

System Interfaces (continued)

functions

3

```

----- AOS/VS, VMS, and OpenVMS -----
TTCLOSE* (ch) return code from closing I/O channel ch
TTOPEN* (specifications,ch) rc from assigning I/O channel ch to a terminal
TTREAD* (ch,buf,<cs>) reads data from channel ch into buf for max size s
TTWRITE* (ch,buf,<cs>) writes data from buffer buf to channel ch for max size s
help methods are listed in OS/2 C&E for Release 6.10, p44
----- OS/2 only -----
OS2HELP*# (h,p1,p2) displays help information using help method h
----- WINDOWS only -----
MCIPISLP*#(s) wait s seconds for a multimedia device to become active
MCPIISTR*#(com) return information from invocation of MCI string command com
PEEK16*@ (addr,<len>) the contents of a numeric variable, returned by a 16-bit
DLL routine, stored at memory location addr for length len
PEEK16C*@ (addr,<len>) the contents of a character variable, returned by a
16-bit DLL routine, stored at memory location addr for length len
WINHELP*# (h,p1,p2) displays help information using help method h
help methods are listed in Windows C&E for Release 6.10, p54
----- WINDOWS and OS/2 -----
DMYTECHC* (str) character count in hex for packet str
DMYTECKS* CALL(str,icks,ccks) checksum value for packet str; icks is initial
checksum and ccks is calculated checksum
DMYTECWD* (str,rstr) number of tokens in packet str; if packet >200 characters,
str is the first 200, rstr is the remainder; else rstr = ''
DMYTERVC* (x) converts DataMyte 2-byte hex number x to an ascii number
WAKEUP* (t) t specifies a time when a data step begins execution; the return
value is the number of seconds slept

```

Product-specific functions

```

-----
inrvars = "independent normal random variables"
type[t] = "for a Type [t] doubling-sampling plan"
-----
GRAPH
note most of the GRAPH functions have varying parameter lists
depending on the value of the first argument; refer to 90NLDOC
GASK CALL(attr,parms...) current setting for attribute 'attr'
GDRAW (ge,parms...) return code from creating graphic element 'ge'
GINIT () return code from initializing DSGI (Data Step Graphics Interface)
GPRINT (code) displays message that corresponds to error code 'code'
GRAPH (tsk,parms...) return code from performing lib management task 'tsk'
GSET (attr,parms...) return code from setting graphic element attribute
'attr'
GTERM () return code from terminating DSGI (Data Step Graphics Interface)
----- IML -----
7MODULEIC (<cntl>,modn,arg1,...argn) character return value from execution of
external routine modn; invoked from within the IML procedure
7MODULEIN (<cntl>,modn,arg1,...argn) numeric return value from execution of
external routine modn; invoked from within the IML procedure
7MODULEI CALL(<cntl>,modn,arg1,...argn) executes external routine modn;
invoked from within the IML procedure
----- QC -----
AOQ2 (rep,N,a1,r1,a2,n1,n2,p) average outgoing quality type[B]
ASN2 (mode,a1,r1,a2,n1,n2,p) average sample number type[B]
ATI2 (N,a1,r1,a2,n1,n2,p) average total inspection type[B]
BAYESACT CALL(k,s,df,a1,...<an>,y1,...<yn>,b1,...<bn>,p) posterior probabili-
ties that observations are contaminated with a larger variance
C4 (n) expected value of the standard deviation of n inrvars
CUSUMARL (type,sd,h,k,<hs>) average run length of a one- or two-sided
cumulative sum control chart scheme
D2 (n) expected value of the sample range of n inrvars
D3 (n) standard deviation of the range of n inrvars
EWMAARL (sd,r,k) average run length for exponentially weighted moving average
PROBACC2 (a1,r1,a2,n1,n2,D,N) acceptance probability type[A]
PROBACC2 (a1,r1,a2,n1,n2,p) acceptance probability type[B]
PROBMED (n,x) probability that the sample median is less than or equal to x
for a sample of n inrvars
STDMED (n) standard deviation of the median of a normally distributed sample
with size n

```

```

-----
THE NEXT THREE GROUPS OF FUNCTIONS DESCRIBED REQUIRE A DIRECTORY-ID, FILE-ID,
OR DATASET-ID TOKEN OBTAINED FROM DOPEN, FOPEN | MOPEN, OR OPEN RESPECTIVELY
Unless otherwise specified, these functions return 0 if the specific operation
was successful and a nonzero value if the operation was not successful
-----

```

External File Functions

```

-----
DCLOSE& (dirid) closes directory dirid
9DCREATE (dirname,mdir) creates directory dirname under current working
directory or parent directory mdir; returns full pathname if
successful, null string if unsuccessful
DINFO& (dirid,itm) host-specific information itm concerning directory dirid
DNUM& (dirid) number of members in directory dirid
DOPEN& (fileref) opens directory fileref and returns a unique numeric
directory identifier, returns 0 if open unsuccessful
DOPTNAME& (dirid,n) host-specific info item number n about directory dirid
DOPTNUM& (dirid) number of information items available for directory dirid
DREAD& (dirid,n) name of the nth member in directory dirid
DROPNOTE& (fid,notid) deletes note marker notid from FNOTE [DROPNOTE can also
be used to delete a notid from NOTE--refer to "Data Step Functions"]
FAPPEND& (fid,<cc>) appends current record in the FDB to file fid with
optional carriage control character cc
FCLOSE& (fid) closes file fid
FCOL& (fid) current column position in the FDB for file fid
FDELETE& (fid) deletes file fid
FGET& (fid,var,<len>) copies data from the FDB into variable var for
optional length len
FINFO& (fid,itm) host-specific information itm concerning file fid
FNOTE& (fid) unique note identifier for the last record read from file fid
FOPEN& (filref,<mode>,<recl>,<recfm>) opens file filref for input or update
and returns a unique numeric file identifier or zero if open fails
FOPTNAME& (fid,n) host-specific information item number n about file fid
FOPTNUM& (fid) number of information items available for file fid
FPOINT& (fid,notid) positions the read pointer to notid (from the FNOTE
function) in file fid
FPOS& (fid,p) positions column pointer in the FDB to column p for file fid
FPUT& (fid,d) moves data d to the FDB of file fid starting at the FDB's
current column position; d can be text or a variable
FREAD& (fid) reads a record from file fid into the FDB
FREWIND& (fid) positions file pointer to the beginning of file fid
FRLEN& (fid) size of last record read or current record size of file fid
FSEP& (fid,sep) sets token delimiter(s) sep for file fid
FWRITE& (fid,<cc>) writes the current record in the FDB to file fid with
optional carriage control character cc
MOPEN& (dirid,mname,<mode>,<recl>,<recfm>) opens member mname in directory
dirid for input or update and returns a unique numeric file
identifier, returns 0 if open unsuccessful; optional record parms

```

Data Step Functions

```

-----
ATTRC& (dsid,attr) value of character attribute attr for data set dsid
ATTRN& (dsid,attr) value of numeric attribute attr for data set dsid
CLOSE& (dsid) closes data set dsid
CUROBS& (dsid) current observation number from data set dsid
DROPNOTE& (dsid,notid) deletes a note marker notid from NOTE [DROPNOTE can also
delete a notid from FNOTE--refer to "External File Functions"]
DSNAME& (dsid) data set name associated with dsid
FETCH& (dsid,<NOSET>) reads the next nondeleted observation from data set
dsid into the DDV
FETCHOBS& (dsid,obs,<opts>) reads observation number obs from data set
dsid into the DDV
NOTE& (dsid) unique note identifier for current obs of data set dsid
OPEN& (dsname,<mode>) opens data set dsname for input and returns a unique
numeric data set identifier, returns 0 if open unsuccessful
POINT& (dsid,notid) locates observation identified by notid (from the NOTE
function) in data set dsid
REWIND& (dsid) positions data set dsid back to beginning
SET& CALL(dsid) automatically sets values of data set variables or macro
variables after a READ; typically follows an OPEN function call

```

Variable Functions

functions

5

```

GETVARC& (dsid,n) character value of the nth variable in data set dsid
GETVARN& (dsid,n) numeric value of the nth variable in data set dsid
VARFMT& (dsid,n) format assigned to the nth variable in data set dsid
VARINFMT& (dsid,n) informat assigned to the nth variable in data set dsid
VARLABEL& (dsid,n) label assigned to the nth variable in data set dsid
VARLEN& (dsid,n) length of the nth variable in data set dsid
VARNAME& (dsid,n) name of the nth variable in data set dsid
VARNUM& (dsid,vname) position of variable vname in data set dsid, 0 if
the variable does not exist in the data set
VARTYPE& (dsid,n) type ('C'|'N') of the nth variable in data set dsid

```

SAS Variable Attributes

```

-----
w/v = "associated with the variable"
var_by_exp = "the variable defined by the expression"
w/var_by_exp = "associated with the variable defined by expression"

7VARRAY (var) 1 if variable var is an array, 0 if not
7VARRAYX (exp) 1 if var_by_exp exp is an array, 0 if not
7VFORMAT (var) format w/v var
7VFORMATD (var) decimal value of the format w/v var
7VFORMATDX (exp) decimal value of the format w/var_by_exp exp
7VFORMATN (var) name of the format w/v var
7VFORMATNX (exp) name of the format w/var_by_exp exp
7VFORMATW (var) width value of the format w/v var
7VFORMATWX (exp) width value of the format w/var_by_exp exp
7VFORMATX (exp) format w/var_by_exp exp
7VINARRAY (var) 1 if variable var is a member of an array, 0 if not
7VINARRAYX (exp) 1 if var_by_exp is a member of an array, 0 if not
7VINFORMAT (var) informat w/v var
7VINFORMATD (var) decimal value of the informat w/v var
7VINFORMATDX (exp) decimal value of the informat w/var_by_exp exp
7VINFORMATN (var) name of the informat w/v var
7VINFORMATNX (exp) name of the informat w/var_by_exp exp
7VINFORMATW (var) width value of the informat w/v var
7VINFORMATWX (exp) width value of the informat w/var_by_exp exp
7VINFORMATX (exp) informat w/var_by_exp exp
7VLABEL (var) label w/v var
7VLABELX (exp) label w/var_by_exp exp
7VLENGTH (var) compile-time size of the variable var
7VLENGTHX (exp) compile-time size of the var_by_exp exp
7VNAME (var) name of the variable var
VNAME CALL(var1,var2) assigns the name of variable var1 as the value of
variable var2
7VNAMEX (exp) name of the var_by_exp exp
9VNEXT CALL(var,type,len) returns name, type, and length of variable var
7VTYPE (var) type ('C'|'N') of the variable var
7VTYPEX (exp) type ('C'|'N') of the var_by_exp exp
9VVALUE (var) formatted value w/v var
9VVALUEX (exp) formatted value w/var_by_exp exp

State and Zipcode
-----
name[type] = "name <= 20 characters in [type] case"

FIPNAME (fcode) converts FIPS code to a state name[upper]
FIPNAMEL (fcode) converts FIPS code to a state name[mixed]
FIPSTATE (fcode) converts FIPS code to a two-char postal state code
STFIPS (pcode) converts two-char postal state code to a FIPS code
STNAME (pcode) converts two-char postal state code to a state name[upper]
STNAMEL (pcode) converts two-char postal state code to a state name[mixed]
9ZIPCITY (zcode) converts five-char zip code to a city name and a two-char
postal state code
ZIPFIPS (zcode) converts five-char zip code to a FIPS code
ZIPNAME (zcode) converts five-char zip code to a state name[upper]
ZIPNAMEL (zcode) converts five-char zip code to a state name[mixed]
ZIPSTATE (zcode) converts five-char zip code to a two-char postal state code

```

Character

functions

6

```

first [type] = "first position of any [type] in string str beginning
at position pos, default pos = 1; a negative value
of pos searches right to left"
parm lists (str,...<strn>) for the CAT* functions can be denoted
using character arrays (of aryl-aryn)

ANSI2OEM* CALL(str,dst,len) [WINDOWS] converts string str from ansi to oem for
length len and stores result in string dst
9ANYALNUM (str,<pos>) first alphanumeric character
9ANYALPHA (str,<pos>) first alphabetic character
9ANYCNTRL (str,<pos>) first control character
9ANYDIGIT (str,<pos>) first digit
9ANYFIRST (str,<pos>) first character valid as the beginning of a SAS variable
name
9ANYGRAPH (str,<pos>) first graphical character
9ANYLOWER (str,<pos>) first lowercase letter
9ANYNAME (str,<pos>) first character valid as part of a SAS variable name
9ANYPRINT (str,<pos>) first printable character
9ANYPUNCT (str,<pos>) first punctuation character
9ANYSPACE (str,<pos>) first white-space character
9ANYUPPER (str,<pos>) first uppercase letter
9ANYXDIGIT (str,<pos>) first hexadecimal character that represents a digit
ASCCEB* (str) [VMS] converts string str from ascii to ebcdic
BYTE (n) nth character in ascii or ebcdic collating sequence
9CAT (str,...<strn>) concatenates strings without removing leading or
trailing blanks
9CATS (str,...<strn>) concatenates strings and removes leading and
trailing blanks
9CATS CALL(result,str,...<strn>) assigns result the value of concatenating
strings str-strn along with removing leading and trailing blanks
9CATT (str,...<strn>) concatenates strings and removes trailing blanks only
9CATT CALL(result,str,...<strn>) assigns result the value of concatenating
strings str-strn along with removing trailing blanks only
9CATX (sep,str,...<strn>) concatenates strings, removes leading and
trailing blanks, and inserts separator sep
9CATX CALL(sep,result,str,...<strn>) assigns result the value of
concatenating strings, removing all leading and trailing blanks,
and inserting separator sep
9COALESCE (str1,...<strn>) first non-missing string of strings str1-strn
COLLATE (s,<e>) string of chars in collating seq fr start pos s to end pos e
(s,<l>) string of chars in collate seq from start pos s for length l
9COMPARE (str1,str2,<mod>) position of the leftmost character by which strings
str1 and str2 differ with optional modifiers mod
9COMPCOST CALL(opl,vall,...<opn,valn>) set the costs of operations for later
use by the COMPGED function
9COMPGED (str1,str2,<cut>,<mod>) general edit distance between strings str1
and str2; for details refer to 9ONLDOC
9COMPLEV (str1,str2,<cut>,<mod>) Levenshtein edit distance between strings
str1 and str2; for details refer to 9ONLDOC
COMPBL+ (str) removes multiple blanks between blank-delimited substrs in str
COMPRESS (str,<rem>,<mod>) removes blanks OR chars specified in rem from str;
added in V9, optional parameter mod modifies 2nd parameter
9COUNT (src,str,<mod>) number of times substr str appears in string src
with modifiers mod
9COUNTC (src,str,<mod>) number of times characters in str appear or do not
appear within src with modifiers mod
DEQUOTE+ (str) removes surrounding quotes, single or double, from str
and removes multiple single and double quotes within str
EBCASC* (str) [VMS] converts string str from ebcdic to ascii
9FIND (src,str,<pos>,<mod>) first position of string str located in string
src using start position pos and modifiers mod
9FINDC (src,str,<mod>) first position of any characters in str that appear
or do not appear within src using start position pos & modifiers mod
INDEX (src,str) first position of string str located in string src
INDEXC (src,str,...<strn>) first position of any character in any of the
strings str-strn located in string src
INDEXW+ (src,str,<dlm>) first position of the blank-delimited substring str
in string src; third parameter added in V9 allows you to specify a
delimiter other than a blank through character expression dlm

```

Character (continued)	functions
-----	7
LEFT (str)	converts leading blanks to trailing blanks in string str
LENGTH (str)	length of string str
9LENGTHC (str)	length of string str including trailing blanks
9LENGTHM (str)	amount of memory in bytes that is allocated for string str
9LENGTHN (str)	length of string str excluding trailing blanks; 0 if blank
LOWCASE+ (str)	converts all uppercase characters in str to lowercase
9NLITERAL (str)	SAS name literal of string str
9NOTALNUM (str,<pos>)	first non-alphanumeric character
9NOTALPHA (str,<pos>)	first non-alphabetic character
9NOTCNTRL (str,<pos>)	first non-control character
9NOTDIGIT (str,<pos>)	first non-digit
9NOTFIRST (str,<pos>)	first character invalid as the beginning of a SAS variable name
9NOTGRAPH (str,<pos>)	first non-graphical character
9NOTLOWER (str,<pos>)	first non-lowercase character
9NOTNAME (str,<pos>)	first character invalid as part of a SAS variable name
9NOTPRINT (str,<pos>)	first non-printable character
9NOTPUNCT (str,<pos>)	first non-punctuation character
9NOTSPACE (str,<pos>)	first non-white-space character
9NOTUPPER (str,<pos>)	first non-uppercase character
9NOTXDIGIT (str,<pos>)	first character that is not a hexadecimal digit
9NVALID (str,)	1 if string str is a valid SAS variable name based on basis b, 0 if not; for values of b refer to 9ONLDOC
OEM2ANSI* CALL(str,dst,len)	[WINDOWS] converts string str from oem to ansi for length len and stores result in string dst
9PROPCASE (str,<dlim>)	converts string str to proper case with delimiters dlm matches a pattern
QUOTE+ (str)	adds surrounding double quotes to string str, and doubles any double quotes found within str
RANK (x)	position of character x in ascii or ebcdic collating sequence
REPEAT (str,n)	string consisting of string str repeated n+1 times
REVERSE (str)	string str with its characters in reverse order
	substring that matches a pattern
RIGHT (str)	converts trailing blanks to leading blanks in string str
SCAN (str,n,<dlim>)	nth substring in str separated by delimiters dlm see 9ONLDOC for default dlm; beginning with V7 a negative value of n scans the string right to left
9SCAN CALL(str,n,pos,len,<dlim>)	position and length of nth substring in string str separated by delimiters dlm; see 9ONLDOC for default dlm
9SCANQ (str,n,<dlim>)	same as SCAN but ignores delimiters within quotation marks
9SCANQ CALL(str,n,pos,len,<dlim>)	same as CALL SCAN but ignores delimiters within quotation marks
SOUNDEX+ (str)	encodes string str according to a patented search algorithm described in SASP222, p64
SPEDIS& (str,key)	a value representing the likelihood that string str matches string key
9STRIP (str)	removes all leading and trailing blanks
9SUBPAD (str,pos,<n>)	n characters of str beginning at position pos: if n=0, null string is returned; if pos + n-1 > length(str), result is padded with blanks
SUBSTR (str,pos,<n>)	"x = substr()" returns n characters of str beginning at position pos "substr() = x" assigns n chars of x to str beginning at position pos beginning in V9, if n is omitted str will be assigned a length of 8
9SUBSTRN (str,pos,<n>)	n characters of str beginning at position pos, allowing negative values for both pos and n, and possible null string return
TRANSLATE (str,tol,from1,...<ton,fromn>)	converts all characters in str that occur in from1 to their respective character in tol for every fromn-ton pair of strings
TRANWRD+ (str,to,from)	converts all blank-delimited occurrences of string from in string str to string to
TRIM (str)	removes all trailing blanks
TRIMN+ (str)	removes all trailing blanks, NULL if result is a blank
UPCASE (str)	converts all lowercase characters in str to uppercase
VERIFY (str,excerpt1,...<excerptn>)	position of the first character in str that is not present in any of the strings excerpt1-excerptn

Arithmetic	functions
-----	8
ABS (num)	absolute value of num
AIRY+ (num)	value of the airy function; the differential equation w''-xw=0
ARCOS (num)	arccosine in radians; -1 < num < 1
ARSIN (num)	arcsine in radians; -1 < num < 1
ATAN (num)	arctangent in radians
9ATAN2 (num1,num2)	arctangent in of two numeric variables in radians
9BETA (a,b)	beta function of first shape parm a and second shape parm b
CEIL (num)	smallest integer greater than or equal to num
9CEILZ (num)	smallest integer greater than or equal to num without fuzzing
7COMB (n,r)	combinations of n elements taken r at a time
7CONSTANT (c)	machine or mathematical constant c, values of c are:
E	natural base
LOGBIG	log w/respect to base of big
PI	pi
LOGSMALL	log w/respect to base of small
EULER	euler constant
SQRTBIG	square root of big
EXACTINT	exact integer
SQRTSMALL	square root of small
BIG	largest dp num
MACEPS	machine precision constant
SMALL	smallest dp num
LOGMACEPS	log w/respect to base of maceps
[dp = double-precision]	SQRTMACEPS square root of maceps
COS (num)	cosine; num must be in radians
COSH (num)	hyperbolic cosine
DAIRY+ (num)	derivative of the airy function
7DEVIANCE (dist,var,parms...)	deviance from distribution dist using random variable var; parameter list varies depending on distribution
DIGAMMA (num)	derivative of the LGAMMA function; num > 0
ERF (num)	the integral defined in SASLang, p546
ERFC (num)	complement to the ERF function [1-erf(num)]
EXP (num)	the constant e raised to the power of num
7FACT (num)	factorial of num
FLOOR (num)	largest integer less than or equal to num
9FLOORZ (num)	largest integer less than or equal to num without fuzzing
FUZZ (num)	nearest integer value if num is within 1e-12 of the integer
GAMMA (num)	the integral defined in SASLang, p551
IBESSEL+ (nu,x,kode)	bessel function when kode=0, and modified bessel function when kode != 0 of order nu evaluated at x
INT (num)	truncates decimal portion of num
9INTZ (num)	truncates decimal portion of num without fuzzing
JBESSEL+ (nu,x)	bessel function of order nu evaluated at x
LGAMMA (num)	natural logarithm of GAMMA(num)
LOG (num)	natural logarithm of num
LOG10 (num)	common logarithm of num
LOG2 (num)	logarithm to the base 2 of num
9LOGISTIC CALL(var1,...<varn>)	replaces each variable with the logistic value of that variable; only accepts variable names
9LOGBETA (a,b)	logarithm of the beta function with shape parameters a and b
MOD (num,div)	remainder of the quotient num/div
9MODZ (num,div)	remainder of the quotient num/div without fuzzing
7PERM (n,r)	permutations of n elements taken r at a time
ROUND (num,<u>)	num rounded to the nearest unit u, default u = 1
9ROUNDE (num,<u>)	num rounded to the nearest unit u, default u = 1; if result halfway between two multiples, an even multiple is returned
9ROUNDZ (num,<u>)	num rounded to the nearest unit u [def = 1] without fuzzing
SIGN (num)	-1 if num < 0, 0 if num = 0, and 1 if num > 0
SIN (num)	sine; num must be in radians
SINH (num)	hyperbolic sine
9SOFTMAX CALL(var1,...<varn>)	replaces each variable with the softmax value of that variable; only accepts variable names
SQRT (num)	square root of num
9STDIZE CALL(<opt1>,...<optn>,var1,...<varn>)	standardizes each variable's value according to options; see 9ONLDOC for option values
TAN (num)	tangent; num must be in radians and not an odd multiple of pi/2
TANH (num)	hyperbolic tangent
9TANH CALL(<var1>,...<varn>)	replaces each variable with the hyperbolic tangent of that variable; only accepts variable names
TRIGAMMA (num)	derivative of the DIGAMMA function; num > 0
TRUNC (num,len)	truncates num stored as a double to len bytes

Probability functions
----- 9

le = "is less than or equal to"
 prob_ = "probability that an observation from a<n>|the"
 frvrdist = "from various continuous and discrete distributions"

BETAINV (p,a,b) pth quantile from beta distribution with shape parms a and b
 CDF& (id,q,<s>,<l>) left cumulative distribution function frvrdist
 CINV (p,df,<nc>) pth quantile from the chi-square distribution with degrees of freedom df and a noncentrality parameter nc
 CNONCT+ (x,df,prob) the noncentrality parameter from a noncentral chi-square distribution whose parameters are x, df, and nc
 FINV (p,ndf,ddf,<nc>) pth quantile from the F distribution with numerator and denominator degrees of freedom ndf and ddf and a noncentrality parameter nc
 FNONCT+ (x,ndf,ddf,<nc>) the noncentrality parameter from an noncentral F distribution whose parameters are x, ndf, ddf, and nc
 GAMINV (p,a) pth quantile from the gamma distribution with shape parameter a
 9LOGCDF (id,q,<s>,<l>) logarithm of the left cumulative distribution function
 LOGPDF& (id,q,<s>,<l>) logarithm of the probability density function frvrdist
 LOGPMF& (id,q,<s>,<l>) logarithm of the probability mass function frvrdist
 LOGSDF& (id,q,<s>,<l>) log of the survival function (log upper tail) frvrdist
 PDF& (id,q,<s>,<l>) probability density function frvrdist
 PMF& (id,q,<s>,<l>) probability mass function frvrdist
 POISSON (m,n) prob_ Poisson distribution, with mean m, le n
 PROBBETA (x,a,b) prob_ beta distribution, with shape parameters a and b, le x
 PROBNML (p,n,m) prob_ binomial distribution, with probability of success p, number of trials n, and number of successes m, le m
 7PROBNRM+ (x,y,r) probability from the bivariate normal distribution [avail with the QC product in 6.07]
 PROBCHI (x,df,<nc>) prob_ chi-square distribution, with degrees of freedom df and noncentrality parameter nc, le x
 PROBF (x,ndf,ddf,<nc>) prob_ F distribution, with numerator and denominator degrees of freedom ndf and ddf, and noncentrality parameter nc, le x
 PROBGAM (x,a) prob_ gamma distribution, with shape parameter a, le x
 PROBHYP (n,k,s,<r>) prob_ extended hypergeometric distribution, with population n, items k, sample size s, and odds ratio r, le x
 PROBIT (p) pth quantile from the standard normal distribution
 7PROBMC+ (dist,q,p,df,np,<parms>) probability or the quantile from various distributions with finite and infinite degrees of freedom for the variance estimate [avail with the QC product in 6.07]
 PROBNEGB (p,n,m) prob_ negative binomial distribution, with probability of success p and number of successes n, le m
 PROBNORM (x) prob_ standard normal distribution le x
 PROBT (x,df,<nc>) prob_ Student's t distribution with degrees of freedom df
 9QUANTILE (dist,prob,<s>,<l>,<sc>) quantile from a specified distribution and noncentrality parameter nc, le x
 SDF& (id,q,<s>,<l>) survival function (upper tail) frvrdist
 TINV (p,df,<nc>) pth quantile from the Student's t distribution with degrees of freedom df and a noncentrality parameter nc
 TNONCT+ (x,df,prob) the noncentrality parameter from a noncentral t distribution whose parameters are x, df, and nc

Programatical

9ALLPERM CALL(k,arg1,...<argn>) all permutations of arguments arg1 to argn
 9CHOOSEC (x,arg1,...<argn>) xth selection among arguments arg1 to argn as a character value, x can be an expression; a negative value of x treats argn as arg1 and counts right to left
 9CHOOSEN (x,arg1,...<argn>) xth selection among arguments arg1 to argn as a numeric value, x can be an expression; a negative value of x treats argn as arg1 and counts right to left
 9IFC (expr,t,f,<m>) character value t, f, or m based on whether expression expr evaluates to true, false, or missing, respectively
 9IFN (expr,t,f,<m>) numeric value t, f, or m based on whether expression expr evaluates to true, false, or missing, respectively

Random Numbers functions
----- 10

var [dist] = "variate generated from a<n>|the [dist] distribution"
 varup [dist] = "updates seed and returns a variate generated from a<n>|the [dist] distribution"

NORMAL (seed) var normal with mean 0 and variance 1
 RANBIN (seed,n,p) var binomial with mean np and variance np(1-p)
 RANCAU (seed) var Cauchy with location parameter 0 and scale parameter 1
 9RAND (dist,param1,...paramn) generates random numbers from distribution dist with parameters appropriate for specified distribution
 RANEXP (seed) var exponential with parameter 1
 RANGAM (seed,a) var gamma with parameter a
 RANNOR (seed) var normal with mean 0 and variance 1
 RANPOI (seed,m) var Poisson with mean m
 RANTBL (seed,p1,...pn) var probability mass function defined by p1-pn
 RANTRI (seed,h) var triangular with parameter h
 RANUNI (seed) var uniform on the interval (0,1)
 RANBIN CALL(seed,n,p) varup binomial with mean np and variance np(1-p)
 RANCAU CALL(seed) varup Cauchy w/location parameter 0 and scale parameter 1
 RANEXP CALL(seed) varup exponential with parameter 1
 RANGAM CALL(seed,a) varup gamma with parameter a
 RANNOR CALL(seed) varup normal with mean 0 and variance 1
 9RANPERK CALL(seed,k,n1,...<nn>) updates seed and returns random permutation of variables n1-nn taken k at a time, "n1,...<nn>" can be specified as "of n1-nn"
 9RANPERM CALL(seed,n1,...<nn>) updates seed and returns random permutation of variables n1-nn, "n1,...<nn>" can be specified as "of n1-nn"
 RANPOI CALL(seed,m) varup Poisson with mean m
 RANTBL CALL(seed,p1,...pn) varup probability mass function defined by p1-pn
 RANTRI CALL(seed,h) varup triangular with parameter h
 RANUNI CALL(seed) varup uniform on the interval (0,1)
 9STREAMINIT CALL(seed) specifies a seed to use for subsequent calls to RAND
 UNIFORM (seed) var uniform on the interval (0,1)

Statistical

of args = "of nonmissing arguments n1-nn"
 parm lists (n1,...<nn>) can be denoted using arrays (of aryl-aryn)

9COALESCE (n1,...<nn>) first non-missing value of arguments n1-nn
 CSS (n1,n2,...<nn>) corrected sum of squares of args
 CV (n1,n2,...<nn>) coefficient of variation of args
 9GEOMEAN (n1,n2,...<nn>) geometric mean of args; fuzzes values near zero to 0
 9GEOMEANZ (n1,n2,...<nn>) geometric mean value of args without fuzzing
 9HARMEAN (n1,n2,...<nn>) harmonic mean of args; fuzzes values near zero to 0
 9HARMEANZ (n1,n2,...<nn>) harmonix mean of args without fuzzing
 9IQR (n1,n2,...<nn>) interquartile range of args
 9KURTOSIS (n1,n2,n3,n4,...<nn>) kurtosis of args
 9LARGEST (k,n1,n2,...<nn>) kth largest value of args
 9MAD (n1,...<nn>) median absolute deviation from the median of args
 MAX (n1,n2,...<nn>) maximum value of args
 9MEDIAN (n1,...<nn>) median of args
 MEAN (n1,n2,...<nn>) mean value of args
 MIN (n1,n2,...<nn>) minimum value of args
 N (n1,...<nn>) number of args
 NMIS (n1,...<nn>) number of missing arguments n1-nn
 ORDINAL (count,n1,n2,...<nn>) largest of the first count arguments n1-nn
 9PCTL<n> (p,n1,...<nn>) nth (1-5) percentile p of args
 RANGE (n1,n2,...<nn>) difference between the largest and smallest of args
 9RMS (n1,...<nn>) root mean square of args
 9SKEWNESS (n1,n2,n3,...<nn>) skewness statistic of args
 9SMALLEST (k,n1,n2,...<nn>) kth smallest value of args
 STD (n1,n2,...<nn>) standard deviation of args
 STDER (n1,n2,...<nn>) standard error of the mean of args
 SUM (n1,n2,...<nn>) sum of nonmissing args n1-nn
 USS (n1,n2,...<nn>) uncorrected sum of squares of args
 VAR (n1,n2,...<nn>) variance of args

Logical	functions
-----	11
BAND+ (x1,x2) the bitwise AND of x1 and x2	
BLSHIFT+ (x1,n) the bitwise left shift of x1 for n bits	
BNOT+ (x1) the bitwise NOT of x1	
BOR+ (x1,x2) the bitwise OR of x1 and x2	
BRSHIFT+ (x1,n) the bitwise right shift of x1 for n bits	
BXOR+ (x1,x2) the bitwise EXCLUSIVE OR of x1 and x2	
Special SAS Operations	

CEXIST& (cent,<U>) 1 if catalog or catalog entry cent exists, 0 if not optional 'U' keyword parameter verifies ability to update	
DIF<n> (var) first differences between var and its nth lag; default n = 1	
DIM (array,dim) nth dimension dim of a multidimensional array	
DIM<n> (array) nth dimension of an array; default n = 1	
EXECUTE+ CALL(mac) executes resolved value of sas macro mac following the current data step; the data step must end with a run statement	
EXIST& (mem,<mtyp>,<gen>) 1 if SAS data library member exists, 0 if not; opt member type mtyp; optional generation number gen added in V9	
FILEXIST& (fref) 1 if external file identified by fileref fref exists, 0 if not	
PHYSXIST& (pname) 1 if external file identified by physical name pname exists, 0 if not	
FILENAME& (lname,<pname>,<dev>,<opts>,<dir>) if pname is present, return code from assigning fileref lname to physical file pname with device dev and host-specific options opts; if pname absent, deassigns fileref	
FILEREF& (fn) zero if filename fn has been assigned, nonzero if it has not	
GETOPTION& (opt,<rep>) value of option opt with optional reporting keyword rep	
HBOUND (array,dim) upper bound of nth dimension dim of a multidim array	
HBOUND<n> (array) upper bound of the nth dimension of array; default n = 1	
INPUT (src,inf) read the value of src using informat inf use this to convert character data to numeric data	
INPUTC+ (src,inf,<w>,<d>) read the value of src using character informat inf specified at run time; w = width and d = decimal value for informat	
INPUTN+ (src,inf,<w>,<d>) read the value of src using numeric informat inf specified at run time; w = width and d = decimal value for informat	
IORCMMSG () formatted error message associated with current value of _IORC_	
LABEL CALL(var,str) assigns label str to the variable var	
LAG<n> (var) nth lag value of variable var stored/retrieved in a queue	
LBOUND (array,dim) lower bound of the nth dimension dim of a multidim array	
LBOUND<n> (array) lower bound of the nth dimension of array; default n = 1	
LIBNAME& (lname,<pname>,<eng>,<opts>) if pname is present, return code from assigning libref lname to physical library pname with engine eng and host-specific options opts; if pname is absent, deassigns libref	
LIBREF& (lib) zero if libname lib has been assigned, nonzero if it has not	
7MISSING (e) 1 if variable or expression e contains a missing value, 0 if not	
9MISSING CALL(var1,...<varn>) assigns a missing value to all variable names var1-varn; can use "of" construct; eg, call missing(of _all_)	
PATHNAME& (fref) physical name of SAS data library or external file fref	
PUT (src,f) write the value of src using format f use this to convert numeric data to character data	
PUTC+ (src,for,<w>,<d>) write the value of src using character format for specified at run time; w = width and d = decimal value for format	
PUTN+ (src,for,<w>,<d>) write the value of src using numeric format for specified at run time; w = width and d = decimal value for format	
RESOLVE+ (mac) resolved value of sas macro mac [more flexible than SYMGET]	
9SYMEXIST (mac) 1 if sas macro mac exists, 0 if it does not exist	
SYMGET (mac) value of sas macro mac during datastep execution	
9SYMGLOBL (mac) 1 if sas macro mac is global in scope, 0 if not	
9SYMLOCAL (mac) 1 if sas macro mac is local in scope, 0 if not	
SYMPUT CALL(mac,str) assigns the value str to sas macro mac	
9SYMPUTX CALL(mac,str,<syntab>) assigns the value str to sas macro mac in symbol table syntab and removes both leading and trailing blanks	
SYMSG& () message produced from a data set or external file function call	
SYSPROD+ (prod) 1 if sas product prod is licensed, 0 if prod is not licensed, and -1 if prod is not recognized	
SYSRC& () return code from a data set or external file function call	
SYS Parm () value of the string specified with the SYS Parm option	
9UUIDGEN (<warn>,<form>) Universal Unique Identifier in form form (binary or character[default]) with maximum warnings warn [default=1]	

Financial	functions
-----	12
	sld = "straight-line depreciation"
COMPOUND (a,f,r,n) the missing argument of the four; f = a(1 + r)**n	
8CONVX (y,f,c1,...cn) convexity for an enumerated cashflow	
8CONVXP (a,c,n,k,k0,y) convexity for a periodic cashflow stream	
DACCDB (p,v,y,r) accumulated depreciation using a declining balance method	
DACCDBSL (p,v,y,r) same as DACCDB but with conversion to an sld function	
DACCSSL (p,v,y) accumulated depreciation using the straight-line method	
DACCSSD (p,v,y) accumulated depreciation using the sum-of-years-digits method	
DACCTAB (p,v,t1,...<tn>) accumulated depreciation using user-specified tables	
DEPDB (p,v,y,r) depreciation using the declining balance method	
DEPDBSL (p,v,y,r) same as DEPDB but with conversion to an sld function	
DEPSL (p,v,y) straight-line depreciation	
DEPSYD (p,v,y) sum-of-years-digits depreciation	
DEPTAB (p,v,t1,...<tn>) depreciation using specified tables	
8DUR (y,f,c1,...cn) modified duration for an enumerated cashflow	
8DURP (a,c,n,k,k0,y) modified duration for a periodic cashflow stream	
INTRR (freq,c0,c1,...cn) internal rate of return as a fraction	
IRR (freq,c0,c1,...cn) internal rate of return as a percentage	
MORT (amt,pay,rate,period) missing argument of the four amortization parms	
NETPV (rate,freq,c0,c1,...cn) net present value with rate as a fraction	
NPV (rate,freq,c0,c1,...cn) net present value with rate as a percentage	
8PVP (a,c,n,k,k0,y) present value for a periodic cashflow stream	
SAVING (f,p,r,n) the missing argument of the four from a periodic saving f = (p(1 + r)((1 + r)**n - 1) / r	
8YIELDP (a,c,n,k,k0,y) yield-to-maturity for a periodic cashflow stream	
Dates and Times	

7DATDIF (d1,d2,b) number of days between sas date values d1 and d2 according to basis b ('30/360' or 'Actual')	
DATE () sas date equal to the current date	
DATEJUL (n) converts julian date n to a sas date; n = yyddd or yyyyddd	
DATEPART (dt) date portion of the sas datetime value dt	
DATETIME () sas datetime equal to the current date and time	
DAY (d) day-of-month [1-31] from the sas date value d	
DHMS (d,h,m,s) sas datetime from date, hour, minute, and second	
HMS (h,m,s) sas time from hour, minute, and second	
HOUR (tdt) hour [0-23] from either sas time or sas datetime value tdt	
INTCK (int,fr,to) number of time intervals 'int' from 'fr' to 'to'; fr and to are sas dates, times, or datetimes; interval values are listed in SASLang, p558 and SASP222+, p58	
INTNX (int,fr,num) adds 'num' 'int' intervals to starting 'fr' fr and to are sas dates, times, or datetimes; interval values are listed in SASLang, p560 and SASP222+, p59	
INTX (int,fr,num,align) 'align' parm added in Rel 6.11 and 6.09E; advances result to the 'beginning', 'middle', or 'end' of the interval	
JULDATE (d) julian date equivalent of the sas date d	
JULDATE7& (d) julian date (with a 4-digit year) equivalent of the sas date d	
MDY (m,d,y) sas date month, day, and year	
MINUTE (tdt) minute [0-59] from either sas time or sas datetime value tdt	
MONTH (d) month [1-12] from the sas date value d	
QTR (d) quarter of the year [1-4] during which sas date value d falls	
SECOND (tdt) second [0-59] from either sas time or sas datetime value tdt	
TIME () sas time equal to the current time	
TIMEPART (dt) time portion of the sas datetime value dt	
TODAY () sas date equal to the current date	
WEEKDAY (d) day-of-week [1-7] from the sas date value d, 1 = Sunday, etc.	
YEAR (d) year as a four digit number from the sas date value d	
7YRDIF (d1,d2,b) number of years between sas date values d1 and d2 according to basis b ('30/360', 'Actual', 'ACT/360', or 'ACT/365')	
YYQ (y,q) sas date equal to the first day of quarter q in year y	

Web-based functions
----- 13

8HTMLDECODE(str) decoded string from str containing HTML numeric character references or HTML character entity references
8HTMLENCODE(str) encoded string from str using HTML character entity references
8URLDECODE (str) decoded string from str using the URL escape syntax
8URLENCODE (str) encoded string from str using the URL escape syntax

Pattern Matching

9PRXCHANGE (expr|rxid,t,src) performs pattern-matching replacement on str src
9PRXCHANGE CALL(rxid,t,ostr,<nstr>,<len>,<trunc>,<n>) performs pattern-matching
9PRXDEBUG CALL(b) if b > 0 then enables debug output; if zero, disables it
9PRXFREE CALL(rxid) frees memory allocated by expression id from PRXPARSE
9PRXMATCH (expr|rxid,src) position of beginning of pattern expr or id rxid
9PRXNEXT CALL(rxid,beg,end,src,pos,len) position and length of a substring that matches a pattern and iterates over multiple matches
9PRXPAREN (rxid) last bracket match for which there is a match in a pattern
9PRXPARSE (expr) unique numeric id to compiled Perl regular expression expr
9PRXPOSN (rxid,cb,src) capture buffer cb from string src using pattern id rxid
9PRXPOSN CALL(rxid,cb,beg,<len>) start position and length for capture buffer
9PRXSUBSTR CALL(rxid,src,pos,<len>) position and length of a substring that
7RXCHANGE CALL(rx,n,src,str) changes substring(s) that match a pattern
7RXFREE CALL(rx) frees memory allocated by other RX functions
7RXMATCH (rx,str) position of beginning of a substring that matches a pattern
7RXPARSE (expr) unique identifier value from parsing pattern expression expr
7RXSUBSTR CALL(rx,str,<pos>,<len>,<n>) finds position, length, and score of a

National Language Support (NLS)

note: no SAS version origin data provided; for details refer to 90NLDOC

EUROCURR converts one European currency to another
KCOMPARE returns the result of a comparison of character strings
KCOMPRESS removes specific characters from a character string
KCOUNT returns the number of double-byte characters in a string
KCVT converts data from an encoding code to another encoding code
KINDEX searches a character expression for a string of characters
KINDEXC searches a character expression for specific characters
KLEFT left-aligns a char expr by removing leading DBCS blanks and SO/SI
KLENGTH returns the length of an argument
KLOWCASE converts all letters in an argument to lowercase
KREVERSE reverses a character expression
KRIGHT right-aligns a char expr by trimming trailing DBCS blanks and SO/SI
KSCAN selects a specific word from a character expression
KSTRCAT concatenates two or more character strings
KSUBSTR returns a substring from an argument
KSUBSTRB extracts a substring from an argument according to byte position
KTRANSLATE replaces specific characters in a character expression
KTRIM removes trailing DBCS blanks and SO/SI from character expressions
KTRUNCATE truncates a numeric value to a specified length
KUPCASE converts all single-byte letters in an argument to uppercase
KUPDATE inserts, deletes, and replaces character value contents
KUPDATEB inserts, deletes, and replaces the contents of the character value according to the byte position of the character value in the argument
KVERIFY returns position of first character that is unique to an expression
NLDATE converts SAS-date value to the date value of the specified locale
NLDATM converts SAS-datetime value to the time value of the specified locale
NLTIME converts SAS time or datetime value to time value of specified locale
TRANSTAB transcodes a data string by using a translation table
VARTRANSCODE returns the transcode attribute of a SAS-data set variable
VTRANSCODE returns a value that indicates whether transcoding is on or off for the specified variable
VTRANSCODEX returns a value that indicates whether transcoding is on or off for the specified argument
WEEK returns the week-number value

Some Function Q and A functions
----- 14

Q) What is the best way to create a sas datetime value from a sas date variable and a sas time variable?
A) datetime = dhms(date,0,0,time);
(no need to use the hour, minute, and/or second functions)

Q) There is no cube root function; how do I get the cube root of a number?
A) Use fractional exponents:
y = x**(1/3);

Q) LAG lets me look at previous values of a variable; how do I look ahead at subsequent values?

A) The data set must be merged with itself:
for example, using the data set created by the following:
data one; do i = 1 to 20; output; end;
to create the variable 'nexti' containing the next value of i:
data next; merge one one(firstobs=2 rename=(i=nexti) keep=i);
to create the variable 'fifthi' containing the value of i five observations ahead:
data fifth; merge one one(firstobs=5 rename=(i=fifthi) keep=i);

Q) How do i convert a numeric variable to a character variable?
A) You must create a differently-named variable using the PUT function.

Q) How do i convert a character variable to a numeric variable?
A) You must create a differently-named variable using the INPUT function.

Q) How do I compute the factorial of a number?

A) In V7 and above, use the FACT function:
factor = fact(x);
In prior releases, use the increment of the number and the GAMMA function:
factor = gamma(x + 1);
Note that you are limited to the architecture of the machine as to what magnitude of number will cause an overflow error. On the IBM mainframe, the largest factorial that can be stored is 56!, which is approximately 7.1099859e74. IBM PC's have a much greater range limit.

Q) How do I use the constant pi within SAS?

A) In V7 and above, use the CONSTANT function:
pi = constant('pi');
In prior releases, here are two efficient methods of computing pi:
pi = 4 * atan(1);
pi = acos(-1); [slightly less efficient]

Q) How can I compute the integral age (in years) of something?

A) Given two sas date variables born and calc:
age = int(intck('month',born,calc) / 12);
if month(born) = month(calc) then
age = age - (day(born) > day(calc));

Q) How can I compute the number of months between two dates?

A) Given two sas date variables begin and end:
months = intck('month',begin,end) - (day(end) < day(begin));

Q) How can I determine the position of the nth word within a character string?

A) In V9 and above, use the CALL SCAN routine:
call scan(string,n,position,length);
In prior releases, use a combination of the INDEXW and SCAN functions:
position = indexw(string,scan(string,n));

Q) Why is there no WEEK function that returns 1-52?

A) Since neither 365 nor 366 is divisible by 7, some dates will have to be in week '0' or week '53'. The following code can be used if week of year is necessary:
weekofyr = intck('week',intnx('year',date,0),date);
(refer to SAS Communications, First Quarter 1992, p48)
Note: beginning in Version 9, a WEEK function is provided via National Language Support.

Some Function Q and A (continued)

functions
15

Q) I need to reorder characters within a string...use SUBSTR?
 A) You can do this using only one function call with TRANSLATE versus two functions calls with SUBSTR. The following lines each move the first character of a 4-character string to the last:

```
reorder = translate('2341',string,'1234');
reorder = substr(string,2,3) || substr(string,1,1);
```

Numeric (standard input)

informat
16

BESTw.d standard numeric integer and decimal data and scientific notation refer to W.d for examples

BINARYw.d binary data as positive numbers
 00001111 | num binary8 | 15
 00001111 | num binary8.2 | 0.15

BZw.d translates trailing and embedded blanks to zeroes
 34 | num bz4. | 3400
 -2 1 | num bz4.1 | -20.1

COMMAw.d removes commas, blanks, dollar and percent signs, dashes and right parantheses, and translates left parantheses to minus signs
 \$1,000,000 | comma10. | 1000000
 500 | comma10.1 | 50
 (500) | comma10. | -500
 (-500) | comma10. | -500

COMMAXw.d removes periods, commas, blanks, dollar and percent signs, dashes and right ')', and translates left '(' to minus signs
 \$1.000.000 | commax10. | 1000000
 (500) | commax10. | -500
 (500) | commax10.1 | -50

Dw.d standard numeric integer and decimal data and scientific notation refer to W.d for examples

DOLLARw.d same as COMMAw.d

DOLLARXw.d same as COMMAXw.d

Ew.d scientific notation
 1.257E3 | e7. | 1257
 1.257E3 | e7.1 | 125.7

Fw.d standard numeric integer and decimal data and scientific notation refer to W.d for examples

HEXw. if w < 16, hexadecimal integers; if w = 16, hexadecimal signed floating point numbers
 01F | hex3. | 31
 4152000000000000 | hex16. | 5.125
 C310000000000000 | hex16. | -256

NUMXw.d& numeric values with a comma for the decimal point
 896.48 | numx6. | 896.48

OCTALw.d values stored in base eight
 177 | octal3. | 127
 177 | octal3.1 | 12.7

PERCENTw. percentages; removes commas, blanks, dollar signs, and dashes, translates left parantheses to minus signs, and interprets right parantheses as division by 100
 1% | percent6. | 0.01
 -1% | percent6. | -0.01
 20-% | percent6. | 0.2
 (20%) | percent6. | -0.2
 (-20%) | percent6. | -0.2

9TRAILSGNw. numeric values that may contain trailing plus or minus signs
 5- | trailsgn5. | -5
 14.3+ | trailsgn5. | 14.3
 14.3 | trailsgn5. | 14.3

w.d standard numeric integer and decimal data and scientific notation
 23 | 5. | 23
 23 | 5.5 | 0.00023
 -23 | 5. | -23
 23.2 | 5. | 23.2
 23.2 | 5.5 | 23.2
 2.3E1 | 5. | 23

Character

\$ASCIIW. converts ascii to native format
 abc | \$ascii3. | [e]'018283'x [a]'616263'x

\$BINARYw. binary data (every 8 bits = 1 character)
 01001100010011101 | \$binary16. | [e]'LM'x [a]'<('

\$CBw. column-binary files in punchcard code refer to SASLang, p638 for example

Character (continued)	informat	17
\$CHARw.	preserves leading and trailing blanks, '.' is not read as missing abc \$char5. ' abc '	
\$CHARZBw.	translates nulls to blanks '81820083'x \$charzb4. [e]'ab c'	
\$EBCDICw.	converts ebcdic to native format abc \$asci3. [e]'818283'x [a]'616263'x	
\$Fw.	trims leading blanks, '.' is read as missing refer to \$W. for examples	
\$HEXw.	hexadecimal data 5A5A \$hex4. [e]!!! [a]'ZZ'x	
\$OCTALw.	octal data, width w is the number of octal digits multiplied by 3 132132 \$octal18. [e]!!! [a]'ZZ'x	
\$PHEXw.	packed decimal notation, the low-order nibble is ignored '1E0F'x \$phex2. '1E0'	
\$QUOTEw.+	removes matching beginning and ending quotes 'SAS' \$quote3. SAS "SAS" \$quote3. SAS "6'2" \$quote3. 6'2	
\$REVERJw.@	inputs text right to left, preserves leading and trailing blanks ABCD \$reverj6. ' DCBA'	
\$REVERSw.@	inputs text right to left, left justifies result ABCD \$revers6. 'DCBA'	
\$UPCASEw.+	converts all lowercase characters to uppercase sas \$upcase3. 'SAS'	
\$VARYINGw.	varying lengths of data, a length variable must follow, w=max len 5floyd lv 1. str \$varying9. lv str='floyd'	
\$w.	trims leading blanks, '.' is read as missing abc \$5. 'abc '	

Numeric (hexadecimal input)

BITSw.d	extracts bits as positive numbers, d is 0-based offset [0-63] 'C2'x bits4.4 2 'C2'x bits6.1 33
CBw.d	column-binary files in punchcard code refer to SASLang, p647 for example
FLOATw.g+	floating point numbers; compare with RBw.d--different results for truncated 8-byte floating point numbers under operating systems using IEEE floating point standard
IBw.d	signed integers '00000080'x ib4. 128 '00000080'x ib4.2 1.28 'FFFFFFFE'x ib4. -2
7IBRw.d	integer binary (fixed-point) in Intel and DEC format 'A900'x ibr2. 169 '31420000'x ibr4.2 169.45
IEEEw.d+ PDw.d	floating point numbers stored in IEEE standard format signed packed decimal data '0000128A'x pd4. 128 '0000128B'x pd4. -128 '0000128C'x pd4. 128 '0000128D'x pd4. -128 '0000128E'x pd4. 128 '0000128F'x pd4. 128 '0000128F'x pd4.2 1.28
PIBw.d	positive integers A pib1. [e]193 [a]65 '65'x pib1. 101 '65'x pib1.1 10.1 '0100'x pib2. 256
7PIBRw.d	positive integer binary (fixed-point) in Intel and DEC format '0001'x pibr2. 256
PKw.d	unsigned packed decimal data '001234'x pk3. 1234 '001234'x pk3.2 12.34
PUNCH.d	whether a row of column-binary data is punched; refer to 9ONLDOC

Numeric (hexadecimal input, continued)	informat	18
RBw.d	floating point numbers '4280000000000000'x rb8. 128 '4280000000000000'x rb8.1 12.8 '4280089345600000'x rb8. 128.03349718 'C110000000000000'x rb8. -1	
ROWw.d	column-binary field down a card column; refer to 9ONLDOC	
S370FFw.d=	standard numeric data stored in IBM mainframe format 'F1F2F3'x s370ff3. 123 'F1F2F3'x s370ff3.2 1.23	
S370FIBw.d	signed integers stored in IBM mainframe format refer to IBw.d for examples	
S370FIBUw.d=	unsigned integers stored in IBM mainframe format identical to S370FPBw.d	
S370FPDw.d	signed packed decimal data stored in IBM mainframe format refer to PDw.d for examples	
S370FPDUw.d=	unsigned packed decimal data stored in IBM mainframe format, similar to S370FPDw.d, but all sign digits except 'F' are rejected	
S370FPBw.d	positive integers stored in IBM mainframe format refer to PIBw.d for examples	
S370FRBw.d	floating point numbers stored in IBM mainframe format refer to RBw.d for examples	
S370FZDw.d=	zoned decimal data stored in IBM mainframe format refer to ZDw.d for examples	
S370FZDLw.d=	zoned decimal leading sign data stored in IBM mainframe format 'C1F2F3'x s370fzdl3. 123 'C1F2F3'x s370fzdl3.2 1.23 'D1F2F3'x s370fzdl3.2 -123	
S370FZDSw.d=	zoned decimal separate leading sign data stored in IBM mainframe format '4EF1F2F3'x s370fzds4. 123 '60F1F2F3'x s370fzds4.2 -1.23	
S370FZDTw.d=	zoned decimal separate trailing sign data stored in IBM mainframe format 'F1F2F34E'x s370fzdt4. 123 'F1F2F360'x s370fzdt4.2 -1.23 'F1F2F340'x s370fzdt4.2 [invalid]	
S370FZDUw.d=	unsigned zoned decimal data stored in IBM mainframe similar to S370FZDw.d, but all sign digits except 'F' are rejected	
VAXRBw.d	floating point numbers stored in VMS format refer to RBw.d for examples	
VMSZNd.*+	[VMS] zoned decimal data in VMS format, last digit is '0'-'9' for positive values and 'p'-'y' for negative values 1234 vmszn4. 1234 123t vmszn4.1 -123.4	
ZDw.d	zoned decimal data, ignores high order nibbles, last digit is '{-'I' for positive values and '}'-'R' for negative values 'F0F1F2C8'x zd4. 128 'F1F0F2D8'x zd4. -1028 '81F0F2F8'x zd4. 1028 'F1F2F3D9'x zd4.1 -123.9	
ZBw.d	zoned decimal data produced in IBM 1410, 1401, and 1620 form (zeroes are '40'x instead of 'F0'x) '40F140C8'x zd4. 108 'F1F240D9'x zd4.1 -120.9	
ZDVw.d+	zoned decimal data; validates high order nibbles 'F1F0F2D8'x zd4. -1028 '81F0F2F8'x zd4. [invalid]	

Dates and Times

w/dlm = "delimiters may separate day, month, and year values, but the delimiter must be consistent and used throughout"

9ANYDTDEw.	date values of any of the following forms: DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MONYY, TIME, YMMDD, or YYQ
9ANYDTDTw.	datetime values of any of the following forms: DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MONYY, TIME, YMMDD, or YYQ
9ANYDTTMEw.	time values of any of the following forms: DATE, DATETIME, DDMMYY, JULIAN, MMDDYY, MONYY, TIME, YMMDD, or YYQ

Dates and Times (continued)

informat
19

DATEw. date of the form ddmmyy<yy>, w/dlm
1jan1990 | date10. | 10958
01 jan 90 | date10. | 10958
1-jan-1990 | date10. | 10958

DATETIMEw. datetime of the form ddmmyy<yy>chh:mm<:ss.ss>, w/dlm, c is any
delimiter
23dec89 10:03:17.2 | datetime22. | 946029797.2
23dec1989D10:03:17.2 | datetime22. | 946029797.2
23-dec-1989/10:03:17.2 | datetime22. | 946029797.2

DDMMYYw. date of the form ddmmyy<yy>, w/dlm
231090 | ddmmyy8. | 11253
23/10/90 | ddmmyy8. | 11253
23 10 90 | ddmmyy8. | 11253
23/10/1990 | ddmmyy10. | 11253

JULIANw. date of the form <yy>yyddd
90091 | julian5. | 11048

MMDYYw. date of the form mmddy<yy>, w/dlm
010190 | mmdyy8. | 10958
1/1/90 | mmdyy8. | 10958
01 1 90 | mmdyy8. | 10958
01/01/1990 | mmdyy10. | 10958

MONYYw. date of the form mmyy<yy>, the first day of the month is used
jan90 | monyy5. | 10958

MSEC8. time stored in IBM mainframe TIME MIC values
'0000EA044E65A000'x | msec8. | 62818.412122

PDJULGW.& IBM packed decimal julian dates in the hexadecimal form yyydddF
'1999003F'x | pdjulg4. | [jan 03, 1999]
'2000003F'x | pdjulg4. | [jan 03, 2000]

PDJULIW.& IBM packed decimal julian dates in the hexadecimal form cyydddF
where cc is the century indicator ('00'x = 1900, '01'x = 2000)
'0001003F'x | pdjuli4. | [jan 03, 1901]
'0101003F'x | pdjuli4. | [jan 03, 2001]

PDTIME4. time stored in IBM mainframe SMP and RMF records
'0142225F'x | pdtime4. | 51745

RMFDUR4. time from RMF measurement intervals of IBM mainframe RMF records
'3552226F'x | rmfdur4. | 2152.266

RMFSTAMP8. datetime stored in IBM mainframe RMF records
'0142225F0089286F'x | rmfstamp8. | 939910945

7SHRSTAMP8. datetime stored in IBM mainframe SHR records
'0097239F12403576'x | shrstamp8. | 1188304835.8

SMFSTAMP8. datetime stored in IBM mainframe SMF records
'004EF5280089286F'x | smfstamp8. | 939910945.68

9STIMERw. time values written to the log via the STIMER option
54 | stimer11. | 54
5:54 | stimer11. | 354
10:05:54.98 | stimer11. | 36354.98

TIMEw. time of the form hh:mm<:ss.ss><am|pm>, the ':' may be substituted
with any non-alphanumeric character
14:22:25 | time8. | 51745
02:22:25pm | time10. | 51745
02:22:25am | time10. | 8545

TODSTAMP8. datetime stored as IBM mainframe time-of-day clock value
'93B200C19E7A2000'x | todstamp8. | 704914018.41

TU4. time stored as IBM mainframe timer unit value
'8FC7A9BC'x | tu4. | 62818.411563

VMSTIME.* [VMS] time stored in VMS format timestamp

YYMMDDw. date of the form <yy>ymmdd, w/dlm
900101 | yymmdd8. | 10958
90/1/1 | yymmdd8. | 10958
1990 1 1 | yymmdd8. | 10958
1990/01/01 | yymmdd10. | 10958

YYMMNW.& date in the form of <yy>yymm, the first day of the month is used;
prior to 6.09E TS470 and 6.12 TS060, this informat is available
only to sites licensing ETS
9001 | yymmn4. | 10958
199001 | yymmn6. | 10958

YYQw. date in the form of <yy>yyQq, resulting in the first day of the
quarter 'q' for year '<yy>yy'
90Q2 | yyq4. | 11048

National Language Support (NLS)

informat
20

note: no SAS version origin data provided
for details refer to the SAS Version 9 online documentation

SCPTDw. converts Hebrew DOS (cp862) encoding to Windows (cp1255) encoding

SCPTDw. converts Windows (cp1255) encoding to Hebrew DOS (cp862) encoding

\$KANJW. removes shift code data from DBCS data

\$KANJXw. adds shift-code data to DBCS data

\$LOGVSw. converts left-to-right logical order to visual order

\$LOGVSRw. converts right-to-left logical order to visual order

\$VSLGw. converts visual order to left-to-right logical order

\$VSLGRw. converts visual order to right-to-left logical order

\$UCS2Bw. converts big-endian, 16-bit, UCS2, Unicode to current encoding

\$UCS2BEw. converts current encoding to big-endian, 16-bit, UCS2, Unicode

\$UCS2Lw. converts little-endian, 16-bit, UCS2, Unicode to current encoding

\$UCS2LEw. converts current encoding to little-endian, 16-bit, UCS2, Unicode

\$UCS2Xw. converts 16-bit, UCS2, Unicode to current SAS session encoding

\$UCS2XEw. converts current SAS session encoding to 16-bit, UCS2, Unicode

\$UCS4Bw. converts big-endian, 32-bit, UCS4, Unicode to current encoding

\$UCS4Lw. converts little-endian, 32-bit, UCS4, Unicode to current encoding

\$UCS4Xw. converts 32-bit, UCS4, Unicode to current SAS session encoding

\$UCS4XEw. converts current SAS session encoding to 32-bit, UCS4, Unicode

\$UESCw. converts string encoded in UESC representation to current encoding

\$UESCW. converts string of current encoding to UESC representation

\$UNCRw. converts NCR character string to current SAS session encoding

\$UNCREw. converts current SAS session encoded string to NCR string

\$UPARENw. converts str encoded in UPAREN representation to current encoding

\$UPARENw. converts string of current encoding to UPAREN representation

\$UPARENpw. same as UPAREN but national chars remain in UPAREN representation

\$UTF8w. converts string encoded in UTF-8 to current SAS session encoding

EURDFDEw. date in an international format similar to DATE.

EURDFDTw. datetime in an international format

EURDFMYw. date in an international format similar to MONYY.

EUROWd. removes chars in European currency; reverses comma & decimal point

EUROXw.d. removes embedded characters in European currency

JDATEYMDw. Japanese kanji date in the form <yy>yymmdd

JNENGow. Japanese kanji date in the form yymmdd

MINGUow. date of the Taiwanese form yymmdd

NENGow. date stored in Japanese form r.yymmdd

NLDATEw. converts date value of specified locale to local SAS date value

NLDATMw. converts datetime value of specified locale to local SAS datetime

NLMNYw.d. monetary data in the specified locale for the local expression

NLMNYIw.d. monetary data in specified locale for the international expression

NLNUMw.d. numeric data in the specified locale for local expressions

NLNUMIw.d. numeric data in the specified locale for international expressions

NLPCTw.d. percentage data in the specified locale for local expressions

NLPCTIw.d. percentage data in specified locale for international expressions

NLTIMAPw. converts am/pm time value of specified locale to local SAS time

NLTIMEw. converts time value of specified locale to local SAS time value

WEEKUw. converts number-of-week value to SAS date value using U algorithm

WEEKVw. converts number-of-week value to SAS date value using V algorithm

WEEKWw. converts number-of-week value to SAS date value using W algorithm

YENW.d. removes commas, decimal points, and yen signs

Numeric (hexadecimal output)	formats

abs(neg) = 'takes absolute value of negative values'	21
FLOATw.d+	floating point numbers; compare with RBw.d--different results for truncated 8-byte floating point numbers under operating systems using IEEE floating point standard
IBw.d	signed integers
	128 ib4. '00000080'x
	1.28 ib4.2 '00000080'x
	-2 ib4. 'FFFFFFFE'x
7IBRw.d	integer binary (fixed-point) in Intel and DEC format
	169 ibr2. 'A900'x
	169.45 ibr4.2 '31420000'x
IEEEw.d+	floating point numbers in IEEE standard format
PDw.d	signed packed decimal values
	128 pd4. '0000128C'x
	1.28 pd4.2 '0000128C'x
	-128 pd4. '0000128D'x
PIBw.d	positive integers
	193 pib1. 'C1'x
	10.1 pib1.1 '65'x
	256 pib2. '0100'x
7PIBRw.d	positive integer binary (fixed-point) in Intel and DEC format
	256 pibr2. '0001'x
PKw.d	unsigned packed decimal values
	1234 pk3. '001234'x
	12.34 pk3.2 '001234'x
RBw.d	floating point numbers
	128 rb8. '4280000000000000'x
	12.8 rb8.1 '427FFFFFFFFFFFFF'x
	128.03349718 rb8. '42800893456C9BB4'x
	-1 rb8. 'C1100000000000001'x
7S370FFw.d	standard numeric data stored in IBM mainframe format
	123 s370ff3. 'F1F2F3'x
S370FIBw.d	signed integers in IBM mainframe format
	refer to IBw.d for examples
S370FIBUw.d=	unsigned integers in IBM mainframe format, abs(neg)
	245 s370fibu1. 'F5'x
	-245 s370fibu1. 'F5'x
S370FPDw.d	signed packed decimal values in IBM mainframe format
	refer to PDw.d for examples
S370FPDUw.d=	unsigned packed decimal data in IBM mainframe format, abs(neg)
	123 s370fpdu2. '123F'x
	-123 s370fpdu2. '123F'x
S370FPIBw.d	positive integers in IBM mainframe format
	refer to PIBw.d for examples
S370FRBw.d	floating point numbers in IBM mainframe format
	refer to RBw.d for examples
S370FZDw.d=	zoned decimal data in IBM mainframe format
	refer to ZDw.d for examples
S370FZDLw.d=	zoned decimal leading sign data in IBM mainframe format
	123 s370fzdl3. 'C1F2F3'x
	-123 s370fzdl3. 'D1F2F3'x
S370FZDSw.d=	zoned decimal separate leading sign data in IBM mainframe format
	123 s370fzds4. '4E1F2F3'x
	-123 s370fzds4. '601F2F3'x
S370FZDTw.d=	zoned decimal separate trailing sign data in IBM mainframe format
	123 s370fzdt4. 'F1F2F34E'x
	-123 s370fzdt4. 'F1F2F360'x
S370FZDUw.d=	unsigned zoned decimal data in IBM mainframe format, abs(neg)
	123 s370fzdu3. 'F1F2F3'x
	-123 s370fzdu3. 'F1F2F3'x
9VAXRBw.d	floating point numbers in VMS format
	1 vaxrb8. '8040000000000000'x
VMSZNw.d*+	[VMS] zoned decimal data in VMS format
	-1234 vmszn5. 123t
ZDw.d	zoned decimal data, last digit is {'-' 'I' for positive values and '-' 'R' for negative values
	128 zd4. 'F0F1F2C8'x
	-1028 zd4. 'F1F0F2D8'x
	-123.9 zd4.1 'F1F2F3D9'x

Dates and Times	formats

w/x_sep = "with x being C, D, N, P, or S to designate comma, dash, "null, period, or slash, respectively as the separator"	22
DATEw.	date value in the form ddmmyy<yy>
	10847 date5. 12SEP
	10847 date7. 12SEP89
	10847 date9. 12SEP1989
DATEAMPw.d&	datetime value in the form ddmmyy<yy>:hh:mm:ss.ss AM PM
	937192783 dateampm10. 12SEP89:03
	937192783 dateampm13. 12SEP89:03 AM
	937192783 dateampm19. 12SEP89:03:19:43 AM
DATETIMEw.d	datetime value in the form ddmmyy<yy>:hh:mm:ss.ss
	937192783 datetime7. 12SEP89
	937192783 datetime16. 12SEP89:03:19:43
	937192783 datetime18.1 12SEP89:03:19:43.0
DAYw.	day of month [1-31] from a date value
	10919 day2. 23
DDMMYYw.	date value in the form ddmmyy<yy>
	11316 ddmmyy5. 25/12
	11316 ddmmyy6. 251290
	11316 ddmmyy10. 25/12/1990
7DDMMYYxw.&	date value in the form ddmmyy<yy>, w/x_sep; avail in 6.12 and 6.09E as DDMMYYNw. (with only the 'N' separator)
	11316 ddmmyy6. 120989
	11316 ddmmyy8. 12091989
	11316 ddmmyyd8. 12-09-89
DOWNAMEw.	day of week from a date value
	10621 downame6. Sunday
	10621 downame3. Sun
9DTDATEw.	same as DATEw. but accepts a datetime value
9DTMONYYw.	same as MONYYw. but accepts a datetime value
9DTWKDATXw.	same as WEEKDATXw. but accepts a datetime value
9DTYEARw.	same as YEARw. but accepts a datetime value
9DTYYQCw.	year and quarter from a datetime value with a colon as separator
	1447343794 dtyyqc6. 2005:4
HHMMw.d	hours and minutes from a time value
	10530 hhmm. 2:55
	10530 hhmm7.2 2:55.50
HOURLw.d	hours from a time value
	10530 hour. 3
	10530 hour6.2 2.93
JULDAYw.	julian day of the year from a date value
	11048 julday3. 91
JULDATEw.	julian date from a date value
	refer to JULIANw. for examples
JULIANw.	julian date from a date value
	11048 julian5. 90091
	11048 julian7. 1990091
MDDYYw.	date value in the form mmddy<yy>
	10847 mmdyy4. 0912
	10847 mmdyy5. 09/12
	10847 mmdyy6. 091289
	10847 mmdyy7. 091289
	10847 mmdyy8. 09/12/89
	10847 mmdyy10. 09/12/1989
7MDDYYxw.&	date value in the form mmddy<yy>, w/x_sep; avail in 6.12 and 6.09E as MDDYYNw. (with only the 'N' separator)
	10847 mmdyy6. 091289
	10847 mmdyy8. 09121989
	10847 mmdyyd8. 09-12-89
MMSSw.d	minutes and seconds from a time value
	4530 mmss. 75:30
	4530.34 mmss8.2 75:30.34
MMYYw.	month and year from a date value, w/x_sep
	10741 mmyy5. 05M89
	10741 mmyy7. 05-1989
	10741 mmyys7. 05/1989
MONNAMEw.	month name from a date value
	10919 monname9. November
	10919 monname3. Nov

Dates and Times (continued)

formats
23

MONTHw. month of the year [1-12] from a date value
10919 | month2. | 11

MONYYw. date value in the form mmmyy<yy>
10958 | monyy5. | JAN90
10958 | monyy7. | JAN1990

PDJULGw.& IBM packed decimal julian dates in the hexadecimal form yyyydddF
14247 | pdjulg4. | '1999003F'x
14612 | pdjulg4. | '2000003F'x

PDJULIw.& IBM packed decimal julian dates in the hexadecimal form cyyyddF
where cc is the century indicator ('00'x = 1900, '01'x = 2000)
-21547 | pdjuli4. | '0001003F'x
14978 | pdjuli4. | '0101003F'x

QTRw. quarter of the year from a date value
10741 | qtr1. | 2

QTRRw. quarter of the year from a date value using Roman numerals
10741 | qtr3. | II

TIMEAMPw.d& time in the form hh:mm:ss.ss AM|PM from a time value
51745 | timeampm5. | 2 PM
51745 | timeampm11. | 2:22:25 PM
8545 | timeampm11. | 2:22:25 AM

TIMew.d time in the form hh:mm:ss.ss from a time value
51745 | time8. | 14:22:25
51745.23 | time11.2 | 14:22:25.23

TODw. time portion from a datetime value
956978640 | tod8. | 3:24:00

VMSTIMEF.* [VMS] datetime value in VMS date and time format

WEEKDATEw. day name, month name, day, and year from a date value
10848 | weekdate3. | Wed
10848 | weekdate9. | Wednesday
10848 | weekdate17. | Wed, Sep 13, 1989
10848 | weekdate29. | Wednesday, September 13, 1989

WEEKDATXw. day name, day, month name, and year from a date value
10848 | weekdatx17. | Wed, 13 Sep 1989
10848 | weekdatx29. | Wednesday, 13 September 1989

WEEKDAYw. day of week [1-7] from a date value
10848 | weekday1. | 4

WORDDATEw. month name, day, and year from a date value
10848 | worddate3. | Sep
10848 | worddate9. | September
10848 | worddate12. | Sep 13, 1989
10848 | worddate18. | September 13, 1989

WORDDATXw. day, month name, and year from a date value
10848 | worddatx3. | Sep
10848 | worddatx9. | September
10848 | worddatx12. | 13 Sep 1989
10848 | worddatx18. | 13 September 1989

YEARw. year from a date value
10848 | year2. | 89
10848 | year4. | 1989

YYMMXw. year and month from a date value, w/x_sep
10741 | yymm5. | 89M05
10741 | yymm7. | 1989:05

YYMMDDw. date value in the form <yy>yymmdd
10669 | yymmdd4. | 8903
10669 | yymmdd6. | 890318
10669 | yymmdd10. | 1989-03-18

7YYMMDDXw.& date value in the form <yy>yymmdd, w/x_sep; avail in 6.12 and 6.09E as YYMMDDNw. (with only the 'N' separator)
10669 | yymmddn6. | 890318
10669 | yymmddn8. | 19890318
10669 | yymmddn8. | 89-03-18

YYMONw. date value in the form <yy>yymmm
10958 | yymon5. | 90JAN
10958 | yymon7. | 1990JAN

YYQXw. year and quarter from a date value, w/x_sep
10741 | yyqd6. | 1989-2
10741 | yyqn5. | 19892

YYQRXw. year and quarter (Roman numerals) from a date value, w/x_sep
10741 | yyqrp8. | 1989.II

Numeric (standard output)

formats
24

BESTw. SAS chooses best notation for specified width
1257000 | best7. | 1257000
1257000 | best6. | 1.26E6
1257000 | best3. | 1E6

BINARYw. binary representation; truncates decimal, writes all negative values as all '1's
123.45 | binary8. | 01111011
123 | binary8. | 01111011
-123 | binary8. | 11111111

COMMAw.d inserts commas, prior to V7 d must be zero or two
23451.23 | comma9.2 | 23,451.23

COMMAXw.d inserts periods with a comma separating the decimal fraction, prior to V7 d must be zero or two
23451.23 | commax9.2 | 23.451,23

Dw.s# numeric values using at least s significant digits (this writes numbers in similar ranges with same number of decimal places)
12345 | d10.4 | 12345.0
1234.5 | d10.4 | 1234.5
123.45 | d10.4 | 123.45000
12.345 | d10.4 | 12.34500
1.2345 | d10.4 | 1.23450
.12345 | d10.4 | 0.12345

DOLLARw.d prefixes '\$', inserts commas, prior to V7 d must be zero or two
1254.71 | dollar9.2 | \$1,254.71

DOLLARXw.d prefixes '\$', inserts periods with a comma separating the decimal fraction, prior to V7 d must be zero or two
1254.71 | dollar9.2 | \$1.254,71

Ew.d scientific notation
1257 | e10. | 1.257E+03

Fw.d standard numeric integer and decimal data refer to W.d for examples

FRACTw. converts decimal data to fractions in reduced form
0.666667 | fract8. | 2/3
0.2784 | fract10. | 174/625

HEXw. if w < 16, hexadecimal integers; if w = 16, hexadecimal signed floating point numbers
31 | hex8. | 0000001F
5.125 | hex16. | 4152000000000000
-256 | hex16. | C310000000000000

NEGPARENw.d displays negative numbers in parentheses, inserts commas
1000 | negparen10. | b 1,000b [b = blank]
-1000 | negparen10. | (1,000)

NUMXw.d& numeric values with a comma for the decimal point
896.48 | numx6.2 | 896,48
896.48 | numx6 | 896

OCTALw octal integers
3592 | octal6. | 007010

PERCENTw.d percentages; uses BESTw. format, displays negative values in parentheses, width must be between 0 and 2
0.1 | percent10. | b 10%b [b = blank]
1.2 | percent10. | b 120%b [b = blank]
-0.05 | percent10. | (5%)
-0.053 | percent10.2 | (5.30%)

PVALUEw.d# writes p-values
-1 | pvalue. | 0.0
0 | pvalue. | 0.0001
1 | pvalue. | 1.0000

ROMANw. values in Roman numerals
1992 | roman7. | MCMXCII

SSNw. social security numbers with dashes
263878439 | ssn11. | 263-87-8439

UICw.* [VMS] converts numeric values to VMS UIC strings

VMSMSGw.*+ [VMS] numeric values as character strings containing the equivalent VMS message

w.d standard numeric integer and decimal data
23.45 | 6.3 | 23.450

Numeric (standard output, continued)

formats
25

WORDFw. numbers in English with fractions in hundredths
2.5 | wordf15. | two and 50/100
2.5 | wordf10. | two and 5*
-2.5 | wordf19. | minus two and 50/1*
-2.5 | wordf21. | minus two and 50/100

WORDSw. numbers and fractions in English with fractions in hundredths
2.1 | words23. | two and ten hundredths
-2.1 | words23. | minus two and ten hund*-2.1 | words28. | minus two and ten hundredths

Zw.d standard numeric integer and decimal data with leading zeroes
23.45 | z9.3 | 00023.450

Character

\$ASCIW. converts native format to ascii
abc | \$ascii3. | [a]'616263'x

\$BINARYw. converts to binary data (every character = 8 binary digits)
AB | \$binary16. | [e]'1100001111000010' [a]'0100000101000010'

\$CHARw. standard character data, does not trim leading blanks
refer to \$W. for examples

\$EBDCIW. converts native format to ebcdic
abc | \$ascii3. | [e]'818283'x

\$Fw. standard character data, does not trim leading blanks
refer to \$W. for examples

\$HEXw. converts to hexadecimal data (every character = 2 hex digits)
AB | \$hex4. | [e]'C1C2' [a]'4142'x

\$MSGCASEw.+ converts lowercase characters to uppercase based on the value of the MSGCASE system option

\$OCTALw. converts to octal data (every character = 3 octal digits)
A | \$octal9. | [e]'301100100' [a]'101040040'x

\$QUOTEw.+ adds enclosing double quotation marks
SAS | \$quote. | "SAS"

\$REVERJw.@ reverses character order, does not justify result
ABCD | \$reverb6. | ' DCBA'

\$REVERSw.@ reverses character order, left justifies result
ABCD | \$reverb6. | 'DCBA'

\$UPCASEw.+ converts lowercase characters to uppercase
sas | \$upcase3. | SAS

\$VARYINGw. varying lengths of data, length variable must follow, w=max len
str='floyd' lv=5 | str \$varying9. lv | 'floyd'

\$w. standard character data, does not trim leading blanks
abc | \$char5. | ' abc'

National Language Support (NLS)

note: no SAS version origin data provided

for details refer to the SAS Version 9 online documentation

\$BIDIw. reverses order of Hebrew chars; preserves order of Latin words

\$CPTDWw. Hebrew text in IBM-PC (cp862) in Windows Hebrew encoding (cp 1255)

\$CPTWDWw. Hebrew text in Windows (cp1255) in Hebrew DOS (cp862) encoding

\$KANJlw. adds shift-code data to DBCS data

\$KANJlw. removes shift-code data from DBCS data

\$LOGVSw. string that is in left-to-right-logical order in visual order

\$LOGVSRw. string that is in right-to-left-logical order in visual order

\$UCS2Bw. current SAS session encoding in big-endian, 16-bit, UCS2, Unicode

\$UCS2BEw. big-endian, 16-bit, UCS2, Unicode in current session SAS encoding

\$UCS2Lw. current session encoding in little-endian, 16-bit, UCS2, Unicode

\$UCS2LEw. little-endian, 16-bit, UCS2, Unicode in current session encoding

\$UCS2Xw. current session encoding in native-endian, 16-bit, UCS2, Unicode

\$UCS2Xew. native-endian, 16-bit, UCS2, Unicode in current session encoding

\$UCS4Bw. current session encoding in big-endian, 32-bit, UCS4, Unicode

\$UCS4BEw. big-endian, 32-bit, UCS4, Unicode in current SAS session encoding

\$UCS4Lw. current session encoding in little-endian, 32-bit, UCS4, Unicode

\$UCS4LEw. little-endian, 32-bit, UCS4, Unicode in current session encoding

\$UCS4Xw. current session encoding in native-endian, 32-bit, UCS4, Unicode

\$UCS4Xew. native-endian, 32-bit, UCS4, Unicode in current session encoding

\$UESCW. current session encoding in Unicode escape (UESC) representation

National Language Support (NLS) (continued)

formats
26note: no SAS version origin data provided
for details refer to the SAS Version 9 online documentation

\$UESCEw. Unicode escape (UESC) representation in current session encoding

\$UNCRw. current session encoding in numeric character representation (NCR)

\$UNCREw. numeric character representation (NCR) in current session encoding

\$UPARENw. current session encoding in Unicode parenthesis representation

\$UPARENw. Unicode parenthesis representation in current session encoding

\$UTF8Xw. current session encoding in universal transformation format

\$VSLGW. string that is in visual order in left-to-right logical order

\$VSLGRw. string that is in visual order in right-to-left logical order

EURDFDDw. date in an international format similar to DDMMYY.

EURDFDEw. date in an international format similar to DATE.

EURDFDNw. date in an international format similar to WEEKDAY.

EURDFDTw. datetime in an international format, refto610

EURDFDWNw. date in an international format similar to DOWNNAME.

EURDFMNw. date in an international format similar to MONNAME.

EURDFMYw. date in an international format similar to MONYY.

EURDFWDXw. date in an international format similar to WORDDATX.

EURDFWKXw. date in an international format similar to WEEKDATX.

EURFRATSw.d converts an amount from Austrian schillings to euros

EURFRBEFw.d converts an amount from Belgian francs to euros

EURFRCHFw.d converts an amount from Swiss francs to euros

EURFRCKw.d converts an amount from Czech koruny to euros

EURFRDEMw.d converts an amount from Deutsche marks to euros

EURFRDKw.d converts an amount from Danish kroner to euros

EURFRPw.d converts an amount from Spanish pesetas to euros

EURFRFIMw.d converts an amount from Finnish markkaa to euros

EURFRFRFw.d converts an amount from French francs to euros

EURFRGBPw.d converts an amount from British pounds to euros

EURFRGRDw.d converts an amount from Greek drachmas to euros

EURFRHUFw.d converts an amount from Hungarian forints to euros

EURFRIEPw.d converts an amount from Irish pounds to euros

EURFRITLw.d converts an amount from Italian lire to euros

EURFRLUFw.d converts an amount from Luxembourg francs to euros

EURFRNLGw.d converts an amount from Dutch guilders to euros

EURFRNOKw.d converts an amount from Norwegian krone to euros

EURFRPLZw.d converts an amount from Polish zlotys to euros

EURFRPTEw.d converts an amount from Portuguese escudos to euros

EURFRROLw.d converts an amount from Romanian lei to euros

EURFRRURw.d converts an amount from Russian rubles to euros

EURFRSEKw.d converts an amount from Swedish kronor to euros

EURFRSITw.d converts an amount from Slovenian tolar to euros

EURFRTRLw.d converts an amount from Turkish liras to euros

EURFRYUw.d converts an amount from Yugoslavian dinars to euros

EUROW.d prefixes euro symbol, inserts commas, period separates decimal

EUROXw.d prefixes euro symbol, inserts periods, comma separates decimal

EURTOATSw.d converts an amount from euros to Austrian schillings

EURTOBEFw.d converts an amount from euros to Belgian francs

EURTOCHFw.d converts an amount from euros to Swiss francs

EURTOCZKw.d converts an amount from euros to Czech koruny

EURTODEMw.d converts an amount from euros to Deutsche marks

EURTODKKw.d converts an amount from euros to Danish kroner

EURTOESPw.d converts an amount from euros to Spanish pesetas

EURTOFIMw.d converts an amount from euros to Finnish markkaa

EURTOFRFw.d converts an amount from euros to French francs

EURTOGBPw.d converts an amount from euros to British pounds

EURTOGRDw.d converts an amount from euros to Greek drachmas

EURTOHUFw.d converts an amount from euros to Hungarian forints

EURTOIEPw.d converts an amount from euros to Irish pounds

EURTOTLw.d converts an amount from euros to Italian lire

EURTOLUFw.d converts an amount from euros to Luxembourg francs

EURTONLGw.d converts an amount from euros to Dutch guilders

EURTONOKw.d converts an amount from euros to Norwegian krone

EURTOPLZw.d converts an amount from euros to Polish zlotys

EURTOPTEw.d converts an amount from euros to Portuguese escudos

EURTOROLw.d converts an amount from euros to Romanian lei

EURTORURw.d converts an amount from euros to Russian rubles

EURTOSEKw.d converts an amount from euros to Swedish kronor

EURTOSITw.d converts an amount from euros to Slovenian tolar

