CHAPTER
18

Informats

SAS Informats under Windows

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 Windows.

Many of the SAS informats that have details specific to the Windows operating environment are used to read binary data. In using these informats, it is important that you understand the concepts that are presented in "Reading Binary Data" on page 345.

If you have informats that you created for use in earlier releases of the SAS System, see "Converting User-Written Informats from Earlier Releases to Version 8" on page 346 for information about how to convert those informats for use with Version 8.

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 compatible with IBM microcomputers and some computers manufactured by Digital Equipment Corporation store bytes in a different order called *byte-reversed*.

Binary data 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 on the microcomputer you run another SAS program that uses the PIB*w.d* informat to read the data. The data are read correctly because both 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.

Converting User-Written Informats from Earlier Releases to Version 8

You must convert Release 6.04, Release 6.06, and Release 6.08 user-written informats and formats to their Version 8 counterparts before you can use them in a Version 8 SAS program. The only exception to this rule is user-written informats and formats created by Release 6.08 or later under Windows; these informats and formats can be read directly from your Windows SAS session. *

Converting Version 6 User-Written Informats

You can convert Release 6.04, 6.06, and 6.08 SAS catalogs that contain user-written informats and formats by using one of the following methods:

Converting Release 6.04 catalogs
: use the CNTLOUT= option in the PROC FORMAT statement in Release 6.04 to create an output data set, and then use the CNTLIN= option in the PROC FORMAT statement in Version 8 to create the Version 8 informats or formats. You must use the V604 engine in your Version 8 SAS session to read the data set. This method also works for converting from Release 6.06 or Release 6.08.

Converting Release 6.06 or Release 6.08 catalogs
: use the CPORT and CIMPORT procedures to convert the informats and formats. For more information about the CPORT and CIMPORT procedures, see *SAS*

* However, it is recommended that you use PROC CPORT and PROC CIMPORT to convert older Windows catalogs that contain user-written informats and formats to Version 8 if you no longer need to use them in previous releases.

Procedures Guide. This method works for converting from Release 6.06 or Release 6.08 only; it does not work for converting from Release 6.04.

Converting Version 5 User-Written Informats

You must also convert Version 5 user-written informats and formats to their Version 8 counterparts before you can use them in a Version 8 SAS program. (This implies that you are not only converting these files, but are also transferring them from a remote operating system to your PC). You can convert them using one of the following methods:

- Use the V5TOV6 procedure on the remote operating environment to convert the informats and formats to Version 6 format. This implies that the remote operating environment has access to Version 6 SAS software. Then, transport the converted informats and formats (as binary files) to your Windows operating environment and use the CIMPORT procedure to complete the conversion.

  Note: The V5TOV6 procedure is not available in Version 8 of the SAS System. You must use this procedure in Release 6 of the SAS System. △
- Use the SUGI supplemental procedure FMTLIB under Version 5 on the remote operating environment to create an output data set, transport that data set to your PC, and then use the CNTLIN= option in the PROC FORMAT statement in Version 8 to create the Version 8 informats or formats.

HEX*w*.

Converts hexadecimal positive binary values to fixed-point or floating-point binary values

Category numeric
Width range: 1–16
Default width: 8
Alignment: left
Windows specifics: native floating-point representation

Syntax

HEX*w*.

w
specifies whether the input represents an integer (fixed-point) or a real (floating-point) binary number. When you specify a *w* value of 1 through 15, the input hexadecimal value represents an integer binary number. When you specify 16 for the *w* value, the input hexadecimal value represents a floating-point value.

Details

The HEX*w*. informat expects input that is not byte-reversed, that is, not in Windows form. (The IB, PIB, and RB informats for binary numbers expect the bytes to be reversed.) This means that you can use the HEX*w*. informat to read hexadecimal literals from SAS programs that were created in another environment.

See Also

- Informats: HEX*w*. in *SAS Language Reference: Dictionary* and "$HEX*w*." on page 348
- Format: "HEX*w*." on page 319

$HEX*w*.

Converts hexadecimal data to character data

Category character
Width range: 1–32767
Default width: 2
Alignment: left
Windows specifics: ASCII character-encoding system

Syntax

$HEX*w*.

w
specifies width of the input value.

Details

The $HEX*w*. informat is like the HEX*w*. informat in that it reads values in which each hexadecimal digit occupies 1 byte. Use the $HEX*w*. informat to encode hexadecimal information into a character variable when your input data are limited to printable characters. The conversion is based on the ASCII character set.

See Also

- Informats: $HEX*w*. in *SAS Language Reference: Dictionary* and "HEX*w*." on page 347
- Format: "$HEX*w*." on page 320

IB*w.d*

Reads integer binary (fixed-point) data

Category numeric
Width range: 1–8
Default width: 4
Decimal range: 0–10
Windows specifics: native floating-point representation

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. SAS uses the *d* value even if the input data contain decimal points.

Details

For integer binary data, the high-order bit is the value's sign: 0 for positive values, 1 for negative. Negative values are represented in twos-complement notation. If the informat includes a *d* value, the data value is divided by 10^d.

Using the IB*w.d* informat requires you to understand twos complements and byte-swapped data format.

For more information about microcomputer fixed-point values, see Intel Corporation's *i486 Microprocessor Programmer's Reference Manual*.

Comparison of IB and PIB

The IB*w.d* informat and the PIB*w.d* informat give you different results. The IB*w.d* informat processes both positive and negative numbers and it uses the high-order bit as the sign bit. In contrast, the PIB*w.d* informat is used only for positive numbers and it does not look for a sign bit. As an example, suppose your data contain the following two-byte (byte-swapped) value:

```
01 80
```

When you read this value using the IB2. informat, the informat looks for the sign bit, sees that it is on, and reads the value as –32,767. However, if you read this value with the PIB2. informat, no sign bit is used, and the result is 32,769.

Example

Suppose that your data contain the following 6-byte (byte-swapped) value:

```
64 00 00 00 00 00
```

If you read this value using the IB6. informat, it is read as the fixed-point value 100.0. Now suppose that your data contain the following (byte-swapped) value:

```
01 80
```

Because the sign bit is set, the value is read as –32,767.

See Also

- Informat: IB*w.d* in *SAS Language Reference: Dictionary*
- Format: "IB*w.d*" on page 320
- "Reading Binary Data" on page 345

PD*w.d*

Reads packed decimal data

Category numeric

Width range: 1–16

Default width: 1

Decimal range: 0–31

Windows 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 the data contain decimal points, then SAS ignores the *d* value.

Details

In packed decimal data, each byte contains two digits. The value's sign is in the first bit of the first byte (although the entire first byte is used for the sign). Although it is usually impossible to key in packed decimal data directly from a terminal, many programs write packed decimal data. The decimal range is 1 through 31.

Example

Suppose your data contain the following packed decimal number:

```
80 00 11 43
```

If you use the PD4. informat, this value is read as the double-precision value –1143.0. Similarly, the following value is read as 1500.0:

```
00 00 15 00
```

See Also

- Informat: PD*w.d* in *SAS Language Reference: Dictionary*
- Format: "PD*w.d*" on page 321
- "Reading Binary Data" on page 345

PIB*w.d*

Reads positive integer binary (fixed-point) data

Category numeric
Width range: 1–8
Default width: 1
Decimal range: 0–10
Windows specifics: native byte-swapped integers

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. SAS uses the *d* value even if the input data contain decimal points.

Details

Positive integer binary values are the same as integer binary (see the informat "IB*w.d*" on page 348), except that all values are treated as positive. Thus, the high-order bit is part of the value rather than the value's sign.

Comparison of PIB and IB

The PIB*w.d* informat and the IB*w.d* informat give you different results, and you should differentiate carefully between these two informats. The IB*w.d* informat processes both positive and negative numbers and uses the high-order bit as the sign bit. In contrast, the PIB*w.d* informat is used only for positive numbers and it does not look for a sign bit. As an example, suppose your data contain the following two-byte (byte-swapped) value:

```
01 80
```

When you read this value using the IB2. informat, the informat looks for the sign bit, sees that it is on, and reads the value as –32,767. However, if you read this value with the PIB2. informat, no sign bit is used, and the result is 32,769.

Example

Suppose your data contain the following one-byte value:

```
FF
```

If you read this value using the PIB1. informat, it is read as the double-precision value 255.0. Using this informat requires you to understand twos complements and byte-swapped data format.

See Also

- Informat: PIB*w.d* in *SAS Language Reference: Dictionary*
- Format: "PIB*w.d*" on page 323
- "Reading Binary Data" on page 345

RB*w.d*

Reads real binary (floating-point) data

Category numeric
Width range: 2–8
Default width: 4
Decimal range: 0–10
Windows 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. 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 microcomputer 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.

See Also

- Informat: RB*w.d* in *SAS Language Reference: Dictionary*
- Format: "RB*w.d*" on page 323
- "Reading Binary Data" on page 345

ZD*w.d*

Reads zoned decimal data

Category numeric
Width range: 1–32
Default width: 1
Decimal range: 1–10
Windows specifics: 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 the data contain decimal points, then SAS ignores the *d* value.

Details

This is also known as an overprint trailing numeric format. Under Windows, the last byte of the field contains the sign information of the number. The following table gives the conversion for the last byte:

Digit	ASCII Character	Digit	ASCII Character
0	{	–0	}
1	A	–1	J
2	B	–2	K
3	C	–3	L
4	D	–4	M
5	E	–5	N
6	F	–6	O
7	G	–7	P

Digit	ASCII Character	Digit	ASCII Character
8	H	–8	Q
9	I	–9	R

See Also

- Informat: ZD*w.d* in *SAS Language Reference: Dictionary*
- Format: “ZD*w.d*” on page 324

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS Companion for the Microsoft Windows Environment, Version 8,* Cary, NC: SAS Institute Inc., 1999. pp.555.

SAS Companion for the Microsoft Windows Environment, Version 8

ISBN 1–58025–524–8

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

1st printing, September 1999

The Institute is a private company devoted to the support and further development of its software and related services.