A format converts a character or numeric value to a text string, as when printing. The use and syntax of formats parallels that of informats. However, the decimal argument is more important for formats than for informats. For most numeric formats, it determines the number of decimal places that are produced.

To create an expression from a format, use the PUT function. To create formats for a specific purpose, use the FORMAT proc.

Character Formats

Character formats use an optional width argument, shown as \( w \), which can be between 1 and 32767. Character formats left-align when the output they produce is shorter than the width of the format.

$ASCII$

$ASCIIw$. Writes ASCII text.

$BIDI$

$BIDIw$. Writes a visually oriented string. \( \in \ 9^+ \)

$BINARY$

$BINARYw$. Character binary: converts each character to 8 output binary digit characters.

$CHAR$

$CHARw$. Writes a character value unchanged. Same as the standard character format.

$EBCDIC$

$EBCDICw$. Writes EBCDIC text.

$F$

$w$. Standard character format. Writes a character value unchanged.

Alias $ A width argument is required when using this alias.
$HEX
$HEXw. Character hexadecimal: writes each character as two hexadecimal digits.

$KANJI
$KANJIw. Adds shift codes to DBCS data. \( \in 7+ \)

$KANJI
$KANJIw. Removes shift codes from DBCS data. \( \in 7+ \)

$MSGCASE
$MSGCASEw. Converts letters to uppercase if the system option MSGCASE is on.

$OCTAL
$OCTALw. Character octal: writes each character as 3 octal digits.

$QUOTE
$QUOTEw. \( w \geq 3 \) Writes string surrounded by double quotes (").

$REVERJ
$REVERJw. Reversed; right to left.

$REVERS
$REVERSw. Reversed, with leading spaces removed.

$UCS2B
$UCS2Bw. Writes big-endian 16-bit UCS2 Unicode without a byte-order mark. \( \in 8.2+ \)

$UCS2L
$UCS2Lw. Writes little-endian 16-bit UCS2 Unicode without a byte-order mark. \( \in 8.2+ \)

$UCS2X
$UCS2Xw. Writes 16-bit UCS2 Unicode. \( \in 8.2+ \)

$UPCASE
$UPCASEw. Converts lowercase letters to uppercase.

$UTF8X
$UTF8Xw. Writes 8-bit UCS2 Unicode. \( \in 8.2+ \)

$VARYING
$VARYINGw. length variable Varying-length. Use only in the PUT statement.
Numeric Formats

Numeric formats use an optional width argument, $w$, which can range from 1 to 32. Most also use an optional decimal argument, $d$, which can range from 0 to 31. For formats that print decimal points, $d$ has to be less than $w$. In binary formats, which cannot contain a decimal point, the $d$ argument tells the format to multiply the number by that power of 10 before formatting. In these binary formats, $d$ can be no more than 31, but it does not have to be less than $w$. Some binary formats can write only positive values; if you use such a format to write a negative value, it writes it as positive. Many formats can write only integer values; they truncate fractional values. Most numeric formats right-align when they produce output that is shorter than the width of the format, but formats that produce words or output that begins with a letter usually left-align.

**BEST**

**BEST**$w$. Writes a number as precisely as possible in the width.

**BINARY**

**BINARY**$w$. $w \leq 64$ Writes binary integers.

**COMMA**

**COMMA**$w.d$ $w \geq 2$ Commas separate every three digits.

**COMMAX**

**COMMAX**$w.d$ $w \geq 2$ The same as COMMA, but with periods and commas interchanged.

**D**

**D**$w.s$ $s < w$, $s \leq 16$ Writes numbers with at least $s$ significant digits using, within certain ranges, the same number of decimal places.

**DOLLAR**

**DOLLAR**$w.d$ $w \geq 2$ The same as COMMA, but preceded by a dollar sign.

**DOLLARX**

**DOLLARX**$w.d$ $w \geq 2$ The same as DOLLAR, but with periods and commas interchanged.

**E**

**E**$w$. $w \geq 7$ Scientific (exponential) notation using E.

**Example**

$1.25E+10$

**EUR...**

**EUR**$currencyclw.d$ $w \geq 2$ Converts currencies to euros.

**EUR**$currencyclw.d$ $w \geq 2$ Converts euros to other currencies. See the EUROCURR function for the three-letter currency codes.

**EURO**$w.d$ $w \geq 2$ Similar to DOLLAR, but with a euro symbol.

**EURO**$w.d$ $w \geq 2$ Similar to DOLLARX, but with a euro symbol.

$\in 8.1+$
F
F_{w.d} \quad w.d \quad \text{Standard numeric format. Writes a decimal point and a fixed number of decimal places if } d > 0.

FLOAT
FLOAT_{w.d} \quad w = 4 \quad \text{Single-precision floating point.}

FRACT
FRACT_{w.} \quad w \geq 4 \quad \text{Writes fractions in reduced form.}

HEX
HEX_{w.} \quad w < 16 \quad \text{Writes hexadecimal integers.}
HEX_{w.} \quad w = 16 \quad \text{Writes the hexadecimal representation of an 8-byte floating point value.}

IB
IB_{w.d} \quad w \leq 8 \quad \text{Signed integer binary.}

IBR
IBR_{w.d} \quad w \leq 8 \quad \text{Signed integer binary with little-endian byte ordering.} \in 7+

IEEE
IEEE_{w.d} \quad 3 \leq w \leq 4 \quad \text{IEEE single-precision floating point.}
IEEE_{w.d} \quad 5 \leq w \leq 8 \quad \text{IEEE double-precision floating point.}

MRB
MRB_{w.d} \quad 2 \leq w \leq 8 \quad \text{Microsoft real binary (floating point).}

NEGPAREN
NEGPAREN_{w.d} \quad d \leq 2 \quad \text{Commas separate every three digits; negative numbers are in parentheses.}

NUMX
NUMX_{w.d} \quad \text{The same as the standard numeric format, but with the decimal point written as a comma.}

OCTAL
OCTAL_{w.} \quad w \leq 24 \quad \text{Writes octal integers.}

PD
PD_{w.d} \quad w \leq 16 \quad \text{Packed decimal.}

PERCENT
PERCENT_{w.d} \quad w \geq 4, d \leq 2 \quad \text{Writes numbers as percents, followed by a percent sign, with negative values enclosed in parentheses.}

PIB
PIB_{w.d} \quad w \leq 8 \quad \text{Unsigned (positive) integer binary.}

PIBR
PIBR_{w.d} \quad w \leq 8 \quad \text{Unsigned (positive) integer binary with little-endian byte ordering.} \in 7+

PK
PK_{w.d} \quad w \leq 16 \quad \text{Unsigned packed decimal.}
RB
RB_{w.d} \quad 2 \leq w \leq 8, \quad d \leq 10 \quad \text{Real binary; floating point.}

ROMAN
ROMAN_{w} \quad w \geq 2 \quad \text{Writes Roman numerals using capital letters.}

S370F...
Formats for compatibility with native IBM mainframe (System/370) data formats.
S370FF_{w.d} \quad \text{EBCDIC numeric.} \quad \in {7+}
S370FHEX_{w.d} \quad \text{EBCDIC hexadecimal.} \quad \in {8.2+}
S370FIB_{w.d} \quad w \leq 8 \quad \text{Signed integer binary with big-endian byte ordering.}
S370FIBU_{w.d} \quad w \leq 8 \quad \text{Unsigned integer binary (absolute value) with big-endian byte ordering.}
S370FPD_{w.d} \quad w \leq 16 \quad \text{IBM mainframe packed decimal.}
S370FPDU_{w.d} \quad w \leq 16 \quad \text{IBM mainframe unsigned packed decimal (absolute value).}
S370FPIB_{w.d} \quad w \leq 8 \quad \text{Unsigned (positive) integer binary with big-endian byte ordering. Writes negative values as 'FF'X.}
S370FRB_{w.d} \quad 2 \leq w \leq 8 \quad \text{IBM mainframe real binary.}
S370FZD_{w.d} \quad \text{IBM mainframe zoned decimal.}
S370FZDL_{w.d} \quad \text{IBM mainframe zoned decimal leading sign: EBCDIC zoned decimal with the sign bit in the first byte.}
S370FZDS_{w.d} \quad w \geq 2 \quad \text{IBM mainframe zoned decimal separate leading sign: EBCDIC numeric digits with the first character blank or a minus sign.}
S370FZDT_{w.d} \quad w \geq 2 \quad \text{IBM mainframe zoned decimal separate trailing sign: EBCDIC numeric digits with the last character blank or a minus sign.}
S370FZDU_{w.d} \quad \text{IBM System/370 unsigned zoned decimal. The same as S370FZD, but writes the absolute value.}

SSN
SSN_{w} \quad w = 11 \quad \text{Nine-digit number with hyphens after the third and fifth digits.}

WORDF
WORDF_{w} \quad 5 \leq w \leq 32767 \quad \text{Writes number in words, with hundredths written as a fraction.}

WORDS
WORDS_{w} \quad 5 \leq w \leq 32767 \quad \text{Writes number in words, with hundredths written in words.}

YEN
YEN_{w.d} \quad w \geq 2, \quad d: 0, 2 \quad \text{The same as COMMA, but preceded by a yen sign.}

Z
Z_{w.d} \quad \text{Writes number with leading zeros.}

ZD
ZD_{w.d} \quad d \leq 10 \quad \text{Zoned decimal.}
Time Formats

Time formats are numeric formats that write SAS date, SAS time, and SAS datetime values. Widths up to 32 are often allowed, but the useful widths are shown in the entries. Instead of detailed descriptions, examples show the output that each format produces.

Formats marked ✈ have international equivalents. See the entry for ...DF... for details. The symbol ✈ in a format name indicates an optional suffix to change the symbol that the format writes as a delimiter. Symbol suffixes are:

- B: space (blank) ∈ 7+
- C: colon
- D: hyphen (dash)
- N: none (the format width is smaller)
- P: period
- S: slash

...DF...

International date and datetime formats. The prefix of the format name indicates the language. The suffix identifies the equivalent format. *∈ 7+

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Language</th>
<th>Prefix</th>
<th>Language</th>
<th>Prefix</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFR</td>
<td>Afrikaans</td>
<td>ESP</td>
<td>Spanish</td>
<td>NOR</td>
<td>Norwegian</td>
</tr>
<tr>
<td>CAT</td>
<td>Catalan</td>
<td>FIN</td>
<td>Finnish</td>
<td>POL</td>
<td>Polish</td>
</tr>
<tr>
<td>CRO</td>
<td>Croatian</td>
<td>FRA</td>
<td>French</td>
<td>PTG</td>
<td>Portuguese</td>
</tr>
<tr>
<td>CSY</td>
<td>Czech</td>
<td>FRS</td>
<td>Swiss_French</td>
<td>RUS</td>
<td>Russian</td>
</tr>
<tr>
<td>DES</td>
<td>Swiss_German</td>
<td>ITA</td>
<td>Italian</td>
<td>SVE</td>
<td>Swedish</td>
</tr>
<tr>
<td>DEU</td>
<td>German</td>
<td>MAC</td>
<td>Macedonian</td>
<td>EUR</td>
<td>Selected in the DFLANG= system option</td>
</tr>
<tr>
<td>ENG</td>
<td>English</td>
<td>NLD</td>
<td>Dutch</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Equivalent</th>
<th>Suffix</th>
<th>Equivalent</th>
<th>Suffix</th>
<th>Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>DD</td>
<td>DDMMYY</td>
<td>DT</td>
<td>DATETIME</td>
<td>MY</td>
<td>MONYY</td>
</tr>
<tr>
<td>DE</td>
<td>DATE</td>
<td>DWN</td>
<td>DOWNAME</td>
<td>WDX</td>
<td>WORDDATX</td>
</tr>
<tr>
<td>DN</td>
<td>WEEKDAY</td>
<td>MN</td>
<td>MONNAME</td>
<td>WKX</td>
<td>WEEKDATX</td>
</tr>
</tbody>
</table>

Example

FINDFMN is the Finnish-language equivalent to the MONNAME format.

DATE

DATEw. w: 5, 7, 9  Writes a SAS date value: 06SEP   06SEP93 06SEP1993 ✈

DATEAMPM

DATEAMPMw.d w: 7, 10, 13, 16, 19, d + 20  Writes a SAS datetime value using a 12-hour clock (for w ≥ 13): 01JAN60 01JAN60:10 01JAN60:10 AM 01JAN60:10:00 AM 01JAN60:10:00:08 AM
DATETIME
DATETIMEw.d  \( w: 7, 10, 13, 16, d + 17 \)  Writes a SAS datetime value: 06SEP93 06SEP93:14 06SEP93:14:03 06SEP93:14:03:17

DAY
DAYw.  \( w = 2 \)  Writes the day of the month of a SAS date value.

DDMMYY+

DDMMYYw.  \( w: 2, 4, 5, 6, 8, 10 \)  Writes a SAS date value: 06 0609 06/09 06/09/93 06/09/1993 \( \in 7^+ \)
DDMMYYBw.  \( w: 2, 4, 5, 6, 8, 10 \)  W 06 06 09 93 06 09 1993 \( \in 7^+ \)
DDMMYYCw.  \( w: 2, 4, 5, 6, 8, 10 \) 06:09 06:09:93 06:09:1993 \( \in 7^+ \)
DDMMYYDw.  \( w: 2, 4, 5, 6, 8, 10 \) 06-09 06-09-93 06-09-1993 \( \in 7^+ \)
DDMMYYNw.  \( w: 2, 4, 5, 8 \) 0609 060993 06091993 \( \in 7^+ \)
DDMMYYYYw.  \( w: 2, 4, 5, 6, 8, 10 \) 06.09 06.09.93 06.09.1993 \( \in 7^+ \)

DOWNAME
DOWNAMEw.  \( w: 1, 2, 3, 9 \)  Writes the name of the day of the week of a SAS date value. 

DT...

DTDATEw.  \( w: 5, 7, 9 \)  Writes the date of a SAS datetime value: 06SEP 06SEP93 06SEP1993 \( \in 8.1^+ \)
DTMONEYw.  \( w: 5, 7 \)  Writes the month and year of a SAS datetime value: APR11 APR2011 \( \in 8.2^+ \)
DWTWDATEw.  Writes the date of a SAS datetime value, similar to the WEEKDATE format. \( \in 8.2^+ \)
DTYEARw.  \( w: 2, 4 \)  Writes the year of a SAS datetime value: 07 2007 \( \in 8.2^+ \)
DTYQw.  \( w: 4, 6 \)  Writes the year and quarter of a SAS datetime value: 00:4 2000:4 \( \in 8.2^+ \)

HHMM

HHMMw.d  \( w: 2, 4, 5, d + 6 \)  Writes a SAS time value: 15 1547 15:47

HOUR

HOURw.d  \( 2 \leq w \leq 20, d < w - 2 \)  Writes the hour of a SAS time value.

JULDAY

JULDAYw.  \( w = 3 \)  Writes the day of the year of a SAS date value.

JULIAN

JULIANw.  \( w: 5, 7 \)  Writes the year and day of the year of a SAS date value.

MINGGUO

MINGGUOw.  \( w: 6–7, 8–10 \)  Writes a SAS date value as a Taiwan-era date (year 1 = 1912): 840922 0084/09/22
MMDDYY+

\[ \text{MMDDYYw. } w = 2, 4, 5, 6, 8, 10 \quad \text{Writes a SAS date value: } 08\ 08/27 \ 082799 \ 08/27/99 \ 08/27/1999 \in 7^+ \]

\[ \text{MMDDYYBw. } w = 2, 4, 5, 6, 8, 10 \quad \text{08 27 08 27 99 08 27 1999 } \in 7^+ \]

\[ \text{MMDDYYCw. } w = 2, 4, 5, 6, 8, 10 \quad 08 27 082799 08 27 1999 \in 7^+ \]

\[ \text{MMDDYYDw. } w = 2, 4, 5, 6, 8, 10 \quad 08-27 08-27-99 08-27-1999 \in 7^+ \]

\[ \text{MMDDYYNW. } w = 2, 4, 6, 8 \quad 0827 082799 08271999 \in 7^+ \]

\[ \text{MMDDYYPW. } w = 2, 4, 5, 6, 8, 10 \quad 08.27 08.27.99 08.27.1999 \in 7^+ \]

\[ \text{MMDDYYSW. } w = 2, 4, 5, 6, 8, 10 \quad 08/27 08/27/99 08/27/1999 \in 7^+ \]

\[ \text{MMSSw.d } w = 2, 5, 7–20, d = w - 6 \quad \text{Writes a SAS time value or a number of seconds as minutes and seconds, or writes a number of minutes as hours and minutes: } 24:10 \]

\[ \text{MMYY+} \]

\[ \text{MMYYw. } w = 5, 7 \quad \text{Writes the month and year of a SAS date value: } 09M93 \ 09M1993 \]

\[ \text{MMYYCW. } w = 5, 7 \quad 08:99 \ 08:1999 \]

\[ \text{MMYYDW. } w = 5, 7 \quad 08-99 \ 08-1999 \]

\[ \text{MMYYNW. } w = 4, 6 \quad 0899 \ 081999 \]

\[ \text{MMYYPW. } w = 5, 7 \quad 08.99 \ 08.1999 \]

\[ \text{MMYYSW. } w = 5, 7 \quad 08/99 \ 08/1999 \]

\[ \text{MONNAME} \]

\[ \text{MONNAMEw. } w = 1, 3, 9 \quad \text{Writes the name of the month of a SAS date value: September} \rightarrow \]

\[ \text{MONTH} \]

\[ \text{MONTHw. } w = 2 \quad \text{Writes the month number of a SAS date value.} \]

\[ \text{MONYY} \]

\[ \text{MONYYw. } w = 5, 7 \quad \text{Writes the month and year of a SAS date value: APR11 APR2011 } \rightarrow \]

\[ \text{NENGO} \]

\[ \text{NENGOw. } 2 \leq w \leq 10 \quad \text{Writes the Japanese era (M, T, S, or H), year, month, and day of a SAS date value.} \]

\[ \text{QTR} \]

\[ \text{QTRw. } w = 1 \quad \text{Writes the quarter number of a SAS date value.} \]

\[ \text{QTRR} \]

\[ \text{QTRRW. } w = 3 \quad \text{Writes the quarter number of a SAS date value as a Roman numeral.} \]

\[ \text{TIMEw.d } w = 2, 5, 8, 10–20, d = w - 9 \quad \text{Writes a SAS time value as hours and optional minutes, seconds, and fractional seconds: 05 05:15 05:15:00 05:15:00.00} \]
FORMATS

TIMEAMPM
TIMEAMPMw.d  w: 2, 5, 8, 11, 13–20, d = w – 12  Writes a SAS time value or the time of day of a SAS datetime value using a 12-hour clock: AM 10 AM 10:00 AM 10:00:08 AM

TOD
TODw.d  w: 2, 5, 8, 10–20, d = w – 9  Writes the time of day of a SAS datetime value or a SAS time value as hours and optional minutes, seconds, and fractional seconds (the same way the TIME format writes SAS time values).

WEEKDATE
WEEKDATEw.  w: 3, 9, 15, 17, 23, 29  Writes a SAS date value:
Mon  Monday  Mon, Oct 12, 92  Mon, Oct 12, 1992
Monday, Oct 12, 1992  Monday, October 12, 1992

WEEKDATX
WEEKDATXw.  w: 3, 9, 15, 17, 23, 29  Writes a SAS date value: Mon
Monday  Mon, 12 Oct 92  Mon, 12 Oct 1992
Monday, 12 Oct 1992  Monday, 12 October 1992

WEEKDAY
WEEKDAYw.  w = 1  Writes the number of the day of the week of a SAS date value, with Sunday=1, Saturday=7.

WORDDATE
WORDDATEw.  w: 3, 9, 12, 18  Writes a SAS date value: Oct  October
Oct 12, 1992  October 12, 1992

WORDDATX
WORDDATXw.  w: 3, 9, 12, 18  Writes a SAS date value: Oct  October
12 Oct 1992  12 October 1992

YEAR
YEARw.  w: 2, 4  Writes the year of a SAS date value: 07  2007

YYMM+
YYMMw.  w: 5, 7  Writes the year and month of a SAS date value:
93M09  1993M09
YYMMCw.  w: 5, 7 93:09  1993:09
YYMMDw.  w: 5, 7 93–09  1993–09
YYMMNW.  w: 4, 6 9309  199309
YYMMPw.  w: 5, 7 93.09  1993.09
YYMMSw.  w: 5, 7 93/09  1993/09

YYMMD+D+
YYMMDw.  w: 2, 4, 5, 6, 8, 10  Writes the year, month, and day of a SAS date value: 99  9908  99–08
1999–08–27
YYMMDDBw.  w: 2, 4, 5, 6, 8, 10 99 08  99 08 27
1999 08 27 ∈7+
YYMMDDCw.  w: 2, 4, 5, 6, 8, 10 99:08  99:08:27  1999:08:27 ∈7+
YYMMDDDDw.  w: 2, 4, 5, 6, 8, 10 99–08  99–08–27
1999–08–27 ∈7+
YYMMDDNw.  $w$: 2, 4, 6, 8  990827  99082799  19990827  $\in 7+$
YYMMDDPw.  $w$: 2, 4, 5, 6, 8, 10  99.08  99.08.27  1999.08.27  $\in 7+$
YYMMDDSw.  $w$: 2, 4, 5, 6, 8, 10  99/08  99/08/27  1999/08/27  $\in 7+$

YYMON
YYMONw.  $w$: 5, 7  Writes the year and month of a SAS date value: 93SEP  1993SEP

YYQ+
YYQw.  $w$: 4, 6  Writes the year and quarter of a SAS date value: 00Q4  2000Q4
YYQCw.  $w$: 4, 6  00:4  2000:4
YYQDw.  $w$: 4, 6  00-4  2000-4
YYQNW.  $w$: 3, 5  004  2004
YYQPw.  $w$: 4, 6  00.4  2000.4
YYQSw.  $w$: 4, 6  00/4  2000/4

YYQR+
YYQRw.  $w$: 6, 8  Writes the year and quarter of a SAS date value: 00QIV  2000QIV
YYQRCw.  $w$: 6, 8  00:IV  2000:IV
YYQRDw.  $w$: 6, 8  00-IIV  2000-IIV
YYQRNw.  $w$: 5, 7  00IV  2000IV
YYQRPW.  $w$: 6, 8  00.IIV  2000.IIV
YYQRSw.  $w$: 6, 8  00/IIV  2000/IIV