Introduction

In general, formats are completely portable. Only the formats that have aspects specific to OS/390 are documented in this chapter. All portable formats are described in SAS Language Reference: Dictionary; that information is not repeated here. Instead, you are given details on how the format behaves in the OS/390 environment, then you are referred to SAS Language Reference: Dictionary for further details.

Considerations for Using Formats in the OS/390 Environment

EBCDIC and Character Data

The following character formats produce different results on different computing platforms, depending on which character-encoding system the platform uses. Because OS/390 uses the EBCDIC character-encoding system, all of the following formats convert data from EBCDIC.
These formats are not discussed in detail in this chapter because the EBCDIC character-encoding system is their only host-specific aspect.

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

$BINARY w.
converts EBCDIC character data to binary representation, where each character is represented by eight binary characters.

$EBCDIC w.
converts EBCDIC data to character data. Under OS/390, $EBCDIC w. and $CHAR w. are equivalent.

$HEX w.
converts EBCDIC character data to hexadecimal representation.

$OCTAL w.
converts EBCDIC character data to octal representation.

All the information that you need in order to use these formats under OS/390 is in SAS Language Reference: Dictionary.

---

**Floating-Point Number Format and Portability**

The manner in which OS/390 stores floating-point numbers can affect your data. See “Representation of Floating-Point Numbers” on page 139 for details.

---

**Writing Binary Data**

If a SAS program that writes binary data is run in only one operating environment, you can use the following native-mode formats:

* Native-mode formats use the byte-ordering system that is standard for the operating environment.
S370FFw.d
  is used on other computer systems to read EBCDIC data from IBM mainframe files.

S370FIBw.d
  writes integer binary data in IBM mainframe format.

S370FIBUw.d
  writes unsigned integer binary data in IBM mainframe format.

S370FP Dw.d
  writes packed decimal data in IBM mainframe format.

S370FPDUw.d
  writes unsigned packed decimal data in IBM mainframe format.

S370FPI Bw.d
  writes positive integer binary data in IBM mainframe format.

S370FR Bw.d
  writes real binary data in IBM mainframe format.

S370FZ Dw.d
  writes zoned decimal data in IBM mainframe format.

S370FZDLw.d
  writes zoned decimal leading sign data in IBM mainframe format.

S370FZDSw.d
  writes zoned decimal separate leading sign data in IBM mainframe format.

S370FZDTw.d
  writes zoned decimal separate trailing sign data in IBM mainframe format.

S370FZDUw.d
  writes unsigned zoned decimal data in IBM mainframe format.

These IBM 370 formats 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 formats, see SAS Language Reference: Dictionary.

---

**Dictionary**

**BESTw.**

SAS System chooses the best notation

Numeric
Details

Numbers are written using EBCDIC code with one digit per byte. Because the value is output as EBCDIC text characters, you can print it without further formatting.

The range of the magnitude of numbers is from $5.4 \times 10^{-79}$ to $7.2 \times 10^{75}$. Any number that is outside this range causes an overflow error. All numeric variables that are represented by the SAS System are within this range.

The following examples illustrate the use of BESTw. under OS/390:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>best6.</td>
<td>b1234</td>
<td></td>
</tr>
<tr>
<td>-1234</td>
<td>best6.</td>
<td>b-1234</td>
<td></td>
</tr>
<tr>
<td>12.34</td>
<td>best6.</td>
<td>b12.34</td>
<td></td>
</tr>
<tr>
<td>12345678</td>
<td>best8.</td>
<td>1.2346E8</td>
<td>truncated and rounded</td>
</tr>
</tbody>
</table>

Note: In these examples, the Value column represents the value of the SAS numeric variable. The Results column shows what the numeric output looks like when viewed from a text editor. The b characters in the Results column indicate blank spaces. See Table 9.1 on page 142 for a table of commonly used EBCDIC characters.

See Also

- Format: BESTw. in SAS Language Reference: Dictionary
The range of the magnitude of numbers is from \(5.4 \times 10^{-79}\) to \(7.2 \times 10^{75}\). Any number that is outside of this range causes an overflow error. All numeric variables that are represented by the SAS System are within this range.

The following examples illustrate the use of \(Ew\). under OS/390:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>e10.</td>
<td>(b1.230E+02)</td>
<td></td>
</tr>
<tr>
<td>-123</td>
<td>e10.</td>
<td>(-1.230E+02)</td>
<td></td>
</tr>
<tr>
<td>12.3</td>
<td>e10.</td>
<td>(b1.230E+01)</td>
<td></td>
</tr>
<tr>
<td>12345678</td>
<td>e10.</td>
<td>(b1.235E+07)</td>
<td>truncated and rounded</td>
</tr>
</tbody>
</table>

Note: In these examples, the Value column represents the value of the SAS numeric variable. The Results column shows what the numeric value looks like when viewed from a text editor. The \(b\) characters in the Results column indicate blank spaces. See Table 9.1 on page 142 for a table of commonly used EBCDIC characters.

See Also

- Format: \(Ew\). in SAS Language Reference: Dictionary
- Informat: “\(Ew.d\)” on page 206

### \(HEXw\).

**Converts real binary (floating-point) values to hexadecimal representation**

**Numeric**

- Width range: 1-16 bytes
- Default width: 8
- Alignment: left
- OS/390 specifics: writes output as EBCDIC, IBM floating-point format

**Details**

Each hexadecimal digit is written using the EBCDIC code, which requires one byte per digit. See Table 9.1 on page 142 for a table of commonly used EBCDIC characters.

The format of floating-point numbers is host-specific. See “Representation of Floating-Point Numbers” on page 139 for a description of the IBM floating-point format that is used under OS/390.

The \(w\) value of the \(HEXw\). format determines whether the number is written as a floating-point number or as an integer. When you specify a width value of 1 through 15, the real binary numbers are truncated to fixed-point integers before being converted to hexadecimal representation. When you specify 16 for the width, the floating point values are used, and the numbers are not truncated.

The following examples illustrate the use of \(HEXw\). under OS/390:
<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results (Hex)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.5</td>
<td>hex16.</td>
<td>421f800000000000</td>
<td>floating-point number</td>
</tr>
<tr>
<td>31.5</td>
<td>hex15.</td>
<td>000000000000001f</td>
<td>integer</td>
</tr>
<tr>
<td>-31.5</td>
<td>hex16.</td>
<td>c21f800000000000</td>
<td>floating-point number</td>
</tr>
<tr>
<td>-31.5</td>
<td>hex15.</td>
<td>ffffffff00000000</td>
<td>integer</td>
</tr>
</tbody>
</table>

Note: In these examples, the Value column represents the value of the SAS numeric variable. The Results column shows what the numeric value looks like when viewed from a text editor.

### See Also

- Format: HEXw. in SAS Language Reference Dictionary
- Informat: “HEXw.” on page 207
- “Representation of Numeric Variables” on page 139

---

**IBw.d**

Writes numbers in integer binary (fixed-point) format

**Numeric**

Width range: 1-8 bytes

Default width: 4

Decimal range: 0-10

Alignment:

**OS/390 specifics**: two’s complement notation

### Details

On an IBM mainframe system, integer values are stored in two’s complement notation. If an overflow occurs, the value written is the largest value that fits into the output field; the value will be positive, negative, or unsigned, as appropriate. If the format includes a d value, the number is multiplied by \(10^d\).

Here are some examples of the IBw.d format:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results (Hex)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1234</td>
<td>ib4.</td>
<td>fffffdb2e</td>
<td></td>
</tr>
<tr>
<td>12.34</td>
<td>ib4.</td>
<td>0000000c</td>
<td></td>
</tr>
<tr>
<td>123456789</td>
<td>ib4.</td>
<td>075bcd15</td>
<td></td>
</tr>
<tr>
<td>1234</td>
<td>ib6.2</td>
<td>00000001e208</td>
<td>a d value of 2 causes the number to be multiplied by (10^2)</td>
</tr>
<tr>
<td>Value</td>
<td>Format</td>
<td>Results (Hex)</td>
<td>Notes</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>---------------</td>
<td>-------</td>
</tr>
<tr>
<td>-1234</td>
<td>ib6.2</td>
<td>FFFFFFFFE1DF8</td>
<td>a d value of 2 causes the number to be multiplied by $10^2$</td>
</tr>
<tr>
<td>1234</td>
<td>ib1.</td>
<td>7F</td>
<td>overflow occurred</td>
</tr>
<tr>
<td>-1234</td>
<td>ib1.</td>
<td>80</td>
<td>overflow occurred</td>
</tr>
</tbody>
</table>

Note: In these examples, the Value column represents the value of the numeric variable. The Results column shows a hexadecimal representation of the bit pattern written by the corresponding format. (You cannot view this data in a text editor, unless you can view it in hexadecimal representation.)

See Also

- Formats: IBw.d, S370FIBw.d, and S370FPIBw.d in SAS Language Reference Dictionary
- Informat: “IBw.d "on page 208

**PDw.d**

Writes values in IBM packed decimal format

**Numeric**

- **Width range:** 1-16 bytes
- **Default width:** 1
- **Decimal range:** 0-31
- **Alignment:** left
- **OS/390 specifics:** IBM packed decimal format

**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 of $10^{31} - 1$ to $-10^{31} + 1$. The sign is written in the rightmost nibble. (A nibble is four bits or half a byte.) A hexadecimal C indicates a plus sign, and a hexadecimal D indicates a minus sign. 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.

If an overflow occurs, the value that is written is the largest value that fits into the output field; the value will be positive, negative, or unsigned, as appropriate.

Here are several examples of packed decimal format:
### RBw.d

**Writes numeric data in real binary (floating-point) notation**

**Numeric**
- **Width range:** 2-8 bytes
- **Default width:** 4
- **Decimal range:** 0-10
- **Alignment:** left
- **OS/390 specifics:** IBM floating-point format

**Details**

The format of floating-point numbers is host-specific. See “Representation of Floating-Point Numbers” on page 139 for a description of the format that is used to store floating-point numbers under OS/390.

Here are some examples of how decimal numbers are written as floating-point numbers using the RBw.d format:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results (Hex)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>rb8.1</td>
<td>434CE000000000000</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>rb8.2</td>
<td>44300C000000000000</td>
<td></td>
</tr>
<tr>
<td>-123</td>
<td>rb8.</td>
<td>C27B00000000000000</td>
<td></td>
</tr>
<tr>
<td>1234</td>
<td>rb8.</td>
<td>434D20000000000000</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** In these examples, the Value column represents the value of the data, and the Results column shows a hexadecimal representation of the bit pattern written by the corresponding format. (You cannot view this data in a text editor, unless you can view it in hexadecimal representation.)
w.d

**Writes numeric data**

**Numeric**

*Width range:* 1-32 bytes  
*Default width:* 12  
*Decimal range:* \(d < w\)  
*Alignment:* right  
*OS/390 specifics:* writes output as EBCDIC, minimum and maximum values

**Details**

The w.d format writes numeric values one digit per byte using EBCDIC code. Because the values are stored in EBCDIC, they can be printed without further formatting.

Numbers written with the w.d format are rounded to the nearest number that can be represented in the output field. If the number is too large to fit, the BESTw.d format is used. Under OS/390, the range of the magnitude of numbers that can be written with the BESTw.d format is from \(5.4 \times 10^{-79}\) to \(7.2 \times 10^{75}\).

The following examples illustrate the use of the w.d format:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>4.</td>
<td>1234</td>
</tr>
<tr>
<td>1234</td>
<td>5.</td>
<td>b1234</td>
</tr>
<tr>
<td>12345</td>
<td>4.</td>
<td>12E3</td>
</tr>
<tr>
<td>123.4</td>
<td>6.2</td>
<td>123.40</td>
</tr>
<tr>
<td>-1234</td>
<td>6.</td>
<td>b-1234</td>
</tr>
</tbody>
</table>

**Note:** In these examples, the Value column represents the value of the data, and the Results column shows what the numeric value looks like when viewed from a text editor. The b characters in the Results column indicate blank spaces. See Table 9.1 on page 142 for a table of commonly used EBCDIC characters.
ZDw.d

Writes zoned decimal data

Numeric
Width range: 1-32 bytes
Default width: 1
Decimal range: 0-32
Alignment: left
OS/390 specifics: IBM zoned decimal format

Details
Like standard format, zoned decimal digits are represented as EBCDIC characters. Each digit requires one byte. 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 written as the EBCDIC characters 0 through 9. The character that is written for the least significant digit depends on the sign of the number. Negative numbers are represented as the EBCDIC printable hexadecimal characters D0 through D9 in the least significant digit position, and positive numbers are represented as hexadecimal C0 through C9.

If an overflow occurs, the value that is written is the largest value that fits into the output field; the value will be positive, negative, or unsigned, as appropriate.

The following examples illustrate the use of the zoned decimal format:

<table>
<thead>
<tr>
<th>Value</th>
<th>Format</th>
<th>Results (Hex)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>zd8.</td>
<td>F0F0F0F01F2F3C4</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>zd8.1</td>
<td>F0F0F0F01F2F3C0</td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>zd8.2</td>
<td>F0F0F0F01F2F3C0</td>
<td></td>
</tr>
<tr>
<td>-123</td>
<td>zd8.</td>
<td>F0F0F0F01F2F3D3</td>
<td></td>
</tr>
<tr>
<td>0.000123</td>
<td>zd8.6</td>
<td>F0F0F0F01F2C3</td>
<td></td>
</tr>
<tr>
<td>0.00123</td>
<td>zd8.6</td>
<td>F0F0F0F01F2C0</td>
<td></td>
</tr>
<tr>
<td>1E-6</td>
<td>zd8.6</td>
<td>F0F0F0F0F0F01C1</td>
<td>overflow occurred</td>
</tr>
</tbody>
</table>

Note: In these examples, the Value column represents the value of the data, and the Results column shows a hexadecimal representation of the bit pattern that is written by the corresponding format. (You cannot view this data in a text editor unless you view it in hexadecimal representation.) See Table 9.1 on page 142 for a table of commonly used EBCDIC characters.
See Also

- Formats: ZDw.d, S370FZDLw.d, S370FZDSw.d, S370FZDTw.d, and S370FZDw.d in SAS Language Reference Dictionary
- Informats: “ZDw.d” on page 211, “ZDBw.d” on page 212, and “S370FZDw.d” in SAS Language Reference Dictionary