THE POWER TO KNOW.

SAS® 9.3 出力形式と入力形式 リファレンス

SAS® ドキュメント

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® 9.3 Formats and Informats: Reference*. Cary, NC: SAS Institute Inc.

SAS® 9.3 Formats and Informats: Reference

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

Printing 2, 2012 年 8 月
Electronic book 1, 2011 年 11 月

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/publishing** or call 1-800-727-3228.

目次

本書について

SAS 言語の構文規則

SAS 言語の構文規則の概要

SAS では、SAS 言語要素の構文ドキュメントに共通の規則を使用しています。これらの規則により、SAS 構文の構成要素を簡単に識別できます。規則は、次の項目に分類されます。

- 構文の構成要素
- スタイル規則
- 特殊文字
- SAS ライブラリと外部ファイルの参照

構文の構成要素

言語要素の多くでは、その構文の構成要素はキーワードと引数から構成されます。キーワードのみ必要な言語要素もあります。また、キーワードに等号(=)が続く言語要素もあります。

キーワード
プログラムの作成時に使用する SAS 言語要素名です。キーワードはリテラルであり、通常、構文の先頭の単語です。CALL ルーチンでは、最初の 2 つの単語がキーワードです。

次の SAS 構文の例では、構文の最初の単語がキーワードです。

CHAR (*string, position*)

CALL RANBIN (*seed, n, p, x*);

ALTER (*alter-password*)

BEST *w.*

REMOVE <*data-set-name*>

次の例では、CALL ルーチンの最初の 2 つの単語がキーワードです。

CALL RANBIN(*seed*, *n*, *p*, *x*)

引数なしで 1 つのキーワードから構成される SAS ステートメント構文もあります。

DO;
... SAS code ...
END;

2 つのキーワード値のいずれか 1 つの指定が必要なシステムオプションもあります。

DUPLEX | **NODUPLEX**

引数
数値定数、文字定数、変数、式のいずれかです。引数は、キーワードに続くか、キーワードの後ろの等号に続きます。SAS では、引数を使用して、言語要素を処理します。引数が必須の場合もオプションの場合もあります。構文では、オプションの引数にはかぎかっこが付けられます。

次の例では、*string* と *position* がキーワード CHAR に続きます。これらの引数は、CHAR 関数の必須引数です。

CHAR (*string, position*)

引数ごとに値が指定されます。次の例の SAS コードでは、引数 *string* の値として 'summer'、引数 *position* の値として 4 が指定されています。`x=char('summer', 4);`

次の例では、*string* と *substring* は必須引数ですが、*modifiers* と *startpos* はオプションの引数です。

FIND(*string, substring* <,*modifiers*> <,*startpos*>

注: 通常、SAS ドキュメントのサンプルコードは、小文字の固定幅フォントを使用して表記されます。コードの作成には、大文字も、小文字も、大文字と小文字の両方も使用できます。

スタイル規則

SAS 構文の説明に使用されるスタイル規則には、大文字太字、大文字、斜体の規則も含まれます。

大文字太字
関数名やステートメント名などの SAS キーワードを示します。次の例では、キーワード ERROR の表記には大文字太字が使用されています。

ERROR<*message*>;

大文字
リテラルの引数を示します。

次の CMPMODEL=システムオプションの例では、BOTH、CATALOG、XML がリテラルです。

CMPMODEL = BOTH | CATALOG | XML

斜体
ユーザー指定の引数または値を示します。斜体表記の項目は、ユーザー指定値であり、次のいずれかを表します。

- 非リテラルの引数。次の LINK ステートメントの例では、引数 *label* はユーザー指定値であるため、斜体で表記されています。

  LINK *label*;

- 引数に割り当てられる非リテラル値。

  次の FORMAT ステートメントの例では、引数 DEFAULT に変数の *default-format* が割り当てられます。

  FORMAT = *variable-1* <, ..., *variable-nformat*><DEFAULT = *default-format*>;

斜体表記の項目は、選択可能な引数リストの総称でもあります(*attribute-list* など)。複数の斜体表記の項目が使用される場合、項目は *item-1, ..., item-n* という形式で表記されます。

特殊文字

SAS 言語要素の構文には、次の特殊文字も使用されます。

=
等号は、一部の言語要素(システムオプションなど)のリテラル値を示します。

次の MAPS システムオプションの例では、等号は MAPS の値を設定します。

MAPS = *location-of-maps*

< >
かぎかっこはオプションの引数を示します。かぎかっこ付きでない引数は必須引数です。

次の CAT 関数の例では、少なくとも項目が 1 つ必要です。

CAT (*item-1 <, ..., item-n>*)

|
縦棒は、値グループから 1 つの値を選択できることを示します。縦棒で区切られている値は、相互排他です。

次の CMPMODEL=システムオプションの例では、属性を 1 つのみ選択できます。

CMPMODEL = BOTH | CATALOG | XML

...
省略記号は、省略記号に続く引数や引数グループの繰り返しを示します。省略記号とその後の引数にかぎかっこが付けられている場合、その引数はオプションです。

次の CAT 関数の例では、省略記号はオプションの項目を複数指定できることを示しています。

CAT (*item-1 <, ..., item-n>*)

'*value*' or "*value*"
単一引用符や二重引用符付きの引数は、その値も単一引用符または二重引用符を付ける必要があることを示します。

次の FOOTNOTE ステートメントの例では、引数 *text* には引用符が付けられています。

FOOTNOTE <*n*> <*ods-format-options* '*text*' | "*text*">;

;
セミコロンは、ステートメントまたは CALL ルーチンの終わりを示します。

次の例では、それぞれのステートメントはセミコロンで終了しています。
```
data namegame; length color name $8; color = 'black'; name =
'jack'; game = trim(color) || name; run;
```

SAS ライブラリと外部ファイルの参照

多くの SAS ステートメントなどの言語要素では、SAS ライブラリと外部ファイルを参照します。論理名(ライブラリ参照名またはファイル参照名)から参照を作成するのか、引用符付きの物理ファイル名を使用するかを選択できます。論理名を使用する場合、通

常、関連付けに SAS ステートメント(LIBNAME または FILENAME)を使用するのか、動作環境のコントロール言語を使用するのかを選択します。複数の方法を使用して、SAS ライブラリと外部ファイルを参照できます。動作環境によっては使用できない方法があります。

SAS ドキュメントでは、外部ファイルを使用する例には斜体のフレーズ *file-specification* を使用します。また、SAS ライブラリを使用する例には斜体フレーズ *SAS-library* を使用します。*SAS-library* は引用符付きであることに注意してください。

```
infile file-specification obs = 100;
libname libref 'SAS-library';
```

SAS 9.3 の出力形式と入力形式の新機能

概要

出力形式と入力形式は、別のドキュメントとして発行されるようになりました。*SAS Language Reference: Dictionary* に含まれなくなりました。詳細については、“SAS 言語リファレンス: ディクショナリの変更” (ix ページ)を参照してください。

SAS 9.3 では、新しい出力形式の追加や出力形式の拡張は行われていません。

新しい入力形式は、世紀マーカーを含む IBM 日時値、Java 日時値を読み取ります。また、*hhmmss* または *hh:mm:ss* 形式の時、分、秒を読み取ります。

新しい SAS 入力形式

次の入力形式が追加されました。

B8601CI*w*. (p. 243)
: 世紀マーカーを含む、形式 *cyymmddhhmmss<fff>*の IBM 日時値を読み取ります。

B8601DJ*w*. (p. 246)
: 形式 *yyyymmddhhmmssffffff* の Java 日時値を読み取ります。

HHMMSS*w*. (p. 277)
: 形式 *hhmmss* または *hh:mm:ss* の時、分、秒を読み取ります。

*SAS 言語リファレンス: ディクショナリ*の変更

SAS 9.3 より前では、このドキュメントは *SAS Language Reference: Dictionary* に組み込まれていました。SAS 9.3 から、*SAS Language Reference: Dictionary* は次の 7 つのドキュメントに分割されました。

- *SAS データセットオプション: リファレンス*
- *SAS 出力形式と入力形式: リファレンス*
- *SAS 関数と CALL ルーチン: リファレンス*
- *SAS ステートメント: リファレンス*
- *SAS システムオプション: リファレンス*

- *SAS コンポーネントオブジェクト: リファレンス* (ハッシュオブジェクトと Java オブジェクトに関するドキュメントを含む)
- *Base SAS Utilities: リファレンス* (SAS DATA ステップデバッガと SAS ユーティリティマクロ%DS2CSV に関するドキュメントを含む)

推奨資料

推奨図書

- *An Array of Challanges-Test Your SAS Skills*
- **Base SAS Glossary**
- *Base SAS プロシジャガイド*
- *Debugging SAS Programs: A Handbook of Tools and Techniques*
- *The Little SAS Book: A Primer*
- *UNIX 版 SAS*
- *Windows 版 SAS*
- *z/OS 版 SAS*
- *SAS Guide to Report Writing: Examples*
- *SAS 言語リファレンス: 解説編*
- *SAS 各国語サポート(NLS): リファレンスガイド*
- *SAS Programming by Example*
- *The SAS Workbook*
- *Step-by-Step Programming with Base SAS Software*
- *Using the SAS Windowing Environment: A Quick Tutorial*

SAS の刊行物の総一覧については、support.sas.com/bookstore にてご確認ください。
必要な書籍についてのご質問は、下記までお寄せください。

SAS Publishing Sales
SAS Campus Drive
Cary, NC 27513-2414
電話: 1-800-727-3228
ファクシミリ: 1-919-677-8166
電子メール: sasbook@sas.com
Web アドレス: support.sas.com/bookstore

1 部

SAS 出力形式

1 章

出力形式について

出力形式について

出力形式は、出力として表示または出力されるデータ値へのパターンの適用や命令の実行を行う SAS 言語要素です。出力形式の種類は、データの種類(数値、文字、日付、時間、タイムスタンプ)に対応します。ユーザー定義の出力形式を作成することもできます。SAS 出力形式の例として、BINARY、DATE、WORDS などがあります。たとえば、数値を英語表記に相当する値に変換する WORDS22.は、数値 692 を `six hundred ninety-two` として出力します。

構文

SAS 出力形式の形式は次のようになります。

<$>format<w>.<d>

$
文字出力形式であることを表します。数値出力形式には使用されません。

format
出力形式名です。出力形式には、SAS 出力形式、PROC FORMAT の VALUE ステートメントを使用して定義されたユーザー定義の出力形式があります。

参照項目: ユーザー定義の出力形式の詳細については、Chapter 27, "FORMAT Procedure" in *Base SAS Procedures Guide* を参照してください。

w
出力形式の幅です。大部分の出力形式では、出力データのカラム数となります。

d
オプションの小数点以下の桁数です(数値出力形式)。

出力形式は、名前の一部として必ずピリオド(.)を含みます。出力形式の *w* 値と *d* 値を省略すると、デフォルト値が使用されます。出力形式で指定する *d* 値に基づき、SAS は小数点以下の桁数を表示します。出力形式は、内部的に保存されているデータ値は変更または切り捨てるものではありません。

たとえば、DOLLAR10.2 では、*w* 値に 10 が指定されているため、値の最大カラム数は 10 となります。*d* 値には 2 が指定されているため、10 カラムのうち 2 カラムが値の小数部分に使用されます。値の残りの文字には 8 カラムが使用されます。この残りのカラムには、小数点、小数以外の数値、マイナス記号(値が負の場合)、ドル記号、およびカンマが必要に応じてが含まれます。

値を表すには出力形式の幅が狭すぎる場合、値は利用可能なスペースに合うように圧縮されます。文字出力形式では、値の右側を切り捨てます。数値出力形式では、BEST*w.d* 出力形式が使用される場合があります。十分な幅を指定していない場合、アスタリスクが出力されます。次の例では、結果は x=**.となります。

```
x=123;
put x= 2.;
```

数値出力形式を使用して文字値を書き出すなど、矛盾する出力形式を使用する場合、まず他の種類の類似した出力形式が試されます。失敗時には、問題を説明するエラーメッセージが SAS ログに表示されます。

d の値が 15 より大きい場合、有効桁数 15 桁以降の桁の小数値が正確でない場合があります。

出力形式の使用

出力形式の指定方法

出力形式の指定について

出力形式を次の方法で使用できます。

- PUT ステートメントで使用する
- PUT 関数、PUTC 関数、または PUTN 関数と使用する
- %SYSFUNC マクロ関数と使用する
- DATA ステップまたは PROC ステップの FORMAT ステートメントで使用する
- DATA ステップまたは PROC ステップの ATTRIB ステートメントで使用する

PUT ステートメント

変数名の後に出力形式が指定された PUT ステートメントは、DATA ステップでその出力形式を使用してデータ値を書き出します。たとえば、次の PUT ステートメントは、DOLLAR*w.d* 出力形式を使用して、AMOUNT の数値をドルで書き出します。

```
amount=1145.32;
put amount dollar10.2;
```

DOLLAR*w.d* 出力形式の結果は次のようになります。

```
$1,145.32
```

詳細については、“PUT Statement” in *SAS Statements: Reference* を参照してください。

PUT 関数

PUT 関数は、数値変数、文字変数、定数を有効な出力形式を使用して変換し、変換後の文字値を返します。たとえば、次のステートメントは、数値変数の値を 2 桁の 16 進表現に変換します。

```
num=15;
char=put(num,hex2.);
```

この PUT 関数は、値 0F を返します。この値は、変数 CHAR に割り当てられます。

PUT 関数は、数値を文字値に変換する場合に使用します。

詳細については、“PUT Function” in *SAS Functions and CALL Routines: Reference* を参照してください。

%SYSFUNC マクロ関数

%SYSFUNC (または%QSYSFUNC)マクロ関数は、SAS 関数またはユーザー定義関数を実行し、DATA ステップ外でオプションの出力形式を関数の結果に適用します。たとえば、次のプログラムは、マクロ変数の数値をドルとして書き出します。

```
 %macro tst(amount);
%put %sysfunc(putn(&amount,dollar10.2));
%mend tst;

%tst (1154.23);
```

詳細については、“%SYSFUNC and %QSYSFUNC Functions” in *SAS Macro Language: Reference* を参照してください。

FORMAT ステートメント

FORMAT ステートメントは、文字変数を文字出力形式に、数値変数を数値出力形式に恒久的に関連付けます。

SAS は出力形式を使用して、ユーザーが指定する変数の値を書き出します。たとえば、DATA ステップの次のステートメントは、COMMA*w.d* 数値出力形式を SALES1 から SALES3 までの変数に関連付けます。

```
format sales1-sales3 comma10.2;
```

FORMAT ステートメントは恒久的に出力形式を変数に関連付けるため、次に続く DATA ステップや PROC ステップは、COMMA10.2 を使用して、SALES1、SALES2、SALES3 の値を書き出します。

詳細については、“FORMAT Statement” in *SAS Statements: Reference* を参照してください。

注: PUT ステートメントの前に FORMAT ステートメントを使用して出力形式を割り当てる場合、先頭の空白はすべて切り捨てられます。FORMAT ステートメントを使用して変数に関連付けられた出力形式は、後続の PUT ステートメントでコロン(:)修飾子も一緒に指定された出力形式と同じ結果を出力します。コロンフォーマット修飾子の使用に関する詳細については、“PUT Statement, List” in *SAS Statements: Reference* を参照してください。

ATTRIB ステートメント

ATTRIB ステートメントは、他の属性と同様に、出力形式を 1 つ以上の変数と関連付けることができます。たとえば、次のステートメントの ATTRIB ステートメントは、恒久的に COMMA*w.d* 出力形式を SALES1 から SALES3 までの変数に関連付けます。

```
attrib sales1-sales3 format=comma10.2;
```

ATTRIB ステートメントは恒久的に出力形式を変数に関連付けるため、次に続く DATA ステップや PROC ステップは、COMMA10.2 を使用して、SALES1、SALES2、SALES3 の値を書き出します。

詳細については、“ATTRIB Statement” in *SAS Statements: Reference* を参照してください。

恒久的な関連付けと一時関連付け

出力形式を PUT ステートメントに指定するとき、DATA ステップ中に出力形式を使用してデータ値を書き出しますが、出力形式と変数を恒久的に関連付けるものではありません。出力形式を変数と恒久的に関連付けるには、DATA ステップで FORMAT ステートメントまたは ATTRIB ステートメントを使用します。SAS では、SAS データセットのディスクリプタ情報を変更することにより、出力形式と変数を恒久的に関連付けます。

PROC ステップで FORMAT ステートメントまたは ATTRIB ステートメントを使用すると、この PROC ステップに加えて、このプロシジャが作成して出力形式が適用される変数を含む出力データセットに対しても、出力形式を変数に関連付けます。

SAS プロシジャでの出力形式の使用に関する詳細については、“Formatted Values” in Chapter 2 of *Base SAS Procedures Guide* を参照してください。

ユーザー定義の出力形式

Base SAS が提供する出力形式に加えて、独自の出力形式を作成できます。Base SAS では、PROC FORMAT を使用して、文字変数と数値変数の両方に対し独自の出力形式を作成できます。

詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

ユーザー定義の出力形式を使用する SAS プログラムの実行時に、これらの出力形式が使用できる必要があります。これらの出力形式を使用可能にするには 2 つの方法があります。

- 一時的でなく、恒久的な出力形式を PROC FORMAT で作成する

- 出力形式を作成する(PROC FORMAT ステップ)ソースコードを、それを使用する SAS プログラムと一緒に保存する

恒久的な SAS 出力形式を作成するには、Chapter 27, "FORMAT Procedure" in *Base SAS Procedures Guide* を参照してください。

ユーザー定義の出力形式を見つけられないプログラムを実行する場合、結果は FMTERR システムオプションの設定によって異なります。ユーザー定義の出力形式が見つからない場合、次のシステムオプションの結果は次のようになります。

システムオプション	結果
FMTERR	現在の DATA ステップまたは PROC ステップを停止するエラーを発生します。
NOFMTERR	処理は続行され、デフォルトの出力形式を置き換えます。通常は BEST*w.*または$*w.*出力形式になります。

NOFMTERR を使用すると変数を処理できますが、ユーザー定義の出力形式によって提供される情報は失われます。

問題を避けるために、使用されるすべてのユーザー定義の出力形式にプログラムがアクセスできるかを確認します。

ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング

定義

バイナリ整数データの整数値は、通常、1 バイト、2 バイト、3 バイトの 3 つのサイズのいずれかで保存されます。整数のバイトオーダリングは、整数が生成されるプラットフォーム(動作環境)によって異なります。

バイトオーダリングは、"ビッグエンディアン"プラットフォームと"リトルエンディアン"プラットフォームとで異なります。これらの俗称は、IBM メインフレーム(ビッグエンディアン)と Intel 基準プラットフォーム(リトルエンディアン)のバイトオーダリングを表すために使用されます。SAS System では、AIX、HP-UX、IBM メインフレーム、Macintosh、および SPARC の Solaris のプラットフォームをビッグエンディアンとします。Intel ABI、Linux、OpenVMS Alpha、OpenVMS Integrity、x64 の Solaris、Tru64 UNIX および Windows のプラットフォームをリトルエンディアンとします。

バイトオーダリングの相違点

ビッグエンディアンプラットフォームでは、値 1 はバイナリで保存され、16 進表記で表されます。1 バイトは 01、2 バイトは 00 01、4 バイトは 00 00 00 01 としてそれぞれ保存されます。リトルエンディアンプラットフォームでは、値 1 は 1 バイトで 01 (ビッグエンディアンと同じ)、2 バイトで 01 00、4 バイトで 01 00 00 00 としてそれぞれ保存されます。

負の整数には、"2 つの補数"表現が使用されます。整数の最大有効バイトの高位ビットに設定されます。たとえば、–2 は、ビッグエンディアンプラットフォームでは、1 バイト、2 バイト、4 バイトの場合それぞれ FE、FF FE、FF FF FF FE として表されます。リト

ルエンディアンプラットフォームでは、それぞれ FE、FE FF、FE FF FF FF として表されます。これらは、16 進表現のバイナリ整数値–2 の出力結果です。

ビッグエンディアンまたはリトルエンディアンプラットフォームで生成されたデータの書き出し

SAS では、符号付き整数または符号なし整数をビッグエンディアンシステムまたはリトルエンディアンシステムで生成されていても読み込むことができます。同様に、ビッグエンディアン形式とリトルエンディアン形式の符号付き整数および符号なし整数を書き出すことができます。これらの整数の長さは、最大で 8 バイトまで可能です。

次の表は、プラットフォームのさまざまな組み合わせに使用する出力形式を示したものです。符号付き整数列の“いいえ”は、符号なしで、負でない数字を示します。“はい”は、数字が負または正のいずれかであることを示します。

表 1.1 SAS 出力形式とバイトオーダリング

データが作成されたプラットフォーム	データを書き出すプラットフォーム	符号付き整数	出力形式
ビッグエンディアン	ビッグエンディアン	はい	IB または S370FIB
ビッグエンディアン	ビッグエンディアン	いいえ	PIB、S370FPIB、S370FIBU
ビッグエンディアン	リトルエンディアン	はい	S370FIB
ビッグエンディアン	リトルエンディアン	いいえ	S370FPIB
リトルエンディアン	ビッグエンディアン	はい	IBR
リトルエンディアン	ビッグエンディアン	いいえ	PIBR
リトルエンディアン	リトルエンディアン	はい	IB または IBR
リトルエンディアン	リトルエンディアン	いいえ	PIB または PIBR
ビッグエンディアン	いずれか	はい	S370FIB
ビッグエンディアン	いずれか	いいえ	S370FPIB
リトルエンディアン	いずれか	はい	IBR
リトルエンディアン	いずれか	いいえ	PIBR

バイナリ整数表記と各種プログラミング言語

次の表は、プログラミング言語別のバイナリ整数表記を比較したものです。

言語	2 バイト	4 バイト
SAS	IB2. 、IBR2.、PIB2.、PIBR2.、S370FIB2.、S370FIBU2.、S370FPIB2.	IB4.、IBR4.、PIB4.、PIBR4.、S370FIB4.、S370FIBU4.、S370FPIB4.

言語	2 バイト	4 バイト
PL/I	FIXED BIN(15)	FIXED BIN(31)
Fortran	INTEGER*2	INTEGER*4
COBOL	COMP PIC 9(4)	COMP PIC 9(8)
IBM アセンブラ	H	F
C	short	long

データの変換とエンコーディング

エンコーディングは、文字セットの各文字を一意の数値表現にマップし、すべてのコードポイントを含む表を作成します。各文字は、エンコーディングごとに異なる数値表現がされます。たとえば、ドル記号$の ASCII エンコーディングは、24 (16 進表現)となります。ドル記号$のデンマーク語の EBCDIC エンコーディングは、67 (16 進表現)となります。通常は ASCII を使用するバージョンの SAS がデンマーク語の EBCDIC でエンコーディングされたデータセットを適切に解釈するためには、データをトランスコードする必要があります。

トランスコーディングとは、1 つのエンコーディングから別のエンコーディングにデータを移動する処理のことです。ASCII のドル記号がデンマーク語の EBCDIC のドル記号にトランスコードされると、この文字の 16 進表現が値 24 から 67 に変換されます。

特定の SAS データセットのエンコーディングを確認するには、SAS 9 以降では次の操作を行います。

1. SAS エクスプローラでデータセットを検索します。
2. データセットを右クリックします。
3. メニューからプロパティを選択します。
4. 詳細タブをクリックします。
5. データセットのエンコーディングが、他の情報と一緒に表示されます。

データがトランスコードされる頻度が高いのは、次のような状況です。

- 異なるロケールまたは異なる動作環境で実行している、2 つの異なる SAS セッション間でデータを共有する場合
- 大文字または小文字への変換などの、テキスト文字列の操作を実行する場合
- 別の言語の文字を表示または出力する場合
- 異なるロケールで実行している SAS セッション間でデータをコピーして貼り付ける場合

異なるエンコーディングまたは動作環境からの各国語サポート関連のトランスコーディングを処理するために設計された SAS 機能の詳細については、*SAS 各国語サポート (NLS): リファレンスガイド*を参照してください。

パック 10 進データとゾーン 10 進データの処理

定義

パック 10 進
: 10 進数の 2 桁を 1 バイトで表現する、10 進数のエンコード方法です。パック 10 進表現は、正確な精度で 10 進データを保存します。数字の小数部分は、別に仮数や指数がないため、入力形式または出力形式によって設定されます。

  パック 10 進データを使用する利点は、正確な精度を維持できることです。ただし、10 進データを含む計算はネイティブ命令を欠く場合は不正確になることがあります。

ゾーン 10 進
: 1 桁がストレージの 1 バイトを使用する 10 進数のエンコード方法です。最終バイトには、最終桁と数字の符号が含まれます。ゾーン 10 進データは、出力可能な表現を生成します。

ニブル
: 0.5 バイトです。

データの種類

パック10 進データ

パック 10 進表現は、10 進数の桁をバイトの"ニブル"に保存します。バイトは 2 つのニブルを含み、ニブルは 1 桁の 16 進数によって示されます。たとえば、値 15 は、16 進数の 1 と 5 を使用して 2 つのニブルに保存されます。

符号の表示は動作環境によって異なります。IBM メインフレームでは、符号は最後のニブルによって示されます。出力形式の使用時、C は正の値、D は負の値を示します。入力形式の使用時、A、C、E および F は正の値、B と D は負の値を示します。他のニブルは、符号付きパック 10 進データに使用できません。他のすべての動作環境では、符号はその独自のバイトで示されます。高位ビットが 1 の場合、数字は負になります。他の場合、数字は正になります。

次の事柄がパック 10 進データ表現に当てはまります。

- すべてのプラットフォーム上で S370FPD 出力形式を使用して、IBM メインフレーム形式の 10 進データを取得できます。
- 符号インジケータのない符号なしパックデータを使用できます。パック 10 進の出力形式および入力形式により、パック 10 進データ表現が処理されます。ASCII プラットフォームでも、EBCDIC プラットフォームでも同じです。
- S370FPDU の出力形式および入力形式では最後のニブルに F が必要ですが、パック 10 進では符号のニブルは必要ありません。

ゾーン10 進データ

次の事柄がゾーン 10 進データ表現に当てはまります。

- ゾーン 10 進表現は、10 進数の桁をバイトの下位ニブルに保存します。符号を含まないバイトのすべてにおいて、上位ニブルは数値ゾーンニブル(EBCDIC では F、ASCII では 3)になります。

- 符号はその表現によって、桁を含むバイトに結合することも、切り離すことも可能です。ただし、標準ゾーン 10 進の出力形式および入力形式では、符号が最後のバイトに結合されます。
- EBCDIC および ASCII のゾーン 10 進出力形式は、同じ出力可能な表現で数字を生成します。バイトは 2 つのニブルを含み、それぞれが 1 桁の 16 進数によって示されます。たとえば、値 15 は 2 バイトで保存されます。1 番目のバイトには 16 進値の F1 が、2 番目のバイトには 16 進値の C5 がそれぞれ含まれます。

パックユリウス暦の日付

次の事柄がパックユリウス暦の日付に当てはまります。

- パック 10 進表現のユリウス暦の日付を処理する 2 つの出力形式と入力形式は、PDJULI と PDJULG です。PDJULI は IBM メインフレームの年計算を使用し、PDJULG はグレゴリオ暦の計算を使用します。
- IBM メインフレームの計算では、1900 を基準年とし、データの年値は 1900 からのオフセットを示します。たとえば、98 は 1998、100 は 2000、102 は 2002 をそれぞれ表します。また、1998 は 3898 を表します。
- グレゴリオ暦の計算では、2 桁または 4 桁の年を利用できます。2 桁の年を使用する場合、YEARCUTOFF=システムオプションの設定を使用して実際の年を決定します。

パック 10 進データとゾーン 10 進データをサポートするプラットフォーム

一部のプラットフォームには、パックおよびゾーン 10 進データをサポートするためのネイティブ命令が含まれています。他のプラットフォームでは、ソフトウェアを使用して計算をエミュレートする必要があります。たとえば、IBM メインフレームにはパック 10 進データを追加するためのパック追加命令が含まれていますが、Intel 基準のプラットフォームにはそのような命令は含まれていないため、10 進データを他の形式に変換する必要があります。

パック 10 進データとゾーン 10 進データをサポートする言語

複数の言語で、パック 10 進データとゾーン 10 進データがサポートされています。次の表は、COBOL picture 句に対応する SAS 出力形式および入力形式を示したものです。

IBM VS COBOL II 句	対応する S370Fxxx 出力形式および入力形式
PIC S9(X) PACKED-DECIMAL	S370FPDw.
PIC 9(X) PACKED-DECIMAL	S370FPDUw.
PIC S9(W) DISPLAY	S370FZDw.
PIC 9(W) DISPLAY	S370FZDUw.
PIC S9(W) DISPLAY SIGN LEADING	S370FZDLw.
PIC S9(W) DISPLAY SIGN LEADING SEPARATE	S370FZDSw.
PIC S9(W) DISPLAY SIGN TRAILING SEPARATE	S370FZDTw.

表内のパック 10 進表現では、X は表す桁数、W はバイト数を表します。PIC S9(X) PACKED-DECIMAL では、W は **ceil((x+1)/2)** になります。PIC 9(X) PACKED-DECIMAL では、W は **ceil(x/2)** になります。たとえば、PIC S9(5) PACKED-DECIMAL は 5 桁を表します。符号が含まれる場合、6 つのニブルが必要です。**ceil((5+1)/2)** の長さは 3 バイトであり、W の値は 3 になります。

PACKED-DECIMAL の代わりに COMP-3 を使用できます。

IBM アセンブリ言語では、P ディレクティブはパック 10 進を、Z ディレクティブはゾーン 10 進をそれぞれ示します。次はアセンブリ言語リストの抜粋であり、オフセット、値、DC ステートメントが示されています。

```
offset value (in hex) inst label directive

+000000 00001C 2 PEX1 DC PL3'1'
+000003 00001D 3 PEX2 DC PL3'-1'
+000006 F0F0C1 4 ZEX1 DC ZL3'1'
+000009 F0F0D1 5 ZEX2 DC ZL3'1'
```

PL/I では、FIXED DECIMAL 属性はパック 10 進データとともに使用されます。ゾーン 10 進データを表すには、PICTURE 指定を使用する必要があります。Fortran または C 言語では、10 進データの統一した表現はありません。

パック10 進とゾーン10 進の出力形式と入力形式の概要

SAS では、パック 10 進データとゾーン 10 進データを処理するために、一連の出力形式および入力形式を使用します。次の表は、これらの出力形式および入力形式のデータ表現の種類を示したものです。S370 で始まる出力形式および入力形式は、IBM メインフレーム表現を示します。

出力形式	データ表現の種類	対応する入力形式	コメント
PD	パック 10 進	PD	ローカル符号付きパック 10 進
PK	パック 10 進	PK	符号なしパック 10 進; 動作環境に固有でない
ZD	ゾーン 10 進	ZD	ローカルゾーン 10 進
なし	ゾーン 10 進	ZDB	EBCDIC の空白(16 進 40) を EBCDIC のゼロ(16 進 F0)に変換; ゾーン 10 進の入力形式に相当
なし	ゾーン 10 進	ZDV	IBM 形式でないゾーン 10 進表現
S370FPD	パック 10 進	S370FPD	最後のニブルが C (正)または D (負)
S370FPDU	パック 10 進	S370FPDU	最後のニブルは常に F (正)
S370FZD	ゾーン 10 進	S370FZD	最終バイトの上位ニブルに符号が含まれる: C (正)または D (負)
S370FZDU	ゾーン 10 進	S370FZDU	符号なし; 符号ニブルは常に F

出力形式	データ表現の種類	対応する入力形式	コメント
S370FZDL	ゾーン 10 進	S370FZDL	先頭バイトに符号ニブルが含まれる(入力形式); 先頭に 16 進数の C0 (正)または D0 (負)を含む符号バイトが個別にある(出力形式)
S370FZDS	ゾーン 10 進	S370FZDS	先頭に符号- (16 進数の 60)または+ (16 進数の 4E)あり
S370FZDT	ゾーン 10 進	S370FZDT	末尾に符号- (16 進数の 60)または+ (16 進数の 4E)があり
PDJULI	パック 10 進	PDJULI	パック表現のユリウス暦の日付 - IBM 計算
PDJULG	パック 10 進	PDJULG	パック表現のユリウス暦の日付 - クレゴリオ暦計算
なし	パック 10 進	RMFDUR	入力レイアウト: *mmssttt*F
なし	パック 10 進	SHRSTAMP	入力レイアウト: *yyyydddd*F*hhmmssth*。*yyyydddd*F はパックユリウス暦の日付 ; *yyyy* は 1900 を基準年とする年。
なし	パック 10 進	SMFSTAMP	入力レイアウト: *xxxxxxxxyyyydddd*F。*yyyydddd*F はパックユリウス履歴の日付; *yyyy* は 1900 を基準年とする年。
なし	パック 10 進	PDTIME	入力レイアウト: 0*hhmmss*F
なし	パック 10 進	RMFSTAMP	入力レイアウト: 0*hhmmss*F*yyyydddd*F。*yyyydddd*F はパックユリウス履歴の日付; yyyy は 1900 を基準年とする年。

ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理

ISO 8601 規格のフォーマッティングシンボル

次のリストは、ISO 8601 規格の日付値、時間値、日時値、期間値、間隔値の表記に使用されるフォーマッティングシンボルについて説明したものです。

n
年数、月数、日数を表す数字です。

P
年数、月数、日数、時間数、分数、秒数で示される期間が続くことを示します。

T
時間値が続くことを示します。時間を含む値は T で始まる必要があります。

要件 文字 E8601 で始まる拡張表記入力形式から読み込む時間値には、大文字の T を使用する必要があります。

W
期間が週数で示されることを示します。

Z
時間値がイギリスのグリニッジに対応した時間、つまり UTC 時間であることを示します。

+|-
+は、イギリスのグリニッジの東部タイムゾーンのオフセットを示します。-は、イギリスのグリニッジの西部タイムゾーンのオフセットを示します。

yyyy
4 桁の年を示します。

mm
日付の一部。2 桁の月、01-12 を示します。

dd
2 桁の日、01–31 を示します。

hh
2 桁の時間、00–24 を示します。

mm
時間の一部。2 桁の時間、00–59 を示します。

ss
2 桁の秒、00–59 を示します。

fff | ***ffffff***
秒の端数 0–9 を示します(オプション)。

fff $N8601B 入力形式と$N8601E 入力形式では、小数点以下の桁数 1 - 3 の値を読み込みます

ffffff $N8601B 入力形式および$N8601E 入力形式以外の入力形式では、小数点以下の桁数 1 - 6 桁の値を読み込みます

Y
期間の年数値が次に続くことを示します。

M
日付の一部。期間の月数値が次に続くことを示します。

D
期間の日数値が次に続くことを示します。

H
期間の時間値が次に続くことを示します。

M
時間の一部。期間の分数値が次に続くことを示します。

S
期間の秒数値が次に続くことを示します。

ISO 8601 規格の日付値、時間値、日時値の書き出し

SAS では次の表の出力形式を使用して、SAS 日付値、SAS 時間値、SAS 日時値から日付値、時間値、日時値を ISO 8601 規格の基本表記および拡張表記で書き出します。

日付、時間、日時	ISO 8601 表記	例	出力形式
基本表記			
日付	*yyyymmdd*	20120915	B8601DA*w*.
時間	*hhmmssffffff*	155300322348	B8601TM*w.d*
タイムゾーン付き時間	*hhmmss*+|-*hhmm*	155300+0500	B8601TZ*w.d*
	*hhmmss*Z	155300Z	B8601TZ*w.d*
タイムゾーン付きローカル時間への変換	hhmmss+|-hhmm	155300+0500	B8601LZ*w.d*
日時	*yyyymmdd*T*hhmmssffffff*	20120915T155300	B8601DT*w.d*
タイムゾーン付き日時	*yyyymmdd*T*hhmmss*+|-*hhmm*	20120915T155300+0500	B8601DZ*w.d*
	*yyyymmdd*T*hhmmss*Z	20120915T155300Z	B8601DZ*w.d*
日時からの日付の書き出し	*yyyymmdd*	20120915	B8601DN*w*.
拡張表記			
日付	*yyyy-mm-dd*	2012-09-15	E8601DA*w*.
時間	*hh:mm:ss.ffffff*	15:53:00.322348	E8601TM*w.d*
タイムゾーン付き時間	*hh:mm:ss.ffffff*+|-*hh:mm*	15:53:00+05:00	E8601TZ*w.d*
タイムゾーン付きローカル時間への変換	*hh:mm:ss.ffffff*+|-*hh:mm*	15:53:00+05:00	E8601LZ*w.d*
日時	*yyyy-mm-dd*T*hh:mm:ss.ffffff*	2012-09-15T15:53:00	E8601DT*w.d*
タイムゾーン付き日時	*yyyy-mm-dd*T*hh:mm:ss.nnnnnn*+|-*hh:mm*	2012-09-15T15:53:00+05:00	E8601DZ*w.d*
日時からの日付の書き出し	*yyyy-mm-dd*	2012-09-15	E8601DN*w*.

アスタリスク(*)は、出力形式が適用された日付値または時間値が範囲外の場合に使用されます。

ISO 8601 規格の期間値、日時値、間隔値の書き出し

期間、日時、間隔の出力形式

SAS では、次の出力形式を使用して、文字データから期間値、日時値、間隔値を書き出します。

表 1.2 構成要素をすべて含む形式

時間要素	ISO 8601 表記	例	出力形式
期間 - 基本表記	P*yyyymmdd*T*hhmmssfff*	P20120915T155300	$N8601BA
	-P*yyyymmdd*T*hhmmssfff*	-P20120915T155300	$N8601BA
期間 - 拡張表記	P*yyyy-mm-dd*T*hh:mm:ss.fff*	P2012-09-15T15:53:00	$N8601EA
	-P*yyyy-mm-dd*T*hh:mm:ss.fff*	-P2012-09-15T15:53:00	$N8601EA
期間 - 基本表記と拡張表記	P*n*Y*n*M*n*DT*n*H*n*M*n*S	P2y10m14dT20h13m45s	$N8601B $N8601E
	-P*n*Y*n*M*n*DT*n*H*n*M*n*S	-P2y10m14dT20h13m45s	$N8601B $N8601E
	P*n*W (週)	P6w	$N8601B $N8601E
間隔 - 基本表記	*yyyymmdd*T*hhmmssfff*/ yyyymmddT*hhmmssfff*	20120915T155300/2014111 3T000000	$N8601BA
	P*n*Y*n*M*n*DT*n*H*n*M*n*S/ *yyyymmdd*T*hhmmssfff*	P2y10M14dT20h13m45s/ 20120915T155300	$N8601B
	*yyyymmdd*T*hhmmssfff*/ P*n*Y*n*M*n*DT*n*H*n*M*n*S	20120915T155300/ P2y10M14dT20h13m45s	$N8601BA
間隔 - 拡張表記	*yyyy-mm-dd*T*hh:mm:ss.fff*/ *yyyy-mm-dd*T*hh:mm:ss.fff*	2012-09-15T15:53:00/2014 -11-13T00:00:00	$N8601EA
	P*n*Y*n*M*n*DT*n*H*n*M*n*S/*yyyy- mm-dd*T*hh:mm:ss.fff*	P2y10M14dT20h13m45s/ 2012-09-15T15:53:00	$N8601E
	*yyyy-mm-dd*T*hh:mm:ss.fff*/ P*n*Y*n*M*n*DT*n*H*n*M*n*S	2012-09-15T15:53:00/ P2y10M14dT20h13m45s	$N8601EA
日時 - 基本表記	*yyyymmdd*T*hhmmss.fff*+|- *hhmm*	20120915T155300	$N8601BA
	(すべて空白)		$N8601B $N8601BA $N8601E $N8601EA

時間要素	ISO 8601 表記	例	出力形式
日時 - 拡張表記	*yyyy-mm-dd*T*hh:mm:ss.fff* +|-*hhmm*	2012-09-15T15:53:00 +04:30	$N8601EA
	(すべて空白)		$N8601B $N8601BA $N8601E $N8601EA

省略構成要素の書き出し

省略構成要素は、ハイフン(-) または x を使用して、拡張日時形式 *yyyy-mm-dd*T*hh:mm:ss* および拡張期間形式 P*yyyy-mm-dd*T*hh:mm:ss* では示せます。

期間形式 P*n*Y*n*M*n*DT*n*H*n*M*n*S では、省略構成要素は削除するため、ハイフンまたは x を含みません。たとえば、P2mT4H のようになります。

次の出力形式は、省略構成要素をハイフンおよび x を使用して書き出します。

出力形式	日時形式	期間形式	例
$N8601H	*yyyy-mm-dd*T*hh:mm:ss*	P*n*Y*n*M*n*DT*n*H*n*M*n*S	--09-15T15:-:53 P2Y2DT4H5M6S/--09-15T15:-:00
$N8601EH	*yyyy-mm-dd*T*hh:mm:ss*	P*yyyy-mm-dd*T*hh:mm:ss*	P000---02T02:55:20/2012---15T-:-:45
$N8601X	*yyyy-mm-dd*T*hh:mm:ss*	P*n*Y*n*M*n*DT*n*H*n*M*n*S	P2Y2DT4H5M6S/x-09-15T15:x:00
$N8601EX	*yyyy-mm-dd*T*hh:mm:ss*	P*yyyy-mm-dd*T*hh:mm:ss*	P0003-x-02T02:55:20/2012-x-15Tx:x:45

省略構成要素がある日時値に$N8601B 出力形式または$N8601BA 出力形式のいずれかを適用する場合、省略構成要素にハイフンを使用する拡張表記で書き出され、正確なデータが提供されます。たとえば、月が省略構成要素の場合、値 2012-15 ではなく、値 2012---15 が書き出されます。

また、ハイフンを使用する拡張表記は、期間が$N8601BA 出力形式で書き出される場合にも基本表記の代わりに使用されます。上述と同じ日付の場合、P2012-15 ではなく P2012---15 が書き出されます。

切り捨て期間値、日時値、間隔値の書き出し

期間値、日付値、間隔値では、下位値が 0 または有効でない場合に切り捨てられます。出力形式$N8601B、$N8601BA、$N8601E、$N8601EA を使用して切り捨て値を書き出す場合、値は最後の非欠損構成要素まで書き出されます。

$N8601H 出力形式または$N8601EH 出力形式のいずれかを使用して切り捨て値を書き出す場合、下位構成要素はハイフンを使用して書き出されます。$N8601X 出力形式または$N8601EX 出力形式を使用して切り捨て値を書き出す場合、下位構成要素は x を使用して書き出されます。

次に、切り捨て値の例を示します。

- **p00030202T1031**
- **2012-09-15T15/2014-09-15T15:53**
- **-p0003-03-03T-:-:-**
- **P2y3m4dT5h6m**
- **2012-09-xTx:x:x**
- **2012**

期間構成要素の標準化

期間構成要素の値が構成要素の最大標準値を超えると、構成要素はその期間構成要素が単一の構成要素である場合を除いて標準化されます。次の表は、標準化された期間構成要素の例を示したものです。

期間	標準化された期間(拡張表記)
p3y13m	p0004-01
pt24h24m65s	P----01T-:25:05
p3y13mT24h61m	P0004-01-01T01:01
p0004-13	p0005-01
p0003-02-61T15:61:61	P0003-04-01T16:02:01
p13m	P13M

構成要素に最大値(分や秒の場合は 60)が含まれている場合、その値は標準化され、ハイフンと置き換えられます。たとえば、**pT12:60:13** は **PT13:-:13** になります。

30 日で月が標準化されます。

日時値の日付と時間が構成要素の標準値を超える場合は、標準化されません。この場合、エラーが発生します。

期間値、日時値、間隔値の端数

終わりの構成要素には、1 つのピリオドまたはカンマとそれに続く 1 桁から 3 桁までの小数点以下の桁数からなる端数を含めることができます。次の例は、期間値、日付値、間隔値での端数の使用を表したものです。

- **201209.5**
- **P2012-09-15T10.33**
- **2012-09-15/P0003-03-03,333**

2 章

出力形式のディクショナリ

他のドキュメントで説明されている出力形式

各国語サポートの入力形式については、Chapter 9, “Dictionary of Formats for NLS,” in *SAS National Language Support (NLS): Reference Guide* を参照してください。

カテゴリ別の出力形式

このリストには、次の 4 つのカテゴリの出力形式があります。

カテゴリ	説明
文字	文字変数から文字データ値を書き出すように指示します。
日付と時間	日付、時刻および日時を表す変数からデータ値を書き出すように指示します。
ISO 8601	日付値、時間値、日付値を ISO 8601 規格を使用して書き出すように指示します。
数値	数値変数から数値データ値を書き出すように指示します。

各国語サポートの出力形式については、*SAS 各国語サポート(NLS): リファレンスガイド*を参照してください。

ユーザー定義の出力形式の保存は、永久 SAS データセット、特に他のユーザーと共有するデータセットの変数にその出力形式を関連付ける場合に大変に重要になります。ユーザー定義の出力形式の作成および保存の詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

次の表に、SAS 出力形式の概要を示します。詳細については、各出力形式のディクショナリエントリを参照してください。

カテゴリ	言語要素	説明
ISO 8601	$N8601Bw.d 出力形式 (p. 37)	基本表記 PnYnMnDTnHnMnS と yyyymmddThhmmss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601BAw.d 出力形式 (p. 39)	基本表記 PyyyymmddThhmmss と yyyymmddThhmmss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601Ew.d 出力形式 (p. 40)	拡張表記 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EAw.d 出力形式 (p. 41)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EHw.d 出力形式 (p. 42)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EXw.d 出力形式 (p. 43)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、省略した構成要素の各数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601Hw.d 出力形式 (p. 44)	期間値の省略した構成要素を削除し、日時値の省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を書き出します。
	$N8601Xw.d 出力形式 (p. 46)	期間値の省略した構成要素を削除し、日時値の省略した構成要素の数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を書き出します。

カテゴリ	言語要素	説明
	B8601DAw. 出力形式 (p. 58)	ISO 8601 規格の基本表記 yyyymmdd を使用して日付値を書き出します。
	B8601DNw. 出力形式 (p. 59)	ISO 8601 規格の基本表記 yyyymmdd を使用して、日時値から日付を書き出します。
	B8601DTw.d 出力形式 (p. 60)	日時値を ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して日時値を書き出します。
	B8601DZw.出力形式 (p. 61)	ISO 8601 規格の日時とタイムゾーンの基本表記 yyyymmddThhmmss+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	B8601LZw.出力形式 (p. 62)	ISO 8601 規格の基本表記 hhmmss+|-hhmm を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。
	B8601TMw.d 出力形式 (p. 64)	ISO 8601 規格の基本表記 hhmmss<ffff>を使用して、時間値を書き出します。
	B8601TZw.出力形式 (p. 65)	ISO 8601 規格の基本時間表記 hhmmss+|–hhmm を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。
	E8601DAw.出力形式 (p. 88)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して日付値を書き出します。
	E8601DNw. 出力形式 (p. 89)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して、SAS 日時値から日付を書き出します。
	E8601DTw.d 出力形式 (p. 90)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.ffffff を使用して日時値を書き出します。
	E8601DZw.出力形式 (p. 91)	ISO 8601 規格の日時とタイムゾーンの拡張表記 yyyy-mm-ddThh:mm:ss+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	E8601LZw.出力形式 (p. 92)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|-hh:mm を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。
	E8601TMw.d 出力形式 (p. 94)	ISO 8601 規格の拡張表記 hh:mm:ss.ffffff を使用して時間値を書き出します。
	E8601TZw.d 出力形式 (p. 95)	時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 hh:mm:ss+|-hh:mm を使用して書き出します。
数値	BESTw. 出力形式 (p. 54)	SAS により最適な出力が選択されます。
	BESTDw.p 出力形式 (p. 55)	同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。整数は小数なしで出力します。
	BINARYw. 出力形式 (p. 57)	数値を 2 進表現に変換します。

カテゴリ	言語要素	説明
	COMMAw.d 出力形式 (p. 66)	3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。
	COMMAXw.d 出力形式 (p. 67)	3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。
	Dw.p 出力形式 (p. 68)	値の桁数をできる限り最大にし、同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。
	DOLLARw.d 出力形式 (p. 78)	先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。
	DOLLARXw.d 出力形式 (p. 79)	先頭にドル($)記号を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。
	Ew. 出力形式 (p. 87)	数値を指数表記で書き出します。
	FLOATw.d 出力形式 (p. 97)	数値に 10 の d 乗を掛けた、単精度のネイティブ浮動小数点値を書き出します。
	FRACTw. 出力形式 (p. 98)	数値を分数に変換します。
	HEXw. 出力形式 (p. 99)	バイナリ実数(浮動小数点)値を 16 進表現に変換します。
	IBw.d 出力形式 (p. 103)	負の値を含む、ネイティブのバイナリ整数(固定小数点)値を書き出します。
	IBRw.d 出力形式 (p. 104)	Intel および DEC 形式のバイナリ整数(固定小数点)値を書き出します。
	IEEEw.d 出力形式 (p. 105)	数値を 10 の d 乗し、IEEE 浮動小数点値を生成します。
	NEGPARENw.d 出力形式 (p. 120)	負の数値をかっこで囲んで書き出します。
	NUMXw.d 出力形式 (p. 121)	小数点をカンマにして数値を書き出します。
	OCTALw. 出力形式 (p. 122)	数値を 8 進表現に変換します。
	PDw.d 出力形式 (p. 123)	パック 10 進形式のデータを書き出します。
	PERCENTw.d 出力形式 (p. 127)	数値を百分率として書き出します。
	PERCENTNw.d 出力形式 (p. 128)	百分率を書き出します。負の値にはマイナス符号を使用します。
	PIBw.d 出力形式 (p. 129)	正のバイナリ整数(固定小数点)値を書き出します。
	PIBRw.d 出力形式 (p. 131)	Intel 形式と DEC 形式の正のバイナリ整数(固定小数点)値を書き出します。
	PKw.d 出力形式 (p. 132)	符号なしパック 10 進形式のデータを書き出します。

カテゴリ	言語要素	説明
	PVALUEw.d 出力形式 (p. 133)	p 値を書き出します。
	RBw.d 出力形式 (p. 136)	バイナリ実数データ(浮動小数点)をバイナリ実数形式で書き出します。
	ROMANw. 出力形式 (p. 137)	数値をローマ数字で書き出します。
	S370FFw.d 出力形式 (p. 138)	IBM メインフレーム形式の標準数値データを書き出します。
	S370FIBw.d 出力形式 (p. 139)	IBM メインフレーム形式の負の値を含むバイナリ整数(固定小数点)値を書き出します。
	S370FIBUw.d 出力形式 (p. 140)	IBM メインフレーム形式の符号なしバイナリ整数(固定小数点)値を書き出します。
	S370FPDw.d 出力形式 (p. 141)	IBM メインフレーム形式のパック 10 進データを書き出します。
	S370FPDUw.d 出力形式 (p. 142)	IBM メインフレーム形式の符号なしパック 10 進データを書き出します。
	S370FPIBw.d 出力形式 (p. 143)	IBM メインフレーム形式の正のバイナリ整数(固定小数点)値を書き出します。
	S370FRBw.d 出力形式 (p. 145)	IBM メインフレーム形式のバイナリ実数(浮動小数点)を書き出します。
	S370FZDw.d 出力形式 (p. 146)	IBM メインフレーム形式のゾーン 10 進データを書き出します。
	S370FZDLw.d 出力形式 (p. 147)	IBM メインフレーム形式の前符号付きゾーン 10 進データを書き出します。
	S370FZDSw.d 出力形式 (p. 148)	IBM メインフレーム形式の分離した前符号付きゾーン 10 進データを書き出します。
	S370FZDTw.d 出力形式 (p. 149)	IBM メインフレーム形式の分離した後符号付きゾーン 10 進データを書き出します。
	S370FZDUw.d 出力形式 (p. 150)	IBM メインフレーム形式の符号なしゾーン 10 進データを書き出します。
	SSNw. 出力形式 (p. 151)	アメリカの社会保障番号形式で書き出します。
	VAXRBw.d 出力形式 (p. 157)	VMS 形式のバイナリ実数(浮動小数点)を書き出します。
	VMSZNw.d 出力形式 (p. 158)	VMS および MicroFocus COBOL ゾーン数値データを生成します。
	w.d 出力形式 (p. 159)	標準数値でデータを書き出します(1 バイト 1 桁)。
	WORDFw. 出力形式 (p. 171)	数値を英語表現で書き出します。分数は数値で表示します。

カテゴリ	言語要素	説明
	WORDSw. 出力形式 (p. 172)	数値を英語表現で書き出します。
	Zw.d 出力形式 (p. 187)	先頭に 0 を含む標準数値データを書き出します。
	ZDw.d 出力形式 (p. 188)	ゾーン 10 進形式の数値データを書き出します。
日付と時間	$N8601Bw.d 出力形式 (p. 37)	基本表記 PnYnMnDTnHnMnS と yyyymmddThhmmss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601BAw.d 出力形式 (p. 39)	基本表記 PyyyymmddThhmmss と yyyymmddThhmmss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601Ew.d 出力形式 (p. 40)	拡張表記 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EAw.d 出力形式 (p. 41)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EHw.d 出力形式 (p. 42)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601EXw.d 出力形式 (p. 43)	拡張表記 Pyyyy-mm-ddThh:mm:ss と yyyy-mm-ddThh:mm:ss を使用して、省略した構成要素の各数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。
	$N8601Hw.d 出力形式 (p. 44)	期間値の省略した構成要素を削除し、日時値の省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を書き出します。
	$N8601Xw.d 出力形式 (p. 46)	期間値の省略した構成要素を削除し、日時値の省略した構成要素の数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式 PnYnMnDTnHnMnS と yyyy-mm-ddThh:mm:ss を書き出します。
	B8601DAw. 出力形式 (p. 58)	ISO 8601 規格の基本表記 yyyymmdd を使用して日付値を書き出します。
	B8601DNw. 出力形式 (p. 59)	ISO 8601 規格の基本表記 yyyymmdd を使用して、日時値から日付を書き出します。
	B8601DTw.d 出力形式 (p. 60)	日時値を ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して日時値を書き出します。
	B8601DZw.出力形式 (p. 61)	ISO 8601 規格の日時とタイムゾーンの基本表記 yyyymmddThhmmss+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	B8601LZw.出力形式 (p. 62)	ISO 8601 規格の基本表記 hhmmss+|-hhmm を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。

カテゴリ	言語要素	説明
	B8601TMw.d 出力形式 (p. 64)	ISO 8601 規格の基本表記 hhmmss<ffff>を使用して、時間値を書き出します。
	B8601TZw.出力形式 (p. 65)	ISO 8601 規格の基本時間表記 hhmmss+|–hhmm を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。
	DATEw. 出力形式 (p. 70)	SAS 日付値を ddmmmyy、ddmmmyyyy または dd-mmm-yyyy 形式で書き出します。
	DATEAMPMw.d 出力形式 (p. 71)	SAS 日時値を、AM または PM を付けて ddmmmyy:hh:mm:ss.ss 形式で書き出します。
	DATETIMEw.d 出力形式 (p. 72)	SAS 日時値を ddmmmyy:hh:mm:ss.ss 形式で書き出します。
	DAYw. 出力形式 (p. 74)	SAS 日付値の日付部分を書き出します。
	DDMMYYw. 出力形式 (p. 75)	SAS 日付値を ddmm<yy>yy または dd/mm/<yy>yy 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。
	DDMMYYxw. 出力形式 (p. 76)	SAS 日付値を ddmm<yy>yy or dd-mm-yy<yy>形式で書き出します。出力形式名の x は、日、月、年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	DOWNAMEw. 出力形式 (p. 81)	SAS 日付値から曜日名を書き出します。
	DTDATEw. 出力形式 (p. 81)	入力として日時値が必要です。SAS 日付値を ddmmmyy または ddmmmyyyy 形式で書き出します。
	DTMONYYw. 出力形式 (p. 82)	SAS 日時値を mmmyy または mmmyyyy 形式で書き出します。
	DTWKDATXw. 出力形式 (p. 84)	SAS 日時値を day-of-week, dd month-name yy (または yyyy)形式で書き出します。
	DTYEARw. 出力形式 (p. 85)	SAS 日時値を yy または yyyy 形式で書き出します。
	DTYYQCw. 出力形式 (p. 86)	SAS 日時値から、年と四半期をコロン(:)で区切って書き出します。
	E8601DAw.出力形式 (p. 88)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して日付値を書き出します。
	E8601DNw. 出力形式 (p. 89)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して、SAS 日時値から日付を書き出します。
	E8601DTw.d 出力形式 (p. 90)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.ffffff を使用して日時値を書き出します。

カテゴリ	言語要素	説明
	E8601DZw.出力形式 (p. 91)	ISO 8601 規格の日時とタイムゾーンの拡張表記 yyyy-mm-ddThh:mm:ss+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。
	E8601LZw.出力形式 (p. 92)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|-hh:mm を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。
	E8601TMw.d 出力形式 (p. 94)	ISO 8601 規格の拡張表記 hh:mm:ss.ffffff を使用して時間値を書き出します。
	E8601TZw.d 出力形式 (p. 95)	時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 hh:mm:ss+|-hh:mm を使用して書き出します。
	HHMMw.d 出力形式 (p. 100)	SAS 時間値を時間および分として hh:mm 形式で書き出します。
	HOURw.d 出力形式 (p. 101)	SAS 時間値を時間および時間の小数として書き出します。
	JULDAYw. 出力形式 (p. 106)	SAS 日付値からユリウス暦の日付部分を書き出します。
	JULIANw. 出力形式 (p. 107)	SAS 日付値を yyddd または yyyyddd 形式のユリウス日付として書き出します。
	MDYAMPMw.d 出力形式 (p. 108)	SAS 日時値を mm/dd/yy<yy> hh:mm AM|PM 形式で書き出します。年は 2 桁または 4 桁で表示されます。
	MMDDYYw. 出力形式 (p. 110)	SAS 日付値を mmdd<yy>yy または mm/dd/<yy>yy 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。
	MMDDYYxw. 出力形式 (p. 111)	SAS 日付値を mmdd<yy>yy または mm-dd-<yy>yy 形式で書き出します。出力形式名の x は、月、日、年を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	MMSSw.d 出力形式 (p. 113)	SAS 時間値を午前 0 時から経過した分数および秒数として書き出します。
	MMYYw. 出力形式 (p. 114)	SAS 日付値を mmM<yy>yy 形式で書き出します。M は区切り文字を表し、年は 2 桁または 4 桁の数字で書き出されます。
	MMYYxw. 出力形式 (p. 116)	SAS 日付値を mm<yy>yy または mm-<yy>yy 形式で書き出します。出力形式名の x は、月と年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	MONNAMEw. 出力形式 (p. 117)	SAS 日付値を月の名前として書き出します。
	MONTHw. 出力形式 (p. 118)	SAS 日付値を月として書き出します。

カテゴリ	言語要素	説明
	MONYYw. 出力形式 (p. 119)	SAS 日付値を月および年として mmmyy または mmmyyyy 形式で書き出します。
	PDJULGw. 出力形式 (p. 124)	パックユリウス日付値を、IBM で使用する 16 進の yyyydddF 形式で書き出します。
	PDJULIw. 出力形式 (p. 126)	パック表現のユリウス日付値を、IBM で使用する 16 進の ccyydddF 形式で書き出します。
	QTRw. 出力形式 (p. 134)	SAS 日付値を年の四半期として書き出します。
	QTRRw. 出力形式 (p. 135)	SAS 日付値を年の四半期としてローマ数字で書き出します。
	TIMEw.d 出力形式 (p. 152)	SAS 時間値を hh:mm:ss.ss 形式で書き出します。
	TIMEAMPMw.d 出力形式 (p. 154)	SAS 時間値と SAS 日時値を AM または PM を使用して hh:mm:ss.ss 形式で書き出します。
	TODw.d 出力形式 (p. 155)	SAS 時間値と、SAS 日時値の時間部分を hh:mm:ss.ss 形式で書き出します。
	WEEKDATEw. 出力形式 (p. 160)	SAS 日付値を day-of-week, month-name dd, yy (または yyyy)形式で書き出します。
	WEEKDATXw. 出力形式 (p. 162)	SAS 日付値を day-of-week, dd month-name yy (または yyyy)形式で書き出します。
	WEEKDAYw. 出力形式 (p. 163)	SAS 日付値から曜日の値を書き出します。
	WEEKUw. 出力形式 (p. 164)	U アルゴリズムを使用して、10 進数の週番号を書き出します。
	WEEKVw. 出力形式 (p. 166)	V アルゴリズムを使用して、10 進数の週番号を書き出します。
	WEEKWw. 出力形式 (p. 168)	W アルゴリズムを使用して、10 進数の週番号を書き出します。
	WORDDATEw. 出力形式 (p. 169)	SAS 日付値を month-name dd, yyyy 形式で書き出します。
	WORDDATXw. 出力形式 (p. 170)	SAS 日付値を dd month-name yyyy 形式で書き出します。
	YEARw. 出力形式 (p. 173)	SAS 日付値から年部分を書き出します。
	YYMMw. 出力形式 (p. 174)	SAS 日付値を<yy>yyMmm 形式で書き出します。M は月番号が M の後に続くことを示す区切り文字で、年は 2 桁または 4 桁の数字で書き出されます。
	YYMMDDw. 出力形式 (p. 175)	SAS 日付値を yymmdd または<yy>yy-mm-dd 形式で書き出します。ハイフンが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ	言語要素	説明
	YYMMDDxw. 出力形式 (p. 177)	SAS 日付値を yymmdd または<yy>yy-mm-dd 形式で書き出します。出力形式名の x は、年、月、日を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。
	YYMONw. 出力形式 (p. 181)	SAS 日付値を yymmm または yyyymmm 形式で書き出します。
	YYQw. 出力形式 (p. 182)	SAS 日付値を<yy>yyQq 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また、q は四半期を表します。
	YYQxw. 出力形式 (p. 183)	SAS 日付値を<yy>yyq または<yy>yy-q 形式で書き出します。出力形式名の x は、年と四半期または年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。
	YYQRw. 出力形式 (p. 184)	SAS 日付値を<yy>yyQqr 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また qr はローマ数字表現の四半期を表します。
	YYQRxw. 出力形式 (p. 185)	SAS 日付値を<yy>yyqr または<yy>yy-qr 形式で書き出します。出力形式名の x は、年と四半期、または年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます、また qr は、ローマ数字表現の四半期です。
文字	$ASCIIw. 出力形式 (p. 31)	ネイティブなフォーマットの文字データを ASCII 表現に変換します。
	$BASE64Xw. 出力形式 (p. 32)	Base 64 エンコーディングを使用して、文字データを ASCII テキストに変換します。
	$BINARYw. 出力形式 (p. 33)	文字データを 2 進表現に変換します。
	$CHARw. 出力形式 (p. 34)	標準文字データを書き出します。
	$EBCDICw. 出力形式 (p. 35)	ネイティブな形式の文字データを EBCDIC 表現に変換します。
	$HEXw. 出力形式 (p. 36)	文字データを 16 進表現に変換します。
	$MSGCASEw. 出力形式 (p. 37)	MSGCASE システムオプションが有効になっているときに、文字データを大文字で書き込みます。
	$OCTALw. 出力形式 (p. 47)	文字データを 8 進表現に変換します。
	$QUOTEw. 出力形式 (p. 48)	データ値を二重引用符で囲んで書き込みます。
	$REVERJw. 出力形式 (p. 49)	文字データの末尾から順に書き込みます。空白も保持されます。

カテゴリ	言語要素	説明
	$REVERSw. 出力形式 (p. 50)	文字データの末尾から順に、左揃えで書き込みます。
	$UPCASEw. 出力形式 (p. 51)	文字データを大文字に変換します。
	$VARYINGw. 出力形式 (p. 51)	可変長の文字データを書き込みます。
	$w. 出力形式 (p. 53)	標準文字データを書き出します。

ディクショナリ

$ASCIIw. 出力形式

ネイティブなフォーマットの文字データを ASCII 表現に変換します。

カテゴリ: 文字

配置: 左

構文

$ASCII*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 1
範囲: 1–32767

詳細

ASCII がネイティブな形式の場合、変換は実行されません。

比較

- $ASCII*w*.を EBCDIC システムで使用すると、EBCDIC 形式の文字データは ASCII*w* 形式に変換されます。
- その他のシステム上では、$ASCII*w*.は$CHAR*w*.出力形式と同じ結果になります。

例

```
put x $ascii3.;
```

x の値	結果*
abc	616263
ABC	414243
();	28293B

* 上記の結果は、ASCII 形式のコード値を 16 進表現で出力したものです。16 進数の 2 桁が ASCII コードの 1 バイトに対応しています。

$BASE64X*w*. 出力形式

Base 64 エンコーディングを使用して、文字データを ASCII テキストに変換します。

カテゴリ: 文字

配置: 左

構文

$BASE64X*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1-32767

詳細

Base 64 は、ポジション指定スキームに基づき、ASCII 文字のみ使用したエンコード文字を指定するエンコーディング方式です。複数の Base 64 エンコーディングスキームが、電子メールやコンテンツマスキングなどの特定の用途に定義されています。SAS では、ポジション 0 - 61 を文字 A - Z、a - z、0 - 9 にマップします。ポジション 62 は文字 +にマップし、ポジション 63 は文字/にマップします。

次に、Base 64 エンコーディングの使用例を示します。

- XML ファイルの埋め込みバイナリデータ
- パスワードのエンコード
- URL のエンコード

エンコード結果の'='文字は、結果にゼロビットが埋め込まれたことを示します。エンコード文字をデコードするには、'='をデコードする値に含める必要があります。

例

put x $base64x64.;

x の値	結果
"FCA01A7993BC"	RkNBMDFBNzk5M0JD
"MyPassword"	TXlQYXNzd29yZA==
"www.mydomain.com/myhiddenURL"	d3d3Lm15ZG9tYWluLmNvbi9teWhpZGRlblVSTA==

関連項目:

- LIBNAME ステートメントオプション“XMLDOUBLE=DISPLAY | INTERNAL” in Chapter 8 of *SAS XML LIBNAME Engine: User's Guide*

入力形式:

- “$BASE64Xw. 入力形式” (219 ページ)

$BINARY*w.* 出力形式

文字データを 2 進表現に変換します。

カテゴリ: 文字

配置: 左

構文

$BINARY*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1–32767

比較

$BINARY*w.*出力形式は文字値を 2 進表現に変換します。BINARY*w.*出力形式は数値を 2 進表現に変換します。

例

```
put @1 name $binary16.;
```

name の値	結果	
	ASCII	EBCDIC
AB	0100000101000010	1100000111000010

$CHAR*w*. 出力形式

標準文字データを書き出します。

カテゴリ: 文字

配置: 左

構文

$CHAR*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 変数の長さが定義されていなければ 8。その他の場合は変数の長さ。
範囲: 1–32767

比較

- $CHAR*w*.出力形式は$*w*.出力形式と同じ結果になります。
- $CHAR*w*.出力形式と $*w*.出力形式は先頭の空白を切り捨てません。先頭の空白を切り捨てるには、LEFT 関数を使用して、文字データを左寄せにします。または、PUT ステートメントにフォーマット修飾子のコロン(:)と選択した出力形式を指定して使用して、リスト出力します。
- 次の表は、SAS 出力形式の$CHAR8.と他のプログラミング言語での表記方法を比較したものです。

言語	表記
SAS	$CHAR8.
C	char [8]
COBOL	PIC x(8)
Fortran	A8
PL/I	A(8)

例

```
put @7 name $char4.;
```

name の値	結果
	----+----1

name の値	結果
XYZ	XYZ

$EBCDICw. 出力形式

ネイティブな形式の文字データを EBCDIC 表現に変換します。

カテゴリ: 文字

配置: 左

構文

$EBCDIC*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–32767

詳細

EBCDIC がネイティブな形式の場合、変換は実行されません。

比較

- $EBCDIC*w.*を ASCII システム上で使用すると、ASCII 形式の文字データは EBCDIC 形式に変換されます。
- その他のシステム上では、$EBCDIC*w.*は$CHAR*w.*出力形式と同じ結果になります。

例

```
put name $ebcdic3.;
```

name の値	結果*
qrs	9899A2
QRS	D8D9E2
+;>	4E5E6E

* 上記の結果は、EBCDIC 形式のコード値を 16 進表現で出力したものです。16 進数の 2 桁が EBCDIC コードの 1 バイトに対応しています。

$HEX*w*. 出力形式

文字データを 16 進表現に変換します。

カテゴリ: 文字

配置: 左

参照項目: “$HEXw. Format: UNIX” in *SAS Companion for UNIX Environments*
“$HEXw. Format: Windows” in *SAS Companion for Windows*

構文

$HEX*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1–32767

ヒント:
データに相当する 16 進表現を完全に書き出すには、*w* に、変換した変数またはフィールドの長さを 2 倍した値を指定する必要があります。

w が変換したい変数の長さの 2 倍を超えている場合、$HEX*w* により空白が埋め込まれます。

詳細

$HEX*w*.出力形式は、1 文字を 16 進数の 2 桁に変換します。末尾の空白を含め、空白は 1 文字として扱われます。

比較

HEX*w*.出力形式は、バイナリ実数を 16 進数の表現に変換します。

例

```
put @5 name $hex4.;
```

name の値	結果	
	EBCDIC	ASCII
	----+----1	----+----1
AB	C1C2	4142

$MSGCASE*w.* 出力形式

MSGCASE システムオプションが有効になっているときに、文字データを大文字で書き込みます。

カテゴリ: 文字

配置: 左

構文

$MSGCASE*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 変数の長さが定義されていなければ 1。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

MSGCASE=システムオプションが有効な場合、SAS が生成する NOTES、WARNINGS、ERROR メッセージはすべて大文字で表示されます。その他の場合、NOTES、WARNINGS、ERROR メッセージはすべて大文字小文字混在で表示されます。MSGCASE=システムオプションは、構成ファイルに指定するか、または SAS 起動時に指定します。

例

```
put name $msgcase.;
```

name の値	結果
sas	SAS

関連項目:

システムオプション:

- “MSGCASE System Option: UNIX” in *SAS Companion for UNIX Environments*
- “MSGCASE System Option: Windows” in *SAS Companion for Windows*
- “MSGCASE System Option: z/OS” in *SAS Companion for z/OS*

$N8601B*w.d* 出力形式

基本表記 P*n*Y*n*M*n*DT*n*H*n*M*n*S と *yyyymmdd*T*hhmmss* を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601B*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–3

詳細

$N8601B 出力形式は、ISO 8601 規格の期間、日時、間隔の値を次の基本表記の文字データとして書き出します。

- P*n*Y*n*M*n*DT*n*H*n*M*n*S
- *yyyymmdd*T*hhmmss*
- P*n*Y*n*M*n*DT*n*H*n*M*n*S/*yyyymmdd*T*hhmmss*
- *yyyymmdd*T*hhmmss*T/P*n*Y*n*M*n*DT*n*H*n*M*n*S

最小単位の構成要素には、次の例のように端数を含めることができます。

- **p2y3.5m**
- **p00020304T05.335**

例

```
put nb $n8601b.;
```

nb の値	結果
0002405050112FFC	P2Y4M5DT5H1M12S
2012915155300FFD	20120915T155300
2012915000000FFD2014915000000FFD	20120915T000000/20140915T000000
0033104030255FFC2012915155300FFD	P33Y1M4DT3H2M55S/20120915T155300

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

$N8601BA*w.d* 出力形式

基本表記 P*yyyymmdd*T*hhmmss* と *yyyymmdd*T*hhmmss* を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.5.4.2, alternative format

構文

$N8601BA*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–3

詳細

$N8601BA 出力形式は、ISO 8601 規格の期間、日時、間隔の値を次の基本表記の文字データとして書き出します。

- P*yyyymmdd*T*hhmmss*
- *yyyymmdd*T*hhmmss*
- P*yyyymmdd*T*hhmmss*/*yyyymmdd*T*hhmmss*
- *yyyymmdd*T*hhmmss*/P*yyyymmdd*T*hhmmss*

最小単位の構成要素には、次の例のように端数を含めることができます。

- `p00023.5`
- `00020304T05.335`

例

```
put @1 nba $N8601ba.;
```

nba の値	結果
00024050501127D0	P00020405T050112.5
2012915155300FFD	20120915T155300
000230405060752820129151553 00FFD	P00020304T050607.33/20120915T155300

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$N8601E*w.d* 出力形式

拡張表記 P*n*Y*n*M*n*DT*n*H*n*M*n*S と *yyyy-mm-dd*T*hh:mm:ss* を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601E*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 50

範囲: 1–200

要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–3

詳細

$N8601E 出力形式は、ISO 8601 規格の期間、日時、間隔の値を次の基本表記の文字データとして書き出します。

- P*n*Y*n*M*n*DT*n*H*n*M*n*S

- *yyyy-mm-dd*T*hh:mm:ss*
- P*n*Y*n*M*n*DT*n*H*n*M*n*S/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*T/P*n*Y*n*M*n*DT*n*H*n*M*n*S

最小単位の構成要素には、次の例のように端数を含めることができます。

- **p2y3.5m**
- **p0002-03-04T05.335**

例

```
put @1 ne $n8601e.;
```

ne の値	結果
00024050501127D0	P2Y4M5DT5H1M12.5S
2012915155300FFD	2012-09-15T15:53:00
2012915000000FFD2014915000000FFD	2012-09-15T00:00:00/2013-09-15T00:00:00
0033104030255FFC2012915155300FFD	P33Y1M4DT3H2M55S/2012-09-15T15:53:00

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$N8601EA*w.d* 出力形式

拡張表記 P*yyyy-mm-dd*T*hh:mm:ss* と *yyyy-mm-dd*T*hh:mm:ss* を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601EA*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d

最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–3

詳細

$N8601EA 出力形式は、ISO 8601 規格の期間、日時、間隔の値を次の基本表記の文字データとして書き出します。

- P*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*
- P*yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*/P*yyyy-mm-dd*T*hh:mm:ss*

最小単位の構成要素には、次の例のように端数を含めることができます。

- **p00023.5**
- **0002-03-04T05.335**

例

```
put @1 nea $N8601ea.;
```

nea の値	結果
00024050501127D0	P0002-04-05T05:01:12.500
2012915155300FFD	2012-09-15T15:53:00
000230405060752820129151553 00FFD	P0002-03-04T05:06:07.330/2012-09-15T15:53:00

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$N8601EH*w.d* 出力形式

拡張表記 P*yyyy-mm-dd*T*hh:mm:ss* と *yyyy-mm-dd*T*hh:mm:ss* を使用して、省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601EH*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 50

範囲: 1–200

要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d

最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–3

詳細

$N8601EH 出力形式は、省略した構成要素を表すためにハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の値を次の拡張表記の文字データとして書き出します。

- P*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*
- P*yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*/P*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*

省略した日時構成要素は常に表示され、切り詰められません。

例

```
put a $n8601eh.;
```

a の値	結果
00023FFFFFFFFFFC2012FFF15FFFFFFD	P0002-03—T-:-:-/2012—T15:-:-
2012FFF15FFFFFFdFFFF3FF1553FFFFC	2012--T15:-:-/P—03—T15:53:-

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

$N8601EX*w.d* 出力形式

拡張表記 P*yyyy-mm-dd*T*hh:mm:ss* と *yyyy-mm-dd*T*hh:mm:ss* を使用して、省略した構成要素の各数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Elements 5.5.3, 5.5.4.1, and 5.5.4.2

構文

$N8601EX*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 50

範囲: 1–200

要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d

最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–3

詳細

$N8601EX 出力形式は、省略した構成要素を表すためにハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の値を次の拡張表記の文字データとして書き出します。

- P*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*
- P*yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*/P*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*

省略した日時構成要素は常に表示され、切り詰められません。

例

```
put nex $n8601ex.;
```

nex の値	結果
00023FFFFFFFFFC2012FFF15FFFFFFD	P0002-03xxTxx:xx:xx/2012-xx-xxT15:xx:xx
2012FFF15FFFFFFdFFFF3FF1553FFFFC	2012-xx-xxT15:xx:xx/Pxxxx-03-xxT15:53:xx

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$N8601H*w.d* 出力形式

期間値の省略した構成要素を削除し、日時値の省略した構成要素にハイフン(-)を使用して、ISO 8601 規格の期間、日時、間隔の形式 P*n*Y*n*M*n*DT*n*H*n*M*n*S と *yyyy-mm-dd*T*hh:mm:ss* を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Elements 5.5.3, 5.5.4.1, and 5.5.4.2

構文

$N8601H*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–3

詳細

$N8601H 出力形式は、P*n*Y*n*M*n*DT*n*H*n*M*n*S の形式の構成要素を省略し、省略した日時形式の構成要素を表すためにハイフン (-)を使用して、ISO 8601 規格の期間、間隔、日時を次の形式で書き出します。

- P*n*Y*n*M*n*DT*n*H*n*M*n*S
- *yyyy-mm-dd*T*hh:mm:ss*
- P*n*Y*n*M*n*DT*n*H*n*M*n*S/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*T/P*n*Y*n*M*n*DT*n*H*n*M*n*S
- *yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*

省略した日時構成要素は常に表示され、切り詰められません。

例

```
put nh $n8601h.;
```

nh の値	結果
0002304FFFFFFFFC2012FFF15FFFFFFD	P2Y3M4D/2012—-T15:-:-
FFFF102FFFFFFFFD2012FFF15FFFFFFD	—01-02T-:-:-0/2012—-T15:-:-

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$N8601X*w.d* 出力形式

期間値の省略した構成要素を削除し、日時値の省略した構成要素の数字に x を使用して、ISO 8601 規格の期間、日時、間隔の形式 P*n*Y*n*M*n*DT*n*H*n*M*n*S と *yyyy-mm-dd*T*hh:mm:ss* を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Elements 5.5.3, 5.5.4.1, and 5.5.4.2

構文

$N8601X*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
最小単位の構成要素に小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–3

詳細

$N8601X 出力形式は、P*n*Y*n*M*n*DT*n*H*n*M*n*S の形式の構成要素を省略し、省略した日時形式の構成要素を表すために x を使用して、ISO 8601 規格の期間、間隔、日時を次の形式で書き出します。

- P*n*Y*n*M*n*DT*n*H*n*M*n*S
- *yyyy-mm-dd*T*hh:mm:ss*
- P*n*Y*n*M*n*DT*n*H*n*M*n*S/*yyyy-mm-dd*T*hh:mm:ss*
- *yyyy-mm-dd*T*hh:mm:ss*T/P*n*Y*n*M*n*DT*n*H*n*M*n*S
- *yyyy-mm-dd*T*hh:mm:ss*/*yyyy-mm-dd*T*hh:mm:ss*

省略した日時構成要素は常に表示され、切り詰められません。

例

```
put nx $n8601x.;
```

nx の値	結果
0002304FFFFFFFFC2011FFF15FFFFFFD	P2Y3M4D/2011-xx-xxT15:xx:xx
FFFF102FFFFFFFFD2011FFF15FFFFFFd	xxxx-01-02Txx:xx:xx/2011x-xxT15:xx:xx

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

$OCTALw. 出力形式

文字データを 8 進表現に変換します。

カテゴリ: 文字

配置: 左

構文

$OCTAL*w*.

説明

w

出力フィールドの幅を指定します。

デフォルト: デフォルトの幅は、出力対象となる変数の長さに基づいて計算されます。

範囲: 1–32767

ヒント: 文字値はそれぞれ 3 つの 8 進表現の文字を生成するため、*w* の値には文字値の長さを 3 倍した値を指定します。

比較

$OCTAL*w*.出力形式は、文字値の文字コードを 8 進表現に変換します。OCTAL*w*.出力形式は、数値を 8 進表現に変換します。

例

次の例は、$OCTAL*w*.出力形式を使用した場合の ASCII 出力です。

```
data _null_;
infile datalines truncover;
input item $5.;
put item $octal15.;
datalines;
art
rice
bank
;
run;
```

SAS は次の結果をログに書き込みます。

```
141162164040040
162151143145040
142141156153040
```

$QUOTE*w*. 出力形式

データ値を二重引用符で囲んで書き込みます。

カテゴリ: 文字

配置: 左

構文

$QUOTE*w*.

説明

w

出力フィールドの幅を指定します。

デフォルト: 変数の長さが定義されていなければ 2。その他の場合、デフォルトは変数の長さ+ 2。

範囲: 2–32767

ヒント: *w* の幅を左と右の引用符を含められるよう十分な大きさにします。

詳細

次のリストは、$QUOTE*w*.出力形式を使用した場合に作成される出力です。これらの項目の例については、次の例を参照してください。

- データ値が引用符で囲まれていない場合、出力が二重引用符で囲まれます。
- データ値が引用符で囲まれていないが値に単一引用符が含まれている場合、次が実行されます。
  - データ値を二重引用符で囲む
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、値に二重引用符が含まれている場合、次が実行されます。
  - データ値を二重引用符で囲む
  - データ値に見つかった二重引用符を複製する
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、値に 2 つの隣接した単一引用符が含まれている場合、次が実行されます。
  - 値を二重引用符で囲む
  - 単一引用符を変更しない
- データ値が単一引用符で始まり終わっていて、二重引用符と隣接した単一引用符が含まれている場合、次が実行されます。
  - 値を二重引用符で囲む

- データ値に見つかった二重引用符を複製する
- 単一引用符を変更しない

- ターゲットフィールドの長さが文字列と引用符を含めるのに十分でない場合、そのフィールドの長さに合う長さの引用符付き文字列が返されます。

例

```
put name $quote20.;
```

name の値	結果
	----+----1----+----2
SAS	"SAS"
SAS's	"SAS's"
'ad"verb"'	"'ad""verb""'"
'ad''verb'	"'ad''verb'"
'"ad"''"verb"'	----+----1----+----2 "'""ad""''""verb""'"
deoxyribonucleotide	"deoxyribonucleotid" *

* deoxyribonucleotide は 19 文字です。引用符が追加される場合、文字列の長さは 21 文字です。テキストの末尾の文字 e が切り捨てられ、引用符が挿入されます。

$REVERJ*w*. 出力形式

文字データの末尾から順に書き込みます。空白も保持されます。

カテゴリ: 文字

配置: 右

構文

$REVERJ*w*.

説明

w

出力フィールドの幅を指定します。

デフォルト: *w* が指定されていない場合は 1。

範囲: 1–32767

比較

$REVERJ*w*.出力形式は$REVERS*w*.出力形式とほぼ同じですが、$REVERS*w*.では先頭の空白がすべて削除され、結果を左寄せにします。

例

```
put @1 name $reverj7.;
```

名前*	結果
	----+----1
ABCD###	DCBA
###ABCD	DCBA

* #文字は空白を表します。

$REVERS*w*. 出力形式

文字データの末尾から順に、左揃えで書き込みます。

カテゴリ: 文字

配置: 左

構文

$REVERS*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: *w* が指定されていない場合は 1。
範囲: 1–32767

比較

$REVERS*w*.出力形式は$REVERJ*w*.出力形式とほぼ同じですが、$REVERJ*w*.では結果を左寄せにしません。

例

```
put @1 name $revers7.;
```

名前*	結果
	----+----1
ABCD###	DCBA
###ABCD	DCBA

* #文字は空白を表します。

$UPCASEw. 出力形式

文字データを大文字に変換します。

カテゴリ: 文字

配置: 左

構文

$UPCASE*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

ハイフンやその他の記号などの特殊文字は、変更されません。

例

```
put @1 name $upcase9.;
```

name の値	結果
	----+----1
coxe-ryan	COXE-RYAN

$VARYINGw. 出力形式

可変長の文字データを書き込みます。

該当要素: DATA ステップ

カテゴリ: 文字

配置: 左

構文

$VARYING*w.length-variable*

説明

w
出力行または出力ファイルレコードに対する出力フィールドの最大幅を指定します。

デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

length-variable
文字変数の現在の値の長さを含む数値変数を指定します。*length-variable* の値は、INPUT ステートメントに記述されているフィールドから直接読み込むか、既存の SAS データセットから読み込むか、あるいは値を計算して求めます。

制限事項: *length-variable* には、配列参照は使えません。

要件 *length-variable* は、SAS ステートメント内で$VARYING*w*.の直後に指定しなければなりません。

ヒント:

length-variable の値が 0、負の値、あるいは欠損値ならば、出力フィールドには何も書き出しません。

length-variable の値が 0 以上で *w* 未満の場合は、*length-variable* によって指定された文字数を書き出します。

length-variable の値が *w* と同じか、それよりも大きい場合は、*w* 幅の列を書き出します。

詳細

文字値の長さがレコード間で異なる場合は、$VARYING*w*.を使用します。データ値を$VARYING*w*.で書き出した後、ポインタはこのデータ値の次の列に移動します。

例

例 1: 変数長を直接的に取得する

既存のデータセット変数に変数の長さが含まれています。データ値とその結果は、この SAS ステートメントの説明の後に示します。

```
put @10 name $varying12. varlen;
```

NAME は長さが 12 の文字変数です。文字長が 1 から 12 までの値が含まれます。VARLEN は、同一のデータセットに存在し、現在のオブザベーションの NAME の実際の長さを含む数値変数です。

name*の値	結果
	----+----1----+----2----+
New York 8	New York
Toronto 7	Toronto
Buenos Aires 12	Buenos Aires
Tokyo 5	Tokyo

* NAME の値が VARLEN の値の前に表示されます。

例 2: 変数長を間接的に取得する

LENGTH 関数を使用して、変数の長さを設定します。データ値とその結果は、これらの SAS ステートメントの説明の後に示します。

```
varlen=length(name);
put @10 name $varying12. varlen;
```

割り当てステートメントにより、可変長変数の長さが決定されます。変数 VARLEN には長さが含まれ、この変数は$VARYING12.出力形式の *length-variable* 引数になります。

値*	結果
	----+----1----+----2----+
New York	New York
Toronto	Toronto
Buenos Aires	Buenos Aires
Tokyo	Tokyo

* NAME の値が VARLEN の値の前に表示されます。

$*w*. 出力形式

標準文字データを書き出します。

カテゴリ: 文字

配置: 左

別名: $F*w*.

構文

$*w*.

説明

w

出力フィールドの幅を指定します。数値または列範囲を指定できます。

デフォルト: 変数の長さが定義されていなければ 1。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

比較

$*w*.出力形式と$CHAR*w*.出力形式は同じです。先頭の空白を切り捨てません。先頭の空白を切り捨てるには、LEFT 関数を使用して文字データを左寄せにするか、リスト出力でコロン(:)フォーマット修飾子と選択した出力形式と使用します。

例

```
put @10 name $5.;
put name $ 10-15;
```

name *の値	結果
	----+----1----+----2
#Cary	Cary
Tokyo	Tokyo

* #文字は空白を表します。

BESTw. 出力形式

SAS により最適な出力が選択されます。

カテゴリ: 数値

配置: 右

参照項目: “BESTw. Format: z/OS” in *SAS Companion for z/OS*

構文

BEST*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 12

範囲: 1–32

ヒント: 0 から.01 までの数字を出力する場合、過度に丸めないように 7 以上のフィールド幅を使用します。0 から-.01 までの数字を出力する場合は、8 以上のフィールド幅を使用します。

詳細

出力形式が数値の書き出しに指定されていない場合、BEST*w*.出力形式がデフォルトの出力形式として使用されます。BEST*w*.出力形式は、数字を次のように書き出します。

- 値は、幅によって決定される最大有効桁数で書き出されます。
- 整数は小数なしで書き出されます。
- 小数を含む数字は、小数点の前と後に必要な桁数、または幅によって許可される桁数で書き出されます。
- 指定幅内で書き出し可能な値は、末尾のゼロなしで書き出されます。
- 指定幅内で書き込みできない値は、幅によって決定される可能な小数点以下の最大桁数で書き出されます。

- 極値は、指数表記で書き出されることがあります。

SAS では使用される出力形式にかかわらず完全な値を保存します。

比較

- BEST*w*.出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。整数は小数なしで出力されます。
- D*w.p* 出力形式は数字を指定した有効桁数で書き出し、BEST*w* 出力形式よりも小数点揃えを行います。
- BESTD*w.p* 出力形式は BEST*w*.出力形式と D*w.p* 出力形式を組み合わせたものです。すべての数値データに適用されます。また、BEST*w*.出力形式よりも小数点揃えに優れています。
- *w.d* 出力形式は小数点揃えを行いますが、すべての数字が必ずしも同じ有効桁数で表示されません。

例

ステートメントによって、次の結果が作成されます。

SAS ステートメント	結果
	`----+----1`
`x=1257000;` `put x best6.;`	`1.26E6`
`x=1257000;` `put x best3.;`	`1E6`

関連項目:

出力形式:

- “BESTDw.p 出力形式” (55 ページ)

BESTD*w.p* 出力形式

同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。整数は小数なしで出力します。

カテゴリ: 数値

配置: 右

構文

BESTD*w.p*

説明

w

出力フィールドの幅を指定します。

デフォルト: 12

範囲: 1–32

p

有効桁数を指定します。この引数はオプションです。

デフォルト: 3

範囲: 0 - *w*–1

要件 *w* より小さい数にします。

ヒント: *p* が省略される、または 0 が指定されると、*p* は 3 に設定されます。

詳細

BESTD*w.p* 出力形式は、同じような大きさの値グループ内で小数点を揃えて値を書き出します。整数は小数点なしで出力されます。*p* の値が大きくなると、出力されるデータ値の有効桁数が増え、小数点揃えのシフトが増す可能性があります。*p* の値が小さくなると、出力されるデータ値の有効桁数が減り、小数点揃えの可能性が高くなります。

出力形式は、値の範囲に対して、値をより少ない小数点以下の桁数で表示できる場合でも、出力に適した小数点以下の桁数を選択します。

比較

- BEST*w.*出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。整数は小数なしで出力されます。
- D*w.p* 出力形式は数字を指定した有効桁数で書き出し、BEST*w* 出力形式よりも小数点揃えを行います。
- BESTD*w.p* 出力形式は BEST*w.*出力形式と D*w.p* 出力形式を組み合わせたものです。すべての数値データに適用されます。また、BEST*w.*出力形式よりも小数点揃えに優れています。
- *w.d* 出力形式は小数点揃えを行いますが、すべての数字が必ずしも同じ有効桁数で表示されません。

例

```
put x bestd14.;
```

データ行	結果
	----+----1----+
12345	12345
123.45	123.4500000
1.2345	1.2345000
.12345	0.1234500

データ行	結果
1.23456789	1.2345679

関連項目:

出力形式:

- “BESTw. 出力形式” (54 ページ)
- “Dw.p 出力形式” (68 ページ)

BINARY*w.* 出力形式

数値を 2 進表現に変換します。

カテゴリ: 数値

配置: 左

構文

BINARY*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 8
範囲: 1–64

比較

BINARY*w.*は数値を 2 進表現に変換します。$BINARY*w.*出力形式は文字値を 2 進表現に変換します。

例

```
put @1 x binary8.;
```

x の値	結果
	----+----1
123.45	01111011
123	01111011
-123	10000101

B8601DA*w*. 出力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して日付値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DA*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 10
範囲: 8–10

詳細

B8601DA 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd* を使用して日付値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
0 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
put bda b8601da.;
```

bda の値	結果
18885	20110915
18628	20110101

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

B8601DN*w*. 出力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して、日時値から日付を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DN*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 10
範囲: 8–10

詳細

B8601DN 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd* を使用して、日時値から日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
put bdn b8601dn.;
```

bdn の値	結果
1631664000	20110915

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

B8601DT*w.d* 出力形式

日時値を ISO 8601 規格の基本表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

B8601DT*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 19
範囲: 15–26

d
秒の端数を表す秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

B8601DT 出力形式は、ISO 8601 規格の基本表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して、日時値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
put bdt b8601dt.;
```

bdt の値	結果
---- +----1	
1631721180	20110915T155300

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

B8601DZw.出力形式

ISO 8601 規格の日時とタイムゾーンの基本表記 *yyyymmdd*T*hhmmss*+0000 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Element 5.4.1, complete representation

構文

B8601DZ*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 26
範囲: 20–35

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601DZ 出力形式は、次の ISO 8601 規格の基本日時表記のいずれかを使用して、基準子午線の日時に対応した SAS 日時値を書き出します。

- *yyyymmdd*T*hhmmss*+0000

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *yyyymmdd*T*hhmmss*Z

  注: この形式は、*w* の長さが+0000 タイムゾーン表記に十分でない場合に使用します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+0000
基準子午線(イギリスのグリニッジ)の UTC 時間を示します。

タイムゾーンオフセットが指定された ISO 860 規格 1 の時間値または日時値は、オフセットに示された時間数と分数によって調整され、基準子午線(イギリスのグリニッジ)の時間または日時として処理されます。B8601DZ 出力形式は、常に基準子午線オフセット値として+0000 を使用して日時値を書き出します。+0000 以外の UTC オフセットを使用して日時を書き出すには、“B8601LZw.出力形式” (62 ページ)を参照してください。

制限:短い形式+00 は、サポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000UTC 時間であることを示します。出力形式の幅が+0000 表記をサポートしない場合に Z が使用されます。

例

```
put bdz b8601dz20.;
```

日時値	bdz の値	結果
20110915T155300+0500	1631703180*	20110915T105300+0000
20110915T155300Z	1631721180	20110915T155300+0000

* この ISO 8601 値のタイムゾーンオフセットは、5 時間です。SAS にこの値が読み込まれる際、SAS 日時値に 5 時間の調整が行われます。結果列には 5 時間が調整された日時値が表示されます。

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

B8601LZ*w*.出力形式

ISO 8601 規格の基本表記 *hhmmss*+|-*hhmm* を使用して、ローカル時間と UTC 間のタイムゾーンオフセット差を追加することにより、時間値をローカル時間として書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Elements 5.3.3 and 5.3.4.2

構文

B8601LZ*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 14

範囲: 9–20

詳細

B8601LZ 出力形式は、ISO 8601 規格の基本表記 *hhmmss+|–hhmm* を使用して時間値を調整せずに書き出し、ローカル SAS セッションに対応した UTC タイムゾーンオフセットを追加します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|-*hhmm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hhmm* (つまり+または-を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

SAS では UTC 時間を B8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 000000 と 235959 の間になるように値を調整します。B8601LZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

例

この PUT ステートメントは東部標準タイムゾーンに対応した時間を書き出します。

```
put blz b8601lz.;
```

blz の値	結果
46380	125300-0500

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

B8601TM*w.d* 出力形式

ISO 8601 規格の基本表記 *hhmmss<ffff>*を使用して、時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

B8601TM*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 8
範囲: 6–15

d
秒の端数を表す秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

B8601TM 出力形式は、ISO 8601 規格の基本時間表記 *hhmmss<ffffff>*を使用して、SAS 時間値を書き出します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
put btm b8601tm.;
```

btm の値	結果
57180	155300

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

B8601TZw.出力形式

ISO 8601 規格の基本時間表記 *hhmmss+|–hhmm* を使用して、時間値を協定世界時(UTC)に調整し、時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

サポート: ISO 8601 Elements 5.3.3 and 5.3.4

構文

B8601TZ*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 14

範囲: 9–20

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601TZ 出力形式は、時間値を基準子午線の時間に調整し、その時間値を次の ISO 8601 規格の基本時間表記のいずれかで書き出します。

- *hhmmss+|–hhmm*

  注: この形式は、*w* の長さがこの時間表記に十分である場合に使用します。

- *hhmmss*Z

  注: この形式は、*w* の長さが+|-*hhmm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|–*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hhmm* (つまり+または-を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000 UTC 時間であることを示します。

SAS では UTC 時間を B8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 000000 と 240000 の間になるように値を調整します。B8601TZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

比較

000000 と 240000 の間の時間値に対して、B8601TZ 出力形式は時間値を基準子午線の時間に調整し、調整後の値を国際規格の拡張時間表記で書き出します。B8601LZ 出力形式は、時間を調整せず、ローカル SAS セッションの UTC タイムゾーンオフセットを使用して時間値を国際規格の拡張時間表記で書き出します。

例

```
put btz b8601tz.;
```

b8z の値	結果
73441	202401+0000

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

COMMA*w.d* 出力形式

3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

COMMA*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 6

範囲: 1–32

ヒント: *w* の幅に数値、カンマ、オプションの小数点の書き出しに十分な値を指定します。

d
数値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–31

要件 *w* より小さい数にします。

詳細

COMMA*w.d* 出力形式は、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

比較

- COMMA*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMAX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- COMMA*w.d* 出力形式は DOLLAR*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

```
put @10 sales comma10.2;
```

sales の値	結果
	----+----1----+----2
23451.23	23,451.23
123451.234	123,451.23

関連項目:

出力形式:

- “COMMAXw.d 出力形式” (67 ページ)
- “DOLLARw.d 出力形式” (78 ページ)

COMMAX*w.d* 出力形式

3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

COMMAX*w.d*

説明

w
出力フィールドの幅を指定します。この引数はオプションです。

デフォルト: 6

範囲: 1–32

ヒント: *w* の幅に数値、カンマ、オプションの小数点の書き出しに十分な値を指定します。

d

数値の小数点以下の桁数を指定します。

範囲: 0–31

要件 *w* より小さい数にします。

詳細

COMMAX*w.d* 出力形式は、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

比較

COMMA*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMAX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。

例

```
put @10 sales commax10.2;
```

sales の値	結果
	----+----1----+----2
23451.23	23.451,23
123451.234	123.451,23

D*w.p* 出力形式

値の桁数をできる限り最大にし、同じような大きさの数値に対しては小数点以下の桁数を揃えて出力します。

カテゴリ: 数値

配置: 右

構文

D*w.p*

説明

w

出力フィールドの幅を指定します。この引数はオプションです。

デフォルト: 12

範囲: 1–32

p

有効桁数を指定します。この引数はオプションです。

デフォルト: 3

範囲: 0–9

要件 *p* は *w* より小さい数にします。

ヒント:

p が省略される、または 0 が指定されると、*p* は 3 に設定されます。

0 が指定した有効桁数の場合、*w.d* 出力形式を D*w.p* 出力形式の代わりに使用します。

詳細

D*w.p* 出力形式は、同じような大きさの値グループ内で小数点を揃えて値を書き出します。*p* の値が大きくなると、出力されるデータ値の有効桁数が増え、小数点揃えのシフトが増す可能性があります。*p* の値が小さくなると、出力されるデータ値の有効桁数が減り、小数点揃えの可能性が高くなります。

比較

- BEST*w.*出力形式は、出力フィールドに可能な限りの有効桁数を書き出しますが、数字の大きさが異なる場合に小数点以下の桁数を揃えません。
- D*w.p* は数字を指定した有効桁数で書き出し、BEST*w* 出力形式よりも小数点揃えを行います。
- BESTD*w.p* 出力形式は BEST*w.*出力形式と D*w.p* 出力形式を組み合わせたものです。すべての数値データに適用されます。また、BEST*w.*出力形式よりも小数点揃えに優れています。
- *w.d* 出力形式は小数点揃えを行いますが、すべての数字が必ずしも同じ有効桁数で表示されません。

例

```
put @1 x d10.4;
```

x の値	結果
	----+----1
12345	12345.0
1234.5	1234.5
123.45	123.45000
12.345	12.34500
1.2345	1.23450
.12345	0.12345

関連項目:

出力形式:

- “BESTDw.p 出力形式” (55 ページ)

DATE*w.* 出力形式

SAS 日付値を *ddmmmyy*、*ddmmmyyyy* または *dd-mmm-yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DATE*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 7

範囲: 5–11

ヒント: 4 桁の年を日、月、年の間を区切らずに出力する場合は、幅に 9 を指定します。4 桁の年を日、月、年の間をハイフンで区切って出力する場合は、幅に 11 を指定します。

詳細

DATE*w.*出力形式は、SAS 日付値を *ddmmmyy*、*ddmmmyyyy* または *dd-mmm-yyyy* 形式で書き出します。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

例

例の表では、19068 を入力値として使用します。この値は 2012 年 3 月 16 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put day date5.;`	`16MAR`
`put day date6.;`	`16MAR`
`put day date7.;`	`16MAR12`
`put day date8.;`	`16MAR12`

SAS ステートメント	結果
put day date9.;	16MAR2012
put day date11.;	16-MAR-2012

関連項目:

関数:

- “DATE Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)

DATEAMPM*w.d* 出力形式

SAS 日時値を、AM または PM を付けて *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DATEAMPM*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 19

範囲: 7–40

ヒント: AM または PM を出力するには、*w* に少なくとも 13 を指定する必要があります。幅の値が 10 と 12 の間の場合、24 時間表記が出力されます。

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–39

要件 *w* より小さい数にします。

注: *w*–*d*< 17 の場合、小数点以下の値は切り捨てられます。

詳細

DATEAMPM*w.d* 出力形式は、SAS 日付値を *ddmmmyy:hh:mm:ss.ss* の形式で書き出します。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy
年を表す 2 桁の整数です。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

比較

DATEAMPM*w.d* 出力形式は DATETIME*w.d* 出力形式とほぼ同じですが、DATEAMPM*w.d* は時間の最後に AM または PM を出力します。

例

例の表では 1650538894 を入力値として使用します。この値は 2012 年 4 月 20 日、午前 11:01:34 に相当する SAS 日時値です。

SAS ステートメント	結果
	----+----1----+----2----+
put event dateampm.;	20APR12:11:01:34 AM
put event dateampm7.;	20APR12
put event dateampm10.;	20APR:11
put event dateampm13.;	20APR12:11 AM
put event dateampm22.2;	20APR12:11:01:34.00 AM

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)

DATETIME*w.d* 出力形式

SAS 日時値を *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

制限事項: *w–d*< 17 の場合、小数点以下の値は切り捨てられます。

構文

DATETIME*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 16

範囲: 7–40

ヒント: 日付、時間、秒を含む SAS 日時値の出力には、*w* 値として少なくとも 16 を指定する必要があります。秒の端数も含めて値を返す場合、*w* にさらに 2 桁、*d* に値を追加します。

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–39

要件 *w* より小さい数にします。

詳細

DATETIME*w.d* 出力形式は、SAS 日時値を *ddmmmyy:hh:mm:ss.ss* 形式で書き出します。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy
年を表す 2 桁の整数です。

hh
時間を 24 時間表記で表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

例

例の表では 1668138559 を入力値として使用します。この値は 2012 年 11 月 10 日、午前 3:49:19 に相当する SAS 日時値です。

SAS ステートメント	結果
	----+----1----+----2----+
put event datetime.;	10NOV12:03:49:19
put event datetime7.;	10NOV12
put event datetime12.;	10NOV12:03
put event datetime18.;	10NOV12:03:49:19
put event datetime18.1;	10NOV12:03:49:19.0
put event datetime19.;	10NOV2012:03:49:19

SAS ステートメント	結果
put event datetime20.1;	10NOV2012:03:49:19.0
put event datetime21.2;	10NOV2012:03:49:19.00

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “TIMEw.d 出力形式” (152 ページ)

関数:

- “DATETIME Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DATETIMEw. 入力形式” (261 ページ)
- “TIMEw. 入力形式” (324 ページ)

DAY*w.* 出力形式

SAS 日付値の日付部分を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DAY*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 2
範囲: 2–32

例

例の表では、19158 を入力値として使用します。この値は 2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+

SAS ステートメント	結果
`put date day2.;`	`14`

DDMMYY*w*. 出力形式

SAS 日付値を *ddmm<yy>yy* または *dd*/*mm*/*<yy>yy* 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

DDMMYY*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作: *w* の値が 2 から 5 までの場合、可能な限りで日と月だけを出力します。*w* が 7 の場合、年を 2 桁で表し、スラッシュを使わずに出力します。

詳細

DDMMYY*w*.出力形式は、SAS 日付値を *ddmm<yy>yy* または *dd*/*mm*/*<yy>yy* 形式で書き出します。

dd
月の日を表す整数です。

/
区切り文字です。

mm
月を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

これらの例では、19351 を入力値として使用します。この値は、2012 年 12 月 24 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date ddmmyy5.;`	`24/12`

SAS ステートメント	結果
put date ddmmyy6.;	241212
put date ddmmyy7.;	241212
put date ddmmyy8.;	24/12/12
put date ddmmyy10.;	24/12/2012

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYxw. 出力形式” (76 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “MDY Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

DDMMYY*xw.* 出力形式

SAS 日付値を *ddmm<yy>yy* or *dd-mm-yy<yy>*形式で書き出します。出力形式名の *x* は、日、月、年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

DDMMYY*xw.*

説明

x

区切り文字を示します。または、区切り文字を日、月、年の間に挿入しないことを示します。*x* の値として、次の値が有効です。

B
空白で区切ります。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作:
w の値が 2 から 5 までの場合、日と月が可能な範囲で表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。
x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

DDMMYY*xw*.出力形式は、SAS 日付値を *ddmm<yy>yy* または *ddxmmx<yy>yy* 形式で書き出します。

dd
月の日を表す整数です。

x
指定された区切り文字です。

mm
月を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19137 を入力値として使用します。この値は、2012 年 5 月 24 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date ddmmyyc5.;`	`24:05`
`put date ddmmyyd8.;`	`24-05-12`
`put date ddmmyyp10.;`	`24.05.2012`

SAS ステートメント	結果
put date ddmmyyn8.;	24052012

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYxw. 出力形式” (111 ページ)
- “YYMMDDxw. 出力形式” (177 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DDMMYYw. 入力形式” (263 ページ)

DOLLAR*w.d* 出力形式

先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

DOLLAR*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 2–32

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–31

要件 *w* より小さい数にします。

詳細

DOLLAR*w.d* 出力形式は、先頭にドル($)記号を付け、3 桁ごとにカンマを入れ、小数部分との区切りにはピリオドを使用して数値を出力します。

ドル記号文字($)のコードの 16 進表現は、EBCDIC のシステムでは 5B、ASCII のシステムでは 24 です。これらのコードが表す通貨記号は、他の国では異なる場合があります。しかし、DOLLAR*w.d* は、常にこれらのコードのどちらかを生成します。別の通貨記号が必要な場合、独自の出力形式を FORMAT プロシジャを使用して定義します。詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

比較

- DOLLAR*w.d* 出力形式は DOLLARX*w.d* 出力形式とほぼ同じですが、DOLLARX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- DOLLAR*w.d* 出力形式は COMMA*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

```
put @3 netpay dollar10.2;
```

netpay の値	結果
	----+----1----+
1254.71	$1,254.71

関連項目:

出力形式:

- “COMMAw.d 出力形式” (66 ページ)
- “DOLLARXw.d 出力形式” (79 ページ)

DOLLARX*w.d* 出力形式

先頭にドル($)記号を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

カテゴリ: 数値

配置: 右

構文

DOLLARX*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 2–32

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

要件 *w* より小さい数にします。

詳細

DOLLARX*w.d* 出力形式は、先頭にドル($)記号を付け、3 桁ごとにピリオドを入れ、小数部分との区切りにはカンマを使用して数値を出力します。

ドル記号文字($)のコードの 16 進表現は、EBCDIC のシステムでは 5B、ASCII のシステムでは 24 です。これらのコードが表す通貨記号は、他の国では異なる場合があります。しかし、DOLLARX*w.d* は、常にこれらのコードのどちらかを生成します。別の通貨記号が必要な場合、独自の出力形式を FORMAT プロシジャを使用して定義します。詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

比較

- DOLLARX*w.d* 出力形式は DOLLAR*w.d* 出力形式とほぼ同じですが、DOLLARX*w.d* 出力形式では小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。
- DOLLARX*w.d* 出力形式は COMMAX*w.d* 出力形式とほぼ同じですが、COMMA*w.d* 出力形式は先頭のドル記号を出力しません。

例

```
put @3 netpay dollarx10.2;
```

netpay の値	結果
	----+----1----+
1254.71	$1.254,71

関連項目:

出力形式:

- “COMMAXw.d 出力形式” (67 ページ)
- “DOLLARw.d 出力形式” (78 ページ)

DOWNAMEw. 出力形式

SAS 日付値から曜日名を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DOWNAME*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 9

範囲: 1–32

ヒント: *w* を省略すると、曜日名全体が出力されます。

詳細

必要に応じて、SAS により出力形式の幅に合うように曜日名が切り捨てられます。たとえば、DOWNAME2.出力形式を指定すると、曜日名の最初の 2 文字が出力されます。

例

例の表では、19137 を入力値として使用します。この値は 2012 年 5 月 24 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date downame.;`	`Thursday`

関連項目:

出力形式:

- “WEEKDAYw. 出力形式” (163 ページ)

DTDATEw. 出力形式

入力として日時値が必要です。SAS 日付値を *ddmmmyy* または *ddmmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTDATE*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 7
範囲: 5–9
ヒント: 4 桁の年を出力する場合は幅に 9 を指定します。

詳細

DTDATE*w* は、SAS 日付値を *ddmmmyy* または *ddmmmyyyy* 形式で書き出します。

dd
月の日を表す整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTDATE*w.*出力形式は、DATE*w.*出力形式と同じ種類の出力を作成します。相違点は、DTDATE*w.*出力形式には日時値が必要な点です。

例

例の表では日時値の 16APR2012:10:00:00 を入力として使用し、DTDATE*w.*出力形式に対応した 2 桁の年と 4 桁の年を出力します。

SAS ステートメント	結果
	`----+----1`
`put trip_date=dtdate.;`	`16APR12`
`put trip_date=dtdate9.;`	`16APR2012`

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)

DTMONYY*w.* 出力形式

SAS 日時値を *mmmyy* または *mmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTMONYY*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 5

範囲: 5–7

詳細

DTMONYY*w.*出力形式は、SAS 日時値を *mmmyy* または *mmmyyyy* 形式で書き出します。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTMONYY*w.*出力形式と MONYY*w.*出力形式は、日付値を書き出すという点でほぼ同じです。相違点は、DTMONYY*w.*が入力として日時値が必要であり、MONYY*w.*が SAS 日付値が必要である点です。

例

例の表では 1665986932 を入力値として使用します。この値は 2012 年 10 月 16 日、午前 06:08:52 に相当する SAS 日時値です。

SAS ステートメント	結果
	`----+----1`
`put date dtmonyy.;`	`OCT12`
`put date dtmonyy5.;`	`OCT12`
`put date dtmonyy6.;`	` OCT12`
`put date dtmonyy7.;`	`OCT2012`

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)
- “MONYYw. 出力形式” (119 ページ)

DTWKDATX*w*. 出力形式

SAS 日時値を *day-of-week*, *dd month-name yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTWKDATX*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 29

範囲: 3–37

詳細

DTWKDATX*w*.出力形式は、SAS 日付値を *day-of-week*, *dd month-name*, *yy* (または *yyyy*)形式で書き出します。

day-of-week
曜日名の最初の 3 文字か、曜日名全体です。

dd
月の日を表す整数です。

month-name
月名の最初の 3 文字か、月名全体です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

DTWKDATX*w*.出力形式は、日付値を書き出すという点で WEEKDATX*w*.出力形式とほぼ同じです。相違点は、DTWKDATX*w*.が入力として日時値が必要であり、WEEKDATX*w*.が SAS 日付値が必要である点です。

例

例の表では 1665986932 を入力値として使用します。この値は 20012 年 10 月 16 日、午前 06:08:52 に相当する SAS 日時値です。

SAS ステートメント	結果
	`----+----1----+----2----+`
`put date dtwkdatx.;`	`Tuesday, 16 October 2012`
`put date dtwkdatx3.;`	`Tue`

SAS ステートメント	結果
put date dtwkdatx8.;	Tue
put date dtwkdatx25.;	Tuesday, 16 Oct 2012

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)
- “WEEKDATXw. 出力形式” (162 ページ)

DTYEARw. 出力形式

SAS 日時値を *yy* または *yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTYEAR*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 4

範囲: 2–4

詳細

DTYEAR*w.*出力形式は、日付値を書き出すという点で YEAR*w.*出力形式とほぼ同じです。相違点は、DTYEAR*w.*が入力として日時値が必要であり、YEAR*w.*が SAS 日付値が必要である点です。

例

例の表では 1665986932 を入力値として使用します。この値は 2012 年 10 月 16 日、午前 06:08:52 に相当する SAS 日時値です。

SAS ステートメント	結果
	----+----1
put date dtyear.;	2012
put date dtyear2.;	12

SAS ステートメント	結果
put date dtyear3.;	12
put date year4.;	2012

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)
- “YEARw. 出力形式” (173 ページ)

DTYYQC*w*. 出力形式

SAS 日時値から、年と四半期をコロン(:)で区切って書き出します。

カテゴリ: 日付と時間

配置: 右

構文

DTYYQC*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 4–6

詳細

DTYYQC*w*.出力形式は、SAS 日時値を *yy* または *yyyy* 形式の年にコロン(:)と四半期を示す数値を続けて書き出します。

例

例の表では 1665986932 を入力値として使用します。この値は 2012 年 10 月 16 日、午後 06:08:52 に相当する SAS 日時値です。

SAS ステートメント	結果
	----+----1
put date dtyyqc.;	12:4
put date dtyyqc4.;	12:4
put date dtyyqc5.;	12:4

SAS ステートメント	結果
put date dtyyqc6.;	2012:4

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)

Ew. 出力形式

数値を指数表記で書き出します。

カテゴリ: 数値

配置: 右

参照項目: “Ew. Format: z/OS” in *SAS Companion for z/OS*

構文

E*w*.

説明

w
出力フィールドの幅を指定します。出力フィールドには 14 桁までの有効桁数を表示できます。

デフォルト: 12

範囲: 7–32

詳細

値を指数表記でフォーマットする際に、E 出力形式は結果の最初のカラムをマイナス記号を書き込む場所として確保し、14 桁までの有効桁数にフォーマットします。

例

```
put @1 x e10.;
```

x の値	結果
	----+----1
1257	1.257E+03
-1257	-1.257E+03

E8601DA*w*.出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して日付値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DA

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DA*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 10
要件 10 である必要があります。

詳細

E8601DA 出力形式は、ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
put eda e8601da.;
```

eda の値	結果
19251	2012-09-15

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

E8601DN*w*. 出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して、SAS 日時値から日付を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DN

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DN*w*.

説明

w
幅を指定します。
デフォルト: 10
要件 幅は 10 である必要があります。

詳細

E8601DN 出力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd* を使用して日付を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
put edn e8601dn.;
```

edn の値	結果
1663308532	2012-09-15

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

E8601DT*w.d* 出力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd*T*hh:mm:ss.ffffff* を使用して日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DT

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DT*w.d*

説明

w
幅を指定します。
デフォルト: 19
範囲: 19–26

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

E8601DT 出力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd*T*hh:mm:ss.ffffff* を使用して日時値を書き出します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
put edt e8601dt25.3.;
```

edt の値	結果
1663343580.2	2012-09-15T15:53:00.234

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

E8601DZ*w*.出力形式

ISO 8601 規格の日時とタイムゾーンの拡張表記 *yyyy-mm-dd*T*hh:mm:ss*+00:00 を使用して、基準子午線の協定世界時(UTC)の日時値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DZ

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DZ*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 26
範囲: 20–35

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601DZ 出力形式は、SAS 日時値を次の ISO 8601 規格の拡張日時表記のいずれかを使用して書き出します。

- *yyyy-mm-dd*T*hh:mm:ss*+00:00

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *yyyy-mm-dd*T*hh:mm:ss*Z

  注: この形式は、*w* の長さが+00:00 タイムゾーン表記に十分でない場合に使用します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 24 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+00:00
時間が基準子午線(イギリスのグリニッジ)に対応した時間であることを示します。

タイムゾーンオフセットが指定された ISO 860 規格 1 の時間値または日時値は、オフセットに示された時間数と分数によって調整され、基準子午線(イギリスのグリニッジ)の時間または日時として処理されます。E8601DZ 出力形式は、常に+00:00 の基準子午線オフセット値を使用して日時値を書き出します。+00:00 以外の UTC オフセットを使用して日時を書き出すには、“E8601LZw.出力形式” (92 ページ)を参照してください。

制限:短い形式+00 は、サポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。出力形式の幅が+00:00 表記に十分でない場合は、Z を使用します。

例

```
put edz e8601dz.;
```

edz の値	結果
1663332780	2012-09-15T12:53:00+00:00

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

E8601LZ*w*.出力形式

ISO 8601 規格の拡張時間表記 *hh:mm:ss*+|-*hh:mm* を使用して、ローカル SAS セッションに対応した協定世界時(UTC)のオフセットを追加することにより、時間値をローカル時間として書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601LZ

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601LZ*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 14

範囲: 9–20

詳細

E8601LZ 出力形式は、次の ISO 8601 規格の拡張時間表記のいずれかを使用して時間値を調整せずに書き出し、ローカル SAS セッションに対応した UTC タイムゾーンオフセットを追加します。

- *hh:mm:ss*+|–*hh:mm*

  注: この形式は、*w* の長さがこの時間表記に十分である場合に使用します。

- *hh:mm:ss*Z

  注: この形式は、*w* の長さが+|- *hh:mm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|-*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hh:mm* (つまり+または-に続く 5 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+02:00 は基準子午線の東部での 2 時間差を表し、–06:00 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。

SAS では、時間値を *hh:mm.ffffff* 形式を使用して書き出し、ローカル SAS セッションに対応した基準子午線とのタイムゾーンオフセットのタイムゾーンインジケータ+|-*hh:mm* または Z を追加します。Z タイムゾーンインジケータは、出力形式の長さの値が 14 未満の場合に使用されます。

両方のゾーンインジケータを使用して同じ時間を書き出す場合、UTC に基づく 2 つの異なる時間を示します。たとえば、ローカル SAS セッションでアメリカの東部標準時が使用され、その時間値が 45824 である場合、12:43:44-04:00 または 12:43:44Z が書き出されます。時間 12:43:44–04:00 は、基準子午線の時間の 16:43:44+00:00 です。Z は、その時間が基準子午線の時間である、つまり、12:43:44+00:00 であることを示します。

SAS では UTC 時間を E8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間

が 00:00:00 と 24:00:00 の間になるように値を調整します。E8601LZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

例

この PUT ステートメントは東部標準タイムゾーンに対応した時間を書き出します。

```
put elz e8601lz.;
```

elz の値	結果
46380	12:53:00-5:00

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理” (13 ページ)

E8601TM*w.d* 出力形式

ISO 8601 規格の拡張表記 *hh:mm:ss.ffffff* を使用して時間値を書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TM

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.3.1.1, complete representation, and 5.3.1.3, representation of decimal fractions

構文

E8601TM*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 8
範囲: 8–15

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

E8601TM 出力形式は、SAS 時間値を ISO 8601 規格の拡張時間表記 *hh:mm:ss* を使用して書き出します。*ffffff*:

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

.ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
put etm e8601tm.;
```

etm の値	結果
57180	15:53:00

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

E8601TZ*w.d* 出力形式

時間値を協定世界時(UTC)に調整し、時間値を ISO 8601 規格の拡張表記 *hh*:*mm*:*ss*+|-*hh*:*mm* を使用して書き出します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TZ

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601TZ*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 14
範囲: 9–20

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601TZ 出力形式は、時間値を次の ISO8601 規格の拡張時間表記のいずれかを使用して書き出します。

- *hh:mm:ss*+|–*hh:mm*

  注: この形式は、*w* の長さがこのタイムゾーン表記に十分な場合に使用します。

- *hh:mm:ss*Z

  注: この形式は、*w* の長さが+|- *hh:mm* タイムゾーン表記に十分でない場合に使用します。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

+|-*hh:mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hh:mm* (つまり+または-に続く 5 文字)である必要があります。

制限: 短い形式+|–*hh* は、サポートされていません。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+02:00 は基準子午線の東部での 2 時間差を表し、–06:00 は基準子午線の西部での 6 時間差を表します。

Z
基準子午線(イギリスのグリニッジ)に対応した時間、つまり+00:00 UTC 時間であることを示します。

SAS では UTC 時間を B8601TZ 入力形式を使用して読み込む際に、オフセット調整後の時間が 24 時間を超えていたり 00 時間未満である場合、オフセット調整後の時間が 00:00:00 と 24:00:00 の間になるように値を調整します。B8601TZ 出力形式をこの時間範囲外の時間に適用する場合、時間はアスタリスクを使用してフォーマットされ、その値が範囲外であることが示されます。

比較

00:00:00 と 24:00:00 の間の時間値に対して、B8601TZ 出力形式は時間値を基準子午線の時間に調整し、調整後の値を国際規格の拡張時間表記で書き出します。B8601LZ 出力形式は時間を調整せず、ローカル SAS セッションの UTC タイムゾーンオフセットを使用して時間値を国際規格の拡張時間表記で書き出します。

例

```
put etz e8601tz.;
```

etz の値	結果
17024	04:43:44+00:00
85424	23:43:44+00:00

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の処理" (13 ページ)

FLOAT*w.d* 出力形式

数値に 10 の *d* 乗を掛けた、単精度のネイティブ浮動小数点値を書き出します。

カテゴリ: 数値

配置: 左

構文

FLOAT*w.d*

説明

w
出力フィールドの幅を指定します。
要件 幅は 4 である必要があります。

d
値を乗算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0-31

詳細

この出力形式は、浮動数値が倍精度の切り捨て値と異なる動作環境に使用します。FLOAT4.によって書き出される値は通常、使用している動作環境で実行される、単精度値を扱う他の外部プログラムによって読み込まれるための値です。

注: フォーマットされる値が欠損値、あるいは単精度のネイティブ浮動小数点の範囲外である場合、ゼロの単精度の値が生成されます。

IBM メインフレームシステム上では、4 バイトの浮動小数点数は 8 バイトに切り捨てられた浮動小数点数と同じです。ただし、IEEE 浮動小数点数規格の動作環境(IBM PC、UNIX など)では、4 バイトの浮動小数点数は倍精度の切り捨て値と異なります。つまり、RB4.出力形式は、FLOAT4.出力形式と同じ結果を生成しません。IEEE 以外の浮動小数点表現にもこの同じ特徴がみられます。

比較

次の表は、各種プログラミング言語での浮動表記を比較したものです。

言語	浮動表記
SAS	FLOAT4
Fortran	REAL+4
C	float
IBM 370 ASM	E

言語	浮動表記
PL/I	FLOAT BIN(21)

例

```
put x float4.;
```

x の値	結果*
1	3F800000

* 結果は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

FRACT*w*. 出力形式

数値を分数に変換します。

カテゴリ: 数値

配置: 右

構文

FRACT*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 10
範囲: 4–32

詳細

1 を 3 で割ると、値は 0.33333333 となります。この値を 1/3 として書き出すには、FRACT*w*.出力形式を使用します。FRACT*w*.は分数を約分して書き出します。50/100 は 1/2 となります。

例

```
put x fract8.;
```

x の値	結果
	----+----1
0.6666666667	2/3
0.2784	174/625

HEX*w*. 出力形式

バイナリ実数(浮動小数点)値を 16 進表現に変換します。

カテゴリ: 数値

配置: 左

参照項目: “HEXw. Format: Windows” in *SAS Companion for Windows*
“HEXw. Format: UNIX” in *SAS Companion for UNIX Environments*
“HEXw. Format: z/OS” in *SAS Companion for z/OS*

構文

HEX*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 1–16

ヒント: *w*< 16 の場合、HEX*w*.出力形式はバイナリ実数を固定小数点数に変換してから 16 進表現に変換します。また、負数を 2 つの補数にして右寄せで書き出します。*w* が 16 の場合、HEX*w*.は浮動小数点数値を 16 進表現で表示します。

詳細

どの動作環境でも、HEX*w*.によって書き出される最下位のバイトは右端のバイトになります。一部の動作環境では、最小有効数字を先頭のバイトとして整数を保存します。HEX*w*.出力形式は、バイトの桁順序にかかわらず、どの動作環境でも同じように結果を生成します。

注: 動作環境によって浮動小数点数の保存方法は異なります。しかし、HEX16.出力形式は、使用されている動作環境での保存方法と同じように、16 進表現の浮動小数点数を書き出します。

比較

HEX*w*.数値出力形式と$HEX*w*.文字出力形式は、ともに値を 16 進表現に変換します。

例

```
put @8 x hex8.;
```

x の値	結果
	----+----1----+----2
35.4	00000023

x の値	結果
88	00000058
2.33	00000002
-150	FFFFFF6A

HHMM*w.d* 出力形式

SAS 時間値を時間および分として *hh:mm* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

HHMM*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 5

範囲: 2–20

d
分値の小数点以下の桁数を指定します。小数点以下の桁数は、分の端数を示します。この引数はオプションです。

デフォルト: 0

範囲: 0–19

要件 *w* より小さい数にします。

詳細

HHMM*w.d* 出力形式は、SAS 時間値を *hh:mm* 形式で書き出します。

hh
整数です。

注: *hh* が 1 桁の場合、HHMM*w.d* はその桁の前に先頭の空白を置きます。たとえば、HHMM*w.d.*出力形式は、09:00 ではなく 9:00 と書き出します。

mm
分を表す 00 から 59 までの整数です。

時間と分は、SAS 時間値の秒値に基づき丸められます。

HHMM 出力形式では、アスタリスクを使用して、0-24 時間の時間範囲外の値(日時値など)をフォーマットします。

比較

HHMM*w.d* 出力形式は TIME*w.d* 出力形式とほぼ同じですが、HHMM*w.d* 出力形式は秒を出力しません。

HHMM*w.d* 出力形式は、1 桁の時間にはその先頭に空白を書き出します。TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。

例

例の表では、46796 を入力値として使用します。この値は、午後 12:59:56 に相当する SAS 時間値です。

SAS ステートメント	結果
	`----+----1`
`put time hhmm.;`	`13:00`
`put time hhmm8.2;`	`12:59.93`

最初の例では、SAS 時間値の秒値に基づき、時間値が 4 秒切り上げられます。2 番目の例では、出力形式に小数点以下の桁数として 2 を追加指定することにより、56 秒が 1 分の 93%として表示されています。

関連項目:

出力形式:

- “HOURw.d 出力形式” (101 ページ)
- “MMSSw.d 出力形式” (113 ページ)
- “TIMEw.d 出力形式” (152 ページ)
- “TODw.d 出力形式” (155 ページ)

関数:

- “HMS Function” in *SAS Functions and CALL Routines: Reference*
- “HOUR Function” in *SAS Functions and CALL Routines: Reference*
- “MINUTE Function” in *SAS Functions and CALL Routines: Reference*
- “SECOND Function” in *SAS Functions and CALL Routines: Reference*
- “TIME Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “TIMEw. 入力形式” (324 ページ)

HOUR*w.d* 出力形式

SAS 時間値を時間および時間の小数として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

HOUR*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 2
範囲: 2–20

d
時間値の小数点以下の桁数を指定します。時間の小数が出力されます。この引数はオプションです。
範囲: 0-19
要件 *w* より小さい数にします。

詳細

時間は、SAS 時間値の分値に基づき丸められます。

HOUR 出力形式では、アスタリスクを使用して、0-24 時間の時間範囲外の値(日時値など)をフォーマットします。

例

例の表では、41400 を入力値として使用します。この値は、午前 11:30 に相当する SAS 時間値です。

SAS ステートメント	結果
	`----+----1`
`put time hour4.1;`	`11.5`

関連項目:

出力形式:

- “HHMMw.d 出力形式” (100 ページ)
- “MMSSw.d 出力形式” (113 ページ)
- “TIMEw.d 出力形式” (152 ページ)
- “TODw.d 出力形式” (155 ページ)

関数:

- “HMS Function” in *SAS Functions and CALL Routines: Reference*
- “HOUR Function” in *SAS Functions and CALL Routines: Reference*
- “MINUTE Function” in *SAS Functions and CALL Routines: Reference*

- “SECOND Function” in *SAS Functions and CALL Routines: Reference*
- “TIME Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “TIMEw. 入力形式” (324 ページ)

IB*w.d* 出力形式

負の値を含む、ネイティブのバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

参照項目: “IBw.d Format: UNIX” in *SAS Companion for UNIX Environments*
“IBw.d Format: Windows” in *SAS Companion for Windows*
“IBw.d Format: z/OS” in *SAS Companion for z/OS*

構文

IB*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 1–8

d
10^d で数値を乗算するよう指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–10

詳細

IB*w.d* 出力形式では、2 の補数で表される負の値を含む、バイナリ整数(固定小数点)値を書き出します。IB*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、バイナリ整数値を書き出します。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

IB*w.d* と PIB*w.d* 出力形式は、ネイティブな形式の整数を書き出すために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 出力形式は、動作環境に関係なくリトルエンディアン整数を書き出すために使用されます。

ビッグエンディアン整数とリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,ib4.);
put y $hex8.;
```

x の値	ビッグエンディアンプラットフォームでの結果*	リトルエンディアンプラットフォームでの結果*
	----+----1	----+----1
128	00000080	80000000

* 結果は、4 バイトのバイナリ整数を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

関連項目:

出力形式:

- “IBRw.d 出力形式” (104 ページ)

IBR*w.d* 出力形式

Intel および DEC 形式のバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

構文

IBR*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 4

範囲: 1–8

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

詳細

IBR*w.d* 出力形式では、2 の補数で表される負の値を含む、バイナリ整数(固定小数点)値を書き出します。IBR*w.d* は、Intel および DEC の動作環境で生成されるバイナリ整数値を書き出します。IBR*w.d* を使用して、Intel または DEC の環境のバイナリ整

数データを他の動作環境に書き出します。SAS コードの IBR*w.d* 出力形式によって、動作環境に関係なくデータを書き出すためのポータブルな実装が用意されます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- IB*w.d* と PIB*w.d* 出力形式は、ネイティブな形式の整数を書き出すために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。)
- IBR*w.d* と PIBR*w.d* 出力形式は、書き出しを実行する動作環境に関係なくリトルエンディアン整数を書き出すために使用されます。
- Intel および DEC の環境では、IB*w.d* と IBR*w.d* 出力形式は同じものです。

ビッグエンディアン整数とリトルエンディアン整数に関連して使用する出力形式の種類を表示するには、表 1.1 (8 ページ)を参照してください。

各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,ibr4.);
put y $hex8.;
```

x の値	結果*
	----+----1
128	80000000

* 結果は、4 バイトのバイナリ整数を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

関連項目:

出力形式:

- “IBw.d 出力形式” (103 ページ)

IEEE*w.d* 出力形式

数値を 10 の *d* 乗し、IEEE 浮動小数点値を生成します。

カテゴリ: 数値

配置: 左

注意: 値が大きい浮動小数点数値や桁数が必要な浮動小数点数値は、IEEE 出力形式を使用して IBM メインフレームに書き出し、IEE 入力形式で読み込みなおす場合、元の SAS 値とは異なる場合があります。

構文

IEEE*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 3–8

ヒント: *w* が 8 の場合、IEEE の倍精度浮動小数点数が書き出されます。*w* が 5、6 または 7 の場合、相当するバイト数が切り捨てられたと仮定して、IEEE の倍精度浮動小数点数が書き出されます。*w* が 4 の場合、IEEE の単精度浮動小数点数が書き出されます。*w* が 3 の場合、1 バイトが切り捨てられたと仮定して、IEEE の単精度浮動小数点数が書き出されます。

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

詳細

この出力形式は、IEEE*w.d* が使用される浮動小数点数表現である動作環境で使用されます。また、IEEE*w.d* 出力形式を使用して、IEEE 浮動小数点数表現を使用する動作環境のプログラムによって使用されるファイルを作成することもできます。

通常プログラムは、単精度(4 バイト)または倍精度(8 バイト)で IEEE 値を生成します。プログラムは、出力ファイルのスペースを削減する場合にのみ切り捨てを実行します。機械語命令では、浮動小数点数が 4 バイトまたは 8 バイトのいずれかである必要があります。IEEE*w.d* 出力形式では、他の長さも処理できるため、スペースの削減のために切り捨てられたデータを含むファイルにデータを書き込めます。

例

```
test1=put(x,ieee4.);
put test1 $hex8.;
test2=put(x,ieee5.);
put test2 $hex10.;
```

x の値	結果*
1	3F800000
	3FF0000000

* 結果は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

JULDAY*w.* 出力形式

SAS 日付値からユリウス暦の日付部分を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

JULDAY*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 3
範囲: 3–32

詳細

JULDAY*w.*出力形式では、SAS 日付値を *ddd* 形式で書き出します。

ddd
1 から 365 まで(うるう年の場合は 1 から 366 まで)の日数です。

例

例の表では、18993 と 19068 を入力値として使用します。18993 は 2012 年 1 月 1 日に、19068 は 2012 年 3 月 16 日にそれぞれ相当する SAS 日付値です。

入力値	SAS ステートメント	結果
		`----+----1`
`18993`	`put date julday3.;`	`1`
`19068`	`put date julday3.;`	`76`

JULIAN*w.* 出力形式

SAS 日付値を *yyddd* または *yyyyddd* 形式のユリウス日付として書き出します。

カテゴリ: 日付と時間
配置: 左

構文

JULIAN*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 5
範囲: 5–7

ヒント: *w* が 5 の場合、JULIAN*w*.出力形式では日付の年の部分を 2 桁で書き出します。*w* が 7 の場合、JULIAN*w*.出力形式では日付の年の部分を 4 桁で書き出します。

詳細

JULIAN*w*.出力形式では、SAS 日付値を *yyddd* または *yyyyddd* 形式で書き出します。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

ddd
1 から 365 まで(うるう年の場合は 1 から 366 まで)の年の日数です。

例

例の表では、19114 を入力値として使用します。この値は 2012 年 5 月 1 日(年の 122 日目)に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date julian5.;`	`12122`
`put date julian7.;`	`2012122`

関連項目:

関数:

- "DATEJUL Function" in *SAS Functions and CALL Routines: Reference*
- "JULDATE Function" in *SAS Functions and CALL Routines: Reference*

入力形式:

- "JULIANw. 入力形式" (282 ページ)

MDYAMPM*w.d* 出力形式

SAS 日時値を *mm/dd/yy<yy> hh:mm* AM|PM 形式で書き出します。年は 2 桁または 4 桁で表示されます。

カテゴリ: 日付と時間

配置: 右

注: デフォルトの時間は AM です。

構文

MDYAMPM*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 19
範囲: 8−40

詳細

MDYAMPM*w.d* 出力形式は、SAS 日時値を次の形式で書き出します。

mm/*dd*/*yy*<yy> *hh*:*mm*<AM | PM>:

mm
月を表す 1 から 12 までの整数です。

dd
月の日を表す 1 から 31 の整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

hh
時間を表す 00 から 23 までの整数です。

mm
分を表す 00 から 59 までの整数です。

AM | PM
00:01−12:00(AM)または 12:01−12:00(PM)のいずれかです。デフォルトは AM です。

日時の区切り文字
SAS で日付要素と時間要素を区切るために使用するスラッシュ(/)、コロン(:)、空白文字などの特殊文字のいずれか 1 つです。

比較

MDYAMPM*w.*出力形式は、日時値を区切り文字付きの *mm/dd/yy*<*yy*> *hh:mm* AM | PM 形式で書き出します。日付と時間の間にスペースが挿入されます。

DATETIME*w.d* 出力形式は、日時値を区切り文字付きの *ddmmmyy*<*yy*>*: hh:mm:ss.ss* 形式で書き出します。

例

例の表では、1663343580 を入力値として使用します。この値は 2012 年 9 月 15 日の午後 3:53:00 に相当する SAS 日時値です。

SAS ステートメント	結果
`put dt mdyampm25.`	`9/15/2012 3:53 PM`

関連項目:

出力形式:

- “DATETIMEw.d 出力形式” (72 ページ)

入力形式:

- “MDYAMPMw.d 入力形式” (283 ページ)

MMDDYY*w*. 出力形式

SAS 日付値を *mmdd<yy>yy* または *mm/dd/<yy>yy* 形式で書き出します。スラッシュが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

MMDDYY*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作: *w* の値が 2 から 5 までの場合、可能な限りで日と月だけを出力します。*w* が 7 の場合、年を 2 桁で表し、スラッシュを使わずに出力します。

詳細

MMDDYY*w*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

mmdd<yy>yy

mm/dd/<yy>yy:

mm
月を表す整数です。

/
区切り文字です。

dd
月の日を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19291 を入力値として使用します。この値は、2012 年 10 月 25 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put day mmddyy2.;`	`10`

SAS ステートメント	結果
put day mmddyy3.;	10
put day mmddyy4.;	1025
put day mmddyy5.;	10/25
put day mmddyy6.;	102512
put day mmddyy7.;	102512
put day mmddyy8.;	10/25/12
put day mmddyy10.;	10/25/2012

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYxw. 出力形式” (111 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

MMDDYY*xw*. 出力形式

SAS 日付値を *mmdd<yy>yy* または *mm-dd-<yy>yy* 形式で書き出します。出力形式名の *x* は、月、日、年を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

MMDDYY*xw*.

説明

x
区切り文字を示します。または、区切り文字を日、月、年の間に挿入しないことを示します。*x* の値として、次の値が有効です。

B
空白で区切ります。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作:
w の値が 2 から 5 までの場合、日と月が可能な範囲で表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。
x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

MMDDYY*xw*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

mmdd<yy>yy

mmxddx<yy>yy

mm
月を表す整数です。

x
指定された区切り文字です。

dd
月の日を表す整数です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19127 を入力値として使用します。この値は、2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1----+
put day mmddyyc5.;	05:14
put day mmddyyd8.;	05-14-12
put day mmddyyp10.;	05.14.2012
put day mmddyyn8.;	05142012

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYxw. 出力形式” (76 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDxw. 出力形式” (177 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “MMDDYYw. 入力形式” (285 ページ)

MMSS*w.d* 出力形式

SAS 時間値を午前 0 時から経過した分数および秒数として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MMSS*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 5

範囲: 2–20

ヒント: 分数と秒数を表す値を書き出すには、*w* に少なくとも 5 を指定します。

d

秒値の小数点以下の桁数を指定します。このため、SAS 時間値には秒の端数が含まれます。この引数はオプションです。

範囲: 0–19

制限事項: *w* より小さい値にします。

詳細

MMSS 出力形式では、アスタリスクを使用して、0-24 時間の時間範囲外の値(日時値など)をフォーマットします。

例

次の例では、入力値として 4530 を使用します。

SAS ステートメント	結果
	`----+----1`
`put time mmss.;`	`75:30`

関連項目:

出力形式:

- “HHMMw.d 出力形式” (100 ページ)
- “TIMEw.d 出力形式” (152 ページ)

関数:

- “HMS Function” in *SAS Functions and CALL Routines: Reference*
- “MINUTE Function” in *SAS Functions and CALL Routines: Reference*
- “SECOND Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “TIMEw. 入力形式” (324 ページ)

MMYY*w.* 出力形式

SAS 日付値を *mm*M<*yy*>*yy* 形式で書き出します。M は区切り文字を表し、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

MMYY*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 7

範囲: 5–32

操作: *w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

詳細

MMYY*w*.出力形式は、SAS 日付値を *mm*M<*yy*>*yy* 形式で書き出します。

mm
月を表す整数です。

M
区切り文字です。

<*yy*>*yy*
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19291 を入力値として使用します。この値は、2012 年 10 月 25 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date mmyy5.;`	`10M12`
`put date mmyy6.;`	`10M12`
`put date mmyy.;`	`10M2012`
`put date mmyy7.;`	`10M2012`
`put date mmyy10.;`	`10M2012`

関連項目:

出力形式:

- “MMYYxw. 出力形式” (116 ページ)
- “YYMMw. 出力形式” (174 ページ)

MMYY*xw.* 出力形式

SAS 日付値を *mm*<*yy*>*yy* または *mm*-<*yy*>*yy* 形式で書き出します。出力形式名の *x* は、月と年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

MMYY*xw.*

説明

x
区切り文字を示します。または、区切り文字が月と年の間に挿入しないことを示します。*x* の値として、次の値が有効です。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト: 7

範囲: 5–32

操作:
x を N に設定すると、区切り文字なしを指定します。この場合の幅範囲は 4–32 になり、デフォルトは 6 に変化します。

x の値が C、D、P または S で、*w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 4 または 5 の場合、日付の年は下 2 桁のみ表示されます。*x* の値が *N* で、*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

MMYY*xw.*出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

mm<*yy*>*yy*

mmx<*yy*>*yy*

mm
月を表す整数です。

x
指定された区切り文字です。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

例

次の例では、19127 を入力値として使用します。この値は、2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1----+
put date mmyyc5.;	05:12
put date mmyyd.;	05-2012
put date mmyyn4.;	0512
put date mmyyp8.;	05.2012
put date mmyys10.;	05/2012

関連項目:

出力形式:

- “MMYYw. 出力形式” (114 ページ)
- “YYMMxw. 出力形式” (179 ページ)

MONNAME*w*. 出力形式

SAS 日付値を月の名前として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONNAME*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 9
範囲: 1–32
ヒント: MONNAME3.を使用すると、月名の最初の 3 文字が出力されます。

詳細

必要に応じて、出力形式の幅に合うように月名が切り捨てられます。

例

例の表では、19057 を入力値として使用します。この値は 2012 年 3 月 5 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date monname1.;`	`M`
`put date monname3.;`	`Mar`
`put date monname5.;`	`March`

関連項目:

出力形式:

- “MONTHw. 出力形式” (118 ページ)

MONTH*w*. 出力形式

SAS 日付値を月として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONTH*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 2

範囲: 1–32

ヒント: MONTH1.を使用すると、16 進値が出力されます。

詳細

MONTH*w*.出力形式は、SAS 日付値から月(1 から 12 まで)を書き出します。月が 1 桁の場合、MONTH*w*.出力形式はその桁の前に先頭の空白を置きます。たとえば、MONTH*w*.出力形式は 04 ではなく 4 と書き出します。

例

例の表では、19127 を入力値として使用します。この値は 2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1
put date month.;	5

関連項目:

出力形式:

- “MONNAMEw. 出力形式” (117 ページ)

MONYY*w.* 出力形式

SAS 日付値を月および年として *mmmyy* または *mmmyyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

MONYY*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 5

範囲: 5–7

詳細

MONYY*w.*出力形式は、SAS 日付値を *mmmyy* または *mmmyyyy* 形式で書き出します。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

比較

MONYY*w.*出力形式と DTMONYY*w.*出力形式は、日付値を書き出すという点でほぼ同じです。相違点は、MONYY*w.*が入力として SAS 日付値が必要であり、DTMONYY*w.*が日時値が必要である点です。

例

例の表では、19127 を入力値として使用します。この値は 2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date monyy5.;`	`MAY12`
`put date monyy7.;`	`MAY2012`

関連項目:

出力形式:

- “DTMONYYw. 出力形式” (82 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “MONYYw. 入力形式” (286 ページ)

NEGPARENw.d 出力形式

負の数値をかっこで囲んで書き出します。

カテゴリ: 数値

配置: 右

構文

NEGPAREN*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 1–32

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

NEGPARENw.d 出力形式は、出力値を右揃えにします。入力値が負の場合、NEGPARENw.d は、指定したフィールドの幅が十分な場合は値をかっこで囲って出力を表示します。フィールドの幅が十分でない場合は、負の値を表すためにマイナス符号が使用されます。入力値が負でない場合、NEGPARENw.d は、適切なカラム配置を確実にするために値を先頭および末尾の空白とともに表示します。値が正の場合でも、最後のカラムは右かっこ用に確保されます。

比較

NEGPARENw.d 出力形式と COMMAw.d 出力形式は、各値を 3 桁ごとにカンマで区切るという点では同じです。

例

```
put @1 sales negparen8.;
```

sales の値	結果
	----+----1----+
100	100
1000	1,000
-200	(200)
-2000	(2,000)

NUMXw.d 出力形式

小数点をカンマにして数値を書き出します。

カテゴリ: 数値

配置: 右

構文

NUMX*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 12

範囲: 1–32

d

数値の小数点(カンマ)以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

NUMX*w.d* 出力形式は、小数点をカンマにして数値を書き出します。

比較

NUMX*w.d* 出力形式は、*w.d* 出力形式とほぼ同じですが、NUMX*w.d* では小数点をカンマにして数値を書き出します。

例: 例

```
put x numx10.2;
```

x の値	結果
----+----1----+	
896.48	896,48
64.89	64,89
3064.10	3064,10

関連項目:

出力形式:

- “w.d 出力形式” (159 ページ)

入力形式:

- “NUMXw.d 入力形式” (289 ページ)

OCTAL*w.* 出力形式

数値を 8 進表現に変換します。

カテゴリ: 数値

配置: 左

構文

OCTAL*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 3
範囲: 1–24

詳細

必要に応じて、OCTAL*w.*出力形式は数値を整数に変換してから 8 進表現で表示します。

比較

OCTAL*w.*は、数値を 8 進表現に変換します。$OCTAL*w.*出力形式は、文字値を 8 進表現に変換します。

例

```
put x octal6.;
```

x の値	結果
	----+----1
3592	007010

PD*w.d* 出力形式

パック 10 進形式のデータを書き出します。

カテゴリ: 数値
配置: 左
参照項目: “PDw.d Format: UNIX” in *SAS Companion for UNIX Environments*
“PDw.d Format: Windows” in *SAS Companion for Windows*
“PDw.d Format: z/OS” in *SAS Companion for z/OS*

構文

PD*w.d*

説明

w
出力フィールドの幅を指定します。*w* 値は、桁数ではなくバイト数を指定します。(パック 10 進データは、1 バイトに 2 桁を含みます。)
デフォルト: 1
範囲: 1–16

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

動作環境によってパック 10 進値の保存方法は異なります。ただし、PD*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、パック 10 進値を書き出します。

PD*w.d* 出力形式は、欠損数値データを–0 として書き出します。PD*w.d* 入力形式で–0 が読み出されると、0 として保存されます。

比較

次の表は、各種プログラミング言語でのパック 10 進表記を比較したものです。

言語	表記
SAS	PD4.
COBOL	COMP-3 PIC S9(7)
IBM 370 アセンブラ	PL4
PL/I	FIXED DEC

例

```
y=put(x,pd4.);
put y $hex8.;
```

x の値	結果*
	----+----1
128	00000128

* 結果は、パック 10 進形式で書かれたバイナリ値を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

PDJULG*w.* 出力形式

パックユリウス日付値を、IBM で使用する 16 進の *yyyyddd*F 形式で書き出します。

カテゴリ: 日付と時間

構文

PDJULG*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 3-16

詳細

PDJULG*w*.出力形式は、SAS 日付値を *yyyyddd*F 形式で書き出します。

yyyy
4 桁のグレゴリオ暦の年を 2 バイトで表したものです。

ddd
ユリウス日 1–365 (うるう年の場合は 1–366)に対応する 3 桁の整数を 1.5 バイトで表したものです。

F
2 進表現の 1 のみで構成される、値を正とする 0.5 バイトです。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

SAS ステートメント	結果
	`----+----1`

`date = '17mar2012'd;`
`juldate = put(date,pdjulg4.);`
`put juldate $hex8.;`
	`2012077F`

関連項目:

出力形式:

- “PDJULIw. 出力形式” (126 ページ)
- “JULIANw. 出力形式” (107 ページ)
- “JULDAYw. 出力形式” (106 ページ)

関数:

- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*
- “DATEJUL Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “PDJULIw. 入力形式” (293 ページ)
- “PDJULGw. 入力形式” (292 ページ)
- “JULIANw. 入力形式” (282 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

PDJULI*w.* 出力形式

パック表現のユリウス日付値を、IBM で使用する 16 進の *ccyyddd*F 形式で書き出します。

カテゴリ: 日付と時間

構文

PDJULI*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 3-16

詳細

PDJULI*w.*出力形式は、SAS 日付値を *ccyyddd*F 形式で書き出します。

cc
世紀を表す 2 桁の整数を 1 バイトで表したものです。

yy
年を表す 2 桁の整数を 1 バイトで表したものです。PDJULI*w.*出力形式は、正しいパック 10 進 *ccyy* 表現を生成するために 4 桁のグレゴリオ暦の年から 1900 を減算して、世紀バイトの調整を行います。年値 1998 は *ccyy* で 0098 として保存され、年値 2011 は 0111 として保存されます。

ddd
ユリウス日 1–365 (うるう年の場合は 1–366)に対応する 3 桁の整数を 1.5 バイトで表したものです。

F
2 進表現の 1 のみで構成される、値を正とする 0.5 バイトです。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

SAS ステートメント	結果
	`----+----1`

`date = '17mar2012'd;`
`juldate = put(date,pdjuli4.);`
`put juldate $hex8.;`
	`0112077F`

SAS ステートメント	結果

`date = '31dec2013'd;`
`juldate = put(date,pdjuli4.);`
`put juldate $hex8.;`
	`0113365F`

関連項目:

出力形式:

- “PDJULGw. 出力形式” (124 ページ)
- “JULIANw. 出力形式” (107 ページ)
- “JULDAYw. 出力形式” (106 ページ)

関数:

- “DATEJUL Function” in *SAS Functions and CALL Routines: Reference*
- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “PDJULGw. 入力形式” (292 ページ)
- “PDJULIw. 入力形式” (293 ページ)
- “JULIANw. 入力形式” (282 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

PERCENT*w.d* 出力形式

数値を百分率として書き出します。

カテゴリ: 数値

配置: 右

構文

PERCENT*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 4–32

ヒント: 出力フィールドの幅数には、数字が正か負かにかかわらず、パーセント記号(%)と負の数字のためのかっこを含める必要があります。

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–31

要件 *w* より小さい数にします。

詳細

PERCENT*w.d* 出力形式は、値を 100 で乗算した後に BEST*w.d* 出力形式を適用し、適用後の値の末尾にパーセント記号(%)を追加します。負の値はかっこで囲みます。

例

```
put @10 gain percent10.;
```

x の値	結果
	----+----1----+----2
0.1	10%
1.2	120%
-0.05	(5%)

関連項目:

出力形式:

- “PERCENTNw.d 出力形式” (128 ページ)

PERCENTN*w.d* 出力形式

百分率を書き出します。負の値にはマイナス符号を使用します。

カテゴリ: 数値

配置: 右

構文

PERCENTN*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 4–32

ヒント: 出力フィールドの幅数には、数値が負か正かにかかわらず、マイナス記号(-)、パーセント記号(%)、末尾の空白分を含める必要があります。

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–31

要件 *w* より小さい数にします。

詳細

PERCENTN*w.d* 出力形式は、負の値に 100 を乗算し、BEST*w.d* 出力形式を適用し、適用後の値の先頭と末尾にマイナス記号とパーセント記号(%)をそれぞれ追加します。

比較

PERCENTN*w.d* 出力形式は、負の値にかっこではなくマイナス記号を使用して、百分率を生成します。PERCENT*w.d* 出力形式は、負の値にかっこを使用して、百分率を生成します。

例

```
put x percentn10.;
```

x の値	結果
-0.1	-10%
.2	20%
.8	80%
-0.05	-5%
-6.3	-630%

関連項目:

出力形式:

- “PERCENTw.d 出力形式” (127 ページ)

PIB*w.d* 出力形式

正のバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

参照項目: “PIBw.d Format: UNIX” in *SAS Companion for UNIX Environments*

“PIBw.d Format: Windows” in *SAS Companion for Windows*

構文

PIB*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–8

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

すべての値は、正として処理されます。PIB*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、正のバイナリ整数値を書き出します。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- 正のバイナリ整数値は、符号ビットが値の一部である点を除きバイナリ整数値と同じであり、常に正の整数になります。PIB*w.d* 出力形式はすべての値を正として処理し、値の一部として符号ビットを含めます。
- PIB*w.d* 出力形式は、幅が 1 の場合、1 バイトのコンテンツを 2 進表現した値を出力します。1 バイトのコンテンツを 2 進表現する値は、データに 16 進数の 80 と 16 進数の FF の間の値が含まれる、高位ビットが負の符号として誤って解釈される可能性がある場合に使用します。
- PIB*w.d* 出力形式は IB*w.d* 形式と同じですが、PIB*w.d* はすべての値を正の値として処理します。
- IB*w.d* と PIB*w.d* 出力形式は、ネイティブな形式の整数を書き出すために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 出力形式は、動作環境に関係なくリトルエンディアン整数を書き出すために使用されます。

  ビッグエンディアン整数とリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,pib1.);
put y $hex2.;
```

x の値	結果*
	----+----1

x の値	結果*
12	0C

* 結果は、正のバイナリ整数形式で書かれた 1 バイトのバイナリ値を 16 表現で表したものです。出力フィールドの 1 カラムを使用します。

関連項目:

出力形式:

- “PIBRw.d 出力形式” (131 ページ)

PIBR*w.d* 出力形式

Intel 形式と DEC 形式の正のバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

構文

PIBR*w.d*

説明

w
幅を指定します。
デフォルト: 1
範囲: 1–8

d
10^d で数値を乗算するよう指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–10

詳細

すべての値は、正として処理されます。PIBR*w.d* は、Intel および DEC の動作環境で生成される正のバイナリ整数値を書き出します。PIBR*w.d* を使用して、Intel または DEC の環境の正のバイナリ整数データを他の動作環境に書き出します。PIBR*w.d* 出力形式によって、動作環境に関係なくデータを書き出すためのポータブルな実装が用意されます。

注: 動作環境によって正のバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- 正のバイナリ整数値は、符号ビットが値の一部である点を除きバイナリ整数値と同じであり、常に正の整数になります。PIBR*w.d* 出力形式はすべての値を正として処理し、値の一部として符号ビットを含めます。

- PIBR*w.d* 出力形式は、幅が 1 の場合、1 バイトのコンテンツを 2 進表現した値を出力します。1 バイトのコンテンツを 2 進表現する値は、データに 16 進数の 80 と 16 進数の FF の間の値が含まれる、高位ビットが負の符号として誤って解釈される可能性がある場合に使用します。
- Intel および DEC の動作環境では、PIB*w.d* と PIBR*w.d* 出力形式は同じものです。
- IB*w.d* と PIB*w.d* 出力形式は、ネイティブな形式の整数を書き出すために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 出力形式は、動作環境に関係なくリトルエンディアン整数を書き出すために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,pibr2.);
put y $hex4.;
```

x の値	結果*
	----+----1
128	8000

* 結果は、正のバイナリ整数形式で書かれた 2 バイトのバイナリ値を 16 表現で表したものです。出力フィールドの 1 カラムを使用します。

関連項目:

入力形式:

- “PIBw.d 入力形式” (296 ページ)

PK*w.d* 出力形式

符号なしパック 10 進形式のデータを書き出します。

カテゴリ: 数値

配置: 左

構文

PK*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–16

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

要件 *w* より小さい数にします。

詳細

符号なしのパック 10 進データは、1 バイトに 2 桁を含みます。

比較

PK*w.d* 出力形式は PD*w.d* 出力形式と同じですが、PK*w.d* は下位バイトに符号を書き出しません。

例

```
y=put(x,pk4.);
put y $hex8.;
```

x の値	結果*
	----+----1
128	00000128

* 結果は、パック 10 進形式で書かれた 4 バイトの数値を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

PVALUE*w.d* 出力形式

p 値を書き出します。

カテゴリ: 数値

配置: 右

構文

PVALUE*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 3–32

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 最小 4 および *w*–2

範囲: 1–30

制限事項: *w* より小さい数にします。

比較

PVALUE*w.d* 出力形式は、次の場合を除いて *w.d* 出力形式の規則に従います。

- 値 x が 0 <= x < 10^{-d} の場合、x は *d*-1 のゼロを使用して“<.0...01”と出力される。
- 欠損値は、“.”として出力される。(MISSING=システムオプションを使用して別の文字を指定する場合を除く)

例

```
put x pvalue6.4;
```

x の値	結果
	----+----1
.05	0.0500
0.000001	<.0001
0	<.0001
.0123456	0.0123

QTR*w.* 出力形式

SAS 日付値を年の四半期として書き出します。

カテゴリ: 日付と時間

配置: 右

構文

QTR*w.*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–32

例

例の表では、19057 を入力値として使用します。この値は 2012 年 3 月 5 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1
put date qtr.;	1

関連項目:

出力形式:

- “QTRRw. 出力形式” (135 ページ)

QTRR*w.* 出力形式

SAS 日付値を年の四半期としてローマ数字で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

QTRR*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 3
範囲: 3–32

例

例の表では、19251 を入力値として使用します。この値は 2012 年 9 月 15 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1
put date qtrr.;	III

関連項目:

出力形式:

- “QTRw. 出力形式” (134 ページ)

RB*w.d* 出力形式

バイナリ実数データ(浮動小数点)をバイナリ実数形式で書き出します。

カテゴリ: 数値

配置: 左

参照項目: “RBw.d Format: UNIX” in *SAS Companion for UNIX Environments*
“RBw.d Format: Windows” in *SAS Companion for Windows*
“RBw.d Format: z/OS” in *SAS Companion for z/OS*

構文

RB*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 2–8

d
10^d で数値を乗算するよう指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–10

詳細

RB*w.d* 出力形式は、SAS での数値データの保存形式と同じ形式で書き出します。データ変換が不要なため、RB*w.d* はデータを SAS で書き出すための最も効率的な方法です。

注: 動作環境によってバイナリ実数値の保存方法は異なります。ただし、RB*w.d* は、SAS の実行に使用する動作環境に合ったバイナリ実数値を書き出します。

注意:
IEEE の標準基準に準拠した環境へのバイナリ実数データの書き出しに RB4.を使用すると、浮動小数点数は実際の 4 バイト(単精度)の浮動小数点数ではなく、8 バイト(倍精度)の数字に切り捨てられます。

比較

次の表は、各種プログラミング言語でのバイナリ実数表記名を比較したものです。

言語	4 バイト	8 バイト
SAS	RB4.	RB8.
Fortran	REAL*4	REAL*8
C	float	double

言語	4 バイト	8 バイト
COBOL	COMP-1	COMP-2
IBM 370 アセンブラ	E	D

例

```
y=put(x,rb8.);
put y $hex16.;
```

x の値	結果*
	----+---1----+----2
128	4280000000000000

* 結果は、IBM メインフレームで見られる、8 バイトのバイナリ実数を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

ROMAN*w*. 出力形式

数値をローマ数字で書き出します。

カテゴリ: 数値

配置: 左

構文

ROMAN*w*.

説明

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 2–32

詳細

ROMAN*w*.出力形式は、浮動小数点数を整数に切り捨ててから値を書き出します。

例

```
put @5 year roman10.;
```

year の値	結果
2012	MMXII

S370FF*w.d* 出力形式

IBM メインフレーム形式の標準数値データを書き出します。

カテゴリ: 数値

構文

S370FF*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 12
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–31

詳細

S370FF*w.d* 出力形式は、IBM メインフレーム形式(EBCDIC)の数値データを書き出します。EBCDIC 数値は、1 桁 1 バイトで表されます。EBCDIC がネイティブな形式の場合、S370FF*w.d* は変換を実行しません。

値が負の場合、EBCDIC マイナス記号が値の前に置かれます。欠損値は、1 つの EBCDIC ピリオドで表されます。

比較

- EBCDIC システムでは、S370FF*w.d* は *w.d* 出力形式のように機能します。
- それ以外のシステムでは、S370FF*w.d* は、文字データに対する$EBCDIC*w.*出力形式の処理と同じ処理を数値データに行います。

例

```
y=put(x,s370ff5.);
put y $hex10.;
```

x の値	結果*
	----+----1
12345	F1F2F3F4F5

* 結果は、整数を 16 進表現で表したものです。

関連項目:

出力形式:

- “$EBCDICw. 出力形式” (35 ページ)
- “w.d 出力形式” (159 ページ)

S370FIB*w.d* 出力形式

IBM メインフレーム形式の負の値を含むバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

構文

S370FIB*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 4

範囲: 1–8

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

詳細

S370FIB*w.d* 出力形式は、2 の補数で表される負の値を含む、IBM フレームワーク形式で保存されるバイナリ整数(固定小数点)値を書き出します。S370FIB*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、バイナリ整数値を書き出します。

S370FIB*w.d* を使用して、IBM メインフレーム形式のバイナリ整数データを他の動作環境で作成されたデータから書き出します。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- SAS を IBM メインフレームで使用する場合、S370FIB*w.d* と IB*w.d* は同じものです。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を書き出すために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,s370fib4.);
put y $hex8.;
```

x の値	結果*
	----+----1
128	00000080

* 結果は、4 バイトのバイナリ整数を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

関連項目:

出力形式:

- “S370FIBUw.d 出力形式” (140 ページ)
- “S370FPIBw.d 出力形式” (143 ページ)

S370FIBU*w.d* 出力形式

IBM メインフレーム形式の符号なしバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

構文

S370FIBU*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 4

範囲: 1–8

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

詳細

S370FIBU*w.d* 出力形式は、2 の補数で表される負の値を含む、IBM フレームワーク形式で保存される符号なしバイナリ整数(固定小数点)値を書き出します。符号なしバイナリ整数値はバイナリ整数値と同じですが、すべての値が正として処理される点が異なります。S370FIBU*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、バイナリ整数値を書き出します。

S370FIBU*w.d* を使用して、IBM メインフレーム形式の符号なしバイナリ整数データを他の動作環境で作成されたデータから書き出します。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- S370FIBU*w.d* 出力形式は、COBOL の表記 PIC 9 (*n*) BINARY と同じです。*n* は桁数です。
- S370FIBU*w.d* 出力形式は S370FIB*w.d* 出力形式と同じですが、S370FIBU*w.d* 出力形式は符号の付いた値ではなく絶対値を常に使用します。
- S370FPIB*w.d* 出力形式はすべての負の数を FF として書き出し、S370FIBU*w.d* 出力形式は絶対値を書き出します。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を書き出すために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,s370fibu1.);
put y $hex2.;
```

x の値	結果*
245	F5
-245	F5

* 結果は、1 バイトのバイナリ整数を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

関連項目:

出力形式

- “S370FIBw.d 出力形式” (139 ページ)
- “S370FPIBw.d 出力形式” (143 ページ)

S370FPD*w.d* 出力形式

IBM メインフレーム形式のパック 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FPD*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–16

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FPD*w.d* は、IBM メインフレームコンピュータと同じ形式でパック 10 進データを書き出すために他の動作環境にて使用します。

比較

次の表は、各種プログラミング言語でのパック 10 進形式の表記を示したものです。

言語	パック 10 進表記
SAS	S370FPD4.
PL/I	FIXED DEC(7,0)
COBOL	COMP-3 PIC S9(7)
IBM 370 アセンブラ	PL4

例

```
y=put(x,s370fpd4.);
put y $hex8.;
```

x の値	結果*
	`----+----1`
128	`0000128C`

* 結果は、パック 10 進形式で書かれたバイナリ値を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

S370FPDU*w.d* 出力形式

IBM メインフレーム形式の符号なしパック 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FPDU*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–16

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FPDU*w.d* は、IBM メインフレームコンピュータと同じ形式で符号なしパック 10 進データを書き出すために他の動作環境にて使用します。

比較

- S370FPDU*w.d* 出力形式は S370FPD*w.d* 出力形式とほぼ同じですが、S370FPD*w.d* 出力形式は符号の付いた値ではなく絶対値を常に使用します。
- S370FPDU*w.d* 出力形式は、COBOL の表記 PIC 9 (*n*) PACKED-DECIMAL と同じです。*n* 値は桁数です。

例

```
y=put(x,s370fpdu2.);
put y $hex4.;
```

x の値	結果*
123	123F
-123	123F

* 結果は、パック 10 進形式で書かれたバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FPIB*w.d* 出力形式

IBM メインフレーム形式の正のバイナリ整数(固定小数点)値を書き出します。

カテゴリ: 数値

配置: 左

構文

S370FPIB*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 4

範囲: 1–8

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–10

詳細

正のバイナリ整数値はバイナリ整数値と同じですが、すべての値が正として処理される点が異なります。S370FPIB*w.d* では、SAS の実行に使用する動作環境で値が作成された場合と同じように、バイナリ整数値を書き出します。

S370FPIB*w.d* を使用して、IBM メインフレーム形式の正のバイナリ整数データを他の動作環境で作成されたデータから書き出します。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (7 ページ)を参照してください。

比較

- SAS を IBM メインフレームで使用する場合、S370FPIB*w.d* と PIB*w.d* は同じものです。
- S370FPIB*w.d* 出力形式は S370FIB*w.d* 出力形式と同じですが、S370FPIB*w.d* 出力形式はすべての値を正の値として処理します。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を書き出すために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する出力形式の種類の表については、表 1.1 (8 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

```
y=put(x,s370fpib1.);
put y $hex2.;
```

x の値	結果*
	----+----1

x の値	結果*
12	0C

* *結果は、正のバイナリ整数形式で書かれた 1 バイトのバイナリ値を 16 表現で表したものです。出力フィールドの 1 カラムを使用します。

関連項目:

出力形式:

- “S370FIBw.d 出力形式” (139 ページ)
- “S370FIBUw.d 出力形式” (140 ページ)

S370FRB*w.d* 出力形式

IBM メインフレーム形式のバイナリ実数(浮動小数点)を書き出します。

カテゴリ: 数値

配置: 左

構文

S370FRB*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 4
範囲: 2–8

d
10^d で数値を乗算するよう指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–10

詳細

浮動小数点数は、値を表す仮数部と値の大きさを表す指数部で構成されています。

S370FRB*w.d* は、IBM メインフレームコンピュータと同じ形式で浮動小数点数バイナリデータを書き出すために他の動作環境にて使用します。

比較

次の表は、各種プログラミング言語での浮動小数点数進形式の表記を示したものです。

言語	4 バイト	8 バイト
SAS	S370FRB4.	S370FRB8.

言語	4 バイト	8 バイト
PL/I	FLOAT BIN(21)	FLOAT BIN(53)
Fortran	REAL*4	REAL*8
COBOL	COMP-1	COMP-2
IBM 370 アセンブラ	E	D
C	float	double

例

```
y=put(x,s370frb6.);
put y $hex8.;
```

x の値	結果*
128	42800000
-123	C2800000

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FZD*w.d* 出力形式

IBM メインフレーム形式のゾーン 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FZD*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 1–32

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FZD*w.d* は、IBM メインフレームコンピュータと同じ形式でゾーン 10 進データを書き出すために他の動作環境にて使用します。

比較

次の表は、各種プログラミング言語でのゾーン 10 進形式の表記を示したものです。

言語	ゾーン 10 進表記
SAS	S370FZD3.
PL/I	PICTURE '99T'
COBOL	PIC S9(3) DISPLAY
アセンブラ	ZL3

例

```
y=put(x,s370fzd3.);
put y $hex6.;
```

x の値	結果*
123	F1F2C3
-123	F1F2D3

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FZDL*w.d* 出力形式

IBM メインフレーム形式の前符号付きゾーン 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FZDL*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 1–32

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FZDL*w.d* は、IBM メインフレームコンピュータと同じ形式で前符号付きゾーン 10 進データを書き出すために他の動作環境にて使用します。

比較

- S370FZDL*w.d* 出力形式は S370FZD*w.d* 出力形式と同じですが、S370FZDL*w.d* 出力形式は出力形式が適用された出力の最初のバイトに数値の符号を表示します。
- S370FZDL*w.d* 出力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN LEADING と同じです。*n* 値は桁数です。

例

```
y=put(x,s370fzdl3.);
put y $hex6.;
```

x の値	結果*
123	C1F2F3
-123	D1F2F3

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FZDS*w.d* 出力形式

IBM メインフレーム形式の分離した前符号付きゾーン 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FZDS*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–32

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FZDS*w.d* は、IBM メインフレームコンピュータと同じ形式で分離した前符号付きゾーン 10 進データを書き出すために他の動作環境にて使用します。

比較

- S370FZDS*w.d* 出力形式は、S370FZDL*w.d* 出力形式と同じですが、S370FZDS*w.d* 出力形式は数値の符号をゾーン出力に埋め込みません。
- S370FZDS*w.d* 出力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN LEADING SEPARATE と同じです。*n* 値は桁数です。

例

```
y=put (x,s370fzds4.);
put y $hex8.;
```

x の値	結果*
123	4EF1F2F3
-123	60F1F2F3

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FZDT*w.d* 出力形式

IBM メインフレーム形式の分離した後符号付きゾーン 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FZDT*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–32

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FZDT*w.d* は、IBM メインフレームコンピュータと同じ形式で分離した後符号付きゾーン 10 進データを書き出すために他の動作環境にて使用します。

比較

- S370FZDT*w.d* 出力形式は S370FZDS*w.d* 出力形式と同じですが、S370FZDT*w.d* 出力形式は、出力形式が適用された出力の最後に数値の符号を表示します。
- S370FZDT*w.d* 出力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN TRAILING SEPARATE と同じです。*n* 値は桁数です。

例

```
y=put (x,s370fzdt4.); ;
put y $hex8.;
```

x の値	結果*
123	F1F2F34E
-123	F1F2F360

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

S370FZDU*w.d* 出力形式

IBM メインフレーム形式の符号なしゾーン 10 進データを書き出します。

カテゴリ: 数値

配置: 左

構文

S370FZDU*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 1–32

d

10^d で数値を乗算するよう指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

S370FZDU*w.d* は、IBM メインフレームコンピュータと同じ形式で符号なしゾーン 10 進データを書き出すために他の動作環境にて使用します。

比較

- S370FZDU*w.d* 出力形式は S370FZD*w.d* 出力形式と同じですが、S370FZDU*w.d* 出力形式は数値の絶対値を常に使用します。
- S370FZDU*w.d* 出力形式は、COBOL の表記 PIC 9 (*n*) DISPLAY と同じです。*n* 値は桁数です。

例

```
y=put (x,s370fzdu3.);
put y $hex6.;
```

x の値	結果*
123	F1F2F3
-123	F1F2F3

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。16 進数のペア(F1 など)がバイナリデータの 1 バイトに対応します。1 バイトは、出力フィールドの 1 カラムに対応します。

SSN*w.* 出力形式

アメリカの社会保障番号形式で書き出します。

カテゴリ: 数値

構文

SSN*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 11

制限事項: *w* は 11 である必要があります。

詳細

欠損値は 9 つのシングルピリオドとして書き出され、3 つ目のピリオドと 4 つ目のピリオドの間、5 つ目のピリオドと 6 つ目のピリオドの間にそれぞれハイフンが置かれます。値が 9 桁に満たない場合は値を右揃えにし、左側にゼロを挿入します。値が 9 桁を超える場合は欠損値として書き出します。

例

```
put id ssn.;
```

id の値	結果
	----+----1----+
263878439	263-87-8439

TIME*w.d* 出力形式

SAS 時間値を *hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

TIME*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–20

ヒント: *w* には、必要な結果を生成するために十分な大きさを指定します。小数点以下 3 桁までをすべて含む時間値を取得するには、少なくとも 12 スペース(小数点の左側に 8 スペース、小数点自体に 1 スペース、秒の小数部分に 3 スペース)を指定する必要があります。

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–19

要件 *w* より小さい数にします。

詳細

TIME*w.d* 出力形式は、SAS 時間値を *hh:mm:ss.ss* 形式で書き出します。

hh
整数です。

注: hh が 1 桁の場合、TIME*w.d* はその桁の前に先頭の空白を置きます。たとえば、TIME*w.d.*出力形式は、09:00 ではなく 9:00 と書き出します。

mm
分を表す 00 から 59 までの整数です。

ss.ss
小数点以下の秒の端数を含む、00 から 59 の範囲の秒数です。

比較

TIME*w.d* 出力形式は HHMM*w.d* 出力形式と同じですが、TIME*w.d* は秒を含めます。

TIME*w.d* 出力形式は、1 桁の時間にはその先頭に空白を書き出します。TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。

例

例 1

この例では、59083 を入力値として使用します。この値は、午後 4:24:43 に相当する SAS 時間です。

SAS ステートメント	結果
	`----+----1`
`put begin time.;`	`16:24:43`

例 2

この例では、32083 を入力値として使用します。この値は、午前 8:54:43 に相当する SAS 時間値です。

SAS ステートメント	結果
	`----+----1`
`put begin time.;`	`8:54:43`

関連項目:

出力形式:

- “HHMMw.d 出力形式” (100 ページ)
- “HOURw.d 出力形式” (101 ページ)
- “MMSSw.d 出力形式” (113 ページ)
- “TODw.d 出力形式” (155 ページ)

関数:

- “HOUR Function” in *SAS Functions and CALL Routines: Reference*
- “MINUTE Function” in *SAS Functions and CALL Routines: Reference*
- “SECOND Function” in *SAS Functions and CALL Routines: Reference*
- “TIME Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “TIMEw. 入力形式” (324 ページ)

TIMEAMPM*w.d* 出力形式

SAS 時間値と SAS 日時値を AM または PM を使用して *hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

TIMEAMPM*w.d*

説明

w
出力フィールドの幅を指定します。

デフォルト: 11

範囲: 2–20

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–19

要件 *w* より小さい数にします。

詳細

TIMEAMPM*w.d* 出力形式は、SAS 時間値と SAS 日付値を AM または PM を使用して *hh:mm:ss.ss* 形式で書き出します。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

23:59:59 PM を超える時間は、次の日として表示されます。

w には、必要な結果を生成するために十分な大きさを指定します。小数点以下 3 桁と AM または PM をすべて含む時間値を取得するには、少なくとも 11 スペース (*hh:mm:ss* PM)を指定する必要があります。*w* が 5 未満の場合、AM または PM だけが書き出されます。

比較

- TIMEAMPMM*w.d* 出力形式は TIMEM*w.d* 出力形式と同じですが、TIMEAMPMM*w.d* は時間の最後に AM または PM を出力します。
- TIME*w.d* は 23:59:59 PM を超える時間を書き出しますが、TIMEAMPM*w.d* は書き出しません。

例

例の表では、59083 を入力値として使用します。この値は、午後 4:24:43 に相当する SAS 時間値です。

SAS ステートメント	結果
	----+----1----+
put begin timeampm3.;	PM
put begin timeampm5.;	4 PM
put begin timeampm7.;	4:24 PM
put begin timeampm11.;	4:24:43 PM

関連項目:

出力形式:

- “TIMEw.d 出力形式” (152 ページ)

TOD*w.d* 出力形式

SAS 時間値と、SAS 日時値の時間部分を *hh:mm:ss.ss* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

TOD*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–20

ヒント: 指定した幅が十分な場合は、ゼロ時間に対しゼロが書き出されます。たとえば、02:30 または 00:30 となります。

d

秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–19

要件 *w* より小さい数にします。

詳細

TOD*w.d* 出力形式は、SAS 時間値および SAS 日時値を *hh:mm:ss.ss* 形式で書き出します。

hh
時間を表す整数です。

mm
分を表す整数です。

ss.ss
小数点以下 2 桁の秒数です。

比較

TOD*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。TIME*w.d* 出力形式と HHMM*w.d* 出力形式は、1 桁の時間にはその先頭にゼロを書き出します。

例

例 1

この例では、SAS 日時値 1661437223 は 2012 年 8 月 24 日の午後 2:20:23 に相当します。

SAS ステートメント	結果
	`----+----1`
`begin = '1:30't;` `put begin tod5.;`	`01:30`
`begin = 1661437223;` `put begin tod9.;`	`14:20:23`

例 2

この例では、SAS 時間値 32083 は午前 8:54:43 に相当します。

SAS ステートメント	結果
	`----+----1`
`begin = 32083;` `put begin tod9.;`	`08:54:43`

関連項目:

出力形式:

- “HHMMw.d 出力形式” (100 ページ)
- “TIMEw.d 出力形式” (152 ページ)
- “TIMEAMPMw.d 出力形式” (154 ページ)

関数:

- “TIMEPART Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “TIMEw. 入力形式” (324 ページ)

VAXRB*w.d* 出力形式

VMS 形式のバイナリ実数(浮動小数点)を書き出します。

カテゴリ: 数値

配置: 右

構文

VAXRB*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 8
範囲: 2-8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

VAXRB*w.d* 出力形式を使用して、ネイティブな VAX 浮動小数点数表記または VMS 浮動小数点数表記のデータを書き出します。

比較

VAX または VMS で実行している SAS を使用する場合、VAXRB*w.d* と RB*w.d* 出力形式は同じものです。

例

```
x=1;
y=put(x,vaxrb8.);
put y=$hex16.;
```

x の値	結果*
----+----1	

x の値	結果*
1	8040000000000000

* 結果は、整数を 16 進表現で表したものです。

VMSZN*w.d* 出力形式

VMS および MicroFocus COBOL ゾーン数値データを生成します。

カテゴリ: 数値

配置: 左

構文

VMSZN*w.d*

必須引数

w
出力フィールドの幅を指定します。
デフォルト: 1
範囲: 1–32

d
数値の小数点以下の桁数を指定します。この引数はオプションです。

詳細

VMSZN*w.d* 出力形式は ZD*w.d* 出力形式と同じです。どちらの出力形式も桁を ASCII で表現した文字列を生成します。最後の桁は、値の最後の桁の大きさと値の符号を示す特殊文字になります。相違点は、最後の桁に使用される特殊文字が異なります。次の表は、VMSZN*wd* 出力形式が使用する特殊文字を示したものです。

表現する桁	特殊文字	表現する桁	特殊文字
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

表現する桁	特殊文字	表現する桁	特殊文字
8	8	–8	x
9	9	–9	y

VMSZN*w.d* 出力形式が適用されたデータは、ASCII 文字列になります。

フィールドの指定幅に対して出力形式の適用後の値が長いすぎる場合、VMSZN*w.d* 出力形式は次の調整を行います。

- 正の値の場合、出力を指定幅におさまる最大の正の値に設定します。
- 負の値の場合、出力を指定幅におさまれる最大の大きさの負の数に設定します。

例

SAS ステートメント	結果
	`----+----1`
`x=1234;` `put x vmszn4.;`	`1234`
`x=1234;` `put x vmszn5.1;`	`12340`
`x=1234;` `put x vmszn6.2;`	`123400`
`-1234;` `put x vmszn5.;`	`0123t`

関連項目:

出力形式:

- “ZDw.d 出力形式” (188 ページ)

入力形式:

- “VMSZNw.d 入力形式” (328 ページ)

w.d 出力形式

標準数値でデータを書き出します(1 バイト 1 桁)。

カテゴリ: 数値

配置: 右

別名: F*w.d*

参照項目: “w.d Format: z/OS” in *SAS Companion for z/OS*

構文

w.d

説明

w
出力フィールドの幅を指定します。

範囲: 1–32

ヒント: 必要に応じて、値、小数点、マイナス記号を書き出せる十分なスペースを指定します。

d
数値の小数点以下の桁数を指定します。この引数はオプションです。

範囲: 0–31

要件 *w* より小さい数にします。

ヒント: *d* が 0、または *d* を指定しない場合、*w.d* は値を小数点なしで書き出します。

詳細

w.d 出力形式は、出力フィールドにおさまる近傍値になるように値を丸めます。*w.d* が小さすぎる場合、小数点揃えを BEST*w*.出力形式に合わせます。*w.d* 出力形式は、負の数には先頭のマイナス記号を付けて書き出します。また、*w.d* は、出力を右揃えにしてから書き出し、先頭に空白を挿入します。

比較

Z*w.d* 出力形式は *w.d* 出力形式と同じですが、Z*w.d* は右揃えにした出力に空白ではなく 0 を挿入します。

例

```
put @7 x 6.3;
```

x の値	結果
	----+----1----+
23.45	23.450

WEEKDATE*w.* 出力形式

SAS 日付値を *day-of-week*, *month-name dd*, *yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDATE*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 29

範囲: 3–37

詳細

WEEKDATE*w*.出力形式は、SAS 日付値を *day-of-week*, *month-name dd*, *yy* (または *yyyy*)形式で書き出します。

dd
月の日を表す整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

w が小さすぎて曜日と月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WEEKDATE*w*.出力形式は WEEKDATX*w*.出力形式と同じですが、WEEKDATX*w*. は月名の前に *dd* を出力します。

例

例の表では、19158 を入力値として使用します。この値は 2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+----2`
`put date weekdate3.;`	`Thu`
`put date weekdate9.;`	`Thursday`
`put date weekdate15.;`	`Thu, Jun 14, 12`
`put date weekdate17.;`	`Thu, Jun 14, 2012`

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “TODw.d 出力形式” (155 ページ)

- “WEEKDATXw. 出力形式” (162 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “WEEKDAY Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

WEEKDATX*w.* 出力形式

SAS 日付値を *day-of-week*, *dd month-name yy* (または *yyyy*)形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDATX*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 29
範囲: 3–37

詳細

WEEKDATX*w.*出力形式は、SAS 日付値を *day-of-week*, *dd month-name*, *yy* (または *yyyy*)形式で書き出します。

dd
月の日を表す整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

w が小さすぎて曜日と月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WEEKDATE*w.*出力形式は WEEKDATX*w.*出力形式と同じですが、WEEKDATE*w.* は月名の後に *dd* を出力します。

WEEKDATX*w*.出力形式は DTWKDATX*w*.出力形式と同じですが、DTWKDATX*w*. は入力として日時値が必要です。

例

例の表では、19046 を入力値として使用します。この値は 2012 年 2 月 23 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1----+----2----+----3
put date weekdatx.;	Thursday, 23 February 2012

関連項目:

出力形式:

- “DTWKDATXw. 出力形式” (84 ページ)
- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “TODw.d 出力形式” (155 ページ)
- “WEEKDATEw. 出力形式” (160 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “WEEKDAY Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

WEEKDAY*w*. 出力形式

SAS 日付値から曜日の値を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WEEKDAY*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 1
範囲: 1–32

詳細

WEEKDAY*w.*出力形式は、SAS 日付値を曜日の値(1=日曜日、2=月曜日など)として書き出します。

例

例の表では、19025 を入力値として使用します。この値は 2012 年 2 月 2 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date weekday.;`	`5`

関連項目:

出力形式:

- “DOWNAMEw. 出力形式” (81 ページ)

WEEKU*w.* 出力形式

U アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間
配置: 左

構文

WEEKU*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 11
範囲: 3–200

詳細

WEEKU*w*.出力形式では、週番号形式で書き出します。WEEKU*w*.出力形式では、指定された幅に応じてさまざまな形式で書き出します。U アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(日曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

幅、出力形式および例については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKV*w*.出力形式では、01 から 53 の範囲の 10 進数として週番号を書き出します。各週は月曜日から始まり、年の第 1 週には 1 月 4 日と年の最初の木曜日の両方が含まれます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w*.出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き出します。第 1 週の最初の日は月曜日になります。WEEKU*w*.出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き出します(日曜日が週の最初の日です)。

例

```
sasdate = '31JAN2012'd;
```

ステートメント	結果
	`----+----1----+`

`v=put(sasdate,weeku3.);`
`w=put(sasdate,weeku5.);`
`x=put(sasdate,weeku7.);`
`y=put(sasdate,weeku9.);`
`z=put(sasdate,weeku11.);`
`put v;`
`put w;`
`put x;`
`put y;`
`put z;`
	`W05`
	`12W05`
	`12W0503`
	`2012W0503`
	`2012-W05-03`

関連項目:

出力形式:

- “WEEKVw. 出力形式” (166 ページ)
- “WEEKWw. 出力形式” (168 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKUw. 入力形式” (331 ページ)
- “WEEKVw. 入力形式” (333 ページ)
- “WEEKWw. 入力形式” (334 ページ)

WEEKV*w.* 出力形式

V アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間

配置: 左

構文

WEEKV*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 11
範囲: 3–200

詳細

WEEKV*w.*出力形式では、指定した幅に応じてさまざまな形式を書き出します。V アルゴリズムでは、先頭に 0 を付けた 01 から 53 の範囲の 10 進数として表され、最大値が 53 になる週番号値を使用して、SAS 日付値を計算します。各週は月曜日から始まり、年の第 1 週は 1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。たとえば、年の 5 週目は 06 として表されます。

幅、出力形式および例については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101

幅	出力形式	例
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKV*w*.出力形式では、01 から 53 の範囲の 10 進数として週番号を書き出します。各週は月曜日から始まり、年の第 1 週には 1 月 4 日と年の最初の木曜日の両方が含まれます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w*.出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き出します。第 1 週の最初の日は月曜日になります。WEEKU*w*.出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き出します(日曜日が週の最初の日です)。

例

```
sasdate='31JAN2012'd;
```

ステートメント	結果
	----+----1----+

v=put(sasdate,weekv3.);
w=put(sasdate,weekv5.);
x=put(sasdate,weekv7.);
y=put(sasdate,weekv9.);
z=put(sasdate,weekv11.);
put v;
put w;
put x;
put y;
put z;
	
	W05
	12W01
	12W0502
	2012W0502
	2012-W05-02

関連項目:

出力形式:

- “WEEKUw. 出力形式” (164 ページ)
- “WEEKWw. 出力形式” (168 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKUw. 入力形式” (331 ページ)
- “WEEKVw. 入力形式” (333 ページ)
- “WEEKWw. 入力形式” (334 ページ)

WEEKW*w.* 出力形式

W アルゴリズムを使用して、10 進数の週番号を書き出します。

カテゴリ: 日付と時間

配置: 左

構文

WEEKW*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 11

範囲: 3–200

詳細

WEEKW*w.*出力形式では、指定した幅に応じてさまざまな形式を書き出します。W アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(月曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

幅、出力形式および例については、次の表を参照してください。

幅	出力形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKV*w.*出力形式では、01 から 53 の範囲の 10 進数として週番号を書き出します。月曜日と年の第 1 週から始まる週には、1 月 4 日と年の最初の木曜日が含まれます。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。WEEKW*w.*出力形式では、00 から 53 の範囲の 10 進数として年の週番号を書き出します。第 1 週の最初の日は月曜日になります。WEEKU*w.*出力形式では、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として年の週番号を書き出します(日曜日が週の最初の日です)。

例

```
sasdate = '31JAN2012'd;
```

ステートメント	結果
	----+----1----+

v=put(sasdate,weekw3.);
w=put(sasdate,weekw5.);
x=put(sasdate,weekw7.);
y=put(sasdate,weekw9.);
z=put(sasdate,weekw11.);
put v;
put w;
put x;
put y;
put z;
	
	W05
	12W05
	12W0502
	2012W0502
	2012-W05-02

関連項目:

出力形式:

- “WEEKUw. 出力形式” (164 ページ)
- “WEEKVw. 出力形式” (166 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKUw. 入力形式” (331 ページ)
- “WEEKVw. 入力形式” (333 ページ)
- “WEEKWw. 入力形式” (334 ページ)

WORDDATE*w.* 出力形式

SAS 日付値を *month-name dd*, *yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WORDDATE*w.*

説明

w
出力フィールドの幅を指定します。
デフォルト: 18
範囲: 3–32

詳細

WORDDATE*w*.出力形式は、SAS 日付値を *month-name dd, yyyy* 形式で書き出します。

dd
月の日を表す整数です。

yyyy
年を表す 4 桁の整数です。

幅が小さすぎて月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WORDDATE*w*.出力形式は WORDDATX*w*.出力形式と同じですが、WORDDATX*w*. は月名の前に *dd* を出力します。

例

例の表では、19158 を入力値として使用します。この値は 2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
`put term worddate3.;`	`Jun`
`put term worddate9.;`	`----+----1----+----2` ` June`
`put term worddate12.;`	`Jun 14, 2012`
`put term worddate20.;`	`----+----1----+----2` ` June 14, 2012`

関連項目:

出力形式:

- “WORDDATXw. 出力形式” (170 ページ)

WORDDATX*w*. 出力形式

SAS 日付値を *dd month-name yyyy* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

WORDDATX*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 18
範囲: 3–32

詳細

WORDDATX*w*.出力形式は、SAS 日付値を *dd month-name, yyyy* 形式で書き出します。

dd
月の日を表す整数です。

yyyy
年を表す 4 桁の整数です。

幅が小さすぎて月の名前を完全な形で書き出すことができない場合、必要に応じて短縮形で書き出します。

比較

WORDDATX*w*.出力形式は WORDDATE*w*.出力形式と同じですが、WORDDATE*w*. は月名の後に *dd* を出力します。

例

例の表では、19057 を入力値として使用します。この値は 2012 年 3 月 5 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+----2`
`put term worddatx.;`	`05 March 2012`

関連項目:

出力形式:

- “WORDDATEw. 出力形式” (169 ページ)

WORDF*w*. 出力形式

数値を英語表現で書き出します。分数は数値で表示します。

カテゴリ: 数値
配置: 左

構文

WORDF*w*.

説明

w
出力フィールドの幅を指定します。
デフォルト: 10
範囲: 5–32767

詳細

WORDF*w*.出力形式は、数値を分数付きの英語表現に変換します。たとえば、8.2 は eight and 20/100 と出力されます。

負数は、minus という英語表現が値の前に付きます。英語表現に相当する値は指定したフィールドに表示しきれない場合、右側が切り捨てられ、最後の文字がアスタリスクとして出力されます。

比較

WORDF*w*.出力形式は WORDS*w*.出力形式とほぼ同じですが、WORDF*w*.は小数を英語表現でなく数値として出力します。

例

```
put price wordf15.;
```

price の値	結果
	----+----1----+
2.5	two and 50/100

関連項目:

出力形式:

- “WORDSw. 出力形式” (172 ページ)

WORDS*w*. 出力形式

数値を英語表現で書き出します。

カテゴリ: 数値
配置: 左

構文

WORDS*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 10

範囲: 5–32767

詳細

WORDS*w*.出力形式を使用して、受取人行の下に金額が書かれた小切手を出力できます。

負数は、minus という英語表現が値の前に付きます。数値が整数でない場合、小数部分は hundredths と表現されます。たとえば、5.3 は five and thirty hundredths と出力されます。英語表現に相当する値は指定したフィールドに表示しきれない場合、右側が切り捨てられ、最後の文字がアスタリスクとして出力されます。

比較

WORDS*w*.出力形式は WORDF*w*.出力形式とほぼ同じですが、WORDS*w*.は小数を数値ではなく英語表現で出力します。

例

```
put price words23.;
```

price の値	結果
	----+----1----+----2----+
2.1	two and ten hundredths

関連項目:

出力形式:

- “WORDFw. 出力形式” (171 ページ)

YEAR*w*. 出力形式

SAS 日付値から年部分を書き出します。

カテゴリ: 日付と時間

配置: 右

構文

YEAR*w*.

説明

w

出力フィールドの幅を指定します。

デフォルト: 4

範囲: 2–32

ヒント: *w* が 4 より小さい場合、年の最後の 2 桁が出力されます。4 より大きい場合は、年値は 4 桁で出力されます。

詳細

YEAR*w.*出力形式は、日付値を書き出すという点で DTYEAR*w.*出力形式とほぼ同じです。相違点は、YEAR*w.*が入力として SAS 日付値が必要であり、DTYEAR*w.*が日時値が必要である点です。

例

例の表では、19158 を入力値として使用します。この値は 2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date year2.;`	`12`
`put date year4.;`	`2012`

関連項目:

出力形式:

- “DTYEARw. 出力形式” (85 ページ)

YYMM*w.* 出力形式

SAS 日付値を<*yy*>*yy*M*mm* 形式で書き出します。M は月番号が M の後に続くことを示す区切り文字で、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYMM*w.*

説明

w

出力フィールドの幅を指定します。

デフォルト: 7

範囲: 5–32

操作: *w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYMM*w*.出力形式は、SAS 日付値を<*yy*>*yy*M*mm* 形式で書き出します。

<*yy*>*yy*
年を表す 2 桁または 4 桁の整数です。

M
月の番号が後に続くことを示す区切り文字です。

mm
月を表す整数です。

例

次の例では、19291 を入力値として使用します。この値は、2012 年 10 月 25 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date yymm5.;`	`12M10`
`put date yymm6.;`	` 12M10`
`put date yymm.;`	`2012M10`
`put date yymm7.;`	`2012M10`
`put date yymm10.;`	`----+----1----+` ` 2012M10`

関連項目:

出力形式:

- “MMYYw. 出力形式” (114 ページ)
- “YYMMxw. 出力形式” (179 ページ)

YYMMDD*w*. 出力形式

SAS 日付値を *yymmdd* または<*yy*>*yy-mm-dd* 形式で書き出します。ハイフンが区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYMMDD*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作: *w* の値が 2 から 5 までの場合、日付は可能な限りで年と月だけを表示されます。*w* が 7 の場合、日付は 2 桁の年としてハイフンなしで表示されます。

詳細

YYMMDD*w* 出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

yymmdd

<yy>yy–mm–dd

<yy>yy
年を表す 2 桁または 4 桁の整数です。

–
区切り文字です。

mm
月を表す整数です。

dd
月の日を表す整数です。

4 桁の年表示で区切り文字のない日付を出力するには、YYMMDD*x*.出力形式を使用します。

例

次の例では、19086 を入力値として使用します。この値は、2012 年 4 月 3 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put day yymmdd2.;`	`12`
`put day yymmdd3.;`	`12`
`put day yymmdd4.;`	`1204`
`put day yymmdd5.;`	`12-04`
`put day yymmdd6.;`	`120403`
`put day yymmdd7.;`	`120403`
`put day yymmdd8.;`	`12-04-03`
`put day yymmdd10.;`	`2012-04-03`

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDxw. 出力形式” (177 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式::

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)

YYMMDD*xw.* 出力形式

SAS 日付値を *yymmdd* または<*yy*>*yy-mm-dd* 形式で書き出します。出力形式名の *x* は、年、月、日を区切る特殊文字を表す文字です。特殊文字は、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

YYMMDD*xw.*

説明

x

区切り文字を示します。または、区切り文字を年、月、日の間に挿入しないことを示します。*x* の値として、次の値が有効です。

B

空白で区切ります。

C

コロンで区切ります。

D

ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 2–10

操作:
w の値が 2 から 5 までの場合、日付は可能な限りで年と月だけを表示されます。*w* が 7 の場合、日付の年は 2 桁で、区切り文字を使用せずに表示されます。
x の値が N の場合、幅の範囲は 2–8 に変化します。

詳細

YYMMDD*xw*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

yymmdd

<yy>yyxmmxdd

<yy>yy
年を表す 2 桁または 4 桁の整数です。

x
指定された区切り文字です。

mm
月を表す整数です。

dd
月の日を表す整数です。

例

次の例では、19127 を入力値として使用します。この値は、2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put day yymmddc5.;`	`12:05`
`put day yymmddd8.;`	`12-05-14`
`put day yymmddp10.;`	`2012.05.14`
`put day yymmddn8.;`	`20120514`

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYxw. 出力形式” (76 ページ)
- “MMDDYYxw. 出力形式” (111 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “YYMMDDw. 入力形式” (338 ページ)

YYMM*xw.* 出力形式

SAS 日付値を<*yy*>*yymm* または<*yy*>*yy-mm* 形式で書き出します。出力形式名の *x* は、年と月を区切る特殊文字を表します。この特殊文字は、ハイフン(-)、ピリオド(.)、スラッシュ(/)、コロン(:)または区切り文字なしになります。年は 2 桁または 4 桁で表示されます。

構文

YYMM*xw.*

説明

x

区切り文字を示します。または、区切り文字を年と月の間に挿入しないことを示します。*x* の値として、次の値が有効です。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w

出力フィールドの幅を指定します。

デフォルト: 7

範囲: 5–32

操作:

x を *N* に設定すると、区切り文字なしを指定します。この場合の幅範囲は 4–32 になり、デフォルトは 6 に変化します。

x の値が C、D、P または S で、*w* の値が 5 または 6 の場合、日付の年は下 2 桁のみ表示されます。*w* が 7 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 4 または 5 の場合、日付の年は下 2 桁のみ表示されます。*x* の値が *N* で、*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYMM*xw*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

<*yy*>*yymm*

<*yy*>*yy*X*mm*

<*yy*>*yy*
年を表す 2 桁または 4 桁の整数です。

x
指定された区切り文字です。

mm
月を表す整数です。

例

次の例では、19127 を入力値として使用します。この値は、2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date yymmc5.;`	`12:05`
`put date yymmd.;`	`2012-05`
`put date yymmn4.;`	`1205`
`put date yymmp8.;`	` 2012.05`
`put date yymms10.;`	` 2012/05`

関連項目:

出力形式:

- “MMYYxw. 出力形式” (116 ページ)
- “YYMMw. 出力形式” (174 ページ)

YYMON*w.* 出力形式

SAS 日付値を *yymmm* または *yyyymmm* 形式で書き出します。

カテゴリ: 日付と時間

配置: 右

構文

YYMON*w.*

説明

w
出力フィールドの幅を指定します。出力形式の幅が小さすぎて 4 桁の年を出力できない場合、年の下 2 桁だけが出力されます。

デフォルト: 7

範囲: 5–32

詳細

YYMON*w.*出力形式は、SAS 日付値を*<yy>yymmm* 形式で書き出します。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

mmm
3 文字に省略された月名です。

例

例の表では、19158 を入力値として使用します。この値は 2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1`
`put date yymon6.;`	`02JUN`
`put date yymon7.;`	`2012JUN`

関連項目:

出力形式:

- “MMYYw. 出力形式” (114 ページ)

YYQ*w.* 出力形式

SAS 日付値を<*yy*>*yy*Q*q* 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また、*q* は四半期を表します。

カテゴリ: 日付と時間

配置: 右

構文

YYQ*w.*

説明

w
出力フィールドの幅を指定します。

デフォルト: 6

範囲: 4–32

操作: *w* の値が 4 または 5 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYQ*w.*出力形式は、SAS 日付値を<*yy*>*yy*Q*q* 形式で書き出します。

<*yy*>*yy*
年を表す 2 桁または 4 桁の整数です。

Q
区切り文字です。

q
四半期を表す整数(1、2、3、4 のいずれか)です。

例

次の例では、19158 を入力値として使用します。この値は、2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date yyq4.;`	`12Q2`
`put date yyq5.;`	`12Q2`
`put date yyq.;`	`2012Q2`
`put date yyq6.;`	`2012Q2`
`put date yyq10.;`	`2012Q2`

関連項目:

出力形式:

- “YYQxw. 出力形式” (183 ページ)
- “YYQRw. 出力形式” (184 ページ)

YYQ*xw*. 出力形式

SAS 日付値を*<yy>yyq* または*<yy>yy-q* 形式で書き出します。出力形式名の *x* は、年と四半期または年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます。

カテゴリ: 日付と時間

配置: 右

構文

YYQ*xw*.

説明

x

区切り文字を示します。または、区切り文字を年と四半期の間に挿入しないことを示します。*x* の値として、次の値が有効です。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w

出力フィールドの幅を指定します。

デフォルト: 6

範囲: 4–32

操作:

x を *N* に設定すると、区切り文字なしを指定します。この場合の幅範囲は 3–32 になり、デフォルトは 5 に変化します。

w の値が 4 または 5 の場合、日付の年は下 2 桁でのみ表示されます。*w* が 6 以上の場合、日付の年は 4 桁で表示されます。

x の値が *N* で、*w* の値が 3 または 4 の場合、日付の年は下 2 桁でのみ表示されます。*x* の値が *N* で、*w* の値が 5 以上の場合、日付の年は 4 桁で表示されます。

詳細

YYQ*xw*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

<yy>yyq

<yy>yyxq

<yy>yy
年を表す 2 桁または 4 桁の整数です。

x
指定された区切り文字です。

q
四半期を表す整数(1、2、3、4 のいずれか)です。

例

次の例では、19188 を入力値として使用します。この値は、2012 年 7 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	`----+----1----+`
`put date yyqc4.;`	`12:3`
`put date yyqd.;`	`2012-3`
`put date yyqn3.;`	`123`
`put date yyqp6.;`	`2012.3`
`put date yyqs8.;`	`2012/3`

関連項目:

出力形式:

- “YYQw. 出力形式” (182 ページ)
- “YYQRxw. 出力形式” (185 ページ)

YYQR*w*. 出力形式

SAS 日付値を*<yy>yyQqr* 形式で書き出します。Q は区切り文字として使用され、年は 2 桁または 4 桁の数字で書き出されます。また *qr* はローマ数字表現の四半期を表します。

カテゴリ: 日付と時間

配置: 右

構文

YYQR*w*.

説明

w
出力フィールドの幅を指定します。

デフォルト: 8

範囲: 6–32

操作: *w* の値が小さすぎて 4 桁の年を書き出せない場合、日付の年は下 2 桁でのみ表示されます。

詳細

YYQR*w*.出力形式は、SAS 日付値を*<yy>yy*Q*qr* 形式で書き出します。

<yy>yy
年を表す 2 桁または 4 桁の整数です。

Q
区切り文字です。

qr
四半期を表すローマ数字表現(I、II、III、IV のいずれか)です。

例

次の例では、19158 を入力値として使用します。この値は、2012 年 6 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1----+
put date yyqr6.;	12QII
put date yyqr7.;	2012QII
put date yyqr.;	2012QII
put date yyqr8.;	2012QII
put date yyqr10.;	2012QII

関連項目:

出力形式:

- “YYQw. 出力形式” (182 ページ)
- “YYQRxw. 出力形式” (185 ページ)

YYQR*xw*. 出力形式

SAS 日付値を*<yy>yyqr* または*<yy>yy-qr* 形式で書き出します。出力形式名の *x* は、年と四半期、または年を区切る特殊文字を表す文字で、ハイフン(-)、ピリオド(.)、空白文字、スラッシュ(/)、コロン(:)、または区切り文字なしになります。年は 2 桁または 4 桁の数字で書き出されます、また *qr* は、ローマ数字表現の四半期です。

カテゴリ: 日付と時間

配置: 右

構文

YYQR*xw*.

説明

x

区切り文字を示します。または、区切り文字を年と四半期の間に挿入しないことを示します。*x* の値として、次の値が有効です。

C
コロンで区切ります。

D
ハイフンで区切ります。

N
区切り文字なしを表します。

P
ピリオドで区切ります。

S
スラッシュで区切ります。

w

出力フィールドの幅を指定します。

デフォルト: 8

範囲: 6–32

操作:

x を *N* に設定すると、区切り文字なしを指定します。幅範囲は 5–32 になり、デフォルトは 7 に変化します。

w の値が小さすぎて 4 桁の年を書き出せない場合、日付の年は下 2 桁でのみ表示されます。

詳細

YYQR*xw*.出力形式は、SAS 日付値を次の形式のいずれかで書き出します。

<yy>yyqr

<yy>yyxqr

<yy>yy
年を表す 2 桁または 4 桁の整数です。

x
指定された区切り文字です。

qr
四半期を表すローマ数字表現(I、II、III、IV のいずれか)です。

例

次の例では、19127 を入力値として使用します。この値は、2012 年 5 月 14 日に相当する SAS 日付値です。

SAS ステートメント	結果
	----+----1----+
put date yyqrc6.;	12:II
put date yyqrd.;	2012-II
put date yyqrn5.;	12II
put date yyqrp8.;	2012.II
put date yyqrs10.;	2012/II

関連項目:

出力形式:

- “YYQxw. 出力形式” (183 ページ)
- “YYQRw. 出力形式” (184 ページ)

Zw.d 出力形式

先頭に 0 を含む標準数値データを書き出します。

カテゴリ: 数値

配置: 右

構文

Z*w.d*

説明

w

出力フィールドの幅を指定します。

デフォルト: 1

範囲: 1–32

ヒント: 必要に応じて、値、小数点、マイナス記号を書き出せる十分なスペースを指定します。

d

数値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

ヒント: *d* が 0 の場合、または *d* の指定を省略する場合、*Zw.d* は値を小数点なしで書き出します。

詳細

Z*w.d* 出力形式は、標準数値(1 バイト 1 桁)を書き出し、データ値の左側に 0 を入力します。

Z*w.d* 出力形式は、出力フィールドに合う近傍値になるように値を丸めます。*w.d* が大きすぎておさまらない場合、小数点揃えを BEST*w.*出力形式に合わせます。Z*w.d* 出力形式は、負の数には先頭のマイナス記号を付けて書き出します。また、出力を右揃えしてから書き出し、先頭にゼロを挿入します。

比較

Z*w.d* 出力形式は *w.d* 出力形式とほぼ同じですが、Z*w.d* は右揃えにした出力に空白ではなく 0 を挿入します。

例

```
put @5 seqnum z8.;
```

seqnum の値	結果
	----+----1
1350	00001350

ZD*w.d* 出力形式

ゾーン 10 進形式の数値データを書き出します。

カテゴリ: 数値

配置: 左

参照項目: “ZDw.d Format: UNIX” in *SAS Companion for UNIX Environments*
“ZDw.d Format: Windows” in *SAS Companion for Windows*
“ZDw.d Format: z/OS” in *SAS Companion for z/OS*

構文

ZD*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 1
範囲: 1–32

d
10^d で数値を乗算するよう指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

ゾーン 10 進出力形式は、標準数値形式と同じよう 1 桁には 1 バイトが必要です。ただし、値の符号は最後の桁とともに最下位バイトに示されます。

注: 動作環境によってゾーン 10 進値の保存方法は異なります。ただし、ZD*w.d* 出力形式では、SAS の実行に使用する動作環境で値が作成された場合と同じように、ゾーン 10 進値を書き出します。

比較

次の表は、各種プログラミング言語でのゾーン 10 進形式の表記を比較したものです。

言語	ゾーン 10 進表記
SAS	ZD3.
PL/I	PICTURE '99T'
COBOL	DISPLAY PIC S 999
IBM 370 アセンブラ	ZL3

例

```
y=put(x,zd4.);
put y $hex8.;
```

x の値	結果*
120	F0F1F2C0

* 結果は、IBM メインフレームで見られる、ゾーン 10 進形式のバイナリ値を 16 進表現で表したものです。1 バイトは、出力フィールドの 1 カラムを使用します。

2 部

SAS 入力形式

3 章

入力形式について

入力形式について

入力形式は、入力として読み込まれるデータ値へのパターンの適用や命令の実行を行う SAS 言語要素です。入力形式の種類は、データの種類(数値、文字、日付、時間、タイムスタンプ)に対応します。ユーザー定義の入力形式を作成することもできます。SAS 入力形式の例として、BINARY、DATE、COMMA などがあります。たとえば、次の値にはドル記号とカンマが含まれています。

```
$1,000,000
```

数値 1000000 を変数に保存する前に、ドル記号($)とカンマ(,)を削除するには、この値を COMMA11.入力形式で読み込みます。

最初に変数を明示的に定義しない限り、SAS では入力形式を使用して、変数が数値または文字かどうかを特定します。入力形式は、文字変数の長さを特定するためにも使用されます。

構文

SAS 入力形式の形式は次のようになります。

<$>informat<w>.<d>

$
文字出力形式であることを表します。数値出力形式には使用されません。

informat
入力形式名です。入力形式には、SAS 入力形式、PROC FORMAT の INVALUE ステートメントを使用して定義されたユーザー定義の入力形式があります。Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

w
入力形式の幅です。大部分の入力形式では、入力データのカラム数となります。

d
オプションの小数点以下の桁数です(数値入力形式)。入力データを 10 の *d* 乗で除算します。

注: 32 桁まで読み込めるように数値入力形式を指定している場合でも、多くのコンピュータで使用されている 8 バイトの浮動小数点数表記の制限により、有効桁数が 15 桁を超える数字では精度が失われることがあります。

入力形式は、名前の一部として必ずピリオド(.)を含みます。入力形式の *w* 値と *d* 値を省略すると、デフォルト値が使用されます。データに小数点が含まれる場合、*d* 値は無視され、実際に入力データにある小数点以下桁数が読み込まれます。

入力形式の幅が入力データのすべてのカラムを読み込むには狭すぎる場合、予期しない結果が発生することがあります。日付入力形式と時間入力形式では、この問題は頻繁に発生します。日、月、年、または時間の間に空白または特殊文字を含めるように入力形式の幅を調整する必要があります。日付値と時間値の詳細については、“Dates, Times, and Intervals” in Chapter 7 of *SAS Language Reference: Concepts* を参照してください。

入力形式に関する問題が発生した場合、SAS ログに記述され、欠損値が変数に割り当てられます。問題が発生するのは、文字データの読み込みに数値入力形式を使用するなど、互換性のない入力形式を使用する場合、または最後のカラムの特殊文字が読み込まれる日時入力形式の幅を指定する場合です。

入力形式の使用

入力形式の指定方法

入力形式指定の概要

入力形式を次の方法で指定できます。

- INPUT ステートメントで使用する
- 関数 INPUT、INPUTC、INPUTN と使用する

- DATA ステップまたは PROC ステップの INFORMAT ステートメントで使用する
- DATA ステップまたは PROC ステップの ATTRIB ステートメントで使用する

INPUT ステートメント

INPUT ステートメントで変数名の後に入力形式を使用する方法が、値を変数に読み込む最も簡単な方法です。たとえば、次の INPUT ステートメントでは、2 つの入力形式が使用されます。

```
input @15 style $3. @21 price 5.2;
```

$*w*.文字入力形式は、値を変数 STYLE に読み込みます。*w.d* 数値入力形式は、値を変数 PRICE に読み込みます。

INPUT ステートメントの詳細については、“INPUT Statement” in *SAS Statements: Reference* を参照してください。

INPUT 関数

INPUT 関数は、指定された入力形式を使用して、SAS 文字式を変換します。入力形式によって、結果の値が数値か文字かどうかが特定されます。そのため、データの変換に INPUT 関数を使用します。例:

```
TempCharacter='98.6';
TemperatureNumber=input(TempCharacter,4.);
```

ここでは、INPUT 関数を *w.d* 入力形式と組み合わせて、TempCharacter の文字値を数値に変換し、数値 98.6 を TemperatureNumber に割り当てます。

PUT 関数を SAS 出力形式と使用して、数値を文字値に変換します。数値から文字への変換の例については、“PUT Function” in *SAS Functions and CALL Routines: Reference* を参照してください。INPUT 関数の詳細については、“INPUT Function” in *SAS Functions and CALL Routines: Reference* を参照してください。

INFORMAT ステートメント

INFORMAT ステートメントは入力形式を変数と関連付けます。SAS では、入力形式を後続の INPUT ステートメントで使用して、値を変数に読み込みます。たとえば、次のステートメントで INFORMAT ステートメントは DATE*w.*入力形式を変数 Birthdate および Interview と関連付けます。

```
informat Birthdate Interview date9.;
input @63 Birthdate Interview;
```

INFORMAT ステートメントと関連付けられた入力形式は、INPUT ステートメントでコロン(:)フォーマット修飾子も一緒に指定された入力形式と同じ結果をもたらします。コロン(:)修飾子の使用の詳細については、“INPUT Statement, List” in *SAS Statements: Reference* を参照してください。そのため、SAS では、修飾リスト入力を使用して変数を読み込むため、次の事柄が発生します。

- 入力形式の *w* 値によって、外部ファイルのカラム位置、入力フィールド幅が特定されない
- 入力データに埋め込まれている空白は、INFILE ステートメントの DLM=オプションまたは DLMSTR=オプションを変更しない限り、区切り文字として処理される
- 文字入力形式の場合、入力形式の *w* 値が文字変数の長さを指定する
- 数値入力形式の場合、*w* 値が無視される
- 数値入力形式の場合、入力形式の *d* 値の機能は通常と変わらない

フォーマット入力やカラム入力などの別の入力スタイルを使用するように INPUT ステートメントをコード化した場合、INFORMAT ステートメントの使用時にはその入力スタイルは使用されません。

データの読み込みに修飾リスト入力を使用する方法の詳細については、“INPUT Statement, List” in *SAS Statements: Reference* を参照してください。

注: テキストファイルがローカルエンコーディング環境以外で作成された場合はいつでも、ENCODING=オプションに ASCII または EBCDIC 環境を指定する必要があります。たとえば、EBCDIC テキストファイルを ASCII プラットフォームで読み込む場合、FILENAME ステートメントまたは INFILE ステートメントの ENCODING=オプションを指定することをお勧めします。だたし、DSD を使用し、FILENAME ステートメントまたは INFILE ステートメントの DLM=オプションまたは DLMSTR=オプションを使用する場合、ENCODING=オプションが必要条件となります。これらのオプションは、セッションエンコーディングで示された特定の文字(引用符、カンマ、空白など)が必要なためです。エンコーディング固有の入力形式は、バイナリファイルに使用するためのものです。つまり、文字フィールドと非文字フィールドの両方が含まれます。

ATTRIB ステートメント

ATTRIB ステートメントは、他の属性と同様に、入力形式を 1 つ以上の変数と関連付けることができます。たとえば、次のステートメントの ATTRIB ステートメントは、DATE*w*.入力形式を変数 Birthdate および Interview と関連付けます。

```
attrib Birthdate Interview informat=date9.;
input @63 Birthdate Interview;
```

ATTRIB ステートメントで INFORMAT=オプションを使用して関連付けられた入力形式は、INPUT ステートメントでコロン(:)フォーマット修飾子も一緒に指定された入力形式と同じ結果をもたらします。コロン(:)修飾子の使用に関する詳細については、“INPUT Statement, List” in *SAS Statements: Reference* を参照してください。そのため、SAS では、INFORMAT ステートメントの場合と同じように、修飾リスト入力を使用して変数を読み込みます。

詳細については、“ATTRIB Statement” in *SAS Statements: Reference* を参照してください。

恒久的な関連付けと一時関連付け

INPUT ステートメントに入力形式を指定した場合、DATA ステップにおける入力データ値の読み込みにこの入力形式が使用されます。ただし、入力形式は恒久的には変数と関連付けられません。入力形式を恒久的に変数と関連付けるには、INFORMAT ステートメントまたは ATTRIB ステートメントを使用します。SAS では、SAS データセットのディスクリプタ情報を変更することにより、入力形式と変数を恒久的に関連付けます。

ユーザー定義の入力形式

Base SAS が提供する入力形式に加えて、独自の入力形式を作成できます。Base SAS では、PROC FORMAT を使用して、文字変数と数値変数の両方に対し独自の入力形式と出力形式を作成できます。ユーザー定義の入力形式の詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。

ユーザー定義の入力形式を使用する SAS プログラムの実行時に、これらの入力形式が使用できる必要があります。これらの入力形式を使用可能にするには 2 つの方法があります。

- 一時的でなく、恒久的な入力形式を PROC FORMAT で作成する

- 入力形式を作成する(PROC FORMAT ステップ)ソースコードを、それを使用する SAS プログラムと一緒に保存する

ユーザー定義の入力形式を見つけられないプログラムを実行する場合、結果は FMTERR=システムオプションの設定によって異なります。ユーザー定義の入力形式が見つからない場合、次のシステムオプションの結果は次のようになります。

システムオプション	結果
FMTERR	現在の DATA ステップまたは PROC ステップを停止するエラーを発生します。
NOFMTERR	処理は続行され、デフォルトの出力形式を置き換えます。

NOFMTERR を使用すると変数を処理できますが、ユーザー定義の入力形式によって提供される情報は失われます。このオプションにより DATA ステップでデータが誤って読み込まれ、結果が正しくない可能性があります。詳細については、“FMTERR System Option” in *SAS System Options: Reference* を参照してください。

問題を避けるために、使用されるすべてのユーザー定義の入力形式にプログラムのユーザーがアクセスできるかを確認します。

ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング

定義

バイナリ整数データの整数値は、通常、1 バイト、2 バイト、3 バイトの 3 つのサイズのいずれかで保存されます。整数のバイトオーダリングは、整数が生成されるプラットフォーム(動作環境)によって異なります。

バイトオーダリングは、“ビッグエンディアン”プラットフォームと“リトルエンディアン”プラットフォームとで異なります。これらの俗称は、IBM メインフレーム(ビッグエンディアン)と Intel 基準プラットフォーム(リトルエンディアン)のバイトオーダリングを表すために使用されます。SAS System では、IBM メインフレーム、HP-UX、AIX、SPARC の Solaris および Macintosh のプラットフォームをビッグエンディアンとします。SAS では、Intel ABI、Linux、OpenVMS Alpha、OpenVMS Integrity、x64 の Solaris、Tru64 UNIX および Windows のプラットフォームをリトルエンディアンとします。

バイトオーダリングについて

ビッグエンディアンプラットフォームでは、値 1 はバイナリで保存され、16 進表記で表されます。1 バイトは 01、2 バイトは 00 01、4 バイトは 00 00 00 01 としてそれぞれ保存されます。リトルエンディアンプラットフォームでは、値 1 は 1 バイトで 01 (ビッグエンディアンと同じ)、2 バイトで 01 00、4 バイトで 01 00 00 00 としてそれぞれ保存されます。

負の整数には、“2 つの補数”表現が使用されます。整数の最大有効バイトの高位ビットに設定されます。たとえば、–2 は、ビッグエンディアンプラットフォームでは、1 バイト、2 バイト、4 バイトの場合それぞれ FE、FF FE、FF FF FF FE として表されます。リトルエンディアンプラットフォームでは、それぞれ FE、FE FF、FE FF FF FF として表されます。これらは、16 進表現のバイナリ整数値–2 の出力結果です。

ビッグエンディアンまたはリトルエンディアンプラットフォームで生成されたデータの読み込み

SAS では、符号付き整数または符号なし整数をビッグエンディアンシステムまたはリトルエンディアンシステムで生成されていても読み込むことができます。同様に、ビッグエンディアン形式とリトルエンディアン形式の符号付き整数および符号なし整数を書き出すことができます。これらの整数の長さは、最大で 8 バイトまで可能です。

次の表は、プラットフォームのさまざまな組み合わせに使用する入力形式を示したものです。符号付き整数 列の“いいえ”は、符号なしで、負でない数字を示します。“はい”は、数字が負または正のいずれかであることを示します。

表 3.1 SAS 入力形式とバイトオーダリング

データが作成されたプラットフォーム	データが読み込まれるプラットフォーム	符号付き整数	入力形式
ビッグエンディアン	ビッグエンディアン	はい	IB または S370FIB
ビッグエンディアン	ビッグエンディアン	いいえ	PIB、S370FPIB、S370FIBU
ビッグエンディアン	リトルエンディアン	はい	IBR
ビッグエンディアン	リトルエンディアン	いいえ	PIBR
リトルエンディアン	ビッグエンディアン	はい	IBR
リトルエンディアン	ビッグエンディアン	いいえ	PIBR
リトルエンディアン	リトルエンディアン	はい	IB または IBR
リトルエンディアン	リトルエンディアン	いいえ	PIB または PIBR
ビッグエンディアン	いずれか	はい	S370FIB
ビッグエンディアン	いずれか	いいえ	S370FPIB
リトルエンディアン	いずれか	はい	IBR
リトルエンディアン	いずれか	いいえ	PIBR

各種プログラミング言語でのバイナリ整数表記

次の表は、プログラミング言語別のバイナリ整数表記を比較したものです。

言語	2 バイトまたは 8 ビットシステム	4 バイトまたは 16 ビットシステム	8 バイトまたは 64 ビットシステム
SAS	IB2.、IBR2.、PIB2、PIBR2.、S370FIB2.、S370FIBU2.、S370FPIB2.	IB4.、IBR4.、PIB4.、PIBR4.、S370FIB4.、S370FIBU4.、S370FPIB4.	IB8.、IBR8.、PIB8.、PIBR8.、S370FIB8.、S370FIBU8.、S370FPIB8.

言語	2 バイトまたは 8 ビットシステム	4 バイトまたは 16 ビットシステム	8 バイトまたは 64 ビットシステム
C	short	INT	long *
Java	short	INT	long *
Visual Basic 6.0	short	long*	なし
Visual Basic.NET	short	integer	long *
PL/I	FIXED BIN(15)	FIXED BIN(31)	FIXED BIN (63)
Fortran	INTEGER*2	INTEGER*4	INTEGER*8
COBOL	COMP PIC 9(4)	COMP PIC 9(8)	COMP PIC 9(16)
IBM アセンブラ	H	F	FD

* long として宣言された整数のサイズは、動作環境によって異なります。

パック 10 進データとゾーン 10 進データの処理

定義

パック 10 進
: 10 進数の 2 桁を 1 バイトで表現する、10 進数のエンコード方法です。パック 10 進表現は、正確な精度で 10 進データを保存します。数字の小数部分は、別に仮数や指数がないため、入力形式または出力形式によって設定されます。

  パック 10 進データを使用する利点は、正確な精度を維持できることです。ただし、10 進データを含む計算はネイティブ命令を欠く場合は不正確になることがあります。

ゾーン 10 進
: 1 桁がストレージの 1 バイトを使用する 10 進数のエンコード方法です。最終バイトには、最終桁と数字の符号が含まれます。ゾーン 10 進データは、出力可能な表現を生成します。

ニブル
: 0.5 バイトです。

データの種類

パック 10 進データ

パック 10 進表現は、10 進数の桁をバイトの"ニブル"に保存します。バイトは 2 つのニブルを含み、ニブルは 1 桁の 16 進数によって示されます。たとえば、値 15 は、16 進数の 1 と 5 を使用して 2 つのニブルに保存されます。

符号の表示は動作環境によって異なります。IBM メインフレームでは、符号は最後のニブルによって示されます。出力形式の使用時、C は正の値、D は負の値を示します。入力形式の使用時、A、C、E および F は正の値、B と D は負の値を示します。

他のニブルは、符号付きパック 10 進データに使用できません。他のすべての動作環境では、符号はその独自のバイトで示されます。高位ビットが 1 の場合、数字は負になります。他の場合、数字は正になります。

次の事柄がパック 10 進データ表現に当てはまります。

- すべてのプラットフォーム上で S370FPD 出力形式を使用して、IBM メインフレーム形式の 10 進データを取得できます。
- 符号インジケータのない符号なしパックデータを使用できます。パック 10 進の出力形式および入力形式により、パック 10 進データ表現が処理されます。ASCII プラットフォームでも、EBCDIC プラットフォームでも同じです。
- S370FPDU の出力形式および入力形式では最後のニブルに F が必要ですが、パック 10 進では符号のニブルは必要ありません。

ゾーン 10 進データ

次の事柄がゾーン 10 進データ表現に当てはまります。

- ゾーン 10 進表現は、10 進数の桁をバイトの下位ニブルに保存します。符号を含まないバイトのすべてにおいて、上位ニブルは数値ゾーンニブル(EBCDIC では F、ASCII では 3)になります。
- 符号はその表現によって、桁を含むバイトに結合することも、切り離すことも可能です。ただし、標準ゾーン 10 進の出力形式および入力形式では、符号が最後のバイトに結合されます。
- EBCDIC および ASCII のゾーン 10 進出力形式は、同じ出力可能な表現で数字を生成します。バイトは 2 つのニブルを含み、それぞれが 1 桁の 16 進数によって示されます。たとえば、値 15 は 2 バイトで保存されます。1 番目のバイトには 16 進値の F1 が、2 番目のバイトには 16 進値の C5 がそれぞれ含まれます。

パックユリウス暦の日付

次の事柄がパックユリウス暦の日付に当てはまります。

- パック 10 進表現のユリウス暦の日付を処理する 2 つの出力形式と入力形式は、PDJULI と PDJULG です。PDJULI は IBM メインフレームの年計算を使用し、PDJULG はグレゴリオ暦の計算を使用します。
- IBM メインフレームの計算では、1900 を基準年とし、データの年値は 1900 からのオフセットを示します。たとえば、98 は 1998、100 は 2000、102 は 2002 をそれぞれ表します。また、1998 は 3898 を表します。
- グレゴリオ暦の計算では、2 桁または 4 桁の年を利用できます。2 桁の年を使用する場合、YEARCUTOFF=システムオプションの設定を使用して実際の年を決定します。

パック 10 進データとゾーン 10 進データをサポートするプラットフォーム

一部のプラットフォームには、パックおよびゾーン 10 進データをサポートするためのネイティブ命令が含まれています。他のプラットフォームでは、ソフトウェアを使用して計算をエミュレートする必要があります。たとえば、IBM メインフレームにはパック 10 進データを追加するためのパック追加命令が含まれていますが、Intel 基準のプラットフォームにはそのような命令は含まれていないため、10 進データを他の形式に変換する必要があります。

パック 10 進データとゾーン 10 進データをサポートする言語

複数の言語で、パック 10 進データとゾーン 10 進データがサポートされています。次の表は、COBOL picture 句に対応する SAS 出力形式および入力形式を示したものです。

IBM VS COBOL II 句	対応する S370Fxxx 出力形式および入力形式
PIC S9(X) PACKED-DECIMAL	S370FPDw.
PIC 9(X) PACKED-DECIMAL	S370FPDUw.
PIC S9(W) DISPLAY	S370FZDw.
PIC 9(W) DISPLAY	S370FZDUw.
PIC S9(W) DISPLAY SIGN LEADING	S370FZDLw.
PIC S9(W) DISPLAY SIGN LEADING SEPARATE	S370FZDSw.
PIC S9(W) DISPLAY SIGN TRAILING SEPARATE	S370FZDTw.

表内のパック 10 進表現では、X は表す桁数、W はバイト数を表します。PIC S9(X) PACKED-DECIMAL では、W は `ceil((x+1)/2)` になります。PIC 9(X) PACKED-DECIMAL では、W は `ceil(x/2)` になります。たとえば、PIC S9(5) PACKED-DECIMAL は 5 桁を表します。符号が含まれる場合、6 つのニブルが必要です。`ceil((5+1)/2)` の長さは 3 バイトであり、W の値は 3 になります。

PACKED-DECIMAL の代わりに COMP-3 を使用できます。

IBM アセンブリ言語では、P ディレクティブはパック 10 進を、Z ディレクティブはゾーン 10 進をそれぞれ示します。次はアセンブリ言語リストの抜粋であり、オフセット、値、DC ステートメントが示されています。

```
offset value (in hex) inst label directive

+000000 00001C 2 PEX1 DC PL3'1'
+000003 00001D 3 PEX2 DC PL3'-1'
+000006 F0F0C1 4 ZEX1 DC ZL3'1'
+000009 F0F0D1 5 ZEX2 DC ZL3'1'
```

PL/I では、FIXED DECIMAL 属性はパック 10 進データとともに使用されます。ゾーン 10 進データを表すには、PICTURE 指定を使用する必要があります。Fortran または C 言語では、10 進データの統一した表現はありません。

パック 10 進とゾーン 10 進の出力形式と入力形式の概要

SAS では、パック 10 進データとゾーン 10 進データを処理するために、一連の出力形式および入力形式を使用します。次の表は、これらの出力形式および入力形式のデータ表現の種類を示したものです。S370 で始まる出力形式および入力形式は、IBM メインフレーム表現を示します。

出力形式	データ表現の種類	対応する入力形式	コメント
PD	パック 10 進	PD	ローカル符号付きパック 10 進
PK	パック 10 進	PK	符号なしパック 10 進; 動作環境に固有でない
ZD	ゾーン 10 進	ZD	ローカルゾーン 10 進
なし	ゾーン 10 進	ZDB	EBCDIC の空白(x'40')を EBCDIC のゼロ(x'F0')に変換してから、ゾーン 10 進の入力形式に相当
なし	ゾーン 10 進	ZDV	IBM 形式でないゾーン 10 進表現
S370FPD	パック 10 進	S370FPD	最後のニブルが C (正)または D (負)
S370FPDU	パック 10 進	S370FPDU	最後のニブルは常に F (正)
S370FZD	ゾーン 10 進	S370FZD	最終バイトの上位ニブルに符号が含まれる: C (正)または D (負)
S370FZDU	ゾーン 10 進	S370FZDU	符号なし; 符号ニブルは常に F
S370FZDL	ゾーン 10 進	S370FZDL	先頭バイトに符号ニブルが含まれる(入力形式); 先頭に x'C0' (正)または x'D0' (負)を含む符号バイトが個別にある(出力形式)
S370FZDS	ゾーン 10 進	S370FZDS	先頭に符号- (x'60')または+ (x'4E')あり
S370FZDT	ゾーン 10 進	S370FZDT	末尾に符号- (x'60')または+ (x'4E')あり
PDJULI	パック 10 進	PDJULI	パック表現のユリウス暦の日付 - IBM 計算
PDJULG	パック 10 進	PDJULG	パック表現のユリウス暦の日付 - クレゴリオ暦計算
なし	パック 10 進	RMFDUR	入力レイアウト: *mmssttt*F
なし	パック 10 進	SHRSTAMP	入力レイアウト: *yyyydddd*F*hhmmssth*。*yyyydddd*F はパックユリウス暦の日付; *yyyy* は 1900 を基準年とする年。

出力形式	データ表現の種類	対応する入力形式	コメント
なし	パック 10 進	SMFSTAMP	入力レイアウト: *xxxxxxxxyyyydddF*。*yyyydddF* はパックユリウス履歴の日付; *yyyy* は 1900 を基準年とする年。
なし	パック 10 進	PDTIME	入力レイアウト: 0*hhmmss*F
なし	パック 10 進	RMFSTAMP	入力レイアウト: 0*hhmmss*F*yyyydddF*。*yyyydddF* はパックユリウス履歴の日付; yyyy は 1900 を基準年とする年。

ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み

ISO 8601 規格のフォーマッティングシンボル

次のリストは、ISO 8601 規格の日付値、時間値、日時値、期間値、間隔値の表記に使用されるフォーマッティングシンボルについて説明したものです。

n
年数、月数、日数を表す数字です。

P
年数、月数、日数、時間数、分数、秒数で示される期間が続くことを示します。

T
時間値が続くことを示します。時間を含む値は T で始まる必要があります。

要件 文字 E8601 で始まる拡張表記入力形式から読み込む時間値には、大文字の T を使用する必要があります。

W
期間が週数で示されることを示します。

Z
時間値がイギリスのグリニッジに対応した時間、つまり UTC 時間であることを示します。

+|-
+は、イギリスのグリニッジの東部タイムゾーンのオフセットを示します。-は、イギリスのグリニッジの西部タイムゾーンのオフセットを示します。

yyyy
4 桁の年を示します。

mm
日付の一部。2 桁の月、01-12 を示します。

dd
2 桁の日、01–31 を示します。

hh
2 桁の時間、00–24 を示します。

mm
時間の一部。2 桁の時間、00–59 を示します。

ss
2 桁の秒、00–59 を示します。

fff | ffffff
秒の端数 0–9 を示します(オプション)。

fff $N8601B 入力形式と$N8601E 入力形式では、小数点以下の桁数 1 - 3 の値を読み込みます

ffffff $N8601B 入力形式および$N8601E 入力形式以外の入力形式では、小数点以下の桁数 1 - 6 桁の値を読み込みます

Y
期間の年数値が次に続くことを示します。

M
日付の一部。期間の月数値が次に続くことを示します。

D
期間の日数値が次に続くことを示します。

H
期間の時間値が次に続くことを示します。

M
時間の一部。期間の分数値が次に続くことを示します。

S
期間の秒数値が次に続くことを示します。

ISO 8601 規格の日付値、時間値、日時値の読み込み

さまざまな入力形式を使用して ISO 8601 日付、時間、日時を読み込み、SAS 日付値、時間値、または日時値を生成します。次の表は、各種の日付、時間、日時の形式と、それらの読み込みに使用する入力形式を示したものです。

日付、時間、日時	ISO 8601 表記	例	入力形式
基本表記			
日付	YYYYMMDD	20120915	B8601DA*w.*
時間	hhmmssnnnnnn	155300322348	B8601TM*w.d*
タイムゾーン付き時間	hhmmss+|-hhmm	155300+0500	B8601TZ*w.d*
	hhmmssZ	155300Z	B8601TZ*w.d*
タイムゾーン付きローカル時間への変換	hhmmss+|-hhmm	155300+0500	B8601TZ*w.d*
日時	YYYYMMDDThhmmssnnnnnn	20120915T155300	B8601DT*w.d*

日付、時間、日時	ISO 8601 表記	例	入力形式
タイムゾーン付き日時	YYYYMMDDThhmmss +|-hhmm	20120915T155300+0 500	B8601DZ*w.d*
	YYYYMMDDThhmmssZ	20120915T155300Z	B8601DZ*w.d*
日時からの日付の読み込み	YYYYMMD	20120915	B8601DN*w.*
拡張表記			
日付	YYYY-MM-DD	2012-09-15	E8601DA*w.*
時間	hh:mm:ss.nnnnnn	15:53:00.322348	E8601TM*w.d*
タイムゾーン付き時間	hh:mm:ss.nnnnnn+|-hh:mm	15:53:00+05:00	E8601TZ*w.d*
タイムゾーン付きローカル時間への変換	hh:mm:ss.nnnnnn+|-hh:mm	15:53:00+05:00	E8601LZ*w.d*
日時	YYYY-MM-DDThh:mm:ss.nnnnnn	2012-09-15T15:53:0 0	E8601DT*w.d*
タイムゾーン付き日時	YYYY-MM-DDThh:mm:ss.nnnnnn+|-hh:mm	2012-09-15T15:53:0 0+05:00	E8601DZ*w.d*
日時からの日付の読み込み	YYYY-MM-DD	2012-09-15	E8601DN*w.*

タイムゾーンオフセット(+|-*hh*:*mm* または+|-*hhmm*)が指定された ISO 8601 値が読み込まれると、時間値または日時値がオフセットに従って調整されます。タイムゾーンオフセット付きの ISO 8601 値に対応する SAS 時間値または日時値は、基準子午線(イギリスのグリニッジ)の時間または日時になります。たとえば、日時 2011–09–15T15:53:00+05:00 が E8601DZ 入力形式を使用して読み込まれた場合、日時値は 5 時間のタイムゾーンの差が調整されて 1631703180 になります。この日時値は基準子午線の日時値です。この値を E8601DZ 出力形式を使用して書き出した場合、値は 2011–09–15T10:53:00+00:00 です。T の後には、5 時間の調整された時間値が表示されます。

ISO 8601 規格の期間値、間隔値、日時値の読み込み

期間値、間隔値、日時値を読み込む入力形式

SAS では、ISO 規格の日時値、期間値、間隔値を読み込む 2 つの入力形式を使用します。

$N8601B 入力形式
: 基本表記、拡張表記のいずれかで指定される期間値、間隔値、日時値を読み込みます。

$N8601E 入力形式
: 拡張表記でのみ指定される期間値、間隔値、日時値を読み込みます。

拡張表記の準拠を確保したい場合は、$N8601E 入力形式を使用します。

これらの入力形式によって読み込まれる日時値は、SAS 文字表現になります。日時値を数値として読み込む場合、B8601DT 入力形式、B8601DZ 入力形式、E8601DT 入力形式、または E8601DZ 入力形式を使用します。

省略されていない期間、日時、間隔の表記

次の表は、要素が省略されていない形式で読み込み可能な期間値、日時値、間隔値の形式を示したものです。

表 3.2 構成要素をすべて含む形式

時間要素	ISO 8601 表記	例
期間 - 基本表記	PYYYYMMDDThhmmss	P20120915T155300
	-PYYYYMMDDThhmmss	-P20080915T155300
期間 - 拡張表記	PYYYY-MM-DDThh:mm:ss	P2012-09-15T15:53:00
	-PYYYY-MM-DDThh:mm:ss	-P2012-09-15T15:53:00
期間 - 基本表記と拡張表記	PnYnMnDTnHnMnS	P2y10m14dT20h13m45s
	-PnYnMnDTnHnMnS	-P2n10m14dT20h13m45s
	PnW (週)	P6w
間隔 - 基本表記	YYYYMMDDThhmmss/YYYYMMDDThhmmss	20120915T155300/20141113T000000
	PnYnMnDTnHnMnS/YYYYMMDDThhmmss	P2y10M14dT20h13m45s/20120915T155300
	YYYYMMDDThhmmss/PnYnMnDTnHnMnS	20120915T155300/P2y10M14dT20h13m45s
間隔 - 拡張表記	YYYY-MM-DDThh:mm:ss/YYYY-MM-DDThh:mm:ss	2012-09-15T15:53:00/2014-11-13T00:00:00
	PnYnMnDTnHnMnS/YYYY-MM-DDThh:mm:ss	P2y10M14dT20h13m45s/2012-09-15T15:53:00
	YYYY-MM-DDThh:mm:ss/PnYnMnDTnHnMnS	2012-09-15T15:53:00/P2y10M14dT20h13m45s
日時 - 基本表記	YYYYMMDDThhmmss.fff+|-hhmm	20120915T155300
	(すべて空白)	
日時 - 拡張表記	YYYY-MM-DDThh:mm:ss.fff+|-hhmm	2012-09-15T15:53:00 +04:30
	(すべて空白)	

省略要素の読み込み

1 つ以上の日付要素または時間要素を、P*yyyymmdd* 形式の日時値または期間値から省略できます。省略要素は$N8601B 入力形式または $N8601E 入力形式を使用して読み込みます。また、省略要素は、ハイフン(-)で表す必要があります。

次は、省略要素を含む期間値、日時値、間隔値の例です。

p0003-02--T10:31:33
省略要素は日数です。

-p0003-02-02T-:31:33
省略要素は時間数です。

x-09-15T15:x:x
省略要素は、年数、分数、秒数です。

2012-09-15T15:x:00/2010-09-15T15:x:00
省略要素は分です。

タイムゾーンオフセットを含む値を読み込む場合、省略要素は許可されません。省略要素の代わりに 00 を使用します。

切り捨て値

SAS では、値が 0 または有効でないため 1 つ以上の下位要素が切り捨てられた、切り捨て期間値、日時値、間隔値を読み込みます。

次のリストは、切り捨て値の例を表したものです。

- **p00030202T1031**
- **2012-09-15T15/2014-09-15T15:53**
- **-p0003-03-03T-:-:-**
- **P2y3m4dT5h6m**
- **2012-09-xTx:x:x**
- **2012**

タイムゾーンオフセットを含む値を読み込む場合、切り捨ては許可されません。切り捨て値の代わりに 00 を使用します。

期間構成要素の標準化

期間構成要素の値が構成要素の最大標準値を超えると、構成要素はその期間構成要素が単一の構成要素である場合を除いて標準化されます。次の表は、標準化された期間構成要素の例を示したものです。

期間	標準化された期間(拡張表記)
p3y13m	p0004-01
pt24h24m65s	P----01T-:25:05
p3y13mT24h61m	P0004-01-01T01:01
p0004-13	p0005-01
p0003-02-61T15:61:61	P0003-04-01T16:02:01
p13m	P13M

構成要素に最大値(分や秒の場合は 60)が含まれている場合、その値は標準化され、ハイフンと置き換えられます。たとえば、**pT12:60:13** は **PT13:-:13** になります。

30 日で月が標準化されます。

日時値の日付と時間が構成要素の標準値を超える場合は、標準化されません。この場合、エラーが発生します。

期間値、日時値、間隔値の端数

終わりの構成要素には、1 つのピリオドまたはカンマとそれに続く 1 桁から 3 桁までの小数点以下の桁数からなる端数を含めることができます。次の例は、期間値、日付値、間隔値での端数の使用を表したものです。

- **201209.5**
- **P2012-09-15T10.33**
- **2012-09-15/P0003-03-03,333**

4 章

入力形式のディクショナリ

他のドキュメントで説明されている入力形式

各国語サポートの入力形式については、Chapter 12, “Dictionary of Informats for NLS,” in *SAS National Language Support (NLS): Reference Guide* を参照してください。

カテゴリ別の入力形式

次の 5 つのカテゴリの入力形式があります。

カテゴリ	説明
文字	文字変数に文字データ値を読み込むように指示します。
カラムバイナリ	文字変数または数値変数にカラムバイナリ形式またはマルチパンチ形式で保存されているデータを読み込むように指示します。
日付と時間	日付、時間および日時を表す変数に日付値を読み込むように指示します。
ISO 8601	数値変数または文字変数に ISO 8601 規格で書かれた日付値、時間値、日時値を読み込むように指示します。
数値	数値変数に数値データ値を読み込むように指示します。

カラムバイナリデータの詳細については、“Reading Column-Binary Data” in Chapter 19 of *SAS Language Reference: Concepts* を参照してください。ユーザー定義の入力形式作成の詳細については、Chapter 27, “FORMAT Procedure” in *Base SAS Procedures Guide* を参照してください。.

次の表に、SAS 入力形式の概要を示します。詳細については、各入力形式のディクショナリエントリを参照してください。

カテゴリ	言語要素	説明
ISO 8601	$N8601Bw.d 入力形式 (p. 226)	基本表記または拡張表記で示される ISO 8601 規格の期間値、日時値、間隔値の完全形式、切り捨て形式、省略形式を読み込みます。

カテゴリ	言語要素	説明
	$N8601Ew.d 入力形式 (p. 228)	拡張表記で示される ISO 8601 規格の期間、日時、間隔の値を読み込みます。
	B8601CIw.d 入力形式 (p. 243)	世紀マーカーを含む、cyymmddhhmmss<fff>形式の IBM 日時値を読み込みます。
	B8601DAw. 入力形式 (p. 245)	ISO 8601 規格の基本表記 yyyymmdd を使用して示される日付値を読み込みます。
	B8601DJw.d 入力形式 (p. 246)	yyyymmddhhmmss<ffffff>形式の Java 日時値を読み込みます。
	B8601DNw. 入力形式 (p. 247)	ISO 8601 規格の基本表記 yyyymmdd を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。
	B8601DTw.d 入力形式 (p. 248)	ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して示される日時値を読み込みます。
	B8601DZw.d 入力形式 (p. 250)	ISO 8601 規格の日時基本表記 yyyymmddThhmmss+|-hhmm または yyyymmddThhmmss<ffffff>Z を使用して示される協定世界時(UTC)の日時値を読み込みます。
	B8601TMw.d 入力形式 (p. 251)	ISO 8601 規格の基本表記 hhmmss<ffffff>を使用して示される時間値を読み込みます
	B8601TZw.d 入力形式 (p. 252)	ISO 8601 規格の基本時間表記 hhmmss<ffff>+|–hhmm または hhmmss<ffffff>Z で示される時間値を読み込みます。
	E8601DAw. 入力形式 (p. 265)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して示される日付値を読み込みます。
	E8601DNw. 入力形式 (p. 266)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。
	E8601DTw.d 入力形式 (p. 267)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.<ffffff>を使用して示される日時値を読み込みます。
	E8601DZw.d 入力形式 (p. 269)	ISO 8601 規格の日時拡張表記 yyyy-mm-ddThh:mm:ss+|-hh:mm.<fffff>または yyyy-mm-ddThh:mm:ss.<fffff>Z を使用して示される、協定世界時(UTC)の日時値を読み込みます。
	E8601LZw.d 入力形式 (p. 270)	ISO 8601 規格の拡張表記 hh:mm:ss+|–hh:mm.<fffff>または hh:mm:ss.<fffff>Z を使用して示される協定世界時(UTC)値を読み込み、ローカル時間に変換します。
	E8601TMw.d 入力形式 (p. 272)	ISO 8601 規格の拡張表記 hh:mm:ss.<ffffff>を使用して示される時間値を読み込みます。
	E8601TZw.d 入力形式 (p. 273)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|–hh:mm.<ffffff>または hh:mm:ssZ を使用して示される時間値を読み込みます。
カラムバイナリ	$CBw. 入力形式 (p. 221)	カラムバイナリファイルから標準文字データを読み込みます。

カテゴリ	言語要素	説明
	CBw.d 入力形式 (p. 257)	カラムバイナリファイルから標準数値を読み込みます。
	PUNCH.d 入力形式 (p. 300)	カラムバイナリデータ行のパンチの有無を読み込みます。
	ROWw.d 入力形式 (p. 305)	カードカラムに記録されているカラムバイナリフィールドを読み込みます。
数値	BINARYw.d 入力形式 (p. 254)	正のバイナリ値を整数に変換します。
	BITSw.d 入力形式 (p. 255)	ビットを抽出します。
	BZw.d 入力形式 (p. 256)	空白をゼロに変換します。
	COMMAw.d 入力形式 (p. 258)	埋め込み文字を取り除きます。
	COMMAXw.d 入力形式 (p. 259)	埋め込まれているピリオド、空白、ドル記号、パーセント記号、ダッシュ、閉じかっこを入力データから取り除きます。フィールドの先頭にある開きかっこはマイナス記号に変換されます。COMMAX 入力形式では、小数点とカンマの役割が逆になっています。
	Ew.d 入力形式 (p. 264)	指数表記および倍精度の指数表記で保存されている数値を読み込みます。
	FLOATw.d 入力形式 (p. 275)	正の単精度浮動小数点値を読み込み、その値を 10 の d 乗で除算します。
	HEXw. 入力形式 (p. 276)	16 進の正のバイナリ値を整数(固定小数点)または実数(浮動小数点)のバイナリ値に変換します。
	IBw.d 入力形式 (p. 278)	負の値を含む、ネイティブのバイナリ整数(固定小数点)値 を読み込みます。
	IBRw.d 入力形式 (p. 280)	Intel および DEC 形式のバイナリ整数(固定小数点)値を読み込みます。
	IEEEw.d 入力形式 (p. 281)	IEEE 形式の浮動小数点値を読み込み、その値を 10 の d 乗で除算します。
	NUMXw.d 入力形式 (p. 289)	小数点をカンマとして数値を読み込みます。
	OCTALw.d 入力形式 (p. 290)	正の 8 進値を整数に変換します。
	PDw.d 入力形式 (p. 290)	IBM のパック 10 進形式で保存されているデータを読み込みます。
	PERCENTw.d 入力形式 (p. 296)	百分率を数値として読み込みます。
	PIBw.d 入力形式 (p. 296)	正のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ	言語要素	説明
	PIBRw.d 入力形式 (p. 298)	Intel 形式と DEC 形式の正のバイナリ整数(固定小数点)値を読み込みます。
	PKw.d 入力形式 (p. 299)	符号なしパック 10 進データを読み込みます。
	RBw.d 入力形式 (p. 301)	バイナリ実数(浮動小数点)表記で格納されている数値データを読み込みます。
	S370FFw.d 入力形式 (p. 306)	EBCDIC 形式の数値データを読み込みます。
	S370FIBw.d 入力形式 (p. 307)	負の値を含む、IBM メインフレーム形式のバイナリ整数(固定小数点)値を読み込みます。
	S370FIBUw.d 入力形式 (p. 309)	IBM メインフレーム形式の符号なしバイナリ整数(固定小数点)値を読み込みます。
	S370FPDw.d 入力形式 (p. 310)	IBM メインフレーム形式のパック 10 進データを読み込みます。
	S370FPDUw.d 入力形式 (p. 311)	IBM メインフレーム形式の符号なしパック 10 進データを読み込みます。
	S370FPIBw.d 入力形式 (p. 312)	IBM メインフレーム形式の正のバイナリ整数(固定小数点)値を読み込みます。
	S370FRBw.d 入力形式 (p. 313)	IBM メインフレーム形式のバイナリ実数(浮動小数点)を読み込みます。
	S370FZDw.d 入力形式 (p. 314)	IBM メインフレーム形式のゾーン 10 進データを読み込みます。
	S370FZDBw.d 入力形式 (p. 316)	空白を含むゾーン 10 進データを読み込みます。
	S370FZDLw.d 入力形式 (p. 317)	IBM メインフレーム形式の前符号付きゾーン 10 進データを読み込みます。
	S370FZDSw.d 入力形式 (p. 318)	IBM メインフレーム形式の分離した前符号付きゾーン 10 進データを読み込みます。
	S370FZDTw.d 入力形式 (p. 319)	IBM メインフレーム形式の分離した後符号付きゾーン 10 進データを読み込みます。
	S370FZDUw.d 入力形式 (p. 320)	IBM メインフレーム形式の符号なしゾーン 10 進データを読み込みます。
	TRAILSGNw. 入力形式 (p. 326)	後置のプラス符号(+)とマイナス符号(–)を読み込みます。
	VAXRBw.d 入力形式 (p. 328)	VMS 形式のバイナリ実数(浮動小数点)データを読み込みます。
	VMSZNw.d 入力形式 (p. 328)	VMS および MicroFocus COBOL ゾーン数値データを読み込みます。

カテゴリ	言語要素	説明
	w.d 入力形式 (p. 330)	標準数値データを読み込みます。
	ZDw.d 入力形式 (p. 342)	ゾーン 10 進データを読み込みます。
	ZDBw.d 入力形式 (p. 344)	空白を含むゾーン 10 進データを読み込みます。
	ZDVw.d 入力形式 (p. 344)	ゾーン 10 進データを読み込んで検証します。
日付と時間	$N8601Bw.d 入力形式 (p. 226)	基本表記または拡張表記で示される ISO 8601 規格の期間値、日時値、間隔値の完全形式、切り捨て形式、省略形式を読み込みます。
	$N8601Ew.d 入力形式 (p. 228)	拡張表記で示される ISO 8601 規格の期間、日時、間隔の値を読み込みます。
	ANYDTDTEw. 入力形式 (p. 236)	さまざまな日付形式、時間形式、および日時形式から日付値を読み込み、抽出します。
	ANYDTDTMw. 入力形式 (p. 238)	さまざまな日付形式、時間形式、および日時形式から日時値を読み込み、抽出します。
	ANYDTTMEw. 入力形式 (p. 241)	さまざまな日付形式、時間形式、および日時形式から時間値を読み込み、抽出します。
	B8601CIw.d 入力形式 (p. 243)	世紀マーカーを含む、cyymmddhhmmss<fff>形式の IBM 日時値を読み込みます。
	B8601DAw. 入力形式 (p. 245)	ISO 8601 規格の基本表記 yyyymmdd を使用して示される日付値を読み込みます。
	B8601DJw.d 入力形式 (p. 246)	yyyymmddhhmmss<ffffff>形式の Java 日時値を読み込みます。
	B8601DNw. 入力形式 (p. 247)	ISO 8601 規格の基本表記 yyyymmdd を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。
	B8601DTw.d 入力形式 (p. 248)	ISO 8601 規格の基本表記 yyyymmddThhmmss<ffffff>を使用して示される日時値を読み込みます。
	B8601DZw.d 入力形式 (p. 250)	ISO 8601 規格の日時基本表記 yyyymmddThhmmss+|-hhmm または yyyymmddThhmmss<ffffff>Z を使用して示される協定世界時(UTC)の日時値を読み込みます。
	B8601TMw.d 入力形式 (p. 251)	ISO 8601 規格の基本表記 hhmmss<ffffff>を使用して示される時間値を読み込みます
	B8601TZw.d 入力形式 (p. 252)	ISO 8601 規格の基本時間表記 hhmmss<ffff>+|–hhmm または hhmmss<ffffff>Z で示される時間値を読み込みます。
	DATEw. 入力形式 (p. 260)	ddmmmyy または ddmmmyyyy 形式の日付値を読み込みます。
	DATETIMEw. 入力形式 (p. 261)	ddmmmyy hh:mm:ss.ss または ddmmmyyyy hh:mm:ss.ss 形式の日時値を読み込みます。

カテゴリ	言語要素	説明
	DDMMYYw. 入力形式 (p. 263)	ddmmyy<yy>または dd-mm-yy<yy>形式の日付値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)などの特殊文字で日、月、年を区切ります。年は 2 桁または 4 桁のいずれかになります。
	E8601DAw. 入力形式 (p. 265)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して示される日付値を読み込みます。
	E8601DNw. 入力形式 (p. 266)	ISO 8601 規格の拡張表記 yyyy-mm-dd を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。
	E8601DTw.d 入力形式 (p. 267)	ISO 8601 規格の拡張表記 yyyy-mm-ddThh:mm:ss.<ffffff>を使用して示される日時値を読み込みます。
	E8601DZw.d 入力形式 (p. 269)	ISO 8601 規格の日時拡張表記 yyyy-mm-ddThh:mm:ss+|-hh:mm.<fffff>または yyyy-mm-ddThh:mm:ss.<fffff>Z を使用して示される、協定世界時(UTC)の日時値を読み込みます。
	E8601LZw.d 入力形式 (p. 270)	ISO 8601 規格の拡張表記 hh:mm:ss+|–hh:mm.<fffff>または hh:mm:ss.<fffff>Z を使用して示される協定世界時(UTC)値を読み込み、ローカル時間に変換します。
	E8601TMw.d 入力形式 (p. 272)	ISO 8601 規格の拡張表記 hh:mm:ss.<ffffff>を使用して示される時間値を読み込みます。
	E8601TZw.d 入力形式 (p. 273)	ISO 8601 規格の拡張時間表記 hh:mm:ss+|–hh:mm.<ffffff>または hh:mm:ssZ を使用して示される時間値を読み込みます。
	HHMMSSw. 入力形式 (p. 277)	形式 hh:mm:ss または hhmmss.の時、分、秒を読み取ります。
	JULIANw. 入力形式 (p. 282)	yyddd または yyyyddd 形式のユリウス日付を読み込みます。
	MDYAMPMw.d 入力形式 (p. 283)	mm-dd-yy<yy> hh:mm:ss.ss AM|PM 形式の日時値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)またはコロン(:)などの特殊文字で月、日、年は区切られます。年は 2 桁または 4 桁のいずれかになります。
	MMDDYYw. 入力形式 (p. 285)	mmddyy または mmddyyyy 形式の日付値を読み込みます。
	MONYYw. 入力形式 (p. 286)	mmmyy または mmmyyyy 形式の年月を読み込みます。
	MSECw. 入力形式 (p. 288)	TIME MIC の値を読み込みます。
	PDJULGw. 入力形式 (p. 292)	IBM で使用する 16 進の yyyydddF 形式のパックユリウス日付値を読み込みます。
	PDJULIw. 入力形式 (p. 293)	IBM で使用する 16 進の ccyydddF 形式のパックユリウス日付を読み込みます。
	PDTIMEw. 入力形式 (p. 295)	SMF レコードと RMF レコードのパック 10 進の時間を読み込みます。

カテゴリ	言語要素	説明
	RMFDURw. 入力形式 (p. 302)	RMF レコードの継続間隔を読み込みます。
	RMFSTAMPw. 入力形式 (p. 304)	RMF レコードの日付フィールドと時間フィールドを読み込みます。
	SHRSTAMPw. 入力形式 (p. 321)	SHR レコードの日付値と時間値を読み込みます。
	SMFSTAMPw. 入力形式 (p. 322)	SMF レコードの日時値を読み込みます。
	STIMERw. 入力形式 (p. 323)	時間値を読み込み、読み込んだ値が時間、分、秒であるかを識別します。STIMER システムオプションの出力を読み込みます。
	TIMEw. 入力形式 (p. 324)	hh:mm:ss.ss 形式の時、分、秒を読み込みます。コロン(:)やピリオド(.)などの特殊文字が時、分、秒を区切るために使用されます。
	TODSTAMPw. 入力形式 (p. 325)	8 バイトの TOD スタンプを読み込みます。
	TUw. 入力形式 (p. 327)	タイマーユニットを読み込みます。
	WEEKUw. 入力形式 (p. 331)	週番号が使用される形式の値を読み込み、U アルゴリズムを使用して SAS 日付値を返します。
	WEEKVw. 入力形式 (p. 333)	週番号が使用される形式の値を読み込み、V アルゴリズムを使用して SAS 日付値を返します。
	WEEKWw. 入力形式 (p. 334)	週番号が使用される形式の値を読み込み、W アルゴリズムを使用して SAS 日付値を返します。
	YMDDTTMw.d 入力形式 (p. 336)	<yy>yy-mm-dd hh:mm:ss.ss 形式の日時値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)またはコロン(:)などの特殊文字で年、月、日、時間、分、秒が区切られます。年は 2 桁または 4 桁のいずれかになります。
	YYMMDDw. 入力形式 (p. 338)	yymmdd または yyyymmdd 形式の日付値を読み込みます。
	YYMMNw. 入力形式 (p. 339)	yyyymm または yymm 形式の日付値を読み込みます。
	YYQw. 入力形式 (p. 341)	yyQq または yyyyQq 形式の年の四半期を読み込みます。
文字	$ASCIIw. 入力形式 (p. 218)	ASCII 文字データをネイティブな形式に変換します。
	$BASE64Xw. 入力形式 (p. 219)	Base 64 エンコーディングを使用して、ASCII テキストを文字データに変換します。
	$BINARYw. 入力形式 (p. 220)	バイナリデータを文字データに変換します。
	$CHARw. 入力形式 (p. 222)	空白を含む文字データを読み込みます。

カテゴリ	言語要素	説明
	$CHARZBw. 入力形式 (p. 223)	2 進ゼロを空白に変換します。
	$EBCDICw. 入力形式 (p. 224)	EBCDIC 文字データをネイティブな形式に変換します。
	$HEXw. 入力形式 (p. 225)	16 進データを文字データに変換します。
	$OCTALw. 入力形式 (p. 230)	8 進データを文字データに変換します。
	$PHEXw. 入力形式 (p. 231)	パック 16 進データを文字データに変換します。
	$QUOTEw. 入力形式 (p. 232)	対になっている引用符を文字データから削除します。
	$UPCASEw. 入力形式 (p. 233)	文字データを大文字に変換します。
	$VARYINGw. 入力形式 (p. 233)	可変長の文字データを読み込みます。
	$w. 入力形式 (p. 235)	標準文字データを読み込みます。

ディクショナリ

$ASCIIw. 入力形式

ASCII 文字データをネイティブな形式に変換します。

カテゴリ: 文字

構文

$ASCII*w*.

説明

w
入力幅を指定します。

デフォルト: 変数の長さが定義されていなければ 1。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

ASCII がネイティブな形式の場合、変換は実行されません。

比較

- IBM メインフレームシステム上では、$ASCII*w*.は ASCII データを EBCDIC に変換します。
- その他すべてのシステム上では、$ASCII*w*.は、デフォルト長が異なる以外は $CHAR*w*.入力形式と同じ結果になります。

例

```
input @1 name $ascii3.;
```

データ行	結果*	
----+----1	EBCDIC	ASCII
abc	818283	616263
ABC	C1C2C3	414243
();	4D5D5E	28293B

* 上記の結果は、文字のコード値を 16 進表現で出力したものです。16 進の 2 文字がバイナリデータの 1 バイトに対応します。1 バイトが 1 文字値に対応します。

$BASE64X*w*. 入力形式

Base 64 エンコーディングを使用して、ASCII テキストを文字データに変換します。

カテゴリ: 文字

配置: 左

構文

$BASE64X*w*.

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–32767

詳細

Base 64 は、ポジション指定スキームに基づき、ASCII 文字のみ使用したエンコード文字を指定するエンコーディング方式です。複数の Base 64 エンコーディングスキームが、電子メールやコンテンツマスキングなどの特定の用途に定義されています。ポジション 0 - 61 を文字 A - Z、a - z、0 - 9 にマップします。ポジション 62 は文字+にマップし、ポジション 63 は文字/にマップします。

次に、Base 64 エンコーディングの使用例を示します。

- XML ファイルの埋め込みバイナリデータ

- パスワードのエンコード
- URL のエンコード

エンコード結果の'='文字は、結果にゼロビットが埋め込まれたことを示します。エンコード文字をデコードするには、'='をデコードする値に含める必要があります。

例

```
input @1 b64exmpl $base64x64.;
```

データ行	結果
RkNBMDFBNzk5M0JD	FCA01A7993BC
TXlQYXNzd29yZA==	MyPassword
d3d3Lm15ZG9tYWluLmNvbi9teWhpZGRlblVSTA==	www.mydomain.com/myhiddenURL

関連項目:

- “LIBNAME Statement Syntax” in *SAS XML LIBNAME Engine: User's Guide* の MLDOUBLE オプション

出力形式:

- “$BASE64Xw. 出力形式” (32 ページ)

$BINARY*w*. 入力形式

バイナリデータを文字データに変換します。

カテゴリ: 文字

構文

$BINARY*w*.

説明

w

入力幅を指定します。8 ビットのバイナリ情報で 1 文字を表すため、$BINARY*w*.が読み込む 8 ビットごとに入力が文字値に変換され、変数に保存されます。

w< 8 の場合、$BINARY*w*.は *w* の後に 0 が続いているとしてデータを読み込みます。そのため、$BINARY4.は文字 0101 を 01010000 として読み込み、EBCDIC の場合は`&`に、ASCII の場合は `P` に変換します。*w*> 8 かつ 8 の倍数でない場合、$BINARY*w*.はデータ変換の前に *w* 内で最大の 8 の倍数の幅まで読み込みます。

デフォルト: 8

範囲: 1–32767

詳細

$BINARY*w*.入力形式はバイナリデータを解釈しませんが、0 または 1 のみを含む文字列をバイナリ情報として変換します。そのため、入力には数字 1 と 0 のみを使用します。空白は挿入できません。$BINARY*w*.入力形式では、先頭と末尾の空白は無視されます。

印字不可文字のバイナリコード表現を読み込むには、特定の文字の ASCII 表現または EBCDIC 表現に対応する、0 および 1 からなる文字列を入力します。$BINARY*w*.入力形式はその文字列を対応する文字値に変換します。

比較

- BINARY*w*.入力形式は、0 または 1 のみを含む 8 文字の入力を 1 バイトの数値データの 2 進表現として読み込みます。
- $HEX*w*.入力形式は、文字データの ASCII 表現または EBCDIC 表現を 16 進表現で表したものを読み込みます。

例

```
input @1 name $binary16.;
```

データ行	結果	
----+----1----+----2	ASCII	EBCDIC
0100110001001101	LM	<(

$CB*w*. 入力形式

カラムバイナリファイルから標準文字データを読み込みます。

カテゴリ: カラムバイナリ

構文

$CB*w*.

説明

w
入力幅を指定します。
デフォルト: なし
範囲: 1–32767

詳細

カラムバイナリデータストレージは、80 項目を超えるデータを 1 つの"仮想"パンチカードに保存できるように、データを圧縮します。

$CB*w*.入力形式は、カラムバイナリファイルから標準文字データを読み込みます。各カードカラムは 2 バイトで表されます。$CB*w*.入力形式はデータを標準文字コードに変

換します。その組み合わせが無効なパンチコードの場合、空白が返され、自動変数 _ERROR_が 1 に設定されます。

例

```
input @1 name $cb2.;
```

データ行*	結果	
----+----1	EBCDIC	ASCII
200A	+	N

* データ行はカラムバイナリの 16 進表現です。サンプルデータの“仮想”パンチカードカラムでは、行 12、行 6、行 8 がパンチされています。2 進表現は、0010 0000 0000 1010 となります。

関連項目:

- “How to Read Column-Binary Data” in Chapter 19 of *SAS Language Reference: Concepts*

入力形式:

- “CBw.d 入力形式” (257 ページ)
- “PUNCH.d 入力形式” (300 ページ)
- “ROWw.d 入力形式” (305 ページ)

$CHAR*w*. 入力形式

空白を含む文字データを読み込みます。

カテゴリ: 文字

構文

$CHAR*w*.

説明

w
入力幅を指定します。

デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

$CHAR*w*.入力形式は、保存前に、前後の空白を切り捨てたり、入力データフィールドの 1 つのピリオドを値の空白に変換したりしません。DATA ステップ内の INFORMAT ステートメントまたは ATTRIB ステートメントに$CHAR*w*.を使用して、リスト入力を読み込む場合、先頭の空白を含め、データに埋め込まれている空白はフィールド区切り文字として解釈されます。

比較

- $CHAR*w*.入力形式は、$*w*.入力形式とほとんど同じです。ただし、$CHAR*w*. は先頭の空白を切り捨てず、入力データフィールドの 1 つのピリオドを空白に変換しませんが、$*w*.入力形式はそれらを行います。
- 次の表は、SAS 入力形式の$CHAR8.と他のプログラミング言語での表記方法を比較したものです。

言語	文字表記
SAS	$CHAR8.
IBM 370 アセンブラ	CL8
C	char [8]
COBOL	PIC x(8)
Fortran	A8
PL/I	CHAR(8)

例

```
input @1 name $char5.;
```

データ行	結果*
----+----1	
XYZ	XYZ##
XYZ	#XYZ#
.	##.##
X YZ	#X#YZ

* #文字は空白を表します。

$CHARZBw. 入力形式

2 進ゼロを空白に変換します。

カテゴリ: 文字

構文

$CHARZB*w*.

説明

w
入力幅を指定します。

デフォルト: 変数の長さが定義されていなければ 1。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

$CHARZB*w*.入力形式は、文字データの前後の空白を値の保存前に切り捨てません。

比較

$CHARZB*w*.入力形式は、$CHAR*w*.入力形式と同じですが、$CHARZB*w*.はバイナリゼロを含むバイトを空白文字に変換します。

例

```
input @1 name $charzb5.;
```

データ行*		結果**
EBCDIC	ASCII	
E7E8E90000	58595A0000	XYZ##
00E7E8E900	0058595A00	#XYZ#
00E700E8E9	005800595A	#X#YZ

* データ行は、文字コードを 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトが 1 文字値に対応します。

** #文字は空白を表します。

$EBCDIC*w*. 入力形式

EBCDIC 文字データをネイティブな形式に変換します。

カテゴリ: 文字

構文

$EBCDIC*w*.

説明

w
入力幅を指定します。

デフォルト: 変数の長さが定義されていなければ 1。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

詳細

EBCDIC がネイティブな形式の場合、変換は実行されません。

注: テキストファイルがローカルエンコーディング環境以外で作成された場合はいつでも、ENCODING=オプションに ASCII または EBCDIC 環境を指定する必要があります。EBCDIC テキストファイルを ASCII プラットフォームで読み込む場合、FILENAME ステートメントまたは INFILE ステートメントの ENCODING=オプションを指定することをお勧めします。だたし、DSD を使用し、FILENAME ステートメントまたは INFILE ステートメントの DLM=オプションまたは DLMSTR=オプションを使用する場合、ENCODING=オプションが必要条件となります。これらのオプションは、セッションエンコーディングで示された特定の文字(引用符、カンマ、空白など)が必要なためです。エンコーディング固有の入力形式は、バイナリファイルに使用するためのものです。つまり、文字フィールドと非文字フィールドの両方が含まれます。

比較

- IBM メインフレームシステム上では、$EBCDIC*w*.は$CHAR*w*.入力形式と同じ結果になります。
- 他のすべてのシステム上では、$EBCDIC*w*.は EBCDIC 形式のデータを ASCII 形式に変換します。

例

```
input @1 name $ebcdic3.
```

データ行	結果*	
`----+----1`	ASCII	EBCDIC
`qrs`	`717273`	`9899A2`
`QRS`	`515253`	`D8D9E2`
`+;>`	`2B3B3E`	`4E5E6E`

* 上記の結果は、文字のコード値を 16 進表現で出力したものです。16 進の 2 文字がバイナリデータの 1 バイトに対応します。1 バイトが 1 文字値に対応します。

$HEX*w*. 入力形式

16 進データを文字データに変換します。

カテゴリ: 文字

参照項目: “$HEXw. Informat: UNIX” in *SAS Companion for UNIX Environments*
“$HEXw. Informat: Windows” in *SAS Companion for Windows*

構文

$HEX*w*.

説明

w
16 進データの桁数を指定します。

w=1 の場合、$HEX*w*.は値の後に 16 進の 0 を埋め込みます。*w* が 1 より大きい奇数の場合、$HEX*w*.は *w*–1 桁の 16 進データを読み込みます。

デフォルト: 2

範囲: 1–32767

詳細

$HEX*w*.入力形式は、2 桁の 16 進データを 1 バイトの文字データに変換します。入力方法が印字可能な文字に限定される場合は、$HEX*w*.を使用して、16 進値を文字変数にエンコードします。

比較

HEX*w*.入力形式は、2 桁の 16 進データを一度に読み込み、それらを 1 バイトの数値データに変換します。

例

```
input @1 name $hex4.;
```

データ行	結果	
----+----1	ASCII	EBCDIC
6C6C	ll	%%

$N8601B*w.d* 入力形式

基本表記または拡張表記で示される ISO 8601 規格の期間値、日時値、間隔値の完全形式、切り捨て形式、省略形式を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601B*w.d*

説明

w
入力幅を指定します。

デフォルト: 50

範囲: 1–200

要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d

秒値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–3

詳細

$N8601B 入力形式は、ISO 8601 規格の期間、日時、間隔の値を次の基本表記の文字データとして読み込みます。

時間要素	ISO 8601 表記	例
期間	P*yyyy-mm-dd*T*hh:mm:ss.fff*	P2012-09-15T15:53:00
	P*yyyymmdd*T*hhmmss*	P00020304T050607
	P*n*Y*n*M*n*DT*n*H*n*M*n.fff*S	P2y10m14dT20h13m45.222s
	P*n*W	P6w
間隔	*yyyy-mm-dd*T*hh:mm:ss.fff*/ *yyyy-mm-dd*T*hh:mm:ss.fff*	2012-09-15T15:53:00/2014-11-13T00:00:00
	*yyyymmdd*T*hhmmss.fff*/ *yyyymmdd*T*hhmmss.fff*	20120915T155300/20141115T120000
	P*n*Y*n*M*n*DT*n*H*n*M*n.fff*S/ *yyyy-mm-dd*T*hh:mm:ss.fff*	P2y10M14dT20h13m45s/ 2012-09-15T15:53:00
	*yyyy-mm-dd*T*hh:mm:ss.fff*/ P*n*Y*n*M*n*DT*n*H*n*M*n.fff*S	2012-09-15T15:53:00/ P2y10M14dT20h13m45s
日時	*yyyy-mm-dd*T*hh:mm:ss.fff*	2012-09-15T15:53:00
	*yyyymmdd*T*hhmmss.fff*	20120915T155300

$N8601B 入力形式は、省略要素または切り捨て要素を含む ISO 8601 規格の期間要素、間隔要素、日時要素も読み込みます。省略要素は、1 つのハイフン(-)を使用して、その省略要素を示す必要があります。

比較

$N8601B 入力形式は、基本表記または拡張表記で示される期間、間隔、日時を読み込みます。

$N8601E 入力形式は、拡張表記でのみ示される期間、間隔、日時を読み込みます。拡張表記の準拠を確保したい場合は$N8601E 入力形式を使用します。

例

```
input @1 i860 $n8601b.;
```

データ行	結果
p0002-04-05t5:1:12	0002405050112FFC
2012-09-15T15:53:00/2010-09-15T00:00:00	2012915155300FFD2010915000000FFD
p0033-01-04T3:2:55/2012-09-15T15:53:00	0033104030255FFC2012915155300FFD

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

$N8601E*w.d* 入力形式

拡張表記で示される ISO 8601 規格の期間、日時、間隔の値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.4, complete representation

構文

$N8601E*w.d*

説明

w
入力幅を指定します。
デフォルト: 50
範囲: 1–200
要件 期間値または日時値の最小長は 16 です。間隔値の最小長は 16 です。

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–3

詳細

$N8601E 入力形式は、次の拡張表記で示される ISO 8601 規格の期間、間隔、日時の値を読み込みます。

時間要素	ISO 8601 表記	例
期間	P*yyyy-mm-dd*T*hh:mm:ss.fff*	P2012-09-15T15:53:00
	PnW	P6w

時間要素	ISO 8601 表記	例
間隔	*yyyy-mm-dd*T*hh:mm:ss.fff*/ *yyyy-mm-dd*T*hh:mm:ss.fff*	2012-09-15T15:53:00/2014-11-13T00:00:00
	*yyyy-mm-dd*T*hh:mm:ss.fff*/ P*n*Y*n*M*n*DT*n*H*n*M*n*s.*fff*S	2012-09-15T15:53:00/ P2Y10M14DT20H13M45S
日時	*yyyy-mm-dd*T*hh:mm:ss.fff*	2012-09-15T15:53:00

n
年数、月数、日数を表す数字です。

P
年数、月数、日数、時間数、分数、秒数で示される期間が続くことを示す文字です。

W
期間が週数で示されることを表す文字です。

T
時間値が続くことを示すために使用される文字です。すべての時間値が 0 の場合、T は不要です。

/
間隔の開始日時値と終了日時値を区切るために使用されます。

yyyy
4 桁の年を示します。

mm
01 から 12 までの 2 桁の月を示します。

dd
01 から 31 までの 2 桁の日を示します。

hh
00 から 23 までの 2 桁の時間を示します。

mm
00 から 59 までの 2 桁の分を示します。

ss
00 から 59 までの 2 桁の秒を示します。

fff
3 桁までの有効桁数を持つ、オプションの秒の端数を示します。各桁は 0 から 9 までです。

Y
期間の年数を示すために使用される文字です。

M
期間の月数を示すために使用される文字です。

D
期間の日数を指定するために使用される文字です。

H
期間の時間数を示すために使用される文字です。

M
期間の分数を指定するために使用される文字です。

S
期間の秒数を指定するために使用される文字です。

比較

$N8601E 入力形式は、拡張表記でのみ示される有効な期間、間隔、日時を読み込みます。

$N8601B 入力形式は、基本表記または拡張表記で示される有効な期間、間隔、日時を読み込みます。

拡張表記の準拠を確保したい場合は$N8601E 入力形式を使用します。

例

```
input @1 i860 $n8601e.;
```

データ行	結果
p0002-04-05t5:1:12s	0002405050112FFC
2012-09-15T15:53:00/2014-09-15T00:00:00	2012915155300FFD2014915000000FFD
p0033-01-04T3:2:55/2012-09-15T15:53:00	0033104030255FFC2012915155300FFD

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

$OCTAL*w*. 入力形式

8 進データを文字データに変換します。

カテゴリ: 文字

構文

$OCTAL*w*.

説明

w
入力幅をビットで指定します。1 桁の 8 進データは 3 ビットの 2 進情報を表すため、*w* の値を$OCTAL*w*.が読み込む 8 進データのカラムごとに 3 ずつ増分します。

デフォルト: 3

範囲: 1–32767

詳細

8 ビットの 2 進データは、1 桁の文字データのコードを表します。そのため、1 桁の文字データを表すために少なくとも 3 桁の 8 進データが必要です。ビットが追加されるこ

ともあります。$OCTAL*w*.は、3 桁の 8 進データを 1 桁の文字データとして処理します。追加ビットは無視します。

印字不可文字のバイナリコードを 8 進表現で表したものを読み込むには、$OCTAL*w*.を使用します。特定の文字の ASCII コードまたは EBCDIC コードを 8 進表記で入力します。次に、$OCTAL*w*.を使用して、文字データに変換します。

入力には、0 から 7 までの数字のみを使用します。空白は挿入できません。$OCTAL*w*.では、先頭と末尾の空白は無視されます。

比較

OCTAL*w*.入力形式は 8 進データを読み込み、数値データに変換します。

例

```
input @1 name $octal9.;
```

データ行	結果	
----+----1	EBCDIC	ASCII
114	<	L

$PHEX*w*. 入力形式

パック 16 進データを文字データに変換します。

カテゴリ: 文字

構文

$PHEX*w*.

説明

w
入力のバイト数を指定します。

$PHEX*w*.を使用してパック 16 進データを読み込む場合、変数の長さは *w* ではなく、その文字値の保存に必要なバイト数です。通常、$PHEX*w*.によって暗黙的に長さ定義される文字変数では、その長さは 2*w*–1 となります。

デフォルト: 2

範囲: 1–32767

詳細

パック 16 進データは、パック 10 進データに似ていますが、すべての 16 進文字が有効である点が異なります。パック 16 進データでは、下位ニブルの値には意味がありません。パック 10 進データでは、下位ニブルの値はデータが表す数値の符号を示します。$PHEX*w*.入力形式は文字値を返し、実際の値に関係なく符号ニブルの値を **X'F'** として扱います。

比較

PD*w.d.*入力形式は、パック 10 進データを読み込み、それを数値データに変換します。

例

```
input @1 devaddr $phex2.;
```

データ行*	結果
0001111000001111	1E0

* データ行は 2 バイトのバイナリデータを表しています。0.5 バイトが 16 進数の 1 桁に対応します。データ行は 16 進表記で表すと、1E0F に相当します。

$QUOTE*w.* 入力形式

対になっている引用符を文字データから削除します。

カテゴリ: 文字

構文

$QUOTE*w.*

説明

w

入力幅を指定します。

デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

例

```
input @1 name $quote7.;
```

データ行	結果
----+----1	
'SAS'	SAS
"SAS"	SAS
"SAS's"	SAS's

$UPCASEw. 入力形式

文字データを大文字に変換します。

カテゴリ: 文字

構文

$UPCASE*w.*

説明

w
入力幅を指定します。
デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。
範囲: 1–32767

詳細

ハイフンなどの特殊文字は、変更されません。

例

```
input @1 name $upcase3.;
```

データ行	結果
`----+----1`	
`sas`	`SAS`

$VARYINGw. 入力形式

可変長の文字データを読み込みます。

カテゴリ: 文字

構文

$VARYING*w.length-variable*

説明

w
入力ファイルのすべてのレコードに対する文字フィールドの最大幅を指定します。
デフォルト: 変数の長さが定義されていなければ 8。その他の場合、デフォルトは変数の長さ。

範囲: 1–32767

length-variable
現在のレコードの文字フィールドの幅を含む数値変数を指定します。*length-variable* の値は、INPUT ステートメントに記述されたフィールドから直接読み込むか、DATA ステップで値を計算して求めます。

制限事項: *Length-variable* には、配列参照は使えません。

要件 *length-variable* は、INPUT ステートメント内で$VARYING*w*.の直後に指定しなければなりません。

ヒント:

length-variable の値が負の値または欠損値の場合、そのレコードのデータは読み込まれません。

length-variable の値が 0 の場合、変数の値は空白文字になります。*length-variable* の値に 0 を指定することで、ゼロ長のレコードとフィールドの読み込みが可能です。

$VARYING.入力形式以外の入力形式を使用して読み込まれた変数に対して、同一のデータを *length-variable* が 0 の$VARYING.入力形式で読み込むと、前の値が空白値で上書きされます。

length-variable が 0 より大きく、*w* より小さい場合、*length-variable* によって指定されるカラム数が読み込まれます。次に、変数に割り当てられる最大幅になるように値の後に空白が埋め込まれます。

length-variable が *w* 以上の場合、*w* カラムが読み込まれます。

詳細

文字値の長さがレコード間で異なる場合は、$VARYING*w*.を使用します。データ値を$VARYING*w*.で読み込んだ後、ポインタはこのデータ値の次の列に設定されます。

例

例 1: 現在のレコード長を直接取得する

```
input fwidth 1. name $varying9. fwidth;
```

データ行	結果*
----+----1	
5shark	shark
3sunfish	sun
8bluefish	bluefish

* 2 番目のデータ行を読み込んだ結果に注意してください。

例 2: レコード長を間接的に取得する

INFILE ステートメントの LENGTH=オプションを使用して、レコード長を間接的に取得します。入力データ行とその結果は、SAS ステートメントの説明の後に示します。

```
data one;
infile file-specification length=reclen;
input @;
fwidth=reclen-9;
```

```
input name $ 1-9
@10 class $varying20. fwidth;
run;
```

INFILE ステートメントの LENGTH=オプションは、INPUT ステートメントの初回実行時に内部的に保存されたレコード長を RECLEN に割り当てます。後置@は、別の INPUT ステートメントのレコードを保持します。次に、割り当てステートメントは、レコード長の合計からレコードの固定長部分を引いて可変長フィールドの値を計算します。変数 FWIDTH は、最後のフィールドの長さを含み、$VARYING20.入力形式の *length-variable* 引数になります。

データ行	結果
----+----1----+----2	
PATEL CHEMISTRY	PATEL CHEMISTRY
JOHNSON GEOLOGY	JOHNSON GEOLOGY
WILCOX ART	WILCOX ART

$*w*. 入力形式

標準文字データを読み込みます。

カテゴリ: 文字

別名: $F*w*.

構文

$*w*.

説明

w
入力幅を指定します。デフォルト値は提供されていないため、*w* は必ず指定してください。

範囲: 1–32767

詳細

$*w*.入力形式は、テキストの保存前に値の前の空白を切り捨て、値を左揃えにします。また、フィールドに空白と 1 つのピリオドのみ含まれている場合、$*w*.はピリオドを空白に変換します。ピリオドは欠損値として解釈されるためです。$*w*.入力形式は、フィールド内の 2 つ以上のピリオドは文字データとして処理します。

比較

$*w*.入力形式は$CHAR*w*.入力形式とほぼ同じです。ただし、$CHAR*w*.は値の前の空白を切り捨てることも入力フィールドの 1 つのピリオドを空白に変換することも行いませんが、$*w*.はこれらを行います。

例

```
input @1 name $5.;
```

データ行	結果*
----+----1	
XYZ	XYZ##
XYZ	XYZ##
.	
X YZ	X#YZ#

* #文字は空白を表します。

ANYDTDTE*w*. 入力形式

さまざまな日付形式、時間形式、および日時形式から日付値を読み込み、抽出します。

カテゴリ: 日付と時間

構文

ANYDTDTE*w*.

説明

w
入力幅を指定します。
デフォルト: 9
範囲: 5–32

詳細

ANYDTDTE 入力形式は、次の入力形式のいずれか、または日付形式、時間形式、日時形式に対応する入力データを読み込み、派生した値から日付部分を抽出します。

入力形式または入力値の形式	サンプルデータ	入力形式または入力値の形式	サンプルデータ
DATE	01JAN12 01JAN2012	MONYY	JAN12 JAN2012
DATETIME	01JAN12 14:30:08 01JAN2012 14:30:08.5	TIME	14:30 14:30:08.05
DDMMYY	010112 01012012	YMDDTTM	12-01-01 11:23

入力形式または入力値の形式	サンプルデータ	入力形式または入力値の形式	サンプルデータ
JULIAN	12001 2012001	YYMMDD	120101 20120101
MDYAMPM	01-01-12 3:53 pm	YYQ	12Q1 2012Q1
MMDDYY	010112 01012012	YY<YY>*x*MM *	12/01 2012-01
MM*x*YY<YY> *	01/12 01-2012	*month-day-year*	January 1, 2012

* *x* 年から月を区切る特殊文字です。

入力値が時間のみの値である場合、日付は 01JAN1960 と仮定されます。

01-02-03、01-02 などの入力データでは、月、日、年の区別が不明確になる可能性があります。この場合、DATESTYLE システムオプションを使用して、月、日、年の順序を示します。

比較

ANYDTDTE 入力形式は、派生した値から日付部分を抽出します。ANYDTDTM 入力形式は、日時部分を抽出します。ANYDTTME 入力形式は、時間部分を抽出します。

例

```
input dateinfo anydtdte21.;
```

データ行	入力形式	結果	DATE*w*. 出力形式
----+----1----+----2			
01JAN12	DATE	18993	01JAN12
01JAN2012 14:30:08.5	DATETIME	18993	01JAN12
01012012	DDMMYY	18993	01JAN12
2012001	JULIAN	18993	01JAN12
01/01/12	MMDDYY	18993	01JAN12
JAN2012	MONYY	18993	01JAN12
14:30	TIME	0	01JAN60
20120101	YYMMDD	18993	01JAN12
12q1	YYQ	18993	01JAN12

データ行	入力形式	結果	DATE*w*. 出力形式
January 1, 2012	なし	18993	01JAN12

関連項目:

入力形式:

- “ANYDTDTMw. 入力形式” (238 ページ)
- “ANYDTTMEw. 入力形式” (241 ページ)
- “DATEw. 入力形式” (260 ページ)
- “DATETIMEw. 入力形式” (261 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “JULIANw. 入力形式” (282 ページ)
- “MDYAMPMw.d 入力形式” (283 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “MONYYw. 入力形式” (286 ページ)
- “TIMEw. 入力形式” (324 ページ)
- “YMDDTTMw.d 入力形式” (336 ページ)
- “YYMMDDw. 入力形式” (338 ページ)
- “YYQw. 入力形式” (341 ページ)

ANYDTDTM*w*. 入力形式

さまざまな日付形式、時間形式、および日時形式から日時値を読み込み、抽出します。

カテゴリ: 日付と時間

構文

ANYDTDTM*w*.

説明

w
入力幅を指定します。
デフォルト: 19
範囲: 1–32

詳細

ANYDTDTM 入力形式は、次の入力形式のいずれかの形式、または日付/時間形式のデータを読み込み、派生した値から日時部分を抽出します。

入力形式または入力値の形式	サンプルデータ
DATE	01JAN12 01JAN2012
DATETIME	01JAN12 14:30:08 01JAN2012 14:30:08.5
DDMM<YY>YY	010112 01012012
JULIAN	12001 2012001
MMDD<YY>YY *	010112 01012012
MM*x*<YY>YY *	01/12 01-2012
MDYAMPM**	01/01/12 02:30:08 AM 01/01/2012 02:30:08 AM
MONYY	JAN12 JAN2012
TIME	14.30 14:30:08.05
<YY>YYMMDD	120101 20120101
<YY>YYQ	12Q1 2012Q1
<YY>YY*x*MM *	12/01 2012/01
month-day-year	January 1, 2012

* *x* 年から月を区切る特殊文字です。<YY> は、世紀がオプションであることを示します。

** AM | PM がなく、月値と日値の区別が不明確である場合、DATESTYLE=システムオプションの値を使用してその順序を指定します。

入力値が時間のみの値である場合、日付は 01JAN1960 と仮定されます。入力値が日付のみの値である場合、時間は 12:00 (午前 0 時)と仮定されます。入力値の時間値には、時間と分を含める必要があります。入力値にいずれかの日付構成要素が含まれない場合や、時間値の時間および分がないまたは範囲外である場合、読み込まれる値は SAS 欠損値になります。

前述の入力形式の入力値は、MMDDYY、DDMMYY、または YYMMDD を除き、2 桁の年の使用時は相互排他になります。01-02-03、01-02 などの入力データでは、月、日、年の区別が不明確になる可能性があります。この場合、DATESTYLE システムオプションを使用して、月、日、年の順序を示します。

ANYDTTME 入力形式では、時間値のコロン、ピリオドの読み込み時に次のルールが使用されます。

コロンとピリオドの使用	例
値 *h*:*m* の 1 つのコロンは、時間と分を示します。	14:30
値 h:m:s の 2 つのコロンは、時間、分、秒を示します。	14:30:08
値 m:s.ff (ff は秒の分数)の 1 つのピリオドは、ピリオドの前の数字が秒数であることを示します。	2:39.66
値の複数のピリオドは、ピリオドが日付の区切り文字であることを示します。値は時間値ではありません。	12.25.2012

比較

ANYDTDTE 入力形式は、派生した値から日付部分を抽出します。ANYDTDTM 入力形式は、日時部分を抽出します。ANYDTTME 入力形式は、時間部分を抽出します。

例

```
input dateinfo anydtdtm21.;
```

データ行	入力形式またはデータ形式	結果	DATETIME *w.d* 出力形式を適用
----+----1----+----2			
01JAN2012	DATE	1640995200	01JAN12:00:00:00
01JAN2012 14:30:08.5	DATETIME	1641047408.5	01JAN12:14:30:09
01012012	DDMMYY	1640995200	01JAN12:00:00:00
2012001	JULIAN	1546387200	01JAN12:00:00:00
01/01/12	MMDDYY	1546387200	01JAN12:00:00:00
01-12	MMxYY	1546387200	01JAN12:00:00:00
JAN2012	MONYY	1546387200	01JAN12:00:00:00
14:30	TIME	52200	01JAN60:14:30:00
20120101	YYMMDD	1546387200	01JAN12:00:00:00
12Q1	YYQ	1546387200	01JAN12:00:00:00
January 1, 2012	*month-day-year*	1546387200	01JAN12:00:00:00

関連項目:

入力形式:

- “ANYDTDTEw. 入力形式” (236 ページ)
- “ANYDTTMEw. 入力形式” (241 ページ)
- “DATEw. 入力形式” (260 ページ)
- “DATETIMEw. 入力形式” (261 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “JULIANw. 入力形式” (282 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “MONYYw. 入力形式” (286 ページ)
- “TIMEw. 入力形式” (324 ページ)
- “YYMMDDw. 入力形式” (338 ページ)
- “YYQw. 入力形式” (341 ページ)

ANYDTTME*w*. 入力形式

さまざまな日付形式、時間形式、および日時形式から時間値を読み込み、抽出します。

カテゴリ: 日付と時間

構文

ANYDTTME*w*.

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 1–32

詳細

ANYDTTME 入力形式は、次の入力形式または形式のいずれかに対応する入力データを読み込みます。

入力形式または入力値の形式	サンプルデータ	入力形式または入力値の形式	サンプルデータ
DATE	01JAN12 01JAN2012	MONYY	JAN12 JAN2012
DATETIME	01JAN12 14:30:08 01JAN2012 14:30:08.5	YYMMDD	120101 20120101

入力形式または入力値の形式	サンプルデータ	入力形式または入力値の形式	サンプルデータ
DDMMYY	010112 01012012	YYQ	12Q1 2012Q1
JULIAN	12001 2012001	YYQ	12Q1 2012Q1
MMDDYY	010112 01012012	*month-day-year*	January 1, 2012 2012-01

入力値が時間のみの値である場合、日付は 01JAN1960 と仮定されます。入力値が日付のみの値である場合、時間は 12:00 (午前 0 時)と仮定されます。

01-02-03、01-02 などの入力データでは、月、日、年の区別が不明確になる可能性があります。この場合、DATESTYLE システムオプションを使用して、月、日、年の順序を示します。

ANYDTTME 入力形式では、時間値のコロン、ピリオドの読み込み時に次のルールが使用されます。

コロンとピリオドの使用	例
値 *h*:*m* の 1 つのコロンは、時間と分を示します。	14:30
値 h:m:s の 2 つのコロンは、時間、分、秒を示します。	14:30:08
値 m:s.ff (ff は秒の分数)の 1 つのピリオドは、ピリオドの前の数字が秒数であることを示します。	2:39.66
値の複数のピリオドは、ピリオドが日付の区切り文字であることを示します。値は時間値ではありません。	12.25.2012

比較

ANYDTDTE 入力形式は、派生した値から日付部分を抽出します。ANYDTDTM 入力形式は、日時部分を抽出します。ANYDTTME 入力形式は、時間部分を抽出します。

例

```
input dateinfo anydttme21.;
```

データ行	入力形式	結果	TIME*w.d* 出力形式を適用
----+----1----+----2			

データ行	入力形式	結果	TIME*w.d* 出力形式を適用
01JAN12	DATE	0	00:00:00
01JAN2012 14:30:08.5	DATETIME	52208.5	14:30:09
010112	DDMMYY	0	00:00:00
2012001	JULIAN	0	00:00:00
01012012	MMDDYY	0	00:00:00
JAN2012	MONYY	0	00:00:00
14:30:08.5	TIME	52208.5	14:30:09
20120101	YYMMDD	0	00:00:00
12Q1	YYQ	0	00:00:00
January 1, 2012	*month-day-year*	0	00:00:00

関連項目:

入力形式:

- “ANYDTDTEw. 入力形式” (236 ページ)
- “ANYDTDTMw. 入力形式” (238 ページ)
- “DATEw. 入力形式” (260 ページ)
- “DATETIMEw. 入力形式” (261 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “JULIANw. 入力形式” (282 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “MONYYw. 入力形式” (286 ページ)
- “TIMEw. 入力形式” (324 ページ)
- “YYMMDDw. 入力形式” (338 ページ)
- “YYQw. 入力形式” (341 ページ)

B8601Clw.d 入力形式

世紀マーカーを含む、*cyymmddhhmmss<fff>*形式の IBM 日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

構文

B8601CI*w.d*

説明

w
入力幅を指定します。
デフォルト: 16
範囲: 10–26

d
秒値の小数点以下の桁数を指定します。
デフォルト: 0
範囲: 0–6

詳細

B8601CI 入力形式は、IBM 時間表記 *cyymmddhhmmss<fff>*で示される時間値を読み込みます。

c
世紀を表す 1 桁の数字です。

0 は、年 1900–1999 を示します。

1 は、年 2000–2099 を示します。

2 は、年 2100–2199 を示します。

n は、2199 を超える年から計算して決定される世紀の年 00–99 を示します。世紀マーカーを決定するには、年から 1900 を引き、その結果を 100 で割ります。余りは切り捨てます 残った整数が世紀マーカーです。たとえば、年 2382 に対して世紀マーカーを決定するには、計算(2382-1900)/100=4.82 を実行します。.82 を切り捨てます。この世紀マーカーは 4 になります。

yy
00 から 99 までの 2 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

fff
3 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
input @1 bci b8601ci.;
```

日付と時間	データ行	結果
	----+----1----+	
January 1, 1900 12:00:00	0000101120000	-1893326400
October 1, 2011 12:15:30.25	111100112153025	1633090530.3

B8601DAw. 入力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して示される日付値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: ND8601DA

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DA*w*.

説明

w
入力幅を指定します。
デフォルト: 10
要件 10 である必要があります。

詳細

B8601DA 入力形式は、ISO 8601 規格の基本表記 *yyyymmdd* を使用して示される日付値を読み込みます。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

月または日の値が省略される場合、SAS では、月または日に 1 の値が使用されます。時間、分または秒の値が省略される場合、SAS では、時間、分または秒に 0 の値が使用されます。

例

```
input @1 bda b8601da.;
```

データ行	結果	B8601DA 出力形式の適用後
----+----1		
20120915	19251	20120915
2012	18993	20120101

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

B8601DJ*w.d* 入力形式

*yyyymmddhhmmss<ffffff>*形式の Java 日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

構文

B8601DJ*w.d*

説明

w
入力幅を指定します。
デフォルト: 16
範囲: 10–26

d
秒値の小数点以下の桁数を指定します。
デフォルト: 0
範囲: 0–6

詳細

B8601DJ 入力形式は、Java 日時表記 *yyyymmddhhmmss<ffffff>*を使用して示される日時値を読み込みます。

yyyy
0000 から 9999 までの 4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

比較

B8601DJ 入力形式は、時間から日付を区切るための T を含まない日時値を読み込みます。

Java 日時値は、T を含みません。たとえば、2011 年 1 月 1 日の午前 4:30:25 は、20110101043025 と表記されます。

ISO 8601 規格の日時値は、T を含みます。たとえば、2011 年 1 月 1 日の午前 4:30:25 は、20110101T043025 と表記されます。

例

```
input @1 bdj b8601dj.;
```

日付と時間	データ行	結果
	`----+----1----+-----`	
July 31, 2011 2:33:35 p.m.	`20110731142335`	`1627741415`
September 1, 2012 7:30:00.33 a.m.	`2012090107300033`	`1662103800.3`

B8601DNw. 入力形式

ISO 8601 規格の基本表記 *yyyymmdd* を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: ND8601DN

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

B8601DN*w*.

説明

w
入力幅を指定します。
デフォルト: 10
要件 入力幅は 10 である必要があります。

詳細

B8601DN 入力形式は、ISO 8601 規格の基本日付表記 *yyyymmdd* を使用して示される日付値を読み込み、SAS 日時値の日付を返します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
input @1 bdn b8601dn.;
```

データ行	結果
`----+----1`	
`20120915`	`1663286400`

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

B8601DT*w.d* 入力形式

ISO 8601 規格の基本表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して示される日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601
配置: 左
別名: ND8601DT
制限事項: UTC タイムゾーンオフセット値はサポートされていません。

構文

B8601DT*w.d*

説明

w
入力幅を指定します。
デフォルト: 19
範囲: 19–26

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

B8601DT 入力形式は、ISO 8601 規格の基本日時表記 *yyyymmdd*T*hhmmss*<*ffffff*>を使用して示される日時値を読み込みます。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

月または日の値が省略される場合、SAS では、月または日に 1 の値が使用されます。時間、分または秒の値が省略される場合、SAS では、時間、分または秒に 0 の値が使用されます。

例

```
input @1 bdt b8601dt.;
```

データ行	結果	B8601DT 出力形式の適用後
----+----1		
20120915T155300	1663343580	20120915T155300
2012	1640995200	20120101T000000

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

B8601DZ*w.d* 入力形式

ISO 8601 規格の日時基本表記 *yyyymmdd*T*hhmmss*+|-*hhmm* または *yyyymmdd*T*hhmmss*<*ffffff*>Z を使用して示される協定世界時(UTC)の日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: ND8601DZ

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.4.1, complete representation

構文

B8601DZ*w.d*

説明

w
入力幅を指定します。
デフォルト: 26
範囲: 20–35

d
秒の端数を表す秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601DZ 入力形式は、次の ISO 8601 規格の基本日時表記のいずれかで示される日時値を読み込みます。

- *yyyymmdd*T*hhmmss*+|–*hhmm*
- *yyyymmdd*T*hhmmss*<*ffffff*>Z

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 24 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

+|–*hhmm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hhmm* (つまり+または-を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000 UTC 時間であることを示します。

例

```
input @1 bdz b8601dz.;
```

データ行	結果
`----+----1`	
`20120915T155300+0500`	`1663325580`

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

B8601TM*w.d* 入力形式

ISO 8601 規格の基本表記 *hhmmss<ffffff>*を使用して示される時間値を読み込みます

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: ND8601TM

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Elements 5.3.1.1 and 5.3.1.3, complete representation and representation of decimal fractions

構文

B8601TM*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 6–15

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

B8601TM 入力形式は、ISO 8601 規格の基本時間表記 *hhmmss<ffffff>*を使用して示される時間値を読み込みます。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
input @1 btm b8601tm;
```

データ行	結果
----+----1	
155300	57180

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

B8601TZ*w.d* 入力形式

ISO 8601 規格の基本時間表記 *hhmmss<ffff>*+|–*hhmm* または *hhmmss<ffffff>*Z で示される時間値を読み込みます。

カテゴリ: 日付と時間

ISO 8601

配置: 左

別名: ND8601TZ

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

B8601TZ*w.d*

説明

w

入力幅を指定します。

デフォルト: 14

範囲: 9–20

d

(オプション)秒値の小数点以下の桁数を指定します。

デフォルト: 0

範囲: 0–6

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。B8601TZ 入力形式は、次の ISO 8601 規格の基本時間表記のいずれかで示される時間値を読み込みます。

- *hhmmss*<*ffffff*>+|–*hhmm*
- *hhmmss*<*ffffff*>Z

hh

00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm

00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss

00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff

6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

+|–*hh:mm*

基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hhmm* (つまり+または-を伴う 4 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+0200 は基準子午線の東部での 2 時間差を表し、-0600 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z

時間が基準子午線(イギリスのグリニッジ)に対応した時間、つまり+0000 UTC 時間であることを示します。

B8601TZ 入力形式を使用して UTC 時間を読み込み、調整された時間が 240000 より大きい、または 000000 より小さい場合、000000 と 240000 の間になるように時間値を調整します。たとえば、B8601TZ 入力形式を使用して UTC 時間 234344-0500 を読み込む場合、この UTC 時間に 5 時間を追加して 284344 としてから、時間調整を行います。時間 044344+0000 を示す値が保存されます。

例

```
input @1 btz b8601tz.;
```

データ行	結果
----+----1	
202401-0500	5041
202401Z	73441
202401+0000	73441

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

BINARY*w.d* 入力形式

正のバイナリ値を整数に変換します。

カテゴリ: 数値

構文

BINARY*w.d*

説明

w
入力幅を指定します。

デフォルト: 8

範囲: 1–64

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれている場合も *d* 値が使用されます。この引数はオプションです。

範囲: 0–31

詳細

入力には、数字 1 と 0 のみを使用します。空白は挿入できません。BINARY*w.d* では、先頭と末尾の空白は無視されます。

BINARY*w.d* では、負の値を読み込むことはできません。すべての入力値は正数(符号なし)として処理されます。

例

```
input @1 value binary8.1;
```

データ行	結果
----+----1	
00001111	1.5

BITS*w.d* 入力形式

ビットを抽出します。

カテゴリ: 数値

構文

BITS*w.d*

説明

w
読み込むビット数を指定します。
デフォルト: 1
範囲: 1–64

d
ゼロを基点とするオフセットを指定します。
範囲: 0–63

詳細

BITS*w.d* 入力形式は入力ストリームから指定したビットを抽出して、抽出したビット文字列に相当する数値を変数に割り当てます。同時に、*w* と *d* の値により、文字列の読み込む場所が指定されます。

この入力形式は、1 バイトに多くの情報を含むシステムレコードからデータを抽出するときに使用します。

例

```
input @1 value bits4.1;
```

データ行	結果*
----+----1----+	

データ行	結果*
B	8

* 大文字 B は、EBCDIC バイナリコードでは 11000010、ASCII バイナリコードでは 01000010 となります。

入力ポインタはカラム 2 (*d*=1)に移動します。次に、INPUT ステートメントは 4 ビット (*w*=4)、ここではビット文字列 1000 を読み込み、このバイナリの組み合わせに相当する数値 8 を保存します。

BZ*w.d* 入力形式

空白をゼロに変換します。

カテゴリ: 数値

構文

BZ*w.d*

説明

w
入力幅を指定します。

デフォルト: 1

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。

範囲: 0–31

詳細

BZ*w.d* 入力形式は、数値を読み込み、末尾の空白や埋め込まれている空白を 0 に変換しますが、先頭の空白は無視します。

BZ*w.d* 入力形式は、フィールド内の数値はその位置にかかわらず読み込むことができます。空白は数値の前後に置くことが可能です。マイナス記号は負の値の前に置く必要があります。BZ*w.d* 入力形式は、入力フィールド内のマイナス記号と数値の間の空白を無視します。

BZ*w.d* 入力形式は、フィールド内の 1 つのピリオドを 0 として解釈します。フィールド内の複数のピリオドまたはその他の数値以外の文字は欠損値として解釈します。

BZ*w.d* を DATA ステップのリスト入力で使用するには、INFILE ステートメントの DLM=オプションまたは DLMSTR=オプションを使用して、リスト入力の区切り文字を変更します。デフォルトでは、データ行の値の間の空白は 0 ではなく、区切り文字として解釈されます。

比較

BZ*w.d* 入力形式は、末尾の空白と埋め込まれている空白を 0 に変換します。末尾の空白を 0 に変換しない場合(E 表記で値を読み込む場合など)は、*w.d* 入力形式か、E*w.d* 入力形式を代わりに使用します。

例

```
input @1 x bz4.;
```

データ行	結果
----+----1	
34	3400
-2	-200
-2 1	-201

CB*w.d* 入力形式

カラムバイナリファイルから標準数値を読み込みます。

カテゴリ: カラムバイナリ

構文

CB*w.d*

説明

w
入力幅を指定します。
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれている場合も *d* 値が使用されます。この引数はオプションです。

詳細

カラムバイナリデータストレージは、80 項目を超えるデータを 1 つの"仮想"パンチカードに保存できるように、データを圧縮します。

CB*w.d* 入力形式は標準数値をカラムバイナリファイルから読み込み、そのデータを標準バイナリ形式に変換します。

まず、CB*w.d* を使用して読み込んだカラムバイナリデータの各カラムを 2 進表記で保存します。このとき、各バイトの高位 2 ビットは無視されます。パンチコードが有効な場合、それに相当する数値を指定の変数に保存します。組み合わせが無効な場合、変数に欠損値を割り当て、自動変数_ERROR_を 1 に設定します。

例: 例

```
input @1 x cb8.;
```

データ行*	結果
----+----1	
0009	9

* データ行はカラムバイナリの 16 進表現です。サンプルデータの“仮想”パンチカードカラムでは、行 9 がパンチされています。2 進表現は、0000 0000 0000 1001 となります。

関連項目:

- “How to Read Column-Binary Data” in Chapter 19 of *SAS Language Reference: Concepts* in *SAS 言語リファレンス: 解説編*

入力形式

- “$CBw. 入力形式” (221 ページ)
- “PUNCH.d 入力形式” (300 ページ)
- “ROWw.d 入力形式” (305 ページ)

COMMA*w.d* 入力形式

埋め込み文字を取り除きます。

カテゴリ: 数値

別名: DOLLAR*w.d*

構文

COMMA*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。
範囲: 0–31

詳細

COMMA*w.d* 入力形式では数値を読み込み、埋め込まれているカンマ、空白、ドル記号、パーセント記号、ハイフン、閉じかっこを入力データから取り除きます。
COMMA*w.d* 入力形式では、フィールドの先頭にある開きかっこがマイナス記号に変換されます。

比較

COMMA*w.d* 入力形式は、COMMAX*w.d* 入力形式と機能は似ていますが、小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。

例

```
input @1 x comma10.;
```

データ行	結果
----+----1----+	
$1,000,000	1000000
(500)	-500

COMMAX*w.d* 入力形式

埋め込まれているピリオド、空白、ドル記号、パーセント記号、ダッシュ、閉じかっこを入力データから取り除きます。フィールドの先頭にある開きかっこはマイナス記号に変換されます。COMMAX 入力形式では、小数点とカンマの役割が逆になっています。

カテゴリ: 数値

別名: DOLLARX*w.d*

構文

COMMAX*w.d*

説明

w
入力幅を指定します。

デフォルト: 1

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点を表すカンマが含まれる場合、*d* 値は無視されます。この引数はオプションです。

範囲: 0–31

詳細

COMMAX*w.d* 入力形式では数値を読み込み、埋め込まれているピリオド、空白、ドル記号、パーセント記号、ハイフン、閉じかっこを入力データから取り除きます。
COMMAX*w.d* 入力形式では、フィールドの先頭にある開きかっこがマイナス記号に変換されます。

比較

COMMAX*w.d* 入力形式は、COMMA*w.d* 入力形式と機能は似ていますが、小数点とカンマの役割が逆になっています。この規則はヨーロッパの国で共通です。

例

```
input @1 x commax10.;
```

データ行	結果
----+----1----+	
$1.000.000	1000000
1.234,56	1234.56
(500)	-500

DATE*w.* 入力形式

ddmmmyy または *ddmmmyyyy* 形式の日付値を読み込みます。

カテゴリ: 日付と時間

構文

DATE*w.*

説明

w
入力幅を指定します。
デフォルト: 7
範囲: 7–32
ヒント: 4 桁の年を読み込む場合は幅に 9 を指定します。

詳細

日付値は *ddmmmyy* または *ddmmmyyyy* 形式にする必要があります。

dd
月の日を表す 01 から 31 の整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

年、月、日の値は、空白や特殊文字で区切ることができます。入力幅には、空白および特殊文字のスペースを考慮する必要があります。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input calendar_date date11.;
```

データ行	結果
----+----1----+	
16mar12	19068
16 mar 12	19068
16-mar-2012	19068

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)

関数:

- “DATE Function” in *SAS Functions and CALL Routines: Reference*

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

DATETIME*w.* 入力形式

ddmmmyy hh:mm:ss.ss または *ddmmmyyyy hh:mm:ss.ss* 形式の日時値を読み込みます。

カテゴリ: 日付と時間

構文

DATETIME*w.*

説明

w
入力幅を指定します。
デフォルト: 18
範囲: 13–40

詳細

日時値は *ddmmmyy* または *ddmmmyyyy* 形式にし、その後に空白または特殊文字を挿入し、次に *hh:mm:ss.ss* (時間)を追加する必要があります。

dd
月の日を表す 01 から 31 の整数です。

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

hh
時間を表す 00 から 23 までの整数です。

mm
分を表す 00 から 59 までの整数です。

ss.ss
小数点以下の秒の端数を含む、00 から 59 の範囲の秒数です。

DATETIME*w.*には、日付と時間の両方の値が必須です。ただし、*ss.ss* の部分はオプションです。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

注: AM と PM を含む時間値を読み込むことができます。

比較

DATETIME*w.d* 入力形式は、オプションの区切り文字を含む *dd-mmm-yy<yy> hh:mm:ss.ss* AM|PM 形式の日時値を読み込みます。日付と時間は特殊文字で区切られる場合もあります。

MDYAMPM*w.d* 入力形式は、オプションの区切り文字を含む *mm-dd-yy<yy> hh:mm:ss.ss* AM | PM 形式の日時値を読み込みます。日付と時間の間にはスペースが必要です。

YMDDTTM*w.d* 入力形式は、必須の区切り文字を含む*<yy>yy-mm-dd/hh:mm:ss.ss* 形式の日時値を読み込みます。

例

```
input date_and_time datetime20.;
```

データ行	結果
----+----1----+----2	
16mar12:11:23:07.4	1647516187.4
16mar2012/11:23:07.4	1647516187.4
16mar2012/11:23 PM	1647559380.0

関連項目:

- “About SAS Date, Time, and Datetime Values” in Chapter 7 of *SAS Language Reference: Concepts*

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DATETIMEw.d 出力形式” (72 ページ)
- “TIMEw.d 出力形式” (152 ページ)

関数:

- “DATETIME Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “MDYAMPMw.d 入力形式” (283 ページ)
- “TIMEw. 入力形式” (324 ページ)
- “YMDDTTMw.d 入力形式” (336 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

DDMMYY*w*. 入力形式

ddmmyy<*yy*>または *dd-mm-yy*<*yy*>形式の日付値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)などの特殊文字で日、月、年を区切ります。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

構文

DDMMYY*w*.

説明

w
入力幅を指定します。
デフォルト: 6
範囲: 6–32

詳細

日付値は *ddmmyy*<*yy*>または *ddxmmxyy*<*yy*>形式にする必要があります。

dd
月の日を表す 01 から 31 の整数です。

mm
月を表す 01 から 12 までの整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

x
区切り文字です。特殊文字または空白となります。

区切り文字を使用する場合は、すべての値の間に挿入します。空白は日付の前後に挿入することもできます。入力幅には、空白および特殊文字のスペースを考慮する必要があります。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input calendar_date ddmmyy10.;
```

データ行	結果
----+----1----+	
160308	19068
16/03/08	19068
16-03-2008	19068
16 03 2008	19068

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “MDY Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

E*w.d* 入力形式

指数表記および倍精度の指数表記で保存されている数値を読み込みます。

カテゴリ: 数値

参照項目: E*w.d* 入力形式(z/OS)

構文

E*w.d*

説明

w
数値を含むフィールドの幅を指定します。

デフォルト: 12

範囲: 1–32

d
数値の小数点以下の桁数を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。

範囲: 0–31

比較

E*w.d* 入力形式は、標準数値データの SAS 入力形式である *w.d* 入力形式でも指数表記の数字を読み込めるため、広く使用されるものではありません。

例

```
input @1 x e7.;
```

データ行	結果
----+----1----+	
1.257E3	1257
12d3	12000

E8601DA*w.* 入力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して示される日付値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DA

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DA*w.*

説明

w
入力幅を指定します。
デフォルト: 10
要件 入力幅は 10 である必要があります。

詳細

E8601DA 入力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd* で示される日付値を読み込みます。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
input eda e8601da.;
```

データ行	結果
----+----1	
2012-09-15	19251

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

E8601DN*w*. 入力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd* を使用して示される日付値を読み込み、値の時間部分が 000000 である SAS 日時値を返します。

カテゴリ: 日付と時間
ISO 8601
配置: 左
別名: IS8601DN
制限事項: UTC タイムゾーンオフセット値はサポートされていません。
サポート: ISO 8601 Element 5.2.1.1, complete representation

構文

E8601DN*w*.

説明

w
入力幅を指定します。
デフォルト: 10
要件 入力幅は 10 である必要があります。

詳細

E8601DN 入力形式は、ISO 8601 規格の拡張日付表記 *yyyy-mm-dd* を使用して示される日付値を読み込み、SAS 日時値の日付を返します。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

例

```
input edn e8601dn.;
```

データ行	結果
----+----1	
2012-09-15	1663286400

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

E8601DT*w.d* 入力形式

ISO 8601 規格の拡張表記 *yyyy-mm-dd*T*hh:mm:ss.<ffffff>*を使用して示される日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601
配置: 左
別名: IS8601DT
制限事項: UTC タイムゾーンオフセット値はサポートされていません。
サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DT*w.d*

説明

w
入力幅を指定します。
デフォルト: 19
範囲: 19–26

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

E8601DT 入力形式は、ISO 8601 規格の拡張日時表記 *yyyy-mm-dd*T*hh:mm:ss.<ffffff>*を使用して示される日時値を読み込みます。

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
input @1 edt e8601dt.;
```

データ行	結果
----+----1----+----2----+----3	
2012-09-15T15:53:00	1663343580

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

E8601DZ*w.d* 入力形式

ISO 8601 規格の日時拡張表記 *yyyy-mm-dd*T*hh*:*mm*:*ss*+|-*hh*:*mm*.<*fffff*>または *yyyy-mm-dd*T*hh*:*mm*:*ss*.<*fffff*>Z を使用して示される、協定世界時(UTC)の日時値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601DZ

サポート: ISO 8601 Element 5.4.1, complete representation

構文

E8601DZ*w.d*

説明

w
入力幅を指定します。
デフォルト: 26
範囲: 20–35

d
最小単位の構成要素に対し値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601DZ 入力形式は、UTC 時間オフセットを含み、次の ISO 8601 規格の拡張日時表記のいずれかで示される日時値を読み込みます。

- *yyyy-mm-dd*T*hh:mm:ss*.<*ffffff*>+|–*hh:mm*
- *yyyy-mm-dd*T*hh:mm:ss*.<*ffffff*>Z

yyyy
4 桁の年です。

mm
01 から 12 までの 2 桁の月です(ゼロ埋め込み)。

dd
01 から 31 までの 2 桁の日です(ゼロ埋め込み)。

hh
00 から 24 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

+|–*hh*:*mm*
基準子午線時間からの時間と分の符号付きオフセットです。オフセットは +|-*hh*:*mm* (つまり+または-に続く 5 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。たとえば、+02:00 は基準子午線の東部での 2 時間差を表し、–06:00 は基準子午線の西部での 6 時間差を表します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
時間が基準子午線(イギリスのグリニッジ)上の UTC 時間であることを示します。

例

INPUT ステートメント	データ行	結果
	----1----+----2----+----3	
input edz e8601dz.;	2012-09-15T15:53:00Z	1663343580
input edz e8601dz28.2;	2012-09-15T15:53:00+03:00	1663332780

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

E8601LZ*w.d* 入力形式

ISO 8601 規格の拡張表記 *hh*:*mm*:*ss*+|–*hh*:*mm*.<*fffff*>または *hh*:*mm*:*ss*.<*fffff*>Z を使用して示される協定世界時(UTC)値を読み込み、ローカル時間に変換します。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601LZ

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601LZ*w.d*

説明

w
入力幅を指定します。
デフォルト: 14

範囲: 9–20

要件 Z タイムゾーンインジケータ付きの時間を読み込む場合、同じデータ行にデータが続くときは入力幅に 9 を指定する必要があります。

d

最小単位の構成要素に対し値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–6

詳細

UTC 値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601LZ 入力形式は、次の ISO 8601 規格の拡張時間表記のいずれかで示される UTC 時間値を読み込み、ローカル時間に対応する SAS 時間値を返します。

- *hh:mm:ss.<ffffff>+|–hh:mm*
- *hh:mm:ss.<ffffff>*Z

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

+|–*hh:mm*
基準子午線からの時間と分の符号付きオフセットです。オフセットは +|-*hh:mm* (つまり+または-に続く 5 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
基準子午線に対応した時間、つまり+00:00 UTC 時間であることを示します。

E8601LZ 入力形式を使用して UTC 時間を読み込み、調整された時間が 24:00:00 より大きい、または 00:00:00 より小さい場合、時間が 00:00:00 と 24:00:00 の間になるように値を調整します。たとえば、E8601LZ 入力形式を使用して UTC 時間 23:43:44-05:00 を読み込む場合、この UTC 時間に 5 時間を追加して 28:43:44 としてから、時間調整を行います。時間 04:43:44+00:00 を示す値が保存されます。

例

```
input elz e8601lz.;
```

データ行	結果
`----+----1----+`	
`09:13:21+02:00`	`26001`

データ行	結果
23:43:44Z	85424

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

E8601TM*w.d* 入力形式

ISO 8601 規格の拡張表記 *hh:mm:ss.<ffffff>*を使用して示される時間値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TM

制限事項: UTC タイムゾーンオフセット値はサポートされていません。

サポート: ISO 8601 Elements 5.3.1.1 and 5.3.1.3, complete representation and representation of decimal fractions

構文

E8601TM*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 8–15

d
秒値の小数点以下の桁数を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–6

詳細

E8601TM 入力形式は、ISO 8601 規格の拡張時間表記 *hh:mm:ss.<ffffff>*を使用して示される時間値を読み込みます。

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

例

```
input @1 etm e8601tm.;
```

データ行	結果
----+----1	
15:53:00	57180

関連項目:

"ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み" (203 ページ)

E8601TZ*w.d* 入力形式

ISO 8601 規格の拡張時間表記 *hh*:*mm*:*ss*+|–*hh*:*mm*.<*ffffff*>または *hh*:*mm*:*ss*Z を使用して示される時間値を読み込みます。

カテゴリ: 日付と時間
ISO 8601

配置: 左

別名: IS8601TZ

サポート: ISO 8601 Element 5.3.1.1, complete representation

構文

E8601TZ*w.d*

説明

w
入力幅を指定します。

デフォルト: 14

範囲: 9–20

要件 Z タイムゾーンインジケータ付きの時間を読み込む場合、同じデータ行にデータが続くときは入力幅に 9 を指定する必要があります。

d
最小単位の構成要素に対し値の小数点以下の桁数を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–6

詳細

UTC 時間値は、イギリスのグリニッジの基準子午線に基づく時間とタイムゾーンを示します。E8601TZ 入力形式は、次の ISO 8601 規格の拡張表記のいずれかで示される UTC 時間値を読み込みます。

- *hh:mm:ss*+|-*hh:mm.*<*ffffff*>
- *hh:mm:ss*Z

hh
00 から 23 までの 2 桁の時間です(ゼロ埋め込み)。

mm
00 から 59 までの 2 桁の分です(ゼロ埋め込み)。

ss
00 から 59 までの 2 桁の秒です(ゼロ埋め込み)。

ffffff
6 桁までの有効桁数を持つ、オプションの秒の分数です。各桁は 0 から 9 までです。

+|–*hh:mm*
基準子午線からの時間と分の符号付きオフセットです。オフセットは +|-*hh:mm* (つまり+または-に続く 5 文字)である必要があります。

+は基準子午線の東部タイムゾーンに、-は基準子午線の西部タイムゾーンにそれぞれ使用します。

制限: 短い形式+|–*hh* は、サポートされていません。

Z
基準子午線に対応した時間、つまり+00:00 UTC 時間であることを示します。

E8601TZ 入力形式を使用して UTC 時間を読み込み、調整された時間が 24:00:00 より大きい、または 00:00:00 より小さい場合、時間が 00:00:00 と 24:00:00 の間になるように値を調整します。たとえば、E8601TZ 入力形式を使用して UTC 時間 23:43:44-05:00 を読み込む場合、この UTC 時間に 5 時間を追加して 28:43:44 としてから、時間調整を行います。時間 04:43:44+00:00 を示す値が保存されます。

例

```
input @1 etz e8601tz.;
```

データ行	結果
`----+----1----+`	
`23:43:44-05:00`	`17024`
`23:43:44Z`	`85424`

関連項目:

“ISO 8601 規格の基本表記と拡張表記を使用した日付と時間の読み込み” (203 ページ)

FLOAT*w.d* 入力形式

正の単精度浮動小数点値を読み込み、その値を 10 の *d* 乗で除算します。

カテゴリ: 数値

構文

FLOAT*w.d*

説明

w
入力幅を指定します。
要件 *w* は 4 である必要があります。

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。

詳細

FLOAT*w.d* 入力形式は、浮動数値が倍精度の切り捨て値と異なる動作環境に使用します。

IBM メインフレームシステム上では、4 バイトの浮動小数点数は 8 バイトに切り捨てられた浮動小数点数と同じです。IEEE 浮動小数点数規格を使用する動作環境(IBM PC、UNIX など)では、4 バイトの浮動小数点数は倍精度の切り捨て値と異なります。そのため、RB4.入力形式は FLOAT4.と同じ結果を生成しません。IEEE 以外の浮動小数点表現にもこの同じ特徴がみられます。FLOAT4.で読み込まれる値は通常、使用している動作環境で実行されている他の外部プログラムから取り込まれます。

比較

次の表は、各種プログラミング言語での浮動表記を比較したものです。

言語	浮動表記
SAS	FLOAT4.
Fortran	REAL*4
C	float
IBM 370 ASM	E
PL/I	FLOAT BIN(21)

例

```
input x float4.;
```

データ行*	結果
----+----1----+----2	
3F800000	1

* データ行は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

HEX*w*. 入力形式

16 進の正のバイナリ値を整数(固定小数点)または実数(浮動小数点)のバイナリ値に変換します。

カテゴリ: 数値

参照項目: “HEXw. Informat: UNIX” in *SAS Companion for UNIX Environments*
“HEXw. Informat: Windows” in *SAS Companion for Windows*
“HEXw. Informat: z/OS” in *SAS Companion for z/OS*

構文

HEX*w*.

説明

w
入力値のフィールド幅を指定し、最終値を固定小数点または浮動小数点にするかも指定します。

デフォルト: 8

範囲: 1–16

ヒント: *w*<16 の場合、HEX*w*.は入力値を正のバイナリ整数値に変換し、すべての入力値を正数(符号なし)として処理します。*w* が 16 の場合、HEX*w*.は負の値を含めて入力値をバイナリ実数(浮動小数点数)値に変換します。

詳細

動作環境の情報
動作環境によって浮動小数点数の保存方法は異なります。ただし、HEX16.は、使用されている動作環境での保存方法と同じ方法で表現されているとして、16 進表現の浮動小数点数を読み込みます。

HEX*w*.入力形式は、先頭の空白または末尾の空白を無視します。

例

```
input @1 x hex3. @5 y hex16.;
```

データ行*	結果
----+----1----+----2	

データ行*	結果
88F 4152000000000000	2191 5.125

* データ行は IBM メインフレームの 16 進データです。

HHMMSS*w*. 入力形式

形式 *hh*:*mm*:*ss* または *hhmmss*.の時、分、秒を読み取ります。

カテゴリ: 日付と時間

構文

HHMMSS*w*.

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 1–20

詳細

HHMMSS*w*.入力形式は、次のうちいずれかの形式の時間値を読み取ります。

- *hh*:*mm*:*ss*
- *hhmmss*

hh
時間数を表す整数です。

:
時間、分、秒を区切る特殊文字を表します。

mm
分数を表す整数です。

ss
秒数を表す整数です。秒数の端数は無視されます。

入力データが 6 桁以下の場合、データは左から右へ時間、分、秒として読み込まれます。6 桁未満のデータには右側にゼロが埋め込まれます。最初の 2 桁は時間として読み込まれます。3 桁目と 4 桁目は分として読み込まれます。5 桁目と 6 桁目は秒として読み込まれます。

1 は 100000 または 10:00:00 と同じです。
02 は 020000 または 02:00:00 と同じです。
124 は 124000 または 12:40:00 と同じです。
1435 は 143500 または 14:35:00 と同じです。
20345 は 203450 または 20:34:50 と同じです。
165532 は 16:55:32 と同じです。

6 桁を超えている場合、右端の 2 桁が秒として読み込まれます。右から 3 桁目と 4 桁目が分として読み込まれます。分の左の残り桁が時間として読み込まれます。

2358444 は 235:84:44 と同じです。

12545533 は 1254:55:33 と同じです。

入力データに含まれるコロンが 1 つだけの場合(たとえば、17:35)、コロンの前の 2 桁は時間として読み込まれます。コロンの後の 2 桁は秒として読み込まれます。秒数は 0 です。

注: 12:3400 のように分と秒の間のコロンが省略される場合、3400 は 3400 分として読み込まれます。3400 分では、12 時間に 56 時間と 40 分が追加されるため、68:40:00 となります。次の例を参照してください。

例

input tm hhmmss.;

データ行	結果	TIME*w* を適用
23	82800	23:00:00
12:45:44	45344	12:45:44
2358444	851084	236:24:44
17:35	63300	17:35:00
12:3400	247200	68:40:00

関連項目:

入力形式:

- “TIMEw. 入力形式” (324 ページ)

IB*w.d* 入力形式

負の値を含む、ネイティブのバイナリ整数(固定小数点)値 を読み込みます。

カテゴリ: 数値

参照項目: “IBw.d Informat: UNIX” in *SAS Companion for UNIX Environments*
“IBw.d Informat: Windows” in *SAS Companion for Windows*
“IBw.d Informat: z/OS” in *SAS Companion for z/OS*

構文

IB*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

IB*w.d* 入力形式では、2 の補数で表される負の値を含む、バイナリ整数(固定小数点)値を読み込みます。IB*w.d* では、バイナリ整数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

IB*w.d* と PIB*w.d* 入力形式は、ネイティブな形式の整数を読み込むために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 入力形式は、動作環境に関係なくリトルエンディアン整数を読み込むために使用されます。

ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、IB 入力形式を指定できます。ただし、これらの例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('0080'x,ib2.);
y=input('8000'x,ib2.);
```

SAS ステートメント	ビッグエンディアンプラットフォームでの結果	リトルエンディアンプラットフォームでの結果
put x=;	128	-32768
put y=;	-32768	128

関連項目:

入力形式:

- “IBRw.d 入力形式” (280 ページ)

IBR*w.d* 入力形式

Intel および DEC 形式のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

構文

IBR*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

IBR*w.d* 入力形式では、2 の補数で表される負の値を含む、バイナリ整数(固定小数点)値を読み込みます。IBR*w.d* では、Intel および DEC のプラットフォームで生成されるバイナリ整数値を読み込みます。IBR*w.d* を使用して、Intel または DEC の環境のバイナリ整数データを他の動作環境で読み込みます。IBR*w.d* 入力形式によって、動作環境に関係なくデータを読み込むためのポータブルな実装が用意されます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

IB*w.d* と PIB*w.d* 入力形式は、ネイティブな形式の整数を読み込むために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 入力形式は、動作環境に関係なくリトルエンディアン整数を読み込むために使用されます。

Intel および DEC の動作環境では、IB*w.d* と IBR*w.d* 入力形式は同じものです。

ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、IBR 入力形式を指定できます。ただし、これらの例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('0100'x,ibr2.);
y=input('0001'x,ibr2.);
```

SAS ステートメント	ビッグエンディアンプラットフォームでの結果	リトルエンディアンプラットフォームでの結果
put x=;	1	1
put y=;	256	256

関連項目:

入力形式:

- “IBw.d 入力形式” (278 ページ)

IEEE*w.d* 入力形式

IEEE 形式の浮動小数点値を読み込み、その値を 10 の *d* 乗で除算します。

カテゴリ: 数値

構文

IEEE*w.d*

説明

w
入力幅を指定します。

デフォルト: 8

範囲: 2–8

ヒント: *w* が 8 の場合、IEEE の倍精度浮動小数点数が読み込まれます。*w* が 5、6 または 7 の場合、相当するバイト数が切り捨てられたと仮定して、IEEE の倍精度浮動小数点数が読み込まれます。*w* が 4 の場合、IEEE の単精度浮動小数点数が読み込まれます。*w* が 3 の場合、1 バイトが切り捨てられたと仮定して、IEEE の単精度浮動小数点数が読み込まれます。

d
値を除算する 10 のべき乗を指定します。

詳細

IEEE*w.d* 入力形式は、IEEE 浮動小数点表現を使用する動作環境で使用します。また、IEEE*w.d* 入力形式を使用して、IEEE 浮動小数点数表現を使用する動作環境のプログラムによって作成されるファイルを読み込むこともできます。

通常プログラムは、単精度(4 バイト)または倍精度(8 バイト)で IEEE 値を生成します。切り捨ては、出力ファイルのスペースを削減する場合にのみプログラムによって実行されます。機械語命令では、浮動小数点数が 4 バイトまたは 8 バイトのいずれかである必要があります。IEEE*w.d* 入力形式では、他の長さも処理できるため、スペースの削減のために切り捨てられたデータを含むファイルからデータを読み込めます。

例

```
input test1 ieee4.;
input test2 ieee5.;
```

データ行*	結果
----+----1----+	
3F800000	1
3FF0000000	1

* データ行は、IEEE 形式で保存したバイナリ値を 16 進表現で表したものです。

最初の INPUT ステートメントは最初のデータ行を読み込み、2 番目の INPUT ステートメントは次のデータ行を読み込みます。

JULIAN*w*. 入力形式

yyddd または *yyyyddd* 形式のユリウス日付を読み込みます。

カテゴリ: 日付と時間

構文

JULIAN*w*.

説明

w
入力幅を指定します。
デフォルト: 5
範囲: 5–32

詳細

日付値は、次のうちいずれかの形式である必要があります。

- *yyddd*
- *yyyyddd*

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

dd または *ddd*
年の日を表す 01 から 365 の整数です。

ユリウス日付は、数字が連続して続く文字列であり、年値と日値の間のスペースにはゼロが埋め込まれます。

1582 年より前の年値を含むユリウス日付は、グレゴリオ日付に変換できません。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input julian_date julian7.;
```

データ行	結果*
----+----1	
12076	19068
2012076	19068

* 入力値は、2012 年の 76 日目、つまり 3 月 16 日に対応しています。

関連項目:

出力形式:

- “JULIANw. 出力形式” (107 ページ)

関数:

- “DATEJUL Function” in *SAS Functions and CALL Routines: Reference*
- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

MDYAMPM*w.d* 入力形式

mm-dd-yy<yy> hh:mm:ss.ss AM|PM 形式の日時値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)またはコロン(:)などの特殊文字で月、日、年は区切られます。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

要件 スペースで日付と時間を区切る必要があります。

注: デフォルトの時間は AM です。

構文

MDYAMPM*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 19
範囲: 8–40

d
秒値の小数点以下の桁数を指定します。小数点以下の桁数は、秒の端数を示します。この引数はオプションです。

デフォルト: 0

範囲: 0–39

詳細

MDYAMPMw.d 入力形式は、*mm-dd-yy<yy> hh:mm<:ss<.ss>>* <AM | PM>形式の日時値を読み込みます。

mm
月を表す 01 から 12 までの整数です。

dd
月の日を表す 01 から 31 の整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

hh
時間を表す 00 から 23 までの整数です。

mm
分を表す 00 から 59 までの整数です。

ss.ss
小数点以下の秒の端数を含む、00 から 59 の範囲の秒数です。

要件: 秒の端数が指定される場合、小数点が必須となり、その表記にはピリオドのみ使用できます。

AM | PM
00:01–12:00 (AM)または 12:01–12:00 (PM)のいずれかです。

- または :
日付要素と時間要素を区切るために使用するスラッシュ(/)、ハイフン(-)、コロン(:)、空白文字などの特殊文字のいずれか 1 つを表します。特殊文字は、日付要素間または時間要素間、日付と時間の間の区切り文字として使用できます。

比較

MDYAMPM*w.d* 入力形式は、オプションの区切り文字を含む *mm-dd-yy<yy> hh:mm:ss.ss* AM | PM 形式の日時値を読み込みます。日付と時間の間にはスペースが必要です。

DATETIME*w.d* 入力形式は、オプションの区切り文字を含む *dd-mmm-yy<yy> hh:mm:ss.ss* AM|PM 形式の日時値を読み込みます。日付と時間は特殊文字で区切られる場合もあります。

YMDDTTM*w.d* 入力形式は、必須の区切り文字を含む*<yy>yy-mm-dd/hh:mm:ss.ss* 形式の日時値を読み込みます。

例

```
input @1 dt mdyampm25.2.;
```

データ行	結果
09.15.2012 03:53:00 pm	1663343580

データ行	結果
09-15-12 3.53 pm	1663343580

関連項目:

入力形式:

- “DATETIMEw. 入力形式” (261 ページ)
- “YMDDTTMw.d 入力形式” (336 ページ)

MMDDYYw. 入力形式

mmddyy または *mmddyyyy* 形式の日付値を読み込みます。

カテゴリ: 日付と時間

構文

MMDDYY*w*.

説明

w
入力幅を指定します。
デフォルト: 6
範囲: 6–32

詳細

日付値は、次のうちいずれかの形式である必要があります。

- *mmddyy*
- *mmddyyyy*

mm
月を表す 01 から 12 までの整数です。

dd
月の日を表す 01 から 31 の整数です。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

月、日、年のフィールドは、空白や特殊文字で区切ることができます。ただし、区切り文字を使用する場合は、値のすべてのフィールドの間に挿入します。空白は日付の前後に挿入することもできます。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input calendar_date mmddyy8.;
```

データ行	結果
----+----1----+	
031612	19068
03/16/12	19068
03 16 12	19068
03162012	19068

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

MONYY*w.* 入力形式

mmmyy または *mmmyyyy* 形式の年月を読み込みます。

カテゴリ: 日付と時間

構文

MONYY*w.*

説明

w
入力幅を指定します。
デフォルト: 5
範囲: 5–32

詳細

日付値は、次のうちいずれかの形式である必要があります。

- *mmmyy*
- *mmmyyyy*

mmm
月名の最初の 3 文字になります。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

MONYY*w.*入力形式によって読み込まれる値は SAS 日付値となり、指定した月の第 1 日に対応します。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input month_and_year monyy7.;
```

データ行	結果
----+----1	
mar 12	19053
mar2012	19053

関連項目:

出力形式:

- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “MONYYw. 出力形式” (119 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DDMMYYw. 入力形式” (263 ページ)

- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

MSEC*w*. 入力形式

TIME MIC の値を読み込みます。

カテゴリ: 日付と時間

構文

MSEC*w*.

説明

w
入力幅を指定します。

要件 *w* は 8 である必要があります。IBM メインフレーム上の OS タイムマクロまたは STCK システム/370 命令がそれぞれ 8 バイト値を返すためです。

詳細

MSEC*w*.入力形式は、IBM メインフレーム動作環境によって生成される時間値を読み込み、時間値を SAS 時間値に変換します。

2 つの IBM メインフレームの TIME 値の差異をおよそマイクロ秒の精度で求めるには、MSEC*w*.入力形式を使用します。

比較

MSEC*w*.入力形式と TODSTAMP*w*.入力形式はともに IBM の TOD クロック値を読み込みますが、MSEC*w*.入力形式は時間値を変数に割り当て、TODSTAMP*w*.入力形式は日時値を割り当てます。

例

```
input btime msec8.;
```

データ行*	結果
0000EA044E65A000	62818.412122

* データ行は、8 バイトの TOD クロックのバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、午後 5:26:58.41 に対応する SAS 時間値です。

関連項目:

入力形式:

- “TODSTAMPw. 入力形式” (325 ページ)

NUMX*w.d* 入力形式

小数点をカンマとして数値を読み込みます。

カテゴリ: 数値

構文

NUMX*w.d*

説明

w
入力幅を指定します。

デフォルト: 12

範囲: 1–32

d
小数点以下の桁数を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。

範囲: 0–31

詳細

NUMX*w.d* 入力形式は数値を読み込み、カンマを小数点として解釈します。

比較

NUMX*w.d* 入力形式は、*wd* 入力形式とほぼ同じですが、小数点としてカンマを含む数値を読み込みます。

例

```
input @1 x numx10.;
```

データ行	結果
----+----1----+	
896,48	896.48
3064,1	3064.1
6489	6489

関連項目:

出力形式:

- “NUMXw.d 出力形式” (121 ページ)

- “w.d 出力形式” (159 ページ)

OCTAL*w.d* 入力形式

正の 8 進値を整数に変換します。

カテゴリ: 数値

構文

OCTAL*w.d*

説明

w
入力幅を指定します。
デフォルト: 3
範囲: 1–24

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 1–31
制限事項: *w* 値以上である必要があります。

詳細

入力には、0 から 7 までの数字のみを使用します。空白は挿入できません。
OCTAL*w.d* 入力形式では、先頭と末尾の空白は無視されます。

OCTAL*w.d* では、負の値を読み込むことはできません。すべての入力値は正数(符号なし)として処理されます。

例

```
input @1 value octal3.1;
```

データ行	結果
----+----1	
177	12.7

PD*w.d* 入力形式

IBM のパック 10 進形式で保存されているデータを読み込みます。

カテゴリ: 数値

参照項目: “PDw.d Informat: UNIX” in *SAS Companion for UNIX Environments*
“PDw.d Informat: Windows” in *SAS Companion for Windows*

“PDw.d Informat: z/OS” in *SAS Companion for z/OS*

構文

PD*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–16

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

PD*w.d* 入力形式は、多くのプログラムでは保存域の効率を考慮してデータをパック 10 進形式で書き出されるために使用されます。パック 10 進形式では、2 桁を 1 バイトに当てはめ、符号には 4 ビットを使用します。

注: 動作環境によってパック 10 進値の保存方法は異なります。ただし、PD*w.d* では、パック 10 進値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

PD*w.d* 出力形式は、欠損数値データを-0 として書き出します。PD*w.d* 入力形式で -0 が読み出されると、0 として保存されます。

比較

次の表は、各種プログラミング言語でのパック 10 進表記を比較したものです。

言語	表記
SAS	PD4.
COBOL	COMP-3 PIC S9(7)
IBM 370 アセンブラ	PL4
PL/I	FIXED DEC

例

例 1: パック 10 進データの読み込み

```
input @1 x pd4.;
```

データ行*	結果
`----+----1`	

データ行*	結果
0000128C	128

* データ行は、パック 10 進形式で保存したバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。

例 2: パック 10 進データを読み込んで SAS 日付値を作成する

```
input x: $hex10.;
mnth=input(x, pd5.);
date=input(put(mnth,8.),mmddyy6.);
```

データ行*	結果
----+----1	
012252010C	18621

* データ行は、IBM メインフレーム動作環境上のパック 10 進形式で保存したバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、2010 年 12 月 25 日に対応する SAS 日付値です。

PDJULG*w*. 入力形式

IBM で使用する 16 進の *yyyyddd*F 形式のパックユリウス日付値を読み込みます。

カテゴリ: 日付と時間

構文

PDJULG*w*.

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 4

詳細

PDJULG*w*.入力形式は、IBM で使用する *yyyyddd*F 形式のパックユリウス日付値を読み込みます。

yyyy
4 桁のグレゴリオ暦の年を 2 バイトで表したものです。

ddd
ユリウス日 1–365 (うるう年の場合は 1–366)に対応する 3 桁の整数を 1.5 バイトで表したものです。

F
2 進表現の 1 のみで構成される、値を正とする 0.5 バイトです。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input date pdjulg4.;
```

データ行	結果*
----+----1	
2012003F	18995

* SAS 日付値 18995 は、2012 年 1 月 3 日を表します。

関連項目:

出力形式:

- “JULDAYw. 出力形式” (106 ページ)
- “JULIANw. 出力形式” (107 ページ)
- “PDJULGw. 出力形式” (124 ページ)
- “PDJULIw. 出力形式” (126 ページ)

関数:

- “DATEJUL Function” in *SAS Functions and CALL Routines: Reference*
- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “JULIANw. 入力形式” (282 ページ)
- “PDJULIw. 入力形式” (293 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

PDJULIw. 入力形式

IBM で使用する 16 進の *ccyyddd*F 形式のパックユリウス日付を読み込みます。

カテゴリ: 日付と時間

構文

PDJULI*w.*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 4

詳細

PDJULI*w*.入力形式は、*ccyyddd*F 形式の IBM で使用するパックユリウス日付値を読み込みます。

cc
世紀を表す 2 桁の整数を 1 バイトで表したものです。

yy
年を表す 2 桁の整数を 1 バイトで表したものです。PDJULI*w* 入力形式は、正しい 4 桁のグレゴリオ暦の年を生成するために、1900 を 2 バイトの *ccyy* 値に加算して、1 バイトの世紀表現の調整を行います。この調整の結果、*ccyy* の値 0098 が 1998 に、値 0101 が 2001 に、値 0218 が 2118 になります。

ddd
ユリウス日 1–365 (うるう年の場合は 1–366)に対応する 3 桁の整数を 1.5 バイトで表したものです。

F
2 進表現の 1 のみで構成される、値を正とする 0.5 バイトです。

例

```
input date pdjuli4.;
```

データ行	結果*
----+----1	
0099001F	14245
0112015F	19007

* SAS 日付値 14245 は 1999 年 1 月 1 日、SAS 日付値 19007 は 2012 年 1 月 15 日になります。

関連項目:

出力形式:

- “JULDAYw. 出力形式” (106 ページ)
- “JULIANw. 出力形式” (107 ページ)
- “PDJULGw. 出力形式” (124 ページ)
- “PDJULIw. 出力形式” (126 ページ)

関数:

- “DATEJUL Function” in *SAS Functions and CALL Routines: Reference*
- “JULDATE Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “JULIANw. 入力形式” (282 ページ)
- “PDJULGw. 入力形式” (292 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

PDTIME*w*. 入力形式

SMF レコードと RMF レコードのパック 10 進の時間を読み込みます。

カテゴリ: 日付と時間

構文

PDTIME*w*.

説明

w
入力幅を指定します。

要件 *w* には 4 を指定します。RMF レコードと SMF レコードのパック 10 進の時間値に 4 バイトの情報が含まれています。

詳細

PDTIME*w*.入力形式は、IBM メインフレームシステムによって生成される SMF レコードと RMF レコードに含まれるパック 10 進の時間値を読み込み、その値を SAS 時間値に変換します。

パック 10 進の時間値を 16 進表記で表した一般的な形式は、0*hhmmss*F です。

0
0 のみを含む 0.5 バイトです。

hh
時間に対応する 2 桁を表す 1 バイトです。

mm
分に対応する 2 桁を表す 1 バイトです。

ss
秒に対応する 2 桁を表す 1 バイトです。

F
1 のみを含む 0.5 バイトです。

フィールドに 0 のみが含まれる場合、PDTIME*w*.では欠損値として処理します。

PDTIME*w*.を使用して、動作環境に関係なく IBM メインフレームで作成されたファイルからパック 10 進の時間値を読み込むことができます。

例

```
input begin pdtime4.;
```

データ行*	結果
0142225F	51745

* データ行は、パック 10 進形式で保存したバイナリ時間値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、午後 2:22:25 に対応する SAS 時間値です。

PERCENT*w.d* 入力形式

百分率を数値として読み込みます。

カテゴリ: 数値

構文

PERCENT*w.d*

説明

w
入力幅を指定します。

デフォルト: 6

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。

範囲: 0–31

詳細

PERCENT*w.d* 入力形式は、COMMA*w.d* 入力形式と同じ方法を使用して、入力データの数値部分を数字に変換します。入力フィールドでパーセント記号(%)が数字の後にある場合、PERCENT*w.d* はその数字を 100 で除算します。

例

```
input @1 x percent3. @4 y percent5.;
```

データ行	結果
----+----1	
1% (20%)	0.01 -0.2

PIB*w.d* 入力形式

正のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

参照項目: “PIBw.d Informat: UNIX” in *SAS Companion for UNIX Environments*
“PIBw.d Informat: Windows” in *SAS Companion for Windows*

構文

PIB*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

すべての値は、正として処理されます。PIB*w.d* では、正のバイナリ整数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

注: 動作環境によって正のバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

- 正のバイナリ整数値は、符号ビットが値の一部である点を除きバイナリ整数値と同じであり、常に正の整数になります。PIB*w.d* 入力形式はすべての値を正として処理し、値の一部として符号ビットを含めます。
- PIB*w.d* 入力形式は、幅が 1 の場合、1 バイトのコンテンツを 2 進表現した値を読み込みます。1 バイトのコンテンツを 2 進表現する値は、データに 16 進数の 80 と 16 進数の FF の間の値が含まれる、高位ビットが負の符号として誤って解釈される可能性がある場合に使用します。
- IB*w.d* と PIB*w.d* 入力形式は、ネイティブな形式の整数を読み込むために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 入力形式は、動作環境に関係なくリトルエンディアン整数を読み込むために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、PIB 入力形式を指定できます。ただし、これらの例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('0100'x,pib2.);
y=input('0001'x,pib2.);
```

SAS ステートメント	ビッグエンディアンプラットフォームでの結果	リトルエンディアンプラットフォームでの結果
put x=;	256	1
put y=;	1	256

関連項目:

入力形式:

- “PIBRw.d 入力形式” (298 ページ)

PIBR*w.d* 入力形式

Intel 形式と DEC 形式の正のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

構文

PIBR*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

すべての値は、正として処理されます。PIBR*w.d* は、Intel および DEC の動作環境で生成される正のバイナリ整数値を読み込みます。PIBR*w.d* を使用して、Intel または DEC の環境の正のバイナリ整数データを他の動作環境から読み込みます。PIBR*w.d* 入力形式によって、動作環境に関係なくデータを読み込むためのポータブルな実装が用意されます。

注: 動作環境によって正のバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

- 正のバイナリ整数値は、符号ビットが値の一部である点を除きバイナリ整数値と同じであり、常に正の整数になります。PIBR*w.d* 入力形式はすべての値を正として処理し、値の一部として符号ビットを含めます。
- PIBR*w.d* 入力形式は、幅が 1 の場合、1 バイトのコンテンツを 2 進表現した値を読み込みます。この値は、データに 16 進数の 80 と 16 進数の FF の間の値が含まれる、高位ビットが負の符号として誤って解釈される可能性がある場合に使用します。
- Intel および DEC の動作環境では、PIB*w.d* と PIBR*w.d* 入力形式は同じものです。
- IB*w.d* と PIB*w.d* 入力形式は、ネイティブな形式の整数を読み込むために使用されます。(ネイティブな形式により、同じ動作環境で作成される値の読み込みと書き出しが可能になります。) IBR*w.d* と PIBR*w.d* 入力形式は、動作環境に関係なくリトルエンディアン整数を読み込むために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、PIBR 入力形式を指定できます。ただし、これらの例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
 x=input('0100'x,pibr2.);
y=input('0001'x,pibr2.);
```

SAS ステートメント	ビッグエンディアンプラットフォームでの結果	リトルエンディアンプラットフォームでの結果
put x=;	1	1
put y=;	256	256

関連項目:

入力形式

- “PIBw.d 入力形式” (296 ページ)

PK*w.d* 入力形式

符号なしパック 10 進データを読み込みます。

カテゴリ: 数値

構文

PK*w.d*

説明

w

符号なしパック 10 進データのバイト数を指定します。1 バイトに 2 桁を含みます。

デフォルト: 1

範囲: 1–16

d

値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–10

詳細

符号なしのパック 10 進データは、1 バイトに 2 桁を含みます。

比較

PK*w.d* 入力形式は PD*w.d* 入力形式と同じですが、PK*w.d* ではフィールドの最終バイトの半分を値の符号としてではなく、値の一部として処理します。

例

```
input @1 x pk3.;
```

データ行*	結果
`----+----1`	
`001234`	`1234`

* データ行は、符号なしパック 10 進形式で保存したバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。

PUNCH.*d* 入力形式

カラムバイナリデータ行のパンチの有無を読み込みます。

カテゴリ: カラムバイナリ

構文

PUNCH*.d*

説明

d

読み込むカードカラムの行を指定します。

範囲: 1–12

詳細

カラムバイナリデータストレージは、80 項目を超えるデータを 1 つの"仮想"パンチカードに保存できるように、データを圧縮します。

この入力形式は、現在のカードカラムの行 *d* にパンチがある場合は変数に値 1 を割り当てます。現在のカードカラムの行 *d* にパンチがない場合は 0 を割り当てます。PUNCH.*d* では、フィールドの読み込みの終了後、ポインタは次のカラムに移動しません。

例

データ行 *	SAS ステートメント	結果
12-7-8	input x punch.12	1
	input x punch.11	0
	input x punch0.7	1

* データ行は“仮想”パンチカードコードです。サンプルデータのパンチカードカラムでは、行 12、行 7、行 8 がパンチされています。

関連項目:

- “How to Read Column-Binary Data” in Chapter 19 of *SAS Language Reference: Concepts*

入力形式:

- “$CBw. 入力形式” (221 ページ)
- “CBw.d 入力形式” (257 ページ)
- “ROWw.d 入力形式” (305 ページ)

RB*w.d* 入力形式

バイナリ実数(浮動小数点)表記で格納されている数値データを読み込みます。

カテゴリ: 数値

参照項目: “RBw.d Informat: UNIX” in *SAS Companion for UNIX Environments*
“RBw.d Informat: Windows” in *SAS Companion for Windows*
“RBw.d Informat: z/OS” in *SAS Companion for z/OS*

構文

RB*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 2–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–10

詳細

注: 動作環境によってバイナリ実数値の保存方法は異なります。ただし、RB*w.d* 入力形式では、バイナリ実数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

比較

次の表は、各種プログラミング言語でのバイナリ実数表記名を比較したものです。

	バイナリ実数表記	
言語	**4 バイト**	**8 バイト**
SAS	RB4.	RB8.
Fortran	REAL*4	REAL*8
C	float	double
IBM 370 アセンブラ	F	D
PL/I	FLOAT BIN(21)	FLOAT BIN(53)

注意:

IEEE の標準基準に準拠した環境からのバイナリ実数情報の読み込みに RBw.d 入力形式を使用すると、浮動小数点数は実際の 4 バイトの浮動小数点数(単精度)ではなく、8 バイトの数字(倍精度)に切り捨てられます。

例

```
input @1 x rb8.;
```

データ行*	結果
`----+----1`	
`4280000000000000`	`128`

* データ行は、IBM メインフレーム動作環境上のバイナリ実数(浮動小数点数)を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。

関連項目:

入力形式:

- “IEEEw.d 入力形式” (281 ページ)

RMFDURw. 入力形式

RMF レコードの継続間隔を読み込みます。

カテゴリ: 日付と時間

構文

RMFDUR*w.*

説明

w
入力幅を指定します。

要件 *w* には 4 を指定します。RMF レコードのパック 10 進の継続間隔値に 4 バイトの情報が含まれています。

詳細

RMFDUR*w.*入力形式では、IBM メインフレームシステムによってパック 10 進データとして生成される RMF レコードの RMF 測定継続間隔を読み込み、SAS 時間値に変換します。

RMF レコードの継続間隔データを 16 進表記で表した一般的な形式は、*mmssttt*F です。

mm
分に対応する 2 桁を表す 1 バイトです。

ss
秒に対応する 2 桁を表す 1 バイトです。

ttt
1000 分の 1 秒に対応する 3 桁を表す 1.5 バイトです。

F
2 進表現の 1 のみで構成される、値を正とする 0.5 バイトです。

フィールドに 10 進パックデータが含まれていない場合、RMFDUR*w.*は欠損値を返します。

比較

- RMFDUR*w.*入力形式と RMFSTAMP*w.*入力形式はどちらも IBM メインフレームシステムによって生成される RMF レコードからパック 10 進情報を読み込みます。
- RMFDUR*w.*入力形式は継続間隔データを読み込み、時間値に変換します。
- RMFSTAMP*w.*入力形式は TOD データを読み込み、日時値に変換します。

例

```
input dura rmfdur4.;
```

データ行 *	結果
----+----1	
3552226F	2152.226

* データ行は、RMF レコード形式のパック 10 進で保存したバイナリ継続間隔値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、00:35:52.226 に対応する SAS 時間値です。

関連項目:

入力形式:

- “RMFSTAMPw. 入力形式” (304 ページ)
- “SMFSTAMPw. 入力形式” (322 ページ)

RMFSTAMP*w.* 入力形式

RMF レコードの日付フィールドと時間フィールドを読み込みます。

カテゴリ: 日付と時間

構文

RMFSTAMP*w.*

説明

w
入力幅を指定します。

要件 *w* には、8 を指定します。RMF レコード内のパック 10 進の時間値と日付値に 8 バイトの情報が含まれているためです。4 バイトの時間データの後に 4 バイトの日付データが続きます。

詳細

RMFSTAMP*w.*入力形式は、IBM メインフレームシステムによって生成される RMF レコードのパック 10 進の時間値と日付値を読み込み、それらの値を SAS 日時値に変換します。

RMF レコードの日時情報を 16 進表記で表した一般的な形式は、0*hhmmss*F*ccyyddd*F です。

0
バイナリ 0 のみを含む 0.5 バイトです。

hh
その日の時間に対応する 2 桁を表す 1 バイトです。

mm
分に対応する 2 桁を表す 1 バイトです。

ss
秒に対応する 2 桁を表す 1 バイトです。

cc
世紀に対応する 2 桁を表す 1 バイトです。

yy
世紀に対応する 2 桁を表す 1 バイトです。

ddd
年の日に対応する 3 桁を含む 1.5 バイトです。

F
バイナリ 1 のみを含む 0.5 バイトです。

世紀インジケータ 00 は 1900、01 は 2000、02 は 2100 に対応します。

RMFSTAMP*w.*を使用して、動作環境に関係なく IBM メインフレームで作成されたファイルからパック 10 進の時間値と日付値を読み込むことができます。

比較

RMFSTAMP*w.*入力形式と PDTIME*w.*入力形式はどちらも RMF レコードからパック 10 進値を読み込みます。RMFSTAMP*w.*入力形式は時間値と日付値を読み込み、SAS 日時値に変換します。PDTIME*w.*入力形式は時間値のみを読み込み、SAS 時間値に変換します。

例

```
input begin: $hex16.;
y=input(begin, rmfstamp8.);
```

データ行 *	結果
----+----1----+----2	
0142225F2612200F	80550512545

* データ行は、RMF レコード形式のパック 10 進で保存したバイナリ日時値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、2012 年 7 月 18 日午後 2:22.25 に対応する SAS 日時値です。

ROW*w.d* 入力形式

カードカラムに記録されているカラムバイナリフィールドを読み込みます。

カテゴリ: カラムバイナリ

構文

ROW*w.d*

説明

w

フィールドが始まる行を指定します。

範囲: 0–12

d

フィールドの行の長さを指定します。

デフォルト: 1

範囲: 1–25

詳細

カラムバイナリデータストレージは、80 項目を超えるデータを 1 つの"仮想"パンチカードに保存できるように、データを圧縮します。

ROW*w.d* 入力形式は、フィールドのパンチの相対位置を数値変数に割り当てます。

指定したフィールドに複数のパンチがある場合、ROW*w.d* は変数に欠損値を割り当て、自動変数_ERROR_を 1 に設定します。フィールドにパンチがない場合、ROW*w.d* は変数に欠損値を割り当てます。

ROW*w.d* は、すべてのカラムのフィールドを読み込むことができます。新しいカラムの 12 行目まで読み込み、次に残りの行を読み込みます。ROW*w.d* では、フィールドの読み込みの終了後、ポインタは次の行に移動します。

例

```
input x row5.3
input x row7.1
input x row5.2
input x row3.5
```

データ行 *	結果
----+----1	
00	
04	3
	1
	.
	5

* データ行はカラムバイナリの 16 進表現です。サンプルデータの“仮想”パンチカードカラムでは、行 7 がパンチされています。2 進表現は、0000 0000 0000 0100 となります。

関連項目:

- “How to Read Column-Binary Data” in Chapter 19 of *SAS Language Reference: Concepts*

入力形式:

- “$CBw. 入力形式” (221 ページ)
- “CBw.d 入力形式” (257 ページ)
- “PUNCH.d 入力形式” (300 ページ)

S370FF*w.d* 入力形式

EBCDIC 形式の数値データを読み込みます。

カテゴリ: 数値

構文

S370FF*w.d*

説明

w

入力幅を指定します。

デフォルト: 12

範囲: 1–32

d

値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–31

詳細

S370FF*w.d* 入力形式は、EBCDIC で表されている数値データを読み込み、データをネイティブ形式に変換します。EBCDIC がネイティブな形式の場合、S370FF*w.d* は変換を実行しません。

S370FF*w.d* は、1 桁 1 バイトで表される EBCDIC 数値を読み込みます。S370FF*w.d* を使用して、IBM メインフレームファイルの数値データを他の動作環境で読み込みます。

S370FF*w.d* は、フィールド内の数値はその位置にかかわらず読み込みます。数値の前後に挿入されている EBCDIC 空白による影響はありません。値が負の場合、EBCDIC マイナス記号を値の直前に置く必要があります。S370FF*w.d* は、EBCDIC 小数点を含む値と科学表記の値を読み込みます。1 つの EBCDIC ピリオドは、欠損値として扱います。

比較

S370FF*w.d* 入力形式は、$EBCDIC*w.d* 入力形式が実行する文字データへの役割と同じ役割を数値データに対して実行します。つまり、IBM メインフレームシステム上では、S370FF*w.d* は、*w.d* 標準入力形式と同じ処理を行います。それ以外のシステムでは、S370FF*w.d* を使用することは、*w.d* 標準入力形式だけでなく$EBCDIC*w.d* も使用することに相当します。

例

```
input @1 x s370ff3.;
```

データ行*	結果
----+----1	
F1F2F3	123
F2F4F0	240

* データ行は、文字コードを 16 進表現で表したものです。16 進の 2 文字がバイナリデータの 1 バイトに対応します。1 バイトが 1 文字値に対応します。

S370FIB*w.d* 入力形式

負の値を含む、IBM メインフレーム形式のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

構文

S370FIB*w.d*

説明

w
入力幅を指定します。

デフォルト: 4

範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–10

詳細

S370FIB*w.d* 入力形式は、2 の補数で表される負の値を含む、IBM メインフレーム形式で保存されるバイナリ整数(固定小数点)値を読み込みます。S370FIB*w.d* では、バイナリ整数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

S370FIB*w.d* を使用して、IBM メインフレーム形式で作成されるバイナリ整数データを他の動作環境で読み込みます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

- SAS を IBM メインフレームで使用する場合、S370FIB*w.d* と IB*w.d* は同じものです。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を読み込むために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、S370FIB 入力形式を指定できます。ただし、この例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('0080'x,s370fib2.);
```

SAS ステートメント	結果
`put x=;`	`128`

関連項目:

入力形式

- “S370FIBUw.d 入力形式” (309 ページ)
- “S370FPIBw.d 入力形式” (312 ページ)

S370FIBU*w.d* 入力形式

IBM メインフレーム形式の符号なしバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

構文

S370FIBU*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれている場合も *d* 値が使用されます。この引数はオプションです。
範囲: 0–10

詳細

S370FIBU*w.d* 入力形式は、2 の補数で表される負の値を含む、IBM メインフレーム形式で保存される符号なしバイナリ整数(固定小数点)値を読み込みます。符号なしバイナリ整数値はバイナリ整数値と同じですが、すべての値が正として処理される点が異なります。S370FIBU*w.d* では、バイナリ整数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

S370FIBU*w.d* を使用して、IBM メインフレーム形式で作成される符号なしバイナリ整数データを他の動作環境で読み込みます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

- S370FIBU*w.d* 入力形式は、COBOL の表記 PIC 9(*n*) BINARY と同じです。*n* は桁数です。
- S370FIBU*w.d* 入力形式と S370FPIB*w.d* 入力形式は同じものです。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を読み込むために使用されます。

ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、S370FIBU 入力形式を指定できます。ただし、これらの例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('7F'x,s370fibu1.);
y=input('F6'x,s370fibu1.);
```

SAS ステートメント	結果
put x=;	127
put y=;	246

関連項目:

入力形式:

- “S370FIBw.d 入力形式” (307 ページ)
- “S370FPIBw.d 入力形式” (312 ページ)

S370FPD*w.d* 入力形式

IBM メインフレーム形式のパック 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FPD*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–16

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

パック 10 進データでは、1 バイトに 2 桁が含まれますが、入力フィールドの 1 桁のみが符号を表します。最後のバイトの下位 4 ビットは符号を表します。C または F の場合は正数、D の場合は負数になります。

S370FPD*w.d* を使用して、IBM メインフレームファイルのパック 10 進データを他の動作環境で読み込みます。

比較

- SAS を IBM メインフレームで使用する場合、S370FPD*w.d* 入力形式と PD*w.d* 入力形式は同じものです。
- 次の表は、各種プログラミング言語でのパック 10 進表記を比較したものです。

言語	パック 10 進表記
SAS	S370FPD4.
PL/I	FIXED DEC(7,0)
COBOL	COMP-3 PIC 9(7)
アセンブラ	PL4

S370FPDU*w.d* 入力形式

IBM メインフレーム形式の符号なしパック 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FPDU*w.d*

説明

w

入力幅を指定します。

デフォルト: 1

範囲: 1–16

d

値を除算する 10 のべき乗を指定します。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

パック 10 進データは、1 バイトに 2 桁を含みます。最後のバイトの下位 4 ビットは、符号付きパックデータの符号を示します。符号なしパックデータの場合、常に F になります。

S370FPDU*w.d* を使用して、IBM メインフレームファイルの符号なしパック 10 進データを他の動作環境で読み込みます。

比較

- S370FPDU*w.d* 入力形式は、S370FPD*w.d* 入力形式と似ていますが、S370FPDU*w.d* 入力形式では F 以外の符号は拒否されます。
- S370FPDU*w.d* 入力形式は、COBOL の表記 PIC 9 (*n*) PACKED-DECIMAL と同じです。*n* 値は桁数です。

例

```
input @1 x s370fpdu3.;
```

データ行*	結果
----+----1	
12345F	12345

* データ行は、パック 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

S370FPIB*w.d* 入力形式

IBM メインフレーム形式の正のバイナリ整数(固定小数点)値を読み込みます。

カテゴリ: 数値

構文

S370FPIB*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 1–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–10

詳細

正のバイナリ整数値はバイナリ整数値と同じですが、すべての値が正として処理される点が異なります。S370FPIB*w.d* では、バイナリ整数値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

S370FPIB*w.d* を使用して、IBM メインフレーム形式で作成される正のバイナリ整数データを他の動作環境で読み込みます。

注: 動作環境によってバイナリ整数値の保存方法は異なります。この概念をバイトオーダリングといいます。バイトオーダリングの詳細については、“ビッグエンディアンプラットフォームとリトルエンディアンプラットフォーム上でのバイナリ整数データのバイトオーダリング” (197 ページ)を参照してください。

比較

- SAS を IBM メインフレームで使用する場合、S370FPIB*w.d* と PIB*w.d* は同じものです。
- S370FPIB*w.d*、S370FIBU*w.d* および S370FIB*w.d* は、動作環境に関係なくビッグエンディアン整数を読み込むために使用されます。

  ビッグエンディアン整数およびリトルエンディアン整数に関連して使用する入力形式の種類の表については、表 3.1 (198 ページ)を参照してください。

  各種プログラミング言語でのバイナリ整数表記の比較表については、“バイナリ整数表記と各種プログラミング言語” (8 ページ)を参照してください。

例

INPUT ステートメントを使用して、S370FPIB 入力形式を指定できます。ただし、この例では入力形式を INPUT 関数と使用します。バイナリ入力値は 16 進リテラルを使用して表されます。

```
x=input('0100'x,s370fpib2.);
```

SAS ステートメント	結果
`put x=4;`	`256`

関連項目:

入力形式:

- “S370FIBw.d 入力形式” (307 ページ)
- “S370FIBUw.d 入力形式” (309 ページ)

S370FRB*w.d* 入力形式

IBM メインフレーム形式のバイナリ実数(浮動小数点)を読み込みます。

カテゴリ: 数値

構文

S370FRB*w.d*

説明

w
入力幅を指定します。
デフォルト: 6

範囲: 2–8

d

値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–10

詳細

バイナリ実数値は、値を表す仮数部と値の大きさを表す指数部で構成されています。

S370FRB*w.d* を使用して、IBM メインフレームフィルのバイナリ実数データを他の動作環境で読み込みます。

比較

- SAS を IBM メインフレームで使用する場合、S370FRB*w.d* と RB*w.d* は同じものです。
- 次の表は、各種プログラミング言語でのバイナリ実数表記を示したものです。

	バイナリ実数表記	
言語	**4 バイト**	**8 バイト**
SAS	S370FRB4.	S370FRB8.
PL/I	FLOAT BIN(21)	FLOAT BIN(53)
Fortran	REAL*4	REAL*8
COBOL	COMP-1	COMP-2
アセンブラ	E	D
C	float	double

関連項目:

入力形式:

- “RBw.d 入力形式” (301 ページ)

S370FZD*w.d* 入力形式

IBM メインフレーム形式のゾーン 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FZD*w.d*

説明

w
入力幅を指定します。

デフォルト: 8

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。

デフォルト: 0

範囲: 0–31

詳細

ゾーン 10 進データは、標準 10 進データと同じように 1 桁には 1 バイトが必要です。ただし、値の符号は最後の桁とともに最後のバイトに保存されます。

S370FZD*w.d* を使用して、IBM メインフレームファイルのゾーン 10 進データを他の動作環境で読み込みます。

比較

- SAS を IBM メインフレームで使用する場合、S370FZD*w.d* と ZD*w.d* は同じものです。
- 次の表は、各種プログラミング言語での 10 進表記を示したものです。

言語	ゾーン 10 進表記
SAS	S370FZD3.
PL/I	PICTURE'99T'
COBOL	PIC S9(3) DISPLAY
アセンブラ	ZL3

例

```
input @1 x s370fzd3.;
```

データ行*	結果
----+----1	
F1F2C3	123
F1F2D3	-123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

関連項目:

入力形式:

- “ZDw.d 入力形式” (342 ページ)

S370FZDB*w.d* 入力形式

空白を含むゾーン 10 進データを読み込みます。

カテゴリ: 数値

参照項目: “ZDBw.d Informat: z/OS” in *SAS Companion for z/OS*

構文

S370FZBD*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

S370FZDB*w.d* 入力形式を使用して、IBM メインフレームファイルのゾーン 10 進データを他の動作環境で読み込みます。

例

```
input @1 x s370fzdb8.;
```

データ行*	結果
----+----1	
40404040F14040C0	1000
4040404040F1F2D3	–123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

S370FZDL*w.d* 入力形式

IBM メインフレーム形式の前符号付きゾーン 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FZDL*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

S370FZDL*w.d* を使用して、IBM メインフレームファイルのゾーン 10 進データを他の動作環境で読み込みます。

比較

- 前符号付きゾーン 10 進データは、標準ゾーン 10 進データと似ていますが、最初のバイトに最初の桁と一緒に値の符号が保存される点が異なります。
- S370FZDL*w.d* 入力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN LEADING と同じです。*n* 値は桁数です。

例

```
input @1 x s370fzdl3.;
```

データ行*	結果
----+----1	
C1F2F3	123
D1F2F3	-123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

S370FZDS*w.d* 入力形式

IBM メインフレーム形式の分離した前符号付きゾーン 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FZDS*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 2–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

S370FZDS*w.d* を使用して、IBM メインフレームファイルのゾーン 10 進データを他の動作環境で読み込みます。

比較

- 分離した前符号付きゾーン 10 進データは、標準のゾーン 10 進データと似ていますが、値の符号は最初のバイトに保存され、値の最初の桁は 2 バイト目に保存される点が異なります。
- S370FZDS*w.d* 入力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN LEADING SEPARATE と同じです。*n* 値は桁数です。

例

```
input @1 x s370fzds4.;
```

データ行*	結果
----+----1	
4EF1F2F3	123
60F1F2F3	-123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

S370FZDT*w.d* 入力形式

IBM メインフレーム形式の分離した後符号付きゾーン 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FZDT*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 2–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

S370FZDT*w.d* を使用して、IBM メインフレームファイルのゾーン 10 進データを他の動作環境で読み込みます。

比較

- 分離した後符号付きゾーン 10 進データは、分離した前符号付きゾーン 10 進データと似ていますが、値の符号が最後のバイトに保存される点が異なります。
- S370FZDT*w.d* 入力形式は、COBOL の表記 PIC S9 (*n*) DISPLAY SIGN TRAILING SEPARATE と同じです。*n* 値は桁数です。

例

```
input @1 x s370fzdt4.;
```

データ行*	結果
----+----1	
F1F2F34E	123
F1F2F360	-123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

S370FZDU*w.d* 入力形式

IBM メインフレーム形式の符号なしゾーン 10 進データを読み込みます。

カテゴリ: 数値

構文

S370FZDU*w.d*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
デフォルト: 0
範囲: 0–31

詳細

S370FZDU*w.d* を使用して、IBM メインフレームファイルの符号なしゾーン 10 進データを他の動作環境で読み込みます。

比較

- S370FZDU*w.d* 入力形式は、S370FZD*w.d* 入力形式と似ていますが、S370FZDU*w.d* 入力形式では F 以外の符号は拒否されます。
- S370FZDU*w.d* 入力形式は、COBOL の表記 PIC 9 (*n*) DISPLAY と同じです。*n* 値は桁数です。

例

```
input @1 x s370fzdu3.;
```

データ行*	結果
----+----1	
F1F2F3	123

* データ行は、IBM メインフレーム動作環境上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。16 進数の 2 桁がバイナリデータの 1 バイトに対応します。1 バイトは、入力フィールドの 1 カラムに対応します。

SHRSTAMP*w*. 入力形式

SHR レコードの日付値と時間値を読み込みます。

カテゴリ: 日付と時間

構文

SHRSTAMP*w*.

説明

w
入力幅を指定します。

要件 *w* には、8 を指定します。SHR レコード内のパック 10 進の日付値と時間値には 8 バイトの情報が含まれているためです。4 バイトの日付データの後に 4 バイトの時間データが続きます。

詳細

SHRSTAMP*w*.入力形式は、IBM メインフレーム環境によって生成される SHR レコードのパック 10 進の日付値と時間値を読み込み、それらの値を SAS 日時値に変換します。

SHR レコードの日時情報を 16 進表記で表した一般的な形式は、*ccyyddd*F*hhmmssth* です。

ccyy
年を 2 バイトで表したものです。*cc* は、世紀を表す 2 桁の整数を 1 バイトで表したものです。*yy* は、年に対応する 2 桁を 1 バイトで表したものです。

cc は、世紀インジケータです。たとえば、00 は 19yy、01 は 20yy、02 は 21yy のように表します。16 進表記の年値 0115 は、年 2015 と同じです。

ddd
年の日に対応する 3 桁を含む 1.5 バイトです。

F
バイナリ 1 のみを含む 0.5 バイトです。

hh
その日の時間に対応する 2 桁を表す 1 バイトです。

mm
分に対応する 2 桁を表す 1 バイトです。

ss
秒に対応する 2 桁を表す 1 バイトです。

th
秒の 100 分の 1 に対応する 2 桁を表す 1 バイトです。

SHRSTAMP*w*.入力形式を使用して、動作環境に関係なく IBM メインフレームで作成されたファイルからパック 10 進の日付値と時間値を読み込むことができます。

例

```
input begin: $hex16.;
y=input(begin, shrstamp8.);
```

データ行*	結果
----+----1----+----2	
0110239F12403576	1598532035.8

* データ行は、SHR レコード形式で保存したパック 10 進の日付値と時間値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、2010 年 8 月 27 日、12:40:36 に対応する SAS 日時値です。

SMFSTAMP*w.* 入力形式

SMF レコードの日時値を読み込みます。

カテゴリ: 日付と時間

構文

SMFSTAMP*w.*

説明

w
入力幅を指定します。

要件 *w* には、8 を指定します。SMF レコード内の時間値と日付値には 8 バイトの情報が含まれているためです。4 バイトの時間データの後に 4 バイトの日付データが続きます。

ヒント: SMF レコードの時間部分は、4 バイトのバイナリ整数です。これは午前 0 時からの 100 分の 1 秒単位の経過時間を表しています。

詳細

SMFSTAMP*w.*入力形式は、IBM メインフレームシステムによって生成される SMF レコードのバイナリ整数の時間値とパック 10 進の日付値を読み込み、それらの値を SAS 日時値に変換します。

SMF レコードの日付部分を 16 進表記で表した形式は、*ccyydddF* です。

cc
世紀に対応する 2 桁を表す 1 バイトです。

yy
世紀に対応する 2 桁を表す 1 バイトです。

ddd
年の日に対応する 3 桁を含む 1.5 バイトです。

F
バイナリ 1 のみを含む 0.5 バイトです。

SMFSTAMP*w.*入力形式を使用して、動作環境に関係なく IBM メインフレームで作成されたファイルからバイナリ整数の時間値とパック 10 進の日付値を読み込みます。

例

```
input begin: $hex16.;
y=input(begin, smfstamp8.);
```

データ行*	結果
----+----1----+----2	
0058DC0C0108200F	1532016635

* データ行は、SMF レコード形式で保存したバイナリの時間値と日付値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、2008 年 7 月 18 日午後 4:10:35 に対応する SAS 日時値です。

STIMERw. 入力形式

時間値を読み込み、読み込んだ値が時間、分、秒であるかを識別します。STIMER システムオプションの出力を読み込みます。

カテゴリ: 日付と時間

構文

STIMER*w.*

説明

w
入力幅を指定します。

詳細

STIMER 入力形式は、STIMER システムオプションが SAS ログに書き出すパフォーマンス統計量を読み込みます。

この入力形式は時間値を読み込み、値が時間、分または秒であるかどうかを小数点とコロンに基づいて次のように識別します。

- コロンがない場合、値は秒数とします。
- コロンが 1 つある場合、コロンの前の値は分数とします。コロンの後の値は秒数とします。
- コロンが 2 つある場合、時、分、秒の順に時間が示されているとします。

いずれの場合でも、結果は SAS 時間値となります。

STIMER の入力値は、次のうちいずれかの形式である必要があります。

- *ss*
- *ss.ss*
- *mm:ss*
- *mm:ss.ss*
- *hh:mm:ss*

- *hh:mm:ss.ss*

ss
秒数を表す整数です。

mm
分数を表す整数です。

hh
時間数を表す整数です。

TIME*w.* 入力形式

hh:mm:ss.ss 形式の時、分、秒を読み込みます。コロン(:)やピリオド(.)などの特殊文字が時、分、秒を区切るために使用されます。

カテゴリ: 日付と時間

構文

TIME*w.*

説明

w
入力幅を指定します。
デフォルト: 8
範囲: 5–32

詳細

TIME*w.*入力形式は、*hh:mm:ss*<.*ss*> <AM | PM>形式の時間値を読み込みます。

hh
時間数を表す整数です。

:
時間、分、秒を区切る特殊文字を表します。

mm
分を表す 00 から 59 までの整数です。

ss<.*ss*>
秒数を表す整数です。必要に応じて、10 分の 1 秒も表せます。秒と 10 分の 1 秒は、常にピリオドで区切る必要があります。

AM | PM
AM は、12:00 (午前 0 時)から午前 11:59 までの時間を示します。PM は、12:00 (午後 0 時)から午後 11:59 までの時間を示します。

hh、*mm* および *ss* を特殊文字で区切ります。ピリオドが特殊文字として使用される場合、時間は時、分、秒の順に解釈されます。たとえば、23.22 は 23 時間と 22 分で、23 分と 22 秒でも 23 秒と 220 ミリ秒でもありません。

秒の値を入力しない場合、SAS は値が 0 であるとします。

保存値は、時間値の秒数合計です。

例

```
input begin time10.;
```

データ行	結果	TIMEw を適用
----+----1		
12.56	46560	12:56:00
120:120	439200	122:00:00
1:13 pm	47580	13:13:00

関連項目:

出力形式:

- “HHMMw.d 出力形式” (100 ページ)
- “HOURw.d 出力形式” (101 ページ)
- “MMSSw.d 出力形式” (113 ページ)
- “TIMEw.d 出力形式” (152 ページ)

関数:

- “HOUR Function” in *SAS Functions and CALL Routines: Reference*
- “MINUTE Function” in *SAS Functions and CALL Routines: Reference*
- “SECOND Function” in *SAS Functions and CALL Routines: Reference*
- “TIME Function” in *SAS Functions and CALL Routines: Reference*

TODSTAMPw. 入力形式

8 バイトの TOD スタンプを読み込みます。

カテゴリ: 日付と時間

構文

TODSTAMP*w*.

説明

w
入力幅を指定します。

要件 *w* は 8 である必要があります。IBM メインフレーム上の OS タイムマクロまたは STCK 命令がそれぞれ 8 バイト値を返すためです。

詳細

TODSTAMP*w.*入力形式は、IBM メインフレームオペレーティングシステムによって生成される TOD クロックを読み込み、そのクロック値を SAS 日時値に変換します。

TOD 値がすべて 0 の場合、TODSTAMP*w.*は欠損値を返します。

TODSTAMP*w.*を使用して、IBM メインフレームによって生成された TOD 値を他の動作環境で読み込みます。

例

```
input btime: $hex16.;
y=input(btime, todstamp8.);
```

データ行*	結果
----+----1----+----2	
B591183D5FB80000	1300786905

* データ行は、8 バイトの TOD クロックのバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、2001 年 3 月 21 日 09:41:45 に対応する SAS 日時値です。

TRAILSGN*w.* 入力形式

後置のプラス符号(+)とマイナス符号(–)を読み込みます。

カテゴリ: 数値

構文

TRAILSGN*w.*

説明

w
入力幅を指定します。

デフォルト: 6

範囲: 1–32

詳細

データに小数点が含まれている場合、TRAILSGN 入力形式は入力データ内の小数点以下桁数を保持します。データにカンマが含まれている場合は。カンマを無視して値を読み込みます。

例

```
input x trailsgn8.;
```

データ行	結果
----+----1	
1	1
1,000	1000
1+	1
1-	-1
1.2	1.2
1.2+	1.2
1.2-	-1.2

TU*w*. 入力形式

タイマーユニットを読み込みます。

カテゴリ: 日付と時間

構文

TU*w*.

説明

w
入力幅を指定します。
要件 *w* には、4 を指定します。OS タイムマクロが 4 バイトの値を返すためです。

詳細

TU*w*.入力形式は、IBM メインフレーム動作環境によって生成されるタイマーユニット値を読み込み、それらの値を SAS 時間値に変換します。

1 秒は 38,400 ソフトウェアタイマーユニットです。タイマーユニット値の下位ビットは、約 26.041667 マイクロセカンドを表します。

TU*w*.入力形式を使用して、IBM メインフレームによって生成されたタイマーユニットを他の動作環境で読み込みます。

例

```
input btime tu4.;
```

データ行*	結果
----+----1	

データ行*	結果
8FC7A9BC	62818.411563

* データ行は、4 バイトのバイナリタイマーユニット値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。結果は、午後 5:26:58.41 に対応する SAS 時間値です。

VAXRB*w.d* 入力形式

VMS 形式のバイナリ実数(浮動小数点)データを読み込みます。

カテゴリ: 数値

構文

VAXRB*w.d*

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 2–8

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 0–10

詳細

VAXRB*w.d* 入力形式を使用して、VMS ファイルの浮動小数点データを他の動作環境で読み込みます。

比較

VMS で実行している SAS を使用する場合、VAXRB*w.d* と RB*w.d* 入力形式は同じものです。

関連項目:

入力形式:

- “RBw.d 入力形式” (301 ページ)

VMSZN*w.d* 入力形式

VMS および MicroFocus COBOL ゾーン数値データを読み込みます。

カテゴリ: 数値

構文

VMSZN*w.d*

必須引数

w
出力フィールドの幅を指定します。
デフォルト: 1
範囲: 1–32

d
数値の小数点以下の桁数を指定します。この引数はオプションです。

詳細

VMSZN*w.d* 入力形式は、ZD*w.d* 入力形式とほぼ同じです。どちらも桁を ASCII で表現した文字列を読み込みます。最後の桁は、値の最後の桁の大きさと値の符号を示す特殊文字になります。VMSZN*w.d* 入力形式と ZD*w.d* 入力形式の相違点は、最後の桁に使用される特殊文字です。次の表は、VMSZN*w.d* 入力形式によって使用される特殊文字を示したものです。

表現する桁	特殊文字	表現する桁	特殊文字
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

VMSZN*w.d* 入力形式を使用して読み込まれるデータは、ASCII 文字列です。

例

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

データ行	結果
`----+----1`	

データ行	結果
1234	1234
123t	-1234

関連項目:

出力形式:

- “VMSZNw.d 出力形式” (158 ページ)

入力形式:

- “ZDw.d 入力形式” (342 ページ)

w.d 入力形式

標準数値データを読み込みます。

カテゴリ: 数値

別名: BEST*w.d*、D*w.d*、E*w.d*、F*w.d*

構文

w.d

説明

w
入力幅を指定します。
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。データに小数点が含まれる場合、*d* 値は無視されます。この引数はオプションです。
範囲: 0–31

詳細

w.d 入力形式は、フィールド内の数値はその位置にかかわらず読み込みます。数値の前後に挿入されている空白による影響はありません。マイナス符号は、空白を区切りとして挿入せずに、負の値の直前に置く必要があります。*w.d* 入力形式は、小数点を含む値と科学 E 表記の値を読み込みます。1 つのピリオドは、欠損値として扱います。

比較

- *w.d* 入力形式は、BZ*w.d* 入力形式と同じですが、*w.d* 入力形式は数値の後の空白を無視します。数値の後の空白を 0 として読み込むには、BZ*w.d* 入力形式を使用します。
- *w.d* 入力形式は、科学 E 表記の値を E*w.d* 入力形式と同じように読み込みます。

例

```
input @1 x 6. @10 y 6.2;
put x @7 y;
```

データ行	結果
----+----1----+	----+----1----+
23 2300	23 23
23 2300	23 0
23 -2300	23 -23
23.0 23.	23 23
2.3E1 2.3	23 2.3
-23 0	-23 .

WEEKU*w*. 入力形式

週番号が使用される形式の値を読み込み、U アルゴリズムを使用して SAS 日付値を返します。

カテゴリ: 日付と時間

構文

WEEKU*w*.

説明

w
入力幅を指定します。
デフォルト: 11
範囲: 3–200

詳細

WEEKU*w*.入力形式は、年内の週番号が使用される形式の値を読み込み、U アルゴリズムを使用して SAS 日付値を返します。入力に年表現が含まれていない場合、WEEKU*w*.は現在の年をデフォルトの年表現として使用します。入力に日表現が含まれていない場合、WEEKU*w*.は週の最初の日をデフォルトの日表現として使用します。

U アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(日曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

次の例の WEEKU*w*.入力形式の入力値は、すべて同じ日付となります。現在の年は 2012 です。

幅	形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKU*w.*入力形式は、週番号値を 0 から 53 の範囲の 10 進数として読み込みます。このとき、日曜日を週の最初の日とします。

WEEKV*w.*入力形式は、週番号値を 01 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。年の第 1 週は、1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の木曜日が 2 日、3 日、4 日の場合、それより前の日は前年の最後の週に組み込まれます。

WEEKW*w.*入力形式は、週番号値を 00 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。

例

次の例では、現在の年は 2012 です。

ステートメント	結果
	`----+----1`

`v=input('W01',weeku3.);`
`w=input('03W01',weeku5.);`
`x=input('03W0101',weeku7.);`
`y=input('2003W0101',weeku9.);`
`z=input('2003-W01-01',weeku11.);`
`put v;`
`put w;`
`put x;`
`put y;`
`put z;`
	
	`18993`
	`18993`
	`18993`
	`18993`
	`18993`

関連項目:

出力形式:

- “WEEKUw. 出力形式” (164 ページ)
- “WEEKVw. 出力形式” (166 ページ)
- “WEEKWw. 出力形式” (168 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKVw. 入力形式” (333 ページ)
- “WEEKWw. 入力形式” (334 ページ)

WEEKV*w*. 入力形式

週番号が使用される形式の値を読み込み、V アルゴリズムを使用して SAS 日付値を返します。

カテゴリ: 日付と時間

構文

WEEKV*w*.

説明

w
入力幅を指定します。
デフォルト: 11
範囲: 3–200

詳細

WEEKV*w*.入力形式は、年内の週番号値を読み込みます。入力に年表現が含まれていない場合、WEEKV*w*.は現在の年をデフォルトの年表現として使用します。入力に日表現が含まれていない場合、WEEKV*w* は週の最初の日をデフォルトの日表現として使用します。

V アルゴリズムでは、SAS 日付値を計算します。週番号値は、01 から 53 の範囲の 10 進数で表し、先頭に 0 を使用します。最大値は 53 です。各週は月曜日から始まり、年の第 1 週は 1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の月曜日が 2 日、3 日または 4 日の場合、それより前の日は前年の最後の週に組み込まれます。たとえば、年の 5 週目は 06 として表されます。

次の例の WEEKV*w*.入力形式の入力値は、すべて同じ日付となります。現在の年は 2012 です。

幅	形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKV*w.*入力形式は、週番号値を 01 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。年の第 1 週は、1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の木曜日が 2 日、3 日、4 日の場合、それより前の日は前年の最後の週に組み込まれます。

WEEKU*w.*入力形式は、週番号値を 0 から 53 の範囲の 10 進数として読み込みます。このとき、日曜日を週の最初の日とします。

WEEKW*w.*入力形式は、年の週番号値を 00 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。

例

次の例では、現在の年は 2012 です。

ステートメント	結果
	----+----1

v=input('W01',weekv3.);
w=input('03W01',weekv5.);
x=input('03W0101',weekv7.);
y=input('2003W0101',weekv9.);
z=input('2003-W01-01',weekv11.);
put v;
put w;
put x;
put y;
put z;
	
	18994
	18994
	18994
	18994
	18994

関連項目:

出力形式:

- “WEEKUw. 出力形式” (164 ページ)
- “WEEKVw. 出力形式” (166 ページ)
- “WEEKWw. 出力形式” (168 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKUw. 入力形式” (331 ページ)
- “WEEKWw. 入力形式” (334 ページ)

WEEKW*w.* 入力形式

週番号が使用される形式の値を読み込み、W アルゴリズムを使用して SAS 日付値を返します。

カテゴリ: 日付と時間

構文

WEEKW*w.*

説明

w
入力幅を指定します。
デフォルト: 11
範囲: 3–200

詳細

WEEKW*w.*入力形式は、年内の週番号値を読み込みます。入力に年表現が含まれていない場合、WEEKW*w.*入力形式は現在の年をデフォルトの年表現として使用します。入力に日表現が含まれていない場合、WEEKW*w.*入力形式は週の最初の日をデフォルトの日表現として使用します。W アルゴリズムでは、年内の週番号を使用して SAS 日付値を計算します(月曜日を週の最初の日と見なします)。週番号値は、先頭に 0 を付けた 0 から 53 の範囲の 10 進数として表され、最大値は 53 になります。たとえば、年の 5 週目は 05 として表されます。

次の例の WEEKW*w.*入力形式の入力値は、すべて同じ日付となります。現在の年は 2012 です。

幅	形式	例
3-4	Www	w01
5-6	yyWww	12W01
7-8	yyWwwdd	12W0101
9-10	yyyyWwwdd	2012W0101
11-200	yyyy-Www-dd	2012-W01-01

比較

WEEKW*w.*入力形式は、週番号値を 00 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。

WEEKU*w.*入力形式は、週番号値を 00 から 53 の範囲の 10 進数として読み込みます。このとき、日曜日を週の最初の日とします。

WEEKV*w.*入力形式は、週番号値を 01 から 53 の範囲の 10 進数として読み込みます。このとき、月曜日を週の最初の日とします。年の第 1 週は、1 月 4 日と年の最初の木曜日の両方を含む週です。1 月の最初の木曜日が 2 日、3 日、4 日の場合、それより前の日は前年の最後の週に組み込まれます。

例

次の例では、現在の年は 2012 です。

ステートメント	結果
	----+----1

v=input('W01',weekw3.);
w=input('03W01',weekw5.);
x=input('03W0101',weekw7.);
y=input('2003W0101',weekw9.);
z=input('2003-W01-01',weekw11.);
put v;
put w;
put x;
put y;
put z;
	18994
	18994
	18994
	18994
	18994

関連項目:

出力形式:

- “WEEKUw. 出力形式” (164 ページ)
- “WEEKVw. 出力形式” (166 ページ)
- “WEEKWw. 出力形式” (168 ページ)

関数:

- “WEEK Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “WEEKUw. 入力形式” (331 ページ)
- “WEEKVw. 入力形式” (333 ページ)

YMDDTTM*w.d* 入力形式

<yy>yy-mm-dd hh:mm:ss.ss 形式の日時値を読み込みます。ハイフン(-)、ピリオド(.)、スラッシュ(/)またはコロン(:)などの特殊文字で年、月、日、時間、分、秒が区切られます。年は 2 桁または 4 桁のいずれかになります。

カテゴリ: 日付と時間

配置: 右

構文

YMDDTTM*w.d*

説明

w
出力フィールドの幅を指定します。
デフォルト: 19
範囲: 13–40

d
秒値の小数点以下の桁数を指定します。小数点以下の桁数は、秒の端数を示します。この引数はオプションです。

デフォルト: 0

範囲: 0–39

詳細

YMDDTTMw.d 入力形式は、*<yy>yy-mm-dd hh*:*mm*:*<ss<.ss>>*形式の日時値を読み込みます。

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

mm
月を表す 01 から 12 までの整数です。

dd
月の日を表す 01 から 31 の整数です。

hh
時間を表す 00 から 23 までの整数です。

mm
分を表す 00 から 59 までの整数です。

ss.ss
小数点以下の秒の端数を含む、00 から 59 の範囲の秒数です。

要件　秒の端数が指定される場合、小数点が必須となり、その表記にはピリオドのみ使用できます。。

- または :
日付要素と時間要素を区切るために使用するスラッシュ(/)、ハイフン(-)、コロン(:)、空白文字などの特殊文字のいずれか 1 つを表します。特殊文字は、日付要素間または時間要素間、日付と時間の間の区切り文字として使用できます。

比較

YMDDTTM*w.d* 入力形式は、必須の区切り文字を含む*<yy>yy-mm-dd/hh:mm:ss.ss* 形式の日時値を読み込みます。

MDYAMPM*w.d* 入力形式は、オプションの区切り文字を含む *mm-dd-yy<yy> hh:mm:ss.ss* AM | PM 形式の日時値を読み込みます。日付と時間の間にはスペースが必要です。

DATETIME*w.d* 入力形式は、オプションの区切り文字を含む *dd-mmm-yy<yy> hh:mm:ss.ss* AM|PM 形式の日時値を読み込みます。日付と時間は特殊文字で区切られる場合もあります。

例

```
input @1 dt ymddttm24.;
```

データ行	結果
2012-03-16 11:23:07.4	1647516187.4
2012 03 16 11 23 07.4	1647516187.4

データ行	結果
12.3.16/11:23	1647516180

関連項目:

入力形式:

- “DATETIMEw. 入力形式” (261 ページ)
- “MDYAMPMw.d 入力形式” (283 ページ)

YYMMDDw. 入力形式

yymmdd または *yyyymmdd* 形式の日付値を読み込みます。

カテゴリ: 日付と時間

構文

YYMMDD*w*.

説明

w
入力幅を指定します。
デフォルト: 6
範囲: 6–32

詳細

次のうちいずれかの形式の日付値を読み込みます。

- *yymmdd*
- *yyyymmdd*

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

mm
月の日を表す 01 から 12 の整数です。

dd
月の日を表す 01 から 31 の整数です。

年、月、日の値は、空白や特殊文字で区切ることができます。ただし、区切り文字を使用する場合は、すべての値の間に挿入します。空白は日付の前後に挿入することもできます。入力幅には、空白および特殊文字のスペースを考慮する必要があります。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input calendar_date yymmdd10.;
```

データ行	結果
----+----1	
120316	19068
12/03/16	19068
12 03 16	19068
2012-03-16	19068

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “MMDDYYw. 出力形式” (110 ページ)
- “YYMMDDw. 出力形式” (175 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

YYMMN*w*. 入力形式

yyyymm または *yymm* 形式の日付値を読み込みます。

カテゴリ: 日付と時間

構文

YYMMN*w*.

説明

w
入力幅を指定します。
デフォルト: 4
範囲: 4–6

詳細

次のうちいずれかの形式の日付値を読み込みます。

- *yyyymm*
- *yymm*

yy または *yyyy*
年を表す 2 桁または 4 桁の整数です。

mm
月を表す 2 桁の整数です。

入力形式名として *N* も必ず使用する必要があります。これは、年値と月値が空白または特殊文字で区切られないことを示します。日値 01 がこの値に自動的に追加され、有効な SAS 日付変数が作成されます。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input date1 yymmn6.;
```

データ行	結果
`----+----1`	
`201208`	`19206`

関連項目:

出力形式:

- “DATEw. 出力形式” (70 ページ)
- “DDMMYYw. 出力形式” (75 ページ)
- “YYMMDDw. 出力形式” (175 ページ)
- “YYMMw. 出力形式” (174 ページ)
- “YYMONw. 出力形式” (181 ページ)

関数:

- “DAY Function” in *SAS Functions and CALL Routines: Reference*

- “MONTH Function” in *SAS Functions and CALL Routines: Reference*
- “MDY Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*

入力形式:

- “DATEw. 入力形式” (260 ページ)
- “DDMMYYw. 入力形式” (263 ページ)
- “MMDDYYw. 入力形式” (285 ページ)
- “YYMMDDw. 入力形式” (338 ページ)

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

YYQ*w*. 入力形式

*yy*Q*q* または *yyyy*Q*q* 形式の年の四半期を読み込みます。

カテゴリ: 日付と時間

構文

YYQ*w*.

説明

w
入力幅を指定します。
デフォルト: 6 (SAS 6 の場合、デフォルトは 4)
範囲: 4-32 (SAS 6 の場合、範囲は 4-6)

詳細

次のうちいずれかの形式のデータを読み込みます。

- *yy*Q*q*
- *yyyy*Q*q*

yy または *yyyy*
2 桁または 4 桁の年を表す整数です。

q
四半期を表す整数(1、2、3、4 のいずれか)です。四半期を 01、02、03、04 と表すこともできます。

文字 Q で年値と四半期値を区切る必要があります。年値、文字 Q および四半期値を空白で区切ることはできません。YYQ*w*.で読み込まれる値により、指定された四半期の最初の日に対応する SAS 日付値が生成されます。

注: SAS では、2 桁の年は YEARCUTOFF=システムオプションで定義された 100 年の期間内であると解釈します。

例

```
input quarter yyq9.;
```

データ行	結果
----+----1----+	
12Q2	19084
12Q02	19084
2012Q02	19084

関連項目:

関数:

- “QTR Function” in *SAS Functions and CALL Routines: Reference*
- “YEAR Function” in *SAS Functions and CALL Routines: Reference*
- “YYQ Function” in *SAS Functions and CALL Routines: Reference*

システムオプション:

- “YEARCUTOFF= System Option” in *SAS System Options: Reference*

ZD*w.d* 入力形式

ゾーン 10 進データを読み込みます。

カテゴリ: 数値

参照項目: “ZDw.d Informat: UNIX” in *SAS Companion for UNIX Environments*
“ZDw.d Informat: Windows” in *SAS Companion for Windows*
“ZDw.d Format: z/OS” in *SAS Companion for z/OS*

構文

ZD*w.d*

説明

w
入力幅を指定します。
デフォルト: 1
範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。
範囲: 1–31

詳細

ZD*w.d* 入力形式は、ゾーン 10 進データを読み込みます。ゾーン 10 進データは、1 桁に 1 バイト必要で、最後のバイトに値の符号が最後の桁とともに含まれます。

注: 動作環境によってゾーン 10 進値の保存方法は異なります。ただし、ZD*w.d* では、ゾーン 10 進値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

ゾーン 10 進形式の正数値を PC から入力できます。一部のキー装置では、マイナス符号を最後の桁に重ね打ちすることにより、負数値を入力できます。

比較

- *w.d* 入力形式のように、ZD*w.d* 入力形式は、1 桁に 1 バイトが必要なデータを読み込みます。ZDV*w.d* または ZD*w.d* を使用して、最後のバイトに最後の桁と符号を含むゾーン 10 進データを読み込みます。
- ZD*w.d* 入力形式は、ZDV*w.d* 入力形式と同じように機能しますが、ZDV*w.d* では入力文字列を検証し、無効なデータを許可しません。
- 次の表は、各種プログラミング言語でのゾーン 10 進入力形式の表記を比較したものです。

言語	ゾーン 10 進表記
SAS	ZD3.
PL/I	PICTURE'99T'
COBOL	DISPLAY PIC S 999
IBM アセンブラ	ZL3

例

```
input @1 x zd4.;
```

データ行*	結果
`----+----1`	
`F0F1F2C8`	128

* データ行は、IBM メインフレームコンピュータシステム上のゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。

関連項目:

入力形式

- “w.d 入力形式” (330 ページ)
- “ZDVw.d 入力形式” (344 ページ)

ZDB*w.d* 入力形式

空白を含むゾーン 10 進データを読み込みます。

カテゴリ: 数値

参照項目: “ZDBw.d Informat: z/OS” in *SAS Companion for z/OS*

構文

ZDB*w.d*

説明

w
入力幅を指定します。

デフォルト: 1

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 0–31

詳細

ZDB*w.d* 入力形式は、IBM 1410、1401 および 1620 形式で生成されたゾーン 10 進データを読み込みます。このゾーン 10 進データでは、0 はパンチされるのではなく、空白で残されます。

例

```
input @1 x zdb3.;
```

データ行 *	結果
----+----1	
F140C2	102

* データ行は、IBM メインフレーム動作環境上でスペース用コードを含むゾーン 10 進形式で格納したバイナリ値を 16 進表現で表したものです。1 バイトは、入力フィールドの 1 カラムを使用します。

ZDV*w.d* 入力形式

ゾーン 10 進データを読み込んで検証します。

カテゴリ: 数値

構文

ZDV*w.d*

説明

w
入力幅を指定します。

デフォルト: 1

範囲: 1–32

d
値を除算する 10 のべき乗を指定します。この引数はオプションです。

範囲: 1–31

詳細

ZDV*w.d* 入力形式は、1 桁に 1 バイトが必要であり、最後のバイトに値の符号が最後の桁とともに含まれるデータを読み込みます。また、入力文字列を検証し、無効なデータを許可しません。

ZDV*w.d* は、動作環境に依存します。たとえば、IBM メインフレーム上では、ZDV*w.d* は最後の上位ニブルを除くすべての上位ニブルで F が必要です。(これに対応して、ZD*w.d* 入力形式では、符号と関連付けられているニブル以外のすべてのバイトの上位ニブルを無視します。) 最後の上位ニブルには、A から F までの値を使用できます。A、C、E および F が正の値で、B と D が負の値です。IBM メインフレーム上の下位ニブルは、ZD と同じように、0 から 9 までの数値である必要があります。

注: 動作環境によってゾーン 10 進値の保存方法は異なります。ただし、ZDV*w.d* 入力形式では、ゾーン 10 進値を、SAS の実行に使用する動作環境で値が作成されたものとして読み込みます。

比較

ZDV*w.d* 入力形式は、ZD*w.d* 入力形式と同じように機能しますが、ZDV*w.d* では入力文字列を検証し、無効なデータを許可しません。

例

```
input @1 test zdv4.;
```

データ行*	結果
----+----1	
F0F1F2C8	128

* データ行は、ゾーン 10 進形式で保存したバイナリ値を 16 進表現で表したものです。この例は、IBM メインフレームワークで実行されました。結果は、動作環境によって異なります。

関連項目:

入力形式:

- “w.d 入力形式” (330 ページ)
- “ZDw.d 入力形式” (342 ページ)

キーワード

S

T

V

W

Y