CHAPTER 15 Informats

SAS Informats under OpenVMS

A SAS informat is an instruction or template that the SAS System uses to read data values into a variable. Most SAS informats are described completely in *SAS Language Reference: Dictionary*. The informats that are described here have behavior that is specific to the SAS System under OpenVMS.

Many of the SAS informats that have details that are specific to the OpenVMS operating environment are used to read binary data. For more information, see "Reading Binary Data" on page 321.

Reading Binary Data

Different computers store numeric binary data in different forms. IBM 370 and Hewlett-Packard 9000 computers store bytes in one order. Microcomputers that are IBM compatible and some computers manufactured by Digital Equipment Corporation store bytes in a different order called *byte-reversed*.

Binary data that are stored in one order cannot be read by a computer that stores binary data in the other order. When you are designing SAS applications, try to anticipate how your data will be read and choose your formats and informats accordingly.

The SAS System provides two sets of informats for reading binary data and corresponding formats for writing binary data.

- The IB*w.d*, PD*w.d*, PIB*w.d*, and RB*w.d* informats and formats read and write in native mode, that is, using the byte-ordering system that is standard for the machine.
- The S370FIB*w.d*, S370FPD*w.d*, S370FRB*w.d*, and S370FPIB*w.d* informats and formats read and write according to the IBM 370 standard, regardless of the native mode of the machine. These informats and formats allow you to write SAS programs that can be run in any SAS environment, regardless of how numeric data are stored.

If a SAS program that reads and writes binary data runs on only one type of machine, you can use the native mode informats and formats. However, if you want to write SAS programs that can be run on multiple machines using different byte-storage systems, use the IBM 370 formats and informats. The purpose of the IBM 370 informats and formats is to enable you to write SAS programs that can be run in any SAS environment, no matter what standard you use for storing numeric data.

For example, suppose you have a program that writes data with the PIB*w.d* format. You execute the program on a microcomputer so that the data are stored in byte-reversed mode. Then you run another SAS program on the microcomputer that uses the PIB*w.d* informat to read the data. The data are read correctly because both of the programs are run on the microcomputer using byte-reversed mode. However, you cannot upload the data to a Hewlett-Packard 9000-series machine and read the data correctly because they are stored in a form native to the microcomputer but foreign to the Hewlett-Packard 9000. To avoid this problem, use the S370FPIB*w.d* format to write the data; even on the microcomputer, this causes the data to be stored in IBM 370 mode. Then read the data using the S370FPIB*w.d* informat. Regardless of what type of machine you use when reading the data, they are read correctly.

HEX*w*.

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

Language element: informat

Category: numeric

Width range: 1 to 16

Default width: 8

OpenVMS specifics: ASCII character-encoding system

Syntax

HEX*w*.

w

specifies the field width of the input value. If you specify a *w* value of 1 through 15, the input hexadecimal value is converted to an fixed-point number. If you specify 16 for the *w* value, the input hexadecimal value is converted to a floating-point number.

Details

Under OpenVMS, the hexadecimal format of the number is stored in ASCII representation.

For more information about OpenVMS floating-point representation, see *Architecture Reference Manual for Alpha* and *Architecture Reference Manual for VAX*.

See Also

- □ Informats: HEX*w*. in *SAS Language Reference: Dictionary* and “$HEX*w*.” on page 323
- □ Format: “HEX*w*.” on page 264

$HEX*w*.

Converts hexadecimal data to character data

Language element: informat

Category: character

Width range: 1 to 32767

Default width: 2

OpenVMS specifics: ASCII character-encoding system

Syntax

$HEX*w*.

w

specifies width of the input value.

Details

The $HEX*w*. informat is similar to the HEX*w*. informat. The $HEX*w*. informat converts character values that are stored as the hexadecimal equivalent of ASCII character codes to the corresponding character values. The conversion is based on the ASCII character-encoding system.

In the ASCII system, the conversion for 8-bit hexadecimal input (x’80’ and above) is for the Multinational Character Set, which includes national characters such as Ä and ß. For more information about the Multinational Character Set, see *Guide to Using OpenVMS*.

See Also

- □ Informats: $HEX*w*. in *SAS Language Reference: Dictionary* and “HEX*w*.” on page 322
- □ Format: “$HEX*w*.” on page 265

IB*w.d*

Reads integer-binary (fixed-point) values

Language element: informat

Category: numeric

Width range: 1 to 8

Default width: 4

Decimal range: 0 to 10

OpenVMS specifics: native twos-complement notation

Syntax

IB*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the IB*w.d* informat divides the input value by the 10^d value. SAS uses the *d* value, even if the input data contain decimal points.

Details

The IB*w.d* informat reads integer-binary (fixed-point) values that are represented in the OpenVMS twos-complement notation. For information about OpenVMS native fixed-point values, see *Architecture Reference Manual for Alpha* and *Architecture Reference Manual for VAX*.

See Also

- Informat: IB*w.d* in *SAS Language Reference: Dictionary*
- Format: “IB*w.d*” on page 265
- “Reading Binary Data” on page 321

PD*w.d*

Reads packed decimal data

Language element: informat

Category: numeric

Width range: 1 to 16

Default width: 1

Decimal range: 0 to 10

OpenVMS specifics: How values are interpreted as negative or positive

Syntax

PD*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the PD*w.d* informat divides the input value by the 10^d value. If the data contain decimal points, then SAS ignores the *d* value.

Details

Under OpenVMS, the least significant 4 bits of the least significant byte in the PD value are interpreted as follows:

- If the hexadecimal number in the 4 bits is D or F, then the value is interpreted as negative.
- If the hexadecimal number in the 4 bits is C, then the value is interpreted as positive.

See Also

- Informat: PD*w.d* in *SAS Language Reference: Dictionary*
- Format: "PD*w.d*" on page 266
- "Reading Binary Data" on page 321

PIB*w.d*

Reads positive integer-binary (fixed-point) values

Language element: informat

Category: numeric

Width range: 1 to 8

Default width: 1

Decimal range: 0 to 10

OpenVMS specifics: native integer-binary values; byte reversal

Syntax

PIB*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the PIB*w.d* informat divides the input value by the 10^d value. SAS uses the *d* value even if the input data contain decimal points.

Details

Under OpenVMS, the PIB informat reads native integer-binary values. However, this informat ignores the negativity of data in twos-complement notation and reads it as positive. For more information about OpenVMS native fixed-point values, see *Architecture Reference Manual for Alpha* and *Architecture Reference Manual for VAX*.

See Also

- □ Informat: PIB*w.d* in *SAS Language Reference: Dictionary*
- □ Format: "PIB*w.d*" on page 267
- □ "Reading Binary Data" on page 321

RB*w.d*

Reads real-binary (floating-point) data

Language element: informat

Category: numeric

Width range: 2 to 8

Default width: 4

Decimal range: 0 to 10

OpenVMS specifics: native floating-point representation

Syntax

RB*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the RB*w.d* informat divides the input value by the 10^d value. SAS uses the *d* value even if the input data contain decimal points.

Details

The RB*w.d* informat reads numeric data that are stored in native real-binary (floating-point) notation. Numeric data for scientific calculations are often stored in floating-point notation. (The SAS System stores all numeric values in floating-point notation.) A floating-point value consists of two parts: a mantissa that gives the value and an exponent that gives the value's magnitude.

It is usually impossible to key in floating-point binary data directly from a terminal, but many programs write floating-point binary data. Use caution if you are using the RB*w.d* informat to read floating-point data created by programs other than the SAS System because the RB*w.d* informat is designed to read only double-precision data.

Since the RB*w.d* informat is designed to read only double-precision data, it supports widths of less than 8 bytes only for those applications that truncate numeric data for space-saving purposes. RB4. does *not* expect a single-precision number that is truncated to 4 bytes.

External programs such as those written in C and FORTRAN can produce only single- or double-precision floating-point numbers. No length other than 4 or 8 bytes is allowed. The RB*w.d* informat allows a length of 3 through 8 bytes, depending on the storage you need to save.

The FLOAT4. informat has been created to read a single-precision, floating-point number. If you read the hexadecimal notation **3F800000**with FLOAT4., the result is a value of 1.

To read data created by a C or FORTRAN program, you need to decide on the proper informat to use. If the floating-point numbers require an 8-byte width, you should use the RB8. informat. If the floating-point numbers require a 4-byte width, you should use FLOAT4.

For more information about OpenVMS floating-point representation, see *Architecture Reference Manual for Alpha* and *Architecture Reference Manual for VAX*.

Examples

Example 1: Single- vs. Double-Precision Representation Consider how the value of 1 is represented in single-precision and double-precision notation. For single-precision, the hexadecimal representation of the 4 bytes of data is **3F800000**. For double-precision, the hexadecimal representation is **3FF0000000000000**. The digits at the beginning of the data are different, indicating a different method of storing the data.

Example 2: Reading External Data Consider this C example:

```
#include <stdio.h>

main() {

FILE *fp;
float x[3];

fp = fopen(‘‘test.dat’’,’’wb’’);
x[0] = 1; x[1] = 2; x[2] = 3;

fwrite((char *)x,sizeof(float),3,fp);
fclose(fp);
}
```

The file TEST.DAT will contain, in hexadecimal notation, **3f8000004000000040400000**.

See Also

- □ Informat: RB*w.d* in *SAS Language Reference: Dictionary*
- □ Format: "RB*w.d*" on page 268
- □ "Reading Binary Data" on page 321

VMSTIME.

Converts an 8-byte binary value that is in OpenVMS date and time format to a SAS date-time value

Language element: informat

Category: date and time

Width range: 8

Default width: 8

OpenVMS specifics: All aspects are host-specific

Syntax

VMSTIME.

Details

The VMSTIME. informat is specific to the OpenVMS operating environment. You cannot specify a width with this informat; the width is always 8 bytes.

OpenVMS date and time values that are read in with the VMSTIME. informat retain precision up to 1/100 of a second, even though the SAS System cannot display anything less than whole seconds. If you later use the VMSTIMEF. format to write out the date-time value, the precision is retained.

See Also

- □ Format: "VMSTIMEF." on page 271

VMSZN*w.d*

Reads VMS zoned numeric data

Language element: informat

Category: numeric

Width range: 1 to 32

Default width: 1

OpenVMS specifics: All aspects are host-specific

Syntax

VMSZN*w.d*

w
specifies the width of the output field.

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

Details

The VMSZN*w.d* informat is similar to the ZD*w.d* informat. Both read a string of ASCII digits, and the last digit is a special character denoting the magnitude of the last digit and the sign of the entire number. The difference between the VMSZN*w.d* informat and the ZD*w.d* informat is in the special character used for the last digit. The following table shows the special characters used by the VMSZN*w.d* informat.

Desired Digit	Special Character	Desired Digit	Special Character
0	0	-0	p
1	1	-1	q
2	2	-2	r
3	3	-3	s
4	4	-4	t
5	5	-5	u
6	6	-6	v
7	7	-7	w
8	8	-8	x
9	9	-9	y

Data formatted using the VMSZN*w.d* informat are ASCII strings.

Examples

If you format the ASCII string 1234 using the VMSZN*w.d* informat in the following SAS statement:

```
input @1 vmszn4.;
```

the result is 1234.

If you format the ASCII string 123t using the VMSZN*w.d* informat in the following SAS statement:

```
input @1 vmszn4.;
```

the result is -1234.

See Also

- Informats: ZD*w.d* in *SAS Language Reference: Dictionary* and "ZD*w.d*" on page 330
- Format: "VMSZN*w.d*" on page 271

ZD*w.d*

Reads zoned decimal data

Language element: informat

Category: numeric

Width range: 1 to 32

Default width: 1

OpenVMS specifics: the last byte includes the sign

Syntax

ZD*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the ZD*w.d* informat divides the input value by the 10^d value. If the data contain decimal points, then SAS ignores the *d* value.

Details

The ZD*w.d* informat accepts true zoned decimal, trailing numeric strings with the overpunch format; numeric strings of the form that is read by the standard numeric informat; and hybrid strings. A hybrid string is a zoned decimal string that has an explicit sign.

To achieve the same results as the Release 6.06 implementation of the ZD*w.d* informat, use the ZDV*w.d* informat.

A *zoned decimal*, or trailing numeric string with overpunch format, is a character string consisting of digits. The last character of the string is a special character that specifies both the value of the last digit and the sign of the entire number. The special characters are listed in the following table.

Digit	ASCII Character	Digit	ASCII Character
0	{	-0	}
1	A	-1	J
2	B	-2	K

Digit	ASCII Character	Digit	ASCII Character
3	C	-3	L
4	D	-4	M
5	E	-5	N
6	F	-6	O
7	G	-7	P
8	H	-8	Q
9	I	-9	R

The data formatted using the ZD*w.d* informat are ASCII strings.

Examples

If you format the ASCII string 123{ using ZD*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is 1230.

If you format the ASCII string 123} using the ZD*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is -1230.

If you format the ASCII string 1230 using the ZD*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is 1230.

If you format the ASCII string -1230 using the ZD*w.d* informat in the following SAS statement:

```
input i zd5.;
```

the result is -1230.

If you format the ASCII string +123{ using the ZD*w.d* informat in the following SAS statement:

```
input i zd5.;
```

the result is 1230.

See Also

- Informats: ZD*w.d* in *SAS Language Reference: Dictionary* and “ZDV*w.d*” on page 332

ZDV*w.d*

Reads zoned decimal data

Language element: informat

Category: Numeric

Width range: 1 to 32

Default width: 1

OpenVMS specifics: All aspects are host-specific

Syntax

ZDV*w.d*

w
specifies the width of the input field.

d
optionally specifies the power of 10 by which to divide the input value. If you specify *d*, the ZDV*w.d* informat divides the input value by the 10^d value. If the data contain decimal points, then SAS ignores the *d* value.

Details

In Release 6.07 of the SAS System, the ZDV*w.d* informat replaced the ZD*w.d* informat with one change: although the string must be a true trailing numeric with overpunch format, it no longer needs to be right-justified within its field. Trailing blanks are trimmed before the number is converted.

A *zoned decimal*, or trailing numeric string with overpunch format, is a character string that consists of digits. The last character of the string is a special character that specifies both the value of the last digit and the sign of the entire number. The special characters are the same as those that are documented for the ZD*w.d* informat, as shown in the following table.

	ASCII		ASCII
Digit	**Character**	**Digit**	**Character**
0	{	-0	}
1	A	-1	J
2	B	-2	K
3	C	-3	L
4	D	-4	M

	ASCII		ASCII
Digit	**Character**	**Digit**	**Character**
5	E	-5	N
6	F	-6	O
7	G	-7	P
8	H	-8	Q
9	I	-9	R

The data formatted using the ZDV*w.d* informat are ASCII strings.

Examples

If you format the ASCII string 123{ using ZDV*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is 1230.

If you format the ASCII string 123} using the ZDV*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is -1230.

If you format the ASCII string 1230 using the ZDV*w.d* informat in the following SAS statement:

```
input i zd4.;
```

the result is invalid data.

If you format the ASCII string -1230 using the ZDV*w.d* informat in the following SAS statement:

```
input i zd5.;
```

the result is invalid data.

See Also

- Informats: ZD*w.d* in *SAS Language Reference: Dictionary* and “ZD*w.d*” on page 330

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OpenVMS Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. 518 pp.

SAS® Companion for the OpenVMS Environment, Version 8

1–58025–526–4

1st printing, October 1999

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