CHAPTER

15

Informats

Informats in the CMS Environment

In general, informats are completely portable. Only the informats that have host-specific behavior are documented in this section.

All of these informats are described in *SAS Language Reference: Dictionary*; that information is not repeated here. Instead, each format description includes “CMS specifics” information that tells how the informat behaves under CMS. Then you are referred to *SAS Language Reference: Dictionary*.

Considerations for Using Informats under CMS

EBCDIC and Character Data

The following character informats produce different results on different computing platforms, depending on which character-encoding system the platform uses. Because CMS uses the EBCDIC character-encoding system, all of the following informats convert data to EBCDIC.

These informats are not discussed in detail in this section because the EBCDIC character-encoding system is their only host-specific aspect.

$ASCII*w*.
converts ASCII character data to EBCDIC character data.

$BINARY*w*.
: converts binary values to EBCDIC character data.

$CHAR*w*.
: reads character data with blanks.

$CHARZB*w*.
: reads character data and converts any byte that contains a binary zero to a blank.

$EBCDIC*w*.
: converts character data to EBCDIC. Under CMS, $EBCDIC and $CHAR are equivalent.

$HEX*w*.
: converts hexadecimal data to EBCDIC character data.

$OCTAL*w*.
: converts octal data to EBCDIC character data.

$PHEX*w*.
: converts packed hexadecimal data to EBCDIC character data.

$VARYING*w*.
: reads character data with blanks.

$*w*.
: reads standard character data.

All the information that you need in order to use these informats under CMS is included in *SAS Language Reference: Dictionary*.

Floating-Point Number Format and Portability

The manner in which CMS stores floating-point numbers can affect your data. See “Representation of Floating-Point Numbers” on page 111 for details.

Reading Binary Data

If a SAS program that reads binary data is run on only one type of machine, you can use the following native-mode informats. Native mode means that these informats use the byte-ordering system and floating-point representation that is standard for the machine.

IB*w.d*	reads integer binary (fixed-point) values, including negative values, that are represented in twos complement notation
PD*w.d*	reads data that are stored in the IBM packed decimal format
PIB*w.d*	reads positive integer binary (fixed-point) values
RB*w.d*	reads real binary (floating-point) data.
ZD*w.d*	reads zoned decimal data

If you want to write SAS programs that can be run on multiple machines that use different byte-storage systems, then use the following IBM 370 informats:

S370FIB
: reads integer binary data in the IBM mainframe format

S370FIBU
: reads unsigned integer binary data in the IBM mainframe format

S370FPD
: reads packed decimal data in the IBM mainframe format

S370FPDU
: reads unsigned packed decimal data in the IBM mainframe format

S370FPIB
: reads positive integer binary data in the IBM mainframe format

S370FRB
: reads real binary data in the IBM mainframe format

S370FZD
: reads zoned decimal data in the IBM mainframe format

S370FZDL
: reads zoned decimal leading sign data in the IBM mainframe format

S370FZDS
: reads zoned decimal separate leading sign data in the IBM mainframe format

S370FZDT
: reads zoned decimal separate trailing sign data in the IBM mainframe format

S370FZDU
: reads unsigned zoned decimal data in the IBM mainframe format.

These IBM 370 informats enable you to write SAS programs that can be run in any SAS environment, regardless of the standard for storing numeric data. They also enhance your ability to port raw data between host operating environments.

For more information about the IBM 370 informats, see *SAS Language Reference: Dictionary*.

Date and Time Informats

Several informats are designed to read time and date stamps that have been written by the System Management Facility (SMF), or by the Resource Management Facility (RMF). SMF and RMF are standard features of the OS/390 operating environment. They record information about each job that is processed. The SAS System under CMS can be used to analyze SMF and RMF data from an OS/390 system.

The following informats are used to read time and date stamps that are generated by SMF and RMF:

PDTIME*w*.
: reads the packed decimal time of SMF and RMF records.

RMFDUR.
: reads the duration values of RMF records.

RMFSTAMP*w*.
: reads the time and date fields of RMF records.

SMFSTAMP*w*.
: reads the time and date of SMF records.

TODSTAMP.
: reads the 8-byte time-of-day stamp.

TU*w*.
: reads Timer Unit values.

In order to facilitate the portability of SAS programs, you may use these informats with any operating environment that is supported by the SAS System; therefore, they are documented in *SAS Language Reference: Dictionary*.

Column-Binary Informats

Four informats read data from column-binary files:

$CB*w*.	reads standard character data from column-binary files.
CB*w*.	reads standard numeric values from column-binary files.
PUNCH.*d*	reads whether a row of column-binary data is punched.
ROW*w.d*	reads a column-binary field down a card column.

Data that are stored in column-binary form have usually been read into the SAS System from punch cards. Although column-binary files are not unique to CMS, that method of storing punch card data was used extensively on IBM 370 computer systems, and many files that were originally stored on punch cards are still in use today.

The $CB, CB, PUNCH, and ROW informats are completely portable. They enable SAS programs to read data that are stored in column-binary format regardless of which operating environment your site is running under. The only CMS-specific aspect of these informats is the historical relationship between the Hollerith card-coding system and IBM. See *SAS Language Reference: Dictionary* for complete information about the column-binary informats.

E

Reads numeric values that are stored in scientific notation

Numeric

Width range: 7– 32

Default width: 12

Decimal range: 0– 31

CMS specifics: interprets input as EBCDIC, minimum and maximum values

Syntax

E*w.d*

w
specifies the field width of the input value.

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

Details

Numbers are interpreted using the EBCDIC character-encoding system, with one digit per byte. The range of acceptable values is 5.398E-79 to 7.237E75. Any number outside this range causes an overflow error. The following table illustrates the use of the E informat.

EBCDIC Data Pattern Read	Informat	Numeric Value	Scientific Notation
’40F14BF2F3F0C54EF0F2’x	e10.	123	1.230E+02
’60F14BF2F3F0C54EF0F2’x	e10.	-123	-1.230E+02
’40F14BF2F3F0C54EF0F1’x	e10.	12.3	1.230E+01
’40F14BF2F3F5C54EF0F8’x	e10.	123500000	1.235E+08
’40F14BF2C54EF0F8’x	e8.	120000000	1.2E+08

Note:

- ’40’x=blank
- ’F1’x=1, ’F2’x=2, and so on
- ’4B’x=decimal point
- ’C5’x=E
- ’4E’x=plus sign
- ’60’x=minus sign

△

See Also

- Format: “E” on page 128
- *SAS Language Reference: Dictionary*.

HEX

Converts hexadecimal positive binary values to either integer (fixed-point) or real (floating-point) binary values

Numeric

Width range: 1– 16

Default width: 8

CMS specifics: IBM floating-point format

Syntax

HEX*w*.

w

specifies the field width of the input value and determines whether the input represents an integer or real binary value.

Details

Each hexadecimal digit that is read by the HEX informat is interpreted using the EBCDIC character-encoding system, with one digit per byte. For example, the hexadecimal value ’F1’x (EBCDIC 1) results in the number 1.0 using the HEX1. informat.

When you specify a width value of 1 through 15, the input hexadecimal number represents an integer binary value. When you specify 16 for the field width, the input hexadecimal number represents a real binary number.

For example, suppose a floating-point number has been stored as the following hexadecimal character value:

```
433E800000000000
   ('F4F3F3C5F8F0F0F0F0F0F0F0F0F0F0F0'x)
```

If you read the value using HEX16. as the informat, the number is converted to its correct floating-point value of 1000. However, if you specify HEX15. as the informat, SAS expects the input to represent an integer binary value. Since the number was originally stored in IBM floating-point format, the second result is incorrect.

If a number has been stored as a character representation of an integer binary value, it is correct to use a *w* value of less than 16. For example, if 256 has been stored as the following hexadecimal character value, then using HEX6. gives the correct result:

```
000100 ('F0F0F0F1F0F0'x)
```

However, using HEX16. produces an incorrect result of 3.38E-80 in the preceding example.

You can use the HEX informat to read negative floating-point numbers; however, it cannot be used to read negative integer binary numbers. For example, if you write -10 using the HEX16. format, it is stored in a character representation of a floating-point format, as follows:

```
C1A0000000000000
   ('C3F1C1F0F0F0F0F0F0F0F0F0F0F0F0F0'x)
```

If this character representation of a floating-point number is then read with the HEX16. informat, the value is correctly interpreted as the number -10.

However, if you write -10 as an integer binary number using the HEX2. format, it is written in EBCDIC as the hexadecimal string F6, which is the twos complement notation for -10. If you then read this value using HEX2. as the informat, it is read as a positive number, giving a result of 246.

See Also

- “Representation of Floating-Point Numbers” on page 111
- Format: “HEX” on page 129
- *SAS Language Reference: Dictionary*.

IB

Reads integer binary (fixed-point) values, including negative values

Numeric

Width range: 1– 8

Default width: 4

Decimal range: 0– 10

CMS specifics: twos complement notation

Syntax

IB*w.d*

w
specifies the field width of the input value.

d
specifies a divisor for the input value. If the informat includes a *d* value, the input value is divided by 10^d.

Details

The integer binary values include negative values represented in twos complement notation. The following table shows several examples of the IB*w.d* informat.

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
’000004D2’x	1234	ib4.	1234
’FFFFFB2E’x	-1234	ib4.	-1234
’0000000C’x	12	ib4.	12
’000000003034’x	12340	ib6.2	123.4
’00000001E208’x	123400	ib6.2	1234
’00000012D450’x	1234000	ib6.2	12340

See Also

- Format: “IB” on page 130
- *SAS Language Reference: Dictionary*.

PD

Reads data stored in IBM packed decimal format into a floating-point number

Numeric

Width range: 1– 16

Default width: 1

Decimal range: 0– 10

CMS specifics: IBM packed decimal format

Syntax

PD*w.d*

w
specifies the field width of the input value.

d
specifies a divisor for the input value. If the informat includes a *d* value, the input value is divided by 10^d.

Details

In packed decimal format, each byte represents two decimal digits. An IBM packed decimal number consists of a sign and up to 31 digits, thus giving a range from 10^{31} -1 to 10^{-31} +1. The sign is written in the rightmost nibble, with a ’C’x indicating a positive value and a ’D’x indicating a negative value. The rest of the nibbles to the left of the sign nibble represent decimal digits. The hexadecimal values of these digit nibbles correspond to decimal values; therefore, only values between ’0’x and ’9’x can be used in the digit positions. The following table shows several examples of how data is read using the PD*w.d* informat.

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
’01234C’x	1234	pd3.	1234
’01234D’x	-1234	pd3.	-1234
’0001234C’x	1234	pd4.	1234
’0012340C’x	12340	pd4.1	1234
’0123400C’x	123400	pd4.2	1234

See Also

- Format: “PD” on page 131
- *SAS Language Reference: Dictionary*.

RB

Reads real binary (floating-point) data into a floating-point number

Numeric

Width range: 2– 8

Default width: 4

Decimal range: 0– 10

CMS specifics: IBM floating-point format

Syntax

RB*w.d*

w
: specifies the field width of the input value.

d
: specifies a divisor for the input value. If the informat includes a *d* value, the input value is divided by 10^d.

Details

The format of floating-point numbers is specific to CMS. (See “Representation of Floating-Point Numbers” on page 111 for a description of the format used to store floating-point numbers.) The following table shows how data that represent several decimal numbers are read as floating-point numbers using the RB*w.d* informat.

Data Pattern Read	**Actual Numeric Value**	**Informat**	**Resulting Numeric Value**
’427B000000000000’x	123	rb8.	123
’434CE00000000000’x	1230	rb8.1	123
’44300C0000000000’x	12300	rb8.2	123
’C27B000000000000’x	-123	rb8.	-123
’434D200000000000’x	1234	rb8.	1234
’41C570A3D70A3D70’x	12.34	rb8.	12.34

See Also

- □ “Representation of Floating-Point Numbers” on page 111
- □ Format: “RB” on page 132
- □ *SAS Language Reference: Dictionary*.

w.d

Reads standard numeric data

Numeric

Width range: 1– 32

Decimal range: 0– 31, $d < w$.

CMS specifics: minimum and maximum values

Syntax

w.d

w
specifies the field width of the input value in bytes or character digits.

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

Details

Numbers are interpreted using the EBCDIC character-encoding system, with one digit per byte. The *w.d* informat reads numeric values located anywhere in the field. Blanks can precede or follow a numeric value with no effect. A minus sign with no separating blanks must immediately precede a negative value. Data can be stored with decimal points or in scientific notation.

Include a *d* value in the *w.d* informat when you want SAS to insert a decimal point. When you include a *d* value, values read without decimal points are divided by 10^d. If the value read already has a decimal point, the *d* is ignored.

The range of acceptable values that can be read with the *w.d* informat can range from 5.398E-79 to 7.237E+75. The following table illustrates the use of the *w.d* informat.

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
’F1F2F3F4’x	1234	4.	1234
’40F1F2F3F4’x	1234	5.	1234
’F1F24BF5F0’x	12.50	5.	12.50
’F1F24BF5F0’x	12.50	5.1	12.50
’F1F2F5F040’x	1250	5.1	125
’40F1F0F0F0’x	1000	5.1	100
’4060F1F2F3F4’x	-1234	6.	-1234
’40F14BF0C54EF0F3’x	1.0E+03	8.	1000

Note:

- ’40’x=blank
- ’F1’x=1, ’F2’x=2, and so on
- ’4B’x=decimal point
- ’C5’x=E
- ’4E’x=plus sign
- ’60’x=minus sign

△

See Also

- Format: "w.d" on page 133
- *SAS Language Reference: Dictionary*.

ZD

Reads zoned decimal data

Numeric

Width range: 1– 32

Default width: 1

Decimal range: 0– 31

CMS specifics: IBM zoned decimal format

Syntax

ZD*w.d*

w
specifies the field width of the input value.

d
specifies a divisor for the input value. If the informat includes a *d* value, the input value is divided by 10^{d}.

Details

Like numbers stored in standard format, zoned decimal digits are represented as EBCDIC characters. Each digit requires one byte of storage space. The rightmost byte represents both the least significant digit and the sign of the number. Digits to the left of the least significant digit are represented as the EBCDIC characters 0 through 9. The character printed for the least significant digit depends on the sign of the number.

In the least significant byte, negative numbers are coded with the high-order nibble being a 'D'x and with the low-order nibble being the numeric value. Positive numbers are represented with the high-order nibble being a 'C'x. For example, compare the zoned decimal data for 123 and -123 in the following table.

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
'F0F0F0F0F0F1F2C3'x	123	zd8.	123
'F0F0F0F0F1F2F3C4'x	1234	zd8.	1234
'F0F0F0F0F1F2F3C0'x	1230	zd8.1	123
'F0F0F0F1F2F3F0C0'x	12300	zd8.2	123
'F0F0F0F0F0F1F2D3'x	-123	zd8.	-123

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
'F0F0F0F0F0F1F2C3'x	123	zd8.6	0.000123
'F0F0F0F0F1F2F3C0'x	1230	zd8.6	0.00123
'F0F0F0F0F0F0F0C1'x	1	zd8.6	1E-6

See Also

- □ Informat: "ZDB" on page 174
- □ Format: "ZD" on page 135
- □ *SAS Language Reference: Dictionary*

ZDB

Reads zoned decimal data with blanks

Numeric

Width range: 0– 31

Default width: 1

Decimal range: none

CMS specifics: used on IBM 1410, 1401, and 1620

Syntax

ZDB*w.d*

w
specifies the field width of the input value, in bytes.

d
specifies a divisor for the input value. If the informat includes a *d* value, the input value is divided by 10^d.

Details

The ZDB informat reads zoned decimal data that are produced in IBM 1410, 1401, and 1620 form, in which zeros are left blank rather than being written. Each digit is represented as an EBCDIC character as previously described for the ZD informat. The only differences are the way in which zeros are represented, and that the ZDB informat does not allow you to use a *d* value while ZD does. The ZDB informat treats EBCDIC blanks ('40'x) as zeros. (EBCDIC zeros are also read as zeros.) Like the ZD informat, the ZDB informat also uses the rightmost byte to represent both the least significant digit and the sign.

The table under "ZD" on page 173 shows the convention that is used to store digits in zoned decimal format. The following table shows several examples of how the ZDB informat reads data.

Data Pattern Read	Actual Numeric Value	Informat	Resulting Numeric Value
'40404040404040C1'x	1	zdb8.	1
'404040404040F1C0'x	10	zdb8.	10
'4040404040F140C0'x	100	zdb8.	100
'40404040F14040C0'x	1000	zdb8.	1000
'4040404040F1F2D3'x	-123	zdb8.	-123
'4040404040F1F2C3'x	123	zdb8.	123

See Also

- Informat: "ZD" on page 173
- Format: "ZD" on page 135
- *SAS Language Reference: Dictionary*

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the CMS Environment, Version 8* , Cary, NC: SAS Institute Inc., 1999.

SAS® Companion for the CMS Environment, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1–58025–481–0

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

1st printing, October 1999