15

Formats

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

`$ASCII`*w.* Writes ASCII text.

$BIDI

`$BIDI`*w.* Writes a visually oriented string. ∈9+

$BINARY

`$BINARY`*w.* Character binary: converts each character to 8 output binary digit characters.

$CHAR

`$CHAR`*w.* Writes a character value unchanged. Same as the standard character format.

$EBCDIC

`$EBCDIC`*w.* Writes EBCDIC text.

$F

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

Alias $ A width argument is required when using this alias.

$HEX

`$HEX`*w.* Character hexadecimal: writes each character as two hexadecimal digits.

$KANJI

`$KANJI`*w.* Adds shift codes to DBCS data. ∈ 7+

$KANJIX

`$KANJIX`*w.* Removes shift codes from DBCS data. ∈ 7+

$MSGCASE

`$MSGCASE`*w.* Converts letters to uppercase if the system option MSGCASE is on.

$OCTAL

`$OCTAL`*w.* Character octal: writes each character as 3 octal digits.

$QUOTE

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

$REVERJ

`$REVERJ`*w.* Reversed; right to left.

$REVERS

`$REVERS`*w.* Reversed, with leading spaces removed.

$UCS2B

`$UCS2B`*w.* Writes big-endian 16-bit UCS2 Unicode without a byte-order mark. ∈ 8.2+

$UCS2L

`$UCS2L`*w.* Writes little-endian 16-bit UCS2 Unicode without a byte-order mark. ∈ 8.2+

$UCS2X

`$UCS2X`*w.* Writes 16-bit UCS2 Unicode. ∈ 8.2+

$UPCASE

`$UPCASE`*w.* Converts lowercase letters to uppercase.

$UTF8X

`$UTF8X`*w.* Writes 8-bit UCS2 Unicode. ∈ 8.2+

$VARYING

`$VARYING`*w. 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.* $w \le 64$ Writes binary integers.

COMMA

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

COMMAX

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

D

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

DOLLAR

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

DOLLARX

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

E

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

Example

```
1.25E+10
```

EUR...

`EURFR`*currency*`w.d` $w \ge 2$ Converts currencies to euros.

`EURTO`*currency*`w.d` $w \ge 2$ Converts euros to other currencies. See the EUROCURR function for the three-letter currency codes.

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

`EUROX`*w.d* $w \ge 2$ Similar to DOLLARX, but with a euro symbol.

∈ 8.1+

F

F*w.d* *w.d* Standard numeric format. Writes a decimal point and a fixed number of decimal places if $d > 0$.

FLOAT

FLOAT*w.d* $w = 4$ Single-precision floating point.

FRACT

FRACT*w.* $w \geq 4$ Writes fractions in reduced form.

HEX

HEX*w.* $w < 16$ Writes hexadecimal integers.

HEX*w.* $w = 16$ Writes the hexadecimal representation of an 8-byte floating point value.

IB

IB*w.d* $w \leq 8$ Signed integer binary.

IBR

IBR*w.d* $w \leq 8$ Signed integer binary with little-endian byte ordering. ∈ 7+

IEEE

IEEE*w.d* $3 \leq w \leq 4$ IEEE single-precision floating point.

IEEE*w.d* $5 \leq w \leq 8$ IEEE double-precision floating point.

MRB

MRB*w.d* $2 \leq w \leq 8$ Microsoft real binary (floating point).

NEGPAREN

NEGPAREN*w.d* $d \leq 2$ Commas separate every three digits; negative numbers are in parentheses.

NUMX

NUMX*w.d* The same as the standard numeric format, but with the decimal point written as a comma.

OCTAL

OCTAL*w.* $w \leq 24$ Writes octal integers.

PD

PD*w.d* $w \leq 16$ Packed decimal.

PERCENT

PERCENT*w.d* $w \geq 4$, $d \leq 2$ Writes numbers as percents, followed by a percent sign, with negative values enclosed in parentheses.

PIB

PIB*w.d* $w \leq 8$ Unsigned (positive) integer binary.

PIBR

PIBR*w.d* $w \leq 8$ Unsigned (positive) integer binary with little-endian byte ordering. ∈ 7+

PK

PK*w.d* $w \leq 16$ Unsigned packed decimal.

RB

RB*w.d* $2 \le w \le 8, d \le 10$ Real binary; floating point.

ROMAN

ROMAN*w.* $w \ge 2$ Writes Roman numerals using capital letters.

S370F...

Formats for compatibility with native IBM mainframe (System/370) data formats.

S370FF*w.d* EBCDIC numeric. ∈ 7+

S370FHEX*w.d* EBCDIC hexadecimal. ∈ 8.2+

S370FIB*w.d* $w \le 8$ Signed integer binary with big-endian byte ordering.

S370FIBU*w.d* $w \le 8$ Unsigned integer binary (absolute value) with big-endian byte ordering.

S370FPD*w.d* $w \le 16$ IBM mainframe packed decimal.

S370FPDU*w.d* $w \le 16$ IBM mainframe unsigned packed decimal (absolute value).

S370FPIB*w.d* $w \le 8$ Unsigned (positive) integer binary with big-endian byte ordering. Writes negative values as 'FF'X.

S370FRB*w.d* $2 \le w \le 8$ IBM mainframe real binary.

S370FZD*w.d* IBM mainframe zoned decimal.

S370FZDL*w.d* IBM mainframe zoned decimal leading sign: EBCDIC zoned decimal with the sign bit in the first byte.

S370FZDS*w.d* $w \ge 2$ IBM mainframe zoned decimal separate leading sign: EBCDIC numeric digits with the first character blank or a minus sign.

S370FZDT*w.d* $w \ge 2$ IBM mainframe zoned decimal separate trailing sign: EBCDIC numeric digits with the last character blank or a minus sign.

S370FZDU*w.d* IBM System/370 unsigned zoned decimal. The same as S370FZD, but writes the absolute value.

SSN

SSN*w.* $w = 11$ Nine-digit number with hyphens after the third and fifth digits.

WORDF

WORDF*w.* $5 \le w \le 32767$ Writes number in words, with hundredths written as a fraction.

WORDS

WORDS*w.* $5 \le w \le 32767$ Writes number in words, with hundredths written in words.

YEN

YEN*w.d* $w \ge 2$, d: 0, 2 The same as COMMA, but preceded by a yen sign.

Z

Z*w.d* Writes number with leading zeros.

ZD

ZD*w.d* $d \le 10$ 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+

Prefix	Language	Prefix	Language	Prefix	Language
AFR	Afrikaans*	ESP	Spanish	NOR	Norwegian
CAT	Catalan*	FIN	Finnish	POL	Polish*
CRO	Croatian*	FRA	French	PTG	Portuguese
CSY	Czech*	FRS	Swiss_French	RUS	Russian*
DAN	Danish	HUN	Hungarian*	SLO	Slovenian*
DES	Swiss_German	ITA	Italian	SVE	Swedish
DEU	German	MAC	Macedonian*	EUR	Selected in the DFLANG= system option
ENG	English	NLD	Dutch		

Suffix	Equivalent	Suffix	Equivalent	Suffix	Equivalent
DD	DDMMYY	DT	DATETIME	MY	MONYY
DE	DATE	DWN	DOWNAME	WDX	WORDDATX
DN	WEEKDAY	MN	MONNAME	WKX	WEEKDATX

Example

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

DATE

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

DATEAMPM

DATEAMPM*w.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

DATETIME*w.d* w: 7, 10, 13, 16, $d + 17$ Writes a SAS datetime value: 06SEP93 06SEP93:14 06SEP93:14:03 06SEP93:14:03:17 ✈

DAY

DAY*w.* $w = 2$ Writes the day of the month of a SAS date value.

DDMMYY+

DDMMYY*w.* w: 2, 4, 5, 6, 8, 10 Writes a SAS date value: 06 0609 06/09 060993 06/09/93 06/09/1993 ✈

DDMMYYB*w.* w: 2, 4, 5, 6, 8, 10 06 09 06 09 93 06 09 1993 ∈ 7+

DDMMYYC*w.* w: 2, 4, 5, 6, 8, 10 06:09 06:09:93 06:09:1993 ∈ 7+

DDMMYYD*w.* w: 2, 4, 5, 6, 8, 10 06–09 06–09–93 06–09–1993 ∈ 7+

DDMMYYN*w.* w: 2, 4, 6, 8 0609 060993 06091993 ∈ 7+

DDMMYYP*w.* w: 2, 4, 5, 6, 8, 10 06.09 06.09.93 06.09.1993 ∈ 7+

DDMMYYS*w.* w: 2, 4, 5, 6, 8, 10 06/09 06/09/93 06/09/1993 ∈ 7+

DOWNAME

DOWNAME*w.* w: 1, 2, 3, 9 Writes the name of the day of the week of a SAS date value. ✈

DT*...*

DTDATE*w.* w: 5, 7, 9 Writes the date of a SAS datetime value: 06SEP 06SEP93 06SEP1993 ∈ 8.1+

DTMONYY*w.* w: 5, 7 Writes the month and year of a SAS datetime value: APR11 APR2011 ∈ 8.2+

DTWKDATX*w.* Writes the date of a SAS datetime value, similar to the WEEKDATX format. ∈ 8.2+

DTYEAR*w.* w: 2, 4 Writes the year of a SAS datetime value: 07 2007 ∈ 8.2+

DTYYQC*w.* w: 4, 6 Writes the year and quarter of a SAS datetime value: 00:4 2000:4 ∈ 8.2+

HHMM

HHMM*w.d* w: 2, 4, 5, $d + 6$ Writes a SAS time value: 15 1547 15:47

HOUR

HOUR*w.d* $2 \le w \le 20$, $d < w - 2$ Writes the hour of a SAS time value.

JULDAY

JULDAY*w.* $w = 3$ Writes the day of the year of a SAS date value.

JULIAN

JULIAN*w.* w: 5, 7 Writes the year and day of the year of a SAS date value.

MINGUO

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

MMDDYY+

MMDDYY*w.* $w = 2, 4, 5, 6, 8, 10$ Writes a SAS date value: 08 0827 08/27 082799 08/27/99 08/27/1999

MMDDYYB*w.* w: 2, 4, 5, 6, 8, 10 08 27 08 27 99 08 27 1999 ∈ 7+

MMDDYYC*w.* w: 2, 4, 5, 6, 8, 10 08:27 08:27:99 08:27:1999 ∈ 7+

MMDDYYD*w.* w: 2, 4, 5, 6, 8, 10 08–27 08–27–99 08–27–1999 ∈ 7+

MMDDYYN*w.* w: 2, 4, 6, 8 0827 082799 08271999 ∈ 7+

MMDDYYP*w.* w: 2, 4, 5, 6, 8, 10 08.27 08.27.99 08.27.1999 ∈ 7+

MMDDYYS*w.* w: 2, 4, 5, 6, 8, 10 08/27 08/27/99 08/27/1999 ∈ 7+

MMSS

MMSS*w.d* w: 2, 5, 7–20, $d = w - 6$ 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

MMYY+

MMYY*w.* w: 5, 7 Writes the month and year of a SAS date value: 09M93 09M1993

MMYYC*w.* w: 5, 7 08:99 08:1999

MMYYD*w.* w: 5, 7 08–99 08–1999

MMYYN*w.* w: 4, 6 0899 081999

MMYYP*w.* w: 5, 7 08.99 08.1999

MMYYS*w.* w: 5, 7 08/99 08/1999

MONNAME

MONNAME*w.* w: 1, 3, 9 Writes the name of the month of a SAS date value: September ✈

MONTH

MONTH*w.* $w = 2$ Writes the month number of a SAS date value.

MONYY

MONYY*w.* w: 5, 7 Writes the month and year of a SAS date value: APR11 APR2011 ✈

NENGO

NENGO*w.* $2 \le w \le 10$ Writes the Japanese era (M, T, S, or H), year, month, and day of a SAS date value.

QTR

QTR*w.* $w = 1$ Writes the quarter number of a SAS date value.

QTRR

QTRR*w.* $w = 3$ Writes the quarter number of a SAS date value as a Roman numeral.

TIME

TIME*w.d* w: 2, 5, 8, 10–20, $d = w - 9$ Writes a SAS time value as hours and optional minutes, seconds, and fractional seconds: 05 05:15 05:15:00 05:15:00.00

TIMEAMPM

TIMEAMPM*w.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

TOD*w.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

WEEKDATE*w.* *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

WEEKDATX*w.* *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

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

WORDDATE

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

WORDDATX

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

YEAR

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

YYMM+

YYMM*w.* *w*: 5, 7 Writes the year and month of a SAS date value: 93M09 1993M09

YYMMC*w.* *w*: 5, 7 93:09 1993:09

YYMMD*w.* *w*: 5, 7 93–09 1993–09

YYMMN*w.* *w*: 4, 6 9309 199309

YYMMP*w.* *w*: 5, 7 93.09 1993.09

YYMMS*w.* *w*: 5, 7 93/09 1993/09

YYMMDD+

YYMMDD*w.* *w*: 2, 4, 5, 6, 8, 10 Writes the year, month, and day of a SAS date value: 99 9908 99–08 990827 99–08–27 1999–08–27

YYMMDDB*w.* *w*: 2, 4, 5, 6, 8, 10 99 08 99 08 27 1999 08 27 ∈ 7+

YYMMDDC*w.* *w*: 2, 4, 5, 6, 8, 10 99:08 99:08:27 1999:08:27 ∈ 7+

YYMMDDD*w.* *w*: 2, 4, 5, 6, 8, 10 99–08 99–08–27 1999–08–27 ∈ 7+

YYMMDDN*w.* *w*: 2, 4, 6, 8 990827 99082799 19990827 ∈7+
YYMMDDP*w.* *w*: 2, 4, 5, 6, 8, 10 99.08 99.08.27 1999.08.27 ∈7+
YYMMDDS*w.* *w*: 2, 4, 5, 6, 8, 10 99/08 99/08/27 1999/08/27 ∈7+

YYMON

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

YYQ+

YYQ*w.* *w*: 4, 6 Writes the year and quarter of a SAS date value: 00Q4 2000Q4
YYQC*w.* *w*: 4, 6 00:4 2000:4
YYQD*w.* *w*: 4, 6 00-4 2000-4
YYQN*w.* *w*: 3, 5 004 20004
YYQP*w.* *w*: 4, 6 00.4 2000.4
YYQS*w.* *w*: 4, 6 00/4 2000/4

YYQR+

YYQR*w.* *w* = 6, 8 Writes the year and quarter of a SAS date value: 00QIV 2000QIV
YYQRC*w.* *w* = 6, 8 00:IV 2000:IV
YYQRD*w.* *w* = 6, 8 00-IV 2000-IV
YYQRN*w.* *w* = 5, 7 00IV 2000IV
YYQRP*w.* *w* = 6, 8 00.IV 2000.IV
YYQRS*w.* *w* = 6, 8 00/IV 2000/IV