**
* For other SAS Institute business, call (919) 677-8000.

Customers outside the United States should contact their local SAS office.

Glossary

ANSI (American National Standards Institute)
an organization in the United States that coordinates voluntary standards and conformity to those standards. ANSI works with ISO to establish global standards. See also ISO (International Organization for Standardization).

ASCII (American Standard Code for Information Interchange)
a 7-bit encoding that is the U.S. national variant of ISO 646. The ASCII encoding includes the upper- and lowercase letters A-Z, digits, symbols (such as &, #, and mathematical symbols), punctuation marks, and control characters. This set of 128 characters is also included in most other encodings. See also ISO 646 family.

BIDI (bidirectional) text
a mixture of characters that are read from left to right and characters that are read from right to left. Most Arabic and Hebrew strings of text, for example, are read from right to left, but numbers and embedded Western terms within Arabic and Hebrew text are read from left to right.

CEDA (Cross-Environment Data Access)
a feature of SAS software that enables a SAS data file that was created in any directory-based operating environment (for example, Solaris, Windows, HP-UX, OpenVMS) to be read by a SAS session that is running in another directory-based environment. You can access the SAS data files without using any intermediate conversion steps. See also data representation.

character set
the set of characters that are used by a language or group of languages. A character set includes national characters, special characters (such as punctuation marks and mathematical symbols), the digits 0-9, and control characters that are needed by the computer. Most character sets also include the unaccented upper- and lowercase letters A-Z. See also national character.

code page
an ordered character set in which a numeric index (code point) is associated with each character. See also character set.

code point
a hexadecimal value that represents a character in an encoding or that is associated with a character on a code page. See also code page, encoding.

code position
the row and column location of a character in a code page. See also code page.

code table
another term for code page. See code page.

data representation
the form in which data is stored in a particular operating environment. Different operating environments use different standards or conventions for storing floating-point numbers (for example, IEEE or IBM 390); for character encoding (ASCII or EBCDIC); for the ordering of bytes in memory (big Endian or little Endian); for word alignment (4-byte boundaries or 8-byte boundaries); and for data-type length (16-bit, 32-bit, or 64-bit).

DBCS (double-byte character set)
any East Asian character set (Japanese, Korean, Simplified Chinese, and Traditional Chinese) that requires a mixed-width encoding because most characters occupy more than one byte of computer memory or storage. This term is somewhat misleading because not all characters in a DBCS require more than one byte, and some DBCS characters actually require four bytes. See also character set.

EBCDIC (Extended Binary Coded Decimal Interchange Code)
a group of 8-bit encodings that each include up to 256 characters. EBCDIC is used on IBM mainframes and on most IBM mid-range computers. EBCDIC follows ISO 646 conventions in order to facilitate transcoding between EBCDIC encodings, ASCII, the ISO 646 family of encodings, and 8-bit extensions to ASCII such as the ISO 8859 family. The 95 EBCDIC graphical characters include 82 invariant characters (including the SPACE character), which occupy the same code positions across most single-byte EBCDIC code pages, and 13 variant graphic characters, which occupy varying code positions across most single-byte EBCDIC code pages. See also ASCII (American Standard Code for Information Interchange), encoding, ISO (International Organization for Standardization), ISO 646 family, ISO 8859 family.

encoding
a set of characters (letters, logograms, digits, punctuation marks, symbols, and control characters) that have been mapped to hexadecimal values (called code points) that can be used by computers. An encoding results from applying an encoding method to a specific character set. Groups of encodings that apply the same encoding method to different character sets are sometimes referred to as families of encodings. For example, German EBCDIC is an encoding in the EBCDIC family, Windows Cyrillic is an encoding in the Windows family, and Latin 1 is an encoding in the ISO 8859 family. See also character set, encoding method.

encoding method
the set of rules that is used for assigning numeric representations to the characters in a character set. For example, these rules specify how many bits are used for storing the numeric representation of the character, as well as the ranges in the code page in which characters appear. The encoding methods are standards that have been developed in the computing industry. An encoding method is often specific to a computer hardware vendor. See also character set, encoding.

internationalization
the process of designing a software application without making assumptions that are based on a single language or locale. See also NLS (National Language Support).

ISO (International Organization for Standardization)
an organization that promotes the development of standards and that sponsors related activities in order to facilitate the dissemination of products and services

among nations and to support the exchange of intellectual, scientific, and technological information.

ISO 646 family
a group of 7-bit encodings that are defined in the ISO 646 standard and that each include up to 128 characters. The ISO 646 encodings are similar to ASCII except for 12 code points that are used for national variants. National variants are specific characters that are needed for a particular language. See also ASCII (American Standard Code for Information Interchange), ISO (International Organization for Standardization).

ISO 8859 family
a group of 8-bit extensions to ASCII that support all 128 of the ASCII code points plus an additional 128 code points, for a total of 256 characters. ISO-8859-1 (Latin 1) is a commonly used member of the ISO 8859 family of encodings. In addition to the ASCII characters, ISO-8859-1 contains accented characters, other letters that are needed for languages of Western Europe, and some special characters. See also ASCII (American Standard Code for Information Interchange), ISO (International Organization for Standardization).

language
an aspect of locale that is not necessarily unique to any one country or geographic region. For example, Portuguese is spoken in Brazil as well as in Portugal, but there are separate locales for Portuguese_Portugal and Portuguese_Brazil. See also locale.

locale
a value that reflects the language, local conventions, and culture for a geographic region. Local conventions can include specific formatting rules for dates, times, and numbers, and a currency symbol for the country or region. Collating sequences, paper sizes, and conventions for postal addresses and telephone numbers are also typically specified for each locale. Some examples of locale values are French_Canada, Portuguese_Brazil, and Chinese_Singapore.

localization
the process of adapting a product to meet the language, cultural, and other requirements of a specific target environment or market so that customers can use their own languages and conventions when using the product. Translation of the user interface, system messages, and documentation is part of localization.

MBCS (multi-byte character set)
a synonym for DBCS. See DBCS (double-byte character set).

national character
any character that is specific to a language as it is written in a particular nation or group of nations.

NLS (national language support)
the set of features that enable a software product to function properly in every global market for which the product is targeted.

SBCS (single-byte character set)
a character set in which each character occupies only one byte of computer memory or storage. A single-byte character set can be either 7 bits (providing up to 128 characters) or 8 bits (providing up to 256 characters). An example of an 8-bit SBCS is the ISO-8859-5 character set, which includes the Cyrillic characters that are used in Russian and other languages. See also character set.

transcoding
the process of converting the contents of a SAS file from one encoding to another encoding. Transcoding is necessary if the session encoding and the file encoding are

different, such as when transferring data from a Latin 1 encoding under UNIX to a German EBCDIC encoding on an IBM mainframe. See also encoding, translation table.

translation table
a SAS catalog entry that is used for transcoding data from one encoding to another encoding. SAS language elements that control locale values and encoding properties automatically invoke the appropriate translation table. Translation tables are specific to the operating environment. For example, there is a specific translation table that maps the Windows Latin 2 encoding to the ISO Latin 2 encoding. See also encoding, transcoding.

Unicode
a 16-bit encoding that supports the interchange, processing, and display of characters and symbols from dozens of writing systems, for a total of up to 65,536 characters. Unicode includes all characters from most modern written languages as well as characters from some historical languages.

Unicode Consortium
an organization that develops and promotes the Unicode standard. See also Unicode.

Index

E

F

I

J

K

L

M

N

Z

Your Turn

If you have comments or suggestions about *SAS® 9.1 National Language Support (NLS): User's Guide*, please send them to us on a photocopy of this page, or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**