Contents

What’s New vii
Overview vii
Documentation Enhancements vii
Internationalization Compatibility for SAS String Functions viii
Language Switching viii
Locales viii
Encodings ix
Autocall Macros ix
Formats xi
Informats xii
Functions xiii
System Options xiv

PART 1 NLS Concepts 1

Chapter 1 △ National Language Support (NLS) 3
Overview to National Language Support 3
Definition of Localization and Internationalization 4

Chapter 2 △ Locale for NLS 5
Overview of Locale Concepts for NLS 5
Specifying a Locale 6

Chapter 3 △ Encoding for NLS 9
Overview: Encoding for NLS 9
Difference between Encoding and Transcoding 12
Character Sets for Encoding in NLS 12
Common Encoding Methods 12
Standards Organizations for NLS Encodings 15
Code Point Discrepancies among EBCDIC Encodings 15
Collating Sequence 16
Determining the Encoding of a SAS Session and a Data Set 20
Default SAS Session Encoding 21
Setting the Encoding of a SAS Session 22
Encoding Behavior in a SAS Session 24

Chapter 4 △ Transcoding for NLS 27
Overview to Transcoding 27
Common Reasons for Transcoding 27
Transcoding and Translation Tables 28
SAS Options That Transcode SAS Data 29
Transcoding between Operating Environments 29
What’s New

Overview

In this release, SAS has expanded the scope and capabilities of National Language Support (NLS). NLS is a set of features that enable a software product to function properly in every global market for which the product is targeted. SAS contains NLS features to ensure that you can write SAS applications that conform to local language conventions. Typically, software that is written in the English language works well for users who use the English language and data that is formatted using the conventions that are observed in the United States. However, without NLS, these products might not work as 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.

This topic describes the changes and enhancements that have been made to the NLS documentation and features.

- additional autocall macros
- additional encodings
- additional functions and a new directive for selected functions
- additional locales
- additional system options
- formats that now support Arabic and new formats
- documentation enhancements such as revising the collating sequence topic and moving the EUR language elements to another section
- informats that now support Arabic and new informats
- internationalization compatibility for SAS string functions
- language switching

Documentation Enhancements

- The title of this document was changed for SAS 9.2 NLS. The new title is SAS National Language Support (NLS): Reference Guide.
- The Collating Sequences section, which describes the orders in which characters are sorted, has been revised.
SAS recommends that users use the NL language elements instead of the EUR language elements. The EUR language elements are in an appendix titled Additional Elements.

Internationalization Compatibility for SAS String Functions

The Internationalization Compatibility for SAS String Functions section specifies the level of internationalization compatibility for SAS string functions.

Language Switching

The Language Switching section describes how you can view SAS messages in another language using a Unicode server.

Locales

The following locales have been added in SAS 9.2 NLS. Information on how locales work in SAS programming is provided in Overview of Locale Concepts for NLS:

- Afrikaans_SouthAfrica
- Albanian_Albania
- Arabic_India
- Arabic_Iraq
- Arabic_Libya
- Arabic_Sudan
- Arabic_Syria
- Arabic_Yemen
- Bengali_India
- Bosnian_BosniaHerzegovina
- Catalan_Spain
- Cornish_UnitedKingdom
- Croatian_BosniaHerzegovina
- English_Belgium
- English_Botswana
- English_Caribbean
- English_Philippines
- English_Zimbabwe
- Faroese_FaroeIslands
- Greenlandic_Greenland
- Hindi_India
- Indonesian_Indonesia
- Macedonian_Macedonia
- Malay_Malaysia
- Maltese_Malta
- ManxGaelic_UnitedKingdom
Encodings

The following encodings have been added in SAS 9.2 NLS. Information on how encodings work in SAS programming is provided in Overview of Encoding for NLS:

- e097 - Farsi Bilingual - EBCDIC
- eofa - Farsi Bilingual - EBCDIC
- e137 - Devanagari - EBCDIC
- eoin - Devanagari - EBCDIC
- e153 - EBCDIC Latin 2 Multilingual with euro
- e053 - EBCDIC Latin 2 Multilingual with euro
- e154 - EBCDIC Cyrillic Multilingual with euro
- e054 - EBCDIC Cyrillic Multilingual with euro
- e155 - EBCDIC Turkey with euro
- e055 - EBCDIC Turkey with euro
- e156 - EBCDIC Baltic Multi with euro
- e056 - EBCDIC Baltic Multi with euro
- e157 - EBCDIC Estonia with euro
- e057 - EBCDIC Estonia with euro
- e158 - EBCDIC Cyrillic Ukraine with euro
- e058 - EBCDIC Cyrillic Ukraine with euro
- e905 - Latin 3 - EBCDIC
- e013 - Latin 3 - EBCDIC
- lat8 - ISO 8859/14–latin8
- p806 - PC Indian Script Code (ISCII–91)
- p098 - Farsi - Personal Computer

Autocall Macros

The following SAS 9.2 NLS autocall macros are new:

- %KCMPRES and %QKCMPRES
- %KLEFT and %QKLEFT
%KLOWCASE and %QKLOWCAS
%KTRIM and %QKTRIM
%KVERIFY

Formats

- The following SAS 9.2 NLS formats have been enhanced and now support Arabic:
  - $BIDI
  - $LOGVS
  - $LOGVSR
  - $VSLOG
  - $VSLOGR

- The following numeric formats are new for SAS 9.2 NLS.
  - NLBEST writes the best numerical notation, based on the locale.
  - NLSTRMON writes a numeric value as a day-of-the-month in the specified locale.
  - NLSTRQTR writes a numeric value as the quarter-of-the-year in the specified locale
  - NLSTRWK writes a numeric value as the day-of-the-week in the specified locale
  - NLPVALUE writes p-values of the local expression in the specified locale

- The following date and time formats are new for SAS 9.2 NLS. These formats write locale-specific dates and times.
  - NLDATEYQ converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.
  - NLDATEYR converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.
  - NLDATEYW converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.
  - NLDATMDT converts the SAS datetime value to the datetime value of the specified locale. This format writes the value as the name of the month, day of the month, and year.
  - NLDATMMN converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.
  - NLDATMWN converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.
  - NLDATMYQ converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.
  - NLDATMYR converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.
NLDATMYW converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.

- The following currency formats are new for SAS 9.2 NLS. These formats write the international monetary expression.
  - NLMNIAUD - Australia
  - NLMNICADw.d - Canada
  - NLMNICHFw.d - Liechtenstein
  - NLMNICNYw.d - China
  - NLMNIDKKw.d - Denmark, Faroe Island, and Greenland
  - NLMNIEURw.d - Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain
  - NLMNIGBPw.d - United Kingdom
  - NLMNIILSw.d - Israel
  - NLMNJPYw.d - Japan
  - NLMNIKRWw.d - South Korea
  - NLMNIMYMw.d - Malaysia
  - NLMNINOKw.d - Norway
  - NLMNINZDw.d - New Zealand
  - NLMNIPLNw.d - Poland
  - NLMNIRUBw.d - Russia
  - NLMNISEKw.d - Sweden
  - NLMNISGDw.d - Singapore
  - NLMNITWDw.d - Thailand
  - NLMNIUSDw.d - Puerto Rico, and United States
  - NLMNIZARw.d - South Africa

- The following currency formats for SAS 9.2 NLS are new. These formats write the local monetary expression.
  - NLMNLAUDw.d - Australia
  - NLMNLCADw.d - Canada
  - NLMNLCHFw.d - Liechtenstein
  - NLMNLCNYw.d - China
  - NLMNLDKKw.d - Denmark, Faroe Island, Greenland
  - NLMNLEURw.d - Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Netherlands, Portugal, and Spain
  - NLMNLGBPw.d - United Kingdom
  - NLMNLHKDw.d - Hong Kong
  - NLMNIILSw.d - Israel
  - NLMNJPYw.d - Japan
  - NLMNKRWw.d - South Korea
  - NLMNMYRw.d - Malaysia
  - NLMNOKw.d - Norway
  - NLMNZDw.d - New Zealand
  - NLMNPLNw.d - Poland
  - NLMNRUBw.d - Russia
The following SAS 9.2 NLS informats have been enhanced and now support Arabic:

- $LOGVS
- $LOGVSR
- $VSLOG
- $VSLOGR

The following currency informats are new for SAS 9.2 NLS. These informats read the international monetary expression.

- NLMNIAUDw.d - Australia
- NLMNICADw.d - Canada
- NLMNICHFw.d - Liechtenstein and Switzerland
- NLMNICNYw.d - China
- NLMNIDKKw.d - Denmark, Faroe Island, and Greenland
- NLMNIEURw.d - Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain
- NLMNIGBPw.d - United Kingdom
- NLMNIHKDw.d - Hong Kong
- NLMNIILSw.d - Israel
- NLMNIJPYw.d - Japan
- NLMNIKRWw.d - South Korea
- NLMNIMYRw.d - Malaysia
- NLMNINOKw.d - Norway
- NLMNINZDw.d - New Zealand
- NLMNIPLNW.d - Poland
- NLMNIRUBw.d - Russia
- NLMNISEKw.d - Sweden
- NLMNISGDw.d - Singapore
- NLMNITWDw.d - Taiwan
- NLMNIUSDw.d - Puerto Rico, and the United States
- NLMNIZARw.d - South Africa

The following currency informats are new for SAS 9.2 NLS. These informats read the local monetary expression.

- NLMNLAUDw.d - Australia
- NLMNLCADw.d - Canada
- NLMNLCNFw.d - Liechtenstein and Switzerland
 Functions

The following functions are new for SAS 9.2 NLS:

- **GETPXLANGUAGE**
  displays a transcoding error when illegal data is read from a remote application

- **GETPXLOCALE**
  returns the POSIX locale value for a SAS locale

- **GETPXREGION**
  returns the current, two-letter region code

- **KPROPACASE**
  converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters

- **KPROPCHAR**
  converts special characters to normal characters

- **KPROPDATA**
  removes or converts unprintable characters

- **SORTKEY**
  creates a linguistic sort key

- **UNICODE**
  converts Unicode characters to the current SAS session encoding

- **UNICODEC**
  converts characters in the current SAS session encoding to Unicode characters
UNICODELEN
creates a linguistic sort key

UNICODEWIDTH
specifies the length of a display unit for the Unicode data

☐ A new directive, “#”, was added to the following functions:
  ☐ NLDATE
  ☐ NLDATM
  ☐ NLTIME

System Options

The following system options are new for SAS 9.2 NLS:

BOMFILE
  specifies whether to write the Byte Order Mark (BOM) prefix on Unicode-encoded external files

LOCALELANGCHG
  determines whether the language of the text of the ODS output can be changed

RSASIOTRANSERROR
  displays a transcoding error when illegal data is read from a remote application
NLS Concepts

Chapter 1 . . . . . . National Language Support (NLS)  3
Chapter 2 . . . . . . Locale for NLS  5
Chapter 3 . . . . . . Encoding for NLS  9
Chapter 4 . . . . . . Transcoding for NLS  27
Chapter 5 . . . . . . Double-Byte Character Sets (DBCS)  35
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 action 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 use 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.
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 might 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 might be represented as a number or as a name. The name might be fully spelled or abbreviated. The order of the month, day, and year might 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: 03–10–02
- 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 might 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:

```plaintext
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 15, “Values for the LOCALE= System Option,” on page 545.

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:

```plaintext
sas9 -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 456.
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.

Note: For more details about the differences between the LOCALE= and ENCODING= options, see “Setting the Encoding of a SAS Session” on page 22.

Windows example:

```sas
options locale=Italian_Ialy;
```

The Italian_Ialy 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 “DATESTYLE= System Option” on page 456.

Language Switching

SAS messages are displayed in the language that is specified by the settings in the SAS configuration file during startup. In the Unicode server, you can view SAS messages in another language by using the Language Switching feature. You can access the Language Switching feature with the LOCALELANGCHG system option. If LOCALELANGCHG is enabled, then the value of the LOCALE system option determines the language for procedure output, user interface elements and ODS fonts. If LOCALELANGCHG is disabled, then messages will appear in the language that is set during startup. This feature is supported in the Unicode server. For more information, see the “LOCALELANGCHG System Option” on page 466.
Overview: 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.
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 (1141)

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

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 such as the de_DE locale for German in Germany. 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 27.

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 about how SAS uses SBCS encodings, see Chapter 17, “Encoding Values in SAS Language Elements,” on page 555.

- **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 about how SAS uses DBCS encodings, see Chapter 16, “SAS System Options for Processing DBCS Data,” on page 553.

- **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 15. The common encoding methods are listed here:

**ASCII (American Standard Code for Information Interchange)**
- 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**
- 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 blank 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**
- 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**
- 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 Unicode.

**Unicode**
- provides up to 99,024 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 4 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

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

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

<table>
<thead>
<tr>
<th>Standard</th>
<th>Name of Encoding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB 2312-80</td>
<td>Simplified Chinese</td>
<td>People’s Republic of China</td>
</tr>
<tr>
<td>CNS 11643</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>Big-5</td>
<td>Traditional Chinese</td>
<td>Taiwan</td>
</tr>
<tr>
<td>KS C 5601</td>
<td>Korean National Standard</td>
<td>Korea</td>
</tr>
<tr>
<td>JIS</td>
<td>Japan Industry Standard</td>
<td>Japan</td>
</tr>
<tr>
<td>Shift-JIS</td>
<td>Japan Industry Standard multibyte encoding</td>
<td>Japan</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>EBCDIC Code Points</th>
<th>U.S. English</th>
<th>Finnish</th>
<th>Spanish</th>
<th>Austrian/German</th>
</tr>
</thead>
<tbody>
<tr>
<td>4A</td>
<td>ç</td>
<td>§</td>
<td>[</td>
<td>Ä</td>
</tr>
<tr>
<td>4F</td>
<td>!</td>
<td>!</td>
<td>!</td>
<td></td>
</tr>
<tr>
<td>5A</td>
<td>!</td>
<td>ø</td>
<td>]</td>
<td>Ü</td>
</tr>
<tr>
<td>5B</td>
<td>$</td>
<td>Å</td>
<td>$</td>
<td>$</td>
</tr>
<tr>
<td>5F</td>
<td>¬</td>
<td>^</td>
<td>¬</td>
<td>^</td>
</tr>
<tr>
<td>6A</td>
<td>!</td>
<td>ö</td>
<td>ñ</td>
<td>ö</td>
</tr>
<tr>
<td>79</td>
<td>‘</td>
<td>’</td>
<td>’</td>
<td></td>
</tr>
<tr>
<td>7B</td>
<td>#</td>
<td>Ä</td>
<td>Ñ</td>
<td>#</td>
</tr>
<tr>
<td>7C</td>
<td>@</td>
<td>Ø</td>
<td>@</td>
<td>$</td>
</tr>
<tr>
<td>A1</td>
<td>~</td>
<td>ü</td>
<td>&quot;</td>
<td>ß</td>
</tr>
<tr>
<td>C0</td>
<td>[</td>
<td>ä</td>
<td>[</td>
<td>ä</td>
</tr>
<tr>
<td>D0</td>
<td>]</td>
<td>à</td>
<td>]</td>
<td>ü</td>
</tr>
<tr>
<td>E0</td>
<td>\</td>
<td>È</td>
<td>\</td>
<td>Ö</td>
</tr>
</tbody>
</table>

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 $.

Collating Sequence

Overview to Collating Sequence

The collating sequence is the order in which characters are sorted. For example, when the SORT procedure is executed, the collating sequence determines the sort order (higher, lower, or equal to) of a particular character in relation to other characters.

The default collating sequence is binary collation, which sorts characters according to each character’s location in the code page of the session encoding. (The session encoding is the default encoding for a SAS session. The default encoding can be specified by using various SAS language elements.) The sort order corresponds directly to the arrangement of the code points within the code page. The two single-byte character encoding methods that data processing uses most widely are ASCII and EBCDIC. The OpenVMS, UNIX, and Windows operating environments use ASCII encodings; IBM mainframe computers use EBCDIC encodings.

Binary collation is the fastest type of collation because it is the most efficient for the computer. However, locating characters within a binary-collated report might be difficult if you are not familiar with this method. For example, a binary-collated report lists words beginning with uppercase characters separately from words beginning with lowercase characters, and words beginning with accented characters after words beginning with unaccented characters. Therefore, for ASCII-based encodings, the capital letter Z precedes the lowercase letter a. Similarly, for EBCDIC-based encodings, the lowercase letter z precedes the capital letter A.

You can request an alternate collating sequence that overrides the binary collation. To request an alternate collating sequence, specify one of the following sequences:

- a translation table name
- an encoding value
- linguistic collation

Table 3.4 on page 16 illustrates the results of using different collating sequences to sort a short list of words:

Table 3.4  Results of Different Collating Sequences

<table>
<thead>
<tr>
<th>Binary</th>
<th>Translation Table</th>
<th>Encoding Value</th>
<th>Linguistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aaron</td>
<td>aardvark</td>
<td>Aaron</td>
<td>aardvark</td>
</tr>
<tr>
<td>Aztec</td>
<td>azimuth</td>
<td>Aztec</td>
<td>Aaron</td>
</tr>
<tr>
<td>Zeus</td>
<td>Aaron</td>
<td>Zeus</td>
<td>azimuth</td>
</tr>
<tr>
<td>aardvark</td>
<td>Aztec</td>
<td>aardvark</td>
<td>Aztec</td>
</tr>
<tr>
<td>azimuth</td>
<td>cote</td>
<td>azimuth</td>
<td>cote</td>
</tr>
</tbody>
</table>
The first column shows the results of binary collation on characters that are represented in an ASCII-based encoding. The alphabetization is not consistent because of the separate grouping of words that begin with uppercase and lowercase characters. For example, the word *Zeus* appears before *aardvark* because of the code points that are assigned to the characters within the ASCII-based encoding.

The second column shows the results of specifying a translation table that alternates the ordering of lowercase and uppercase characters. If you use the translation table, the word *aardvark* appears before *Zeus*. However, the word *azimuth* appears before *Aaron* because the translation table assigns a weight value to the lowercase character *a* that is less than the weight value of the uppercase character *A*. In addition, accents are sorted from left to right. For example, *coté* comes before *côte*.

The third column shows the results of specifying the ASCII-based, double-byte latin1 encoding.

The last column shows the results of linguistic collation for the session locale fr_FR (French_France), which uses a collation algorithm to alphabetize words. The algorithm specifies that words beginning with lowercase characters appear before words beginning with uppercase characters. In addition, this linguistic collation sorts accents from right to left because of the French locale specification.

SAS has adopted the International Components for Unicode (ICU) to implement linguistic collation. The ICU and its implementation of the Unicode Collation Algorithm (UCA) have become a standard. The collating sequence is the default provided by the ICU for the specified locale.

### Request Alternate Collating Sequence

To request an alternate collating sequence, use the following SAS language elements:

- **SORTSEQ=** option in the PROC SORT statement. See “Collating Sequence Option” on page 477.
- **SORTSEQ=** system option. See “SORTSEQ= System Option: UNIX, Windows, and z/OS” on page 470.

Note that neither method supports all of the collating sequences. For example, only the SORTSEQ= option in the PROC SORT statement supports linguistic collation. However, both the SORTSEQ= option in the PROC SORT statement and the SORTSEQ= system option support translation table collating sequences.

The BASE (V9) engine and the REMOTE engine for SAS/SHARE support all alternate collating sequences. The V9TAPE sequential engine supports the use of a translation table and an encoding value to sort data, but the V9TAPE engine does not support linguistic collation.

<table>
<thead>
<tr>
<th>Binary</th>
<th>Translation Table</th>
<th>Encoding Value</th>
<th>Linguistic</th>
</tr>
</thead>
<tbody>
<tr>
<td>cote</td>
<td>coté</td>
<td>cote</td>
<td>côté</td>
</tr>
<tr>
<td>coté</td>
<td>côté</td>
<td>coté</td>
<td>coté</td>
</tr>
<tr>
<td>côté</td>
<td>côté</td>
<td>côté</td>
<td>côté</td>
</tr>
<tr>
<td>côté</td>
<td>zebra</td>
<td>côté</td>
<td>zebra</td>
</tr>
<tr>
<td>zebra</td>
<td>zèbre</td>
<td>zebra</td>
<td>zèbre</td>
</tr>
<tr>
<td>zèbre</td>
<td>Zeus</td>
<td>zèbre</td>
<td>Zeus</td>
</tr>
</tbody>
</table>
Specifying a Translation Table

A translation table is a SAS catalog entry that transcodes data from one single-byte encoding to another single-byte encoding. A translation table also reorders characters when sorting them. A translation table can be one that SAS provides, such as a standard collating sequence like ASCII, EBCDIC, or DANISH; or it can be a user-defined translation table.

When you specify a translation table for an alternate collating sequence, the characters are reordered by mapping the code point of each character to an integer weight value in the range of 0 to 255. A binary collation is then performed.

For collating purposes, you can create translation tables that order characters so that lowercase and uppercase characters alternate. For example, you can create a translation table to correct the situation in which Z precedes a in an ASCII-based encoding. (However, regardless of the weight assignments in the translation table, it is difficult to achieve a true alphabetic ordering that takes the character case into account.) You can also create a translation table that orders alphabetic characters of a particular language in their expected order.

The TRANTAB procedure creates, edits, and displays translation tables. For example, you can display a translation table to view the character-weight values. The translation tables that are supplied by SAS 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. See Chapter 14, “The TRANTAB Procedure,” on page 515 for more information about translation tables.

You can specify a translation table with the SORTSEQ= option in the PROC SORT statement or with the SORTSEQ= system option. For example, if your operating environment sorts with the ASCII-based Wlatin1 encoding by default, and you want to sort with a translation table that alternates uppercase and lowercase characters, issue the following statements to specify the SAS translation table FRSOLAT1:

```sas
proc sort data=myfiles.test sortseq=FRSOLAT1;
   by name;
run;
```

A SAS data set that is sorted with a translation table contains a sort indicator that displays the specified translation table name as the collating sequence in CONTENTS procedure output.

Specifying an Encoding Value

An encoding is a set of characters (letters, logograms, digits, punctuation marks, symbols, and control characters) that have been mapped to hexadecimal values, called code points, that computers use. When you specify an encoding value for an alternate collating sequence, the characters are transcoded from the SAS session encoding to the specified encoding, and then a binary collation is performed. You can specify all encoding values that are supported by the ENCODING= option, including multi-byte encodings. Note that specifying a translation table can transcode data, but translation tables are limited to single-byte encodings.
You can specify an encoding value with the SORTSEQ= option in the PROC SORT statement, but you cannot specify an encoding value in the SORTSEQ= system option. For example, you want to sort a SAS data set and then transport it to a Japanese Windows environment. If your session encoding is ASCII-based and binary collation is in effect, you can issue the following statements to specify the ASCII-based double-byte encoding SHIFT-JIS:

```plaintext
proc sort data=myfiles.test sortseq='shift-jis';
  by name;
run;
```

Note that SAS checks the encoding value for any translation tables with the same name. If a translation table name exists, SAS uses the translation table.

A SAS data set that is sorted with an encoding value contains a sort indicator that displays the specified encoding value as the collating sequence in CONTENTS procedure output.

### Specifying Linguistic Collation

Linguistic collation sorts characters according to rules of language and produces results that are intuitive and culturally acceptable. The results are similar to the collation used in printed materials such as dictionaries, phone books, and book indexes. Linguistic collation is useful for generating reports or other data presentations and for achieving compatibility between systems.

SAS incorporates the International Components for Unicode (ICU), which is an open-source library that provides routines for linguistic collation that are compatible with the Unicode Collation Algorithm (UCA). The UCA is a standard by which Unicode strings can be compared and ordered.

To request linguistic collation, you must use the SORTSEQ= option in the PROC SORT statement because the SORTSEQ= system option does not support linguistic collation. For example, the following statements cause the SORT procedure to collate linguistically, in accordance with the French_France locale:

```plaintext
options locale=fr_FR;
proc sort data=myfiles.test sortseq=linguistic;
  by name;
run;
```

When linguistic collation is requested, SAS uses the default linguistic collation algorithm that is provided by the ICU for the SAS session locale. This algorithm reflects the language, local conventions such as data formatting, and culture for a geographical region. You can modify the algorithm by specifying options in parentheses following the LINGUISTIC keyword. For example, you can specify a different locale; you can specify the CASE_FIRST= option to collate lowercase characters before uppercase characters, or vice versa; and so on. Generally, it is not necessary to specify options, because the ICU associates defaults with the various languages and locales. For more information about the linguistic options, see the SORTSEQ= option in “Collating Sequence Option” on page 477 or the SORTSEQ= option in the PROC SORT statement in Base SAS Procedures Guide.

A SAS data set that is sorted linguistically contains a sort indicator that displays the collating sequence LINGUISTIC in CONTENTS procedure output. Along with the sort indicator, the data set also records a complete description of the linguistic collating
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:

```sas
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.
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.

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 3.5  Default SAS Session Encoding Values

<table>
<thead>
<tr>
<th>Operating Environment</th>
<th>Default ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenVMS for integrity servers</td>
<td>Latin1</td>
<td>Western (ISO)</td>
</tr>
<tr>
<td>z/OS</td>
<td>OPEN_ED-1047</td>
<td>OpenEdition EBCDIC</td>
</tr>
<tr>
<td></td>
<td>cp1047-Latin1</td>
<td></td>
</tr>
<tr>
<td>UNIX</td>
<td>Latin1</td>
<td>Western (ISO)</td>
</tr>
<tr>
<td>Windows</td>
<td>WLatin1</td>
<td>Western (Windows)</td>
</tr>
</tbody>
</table>

For a complete list of supported encoding values for a SAS session, see Chapter 18, “Encoding Values for a SAS Session,” on page 561.

---

**Setting the Encoding of a SAS Session**

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

*Note:* Values for the ENCODING= system option depend on the operating environment.

The priority order for setting the encoding is as follows:

1. **ENCODING= system option**
   - The SAS session encoding is determined by the ENCODING= option regardless of whether the DBCS or LOCALE= options are specified. If the ENCODING= option is specified, a set of valid DBCS options are set regardless of whether the user has specified those options. Also, if the ENCODING= option is specified, the LOCALE= option is set to an appropriate value unless a value has been specified by the user.
   
   *Note:* If the ENCODING= option is specified, the TRANTAB= option is implicitly set.

2. **DBCS options**
   - If the ENCODING= option is not specified, the SAS session encoding is determined by the DBCS options regardless of whether the LOCALE= option is specified. The LOCALE= option is set to an appropriate value unless a value has been specified by the user.
   - The encoding is determined by the values of the DBCSLANG and DBCSTYPE options for DBCS languages, such as Japanese, Korean, Simplified Chinese, and Traditional Chinese.
   - The DBCS options are valid only when the DBCS extension directory is included in the path option list. The path of the DBCS extension dynamic link library (DLLs) has to be located at the top of the pathname list of the path option for the DBCS languages when you want to invoke a DBCS SAS session. The DBCS extension DLLs are located in the directory `!SASROOT/dbcs/sasexe` by default.
   - Also you might have to specify the resourcesloc, msg, and sashelp options to use localized resources even if the SAS session encoding is not a DBCS language (for example, Polish, German, and French). The localized resources are located under `!SASROOT/nls/<language identifier>/<sasmsg, sashelp, sasmacro, resource>`. The values for language identifiers are: cs, de, en, es, fr, hu, it, ja, ko, pl, ru, sv, zh, and zt.
You can specify a sasv9.cfg file located in the localized directories such as
%SASROOT/nls/<language identifier> so that you do not have to consider
using the path, resourcesloc, sasmsg, and sashelp options.

If DBCS (which specifies that SAS process DBCS encodings) is specified,
DBCSLANG= and DBCSTYPE= options are implicitly set. The default values for
DBCTYPE= and DBCSLANG= match those values for the Japanese environment
on the host.

3 LOCATE= system option

The SAS session encoding is determined by the LOCATE= option and the
platform, if the ENCODING= or DBCS options are not specified.

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

sas9 -locale Spanish_Spain

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

The following example shows that the wlatin2 encoding is set explicitly when
SAS is invoked:

sas9 -encoding wlatin2

Note: Setting DBCS encodings, DBCS options, or a CJK locale on SAS if the
DBCS extensions are not available will fail to successfully invoke SAS. △

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

In Table 3.6 on page 23, the following values for the CJK locales are based on locale
and platform:

<table>
<thead>
<tr>
<th>Locales</th>
<th>WIN (sas4)</th>
<th>MVS (sas4)</th>
<th>UNX (sas4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>zh_TW</td>
<td>MS-950 (ywin)</td>
<td>IBM-937 (yibm)</td>
<td>sax, s64: EUC-TW (yeuc) others: MS-950 (ywin)</td>
</tr>
<tr>
<td>zh_HK</td>
<td></td>
<td>IBM-935 (zibm)</td>
<td>EUC-CN (zeuc)</td>
</tr>
<tr>
<td>zh_MO</td>
<td>EUC-CN (zeuc)</td>
<td>IBM-939 (jibm)</td>
<td>h64, h6i, r64: SHIFT-JIS (sjs) others: EUC-JP (juec)</td>
</tr>
<tr>
<td>zh_CN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zh_SG</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ja_JP</td>
<td>SHIFT-JIS (sjis)</td>
<td>IBM-939 (jibm)</td>
<td></td>
</tr>
</tbody>
</table>
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. In SAS 9, DBCS recognizes the DBCSTYP value and converts it to the encoding value and specifies it in the descriptor portion of the field, by default.

- Data sets that are created in SAS 7 and SAS 8 do 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. In SAS 8, DBCS has the DBCSTYP field, instead of the encoding field.

- 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 468.

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 29.
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 17, “Encoding Values in SAS Language Elements,” on page 555.

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
Transcoding and Translation Tables

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

The following figure shows a translation table. The area of a translation table 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."

![SAS Windows Latin 2 to ISO Latin 2 Translation Table](image)

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” on page 471. The TRANTAB procedure is used to create, edit, and display customized translation tables. For details, see Chapter 14, “The TRANTAB Procedure,” on page 515.

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 transcodign.

Table 4.1 SAS Options That Transcode SAS Data

<table>
<thead>
<tr>
<th>Option</th>
<th>Where Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARSET=</td>
<td>ODS MARKUP statement</td>
</tr>
<tr>
<td>CORRECTENCODING=</td>
<td>MODIFY statement of the DATASETS procedure</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>%INCLUDE, FILE, FILENAME, INFILE, ODS statements; FILE and INCLUDE commands</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>in a DATA step</td>
</tr>
<tr>
<td>ENCODING=</td>
<td>LIBNAME statement</td>
</tr>
<tr>
<td>OUTENCODING=</td>
<td>LIBNAME statement for XML</td>
</tr>
<tr>
<td>XMLENCODING=</td>
<td>LIBNAME statement for XML</td>
</tr>
</tbody>
</table>

For a list of supported encoding values to use for these options, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 555.

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 have 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. The incorrect encoding stamp can be corrected with the CORRECTENCODING= option in the MODIFY statement in PROC DATASETS. 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 18, “Encoding Values for a SAS Session,” on page 561.

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 newline 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 18, “Encoding Values for a SAS Session,” on page 561.

---

**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 43 and “INENCODING= and OUTENCODING= Options” on page 493.
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 might 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 “ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS” on page 461.

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:

<table>
<thead>
<tr>
<th>Language Element</th>
<th>Type</th>
<th>Use</th>
<th>See</th>
</tr>
</thead>
<tbody>
<tr>
<td>KCVT</td>
<td>function</td>
<td>Converts DBCS data from one operating environment to another</td>
<td>“KCVT Function” on page 268</td>
</tr>
<tr>
<td>CPORT</td>
<td>procedure</td>
<td>Moves files from one environment to another</td>
<td>Base SAS Procedures Guide</td>
</tr>
<tr>
<td>CIMPORT</td>
<td>procedure</td>
<td>Imports a transport file created by CPORT</td>
<td>Base SAS Procedures Guide</td>
</tr>
</tbody>
</table>
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 use 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 might 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). An 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 before 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, 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

SAS Language Elements for NLS Data

Chapter 6. ........... Data Set Options for NLS 43
Chapter 7. ........... Formats for NLS 47
Chapter 8. ........... Functions for NLS 243
Chapter 9. ........... Informats for NLS 309
Chapter 10. .......... Autocall Macros for NLS 449
Chapter 11. .......... System Options for NLS 453
Chapter 12. .......... Options for Commands, Statements, and Procedures for NLS 475
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

<table>
<thead>
<tr>
<th>Category</th>
<th>Data Set Options for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Set Control</td>
<td>“ENCODING= Data Set Option” on page 43</td>
<td>Overrides the encoding to use for reading or writing a SAS data set.</td>
</tr>
<tr>
<td></td>
<td>“OUTREP= Data Set Option” on page 46</td>
<td>Specifies the data representation for the output SAS data set.</td>
</tr>
</tbody>
</table>

### 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 | ASCIANY | 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 Chapter 3, “Encoding for NLS,” on page 9.

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 inherits 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 uses the encoding of the current session.

Note: Character metadata and data output appears garbled if you specify a different encoding from where the data set was created.

In this example, the data set to be printed is internally encoded as ASCII, however the data set option specifies an EBCDIC encoding. SAS attempts to transcode the data from EBCDIC to ASCII, but the data is already in ASCII. The result is garbled data.
data a;
  x=1;
  abc='abc';
run'
proc print data=a (encoding='ebcdic');
run;

Note: The following values for ENCODING= are invalid:
  □ UCS2  
  UCS4  
  UTF16  
  UTF32

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 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, the correct Latin1 characters in a Latin1 session encoding and correct Latin2 characters in a Latin2 session encoding are displayed.

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 action might be necessary, for example, if the normal access to the file uses a different session encoding.

For example, if the current session encoding is W拉丁1, you can specify ENCODING=W拉丁2 in order to create the data set that uses the encoding W拉丁2. The following statements tell SAS to write the data to the new data set using the W LATIN2 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
Options in Statements and Commands:
  “ENCODING= Option” on page 489
  “INENCODING= and OUTENCODING= Options” on page 493
System Options:
  “ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS” on page 461
  “LOCALE System Option” on page 465

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
CHAPTER 7

Formats for NLS

International Date and Datetime Formats 50
Currency Representation 55
Overview to Currency 55
U.S. Dollars 55
Localized Euros 56
Customized Currency Representations 56
Localized National and International Currency Representations 57
Unique National and International Monetary Representations 59
Example: Representing Currency in National and International Formats 61

European Currency Conversion 62
Overview to European Currency Conversion 62
Fixed Rates for Euro Conversion 62
Variable Rates for Euro Conversion 62
Example: Converting between a European Currency and Euros 63
Direct Conversion between European Currencies 64

Formats for NLS by Category 64
$BIDw. Format 73
$SCPTDw. Format 75
$SCPTWw. Format 76
EUROw.d Format 77
EUROXw.d Format 78
HDATEw. Format 79
HEBDATEw. Format 80
$KANJw. Format 82
$KANJIXw. Format 83
$LOGVSw. Format 84
$LOGVSRw. Format 85
MINGUOw. Format 86
NENGOw. Format 87
NLBESTw. Format 89
NLDATEx. Format 91
NLDATEMDw. Format 92
NLDATEMNw. Format 93
NLDATExw. Format 94
NLDATEWNw. Format 95
NLDATEYMw. Format 96
NLDATEYQw. Format 97
NLDATEYRw. Format 98
NLDATEYXw. Format 98
NLDATMw. Format 99
NLDATMAPw. Format 100
<table>
<thead>
<tr>
<th>Code</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLDATMDTw.</td>
<td>101</td>
</tr>
<tr>
<td>NLDATMMDw.</td>
<td>102</td>
</tr>
<tr>
<td>NLDATMMNw.</td>
<td>103</td>
</tr>
<tr>
<td>NLDATMTMw.</td>
<td>104</td>
</tr>
<tr>
<td>NLDATMWNw.</td>
<td>105</td>
</tr>
<tr>
<td>NLDATMWWw.</td>
<td>106</td>
</tr>
<tr>
<td>NLDATMYMw.</td>
<td>107</td>
</tr>
<tr>
<td>NLDATMYQw.</td>
<td>107</td>
</tr>
<tr>
<td>NLDATMYRw.</td>
<td>108</td>
</tr>
<tr>
<td>NLDATMYWw.</td>
<td>109</td>
</tr>
<tr>
<td>NLMNIAEDw.d</td>
<td>110</td>
</tr>
<tr>
<td>NLMNIAUDw.d</td>
<td>111</td>
</tr>
<tr>
<td>NLMNIBGNw.d</td>
<td>112</td>
</tr>
<tr>
<td>NLMNIBRGWw.d</td>
<td>113</td>
</tr>
<tr>
<td>NLMNIDw.d</td>
<td>114</td>
</tr>
<tr>
<td>NLMNICHWw.d</td>
<td>115</td>
</tr>
<tr>
<td>NLMNICNWd</td>
<td>116</td>
</tr>
<tr>
<td>NLMNICZKw.d</td>
<td>117</td>
</tr>
<tr>
<td>NLMNIDkkw.d</td>
<td>118</td>
</tr>
<tr>
<td>NLMNIEEKw.d</td>
<td>119</td>
</tr>
<tr>
<td>NLMNIEGPw.d</td>
<td>120</td>
</tr>
<tr>
<td>NLMNIEURw.d</td>
<td>121</td>
</tr>
<tr>
<td>NLMNIBRPw.d</td>
<td>122</td>
</tr>
<tr>
<td>NLMNIHKDw.d</td>
<td>123</td>
</tr>
<tr>
<td>NLMNIIHRkw.d</td>
<td>124</td>
</tr>
<tr>
<td>NLMNIHUFw.d</td>
<td>125</td>
</tr>
<tr>
<td>NLMNIDRWd</td>
<td>126</td>
</tr>
<tr>
<td>NLMNIILSw.d</td>
<td>127</td>
</tr>
<tr>
<td>NLMNIINRWd</td>
<td>128</td>
</tr>
<tr>
<td>NLMNIPYw.d</td>
<td>129</td>
</tr>
<tr>
<td>NLMNIKRWd</td>
<td>130</td>
</tr>
<tr>
<td>NLMNILLTw.d</td>
<td>131</td>
</tr>
<tr>
<td>NLMNILTVw.d</td>
<td>132</td>
</tr>
<tr>
<td>NLMNIOPw.d</td>
<td>133</td>
</tr>
<tr>
<td>NLMNIXNWd</td>
<td>134</td>
</tr>
<tr>
<td>NLMNMYRw.d</td>
<td>135</td>
</tr>
<tr>
<td>NLMNINOKw.d</td>
<td>136</td>
</tr>
<tr>
<td>NLMNINZDw.d</td>
<td>137</td>
</tr>
<tr>
<td>NLMNIPNLw.d</td>
<td>138</td>
</tr>
<tr>
<td>NLMNIROLw.d</td>
<td>139</td>
</tr>
<tr>
<td>NLMNIRUBw.d</td>
<td>140</td>
</tr>
<tr>
<td>NLMNISEKw.d</td>
<td>141</td>
</tr>
<tr>
<td>NLMNISGDw.d</td>
<td>142</td>
</tr>
<tr>
<td>NLMNISITw.d</td>
<td>143</td>
</tr>
<tr>
<td>NLMNISKKw.d</td>
<td>144</td>
</tr>
<tr>
<td>NLMNITHBw.d</td>
<td>145</td>
</tr>
<tr>
<td>NLMNITRYw.d</td>
<td>146</td>
</tr>
<tr>
<td>NLMNITWDw.d</td>
<td>147</td>
</tr>
<tr>
<td>NLMNIUSDw.d</td>
<td>148</td>
</tr>
<tr>
<td>NLMNIZARw.d</td>
<td>149</td>
</tr>
<tr>
<td>NLMNLAEDx.d</td>
<td>150</td>
</tr>
<tr>
<td>NLMNLAUDw.d</td>
<td>151</td>
</tr>
<tr>
<td>NLMNLABNGWw.d</td>
<td>152</td>
</tr>
<tr>
<td>NLMNLABRLw.d</td>
<td>153</td>
</tr>
</tbody>
</table>
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 7.1 International Date and Datetime Formats

<table>
<thead>
<tr>
<th>Language</th>
<th>English Format</th>
<th>International Format</th>
<th>Min</th>
<th>Max</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEDMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWNN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>3</td>
<td>40</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>2</td>
<td>40</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>2</td>
<td>31</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNAME.</td>
<td>NLDATENWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEW.</td>
<td>2</td>
<td>38</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOWNNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDATX.</td>
<td>2</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>French (FRA)</td>
<td></td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOWNNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONNY.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDATX.</td>
<td>3</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>German (DEU)</td>
<td></td>
<td>DATE.</td>
<td>105</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOWNNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDATX.</td>
<td>3</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td></td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOWNNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MONYY.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDATX.</td>
<td>3</td>
<td>40</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WEEKDAY.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>WORDDATX.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td></td>
<td>DATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATETIME.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DOWNNAME.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>---------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEWX.</td>
<td>3</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEWX.</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDDATX.</td>
<td>EURDFWDX.</td>
<td>1</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Norwegian (NOR)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEWX.</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>20</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEWX.</td>
<td>2</td>
<td>40</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE.</td>
<td>NLDATE.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td></td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEWX.</td>
<td>3</td>
<td>38</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td>Language</td>
<td>English Format</td>
<td>International Format</td>
<td>Min</td>
<td>Max</td>
<td>Default</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------</td>
<td>----------------------</td>
<td>-----</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYIM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEX.</td>
<td>2</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYIM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEX.</td>
<td>1</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYIM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEX.</td>
<td>3</td>
<td>40</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONNAME.</td>
<td>NLDATEMN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>MONYY.</td>
<td>NLDATEYIM.</td>
<td>6</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WEEKDATX.</td>
<td>NLDATEX.</td>
<td>3</td>
<td>26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>WEEKDAY.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>WORDDATX.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>Swiss_French (FRS)</td>
<td>DATE.</td>
<td>NLDATE.</td>
<td>10</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>DATETIME.</td>
<td>NLDATM.</td>
<td>10</td>
<td>200</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>DOWNNAME.</td>
<td>NLDATEWN.</td>
<td>4</td>
<td>200</td>
<td>10</td>
</tr>
</tbody>
</table>
Currency Representation

Overview to Currency

Currency is the medium of exchange, which is specific to a country. SAS provides formats and informats for reading and writing currency.

U.S. Dollars

The DOLLARw.d formats and informats were first introduced to read and write American currency. DOLLARw.d

- uses the dollar sign ($) currency symbol to precede U.S. currency
- uses a comma (,) as the thousands separator and a dot (.) as the decimal separator

Example:

$12,345.00

DOLLARXw.d also writes currency with a leading dollar sign ($), but uses a dot (.) as the thousands separator and a comma (,) as the decimal separator. The reversal of the dot and comma for currency formatting is a convention used in many European countries.

Example:

$12,345,00

Because the dollar sign and some currency symbols used by other countries occupy the same code point location in a code page ('5B' on EBCDIC systems and '24' on
Localized Euros

The EUROw.d formats and informats were introduced to support the euro currency that was established by the European Monetary Union (EMU), which was formed in 1999. EUROw.d

- uses the euro (€) currency symbol to precede Euro currency data
- uses a comma (,) as the thousands separator and a dot (.) as the decimal separator

Example:

```sas
options locale=English_UnitedKingdom;
x=12345;
put x euro10.2;
run;
```

Output:

```
e12.345,00
```

Limitations of the EURO formats and informats are:

- the reversal of the dot and comma for currency formatting is not used by all European countries
- euros are limited only to members of the EMU
- the specific value of the locale is required

Customized Currency Representations

To create a customized currency representation, you can use the FORMAT procedure. The following example shows the creation of unique formats for the Australian dollar, the Swiss franc, and the British pound. For details about the FORMAT procedure, see Base SAS Procedures Guide.

**Example Code 7.1** SAS Code That Customizes Currency Representations

```sas
proc format;

  picture aud low<='0,000,000,009.00'
    (prefix='-AU$' mult=100)
  0--high='0,000,00,009.00' 
    (prefix='AU$' mult=100);

  picture sfr low<='0,000,000,009.00'
    (prefix='-SFr.' mult=100)
  0--high='0,000,00,009.00' 
    (prefix='-SFr.' mult=100);

  picture bpd low<='0,000,000,009.00'
    (prefix='-BPd.' mult=100)
```

Customizing currency representations offers flexibility, but requires a programming solution.

**Localized National and International Currency Representations**

The NLMNYw.d and NLMNYIw.d formats and informats were introduced to represent localized currency in two forms:

- **Localized national currency representation** reflects the customs and conventions of the locale. National formats are specified using the NLMNYw.d formats and informats. You must also use the LOCALE= option to specify the locale when using the NLMNYw.d formats and informats.

  Example:

  ```
  options locale=english_UnitedStates;
  data _null_;
  x=12345;
  put x nlmny15.2;
  run;
  
  Output:
  $12,345.00
  ```
Selected national currency representations follow:

Table 7.2  Localized National Currency Representations

<table>
<thead>
<tr>
<th>LOCALE=</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>$12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12 345,00 $</td>
</tr>
<tr>
<td>French_France</td>
<td>French euros</td>
<td>12 345,00 e</td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>Swiss francs</td>
<td>SFr. 12'345.00</td>
</tr>
<tr>
<td>German_Germany</td>
<td>German euros</td>
<td>12.345,00 e</td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>Luxembourg euros</td>
<td>12.345 e</td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>Spanish pesetas</td>
<td>12.345,00 e</td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>Venezuelan bolivars</td>
<td>Bs12.345,00</td>
</tr>
</tbody>
</table>

The localized renderings show the native customs for representing currency. For example, although these selected EMU countries might use the same euro currency, their depiction of the currency varies. Whereas French_France uses no thousands separator but uses a comma as a decimal separator, German_Germany and Spanish_Spain use a dot as a thousands separator and a comma as a decimal separator.

_Localized International currency representation_ conforms to ISO standard 4217. International forms are specified using the NLMNYIw.d formats and informats. International forms are commonly used to show a comparison of world currencies; for example, for airline ticket, trade, and stock market pricing. You must also use the LOCALE= option to specify the locale when using the NLMNYIw.d formats and informats. The letter “I,” which signifies “International,” is appended to the format and informat names.

Example:

```plaintext
options locale=english_UnitedStates;
data _null_;  
x=12345;  
put x nlmnyi15.2;  
run;
```

Output:

```
USD12,345.00
```

Selected international currency representations follow:

Table 7.3  International Currency Representations by Locale (ISO standard 4217)

<table>
<thead>
<tr>
<th>LOCALE=</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>English_UnitedStates</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>French_Canada</td>
<td>Canadian dollars</td>
<td>12,345.00 CAD</td>
</tr>
<tr>
<td>French_France</td>
<td>French euros</td>
<td>12,345.00 EUR</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>Luxembourg euros</td>
<td>12,345.00 EUR</td>
</tr>
</tbody>
</table>
The international renderings also reflect native customs for representing currency. For example, although all locales use a comma as the thousands separator and a dot as the decimal separator, they vary the placement of the ISO currency code. Whereas the EMU countries put the currency code after the currency, English_UnitedStates, German_Switzerland, and Spanish_Venezuela precede the currency with the ISO code.

For a complete list of the ISO standard 4217 currency codes, see [www.bsi-global.com/Technical%2BInformation/Publications/_Publications/tig90x.doc](http://www.bsi-global.com/Technical%2BInformation/Publications/_Publications/tig90x.doc).

A primary limitation of using localized national and international currency representations is their dependence on a value for the LOCALE= system option.

### Unique National and International Monetary Representations

The NLMNLISOw.d and NLMNIISOw.d formats and informats were introduced to uniquely represent each currency without having to also use the LOCALE= option to specify the locale. Each currency is specified by a unique ISO standard 4217 currency code.

**Unique national monetary representation**

is specified by the unique ISO currency code. National formats are specified using the NLMNLISOw.d formats and informats. In the following example, USD is the ISO currency code for American dollars.

**Note:** When using the NLMNLISOw.d formats and informats, you do not use the LOCALE= option to specify the locale. △

**Example:**

```
data _null_;  
x=12345;  
put x nlmnlusd15.2;  
run;  
```

**Output:**

```
US$12,345.00
```

Selected unique national currency representations follow:

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>National Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>USD$12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CA$12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>French euros</td>
<td>e12,345.00</td>
</tr>
</tbody>
</table>
A currency symbol or a currency code precedes most currencies. Also used are a comma as the thousands separator and a dot as the decimal separator.

Unique international monetary representation is specified by the unique ISO currency code. International formats are specified using the NLMNI/ISOw.d formats and informats. International forms are commonly used to show a comparison of world currencies; for example, for airline ticket, trade, and stock market pricing. The letter “I”, which signifies “International”, is appended to the format and informat names. In the following example, USD is the ISO currency code for American dollars.

Note: When using the NLMNI/ISOw.d formats and informats, you do not use the LOCALE= option to specify the locale.

Example:

```plaintext
data _null_;  
x=12345;  
put x nlmni15.2;  
run;
```

Output:

USD12,345.00

Selected international currency representations follow:

Table 7.5 International Currency Representations by ISO Currency Code

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Currency</th>
<th>International Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>USD</td>
<td>U.S. dollars</td>
<td>USD12,345.00</td>
</tr>
<tr>
<td>CAD</td>
<td>Canadian dollars</td>
<td>CAD12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>French euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>CHF</td>
<td>Swiss francs</td>
<td>CHF12,234.00</td>
</tr>
<tr>
<td>EUR</td>
<td>German euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Luxembourg euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>EUR</td>
<td>Spanish euros</td>
<td>EUR12,345.00</td>
</tr>
<tr>
<td>VEB</td>
<td>Venezuelan bolivars</td>
<td>Not found</td>
</tr>
</tbody>
</table>

The international renderings precede the currency with the appropriate ISO code. Also used are a comma as the thousands separator and a dot as the decimal separator.
Example: Representing Currency in National and International Formats

This SAS program uses the exchange rates for selected Asia-Pacific countries against the U.S. dollar. In the output, each country's currency is represented using a national and an international format.

Example Code 7.2  SAS Code That Formats National and International Currency Formats

data curr;
  input ex_date mmddyy. usd aud hkd jpy sgd 12.2;
datalines;
061704 1.00000 1.45349 7.79930 110.110 1.71900
;
proc print data=curr noobs label;
  var ex_date usd aud hkd jpy sgd;
  format ex_date mmddyy. usd nlmnlusd15.2 aud nlmnlaud15.2 hkd nlmnlhkd15.2
         jpy nlmnljpy15.2 sgd nlmnlsgd15.2;
  label ex_date='Date' usd='US' aud='Australia' hkd='Hong Kong'
         jpy='Japan' sgd='Singapore';
  title 'Exchange Rates for Selected Asian-Pacific Countries
       (Localized Currency Codes)';
proc print data=curr noobs label;
  var ex_date usd aud hkd jpy sgd;
  format ex_date mmddyy. usd nlmniusd15.2 aud nlmniaud15.2 hkd nlmnihkd15.2
         jpy nlmnjpy15.2 sgd nlmnsgd15.2;
  label ex_date='Date' usd='US' aud='Australia' hkd='Hong Kong'
         jpy='Japan' sgd='Singapore';
  title 'Exchange Rates for Selected Asian-Pacific Countries
       (International Currency Codes)';
run;

1 These exchange rates, which were effective June 17, 2004, are specified as data in the SAS program.
2 These NLMNLISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate national formats.
3 These NLMNIISO formats are applied to each of the numeric data items that are specified in the INPUT statement. These formats show currencies in the appropriate international formats.

Display 7.1  National and International Format Output
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 EURFRISO can be used to convert specific European currencies to an amount in euros. ISO represents an ISO standard 4214 currency code. For a complete list of the ISO standard 4217 currency codes, see www.bsi-global.com/Technical%2BInformation/Publications/_Publications/tig90x.doc.

Fixed Rates for Euro Conversion

Twenty-five European countries comprise the EMU (European Monetary Union). The conversion rates for 12 countries are fixed, and are incorporated into the EURFRISO and EURTOISO formats and into the EUROCURR function. The following table lists the currency codes and conversion rates for the specific currencies whose rates are fixed.

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Conversion Rate</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATS</td>
<td>13.7603</td>
<td>Austrian schilling</td>
</tr>
<tr>
<td>BEF</td>
<td>40.3399</td>
<td>Belgian franc</td>
</tr>
<tr>
<td>DEM</td>
<td>1.95583</td>
<td>Deutsche mark</td>
</tr>
<tr>
<td>ESP</td>
<td>166.386</td>
<td>Spanish peseta</td>
</tr>
<tr>
<td>EUR</td>
<td>1</td>
<td>Euro</td>
</tr>
<tr>
<td>FIM</td>
<td>5.94573</td>
<td>Finnish markka</td>
</tr>
<tr>
<td>FRF</td>
<td>6.55957</td>
<td>French franc</td>
</tr>
<tr>
<td>GRD</td>
<td>340.750</td>
<td>Greek drachma</td>
</tr>
<tr>
<td>IEP</td>
<td>0.787564</td>
<td>Irish pound</td>
</tr>
<tr>
<td>ITL</td>
<td>1936.27</td>
<td>Italian lira</td>
</tr>
<tr>
<td>LUF</td>
<td>40.3399</td>
<td>Luxembourg franc</td>
</tr>
<tr>
<td>NLG</td>
<td>2.20371</td>
<td>Dutch guilder</td>
</tr>
<tr>
<td>PTE</td>
<td>200.482</td>
<td>Portuguese escudo</td>
</tr>
</tbody>
</table>

Variable Rates for Euro Conversion

For 13 countries in the EMU, 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 currencies of the EMU countries whose rates fluctuate.

Table 7.7 Variable Rates for Euro Conversion

<table>
<thead>
<tr>
<th>ISO Currency Code</th>
<th>Conversion Rate</th>
<th>Currency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHF</td>
<td>1.60430</td>
<td>Swiss franc</td>
</tr>
<tr>
<td>CZK</td>
<td>34.8563</td>
<td>Czech koruna</td>
</tr>
<tr>
<td>DKK</td>
<td>7.49009</td>
<td>Danish krone</td>
</tr>
<tr>
<td>GBP</td>
<td>0.700132</td>
<td>British pound</td>
</tr>
<tr>
<td>HUF</td>
<td>260.325</td>
<td>Hungarian forint</td>
</tr>
<tr>
<td>NOK</td>
<td>9.19770</td>
<td>Norwegian krone</td>
</tr>
<tr>
<td>PLZ</td>
<td>4.2</td>
<td>Polish zloty</td>
</tr>
<tr>
<td>ROL</td>
<td>13.71</td>
<td>Romanian leu</td>
</tr>
<tr>
<td>RUR</td>
<td>19.7680</td>
<td>Russian ruble</td>
</tr>
<tr>
<td>SEK</td>
<td>9.36591</td>
<td>Swedish krona</td>
</tr>
<tr>
<td>SIT</td>
<td>191</td>
<td>Slovenian tolar</td>
</tr>
<tr>
<td>TRL</td>
<td>336.912</td>
<td>Turkish lira</td>
</tr>
<tr>
<td>YUD</td>
<td>13.0644</td>
<td>Yugoslavian dinar</td>
</tr>
</tbody>
</table>

Example: Converting between a European Currency and Euros

The following example shows the conversion from Belgian francs to euros. The EURFRBEF format divides the country’s currency amount by the exchange rate:

\[
\text{CurrencyAmount} / \text{ExchangeRate}
\]

12345 / 40.3399

Example Code 7.3 Example Code: Conversion from Belgian Francs to Euros

```plaintext
data _null_
x=12345 /*convert from Belgian francs to euros*/
put x eurfrbef15.2;
run;
```

Output:

e306,02

The following example shows the conversion of euros to Belgian francs. The EURTOBEF format multiplies euros by the target currency’s exchange rate:

\[
\text{EurosAmount} \times \text{ExchangeRate}
\]

12345 * 40.3399

Example Code 7.4 Example Code: Conversion from Euros to Belgian Francs

```plaintext
data _null_
x=12345; /*convert from euros to Belgian francs*/
```
put x eurtobef 15.2;  
run; 
Output:  
497996.07

Direct Conversion between European Currencies

The EUROCURR function uses the conversion rate tables to convert between currencies. For conversion between the currencies of two countries,
1 SAS converts the amount to euros.
   Note: SAS stores the intermediate value as precisely as the operating environment allows, and does not round the value. 
2 SAS converts the amount in euros to an amount in the target currency.

SourceCurrencyAmount — EurosAmount — TargetCurrencyAmount

BelgianFrancs — euros
12345 / 40.3399 = 306.02456 euros

Euros — FrenchFrancs
306.02456 * 6.55957 = 2007.3895 French francs

Example Code 7.5 Example Code: Conversion from Belgian Francs to French Francs

data _null_; 
x=eurocurr(12345,'bef','frf'); /*convert from Belgian francs to French francs*/
   put x=; 
run; 
Output: 
x=2007.389499
SAS converts Belgian francs to euros, and then euros to French francs.

Formats for NLS by Category

The following categories relate to NLS issues:

Table 7.8 Categories of NLS Formats

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

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 7.9 Summary of NLS Formats by Category**

<table>
<thead>
<tr>
<th>Category</th>
<th>Formats for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI text handling</td>
<td>“$BIDIw. Format” on page 73</td>
<td>Converts between a logically ordered string and a visually ordered string, by reversing the order of Hebrew and Arabic characters while preserving the order of Latin words and numbers.</td>
</tr>
<tr>
<td></td>
<td>“$LOGVS w. Format” on page 84</td>
<td>Processes a character string that is in left-to-right-logical order, and then writes the character string in visual order.</td>
</tr>
<tr>
<td></td>
<td>“$LOGVSR w. Format” on page 85</td>
<td>Processes a character string that is in right-to-left-logical order, and then writes the character string in visual order.</td>
</tr>
<tr>
<td></td>
<td>“$VSLOG w. Format” on page 229</td>
<td>Processes a character string that is in visual order, and then writes the character string in left-to-right logical order.</td>
</tr>
<tr>
<td></td>
<td>“$VSLOGR w. Format” on page 230</td>
<td>Processes a character string that is in visual order, and then writes the character string in right-to-left logical order.</td>
</tr>
<tr>
<td>Character</td>
<td>“$UCS2B w. Format” on page 206</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS2BE w. Format” on page 208</td>
<td>Processes a character string that is in big-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UCS2L w. Format” on page 209</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS2LE w. Format” on page 210</td>
<td>Processes a character string that is in little-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UCS2X w. Format” on page 211</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS2XE w. Format” on page 212</td>
<td>Processes a character string that is in native-endian, 16-bit, UCS2, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>“$UCS4Bw. Format” on page 213</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS4BEw. Format” on page 215</td>
<td>Processes a character string that is in big-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UCS4Lw. Format” on page 216</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS4LEw. Format” on page 217</td>
<td>Processes a character string that is in little-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UCS4Xw. Format” on page 218</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>“$UCS4XEw. Format” on page 220</td>
<td>Processes a character string that is in native-endian, 32-bit, UCS4, Unicode encoding, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UESCw. Format” on page 221</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode escape (UESC) representation.</td>
</tr>
<tr>
<td></td>
<td>“$UESCEw. Format” on page 222</td>
<td>Processes a character string that is in Unicode escape (UESC) representation, and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UNCRw. Format” on page 223</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in numeric character representation (NCR).</td>
</tr>
<tr>
<td></td>
<td>“$UNCREw. Format” on page 224</td>
<td>Processes a character string that is in numeric character representation (NCR), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UPARENw. Format” on page 225</td>
<td>Processes a character string that is encoded in the current SAS session, and then writes the character string in Unicode parenthesis (UPAREN) representation.</td>
</tr>
<tr>
<td></td>
<td>“$UPARENEw. Format” on page 226</td>
<td>Processes a character string that is in Unicode parenthesis (UPAREN), and then writes the character string in the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>“$UTF8Xw. Format” on page 228</td>
<td>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.</td>
</tr>
<tr>
<td>DBCS</td>
<td>“$KANJIw. Format” on page 82</td>
<td>Adds shift-code data to DBCS data.</td>
</tr>
<tr>
<td></td>
<td>“$KANJIXw. Format” on page 83</td>
<td>Removes shift-code data from DBCS data.</td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------------</td>
<td>------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Date and Time</td>
<td>&quot;HDATEw. Format&quot; on page 79</td>
<td>Writes date values in the form <code>yyyy mmmmm dd</code> where <code>dd</code> is the day-of-the-month, <code>mmm</code> represents the month's name in Hebrew, and <code>yyyy</code> is the year.</td>
</tr>
<tr>
<td></td>
<td>&quot;HEBDATEw. Format&quot; on page 80</td>
<td>Writes date values according to the Jewish calendar.</td>
</tr>
<tr>
<td></td>
<td>&quot;MINGUOW. Format&quot; on page 86</td>
<td>Writes date values as Taiwanese dates in the form <code>yyyymmdd</code></td>
</tr>
<tr>
<td></td>
<td>&quot;NENGOw. Format&quot; on page 87</td>
<td>Writes date values as Japanese dates in the form <code>e.yymmdd</code></td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEw. Format&quot; on page 91</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the date value as a date.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEEMDw. Format&quot; on page 92</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEEMNw. Format&quot; on page 93</td>
<td>Converts a SAS date value to the date value of the specified locale, and then writes the value as the name of the month.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEWw. Format&quot; on page 94</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEWNw. Format&quot; on page 95</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the date of the week.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEYMw. Format&quot; on page 96</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the year and the name of the month.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEYQw. Format&quot; on page 97</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the year and the quarter.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEYRW. Format&quot; on page 98</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the year.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEYWw. Format&quot; on page 98</td>
<td>Converts the SAS date value to the date value of the specified locale, and then writes the value as the year and the week.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDMw. Format&quot; on page 99</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATMAPw. Format&quot; on page 100</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATMDT. Format&quot; on page 101</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.</td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“NLDATMMDDw. Format” on page 102</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMMNw. Format” on page 103</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMTMw. Format” on page 104</td>
<td>Converts the time portion of a SAS datetime value to the time-of-day value of the specified locale, and then writes the value as a time of day.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMWNw. Format” on page 105</td>
<td>Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMWw. Format” on page 106</td>
<td>Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMYMw. Format” on page 107</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMYQw. Format” on page 107</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMYRw. Format” on page 108</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.</td>
<td></td>
</tr>
<tr>
<td>“NLDATMYWw. Format” on page 109</td>
<td>Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.</td>
<td></td>
</tr>
<tr>
<td>“NLTIMEw. Format” on page 204</td>
<td>Converts a SAS time value to the time value of the specified locale, and then writes the value as the time value.</td>
<td></td>
</tr>
<tr>
<td>“NLTIMAPw. Format” on page 205</td>
<td>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.</td>
<td></td>
</tr>
<tr>
<td>“WEEKUw. Format” on page 231</td>
<td>Writes a week number in decimal format by using the U algorithm.</td>
<td></td>
</tr>
<tr>
<td>“WEEKVw. Format” on page 233</td>
<td>Writes a week number in decimal format by using the V algorithm.</td>
<td></td>
</tr>
<tr>
<td>“WEEKWw. Format” on page 235</td>
<td>Writes a week number in decimal format by using the W algorithm.</td>
<td></td>
</tr>
<tr>
<td>“YYWEEKUw. Format” on page 236</td>
<td>Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.</td>
<td></td>
</tr>
<tr>
<td>“YYWEEKVw. Format” on page 238</td>
<td>Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.</td>
<td></td>
</tr>
<tr>
<td>“YYWEEKWw. Format” on page 239</td>
<td>Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Hebrew text handling</td>
<td>“$CPTDWw. Format” on page 75</td>
<td>Processes a character string that is in Hebrew text, encoded in IBM-PC (cp862), and then writes the character string in Windows Hebrew encoding (cp 1255).</td>
</tr>
<tr>
<td></td>
<td>“$CPTWDw. Format” on page 76</td>
<td>Processes a character string that is encoded in Windows (cp1255), and then writes the character string in Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>Numeric</td>
<td>“EUROw.d Format” on page 77</td>
<td>Writes numeric values with a leading euro symbol (E), a comma that separates every three digits, and a period that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>“EUROXw.d Format” on page 78</td>
<td>Writes numeric values with a leading euro symbol (E), a period that separates every three digits, and a comma that separates the decimal fraction.</td>
</tr>
<tr>
<td></td>
<td>“NLBESTw. Format” on page 89</td>
<td>Writes the best numerical notation based on the locale.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIAEDw.d Format” on page 110</td>
<td>Writes the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIAUDw.d Format” on page 111</td>
<td>Writes the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIBGNw.d Format” on page 112</td>
<td>Writes the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIBRILw.d Format” on page 113</td>
<td>Writes the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICADw.d Format” on page 114</td>
<td>Writes the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICHFw.d Format” on page 115</td>
<td>Writes the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICNYw.d Format” on page 116</td>
<td>Writes the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICZKw.d Format” on page 117</td>
<td>Writes the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIDKKw.d Format” on page 118</td>
<td>Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEEKhw.d Format” on page 119</td>
<td>Writes the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEGPw.d Format” on page 120</td>
<td>Writes the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEURw.d Format” on page 121</td>
<td>Writes the monetary format of the international expression for Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIGBPw.d Format” on page 122</td>
<td>Writes the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIHKDw.d Format” on page 123</td>
<td>Writes the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“NLMNIHRKw.d Format”</td>
<td>Writes the monetary format of the international expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>on page 124</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIHUFw.d Format”</td>
<td>Writes the monetary format of the international expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>on page 125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIIDRw.d Format”</td>
<td>Writes the monetary format of the international expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>on page 126</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIISw.d Format”</td>
<td>Writes the monetary format of the international expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>on page 127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIINRWd Format”</td>
<td>Writes the monetary format of the international expression for India.</td>
<td></td>
</tr>
<tr>
<td>on page 128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIJPyw.d Format”</td>
<td>Writes the monetary format of the international expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>on page 129</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIKRWd Format”</td>
<td>Writes the monetary format of the international expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>on page 130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIILTw.d Format”</td>
<td>Writes the monetary format of the international expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>on page 131</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIILVd Format”</td>
<td>Writes the monetary format of the international expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>on page 132</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIOMOp.w Format”</td>
<td>Writes the monetary format of the international expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>on page 133</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIOMXNw.d Format”</td>
<td>Writes the monetary format of the international expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>on page 134</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIOMYRw.d Format”</td>
<td>Writes the monetary format of the international expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>on page 135</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIOKw.d Format”</td>
<td>Writes the monetary format of the international expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>on page 136</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIINZDw.d Format”</td>
<td>Writes the monetary format of the international expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>on page 137</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIPLNw.d Format”</td>
<td>Writes the monetary format of the international expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>on page 138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIROLw.d Format”</td>
<td>Writes the monetary format of the international expression for Romania.</td>
<td></td>
</tr>
<tr>
<td>on page 139</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNIIRUBw.d Format”</td>
<td>Writes the monetary format of the international expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>on page 140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNISEKw.d Format”</td>
<td>Writes the monetary format of the international expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>on page 141</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNISGDw.d Format”</td>
<td>Writes the monetary format of the international expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>on page 142</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNISITw.d Format”</td>
<td>Writes the monetary format of the international expression for Slovenia.</td>
<td></td>
</tr>
<tr>
<td>on page 143</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNISKKw.d Format”</td>
<td>Writes the monetary format of the international expression for Slovakia.</td>
<td></td>
</tr>
<tr>
<td>on page 144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“NLMNITHBw.d Format”</td>
<td>Writes the monetary format of the international expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>on page 145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“NLMNITRYw.d Format” on page 146</td>
<td>Writes the monetary format of the international expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>“NLMNITW Dw.d Format” on page 147</td>
<td>Writes the monetary format of the international expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>“NLMNIUSD w.d Format” on page 148</td>
<td>Writes the monetary format of the international expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>“NLMNIZARw.d Format” on page 149</td>
<td>Writes the monetary format of the international expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLAE Dx.d Format” on page 150</td>
<td>Writes the monetary format of the local expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLAUDw.d Format” on page 151</td>
<td>Writes the monetary format of the local expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLBGNw.d Format” on page 152</td>
<td>Writes the monetary format of the local expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLBR Lu.d Format” on page 153</td>
<td>Writes the monetary format of the local expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCA Dw.d Format” on page 154</td>
<td>Writes the monetary format of the local expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCHFw.d Format” on page 155</td>
<td>Writes the monetary format of the local expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCHNYw.d Format” on page 156</td>
<td>Writes the monetary format of the local expression for China.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCKw.d Format” on page 157</td>
<td>Writes the monetary format of the local expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLDKKW.d Format” on page 158</td>
<td>Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLLEEKw.d Format” on page 159</td>
<td>Writes the monetary format of the local expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLEGPw.d Format” on page 160</td>
<td>Writes the monetary format of the local expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLEURw.d Format” on page 161</td>
<td>Writes the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLGBPw.d Format” on page 162</td>
<td>Writes the monetary format of the local expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHDw.d Format” on page 163</td>
<td>Writes the monetary format of the local expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHRKw.d Format” on page 164</td>
<td>Writes the monetary format of the local expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHFw.d Format” on page 165</td>
<td>Writes the monetary format of the local expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLIDRw.d Format” on page 166</td>
<td>Writes the monetary format of the local expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>“NLMNLILSw.d Format”</td>
<td>Writes the monetary format of the local expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>on page 167</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLINRw.d Format”</td>
<td>Writes the monetary format of the local expression for India.</td>
</tr>
<tr>
<td></td>
<td>on page 168</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLJPYw.d Format”</td>
<td>Writes the monetary format of the local expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>on page 169</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLKRWw.d Format”</td>
<td>Writes the monetary format of the local expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>on page 170</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLTLTw.d Format”</td>
<td>Writes the monetary format of the local expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>on page 171</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLLVLw.d Format”</td>
<td>Writes the monetary format of the local expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>on page 172</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLMOw.d Format”</td>
<td>Writes the monetary format of the local expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>on page 173</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLMXNw.d Format”</td>
<td>Writes the monetary format of the local expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>on page 174</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLMYRw.d Format”</td>
<td>Writes the monetary format of the local expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>on page 175</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLNOw.d Format”</td>
<td>Writes the monetary format of the local expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>on page 176</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLNZDw.d Format”</td>
<td>Writes the monetary format of the local expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>on page 177</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLPLNW.d Format”</td>
<td>Writes the monetary format of the local expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>on page 178</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLROLLw.d Format”</td>
<td>Writes the monetary format of the local expression for Romania.</td>
</tr>
<tr>
<td></td>
<td>on page 179</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLRUBw.d Format”</td>
<td>Writes the monetary format of the local expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>on page 180</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLSKW.d Format”</td>
<td>Writes the monetary format of the local expression for Slovakia.</td>
</tr>
<tr>
<td></td>
<td>on page 181</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLSGDw.d Format”</td>
<td>Writes the monetary format of the local expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>on page 182</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLSITw.d Format”</td>
<td>Writes the monetary format of the local expression for Slovenia.</td>
</tr>
<tr>
<td></td>
<td>on page 183</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLSKKW.d Format”</td>
<td>Writes the monetary format of the local expression for Turkey.</td>
</tr>
<tr>
<td></td>
<td>on page 184</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLTBw.d Format”</td>
<td>Writes the monetary format of the local expression for Taiwan.</td>
</tr>
<tr>
<td></td>
<td>on page 185</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“NLMNLTTRYw.d Format”</td>
<td>Writes the monetary format of the local expression for Puerto Rico and the United States.</td>
</tr>
<tr>
<td></td>
<td>on page 186</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Formats for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>&quot;NLMNLZARw.d Format&quot; on page 189</td>
<td>Writes the monetary format of the local expression for South Africa.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLMNYw.d Format&quot; on page 190</td>
<td>Writes the monetary format of the local expression in the specified locale using local currency.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLMNYIw.d Format&quot; on page 191</td>
<td>Writes the monetary format of the international expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLNUMw.d Format&quot; on page 193</td>
<td>Writes the numeric format of the local expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLNUMIw.d Format&quot; on page 194</td>
<td>Writes the numeric format of the international expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLPCTw.d Format&quot; on page 195</td>
<td>Writes percentage data of the local expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLPCTIw.d Format&quot; on page 197</td>
<td>Writes percentage data of the international expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLPCTNW.d Format&quot; on page 198</td>
<td>Produces percentages, using a minus sign for negative values.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLPCTPW.d Format&quot; on page 199</td>
<td>Writes locale-specific numeric values as percentages.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLVALUEw.d Format&quot; on page 199</td>
<td>Writes p-values of the local expression in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLSTRMONw.d Format&quot; on page 201</td>
<td>Writes a numeric value as a day-of-the-month in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLSTRQTRw.d Format&quot; on page 202</td>
<td>Writes a numeric value as the quarter-of-the-year in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLSTRWKW.d Format&quot; on page 203</td>
<td>Writes a numeric value as the day-of-the-week in the specified locale.</td>
</tr>
<tr>
<td></td>
<td>&quot;YENw.d Format&quot; on page 240</td>
<td>Writes numeric values with yen signs, commas, and decimal points.</td>
</tr>
</tbody>
</table>

**$\text{BIDIw. Format}$**

Converts between a logically ordered string and a visually ordered string, by reversing the order of Hebrew and Arabic characters while preserving the order of Latin words and numbers.

Category: BIDI text handling
Alignment: left

**Syntax**

$\text{$BIDIw.$}$
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 and Arabic text is stored in logical order. 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. SAS users can encounter Hebrew and Arabic 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 and Arabic text from a visual operating environment to a logical one. The $BIDI format is a format that reverses Hebrew and Arabic 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 and Arabic. Some mainframe operating environments might experience unsatisfactory results, because they use the OldCode Hebrew or Arabic encoding. There is a hotfix for this encoding on SAS Institute’s Web site: [http://support.sas.com/](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 or Arabic characters in the text string, the characters in the Hebrew or Arabic word are reversed and the position of the Hebrew or Arabic 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.

```plaintext
data;
  a = 'ףלט ינפנ מab c 123';
  b1 = put (a, $bidi20.);
  put b1=
  b2 = put (b, $bidi20.);
  put b2=
run;
```
The following lines are written to the SAS log:

\[ \text{\texttt{\textbackslash b1=123 \texttt{abc \& \textbackslash b2=123}}} \]

---

**$CPTDWw. 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**

\[ $CPTDWw. \]

**Syntax Description**

\[ w \]

specifies the width of the output field.

*Default:* 200

*Range:* 1–32000

---

**Comparisons**

The $CPTDWw. format performs processing that is the opposite of the $CPTWDw. format.

**Examples**

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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptdw3.;</td>
<td>123N</td>
</tr>
</tbody>
</table>

---

**See Also**

Formats:

“$CPTWDw. Format” on page 76
Informs:

“$CPTDWw. Informat” on page 318
“$CPTWDw. Informat” on page 319

$CPTWDw. 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

$CPTWDw.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 200

Range: 1–32000

Comparisons

The $CPTWDw. format performs processing that is the opposite of the $CPTDWw. format.

Examples

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

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $cptwd3.;</td>
<td>( ε\square, )</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$CPTDWw. Format” on page 75
EURO\textit{w}.\textit{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

\texttt{EURO}\textit{w}.\textit{d}

Syntax Description

\textit{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 77.

\textit{d}

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 \textit{w}

Comparisons

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

Examples

These examples use 1254.71 as the value of amount.

\begin{tabular}{ll}
\textbf{Statements} & \textbf{Results} \\
\hline
\texttt{put amount euro10.2;} & E1,254.71 \\
\texttt{put amount euro5.;} & 1,255 \\
\end{tabular}
EUROXw.d Format

EUROXw.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
EUROXw.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 79.

\( d \)

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 \( w \)

See Also
Formats:
“EUROXw.d Format” on page 78
Informats:
“EUROw.d Informat” on page 320
“EUROXw.d Informat” on page 322

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount euro9.2;</td>
<td>E1,254.71</td>
</tr>
<tr>
<td>put amount euro15.3;</td>
<td>E1,254.710</td>
</tr>
</tbody>
</table>

Statements Results

78 EUROXw.d Format  Chapter 7
Comparisons

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

Examples

These examples use 1254.71 as the value of amount.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put amount eurox10.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox5.;</td>
<td>1.255</td>
</tr>
<tr>
<td>put amount eurox9.2;</td>
<td>E1.254,71</td>
</tr>
<tr>
<td>put amount eurox15.3;</td>
<td>E1.254,710</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “EUROw.d Format” on page 77
Informats:
- “EUROw.d Informat” on page 320
- “EUROXw.d Informat” on page 322

HDATEN. Format

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

Category: Date and Time
Alignment: right

Syntax

HDATENw.

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 HDATEw. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day hdate9.;</td>
<td>03 י&quot;פ 16</td>
</tr>
<tr>
<td>put day hdate11.;</td>
<td>2003 י&quot;פ 16</td>
</tr>
<tr>
<td>put day hdate17.;</td>
<td>2003 י&quot;פ 16</td>
</tr>
</tbody>
</table>

See Also

Formats:

“HEBDATEw. Format” on page 80

HEBDATEw. Format

Writes date values according to the Jewish calendar.

Category: Date and Time
Alignment: right

Syntax

HEBDATEw.
Syntax Description

\( w \)

specifies the width of the output field.

Default: 16
Range: 7–24

Details

The Jewish calendar is a combined solar and lunar calendar. Years are counted from the creation of the world, which according to Jewish history, occurred 3760 years and three months before the commencement of the Christian. You must add 3761, beginning in the autumn of a specified year in the Gregorian calendar to calculate the Hebrew year.

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**: 
  \( \text{ינואר ה' תשסא'} \)
- **default**: 
  \( \text{י' תסה'} \)
- **short**: 
  \( \text{ית} \)

Examples

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put day hebdate13.;}</td>
<td>\text{ינואר 2/1/תשסא}'</td>
</tr>
<tr>
<td>\text{put day hebdate16.;}</td>
<td>\text{י' תסה}'</td>
</tr>
<tr>
<td>\text{put day hebdate24.;}</td>
<td>\text{ראשון ל' תדה-ב' תסה}'</td>
</tr>
</tbody>
</table>

See Also

Informats:

“HDATE\( w \). Format” on page 79
$\texttt{KANJ}w. \textit{Format}$

$\texttt{KANJ}w. \textit{Format}$ adds shift-code data to DBCS data.

**Category:** DBCS

**Alignment:** left

### Syntax

$\texttt{KANJ}w.$

### 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. The width must be equal to or greater than the length of the shift-code data.

  **Range:** The minimum width of the format is $2 + (\text{length of shift code used on the current DBCSTYPE= setting}) \times 2$.

### Details

The $\texttt{KANJ}w.$ format adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added.

The $\texttt{KANJ}w.$ format processes host-mainframe data, but $\texttt{KANJ}w.$ can be used on other platforms. If you use the $\texttt{KANJ}w.$ format on non-EBCDIC (non-modal encoding) hosts, the data does not change.

### See Also

**Formats:**

- “$\texttt{KANJ}Xw. \textit{Format}$” on page 83

**Informats:**

- “$\texttt{KANJ}w. \textit{Informat}$” on page 326
- “$\texttt{KANJ}Xw. \textit{Informat}$” on page 326

**System Options:**

- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458
$KANJIXw. Format

Removes shift-code data from DBCS data.

Category: DBCS
Alignment: left

Syntax
$KANJIXw.

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. The width must be equal to or greater than the length of the shift-code data.

Range: The minimum width of the format is 2.

Details
The $KANJIX format removes shift-code data from DBCS data. The input data length must be $ 2 + (SO/SI length)\times 2 $. The data must start with SO and end with SI, unless single-byte data is returned.

The $KANJIX$ format processes host mainframe data, but $KANJIX$ can be used on other platforms. If you use the $KANJIX$ format on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Formats:
“$KANJJIw. Format” on page 82
Informats:
“$KANJIIw. Informat” on page 326
“$KANJIXw. Informat” on page 326
System Options:
“DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458
$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 or Arabic 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 Hebrew input value of "_flight".

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>&quot;flight&quot;</td>
</tr>
</tbody>
</table>
The following example uses the Arabic input value of “ذات” computer.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12.;</td>
<td>ذات</td>
</tr>
</tbody>
</table>

### See Also

Formats:

“$LOGVSRw. Format” on page 85

Informats:

“$LOGVSRw. Informat” on page 328
“$LOGVSw. Informat” on page 327

### $LOGVSRw. 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

$LOGVSRw.

### Syntax Description

- **w** specifies the width of the output field.
  - **Default:** 200
  - **Range:** 1–32000

### Details

The $LOGVSRw. format is used when you store logical-ordered text on a visual server. The Hebrew or Arabic portion of the text is reversed if the $LOGVSw. format is not on the server.

### Comparisons

The $LOGVSRw. format performs processing that is opposite of the $LOGVSw. format.
Examples

The following example uses the Hebrew input value of “flight”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvsr12.;</td>
<td>flight יבשת</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “computer”.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvsr12.;</td>
<td>ذات computer</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$LOGVSdw. Format” on page 84

Informats:

“$LOGVSdw. Informat” on page 327
“$LOGVSRdw. Informat” on page 328

MINGUOW. Format

W rites date values as Taiwanese dates in the form yyyymmd.

Category: Date and Time

Alignment: left

Syntax

MINGUOW.

Syntax Description

\[ w \]

specifies the width of the output field.
Default: 8
Range: 1–10

Details
The MINGUOw. 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 before 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date minguo10.;</td>
<td>0082/01/01</td>
</tr>
<tr>
<td></td>
<td>0101/01/01</td>
</tr>
<tr>
<td></td>
<td>**********</td>
</tr>
</tbody>
</table>

See Also
Informats:
“MINGUOw. Informat” on page 330

NENGOw. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date nengo3.;</td>
<td>H14</td>
</tr>
<tr>
<td>put date nengo6.;</td>
<td>H14/01</td>
</tr>
<tr>
<td>put date nengo8.;</td>
<td>H.140102</td>
</tr>
<tr>
<td>put date nengo9.;</td>
<td>H14/01/02</td>
</tr>
<tr>
<td>put date nengo10.;</td>
<td>H.14/01/02</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NENGO\(w\). Informat” on page 331
NLBESTw. Format

Writes the best numerical notation based on the locale.

Category: Numeric
Alignment: right

Syntax

NLBESTw.

Syntax Description

w
specifies the width of the output field.

Default: 12

Tip: If you print numbers between 0 and .01 exclusively, then use a field width of at least 7 to avoid excessive rounding. If you print numbers between 0 and -.01 exclusively, then use a field width of at least 8.

Range: 1–32

Details

The NLBEST format writes the best numerical value based on the locale's decimal point and the sign mark's location. NLBEST is similar to the BEST format. For more information, see the BEST format in the *SAS Language Reference: Dictionary*.

Examples

The following code produces results based on the locale:

```sas
x=-1257000
put x nlbest6.;
put x nlbest3.;
put "=====";
x=-0.1
put x nlbest6.;
put x nlbest3.;
put "=====";
x=0.1
put x nlbest6.;
put x nlbest3.;
put "=====";
x=1257000
put x nlbest6.;
put x nlbest3.;
```
<table>
<thead>
<tr>
<th>Locales</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>locale=English_UnitedStates</td>
<td>-126E4</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>-.1</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>1.26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
<tr>
<td>locale=German_Germany</td>
<td>-126E4</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>-0.1</td>
</tr>
<tr>
<td></td>
<td>-.1</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>1,26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
<tr>
<td>locale=ar_BH</td>
<td>126E4-</td>
</tr>
<tr>
<td></td>
<td>***</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>0.1-</td>
</tr>
<tr>
<td></td>
<td>.1-</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td>======</td>
</tr>
<tr>
<td></td>
<td>1.26E6</td>
</tr>
<tr>
<td></td>
<td>1E6</td>
</tr>
</tbody>
</table>
**NLDATExw. 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**

NLDATExw.

**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**

NLDATExw. is similar to DATEw. and WORDDATEw. except that NLDATExw. is locale-specific.

---

**Examples**

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>put day nldate.;</td>
<td>February 24, 2003</td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>put day nldate.;</td>
<td>24. Februar 2003</td>
</tr>
</tbody>
</table>
See Also

Formats:

“NLDATEMNW. Format” on page 93
“NLDATEWNW. Format” on page 95

NLDATEMDw. Format

Converts the SAS date value to the date value of the specified locale, and then writes the value as
the name of the month and the day of the month.

Category: Date and Time
Alignment: left

Syntax

NLDATEMDw.

Syntax Description

w

specifies the width of the output field.
Default: 16
Range: 6-200

Examples

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put 1 nldatemd;</td>
<td>January 02</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLDATEYMW. Format” on page 96
NLDATEMNw. 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
NLDATEMNw.

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
NLDATEMNw. is similar to MONNAMEw. except that NLDATEMNw. is locale-specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>put month nldatefmt.;</td>
<td>February</td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>put month nldatefmt.;</td>
<td>February</td>
</tr>
</tbody>
</table>
See Also

Formats:
“NLDATEw. Format” on page 91
“NLDATEWw. Format” on page 94
“NLDATEWNw. Format” on page 95

NLDATEWw. 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

NLDATEWw.

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

NLDATEWw. is similar to WEEKDATEw. except that NLDATEWw. is locale specific.

Examples

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>Sun, Feb 24, 03</td>
</tr>
<tr>
<td>put date nldatew.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>So, 24. Feb 03</td>
</tr>
<tr>
<td>put date nldatew.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:

“NLDATEw. Format” on page 91
“NLDATEMNw. Format” on page 93
“NLDATEWNw. Format” on page 95

**NLDATEWNw. 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**

NLDATEWNw.

**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**

NLDATEWNw. is similar to DOWNAMEw. except that NLDATEWNw. is locale-specific.

**Examples**

These examples use the input value of 15760, which is the SAS date value that corresponds to February 24, 2003.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td>Sunday</td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>put date nldatewn.;</td>
<td>Sonntag</td>
</tr>
</tbody>
</table>
See Also

Formats:

“NLDATEw. Format” on page 91
“NLDATEMNw. Format” on page 93
“NLDATEWw. Format” on page 94

---

**NLDATEYMw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the name of the month.

**Category:** Date and Time

**Alignment:** left

---

### Syntax

**NLDATEYMw.**

---

### Syntax Description

**w**

specifies the width of the output field.

**Default:** 16

**Range:** 6–200

---

### Examples

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put 1 nldateym.;</code></td>
<td>January 1960</td>
</tr>
</tbody>
</table>

---

**See Also**

Formats:

“NLDATEMDw. Format” on page 92
**NLDATEYQw. Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the quarter.

**Category:** Date and Time  
**Alignment:** left

**Syntax**

NLDATEYQw.

**Syntax Description**

w  
specifies the width of the output field.  
**Default:** 16  
**Range:** 4–200

**Examples**

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+— NLDATEYQ min=4 default=16 max=200 +—</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>16 T3 08</td>
</tr>
<tr>
<td>dy=today();</td>
<td></td>
</tr>
<tr>
<td>dt=datetime();</td>
<td></td>
</tr>
<tr>
<td>put &quot;+— NLDATEYQ min=4 default=16 max=200 +—&quot;;</td>
<td></td>
</tr>
<tr>
<td>put '16' +5 dy nldateyq.;</td>
<td>4 ****</td>
</tr>
<tr>
<td>put '4' +5 dy nldateyq4.;</td>
<td>14 T3 08</td>
</tr>
<tr>
<td>put '14' +5 dy nldateyq14.;</td>
<td></td>
</tr>
<tr>
<td>put '32' +5 dy nldateyq32.;</td>
<td>32 3e trimestre 2008</td>
</tr>
<tr>
<td>put '200' +5 dy nldateyq200.;</td>
<td>200</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3e trimestre 2008</td>
</tr>
</tbody>
</table>
**NLDATEYRW Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year.

**Category:** Date and Time  
**Alignment:** left

**Syntax**

NLDATEYRW.

**Syntax Description**

w

specifies the width of the output field.

**Default:** 16  
**Range:** 2–200

**Examples**

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>dy=today();</td>
<td></td>
</tr>
<tr>
<td>dt=datetime();</td>
<td></td>
</tr>
<tr>
<td>put &quot;+— NLDATEYR min=2 default=16 max=200 —+&quot;;</td>
<td>+— NLDATEYR min=2 default=16 max=200 —+</td>
</tr>
<tr>
<td>put dy nldateyr.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dy nldateyr2.;</td>
<td>08</td>
</tr>
<tr>
<td>put dy nldateyr8.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dy nldateyr200.;</td>
<td>2008</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

---

**NLDATEYW Format**

Converts the SAS date value to the date value of the specified locale, and then writes the date value as the year and the week.

**Category:** Date and Time
Alignment: left

Syntax

NLDATEYWw.

Syntax Description

w
  specifies the width of the output field.
  Default:  16
  Range:   5–200

Examples

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td></td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>dy=today();</td>
<td></td>
</tr>
<tr>
<td>dt=datetime();</td>
<td></td>
</tr>
<tr>
<td>put &quot;+— NLDATEYW min=5 default=16 max=200 —+&quot;;</td>
<td>16 Week 33 2008</td>
</tr>
<tr>
<td>put '16' +5 dy nldateyw.;</td>
<td>5 *****</td>
</tr>
<tr>
<td>put '5' +5 dy nldateyw5.;</td>
<td>8 W33 08</td>
</tr>
<tr>
<td>put '8' +5 dy nldateyw8.;</td>
<td>32 Week 33 2008</td>
</tr>
<tr>
<td>put '32' +5 dy nldateyw32.;</td>
<td>200</td>
</tr>
<tr>
<td>put '200' +5 dy nldateyw200.;</td>
<td>Week 33 2008</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLDATMw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime.

Category: Date and Time
Alignment: left

Syntax

NLDATMw.
Syntax Description

\( w \)

specifies the width of the output field. If necessary, SAS abbreviates the datetime 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>24Feb03:12:39:43</td>
</tr>
<tr>
<td>put day nldatm.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>24. Februar 2003 12.39 Uhr</td>
</tr>
<tr>
<td>put day nldatm.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLDATMAP\( w \) Format” on page 100
“NLDATMTM\( w \) Format” on page 104
“NLDATMW\( w \) Format” on page 106

NLDATMAP\( w \) Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as a datetime with a.m. or p.m.

**Category:** Date and Time

**Alignment:** left
Syntax

NLDATMAPw.

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 NLDATMAPw. format is similar to DATEAMPMw. except that the NLDATMAPw. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>put event nldatmap.;</td>
<td>February 24, 2003 12:39:43 PM</td>
</tr>
<tr>
<td>options locale=Spanish_Mexico;</td>
<td></td>
</tr>
<tr>
<td>put event nldatmap.;</td>
<td>24 de febrero de 2003 12:39:43 PM</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLDATMw. Format” on page 99
“NLDATMTMw. Format” on page 104
“NLDATMWw. Format” on page 106

NLDATMDTw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month, day of the month and year.

Category: Date and Time
Syntax

NLDATMDT<sub>w</sub>.

Syntax Description

\(<w>\)

specifies the width of the output field.

Default: 20

Range: 10–200

Examples

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put $6400,nldatmdt.;</td>
<td>January 02, 1960</td>
</tr>
<tr>
<td>put$6400,dtdate.;</td>
<td>02JAN60</td>
</tr>
</tbody>
</table>

**NLDATMMD<sub>w</sub>. Format**

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month and the day of the month.

Category: Date and Time

Alignment: left

Syntax

NLDATMMD<sub>w</sub>.

Syntax Description

\(<w>\)

specifies the width of the output field.

Default: 16

Range: 6–200

Examples

This example uses the en_US locale option.
See Also

Formats:
“NLDATMYMw. Format” on page 107

NLDATMMNw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the name of the month.

Category: Date and Time
Alignment: left

Syntax
NLDATMMNw.

Syntax Description

\( w \)

specifies the width of the output field.
Default: 10
Range: 4–200

Examples
This example uses the en_US locale option.
Statements Results

data _null_
;;
dt = datetime();
dy = date();
put "+— NLDATEMN min=4 default=10
max=200 —+"
put dt nldatmmn.;
put dt nldatmmn.;
put dt nldatmmn.;
put dt nldatmmn10.;
put dt nldatmmn200.;
run;

NLDATMTMw. Format

Converts the time portion of a SAS datetime 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

NLDATMTMw.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 16
Range: 16–200

Comparisons

The NLDATMTMw. format is similar to the TODw. format except that the NLDATMTMw. 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.
NLDATMWNw. Format

Converts a SAS datetime value to the datetime value of the specified locale, and then writes the value as the day of the week.

Category: Date and Time
Alignment: left

Syntax

NLDATMWNw.

Syntax Description

w

specifies the width of the output field.

Default: 30
Range: 16–200

Examples

This example writes the SAS datetime value as a day of the week.

now = datetime() ;
put now nldatmwn. ;
NLDATMW<sub>w</sub>. Format

Converts SAS datetime values to the locale sensitive datetime string as the day of the week and the datetime.

Category: Date and Time
Alignment: left

Syntax

NLDATMW<sub>w</sub>.

Syntax Description

<sub>w</sub>

specifies the width of the output field. If necessary, SAS abbreviates the day of week and datetime to fit the format width.

Default: 30
Range: 16–200

Comparisons

The NLDATMW<sub>w</sub>. format is similar to the TWMDY<sub>w</sub>. format except that the NLDATMW<sub>w</sub>. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td>Sun, Feb 24, 2003 12:39:43</td>
</tr>
<tr>
<td>put event nldatmw.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td>So, 24. Feb 2003 12.39 Uhr</td>
</tr>
<tr>
<td>put event nldatmw.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLDATMW<sub>w</sub>. Format” on page 99
“NLDATMAP<sub>w</sub>. Format” on page 100
“NLDATMTMw. Format” on page 104

**NLDATMYMw. Format**

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the month.

**Category:** Date and Time

**Alignment:** left

**Syntax**

NLDATMYMw.

**Syntax Description**

w

specifies the width of the output field.

**Default:** 16

**Range:** 6–200

**Examples**

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put 86400 nldatym.;</td>
<td>January 1960</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLDATMMDw. Format” on page 102

**NLDATMYQw. Format**

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the quarter of the year.

**Category:** Date and Time
Alignment: left

Syntax
NLDATMYQw.

Syntax Description

w
specifies the width of the output field.

Default: 16
Range: 4–200

Examples
This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR; data <em>null</em>; dy=today(); dt=datetime(); put &quot;— NLDATMYQ min=4 default=16 max=200 —&quot;; put '16' +5 dt nldatmyq.; put '4' +5 dt nldatmyq4.; put '14' +5 dt nldatmyq14.; put '32' +5 dt nldatmyq32.; put '200' +5 dt nldatmyq200.; run;</td>
<td>+— NLDATMYQ min=4 default=16 max=200 —+ 16 T3 08 4 **** 14 T3 08 32 3e trimestre 2008 200 3e trimestre 2008</td>
</tr>
</tbody>
</table>

NLDATMYRw. Format

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year.

Category: Date and Time
Alignment: left

Syntax
NLDATMYRw.
Syntax Description

\( w \)

specifies the width of the output field.

**Default:** 16

**Range:** 2–200

Examples

This example uses the en_US locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+— NLDATMYR min=2 default=16 max=200 —+</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>2008</td>
</tr>
<tr>
<td>dy=today();</td>
<td>+— NLDATMYR min=2 default=16 max=200 —+</td>
</tr>
<tr>
<td>dt=datetime();</td>
<td></td>
</tr>
<tr>
<td>put &quot;+— NLDATMYR min=2 default=16 max=200 —+&quot;);</td>
<td></td>
</tr>
<tr>
<td>put dt nldatmyr.;</td>
<td>08</td>
</tr>
<tr>
<td>put dt nldatmyr2.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dt nldatmyr32.;</td>
<td>2008</td>
</tr>
<tr>
<td>put dt nldatmyr200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLDATMYWw. Format**

Converts the SAS datetime value to the datetime value of the specified locale, and then writes the value as the year and the name of the week.

**Category:** Date and Time

**Alignment:** left

Syntax

NLDATMYWw.

Syntax Description

\( w \)

specifies the width of the output field.

**Default:** 16
Range: 5–200

Examples

This example uses the fr_FR locale option.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=fr_FR;</td>
<td>+— NLDATMYW min=5 default=16 max=200 —+</td>
</tr>
<tr>
<td>data <em>null</em>;</td>
<td>16 Week 33 2008</td>
</tr>
<tr>
<td>dy=today();</td>
<td>5 *****</td>
</tr>
<tr>
<td>dt=datetime();</td>
<td>8 W33 08</td>
</tr>
<tr>
<td>put &quot;+— NLDATMYW min=5 default=16 max=200 —+&quot;;</td>
<td>32 Week 33 2008</td>
</tr>
<tr>
<td>put '16' +5 dt nldatmyw.;</td>
<td>200</td>
</tr>
<tr>
<td>put '5' +5 dt nldatmyw5.;</td>
<td>Week 33 2008</td>
</tr>
<tr>
<td>put '8' +5 dt nldatmyw8.;</td>
<td></td>
</tr>
<tr>
<td>put '32' +5 dt nldatmyw32.;</td>
<td></td>
</tr>
<tr>
<td>put '200' +5 dt nldatmyw200.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

NLMNIAEDw.d Format

Writes the monetary format of the international expression for the United Arab Emirates.

Category: Numeric
Alignment: left

Syntax

NLMNIAEDw.d

Syntax Description

w
specifies the width of the output field.
Default: 12
Range: 8–32

d
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default: 3
Range: 0–328
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniaed32.2);
```
```
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AED1,234,57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLAEĐx.d Format” on page 150

---

\textbf{NLMNIAUDw.d Format}

Writes the monetary format of the international expression for Australia.

Category: Numeric

Alignment: left

\textbf{Syntax}

\textbf{NLMNIAUD} \textit{w.d}

\textbf{Syntax Description}

\texttt{w} specifies the width of the output field.

\textbf{Default:} 9

\textbf{Range:} 1–32

\texttt{d} specifies to divide the number by $10^d$. If the data contains decimal points, the \texttt{d} value is ignored.

\textbf{Default:} 0

\textbf{Range:} 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmniaud32.2}); \\
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put } x=;</td>
<td>(AUD1,234,57)</td>
</tr>
<tr>
<td>\text{put } y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLAUDw,d Format” on page 151

NLMNIBGNw,d Format

Writes the monetary format of the international expression for Bulgaria.

Category: Numeric
Alignment: left

Syntax

NLMNIBGNw,d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnibgn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BGN1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLBGNw.d Format” on page 152

NLMNIBRLw.d Format

Writes the monetary format of the international expression for Brazil.

Category: numeric
Alignment: left

Syntax

NLMNIBRLw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnibrl32.2}); \\
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(BRL1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLBR\ld d Format” on page 153

NLMNICAD\ld d Format

Writes the monetary format of the international expression for Canada.

Category: Numeric
Alignment: left

Syntax

NLMNICAD\ld d

Syntax Description

\(w\)
specifies the width of the output field.
Default: 9
Range: 1–32

\(d\)
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmmnicad32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x;</code></td>
<td>(CAD1,234,57)</td>
</tr>
<tr>
<td><code>put y;</code></td>
<td>$1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
  “NLMNLCADw.d Format” on page 154

NLMNICHF\w.d Format

Writes the monetary format of the international expression for Liechtenstein and Switzerland.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNICHF\w.d}

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 9
- Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 0
- Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmichf32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(CHF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLCHFw.d Format” on page 155

NLMNICNYw.d Format

Writes the monetary format of the international expression for China.

Category: Numeric
Alignment: left

Syntax

NLMNICNYw.d

Syntax Description

\( w \)

specifies the width of the output field.
- Default: 9
- Range: 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.
- Default: 0
- Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmnicny}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>(CNY1,234.57)</td>
</tr>
<tr>
<td>put (y=);</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLCNYw.d Format” on page 156

NLMNICZKw.d Format

Writes the monetary format of the international expression for the Czech Republic.

Category: Numeric
Alignment: left

Syntax

NLMNICZKw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmniczk32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x=; )</td>
<td>(CZK1,234.57)</td>
</tr>
<tr>
<td>put ( y=; )</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLCZKw.d Format” on page 157

NLMNIDKKw.d Format

Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.

Category: Numeric
Alignment: left

Syntax

NLMNIDKKw.d

Syntax Description

\( w \)
specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnidkk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=</code>;</td>
<td>(DKK1,234.57)</td>
</tr>
<tr>
<td><code>put y=</code>;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLDKKw.d Format” on page 158

---

**NLMNIEEKw.d Format**

Writes the monetary format of the international expression for Estonia.

**Category:** Numeric

**Alignment:** left

---

**Syntax**

**NLMNIEEKw.d**

**Syntax Description**

- **w**  
  Specifies the width of the output field.
  
  **Default:** 9
  
  **Range:** 1–32

- **d**  
  Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  
  **Default:** 0
  
  **Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnieek32.2}); \\
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLEEKw.d Format” on page 159
Examples

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

```
x=put(-1234.56789,nlmmiegp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLEGp.w.d Format” on page 160

NLMNIEURw.d Format

Writes the monetary format of the international expression for Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

Category: Numeric
Alignment: left

Syntax

NLMNIEURw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to Locale=German_Germany.

\[
x=\text{put}(-1234.56789,\text{nlmnieur32.2});
\]
\[
y=\text{put}(-1234.56789,\text{nlmnieur32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-EUR1.234,57</td>
</tr>
<tr>
<td>put y=;</td>
<td>€1.234,57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLEUR\textit{w.d} Format” on page 161

\textbf{NLMNGBP\textit{w.d} Format}

 Writes the monetary format of the international expression for the United Kingdom.

\textbf{Category:} Numeric

\textbf{Alignment:} left

\textbf{Syntax}

\texttt{NLMNGBP\textit{w.d}}

\textbf{Syntax Description}

\textit{w}

specifies the width of the output field.

\textbf{Default:} 9

\textbf{Range:} 1–32

\textit{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

\textbf{Default:} 0

\textbf{Range:} 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnigbp32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(GBP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLGBPw.d Format” on page 162

NLMNIHKDw.d Format

Writes the monetary format of the international expression for Hong Kong.

Category: Numeric
Alignment: left

Syntax

NLMNIHKDw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnihkd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x;</td>
<td>(HKD1,234.57)</td>
</tr>
<tr>
<td>put y;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:  
“NLMNLHKDw.d Format” on page 163

**NLMNIHRKw.d Format**

Writes the monetary format of the international expression for Croatia.

**Category:** Numeric  
**Alignment:** left

**Syntax**

`NLMNIHRKw.d`

**Syntax Description**

`w`

specifies the width of the output field.  
**Default:** 9  
**Range:** 1–32

`d`

specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.  
**Default:** 0  
**Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English United States.

```plaintext
x=put(-1234.56789, nlmnihrk32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HRK1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLHRKw.d Format” on page 164

**NLMNIHUFw.d Format**

Writes the monetary format of the international expression for Hungary.

**Category:** Numeric

**Alignment:** left

**Syntax**

**NLMNIHUFw.d**

**Syntax Description**

\[w\]

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

\[d\]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

**Default:** 0

**Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmnihuf32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HUF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLHUFw.d Format” on page 165

NLMNIIDRw.d Format

Writes the monetary format of the international expression for Indonesia.

Category: Numeric

Alignment: left

Syntax

NLMNIIDRw.d

Syntax Description

`w`

specifies the width of the output field.

Default: 9

Range: 1–32

`d`

specifies to divide the number by $10^d$. If the data contains decimal points, the `d` value is ignored.

Default: 0

Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmniidr32.2}); \\
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(IDR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:  
“NLMNLIDRw.d Format” on page 166

NLMNIILSw.d Format

Writes the monetary format of the international expression for Israel.

Category: Numeric  
Alignment: left

Syntax

NLMNIILSw.d

Syntax Description

\(w\)  
specifies the width of the output field.  
Default: 9  
Range: 1–32

\(d\)  
specifies to divide the number by 10^d. If the data contains decimal points, the \(d\) value is ignored.  
Default: 0  
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmniils32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLILS\text{w.d Format}” on page 167

NLMNIINR\text{w.d Format}

Writes the monetary format of the international expression for India.

Category: Numeric
Alignment: left

Syntax

NLMNIINR\text{w.d}

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmniinr32.2});
\]
\[
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(INR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNINR\(w.d\) Format” on page 168

---

NLMNJPY\(w.d\) Format

Writes the monetary format of the international expression for Japan.

Category: Numeric
Alignment: left

Syntax

NLMNJPY\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

- Default: 9
- Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 0
- Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmnijpy32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(JPY1,234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLJPY\(w,d\) Format” on page 169

NLMNIKRW\(w,d\) Format

Writes the monetary format of the international expression for South Korea.

Category: Numeric
Alignment: left

Syntax

NLMNIKRW\(w,d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789,\text{nlmnikrw}32.2);
y=\text{put}(-1234.56789,\text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put } x=;</td>
<td>(KRW1,234.57)</td>
</tr>
<tr>
<td>\text{put } y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNKRWw.d Format” on page 170

\textbf{NLMNILTLw.d Format}

Writes the monetary format of the international expression for Lithuania.

Category: Numeric

Alignment: left

Syntax

\texttt{NLMNILTLw.d}

Syntax Description

\(w\)

specifies the width of the output field.

Default: \(9\)

Range: \(1–32\)

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: \(0\)

Range: \(0–31\)
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789,\text{nlmnlvl}132.2);
y=\text{put}(-1234.56789,\text{dollar}32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LTL1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLLTLw.d Format” on page 171

NLMNILVLw.d Format

Writes the monetary format of the international expression for Latvia.

Category: Numeric
Alignment: left

Syntax

NLMNILVLw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnlvl32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=';</td>
<td>(LVL1, 234.57)</td>
</tr>
<tr>
<td>put y=';</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLLVLw.d Format” on page 172

NLMNIMOPw.d Format

Writes the monetary format of the international expression for Macau.

Category: Numeric
Alignment: left

Syntax

NLMNIMOPw.d

Syntax Description

`w`
specifies the width of the output field.

**Default:** 9
**Range:** 1–32

`d`
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

**Default:** 0
**Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlnmimop32.2}); \]
\[ y = \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(MOP1,234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLMOP\( w,d \) Format” on page 173

NLMNIMXN\( w,d \) Format

Writes the monetary format of the international expression for Mexico.

Category: Numeric

Alignment: left

Syntax

NLMNIMXN\( w,d \)

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9

Range: 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default: 0

Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmmimxn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MXN1,234,57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLMXNw.d Format” on page 174

NLMNIMYRw.d Format

Writes the monetary format of the international expression for Malaysia.

Category: Numeric
Alignment: left

Syntax

NLMNIMYRw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnmmyr32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MYR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLMYRw.d Format” on page 175

NLMNINOKw.d Format

Writes the monetary format of the international expression for Norway.

Category: Numeric
Alignment: left

Syntax

NLMNINOKw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmninok32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NOK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLNOKw.d Format” on page 176

NLMNINZDw.d Format

Writes the monetary format of the international expression for New Zealand.

Category: Numeric
Alignment: left

Syntax

NLMNINZDw.d

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 9
- Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- Default: 0
- Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmnlnc32.2});
\]
\[
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLNZDw.d Format” on page 177

NLMNLNPlNw.d Format

W 

writes the monetary format of the international expression for Poland.

Category: Numeric

Alignment: left

Syntax

NLMNLNPlNw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9

Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnipln32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(PLN1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLPLNw.d Format” on page 178

NLMNIROLw.d Format

Writes the monetary format of the international expression for Romania.

Category: Numeric
Alignment: left

Syntax

NLMNIROLw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnirol32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ROL1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLROLw.d Format” on page 179

---

NLMNIRUBw.d Format

Writes the monetary format of the international expression for Russia.

Category: Numeric

Alignment: left

Syntax

NLMNIRUBw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9

Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnirub32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RUB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLRUBw.d Format” on page 180

NLMNISEKw.d Format

Writes the monetary format of the international expression for Sweden.

Category: Numeric
Alignment: left

Syntax

NLMNISEKw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnisek32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLSEKw.d Format” on page 181

NLMNISGDw.d Format

Writes the monetary format of the international expression for Singapore.

Category: Numeric
Alignment: left

Syntax

NLMNISGDw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the Locale= system option is set to English_UnitedStates.

\[ x = \text{put}(-1234.56789, \text{nlmnisgd}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(SGD1, 234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLSGDw.d Format” on page 182

**NLMNISITw.d Format**

Writes the monetary format of the international expression for Slovenia.

**Category:** Numeric

**Alignment:** left

**Syntax**

\[ \text{NLMNISITw}.d \]

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

\( d \)

specifies to divide the number by 10\(^d\). If the data contains decimal points, the \( d \) value is ignored.

**Default:** 0

**Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[x = \text{put}(-1234.56789,\text{nlmnsit32.2});\]
\[y = \text{put}(-1234.56789,\text{dollar32.2});\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SIT1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLSITw.d Format” on page 183

NLMNISKKw.d Format

Writes the monetary format of the international expression for Slovakia.

Category: Numeric

Alignment: left

Syntax

NLMNISKKw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9

Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniskk32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SKK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLSKKw.d Format” on page 184

---

**NLMNITHBw.d Format**

Writes the monetary format of the international expression for Thailand.

Category: Numeric

Alignment: left

---

**Syntax**

**NLMNITHBw.d**

**Syntax Description**

**w**

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

**d**

specifies to divide the number by \(10^d\). If the data contains decimal points, the d value is ignored.

- **Default:** 0
- **Range:** 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[ x=\text{put}(-1234.56789,\text{nlmnithb32.2}); \]
\[ y=\text{put}(-1234.56789,\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLTBw.d Format” on page 185

NLMNTRYw.d Format

Writes the monetary format of the international expression for Turkey.

Category: Numeric
Alignment: left

Syntax

NLMNTRYw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789, nlmnlitry32.2);
y=put(-1234.56789, dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TRY1, 234, 57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNTRYw.d Format” on page 186

NLMNITWDw.d Format

Writes the monetary format of the international expression for Taiwan.

Category: Numeric
Alignment: left

Syntax

NLMNITWDw.d

Syntax Description

\( w \)

specifies the width of the output field.

- Default: 9
- Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

- Default: 0
- Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x = \text{put}(-1234.56789, \text{nlmniUSD32.2});
y = \text{put}(-1234.56789, \text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(TWD1, 234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLTWDw.d Format” on page 187

NLMNIUSDw.d Format

Writes the monetary format of the international expression for Puerto Rico and the United States.

Category: Numeric
Alignment: left

Syntax

NLMNIUSDw.d

Syntax Description

\[w\]

specifies the width of the output field.

Default: 9
Range: 1–32

\[d\]

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default: 0
Range: 0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmniusd32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLUSDw.d Format” on page 188

NLMNIZARw.d Format

Writes the monetary format of the international expression for South Africa.

Category:    Numeric
Alignment:   left

Syntax

NLMNIZARw.d

Syntax Description

w
specifies the width of the output field.
Default:  9
Range:  1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.
Default:  0
Range:  0–31
Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmnizar32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ZAR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLZARw.d Format” on page 189

NLMNLAE Dx.d Format

Writes the monetary format of the local expression for the United Arab Emirates.

Category: Numeric

Alignment: left

Syntax

NLMNLAE Dx.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9

Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31
Formats for NLS

NLMNLAUD\(w.d\) Format

Writes the monetary format of the local expression for Australia.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNLAUD}\(w.d\)

Syntax Description

\(w\)

- specifies the width of the output field.
- Default: 9
- Range: 1–32

\(d\)

- specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
- Default: 0
- Range: 0–31

Examples

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.

\begin{verbatim}
x=put(-1234.56789,nlmnlaed32.2);
y=put(-1234.56789,dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>\texttt{(AED1,234.57)}</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNIAED\(w.d\) Format” on page 110
x=put(-1234.56789,nlmlaud32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(AU$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIAUDw.d Format” on page 111

NLMNLBGNw.d Format

Writes the monetary format of the local expression for Bulgaria.

Category: Numeric
Alignment: left

Syntax

NLMNLBGNw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
\[ x = \text{put}(-1234.56789, \	ext{nlmnlbgn}32.2); \]
\[ y = \text{put}(-1234.56789, \text{dollar}32.2); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(BGN1, 234.57)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIBGN\(w,d\) Format” on page 112

---

**NLMNLBR\(Lw,d\) Format**

Writes the monetary format of the local expression for Brazil.

Category: Numeric

Alignment: left

**Syntax**

NLMNLBR\(Lw,d\)

**Syntax Description**

\(w\)

\(w\) specifies the width of the output field.

Default: \(9\)

Range: \(1–32\)

\(d\)

\(d\) optionally specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: \(0\)

Range: \(0–31\)

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLCADw.d Format

Writes the monetary format of the local expression for Canada.

Category: Numeric
Alignment: left

Syntax

NLMNLCADw.d

Syntax Description

* w
  specifies the width of the output field.
  Default: 9
  Range: 1–32

* d
  specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  Default: 0
  Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLCHF\(w.d\) Format

Writes the monetary format of the local expression for Liechtenstein and Switzerland.

Category: Numeric
Alignment: left

Syntax

NLMNLCHF\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
\[ x= \text{put}(-1234.56789, \text{nlmnlchf32.2}); \]
\[ y= \text{put}(-1234.56789, \text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>put x=</code>;</td>
<td>SFr.1,234.57</td>
</tr>
<tr>
<td><code>put y=</code>;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNICHF\(w.d\) Format” on page 115

---

**NLMNLCNY\(w.d\) Format**

Writes the monetary format of the local expression for China.

**Syntax**

\[ \text{NLMNLCNY}\(w.d\) \]

**Syntax Description**

\(w\)

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

- **Default:** 0
- **Range:** 0–31

**Examples**

In the following example, the \(\text{LOCALE=}\) system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlcny32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RMB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNICNYw.d Format” on page 116

NLMNLCZKw.d Format

Writes the monetary format of the local expression for the Czech Republic.

Category: Numeric
Alignment: left

Syntax

NLMNLCZKw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmniczk32.2);
y=put(-1234.56789,dollar32.2);

Statements Results

---------+--------+
put x=; (CZK1,234.57)
put y=; $-1,234.57

See Also

Formats:
“NLMNLCZKw.d Format” on page 117

NLMNLHKKw.d Format

Writes the monetary format of the local expression for Denmark, Faroe Island, and Greenland.

Category: Numeric

Alignment: left

Syntax

NLMNLHKKw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9

Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmlndkk32.2);
y=put(-1234.56789,dollar32.2);

Statements    Results
---+----+----+
put x=;       (kr1,234.57)
put y=;       $-1,234.57

See Also

Formats:
“NLMNIDKK\(w.d\) Format” on page 118

NLMNLEEK\(w.d\) Format

Writes the monetary format of the local expression for Estonia.
Category:    Numeric
Alignment:   left

Syntax

NLMNLEEK\(w.d\)

Syntax Description

\(w\)

specifies the width of the output field.
Default:  9
Range:    1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default:  0
Range:    0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnleek32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EEK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIEEKw.d Format” on page 119

**NLMNLEGPo.w.d Format**

Writes the monetary format of the local expression for Egypt.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNLEGPo.w.d

**Syntax Description**

- **w**
  - specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlegp32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(EGP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIEGPw.d Format” on page 120

NLMNLEURw.d Format

Writes the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

Category: Numeric
Alignment: left

Syntax

NLMNLEURw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to German_Germany.
x=put(-1234.56789,nlmnieur32.2);
y=put(-1234.56789,nlmnieur32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-EUR1.234,57</td>
</tr>
<tr>
<td>put y=;</td>
<td>€1.234,57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNIEUR\w.d Format” on page 121

NLMNLGBP\w.d Format

Writes the monetary format of the local expression for the United Kingdom.

Category: Numeric
Alignment: left

Syntax

NLMNLGBP\w.d

Syntax Description

\w
specifies the width of the output field.
Default: 9
Range: 1–32

\d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
FORMATS FOR NLS

FORMATS FOR NLS /52

FORMAT 163

x=put(-1234.56789,nlmnlgbp32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(£1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIGBPw.d Format” on page 122

NLMNLHKDw.d Format

WRITES THE MONETARY FORMAT OF THE LOCAL EXPRESSION FOR HONG KONG.

Category: Numeric
Alignment: left

Syntax

NLMNLHKDw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlhkd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HK$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIIKDW.d Format” on page 123

NLMNIIKDW.d Format

Writes the monetary format of the local expression for Croatia.

Category: Numeric
Alignment: left

Syntax

NLMNIIKDW.w.d

Syntax Description

w
 specifies the width of the output field.
Default: 9
Range: 1–32

d
 specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlhrk32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HRK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIHRKw.d Format” on page 124

**NLMNLHUFw.d Format**

Writes the monetary format of the local expression for Hungary.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNLHUFw.d

**Syntax Description**

**w**

specifies the width of the output field.

Default: 9

Range: 1–32

**d**

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 0

Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlhuf32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(HUF1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“NLMNIHUFw.d Format” on page 125

---

**NLMNLIDRw.d Format**

Writes the monetary format of the local expression for Indonesia.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNLIDRw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlidr32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(IDR1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“NLMNIIDRw.d Format” on page 126

**NLMNLILSw.d Format**

Writes the monetary format of the local expression for Israel.

Category: Numeric
Alignment: left

**Syntax**

NLMNLILSw.d

**Syntax Description**

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
\begin{verbatim}
x=put(-1234.56789, lmlnls32.2);
y=put(-1234.56789, dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ILS1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNIIILS\textit{w.d} Format” on page 127

\textbf{NLMNIN\textit{Rw.d} Format}

\textit{NLMNINRw.d} Format writes the monetary format of the local expression for India.

\textbf{Category:} Numeric

\textbf{Alignment:} left

\textbf{Syntax}

\textit{NLMNIN\textit{Rw.d}}

\textbf{Syntax Description}

\textit{w}

specifies the width of the output field.

\textbf{Default:} 9

\textbf{Range:} 1–32

\textit{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

\textbf{Default:} 0

\textbf{Range:} 0–31

\textbf{Examples}

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
 Formats for NLS

 NLMNLJPYw.d Format

 Writes the monetary format of the local expression for Japan.

 Category: Numeric
 Alignment: left

 Syntax

 NLMNLJPYw.d

 Syntax Description

 w specifies the width of the output field.
 Default: 9
 Range: 1–32

d specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
 Default: 0
 Range: 0–31

 Examples

 In the following example, the LOCALE= system option is set to English_UnitedStates.

 x=put(-1234.56789,nlmlinr32.2);
y=put(-1234.56789,dollar32.2);

 Statements Results

 put x=; (INR1,234.57)
 put y=; $-1,234.57

 See Also

 Formats:
 “NLMNIINRw.d Format” on page 128
x=put(-1234.56789, nlmnljpy32.2);
y=put(-1234.56789, dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(JPY1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNlJPyw.d Format” on page 129

NLMNLKRWw.d Format

Writes the monetary format of the local expression for South Korea.

Category: Numeric
Alignment: left

Syntax

NLMNLKRWw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

D
specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English UnitedStates.
x=put(-1234.56789,nlmnlkrw32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(KRW1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIKRWw.d Format” on page 130

---

**NLMNLTLw.d Format**

Writes the monetary format of the local expression for Lithuania.

- **Category:** Numeric
- **Alignment:** left

**Syntax**

NLMNLTLw.d

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Examples**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`. 
\texttt{x=put(-1234.56789, nlmnllvl32.2)}; \\
\texttt{y=put(-1234.56789, dollar32.2)};

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{put x=;}</td>
<td>(LTL1,234.57)</td>
</tr>
<tr>
<td>\texttt{put y=;}</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Formats: 
“NLMNLTLL\textit{w.d Format}” on page 131

**NLMNLLVL\textit{w.d Format}**

Writes the monetary format of the local expression for Latvia.

Category: Numeric

Alignment: left

**Syntax**

\texttt{NLMNLLVL\textit{w.d}}

**Syntax Description**

\textit{w}

specifies the width of the output field.

Default: 9

Range: 1–32

\textit{d}

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31

**Examples**

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
x=put(-1234.56789,nlmnlvl32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(LVL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNILVLw.d Format” on page 132

NLMNLMOPlw.d Format

Writers the monetary format of the local expression for Macau.

Category: Numeric
Alignment: left

Syntax

NLMNLMOPlw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by 10^d. If the data contains decimal points, the \( d \) value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlmop32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MOP1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIMOPw.d Format” on page 133

---

**NLMNLMXNw.d Format**

Writes the monetary format of the local expression for Mexico.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNLMXNw.d

**Syntax Description**

*\( w \)*

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

*\( d \)*

specifies to divide the number by 10^d. If the data contains decimal points, the \( d \) value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
 Formats for NLS △ NLMNLMYRw.d Format 175

```
x=put(-1234.56789,nlmnlmxn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(MXN1,234,57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIMXNw.d Format” on page 134

NLMNLMYRw.d Format

Writes the monetary format of the local expression for Malaysia.

Category: Numeric
Alignment: left

Syntax

NLMNLMYRw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by 10^d. If the data contains decimal points, the \( d \) value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
\begin{verbatim}
x=put(-1234.56789, nlmnlmyr32.2);
y=put(-1234.56789, dollar32.2);
\end{verbatim}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>((R1, 234.57))</td>
</tr>
<tr>
<td>put (y=);</td>
<td>($-1, 234.57)</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIMYRw.d Format” on page 135

---

**NLMNLOKw.d Format**

Writes the monetary format of the local expression for Norway.

**Category:** Numeric  
**Alignment:** left

**Syntax**

\texttt{NLMNLOKw.d}

**Syntax Description**

\(w\)

specifies the width of the output field.  
\textbf{Default:} 9  
\textbf{Range:} 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.  
\textbf{Default:} 0  
\textbf{Range:} 0–31

**Examples**

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
x=put(-1234.56789,nlmnlko32.2);  
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNINOKw.d Format” on page 136

NLMNLNZDw.d Format

Writes the monetary format of the local expression for New Zealand.

Category: Numeric

Alignment: left

Syntax

NLMNLNZDw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9

Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlnlnzd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NZ$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNINZDw.d Format” on page 137

NLMNLPLNW.d Format

Writes the monetary format of the local expression for Poland.

Category: Numeric
Alignment: left

Syntax

NLMNLPLNW.d

Syntax Description

\( w \)

specifies the width of the output field.
Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLROL\textit{w.d} Format

Writes the monetary format of the local expression for Romania.

\textbf{Category:} Numeric
\textbf{Alignment:} left

\begin{description}
\item \textit{Syntax}
NLMNLROL\textit{w.d}
\end{description}

\begin{description}
\item \textit{Syntax Description}

\textit{w} specifies the width of the output field.
\textbf{Default:} 9
\textbf{Range:} 1–32

\textit{d} specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.
\textbf{Default:} 0
\textbf{Range:} 0–31
\end{description}

\begin{description}
\item \textbf{Examples}
In the following example, the LOCALE= system option is set to English_UnitedStates.
\end{description}
x=put(-1234.56789,nlmlrol32.2);
y=put(-1234.56789,$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(ROL1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLNIROLw.d Format” on page 139

NLMNLRIUB_w.d Format

Writes the monetary format of the local expression for Russia.

Category: Numeric
Alignment: left

Syntax

NLMNLRIUB_w.d

Syntax Description

\( w \)

- specifies the width of the output field.
  - Default: 9
  - Range: 1–32

\( d \)

- specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - Default: 0
  - Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlrub32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(RUB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIRUBw.d Format” on page 140

NLMNLSEKw.d Format

Writes the monetary format of the local expression for Sweden.

Category: Numeric
Alignment: left

Syntax

NLMNLSEKw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLSGD \(w.d\) Format

Writes the monetary format of the local expression for Singapore.

**Category:** Numeric

**Alignment:** left

**Syntax**

\[\text{NLMNLSGD} \; w.d\]

**Syntax Description**

\(w\)

- specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

\(d\)

- specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Examples**

In the following example, the `LOCALE=` system option is set to `English_UnitedStates`.

\[
x=\text{put}(-1234.56789,nlmnlsek32.2);
y=\text{put}(-1234.56789,dollar32.2);
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(kr1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNISEK \(w.d\) Format” on page 141
x=put(-1234.56789,nlmnlsgd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SG$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNISGDw.d Format” on page 142

---

**NLMNLSITw.d Format**

Writes the monetary format of the local expression for Slovenia.

Category: Numeric

Alignment: left

**Syntax**

NLMNLSITw.d

**Syntax Description**

w

specifies the width of the output field.

Default: 9

Range: 1–32

d

specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.

Default: 0

Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnsit32.2);   
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SIT1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMVISITw.d Format” on page 143

NLMNLSKKw.d Format

Writes the monetary format of the local expression for Slovakia.

Category: Numeric
Alignment: left

Syntax

NLMNLSKKw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default: 9
Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmlskk32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(SKK1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“NLMNISKKw.d Format” on page 144

**NLMNLTHBw.d Format**

Writes the monetary format of the local expression for Thailand.

Category: Numeric

Alignment: left

**Syntax**

NLMNLTHBw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \( d \) value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlthb32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(THB1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNITHBw,d Format” on page 145

NLMLTRYw,d Format

Writes the monetary format of the local expression for Turkey.

Category: Numeric
Alignment: left

Syntax

NLMLTRYw,d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLTW\(w.d\) Format

Writes the monetary format of the local expression for Taiwan.

Category: Numeric
Alignment: left

Syntax

\[\text{NLMNLTW}w.d\]

Syntax Description

\(w\)

specifies the width of the output field.
- Default: 9
- Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
- Default: 0
- Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
x=\text{put}(-1234.56789,\text{nlnltry}32.2);\]
\[
y=\text{put}(-1234.56789,\text{dollar}32.2);\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put (x=);</td>
<td>(TRY1,234,57)</td>
</tr>
<tr>
<td>put (y=);</td>
<td>$-1,234.57$</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNITRY\(w.d\) Format” on page 146
x=put(-1234.56789,nlmnltwd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(NT$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNITWDw.d Format” on page 147

NLMNLUSDw.d Format

Writes the monetary format of the local expression for Puerto Rico and the United States.

Category: Numeric
Alignment: Left

Syntax

NLMNLUSDw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlusd32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(US$1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIUSDw.d Format” on page 148

NLMNLZARw.d Format

Writes the monetary format of the local expression for South Africa.

Category: Numeric
Alignment: left

Syntax

NLMNLZARw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=put(-1234.56789,nlmnlzar32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>(R1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIZARw.d Format” on page 149

NLMNYw.d Format

Writes the monetary format of the local expression in the specified locale using local currency.

Category: Numeric
Alignment: left

Syntax

NLMNYw.d

Syntax Description

\( w \)

specifies the width of the output field.

- **Default:** 9
- **Range:** 1–32

\( d \)

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

- **Default:** 0
- **Range:** 0–31

Details

The NLMNYw.d format reads integer binary (fixed-point) values, including negative values that are represented in two’s-complement notation. The NLMNYw.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 NLMNYw.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 NLMNYw.d and NLMNYIw.d formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYIw.d format uses three-letter international currency codes, such as USD, while NLMNYw.d format uses local currency symbols, such as $.

The NLMNYw.d format is similar to the DOLLARw.d format except that the NLMNYw.d format is locale-specific.

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=put(-1234.56789,nlmny32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>put y=;</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNYIw.d Format” on page 191

Informats:

“NLMNYw.d Informat” on page 414

“NLMNYIw.d Informat” on page 415

NLMNYIw.d Format

Writes the monetary format of the international expression in the specified locale.

Category: Numeric

Alignment: left

Syntax

NLMNYIw.d
**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 9  
**Range:** 1–32

\( d \)

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

**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 international currency code, and locale-dependent thousands and decimal separators. The position of international currency code is also locale dependent.

**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\( w \).\( d \) and NLMNYI\( w \).\( d \) formats write the monetary format with locale-dependent thousands and decimal separators. However, the NLMNYI\( w \).\( d \) format uses three-letter international currency codes, such as USD, while NLMNY\( w \).\( d \) format uses local currency symbols, such as $.

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{put}(-1234.56789, \text{nlmnyi32.2}); \\
y &= \text{put}(-1234.56789, \text{nlmny32.2}); \\
z &= \text{put}(-1234.56789, \text{dollar32.2});
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put} ( x= );</td>
<td>(USD1,234.57)</td>
</tr>
<tr>
<td>\text{put} ( y= );</td>
<td>($1,234.57)</td>
</tr>
<tr>
<td>\text{put} ( z= );</td>
<td>$-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNY\( w \).\( d \) Format” on page 190
Informs:
“NLMNYw.d Informat” on page 414
“NLMNYIw.d Informat” on page 415

**NLNUMw.d Format**

Writes the numeric format of the local expression in the specified locale.

**Category:** Numeric

**Alignment:** left

---

**Syntax**

NLNUMw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 6

**Range:** 1–32

\( d \)

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 NLNUMw.d format reads integer binary (fixed-point) values, including negative values that are represented in two’s-complement notation. The NLNUMw.d format writes numeric values by using the thousands separator and the decimal separator that is used by the locale.

**Comparisons**

The NLNUMw.d format writes the numeric value with locale-dependent thousand and decimal separators. The NLNUMIw.d format writes the numeric value with a comma (,) as thousand separator and a period (.) as a decimal separator.

If the \(w\) or \(d\) values are not large enough to generate a formatted number, the NLNUMw.d format uses an algorithm that prints the thousands-separator characters whenever possible, even if some decimal precision is lost.

**Examples**

\[ x=\text{put}(-1234356.7891, \text{nlnum32.2}); \]
### Statements

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates; put x=;</td>
<td>-1,234,356.79</td>
</tr>
<tr>
<td>options LOCALE=German_Germany; put x=;</td>
<td>-1.234.356,79</td>
</tr>
</tbody>
</table>

### See Also

Formats:
- “NLNUMIw.d Format” on page 194

Informats:
- “NLNUMIw.d Informat” on page 417
- “NLNUMIw.d Informat” on page 418

### NLNUMIw.d Format

Writes the numeric format of the international expression in the specified locale.

**Category:** Numeric  
**Alignment:** left

#### Syntax

**NLNUMIw.d**

#### Syntax Description

- **w**
  - Specifies the width of the output field.  
  - **Default:** 6  
  - **Range:** 1–32

- **d**
  - 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 **NLNUMIw.d** format reads integer binary (fixed-point) values, including negative values that are represented in two’s-complement notation. The **NLNUMIw.d** format
writes numeric values by using a comma (,) as thousands separator and a period (.) as a
decimal separator for all locales.

**Comparisons**

The NLNUMI\(w.d\) format writes the numeric data of the international expression in the
specified locale. The NLNUMI\(w.d\) format writes the numeric value with a comma (,) as
thousand separator and a period (.) as a decimal separator.

If the \(w\) or \(d\) values are not large enough to generate a formatted number, the
NLNUMI\(w.d\) format uses an algorithm that prints the thousands-separator characters
whenever possible, even if some decimal precision is lost.

**Examples**

\[
x = \text{put}(-1234356.7891, \text{nlnumi32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>---+---1-----+</td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234,356.79</td>
</tr>
<tr>
<td>options LOCALE=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>-1,234,356.79</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLNUMI\(w.d\) Format” on page 193

Informats:

“NLNUMI\(w.d\) Informat” on page 417

“NLNUMI\(w.d\) Informat” on page 418

**NLPCT\(w.d\) Format**

**Syntax**

\[
\text{NLPCT}w.d
\]
Syntax Description

\(w\)

specifies the width of the output field.

**Default:** 6

**Range:** 4–32

\(d\)

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 NLPCT\(w.d\) format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTI\(w.d\) format writes the percentage value with a comma (,) as thousand separator and a period (.) as a decimal separator.

The NLPCT\(w.d\) format is similar to the PERCENT\(w.d\) format except the NLPCT\(w.d\) format is locale-specific.

Examples

\[
x=\text{put}(-12.3456789,\text{nlpct32.2});
\]
\[
y=\text{put}(-12.3456789,\text{nlpcti32.2});
\]
\[
z=\text{put}(-12.3456789,\text{percent32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options ( \text{LOCALE=}\text{English_UnitedStates} ); put (x=);</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put (y=);</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put (z=);</td>
<td>(1234.57%)</td>
</tr>
<tr>
<td>options ( \text{LOCALE=}\text{German_Germany} ); put (x=);</td>
<td>-1.234,57%</td>
</tr>
<tr>
<td>put (y=);</td>
<td>-1.234.57%</td>
</tr>
<tr>
<td>put (z=);</td>
<td>(1234.57%)</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLPCTI\(w.d\) Format” on page 197

Informats:

“NLPCT\(w.d\) Informat” on page 419
“NLPCTIw.d Informat” on page 420

NLPCTIw.d Format

Writes percentage data of the international expression in the specified locale.

Category: Numeric
Alignment: left

Syntax

NLPCTIw.d

Syntax Description

\( w \)
specifies the width of the output field.
Default: 6
Range: 4–32

\( d \)
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 NLPCTIw.d format writes percentage data of the international expression in the specified locale. The NLPCTIw.d format writes the percentage value with locale-dependent thousand and decimal separators. The NLPCTIw.d format writes the percentage value with a comma (,) as thousand separator and a period (.) as a decimal separator.

The NLPCTIw.d format is similar to the PERCENTw.d format except the NLPCTIw.d format is locale-specific.

Examples

In the following example, the LOCALE= system option is set to English.UnitedStates.

\[ \begin{align*}
  x &= \text{put}(-12.3456789, \text{nlpcti}32.2); \\
  y &= \text{put}(-12.3456789, \text{percent}32.2);
\end{align*} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-1,234.57%</td>
</tr>
<tr>
<td>put y=;</td>
<td>(1234.57)</td>
</tr>
</tbody>
</table>
See Also

Formats:
“NLPCTnw.d Format” on page 195
Informat:
“NLPCTnw.d Informat” on page 419
“NLPCTIw.d Informat” on page 420

NLPCTnw.d Format

Produces percentages, using a minus sign for negative values.

Category: Numeric
Alignment: right

Syntax

NLPCTnw.d

Syntax Description

\( w \)

specifies the width of the output field.
Default: 6
Range: 4–32
Tip: The width of the output field must account for the minus sign (−), the percent sign (%), and a trailing blank, whether the number is negative or positive.

\( d \)

specifies the number of digits to the right of the decimal point in the numeric value.
This argument is optional.
Range: 0–31
Requirement: must be less than \( w \)

Details

The NLPCTnw.d format multiplies negative values by 100, adds a minus sign to the beginning of the value, and adds a percent sign (%) to the end of the formatted value.

Examples

\( x=-0.02; \)

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctn6;</td>
<td>x=-2%</td>
</tr>
<tr>
<td>put x percentn6;</td>
<td>x=-2%</td>
</tr>
</tbody>
</table>
**NLPCTP\textit{w}.d Format**

Writes locale-specific numeric values as percentages.

**Category:** Numeric

**Alignment:** right

**Syntax**

\texttt{NLPCTP\textit{w}.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

**Default** 6

**Range** 4–32

**Tip:** The width of the output field must account for the percent sign (\%).

\textit{d}

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

This argument is optional. The thousand separator and decimal symbol for the NLPCTP format is locale-specific.

**Range:** 0–31

**Requirement:** must be less than \textit{w}

**Details**

The NLPCTP\textit{w}.d format multiplies values by 100, formats them, and adds a percent sign (%) to the end of the formatted value. The NLPCTP\textit{w}.d format is similar to the PERCENT\textit{w}.d format except that the thousand separator and decimal symbol for the NLPCTP\textit{w}.d format is locale-specific.

**Examples**

\texttt{x=-0.02;}

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x nlpctp6.;</td>
<td>-2%</td>
</tr>
<tr>
<td>put x percent6.;</td>
<td>( 2%)</td>
</tr>
</tbody>
</table>

**NLPVALUE\textit{w}.d Format**

Writes p-values of the local expression in the specified locale.
Syntax

NLPVALUEw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 6
Range: 3–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal separators, the \(d\) value is ignored.

Default: 4
Range: 1–30

Examples

This example uses the german_Germany locale option.

Statements:

```sas
options locale=german_germany;
data _null_;
   put "+---- nlpvalue min=3 default=6 max=32 ----+";
x=0.1248;
   put x= +5 x pvalue. +5 x nlpvalue.;
   put x= +5 x pvalue3.1 +5 x nlpvalue3.1;
   put x= +5 x pvalue20.2 +5 x nlpvalue20.2;
   put x= +5 x pvalue32.3 +5 x nlpvalue32.3;
run;
```

Results:

```sas
+---- nlpvalue min=3 default=6 max=32 ----+
 x=0.1248 0.1248 0,1248
 x=0.1248 0.1 0,1
 x=0.1248 0.12 0,12
 x=0.1248 0.125 0,125
```

See Also

Format:

PVALUEw.d in SAS Language Reference: Dictionary
NLSTRMONw.d Format

Writes a numeric value as a day-of-the-month in the specified locale.

Category: Numeric
Alignment: left

Syntax

NLSTRMONw.d

Syntax Description

\( w \)

specifies the width of the output field

Default: 20
Range: 200-1

\( d \)

specifies the following:
00000001: write abbreviated form.
00000010: write capitalized form.

Default: 0
Range: 0-3

Details

The NLSTRMONw.d format writes a SAS value, 1–12 as the name-of-the-month in the specified locale. The following examples use the English_UnitedStates locale.

1 = the first month (January)
2 = the second month (February)
3 = the third month (March)
4 = the fourth month (April)
5 = the fifth month (May)
6 = the sixth month (June)
7 = the seventh month (July)
8 = the eight month (August)
9 = the ninth month (September)
10 = the tenth month (October)
11 = the eleventh month (November)
12 = the twelfth month (December)

Examples

This example uses the English_UnitedStates session encoding.
Statements Results
Data _null_;
monnum = 1 ; /* January=1, December=12 */
put monnum NLSTRMON20. ; January
put monnum NLSTRMON20.1; /* decimal .1 specified use abbreviation. */
put monnum NLSTRMON20.2; January
put monnum NLSTRMON20.3; Jan
run;

**NLSTRQTRw.d Format**

Writes a numeric value as the quarter-of-the-year in the specified locale.

Category: Numeric
Alignment: left

**Syntax**

**NLSTRQTRw.d**

**Syntax Description**

**w**

specifies the width of the output field

- **Default:** 20
- **Range:** 1–200

**d**

specifies the following:

- 00000001: write abbreviated form.
- 00000010: write capitalized form.

- **Default:** 3
- **Range:** 0–3

**Details**

The NLSTRQTRw.d format writes a SAS value, 1–4 as the name-of-the-quarter for the year in the specified locale. The following examples use the English_UnitedStates locale.

1 = 1st quarter
2 = 2nd quarter
3 = 3rd quarter
4 = 4th quarter

**Examples**

This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>; qtrnum = 1 ; /* January=1, December=12 */ put qtrnum NLSTRQTR20. ;</td>
<td>1st quarter</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.1; /* decimal .1 specified use abbreviation. */</td>
<td>1st quarter</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.2;</td>
<td>1ST QUARTER</td>
</tr>
<tr>
<td>put qtrnum NLSTRQTR20.3; run;</td>
<td>1ST QUARTER</td>
</tr>
</tbody>
</table>

**NLSTRWKw.d Format**

Writes a numeric value as the day-of-the-week in the specified locale.

- **Category:** Numeric
- **Alignment:** left

**Syntax**

NLSTRWKw.d

**Syntax Description**

- **w**
  - specifies the width of the output field
  - **Default:** 20
  - **Range:** 1–200

- **d**
  - specifies the following:
    - 00000001: write abbreviated form.
    - 00000010: write capitalized form.
  - **Default:** 0
  - **Range:** 0–3

**Details**

The NLSTRQTRw.d format writes a SAS value, 1–7 as the name-of-the-week in the specified locale. The following examples use the English_UnitedStates locale.
1 = First day-of-week (Monday)
2 = Second day-of-week (Tuesday)
3 = Third day-of-week (Wednesday)
4 = Fourth day-of-week (Thursday)
5 = Fifth day-of-week (Friday)
6 = Sixth day-of-week (Saturday)
7 = Seventh day-of-week (Sunday)

**Examples**

This example uses the English_UnitedStates session encoding.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data <em>null</em>;</td>
<td></td>
</tr>
<tr>
<td>wknum = 1 ; /* Sunday=1, Saturday=7 */</td>
<td></td>
</tr>
<tr>
<td>put wknum NLSTRWK20. ;</td>
<td>Sunday</td>
</tr>
<tr>
<td>put wknum NLSTRWK20.1; /* decimal .1 specified use abbreviation. */</td>
<td>Sun</td>
</tr>
<tr>
<td>put wknum NLSTRWK20.2;</td>
<td>SUNDAY</td>
</tr>
<tr>
<td>put wknum NLSTRWK20.3;</td>
<td>SUN</td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>

**NLTIMEw. 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**

NLTIMEw.

**Syntax Description**

w

specifies the width of the input field.

**Default:** 20

**Range:** 10–200
Comparisons

The NLTIMEw. format is similar to the TIMEw. format except that the NLTIMEw. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put time nltime.;</td>
<td>4:24:43</td>
</tr>
<tr>
<td>options locale=German_Germany; put time nltime.;</td>
<td>16.24</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLTIMAPw. Format” on page 205

NLTIMAPw. 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

NLTIMAPw.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 10
Range: 4–200
Comparisons

The NLTIMAPw. format is similar to the TIMEAMPMw. format except that the NLTIMAPw. 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; put time nltimap.;</td>
<td>4:24:43 PM</td>
</tr>
<tr>
<td>options locale=German_Germany; put time nltimap.;</td>
<td>16.24 Uhr</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLTIMEw. Format” on page 204

$UCS2Bw. 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

$UCS2Bw.

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 $UCS2Bw$. 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 $UCS2Bw$. format performs processing that is the opposite of the $UCS2BEw$. format.

Examples
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 大;</td>
<td></td>
</tr>
<tr>
<td>put x $ucs2b2.;</td>
<td>‘5927’x (binary)</td>
</tr>
</tbody>
</table>

See Also
Formats:

“$UCS2Lw. Format” on page 209
“$UCS2Xw. Format” on page 211
“$UTF8Xw. Format” on page 228
“$UCS2BEw. Format” on page 208
Informats:
“$UCS2BEw. Informat” on page 426
“$UCS2BEw. Informat” on page 427
“$UCS2Lw. Informat” on page 428
“$UCS2Xw. Informat” on page 430
“$UTF8Xw. Informat” on page 444

$UCS2BEw. 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
$UCS2BEw.

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 $UCS2BEw. 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 $UCS2BEw. format performs processing that is the opposite of the $UCS2Bw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = ’592700410042’x;</td>
<td>AB</td>
</tr>
<tr>
<td>put x $ucs2be4.;</td>
<td>AB</td>
</tr>
</tbody>
</table>
$UCS2Lw. 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

$UCS2Lw.

Syntax Description

\( w \)

\( w \) 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 $UCS2Lw. 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 $UCS2Lw. format performs processing that is the opposite of the $UCS2LEw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.
Statements Result

| x = 'x'; 
| put x $ucs212.; | '2759'x (binary) |

See Also

Formats:
- "$UCS2Bw. Format" on page 206
- "$UCS2LEw. Format" on page 210
- "$UCS2Xw. Format" on page 211
- "$UTF8Xw. Format" on page 228

Informants:
- "$UCS2Bw. Informat" on page 426
- "$UCS2Lw. Informat" on page 428
- "$UCS2LEw. Informat" on page 429
- "$UCS2Xw. Informat" on page 430
- "$UTF8Xw. Informat" on page 444

$UCS2LEw. 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
$UCS2LEw.

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 $UCS2LEw. 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 $UCS2LEw. format performs processing that is the opposite of the $UCS2Lw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = ’275941004200’x;</td>
<td></td>
</tr>
<tr>
<td>put x $ucs2le4.;</td>
<td>ƒab</td>
</tr>
</tbody>
</table>

See Also

Formats:
“$UCS2Lw. Format” on page 209

Informats:
“$UCS2Lw. Informat” on page 428
“$UCS2LEw. Informat” on page 429

$UCS2Xw. 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

$UCS2Xw.

Syntax Description

\( w \)

\( w \) 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 $UCS2Xw. 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 $UCS2Xw. format performs processing that is the opposite of the $UCS2XEw. format. If you are exchanging data within the same operating environment, use the $UCS2Xw. format. If you are exchanging data with a different operating environment, use the $UCS2Bw. format or $UCS2Lw. format.

Example
This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '弋'; put x $ucs2x2.;</td>
<td>'5927'x (binary) or '2759'x (little endian)</td>
</tr>
</tbody>
</table>

See Also
Formats:
- "$UCS2Bw. Format" on page 206
- "$UCS2XEw. Format" on page 212
- "$UCS2Lw. Format" on page 209
- "$UTF8Xw. Format" on page 228
Informats:
- "$UCS2Bw. Informat" on page 426
- "$UCS2Lw. Informat" on page 428
- "$UCS2Xw. Informat" on page 430
- "$UCS2XEw. Informat" on page 431
- "$UTF8Xw. Informat" on page 444

$UCS2XEw. 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

\$UCS2XEw.

Syntax Description

\textit{w}

\textit{w} specifies the width of the output field. Specify enough width to accommodate the 16-bit size of the Unicode characters.

\textbf{Default:} 8

\textbf{Range:} 1–32000

Details

The \$UCS2XEw. 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 \$UCS2XEw. format performs processing that is the opposite of the \$UCS2Xw. format.

Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
</table>
| x = `e5a4a7`x; /* Japanese 'י' in UTF8 */;
| put x $utf8xe10.; | פ |

See Also

Formats:

“\$UCS2Xw. Format” on page 211

Informats:

“\$UCS2Xw. Informat” on page 430

“\$UCS2XEw. Informat” on page 431

\textbf{\$UCS4Bw. 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

$UCS4Bw.

Syntax Description

\texttt{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 $UCS4Bw. 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 $UCS4Bw. format performs processing that is the opposite of the $UCS4BEw. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '＋';</td>
<td>+</td>
</tr>
<tr>
<td>put x $ucs4b4.;</td>
<td>'00005927'x (binary)</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UCS2Lw. Format” on page 209
“$UCS2Xw. Format” on page 211
“$UCS4BEw. Format” on page 215
“$UCS4Lw. Format” on page 216
“$UCS4Xw. Format” on page 218
“$UTF8Xw. Format” on page 228
Informats:
“$UCS2Bw. Informat” on page 426
“$UCS2Lw. Informat” on page 428
“$UCS2Xw. Informat” on page 430
“$UCS4Bw. Informat” on page 432
“$UCS4Lw. Informat” on page 433
“$UCS4Xw. Informat” on page 434
“$UTF8Xw. Informat” on page 444

$UCS4BEw. 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
$UCS4BEw.

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 $UCS4BEw. 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 $UCS4BEw. format performs processing that is the opposite of the $UCS4Bw. format.

Example
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.
Statements Result

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = '000059270000004100000042'x;</td>
<td></td>
</tr>
<tr>
<td>put x $ucs4be4.;</td>
<td>$AB</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UCS4Bw. Format” on page 213

Informats:

“$UCS4Bw. Informat” on page 432

$UCS4Lw. 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

$UCS4Lw.

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 $UCS4Lw. 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 $UCS4Lw. format performs processing that is the opposite of the $UCS4LEw. format.
Examples
This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x = 'مثال' ; ) put x $ucs4l4.;</td>
<td>‘27590000’x (binary)</td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$UCS2Bw. Format” on page 206
- “$UCS2Xw. Format” on page 211
- “$UCS4Bw. Format” on page 213
- “$UCS4LEw. Format” on page 217
- “$UCS4Xw. Format” on page 218
- “$UTF8Xw. Format” on page 228

Informats:
- “$UCS2Bw. Informat” on page 426
- “$UCS2Lw. Informat” on page 428
- “$UCS2Xw. Informat” on page 430
- “$UCS4Bw. Informat” on page 432
- “$UCS4Lw. Informat” on page 433
- “$UCS4Xw. Informat” on page 434
- “$UTF8Xw. Informat” on page 444

$UCS4LEw. 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

$UCS4LEw.

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 `$UCS4LEw$. 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 `$UCS4LEw$. format performs processing that is the opposite of the `$UCS4Lw$. format.

**Example**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x = '275900004100000042000000'x;$</td>
<td>$AB$</td>
</tr>
<tr>
<td>put $x$ $uc4le4.$;</td>
<td>$AB$</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“$UCS4Lw. Format” on page 216

Informats:

“$UCS4Lw. Informat” on page 433

---

**$UCS4Xw. 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**

$UCS4Xw.$
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 \$UCS4Xw. 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 \$UCS4Xw. format performs processing that is the opposite of the \$UCS4XEw.
format. If you are exchanging data within the same operating environment, use the
\$UCS4Xw. format. If you are exchanging data with a different operating environment,
use the \$UCS4Bw. format or \$UCS4Lw. format.

Example

This example uses the Japanese Shift_JIS session encoding, which is supported under
the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>( x = '¥'; )</td>
<td></td>
</tr>
<tr>
<td>put x $ucs4x4.;</td>
<td>‘00005927’x (binary) or ’27590000’x (little endian)</td>
</tr>
</tbody>
</table>

See Also

Formats:

“\$UCS2Lw. Format” on page 209
“\$UCS4XEw. Format” on page 220
“\$UCS2Xw. Format” on page 211
“\$UCS4Bw. Format” on page 213
“\$UCS4Lw. Format” on page 216
“\$UTF8Xw. Format” on page 228

Informats:

“\$UCS2Bw. Informat” on page 426
“\$UCS2Lw. Informat” on page 428
“\$UCS2Xw. Informat” on page 430
“\$UCS4Bw. Informat” on page 432
**$UCS4XEw. 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

$UCS4XEw.

### Syntax Description

$\textit{w}$

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 $UCS4XEw.$ 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 $UCS4XEw.$ format performs processing that is the opposite of the $UCS4Xw.$ format.

### Example

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '275900004100000042000000'x; **AB (little endian)</td>
<td></td>
</tr>
<tr>
<td>put x $ucs4be4.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:

“$UCS4Xw. Format” on page 218

Informats:

“$UCS4Xw. Informat” on page 434

$UESCw. 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

$UESCw.

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. $UESCw. can be nested.

Comparisons

The $UESCw. format performs processing that is opposite of the $UESCEw. format.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.
Statements | Results
---|---
---+----1----+----2

```plaintext
x='';
y='u5927'
z='uu5927';
put x = $uesc10. ;  ¥u5927
put y = $uesc10. ;  ¥uu5927
put z = $uesc10. ;  ¥uuu5927
```

### See Also

**Formats:**
- “$UESCEw. Format” on page 222

**Informats:**
- “$UESCw. Informat” on page 436
- “$UESCEw. Informat” on page 437

### $UESCEw. 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

```
$UESCEw.
```

### 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 remains in UESC.

### Comparisons

The $UESCEw. format performs processing that is the opposite of the $UESCw. format.
Examples
This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=put('¥u5927',$uesce10.) ;</td>
<td>x=¥u5927</td>
</tr>
<tr>
<td>x=put('¥uu5927',$uesce10.) ;</td>
<td>x=¥uu5927</td>
</tr>
<tr>
<td>x=put('¥uuu5927',$uesce10.) ;</td>
<td>x=¥uu5927</td>
</tr>
</tbody>
</table>

See Also

Formats:
"$UESCw. Format" on page 221
Informats:
"$UESCw. Informat" on page 436
"$UESCEw. Informat" on page 437

$UNCRw. 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
$UNCRw.

Syntax Description

w
specifies the width of the output field.
Default: 8
Range: 1–32000

Comparison
The $UNCRw. format performs processing that is the opposite of the $UNCREw. format.
Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>y='abc';</td>
<td>abc</td>
</tr>
<tr>
<td>put y $uncr10.;</td>
<td>&amp;22823</td>
</tr>
</tbody>
</table>

See Also

Formats:
“$UNCREw. Format” on page 224

Informats:
“$UNCRw. Informat” on page 438
“$UNCREw. Informat” on page 439

$UNCREw. 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

$UNCREw.

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 $UNCREw. format performs processing that is the opposite of the $UNCRw. format.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='大abc'; put x $uncr10.;</td>
<td>abc</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UNCRw. Format” on page 223

Informats:

“$UNCRw. Informat” on page 438

“$UNCREw. Informat” on page 439

$UPARENw. 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

$UPARENw.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 8

Range: 27–32000
Details
The character string is encoded with parentheses and Unicode hexadecimal representation.

Comparisons
The $UPARENw. format performs processing that is the opposite of the $UPARENEw. format.

Examples
This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x='ząd';</td>
<td></td>
</tr>
<tr>
<td>y='abc3';</td>
<td></td>
</tr>
<tr>
<td>put x $uparen7.;</td>
<td>&lt;u5927&gt;</td>
</tr>
<tr>
<td>put y $uparen28.;</td>
<td>&lt;u0061&gt; &lt;u0062&gt; &lt;u0063&gt; &lt;u0033&gt;</td>
</tr>
</tbody>
</table>

See Also

Formats:
“$UPARENEw. Format” on page 226

Informats:
“$UPARENw. Informat” on page 440
“$UPARENEw. Informat” on page 442

$UPARENEw. 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
$UPARENEw.

Syntax Description
\(w\)

specifies the width of the output field.

**Default:** 8

**Range:** 1–32000

**Comparisons**

The $\text{SUPARENE}_w$. format performs processing that is the opposite of the $\text{SUPAREN}_w$. format.

**Examples**

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x='\text{\textless u0061}\text{\textgreater u0062}\text{\textless u0063}\text{\textless u0033}');</td>
<td>abc3</td>
</tr>
</tbody>
</table>
See Also

Formats:
“$UPARENw. Format” on page 225

Informats:
“$UPARENw. Informat” on page 440
“$UPARENEw. Informat” on page 442

$UTF8Xw. 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

$UTF8Xw.

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 are 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = '91E5'x; /* Japanese ‘け’ in Shift-JIS */ put x $utf8x10.;</td>
<td>x='e5a4a7'x</td>
</tr>
</tbody>
</table>

See Also
Formats:

“$UCS2Bw. Format” on page 206
“$UCS2Lw. Format” on page 209
“$UCS2Xw. Format” on page 211

Informat:

“$UCS2Bw. Informat” on page 426
“$UCS2Lw. Informat” on page 428
“$UCS2Xw. Informat” on page 430

---

**$VSLOGw. 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

$VSLOGw.$

### Syntax Description

$w$

specifies the width of the output field.

**Default:** 200

**Range:** 1–32000

### Details

The $VSLOGw.$ 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 $VSLOGw.$ format does not correctly process all combinations of data strings. △

### Comparisons

The $VSLOGw.$ format performs processing that is opposite to the $VSLOGRw.$ format.

### Examples

The following example uses the Hebrew input value of “םייגשזל flight”.

The $VSLOGw.$ format processes a character string in visual order and then writes the string in left-to-right logical order.
The following example uses the Arabic input value of “١٢٣” computer.

### See Also

**Formats:**

“$VSLOGRw. Format” on page 230

**Informat:**

“$VSLOGw. Informat” on page 445

“$VSLOGRw. Informat” on page 446

---

**$VSLOGRw. 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

$VSLOGRw.

### Syntax Description

\( w \)

specifies the width of the output field.

**Default:** 200

**Range:** 1–32000
Details

The $VSLOGRw. 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 $VSLOGRw. format does not correctly process all combinations of data strings. △

Comparisons

The $VSLOGRw. format performs processing that is opposite to the $VSLOGw. format.

Examples

The following example uses the Hebrew input value of “-flight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12; flight</td>
<td>נ-half</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “-computer.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put text $logvs12; م-half</td>
<td>computer</td>
</tr>
</tbody>
</table>

See Also

Formats:

$VSLOGw.

Informats:

“$VSLOGw. Informat” on page 445

“$VSLOGRw. Informat” on page 446

WEEKUw. Format

Writes a week number in decimal format by using the U algorithm.

Category: Date and Time

Alignment: left
### Syntax

WEEKUw.

### Syntax Description

**w**

specifies the width of the output field.

**Default:** 11

**Range:** 3–200

### Details

The WEEKUw. format writes a week-number format. The WEEKUw. 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:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

### Comparisons

The WEEKVw. 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 WEEKWw. 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 WEEKUw. 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

```sas
sasdate = '01JAN2003'd;
```
Statements | Results
---+----+----+
v=put(sasdate,weeku3.); | v=---+4---+
w=put(sasdate,weeku5.); | w=03W00
x=put(sasdate,weeku7.); | x=03W0004
y=put(sasdate,weeku9.); | y=2003W0004
z=put(sasdate,weeku11.); | z=2003–W00–04

See Also

Formats:

“WEEKVw. Format” on page 233
“WEEKWw. Format” on page 235

WEEKVw. Format

Writes a week number in decimal format by using the V algorithm.

Category: \( \text{Date and Time} \)
Alignment: \( \text{left} \)

Syntax

WEEKVw.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 11
Range: 3–200

Details

The WEEKVw. 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:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yyWwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyyWwwdd</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

**Comparisons**

The WEEKVw. 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 WEEKWw. 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 WEEKUw. 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**

```sas
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:

“WEEKUw. Format” on page 231
“WEEKWw. Format” on page 235
WEEKWw. Format

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

**Category:** Date and Time  
**Alignment:** left

### Syntax

\[ \text{WEEKWw.} \]

### Syntax Description

\[ w \]

- Specifies the width of the output field.  
- **Default:** 11  
- **Range:** 3–200

### Details

The WEEKWw. 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:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yywww</td>
<td>03W01</td>
</tr>
<tr>
<td>7-8</td>
<td>yywwwdd</td>
<td>03W0101</td>
</tr>
<tr>
<td>9-10</td>
<td>yyyywwwww</td>
<td>2003W0101</td>
</tr>
<tr>
<td>11-200</td>
<td>yyyy-Www-dd</td>
<td>2003-W01-01</td>
</tr>
</tbody>
</table>

### Comparisons

The WEEKWw. 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 WEEKWw. 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 WEEKUw. 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;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>v=put(sasdate,weekw3.);</code></td>
<td></td>
</tr>
<tr>
<td><code>w=put(sasdate,weekw5.);</code></td>
<td></td>
</tr>
<tr>
<td><code>x=put(sasdate,weekw7.);</code></td>
<td></td>
</tr>
<tr>
<td><code>y=put(sasdate,weekw9.);</code></td>
<td></td>
</tr>
<tr>
<td><code>z=put(sasdate,weekw11.);</code></td>
<td></td>
</tr>
<tr>
<td><code>put v;</code></td>
<td>W03</td>
</tr>
<tr>
<td><code>put w;</code></td>
<td>03W03</td>
</tr>
<tr>
<td><code>put x;</code></td>
<td>03W0003</td>
</tr>
<tr>
<td><code>put y;</code></td>
<td>2003W0003</td>
</tr>
<tr>
<td><code>put z;</code></td>
<td>2003-W00-03</td>
</tr>
</tbody>
</table>

See Also

 Formats:

- “WEEKU\textsubscript{w}. Format” on page 231
- “WEEKV\textsubscript{w}. Format” on page 233

YYWEEKU\textsubscript{w}. Format

Writes a week number in decimal format by using the U algorithm, excluding day-of-the-week information.

Category: Date and Time

Alignment: left

Syntax

```
YYWEEKUw.
```

Syntax Description

\textsubscript{w}

specifies the width of the output field.

Default: 7

Range: 3-8
Details

The YYWEEKUw. format writes a week-number format. The YYWEEKUw. 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).

Refer to the following table for widths, formats, and examples:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKUw. format is similar to the WEEKUw. format except that the YYWEEKUw. format does not specify the day-of-week information. Also, the YYWEEKUw. format does not accept any width that is greater than 8.

Examples

```sas
sasdate = '01JAN2007'd;
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweeku3.);</td>
<td>W00</td>
</tr>
<tr>
<td>v=put(sasdate,yyweeku4.);</td>
<td>W00</td>
</tr>
<tr>
<td>w=put(sasdate,yyweeku5.);</td>
<td>07W00</td>
</tr>
<tr>
<td>x=put(sasdate,yyweeku6.);</td>
<td>07W00</td>
</tr>
<tr>
<td>y=put(sasdate,yyweeku7.);</td>
<td>2007W00</td>
</tr>
<tr>
<td>z=put(sasdate,yyweeku8.);</td>
<td>2007-W00</td>
</tr>
</tbody>
</table>

See Also

Formats:

“WEEKUw. Format” on page 231
YYWEEKVw. Format

Writes a week number in decimal format by using the V algorithm, excluding day-of-the-week information.

Category: Date and Time
Alignment: left

Syntax

YYWEEKVw.

Syntax Description

w
specifies the width of the output field.

Default: 7
Range: 3–8

Details

The YYWEEKVw. 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:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>w01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Comparisons

The YYWEEKVw. format is similar to the WEEKVw. format except that the YYWEEKVw. format does not specify the day-of-week information. Also, the YYWEEKVw. format does not accept a width that is greater than 8.

Examples
sasdate = ’01JAN2007’; d;

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweekv3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweekv4.);</td>
<td>07W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweekv5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweekv6.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweekv7.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweekv8.);</td>
<td>2007-W01</td>
</tr>
</tbody>
</table>

See Also

Formats:

“WEEKVw. Format” on page 233

**YYWEEKWw. Format**

Writes a week number in decimal format by using the W algorithm, excluding the day-of-week information.

**Category:** Date and Time

**Alignment:** left

**Syntax**

YYWEEKWw.

**Syntax Description**

**w**

specifies the width of the output field.

**Default:** 7

**Range:** 3–8

**Details**

The YYWEEKWw. 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.
Refer to the following table for widths, formats, and examples:

<table>
<thead>
<tr>
<th>Widths</th>
<th>Formats</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-4</td>
<td>Www</td>
<td>W01</td>
</tr>
<tr>
<td>5-6</td>
<td>yyWww</td>
<td>07W01</td>
</tr>
<tr>
<td>7</td>
<td>yyyyWww</td>
<td>2007W01</td>
</tr>
<tr>
<td>8</td>
<td>yyyy-Www</td>
<td>2007-W01</td>
</tr>
<tr>
<td>9-above</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

**Comparisons**

The YYWEEKWw. format is similar to the WEEKWw. format except that the YYWEEKWw. format does not specify the day-of-week information. Also, the YYWEEKWw. format does not accept any width that is greater than 8.

**Examples**

```
sasdate = '01JAN2007'd
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>u=put(sasdate,yyweekw3.);</td>
<td>W01</td>
</tr>
<tr>
<td>v=put(sasdate,yyweekw4.);</td>
<td>W01</td>
</tr>
<tr>
<td>w=put(sasdate,yyweekw5.);</td>
<td>07W01</td>
</tr>
<tr>
<td>x=put(sasdate,yyweekw6.);</td>
<td>07W01</td>
</tr>
<tr>
<td>y=put(sasdate,yyweekw7.);</td>
<td>2007W01</td>
</tr>
<tr>
<td>z=put(sasdate,yyweekw8.);</td>
<td>2007-W01</td>
</tr>
<tr>
<td>put u;</td>
<td>W01</td>
</tr>
<tr>
<td>put v;</td>
<td>W01</td>
</tr>
<tr>
<td>put w;</td>
<td>07W01</td>
</tr>
<tr>
<td>put x;</td>
<td>07W01</td>
</tr>
<tr>
<td>put y;</td>
<td>2007W01</td>
</tr>
<tr>
<td>put z;</td>
<td>2007-W01</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“WEEKWw. Format” on page 235

**YENw.d Format**

Writes numeric values with yen signs, commas, and decimal points.
Category: Numeric
Alignment: right

Syntax

YENw.d

Syntax Description

w
specifies the width of the output field.
Default: 1
Range: 1–32

d
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 YENw.d writes a decimal point and two decimal digits. If d is 0, then YENw.d does not write a decimal point or decimal digits.

Details

The YENw.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 might be different in other countries.

Examples

```plaintext
put cost yen10.2;
```

<table>
<thead>
<tr>
<th>Cost</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥1,254.71</td>
<td>¥1,254.71</td>
</tr>
</tbody>
</table>

See Also

Informats:
“YENw.d Informat” on page 447
Chapter 8
Functions for NLS

Internationalization Compatibility for SAS String Functions 244
Functions for NLS by Category 260
GETLOCENV Function 262
GETPXLANGUAGE Function 263
GETPXLOCAL Function 264
GETPXREGION Function 265
KCOMPARE Function 266
KCOMPRESS Function 267
KCOUNT Function 268
KCVT Function 268
KINDEX Function 270
KINDEXC Function 270
KLEFT Function 271
KLENGTH Function 272
KLOWCASE Function 272
KPROPCASE Function 273
KPROPCHAR Function 276
KPROPDATA Function 276
KREVERSE Function 279
KRIGHT Function 279
KSCAN Function 280
KSTRCAT Function 281
KSUBSTR Function 281
KSUBSTRB Function 282
KTRANSLATE Function 283
KTRIM Function 284
KTRUNCATE Function 285
KUPCASE Function 285
KUPDATE Function 286
KUPDATEB Function 287
KVERIFY Function 288
NLDATE Function 289
NLDATM Function 291
NLTIME Function 294
SORTKEY Function 295
TRANTAB Function 299
VARTRANSCODE Function 300
VTRANSCODE Function 301
VTRANSCODEX Function 302
UNICODE Function 303
UNICODEC Function 305
Internationalization Compatibility for SAS String Functions

SAS provides string functions and CALL routines that allow you to easily manipulate your character data. Many of the original SAS string functions assume that the size of one character is always one byte. This process works well for data in a single-byte character set (SBCS). However, when some of these functions and CALL routines are used with data in a double-byte character set (DBCS) or multi-byte character set (MBCS), the data is often handled improperly and produce incorrect results.

DBCS encodings require a varying number of bytes to represent each character. MBCS is sometimes used as a synonym for DBCS.

To solve this problem SAS introduced a set of string functions and CALL routines, called K functions, for those string manipulations where DBCS and MBCS data must be handled carefully. This page shows the level of I18N compatibility for each SAS string function. I18N is the abbreviation for internationalization. Compatibility indicates whether a program using a particular string function can be adapted to different languages and locales without program changes.

The user needs to understand the difference between byte-based offset-length and character-based offset-length in order to use the K functions properly. Most K functions require the character-based offset or length. Under SBCS environments, the byte-based unit is identical to character-based unit; however, under DBCS or MBCS environment, there are significant differences, and programmers need to distinguish them. The users might need to change the programming logic in order to use the K functions. Most K functions require strings encoded in current SAS session encoding.

String functions are assigned I18N levels depending on whether the functions can process DBCS, MBCS, or SBCS. Here are descriptions of the levels:

**I18N Level 0**

This function is designed for SBCS data. Do not use this function to process DBCS or MBCS data.

**I18N Level 1**

This function should be avoided, if possible, if you are using a non-English language. The I18N Level 1 functions might not work correctly with DBCS or MBCS encodings under certain circumstances.

**I18N Level 2**

This function can be used for SBCS, DBCS, and MBCS (UTF-8) data.
### Table 8.1 SAS String Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>I18N Level 0</th>
<th>I18N Level 1</th>
<th>I18N Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANYALNUM</td>
<td>Searches a character string for an alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ANYALPHA</td>
<td>Searches a character string for an alphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYCNTRL</td>
<td>Searches a character string for a control character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>ANYDIGIT</td>
<td>Searches a character string for a digit, and returns the first position at which the digit is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYFIRST</td>
<td>Searches a character string for a character that is valid as the first character in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ANYGRAPH</td>
<td>Searches a character string for a graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYLOWER</td>
<td>Searches a character string for a lowercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYNAME</td>
<td>Searches a character string for a character that is valid in a SAS variable name under VALIDVARNAMES=V7, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYPRINT</td>
<td>Searches a character string for a printable character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYPUNCT</td>
<td>searches a character string for a punctuation character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ANYSPACE</td>
<td>Searches a character string for a white-space character (blank, horizontal and vertical tab, carriage return, line feed, and form feed). Returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYUPPER</td>
<td>Searches a character string for an uppercase letter, and returns the first position at which the letter is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ANYXDIGIT</td>
<td>Searches a character string for a hexadecimal character that represents a digit, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>Returns one character in the ASCII or the EBCDIC collating sequence.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAT</td>
<td>Does not remove leading or trailing blanks, and returns a concatenated character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>CATS</td>
<td>Removes leading and trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>CATT</td>
<td>Removes trailing blanks, and returns a concatenated character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATX</td>
<td>Removes leading and trailing blanks, inserts delimiters, and returns a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHOOSEC</td>
<td>Returns a character value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CHOSEN</td>
<td>Returns a numeric value that represents the results of choosing from a list of arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>COALESCEC</td>
<td>Returns the first non-missing value from a list of numeric arguments.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>COLLATE</td>
<td>Returns a character string in ASCII or EBCDIC collating sequence.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPARE</td>
<td>Returns the position of the leftmost character by which two strings differ, or returns 0 if there is no difference.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPBL</td>
<td>Removes multiple blanks from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>COMPGED</td>
<td>Returns the generalized edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPLEV</td>
<td>Returns the Levenshtein edit distance between two strings.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COMPRESS</td>
<td>Returns a character string with specified characters removed from the original string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT</td>
<td>Counts the number of times that a specified substring appears within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNTC</td>
<td>Counts the number of characters in a string that appear or do not appear in a list of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEQUOTE</td>
<td>Removes matching quotation marks from a character string that begins with a quotation mark, and deletes all characters to the right of the closing quotation mark.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIND</td>
<td>Searches for a specific substring of characters within a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FINDC</td>
<td>Searches a string for any character in a list of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>HTMLDECODE</td>
<td>Decodes a string that contains HTML numeric character references or HTML character entity references, and returns the decoded string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>HTMLENCODE</td>
<td>Encodes characters using HTML character entity references, and returns the encoded string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IFC</td>
<td>Returns a character value based on whether an expression is true, false, or missing.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>IFN</td>
<td>Returns a numeric value based on whether an expression is true, false, or missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>Searches a character expression for a string of characters, and returns the position of the string’s first character for the first occurrence of the string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>INDEXC</td>
<td>Searches a character expression for any of the specified characters, and returns the position of that character.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>INDEXW</td>
<td>Searches a character expression for a string that is specified as a word, and returns the position of the first character in the word.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPARE Function” on page 266</td>
<td>Returns the result of a comparison of character expressions.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOMPRESS Function” on page 267</td>
<td>Removes specified characters from a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCOUNT Function” on page 268</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KCVT Function” on page 268</td>
<td>Converts data from one type of encoding data to another encoding data.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KINDEX Function” on page 270</td>
<td>Searches a character expression for a string of characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KINDEXC Function” on page 270</td>
<td>Searches a character expression for specified characters.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KLEFT Function” on page 271</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO-SI.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KLENGTH” Function” on page 272</td>
<td>Returns the length of an argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KLOWCASE” Function” on page 272</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KREVERSE” Function” on page 279</td>
<td>Reverses a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KRIGHT” Function” on page 279</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO-SI.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSCAN” Function” on page 280</td>
<td>Selects a specified word from a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KSTRCAT” Function” on page 281</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KSUBSTR” Function” on page 281</td>
<td>Extracts a substring from an argument.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KSUBSTRB” Function” on page 282</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRANSLATE” Function” on page 283</td>
<td>Replaces specific characters in a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KTRIM” Function” on page 284</td>
<td>Removes trailing DBCS blanks and SO-SI from character expressions.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KTRUNCATE” Function” on page 285</td>
<td>Truncates a numeric value to a specified length.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>“KUPCASE Function” on page 285</td>
<td>Converts all letters in an argument to uppercase.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“KUPDATE Function” on page 286</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>“KUPDATEB Function” on page 287</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>“KVERIFY Function” on page 288</td>
<td>Returns the position of the first character that is unique to an expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LEFT</td>
<td>Left-aligns a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>LENGTH</td>
<td>Returns the length of a non-blank character string, excluding trailing blanks, and returns 1 for a blank character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTHC</td>
<td>Returns the length of a character string, including trailing blanks.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTHM</td>
<td>Returns the amount of memory (in bytes) that is allocated for a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LENGTHN</td>
<td>Returns the length of a character string, excluding trailing blanks.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>LOWCASE</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>MISSING</td>
<td>Returns a numeric result that indicates whether the argument contains a missing value.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NLITERAL</td>
<td>Converts a character string that you specify to a SAS name literal.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NOTALNUM</td>
<td>Searches a character string for a non-alphanumeric character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NOTALPHA</td>
<td>Searches a character string for a nonalphabetic character, and returns the first position at which the character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>NOTCNTRL</td>
<td>Searches a character string for a character that is not a control character, and returns the first position at which that character is found.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>NOTDIGIT</td>
<td>Searches a character string for any character that is not a digit, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NOTFIRST</td>
<td>Searches a character string for an invalid first character in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NOTGRAPH</td>
<td>Searches a character string for a non-graphical character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NOTLOWER</td>
<td>Searches a character string for a character that is not a lowercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>NOTNAME</td>
<td>Searches a character string for an invalid character in a SAS variable name under VALIDVARNAME=V7, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NOTPRINT</td>
<td>Searches a character string for a nonprintable character, and returns the first position at which that character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTPUNCT</td>
<td>Searches a character string for a character that is not a punctuation character, and returns the first position at which that character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOTSPACE</td>
<td>Searches a character string for a character that is not a white-space character (blank, horizontal and vertical tab, carriage return, line feed, and form feed), and returns the first position at which that character is found.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>NOTUPPER</td>
<td>Searches a character string for a character that is not an uppercase letter, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NOTXDIGIT</td>
<td>Searches a character string for a character that is not a hexadecimal character, and returns the first position at which that character is found.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>NVALID</td>
<td>Checks the validity of a character string for use as a SAS variable name.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>PROPCASE</td>
<td>Converts all words in an argument to proper case.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>QUOTE</td>
<td>Adds double quotation marks to a character value.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>RANK</td>
<td>Returns the position of a character in the ASCII or EBCDIC collating sequence.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>REPEAT</td>
<td>Returns a character value that consists of the first argument repeated n+1 times.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>REVERSE</td>
<td>Reverses a character string.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Right-aligns a character expression.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCAN</td>
<td>Returns the nth word from a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SOUNDEX</td>
<td>Encodes a string to facilitate searching.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPEDIS</td>
<td>Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRIP</td>
<td>Returns a character string with all leading and trailing blanks removed.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBPAD</td>
<td>Returns a substring that has a length you specify, using blank padding if necessary.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>SUBSTR</td>
<td>Extracts a substring from an argument.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUBSTRN</td>
<td>Returns a substring, allowing a result with a length of zero.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>TRANSLATE</td>
<td>Replaces specific characters in a character string.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“TRANTAB Function” on page 299</td>
<td>Transcodes data by using the specified translation table.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRANWRD</td>
<td>Replaces or removes all occurrences of a substring in a character string.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Function</td>
<td>Description</td>
<td>I18N Level 0</td>
<td>I18N Level 1</td>
<td>I18N Level 2</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>TRIM</td>
<td>Removes trailing blanks from a character string, and returns one blank if the string is missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRIMN</td>
<td>Removes trailing blanks from character expressions, and returns a string with a length of zero if the expression is missing.</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPCASE</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>URLDECODE</td>
<td>Returns a string that was decoded using the URL escape syntax.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>URLENCODE</td>
<td>Returns a string that was encoded using the URL escape syntax.</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>VERIFY</td>
<td>Returns the position of the first character in a string that is not in any of several other strings.</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
The following categories relate to NLS issues:

Table 8.2  Categories of NLS Formats

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>processes character data</td>
</tr>
<tr>
<td>Currency Conversion</td>
<td>converts one currency to another currency</td>
</tr>
<tr>
<td>DBCS</td>
<td>processes double-byte character set.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>processes data and time data.</td>
</tr>
<tr>
<td>Locale</td>
<td>processes data based on the specified locale.</td>
</tr>
<tr>
<td>Variable Information</td>
<td>processes variable information.</td>
</tr>
</tbody>
</table>

The following table provides brief descriptions of the SAS functions. For more detailed descriptions, see the NLS entry for each function.

Table 8.3  Summary of NLS Functions by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Functions for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>“KCVT Function” on page 268</td>
<td>Converts data from one type of encoding data to another encoding data.</td>
</tr>
<tr>
<td></td>
<td>“TRANTAB Function” on page 299</td>
<td>Transcodes data by using the specified translation table.</td>
</tr>
<tr>
<td></td>
<td>“UNICODE Function” on page 303</td>
<td>converts Unicode characters to the current SAS session encoding.</td>
</tr>
<tr>
<td></td>
<td>“UNICODEDEC Function” on page 305</td>
<td>converts characters in the current SAS session encoding to Unicode characters.</td>
</tr>
<tr>
<td></td>
<td>“UNICODELENG Function” on page 307</td>
<td>specifies the length of the character unit for the Unicode data.</td>
</tr>
<tr>
<td></td>
<td>“UNICODEWIDTH Function” on page 308</td>
<td>specifies the length of a display unit for the Unicode data.</td>
</tr>
<tr>
<td>DBCS</td>
<td>“KCOMPARE Function” on page 266</td>
<td>Returns the result of a comparison of character expressions.</td>
</tr>
<tr>
<td></td>
<td>“KCOMPRESS Function” on page 267</td>
<td>Removes specified characters from a character expression.</td>
</tr>
<tr>
<td>Category</td>
<td>Functions for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“KCOUNT Function” on page 268</td>
<td>Returns the number of double-byte characters in an expression.</td>
<td></td>
</tr>
<tr>
<td>“KINDEX Function” on page 270</td>
<td>Searches a character expression for a string of characters.</td>
<td></td>
</tr>
<tr>
<td>“KINDEXC Function” on page 270</td>
<td>Searches a character expression for specified characters.</td>
<td></td>
</tr>
<tr>
<td>“KLEFT Function” on page 271</td>
<td>Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.</td>
<td></td>
</tr>
<tr>
<td>“KLENGTH Function” on page 272</td>
<td>Returns the length of an argument.</td>
<td></td>
</tr>
<tr>
<td>“KLOWCASE Function” on page 272</td>
<td>Converts all letters in an argument to lowercase.</td>
<td></td>
</tr>
<tr>
<td>“KPROPCASE Function” on page 273</td>
<td>Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.</td>
<td></td>
</tr>
<tr>
<td>“KPROPCCHAR Function” on page 276</td>
<td>Converts special characters to normal characters.</td>
<td></td>
</tr>
<tr>
<td>“KPROPDATA Function” on page 276</td>
<td>Removes or converts unprintable characters.</td>
<td></td>
</tr>
<tr>
<td>“KREVERSE Function” on page 279</td>
<td>Reverses a character expression.</td>
<td></td>
</tr>
<tr>
<td>“KRIGHT Function” on page 279</td>
<td>Right-aligns a character expression by trimming trailing DBCS blanks and SO/SI.</td>
<td></td>
</tr>
<tr>
<td>“KSCAN Function” on page 280</td>
<td>Selects a specified word from a character expression.</td>
<td></td>
</tr>
<tr>
<td>“KSTRCAT Function” on page 281</td>
<td>Concatenates two or more character expressions.</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTR Function” on page 281</td>
<td>Extracts a substring from an argument.</td>
<td></td>
</tr>
<tr>
<td>“KSUBSTRB Function” on page 282</td>
<td>Extracts a substring from an argument according to the byte position of the substring in the argument.</td>
<td></td>
</tr>
<tr>
<td>“KTRANSLATE Function” on page 283</td>
<td>Replaces specific characters in a character expression.</td>
<td></td>
</tr>
<tr>
<td>“KTRIM Function” on page 284</td>
<td>Removes trailing DBCS blanks and SO/SI from character expressions.</td>
<td></td>
</tr>
<tr>
<td>“KTRUNCATE Function” on page 285</td>
<td>Truncates a numeric value to a specified length.</td>
<td></td>
</tr>
<tr>
<td>“KUPCASE Function” on page 285</td>
<td>Converts all letters in an argument to uppercase.</td>
<td></td>
</tr>
<tr>
<td>“KUPDATE Function” on page 286</td>
<td>Inserts, deletes, and replaces character value contents.</td>
<td></td>
</tr>
<tr>
<td>“KUPDATEB Function” on page 287</td>
<td>Inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument.</td>
<td></td>
</tr>
</tbody>
</table>
### GETLOCENV Function

**Returns the current locale/language environment.**

**Category:** Locale

**Syntax**

```
GETLOCENV()
```

**Details**

The GETLOCENV function returns the locale/language environment value for a valid SAS locale. The following environment values are possible:

- **SBCS**
  - The SAS session encoding is SBCS (Single-Byte Character Set).
  - SASWZSD is loaded for string manipulation.

- **DBCS**
  - The SAS session encoding is DBCS (Double-Byte Character Set).
  - SASWZSD is loaded for string manipulation.
MBCS  The SAS session encoding is Unicode(UTF8). SASWZSU is loaded for string manipulation.

If you receive a blank value, then the WZSS subsystem is not available. This action suggests a configuration or installation error.

**Examples**

In the following example, the LOCALE= system option is set to French_France.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td></td>
</tr>
<tr>
<td>environ=getlocenv();</td>
<td></td>
</tr>
<tr>
<td>put environ;</td>
<td>SBCS</td>
</tr>
</tbody>
</table>

---

**GETPXLANGUAGE Function**

Returns the current two-letter language code.

**Category:** Locale

**Syntax**

GETPXLANGUAGE()

**Details**

The GETPXLANGUAGE function returns the two-letter language code based on the current value of the LOCALE= SAS system option. The length of the language name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.

**Examples**

In the first example, the LOCALE= system option is set to French_France. The second example is set to German. The third example is set to English_United States.
Statements | Results
--- | ---
option locale=french_france;
lang=getpxLanguage();
put lang; | fr
option locale=German;
lang=getpxLanguage();
put lang; | de
option locale=en_US;
lang=getpxLanguage();
put lang; | en

See Also

System Options:
   “LOCALE System Option” on page 465
Functions:
   “GETPXREGION Function” on page 265
   “GETPXLOCALE Function” on page 264

GETPXLOCALE Function

Returns the POSIX locale value for a SAS locale.

Category: Locale

Syntax

GETPXLOCALE(<source>)

<source>
is an optional argument that specifies a locale name.

Details

The GETPXLOCALE function returns the POSIX locale value for a valid SAS locale name. If you specify an invalid locale name, then a null string is returned. If you do not specify a value for the <source> argument, then the function returns the POSIX name for the current SAS session. The length of the POSIX locale name is five characters. If the size of the variable that receives the value is less than five characters, the value is truncated.
Examples

In the first example, the LOCALE= system option is set to French_France. In the second example, the <source> argument is set to German_Germany. In the third example, the <source> argument is set to English_United States.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>option locale=french_france;</td>
<td></td>
</tr>
<tr>
<td>locale=getpxLocale();</td>
<td></td>
</tr>
<tr>
<td>put locale;</td>
<td>fr_FR</td>
</tr>
<tr>
<td>locale=getpxLocale(&quot;german_germany&quot;);</td>
<td></td>
</tr>
<tr>
<td>put locale;</td>
<td>de_DE</td>
</tr>
<tr>
<td>locale=getpxLocale(&quot;english_unitedstates&quot;);</td>
<td></td>
</tr>
<tr>
<td>put locale;</td>
<td>en_US</td>
</tr>
</tbody>
</table>

See Also

System Options:
- “LOCALE System Option” on page 465

Functions:
- “GETPXLANGUAGE Function” on page 263
- “GETPXREGION Function” on page 265

GETPXREGION Function

Returns the current two-letter region code.

Category: Locale

Syntax

GETPXREGION()

Details

The GETPXREGION function returns the two-letter region code based on the current LOCALE= SAS system option. The length of the region name is two characters. If the size of the variable that receives the value is less than two characters, the value is truncated.

Examples

In the first example the LOCALE= system option is set to French_France. The second example is set to German. The third example is set to English_United States.
Statements                  Results
option locale=french_france;
region=getpxRegion();
put region;               FR

option locale=German;
region=getpxRegion();
put region;               DE

option locale=en_US;
region=getpxRegion();
put region;               US

See Also

System Options:
   “LOCALE System Option” on page 465
Functions:
   “GETPXLOCALE Function” on page 264
   “GETPXLANGUAGE Function” on page 263

KCOMPARE Function

Returns the result of a comparison of character expressions.

Category:       DBCS

Restriction:    “Internationalization Compatibility for SAS String Functions” on page 244

Tip:     Non-DBCS equivalent function is COMPARE in SAS Language Reference: Dictionary

Syntax

KCOMPARE(source,<pos, <count,>>findstr)

Arguments

source
specifies the character expression 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 expression 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 expression.

Category: DBCS
Restriction: Chapter 8, “Functions for NLS,” on page 243

Syntax

KCOMPRESS(source,<characters-to-remove>)

Arguments

source
specifies a character expression that contains the characters to be removed. When only source is specified, KCOMPRESS returns this expression with all of the single and double-byte blanks removed.

characters-to-remove
specifies the character or characters that KCOMPRESS removes from the character expression.

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 271
“KTRIM Function” on page 284
KCOUNT Function

Returns the number of double-byte characters in an expression.
Category: DBCS
Restrictions: Chapter 8, “Functions for NLS,” on page 243

Syntax

KCOUNT(source)

Arguments

source
specifies the character expression to count.

KCVT Function

Converts data from one type of encoding data to another encoding data.
Category: Character
Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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 555.

Note: ASCIIANY and EBCDICANY 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 555.

Note: ASCIIANY and EBCDICANY are invalid encoding values.

options
specifies character data options. Here are the available options:
Functions for NLS

KCVT Function

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 inype.

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 456
- “DBCSLANG System Option: UNIX, Windows, and z/OS” on page 457
- “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458

Procedure:

Chapter 13, “The DBCSTAB Procedure,” on page 509
KINDEX Function

Searches a character expression for a string of characters.

Category:  DBCS
Restriction:  Chapter 8, “Functions for NLS,” on page 243
Tip:  Non-DBCS equivalent function is INDEX in SAS Language Reference: Dictionary

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 270

KINDEXC Function

Searches a character expression for specified characters.

Category:  DBCS
Restriction:  “Internationalization Compatibility for SAS String Functions” on page 244
Tip:  Non-DBCS equivalent function is INDEXC SAS Language Reference: Dictionary

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 270

KLEFT Function

Left-aligns a character expression by removing unnecessary leading DBCS blanks and SO/SI.
Category: DBCS
Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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 267
“KRIGHT Function” on page 279
“KTRIM Function” on page 284

KLENGTH Function

Returns the length of an argument.

Category: DBCS
Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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
Restriction: “Internationalization Compatibility for SAS String Functions” on page 244
Functions for NLS

KLOWCASE Function

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.

KPROPCASE Function

Converts Chinese, Japanese, Korean, Taiwanese (CJKT) characters.

Category: DBCS

Restriction: Chapter 8, “Functions for NLS,” on page 243

Syntax

str=KPROPCASE(<instr>, (<options>))

Arguments

str
data string that has been converted and is in the current SAS session encoding.

instr
input data string.

options
converts Japanese, Chinese, Korean, and Taiwanese characters based on specified options.

HALF-KATAKANA, FULL-KATAKANA
This option converts half-width Katakana to full-width Katakana and is used only with Japanese encoding.

Restriction: This option cannot be used at the same time with the full-Katakana, half-Katakana option.

FULL-KATAKANA, HALF-KATAKANA
This option converts full-width Katakana to half-width Katakana and is used only with Japanese encoding.

Restriction: This option cannot be used at the same time with the half-Katakana, full-Katakana option.
KATAKANA, ROMAJI
This option converts the Katakana character string to a Romaji character string and is used only with Japanese encoding.

Restriction: This option cannot be used at the same time with the Romaji, Katakana option.

ROMAJI, KATAKANA
This option converts the Romaji character string to a Katakana character string and is used only with Japanese encoding.

Restriction: This option cannot be used at the same time with the Katakana, Romaji option.

FULL-ALPHABET, HALF-ALPHABET
This option converts the Full-Alphabet characters to Half-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction: This option cannot be used at the same time with the Half-Alphabet, Full-Alphabet option.

HALF-ALPHABET, FULL-ALPHABET
This option converts the Half-Alphabet characters to Full-Alphabet characters and is used only with Japanese, Chinese, Korean, and Taiwanese encoding.

Restriction: This option cannot be used at the same time with the Full-Alphabet, Half-Alphabet option.

LOWERCASE, UPPERCASE
This option converts lowercase alphabet characters to uppercase alphabet characters.

Restriction: This option cannot be used at the same time with the Uppercase, Lowercase option.

UPPERCASE, LOWERCASE
This option converts uppercase alphabet characters to lowercase alphabet characters.

Restriction: This option cannot be used at the same time with the Lowercase, Uppercase option.

PROPER
This option specifies the following default options based on the encoding:

Japanese encoding:
- Half-Katakana, Full-Katakana
- Full-alphabet, Half-alphabet
- Lowercase, Uppercase

Korean encoding:
- Full-alphabet, Half-alphabet

Chinese encoding:
- Full-alphabet, Half-alphabet

Taiwanese encoding:
- Full-alphabet, Half-alphabet

Details
This function converts the input string based on the specified options and default options. The KPROPCASE function supports the Chinese, Japanese, Korean, Taiwanese (CJKT) environment.
Example

The following example demonstrates the functionality of the KPROPCASE function:

```asciidoc
length fullkana halfkana upper lower fullalpha $ 200;
length str1 str2 str3 str4 str5 str7 str8 $ 30 str6 $44;

lower = 'do-naxtutsu'; /* Doughnuts in Japanese Roman word. */
upper = 'DO-NAXTUTSU'; /* Doughnuts in Japanese Roman word. */
fullkana = unicode('\u30C9\u30FC\u30CA\u30C3\u30C4');
halfkana = unicode('\uFF84\uFF9E\uFF70\uFF85\uFF6F\uFF82');
fullalpha =
    unicode('\uFF24\uFF2F\uFF0D\uFF2E\uFF38\uFF34\uFF35\uFF33\uFF35');

str1 = kpropcase(fullkana, 'full-katakana,half-katakana');
if (halfkana EQ trim(str1)) then
    put str1= $hex14.;
str2 = kpropcase(halfkana, 'half-katakana,full-katakana');
if (fullkana EQ trim(str2)) then
    put str2= $hex22.;
str3 = kpropcase(fullkana, 'katakana,romaji');
if (trim(str3) EQ upper) then
    put str3=;
str4 = kpropcase(upper, 'romaji,katakana');
if (trim(str4) EQ fullkana) then
    put str4= $hex22.;
str5 = kpropcase(fullalpha, 'full-alphabet,half-alphabet');
if (trim(upper) EQ str5) then
    put str5=;
str6 = kpropcase(upper, 'half-alphabet,full-alphabet');
if (trim(str6) EQ fullalpha) then
    put str6= $hex46.;
str7 = kpropcase(lower, 'lowercase,uppercase');
if (trim(str7) EQ upper) then
    put str7=;
str8 = kpropcase(upper, 'uppercase,lowercase');
if (trim(str8) EQ lower) then
    put str8=;
RESULTS:
strl=C4DEB0C5AFC220
str2=8368815B83698362836320
str3=DO-NAXTUTSU
str4=8368815B83698362836320
str5=DO-NAXTUTSU
str6=8263826E817C826D826082778273827482738272827420
str7=DO-NAXTUTSU
str8=do-naxtutsu
```
KPROPCHAR Function

Converts special characters to normal characters.

Category: DBCS

Syntax

\[
\text{str} = \text{KPROPCHAR}(\text{<instr>})
\]

Arguments

\[
\text{str} \quad \text{result string. Special characters are converted to normal characters.}
\]

\[
\text{instr} \quad \text{input data string.}
\]

Details

This function converts special characters to normal characters. The KPROPCHAR function converts the characters from the following ranges:

- Enclosed alphanumeric values: \u2460 to \u24FF. See http://www.unicode.org/charts/PDF/U2460.pdf.
- Dingbats: \u2776 to \u2793. See http://www.unicode.org/charts/PDF/U2700.pdf.
- Enclosed CJK letters and months: \u3200 to \u32FF. See http://www.unicode.org/charts/PDF/U3200.pdf.

Example

The following example demonstrates the functionality of the KPROPCHAR function:

```plaintext
length in1 out1 $30;
in1=unicode('\u2460\u2473\u277F\u325F');
out1=KPROPCHAR(in1);
put out1;
```

RESULTS:

| 1 | 20 | -10 | 35 |

KPROPDATA Function

Removes or converts unprintable characters.

Category: DBCS
Syntax

\[ \text{str} = \text{KPROPDATA}(\langle \text{instr} \rangle (\langle \text{option}, \text{input encode name}, \text{output encode name} \rangle)) \]

Arguments

\textit{str}

data string that has been converted and is in session encoding.

\textit{instr}

input data string.

\textit{options}

specifies instructions on processing unprintable characters:

- **UESC**
  - Converts unprintable characters using a Unicode escaped string (for example, \u0000\u1234).
- **TRIM**
  - Removes unprintable characters. No replacement character is used.
- **BLANK or "**
  - Replaces each unprintable character with a single-byte blank.
- **QUESTION or "?"**
  - Replaces unprintable characters with a single-byte '?'.
- **HEX**
  - Replaces unprintable characters with a hexadecimal representation (for example, 0x810x82).
- **TRUNCATE or TRUNC**
  - Truncates the data string when the first unprintable character is encountered.
- **REMOVE**
  - Removes the data string if any unprintable characters are found.
- **NCR**
  - Encodes the unprintable characters using NCR representation if the code is available in Unicode.

\textit{input encode name}

specifies the input data’s encoding name if necessary. If the input encode name is not specified, then the KPROPDATA function processes the data as the current SAS session encoded string. For information on SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 555.

\textit{output encode name}

specifies the output data’s encoding name. If the encoding name is not specified, the KPROPDATA function recognizes the output as the current SAS session encoding. For information on SAS encoding names, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 555.

Details

This function converts the input data string to the current SAS session encoding and removes or replaces unprintable characters based on the options.
Example

The following example demonstrates the functionality of the KPROPDATA function:

```plaintext
length instr $12;
length str1 str2 str3 str4 str5 str6 str7 str8 str9 str10$ 50;

instr = "534153"x||"ae"x || " System";
put instr;

str1 = kpropdata(instr);
put str1= +2 str1= $hex26.;
str2 = kpropdata(instr,'UESC');
put str2= +2 str2= $hex26.;
str3 = kpropdata(instr, 'UESC','wlatin1');
put str3= +2 str3= $hex34.;
str4 = kpropdata(instr,'TRIM','wlatin1');
put str4= +2 str4= $hex26.;
str5 = kpropdata(instr,'BLANK', 'wlatin1');
put str5= +2 str5= $hex26.;
str6 = kpropdata(instr,'?', 'wlatin1');
put str6= +2 str6= $hex26.;
str7 = kpropdata(instr,'hex', 'wlatin1');
put str7= +2 str7= $hex26.;
str8 = kpropdata(instr,'TRUNC', 'wlatin1');
put str8= +2 str8= $hex26.;
str9 = kpropdata(instr,'REMOVE', 'wlatin1');
put str9= +2 str9= $hex26.;
str10 = kpropdata(instr,'NCR', 'wlatin1');
put str10= +2 str10= $hex26.;

RESULTS:
SAS? System
str1=SAS? System str1=534153AE2053797374656D2020
str2=SAS? System str2=534153AE2053797374656D2020
str3=SAS\uff6e System str3=5341535C75666636652053797374656D20
str4=SAS System str4=5341532053797374656D202020
str5=SAS System str5=5341532053797374656D2020
str6=SAS System str6=53415333F2053797374656D2020
str7=SAS\xAE System str7=5341535C784145205379737465
str8=SAS str8=5341532020202020202020202020
str9= str9=20202020202020202020202020
str10=SAS® System str10=53415326233137343B20537973
```
KREVERSE Function

Reverses a character expression.

Category: DBCS

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244


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

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244


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 267
“KLEFT Function” on page 271
“KTRIM Function” on page 284

KSCAN Function

Selects a specified word from a character expression.

Category: DBCS

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244


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 expression you want KSCAN to select.

Tip: If $n$ is negative, KSCAN selects the word in the character expression starting from the end of the string. If $|n|$ is greater than the number of words in the character expression, KSCAN returns a blank value.

delimiters
specifies a character variable that produces characters that you want KSCAN to use as word separators in the character expression.

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 expressions.

**Category:** DBCS

**Restriction:** “Internationalization Compatibility for SAS String Functions” on page 244

**Tip:** Non-DBCS equivalent function is CAT in SAS *Language Reference: Dictionary*.

**Syntax**

```
KSTRCAT(argument-1, argument-2<, … argument-n>)
```

**Arguments**

- `argument` specifies any single-byte or double-byte character expression.

**Details**

KSTRCAT concatenates two or more single-byte or double-byte character expressions. It also removes unnecessary SO/SI pairs between the expressions.

---

**KSUBSTR Function**

Extracts a substring from an argument.

**Category:** DBCS

**Restriction:** “Internationalization Compatibility for SAS String Functions” on page 244

**Tip:** SUBSTR in SAS *Language Reference: Dictionary*.

**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 \textit{argument} after \textit{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 \textit{argument}. The portion begins with the character specified by \textit{position} and is the number of characters specified by \( n \).

A variable that is created by KSUBSTR obtains its length from the length of \textit{argument}.

See Also

Functions:

“KSUBSTRB Function” on page 282

---

**KSUBSTRB Function**

Extracts a substring from an argument according to the byte position of the substring in the argument.

Category: DBCS

Restriction: Chapter 8, “Functions for NLS,” on page 243

Syntax

KSUBSTRB(\text{argument}, \textit{position}<,\textit{n}>)

Arguments

\textit{argument}

specifies any SAS character expression.

\textit{position}

specifies the beginning character position in byte units.

\textit{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 \textit{argument} after \textit{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 \textit{argument}. The portion begins with the byte unit specified by \textit{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 281

KTRANSLATE Function

Replaces specific characters in a character expression.

Category: DBCS

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244


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

**Restriction:** Chapter 8, “Functions for NLS,” on page 243

**Tip:** Non-DBCS equivalent function is TRIM in *SAS Language Reference: Dictionary*.

#### 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 267

“KLEFT Function” on page 271

“KRIGHT Function” on page 279
KTRUNCATE Function

Truncates a numeric value to a specified length.

Category: DBCS

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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 letters in an argument to uppercase.

Category: DBCS

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244


Syntax

KUPCASE(argument)

Arguments

argument
  specifies any SAS character expression.
Details
The KUPCASE function copies a character argument, converts all lowercase letters to uppercase letters, and returns the altered value as a result.

KUPDATE Function

Inserts, deletes, and replaces character value contents.

Category: DBCS
Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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 cannot 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 replaces 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.

Note: If you set the NLSCOMPATMODE system option to on, parameter, characters-to-replace, processes the data based on previous SAS releases. If NLSCOMPATMODE is off, then characters-to-replace uses the 9.2 functionality. See the following table for examples.
Functions for NLS

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

Restriction: “Internationalization Compatibility for SAS String Functions” on page 244

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:** $n$ cannot 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 \textit{characters-to-replace} from the function.

Tip:  If you omit $n$, SAS uses all of the characters in \textit{characters-to-replace} to replace the values of \textit{argument}.

\textit{characters-to-replace} specifies a character expression to replace the contents of \textit{argument}.

Restriction: \textit{characters-to-replace} is optional, but you cannot omit both \textit{characters-to-replace} and $n$ from the function.

Tip:  Enclose a literal string of characters in quotation marks.

Details

The \textsc{KUPDATEB} function replaces the value of \textit{argument} with the expression in \textit{characters-to-replace}. \textsc{KUPDATEB} replaces $n$ byte units starting at the byte unit that you specify in \textit{position}.

See Also

Functions:

“\textsc{KUPDATE Function}” on page 286

\textbf{KVERIFY Function}

Returns the position of the first character that is unique to an expression.

Category:  \textsc{DBCS}

Restriction:  “Internationalization Compatibility for SAS String Functions” on page 244

Tip:  VERIFY in \textit{SAS Language Reference: Dictionary}

Syntax

\texttt{KVERIFY(source,excerpt-1<,...,excerpt-n>)}

Arguments

\textit{source} specifies any SAS character expression.

\textit{excerpt} specifies any SAS character expression. If you specify more than one excerpt, separate them with a comma.

Details

The \textsc{KVERIFY} function returns the position of the first character in \textit{source} that is not present in any \textit{excerpt}. If \textsc{KVERIFY} finds every character in \textit{source} in at least one \textit{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 are formatted in output. The following descriptors are case sensitive:

#
removes the leading zero from the result.

%%
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 filename that is created from a SAS date value.
### Statements

```sas
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.

```sas
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:

“NLDATEw. Format” on page 91

### 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**

```sas
NLDATM(datetime,descriptor)
```
Arguments

datetime
specifies a SAS datetime value.

descriptor
is a variable or expression that specifies how dates and times are formatted in output. The following descriptors are case sensitive:

#
removes the leading zero from the result.

%%
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.
## 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:
    - `#`
      - removes the leading zero from the result.
    - `%%`
      - 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.

### See Also

Format:

“NLDATMw. Format” on page 99
%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 a.m. or p.m. time that is created from a SAS time.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nltime('12:39:43't,'%i%p');</td>
<td>00 PM</td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
<tr>
<td>options locale=German;</td>
<td></td>
</tr>
<tr>
<td>time_ampm=nltime('12:39:43't,'%i%p');</td>
<td>00 nachm</td>
</tr>
<tr>
<td>put time_ampm;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Formats:
“NLTIMEw. Format” on page 204

SORTKEY Function
creates a linguistic sort key.

Category: Locale
Syntax

\texttt{sortKey}(\textit{string, \langle locale, strength, case, numeric, order\rangle})

Arguments

\textit{string}  
character expression

\textit{locale}  
specifies the locale name in the form of a POSIX name (ja_JP). See Table 15.1 on page 545 for a list of locale names and Posix values.

\textit{strength}  
The value of strength is related to the collation level. There are five collation-level values. The following table provides information regarding the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY or P</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>SECONDARY or S</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;as&quot; &lt; &quot;às&quot; &lt; &quot;at&quot;).</td>
<td>Other differences between letters can also be considered secondary differences, depending on the language. A secondary difference is ignored when there is a primary difference anywhere in the strings.</td>
</tr>
<tr>
<td>TERTIARY or T</td>
<td>Upper and lower case differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Ao&quot; &lt; &quot;aò&quot;).</td>
<td>An example is the difference between large and small Kana. A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings.</td>
</tr>
</tbody>
</table>
QUATERNARY or Q
When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, "ab" < "a-b" < "aB"). This difference is ignored when there is a primary, secondary, or tertiary difference. The quaternary level should be used if ignoring punctuation is required or when processing Japanese text.

IDENTICAL or I
When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the NFD form of each string are compared at this level, just in case there is no difference at levels 1-4. For example, only Hebrew cantillation marks are distinguished at this level. This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence.

case order
sorts uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL. The following table provides the values and information for the case order argument.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER or U</td>
<td>Sorts upper case letters first, then the lower case letters.</td>
</tr>
<tr>
<td>LOWER or L</td>
<td>Sorts lower case letters first, then the upper case letters.</td>
</tr>
</tbody>
</table>

numeric collation
orders numbers by the numeric value instead of the number's characters.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NUMERIC or N</td>
<td>Order numbers (integers) by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
</tbody>
</table>

collation order
There are two types of collation values: Phonebook and Traditional. If you do not select a collation value, then the user's locale-default collation is selected. The following table provides more information.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHONEBOOK or P</td>
<td>specifies a phonebook style ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>TRADITIONAL or T</td>
<td>specifies a traditional style ordering of characters. Select TRADITIONAL only with the Spanish language.</td>
</tr>
</tbody>
</table>
Details

The SORTKEY function creates a linguistic sort key for data. You must enter at least one argument. If the length of the variable that receives the key is not large enough, the data truncates, and a warning is displayed.

locale

Locale values use the POSIX name (ll_RR). LL represents the two-letter language code, and RR represents the two-letter region code. For example, en_US is the POSIX name for English, United States. en represents the English language, and US represents the United States. If a locale value is not specified, then the session locale is used.

strength

The strength argument determines whether accents or case affect collating or matching text. If no value is specified for strength, then the locale determines the value. The following values can be specified for strength.

- PRIMARY: This value includes base letters, for example, the letters, A, a, and Å are all processed the same.
- SECONDARY: This value processes data the same as PRIMARY, and accents are processed. The letters A and a are processed equally, and Å is processed as an accented character.
- TERTIARY: This value processes data the same as SECONDARY, and the character's case is processed. For example, A, a, and Å are all processed differently.
- QUATERNARY: This value processes data the same as TERTIARY, and punctuation is processed.
- IDENTICAL: This value process data the same as QUATERNARY, and code point is processed.

case order

specifies to sort data using upper case or lower case letter. The following table shows examples of specifying the UPPER value or the LOWER value.

<table>
<thead>
<tr>
<th>UPPER</th>
<th>LOWER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aztec</td>
<td>aztec</td>
</tr>
<tr>
<td>aztec</td>
<td>Aztec</td>
</tr>
<tr>
<td>Mars</td>
<td>mars</td>
</tr>
<tr>
<td>mars</td>
<td>Mars</td>
</tr>
</tbody>
</table>

collation order

The collation order value PHONEBOOK is ignored unless the locale is a German language.

The collation order value TRADITIONAL is ignored unless the locale is a Spanish language.

A warning message displays for other locales.
TRANTAB Function

Transcodes data by using the specified translation table.

Category: Character

Syntax

\texttt{TRANTAB(string, trantab\_name)}

\textit{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 9.2 supports the TRANTAB function for backward compatibility. However, using the locale= system option is preferred in later SAS releases.

Arguments

\textit{string}

input data that is transcoded.

\textit{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.

\textit{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:

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>teststrg=trantab('testing','lat2_ucs');</td>
<td>TESTING</td>
</tr>
<tr>
<td>put teststrg;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Procedures:
  Chapter 14, “The TRANTAB Procedure,” on page 515

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.


Examples

The following example shows how to determine whether a character variable is transcoded:

```sas
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 301
- “VTRANSCODEX Function” on page 302

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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td></td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rcl1 = vtranscode(y);</td>
<td></td>
</tr>
<tr>
<td>put rcl1=;</td>
<td>rcl1=0</td>
</tr>
</tbody>
</table>

See Also

Functions:

- “VTRANSCODEX Function” on page 302

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 `VTRANSCODE` 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**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>attrib x transcode = yes;</td>
<td></td>
</tr>
<tr>
<td>attrib y transcode = no;</td>
<td></td>
</tr>
<tr>
<td>rcl = vtranscodex('y');</td>
<td>rcl=0</td>
</tr>
<tr>
<td>put rcl=;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Functions:

“`VTRANSCODE` Function” on page 301

Statements:

`ATTRIB` in *SAS Language Reference: Dictionary*

---

**UNICODE Function**

Converts Unicode characters to the current SAS session encoding.

**Category:** Character

**Syntax**

```plaintext
STR=UNICODE(<instr>,(<Unicode type>))
```
Arguments

str
Data string that has been converted to the current SAS session encoding.

instr
input data string.

Unicode type
Unicode character formats
ESC    Unicode Escape (for example, \u0042 ). ESC is the default format.
NCR    Numeric Character Representation (for example, &#22823 or &#177 ; )
PAREN  Unicode Parenthesis Escape (for example, <u0061>)
UCS2   UCS2 encoding with native endian.
UCS2B  UCS2 encoding with big endian.
UCS2L  UCS2 encoding with little endian.
UCS4   UCS4 encoding with native endian.
UCS4B  UCS4 encoding with big endian.
UCS4L  UCS4 encoding with little endian.
UTF16  UTF16 encoding with big endian.
UTF16B UTF16 encoding with big endian.
UTF16L UTF16 encoding with little endian.
UTF8   UTF8 encoding.

Details
This function reads Unicode characters and converts them to the current SAS session encoding.
Examples

The following example demonstrates the functionality of the UNICODE function:

```
Examples: (Submitted under Little endian system.)
str1=unicode("\u0041\u0042\u0043");
str2=unicode("\0041\0042\u0043","esc");
str3=unicode("\# 177;","ncr");
str4=unicode("\# 22823;","ncr");
str5=unicode("<u0061><u0062>","paren");
str6=unicode('2759','ucs2');
str7=unicode('5927','ucs2b');
str8=unicode('2759','ucs21');
str9=unicode('27590000','ucs4');
str10=unicode('0005927','ucs4b');
str11=unicode('27590000','ucs41');
str12=unicode('E5A4A7','utf8');
str13=unicode('2759','utf16');
str14=unicode('5927','utf16b');
str15=unicode('2759','utf161');
```

Results:
str1=ABC
str1=ABC
str3=ə
str4=大
str5=ab
str6=大
str7=大
str8=大
str9=大
str10=大
str11=大
str12=大
str13=大
str14=大
str15=大

**UNICODEC Function**

converts characters in the current SAS session encoding to Unicode characters.

Category: Character

**Syntax**

```
STR=UNICODEC(<instr>,<Unicode type>)
```
Arguments

\textit{str} \\
\hspace{10mm} data string that has been converted to Unicode encoding.

\textit{instr} \\
\hspace{10mm} input data string.

\textbf{Unicode type} \\

<table>
<thead>
<tr>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESC</td>
<td>Unicode Escape (for example, \u0042 )ESC is the default format.</td>
</tr>
<tr>
<td>NCR</td>
<td>Numeric Character Representation (for example, &amp;#22823 or &amp;#177 ;)</td>
</tr>
<tr>
<td>PAREN</td>
<td>Unicode Parenthesis Escape (for example, &lt;u0061&gt;)</td>
</tr>
<tr>
<td>UCS2</td>
<td>UCS2 encoding with native endian.</td>
</tr>
<tr>
<td>UCS2B</td>
<td>UCS2 encoding with big endian.</td>
</tr>
<tr>
<td>UCS2L</td>
<td>UCS2 encoding with little endian.</td>
</tr>
<tr>
<td>UCS4</td>
<td>UCS4 encoding with native endian.</td>
</tr>
<tr>
<td>UCS4B</td>
<td>UCS4 encoding with big endian.</td>
</tr>
<tr>
<td>UCS4L</td>
<td>UCS4 encoding with little endian.</td>
</tr>
<tr>
<td>UTF16</td>
<td>UTF16 encoding with big endian.</td>
</tr>
<tr>
<td>UTF16B</td>
<td>UTF16 encoding with big endian.</td>
</tr>
<tr>
<td>UTF16L</td>
<td>UTF16 encoding with little endian.</td>
</tr>
<tr>
<td>UTF8</td>
<td>UTF8 encoding.</td>
</tr>
</tbody>
</table>

Details

This function reads characters that are in the current SAS session encoding and converts them to Unicode encoding.

Examples

The following example demonstrates the functionality of the \texttt{UNICODEC} function:

```plaintext
length str4 $20;
dai=unicode('\\u5927');

str1=unicodec("ABC");
str2=unicodec("ABC","esc");
str3=unicodec(dai, 'ncr');
str4=unicodec("ab","paren");
str5=unicodec(dai, 'ucs2');
str6=unicodec(dai, 'ucs2b');
str7=unicodec(dai, 'ucs2l');
str8=unicodec(dai, 'ucs4');
str9=unicodec(dai, 'ucs4b');
str10=unicodec(dai, 'ucs4l');
str11=unicodec(dai, 'utf8');
```
Functions for NLS

UNICODELEN Function

specifies the length of the character unit for the Unicode data.

Category: Character

Syntax

UNICODELEN()

Details

The UNICODELEN function specifies the length of the character unit for the Unicode data.

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodelen(&quot;abc&quot;);</td>
<td>len1=4</td>
</tr>
<tr>
<td>len2=unicodelen(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;)</td>
<td>len2=4</td>
</tr>
<tr>
<td>len3=unicodelen(&quot;大&quot;,&quot;ncr&quot;);</td>
<td>len3=1</td>
</tr>
<tr>
<td>len4=unicodelen(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>
See Also

Functions:

“UNICODEWIDTH Function” on page 308

UNICODEWIDTH Function

specifies the length of a display unit for the Unicode data.

Category: Character

Syntax

UNICODEWIDTH()

Details

The UNICODEWIDTH function specifies the length of a display unit for the Unicode data. The display unit displays the width of a character when the character is displayed with fixed width font. Characters between 0x3000 and 0x303F, 0x3400 and 0x4DFF, 0x4E00 and 0x9FFF, 0xF900 and 0xFAFF, inclusively, have the value of a display unit 2. Other characters are display unit 1

Examples

This example uses the Japanese Shift_JIS session encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>len1=unicodewidth(&quot;abc&quot;);</td>
<td>len1=5</td>
</tr>
<tr>
<td>len2=unicodewidth(&quot;\u0041\u0042\u0043\u5927&quot;,&quot;esc&quot;);</td>
<td>len2=5</td>
</tr>
<tr>
<td>len3=unicodewidth(&quot;&amp;#22823; &quot;,&quot;ncr&quot;);</td>
<td>len3=2</td>
</tr>
<tr>
<td>len4=unicodewidth(&quot;&lt;u0061&gt;&lt;u0062&gt;&quot;,&quot;paren&quot;);</td>
<td>len4=2</td>
</tr>
</tbody>
</table>

See Also

Functions:

“UNICODELEN Function” on page 307
Informats for NLS

Informats for NLS by Category

$CPTDWw. Informat 318
$CPTWDw. Informat 319
EUROw.d Informat 320
EUROXw.d Informat 322
JDATEYMDw. Informat 323
JNENGOw. Informat 324
$KANJIw. Informat 326
$KANJIXw. Informat 326
$LOGVSw. Informat 327
$LOGVSRw. Informat 328
MINGUOw. Informat 330
NENGOw. Informat 331
NLDATEw. Informat 332
NLDATMw. Informat 333
NLMNIAEDw.d Informat 334
NLMNIAUDw.d Informat 335
NLMNIBGNw.d Informat 336
NLMNIBRLw.d Informat 337
NLMNICADw.d Informat 338
NLMNICHFw.d Informat 339
NLMNICNYw.d Informat 340
NLMNICZKw.d Informat 341
NLMNIDKKw.d Informat 342
NLMNIEEKw.d Informat 343
NLMNIEGPw.d Informat 344
NLMNIEURw.d Informat 345
NLMNIGBPw.d Informat 346
NLMNIHKDw.d Informat 347
NLMNIHRKw.d Informat 348
NLMNIHUFw.d Informat 349
NLMNIIDRw.d Informat 350
NLMNIILSw.d Informat 351
NLMNIINRw.d Informat 352
NLMNIJPYw.d Informat 353
NLMNIKRWw.d Informat 354
NLMNILTLw.d Informat 355
NLMNILVLw.d Informat 356
NLMNIMOPw.d Informat 357
NLMNIMXNw.d Informat 358
NLMNIMYRw.d Informat 359
There are six categories of SAS informats that support NLS:

### Table 9.1 Categories of Informats for NLS

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI text handling</td>
<td>Instructs SAS to read bidirectional data values from data variables.</td>
</tr>
<tr>
<td>Character</td>
<td>Instructs SAS to read character data values into character variables.</td>
</tr>
<tr>
<td>DBCS</td>
<td>Instructs SAS to manage various Asian languages.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>Instructs SAS to read data values into variables that represent dates, times, and datetimes.</td>
</tr>
<tr>
<td>Hebrew text handling</td>
<td>Instructs SAS to read Hebrew data from data variables.</td>
</tr>
<tr>
<td>Numeric</td>
<td>Instructs SAS to read numeric data values into numeric variables.</td>
</tr>
</tbody>
</table>

The following table provides brief descriptions of the SAS informats. For more detailed descriptions, see the NLS entry for each informat.
<table>
<thead>
<tr>
<th>Category</th>
<th>Informs for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIDI text handling</td>
<td>&quot;$LOGVS. Informat&quot; on page 327</td>
<td>Reads a character string that is in left-to-right logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>&quot;$LOGVSR. Informat&quot; on page 328</td>
<td>Reads a character string that is in right-to-left logical order, and then converts the character string to visual order.</td>
</tr>
<tr>
<td></td>
<td>&quot;$VSLOG. Informat&quot; on page 445</td>
<td>Reads a character string that is in visual order, and then converts the character string to left-to-right logical order.</td>
</tr>
<tr>
<td></td>
<td>&quot;$VSLOGR. Informat&quot; on page 446</td>
<td>Reads a character string that is in visual order, and then converts the character string to right-to-left logical order.</td>
</tr>
<tr>
<td>Character</td>
<td>&quot;$REVERJ. Informat&quot; on page 424</td>
<td>Reads character data from right to left and preserves blanks.</td>
</tr>
<tr>
<td></td>
<td>&quot;$REVERS. Informat&quot; on page 425</td>
<td>Reads character data from right to left, and then left aligns the text.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2B. Informat&quot; on page 426</td>
<td>Reads a character string that is encoded in big-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2BE. Informat&quot; on page 427</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2L. Informat&quot; on page 428</td>
<td>Reads a character string that is encoded in little-endian, 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2LE. Informat&quot; on page 429</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2X. Informat&quot; on page 430</td>
<td>Reads a character string that is encoded in 16-bit, UCS2, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS2XE. Informat&quot; on page 431</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS4B. Informat&quot; on page 432</td>
<td>Reads a character string that is encoded in big-endian, 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS4L. Informat&quot; on page 433</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS4X. Informat&quot; on page 434</td>
<td>Reads a character string that is encoded in 32-bit, UCS4, Unicode encoding, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>Category</td>
<td>Informs for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>----------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>&quot;$UCS4XEw. Informat&quot; on page 435</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UESCw. Informat&quot; on page 436</td>
<td>Reads a character string that is encoded in UESC representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;SUESCEw. Informat&quot; on page 437</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UESC representation.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UNCRw. Informat&quot; on page 438</td>
<td>Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UNCREw. Informat&quot; on page 439</td>
<td>Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UPARENw. Informat&quot; on page 440</td>
<td>Reads a character string that is encoded in UPAREN representation, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UPARENEw. Informat&quot; on page 442</td>
<td>Reads a character string that is in the encoding of the current SAS session, and then converts the character string to UPAREN representation.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UPARENpw. Informat&quot; on page 443</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>&quot;$UTF8Xw. Informat&quot; on page 444</td>
<td>Reads a character string that is encoded in UTF-8, and then converts the character string to the encoding of the current SAS session.</td>
</tr>
<tr>
<td>DBCS</td>
<td>&quot;$KANJIw. Informat&quot; on page 326</td>
<td>Removes shift code data from DBCS data.</td>
</tr>
<tr>
<td></td>
<td>&quot;$KANJIXw. Informat&quot; on page 326</td>
<td>Adds shift-code data to DBCS data.</td>
</tr>
<tr>
<td>Date and Time</td>
<td>&quot;JDATEYMDw. Informat&quot; on page 323</td>
<td>Reads Japanese kanji date values in the format yymmmdd or yyyyymmdd.</td>
</tr>
<tr>
<td></td>
<td>&quot;JNENGOw. Informat&quot; on page 324</td>
<td>Reads Japanese kanji date values in the form yymmd.</td>
</tr>
<tr>
<td></td>
<td>&quot;MINGUOw. Informat&quot; on page 330</td>
<td>Reads dates in Taiwanese format.</td>
</tr>
<tr>
<td></td>
<td>&quot;NENGOw. Informat&quot; on page 331</td>
<td>Reads Japanese date values in the form eyymmd.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATEw. Informat&quot; on page 332</td>
<td>Reads the date value in the specified locale, and then converts the date value to the local SAS date value.</td>
</tr>
<tr>
<td></td>
<td>&quot;NLDATMw. Informat&quot; on page 333</td>
<td>Reads the datetime value of the specified locale, and then converts the datetime value to the local SAS datetime value.</td>
</tr>
<tr>
<td>Category</td>
<td>Informs for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>“NLTIMAPw. Informat” on page 422</td>
<td>Reads the time value and uses a.m. and p.m. in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td></td>
<td>“NLTIMEw. Informat” on page 423</td>
<td>Reads the time value in the specified locale, and then converts the time value to the local SAS time value.</td>
</tr>
<tr>
<td>Hebrew text handling</td>
<td>“$CPTDWw. Informat” on page 318</td>
<td>Reads a character string that is in Hebrew DOS (cp862) encoding, and then converts the character string to Windows (cp1255) encoding.</td>
</tr>
<tr>
<td></td>
<td>“$CPTWDw. Informat” on page 319</td>
<td>Reads a character string that is in Windows (cp1255) encoding, and then converts the character string to Hebrew DOS (cp862) encoding.</td>
</tr>
<tr>
<td>Numeric</td>
<td>“EUROw.d Informat” on page 320</td>
<td>Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.</td>
</tr>
<tr>
<td></td>
<td>“EUROXw.d Informat” on page 322</td>
<td>Reads numeric values and removes embedded characters in European currency.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIAEDw.d Informat” on page 334</td>
<td>Reads the monetary format of the international expression for the United Arab Emirates.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIAUDw.d Informat” on page 335</td>
<td>Reads the monetary format of the international expression for Australia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIBGNw.d Informat” on page 336</td>
<td>Reads the monetary format of the international expression for Bulgaria.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIBRLw.d Informat” on page 337</td>
<td>Reads the monetary format of the international expression for Brazil.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICADw.d Informat” on page 338</td>
<td>Reads the monetary format of the international expression for Canada.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICHFW.d Informat” on page 339</td>
<td>Reads the monetary format of the international expression for Liechtenstein and Switzerland.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICNYw.d Informat” on page 340</td>
<td>Reads the monetary format of the international expression for China.</td>
</tr>
<tr>
<td></td>
<td>“NLMNICZKW.d Informat” on page 341</td>
<td>Reads the monetary format of the international expression for the Czech Republic.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIDKKW.d Informat” on page 342</td>
<td>Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEEKW.d Informat” on page 343</td>
<td>Reads the monetary format of the international expression for Estonia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEGPW.d Informat” on page 344</td>
<td>Reads the monetary format of the international expression for Egypt.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIEURW.d Informat” on page 345</td>
<td>Reads the monetary format of the international expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIGBPW.d Informat” on page 346</td>
<td>Reads the monetary format of the international expression for the United Kingdom.</td>
</tr>
<tr>
<td>Category</td>
<td>Informs for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>“NLMNIHKDw.d Informat” on page 347</td>
<td>Reads the monetary format of the international expression for Hong Kong.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIHRKw.d Informat” on page 348</td>
<td>Reads the monetary format of the international expression for Croatia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIHUFw.d Informat” on page 349</td>
<td>Reads the monetary format of the international expression for Hungary.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIIDRw.d Informat” on page 350</td>
<td>Reads the monetary format of the international expression for Indonesia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIILSw.d Informat” on page 351</td>
<td>Reads the monetary format of the international expression for Israel.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIINRw.d Informat” on page 352</td>
<td>Reads the monetary format of the international expression for India.</td>
</tr>
<tr>
<td></td>
<td>“NLMNJJPYw.d Informat” on page 353</td>
<td>Reads the monetary format of the international expression for Japan.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIKRWw.d Informat” on page 354</td>
<td>Reads the monetary format of the international expression for South Korea.</td>
</tr>
<tr>
<td></td>
<td>“NLMNILTLw.d Informat” on page 355</td>
<td>Reads the monetary format of the international expression for Lithuania.</td>
</tr>
<tr>
<td></td>
<td>“NLMNILVLw.d Informat” on page 356</td>
<td>Reads the monetary format of the international expression for Latvia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIMOPw.d Informat” on page 357</td>
<td>Reads the monetary format of the international expression for Macau.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIMXNw.d Informat” on page 358</td>
<td>Reads the monetary format of the international expression for Mexico.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIMYRw.d Informat” on page 359</td>
<td>Reads the monetary format of the international expression for Malaysia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIKoRw.d Informat” on page 360</td>
<td>Reads the monetary format of the international expression for Norway.</td>
</tr>
<tr>
<td></td>
<td>“NLMNINZDw.d Informat” on page 361</td>
<td>Reads the monetary format of the international expression for New Zealand.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIPLNw.d Informat” on page 362</td>
<td>Reads the monetary format of the international expression for Poland.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIROLw.d Informat” on page 363</td>
<td>Reads the monetary format of the international expression for Romania.</td>
</tr>
<tr>
<td></td>
<td>“NLMNIKUlw.d Informat” on page 364</td>
<td>Reads the monetary format of the international expression for Russia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNISEKw.d Informat” on page 365</td>
<td>Reads the monetary format of the international expression for Sweden.</td>
</tr>
<tr>
<td></td>
<td>“NLMNISGDw.d Informat” on page 366</td>
<td>Reads the monetary format of the international expression for Singapore.</td>
</tr>
<tr>
<td></td>
<td>“NLMNISTw.d Informat” on page 367</td>
<td>Reads the monetary format of the international expression for Slovenia.</td>
</tr>
<tr>
<td></td>
<td>“NLMNISKKw.d Informat” on page 368</td>
<td>Reads the monetary format of the international expression for Slovakia.</td>
</tr>
<tr>
<td>Category</td>
<td>Informs for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“NLMNITHBw.d” Informat” on page 369</td>
<td>Reads the monetary format of the international expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>“NLMNITRYw.d Informat” on page 370</td>
<td>Reads the monetary format of the international expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>“NLMNITWDw.d” Informat” on page 371</td>
<td>Reads the monetary format of the international expression for Taiwan.</td>
<td></td>
</tr>
<tr>
<td>“NLMNIUSDw.d Informat” on page 372</td>
<td>Reads the monetary format of the international expression for Puerto Rico and the United States.</td>
<td></td>
</tr>
<tr>
<td>“NLMNIZARw.d Informat” on page 373</td>
<td>Reads the monetary format of the international expression for South Africa.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCAEDw.d Informat” on page 374</td>
<td>Reads the monetary format of the local expression for the United Arab Emirates.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLAUDw.d Informat” on page 375</td>
<td>Reads the monetary format of the local expression for Australia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLBGNw.d Informat” on page 376</td>
<td>Reads the monetary format of the local expression for Bulgaria.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLBRLw.d Informat” on page 377</td>
<td>Reads the monetary format of the local expression for Brazil.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCADw.d Informat” on page 378</td>
<td>Reads the monetary format of the local expression for Canada.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCHFw.d Informat” on page 379</td>
<td>Reads the monetary format of the local expression for Liechtenstein and Switzerland.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCNYw.d Informat” on page 380</td>
<td>Reads the monetary format of the local expression for China.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCHKw.d Informat” on page 381</td>
<td>Reads the monetary format of the local expression for the Czech Republic.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLDKKw.d Informat” on page 382</td>
<td>Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLNIRw.d Informat” on page 383</td>
<td>Reads the monetary format of the local expression for Estonia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLGBPw.d Informat” on page 384</td>
<td>Reads the monetary format of the local expression for Egypt.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLCZKw.d Informat” on page 385</td>
<td>Reads the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLGBPw.d Informat” on page 386</td>
<td>Reads the monetary format of the local expression for the United Kingdom.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHKDw.d Informat” on page 387</td>
<td>Reads the monetary format of the local expression for Hong Kong.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHRKw.d Informat” on page 388</td>
<td>Reads the monetary format of the local expression for Croatia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNHLHUFw.d Informat” on page 389</td>
<td>Reads the monetary format of the local expression for Hungary.</td>
<td></td>
</tr>
<tr>
<td>Category</td>
<td>Informs for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
</tr>
<tr>
<td>“NLMNLIDRw.d Informat” on page 390</td>
<td>Reads the monetary format of the local expression for Indonesia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLILSw.d Informat” on page 391</td>
<td>Reads the monetary format of the local expression for Israel.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLINRw.d Informat” on page 392</td>
<td>Reads the monetary format of the local expression for India.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLPYw.d Informat” on page 393</td>
<td>Reads the monetary format of the local expression for Japan.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLKRw.d Informat” on page 394</td>
<td>Reads the monetary format of the local expression for South Korea.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLITw.d Informat” on page 395</td>
<td>Reads the monetary format of the local expression for Lithuania.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLLVw.d Informat” on page 396</td>
<td>Reads the monetary format of the local expression for Latvia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLMOPw.d Informat” on page 397</td>
<td>Reads the monetary format of the local expression for Macau.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLMXw.d Informat” on page 398</td>
<td>Reads the monetary format of the local expression for Mexico.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLMYw.d Informat” on page 399</td>
<td>Reads the monetary format of the local expression for Malaysia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLNOKw.d Informat” on page 400</td>
<td>Reads the monetary format of the local expression for Norway.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLNZw.d Informat” on page 401</td>
<td>Reads the monetary format of the local expression for New Zealand.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLPw.d Informat” on page 402</td>
<td>Reads the monetary format of the local expression for Poland.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLROLw.d Informat” on page 403</td>
<td>Reads the monetary format of the local expression for Romania.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLREBw.d Informat” on page 404</td>
<td>Reads the monetary format of the local expression for Russia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLSKw.d Informat” on page 405</td>
<td>Reads the monetary format of the local expression for Sweden.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLSGDw.d Informat” on page 406</td>
<td>Reads the monetary format of the local expression for Singapore.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLSITw.d Informat” on page 407</td>
<td>Reads the monetary format of the local expression for Slovenia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLSKKw.d Informat” on page 408</td>
<td>Reads the monetary format of the local expression for Slovakia.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLTHBw.d Informat” on page 409</td>
<td>Reads the monetary format of the local expression for Thailand.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLYTRw.d Informat” on page 410</td>
<td>Reads the monetary format of the local expression for Turkey.</td>
<td></td>
</tr>
<tr>
<td>“NLMNLTWw.d Informat” on page 411</td>
<td>Reads the monetary format of the local expression for Taiwan.</td>
<td></td>
</tr>
</tbody>
</table>
$CPTDWw. 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

$CPTDWw.

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 $CPTWDw. informat.

Examples

The following example uses the input value of 808182.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x=input('808182',cptdw6.);</code></td>
<td></td>
</tr>
<tr>
<td><code>put x;</code></td>
<td>128</td>
</tr>
</tbody>
</table>

See Also

Formats:

“$CPTDW w. Format” on page 75
“$CPTWDw. Format” on page 76

Informats:

“$CPTWDw. Informat” on page 319

$CPTWDw. 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

$CPTWDw.

Syntax Description

\(w\)

specifies the width of the input field.

Default: 200

Range: 1–32000

Comparisons

The $CPTWDw. informat performs processing that is opposite of the $CPTDW w. informat.
Examples

The following example uses the input value of 1234.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x=input (’1234’,$cptwd6.);</td>
<td>[1234]</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
- “$CPTWDw. Format” on page 76
- “$CPTDw. Format” on page 75

Informats:
- “$CPTDw. Informat” on page 318

EUROw.d Informat

Reads numeric values, removes embedded characters in European currency, and reverses the comma and decimal point.

Category: Numeric

Syntax

EUROw.d

Syntax Description

\[w\]

specifies the width of the input field.

Default: 6

Range: 1–32

\[d\]

specifies the power of 10 by which to divide the value. If the data contains decimal points, the \[d\] value is ignored.
Informats for NLS

Default: 0
Range: 0–31

Details
The EUROw.d informat reads numeric values and removes embedded euro symbols (E), commas, blanks, percent signs, dashes, and close parentheses from the input data. A decimal point is assumed to be a separator between the whole number and the decimal portion. The EUROw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons
- The EUROw.d informat is similar to the EUROXw.d informat, but EUROXw.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 EUROw.d and the EUROXw.d informats are interchangeable.

Examples
The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>input x euro10.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>E1.23</td>
<td>input x euro10.;</td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1.23</td>
<td>input x euro10.;</td>
<td>1.23</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>1,234.56</td>
<td>input x euro10.;</td>
<td>1234.56</td>
</tr>
<tr>
<td></td>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Formats:
- “EUROw.d Format” on page 77
- “EUROXw.d Format” on page 78
Informats:
- “EUROXw.d Informat” on page 322
EUROXw.d Informat

Reads numeric values and removes embedded characters in European currency.
Category: Numeric

Syntax
EUROXw.d

Syntax Description

w
specifies the width of the input field.
Default: 6
Range: 1–32

d
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 EUROXw.d informat reads numeric values and removes embedded euro symbols (€), periods, blanks, percent signs, dashes, and close parentheses from the input data. A comma is assumed to be a separator between the whole number and the decimal portion. The EUROXw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

Comparisons

□ The EUROXw.d informat is similar to the EUROw.d informat, but EUROw.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 EUROXw.d and the EUROw.d informats are interchangeable.

Examples

The following table shows input values for currency in euros, the SAS statements that are applied, and the results.

<table>
<thead>
<tr>
<th>Values</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>€1</td>
<td>input x euro10.; put x;</td>
<td>1</td>
</tr>
<tr>
<td>€1.23</td>
<td>input x euro10.; put x;</td>
<td>123</td>
</tr>
</tbody>
</table>
Informats for NLS

JDATEYMD <w> Informat

Reads Japanese kanji date values in the format yymmmdd or yyyyymmmd.

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 yyyyymmmd.

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 data lines 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.

```sas
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:

“JNENGOw. Informat” on page 324

System Options:

YEARCUTOFF= in SAS Language Reference: Dictionary

---

**JNENGOw. Informat**

Reads Japanese kanji date values in the form **yymmdd**.

Category: Date and Time

Alignment: left

**Syntax**

JNENGOw.
Syntax Description

\[ w \]

specifies the width of the output field.

**Default:** 16

**Range:** 16–32

Details

The **JNENGO**. informat reads Japanese kanji values in the form `ymmd`. 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 data lines 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.

```sas
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**. Informat” on page 323

System Options:

**YEARCUTOFF=** in **SAS Language Reference: Dictionary**
$\textit{KANJI}w. \textbf{Informat}

Removes shift code data from DBCS data.

Category: DBCS

\textbf{Syntax}

$\textit{KANJI}w.$

\textbf{Syntax Description}

$w$

specifies the width of the input field.

\textbf{Restriction:} The width must be an even number. If it is an odd number, it is truncated. The width must be equal to or greater than the length of the shift-code data.

\textbf{Range:} The minimum width for the informat is 2.

\textbf{Details}

The $\textit{KANJI}$ informat removes shift-code data from DBCS data. The $\textit{KANJI}$ informat processes host-mainframe data. $\textit{KANJI}$ can be used on other platforms. If you use the $\textit{KANJI}$ informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

The data must start with SO and end with SI, unless single-byte blank data are returned. The input data length must be $2 + (\text{SO/SI length}) \times 2$.

\textbf{See Also}

Formats:

“$\textit{KANJI}w. \text{Format}$” on page 82

“$\textit{KANJI}Xw. \text{Format}$” on page 83

Informats:

“$\textit{KANJI}Xw. \text{Informat}$” on page 326

---

$\textit{KANJIX}w. \textbf{Informat}$

Adds shift-code data to DBCS data.

Category: DBCS

\textbf{Syntax}

$\textit{KANJIX}w.$
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. The width must be equal to or greater than the length of the shift-code data.

**Range:** The minimum width for the informat is \( 2 + (\text{length of shift code used on the current DBCSTYPE= setting}) \times 2 \).

Details

The $KANJIX$ informat adds shift-code data to DBCS data that does not have shift-code data. If the input data is blank, shift-code data is not added. The $KANJIX$ informat processes host-mainframe data, but $KANJIX$ can be used on other platforms. If you use the $KANJIX$ informat on non-EBCDIC (non-modal encoding) hosts, the data does not change.

See Also

Formats:

- “$KANJ\text{I}w$. Format” on page 82
- “$KANJ\text{IX}w$. Format” on page 83

Informats:

- “$KANJ\text{I}w$. Informat” on page 326

$LOGVS\text{w}$. 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**

$LOGVS\text{w}$.  

**Syntax Description**

\( w \)

specifies the width of the input field.

**Default:** 200

**Range:** 1–32000
Comparisons

The $LOGVS\text{w}$. informat performs processing that is opposite to the LOGVSR\text{w}. informat.

Examples

The following example uses the Hebrew input value of “טאמילג flight.”

Statements | Result
--- | ---
$x=$input (’טאמילג flight’,$logvs12.);
put $x$; | טאמילג flight

The following example uses the Arabic input value of “تاذاً computer.”

Statements | Result
--- | ---
$x=$input (’تاذاً computer’,$logvs12.);
put $x$; | تاذاً computer

See Also

Formats:

“$LOGVSR\text{w}$. Format” on page 85
“$LOGVS\text{w}$. Format” on page 84

Informats:

“$LOGVSR\text{w}$. Informat” on page 328

$LOGVSR\text{w}$. 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

$\text{LOGVSR}\text{w}$.
Syntax Description

\[ w \]

specifies the width of the input field.

**Default:** 200

**Range:** 1–32000

Comparisons

The \$LOGVSRw. informat performs processing that is opposite to the \$LOGVSw. informat.

Examples

The following example uses the Hebrew input value of “_flight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x=\text{input ('ףיעס',ilogvsr12.);} ] put [x;]</td>
<td>\text{flight} \text{ףיעס}</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “ضآر computer.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>[x=\text{input ('ضآر',ilogvsr12.);} ] put [x;]</td>
<td>\text{ضآر} \text{computer}</td>
</tr>
</tbody>
</table>

See Also

Formats:

“\$LOGVS\text{w. Format}” on page 84

“\$LOGVSR\text{w. Format}” on page 85

Informats:

“\$LOGVS\text{w. Informat}” on page 327
**MINGUOw. Informat**

Reads dates in Taiwanese format.

**Category:** Date and Time

---

**Syntax**

MINGUOw.

**Syntax Description**

\( w \)

specifies the width of the input field.

**Default:** 6

**Range:** 6–10

---

**Details**

The general form of a Taiwanese date is \( yyyyymmdd \):

- \( yyyy \)
  - 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 before 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 YYMDDw. 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.;
```
### See Also

Formats:
- “MINGUOw. Format” on page 86
Informats:
- YYMMDDw. in *SAS Language Reference: Dictionary*

### NENGOw. Informat

Reads Japanese date values in the form `eyymmdd`.

**Category:** Date and Time

**Syntax**

```
NENGOw.
```

**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.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>h11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>h.11108</td>
<td>08OCT1999</td>
</tr>
<tr>
<td>11/10/08</td>
<td>08OCT1999</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NENGOw. Format” on page 87

---

**NLDATEw. 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**

NLDATEw.

**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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates; y=input('February 24, 2003', nldate17.); put y=nldate.;</td>
<td>y=February 24, 2003</td>
</tr>
<tr>
<td>options locale=German_Germany; y=input('24. Februar 2003', nldate16.); put y=nldate;</td>
<td>y=24. Februar 2003</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLDATEw. Format” on page 91

NLDATMw. 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

NLDATMw.

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**

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>y=input('24.Feb03:12:39:43', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td>1361709583</td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>y=input('24.Februar 2003 12.39 Uhr', nldatm.);</td>
<td></td>
</tr>
<tr>
<td>put y=;</td>
<td>1330171200</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLDATMw. Format” on page 99

---

**NLMNIAEDw.d Informat**

Reads the monetary format of the international expression for the United Arab Emirates.

**Category:** Numeric

**Alignment:** left

**Syntax**

`NLMNIAEDw.d`

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - Default: 9
  - Range: 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - Default: 0
  - Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

x=input('($12,345.67)',nlmiaed32.2);
y=input('($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informat:
“NLMNLAEDw.d Informat” on page 374

NLMNIAUDw.d Informat

Reads the monetary format of the international expression for Australia.

Category:  Numeric
Alignment: left

Syntax

NLMNIAUDw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
$x$=input('($12,345.67)$',nlmniaud32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put $x$=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put $y$=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:

“NLMNLAUD\textit{w.d} Informat” on page 375

---

**NLMNIBGN\textit{w.d} Informat**

Reads the monetary format of the international expression for Bulgaria.

**Category:** Numeric

**Alignment:** left

**Syntax**

\texttt{NLMNIBGN\textit{w.d}}

**Syntax Description**

$w$

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

$d$

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the \texttt{LOCALE=} system option is set to English\_United\_States.
x=put(-1234.56789,nlmnibgn32.2);
y=put(-1234.56789,dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NLMNLBGNw.d Informat” on page 376

**NLMNIBRLw.d Informat**

Reads the monetary format of the international expression for Brazil.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNIBRLw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)\',nlmnibrl32.2);
y=input('($12,345.67)\'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NLMNLBRLw.d Informat” on page 377

**NLMNICADw.d Informat**

Reads the monetary format of the international expression for Canada.

Category: Numeric

Alignment: left

**Syntax**

NLMNICADw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

Default: 9

Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0

Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informs for NLS △ NLMNICHFW.d Informat 339

\[ \text{x=input'}(\$12,345.67)',nlmniaud32.2); \]
\[ \text{y=input'}(\$12,345.67)\text{dollar32.2);} \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNICADw.d Format” on page 114

NLMNICHFW.w.d Informat

Reads the monetary format of the international expression for Liechtenstein and Switzerland.
Category: Numeric
Alignment: left

Syntax

NLMNICHFW.w.d

Syntax Description

\[ w \]
specifies the width of the output field.
Default: 9
Range: 1–32

\[ d \]
specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input’($12,345.67),nlnmicf32.2);  
y=input’($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNICHW.d Format” on page 115

NLMNICNYw.d Informat

Reads the monetary format of the international expression for China.

Category: Numeric

Alignment: left

Syntax

NLMNICNYw.d

Syntax Description

w
specifies the width of the output field.

Default: 9

Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 0

Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnicny32.2);
y=input'($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“NLMNICNYw.d Format” on page 116

---

**NLMNICZKw.d Informat**

Reads the monetary format of the international expression for the Czech Republic.

Category: Numeric
Alignment: left

**Syntax**

NLMNICZKw.d

**Syntax Description**

\(w\)

specifies the width of the output field.
- Default: 9
- Range: 1–32

\(d\)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
- Default: 0
- Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$,nlmniczk32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNLCZKw.d Informat” on page 381

**NLMNIDKKw.d Informat**

Reads the monetary format of the international expression for Denmark, Faroe Island, and Greenland.

Category: Numeric
Alignment: left

**Syntax**

NLMNIDKKw.d

**Syntax Description**

\[ w \]

specifies the width of the output field.
Default: 9
Range: 1–32

\[ d \]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.
Default: 0
Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)$,nlmniaud32.2);
y=input'($12,345.67)$'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

### See Also

Formats:

“NLMNIDKKw.d Format” on page 118

### NLMNIEEKw.d Informat

Reads the monetary format of the international expression for Estonia.

Category: Numeric

Alignment: left

### Syntax

NLMNIEEKw.d

### Syntax Description

**w**

specifies the width of the output field.

Default: 9

Range: 1–32

**d**

specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.

Default: 0

Range: 0–31

### Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)’,nlmnieek32.2);
y=input('($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNLEEkw.d Informat” on page 383

NLMNIEGPw.d Informat

Reads the monetary format of the international expression for Egypt.

Category: Numeric
Alignment: left

Syntax

NLMNIEGPw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$,nlmniegp32.2);
y=input('($12,345.67)$dollar32.2);

Statements Results
----+----1----+
put x=; -12345.67
put y=; -12345.67

See Also

Informats:
“NLMNLEGp.w.d Informat” on page 384

NLMNIEURw.d Informat

Reads the monetary format of the international expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

Category: Numeric
Alignment: left

Syntax
NLMNIEURw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informat/NLMNIGBPw.d

**Syntax**

NLMNIGBPw.d

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32
- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0
  - **Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=input'($12,345.67)',nlmnieur32.2);
y=input'($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIEURw.d Format” on page 121
x=input'($12,345.67)',nlmngbp32.2);
y=input'($12,345.67)‘dollar32.2);

Statements                   Results
----+----1----+
put x=;         -12345.67
put y=;         -12345.67

See Also

Formats:
“NLMNIGBPw.d Format” on page 122

**NLMNIIHKDw.d Informat**

Reads the monetary format of the international expression for Hong Kong.

Category: Numeric
Alignment: left

**Syntax**

NLMNIIHKDw.d

**Syntax Description**

\[w\]

specifies the width of the output field.

Default: 9
Range: 1–32

\[d\]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnihrkd32.2);
y=input'($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIHKDw.d Format” on page 123

---

**NLMNIHRKw.d Informat**

Reads the monetary format of the international expression for Croatia.

*Category:* Numeric  
*Alignment:* left  

**Syntax**

NLMNIHRKw.d  

**Syntax Description**

**w**

specifies the width of the output field.  
**Default:** 9  
**Range:** 1–32

**d**

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.  
**Default:** 0  
**Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$nlmnihrk32.2);
y=input('($12,345.67)$dollar32.2);

Statements Results

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informs:
“NLMNLHRKw.d Informat” on page 388

NLMNIHUFw.d Informat

Reads the monetary format of the international expression for Hungary.

Category: Numeric
Alignment: left

Syntax

NLMNIHUFw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$',nlmnihuf32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informs:

“NLMNLHUFw.d Informat” on page 389

---

**NLMNIIIDRw.d Informat**

Reads the monetary format of the international expression for Indonesia.

Category: Numeric
Alignment: left

**Syntax**

NLMNIIIDRw.d

**Syntax Description**

\( w \)

specifies the width of the output field.

Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default: 0
Range: 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informs for NLS  △  NLMNILLSw.d Informat

\[ x=\text{input}('($12,345.67)$',\text{nlmniidr32.2}); \]
\[ y=\text{input}('($12,345.67)$'\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNLIDRw.d Informat” on page 390

NLMNILLSw.d Informat

Reads the monetary format of the international expression for Israel.

Category:  Numeric
Alignment:  left

Syntax

NLMNILLSw.d

Syntax Description

\[ w \]

specifies the width of the output field.

Default:  9
Range:  1–32

\[ d \]

specifies to divide the number by \(10^d\). If the data contains decimal points, the \(d\) value is ignored.

Default:  0
Range:  0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmniils32.2);
y=input'($12,345.67)'dollar32.2);

Statements                Results
                              ------+----1----+
put x=;                      -12345.67
put y=;                      -12345.67

See Also

Formats:
   “NLMNIILSw.d Format” on page 127

NLMNIINRw.d Informat

Reads the monetary format of the international expression for India.

Category:   Numeric
Alignment:  left

Syntax

NLMNIINRw.d

Syntax Description

w
   specifies the width of the output field.
   Default:  9
   Range:    1–32

d
   specifies to divide the number by $10^d$. If the data contains decimal points, the d value is ignored.
   Default:  0
   Range:    0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.


```plaintext
x=input('($12,345.67)\',nlniinr32.2);
y=input('($12,345.67)\'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:

“NLMNLINRw.d Informat” on page 392

---

**NLMNJPYw.d Informat**

Reads the monetary format of the international expression for Japan.

*Category:* Numeric  
*Alignment:* left

**Syntax**

NLMNJPYw.d

**Syntax Description**

- **w**
  - Specifies the width of the output field.
  - **Default:** 9  
  - **Range:** 1–32

- **d**
  - Specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
  - **Default:** 0  
  - **Range:** 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnijpy32.2);
y=input'($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIJPYw.d Format” on page 129

NLMNIKRWw.d Informat

Reads the monetary format of the international expression for South Korea.

Category: Numeric
Alignment: left

Syntax

NLMNIKRWw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input’($12,345.67)’,nlmnikrw32.2);  
y=input’($12,345.67)’dollar32.2);

See Also

Informs:

“NLMNLKRWw.d Informat” on page 394

NLMNILT Lw.d Informat

Reads the monetary format of the international expression for Lithuania.

Category: Numeric
Alignment: left

Syntax

NLMNILT Lw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
$x=input('($12,345.67)$',nlmnilt{32.2});$
$y=input('($12,345.67)$dollar{32.2});$

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put $x=$;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put $y=$;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informs:

“NLMNLLTLw.d Informat” on page 395

---

**NLMNILVLw.d Informat**

Reads the monetary format of the international expression for Latvia.

**Category:** Numeric

**Alignment:** left

---

**Syntax**

NLMNILVLw.d

**Syntax Description**

$w$

specifies the width of the output field.

**Default:** 9

**Range:** 1–32

$d$

specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

**Default:** 0

**Range:** 0–31

**Examples**

In the following example, the `LOCALE=` system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnilv32.2);
y=input('($12,345.67)\$32.2');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informs:
“NLMNLLVLw.d Informat” on page 396

NLMNIMOPw.d Informat

Reads the monetary format of the international expression for Macau.

Category: Numeric
Alignment: left

Syntax

NLMNIMOPw.d

Syntax Description

\( w \)

specifies the width of the output field.
Default: 9
Range: 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$,nlmnmop32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informs:

“NLMNLMOpw. d Informat” on page 397

NLMNIMXNw.d Informat

Reads the monetary format of the international expression for Mexico.

Category: Numeric

Alignment: left

Syntax

NLMNIMXNw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 9

Range: 1–32

\( d \)

specifies to divide the number by \( 10^d \). If the data contains decimal points, the \( d \) value is ignored.

Default: 0

Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

x=input('($12,345.67)',nlmnimxn32.2);
y=input('($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNLMXNw.d Informat” on page 398

NLMNIMYRw.d Informat

Reads the monetary format of the international expression for Malaysia.

Category: Numeric
Alignment: left

Syntax

NLMNIMYRw.d

Syntax Description

w
specifies the width of the output field.
Default: 9
Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Statements | Results
---|---
put x=; | -12345.67
put y=; | -12345.67

See Also

Formats:
“NLMNIMYRw.d Format” on page 135

NLMNINOK\textit{w.d} Informat

Reads the monetary format of the international expression for Norway.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNINOKw.d}

Syntax Description

\texttt{w}

specifies the width of the output field.

Default: 9
Range: 1–32

\texttt{d}

specifies to divide the number by $10^d$. If the data contains decimal points, the \texttt{d} value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
x=input'($12,345.67)',nlmninok32.2);
y=input'($12,345.67)'dollar32.2);

Statements Results
------+------+
put x=; -12345.67
put y=; -12345.67

See Also

Formats:
“NLMNINOKw.d Format” on page 136

NLMNINZDw.d Informat

Reads the monetary format of the international expression for New Zealand.

Category: Numeric
Alignment: left

Syntax

NLMNINZDw.d

Syntax Description

w
specifies the width of the output field.
  Default: 9
  Range: 1–32

d
specifies to divide the number by 10^d. If the data contains decimal points, the d value is ignored.
  Default: 0
  Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmniaud32.2);
y=input'($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNINZDw.d Format” on page 137

NLMNIPLNw.d Informat

Reads the monetary format of the international expression for Poland.

Category: Numeric

Alignment: left

Syntax

NLMNIPLNw.d

Syntax Description

w
specifies the width of the output field.

Default: 9
Range: 1–32

d
specifies to divide the number by $10^d$. If the data contains decimal points, the $d$ value is ignored.

Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnipln32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNIPLNw.d Format” on page 138

NLMNIROLw.d Informat

Reads the monetary format of the international expression for Romania.

Category: Numeric
Alignment: left

Syntax

NLMNIROLw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnirol32.2);
y=input('($12,345.67)''dollar32.2');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:

“NLMNLROLw.d Informat” on page 403

**NLMNIRUBw.d Informat**

Reads the monetary format of the international expression for Russia.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNIRUBw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)','nlmnirub32.2);
y=input'($12,345.67)\'dollar32.2);

Statements  Results

----+----1----+
put x=;    -12345.67
put y=;    -12345.67

See Also

Formats:
“NLMNIRUBw.d Format” on page 140

NLMNISEKw.d Informat

Reads the monetary format of the international expression for Sweden.

Category:  Numeric
Alignment:  left

Syntax

NLMNISEKw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnisek32.2);
y=input'($12,345.67)''dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNISEKw.d Format” on page 141

NLMNISGDw.d Informat

Reads the monetary format of the international expression for Singapore.

Category: Numeric
Alignment: left

Syntax

NLMNISGDw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informat for NLS

Informats for NLS  △  NLMNISITw.d Informat  367

\begin{verbatim}
x=\text{input'}('($12,345.67)',nlmnisd32.2);};
y=\text{input'}('($12,345.67)'dollar32.2);};
\end{verbatim}

\begin{tabular}{l|l}
\textbf{Statements} & \textbf{Results} \\
\hline
put \texttt{x=} & -12345.67 \\
pay \texttt{y=} & -12345.67 \\
\hline
\end{tabular}

\section*{See Also}

Formats:
“\textit{NLMNISGDw.d Format}” on page 142

\section*{NLMNISITw.d Informat}

Reads the monetary format of the international expression for Slovenia.

\begin{itemize}
\item \textbf{Category:} Numeric
\item \textbf{Alignment:} left
\end{itemize}

\section*{Syntax}

NLMNISIT\textit{w.d}

\section*{Syntax Description}

\begin{itemize}
\item \textit{w} specifies the width of the output field.
  \begin{itemize}
  \item \textbf{Default:} 9
  \item \textbf{Range:} 1–32
  \end{itemize}
\item \textit{d} optionally specifies to divide the number by \textit{10\textsuperscript{d}}. If the data contains decimal points, the \textit{d} value is ignored.
  \begin{itemize}
  \item \textbf{Default:} 0
  \item \textbf{Range:} 0–31
  \end{itemize}
\end{itemize}

\section*{Examples}

In the following example, the \texttt{LOCALE=} system option is set to English_UnitedStates.
x=input('$(12,345.67)$',nlmnisit2);
y=input('$(12,345.67)$'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:  
“NLMNLSITw.d Informat” on page 407

**NLMNISKKw.d Informat**

Reads the monetary format of the international expression for Slovakia.

**Category:** Numeric  
**Alignment:** left

**Syntax**

NLMNISKK\(w.d\)

**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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

$\text{NLMNITHB}w.d$ Informat

Reads the monetary format of the international expression for Thailand.

Category: Numeric
Alignment: left

Syntax

$\text{NLMNITHB}w.d$

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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```bash
x=input('($12,345.67)$',nlmniskk2);
y=input('($12,345.67)\text{dollar32.2}$);

Statements                  Results
----+----+----+
put x=;                      -12345.67
put y=;                      -12345.67
```

See Also

Informats:
“NLMNLSKKw.d Informat” on page 408
x=input('
$12,345.67',nlmnithb2);
y=input('
$12,345.67',dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNLTHBw.d Informat” on page 409

NLMNTRYw.d Informat

Reads the monetary format of the international expression for Turkey.

Category: Numeric
Alignment: left

Syntax

NLMNTRYw.d

Syntax Description

$w$
 specifies the width of the output field.
  Default: 9
  Range: 1–32

$dd$
 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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

NLMNITWD<sub>w.d</sub> Informat

Reads the monetary format of the international expression for Taiwan.

**Category:** Numeric

**Alignment:** left

### Syntax

NLMNITWD<sub>w.d</sub>

### Syntax Description

- **w**
  - Specifies the width of the output field.
  - **Default:** 9
  - **Range:** 1–32

- **d**
  - Optionally specifies to divide the number by 10<sup>d</sup>. If the data contains decimal points, the <i>d</i> value is ignored.
  - **Default:** 0
  - **Range:** 0–31

### Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

```plaintext
x=input('($12,345.67)',nlmnitry32.2);
y=input('($12,345.67')dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
```

See Also

Informats:

“NLMNLTRY<sub>w.d</sub> Informat” on page 410
x=input'($12,345.67)', nlmnitwd32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNITWd\textit{w.d} Format” on page 147

\textbf{NLMNUSD\textit{w.d} Informat}

Reads the monetary format of the international expression for Puerto Rico and the United States.

\textbf{Category:} Numeric
\textbf{Alignment:} left

\textbf{Syntax}

\texttt{NLMNUSD\textit{w.d}}

\textbf{Syntax Description}

\textit{w}

specifies the width of the output field.
\textbf{Default:} 9
\textbf{Range:} 1–32

\textit{d}

optionally specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.
\textbf{Default:} 0
\textbf{Range:} 0–31

\textbf{Examples}

In the following example, the \texttt{LOCALE=} system option is set to English\_UnitedStates.
x=input’($12,345.67)’,nlmniusd32.2);
y=input’($12,345.67)’dollar32.2);

Statements Results
-----+----1----+
put x=; -12345.67
put y=; -12345.67

See Also

Formats:
“NLMNIUSDw.d Format” on page 148

NLMNIZARw.d Informat

Reads the monetary format of the international expression for South Africa.

Category: Numeric
Alignment: left

Syntax

NLMNIZARw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnizar32.2);
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNIZARw.d Format” on page 149

---

**NLMNLAEDw.d Informat**

Reads the monetary format of the local expression for the United Arab Emirates.

Category: Numeric
Alignment: left

**Syntax**

NLMNLAEDw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

```plaintext
x=input('($12,345.67)',nlnlaid32.2);
y=input('($12,345.67)'dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NLMNAEDw.d Informat” on page 334

**NLMNAUDw.d Informat**

Reads the monetary format of the local expression for Australia.

**Category:** Numeric

**Alignment:** left

**Syntax**

**NLMNAUDw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlnlaud32.2);  
y=input'($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNLAUDw.d Format” on page 151

---

**NLMNLBGNw.d Informat**

Reads the monetary format of the local expression for Bulgaria.

Category: Numeric

Alignment: left

**Syntax**

NLMNLBGNw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
In the following example, the LOCALE= system option is set to English_UnitedStates.

```
x=put(-1234.56789,nlmlbgn32.2);
y=put(-1234.56789,dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informs:
“NLMNIBGNw.d Informat” on page 336

**NLMNLBRLw.d Informat**

Reads the monetary format of the local expression for Brazil.

*Category:* Numeric

*Alignment:* left

**Syntax**

NLMNLBRLw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$',nlmnlbrl32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNIBRLw.d Informat” on page 337

NLMNLCADw.d Informat

Reads the monetary format of the local expression for Canada.

Category: Numeric
Alignment: left

Syntax

NLMNLCADw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnlcad32.2);  
y=input'($12,345.67)'
dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:  
“NLMNLCADw.d Format” on page 154

NLMNLCHFw.d Informat

Reads the monetary format of the local expression for Liechtenstein and Switzerland.

Category: Numeric  
Alignment: left

Syntax

NLMNLCHFw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmlchf32.2);
y=input'($12,345.67)\'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLCHFw.d Format” on page 155

**NLMNLCHFw.d Informat**

Reads the monetary format of the local expression for China.

**Category:** Numeric

**Alignment:** left

**Syntax**

NLMNLCHFw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

NLMNLCZK<sup>w</sup>.<sup>d</sup> Informat

Reads the monetary format of the local expression for the Czech Republic.

Category: Numeric
Alignment: left

Syntax

NLMNLCZK<sup>w</sup>.<sup>d</sup>

Syntax Description

<sup>w</sup>

specifies the width of the output field.
Default: 9
Range: 1–32

<sup>d</sup>

optionally specifies to divide the number by 10<sup>d</sup>. If the data contains decimal points, the <sup>d</sup> value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input(’$(12,345.67)$’,nlmnlczk32.2);
y=input(’$(12,345.67)$’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNICZKw.d Informat” on page 341

NLMNLDDKw.d Informat

Reads the monetary format of the local expression for Denmark, the Faroe Island, and Greenland.

Category: Numeric
Alignment: left

Syntax

NLMNLDDKw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS  \( \triangle \)  NLMNLEEK\( w.d \) Informat  383

\[
x = \text{input}'(\$12,345.67)'\text{, nlmnlkk32.2});
\]
\[
y = \text{input}'(\$12,345.67)'\text{dollar32.2});
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLDKK\( w.d \) Format” on page 158

NLMNLEEK\( w.d \) Informat

Reads the monetary format of the local expression for Estonia.

Category: Numeric
Alignment: left

Syntax

NLMNLEEK\( w.d \)

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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
```plaintext
x=input('($12,345.67)$',nlmnlneek32.2);
y=input('($12,345.67)$dollar32.2);
```

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informs:

“NLMNIEEKw.d Informat” on page 343

---

**NLMNLEGPOw.d Informat**

Reads the monetary format of the local expression for Egypt.

Category:  Numeric

Alignment:  left

**Syntax**

NLMNLEGPOw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

\[ x = \text{input('($12,345.67)$',nlnlegp32.2)}; \]
\[ y = \text{input('($12,345.67)$dollar32.2)}; \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put ( x = );</td>
<td>(-12345.67)</td>
</tr>
<tr>
<td>put ( y = );</td>
<td>(-12345.67)</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNIEGPw.d Informat” on page 344

**NLMNLEURw.d Informat**

Reads the monetary format of the local expression for Austria, Belgium, Finland, France, Germany, Greece, Ireland, Italy, Luxembourg, Malta, the Netherlands, Portugal, Slovenia, and Spain.

**Category:** Numeric  
**Alignment:** left

**Syntax**

\texttt{NLMNLEURw.d}

**Syntax Description**

\( w \)

specifies the width of the output field.  
\textbf{Default:} 9  
\textbf{Range:} 1–32

\( d \)

optionally specifies to divide the number by 10\(^{d}\). If the data contains decimal points, the \( d \) value is ignored.  
\textbf{Default:} 0  
\textbf{Range:} 0–31

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)',nlmnlleur32.2);

y=input'($12,345.67)\$32.2');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
"NLMNLEURw.d Format" on page 161

NLMNGBPw.d Informat

Reads the monetary format of the local expression for the United Kingdom.

Category: Numeric

Alignment: left

Syntax

NLMNGBPw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLHKD\texttt{w.d} Informat

Reads the monetary format of the local expression for Hong Kong.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNLHKD\texttt{w.d}}

Syntax Description

\texttt{w} specifies the width of the output field.
\texttt{d} optionally specifies to divide the number by $10^d$. If the data contains decimal points, the \texttt{d} value is ignored.

Examples

In the following example, the \texttt{LOCALE=} system option is set to \texttt{English\_UnitedStates}.
x=input'($12,345.67)' , nlmnlhkd32.2);
y=input'($12,345.67)' , dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNLHKDw.d Format” on page 163

**NLMNLHRKw.d Informat**

Reads the monetary format of the local expression for Croatia.

**Category:** Numeric  
**Alignment:** left

**Syntax**

NLMNLHRKw.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

**Examples**

In the following example, the LOCATE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnlhrk32.2);
y=input('($12,345.67)\$dollar32.2');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-1,234.57</td>
</tr>
</tbody>
</table>

**See Also**

Informats:

“NLMNIHRK\$d Informat” on page 348

---

**NLMNLHUF\$d Informat**

Reads the monetary format of the local expression for Hungary.

Category: Numeric
Alignment: left

**Syntax**

NLMNLHUF\$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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnlhuf32.2);
y=input('($12,345.67)'dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NLMNIHUFw.d Informat” on page 349

NLMNLIDRw.d Informat

Reads the monetary format of the local expression for Indonesia.

Category: Numeric
Alignment: left

Syntax
NLMNLIDRw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)$nlmlidr32.2);
y=input('($12,345.67)$dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNIIDRw.d Informat” on page 350

NLMNLILSw.d Informat

Reads the monetary format of the local expression for Israel.

Category: Numeric
Alignment: left

Syntax

NLMNLILSw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input’($12,345.67),nlmnlils32.2);  
y=input’($12,345.67)‘dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLILS<w>.<d> Format” on page 167

NLMNLINRW<w>.<d> Informat

Reads the monetary format of the local expression for India.

Category: Numeric
Alignment: left

Syntax

NLMNLINRW<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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
Informats for NLS

\[ x=\text{input}(\text{‘($12,345.67$)\text{,nlmnlr32.2}'}); \]
\[ y=\text{input}(\text{‘($12,345.67$)\text{‘dollar32.2}'}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:
“NLMNIINR\textit{w.d} Informat” on page 352

---

**NLMNLJPY\textit{w.d} Informat**

Reads the monetary format of the local expression for Japan.

Category: Numeric
Alignment: left

**Syntax**

\[ \text{NLMNLJPY}\textit{w.d} \]

**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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
\[ x = \text{input'}(\$12,345.67)', \text{nlmnljpy32.2}); \]
\[ y = \text{input'}(\$12,345.67)'\text{dollar32.2}); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{put } x =; )</td>
<td>(-12345.67)</td>
</tr>
<tr>
<td>( \text{put } y =; )</td>
<td>(-12345.67)</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNLJPYw.d Format” on page 169

**NLMNLKRWw.d Informat**

Reads the monetary format of the local expression for South Korea.

**Category:** Numeric  
**Alignment:** left

**Syntax**

\[ \text{NLMNLKRW}w.d \]

**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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLLTLw.d Informat

Reads the monetary format of the local expression for Lithuania.

Category: Numeric
Alignment: left

Syntax

NLMNLLTLw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)’,nlmnlltl32.2);
y=input('($12,345.67)‘dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Informats:

“NLMNILTLw.d Informat” on page 355

---

**NLMNLLVLw.d Informat**

Reads the monetary format of the local expression for Latvia.

Category: Numeric

Alignment: left

**Syntax**

NLMNLLVLw.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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmn1lv32.2);
y=input('($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informs:

“NLMN1LV$w.d Informat” on page 356

NLMN1MV$w.d Informat

Reads the monetary format of the local expression for Macau.

Category: Numeric
Alignment: left

Syntax

NLMN1MV$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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
\[ x = \text{input('($12,345.67)$', nlmnlmop32.2)}; \\
\text{y} = \text{input('($12,345.67)$', dollar32.2)}; \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put x=} ;</td>
<td>\text{-12345.67}</td>
</tr>
<tr>
<td>\text{put y=} ;</td>
<td>\text{-12345.67}</td>
</tr>
</tbody>
</table>

\section*{See Also}

Informs:
“\text{NLMNIMOPw.d Informat}” on page 357

\section*{NLMNLMXNw.d Informat}

\text{Reads the monetary format of the local expression for Mexico.}

\text{Category: Numeric}

\text{Alignment: left}

\section*{Syntax}

\text{NLMNLMXNw.d}

\section*{Syntax Description}

\text{w}

specifies the width of the output field.

\text{Default: 9}
\text{Range: 1–32}

\text{d}

optionally specifies to divide the number by 10\(^d\). If the data contains decimal points, the \text{d} value is ignored.

\text{Default: 0}
\text{Range: 0–31}

\section*{Examples}

In the following example, the \text{LOCALE= system option} is set to \text{English UnitedStates}. 
x=input('($12,345.67)\',nlmnlmxn32.2);
y=input('($12,345.67)\'dollar32.2);

Statements Results

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNIMXNw.d Informat” on page 358

NLMNLMYRw.d Informat

Reads the monetary format of the local expression for Malaysia.

Category: Numeric

Alignment: left

Syntax

NLMNLMYRw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input'($12,345.67)' , nlmnlmyr32.2);
y=input'($12,345.67)' , dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLMYRw.d Format” on page 175

NLMNLNOKw.d Informat

Reads the monetary format of the local expression for Norway.

Category: Numeric
Alignment: left

Syntax

NLMNLNOKw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
**Informat for NLS**

The **NLMNLNZDw.d Informat** reads the monetary format of the local expression for New Zealand.

**Syntax**

\[ \text{NLMNLNZD}w.d \]

**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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.

\[
\begin{align*}
x &= \text{input'}(\$12,345.67)', \text{nlmnlnok32.2}); \\
y &= \text{input'}(\$12,345.67)', \text{dollar32.2}); \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

- Formats: “NLMNLNOKw.d Format” on page 176
x=input’($12,345.67)’,nlmnlnzdl2.2);  
y=input’($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLMNLNZDw.d Format” on page 177

NLMNLPLNw.d Informat

Reads the monetary format of the local expression for Poland.

Category: Numeric  
Alignment: left

Syntax

NLMNLPLNw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input‘($12,345.67)’,nlmnlpln32.2);  
y=input‘($12,345.67)’dollar32.2);

Statements Results

| put x=;                        | -12345.67 |
| put y=;                        | -12345.67 |

See Also

Formats:
“NLMNLPLNw.d Format” on page 178

NLMNLROLw.d Informat

Reads the monetary format of the local expression for Romania.

Category: Numeric

Alignment: left

Syntax

NLMNLROLw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input(‘($12,345.67)’,nlmnlrol32.2);
y=input(‘($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats: “NLMNIROLw.d Informat” on page 363

NLMNLURBw.d Informat

Reads the monetary format of the local expression for Russia.

Category: Numeric
Alignment: left

Syntax

NLMNLURBw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLEKw.d Informat

Reads the monetary format of the local expression for Sweden.

Category: Numeric

Alignment: left

Syntax

NLMNLEKw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.

x=input’($12,345.67)',nlmnlrub32.2);
y=input’($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLRUBw.d Format” on page 180
x = input'($12,345.67)', nlmnlsek32.2);
y = input'($12,345.67)' dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLSEK\textit{w.d} Format” on page 181

NLMNLSGD\textit{w.d} Informat

Reads the monetary format of the local expression for Singapore.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNLSGD\textit{w.d}}

Syntax Description

\textit{w}

specifies the width of the output field.
Default: 9
Range: 1–32

\textit{d}

optionally specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.
Default: 0
Range: 0–31

Examples

In the following example, the \texttt{LOCALE=} system option is set to English\_United\_States.
x=input’($12,345.67)’,nlmnlsgd32.2); 

y=input’($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNLSGDw.d Format” on page 182

NLMNLSTw.d Informat

Reads the monetary format of the local expression for Slovenia.

Category: Numeric

Alignment: left

Syntax

NLMNLSTw.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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnlssit32.2);
y=input('($12,345.67)’dollar32.2);

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNISItw.d Informat” on page 367

NLMNLSKKw.d Informat

Reads the monetary format of the local expression for Slovakia.

Category: Numeric
Alignment: left

Syntax

NLMNLSKKw.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

Examples
In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)',nlmnlskk32.2);
y=input('($12,345.67)\$dollar32.2');

Statements Results

| put x=; | -12345.67 |
| put y=; | -12345.67 |

See Also

Informats:
“NLMNISKK\$d Informat” on page 368

**NLMNLTHB\$d Informat**

Reads the monetary format of the local expression for Thailand.

**Category:** Numeric

**Alignment:** left

**Syntax**

**NLMNLTHB\$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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
x=input('($12,345.67)','nlmnlthb32.2');
y=input('($12,345.67)\text{dollar32.2}');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:

“NLMNITHB\text{w.d} Informat” on page 369

NLMNTRY\text{w.d} Informat

Reads the monetary format of the local expression for Turkey.

Category: Numeric
Alignment: left

Syntax

NLMNTRY\text{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

Examples

In the following example, the \text{LOCALE=} system option is set to English United States.
x=input('($12,345.67)',nlmnltwy32.2);
y=input('($12,345.67)"dollar32.2');

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Informats:
“NLMNTRYw.d Informat” on page 370

**NLMNLWTWDw.d Informat**

Reads the monetary format of the local expression for Taiwan.

Category: Numeric

Alignment: left

**Syntax**

NLMNLWTWD\(w.d\)

**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

**Examples**

In the following example, the LOCALE= system option is set to English_UnitedStates.
NLMNLUSD\textit{w}.\textit{d} Informat

Reads the monetary format of the local expression for Puerto Rico, and the United States.

Category: Numeric
Alignment: left

Syntax

\texttt{NLMNLUSD\textit{w}.\textit{d}}

Syntax Description

\textit{w} specifies the width of the output field.

\begin{itemize}
  \item Default: 9
  \item Range: 1–32
\end{itemize}

\textit{d} optionally specifies to divide the number by $10^d$. If the data contains decimal points, the \textit{d} value is ignored.

\begin{itemize}
  \item Default: 0
  \item Range: 0–31
\end{itemize}

Examples

In the following example, the \texttt{LOCALE=system} option is set to \texttt{English} _\texttt{UnitedStates}.

\begin{verbatim}
x=input'($12,345.67)','nlmnltd32.2);
y=input'($12,345.67)′dollar32.2);
\end{verbatim}

\begin{tabular}{l|c}
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put \texttt{x=};</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put \texttt{y=};</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>
\end{tabular}

See Also

Formats:

“NLMNLTD\textit{w}.\textit{d} Format” on page 187
Informats for NLS

NLMNLZAR\(w.d\) Informat

Reads the monetary format of the local expression for South Africa.

Category: Numeric
Alignment: left

Syntax

NLMNLZAR\(w.d\)

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

Examples

In the following example, the LOCALE= system option is set to English_UnitedStates.
\[ x = \text{input}('($12,345.67)', \text{nlmnlzar32.2}); \]
\[ y = \text{input}('($12,345.67)\text{dollar32.2}'); \]

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{put} x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>\text{put} y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLMNLZAR\text{w.d} Format” on page 189

**NLMNY\text{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\text{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\text{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 close parenthesis from the input data.
Comparisons

The NLMNYw.d informat performs processing that is the opposite of the NLMNYIw.d informat. The NLMNYw.d informat is similar to the DOLLARw.d informat except that the NLMNYw.d informat is locale-specific.

Examples

The following examples use the input value of $12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates; x=input('($12,345.67)',nlmny32.2);</td>
<td></td>
</tr>
<tr>
<td>y=input('($12,345.67)',dollar32.2); put x=;</td>
<td>-12345.67</td>
</tr>
<tr>
<td>put y=;</td>
<td>-12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:
“NLMNYw.d Format” on page 190
“NLMNYIw.d Format” on page 191

Informats:
“NLMNYIw.d Informat” on page 415

NLMNYIw.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

NLMNYIw.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 NLMNYIw.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 close parenthesis from the input data.

Comparisons

The NLMNYIw.d informat performs processing that is the opposite of the NLMNYw.d informat.

Examples

The following examples use the input value of 12,345.67.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates; x=input('(USD12,345.67)',nlmnyi32.2); y=input('$-12,345.67)',dollar32.2); put x=; put y=;</td>
<td>-12345.67 -12345.67</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLNSYw.d Format” on page 190
“NLMNYw.d Format” on page 191

Informats:

“NLMNYw.d Informat” on page 414
**NLNUMw.d Informat**

Reads numeric data in the specified locale for local expressions, and then converts the data to a numeric value.

**Category:** Numeric

**Syntax**

NLNUMw.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 NLNUMw.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 close parenthesis from the input data.

**Comparisons**

The NLNUMw.d informat performs processing that is opposite to the NLNUMIw.d informat.

**Examples**

The following example uses −1234356.78 as the input value.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>x=input(‘−1,234,356.78’,nlnum32.2);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>−1234356.78</td>
</tr>
</tbody>
</table>
See Also

Formats:
“NLNUMIw.d Format” on page 193
“NLMNYYIw.d Format” on page 191
Informats:
“NLNUMIw.d Informat” on page 418

NLNUMIw.d Informat

Reads numeric data in the specified locale for international expressions, and then converts the data to a numeric value.

Category: Numeric

Syntax

NLNUMIw.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 NLNUMIw.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 close parenthesis from the input data.

Comparisons

The NLNUMIw.d informat performs processing that is opposite to the NLNUMw.d informat.

Examples

The following example uses \(-1,234,356.78\) as the input value.
Informats for NLS

Statements | Results
---|---
```
options locale=English_UnitedStates;
```
```
x=input('-1,234,356.78', nlnumi32.2);
```
```
put x=; -1234356.78
```

See Also

Formats:

“NLNUMw.d Format” on page 193
“NLNMIw.d Format” on page 194

Informats:

“NLNUMw.d Informat” on page 417

NLPCTw.d Informat

Reads percentage data in the specified locale for local expressions, and then converts the data to a numeric value.

Category: Numeric

Syntax

NLPCTw.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 NLPCTw.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 close 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>x=input(‘(-12,345.67%)’,nlpct32.2);</td>
<td></td>
</tr>
<tr>
<td>y=input(‘((12,345.67%))’,percent32.2);</td>
<td></td>
</tr>
<tr>
<td>put x=;</td>
<td>(-123.4567)</td>
</tr>
<tr>
<td>put y=;</td>
<td>(-123.4567)</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“NLPCT\(_w.d\) Format” on page 195
“NLPCTI\(_w.d\) Format” on page 197

Informats:

“NLPCTI\(_w.d\) Informat” on page 420

**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 close 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options LOCALE=English_UnitedStates;</td>
<td>-123.4567</td>
</tr>
<tr>
<td>x=input(''-12,345.67%',nlpct32.2);</td>
<td>-123.4567</td>
</tr>
<tr>
<td>y=input(''(12,345.67%)',percent32.2);</td>
<td>-123.4567</td>
</tr>
<tr>
<td>put x=;</td>
<td>-123.4567</td>
</tr>
<tr>
<td>put y=;</td>
<td>-123.4567</td>
</tr>
</tbody>
</table>

**See Also**

**Formats:**

“NLPCT\( w.d \) Format” on page 195

“NLPCTI\( w.d \) Format” on page 197

**Informats:**

“NLPCT\( w.d \) Informat” on page 419
NLTIMAPw. 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

NLTIMAPw.

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>y=input(’04:24:43 PM’,nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td>16:24:43</td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>y=input(’16:24 Uhr’,nltimap11.);</td>
<td></td>
</tr>
<tr>
<td>put y time.;</td>
<td>16:24:43</td>
</tr>
</tbody>
</table>

See Also

Formats:

“NLTIMAPw. Format” on page 205
**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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>options locale=English_UnitedStates;</td>
<td></td>
</tr>
<tr>
<td>y=input('16:24:43',nltime.);</td>
<td>16:24:43</td>
</tr>
<tr>
<td>put y time.;</td>
<td></td>
</tr>
<tr>
<td>options locale=German_Germany;</td>
<td></td>
</tr>
<tr>
<td>y=input('16.24 Uhr',nltime.);</td>
<td>16:24:00</td>
</tr>
<tr>
<td>put y time;</td>
<td></td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“`NLTIME\(w\). Format` on page 204
$\text{REVERJw. Informat}$

Reads character data from right to left and preserves blanks.

Category: Character

Syntax

$\text{REVERJw.}$

Syntax Description

$w$

specifies the width of the input field.

Default: 1 if $w$ is not specified

Range: 1–32767

Comparisons

The $\text{REVERJw.}$ informat is similar to the $\text{REVERS}w$. informat except that $\text{REVERS}w$. informat left aligns the result by removing all leading blanks.

Examples

The following example uses ABCD as the input value.

```
input @1 name $reverj7.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>####DCBA</td>
</tr>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
</tbody>
</table>

* The character # represents a blank space.

See Also

Informats:

"$\text{REVERS}w$. Informat" on page 425
$REVERS w. Informat

Reads character data from right to left, and then left aligns the text.

Category: Character

Syntax

$REVERS w.

Syntax Description

$REVERS w

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.;

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
<tr>
<td>ABCD</td>
<td>DCBA###</td>
</tr>
</tbody>
</table>

* The # character represents a blank space.

See Also

Informats:

“$REVERJ w. Informat” on page 424
$UCS2Bw. 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

$UCS2Bw.

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 $UCS2Bw. informat performs processing that is opposite of the $UCS2BEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>----+----1----+</td>
<td>----+----1----+</td>
</tr>
<tr>
<td>x=input(‘5927’x,$ucs2b.);</td>
<td>x=91e5</td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UCS2Bw. Format” on page 206
“$UCS2Lw. Format” on page 209
“$UCS2Xw. Format” on page 211
“$UTF8Xw. Format” on page 228
Informats:

“$UCS2Lw. Informat” on page 428
“$UCS2Xw. Informat” on page 430
“$UTF8Xw. Informat” on page 444

$UCS2BEw. 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

$UCS2BEw.

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 $UCS2BEw. informat performs processing that is opposite of the $UCS2Bw. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input ('Ａ', $ucs2be2.);</td>
<td></td>
</tr>
<tr>
<td>put ucs2str=$hex4;</td>
<td></td>
</tr>
<tr>
<td>ucs2str=5927</td>
<td></td>
</tr>
</tbody>
</table>
## $UCS2Lw. 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

$UCS2Lw.$

### 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 $UCS2Lw.$ informat performs processing that is opposite of the $UCS2LEw.$ informat. If you are processing data within the same operating environment, then use the $UCS2Xw.$ informat. If you are processing data from different operating environments, then use the $UCS2Bw.$ and $UCS2Lw.$ informats.

### Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('2759'x,$ucs2l.);</td>
<td></td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td></td>
</tr>
<tr>
<td>x=9le5</td>
<td></td>
</tr>
</tbody>
</table>
$UCS2LEw. 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

$UCS2LEw.

Syntax Description

$ \text{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 $UCS2LEw. informat performs processing that is opposite of the $UCS2Lw. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input ('å', $ ucs2le2.);</td>
<td></td>
</tr>
<tr>
<td>put ucs2str=$hex4;</td>
<td></td>
</tr>
<tr>
<td>ucs2str=2759</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:
“$UCS2Lw. Format” on page 209
“$UCS2LEw. Format” on page 210
Informats:
“$UCS2Lw. Informat” on page 428

$UCS2Xw. 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

$UCS2Xw.

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 $UCS2Xw. informat performs processing that is the opposite of the $UCS2XEw. informat. If you are processing data within the same operating environment, then use the $UCS2Xw. informat. If you are processing data from different operating environments, then use the $UCS2Bw. and $UCS2Lw. informats.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment. This example uses little-endian formatting.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('5927'x,$ucs2x.);</td>
<td></td>
</tr>
<tr>
<td>put x=$hex4.;</td>
<td>x=91e5</td>
</tr>
<tr>
<td></td>
<td>------</td>
</tr>
</tbody>
</table>


$UCS2XEw. 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

$UCS2XEw.

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 $UCS2XEw. informat performs processing that is opposite of the $UCS2Xw. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs2str=input (’’, $ ucs2xe2.);</td>
<td></td>
</tr>
<tr>
<td>put ucs2str=$hex6;</td>
<td>ucs2str=5927</td>
</tr>
</tbody>
</table>
$UCS4Bw. 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

$UCS4Bw.

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 $UCS4Xw. informat. If you are processing data from different operating environments, then use the $UCS4Bw. and $UCS4Lw. informats.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put('Zero1', $UCS4B20.);</td>
<td></td>
</tr>
<tr>
<td>x=input(z, $UCS4B20.);</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td>Zero1</td>
</tr>
</tbody>
</table>
See Also

Formats:
“$UCS4Bw. Format” on page 213
Informats:
“$UCS4Lw. Informat” on page 433
“$UCS4Xw. Informat” on page 434

$UCS4Lw. 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
$UCS4Lw.

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 $UCS4Xw. informat. If you are processing data from different operating environments, then use the $UCS4Bw. and $UCS4Lw. informats.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>z=put(’.com’, $UCS4L16.);</td>
<td>2E0000000630000006F0000006D000000</td>
</tr>
<tr>
<td>put z $hex32.;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:

"$UCS4Lw. Format" on page 216

Informats:

"$UCS4Bw. Informat" on page 432
"$UCS4Xw. Informat" on page 434

$UCS4Xw. 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

$UCS4Xw.

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 $UCS4Xw. informat performs processing that is the opposite of the $UCS4XEw. informat. Use the $UCS4Xw. informat when you are processing data within the same operating environment. Use the $UCS4Bw. and $UCS4Lw. 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

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucs4=put(‘91e5’x,$ucs4x.);</td>
<td>ucs4=27590000</td>
</tr>
<tr>
<td>sjis=input(ucs4,$ucs4x.);</td>
<td>sjis=91E52020</td>
</tr>
<tr>
<td>put ucs4=$hex8. sjis=$hex8.;</td>
<td></td>
</tr>
<tr>
<td>run;</td>
<td></td>
</tr>
</tbody>
</table>
See Also

Formats:

“$UCS2Xw. Format” on page 211
“$UCS2Bw. Format” on page 206
“$UCS2Lw. Format” on page 209
“$UCS4Xw. Format” on page 218
“$UTF8Xw. Format” on page 228

Informats:

“$UCS2Bw. Informat” on page 426
“$UCS2Lw. Informat” on page 428
“$UTF8Xw. Informat” on page 444

$UCS4XEw. 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

$UCS4XEw.

Syntax Description

\( w \)

\( w \) 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 $UCS4XEw. informat performs processing that is the opposite of the $UCS4Xw. informat.

Examples

This example uses the Japanese Shift_JIS encoding, which is supported under the UNIX operating environment.
$UESCw. 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

$UESCw.

Syntax Description

$UESCw.

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 $UESCw. informat can be nested.

Comparisons

The $UESCw. informat performs processing that is the opposite of the $UESCEw. informat.
Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input('¥u5927', $uesc10.);</td>
<td>¥u5927</td>
</tr>
<tr>
<td>y=input('¥uu5927', $uesc10.);</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>z=input('¥uuu5927', $uesc10.);</td>
<td>¥uuu5927</td>
</tr>
</tbody>
</table>

See Also

Formats:
“$UESCw. Format” on page 221
“$UESCEw. Format” on page 222

Informats:
“$UESCEw. Informat” on page 437

$UESCEw. 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

$UESCEw.

Syntax Description

\( w \)

specifies the width of the input field.

Default: 8

Range: 1–32000

Details

The $UESCEw. informat can be nested.
Comparisons

The \$UESCEw. informat performs processing that is opposite of the \$UESCw. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{x=input('¥', $uesc10.);}</td>
<td>¥</td>
</tr>
<tr>
<td>\texttt{y=input('¥u5927',$uesc10.);}</td>
<td>¥u5927</td>
</tr>
<tr>
<td>\texttt{z=input('¥uu5927',$uesc10.);}</td>
<td>¥uu5927</td>
</tr>
<tr>
<td>\texttt{put x y z;}</td>
<td>¥uuu5927</td>
</tr>
</tbody>
</table>

See Also

Formats:

“\$UESCw. Format” on page 221
“\$UESCEw. Format” on page 222

Informat:

“\$UESCw. Informat” on page 436

\textbf{\$UNC Rw. Informat}

Reads an NCR character string, and then converts the character string to the encoding of the current SAS session.

\textbf{Category:} Character

\textbf{Syntax}

\texttt{\$UNC Rw.}

\textbf{Syntax Description}

\textit{w}

specifies the width of the input field.

\textbf{Default:} 8

\textbf{Range:} 1–32000
Details
The input string must contain only characters and NCR. Any national characters must be represented in NCR.

Comparison
The $UNCREw. informat performs processing that is opposite of the $UNCREw. informat.

Examples
These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('大', $uncr10.);</td>
<td>abc</td>
</tr>
<tr>
<td>y=input('abc', $uncr10);</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
</tbody>
</table>

See Also
Formats:
“$UNCRw. Format” on page 223
“$UNCREw. Format” on page 224
Informats:
“$UNCREw. Informat” on page 439

$UNCREw. Informat

Reads a character string in the encoding of the current SAS session, and then converts the character string to NCR.

Category: Character

Syntax
$UNCREw.
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 $UNCREw. informat performs processing that is the opposite of the $UNCRw. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('abc', $uncre12.);</td>
<td>&amp;#22823;abc</td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UNCRw. Format” on page 223

“$UNCREw. Format” on page 224

Informats:

“$UNCRw. Informat” on page 438

$UPARENw. 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

$UPARENw.

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 $UPARENw. informat performs processing that is opposite of the $UPARENEw. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('&lt;u0061&gt;',$uparen10.);</td>
<td>a</td>
</tr>
<tr>
<td>w=input('&lt;u0062&gt;',$uparen10.);</td>
<td>b</td>
</tr>
<tr>
<td>x=input('&lt;u0063&gt;',$uparen10.);</td>
<td>c</td>
</tr>
<tr>
<td>y=input('&lt;u0033&gt;',$uparen10.);</td>
<td>3</td>
</tr>
<tr>
<td>z=input('&lt;u5927&gt;',$uparen10.);</td>
<td>×</td>
</tr>
<tr>
<td>put v;</td>
<td></td>
</tr>
<tr>
<td>put w;</td>
<td></td>
</tr>
<tr>
<td>put x;</td>
<td></td>
</tr>
<tr>
<td>put y;</td>
<td></td>
</tr>
<tr>
<td>put z;</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:

“$UPARENw. Format” on page 225

“$UPARENEw. Format” on page 226

Informats:

“$UPARENEw. Informat” on page 442

“$UPARENpw. Informat” on page 443
$UPARENEw. 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

$UPARENEw.

Syntax Description

\[ w \]

specifies the width of the input field.

Default: 8

Range: 1–32000

Comparisons

The $UPARENEw. informat performs processing that is opposite of the $UPARENw. informat.

Examples

These examples use the Japanese Shift_JIS encoding, which is supported under the UNIX operating system.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>v=input('a',$uparen10.);</td>
<td>&lt;u0061&gt;</td>
</tr>
<tr>
<td>w=input('b',$uparen10.);</td>
<td>&lt;u0062&gt;</td>
</tr>
<tr>
<td>x=input('c',$uparen10.);</td>
<td>&lt;u0063&gt;</td>
</tr>
<tr>
<td>y=input('3',$uparen10.);</td>
<td>&lt;u0033&gt;</td>
</tr>
<tr>
<td>z=input('大',$uparen10.);</td>
<td>&lt;u5927&gt;</td>
</tr>
</tbody>
</table>
$UPARENPw. 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

$UPARENPw.

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----+

```plaintext
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:

“$UPARENw. Format” on page 225
“$UPARENw. Format” on page 226

Informats:

“$UPARENw. Informat” on page 440
“$UPARENw. Informat” on page 442

$UTF8Xw. 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**

$UTF8Xw.

**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.
See Also

Formats:
- "$UCS2Bw. Format" on page 206
- "$UCS2Lw. Format" on page 209
- "$UCS2Xw. Format" on page 211
- "$UTF8Xw. Format" on page 228

Informats:
- "$UCS2Bw. Informat" on page 426
- "$UCS2Lw. Informat" on page 428
- "$UCS2Xw. Informat" on page 430

$VSLOGw. 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

$VSLOGw.

Syntax Description

w
- specifies the width of the input field.
  - Default: 200
  - Range: 1–32000

Comparisons

The $VSLOGw. informat performs processing that is opposite of the $VSLOGRw. informat.

Examples

The following example uses the Hebrew input value of "_flight".
Statements | Result |
---|---|
---+----1----+ |
\[
x=\text{input} (\text{"الف행"}, \text{vslog12}.); \\
\text{put} \ x; \\
\]
flight

The following example uses the Arabic input value of "الف행" computer."

Statements | Result |
---|---|
---+----1----+ |
\[
x=\text{input} (\text{"الف행"}, \text{vslog12}.); \\
\text{put} \ x; \\
\]
ذاتcomputer

**See Also**

Formats:

- "$VSLOGRw. Format" on page 230
- "$VSLOGw. Format" on page 229

Informats:

- "$VSLOGRw. Informat" on page 446

---

**$VSLOGRw. 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**

$VSLOGRw.

**Syntax Description**

\[ w \]

specifies the width of the input field.

**Default:** 200

**Range:** 1–32000
Comparisons

The $VSLOGRw. informat performs processing that is opposite of the $VSLOGw. informat.

Examples

The following example uses the Hebrew input value of “_take the flight.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('\u05d9 \u05d4 \u05d5',$vslogr12.);</td>
<td>flight</td>
</tr>
<tr>
<td>put x;</td>
<td>flight</td>
</tr>
</tbody>
</table>

The following example uses the Arabic input value of “هابلل الكمبيوتر.”

<table>
<thead>
<tr>
<th>Statements</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>x=input ('هابلل الكمبيوتر',$vslogr12.);</td>
<td>computer</td>
</tr>
<tr>
<td>put x;</td>
<td>computer</td>
</tr>
</tbody>
</table>

See Also

Formats:
“$VSLOGw. Format” on page 229
“$VSLOGRw. Format” on page 230

Informats:
“$VSLOGw. Informat” on page 445

YENw.d Informat

Removes embedded yen signs, commas, and decimal points.

Category: Numeric

Syntax

YENw.d
Syntax Description

\( w \)
specifies the width of the input field.

**Default:** 1

**Range:** 1–32

\( d \)
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.

```plaintext
input value yen10.2;
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>¥1254.71</td>
<td>1254.71</td>
</tr>
</tbody>
</table>

See Also

Formats:

“YEN\( w.d \) Format” on page 240
## Autocall Macros for NLS by Category

The following table provides brief descriptions of the SAS NLS autocall macros. For more detailed descriptions, see the NLS entry for each macro.

### Table 10.1 Autocall Macros for NLS by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Autocall Macros for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBCS</td>
<td>“%KCMPRES and %QKCMPRES Autocall Macros” on page 449</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“%KLEFT and %QKLEFT Autocall Macros” on page 450</td>
<td>Left-align an argument by removing leading blanks.</td>
</tr>
<tr>
<td></td>
<td>“%KLOWCASE and %QKLOWCAS Autocall Macros” on page 451</td>
<td>Change uppercase characters to lowercase.</td>
</tr>
<tr>
<td></td>
<td>“%KTRIM and %QKTRIM Autocall Macros” on page 451</td>
<td>Trim trailing blanks.</td>
</tr>
<tr>
<td></td>
<td>“%KVERIFY Autocall Macro” on page 452</td>
<td>Returns the position of the first character unique to an expression.</td>
</tr>
</tbody>
</table>

### %KCMPRES and %QKCMPRES Autocall Macros

**Category:** DBCS  
**Requirement:** MAUTOSOURCE system option  

**Syntax**  
%

%KCMPRES (text | text expression)
Note: Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

Details

The %KCMPRES and %KQCMPRES macros compress multiple blanks and remove leading and trailing blanks. If the argument might contain a special character or mnemonic operator, listed below, use %KQCMPRES.

%KCMPRES returns an unquoted result, even if the argument is quoted. %KQCMPRES produces a result with the following special characters and mnemonic operators masked, so the macro processor interprets them as text instead of as elements of the macro language:

\& % " ( ) + - * / < > = ~ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN

%KLEFT and %QKLEFT Autocall Macros

Left-align an argument by removing leading blanks.

Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KLEFT (text | text expression)
%QKLEFT (text | text expression)

Note: Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

Details

The &KLEFT macro and the &KQLEFT macro both left-align arguments by removing leading blanks. If the argument might contain a special character or mnemonic operator, listed below, use &KQLEFT.

%KLEFT returns an unquoted result, even if the argument is quoted. %KQLEFT produces a result with the following special characters and mnemonic operators masked so the macro processor interprets them as text instead of as elements of the macro language:

\& % " ( ) + - * / < > = ~ ^ ~ ; , # blank
AND OR NOT EQ NE LE LT GE GT IN
%KLOWCASE and %QKLOWCAS Autocall Macros

Change uppercase characters to lowercase.

Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KLOWCASE (text | text expression)
%QKLOWCAS (text | text expression)

Note: Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

Details

The %KLOWCASE and %QKLOWCAS macros change uppercase alphabetic characters to their lowercase equivalents. If the argument might contain a special character or mnemonic operator, listed below, use %QKLOWCAS.

%KLOWCASE returns a result without quotation marks, even if the argument has quotation marks. %QKLOWCAS produces a result with the following special characters and mnemonic operators masked so the macro processor interprets them as text instead of as elements of the macro language:

& % ^ ( ) _ + - * / < > = > < 
, blank AND OR NOT EQ NE LE LT GE GT IN

%KTRIM and %QKTRIM Autocall Macros

Trim trailing blanks.

Category: DBCS
Requirement: MAUTOSOURCE system option

Syntax

%KTRIM (text | text expression)
%QKTRIM (text | text expression)

Note: Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel.

Details

The KTRIM macro and the QKTRIM macro trim trailing blanks. If the argument contains a special character or mnemonic operator, listed below, use %QKTRIM.
QKTRIM produces a result with the following special characters and mnemonic operators masked so the macro processor interprets them as text instead of as elements of the macro language:
& % *** ()+ - */ < > = ? ~ ; # blank AND OR NOT EQ NE LE LT GE GT IN

%KVERIFY Autocall Macro

Returns the position of the first character unique to an expression.

Category: DBCS

Requirement: MAUTOSOURCE system option

Syntax

%KVERIFY (source, excerpt)

Syntax

source

is text or a text expression that you want to examine for characters that do not exist in excerpt.

excerpt

is text or a text expression that defines the set of characters that %KVERIFY uses to examine source.

Note: Autocall macros are included in a SAS library. This library might not be installed at your site or might be a site-specific version. If you cannot access this macro or if you want to find out if the library is a site-specific version, see your on-site SAS support personnel. △

Details

%KVERIFY returns the position of the first character in source that is not also present in excerpt. If all characters in source are present in excerpt, %KVERIFY returns 0.
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 11.1  Summary of NLS System Options Category

<table>
<thead>
<tr>
<th>Category</th>
<th>System Options for NLS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environment control:</td>
<td>“DATESTYLE= System Option” on page 456</td>
<td>Identifies the sequence of month, day, and year when the ANYDTDM, ANYDTDTE, or ANYDTTME informats encounter input where the year, month, and day determination is ambiguous.</td>
</tr>
<tr>
<td>Language control</td>
<td>“DBCS System Option: UNIX, Windows, and z/OS” on page 456</td>
<td>Recognizes double-byte character sets (DBCS).</td>
</tr>
<tr>
<td>Category</td>
<td>System Options for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>“DBCSLANG System</td>
<td>Specifies a double-byte character set (DBCS) language.</td>
</tr>
<tr>
<td></td>
<td>Option: UNIX, Windows, and z/OS” on page 457</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“DBCSYSTEM Type</td>
<td>Specifies the encoding method to use for a double-byte</td>
</tr>
<tr>
<td></td>
<td>Option: UNIX, Windows, and z/OS” on page 458</td>
<td>character set (DBCS).</td>
</tr>
<tr>
<td></td>
<td>“DFLANG= System</td>
<td>Specifies the language for international date informats</td>
</tr>
<tr>
<td></td>
<td>Option: OpenVMS, UNIX, Windows, and z/OS” on page 460</td>
<td>and formats.</td>
</tr>
<tr>
<td></td>
<td>“ENCODING System</td>
<td>Specifies the default character-set encoding for the SAS</td>
</tr>
<tr>
<td></td>
<td>Option: OpenVMS, UNIX, Windows, and z/OS” on page 461</td>
<td>session.</td>
</tr>
<tr>
<td></td>
<td>“FSDBTYPE System</td>
<td>Specifies a full-screen double-byte character set (DBCS)</td>
</tr>
<tr>
<td></td>
<td>Option: UNIX” on page 462</td>
<td>encoding method.</td>
</tr>
<tr>
<td></td>
<td>“FSIMM System Option: UNIX” on page 463</td>
<td>Specifies input method modules (IMMs) for full-screen</td>
</tr>
<tr>
<td></td>
<td>“FSIMMOPT System</td>
<td>Specifies options for input method modules (IMMs) that</td>
</tr>
<tr>
<td></td>
<td>Option: UNIX” on page 464</td>
<td>are used with a full-screen double-byte character set</td>
</tr>
<tr>
<td></td>
<td>“LOCALE System Option”</td>
<td>Specifies a set of attributes in a SAS session that reflect</td>
</tr>
<tr>
<td></td>
<td>on page 465</td>
<td></td>
</tr>
<tr>
<td></td>
<td>“LOCALELANGCHG System Option” on page 466</td>
<td>Determines whether the language of the text of the ODS</td>
</tr>
<tr>
<td></td>
<td>“NLSCOMPATMODE System Option: z/OS” on page 468</td>
<td>output can be changed</td>
</tr>
<tr>
<td></td>
<td>“PAPERSIZE= System Option” on page 469</td>
<td>Specifies the paper size for the printer to use.</td>
</tr>
<tr>
<td></td>
<td>“TRANTAB= System Option” on page 471</td>
<td>Specifies the translation tables that are used by various</td>
</tr>
<tr>
<td></td>
<td></td>
<td>parts of SAS.</td>
</tr>
<tr>
<td>Files: External files</td>
<td>“BOMFILE System Option” on page 455</td>
<td>Specifies whether to write the byte order mark (BOM)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>prefix on Unicode-encoded external files.</td>
</tr>
<tr>
<td>Category</td>
<td>System Options for NLS</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Files: SAS files</td>
<td>“RSASIOTRANSERROR</td>
<td>Displays a transcoding error when illegal data is read from a remote application.</td>
</tr>
<tr>
<td></td>
<td>System Option” on page</td>
<td>469</td>
</tr>
<tr>
<td>Sort: Procedure options</td>
<td>Specifies a</td>
<td>language-specific collating sequence for the SORT and SQL procedures to use in the current SAS session.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**BOMFILE System Option**

Specifies whether to write the byte order mark (BOM) prefix on Unicode-encoded external files.

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: External files

PROC OPTIONS GROUP: EXTFILES

**Syntax**

**BOMFILE | NOBOMFILE**

**Syntax Description**

**BOMFILE**

Specifies to write a byte order mark (BOM) prefix when a Unicode-encoded file is written to an external file.

**NOBOMFILE**

Specify the BOM file when a Unicode-encoded file is written to an external file.

**Details**

The BOMFILE system option does not apply when a Unicode-encoded external file is read.

A BOM is a signature at the beginning of a Unicode data stream. The size of the BOM varies depending on the encoding.
**DATESTYLE= System Option**

Identifies the sequence of month, day, and year when the ANYDTDM, 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**

do 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 35
“DBCS Values for a SAS Session” on page 553
Chapter 17, “Encoding Values in SAS Language Elements,” on page 555

System Options:
“DBCSLANG System Option: UNIX, Windows, and z/OS” on page 457
“DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458

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)

language depends on the operating environment. The following table contains valid language values:

<table>
<thead>
<tr>
<th>Language</th>
<th>z/OS</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHINESE (simplified)</td>
<td>yes*</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>JAPANESE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>KOREAN</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>TAIWANESE (traditional)</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>NONE</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>UNKNOWN</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

* 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
   Chapter 5, “Double-Byte Character Sets (DBCS),” on page 35
   “DBCS Values for a SAS Session” on page 553
   Chapter 17, “Encoding Values in SAS Language Elements,” on page 555
System Options:
   “DBCS System Option: UNIX, Windows, and z/OS” on page 456
   “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458

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 specifies: 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 11.3  DBCS Encoding Methods for z/OS

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM</td>
<td>IBM PC encoding method</td>
</tr>
</tbody>
</table>

Table 11.4  DBCS Encoding Methods for UNIX

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>DEC encoding method</td>
</tr>
<tr>
<td>EUC</td>
<td>Extended UNIX Code encoding method</td>
</tr>
<tr>
<td>HP15</td>
<td>Hewlett Packard encoding method</td>
</tr>
<tr>
<td>PCIBM</td>
<td>IBM PC encoding method</td>
</tr>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese</td>
</tr>
<tr>
<td></td>
<td>language only</td>
</tr>
<tr>
<td>NONE</td>
<td>Disables DBCS processing</td>
</tr>
</tbody>
</table>

Table 11.5  DBCS Encoding Methods for Windows

<table>
<thead>
<tr>
<th>DBCSTYPE= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCMS</td>
<td>Microsoft PC encoding method</td>
</tr>
<tr>
<td>WINDOWS</td>
<td>Alias for PCMS</td>
</tr>
<tr>
<td>SJIS</td>
<td>Shift-JIS encoding method for the Japanese</td>
</tr>
<tr>
<td></td>
<td>language only</td>
</tr>
</tbody>
</table>

See Also

Conceptual Information:
- Chapter 5, “Double-Byte Character Sets (DBCS),” on page 35
- “DBCS Values for a SAS Session” on page 553
- Chapter 17, “Encoding Values in SAS Language Elements,” on page 555

System Options:
- “DBCS System Option: UNIX, Windows, and z/OS” on page 456
- “DBCSSLANG System Option: UNIX, Windows, and z/OS” on page 457
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 languages 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.
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

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-ENCODING= ASCIIANY | EBCDICANY | encoding-value (UNIX and Windows)

ENCODING= encoding-value (OpenVMS, UNIX, Windows, and z/OS)

ASCIIANY

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 18, “Encoding Values for a SAS Session,” on page 561.

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: Encoding for NLS” on page 9
Conceptual discussion about “Overview to Transcoding” on page 27
Chapter 15, “Values for the LOCALE= System Option,” on page 545
Chapter 16, “SAS System Options for Processing DBCS Data,” on page 553
Chapter 17, “Encoding Values in SAS Language Elements,” on page 555

### 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 specifies:** 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 11.6 Full-Screen DBCS Encoding Methods**

<table>
<thead>
<tr>
<th>FSDBTYPE= Encoding Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dec</td>
<td>Digital Equipment Corporation encoding method</td>
</tr>
<tr>
<td>euc</td>
<td>Extended UNIX encoding method</td>
</tr>
<tr>
<td>hp15</td>
<td>HP-UX encoding method</td>
</tr>
<tr>
<td>jis7</td>
<td>7-bit Shift-JIS encoding method used in an X windows environment for the Japanese language only</td>
</tr>
</tbody>
</table>
### 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 `saswijms`, which uses the vendor-supplied MOTIF toolkit.

For example, to use the PIPE input method module for X11 drivers, you would specify:

```
-FSIMM X11=PIPE
```

**Note:** The server is specified by using the FSIMMOPT option.
See Also

Conceptual Information:
Chapter 5, “Double-Byte Character Sets (DBCS),” on page 35

System Option:
“FSIMMOPT System Option: UNIX” on page 464

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 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 35

System Option:
“FSIMM System Option: UNIX” on page 463
LOCALE System Option

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

UNIX specifics: Also valid in SASV9_OPTIONS environment variable

PROC OPTIONS GROUP: LANGUAGECONTROL

Syntax

-LOCALE locale-name (UNIX and Windows)

LOCALE=locale-name (UNIX, Windows, and z/OS)

locale-name

For a complete list of locale values (SAS names and POSIX names), see Chapter 15, “Values for the LOCALE= System Option,” on page 545.

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 15, “Values for the LOCALE= System Option,” on page 545

System Options:

- “ENCODING System Option: OpenVMS, UNIX, Windows, and z/OS” on page 461
- DATESTYLE in *SAS Language Reference: Dictionary*
- “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460
- PAPERSIZE in *SAS Language Reference: Dictionary*
- “TRANTAB= System Option” on page 471

---

**LOCALELANGCHG System Option**

Determines whether the language of the text of the ODS output can be changed

**Default:** LOCALELANGCHG is set to off in all servers except for the UNICODE server

**Valid in:** configuration file, SAS invocation

**Category:** Environment control: Language control

**PROC OPTIONS GROUP=** LANGUAGECONTROL

**Tip:** The Language Switching feature, which uses the LOCALELANGCHG option, is supported in a Unicode server (a SAS server with a session encoding of UTF-8, ENCODING=utf8).

**Syntax**

LOCALELANGCHG | NOLOCALELANGCHG
Syntax Description

**LOCALELANGCHG**
Specifies that the language of the SAS message text in ODS output can change when the LOCALE option is set after startup.

**NOLOCALELANGCHG**
Specifies that the language of the SAS message text in ODS output cannot change when the LOCALE option is set after startup.

Details

The Language Switching feature allows you to change the language of SAS messages after startup. You must enable LOCALELANGCHG to use this feature.

During startup, the configuration file and LOCALE option determine the language for SAS messages. After startup, if the LOCALE option and LOCALELANGCHG option are set, then the language for messages and ODS templates can change to reflect the LOCALE setting when the localizations are available.

You can enable LOCALELANGCHG but not translate into the language of the locale. For example, if you enable LOCALELANGCHG, then start a SAS session in French and set the locale to Greek, NLDATE displays in Greek. The output displays in French. The output displays in French because SAS does not translate into Greek.

Comparisons

If LOCALELANGCHG is enabled at startup and LOCALE is changed during the session, the ODS PATH is updated to include the translated template item store if it exists for the language of the new locale. Messages that do not appear in the SAS log appear in the language of the new locale. Also log messages appear in the original language of the session locale.

If LOCALELANGCHG is not enabled at startup and LOCALE is changed during the session, ODS output appears in the language that was set at startup.

Examples

Example 1 is a French server with LOCALELANGCHG not enabled (NOLOCALELANGCHG).

If a French-client application connects to the server, the output appears in French, and dates, formatted by using the NLDATE format, appear in French. If a German-client application connects to the French server, and the locale is set to German on the executive session, then output messages appear in French, and dates formatted with NLDATE appear in German.

Example 2 is a French server with LOCALELANGCHG enabled (LOCALELANGCHG).

If a French-client application connects to the server, the output appears in French, and dates, formatted by using the NLDATE format, appear in French. If a German-client application connects to the French server, and the locale is set to German on the executive session, then output messages appear in German, and dates formatted with NLDATE appear in German.
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

Syntax Description

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. When NONLSCOMPATMODE is in effect, the encoding that SAS uses to process character data is the encoding that is set by the ENCODING or LOCATE options. Compiler and Session encoding characters remain separate.
PAPERSIZE= System Option

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

RSASIOTRANSERROR System Option

Displays a transcoding error when illegal data is read from a remote application.

Default: RSASIOTRANSERROR

Valid in: configuration file, SAS invocation, OPTIONS statement, SAS System Options window

Category: Files: SAS files

PROC OPTIONS GROUP: SASFILES

Syntax

RSASIOTRANSERROR | NOSASIOTRANSERROR

Syntax Description

RSASIOTRANSERROR
specifies to display a transcoding error when illegal values are read from a remote application.

NOSASIOTRANSERROR
specifies not to display a transcoding error when illegal values are read from a remote application.

Details

The RSASIOTRANSERROR system option enables remote users of SASIO, for example SAS Enterprise Guide and SAS Enterprise Miner, to ignore illegal data values. An illegal data value typically will cause a transcoding error when the data is read by a remote application.
SORTSEQ= System Option: UNIX, Windows, and z/OS

Specifies a language-specific collating sequence for the SORT and SQL procedures 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=collating-sequence

Syntax Description

collating-sequence

specifies the collating 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 collating 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 collating 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. Collating 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:

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

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

**Example**

This example demonstrates the functionality of SORTSEQ with PROC SORT and PROC SQL:

```sas
options sortseq=reverse;
proc sort data=sashelp.class out=foo1;
  by name;
run;

proc sql;
  create table foo2 as select * from sashelp.class order by name;
quit;
run;
```

**See Also**

“Collating Sequence” on page 16

System Options:

“TRANTAB= System Option” on page 471

---

**TRANTAB= System Option**

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 9.2 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 list that is enclosed in parentheses and 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:

<table>
<thead>
<tr>
<th>Position</th>
<th>Type of Translation Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>local-to-transport-format</td>
</tr>
<tr>
<td>2nd</td>
<td>transport-to-local-format</td>
</tr>
<tr>
<td>3rd</td>
<td>lowercase-to-uppercase</td>
</tr>
<tr>
<td>4th</td>
<td>uppercase-to-lowercase</td>
</tr>
<tr>
<td>5th</td>
<td>character classification</td>
</tr>
<tr>
<td>6th</td>
<td>scanner translation</td>
</tr>
<tr>
<td>7th</td>
<td>delta characters</td>
</tr>
<tr>
<td>8th</td>
<td>scanner character classification</td>
</tr>
<tr>
<td>9th</td>
<td>not used</td>
</tr>
<tr>
<td>10th</td>
<td>DBCS user table</td>
</tr>
</tbody>
</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:

```sas
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:

```sas
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 14, “The TRANTAB Procedure,” on page 515
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 12.1 Summary of NLS Statements by Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Commands, Statements, and Procedures for NLS by Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Access</td>
<td>“CVPBYTES=, CVPENGINE=, and CVPMULTIPLIER= Options” on page 484</td>
<td>Specifies attributes for character variables that are needed in order to transcode a SAS file.</td>
</tr>
<tr>
<td></td>
<td>“ENCODING= Option” on page 489</td>
<td>Overrides and transcodes the encoding for input or output processing of external files.</td>
</tr>
<tr>
<td></td>
<td>“INENCODING= and OUTENCODING= Options” on page 493</td>
<td>Overrides and changes the encoding when reading or writing SAS data sets in the SAS library.</td>
</tr>
<tr>
<td>Category</td>
<td>Commands, Statements, and Procedures for NLS by Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>“ODSCHARSET= Option” on page 494</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>“ODSTRANTAB= Option” on page 495</td>
<td>Specifies the translation table to use when transcoding an XML document for an output file.</td>
</tr>
<tr>
<td></td>
<td>“RENCODING= Option” on page 497</td>
<td>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 ASCIANY session encoding.</td>
</tr>
<tr>
<td></td>
<td>“XMLENCODING= Option” on page 502</td>
<td>Overrides the encoding of an XML document to import or export an external document.</td>
</tr>
<tr>
<td>Information</td>
<td>“TRANSCODE= Option” on page 499</td>
<td>Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and length with one or more variables) that indicates whether character variables are to be transcoded.</td>
</tr>
<tr>
<td>ODS: Third-Party Formatted</td>
<td>“CHARSET= Option” on page 476</td>
<td>Specifies the character set to be generated in the META declaration for the output.</td>
</tr>
<tr>
<td></td>
<td>“TRANTAB= Option” on page 501</td>
<td>Specifies the translation table to use when you are transcoding character data in a SAS file for the appropriate output file.</td>
</tr>
</tbody>
</table>

**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](http://www.unicode.org/reports/tr22/index.html) and [www.iana.org/assignments/character-sets](http://www.iana.org/assignments/character-sets).
Options for Commands, Statements, and Procedures for NLS  △ Collating Sequence Option  477

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:

Collating Sequence Option

Specifies the collating 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 collating-sequence-option <other option(s)>;

Options

<table>
<thead>
<tr>
<th>Task</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specify the collating sequence</td>
<td></td>
</tr>
<tr>
<td>Specify ASCII</td>
<td>ASCII</td>
</tr>
<tr>
<td>Specify EBCDIC</td>
<td>EBCDIC</td>
</tr>
<tr>
<td>Specify Danish</td>
<td>DANISH</td>
</tr>
<tr>
<td>Specify Finnish</td>
<td>FINNISH</td>
</tr>
<tr>
<td>Specify Norwegian</td>
<td>NORWEGIAN</td>
</tr>
<tr>
<td>Specify Polish</td>
<td>POLISH</td>
</tr>
<tr>
<td>Specify Swedish</td>
<td>SWEDISH</td>
</tr>
</tbody>
</table>
Options can include one `collating-sequence-option` and multiple other `options`. The order of the two types of options does not matter and both types are not necessary in the same PROC SORT step. Only the explanations for the PROC SORT collating-sequence-options follow.

**Operating Environment Information:** For information about behavior specific to your operating environment for the DANISH, FINNISH, NORWEGIAN, or SWEDISH `collating-sequence-option`, see the SAS documentation for your operating environment.

---

**ASCII**

sorts character variables using the ASCII collating sequence. You need this option only when you want to achieve an ASCII ordering on a system where EBCDIC is the native collating sequence.

**DANISH**

sorts characters according to the Danish and Norwegian convention.

The Danish and Norwegian collating sequence is shown in Figure 12.1 on page 479.

**EBCDIC**

sorts character variables using the EBCDIC collating sequence. You need this option only when you want to achieve an EBCDIC ordering on a system where ASCII is the native collating sequence.

**POLISH**

sorts characters according to the Polish convention.

**FINNISH**

**SWEDISH**

sorts characters according to the Finnish and Swedish convention. The Finnish and Swedish collating sequence is shown in Figure 12.1 on page 479.

**NATIONAL**

sorts character variables using an alternate collating 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 whether a customized national sort sequence is available.

**NORWEGIAN**

See DANISH.
SWEDISH  
See FINNISH.

SORTSEQ=collating-sequence  
specifies the collating sequence. The collating-sequence can be a  
collating-sequence-option, a translation table, an encoding, or the keyword  
LINGUISTIC. Only one collating sequence can be specified. For detailed  
information, refer to “Collating Sequence” on page 16. 

Here are descriptions of the collating sequences:

```
collating—sequence—option | translation_table  
specifies either a translation table, which can be one that SAS provides or  
any user-defined translation table, or one of the PROC SORT statement  
Collating-Sequence-Options. For an example of using PROC TRANTAB and  
PROC SORT with SORTSEQ=, see Using Different Translation Tables for  
Sorting.
```

The available translation tables are

- ASCII
- DANISH
- EBCDIC
- FINNISH
- ITALIAN
- NORWEGIAN
- POLISH
- REVERSE
- SPANISH
- SWEDISH

The following figure shows how the alphanumeric characters in each  
language will sort.

![Figure 12.1 National Collating Sequences of Alphanumeric Characters](image)

**Restriction:** You can specify only one collating-sequence-option in a PROC  
SORT step.

**Tip:** The SORTSEQ= collating sequence options are specified without  
parenthesis and have no arguments associated with them. An example of  
how to specify a collating sequence follows:

```
proc sort data=mydata SORTSEQ=ASCII;
```

**encoding-value**  
specifies an encoding value. The result is the same as a binary collation of  
the character data represented in the specified encoding. See the supported  
encoding values in “SBCS, DBCS, and Unicode Encoding Values for  
Transcoding Data” on page 555.
Restriction: PROC SORT is the only procedure or part of the SAS system that recognizes an encoding specified for the SORTSEQ= option.

Tip: When the encoding value contains a character other than an alphanumeric character or underscore, the value needs to be enclosed in quotation marks.

See: The list of the encodings that can be specified in “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 555.

LINGUISTIC<collating—rules > specifies linguistic collation, which sorts characters according to rules of the specified language. The rules and default collating sequence options are based on the language specified in the current locale setting. The implementation is provided by the International Components for Unicode (ICU) library and produces results that are largely compatible with the Unicode Collation Algorithms (UCA).

Alias: UCA

Restriction: The SORTSEQ=LINGUISTIC option is available only on the PROC SORT SORTSEQ= option and is not available for the SAS System SORTSEQ= option.

Restriction Note that linguistic collation is not supported on platforms VMS on Itanium (VMI) or 64-bit Windows on Itanium (W64).

Tip: LINGUISTIC sorting requires more memory with the z/OS mainframe. You might need to set your REGION to 50M or higher. This action must be done in JCL, if you are running in batch mode, or in the VERIFY screen if you are running interactively. This action allows the ICU libraries to load properly and does not affect the memory that is used for sorting.

Tip: The collating-rules must be enclosed in parentheses. More than one collating rule can be specified.

Tip: When BY processing is performed on data sets that are sorted with linguistic collation, the NOBYSORTED system option might need to be specified in order for the data set to be treated properly. BY processing is performed differently than collating sequence processing.

See: The ICU License agreement in the Base SAS Procedures Guide.

See: The “Collating Sequence” on page 16 for detailed information on linguistic collation.

See Also: Refer to http://www.unicode.org Web site for the Unicode Collation Algorithm (UCA) specification.

The following are the collation-rules that can be specified for the LINGUISTIC option. These rules modify the linguistic collating sequence:

**ALTERNATE_HANDLING=SHIFTED** controls the handling of variable characters like spaces, punctuation, and symbols. When this option is not specified (using the default value Non-Ignorable), differences among these variable characters are of the same importance as differences among letters. If the ALTERNATE_HANDLING option is specified, these variable characters are of minor importance.

Default: NON_IGNORABLE

Tip: The SHIFTED value is often used in combination with

STRENGTH= set to Quaternary. In such a case, whitespace, punctuation, and symbols are considered when comparing strings, but only if all other aspects of the strings (base letters, accents, and case) are identical.
CASE_FIRST=
specify order of uppercase and lowercase letters. This argument is valid for only TERTIARY, QUATERNARY, or IDENTICAL levels. The following table provides the values and information for the CASE_FIRST argument:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER</td>
<td>Sorts uppercase letters first, then the lowercase letters.</td>
</tr>
<tr>
<td>LOWER</td>
<td>Sorts lowercase letters first, then the uppercase letters.</td>
</tr>
</tbody>
</table>

COLLATION=
The following table lists the available COLLATION= values: If you do not select a collation value, then the user’s locale-default collation is selected.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIG5HAN</td>
<td>specifies Pinyin ordering for Latin and specifies big5 charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>DIRECT</td>
<td>specifies a Hindi variant.</td>
</tr>
<tr>
<td>GB2312HAN</td>
<td>specifies Pinyin ordering for Latin and specifies gb2312han charset ordering for Chinese, Japanese, and Korean characters.</td>
</tr>
<tr>
<td>PHONEBOOK</td>
<td>specifies a telephone-book style for ordering of characters. Select PHONEBOOK only with the German language.</td>
</tr>
<tr>
<td>PINYIN</td>
<td>specifies an ordering for Chinese, Japanese, and Korean characters based on character-by-character transliteration into Pinyin. This ordering is typically used with simplified Chinese.</td>
</tr>
<tr>
<td>POSIX</td>
<td>is the Portable Operating System Interface. This option specifies a &quot;C&quot; locale ordering of characters.</td>
</tr>
<tr>
<td>STROKE</td>
<td>specifies a nonalphabetic writing style ordering of characters. Select STROKE with Chinese, Japanese, Korean, or Vietnamese languages. This ordering is typically used with Traditional Chinese.</td>
</tr>
<tr>
<td>TRADITIONAL</td>
<td>specifies a traditional style for ordering of characters. For example, select TRADITIONAL with the Spanish language.</td>
</tr>
</tbody>
</table>
LOCATE=locale_name
specifies the locale name in the form of a POSIX name. For example, ja_JP. See the Table 15.1 on page 545 for a list of locale and POSIX values supported by PROC SORT.

Restriction: The following locales are not supported by PROC SORT:
- Afrikaans_SouthAfrica, af_ZA
- Cornish_UnitedKingdom, kw_GB
- ManxGaelic_UnitedKingdom, gv_GB

NUMERIC_COLLATION=
orders integer values within the text by the numeric value instead of characters used to represent the numbers.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>Order numbers by the numeric value. For example, &quot;8 Main St.&quot; would sort before &quot;45 Main St.&quot;.</td>
</tr>
<tr>
<td>OFF</td>
<td>Order numbers by the character value. For example, &quot;45 Main St.&quot; would sort before &quot;8 Main St.&quot;.</td>
</tr>
</tbody>
</table>

Default: OFF

STRENGTH=
The value of strength is related to the collation level. There are five collation-level values. The following table provides information about the five levels. The default value for strength is related to the locale.

<table>
<thead>
<tr>
<th>Value</th>
<th>Type of Collation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIMARY or 1</td>
<td>PRIMARY specifies differences between base characters (for example, &quot;a&quot; &lt; &quot;b&quot;).</td>
<td>It is the strongest difference. For example, dictionaries are divided into different sections by base character.</td>
</tr>
<tr>
<td>SECONDARY or 2</td>
<td>Accents in the characters are considered secondary differences (for example, &quot;as&quot; &lt; &quot;às&quot; &lt; &quot;at&quot;).</td>
<td>A secondary difference is ignored when there is a primary difference anywhere in the strings. Other differences between letters can also be considered secondary differences, depending on the language.</td>
</tr>
<tr>
<td>TERTIARY or 3</td>
<td>Upper and lowercase differences in characters are distinguished at the tertiary level (for example, &quot;ao&quot; &lt; &quot;Aø&quot; &lt; &quot;aò&quot;).</td>
<td>A tertiary difference is ignored when there is a primary or secondary difference anywhere in the strings. Another example is the difference between large and small Kana.</td>
</tr>
</tbody>
</table>
### Value  |  Type of Collation  |  Description
---|---|---
`QUATERNARY` or 4 | When punctuation is ignored at level 1-3, an additional level can be used to distinguish words with and without punctuation (for example, "ab" < "a-b" < "aB"). | The quaternary level should be used if ignoring punctuation is required or when processing Japanese text. This difference is ignored when there is a primary, secondary or tertiary difference.

`IDENTICAL` or 5 | When all other levels are equal, the identical level is used as a tiebreaker. The Unicode code point values of the Normalization Form D (NFD) form of each string are compared at this level, just in case there is no difference at levels 1-4. | This level should be used sparingly, as only code point values differences between two strings is an extremely rare occurrence. For example, only Hebrew cantillation marks are distinguished at this level.

---

**Alias:** LEVEL=

**CAUTION:** If you use a host sort utility to sort your data, then specifying a translation table based collating sequence with the `SORTSEQ=` option might corrupt the character BY variables. For more information, see the PROC SORT documentation for your operating environment.

**See Also**

“Collating Sequence” on page 16

Procedures

- The SORT Procedure in *Base SAS Procedures Guide*.

System Options:

- “`SORTSEQ=` System Option: UNIX, Windows, and z/OS” on page 470
- “`TRANTAB=` System Option” on page 471

---

### 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

\(<\text{CORRECTENCODING}=\text{encoding-value}\)> 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:

\[
\text{modify mydata / correctencoding=latin2;}
\]

For a list of valid encoding values for transcoding, see “SBCS, DBCS, and Unicode Encoding Values for Transcoding Data” on page 555.

Restriction: CORRECTENCODING= can be used only when the SAS file uses the default base engine, which is V9 in SAS 9.

Example

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 before 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 action 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:

\[
\text{Note: CEDA creates a read-only copy. You need to copy the data with PROC COPY or a DATA step to transcode the data permanently.}\n\]

\[
\text{proc datasets library=myfiles;}
\]
\[
\text{modify olddata / correctencoding=ebcdic1142;}
\]
\[
\text{quit;}
\]

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

\[
\text{LIBNAME libref CVPBYTES=bytes} \ \text{CVPENGINE=engine} \ \text{CVPMULTIPLIER=multiplier} \ 'SAS data-library';
\]
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 486
See also: “Avoiding Character Data Truncation by Using the CVP Engine” on page 38

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.

```sas
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 38

### Example

**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.
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 12.1  PROC CONTENTS Output for MYFILES.WLATIN2

The following code is executed with the session encoding WLatin2.

```sas
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.
Chapter 12

Output 12.2  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 data set MYFILES.UTF8.
NOTE: The data step has been abnormally terminated.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: There were 0 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 two 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 12.3  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 12.4 PROC CONTENTS Output for MYFILES.UTF8

---

ENCODING= Option

Overides 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 555.

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  
        attach=('C:\My Files\Test.out'  
          content_type='text/plain'  
          encoding='ebcdic1047'  
          outencoding='latin1');  
    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:

```sas
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 on HP Integrity Servers*
- `%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 on HP Integrity Servers*
- 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 on HP Integrity Servers*
- 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 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.

ing encoding-value
specifies an encoding value. For a list of encoding values, see Chapter 18, “Encoding Values for a SAS Session,” on page 561.

Details

The INENCODING= option is used to read SAS data sets in the SAS library. The OUTENCODING= option is used to write SAS data sets in the SAS 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 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 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.

□ For the COPY procedure, the default CLONE option uses the encoding attribute of the input data set instead of the encoding value specified on the OUTENCODING= option. For more information on CLONE and NOCLONE, see COPY Statement.

Note: This interaction does not apply when using SAS/CONNECT or SAS/SHARE.

See Also

“Overview: 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 461
“LOCALE System Option” on page 465
Data Set Options:
“ENCODING= Data Set Option” on page 43

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:

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 9.2, 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 28
Conceptual discussion of Chapter 2, “Locale for NLS,” on page 5

System Options:
“TRANTAB= System Option” on page 471
“LOCALE System Option” on page 465

Procedures:
Chapter 14, “The TRANTAB Procedure,” on page 515

Statements:

---

**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 27
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 before 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 UNIX and Windows, see Chapter 18, “Encoding Values for a SAS Session,” on page 561.

**EBCDIC-encoding-value**

For a list of valid values for EBCDIC encodings for z/OS, see Chapter 18, “Encoding Values for a SAS Session,” on page 561.

**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 9.2, 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, go to [http://rnd.sas.com/sites/i18n/i18ndocs/i18nsupport/Pages/](http://rnd.sas.com/sites/i18n/i18ndocs/i18nsupport/Pages/)
Background

In SAS 9 and 9.2, 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.

Before 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 method 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 27

- Statements:
  - LIBNAME in SAS/SHARE User’s Guide
TRANSCODE= Option

Specifies an attribute in the ATTRIB statement (which associates a format, informat, label, and 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 27.
Default: YES

Restriction: The TRANSCODE=NO attribute is not supported by some SAS Workspace Server clients. Variables with TRANSCODE=NO are not returned in SAS 9.2. Before SAS 9.2, variables with TRANSCODE=NO are transcoded. Prior releases of SAS cannot access a SAS 9.2 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 prints 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 is 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.

```sas
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 */
  rcl = vtranscode(z);
  put rcl=;
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.

```sas
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 */
  rcl = vtranscode(z);
  put rcl=;
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 301
- "VTRANSCODEX Function" on page 302
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

\[ \text{TRANTAB} = (\text{translation-table}) \]

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 TRANTAB= option for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases.

Options

\text{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 \text{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 28
Chapter 2, “Locale for NLS,” on page 5

System Options:

“TRANTAB= System Option” on page 471
“LOCALE System Option” on page 465

Procedures:

Chapter 14, “The TRANTAB Procedure,” on page 515
XMLENCODING= Option

Overwrites 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 555.

Default: SAS uses the current session encoding.

See Also

Statements:

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 9.2 supports the TRANTAB statement for backward compatibility. However, using the LOCALE= system option is preferred in later SAS releases.

For more information, see TS-639, Data Conversion Issues in V6–V8. This technical support note provides information for customers using non-English languages


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 27

System Options:
- “TRANTAB= System Option” on page 471

Procedures:
- Chapter 14, “The TRANTAB Procedure,” on page 515
- CPORT in Base SAS Procedures Guide
- UPLOAD in SAS/CONNECT User’s Guide
- DOWNLOAD in SAS/CONNECT User’s Guide
PART 3

Procedures for NLS

Chapter 13 ........ The DBCSTAB Procedure 509
Chapter 14 ........ The TRANTAB Procedure 515
Overview: DBCSTAB Procedure

The DBCSTAB procedure produces conversion tables for the double-byte character sets that SAS supports.

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, and then you can specify the use of the new table with the TRANTAB option.

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.
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, DBCSLANGL, and DBCSTYPE options are specified at startup.

The TRANTAB data set is created as follows:

```sas
data trantab;
  pcms='8342'x; dec='b9b3'x;
run;
```

```sas
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:

```sas
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 might 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 data set WORK.JA_JPN.

See Also

Functions:
   “KCVT Function” on page 268

Procedures:
   Chapter 14, “The TRANTAB Procedure,” on page 515

System Options:
   “TRANTAB= System Option” on page 471
   “DBCS System Option: UNIX, Windows, and z/OS” on page 456
   “DBCSLANG System Option: UNIX, Windows, and z/OS” on page 457
   “DBCSTYPE System Option: UNIX, Windows, and z/OS” on page 458
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.

---

### 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:

```sas
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 525 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 Representative 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 Representative, 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:

```sas
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:

```sas
proc sort sortseq=ascii;
```

You can also create a customized translation table with PROC TRANTAB and specify the new table with PROC SORT. This table is useful when you want to specify sorting sequences for languages other than U.S. English.

See Example 6 on page 536 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: UNIX, Windows, and z/OS” on page 470.

#### 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 might 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 503 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.

Note: For more information, see TS-706: How to use the %lswbatch macro [http://support.sas.com/techsup/technote/ts706.pdf].

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 might 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:

```sas
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 consists 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 525 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.


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.
The TRANTAB Procedure

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;

<table>
<thead>
<tr>
<th>Task</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set all positions in the translation table to zero</td>
<td>“CLEAR Statement” on page 521</td>
</tr>
<tr>
<td>Create an inverse of table 1</td>
<td>“INVERSE Statement” on page 521</td>
</tr>
<tr>
<td>Display a translation table in hexadecimal representation</td>
<td>“LIST Statement” on page 521</td>
</tr>
<tr>
<td>Load a translation table into memory for editing</td>
<td>“LOAD Statement” on page 522</td>
</tr>
<tr>
<td>Replace the characters in a translation table with specified values</td>
<td>“REPLACE Statement” on page 523</td>
</tr>
<tr>
<td>Save the translation table in your SASUSER.PROFILE catalog</td>
<td>“SAVE Statement” on page 524</td>
</tr>
<tr>
<td>Exchange table 1 with table 2</td>
<td>“SWAP Statement” on page 525</td>
</tr>
</tbody>
</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 9.2 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 eight 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.

```
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.

```
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 531

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 positions 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 526, Example 3 on page 529, and Example 4 on page 531

```
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 action 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 529 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 526 and Example 4 on page 531

```
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 538

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 ASCII translation table supplied by SAS.

Program

Set the options and specify a translation table.

```sas
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.

```sas
list both;
```
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.

```sas
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.

```sas
replace 0
'000102030405060708090a0b0c0d0e0f'x
'101112131415161718191a1b1c1d1e1f'x
'202122232425262728292a2b2c2d2e2f'x
'303132333435363738393a3b3c3d3e3f'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.

```sas
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.

```sas
list both;
```
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 534.) To edit table 2, you can use the SWAP statement with the REPLACE statement. (See Example 7 on page 538.)
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 526. 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.

```sas
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.

```sas
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 '10111213A500871819C6C51C1D1E1F'x
20 'C7FCE9E2E40A171BEAEBE8EFE0E050607'x
30 'C9E616F4F6F2FB04F06DCA2B6A7501A'x
40 '20E1EDF3FAF1D8AB8FA22E3C282B7C'x
50 '265FACDBCA1ABB5F521242A293BAC'x
60 '2D2F5FA66A6A626A62C255F3E3F'x
80 '2B61625666666692D2B6A62B2B'x
90 '2D6A5B5C6666667071722D6262B2D5'x
A0 '2D7E7374757677787877A2D2B2B2B2B'x
B0 '2B2B2B5F5FA655F5F5F5F5F5F5F5F'x
C0 '7B41424344454647484955F5F5F5F5F'x
D0 '7D4A4B4C4D4E5F505152F55F5F5F5F5F'x
E0 '5C8355555555555555555555555555'x
F0 '30313233343536373839B75F66E25F5F'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;
```
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 method differs from Example 3 on page 529 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.

```sas
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.

```sas
list one;
```

**SAS Log**

```sas
NOTE: Table specified is ASCII.
ASCII table 1:
  0123456789ABCDEF
  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 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACBCCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADDDBDCDDDEF'x
E0 'E0E1E2E3E4E5E6E7E8E9EAEABBCCBDEEFF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDEEFF'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."

```sas
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.

```sas
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.

```sas
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 '11112131415161718191A1B1C1D1E1F'x
  20 '202122232425262728292A2B2C2D2E2F'x
  30 '303132333435363738393A3B3C3D3E3F'x
  40 '404142434445464748494A4B4C4D4E4F'x
  50 '505152535455565758595A5B5C5D5E5F'x
  60 '606162636465666768696A6B6C6D6E6F'x
  70 '707172737475767778797A7B7C7D7E7F'x
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '90919293949596979899A9B9C9D9E9F'x
AO 'A0A1A2A3A4A5A6A7A8A9AAABACADAABF'x
BO 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'x
CO 'C0C1C2C3C4C5C6C7C8C9CABCBCBCDECF'x
DO 'D0D1D2D3D4D5D6D7D8D9DADBDCCDDEDF'x
EO 'E0E1E2E3E4E5E6E7E8E9EAEBCDEDEEEF'x
FO 'F0F1F2F3F4F5F6F7F8F9FABFBCDFDEFF'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 531. 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;
The TRANTAB Procedure

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 '606162636465666768696A6B6C6D6E6F'x
  70 '707172737475767778797A7B7C7D7E7F'x
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '909192939495969798999A9B9C9D9E9F'x
  A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
  B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBEF'x
  C0 'C0C1C2C3C4C5C6C7C8C9CACBCDCEEDCF'x
  D0 'D0D1D2D3D4D5D6D7D8D9DADDCCDDEDFF'x
  E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF'x
  F0 'F0F1F2F3F4F5F6F7F8F9F9FBFCDFFEFF'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 '600000000000000000000000000000000'x
  70 '000000000000000000000000000000000000000000'x
  80 '808182838485868788898A8B8C8D8E8F'x
  90 '909192939495969798999A9B9C9D9E9F'x
  A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
  B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBEF'x
  C0 'C0C1C2C3C4C5C6C7C8C9CACBCDCEEDCF'x
  D0 'D0D1D2D3D4D5D6D7D8D9DADDCCDDEDFF'x
  E0 'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEFF'x
  F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCDFDEFF'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 531, 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 might 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 languages.

---

**Program**

**Set the SAS system options.**

```sas
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.

```sas
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.

```
<table>
<thead>
<tr>
<th>Default Sort Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Always</td>
</tr>
<tr>
<td>Forever</td>
</tr>
<tr>
<td>Later</td>
</tr>
<tr>
<td>Yesterday</td>
</tr>
<tr>
<td>always</td>
</tr>
<tr>
<td>forever</td>
</tr>
<tr>
<td>later</td>
</tr>
<tr>
<td>yesterday</td>
</tr>
</tbody>
</table>
```

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 method 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.

<table>
<thead>
<tr>
<th>Customized Sort Sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
</tr>
<tr>
<td>Always</td>
</tr>
<tr>
<td>always</td>
</tr>
<tr>
<td>Forever</td>
</tr>
<tr>
<td>forever</td>
</tr>
<tr>
<td>Later</td>
</tr>
<tr>
<td>later</td>
</tr>
<tr>
<td>Yesterday</td>
</tr>
<tr>
<td>yesterday</td>
</tr>
</tbody>
</table>

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;
```
The TRANTAB Procedure

The Original UPPER Translation Table

<table>
<thead>
<tr>
<th>NOTE: Table specified is UPPER.</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPPER table 1:</td>
</tr>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>00 '00102030405060708090A0B0C0D0E0F'x ---</td>
</tr>
<tr>
<td>10 '101112131415161718191A1B1C1D1E1F'x</td>
</tr>
<tr>
<td>20 '202122232425262728292A2B2C2D2E2F'x</td>
</tr>
<tr>
<td>30 '303132333435363738393A3B3C3D3E3F'x</td>
</tr>
<tr>
<td>40 '404142434445464748494A4B4C4D4E4F'x</td>
</tr>
<tr>
<td>50 '505152535455565758595A5B5C5D5E5F'x</td>
</tr>
<tr>
<td>60 '606162636465666768696A6B6C6D6E6F'x</td>
</tr>
<tr>
<td>70 '707172737475767778797A7B7C7D7E7F'x</td>
</tr>
<tr>
<td>80 '808182838485868788898A8B8C8D8E8F'x</td>
</tr>
<tr>
<td>90 '90919293949596979899A9B9C9D9E9F'x</td>
</tr>
<tr>
<td>A0 'A0A1A2A3A4A5A6A7A8A9AAABACDABAEAF'x</td>
</tr>
<tr>
<td>B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDEBF'x</td>
</tr>
<tr>
<td>C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x</td>
</tr>
<tr>
<td>D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x</td>
</tr>
<tr>
<td>E0 'E0E1E2E3E4E5E6E7E8E9EAEBCEDEEFF'x</td>
</tr>
<tr>
<td>F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFFFF'x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UPPER table 2:</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7 8 9 A B C D E F</td>
</tr>
<tr>
<td>-------------------------------</td>
</tr>
<tr>
<td>00 '00102030405060708090A0B0C0D0E0F'x ---</td>
</tr>
<tr>
<td>10 '101112131415161718191A1B1C1D1E1F'x</td>
</tr>
<tr>
<td>20 '202122232425262728292A2B2C2D2E2F'x</td>
</tr>
<tr>
<td>30 '303132333435363738393A3B3C3D3E3F'x</td>
</tr>
<tr>
<td>40 '404142434445464748494A4B4C4D4E4F'x</td>
</tr>
<tr>
<td>50 '505152535455565758595A5B5C5D5E5F'x</td>
</tr>
<tr>
<td>60 '606162636465666768696A6B6C6D6E6F'x</td>
</tr>
<tr>
<td>70 '707172737475767778797A7B7C7D7E7F'x</td>
</tr>
<tr>
<td>80 '808182838485868788898A8B8C8D8E8F'x</td>
</tr>
<tr>
<td>90 '90919293949596979899A9B9C9D9E9F'x</td>
</tr>
<tr>
<td>A0 'A0A1A2A3A4A5A6A7A8A9AAABACDABAEAF'x</td>
</tr>
<tr>
<td>B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDBDEBF'x</td>
</tr>
<tr>
<td>C0 'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'x</td>
</tr>
<tr>
<td>D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'x</td>
</tr>
<tr>
<td>E0 'E0E1E2E3E4E5E6E7E8E9EAEBCEDEEFF'x</td>
</tr>
<tr>
<td>F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFFFF'x</td>
</tr>
</tbody>
</table>

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';
```
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.

```sas
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.

```sas
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 '000A0203040506070809A0B0C0D0E0F'x —
10 '10112131415161718191A1B1C1D1E1F'x
20 '202122232425262728292A2B2C2D2E2F'x
30 '303132333435363738393A3B3C3D3E3F'x
40 '404142434445464748494A4B4C4D4E4F'x
50 '505152535455565758595A5B5C5D5E5F'x
60 '606162636465666768696A6B6C6D6E6F'x
70 '707172737475767778797A7B7C7D7E7F'x
80 '808182838485868788898A8B8C8D8E8F'x
90 '90919293949596979899A9B9C9D9E9F'x
A0 'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'x
B0 'B0B1B2B3B4B5B6B7B8B9BABBBCBDDEBF'x
C0 'C0C1C2C3C4C5C6C7C8C9CACCBCDCECF'x
D0 'D0D1D2D3D4D5D6D7D8D9DADBDCDEFDF'x
E0 'E0E1E2E3E4E5E6E7E8E9EABBCBDEEFF'x
F0 'F0F1F2F3F4F5F6F7F8F9FAFBFCFDDEFF'x
```
### See Also

Conceptual discussion about “Transcoding and Translation Tables” on page 28

**System Options:**

“`TRANTAB= System Option`” on page 471
Values for Locale, Encoding, and Transcoding

Chapter 15. . . . . . Values for the LOCATE= System Option 545
Chapter 16. . . . . . SAS System Options for Processing DBCS Data 553
Chapter 17. . . . . . Encoding Values in SAS Language Elements 555
Chapter 18. . . . . . Encoding Values for a SAS Session 561
### Table 15.1 Values for the LOCALE= System Option

<table>
<thead>
<tr>
<th>SAS Name</th>
<th>Posix Locale</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans_SouthAfrica</td>
<td>af_ZA</td>
<td>Afrikaans</td>
</tr>
<tr>
<td>Albanian_Albania</td>
<td>sq.AL</td>
<td>Albanian</td>
</tr>
<tr>
<td>Arabic_Algeria</td>
<td>ar_DZ</td>
<td></td>
</tr>
<tr>
<td>Arabic_Bahrain</td>
<td>ar_BH</td>
<td></td>
</tr>
<tr>
<td>Arabic_Egypt</td>
<td>ar_EG</td>
<td></td>
</tr>
<tr>
<td>Arabic_India</td>
<td>ar_IN</td>
<td></td>
</tr>
<tr>
<td>Arabic_Iraq</td>
<td>ar_IQ</td>
<td></td>
</tr>
<tr>
<td>Arabic_Jordan</td>
<td>ar_JO</td>
<td></td>
</tr>
<tr>
<td>Arabic_Kuwait</td>
<td>ar_KW</td>
<td></td>
</tr>
<tr>
<td>Arabic_Lebanon</td>
<td>ar_LB</td>
<td></td>
</tr>
<tr>
<td>Arabic_Libya</td>
<td>ar_LY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Morocco</td>
<td>ar_MA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Oman</td>
<td>ar_OM</td>
<td></td>
</tr>
<tr>
<td>Arabic_Qatar</td>
<td>ar_QA</td>
<td></td>
</tr>
<tr>
<td>Arabic_SaudiArabia</td>
<td>ar_SA</td>
<td></td>
</tr>
<tr>
<td>Arabic_Sudan</td>
<td>ar_SD</td>
<td></td>
</tr>
<tr>
<td>Arabic_Syria</td>
<td>ar_SY</td>
<td></td>
</tr>
<tr>
<td>Arabic_Tunisia</td>
<td>ar_TN</td>
<td></td>
</tr>
<tr>
<td>Arabic_UnitedArabEmirates</td>
<td>ar_AE</td>
<td>Arabic</td>
</tr>
<tr>
<td>SAS Name</td>
<td>Posix Locale</td>
<td>Alias</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Arabic_Yemen</td>
<td>ar_YE</td>
<td></td>
</tr>
<tr>
<td>Bengali_India</td>
<td>bn_IN</td>
<td>Bengali</td>
</tr>
<tr>
<td>Bosnian_BosniaHerzegovina</td>
<td>bs_BA</td>
<td></td>
</tr>
<tr>
<td>Bulgarian_Bulgaria</td>
<td>bg_BG</td>
<td>Bulgarian</td>
</tr>
<tr>
<td>Byelorussian_Belarus</td>
<td>be_BY</td>
<td>Byelorussian</td>
</tr>
<tr>
<td>Catalan_Spain</td>
<td>ca_ES</td>
<td>Catalan</td>
</tr>
<tr>
<td>Chinese_China</td>
<td>zh_CN</td>
<td>Chinese</td>
</tr>
<tr>
<td>Chinese_HongKong</td>
<td>zh_HK</td>
<td></td>
</tr>
<tr>
<td>Chinese_Macau</td>
<td>zh_MO</td>
<td></td>
</tr>
<tr>
<td>Chinese_Singapore</td>
<td>zh_SG</td>
<td></td>
</tr>
<tr>
<td>Chinese_Taiwan</td>
<td>zh_TW</td>
<td></td>
</tr>
<tr>
<td>Cornish_UnitedKingdom</td>
<td>kw_GB</td>
<td>Cornish</td>
</tr>
<tr>
<td>Croatian_BosniaHerzegovina</td>
<td>hr_BA</td>
<td></td>
</tr>
<tr>
<td>Croatian_Croatia</td>
<td>hr_HR</td>
<td>Croatian</td>
</tr>
<tr>
<td>Czech_CzechRepublic</td>
<td>cs_CZ</td>
<td>Czech</td>
</tr>
<tr>
<td>Danish_Denmark</td>
<td>da_DK</td>
<td>Danish</td>
</tr>
<tr>
<td>Dutch_Belgium</td>
<td>nl_BE</td>
<td></td>
</tr>
<tr>
<td>Dutch_Netherlands</td>
<td>nl_NL</td>
<td>Dutch</td>
</tr>
<tr>
<td>English_Australia</td>
<td>en_AU</td>
<td></td>
</tr>
<tr>
<td>English_Belgium</td>
<td>en_BE</td>
<td></td>
</tr>
<tr>
<td>English_Botswana</td>
<td>en_BW</td>
<td></td>
</tr>
<tr>
<td>English_Canada</td>
<td>en_CA</td>
<td></td>
</tr>
<tr>
<td>English_Caribbean</td>
<td>en_CB</td>
<td></td>
</tr>
<tr>
<td>English_HongKong</td>
<td>en_HK</td>
<td></td>
</tr>
<tr>
<td>English_India</td>
<td>en_IN</td>
<td></td>
</tr>
<tr>
<td>English_Ireland</td>
<td>en_IE</td>
<td></td>
</tr>
<tr>
<td>English_Jamaica</td>
<td>en_JM</td>
<td></td>
</tr>
<tr>
<td>English_NewZealand</td>
<td>en_NZ</td>
<td></td>
</tr>
<tr>
<td>English_Philippines</td>
<td>en_PH</td>
<td></td>
</tr>
<tr>
<td>English_Singapore</td>
<td>en_SG</td>
<td></td>
</tr>
<tr>
<td>English_SouthAfrica</td>
<td>en_ZA</td>
<td></td>
</tr>
<tr>
<td>English_UnitedKingdom</td>
<td>en_GB</td>
<td></td>
</tr>
<tr>
<td>English_UnitedStates</td>
<td>en_US</td>
<td>English</td>
</tr>
<tr>
<td>English_Zimbabwe</td>
<td>en_ZW</td>
<td></td>
</tr>
<tr>
<td>Estonian_Estonia</td>
<td>et_EE</td>
<td>Estonian</td>
</tr>
<tr>
<td>Faroese_FaroerIshlands</td>
<td>fo_FO</td>
<td>Faroese</td>
</tr>
<tr>
<td>Finnish_Finland</td>
<td>fi_FI</td>
<td>Finnish</td>
</tr>
<tr>
<td>SAS Name</td>
<td>Posix Locale</td>
<td>Alias</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>French_Belgium</td>
<td>fr_BE</td>
<td></td>
</tr>
<tr>
<td>French_Canada</td>
<td>fr_CA</td>
<td></td>
</tr>
<tr>
<td>French_France</td>
<td>fr_FR</td>
<td>French</td>
</tr>
<tr>
<td>French_Luxembourg</td>
<td>fr_LU</td>
<td></td>
</tr>
<tr>
<td>French_Switzerland</td>
<td>fr_CH</td>
<td></td>
</tr>
<tr>
<td>German_Austria</td>
<td>de_AT</td>
<td></td>
</tr>
<tr>
<td>German_Germany</td>
<td>de_DE</td>
<td>German</td>
</tr>
<tr>
<td>German_Liechtenstein</td>
<td>de_LI</td>
<td></td>
</tr>
<tr>
<td>German_Luxembourg</td>
<td>de_LU</td>
<td></td>
</tr>
<tr>
<td>German_Switzerland</td>
<td>de_CH</td>
<td></td>
</tr>
<tr>
<td>Greek_Greece</td>
<td>el_GR</td>
<td>Greek</td>
</tr>
<tr>
<td>Greenlandic_Greenland</td>
<td>kl_GL</td>
<td>Greenlandic</td>
</tr>
<tr>
<td>Hebrew_Israel</td>
<td>he_IL</td>
<td>Hebrew</td>
</tr>
<tr>
<td>Hindi_India</td>
<td>hi_IN</td>
<td>Hindi</td>
</tr>
<tr>
<td>Hungarian_Hungary</td>
<td>hu_HU</td>
<td>Hungarian</td>
</tr>
<tr>
<td>Icelandic_Iceland</td>
<td>ia_IS</td>
<td>Icelandic</td>
</tr>
<tr>
<td>Indonesian_Indonesia</td>
<td>id_ID</td>
<td>Indonesian</td>
</tr>
<tr>
<td>Italian_Italy</td>
<td>it_IT</td>
<td>Italian</td>
</tr>
<tr>
<td>Italian_Switzerland</td>
<td>it_CH</td>
<td></td>
</tr>
<tr>
<td>Japanese_Japan</td>
<td>ja_JP</td>
<td>Japanese</td>
</tr>
<tr>
<td>Korean_Korea</td>
<td>ko_KR</td>
<td>Korean</td>
</tr>
<tr>
<td>Latvian_Latvia</td>
<td>lv_LV</td>
<td>Latvian</td>
</tr>
<tr>
<td>Lithuanian_Lithuania</td>
<td>lt_LT</td>
<td>Lithuanian</td>
</tr>
<tr>
<td>Macedonian_Macedonia</td>
<td>mk_MK</td>
<td>Macedonian</td>
</tr>
<tr>
<td>Malay_Malaysia</td>
<td>ms_MY</td>
<td>Malay</td>
</tr>
<tr>
<td>Maltese_Malta</td>
<td>mt_MT</td>
<td>Maltese</td>
</tr>
<tr>
<td>ManxGaelic_UnitedKingdom</td>
<td>gv_GB</td>
<td>ManxGaelic</td>
</tr>
<tr>
<td>Marathi_India</td>
<td>mr_IN</td>
<td>Marathi</td>
</tr>
<tr>
<td>NorwegianBokmal_Norway</td>
<td>nb_NO</td>
<td>NorwegianBokmal</td>
</tr>
<tr>
<td>NorwegianNynorsk_Norway</td>
<td>nn_NO</td>
<td>NorwegianNynorsk</td>
</tr>
<tr>
<td>Norwegian_Norway</td>
<td>no_NO</td>
<td>Norwegian</td>
</tr>
<tr>
<td>Persian_India</td>
<td>fa_IN</td>
<td>Persian</td>
</tr>
<tr>
<td>Persian_Iran</td>
<td>fa_IR</td>
<td>Persian</td>
</tr>
<tr>
<td>Polish_Poland</td>
<td>pl_PL</td>
<td>Polish</td>
</tr>
<tr>
<td>Portuguese_Brazil</td>
<td>pt_BR</td>
<td></td>
</tr>
<tr>
<td>Portuguese_Portugal</td>
<td>pt_PT</td>
<td>Portuguese</td>
</tr>
<tr>
<td>Romanian_Romania</td>
<td>ro_RO</td>
<td>Romanian</td>
</tr>
<tr>
<td>SAS Name</td>
<td>Posix Locale</td>
<td>Alias</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Russian_Russia</td>
<td>ru_RU</td>
<td>Russian</td>
</tr>
<tr>
<td>Russian_Ukraine</td>
<td>ru_UA</td>
<td></td>
</tr>
<tr>
<td>Serbian_BosniaHerzegovina</td>
<td>sr_BA</td>
<td></td>
</tr>
<tr>
<td>Serbian_Montenegro</td>
<td>sr_ME</td>
<td></td>
</tr>
<tr>
<td>Serbian_Serbia</td>
<td>sr_RS</td>
<td></td>
</tr>
<tr>
<td>Serbian_Yugoslavia</td>
<td>sr_YU</td>
<td>Serbian</td>
</tr>
<tr>
<td>SerboCroatian_BosniaHerzegovina</td>
<td>sh_BA</td>
<td></td>
</tr>
<tr>
<td>SerboCroatian_Montenegro</td>
<td>sh_ME</td>
<td></td>
</tr>
<tr>
<td>SerboCroatian_Serbia</td>
<td>sh_RS</td>
<td></td>
</tr>
<tr>
<td>Slovak_Slovakia</td>
<td>sk_SK</td>
<td>Slovak</td>
</tr>
<tr>
<td>Slovenian_Slovenia</td>
<td>sl_SI</td>
<td>Slovenian</td>
</tr>
<tr>
<td>Spanish_Argentina</td>
<td>es_AR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Bolivia</td>
<td>es_BO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Chile</td>
<td>es_CL</td>
<td></td>
</tr>
<tr>
<td>Spanish_Colombia</td>
<td>es_CO</td>
<td></td>
</tr>
<tr>
<td>Spanish_CostaRica</td>
<td>es_CR</td>
<td></td>
</tr>
<tr>
<td>Spanish_DominicanRepublic</td>
<td>es_DO</td>
<td></td>
</tr>
<tr>
<td>Spanish_Ecuador</td>
<td>es_EC</td>
<td></td>
</tr>
<tr>
<td>Spanish_ElSalvador</td>
<td>es_SV</td>
<td></td>
</tr>
<tr>
<td>Spanish_Guatemala</td>
<td>es_GT</td>
<td></td>
</tr>
<tr>
<td>Spanish_Honduras</td>
<td>es_HN</td>
<td></td>
</tr>
<tr>
<td>Spanish_Mexico</td>
<td>es_MX</td>
<td></td>
</tr>
<tr>
<td>Spanish_Nicaragua</td>
<td>es_NI</td>
<td></td>
</tr>
<tr>
<td>Spanish_Panama</td>
<td>es_PA</td>
<td></td>
</tr>
<tr>
<td>Spanish_Paraguay</td>
<td>es_PY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Paraguay</td>
<td>es_PY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Peru</td>
<td>es_PE</td>
<td></td>
</tr>
<tr>
<td>Spanish_PuertoRico</td>
<td>es_PR</td>
<td></td>
</tr>
<tr>
<td>Spanish_Spain</td>
<td>es_ES</td>
<td>Spanish</td>
</tr>
<tr>
<td>Spanish_UnitedStates</td>
<td>es_US</td>
<td></td>
</tr>
<tr>
<td>Spanish_Uruguay</td>
<td>es_UY</td>
<td></td>
</tr>
<tr>
<td>Spanish_Venezuela</td>
<td>es_VE</td>
<td></td>
</tr>
<tr>
<td>Swedish_Sweden</td>
<td>sv_SE</td>
<td>Swedish</td>
</tr>
<tr>
<td>Tamil_India</td>
<td>ta_IN</td>
<td>Tamil</td>
</tr>
<tr>
<td>Telugu_India</td>
<td>te_IN</td>
<td>Telugu</td>
</tr>
<tr>
<td>Thai_Thailand</td>
<td>th_TH</td>
<td>Thai</td>
</tr>
<tr>
<td>Turkish_Turkey</td>
<td>tr_TR</td>
<td>Turkish</td>
</tr>
</tbody>
</table>
Table 15.2 lists the valid Posix values and the default settings for the ENCODING= option, by operating environment. The settings for DFLANG, DATESTYLE, and PAPERSIZE system options are set automatically.

Here is an 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 15.2  Default Values for the ENCODING, DFLANG, DATESTYLE, and PAPERSIZE System Options
Based on the LOCALE= System Option

<table>
<thead>
<tr>
<th>SAS Name</th>
<th>Posix Locale</th>
<th>Alias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ukrainian_Ukraine</td>
<td>uk_UA</td>
<td>Ukrainian</td>
</tr>
<tr>
<td>Vietnamese_Vietnam</td>
<td>vi_VN</td>
<td>Vietnamese</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Posix Locale</th>
<th>Windows Encoding</th>
<th>UNIX Encoding</th>
<th>z/OS Encoding</th>
<th>DFLANG=</th>
<th>DATESTYLE=</th>
<th>PAPERSIZE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>af_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ar_AE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_BH</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_DZ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_EG</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_IQ</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_JO</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_KW</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LB</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_LY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_MA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_OM</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_QA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SA</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SD</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_SY</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_TN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ar_YE</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-425</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>be_BY</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>bg_BG</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>bn_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>Posix Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>ca_ES</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>cs_CZ</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Czech</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>da_DK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Danish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_AT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Swiss</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>German</td>
<td></td>
<td></td>
</tr>
<tr>
<td>de_DE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>de_LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1141</td>
<td>German</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>el_GR</td>
<td>wgnreek</td>
<td>greek</td>
<td>open_ed-875</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_AU</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_BW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>en_CB</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>en_GB</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_HK</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_IN</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_JM</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>en_NZ</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_PH</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_SG</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1146</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>MDY</td>
<td>A4</td>
</tr>
<tr>
<td>en_ZA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>en_ZW</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_AR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_BO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_CR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_DO</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_EC</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_ES</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1145</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_GT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_HN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>Posix Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>---------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>es_MX</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_NI</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_PY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_SV</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>MDY</td>
<td>letter</td>
</tr>
<tr>
<td>es_US</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>es_UY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>es_VE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>Spanish</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>et_EE</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1122</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IN</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fa_IR</td>
<td>warabic</td>
<td>arabic</td>
<td>open_ed-1097</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>fi_FI</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1143</td>
<td>Finnish</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fo_FO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_BE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_CA</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>French</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>fr_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Swiss_French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr_FR</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>fr-LU</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1147</td>
<td>French</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>gv_GB</td>
<td>wlatin1</td>
<td>latin8</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>he_IL</td>
<td>whebrew</td>
<td>hebrew</td>
<td>open_ed-424</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hi_IN</td>
<td>PCISCI806</td>
<td>latin1</td>
<td>open_ed-1137</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>hr_HR</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Croatian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>hu_HU</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Hungarian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ia_IS</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>id_ID</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_CH</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1148</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>it_IT</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1144</td>
<td>Italian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ja JP</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Japanese</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kl_GL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ko_KR</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>kw_GB</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>lt_LT</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1112</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>lv_LV</td>
<td>wbaltic</td>
<td>latin6</td>
<td>open_ed-1112</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>mk_MK</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1154</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>Posix Locale</td>
<td>Windows Encoding</td>
<td>UNIX Encoding</td>
<td>z/OS Encoding</td>
<td>DFLANG=</td>
<td>DATESTYLE=</td>
<td>PAPERSIZE=</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>mr_IN</td>
<td>PCISCII806</td>
<td>latin1</td>
<td>open_ed-1137</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ms_MY</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>mt_MT</td>
<td>wlatin1</td>
<td>latin3</td>
<td>open_ed-905</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nb_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_BE</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1148</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nl_NL</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1140</td>
<td>Dutch</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>nn_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>no_NO</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1142</td>
<td>Norwegian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>pl_PL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Polish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>pt_BR</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-275</td>
<td>Portuguese</td>
<td>DMY</td>
<td>letter</td>
</tr>
<tr>
<td>pt_PT</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1140</td>
<td>Portuguese</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ro_RO</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_RU</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>ru_UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1154</td>
<td>Russian</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sh_YU</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sk_SK</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sl_SL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-870</td>
<td>Slovenian</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sr_YU</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>sq_AL</td>
<td>wlatin2</td>
<td>latin2</td>
<td>open_ed-1153</td>
<td>English</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>sv_SE</td>
<td>wlatin1</td>
<td>latin9</td>
<td>open_ed-1143</td>
<td>Swedish</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>ta_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>te_IN</td>
<td>wlatin1</td>
<td>latin1</td>
<td>open_ed-1047</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>th_TH</td>
<td>pcoem874</td>
<td>thai</td>
<td>open_ed-838</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>tr_TR</td>
<td>wturkish</td>
<td>latin5</td>
<td>open_ed-1026</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>uk-UA</td>
<td>wcyrillic</td>
<td>cyrillic</td>
<td>open_ed-1025</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>vi_VN</td>
<td>vvietnamese</td>
<td>latin1</td>
<td>open_ed-1130</td>
<td>English</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_CN</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_HK</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_MO</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
<tr>
<td>zh_SG</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>DMY</td>
<td>A4</td>
</tr>
<tr>
<td>zh_TW</td>
<td>not applicable</td>
<td>not applicable</td>
<td>not applicable</td>
<td>Locale</td>
<td>YMD</td>
<td>A4</td>
</tr>
</tbody>
</table>
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. △

Note: If an encoding value contains a hyphen (-), enclose the encoding value in quotation marks. △

Table 16.1 DBCS Supported Values for the DBCSLANG= and DBCSTYPE= System Options

<table>
<thead>
<tr>
<th>DBCSLANG=</th>
<th>z/OS DBCSTYPE=</th>
<th>UNIX DBCSTYPE=</th>
<th>Windows DBCSTYPE=</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chinese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Chinese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Japanese</td>
<td>pcibm</td>
<td>euc</td>
<td>sjis</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Japanese</td>
<td>not applicable</td>
<td>sjis</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>ibm</td>
<td>pcibm</td>
<td>pcms</td>
</tr>
<tr>
<td>DBCSLANG=</td>
<td>z/OS DBCSTYPE=</td>
<td>UNIX DBCSTYPE=</td>
<td>Windows DBCSTYPE=</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>dec</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Korean</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>ibm</td>
<td>dec</td>
<td>pcms</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>pcibm</td>
<td>euc</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>hp15</td>
<td>not applicable</td>
</tr>
<tr>
<td>Taiwanese</td>
<td>not applicable</td>
<td>pcms</td>
<td>not applicable</td>
</tr>
</tbody>
</table>
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 27.

SBCS, DBCS, and Unicode Encoding Values for Transcoding Data

Table 17.1 presents a list of SBCS, DBCS, and Unicode encoding values for transcoding data for all operating environments. The encoding values in Table 16.1 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>
<thead>
<tr>
<th>Encoding Name</th>
<th>Short Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic</td>
<td>aara</td>
<td>Arabic Macintosh</td>
</tr>
<tr>
<td>greek</td>
<td>agrk</td>
<td>Greek Macintosh</td>
</tr>
<tr>
<td>hebrew</td>
<td>aheb</td>
<td>Hebrew Macintosh</td>
</tr>
<tr>
<td>iceland</td>
<td>aice</td>
<td>Icelandic Macintosh</td>
</tr>
<tr>
<td>any</td>
<td>anye</td>
<td>no transcoding is specified</td>
</tr>
<tr>
<td>arabic</td>
<td>arab</td>
<td>Arabic ISO</td>
</tr>
<tr>
<td>roman</td>
<td>arom</td>
<td>Roman Macintosh</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ansi</td>
<td></td>
<td>enables you to create a data set that is compatible with all ASCII encodings</td>
</tr>
<tr>
<td>aturkish</td>
<td>atur</td>
<td>Turkish Macintosh</td>
</tr>
<tr>
<td>aukrainian</td>
<td>aukr</td>
<td>Ukrainian Macintosh</td>
</tr>
<tr>
<td>big5</td>
<td>big5</td>
<td>Traditional Chinese Big5</td>
</tr>
<tr>
<td>cyrillic</td>
<td>cyril</td>
<td>Cyrillic ISO</td>
</tr>
<tr>
<td>dec-cn</td>
<td>jums</td>
<td>Simplified Chinese DEC</td>
</tr>
<tr>
<td>dec-jp</td>
<td>jvms</td>
<td>Japanese DEC</td>
</tr>
<tr>
<td>dec-tw</td>
<td>yvms</td>
<td>Traditional Chinese DEC</td>
</tr>
<tr>
<td>ebcdic037</td>
<td>e037</td>
<td>North American EBCDIC</td>
</tr>
<tr>
<td>ebcdic275</td>
<td>e275</td>
<td>Brazil EBCDIC</td>
</tr>
<tr>
<td>ebcdic424</td>
<td>e424</td>
<td>Hebrew EBCDIC</td>
</tr>
<tr>
<td>ebcdic425</td>
<td>e425</td>
<td>Arabic EBCDIC</td>
</tr>
<tr>
<td>ebcdic500</td>
<td>e500</td>
<td>International EBCDIC</td>
</tr>
<tr>
<td>ebcdic838</td>
<td>e838</td>
<td>Thai EBCDIC</td>
</tr>
<tr>
<td>ebcdic870</td>
<td>e870</td>
<td>Central European EBCDIC</td>
</tr>
<tr>
<td>ebcdic875</td>
<td>e875</td>
<td>Greek EBCDIC</td>
</tr>
<tr>
<td>ebcdic924</td>
<td>e924</td>
<td>European EBCDIC</td>
</tr>
<tr>
<td>ebcdic1025</td>
<td>ecyr</td>
<td>Cyrillic EBCDIC</td>
</tr>
<tr>
<td>ebcdic1026</td>
<td>etur</td>
<td>Turkish EBCDIC</td>
</tr>
<tr>
<td>ebcdic1047</td>
<td>elat</td>
<td>Western EBCDIC</td>
</tr>
<tr>
<td>ebcdic1112</td>
<td>ebal</td>
<td>Baltic EBCDIC</td>
</tr>
<tr>
<td>ebcdic1122</td>
<td>eest</td>
<td>Estonian EBCDIC</td>
</tr>
<tr>
<td>ebcdic1130</td>
<td>evie</td>
<td>Vietnamese EBCDIC</td>
</tr>
<tr>
<td>ebcdic1140</td>
<td>e140</td>
<td>North American EBCDIC</td>
</tr>
<tr>
<td>ebcdic1141</td>
<td>e141</td>
<td>Austria/Germany EBCDIC</td>
</tr>
<tr>
<td>ebcdic1142</td>
<td>e142</td>
<td>Denmark/Norway EBCDIC</td>
</tr>
<tr>
<td>ebcdic1143</td>
<td>e143</td>
<td>Finland/Sweden EBCDIC</td>
</tr>
<tr>
<td>ebcdic1144</td>
<td>e144</td>
<td>Italy EBCDIC</td>
</tr>
<tr>
<td>ebcdic1145</td>
<td>e145</td>
<td>Spain EBCDIC</td>
</tr>
<tr>
<td>ebcdic1146</td>
<td>e146</td>
<td>United Kingdom EBCDIC</td>
</tr>
<tr>
<td>ebcdic1147</td>
<td>e147</td>
<td>France EBCDIC</td>
</tr>
<tr>
<td>ebcdic1148</td>
<td>e148</td>
<td>International EBCDIC</td>
</tr>
<tr>
<td>ebedicany</td>
<td>eany</td>
<td>enables you to create a data set that is compatible with all EBCDIC encodings</td>
</tr>
<tr>
<td>euc-cn</td>
<td>zeuc</td>
<td>Simplified Chinese EUC</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>euc-jp</td>
<td>jeuc</td>
<td>Japanese EUC</td>
</tr>
<tr>
<td>euc-kr</td>
<td>keuc</td>
<td>Korean EUC</td>
</tr>
<tr>
<td>euc-tw</td>
<td>yeuc</td>
<td>Traditional Chinese EUC</td>
</tr>
<tr>
<td>fujitsu-cn</td>
<td>zfuj</td>
<td>Simplified Chinese FACOM</td>
</tr>
<tr>
<td>fujitsu-jp</td>
<td>jfuj</td>
<td>Japanese FACOM</td>
</tr>
<tr>
<td>fujitsu-ko</td>
<td>kfuj</td>
<td>Korean FACOM</td>
</tr>
<tr>
<td>fujitsu-tw</td>
<td>yfuj</td>
<td>Traditional Chinese FACOM</td>
</tr>
<tr>
<td>greek</td>
<td>grek</td>
<td>Greek ISO</td>
</tr>
<tr>
<td>hebrew</td>
<td>hebr</td>
<td>Hebrew ISO</td>
</tr>
<tr>
<td>hitachi-cn</td>
<td>zhit</td>
<td>Simplified Chinese HITAC</td>
</tr>
<tr>
<td>hitachi-jp</td>
<td>jhit</td>
<td>Japanese HITAC</td>
</tr>
<tr>
<td>hitachi-ko</td>
<td>khit</td>
<td>Korean HITAC</td>
</tr>
<tr>
<td>hitachi-tw</td>
<td>yhit</td>
<td>Traditional Chinese HITAC</td>
</tr>
<tr>
<td>hitsas-jp</td>
<td>jhts</td>
<td>Japanese XHITAC</td>
</tr>
<tr>
<td>hitsas-ko</td>
<td>khts</td>
<td>Korean XHITAC</td>
</tr>
<tr>
<td>hitsas-tw</td>
<td>yhts</td>
<td>Traditional Chinese XHITAC</td>
</tr>
<tr>
<td>hp15-tw</td>
<td>yhpx</td>
<td>Traditional Chinese HP15</td>
</tr>
<tr>
<td>ibm-1381</td>
<td>zpce</td>
<td>Simplified Chinese PCIBM</td>
</tr>
<tr>
<td>ibm-930</td>
<td>j930</td>
<td>Japanese Katakana - 930</td>
</tr>
<tr>
<td>ibm-933</td>
<td>kibm</td>
<td>Korean IBM</td>
</tr>
<tr>
<td>ibm-935</td>
<td>zibm</td>
<td>Simplified Chinese IBM</td>
</tr>
<tr>
<td>ibm-937</td>
<td>yibm</td>
<td>Traditional Chinese IBM</td>
</tr>
<tr>
<td>ibm-939</td>
<td>jibm</td>
<td>Japanese IBM</td>
</tr>
<tr>
<td>ibm-942</td>
<td>j942</td>
<td>Japanese PCIBM</td>
</tr>
<tr>
<td>ibm-949</td>
<td>kpce</td>
<td>Korean PCIBM</td>
</tr>
<tr>
<td>latin1</td>
<td>lat1</td>
<td>Western ISO</td>
</tr>
<tr>
<td>latin2</td>
<td>lat2</td>
<td>Central European ISO</td>
</tr>
<tr>
<td>latin5</td>
<td>lat5</td>
<td>Turkish ISO</td>
</tr>
<tr>
<td>latin6</td>
<td>lat6</td>
<td>Baltic ISO</td>
</tr>
<tr>
<td>latin9</td>
<td>lat9</td>
<td>European ISO</td>
</tr>
<tr>
<td>macos-1</td>
<td>jmac</td>
<td>Japanese PCMAC</td>
</tr>
<tr>
<td>macos-2</td>
<td>ymac</td>
<td>Traditional Chinese PCMAC</td>
</tr>
<tr>
<td>macos-3</td>
<td>kmac</td>
<td>Korean PCMAC</td>
</tr>
<tr>
<td>macos-25</td>
<td>zmac</td>
<td>Simplified Chinese PCMAC</td>
</tr>
<tr>
<td>ms-932</td>
<td>j932</td>
<td>Japanese PCMS</td>
</tr>
<tr>
<td>ms-936</td>
<td>zwin</td>
<td>Simplified Chinese PCMS</td>
</tr>
<tr>
<td>ms-949</td>
<td>kwin</td>
<td>Korean PCMS</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>ms-950</td>
<td>ywin</td>
<td>Traditional Chinese PCMS</td>
</tr>
<tr>
<td>msdos720</td>
<td>p720</td>
<td>Arabic MS-DOS</td>
</tr>
<tr>
<td>msdos737</td>
<td>p737</td>
<td>Greek MS-DOS</td>
</tr>
<tr>
<td>msdos775</td>
<td>p775</td>
<td>Baltic MS-DOS</td>
</tr>
<tr>
<td>open_ed-275</td>
<td>eobr</td>
<td>Brazil OpenEdition</td>
</tr>
<tr>
<td>open_ed-424</td>
<td>eoiw</td>
<td>Hebrew OpenEdition</td>
</tr>
<tr>
<td>open_ed-425</td>
<td>eo2</td>
<td>Arabic OpenEdition</td>
</tr>
<tr>
<td>open_ed-838</td>
<td>eoth</td>
<td>Thai OpenEdition</td>
</tr>
<tr>
<td>open_ed-870</td>
<td>eolz</td>
<td>Central European OpenEdition</td>
</tr>
<tr>
<td>open_ed-875</td>
<td>eoel</td>
<td>Greek OpenEdition</td>
</tr>
<tr>
<td>open_ed-924</td>
<td>eolt</td>
<td>European OpenEdition</td>
</tr>
<tr>
<td>open_ed-1025</td>
<td>eocy</td>
<td>Cyrillic OpenEdition</td>
</tr>
<tr>
<td>open_ed-1026</td>
<td>eotr</td>
<td>Turkish OpenEdition</td>
</tr>
<tr>
<td>open_ed-1047</td>
<td>eol1</td>
<td>Western OpenEdition</td>
</tr>
<tr>
<td>open_ed-1112</td>
<td>eobl</td>
<td>Baltic OpenEdition</td>
</tr>
<tr>
<td>open_ed-1122</td>
<td>eoet</td>
<td>Estonian OpenEdition</td>
</tr>
<tr>
<td>open_ed-1130</td>
<td>eovi</td>
<td>Vietnamese OpenEdition</td>
</tr>
<tr>
<td>open_ed-1140</td>
<td>eo40</td>
<td>North American OpenEdition</td>
</tr>
<tr>
<td>open_ed-1141</td>
<td>eo41</td>
<td>Austria/Germany OpenEdition</td>
</tr>
<tr>
<td>open_ed-1142</td>
<td>eo42</td>
<td>Denmark/Norway OpenEdition</td>
</tr>
<tr>
<td>open_ed-1143</td>
<td>eo43</td>
<td>Finland/Sweden OpenEdition</td>
</tr>
<tr>
<td>open_ed-1144</td>
<td>eo44</td>
<td>Italy OpenEdition</td>
</tr>
<tr>
<td>open_ed-1145</td>
<td>eo45</td>
<td>Spain OpenEdition</td>
</tr>
<tr>
<td>open_ed-1146</td>
<td>eo46</td>
<td>United Kingdom OpenEdition</td>
</tr>
<tr>
<td>open_ed-1147</td>
<td>eo47</td>
<td>France OpenEdition</td>
</tr>
<tr>
<td>open_ed-1148</td>
<td>eo48</td>
<td>International OpenEdition</td>
</tr>
<tr>
<td>pcoem437</td>
<td>p437</td>
<td>USA IBM-PC</td>
</tr>
<tr>
<td>pcoem850</td>
<td>p850</td>
<td>Western IBM-PC</td>
</tr>
<tr>
<td>pcoem852</td>
<td>p852</td>
<td>Central European IBM-PC</td>
</tr>
<tr>
<td>pcoem857</td>
<td>p857</td>
<td>Turkish IBM-PC</td>
</tr>
<tr>
<td>pcoem858</td>
<td>p858</td>
<td>European IBM-PC</td>
</tr>
<tr>
<td>pcoem860</td>
<td>p860</td>
<td>Portuguese MS-DOS</td>
</tr>
<tr>
<td>pcoem862</td>
<td>p862</td>
<td>Hebrew IBM-PC</td>
</tr>
<tr>
<td>pcoem863</td>
<td>p863</td>
<td>French Canadian IBM-PC</td>
</tr>
<tr>
<td>pcoem864</td>
<td>p864</td>
<td>Arabic IBM-PC</td>
</tr>
<tr>
<td>pcoem865</td>
<td>p865</td>
<td>Nordic IBM-PC</td>
</tr>
<tr>
<td>Encoding Name</td>
<td>Short Name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>pcoem866</td>
<td>p866</td>
<td>Cyrillic IBM-PC</td>
</tr>
<tr>
<td>pcoem869</td>
<td>p869</td>
<td>Greek IBM-PC</td>
</tr>
<tr>
<td>pcoem874</td>
<td>p874</td>
<td>Thai IBM-PC</td>
</tr>
<tr>
<td>pcoem921</td>
<td>p921</td>
<td>Baltic IBM-PC</td>
</tr>
<tr>
<td>pcoem922</td>
<td>p922</td>
<td>Estonia IBM-PC</td>
</tr>
<tr>
<td>pcoem1129</td>
<td>pvie</td>
<td>Vietnamese IBM-PC</td>
</tr>
<tr>
<td>shift-jis</td>
<td>sjis</td>
<td>Japanese SJIS</td>
</tr>
<tr>
<td>thai</td>
<td>thai</td>
<td>Thai ISO</td>
</tr>
<tr>
<td>utf-8</td>
<td>utf8</td>
<td>Unicode (UTF-8)</td>
</tr>
<tr>
<td>u161</td>
<td>utf16</td>
<td>Unicode (UTF-16)</td>
</tr>
<tr>
<td>u321</td>
<td>utf32</td>
<td>Unicode (UTF-32)</td>
</tr>
<tr>
<td>warabic</td>
<td>wara</td>
<td>Arabic Windows</td>
</tr>
<tr>
<td>wbaltic</td>
<td>wbal</td>
<td>Baltic Windows</td>
</tr>
<tr>
<td>wcyrillic</td>
<td>wcyr</td>
<td>Cyrillic Windows</td>
</tr>
<tr>
<td>wgreek</td>
<td>wgrk</td>
<td>Greek Windows</td>
</tr>
<tr>
<td>whebrew</td>
<td>wheb</td>
<td>Hebrew Windows</td>
</tr>
<tr>
<td>wlatin1</td>
<td>wlt1</td>
<td>Western Windows</td>
</tr>
<tr>
<td>wlatin2</td>
<td>wlt2</td>
<td>Central European Windows</td>
</tr>
<tr>
<td>wturkish</td>
<td>wtur</td>
<td>Turkish Windows</td>
</tr>
<tr>
<td>wvietnamese</td>
<td>wvie</td>
<td>Vietnamese Windows</td>
</tr>
</tbody>
</table>
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 18.1 Single-Byte Encodings for OpenVMS

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic</td>
<td>Arabic (ISO)</td>
</tr>
<tr>
<td>cyrillic</td>
<td>Cyrillic (ISO)</td>
</tr>
<tr>
<td>greek</td>
<td>Greek (ISO)</td>
</tr>
<tr>
<td>hebrew</td>
<td>Hebrew (ISO)</td>
</tr>
<tr>
<td>latin1</td>
<td>Western (ISO)</td>
</tr>
<tr>
<td>latin2</td>
<td>Central Europe (ISO)</td>
</tr>
<tr>
<td>latin5</td>
<td>Turkish (ISO)</td>
</tr>
<tr>
<td>latin6</td>
<td>Baltic (ISO)</td>
</tr>
<tr>
<td>latin9</td>
<td>European (ISO)</td>
</tr>
<tr>
<td>thai</td>
<td>Thai (ISO)</td>
</tr>
</tbody>
</table>

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 18.2  Single-Byte Encodings for UNIX

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>arabic</td>
<td>Arabic (ISO 8859-6)</td>
</tr>
<tr>
<td>cyrillic</td>
<td>Cyrillic (ISO 8859-5)</td>
</tr>
<tr>
<td>greek</td>
<td>Greek (ISO 8859-7)</td>
</tr>
<tr>
<td>hebrew</td>
<td>Hebrew (ISO 8859-8)</td>
</tr>
<tr>
<td>latin1</td>
<td>Western (ISO 8859-1)</td>
</tr>
<tr>
<td>latin2</td>
<td>Central Europe (ISO 8859-2)</td>
</tr>
<tr>
<td>latin5</td>
<td>Turkish (ISO 8859-9)</td>
</tr>
<tr>
<td>latin6</td>
<td>Baltic (ISO 8859-4)</td>
</tr>
<tr>
<td>latin8</td>
<td>Celtic (ISO 8859-14)</td>
</tr>
<tr>
<td>latin9</td>
<td>European (ISO 8859-15)</td>
</tr>
<tr>
<td>thai</td>
<td>Thai (ISO 8859-11)</td>
</tr>
</tbody>
</table>

Table 18.3  Double-Byte Encodings for UNIX

<table>
<thead>
<tr>
<th>ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>big5</td>
<td>Traditional Chinese (Big5)</td>
</tr>
<tr>
<td>euc-cn</td>
<td>Simplified Chinese (EUC)</td>
</tr>
<tr>
<td>euc-jp</td>
<td>Japanese (EUC)</td>
</tr>
<tr>
<td>euc-kr</td>
<td>Korean (EUC)</td>
</tr>
<tr>
<td>euc-tw</td>
<td>Traditional Chinese (EUC)</td>
</tr>
<tr>
<td>hp15-tw</td>
<td>Traditional Chinese (HP15)</td>
</tr>
<tr>
<td>ms-936</td>
<td>Simplified Chinese (PCMS)</td>
</tr>
<tr>
<td>ms-949</td>
<td>Korean (PCMS)</td>
</tr>
<tr>
<td>shift-jis</td>
<td>Japanese (SJIS)</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Description</th>
<th>Windows ENCODING= Value</th>
<th>MS-DOS ENCODING= Value</th>
<th>IBM-PC ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arabic</td>
<td>warabic</td>
<td>msdos720</td>
<td>pcoem864</td>
</tr>
<tr>
<td>Baltic</td>
<td>whaltic</td>
<td>msdos775</td>
<td>pcoem921</td>
</tr>
<tr>
<td>Central Europe</td>
<td>wlatin2</td>
<td>not applicable</td>
<td>pcoem852</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cyrillic</td>
<td>wcyrillic</td>
<td>not applicable</td>
<td>pcoem866</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>pcoem855</td>
</tr>
<tr>
<td>Central Europe</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem852</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estonia</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem922</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>European</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem858</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Farsi</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pc1098</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>French Canadian</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem863</td>
</tr>
<tr>
<td>Greek</td>
<td>wgreek</td>
<td>msdos737</td>
<td>not applicable</td>
</tr>
<tr>
<td>Hebrew</td>
<td>whebrew</td>
<td>not applicable</td>
<td>pcoem862</td>
</tr>
<tr>
<td>Indian Script Code</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pciscii806</td>
</tr>
<tr>
<td>Nordic</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem865</td>
</tr>
<tr>
<td>Portuguese</td>
<td>not applicable</td>
<td>pcoe860</td>
<td>not applicable</td>
</tr>
<tr>
<td>Thai</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem874</td>
</tr>
<tr>
<td>Turkish</td>
<td>wturkish</td>
<td>not applicable</td>
<td>pcoem857</td>
</tr>
<tr>
<td>USA</td>
<td>not applicable</td>
<td>not applicable</td>
<td>pcoem437</td>
</tr>
</tbody>
</table>
### Table 18.5  Windows Double-Byte Encodings

<table>
<thead>
<tr>
<th>Description</th>
<th>PCMS ENCODING= Value</th>
<th>No Vendor ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vietnamese</td>
<td>wvietnamese</td>
<td>not applicable</td>
</tr>
<tr>
<td>Western</td>
<td>wlatin1</td>
<td>pcoem850</td>
</tr>
</tbody>
</table>

**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 18.6  Single-Byte Encodings for z/OS

<table>
<thead>
<tr>
<th>Encoding ENCODING= Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EBCDIC037</td>
<td>EBCDIC cp037- Old North America</td>
</tr>
<tr>
<td>EBCDIC275</td>
<td>EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>EBCDIC425</td>
<td>EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>EBCDIC838</td>
<td>EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>EBCDIC870</td>
<td>EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>EBCDIC875</td>
<td>EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>EBCDIC905</td>
<td>EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>EBCDIC924</td>
<td>EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>EBCDIC1025</td>
<td>EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>EBCDIC1026</td>
<td>EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>EBCDIC1047</td>
<td>EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>EBCDIC1097</td>
<td>EBCDIC cp1097-Farsi Bilingual</td>
</tr>
<tr>
<td>EBCDIC1112</td>
<td>EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>EBCDIC1122</td>
<td>EBCDIC cp1122-Estonian</td>
</tr>
<tr>
<td>EBCDIC1130</td>
<td>EBCDIC cp1130-Vietnamese</td>
</tr>
<tr>
<td>EBCDIC1137</td>
<td>EBCDIC cp1137-Devanagari</td>
</tr>
<tr>
<td>Encoding ENCODING= Value</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>EBCDIC1140</td>
<td>EBCDIC cp1140-North America</td>
</tr>
<tr>
<td>EBCDIC1141</td>
<td>EBCDIC cp1141-German/Austrian</td>
</tr>
<tr>
<td>EBCDIC1142</td>
<td>EBCDIC cp1142-Danish/Norwegian</td>
</tr>
<tr>
<td>EBCDIC1143</td>
<td>EBCDIC cp1143-Finnish/Swedish</td>
</tr>
<tr>
<td>EBCDIC1144</td>
<td>EBCDIC cp1144-Italian</td>
</tr>
<tr>
<td>EBCDIC1145</td>
<td>EBCDIC cp1145-Spanish</td>
</tr>
<tr>
<td>EBCDIC1146</td>
<td>EBCDIC cp1146-English (UK)</td>
</tr>
<tr>
<td>EBCDIC1147</td>
<td>EBCDIC cp1147-French</td>
</tr>
<tr>
<td>EBCDIC1148</td>
<td>EBCDIC cp1148-International</td>
</tr>
<tr>
<td>EBCDIC1149</td>
<td>EBCDIC cp1149-Iceland</td>
</tr>
<tr>
<td>EBCDIC1153</td>
<td>EBCDIC cp1153-Latin 2 Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1154</td>
<td>EBCDIC cp1154-Cyrillic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1155</td>
<td>EBCDIC cp1155-Turkey with euro</td>
</tr>
<tr>
<td>EBCDIC1156</td>
<td>EBCDIC cp1156-Baltic Multilingual with euro</td>
</tr>
<tr>
<td>EBCDIC1157</td>
<td>EBCDIC cp1157-Estonia with euro</td>
</tr>
<tr>
<td>EBCDIC1158</td>
<td>EBCDIC cp1158-Cyrillic Ukraine with euro</td>
</tr>
<tr>
<td>OPEN_ED-037</td>
<td>OpenEdition EBCDIC cp037-Old North America</td>
</tr>
<tr>
<td>OPEN_ED-275</td>
<td>OpenEdition EBCDIC cp275-Brazil</td>
</tr>
<tr>
<td>OPEN_ED-425</td>
<td>OpenEdition EBCDIC cp425-Arabic</td>
</tr>
<tr>
<td>OPEN_ED-838</td>
<td>OpenEdition EBCDIC cp838-Thai</td>
</tr>
<tr>
<td>OPEN_ED-870</td>
<td>OpenEdition EBCDIC cp870-Central Europe</td>
</tr>
<tr>
<td>OPEN_ED-875</td>
<td>OpenEdition EBCDIC cp875-Greek</td>
</tr>
<tr>
<td>OPEN_ED-905</td>
<td>OpenEdition EBCDIC cp905-Latin 3</td>
</tr>
<tr>
<td>OPEN_ED-924</td>
<td>OpenEdition EBCDIC cp924-Western Europe</td>
</tr>
<tr>
<td>OPEN_ED-1025</td>
<td>OpenEdition EBCDIC cp1025-Cyrillic</td>
</tr>
<tr>
<td>OPEN_ED-1026</td>
<td>OpenEdition EBCDIC cp1026-Turkish</td>
</tr>
<tr>
<td>OPEN_ED-1047</td>
<td>OpenEdition EBCDIC cp1047-Latin1</td>
</tr>
<tr>
<td>OPEN_ED_1097</td>
<td>OpenEdition EBCDIC cp1097-Farsi Bilingual</td>
</tr>
<tr>
<td>OPEN_ED-1112</td>
<td>OpenEdition EBCDIC cp1112-Baltic</td>
</tr>
<tr>
<td>OPEN_ED-1122</td>
<td>OpenEdition EBCDIC cp1122-Estonian</td>
</tr>
<tr>
<td>OPEN_ED-1130</td>
<td>OpenEdition EBCDIC cp1130-Vietnamese</td>
</tr>
<tr>
<td>OPEN_ED-1137</td>
<td>OpenEdition EBCDIC cp1137-Devanagari</td>
</tr>
<tr>
<td>OPEN_ED-1140</td>
<td>OpenEdition EBCDIC cp1140-North America</td>
</tr>
<tr>
<td>OPEN_ED-1141</td>
<td>OpenEdition EBCDIC cp1141-German/Austrian</td>
</tr>
<tr>
<td>OPEN_ED-1142</td>
<td>OpenEdition EBCDIC cp1142-Danish/Norwegian</td>
</tr>
<tr>
<td>OPEN_ED-1143</td>
<td>OpenEdition EBCDIC cp1143-Finnish/Swedish</td>
</tr>
</tbody>
</table>
### Table 18.7 Double-Byte Encodings for z/OS

<table>
<thead>
<tr>
<th>Description</th>
<th>ENCODING= Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese</td>
<td>IBM-939</td>
</tr>
<tr>
<td>Korean</td>
<td>IBM-933</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>IBM-935</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>IBM-937</td>
</tr>
</tbody>
</table>
Appendixes

Appendix 1. Additional NLS Language Elements 569

Appendix 2. Recommended Reading 655
APPENDIX

1

Additional NLS Language Elements

Additional NLS Language Elements 570
EURDFDDw. Format 570
EURDFDEw. Format 572
EURDFDNw. Format 575
EURDFDTw.d Format 576
EURDFDWNw. Format 578
EURDFMNWw. Format 581
EURDFMYw. Format 582
EURDFWDXw. Format 584
EURDFWKXw. Format 586
EURFRATSw.d Format 589
EURFRBWFw.d Format 590
EURFRCHFWw.d Format 591
EURFRCZKw.d Format 592
EURFRDEMWw.d Format 593
EURFRDKKw.d Format 595
EURFRESPw.d Format 596
EURFRFMw.d Format 597
EURFRFw.d Format 598
EURFRGBPw.d Format 599
EURFRGDKw.d Format 600
EURFRHUFw.d Format 601
EURFRIEPw.d Format 602
EURFRITLw.d Format 603
EURFRLUFw.d Format 605
EURFRNLGw.d Format 606
EURFRNOKw.d Format 607
EURFRPLZw.d Format 608
EURFRPTEw.d Format 609
EURFRROLw.d Format 610
EURFRRURw.d Format 611
EURFRSEKw.d Format 612
EURFRSITw.d Format 613
EURFRTRLw.d Format 615
EURFRYUDw.d Format 616
EURTOATSw.d Format 617
EURTOBEFw.d Format 618
EURTOCHFWw.d Format 619
EURTOCZKw.d Format 620
EURTOEMNWw.d Format 621
EURTODKKw.d Format 622
Additional NLS Language Elements

The following EUR language elements have been replaced with NL language elements. The EUR elements are supported in SAS 9.2, but SAS recommends that you use the NL elements.

**EURDFDDw. Format**

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

Category:  Date and Time

Alignment: right

Syntax

EURDFDDw.
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 460 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.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfdd8.;</td>
<td>02.01.02</td>
</tr>
<tr>
<td>put date espdfdd8.;</td>
<td>02.01.02</td>
</tr>
<tr>
<td>put date fradfdd8.;</td>
<td>02/01/02</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

- DATEw. in *SAS Language Reference: Dictionary*
- DDMYYw. 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*
- DDMYYw. 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 460

---

**EURDFDEw. Format**

Writes international date values in the form *ddmmmyy* or *ddmmmyyyyy*.

**Category:** Date and Time

**Alignment:** right

**Syntax**

EURDFDEw.
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 \( ddmmyy \) or \( ddmmyyyy \):

\( dd \)

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 460 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 do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are 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 that the DFLANG= system option is set to Spanish.

```plaintext
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date espdfde9.;</td>
<td>02ene2002</td>
</tr>
<tr>
<td>put date fradfde9.;</td>
<td>02jan2002</td>
</tr>
</tbody>
</table>

See Also

Formats:

DATEw. in SAS Language Reference: Dictionary

Functions:

DATE in SAS Language Reference: Dictionary

Informats:

“EURDFDEw. Informat” on page 645

System Options:

“DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460
EURFDNW. Format

Wrote international date values as the day of the week.

Category: Date and Time
Alignment: right

Syntax

EURFDNW.

Syntax Description

\( w \)

specifies the width of the output field.

Default: 1
Range: 1–32

Details

The EURFDNW. format writes SAS date values in the form \textit{day-of-the-week}:

\begin{itemize}
\item \textit{day-of-the-week} is represented as 1=Monday, 2=Tuesday, and so forth.
\end{itemize}

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 460 for the list of language prefixes. When you specify the language prefix in the format, SAS ignores the DFLANG= option.

\textbf{Note:} The EUR-date formats require European character sets and encodings. Some formats work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are 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.

\begin{verbatim}
options dflang=spanish;
\end{verbatim}

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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day eurdfdn.;</td>
<td>3</td>
</tr>
<tr>
<td>put day espfdn.;</td>
<td>3</td>
</tr>
<tr>
<td>put day itadfdn.;</td>
<td>3</td>
</tr>
</tbody>
</table>

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 460

**EURDFDTw.d Format**

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

**Category:** Date and Time

**Alignment:** right

**Syntax**

EURDFDTw.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.
\( d \)
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 \( \text{EURDFDT}_w.d \) format writes SAS datetime values in the form \( \text{ddmmmyy}:hh:mm:ss.ss \):

- \( dd \)
  - 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 460 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 p.m. The first PUT statement assumes that the DFLANG= system option is set to German.

```sas
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.
<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfdt20.;</td>
<td>12Sep2002:12:39:43</td>
</tr>
<tr>
<td>put date deudfdt20.;</td>
<td>12Sep2002:12:39:43</td>
</tr>
<tr>
<td>put date itadfdt20.;</td>
<td>12Set2002:12:39:43</td>
</tr>
</tbody>
</table>

**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*

Informat:
- DATEw. in *SAS Language Reference: Dictionary*
- DATETIMEw.d in *SAS Language Reference: Dictionary*
- “EURDFDTw. Informat” on page 646
- TIMEw.d in *SAS Language Reference: Dictionary*

System Options:
- “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460

---

**EURDFDWNw. Format**

writes international date values as the name of the day.

**Category:** Date and Time  
**Alignment:** right

**Syntax**

**EURDFDWNw.**

**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:

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

**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 EURFDOWNw. format writes SAS date values in the form \( \text{day-name} \):

\( \text{day-name} \)

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 460 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 do not work correctly using non-European encodings. When running in a DBCS
environment, the default format width and max width are 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 that the DFLANG= system option is set to French.

```sas
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day eurdfdwn8.;</td>
<td>Vendredi</td>
</tr>
<tr>
<td>put day fradfdwn8.;</td>
<td>Vendredi</td>
</tr>
<tr>
<td>put day espdfdwn8.;</td>
<td>viernes</td>
</tr>
</tbody>
</table>

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*
- “EURDFDTw. Informat” on page 646
- TIMEw.d in *SAS Language Reference: Dictionary*

System Options:

- “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460
EURDFMNw. Format

Writes international date values as the name of the month.

Category: Date and Time
Alignment: right

Syntax
EURDFMNw.

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 EURDFMNw. format writes SAS date values in the form \textit{month-name}:

\textit{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 460 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 do 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 that the DFLANG= system option is set to Italian.

\begin{verbatim}
options dflang=ita;
\end{verbatim}
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfm10.;</td>
<td>janvier</td>
</tr>
<tr>
<td>put date itadfm10.;</td>
<td>Gennaio</td>
</tr>
<tr>
<td>put date deudfm10.;</td>
<td>Januar</td>
</tr>
</tbody>
</table>

See Also

Formats:
- MONNAMEw. in SAS Language Reference: Dictionary

Functions:
- DATE in SAS Language Reference: Dictionary

Informats:
- “EURDFDEw. Informat” on page 645

System Options:
- “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460

---

EURDFMYw. Format

Writes international date values in the form mmmmm or mmmmmmm.

Category: Date and Time
Alignment: right

Syntax

EURDFMYw.

Syntax Description

w
specifies the width of the output field.
Default: 5 (except 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 EURDFMYw. format writes SAS date values in the form `mmmmyy`, where

`mmmm`  
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 460 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 do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are 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.

```sas
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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfmy7.;</td>
<td>ene2002</td>
</tr>
<tr>
<td>put date espdfmy7.;</td>
<td>ene2002</td>
</tr>
<tr>
<td>put date fradfmy7.;</td>
<td>jan2002</td>
</tr>
</tbody>
</table>

See Also

Formats:

- DDMYYw. in *SAS Language Reference: Dictionary*
- MMDDYYw. in *SAS Language Reference: Dictionary*
- MONYYw. in *SAS Language Reference: Dictionary*
- YYMMDDw. in *SAS Language Reference: Dictionary*
EURDFWDXw. 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

EURDFWDXw.

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:

<table>
<thead>
<tr>
<th>Language</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Afrikaans (AFR)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Catalan (CAT)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Croatian (CRO)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Czech (CSY)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Danish (DAN)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Dutch (NLD)</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Finnish (FIN)</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>French (FRA)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>German (DEU)</td>
<td>18</td>
<td>18</td>
</tr>
<tr>
<td>Hungarian (HUN)</td>
<td>40</td>
<td>18</td>
</tr>
<tr>
<td>Italian (ITA)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Macedonian (MAC)</td>
<td>40</td>
<td>17</td>
</tr>
</tbody>
</table>
### Additional NLS Language Elements

#### EURDFWDXw. Format

<table>
<thead>
<tr>
<th>Language</th>
<th>Maximum</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Norwegian (NOR)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Polish (POL)</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>Portuguese (PTG)</td>
<td>37</td>
<td>23</td>
</tr>
<tr>
<td>Russian (RUS)</td>
<td>40</td>
<td>16</td>
</tr>
<tr>
<td>Slovenian (SLO)</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Spanish (ESP)</td>
<td>24</td>
<td>24</td>
</tr>
<tr>
<td>Swedish (SVE)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-French (FRS)</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>Swiss-German (DES)</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

**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 EURDFWDXw. 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 460 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 EURDFWDXw.K format is the same as the EURDFWDXw. format except that EURDFWDX $w$. 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 that 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put day eurdfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day nlddfwdx29.;</td>
<td>2 januari 2002</td>
</tr>
<tr>
<td>put day itadfwdx17.;</td>
<td>02 Gennaio 1998</td>
</tr>
</tbody>
</table>

See Also

Formats:

    WORDDATXw. in SAS Language Reference: Dictionary

System Options:

    “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460

EURDFWKXw. 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

EURDFWKXw.
**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:

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

**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 EURDFWKXw. format writes SAS date values in the form \textit{day-of-week, dd month-name yy (or yyyy)}:

- \textit{day-of-week} is the name of day.
- \textit{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 460 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 do not work correctly using non-European encodings. When running in a DBCS environment, the default format width and max width are 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 EURDFWKXw. format is the same as the EURDFWDXw. format except that EURDFWKXw. 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 that 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.

<table>
<thead>
<tr>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>put date eurdfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date deudfwkx30.;</td>
<td>Freitag, 4. Januar 2002</td>
</tr>
<tr>
<td>put date itadfwkx17.;</td>
<td>Ven, 04 Gen 2002</td>
</tr>
</tbody>
</table>

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
YYMMD Dw. 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
YYMMD Dw. in SAS Language Reference: Dictionary

System Options:
“DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460

---

**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
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 62.
**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrats5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3,63</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrats5.;</td>
<td>E380</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E380,41</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrats5.;</td>
<td>3.804</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrats9.2;</td>
<td>E3.804,06</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOATS\textit{w.d Format}” on page 617

Functions:

“EUROCURR Function” on page 650

---

**EURFRBEF\textit{w.d Format}**

Converts an amount from Belgian francs to euros.

**Category:** Currency Conversion  
**Alignment:** right

**Syntax**

EURFRBEF\textit{w.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.  

**Default:** 6

\textit{d}

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

**Details**

The EURFRBEF\textit{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 EURFRBEFw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrbef5.; E1</td>
<td></td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrbef5.; E130</td>
<td></td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrbef5.; 1.298</td>
<td></td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURTOBEFw.d Format” on page 618
Functions:
“EUROCURR Function” on page 650

EURFRCHFw.d Format

Converts an amount from Swiss francs to euros.

Category: Currency Conversion
Alignment: right

Syntax

EURFRCHFw.d

Syntax Description

w
specifies the width of the output field.
Default: 6
\( d \)

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 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrchf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E31,17</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrchf5.;</td>
<td>E770</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E769,53</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrchf5.;</td>
<td>7.695</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrchf9.2;</td>
<td>E7.694,94</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOCHF\(w.d\) Format” on page 619

Functions:

“EUROCURR Function” on page 650

**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

\( d \)

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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrczk5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrczk9.2;</td>
<td>E1,43</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrczk5.;</td>
<td>E150</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrczk9.2;</td>
<td>E150,18</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrczk5.;</td>
<td>1.502</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrczk9.2;</td>
<td>E1.501,74</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTOCZK\( w.d \) Format” on page 620

Functions:

“EUROCURR Function” on page 650

EURFRDEM\( w.d \) Format

Converts an amount from Deutsche marks to euros.

Category: Currency Conversion

Alignment: right
Syntax

\textbf{EURFRDEM}_w.d

Syntax Description

\textit{w}

specifies the width of the output field.

\textbf{Default:} 6

\textit{d}

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

Details

The EURFRDEM\textit{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\textit{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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrdem5.; put amount eurfrdem9.2;</td>
<td>E26 E25.56</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrdem5.; put amount eurfrdem9.2;</td>
<td>E631 E631.22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrdem5.; put amount eurfrdem9.2;</td>
<td>6.312 E6.311,90</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTODEM\textit{w.d} Format” on page 621

Functions:

“EUROCURR Function” on page 650
EURFRDKK\textit{w.d} Format

Converts an amount from Danish kroner to euros.

Category: Currency Conversion
Alignment: right

Syntax

\texttt{EURFRDKK\textit{w.d}}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 6

\textit{d}

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

Details

The EURFRDKK\textit{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\textit{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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>\texttt{put amount eurfrdkk5.;} \texttt{put amount eurfrdkk9.2;}</td>
<td>E7 \texttt{E6,68}</td>
</tr>
<tr>
<td>1234.56</td>
<td>\texttt{put amount eurfrdkk5.;} \texttt{put amount eurfrdkk9.2;}</td>
<td>E165 \texttt{E164,83}</td>
</tr>
<tr>
<td>12345</td>
<td>\texttt{put amount eurfrdkk5.;} \texttt{put amount eurfrdkk9.2;}</td>
<td>1.648 \texttt{E1.648,18}</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURTODKK\textit{w.d Format}” on page 622
Functions:
“EUROCURR Function” on page 650

\textbf{EURFRESP\textit{w.d Format}}

Converts an amount from Spanish pesetas to euros.

\begin{itemize}
  \item \textbf{Category:} Currency Conversion
  \item \textbf{Alignment:} right
\end{itemize}

\textbf{Syntax}

\texttt{EURFRESP\textit{w.d}}

\textbf{Syntax Description}

\texttt{w}

specifies the width of the output field.

\textbf{Default:} \texttt{6}

\texttt{d}

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

\textbf{Details}

The \texttt{EURFRESP\textit{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 \texttt{EURFRESP\textit{w.d}} format and the \texttt{EUROCURR} function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

\textbf{Examples}

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

\begin{center}
\begin{tabular}{lll}
\hline
\textbf{Amounts} & \textbf{Statements} & \textbf{Results} \\
\hline
200 & \texttt{put amount eurfresp5.;} & E1 \\
 & \texttt{put amount eurfresp9.2;} & E1,20 \\
20234.56 & \texttt{put amount eurfresp5.;} & E122 \\
 & \texttt{put amount eurfresp9.2;} & E121,61 \\
202345 & \texttt{put amount eurfresp5.;} & 1.216 \\
 & \texttt{put amount eurfresp9.2;} & E1.216,12 \\
\hline
\end{tabular}
\end{center}
Additional NLS Language Elements

EURFRFIM \textit{w.d} Format

Converts an amount from Finnish markkaa to euros.

Category: Currency Conversion
Alignment: right

Syntax

\texttt{EURFRFIM \textit{w.d}}

Syntax Description

\texttt{w}

specifies the width of the output field.
Default: 6

\texttt{d}

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

Details

The EURFRFIM \textit{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 \textit{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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrfim5.;</td>
<td>E8</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrfim9.2;</td>
<td>E8,41</td>
</tr>
</tbody>
</table>
### EURFRFRF \(w.d\) Format

Converts an amount from French francs to euros.

**Category:** Currency Conversion  
**Alignment:** right

#### Syntax

\[ \text{EURFRFRF} \ \!w.d \]

#### Syntax Description

\(w\)

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

\(d\)

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 62.

### See Also

**Formats:**

“EURTOFIM \(w.d\) Format” on page 625

**Functions:**

“EUROCURR Function” on page 650
Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrf5.;</td>
<td>E8</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrf9.2;</td>
<td>E7,62</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrf5.;</td>
<td>E188</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrf9.2;</td>
<td>E188,21</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrf5.;</td>
<td>1.882</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrf9.2;</td>
<td>E1.881,98</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURTOFRFw.d Format” on page 626
Functions:
“EUROCURR Function” on page 650

EURFRGBPw.d Format

Converts an amount from British pounds to euros.

Category: Currency Conversion
Alignment: right

Syntax

EURFRGBPw.d

Syntax Description

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

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

Details

The EURFRGBPw.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 EURFRGBPw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrgbp5.;</td>
<td>E71</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgbp9.2;</td>
<td>E71.42</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrgbp5.;</td>
<td>1,763</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgbp9.2;</td>
<td>E1,763.32</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrgbp5.;</td>
<td>17632</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgbp9.2;</td>
<td>17,632.39</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“EURTOGBPw.d Format” on page 627

Functions:
“EUROCURR Function” on page 650

---

**EURFRGRDw.d Format**

Converts an amount from Greek drachmas to euros.

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

EURFRGRDw.d

**Syntax Description**

- \( w \)
  - specifies the width of the output field.
  - **Default:** 6

- \( d \)
specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRGRD\textit{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\textit{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 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>put amount eurfrgrd5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgrd9.2;</td>
<td>E1,17</td>
</tr>
<tr>
<td>40234.56</td>
<td>put amount eurfrgrd5.;</td>
<td>E118</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgrd9.2;</td>
<td>E118,03</td>
</tr>
<tr>
<td>402345</td>
<td>put amount eurfrgrd5.;</td>
<td>1.180</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrgrd9.2;</td>
<td>E1.180,30</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOGRD\textit{w.d} Format” on page 628

Functions:

“EUROCURR Function” on page 650

---

**EURFRHUF\textit{w.d} Format**

Converts an amount from Hungarian forints to euros.

**Category:** Currency Conversion

**Alignment:** right

---

**Syntax**

\texttt{EURFRHUF\textit{w.d}}

**Syntax Description**
EURFRREP \( w \)
specifies the width of the output field.

**Default:** 6

\( d \)
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 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrhuf5.;</td>
<td>€1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>€1,15</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrhuf5.;</td>
<td>€116</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>€116,14</td>
</tr>
<tr>
<td>302345</td>
<td>put amount eurfrhuf5.;</td>
<td>1.161</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrhuf9.2;</td>
<td>€1.161,41</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOHUF \( w.d \) Format” on page 629

Functions:

“EUROCURR Function” on page 650

---

**EURFRREP \( w.d \) Format**

Converts an amount from Irish pounds to euros.

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

```
EURFRREP \( w.d \)
```
Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)

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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurfriep5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>E1.27</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfriep5.;</td>
<td>1,568</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>E1,567.57</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfriep5.;</td>
<td>15675</td>
</tr>
<tr>
<td></td>
<td>put amount eurfriep9.2;</td>
<td>15,674.92</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTOIEP\( w.d \) Format” on page 630

Functions:

“EUROCURR Function” on page 650

EURFRIEw.d Format

Converts an amount from Italian lire to euros.

Category: Currency Conversion

Alignment: right
Syntax
EURFRITLw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)

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

Details

The EURFRITLw.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 EURFRITLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>put amount eurfrtit15.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtit19.2;</td>
<td>E1,03</td>
</tr>
<tr>
<td>7234.56</td>
<td>put amount eurfrtit15.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtit19.2;</td>
<td>E3,74</td>
</tr>
<tr>
<td>72345</td>
<td>put amount eurfrtit15.;</td>
<td>E37</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtit19.2;</td>
<td>E37,36</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTOITLw.d Format” on page 631

Functions:

“EUROCURR Function” on page 650
EURFRLUF\textsubscript{w.d} Format

Converts an amount from Luxembourg francs to euros.

Category: Currency Conversion
Alignment: right

Syntax
EURFRLUF\textsubscript{w.d}

Syntax Description

\textit{w} specifies the width of the output field.

Default: 6

\textit{d} specifies the number of digits to the right of the decimal point in the numeric value.

Details

The EURFRLUF\textsubscript{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\textsubscript{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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrluf5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E1,24</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrluf5.;</td>
<td>E31</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E30,60</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrluf5.;</td>
<td>E306</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrluf9.2;</td>
<td>E306,02</td>
</tr>
</tbody>
</table>

See Also

Formats:
**EURFRNLG\(w.d\) Format**

Converts an amount from Dutch guilders to euros.

**Category:** Currency Conversion  
**Alignment:** right

**Syntax**

\[
\text{EURFRNLG}\(w.d\)
\]

**Syntax Description**

**\(w\)**

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

**\(d\)**

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 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrnlg5.;</td>
<td>€23</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>€22.69</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrnlg5.;</td>
<td>€560</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>€560.22</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrnlg5.;</td>
<td>5.602</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrnlg9.2;</td>
<td>€5.601,92</td>
</tr>
</tbody>
</table>
EURFRNOK\textit{w.d} Format

Converts an amount from Norwegian krone to euros.

Category: Currency Conversion

Alignment: right

Syntax

\texttt{EURFRNOK\textit{w.d}}

Syntax Description

\textit{w}

specifies the width of the output field.

\textbf{Default:} 6

\textit{d}

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

Details

The EURFRNOK\textit{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\textit{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 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>\texttt{put amount eurfrnok5.;}</td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td>\texttt{put amount eurfrnok9.2;}</td>
<td>E5,44</td>
</tr>
</tbody>
</table>
See Also

Formats:

“EURTONOKw.d Format” on page 635

Functions:

“EUROCURR Function” on page 650

EURFRPLZw.d Format

Converts an amount from Polish zlotys to euros.

Category: Currency Conversion

Alignment: right

Syntax

EURFRPLZw.d

Syntax Description

\( \text{w} \)

specifies the width of the output field.

Default: 6

\( \text{d} \)

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

Details

The EURFRPLZw.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 EURFRPLZw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.
Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrplz5.;</td>
<td>E12</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E11,90</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrplz5.;</td>
<td>E294</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E293,94</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrplz5.;</td>
<td>2.939</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrplz9.2;</td>
<td>E2.939,29</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTOPLZw.d Format” on page 636

Functions:

“EUROCURR Function” on page 650

EURFRPTEw.d Format

Converts an amount from Portuguese escudos to euros.

Category: Currency Conversion

Alignment: right

Syntax

EURFRPTEw.d

Syntax Description

w

specifies the width of the output field.

Default: 6

d

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

Details

The EURFRPTEw.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 EURFRPTEw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>put amount eurfrpte5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1,50</td>
</tr>
<tr>
<td>30234.56</td>
<td>put amount eurfrpte5.;</td>
<td>E151</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E150,81</td>
</tr>
<tr>
<td>302345</td>
<td>put amount eurfrpte5.;</td>
<td>1.508</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrpte9.2;</td>
<td>E1.508,09</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOPTETw.d Format” on page 637

Functions:

“EUROCURR Function” on page 650

---

**EURFRROLw.d Format**

Converts an amount from Romanian lei to euros.

**Category:**   Currency Conversion  
**Alignment:**  right  

---

**Syntax**

**EURFRROLw.d**

**Syntax Description**

- \( w \)
  
  specifies the width of the output field.

  **Default:** 6  

- \( d \)
specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURFRROLw.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 EURFRROLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrrol5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrol9.2;</td>
<td>E3,65</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrrol5.;</td>
<td>E382</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrol9.2;</td>
<td>E381,81</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrrol5.;</td>
<td>3.818</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrol9.2;</td>
<td>E3.818,02</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
“EURTOROLw.d Format” on page 638

Functions:
“EUROCURR Function” on page 650

**EURFRRRURw.d Format**

Converts an amount from Russian rubles to euros.

**Syntax**

EURFRRRURw.d

**Syntax Description**
specifies the width of the output field.

**Default:** 6

d

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 62.

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrrur5.;</td>
<td>E3</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2,53</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfrrur5.;</td>
<td>E265</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E264,80</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfrrur5.;</td>
<td>2.648</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrrur9.2;</td>
<td>E2.647,97</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTORUR\(w.d\) Format” on page 639

Functions:

“EUROCURR Function” on page 650

**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

\[ d \]

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 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfrsek5.;</td>
<td>E5</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E5.34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurfrsek5.;</td>
<td>E132</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E131.81</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurfrsek5.;</td>
<td>1.318</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsek9.2;</td>
<td>E1.318,08</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURTOSEK\(w,d\) Format” on page 640

Functions:

“EUROCURR Function” on page 650

---

**EURFRSIT\(w,d\) Format**

Converts an amount from Slovenian tolar to euros.

**Category:** Currency Conversion

**Alignment:** right
Syntax

EURFRSIT\textit{w.d}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 6

\textit{d}

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

Details

The EURFRSIT\textit{w.d} format converts an amount from Slovenian tolaris to an amount in euros and produces a formatted euro value. The conversion rate is a changeable rate that is incorporated into the EURFRSIT\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Note: Slovenia’s currency is the Euro. The information for EURFRSIT is provided for user’s historical data. △

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>put amount eurfrsit5.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E1,05</td>
</tr>
<tr>
<td>20234.56</td>
<td>put amount eurfrsit5.;</td>
<td>E106</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E105,94</td>
</tr>
<tr>
<td>202345</td>
<td>put amount eurfrsit5.;</td>
<td>1.059</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrsit9.2;</td>
<td>E1.059,40</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURTOSIT\textit{w.d} Format” on page 641

Functions:

“EUROCURR Function” on page 650
**EURFRTRL\(w.d\) Format**

Converting an amount from Turkish liras to euros.

Category: Currency Conversion
Alignment: right

---

**Syntax**

\(\text{EURFRTRL}w.d\)

**Syntax Description**

\(w\) specifies the width of the output field.

Default: 6

\(d\) specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The \(\text{EURFRTRL}w.d\) 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 \(\text{EURFRTRL}w.d\) format and the \(\text{EUROCURR}\) function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>put amount eurfrtr15.;</td>
<td>E1</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtr19.2;</td>
<td>E1,19</td>
</tr>
<tr>
<td>40234.56</td>
<td>put amount eurfrtr15.;</td>
<td>E119</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtr19.2;</td>
<td>E119,42</td>
</tr>
<tr>
<td>402345</td>
<td>put amount eurfrtr15.;</td>
<td>1.194</td>
</tr>
<tr>
<td></td>
<td>put amount eurfrtr19.2;</td>
<td>E1.194,21</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
EURFRYUD\textit{w}.\textit{d} Format

Converts an amount from Yugoslavian dinars to euros.

Category: Currency Conversion

Alignment: right

Syntax

\texttt{EURFRYUD\textit{w}.\textit{d}}

Syntax Description

\textit{w}

specifies the width of the output field.

Default: 6

\textit{d}

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

Details

The EURFRYUD\textit{w}.\textit{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\textit{w}.\textit{d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>put amount eurfryud5.;</td>
<td>E4</td>
</tr>
<tr>
<td></td>
<td>put amount eurfryud9.2;</td>
<td>E3,83</td>
</tr>
<tr>
<td>5234.56</td>
<td>put amount eurfryud5.;</td>
<td>E401</td>
</tr>
<tr>
<td></td>
<td>put amount eurfryud9.2;</td>
<td>E400,67</td>
</tr>
<tr>
<td>52345</td>
<td>put amount eurfryud5.;</td>
<td>4.007</td>
</tr>
<tr>
<td></td>
<td>put amount eurfryud9.2;</td>
<td>E4.006,69</td>
</tr>
</tbody>
</table>
See Also

Formats:
  “EURTOYUDw.d Format” on page 643
Functions:
  “EUROCURR Function” on page 650

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

\( d \)
  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 “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoats6.;</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoats12.2;</td>
<td>13.76</td>
</tr>
</tbody>
</table>
Amounts | Statements | Results
---|---|---
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\textit{w.d} Format” on page 589

Functions:
“EUROCURR Function” on page 650

\textbf{EURTOBEF\textit{w.d} Format}

Converts an amount from euros to Belgian francs.

\textbf{Syntax}

\texttt{EURTOBEF\textit{w.d}}

\textbf{Syntax Description}

\textit{w}

specifies the width of the output field.

\textbf{Default:} 6

\textit{d}

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

\textbf{Details}

The EURTOBEF\textit{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\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.
Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtobef6.;</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>put amount eurtobef12.2;</td>
<td>40.34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtobef6.;</td>
<td>49802</td>
</tr>
<tr>
<td></td>
<td>put amount eurtobef12.2;</td>
<td>49802.03</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtobef6.;</td>
<td>497996</td>
</tr>
<tr>
<td></td>
<td>put amount eurtobef12.2;</td>
<td>497996.07</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURFRBEFw.d Format” on page 590

Functions:
“EUROCURR Function” on page 650

EURTOCHFw.d Format

Converts an amount from euros to Swiss francs.

Category: Currency Conversion

Alignment: right

Syntax
EURTOCHFw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)

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

Details

The EURTOCHFw.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 EURTOCHFw.d
format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
</table>
| 1         | put amount eurtochf6.;  
put amount eurtochf12.2; | 2  
1.60 |
| 1234.56   | put amount eurtochf6.;  
put amount eurtochf12.2; | 1981  
1980.60 |
| 12345     | put amount eurtochf6.;  
put amount eurtochf12.2; | 19805  
19805.08 |

See Also

Formats:

“EURFRCHFw.d Format” on page 591

Functions:

“EUROCURRE Function” on page 650

EURTOCZKw.d Format

Converts an amount from euros to Czech koruny.

Category: Currency Conversion

Alignment: right

Syntax

EURTOCZKw.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)
specifies the number of digits to the right of the decimal point in the numeric value.

**Details**

The EURTOCZKw.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 EURTOCZKw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoczk6.;</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>34.86</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoczk6.;</td>
<td>43032</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>43032.19</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoczk6.;</td>
<td>430301</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoczk12.2;</td>
<td>430301.02</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
  “EURFRCZKw.d Format” on page 592

Functions:
  “EUROCURR Function” on page 650

**EURTODEMw.d Format**

**Converts an amount from euros to Deutsche marks.**

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

**EURTODEMw.d**

**Syntax Description**
**EURTODKK w.d Format**

**Converts an amount from euros to Danish kroner.**

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

EURTODKK \( w.d \)

---

**Details**

The EURTODKK \( w.d \) format converts an amount from euros to Danish kroner. It specifies the output field width and the number of digits to the right of the decimal point in the numeric value. The format is used in conjunction with the \( d \) format for numeric values in the output field.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodek6.;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodek12.2;</td>
<td>1.96</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtodek6.;</td>
<td>2415</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodek12.2;</td>
<td>2414.59</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodek6.;</td>
<td>24145</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodek12.2;</td>
<td>24144.72</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

- “EURFRDEM w.d Format” on page 593

Functions:

- “EUROCURR Function” on page 650
**Syntax Description**

\( w \)

specifies the width of the output field.

**Default:** 6

\( d \)

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 “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtodk6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>7.49</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtodk6.;</td>
<td>9247</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>9246.97</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtodk6.;</td>
<td>92465</td>
</tr>
<tr>
<td></td>
<td>put amount eurtodkk12.2;</td>
<td>92465.16</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURFRDKKK\( w.d \) Format” on page 595

Functions:

“EUROCURR Function” on page 650

---

**EURTOESP\( w.d \) Format**

Converts an amount from euros to Spanish pesetas.

**Category:** Currency Conversion

**Alignment:** right
Syntax

EURTOESP\textit{w.d}

Syntax Description

\textit{w}

 specifies the width of the output field.

 Default: 6

\textit{d}

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

Details

The EURTOESP\textit{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\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoesp8.;</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>166.39</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoesp8.;</td>
<td>205414</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>205413.50</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoesp8.;</td>
<td>2054035</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoesp12.2;</td>
<td>2054035.17</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRESP\textit{w.d} Format” on page 596

Functions:

“EUROCURR Function” on page 650
EURTOFIM\textit{w.d} Format

Converts an amount from euros to Finnish markkaa.

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

EURTOFIM\textit{w.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

Default: 6

\textit{d}

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

**Details**

The EURTOFIM\textit{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\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofim6.;</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofim12.2;</td>
<td>5.95</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofim6.;</td>
<td>7340</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofim12.2;</td>
<td>7340.36</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtofim6.;</td>
<td>73400</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofim12.2;</td>
<td>73400.04</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
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

\( d \)

- 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 “Overview to European Currency Conversion” on page 62.

---

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtofrf6.;</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>6.56</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtofrf6.;</td>
<td>8098</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>8098.18</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtofrf6.;</td>
<td>80978</td>
</tr>
<tr>
<td></td>
<td>put amount eurtofrf12.2;</td>
<td>80977.89</td>
</tr>
</tbody>
</table>
**EURTOGBP w.d Format**

Converting 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

- \( d \)
  - 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 “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtogbp6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogbp12.2;</td>
<td>0.70</td>
</tr>
</tbody>
</table>
EURTOGRD\(w.d\) Format

Converts an amount from euros to Greek drachmas.

**Category:** Currency Conversion  
**Alignment:** right

**Syntax**

\[
\text{EURTOGRD}w.d
\]

**Syntax Description**

\(w\)

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

\(d\)

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 “Overview to European Currency Conversion” on page 62.
Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtogr8.;</td>
<td>341</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>340.89</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtogr8.;</td>
<td>420843</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>420842.99</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtogr8.;</td>
<td>4208225</td>
</tr>
<tr>
<td></td>
<td>put amount eurtogr16.2;</td>
<td>4208225.33</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURFRGRDw.d Format” on page 600
Functions:
“EUROCURR Function” on page 650

EURTOHUFw.d Format

Converts an amount from euros to Hungarian forints.

Category: Currency Conversion
Alignment: right

Syntax
EURTOHUFw.d

Syntax Description

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

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

Details

The EURTOHUFw.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\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtohuf8.;</td>
<td>260</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>260.33</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtohuf8.;</td>
<td>321387</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>321386.83</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtohuf8.;</td>
<td>3213712</td>
</tr>
<tr>
<td></td>
<td>put amount eurtohuf14.2;</td>
<td>3213712.13</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURFRHUF\textit{w.d} Format” on page 601

Functions:

“EUROCURR Function” on page 650

---

**EURTOIEP\textit{w.d} Format**

Converts an amount from euros to Irish pounds.

**Category:** Currency Conversion

**Alignment:** right

**Syntax**

\textbf{EURTOIEP}\textit{w.d}

**Syntax Description**

\textit{w}

specifies the width of the output field.

**Default:** 6

\textit{d}
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 **EUROCURRE** function. For more information about European currency conversion and currency conversion rate tables, see "Overview to European Currency Conversion" on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoiep6.;</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>0.79</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoiep6.;</td>
<td>972</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>972.30</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoiep6.;</td>
<td>9722</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoiep12.2;</td>
<td>9722.48</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
- “EURFRIEP** \( w.d \) Format” on page 602

Functions:
- “EUROCURRE Function” on page 650

**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

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

Details

The EURTOITLw.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 EURTOITLw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoitl8.;</td>
<td>1936</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl12.2;</td>
<td>1936.27</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoitl8.;</td>
<td>2390441</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl112.2;</td>
<td>2390441.49</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoitl8.;</td>
<td>23903253</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoitl112.2;</td>
<td>23903253.15</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRITLw.d Format” on page 603

Functions:

“EUROCURR Function” on page 650

EURTOLUFw.d Format

Converts an amount from euros to Luxembourg francs.

Category: Currency Conversion

Alignment: right
Syntax Description

\( w \)

specifies the width of the output field.

**Default:** 6

\( d \)

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 “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoluf6.;</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>40.34</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoluf6.;</td>
<td>49802</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>49802.03</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoluf6.;</td>
<td>497996</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoluf12.2;</td>
<td>497996.07</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRLUF\( w.d \) Format” on page 605

Functions:

“EUROCURR Function” on page 650

EURTONLG\( w.d \) Format

Converts an amount from euros to Dutch guilders.

**Category:** Currency Conversion

**Alignment:** right
Syntax

EURONLGw.d

Syntax Description

\( w \)

specifies the width of the output field.

**Default:** \( 6 \)

\( d \)

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

Details

The EURONLGw.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 EURONLGw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonlg6.;</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>2.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtonlg6.;</td>
<td>2721</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>2720.61</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtonlg6.;</td>
<td>27205</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonlg12.2;</td>
<td>27204.80</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRNLGw.d Format” on page 606

Functions:

“EUROCURR Function” on page 650
EURTONOK\textit{w.d} Format

Converts an amount from euros to Norwegian krone.

**Category:** Currency Conversion  
**Alignment:** right

**Syntax\**

\texttt{EURTONOK\textit{w.d}}

**Syntax Description**

\textit{w}

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

\textit{d}

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

**Details**

The EURTONOK\textit{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\textit{w.d} format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtonok6.;</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>9.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtonok6.;</td>
<td>11355</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>11355.11</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtonok6.;</td>
<td>113546</td>
</tr>
<tr>
<td></td>
<td>put amount eurtonok12.2;</td>
<td>113545.61</td>
</tr>
</tbody>
</table>

**See Also**

Formats:
EURTOPLZw.d Format

Converts an amount from euros to Polish zlotys.

Category: Currency Conversion
Alignment: right

Syntax
EURTOPLZw.d

Syntax Description

\( w \)

specifies the width of the output field.
Default: 6

\( d \)

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

Details

The EURTOPLZw.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 EURTOPLZw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoplz6.;</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplzl2.2;</td>
<td>4.20</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoplzl6.;</td>
<td>5185</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplzl2.2;</td>
<td>5185.15</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoplzl6.;</td>
<td>51849</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoplzl2.2;</td>
<td>51849.00</td>
</tr>
</tbody>
</table>
See Also

Formats:
“EURFRPLZw.d Format” on page 608
Functions:
“EUROCURRE Function” on page 650

EURTOPTEm.d Format

Converts an amount from euros to Portuguese escudos.

Category: Currency Conversion
Alignment: right

Syntax
EURTOPTEm.d

Syntax Description

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

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

Details

The EURTOPTEm.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 EURTOPTEm.d format and the EUROCURRE function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount europte8.;</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td>put amount europte12.2;</td>
<td>200.48</td>
</tr>
</tbody>
</table>
EURTOROL\texttt{w.d} Format

Converts an amount from euros to Romanian lei.

**Category:** Currency Conversion

**Alignment:** right

### Syntax

\texttt{EURTOROLw.d}

### Syntax Description

\texttt{w}

specifies the width of the output field.

**Default:** 6

\texttt{d}

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

### Details

The \texttt{EURTOROLw.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 \texttt{EURTOROLw.d} format and the \texttt{EUROCURRE} function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234.56</td>
<td>\texttt{put amount eurtopte8.} \texttt{;}</td>
<td>247507</td>
</tr>
<tr>
<td></td>
<td>\texttt{put amount eurtopte12.2} \texttt{;}</td>
<td>247507.06</td>
</tr>
<tr>
<td>12345</td>
<td>\texttt{put amount eurtopte8.} \texttt{;}</td>
<td>2474950</td>
</tr>
<tr>
<td></td>
<td>\texttt{put amount eurtopte12.2};</td>
<td>2474950.29</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“\texttt{EURFRPTEw.d Format}” on page 609

Functions:

“\texttt{EUROCURRE} Function” on page 650
Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>put amount eurtor16.;</code></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td><code>put amount eurtor12.2;</code></td>
<td>13.71</td>
</tr>
<tr>
<td>1234.56</td>
<td><code>put amount eurtor16.;</code></td>
<td>16926</td>
</tr>
<tr>
<td></td>
<td><code>put amount eurtor12.2;</code></td>
<td>16925.82</td>
</tr>
<tr>
<td>12345</td>
<td><code>put amount eurtor16.;</code></td>
<td>169250</td>
</tr>
<tr>
<td></td>
<td><code>put amount eurtor12.2;</code></td>
<td>169249.95</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRROLw.d Format” on page 610

EURTORURw.d Format

Converts an amount from euros to Russian rubles.

Category:  Currency Conversion

Alignment:  right

Syntax

EURTORURw.d

Syntax Description

\(w\)

specifies the width of the output field.

Default:  6

\(d\)

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

Details

The EURTORURw.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
EURTORURw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtorur6.;</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>put amount eurtorur12.2;</td>
<td>19.77</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtorur6.;</td>
<td>24405</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>24404.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtorur6.;</td>
<td>244036</td>
</tr>
<tr>
<td></td>
<td>put amount eurtorur12.2;</td>
<td>244035.96</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURFRRURw.d Format” on page 611

Functions:
“EUROCURR Function” on page 650

EURTOSEKw.d Format

Converts an amount from euros to Swedish kronor.

Category: Currency Conversion

Syntax

EURTOSEKw.d

Syntax Description

$w$

specifies the width of the output field.

Default: 6

$d$
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 “Overview to European Currency Conversion” on page 62.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtosek6.;</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>9.37</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtosek6.;</td>
<td>11563</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>11562.78</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtosek6.;</td>
<td>115622</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosek12.2;</td>
<td>115622.16</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURFRSEK\(w.d\) Format” on page 612

Functions:

“EUROCURR Function” on page 650

---

**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

\( d \)

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 “Overview to European Currency Conversion” on page 62.

**Note:** Slovenia’s currency is the Euro. The information for EURTOSIT is provided for user’s historical data.

**Examples**

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtosit8.;</td>
<td>191</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosit14.2;</td>
<td>191.00</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtosit8.;</td>
<td>235801</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosit14.2;</td>
<td>235800.96</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtosit8.;</td>
<td>2357895</td>
</tr>
<tr>
<td></td>
<td>put amount eurtosit14.2;</td>
<td>2357895.00</td>
</tr>
</tbody>
</table>

**See Also**

Formats:

“EURFRSIT\(w.d\) Format” on page 613

Functions:

“EUROCURR Function” on page 650

---

**EURTOTRL\(w.d\) Format**

Converts an amount from euros to Turkish liras.

**Category:** Currency Conversion

**Alignment:** right
Additional NLS Language Elements  △  EURTOYUDw.d Format  643

Syntax

EURTOTRLo.d

Syntax Description

\( w \)

specifies the width of the output field.

Default: 6

\( d \)

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

Details

The EURTOTRLo.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 EURTOTRLo.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtotrl8.; put amount eurtotrl14.2;</td>
<td>337</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtotrl8.; put amount eurtotrl14.2;</td>
<td>415938</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtotrl8.; put amount eurtotrl14.2;</td>
<td>4159179</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRTRLw.d Format” on page 615

Functions:

“EUROCURR Function” on page 650

EURTOYUDw.d Format

Converts an amount from euros to Yugoslavian dinars.
Category: Currency Conversion

Syntax

EURTOYUDw.d

Syntax Description

w
specifies the width of the output field.

Default: 6

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

Details

The EURTOYUDw.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 EURTOYUDw.d format and the EUROCURR function. For more information about European currency conversion and currency conversion rate tables, see “Overview to European Currency Conversion” on page 62.

Examples

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

<table>
<thead>
<tr>
<th>Amounts</th>
<th>Statements</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>put amount eurtoyud6.;</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoyud12.2;</td>
<td>13.06</td>
</tr>
<tr>
<td>1234.56</td>
<td>put amount eurtoyud6.;</td>
<td>16129</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoyud12.2;</td>
<td>16128.79</td>
</tr>
<tr>
<td>12345</td>
<td>put amount eurtoyud6.;</td>
<td>161280</td>
</tr>
<tr>
<td></td>
<td>put amount eurtoyud12.2;</td>
<td>161280.02</td>
</tr>
</tbody>
</table>

See Also

Formats:

“EURFRYUDw.d Format” on page 616

Functions:

“EUROCURR Function” on page 650
**EURDFDE\textit{w}. Informat**

Reads international date values.

Category: Date and Time

**Syntax**

\texttt{EURDFDE\textit{w}.}

\textit{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 \textit{w} range is 10–32 and the default \textit{w} is 10.

**Details**

The date values must be in the form \textit{ddmmmyy} or \textit{ddmmmyyyy}:

\textit{dd}

is an integer from 01–31 that represents the day of the month.

\textit{mmm}

is the first three letters of the month name.

\textit{yy} or \textit{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 \texttt{YEARCUTOFF=} system option.

You can set the language for the SAS session with the \texttt{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 “\texttt{DFLANG=} System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the \texttt{DFLANG=} system option.
Examples

This INPUT statement uses the value of the DFLANG= system option to read the international date values in Spanish.

```plaintext
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.

```plaintext
input day espdfde10.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>01abr1999</td>
<td>14335</td>
</tr>
<tr>
<td>01-abr-99</td>
<td>14335</td>
</tr>
</tbody>
</table>

See Also

Formats:
“EURDFDEw. Format” on page 572

Informats:
DATEw. in SAS Language Reference: Dictionary
“EURDFDTw. Informat” on page 646
“EURDFMYw. Informat” on page 648

System Options:
“DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460
YEARCUTOFF= in SAS Language Reference: Dictionary

EURDFDTw. Informat

Reads international datetime values in the form `ddmmyy hh:mm:ss.ss` or `ddmmyyyy hh:mm:ss.ss`.

Category: Date and Time

Syntax

EURDFDTw.

Syntax Description
Additional NLS Language Elements  △ EURDFDTw. Informat  647

### Additional NLS Language Elements

#### EURDFDTw. Informat

The EURDFDTw. 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 460 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.

```sas
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.

```sas
input date deudfdt20.;
```
## EURDFMY\(w\). Informat

**Reads** month and year date values in the form *mmmy* or *mmmyyyy*.

**Category:** Date and Time

### Syntax

**EURDFMY\(w\).**

### Syntax Description

\(w\)

specifies the width of the input field.

**Default:** 5 (except Finnish)

---

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>23dez99:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
<tr>
<td>23dez1999:10:03:17.2</td>
<td>1261562597.2</td>
</tr>
</tbody>
</table>

---

### See Also

**Formats:**

- DATE\(w\) in *SAS Language Reference: Dictionary*
- DATETIME\(w.d\) in *SAS Language Reference: Dictionary*
- “EURDFDT\(w.d\) Format” on page 576
- TIME\(w.d\) in *SAS Language Reference: Dictionary*

**Functions:**

- DATETIME in *SAS Language Reference: Dictionary*

**Informats:**

- DATETIME\(w\) in *SAS Language Reference: Dictionary*
- “EURDFD\(w\). Informat” on page 645
- “EURDFMY\(w\). Informat” on page 648

**System Options:**

- “DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460
- YEARCUTOFF= in *SAS Language Reference: Dictionary*
**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 \texttt{mmmyy} or \texttt{mmmyyyy}:

- \texttt{mmm} is the first three letters of the month name.
- \texttt{yy} or \texttt{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 \texttt{YEARCUTOFF=} system option.

You can set the language for the SAS session with the \texttt{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 “\texttt{DFLANG=} System Option: OpenVMS, UNIX, Windows, and z/OS” on page 460 for the list of language prefixes. When you specify the language prefix in the informat, SAS ignores the \texttt{DFLANG=} option.

**Examples**

This INPUT statement uses the value of \texttt{DFLANG=} system option to read the international date values in French.

```sas
options dflang=french;
input month eurdfmy7.;
```

The second INPUT statement uses the French language prefix, and DFLANG is not specified.

```sas
input month fradfmy7.;
```

<table>
<thead>
<tr>
<th>Values</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>avr1999</td>
<td>14335</td>
</tr>
<tr>
<td>avr 99</td>
<td>14335</td>
</tr>
</tbody>
</table>
See Also

Formats:

- DDMYYw. in SAS Language Reference: Dictionary
- "EURDFMYw. Format" on page 582
- 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:

- "EURDFDEw. Informat" on page 645
- "EURDFDTw. Informat" on page 646
- MONYYw. in SAS Language Reference: Dictionary

System Options:

- "DFLANG= System Option: OpenVMS, UNIX, Windows, and z/OS" on page 460
- YEARCUTOFF= in SAS Language Reference: Dictionary

---

**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 Codes Table A1.1 on page 651.)
  
  **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.

- **to-currency-code**
  - specifies a three-character currency code that identifies the currency that you are converting to. (See European Currency and Currency Codes Table A1.1 on page 651.)
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. Several countries use the Euro as their currency instead of the currency listed in the following table. This information is provided in order to satisfy user's historical data.

Table A1.1 European Currency and Currency Codes

<table>
<thead>
<tr>
<th>Currency</th>
<th>Currency code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austrian schilling</td>
<td>ATS</td>
</tr>
<tr>
<td>Belgian franc</td>
<td>BEF</td>
</tr>
<tr>
<td>British pound sterling</td>
<td>GBP</td>
</tr>
<tr>
<td>Czech koruna</td>
<td>CZK</td>
</tr>
<tr>
<td>Danish krone</td>
<td>DKK</td>
</tr>
<tr>
<td>Deutsche mark</td>
<td>DEM</td>
</tr>
<tr>
<td>Dutch guilder</td>
<td>NLG</td>
</tr>
<tr>
<td>Euro</td>
<td>EUR</td>
</tr>
<tr>
<td>Finnish markka</td>
<td>FIM</td>
</tr>
<tr>
<td>French franc</td>
<td>FRF</td>
</tr>
<tr>
<td>Greek drachma</td>
<td>GRD</td>
</tr>
<tr>
<td>Hungarian forint</td>
<td>HUF</td>
</tr>
<tr>
<td>Irish pound</td>
<td>IEP</td>
</tr>
<tr>
<td>Italian lira</td>
<td>ITL</td>
</tr>
<tr>
<td>Luxembourg franc</td>
<td>LUF</td>
</tr>
<tr>
<td>Norwegian krone</td>
<td>NOK</td>
</tr>
<tr>
<td>Polish zloty</td>
<td>PLZ</td>
</tr>
<tr>
<td>Portuguese escudo</td>
<td>PTE</td>
</tr>
<tr>
<td>Romanian leu</td>
<td>ROL</td>
</tr>
<tr>
<td>Russian ruble</td>
<td>RUR</td>
</tr>
<tr>
<td>Slovenian tolar</td>
<td>SIT</td>
</tr>
<tr>
<td>Spanish peseta</td>
<td>ESP</td>
</tr>
<tr>
<td>Swedish krona</td>
<td>SEK</td>
</tr>
<tr>
<td>Swiss franc</td>
<td>CHF</td>
</tr>
<tr>
<td>Turkish lira</td>
<td>TRL</td>
</tr>
<tr>
<td>Yugoslavian dinar</td>
<td>YUD</td>
</tr>
</tbody>
</table>

The EUROCURR function converts a specific country's currency to an equivalent amount in another country's currency. It can also 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 652. 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 652. 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 652.

Examples

Example 1: Converting from Deutsche Marks to Euros  
The following example converts one Deutsche mark to an equivalent amount of euros.

```sas
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 one euro to an equivalent amount of Deutsche marks.

```sas
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.

```sas
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.

```sas
data _null_;  
x=50;  
    amount=eurocurr(x,' ', 'dem');  
    put amount=;  
run;
```

The value in the SAS log is: amount=97.7915.
See Also

Formats:
  “EUROw.d Format” on page 77
  “EUROXw.d Format” on page 78

Informats:
  “EUROw.d Informat” on page 320
  “EUROXw.d Informat” on page 322
Recommended Reading

Here is the recommended reading list for this title:

- SAS Language Reference: Concepts
- SAS Language Reference: Dictionary
- Base SAS Procedures Guide
- SAS/CONNECT User's Guide
- SAS/GRAPH: Reference
- SAS Companion for your operating environment

For a complete list of SAS publications, go to support.sas.com/bookstore. If you have questions about which titles you need, please contact a SAS Publishing Sales Representative at:

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513
Telephone: 1-800-727-3228
Fax: 1-919-531-9439
E-mail: sasbook@sas.com
Web address: support.sas.com/bookstore

Customers outside the United States and Canada, please contact your local SAS office for assistance.
Glossary

ANSI (American National Standards Institute)
an organization in the United States that coordinates voluntary standards and
certainty 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
open mathematical symbols), punctuation marks, and control characters. This set of 128
characters is also included in many 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) 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

Numbers
8859 ISO family 13

A
alignment
  character expressions 271, 279
  left-aligning arguments 150
ANSI (American National Standards Institute) 43
Arabic characters
  reversing 74
  storing logical-ordered text on visual server arguments
  converting to lowercase 273
  converting to uppercase 285
  extracting a substring from 281
  extracting a substring from, based on byte position
    length of 272
  transcribing for specified argument 302
ASCII
  transferring data between EBCDIC and
    ASCII option
    PROC SORT statement 178
  ATTRIB statement 109
  TRANSCODE= option 109
Australia
  monetary format 111, 151
Austria
  monetary format 161
Austrian schillings
  converting to euros 589
  by category 149

B
BASETYPE= option
  PROC DBCSTAB statement 310
Belgium
  monetary format 231, 63
$BIDw. format 74
binary collation 66
blanks
  compressing multiple 450
  removing leading and trailing 150
  removing leading DBCS blanks 527
  trimming trailing 271
BOMFILE system option 455
BOTH option
  CLEAR statement (TRANTAB) 521
  LIST statement (TRANTAB) 522
  SAVE statement (TRANTAB) 524
Brazil
  monetary format 133, 53
Bulgaria
  monetary format 133, 52
Byte Order Mark (BOM) prefix
  on Unicode external files 455

C
Canada
  monetary format 114, 154
case
  changing uppercase characters to lowercase 55
    CATALOG= option
    PROC DBCSTAB statement 510
CEDA 53
character data
  avoiding truncation of 124, 125
  reading from right to left
    character expressions
    comparing 166
    compressing 286
    concatenating 286
  deleting character value contents 286
  inserting character value contents, based on byte unit
    inserting character value contents 286
    inserting character value contents, based on byte unit
  left-aligning 273
  number of double-byte characters in position of first unique character
  removing trailing blanks and SO/SI 284
  replacing character value contents 286
  replacing character value contents, based on byte unit
  replacing specific characters 283
  reversing 270
  right-aligning 276
  searching for specific characters 271
  searching for string of characters
    selecting a specified word from
    translating 288
    trimming 284
    updating 286
Index

updating, based on byte unit 288

verifying 288

character-set encoding

for SAS session 461

character sets 12

definition 10

displaying DBCS 86

specifying, for META declaration in output translation tables and 512

character strings

split 38

character variables

transcoding enabled for specified variable 501

CHARSET= option 476

China

monetary format 116, 156

CIMPORT procedure

cross-environment data access (CEDA) 29

transcoding between operating environments translation tables with 29

CLEAR statement

TRANTAB procedure 521

code page 8

collating sequence 16

alternate sequences 17

binary collation 17

encoding value 18

language-specific linguistic collation overview 15

results of different sequences 16

translation tables 16

commas, removing 448

compatibility 445

SAS string functions 244

compatible encodings 84

compressing multiple blanks 150

concatenation

character expressions 291

conversion tables

creating 511

for DBCS 809

Japanese 512

CPORT procedure

cross-environment data access (CEDA) 29

transcoding between operating environments translation tables with $CPTDWw. format 76

CPTDWw. informat 518

$CPTDWw. format 76

CPTDWw. informat 518

Croatia

monetary format 24

cross-environment data access (CEDA) 25

currency 8 85

converting one European currency to another customized representations 56

dollars 55

European conversion example 81

localized euros 82

localized representations 74

unique representations 55

yen 24

CVP engine

avoiding character data truncation 88

Czech Republic

monetary format 114 157

D

DANISH option

PROC SORT statement 478

data conversion between DBCS encodings 2

DATA= option

PROC DBCSTAB statement 510

data set options 8

for transcoding data sets encoding 21

encoding support, by release 9

mixed encodings 5

suppressing transcoding
transcode attributes of variables with a particular encoding 100

date format descriptors

date values

as a date 21

converting to specified locale 28

day of week 21

Hebrew 69

international 643

international month and year 645

Japanese 88 331

Jewish calendar 81

name and day of month 26

Tibetan 89 330

taiwanese 86 330

year 98

year and name of month 96

year and quarter 77

year and week 24

DAYS=

date values as 21

DATESTYLE= system option

default values 524

datetime-format descriptors 92

datetime formats 21

datetime values

as a datetime 104

converting to specified locale 293 294

day of week 105

day of week and datetime 106

international 647

name and day of month 102

name of month 103

name of month, day of month and year 102

time of day 104

with a.m. or p.m. 101

year 109

year and name of month 103

year and name of week year and quarter 108

DBCS

See double-byte character sets (DBCS)

DBCS data

adding shift-code data to 82 83

removing shift-code data from 83 83

DBCS encoding 83 85

See also double-byte character sets (DBCS)

current data truncation 58

data conversion between encodings 57

East Asian languages 52

encoding values for transcoding data 555
full-screen
full-screen input method modules (IMMs)
removing leading blanks
requirements for displaying character sets
SAS on mainframe and
specifying
split character strings
system option values for
system options in SAS session for
trimming blanks and SO/SI
trimming trailing blanks
when you can use
DBC system option
DBC LANG= option
PROC DBCSTAB statement
DBCSTAB system option
creating conversion tables
decimal points, removing
denomination
East Asian languages
double-byte character sets (DBCS)
double-byte characters
decimal points, removing
Denmark
monetary format
desc= option
PROC DBCSTAB statement
dflang= system option
dollars
double-byte character sets (DBCS)
openEdition encoding and
transferring data between ASCII and
encoding method
language for
recognizing
double-byte characters
number in a character expression

E
East Asian languages
double-byte character sets (DBCS)
EBCDIC
code point discrepancies among encodings
German code page
OpenEdition encoding and
transferring data between ASCII and
EBCDIC option
PROC SORT statement
Egypt
monetary format
encoding
behavior in SAS sessions
character sets for
compatibility for transcoding
converting one type of data to another
data set support by release
data sets
default SAS session encoding
definition
for East Asian languages
input processing
mixed
output processing
overriding
reading and writing external files
SAS sessions
setting for SAS sessions
standards organizations for
versus transcoding
z/OS support
ENCODING= data set option
coding methods
for DBCS
ENCODING= system option
default settings
Posix values
encoding values
DBC
default, based on LANGUAGE= system option
default SAS session values
for transcoding data
OpenVMS
SBCS
Unicode
UNIX
Windows
z/OS
escape codes
Estonia
monetary format
EUR language elements
EURFDDW.d format
EURFDEW.d format
EURFD.EW. format
EURFD.EW. informat
EURFDN.W. format
EURFD.TW. format
EURFD.TW. informat
EURFRWNW. format
EURFMNW. format
EURFMYW. format
EURFMYW. informat
EURFDWXW. format
EURFXXW. format
EURFRATS.d format
EURFRBEF.d format
EURFRCHF.d format
EURFCZK.d format
EURFDEM.d format
EURFDD.d format
EURFDDKK.d format
EURFRES d format
EURFRIM.d format
EURFRK.d format
EURFRGBP.d format
EURFRGR.d format
EURFRHF.d format
EURFRJ.d format
EURFRKL.d format
EURFRNL.d format
EURFRNOK.d format
EURFRPL.d format
EURFRPRTE.d format
EURFRROL.d format
EURFRUR.d format
EURFRSEK.d format
EURFRSD.d format
EURFRYUD.d format
euro conversion
Austrian schillings to euros 589
example 53
fixed rates for 52
variable rates for 52
EUROCURR function 551
European currency conversion 52
converting one currency to another 55
direct conversion between currencies example 53
fixed rates for euro conversion 52
variable rates for euro conversion 52
euros
formats for 53, 52
localized 54
EUROw.d format 52
EUROw.d informat 324
EUROXw.d format 52
EUROXw.d informat 324
EUROATSw.d format 517
EUROTOBELw.d format 518
EUROTOCHFw.d format 610
EUROTOCZKw.d format 610
EUROTODEMw.d format 522
EUROTODKKw.d format 525
EUROTFIMw.d format 525
EUROTRFW.d format 525
EUROTOGBPw.d format 610
EUROTOGRw.d format 528
EUROTOHUw.d format 528
EUROTOJEPw.d format 530
EUROTOITw.d format 530
EUROTOULw.d format 530
EUROTOULw.d format 530
EUROTOLGw.d format 530
EUROTOKw.d format 535
EUROTOLZW.d format 535
EUROTONEw.d format 535
EUROTORLw.d format 539
EUROTORGw.d format 539
EUROTOSw.d format 540
EUROTOSITW.d format 542
EUROTOTW.d format 543
EUROUDW.d format 544
external files
BOM prefix on Unicode files encoding and 55

F

Faroe Island monetary format 118, 138
Finland monetary format 21, 161
FINNISH option
PROC SORT statement 170
FORCE option
PROC DBCSSTAB statement 313
formats 313
associating with variables 309
by category 62
international date and datetime language for international dates 50
supporting DBCS on SO/SI systems 57
France monetary format 21, 161
FSDBTYPE system option 462
FSIMM system option 65
FSIMMOPT system option 464
full-screen DBCS encoding
input method modules (IMMs) for functions by category 201
K functions 24
SAS string functions 244

G

German EBCDIC code page 1
Germany monetary format 121, 161
GETPXLANGUAGE function 50
GETPXLOCAL function 50
GETPXREGION function 50
Greece monetary format 21, 161
Greenland monetary format 32, 158

H

HDATEw. format 79
HEBDATEw. format 81
Hebrew characters reversing 74
storing logical-ordered text on visual server 84, 85
Hebrew date values 76
hexadecimal representation translation tables in 522
Hong Kong monetary format 22, 163
Hungary monetary format 25, 165

I

IBw.d informat 112, 117, 118
illegal data 469
incompatible encodings 31
India monetary format 22, 168
Indonesia monetary format 22, 166
informs associating with variables 499
by category 51
language for international dates 160
supporting DBCS on SO/SI systems 57
input method modules (IMMs) for options for 463
input processing 25
overriding encoding 46
integer binary values, reading 418, 416, 417, 418
international date and datetime formats 50
international date formats and informats specifying language for international date values, writing day-of-week and date 59
day-of-week name 57
day-of-week number ddmmyy 57
dd.mm.yy 57
mmmy 82
month name 581, 584
international datetime values, writing
ddmmmyy:hh:mm:ss:ss 576
International Organization for Standardization (ISO) 15
internationalization 4
INVERSE statement
TRANTAB procedure 523
Ireland
monetary format 231, 161
ISO encodings
8859 family 15
Windows family 15
ISO (International Organization for Standardization) 15
Israel
monetary format 231, 161
Italy
monetary format 231, 161
J
Japan
monetary format 286, 169
Japanese conversion tables 51, 121
Japanese dates 88, 331
JDATEYMDw. informat 323
JNENGOW. informat 325
K
K functions 244
%KANJIw. format 82
%KANJIw. informat 326
%KANJIXw. format 83
%KANJIXw. informat 327
%KCMPRES autocall macro 450
KCOMPARE function 266
KCOMPRESS function 267
KCOUNT function 268
KCVT function 268
KINDEX function 270
KINDEXC function 271
%KLEFT autocal macro 271
KLEFT function 271
KLENGTH function 272
KLOWCASE function 273
KLOWCASE macro 273
KREVERSE function 279
KRIGHT function 279
KSCAN function 280
KSTRCAT function 280
KSUBSTR function 281
KSUBSTRB function 282
KTRANSLATE function 283
%KTRIM autocal macro 284
KTRIM function 284
KTRUNCATE function 285
KUPCASE function 285
KUPDATE function 286
KUPDATEB function 287
%KVERIFY autocal macro 288
KVERIFY function 288
L
labels, associating with variables 495
language 65
language codes
current two-letter code 65
language switching 65
changing text language of ODS output 65
languages for international date informat and formats 65
Latin 1 code page 65
Latvia
monetary format 133, 173
leading blanks
removing 451
left-aligning arguments 450
left-aligning character expressions 271
length
associating with variables of arguments 271
Liechtenstein
monetary format 135, 175
line-feed characters 31
linguistic collation 19
linguistic sort keys 296
LIST statement
TRANTAB procedure 523
Lithuania
monetary format 31, 171
LOAD statement
TRANTAB procedure 523
locale 5
best numerical notation based on 29
converting date values to specified locale 29
converting datetime values to specified locale 29
converting time or datetime values to specified locale 29
language switching 5
of SAS sessions 29
POSIX value for specifying 29
specifying at SAS invocation 29
specifying during SAS session 29
LOCALE= system option 465
default encoding values based on 29
values for 23
LOCALELANGCHT system option 465
localization 4
localized euros 54
localized international currency representation 58
localized national currency representation 57
logical-ordered text
storing on visual server 54
LSLOGVSRw. format 55
LSLOGVSRw. informat 324
LSLOGVSw. format 55
LSLOGVSw. informat 323
lowercase characters
changing uppercase characters to 151
converting arguments to 151
Luxembourg
monetary format 231, 161
M
Macau
monetary format 233, 173
mainframes
DBCS and 15
<table>
<thead>
<tr>
<th>Country</th>
<th>Monetary Format</th>
<th>Monetary Formats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Austria</td>
<td>63</td>
<td>151</td>
</tr>
<tr>
<td>Brazil</td>
<td>112</td>
<td>149</td>
</tr>
<tr>
<td>Bulgaria</td>
<td>166</td>
<td>152</td>
</tr>
<tr>
<td>Canada</td>
<td>114</td>
<td>148</td>
</tr>
<tr>
<td>China</td>
<td>167</td>
<td>153</td>
</tr>
<tr>
<td>Croatia</td>
<td>120</td>
<td>146</td>
</tr>
<tr>
<td>Czech Republic</td>
<td>123</td>
<td>147</td>
</tr>
<tr>
<td>Denmark</td>
<td>113</td>
<td>145</td>
</tr>
<tr>
<td>Estonia</td>
<td>119</td>
<td>143</td>
</tr>
<tr>
<td>Faroe Island</td>
<td>122</td>
<td>144</td>
</tr>
<tr>
<td>Finland</td>
<td>124</td>
<td>144</td>
</tr>
<tr>
<td>France</td>
<td>127</td>
<td>145</td>
</tr>
<tr>
<td>Germany</td>
<td>128</td>
<td>146</td>
</tr>
<tr>
<td>Greece</td>
<td>129</td>
<td>148</td>
</tr>
<tr>
<td>Greenland</td>
<td>130</td>
<td>147</td>
</tr>
<tr>
<td>Hong Kong</td>
<td>131</td>
<td>146</td>
</tr>
<tr>
<td>Hungary</td>
<td>132</td>
<td>146</td>
</tr>
<tr>
<td>India</td>
<td>133</td>
<td>147</td>
</tr>
<tr>
<td>Indonesia</td>
<td>134</td>
<td>148</td>
</tr>
<tr>
<td>Ireland</td>
<td>135</td>
<td>149</td>
</tr>
<tr>
<td>Israel</td>
<td>136</td>
<td>147</td>
</tr>
<tr>
<td>Italy</td>
<td>137</td>
<td>146</td>
</tr>
<tr>
<td>Japan</td>
<td>138</td>
<td>148</td>
</tr>
<tr>
<td>Latvia</td>
<td>139</td>
<td>149</td>
</tr>
<tr>
<td>Liechtenstein</td>
<td>140</td>
<td>150</td>
</tr>
<tr>
<td>Lithuania</td>
<td>141</td>
<td>151</td>
</tr>
<tr>
<td>Luxembourg</td>
<td>142</td>
<td>152</td>
</tr>
<tr>
<td>Macau</td>
<td>143</td>
<td>153</td>
</tr>
<tr>
<td>Malaysia</td>
<td>144</td>
<td>154</td>
</tr>
<tr>
<td>Malta</td>
<td>145</td>
<td>155</td>
</tr>
<tr>
<td>Mexico</td>
<td>146</td>
<td>156</td>
</tr>
<tr>
<td>Netherlands</td>
<td>147</td>
<td>157</td>
</tr>
<tr>
<td>New Zealand</td>
<td>148</td>
<td>158</td>
</tr>
<tr>
<td>Norway</td>
<td>149</td>
<td>159</td>
</tr>
<tr>
<td>Poland</td>
<td>150</td>
<td>160</td>
</tr>
<tr>
<td>Portugal</td>
<td>151</td>
<td>161</td>
</tr>
<tr>
<td>Puerto Rico</td>
<td>152</td>
<td>162</td>
</tr>
<tr>
<td>Romania</td>
<td>153</td>
<td>163</td>
</tr>
<tr>
<td>Russia</td>
<td>154</td>
<td>164</td>
</tr>
<tr>
<td>Singapore</td>
<td>155</td>
<td>165</td>
</tr>
<tr>
<td>Slovakia</td>
<td>156</td>
<td>166</td>
</tr>
<tr>
<td>Slovenia</td>
<td>157</td>
<td>167</td>
</tr>
<tr>
<td>South Africa</td>
<td>158</td>
<td>168</td>
</tr>
<tr>
<td>South Korea</td>
<td>159</td>
<td>169</td>
</tr>
<tr>
<td>Spain</td>
<td>160</td>
<td>170</td>
</tr>
<tr>
<td>Sweden</td>
<td>161</td>
<td>171</td>
</tr>
<tr>
<td>Switzerland</td>
<td>162</td>
<td>172</td>
</tr>
<tr>
<td>Taiwan</td>
<td>163</td>
<td>173</td>
</tr>
<tr>
<td>Thailand</td>
<td>164</td>
<td>174</td>
</tr>
<tr>
<td>Turkey</td>
<td>165</td>
<td>175</td>
</tr>
<tr>
<td>United Arab Emirates</td>
<td>166</td>
<td>176</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>167</td>
<td>177</td>
</tr>
<tr>
<td>United States</td>
<td>168</td>
<td>178</td>
</tr>
</tbody>
</table>

### N

**National Language Support**

See NLS (National Language Support)

**NATIONAL option**

PROC SORT statement

**NENGOw. format**

**NENGOw. informat**

Netherlands

monetary format

New Zealand

monetary format

United States

monetary representations
NLMLZARw.d format 189
NLMLZARw.d informat 413
NLMLNY1w.d format 192
NLMLNYw.d format 191
NLNUM1w.d format 194
NLNUMw.d format 193
NLPCw.d format 196
NLPCw.d informat 419
NLPCw.d informat 421
NLS (National Language Support) 3
compatibility data set options 13
DBC5 25
DBCSTAB procedure 509
encoding formats 266
functions 260
iniformats 311
locale 3
statement options 475
system options 453
transcoding 23
TRANTAB procedure 515
NLS option
LOAD statement (TRANTAB) 522
PROC TRANTAB statement 520
NLSCOMPATMODE system option 465
NLTIMApw. format 205
NLTIMApw. informat 422
NLTIME descriptors 294
NLTIME function 294
NLTIMEw. format 204
NLTIMEw. informat 423
NODATM function 292
Norway monetary format 36 176
NORWEGIAN option
PROC SORT statement 478
numbers 3
numeric data
Japanese dates 81
Taiwanese date values 81
yen 34
numeric values
truncating 28
numerical notation
best, based on locale 81

P
paper size and measurement 8
PAPERSIZE= system option
default values 56
Poland monetary format 33 178
Portugal monetary format 33 161
PROC DBCSTAB statement 510
PROC TRANTAB statement 523
Puerto Rico monetary format 48 188

Q
%QKCMPRES autocall macro 450
%QKLEFT autocall macro 450
%QKTRIM autocall macro 450

R
region codes
current two-letter code 168
release compatibility 168
remote applications illegal data 169
Remote Library Services (RLS) translation tables with 518
REPLACE statement
TRANTAB procedure 523
SREVERJw. informat 425
compared to $REVERSw. informat 425
compared to $REVERJw. informat 425
right-alignment character expressions 57
RLS (Remote Library Services) translation tables with 518
Romania monetary format 33 179
RSASIOTRANSERROR system option 469
Russia monetary format 34 180

S
SAS/CONNECT
Compute Services 31
Data Transfer Services 30
Remote Library Services 30
SAS/GRAPH
translation tables in 518
SAS language elements using encoding values 555
SAS sessions
default character-set encoding default encoding 461
coding 31
coding behavior in locale of 465
setting encoding of specifying locale during system options for DBCS 559
SAS/SHARE
Remote Library Services 34

0
ODS output
changing language of the text 467
ONE option
CLEAR statement (TRANTAB) 522
LIST statement (TRANTAB) 522
SAVE statement (TRANTAB) 524
OpenEdition encoding 3
OpenVMS encoding values 561
operating environments transcoding between 24
OPT= option, TRANTAB statement output processing 50
overriding encoding 3
SAS string functions
  internationalization compatibility for 244
SAVE statement
  TRANTAB procedure 524
SBCS encoding 2
  encoding values for transcoding data searching
  for specific characters in character expression 271
  for string of characters in character expression shift-code data
  adding to DBCS data 55, 529
  removing from DBCS data 57
  shift out/shift in (SO/SI) 57
  trimming from character expressions 284
Singapore
  monetary format 143, 182
  single-byte character set (SBCS) 2
  encoding values for transcoding data 555
Slovakia
  monetary format 144, 184
Slovenia
  monetary format 143, 161, 183
  SO/SI (shift out/shift in) 57
  trimming from character expressions 284
sort keys
  linguistic 296
  sort order
    ASCII 478
    EBCDIC 478
SORT procedure
  language-specific collating sequence for translation tables in 475
  sorting
    translation tables for 530
  SORTKEY function 299
SORTSEQ= option
  PROC SORT statement 476
  SORTSEQ= system option 470
South Africa
  monetary format 149, 189
South Korea
  monetary format 144, 189
Spain
  monetary format 141, 181
  split character strings 38
SQL procedure
  language-specific collating sequence for standards organizations 29
  statement options 475
  string functions
    internationalization compatibility for substrings
      extracting from an argument 283
      extracting from an argument, based on byte position 282
SWAP statement
  TRANTAB procedure 524
Sweden
  monetary format 141, 183
SWEDISH option
  PROC SORT statement 476
Switzerland
  monetary format 113, 155
  system options
    by category 153
    DBCS values for transcoding 553
in SAS sessions for DBCS 555
T
  TABLE= option
    SAVE statement (TRANTAB) 524
Taiwan
  monetary format 147, 187
  Taiwanese dates 86, 330
Thailand
  monetary format 145, 185
  time 1
  time values
    converting to specified locale 294
    removing 451
    trimming 453
  transcoding errors 469
  translating character expressions 293
  translation tables 15, 515
    applying to transport files 515
    compatible and incompatible encodings 295
    considerations for 30
    EBCDIC and OpenEdition encodings enabled for specified argument 302
    enabled for specified character variable encoding values for line-feed characters preventing 35
    reasons for 29
    SAS options for suppressing 35
    transferring data between EBCDIC and ASCII translation tables and 31
    versus encoding 62
    transcoding errors 469
    translating character expressions 293
    translation tables 15, 515
    applying to transport files 515
    character sets and 516
    CIMPORT procedure with clearing positions 521
    CPORT procedure with creating 524
    definition 515
    editing 528, 531, 538
    hexadecimal representation of in SAS/GRAPH 527
    in SORT procedure 517
    inverse tables 52, 53
    loading into memory for editing 522
    modifying 517
    outside TRANTAB procedure positions 513
    Remote Library Services (RLS) with replacing characters in 523
    saving sorting data 524
    specifying 536
    storing 516
    swapping 525
    transcoding and 528
    transcoding by specified table 550
Index

Viewing 525
transport files
applying translation tables to 504
TRANTAB function 299
TRANTAB procedure 514, 519
CLEAR statement 516
creating translation tables examples 525
INVERSE statement 521
inverse translation tables 534
LIST statement 522
LOAD statement 522
modifying translation tables PROC TRANTAB statement 520
REPLACE statement 523
SAVE statement 524
storing translation tables SWAP statement 514
syntax 519
task table 519
translation tables and character sets translation tables for sorting viewing translation tables TRANTAB statement UPLOAD procedure 503
TRANTAB= system option 472
trimming trailing blanks 451
truncating numeric values 285
truncation of character data 38
Turkey monetary format 446, 456
TWO option
CLEAR statement (TRANTAB) 522
LIST statement (TRANTAB) 522
SAVE statement (TRANTAB) 522
TYPE= option, TRANTAB statement 504
UNICODELEN function 307
UNICODEWIDTH function 308
unique international monetary representation 59
unique national monetary representation 59
United Arab Emirates monetary format 510
United Kingdom monetary format 72
United States monetary format 458
UNIX encoding values 563
SUPARENEw. format 225
SUPARENEw. informat 432
SUPARENPw. informat 433
SUPARENw. format 226
SUPARENw. informat 441
UPLOAD procedure TRANTAB statement 503
uploading files translation tables 503
uppercase characters changing to lowercase converting arguments to 551
UTF-16 8
UTF-32 8
UTF-8 8
SUTF8Xw. format 228
SUTF8Xw. informat 444
VARIABLES variables associating formats with 199
associating informats with 199
labels 199
length, associating with 199
transcode attributes of 300
transcoding enabled for specified character variable variant characters 15
VARTRANSCODE function 300
VERBOSE option
PROC DBCSTAB statement 510
VERIFY option
PROC DBCSTAB statement 510
visual server
storing logical-ordered text on 84, 85
VSLOGRw. format 230
VSLOGRw. informat 446
VSLOGw. format 229
VSLOGw. informat 445
VTRANSCODE function 301
VTRANSCODEX function 302
WEEKUw. format 232
WEEKVw. format 233
SUNCRw. format 223
SUNCRw. informat 438
Unicode
BOM prefix on external files 555
encoding values for transcoding data length of character unit length of display unit 607
Unicode Consortium 15
UNICODELEN function 307
UNICODEWIDTH function 308
unique international monetary representation 59
unique national monetary representation 59
United Arab Emirates monetary format 510
United Kingdom monetary format 72
United States monetary format 458
UNIX encoding values 563
SUPARENEw. format 225
SUPARENEw. informat 432
SUPARENPw. informat 433
SUPARENw. format 226
SUPARENw. informat 441
UPLOAD procedure TRANTAB statement 503
uploading files translation tables 503
uppercase characters changing to lowercase converting arguments to 551
UTF-16 8
UTF-32 8
UTF-8 8
SUTF8Xw. format 228
SUTF8Xw. informat 444
W
WEEKUw. format 232
WEEKVw. format 233
<table>
<thead>
<tr>
<th>Format</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEEKWw. format</td>
<td>235</td>
</tr>
<tr>
<td>YENw.d format</td>
<td>448</td>
</tr>
<tr>
<td>YYWEEKUw. format</td>
<td>537</td>
</tr>
<tr>
<td>YYWEEKVw. format</td>
<td>538</td>
</tr>
<tr>
<td>YYWEEKWw. format</td>
<td>539</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Encoding Values</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>562</td>
</tr>
<tr>
<td>ISO encodings</td>
<td>1</td>
</tr>
<tr>
<td>Latin1 code page</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ZOS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>encoding support</td>
<td>53</td>
</tr>
<tr>
<td>encoding values</td>
<td>564</td>
</tr>
</tbody>
</table>
Your Turn

We welcome your feedback.

☐ If you have comments about this book, please send them to yourturn@sas.com. Include the full title and page numbers (if applicable).

☐ If you have comments about the software, please send them to suggest@sas.com.
Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at support.sas.com/bookstore.

SAS® Press
Need to learn the basics? Struggling with a programming problem? You’ll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels. support.sas.com/saspress

SAS® Documentation
To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – free on the Web.
- Hard-copy books.

support.sas.com/publishing

SAS® Publishing News
Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site. support.sas.com/spn