Natural for UNIX

システム関数

バージョン 8.4.1

2017 年 10 月

ADABAS & NATURAL

このマニュアルは Natural バージョン 8.4.1 およびそれ以降のすべてのリリースに適用されます。

このマニュアルに記載される仕様は変更される可能性があります。変更は以降のリリースノートまたは新しいマニュアルに記述されます。

ドキュメント IDは: NATUX-NNATFUNCTIONS-841-20200614JA

目次

前書き

このドキュメントでは、特定のステートメントで使用できるNaturalの各種「組み込み」関数について説明します。『*プログラミングガイド*』の「*システム関数*」を参照してください。

このドキュメントは次の項目で構成されています。

処理ループで使用する **Natural** システム関数	プログラムのループコンテキストで使用できるNaturalシステム関数について説明します。
算術システム関数	算術処理ステートメントおよび論理条件基準でサポートされているシステム関数について説明します。
その他のシステム関数	最小値または最大値を評価するシステム関数、フィールド識別用のシステム関数、Natural以外のプログラムからリターンコードを受け取るシステム関数、「不正にソートされた」文字を変換するシステム関数、大文字／小文字を変換するシステム関数、先頭／末尾の空白を削除するシステム関数。
Natural オブジェクトとして提供されている関数	URL エンコードや Base64 の変換などをサポートするための Natural オブジェクトとして提供される関数について説明します。

『*プログラミングガイド*』の「*システム変数とシステム関数の例*」も参照してください。

1

表記規則

規則	説明
太字	画面上の要素を表します。
モノスペースフォント	*folder.subfolder:service* という規則を使用して webMethods Integration Server 上のサービスの保存場所を表します。
大文字	キーボードのキーを表します。同時に押す必要があるキーは、プラス記号 (+) で結んで表記されます。
斜体	独自の状況または環境に固有の値を指定する必要がある変数を表します。本文で最初に出現する新しい用語を表します。
モノスペースフォント	入力する必要があるテキストまたはシステムから表示されるメッセージを表します。Program code.
{ }	選択肢のセットを表します。ここから1つ選択する必要があります。中カッコの内側にある情報のみを入力します。{ } 記号は入力しません。
|	構文行で相互排他的な2つの選択肢を区切ります。いずれかの選択肢を入力します。| 記号は入力しません。
[ ]	1 つ以上のオプションを表します。大カッコの内側にある情報のみを入力します。[ ] 記号は入力しません。
...	同じ種類の情報を複数回入力できることを示します。情報だけを入力してください。実際のコードに繰り返し記号 (...) を入力しないでください。

オンライン情報

Software AG マニュアルの **Web** サイト

マニュアルは、Software AG マニュアルの Web サイト (**http://documentation.softwareag.com**) で入手できます。このサイトでは Empower クレデンシャルが必要です。Empower クレデンシャルがない場合は、TECHcommunity Web サイトを使用する必要があります。

Software AG Empower 製品のサポート **Web** サイト

もしまだEmpowerのアカウントをお持ちでないのなら、こちらへempower@softwareag.com電子メールにて あなたのお名前、会社名、会社の電子メールアドレスをお書きの上、アカウントを請求してください。

いったんアカウントをお持ちになれば、Empower **https://empower.softwareag.com/** のeServiceセクションにてサポートインシデントをオンラインで開くことができます。

製品情報は、Software AG Empower 製品のサポート Web サイト (**https://empower.softwareag.com**) で入手できます。

機能および拡張機能に関するリクエストの送信、製品の可用性に関する情報の取得、製品のダウンロードを実行するには、Products に移動します。

修正に関する情報を取得し、早期警告、技術論文、Knowledge Base の記事を読むには、**Knowledge Center** に移動します。

もしご質問があれは、こちらの**https://empower.softwareag.com/public_directory.asp** グローバルサポート連絡一覧の、あなたの国の電話番号を選んで、わたくし共へご連絡ください。

Software AG TECHcommunity

マニュアルおよびその他の技術情報は、Software AG TECHcommunity Web サイト (**http://techcommunity.softwareag.com**) で入手できます。以下の操作を実行できます。

- TECHcommunity クレデンシャルを持っている場合は、製品マニュアルにアクセスできます。TECHcommunity クレデンシャルがない場合は、登録し、関心事の領域として [マニュアル] を指定する必要があります。
- 記事、コードサンプル、デモ、チュートリアルにアクセスする。
- Software AGの専門家によって承認されたオンライン掲示板フォーラムを使用して、質問したり、ベストプラクティスを話し合ったり、他の顧客が Software AG のテクノロジをどのように使用しているかを学んだりすることが可能です。
- オープンスタンダードや Web テクノロジを取り扱う外部 Web サイトにリンクできます。

データ保護

Software AG製品は、EU一般データ保護規則(GDPR)を尊重した個人データの処理機能を提供します。 該当する場合、適切な手順がそれぞれの管理ドキュメントに記載されています。

I

2 処理ループで使用する Natural システム関数

この章では、プログラムのループコンテキストで使用できる Natural システム関数について説明します。

処理ループでのシステム関数の使用

指定／評価

Natural システム関数は次のステートメントに指定できます。

- 割り当ておよび算術ステートメント：
  - MOVE
  - ASSIGN
  - COMPUTE
  - ADD
  - SUBTRACT
  - MULTIPLY
  - DIVIDE
- 入出力ステートメント：
  - DISPLAY
  - PRINT
  - WRITE

上記のステートメントを次のステートメントブロック内で使用する場合に、システム関数を指定できます。

- AT BREAK
- AT END OF DATA
- AT END OF PAGE

つまり、FIND、READ、HISTOGRAM、SORT、または READ WORK FILE のすべての処理ループに指定できます。

AT END OF PAGE ステートメントでシステム関数を使用する場合、対応する DISPLAY ステートメントに GIVE SYSTEM FUNCTIONS 節を指定する必要があります。

WHERE 節で排除されたレコードは、システム関数で評価されません。

FIND、READ、HISTOGRAM、またはSORT ステートメントで開始した、異なるレベルの処理ループのデータベースフィールドでシステム関数が評価される場合、その値は常にループ階層の位置に従って処理されます。例えば、新しいデータ値が外側のループに対して取得されていた場合、値はそのループに対してだけ処理されます。

ユーザー定義変数でシステム関数が評価される場合、レポーティングモードの処理は、ユーザー定義変数が定義されたループ階層の位置に依存します。処理ループの開始前に定義されているユーザー定義変数の場合、AT BREAK、AT END OF DATA、または AT END OF PAGE ステートメントが定義されたループ内のシステム関数に対して評価されます。ユーザー定義変数が処理ループ内で定義されている場合、処理中のデータベースフィールドと同様に処理されます。

ユーザー定義変数に対するシステム関数評価の参照を選択する場合、値を処理するループを示すためにユーザー定義変数にループ参照を付けて指定することをお勧めします。ループ参照はステートメントラベルまたはソースコード行番号として指定できます。

SORT GIVE ステートメントでの使用

システム関数は、SORT ステートメントの GIVE 節で評価されたときにも参照できます。

SORT GIVE ステートメントで評価されたシステム関数を参照するには、システム関数の名前の先頭にアスタリスク（*）を付ける必要があります。

AVER、NAVER、SUM、TOTAL での桁あふれ

システム関数 AVER、NAVER、SUM、および TOTAL を適用するフィールドには、桁あふれを防止するために十分な長さ（デフォルトまたはユーザー指定）が必要です。桁あふれが起きると、エラーメッセージが発行されます。

通常、長さはシステム関数を適用するフィールドの長さと同じです。この長さでは不十分の場合、SORT GIVE ステートメントの［NL］オプションを使用して、次のように出力長を拡張する必要があります。

```
SUM(field)(NL=nn)
```

この場合、出力長が拡張されるだけでなく、フィールドの内部長も拡張されます。

ステートメント参照（r）

ステートメント参照はシステム関数に対しても使用できます。『*プログラミングガイド*』の「*ユーザー定義変数*」セクションの「*表記（r）を使用したデータベースフィールドの参照*」も参照してください。

ステートメントラベルまたはソースコード行番号 (*r*) を使用して、指定したフィールドに対してシステム関数を評価する処理ループを指定できます。

AVER(r)(field)

フォーマット／長さ：	フィールドと同じ。 例外：フィールドフォーマットが N の場合、AVER(*field*) のフォーマットは P（桁数はフィールドと同じ）。

AVER に指定されたフィールドのすべての値の平均値を持ちます。AVER は、AVER 要求時の条件が真になるたびに更新されます。

COUNT(r)(field)

フォーマット／長さ：	P7

COUNT は置かれた処理ループを通過するたびに 1 ずつ増加します。COUNT の増加には、COUNT に指定されたフィールドの値は関係しません。

MAX(r)(field)

フォーマット／長さ：	フィールドと同じ。

MAX に指定されたフィールドの最大値を持ちます。MAX を含む処理ループが実行されるたびに（必要に応じて）更新されます。

MIN(r)(field)

フォーマット／長さ：	フィールドと同じ。

MIN に指定されたフィールドの最小値を持ちます。MIN の置かれた処理ループが実行されるたびに（必要に応じて）更新されます。

NAVER(r)(field)

フォーマット／長さ：	フィールドと同じ。 例外：フィールドフォーマットがNの場合、NAVER*(field)*のフォーマットはP（桁数はフィールドと同じ）。

NAVER に指定されたフィールドのすべての値（空値を除く）の平均値を持ちます。NAVER は、NAVER 要求時の条件が真になるたびに更新されます。

NCOUNT(r)(field)

フォーマット／長さ：	P7

NCOUNT は置かれた処理ループを通過するたびに、NCOUNT に指定されたフィールドが空値でないときに 1 ずつ増加します。

NCOUNT の結果が配列になるかスカラ値になるかは、その引数（フィールド）によって決まります。結果のオカレンスの数はフィールドと同じです。

NMIN(r)(field)

フォーマット／長さ：	フィールドと同じ。

NMIN に指定されたフィールドの最小値（空値を除く）を持ちます。NMIN の置かれた処理ループが実行されるたびに（必要に応じて）更新されます。

OLD(r)(field)

フォーマット／長さ：	フィールドと同じ。

AT BREAK 条件で指定されたコントロールブレイクの前、またはエンドオブページ条件やエンドオブデータ条件の前に、OLD で指定されたフィールドの値を持ちます。

SUM(r)(field)

フォーマット／長さ：	フィールドと同じ。 例外：フィールドフォーマットが N の場合、SUM*(field)* のフォーマットは P（桁数はフィールドと同じ）。

SUM に指定されたフィールドのすべての値の合計を持ちます。SUM の置かれた処理ループが実行されるたびに更新されます。SUM を AT BREAK 条件に続けて指定すると、値のブレイクごとにリセットされます。ブレイク間の値だけが加算されます。

TOTAL(r)(field)

フォーマット／長さ：	フィールドと同じ。 例外：フィールドフォーマットが N の場合、TOTAL*(field)* のフォーマットは P（桁数はフィールドと同じ）。

TOTAL の置かれているすべてのオープン処理ループで、TOTAL に指定されたフィールドのすべての値の合計を持ちます。

例

- 例 1 - Natural システム関数 OLD、MIN、AVER、MAX、SUM、COUNT を使用した AT BREAK ステートメント
- 例 2 - Natural システム関数 AVER を使用した AT BREAK ステートメント
- 例 3 - システム関数 MAX、MIN、AVER を使用した AT END OF DATA ステートメント

例 1 - Natural システム関数 OLD、MIN、AVER、MAX、SUM、COUNT を使用した AT BREAK ステートメント

```
** Example 'ATBEX3': AT BREAK  (with Natural system functions)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 3
READ EMPLOY-VIEW LOGICAL BY CITY = 'SALT LAKE CITY'
  DISPLAY NOTITLE CITY NAME 'SALARY' SALARY(1) 'CURRENCY' CURR-CODE(1)
  /*
  AT BREAK OF CITY
    WRITE /  OLD(CITY) (EM=X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X^X)
      31T '   MINIMUM:' MIN(SALARY(1))  CURR-CODE(1) /
      31T '   AVERAGE:' AVER(SALARY(1)) CURR-CODE(1) /
      31T '   MAXIMUM:' MAX(SALARY(1))  CURR-CODE(1) /
      31T '       SUM:' SUM(SALARY(1))  CURR-CODE(1) /
      35T COUNT(SALARY(1)) 'RECORDS FOUND' /
  END-BREAK
  /*
  AT END OF DATA
    WRITE 22T 'TOTAL (ALL RECORDS):'
              T*SALARY TOTAL(SALARY(1))  CURR-CODE(1)
  END-ENDDATA
END-READ
*
END      ↩
```

プログラム ATBEX3 の出力：

```
        CITY                    NAME           SALARY   CURRENCY
-------------------- -------------------- ---------- --------

SALT LAKE CITY       ANDERSON                  50000 USD
SALT LAKE CITY       SAMUELSON                 24000 USD

S A L T   L A K E   C I T Y      MINIMUM:       24000 USD
                                 AVERAGE:       37000 USD
                                 MAXIMUM:       50000 USD
                                     SUM:       74000 USD
                                         2 RECORDS FOUND
```

```
SAN DIEGO            GEE                        60000 USD

S A N   D I E G O               MINIMUM:        60000 USD
                                AVERAGE:        60000 USD
                                MAXIMUM:        60000 USD
                                    SUM:        60000 USD
                                        1 RECORDS FOUND

                    TOTAL (ALL RECORDS):      134000 USD ↩
```

例 2 - Natural システム関数 AVER を使用した AT BREAK ステートメント

```
** Example 'ATBEX4': AT BREAK  (with Natural system functions)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
  2 SALARY    (2)
*
1 #INC-SALARY (P11)
END-DEFINE
*
LIMIT 4
EMPL. READ EMPLOY-VIEW BY CITY STARTING FROM 'ALBU'
  COMPUTE #INC-SALARY = SALARY (1) + SALARY (2)
  DISPLAY NAME CITY SALARY (1:2) 'CUMULATIVE' #INC-SALARY
  SKIP 1
  /*
  AT BREAK CITY
    WRITE NOTITLE
      'AVERAGE:'            T*SALARY (1)  AVER(SALARY(1)) /
      'AVERAGE CUMULATIVE:' T*#INC-SALARY AVER(EMPL.) (#INC-SALARY)
  END-BREAK
END-READ
*
END                                                      ↩
```

プログラム ATBEX4 の出力：

```
NAME                 CITY           ANNUAL    CUMULATIVE
                                                SALARY
-------------------- -------------------- ---------- ------------

HAMMOND              ALBUQUERQUE               22000        42200
                                               20200

ROLLING              ALBUQUERQUE               34000        65200
                                               31200

FREEMAN              ALBUQUERQUE               34000        65200
```

```
                                                31200

LINCOLN              ALBUQUERQUE                41000        78700
                                                37700

AVERAGE:                                        32750
AVERAGE CUMULATIVE:                                          62825
```

例 3 - システム関数 MAX、MIN、AVER を使用した AT END OF DATA ステートメント

```
** Example 'AEDEX1S': AT END OF DATA
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 FIRST-NAME
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
LIMIT 5
EMP. FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  IF NO RECORDS FOUND
    ENTER
  END-NOREC
  DISPLAY PERSONNEL-ID NAME FIRST-NAME
          SALARY (1) CURR-CODE (1)
  /*
  AT END OF DATA
    IF *COUNTER (EMP.) = 0
      WRITE 'NO RECORDS FOUND'
      ESCAPE BOTTOM
    END-IF
    WRITE NOTITLE / 'SALARY STATISTICS:'
                  / 7X 'MAXIMUM:' MAX(SALARY(1))  CURR-CODE (1)
                  / 7X 'MINIMUM:' MIN(SALARY(1))  CURR-CODE (1)
                  / 7X 'AVERAGE:' AVER(SALARY(1)) CURR-CODE (1)
  END-ENDDATA
  /*
END-FIND
*
END                                                              ↩
```

プログラム AEDEX1S の出力：

```
PERSONNEL         NAME                FIRST-NAME          ANNUAL   CURRENCY
   ID                                                     SALARY     CODE
--------- -------------------- -------------------- ---------- --------

11100328  BERGHAUS             ROSE                        70800 DM
11100329  BARTHEL              PETER                       42000 DM
11300313  AECKERLE             SUSANNE                     55200 DM
11300316  KANTE                GABRIELE                    61200 DM
11500304  KLUGE                ELKE                        49200 DM

SALARY STATISTICS:
       MAXIMUM:      70800 DM
       MINIMUM:      42000 DM
       AVERAGE:      55680 DM
```

例 4 - システム関数 AVER を使用した AT END OF PAGE ステートメント

```
** Example 'AEPEX1S': AT END OF PAGE (structured mode)
************************************************************************
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 SALARY    (1)
  2 CURR-CODE (1)
END-DEFINE
*
FORMAT PS=10
LIMIT 10
READ EMPLOY-VIEW BY PERSONNEL-ID FROM '20017000'
  DISPLAY NOTITLE GIVE SYSTEM FUNCTIONS
          NAME JOB-TITLE 'SALARY' SALARY(1) CURR-CODE (1)
  /*
  AT END OF PAGE
    WRITE / 28T 'AVERAGE SALARY: ...' AVER(SALARY(1)) CURR-CODE (1)
  END-ENDPAGE
END-READ
*
END
```

プログラム AEPEX1S の出力：

```
        NAME                  CURRENT             SALARY   CURRENCY
                             POSITION                        CODE
-------------------- ------------------------- ---------- --------

CREMER               ANALYST                        34000 USD
MARKUSH              TRAINEE                        22000 USD
GEE                  MANAGER                        39500 USD
KUNEY                DBA                            40200 USD
NEEDHAM              PROGRAMMER                     32500 USD
JACKSON              PROGRAMMER                     33000 USD

                           AVERAGE SALARY: ...      33533 USD
```

3 算術システム関数

次の算術関数は、算術処理ステートメント（ADD、COMPUTE、DIVIDE、MULTIPLY、SUBTRACT）および論理条件基準でサポートされています。

関数	フォーマット／長さ	説明
ABS(*field*)	*field* と同じ	*field* の絶対値。
ATN(*field*)	F8（*）	*field* アークタンジェント。
COS(*field*)	F8（*）	*field* のコサイン。
EXP(*field*)	F8（*）	基数 e、指数 *field* による累乗（すなわち e^{field}。e はオイラー数）。
FRAC(*field*)	*field* と同じ	*field* の小数部分。
INT(*field*)	*field* と同じ	*field* の整数部分。
LOG(*field*)	F8（*）	*field* の自然対数。
SGN(*field*)	*field* と同じ	*field* の符号（-1、0、+1）。
SIN(*field*)	F8（*）	*field* のサイン（正弦）。
SQRT(*field*)	F8 (*)	*field* の平方根。 負の値の引数フィールドは正の値として扱われます。
TAN(*field*)	F8（*）	*field* のタンジェント。
VAL(*field*)	ターゲットフィールドと同じ	英数字 *field* から数値を抽出します。*field* の内容は、数値の英数字（コードページまたはUnicode）文字表現にする必要があります。*field*の先頭または末尾の空白は無視されます。小数点や先行符号文字は処理されます。 ターゲットフィールドの長さが不足している場合、小数は切り捨てられます（『*プログラミングガイド*』の「*演算割り当てのルール*」セクションの「*フィールドの切り捨てと切り上げ*」も参照）。

* これらの関数は次のように評価されます。

- 引数は、フォーマット／長さが F8 に変換された後、計算のためにオペレーティングシステムに渡されます。
- オペレーティングシステムによって返される結果のフォーマット／長さは F8 で、その後ターゲットフォーマットに変換されます。

算術関数（VAL を除く）で使用できる *field* は、定数またはスカラです。フォーマットは数値（N）、パック 10 進（P）、整数（I）、または浮動小数点（F）にする必要があります。

VAL 関数に使用できる *field* は、定数、スカラ、または配列です。フォーマットは英数字にする必要があります。

算術関数の例：

```
** Example 'MATHEX': Mathematical functions
************************************************************************
DEFINE DATA LOCAL
1 #A     (N2.1) INIT <10>
1 #B     (N2.1) INIT <-6.3>
1 #C     (N2.1) INIT <0>
1 #LOGA  (N2.6)
1 #SQRTA (N2.6)
1 #TANA  (N2.6)
1 #ABS   (N2.1)
1 #FRAC  (N2.1)
1 #INT   (N2.1)
1 #SGN   (N1)
END-DEFINE
*
COMPUTE #LOGA = LOG(#A)
WRITE NOTITLE '=' #A 5X 'LOG'         40T #LOGA
*
COMPUTE #SQRTA = SQRT(#A)
WRITE         '=' #A 5X 'SQUARE ROOT' 40T #SQRTA
*
COMPUTE #TANA  = TAN(#A)
WRITE         '=' #A 5X 'TANGENT'     40T #TANA
*
COMPUTE #ABS   = ABS(#B)
WRITE     //  '=' #B 5X 'ABSOLUTE'    40T #ABS
*
COMPUTE #FRAC  = FRAC(#B)
WRITE         '=' #B 5X 'FRACTIONAL'  40T #FRAC
*
COMPUTE #INT   = INT(#B)
WRITE         '=' #B 5X 'INTEGER'     40T #INT
*
COMPUTE #SGN   = SGN(#A)
WRITE      // '=' #A 5X 'SIGN'        40T #SGN
*
COMPUTE #SGN   = SGN(#B)
```

```
WRITE          '=' #B 5X 'SIGN'         40T #SGN
*
COMPUTE #SGN   = SGN(#C)
WRITE          '=' #C 5X 'SIGN'         40T #SGN
*
END                                          ↩
```

プログラム MATHEX の出力：

```
#A:  10.0     LOG                        2.302585
#A:  10.0     SQUARE ROOT                3.162277
#A:  10.0     TANGENT                    0.648360


#B:  -6.3     ABSOLUTE                   6.3
#B:  -6.3     FRACTIONAL                -0.3
#B:  -6.3     INTEGER                   -6.0


#A:  10.0     SIGN                       1
#B:  -6.3     SIGN                      -1
#C:   0.0     SIGN                       0       ↩
```

II その他のシステム関数

以下のトピックについて説明します。

4 *MINVAL/*MAXVAL - 最小値／最大値の評価

```
{ *MINVAL }
{ *MAXVAL } ([(IR=result-format/length)] operand、...)
```

フォーマット／長さ：フォーマットおよび長さは、IR 節を使用して明示的に指定するか、後述の*フォーマット／長さの変換規則表*を使用して自動的に評価できます。

関数

*MINVAL/*MAXVAL システム関数は、指定されたすべてのオペランド値の最小値／最大値を評価します。結果は常にスカラ値です。オペランドとして配列が指定された場合、すべての配列フィールドの最小／最大が評価されます。

英数字データまたはバイナリデータを引数として使用するとき、データが同一であれば（例えば、*MINVAL('AB','AB')）、結果は最小／最大の長さを持つ引数になります。

制限

システム関数 *MINVAL/*MAXVAL を使用する場合、以下の制限が適用されます。

- *MINVAL/*MAXVAL をターゲット変数が予期される位置に使用しないでください。
- *MINVAL/*MAXVAL は、システム関数でネストすることはできません。

構文説明

オペランド定義テーブル：

オペランド	構文要素					フォーマット												オペランド参照	ダイナミック定義
operand	C	S	A	G		A	U	N	P	I	F	B	D	T				○	×

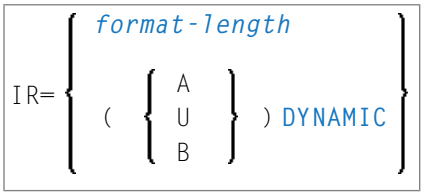
構文要素の説明：

構文要素	説明
*MINVAL	指定されたすべてのオペランド値の最小値を評価します。
*MAXVAL	指定されたすべてのオペランド値の最大値を評価します。
operand	*MINVAL/*MAXVAL システム関数で最小値／最大値が評価されるオペランドです。
result-format-length	結果のフォーマット／長さを明示的に指定するための中間結果節です。下記の「*IR 節*」を参照してください。

IR 節

IR（中間結果）節は、*MINVAL/*MAXVAL システム関数全体の *result-format/length* を明示的に指定するために使用できます。

```
IR=result-format/length
```



結果のフォーマット／長さの有効な組み合わせについては、後述の*フォーマット／長さの変換規則表*を参照してください。

構文要素の説明：

構文要素	説明
format-length	コンパイラは関数全体の結果のフォーマット／長さを決定しようとします。コンパイラが、精度に損失のないことが保証される方法でフォーマット／長さを決定できない場合、プログラマが IR オペランド拡張を使用して *format-length* を設定する必要があります。
A、U または B	フォーマット：ダイナミック変数用の英数字、Unicode、またはバイナリです。
DYNAMIC	固定フォーマット／長さを指定する代わりに、ダイナミック長の英数字フォーマット、Unicode フォーマット、またはバイナリフォーマットを指定できます。

例：

```
DEFINE DATA LOCAL
1 #RESULTI        (I4)
1 #RESULTA        (A20)
1 #RESULTADYN     (A) DYNAMIC
1 #A(I4)          CONST <1234>
1 #B(A20)         CONST <H'30313233'> /* '0123' stored
1 #C(I2/1:3)      CONST <2000, 2100, 2200>
END-DEFINE
*
#RESULTA     := *MAXVAL((IR=A20)         #A, #B)     /*no error, I4->A20 is allowed!
#RESULTADYN  := *MAXVAL((IR=(A)DYNAMIC)  #A, #B)     /*result is (A) dynamic
/* #RESULTI  := *MAXVAL((IR=I4)         #A, #B)     /*compiler error, because conv. ↩
A20->I4 is not allowed!
#RESULTI     := *MAXVAL((IR=I4)          #A, #C(*))  /*maximum of the array is ↩
evaluated
DISPLAY #RESULTA #RESULTADYN (AL=10) #RESULTI
END
```

結果のフォーマット／長さの変換規則表

*MINVAL/*MAXVAL システム関数全体の結果のフォーマット／長さは複数の方法で定義できます。

- 結果のフォーマット／長さの明示的な指定
- 結果のフォーマット／長さの暗黙的な指定

結果のフォーマット／長さの明示的な指定

*MINVAL/*MAXVAL システム関数全体の結果のフォーマット／長さは、IR 節で指定できます。指定されたすべてのオペランドは、精度が損失ないのであれば、この結果のフォーマット／長さに変換されます。その後、変換されたすべてのオペランドの最小／最大が評価され、評価されたフォーマット／長さを持つ単一スカラ値がシステム関数全体の結果として設定されます。

結果のフォーマット／長さの暗黙的な指定

*MINVAL/*MAXVAL システム関数内で IR 節を使用しない場合、結果のフォーマット／長さは、*MINVAL/*MAXVAL システム関数内の引数として指定されたすべてのオペランドのフォーマット／長さに関連して評価されます。各オペランドのフォーマット／長さが取得され、引数リストの次のオペランドのフォーマット／長さと結合されます。そして、フォーマット／長さの変換規則表を使用して 2 つの単一オペランドの結果のフォーマット／長さが評価されます。

フォーマット／長さの変換規則表は、2 つの異なる表に分割されます。下記の 2 つの表に示されていない組み合わせはすべて無効であり、*MINVAL/*MAXVAL システム関数の引数リスト内に適用しないでください。キーワード FLF は、他の方法では精度の損失が生じる可能性があるため、結果のフォーマット／長さを定義するために IR 節を使用する必要があることを示します。

表 **1**

2 つの異なるオペランドの数値の組み合わせをすべて示しています。

		第 2 オペランド				
	フォーマット／長さ	I1	I2	I4	P*a*.*b*、N*a*.*b*	F4、F8
第 1 オペランド	I1	I1	I2	I4	Pmax(3,*a*)。*b*	F8
	I2	I2	I2	I4	Pmax(5,*a*)。*b*	F8
	I4	I4	I4	I4	Pmax(10,*a*)。*b*	F8
	P*x*.*y*、N*x*.*y*	Pmax(3,*x*)。*y*	Pmax(5,*x*)。*y*	Pmax(10,*x*)。*y*	max(*x*,*a*) + max(*y*,*b*) <= 29 の場合 Pmax(*x*,*a*).max(*y*,*b*) else FLF	*y*=0 および *x* <=15 の場合 F8 else FLF
	F4、F8	F8	F8	F8	*b*=0 および *a* <=15 の場合 F8	F8

		第 2 オペランド				
	フォーマット／長さ	I1	I2	I4	P*a.b*、N*a.b*	F4、F8
					else FLF	

凡例：

FLF	フォーマット／長さ宣言が強制されます。結果のフォーマットは IR 節を使用して指定する必要があります。
I*x*	フォーマット／長さは整数です。*x* では、整数の値を保存するために使用されるバイト数を指定します。
F*x*	フォーマット／長さは浮動小数点です。*x* では、浮動小数点の値を保存するために使用されるバイト数を指定します。
P*x.y* **P***a,b*	対応する小数点の前の桁数（*x,a*）および精度（*y,b*）を持つパックフォーマットです。
Nx.y **Na,b**	対応する小数点の前の桁数（*x,a*）および精度（*y,b*）を持つ数値フォーマットです。
Pmax(*c,d***)。** *e*	生成されるフォーマットはパックされています。長さは後続の情報で評価されます。小数点の前の桁数は、*c* および *d* の最大値です。精度の値は、*e* です。
Pmax(*c,d***).max(***e,f***)**	生成されるフォーマットはパックされています。長さは後続の情報で評価されます。小数点の前の桁数は、*c* および *d* の最大値です。精度の値は、*e* および *f* の最大値です。

表 **2**

*MINVAL/*MAXVAL システム関数オペランドに使用できるその他のフォーマットと長さをすべて示しています。

		第 2 オペランド				
	フォーマット／長さ	D	T	A*a*、A ダイナミック	B*a*、B ダイナミック	U*a*、U ダイナミック
第 1 オペランド	D	D	T	NA	NA	NA
	T	T	T	NA	NA	NA
	A*x*、A ダイナミック	NA	NA	A ダイナミック	A ダイナミック	U ダイナミック
	B*x*、B ダイナミック	NA	NA	A ダイナミック	B ダイナミック	U ダイナミック
	U*x*、U ダイナミック	NA	NA	U ダイナミック	U ダイナミック	U ダイナミック

凡例：

NA	この組み合わせは許可されていません。
D	日付フォーマットです。
T	時刻フォーマットです。
B*x*、**B***a*	長さ *x*，*a* のバイナリフォーマットです。
A*x*、**A***a*	長さ *x*，*a* の英数字フォーマットです。
U*x*、**U***a*	長さ *x*，*a* の Unicode フォーマットです。
B ダイナミック	ダイナミック長のバイナリフォーマットです。
A ダイナミック	ダイナミック長の英数字フォーマットです。
U ダイナミック	ダイナミック長の Unicode フォーマットです。

result-format-length の評価

コンパイラは、上記の規則に基づいて、オペランドのペアを考慮し、各ペアの中間結果を計算することで、ソースオペランドを処理できます。1 番目のペアは第 1 と第 2 オペランド、中間結果の 2 番目のペアと第 3 オペランド、などから構成されます。すべてのオペランドが処理された後、最後の結果は、最小／最大を評価するために全オペランドの比較に使用されるフォーマットと長さの比較を示します。フォーマット／長さの評価にこの方法を使用するとき、オペランド *format-lengths* は任意の順序で示すことができます。

例：

```
DEFINE DATA LOCAL
1 A (I2)      INIT <34>
1 B (P4.2)    INIT <1234.56>
1 C (N4.4)    INIT <12.6789>
1 D (I1)      INIT <100>
1 E (I4/1:3)  INIT <32, 6745, 456>
1 #RES-MIN (P10.7)
1 #RES-MAX (P10.7)
END-DEFINE
*
MOVE *MINVAL(A, B, C, D, E(*)) TO #RES-MIN
MOVE *MAXVAL(A, B, C, D, E(*)) TO #RES-MAX
DISPLAY #RES-MIN #RES-MAX
END
```

出力：

```
         #RES-MIN                      #RES-MAX
-------------------------  -------------------------

                12.6789000                  6745.0000000
```

次の表は、例のフォーマット／長さを自動的に評価する単一ステップを示しています。それは全ステップの中間結果（ir）および *result-format/length* として使用される比較フォーマット／長さ（cf）を示します。

評価順序	第1オペランド名	第1オペランドまたは中間結果のフォーマット／長さ	第2オペランド名	第2オペランドまたは中間結果のフォーマット／長さ	中間結果（ir）のフォーマット／長さ
1.	A	I2	B	P4.2	ir1 = P5.2
2.	ir1	P5.2	C	N4.4	ir2 = P5.4
3.	ir2	P5.4	D	I1	ir3 = P5.4
4.	ir3	P5.4	E	I4	**cf = P10.4**

ランタイム時に、すべてのオペランドは cf のフォーマット／長さに変換されます。続いて変換されたすべての値が比較され、対応する最小／最大値が評価されます。

注：

1. オペランドが 1 つだけ指定された場合、*result-format-length* はこのオペランドのフォーマット／長さになります。
2. 1 ~ 4 の長さのバイナリのオペランドが *MINVAL/*MAXVAL システム関数内の引数として英数字オペランドまたは Unicode オペランドとともに指定されている場合、中間結果（*result-format-length*）は、ダイナミック長の英数字フォーマットまたは Unicode フォーマットで評価されます。

   この場合、バイナリのオペランドの値は数値と見なされ、データ転送の規則に従って *result-format-length* に変換され（バイナリ数値はアンパックフォーマットに変換されます）、最小値／最大値が評価されます。

   例：

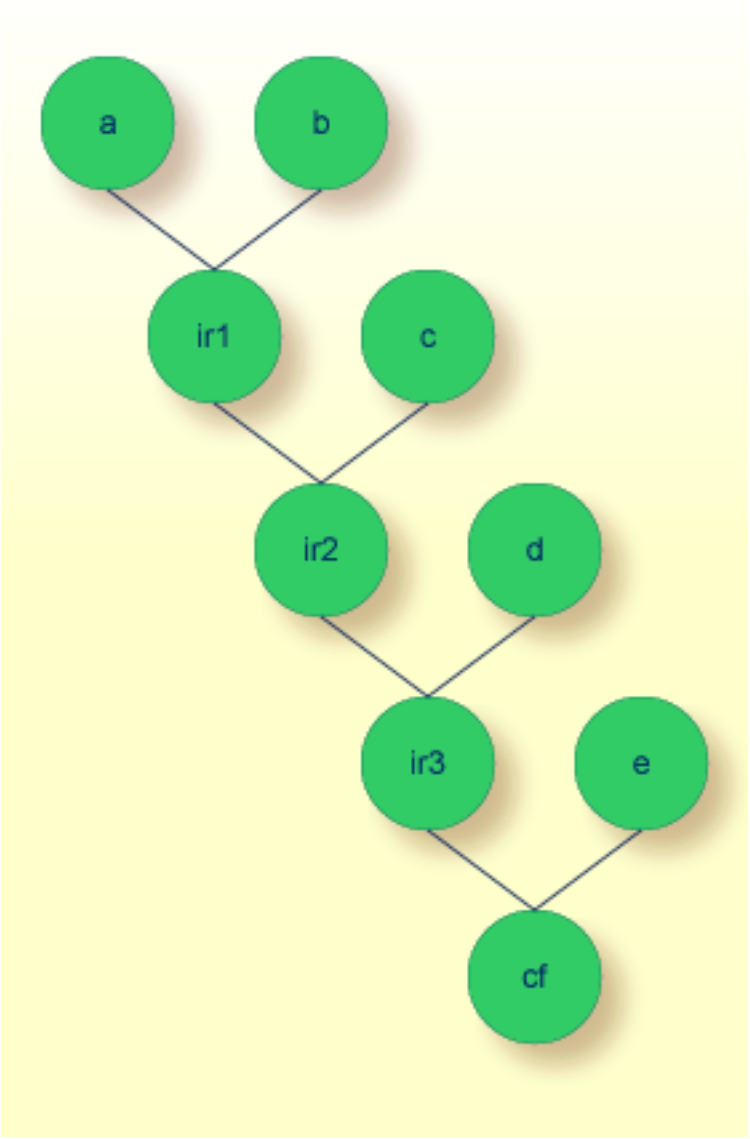
```
DEFINE DATA LOCAL
1 #B4 (B4)  INIT <1>
1 #A10(A10) INIT <"2">
END-DEFINE
WRITE "=" *MAXVAL(#A10, #B4) (AL=60) /* RESULT FORMAT-LENGTH IS (A)DYNAMIC: "2"
WRITE "=" *MINVAL(#A10, #B4) (AL=60) /* RESULT FORMAT-LENGTH IS (A)DYNAMIC: "1"
END
```

   中間 *result-format-length*（#A10、#B4）は A ダイナミックです。

   そのため、まず #A10 が A ダイナミックに変換され、（データ転送の規則を考慮して）#B4 が A ダイナミックに変換された後、両方のオペランドの中間結果が評価されます。

フォーマット／長さの評価順序

下図は、フォーマットと長さが評価される順序を表しています。



凡例：

ir1、ir2、ir3	中間結果 1、2、3 です。
cf	フォーマット／長さの比較結果。

5 *TRANSLATE - 小文字／大文字に変換

```
*TRANSLATE ( operand , { LOWER } )
                       { UPPER }
```

フォーマット／長さ：*operand* と同じ。

関数

Natural システム関数 *TRANSLATE は、英数字またはバイナリのオペランドの文字を大文字または小文字に変換します。オペランドの内容は変更されていません。

*TRANSLATE は、フォーマット A、U、または B のオペランドが許可されるステートメントの任意の位置にオペランドとして指定できます。

制限

システム関数 *TRANSLATE を使用する場合、以下の制限が適用されます。

- *TRANSLATE をターゲット変数が予期される位置に使用しないでください。
- *TRANSLATE は、システム関数でネストすることはできません。

構文説明

オペランド定義テーブル：

オペランド	構文要素					フォーマット											オペランド参照	ダイナミック定義
operand	C	S	A			A	U	B									○	×

構文要素の説明：

構文要素	説明
*TRANSLATE (*operand*,LOWER)	小文字変換 キーワード LOWER が 2 番目の引数として使用される場合、*operand* の文字列は小文字に変換されます。
*TRANSLATE (*operand*,UPPER)	大文字変換 キーワード UPPER が 2 番目の引数として使用される場合、*operand* の文字列は大文字に変換されます。

例

```
DEFINE DATA LOCAL
1 #SRC  (A)DYNAMIC INIT <'aBcDeFg !§$%&/()=?'>
1 #DEST (A)DYNAMIC
END-DEFINE
*
PRINT 'Source string to be translated..........:' #SRC
*
MOVE *TRANSLATE(#SRC, UPPER) TO #DEST
PRINT 'Source string translated into upper case:' #DEST
*
MOVE *TRANSLATE(#SRC, LOWER) TO #DEST
PRINT 'Source string translated into lower case:' #DEST
END
```

出力：

```
Source string to be translated..........: aBcDeFg !§$%&/()=?

Source string translated into upper case: ABCDEFG !§$%&/()=?

Source string translated into lower case: abcdefg !§$%&/()=?
```

6 *TRIM - 先頭または末尾の空白のいずれか、あるいは両方を削除

```
*TRIM ( operand [ , { LEADING  } ] )
                   { TRAILING }
```

フォーマット／長さ：*operand*（A、U、または B）／DYNAMIC と同じ。

関数

Natural システム関数 *TRIM は、英数字またはバイナリの文字列から先頭または末尾の空白（あるいは両方）をすべて削除します。オペランドの内容は変更されていません。ダイナミック変数をオペランドとして使用すると、この変数の長さは結果に従って調整されます。

*TRIM システム関数は、フォーマット A、U または B のオペランドが許可されるステートメントの任意の位置にオペランドとして指定できます。

制限

システム関数 *TRIM を使用する場合、以下の制限が適用されます。

- *TRIM をターゲット変数が予期される位置に使用しないでください。
- *TRIM は、システム関数でネストすることはできません。
- オペランドがスタティック変数であれば、*TRIM を使用して末尾のブランクを削除することはできません。これは、スタティック変数の場合、変数メモリの残りの末尾位置にはスペース文字が入力されるためです。

構文説明

オペランド定義テーブル：

オペランド	構文要素					フォーマット											オペランド参照	ダイナミック定義
operand	C	S	A			A	U	B									○	×

構文要素の説明：

構文要素	説明
*TRIM(*operand*, LEADING)	先頭の空白の削除 キーワード LEADING が 2 番目の引数として使用される場合、*operand* に含まれる文字列から先頭の空白がすべて削除されます。
*TRIM(*operand*, TRAILING)	末尾の空白の削除 キーワード TRAILING が 2 番目の引数として使用される、*operand* に含まれる文字列から末尾の空白がすべて削除されます。
*TRIM(*operand*)	先頭および末尾の両方の空白の削除 2番目の引数としてキーワードを使用しない場合、*operand*に含まれる文字列から先頭と末尾の両方の空白が削除されます。

例

- 例 1 - 英数字引数の使用
- 例 2 - バイナリ引数の使用

例 1 - 英数字引数の使用

```
DEFINE DATA LOCAL
/*******************************
/* STATIC VARIABLE DEFINITIONS
/*******************************
1 #SRC  (A15) INIT <'  ab  CD  '>
1 #DEST (A15)

/* FOR PRINT OUT WITH DELIMITERS
1 #SRC-PRN  (A20)
1 #DEST-PRN (A20)

/*******************************
/* DYNAMIC VARIABLE DEFINITIONS
/*******************************
1 #DYN-SRC (A)DYNAMIC INIT <'  ab  CD  '>
1 #DYN-DEST (A)DYNAMIC

/* FOR PRINT OUT WITH DELIMITERS
1 #DYN-SRC-PRN  (A)DYNAMIC
1 #DYN-DEST-PRN (A)DYNAMIC

END-DEFINE

PRINT 'static variable definition:'
PRINT '---------------------------'
COMPRESS FULL ':' #SRC ':' TO #SRC-PRN LEAVING NO SPACE
```

```
PRINT ' '
PRINT ' 123456789012345      123456789012345'

MOVE *TRIM(#SRC, LEADING) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC, LEADING)'

MOVE *TRIM(#SRC, TRAILING) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC, TRAILING)'

MOVE *TRIM(#SRC) TO #DEST
COMPRESS FULL ':' #DEST ':' TO #DEST-PRN LEAVING NO SPACE
DISPLAY #SRC-PRN #DEST-PRN '*TRIM(#SRC)'

PRINT ' '
PRINT 'dynamic variable definition:'
PRINT '----------------------------'
COMPRESS FULL ':' #DYN-SRC ':' TO #DYN-SRC-PRN LEAVING NO SPACE
PRINT ' '
PRINT ' 1234567890           12345678'

MOVE *TRIM(#DYN-SRC, LEADING) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC, LEADING)'

MOVE *TRIM(#DYN-SRC, TRAILING) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC, TRAILING)'

MOVE *TRIM(#DYN-SRC) TO #DYN-DEST
COMPRESS FULL ':' #DYN-DEST ':' TO #DYN-DEST-PRN LEAVING NO SPACE
DISPLAY (AL=20) #DYN-SRC-PRN #DYN-DEST-PRN '*TRIM(#SRC)'

PRINT ' '
PRINT '":" := delimiter character to show the start and ending of a string!'
END
```

例 1 の出力：

```
#SRC-PRN             #DEST-PRN
-------------------- ----------------------

static variable definition:
----------------------------

 123456789012345         123456789012345
:  ab  CD       :       :ab  CD         :     *TRIM(#SRC, LEADING)
:  ab  CD       :       :  ab  CD       :     *TRIM(#SRC, TRAILING)
:  ab  CD       :       :ab  CD         :     *TRIM(#SRC)

dynamic variable definition:
```

```
----------------------------

 1234567890              12345678
:  ab  CD  :            :ab  CD  :            *TRIM(#SRC, LEADING)
:  ab  CD  :            :  ab  CD:            *TRIM(#SRC, TRAILING)
:  ab  CD  :            :ab  CD:              *TRIM(#SRC)

':' := delimiter character to show the start and ending of a string!
```

例 2 - バイナリ引数の使用

```
DEFINE DATA LOCAL
/*******************************
/* STATIC VARIABLE DEFINITIONS
/*******************************
1 #SRC  (B10) INIT <H'2020FFFF2020FFFF2020'>
1 #DEST (B10)

/*******************************
/* DYNAMIC VARIABLE DEFINITIONS
/*******************************
1 #DYN-SRC (B)DYNAMIC INIT <H'2020FFFF2020FFFF2020'>
1 #DYN-DEST (B)DYNAMIC
END-DEFINE

FORMAT LS=100

PRINT 'static variable definition:
PRINT '----------------------------'
MOVE *TRIM(#SRC, LEADING) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC, LEADING)'

MOVE *TRIM(#SRC, TRAILING) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC, TRAILING)'

MOVE *TRIM(#SRC) TO #DEST
PRINT #SRC #DEST '*TRIM(#SRC)'

PRINT ' '
PRINT 'dynamic variable definition:'
PRINT '----------------------------'

MOVE *TRIM(#DYN-SRC, LEADING) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST '    *TRIM(#SRC, LEADING)'

MOVE *TRIM(#DYN-SRC, TRAILING) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST '    *TRIM(#SRC, TRAILING)'

MOVE *TRIM(#DYN-SRC) TO #DYN-DEST
PRINT #DYN-SRC #DYN-DEST '        *TRIM(#SRC)'
```

```
PRINT ' '

PRINT 'hex."20" := space character'
END
```

例 2 の出力：

```
static variable definition:
---------------------------

2020FFFF2020FFFF2020 0000FFFF2020FFFF2020     *TRIM(#src, leading)
2020FFFF2020FFFF2020 00002020FFFF2020FFFF     *TRIM(#src, trailing)
2020FFFF2020FFFF2020 00000000FFFF2020FFFF     *TRIM(#src)

dynamic variable definition:
---------------------------

2020FFFF2020FFFF2020 FFFF2020FFFF2020         *TRIM(#src, leading)
2020FFFF2020FFFF2020 2020FFFF2020FFFF         *TRIM(#src, trailing)
2020FFFF2020FFFF2020 FFFF2020FFFF             *TRIM(#src)

hex.'20' := space character
```

7 POS - フィールド ID 関数

フォーマット／長さ：	I4

システム関数 POS(*field-name*) は、システム関数で指定される名前を持つフィールドの内部識別子を返します。返される値は、フィールドアドレスの内部表現です。

POS(*field-name*) は、マップ内の位置と関係なく、特定のフィールドを識別するのに使用できます。これは、マップ内のフィールドのシーケンスと数が変更されても、POS(*field-name*) は引き続き同じフィールドを一意に識別できることを意味します。これにより、例えば、プログラムロジックに依存しているとフィールドに MARK を付ける場合、必要なのは 1 つの REINPUT ステートメントのみになります。

例：

```
DECIDE ON FIRST VALUE OF ...
   VALUE ...
      COMPUTE #FIELDX = POS(FIELD1)
   VALUE ...
      COMPUTE #FIELDX = POS(FIELD2)
   ...
END-DECIDE
...
REINPUT ... MARK #FIELDX
```

POS で指定されたフィールドが配列の場合、POS(FIELDX(5)) のように特定のオカレンスを指定する必要があります。POS を配列範囲に適用することはできません。

注意: POS は同じ格納場所で開始する 2 つの異なる変数（REDEFINE 変数）を区別することはできません。これは、POS によって返される内部フィールドのアドレスが両方とも同じであるためです。

POS および *CURS-FIELD

システム関数 POS(*field-name*) は、カーソルが現在位置づけられているフィールドに応じて特定の機能を実行するために、Natural システム変数 *CURS-FIELD と組み合わせて使用できます。

*CURS-FIELD は、カーソルが現在位置づけられているフィールドの内部識別子を持ちます。これは単体では使用できず、POS(*field-name*) と組み合わせて使用する必要があります。これらを使用して、現在カーソルが特定のフィールドに置かれているかどうかを確認し、その状況に応じて処理を実行できます。

例：

```
IF *CURS-FIELD = POS(FIELDX)
   MOVE *CURS-FIELD TO #FIELDY
END-IF
...
REINPUT ... MARK #FIELDY
```

注意:

1. *CURS-FIELD および POS(*field-name*) の値は、フィールドの内部識別子としてのみ機能し、算術演算に使用することはできません。
2. 配列の次元のオカレンス数が EXPAND、RESIZE、または REDUCE ステートメントを使用して変更された後には、X-array（最低でも 1 次元の最低でも 1 つの境界が拡張可能として定義されている配列）のオカレンスに対して POS(*field-name*) から返される値は変わることがあります。
3. Natural RPC：*CURS-FIELD および POS(*field-name*) がコンテキスト変数を参照する場合、結果の情報は同じ会話内でのみ使用できます。
4. Natural for Ajax アプリケーションの場合、入力にフォーカスがあるコントロールの値を表すオペランドが、*CURS-FIELD によって識別されます。*CURS-FIELD を POS 関数と併せて使用することにより、入力にフォーカスがあり、その条件に応じて処理を実行するコントロールをチェックできます。
5. *CURS-FIELD と POS(*field-name*) は、同じ格納場所で開始する 2 つの異なる変数（REDEFINE 変数）を区別することはできません。これは、*CURS-FIELD と POS(*field-name*) によって返される内部フィールドのアドレスが両方とも同じ変数であるためです。

『*プログラミングガイド*』の「*ダイアログの設計*」、「*フィールドに基づいた処理*」、および「*プログラミングの単純化*」も参照してください。

8 RET - リターンコード関数

フォーマット／長さ：	I4

システム関数 RET(*program-name*) を使用して、CALL ステートメントで呼び出された Natural 以外のプログラムからのリターンコードを受け取ることができます。

RET(*program-name*) は、IF ステートメントおよび算術ステートメント ADD、COMPUTE、DIVIDE、MULTIPLY および SUBTRACT で使用できます。

例：

```
DEFINE DATA LOCAL
1 #RETURN (I4)
...
END-DEFINE
...
...
CALL 'PROG1'
IF RET('PROG1') > #RETURN
   WRITE 'ERROR OCCURRED IN PROGRAM 1'
END-IF
...
```

9 SORTKEY ソートキー関数

SORTKEY (*character-string*)

このシステム関数は、「不正にソートされた」 文字（または文字の組み合わせ）を、ソートプログラムまたはデータベースシステムでアルファベット順に 「正しくソートされる」 他の文字（または文字の組み合わせ）に変換するために使用します。

フォーマット／長さ：	A253

国によっては、ソートプログラムまたはデータベースシステムによって正確なアルファベット順にソートされない文字（または文字の組み合わせ）が言語に含まれます。コンピュータで使用される文字セットの文字のシーケンスが必ずしもアルファベット順に相当するとは限らないためです。

例えば、スペイン語の "CH" は、ソートプログラムやデータベースシステムでは 2 文字の文字列とみなされ、"CG" と "CI" の間にソートされます。しかし、実際、スペイン語では、これは "C" と "D" の間に属する 1 文字となります。

または、要求に反して、小文字と大文字がソート順で同等に扱われないこともあります。文字は（数字の前にソートされることを希望しているにもかかわらず）数字の後にソートされることもあります。あるいは、特殊文字（例：ダブル名のハイフン）によって予期しないソート順が生じることもあります。

このような場合、システム関数 SORTKEY(*character-string*) を使用できます。SORTKEY で計算された値はソート条件にしか使用できず、エンドユーザーとのやり取りには元の値が使用されます。

COMPUTE ステートメントおよび論理条件の算術演算オペランドとして SORTKEY 関数を使用できます。

character-string として、英数字定数や変数、または英数字配列の単一オカレンスを指定できます。

Natural プログラムに SORTKEY 関数を指定すると、ユーザー出口 NATUSK*nn* が呼び出されます。*nn* は現在の言語コード（システム変数 *LANGUAGE の現在の値）です。

このユーザー出口は、標準 CALL インターフェイスを提供する任意のプログラミング言語で記述できます。SORTKEY に指定された *character-string* は、そのユーザー出口に渡されます。ユーザー出口は、この文字列内の「不正にソートされた」文字を「正しくソートされた」文字に変換するようにプログラミングされている必要があります。続けて、変換された文字列がNaturalプログラムで使用され、さらに処理が実行されます。

外部プログラムの一般的な呼び出し規則については、CALL ステートメントの説明を参照してください。

ユーザー出口の呼び出し規則の詳細については、「*ユーザー出口*」を参照してください。

例：

```
DEFINE DATA LOCAL
1 CUST VIEW OF CUSTOMERFILE
  2 NAME
  2 SORTNAME
END-DEFINE
...
*LANGUAGE := 4
...
REPEAT
  INPUT NAME
  SORTNAME := SORTKEY(NAME)
  STORE CUST
  END TRANSACTION
  ...
END-REPEAT
...
READ CUST BY SORTNAME
  DISPLAY NAME
END-READ
...
```

上記の例では、INPUT ステートメントの繰り返しの実行で、次の値が入力されるものと仮定しています。"Sanchez", "Sandino" および "Sancinto".

SORTNAME への SORTKEY(NAME) の割り当てで、ユーザー出口 NATUSK04 が呼び出されます。このユーザー出口は、最初にすべての小文字を大文字に変換し、次に文字の組み合わせ "CH" を "C*x*" に変換します。この場合、*x* は使用する文字セットの最終文字、つまり 16 進の H'FF'（この最終文字は出力不可能な文字とみなされる）に対応します。

「元」の名前（NAME）は、希望するソート（SORTNAME）に使用する変換済みの名前とともに保存されます。ファイルを読み込むには、SORTNAME を使用します。DISPLAY ステートメントは、次のように正しいスペイン語のアルファベット順に名前を出力します。

```
Sancinto
Sanchez
Sandino
```

III Natural オブジェクトとして提供されている関数

10 Natural オブジェクトとして提供されている関数

このドキュメントでは、タイプがファンクションのNaturalオブジェクトを使用して実装される関数について説明します。

これらのファンクションオブジェクト（およびそのプロトタイプ定義）のうち、その名前が SAG で始まるものは、システムファイル FNAT の Natural システムライブラリ SYSTEM で提供されています。システムライブラリ SYSEXPG には、サンプルファンクションコールが用意されています。

ファンクションコールの詳細については、『*プログラミングガイド*』ドキュメントの関連セクションを参照してください。

URL のエンコード

Natural アプリケーションを HTTP 要求とインターフェイスで接続する場合、URI（Uniform Resource Identifier）が URL エンコードされている必要があります。REQUEST DOCUMENT ステートメントには、ドキュメントにアクセスするためにこのような URL が必要です。

URL エンコード（またはパーセントのエンコーディング）とは、URL の一部の特殊文字を置き換えるメカニズムです。US-ASCII 文字セットの文字のみを使用して URL を形成できます。US-ASCII 文字セットの一部の文字は、URL で使用される場合に特別な意味を持ちます。これらの文字は「予約されている」制御文字として分類され、URL 文字列を異なる意味を持つサブコンポーネントに構造化します。URL の一般的な構文に関する準標準は、RFC 3986（インターネットコミュニティで構成されるドキュメント）に規定されています。URL エンコードが必要な条件について説明しています。これには、US-ASCII 文字セット（ユーロ記号など）に含まれていない文字の表現、および予約されている文字の使用についての説明が含まれます。

予約されている文字は以下のとおりです。

?	=	&	#	!	$	%	'	(	)	*	+	,	/	:	;	@	[	]

予約されていない文字は以下のとおりです。

A～Z	a～z	0-9	-	_	.	~

URL には、予約されている文字と予約されていない文字のみを使用できます。他の文字は使用できません。（予約されている文字と予約されていない文字のいずれにも該当しない）他のバイト値が必要な場合、または予約されている文字が（URL コンテキストで特別な意味を持たない）データとして使用されている場合、Windows-1252 エンコードスキーマにより、直後にコードポイントを表す 2 桁の 16 進数が続くこれらのコードは、「%-encoding」形式（パーセント記号）に変換する必要があります。これにより、文字列でプラス記号（+）が %2B、パーセント記号（%）が %25、アットマーク（@）が %40 として表示されます。

以下のエンコード関数は、入力文字列全体を処理します。パーセント形式に変換してはいけない制御文字（予約されている文字）が含まれている場合、URL 全体またはその一部をエンコード

しないでください。これらの関数は、URL での使用を許可されていない文字、およびデータ項目として提供される URL コンテキスト内で特別な意味を持つ文字にのみ適用する必要があります。

単一のエンコード

単一の入力パラメータに、エンコードまたはデコードされる文字列が含まれます。内部のすべてのデータは、現在どのセッションコードページが実際にアクティブになっているかにかかわらず、コードページ Windows-1252 で表されているものと見なされます。SAGENC/SAGDEC 関数の実行には Unicode のサポートは必要ありません。以下の文字は、対応する US-ASCII 16 進数で置き換えられます。

文字	<	(	+	|	&	!	$	*	)	;	/	,	%	>	?	`	:	#	@	'	=	"	^	[	]	{	}	\
%*nn* にエンコードされます	3C	28	2B	7C	26	21	24	2A	29	3B	2F	2C	25	3E	3F	60	3A	23	40	27	3D	22	5E	5B	5D	7B	7D	5C

さらにスペースはプラス記号（+）に置き換えられます。他のすべての文字は元のままで変換されません。単一のエンコード関数は、ほとんどの場合、URL エンコードで十分です。返される結果フィールドのフォーマットは、A ダイナミックです。

次の機能を使用できます。

- SAGENC - 単一のエンコード（フォーマット A からフォーマット A）
- SAGDEC - 単一のデコード（フォーマット A からフォーマット A）

SAGENC - 単一のエンコード（フォーマット A からフォーマット A）

この関数 SAGENC は、文字列をパーセントエンコード形式にエンコードします。標準 RFC3986 に従い、（URL では許可されていない）US-ASCII x'7F' 未満の文字と予約されている文字はパーセントでエンコードされ、スペース文字はプラス記号（+）に置き換えられます。予約されていない文字は RCF3986 に準拠し、（ドイツ語のウムラウトなど）US-ASCII x'7F' より上の文字はエンコードされません。このような文字をエンコードする場合は、拡張エンコード関数 SAGENCE を使用します。

オブジェクト	説明
SAGENC	これは単一のエンコードファンクションコールです。
SAGENCP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAGENCP はオプションです。
URLX01	ライブラリ SYSEXPG に含まれるプログラム例。 `#URL-ENC := SAGENC(<#URL-DEC>)`

SAGDEC - 単一のデコード（フォーマット A からフォーマット A へ）

関数 SAGDEC によって、関数 SAGENC が提供したパーセントのエンコーディングがデコードされます。入力パラメータはデコード文字列以外に必要ありません。

オブジェクト	説明
SAGDEC	これは単一のデコードファンクションコールです。
SAGDECP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAGDECP はオプションです。
URLX01	ライブラリ SYSEXPG に含まれるプログラム例。 `#URL-DEC := SAGDEC(<#URL-ENC>) ↩`

拡張エンコード

拡張関数は、RFC3986で指定または推奨されているすべての問題を考慮します。以下のパラメータが考慮されます（デフォルト設定は太字で表示）。

1. エンコード／デコードされる *<dynamic U-string>*
2. リターンコード：<>0（Naturalエラー）MOVE ENCODEDステートメントにエラーがある場合。
3. リターンコード <> 0 の場合エラー文字
4. スペース文字：%20/+/エンコードしない（デフォルト：+）
5. 予約されていない文字：エンコード／エンコードしない
6. 予約されている文字：エンコード／エンコードしない
7. その他の特殊文字（予約されていない文字、予約されている文字のいずれでもない）：エンコード／エンコードしない
8. 文字のパーセントによるエンコード：ISO-8859-1/UTF-8/その他のコードページ/if='' then *CODEPAGE（デフォルトのエンコードコードコードページではなく、デフォルトの Natural コードページ）
9. フォーマットUのX-arrayでユーザーが選択した文字。この文字は、上記のパラメータに従ったパーセントによるエンコードは行われません。例えば、ISO-8859-1 コードページにはないユーロ記号の場合、または文字のパーセントによるエンコードを防ぐ場合などです。
10. X-array と同じオカレンスにある、ユーザーが選択した文字に対する、フォーマット A のX-array 内でのユーザー定義のパーセントによるエンコード。

文字列の入力パラメータは Natural フォーマット U になります。これは、入力文字列にはすべてのUnicode文字を含めることができることを意味します。拡張関数の出力文字列は、Naturalデフォルトコードページ（*CODEPAGE）のフォーマットAです。パーセントによるエンコードのコードページを選択できます。UTF-8、ISO-8859-1、ユーロ記号のパーセントによるエンコードは、MOVE ENCODED ステートメントによって実行されます。パーセントによるエンコードに使用されるターゲットコードページに入力文字が存在しない場合、その文字はエンコードされません。これは、デフォルトのNaturalコードページで文字が変更されずに返されることを意味します。文字がデフォルトの Natural コードページに存在しない場合、その文字は MOVE ENCODED ステートメントによって返される置換文字に置き換えられます。置換文字はパーセントによりエンコードされます。これは、パーセントによるエンコードのコードページが UTF-8 でない場合にのみ発生する可能性があります。最後の MOVE ENCODED エラーが返されます。

パラメータはオプションのパラメータです。ユーザーがパラメータを指定しない場合は、デフォルト値が使用されます。ユーザーが独自の文字変換テーブルを指定した場合、テーブル内の文字は、他のパラメータではなく、このテーブルに従ってパーセントによりエンコードされます。ユーザー定義の変換テーブルの文字をパーセントによりエンコードしたものが同じ文字または空白に等しい場合、この文字はエンコードされません。このようにして、予約されている文字セットまたは予約されていない文字セットから単一の文字を除外できます。

次の機能を使用できます。

- SAGENCE - 拡張エンコード（フォーマット U からフォーマット A、オプションのパラメータ）
- SAGDECE - 拡張デコード（フォーマット A からフォーマット U、オプションのパラメータ）

SAGENCE-拡張エンコード（フォーマットUからフォーマットA、オプションのパラメータ）。

関数 SAGENCE は選択したコードページ（デフォルトは UTF-8）の 16 進値を使用して文字列をパーセントによりエンコードします。標準 RFC3986 に従い、予約されている文字と（URL では許可されていない）US-ASCII x'7F' 未満の文字はパーセントによりエンコードされます。また、スペースとパーセント記号（%）もエンコードされます。

さらに、RCF3986 に準拠した予約されていない文字と、ドイツ語のウムラウトなどの US-ASCII x'7F' より上位の文字がこの関数によってエンコードされます。

SAGENCE には Natural Unicode のサポートが必要です。

オブジェクト	説明
SAGENCE	これは拡張エンコードファンクションコールです。
	Parameters:
	P-DEC-STR-E (U) P-RET (I4) OPTIONAL /* 0: ok /* else: Natural error returned /* by the GIVING clause of /* MOVE ENCODED. /* This is the error which /* comes up when a character /* cannot be converted into /* the target code page. /* Error strategy: /* Step 1: If a character shall be %-encoded and is not available /* in the code page for %-encoding, the character will not be /* %-encoded. It will be copied. /* Step 2: If a character will not be %-encoded but copied from the /* input format U-variable to a format A-variable (in *CODEPAGE) /* and the character is not available in *CODEPAGE, a substitution /* character will be used instead. The substitution character will /* be %-encoded. /* The last error will be returned in P-RET. P-ERR-CHAR (U1) OPTIONAL /* Character causing the error P-SPACE (A1) OPTIONAL /* '%' => %20 /* ' ' => ' ' /* else => '+' (default) P-UNRES (A1) OPTIONAL /* 'E' => encode /* else => don't encode (default) P-RES (A1) OPTIONAL /* 'E' => encode (default) /* else => don't encode P-OTHER (A1) OPTIONAL /* 'E' => encode (default)

オブジェクト	説明
	/* else => don't encode P-CP (A64) OPTIONAL /* IANA name e.g. UTF-8 (default) /* or ISO-8859-1 /* On mainframe only code page names defined with the macro NTCPAGE /* in the source module NATCONFG can be used. Other code page names /* are rejected with a corresponding runtime error. /* P-CP-TABLE-CHAR(U1/1:*) OPTIONAL /* user selected char to be /* %-encoded, e.g. 'ö' or '/' P-CP-TABLE-ENC (A12/1:*) OPTIONAL /* user %-encoding /* e.g. character 'ö' /* '%F6' -> ISO-8859-1 /* '%C3%B6' -> UTF-8 /* e.g. character '/' /* '/' -> '/' not encoded /* although P-RES = 'E' /* Characters in this table will be encoded according to the /* specifed %-encoding. If the U12 encoding part is blank (space /* according to *CODEPAGE) or the P-CP-TABLE-ENC value is equal to /* the character, then the character will not be encoded at all. /*
SAGENCEP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAGENCEP はオプションです。
URLX01	ライブラリ SYSEXPG に含まれるプログラム例。
	サンプル呼び出し 以下のデフォルト値が使用されます。 #URL-ENC := SAGENCE(<#URL-DEC-U>) 使用可能なパラメータがすべて指定されます。 #URL-ENC := SAGENCE(<#URL-DEC-U,L-RET,L-ERR-CHAR,L-SPACE,L-UNRES, ↩ L-RES,L-OTHER,L-CP,L-CP-TAB-CHAR(*),L-CP-TAB-ENC(*) >)

SAGDECE - 拡張デコーディング（フォーマット A からフォーマット U、オプションのパラメータ）

関数 SAGDECE によって、関数 SAGENCE が提供したパーセントのエンコーディングがデコードされます。スペース文字および／またはコードページが指定されている場合、値はエンコードに指定されている値と同じであることが必要です。

SAGDECE には Natural Unicode のサポートが必要です。

オブジェクト	説明
SAGDECE	これは拡張デコーディングファンクションコールです。

	Parameters:
	`1 P-ENC-STR-E (A)`
	`1 P-RET (I4) OPTIONAL /* 0: ok`
	`/* else: Natural error returned`
	`/* by the GIVING clause of`
	`/* MOVE ENCODED.`
	`/* This error comes up`
	`/* when a %-encoded`
	`/* character cannot be`
	`/* converted into the`
	`/* target code page.`
	`/* The last error will be returned in P-RET.`
	`1 P-ERR-CHAR (A12) OPTIONAL /* Error character %-encoded`
	`1 P-SPACE (A1) OPTIONAL /* ' ' => ' '`
	`/* else => '+' (default)`
	`1 P-CP (A64) OPTIONAL /* IANA name e.g. UTF-8 (default)`
	`/* or ISO-8859-1`
	`/* On mainframe only code page names defined with the macro NTCPAGE`
	`/* in the source module NATCONFG can be used. Other code page names`
	`/* are rejected with a corresponding runtime error.`
	`/*`

SAGDECEP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAGDECEP はオプションです。
URLX01	ライブラリ SYSEXPG に含まれるプログラム例。
	サンプル呼び出し 以下のデフォルト値が使用されます。

オブジェクト	説明

	`#URL-DEC-U := SAGDECE(<#URL-ENC>)`
	使用可能なパラメータがすべて指定されます。
	`#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,L-ERR-CHAR-DEC,L-SPACE,L-CP>)`

Example Program

ライブラリ SYSEXPG に含まれるプログラム例。

```
** Example 'URLX01': ENCODED-STR := SAGENC(<DECODED-STR>)
************************************************************************
DEFINE DATA
LOCAL
1 SAMPLE-STRING (A72)
/*
1 #URL-DEC      (A) DYNAMIC
1 #URL-ENC      (A) DYNAMIC
/*
1 #URL-DEC-U    (U) DYNAMIC
/*
1 L-RET         (I4)   /* Return code
1 L-ERR-CHAR    (U1)   /* Error character
1 L-ERR-CHAR-DEC(A12)  /* Decoded error character
1 L-SPACE       (A1)   /* '%'  => %20, ' '  => ' ',
                       /* else => '+' (default)
1 L-UNRES       (A1)   /* 'E' => encode, else => don't encode (default)
1 L-RES         (A1)   /* 'E' => encode (default), else => don't encode
1 L-OTHER       (A1)   /* 'E' => encode (default), else => don't encode
1 L-CP          (A64)  /* default *CODEPAGE
1 L-CP-TAB-CHAR (U1/1:1)
1 L-CP-TAB-ENC  (A12/1:1)
1 L-MSG         (U72)
END-DEFINE
/*
/*
/*
WRITE 'Sample string to be processed:'
/* The string below shall be encoded and decoded again.
/* After decoding it should be unchanged.
SAMPLE-STRING := '"Decoded data!"'
WRITE SAMPLE-STRING (AL=72) /
/*
/* Assign the sample string to the input variable #URL-DEC of the
/* simple encoding function.
#URL-DEC      := SAMPLE-STRING
/*
```

```
/* Copycode SAGENCP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGENCP is optional.
INCLUDE SAGENCP
/*
/* SAGENC(<#URL-DEC>) is the simple encoding function call.
/*
/* Function SAGENC %-encodes a string to code page ISO-8859-1.
/* According to standard RFC3986 reserved characters and characters
/* below US-ASCII x'7F' which are not allowed in a URL will be
/* %-encoded.
/* Also the space and the percent sign will be encoded.
/* Unreserved characters according to RCF3986 and characters above
/* US-ASCII x'7F' will not be encoded. If you want to encode such
/* characters, use the extended encoding function.
/*
/* ---- Space                         ' ' -> '+'
/* ---- Percent sign                  '%' -> '%25'
/*
/* Unreserved characters according to RFC3986 (will not be encoded!):
/* ---- Period (fullstop)             '.' -- '%2E'
/* ---- Tilde                         '~' -- '%7E'
/* ---- Hyphen                        '-' -- '%2D'
/* ---- Underscore character          '_' -- '%5F'
/* ---- digits, lower and upper case characters
/* ---- 0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
/*
/* Reserved characters according to RFC3986:
/* ---- Exclamation mark              '!' -> '%21'
/* ---- Number sign                   '#' -> '%23'
/* ---- Dollar sign                   '$' -> '%24'
/* ---- Ampersand                     '&' -> '%26'
/* ---- Apostrophe                    ''' -> '%27'
/* ---- Left parenthesis              '(' -> '%28'
/* ---- Right parenthesis             ')' -> '%29'
/* ---- Asterisk                      '*' -> '%2A'
/* ---- Plus sign                     '+' -> '%2B'
/* ---- Comma                         ',' -> '%2C'
/* ---- Reverse solidus (backslash) '/' -> '%2F'
/* ---- Colon                         ':' -> '%3A'
/* ---- Semi-colon                    ';' -> '%3B'
/* ---- Equals sign                   '=' -> '%3D'
/* ---- Question mark                 '?' -> '%3F'
/* ---- Commercial at                 '@' -> '%40'
/* ---- Square bracket open           '[' -> '%5B'
/* ---- Square bracket close          ']' -> '%5D'
/*
/* Other characters below x'7F' (US-ASCII) but not allowed in URL
/* ---- Quotation mark                '"' -> '%22'
/* ---- Less than                     '<' -> '%3C'
/* ---- Greater than                  '>' -> '%3E'
```

```
/* ---- Reverse solidus (backslash) '\' -> '%5C'
/* ---- Accent, Circumflex          '^' -> '%5E'
/* ---- Accent, Grave               '`' -> '%60'
/* ---- Opening brace               '{' -> '%7B'
/* ---- Vertical bar                '|' -> '%7C'
/* ---- Closing brace               '}' -> '%7D'
/*
#URL-ENC := SAGENC(<#URL-DEC>)
/*
/*
WRITE 'Simple function, encoded:'
WRITE #URL-ENC (AL=72)
/*
/* Copycode SAGDECP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGDECP is optional.
INCLUDE SAGDECP
/*
/* SAGDEC(<#URL-ENC>) is the simple decoding function call.
/* It decodes the above described %-encodings.
/*
#URL-DEC := SAGDEC(<#URL-ENC>)
/*
/*
/* The result after encoding and decoding must be equal to the original
/* SAMPLE-STRING.
WRITE 'Simple function, decoded:'
WRITE #URL-DEC (AL=72)
/*
/*
/*
WRITE /
/*
/*
/*
/* Assign the sample string to the input variable #URL-DEC-U of the
/* enhanced encoding function.
#URL-DEC-U := SAMPLE-STRING
/*
/* Copycode SAGENCEP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGENCEP is optional.
INCLUDE SAGENCEP
/*
/* This is the enhanced encoding function call.
/* The way, characters will be %-encoded dependes on the input
/* parameter of the function.
/* The parameters of the encoding and decoding function are preset
/* with the default values.
/* L-CP-TAB-CHAR(*) and L-CP-TAB-ENC(*) don't have default values.
```

```
/* L-CP-TAB-CHAR(1) = 'ä' and L-CP-TAB-ENC(1) = '%C3%A4' will not be
/* used for the sample string '"Decoded data!"'. The string does not
/* contain an 'ä'.
L-SPACE     := '+'           /* encoding and decoding
L-UNRES     := 'D'           /* encoding only
L-RES       := 'E'           /* encoding only
L-OTHER     := 'E'           /* encoding only
L-CP        := 'UTF-8'       /* encoding and decoding
                             /* e.g. ISO-8859-1, UTF-16BE, UTF-32BE
L-CP-TAB-CHAR(1) := 'ä'      /* encoding only
L-CP-TAB-ENC (1) := '%C3%A4' /* encoding only
/*
/* Note that all possible parameters are specified for this sample
/* call.
/* If the default values shall be used and no return code is wanted,
/* all parameters can be omitted, besides the string #URL-DEC-U.
/*
#URL-ENC := SAGENCE(<#URL-DEC-U,L-RET,L-ERR-CHAR,L-SPACE,L-UNRES,
  L-RES,L-OTHER,L-CP,L-CP-TAB-CHAR(*),L-CP-TAB-ENC(*) >)
WRITE 'Extended function, encoded:'
WRITE #URL-ENC (AL=72)
IF L-RET NE 0 THEN
  /* If L-RET = 0, the function worked ok. Else L-RET contains the
  /* Natural error returned by the GIVING clause of MOVE ENCODED.
  /* The error comes up when a character cannot be converted into
  /* the target codepage, e.g. because a character does not exist
  /* in the target codepage.
  COMPRESS 'Error' L-RET 'with MOVE ENCODED of' L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
END-IF
/*
/* Copycode SAGDECEP containing the prototype definition is used at
/* compilation time only in order to determine the type of the return
/* variable for function call reference and to check the parameters,
/* if this is desired. SAGDECEP is optional.
INCLUDE SAGDECEP
/*
/* This is the 1st enhanced decoding function call with 5 parameters.
/* Note that all possible parameters are specified for this sample
/* call.
/* Since the parameters have the default values, the subsequent
/* function calls return the same result although parameters
/* have been omitted.
#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,L-ERR-CHAR-DEC,L-SPACE,L-CP>)
WRITE 'Extended function, decoded:'
WRITE #URL-DEC-U (AL=72)
IF L-RET NE 0 THEN
  /* If L-RET = 0, the function worked ok. Else L-RET contains the
  /* Natural error returned by the GIVING clause of MOVE ENCODED.
  /* The error comes up when a %-encoded character cannot be converted
  /* into the target codepage, e.g. because a character does not exist
  /* in the target codepage.
```

```
  COMPRESS 'Error' L-RET 'with MOVE ENCODED of' L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
  RESET L-RET
END-IF
/*
/* This is the 2nd enhanced decoding function call with one parameter.
#URL-DEC-U := SAGDECE(<#URL-ENC>)
WRITE #URL-DEC-U (AL=72)
/* L-RET will not be returned
/*
/* This is the 3rd enhanced decoding function call with 3 parameters.
#URL-DEC-U := SAGDECE(<#URL-ENC,L-RET,2X,L-CP>)
WRITE #URL-DEC-U (AL=72)
IF L-RET NE 0 THEN
  COMPRESS 'Error' L-RET 'with MOVE ENCODED of' L-ERR-CHAR INTO L-MSG
  WRITE L-MSG
  RESET L-RET
END-IF
/*
END
```

Base64 エンコード

このセクションでは、Base64 変換を使用して、バイナリデータを出力可能なネットワーク互換データに、またはその逆方向に変換するために使用できる Natural 関数について説明します。

Base64 変換とは、フォーマット B から フォーマット A に変換し、フォーマット A からフォーマット B に戻すことを意味します。ここで、6（バイナリ）ビットは 8（英数字）ビットに変換されます。例えば、B3 の値は A4 の値に変換されます。

注意: 各バイナリ値は、あいまいでない英数字の値に変換されます。この英数字の値を再変換すると、元の 2 進値になります。ただし、フォーマット A からフォーマット B、フォーマット B からフォーマット A への変換のほとんどが、これに該当しません。

この変換は、TCP/IP 経由で .bmp ファイルを転送したり、ユーティリティプロトコルを介して Natural バイナリ値や整数値を転送したりするために使用できます。

Open Systems のみ：次の 3 つのモードを使用できます。RFC3548、RFC2045、および NATRPC（デフォルト）。NATRPC は、変換が NATRPC ロジックに従って実行されることを意味します。これは 100% メインフレーム対応です。RFC2045 は、CMBASE64 コールのデフォルトです。RFC3548 は NATRPC と同じですが、必要のない英数字バイトには等号（=）が入力されます。

次の機能を使用できます。

- SAG64BA - バイナリから英数字への変換
- SAG64AB - 英数字からバイナリへの変換

これらの 2 つの関数は、ライブラリ SYSEXT に提供される Natural アプリケーションプログラムインターフェイス USR4210N と同じ機能を提供します。

SAG64BA - バイナリから英数字への変換

この関数 SAG64BA は、Base64 エンコードを使用して、バイナリデータを出力可能なネットワーク互換データに変換します。

オブジェクト	説明
SAG64BA	これは、バイナリから英数字フォーマットへの変換関数です。
	Parameters:
	1 PARM-B (B) DYNAMIC BY VALUE /* Binary source input/target output 1 PARM-RC (I4) OPTIONAL /* 0: ok /* Mainframe /* 1 Source is not numeric /* 2 Source is not packed /* 3 Source is not floating point /* 4 Overflow, source doesn't fit into target /* 5 Integer overflow /* 6 Source is not a valid date or time /* 7 Length error (hex input not even) /* 8 Target precision is less than source precision /* 9 Float underflow (result->0) /* 10 Alpha source contains non-hex characters /* 20 Invalid function code /* Open Systems /* 1 Invalid value for RFC parameter /* 2 Invalid function code /* 3 CMBASE64: Overflow, source doesn't fit into /* target /* 4 CMBASE64: Non-base64 character found in encoded /* data /* 5 CMBASE64: Out of memory /* 6 CMBASE64: Invalid number of parameters /* 7 CMBASE64: Invalid parameter type /* 8 CMBASE64: Invalid parameter length /* 9 CMBASE64: Invalid function code /* 10 CMBASE64: Unkown return code 1 PARM-ERRTXT (A72) OPTIONAL /* blank, if ok no error /* else error text 1 PARM-RFC (B1) OPTIONAL /* OS only, not used for MF /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;

オブジェクト	説明
SAG64BAP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAG64BAP はオプションです。

B64X01	ライブラリ SYSEXPG に含まれるプログラム例。
	
	以下のデフォルト値が使用されます。
	
	`PARM-A := SAG64BA(<PARM-B>)`
	
	使用可能なパラメータがすべて指定されます（PARM-RFC はメインフレームに適用されません）。
	
	`PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)`

SAG64AB - 英数字からバイナリへの変換

この関数 SAG64AB は、Base64 エンコードを使用して、出力可能なネットワーク互換データをバイナリデータに変換します。

オブジェクト	説明
SAG64AB	これは、英数字からバイナリフォーマットへの変換関数です。
	Parameters:

	`1 PARM-A (A)`
	`/* Alpha source input/target output`
	`1 PARM-RC (I4) OPTIONAL`
	`/* 0: ok`
	`/* Mainframe`
	`/* 1 Source is not numeric`
	`/* 2 Source is not packed`
	`/* 3 Source is not floating point`
	`/* 4 Overflow, source doesn't fit into target`
	`/* 5 Integer overflow`
	`/* 6 Source is not a valid date or time`
	`/* 7 Length error (hex input not even)`
	`/* 8 Target precision is less than source precision`
	`/* 9 Float underflow (result->0)`
	`/* 10 Alpha source contains non-hex characters`
	`/* 20 Invalid function code`
	`/* Open Systems`
	`/* 1 Invalid value for RFC parameter`
	`/* 2 Invalid function code`
	`/* 3 CMBASE64: Overflow, source doesn't fit into`

オブジェクト	説明
	/* target /* 4 CMBASE64: Non-base64 character found in encoded /* data /* 5 CMBASE64: Out of memory /* 6 CMBASE64: Invalid number of parameters /* 7 CMBASE64: Invalid parameter type /* 8 CMBASE64: Invalid parameter length /* 9 CMBASE64: Invalid function code /* 10 CMBASE64: Unkown return code 1 PARM-ERRTXT (A72) OPTIONAL /* blank, if ok no error /* else error text 1 PARM-RFC (B1) OPTIONAL /* OS only, not used for MF /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;
SAG64ABP	プロトタイプ定義を含むコピーコードは、必要に応じて、ファンクションコール参照の戻り変数のタイプを決定してパラメータをチェックするのみの目的で、コンパイル時に使用されます。 SAG64ABP はオプションです。
B64X01	ライブラリ SYSEXPG に含まれるプログラム例。 以下のデフォルト値が使用されます。 PARM-B := SAG64AB(<PARM-A>) 使用可能なパラメータがすべて指定されます（PARM-RFC はメインフレームに適用されません）。 PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)

Example Program

ライブラリ SYSEXPG に含まれるプログラム例 B64X01。

```
** Example 'B64X01': BASE64-A-STR := SAG64BA(<BASE64-B-STR>)
************************************************************************
* Function ......... Convert binary data into printable,
*                     network-compatible data or vice versa using
*                     Base64 encoding.
*
*                     Base64 encoding means (B) -> (A) -> (B),
*                     where 6 (binary) bits will be encoded into 8
*                     (alpha) bits, e.g a (B3) value will be encoded
*                     into a (A4) value.
*
```

```
*                    Note: Every binary value will be encoded into
*                    a non-ambiguous alpha value. Re-encoding this
*                    alpha value again will result in the original
*                    binary value. However, this is not the case with
*                    most of the (A) -> (B) -> (A) encodings.
*
*                    The encoding may be used to transfer a .bmp
*                    file via TCP/IP, or to transfer Natural binary or
*                    integer values via the utility protocol.
*
*                    Open Systems only:
*                    On Open Systems, there are 3 modes:
*                    RFC3548, RFC2045 and NATRPC (default).
*                    NATRPC means the encoding follows
*                    the NATRPC logic. This is 100% MF compatible.
*                    RFC2045 is the default of the CMBASE64 call.
*                    RFC3548 is like NATRPC, but alpha bytes not
*                    needed are filled with '='.
*
DEFINE DATA
LOCAL
1 FUNCTION        (A2)
                  /* 'AB' Alpha to binary encoding
                  /* 'BA' Binary to alpha encoding
1 PARM-RC         (I4)
                  /* 0:    ok
                  /* Mainframe
                  /* 1  Source is not numeric
                  /* 2  Source is not packed
                  /* 3  Source is not floating point
                  /* 4  Overflow, source doesn't fit into target
                  /* 5  Integer overflow
                  /* 6  Source is not a valid date or time
                  /* 7  Length error (hex input not even)
                  /* 8  Target precision is less than source precision
                  /* 9  Float underflow (result->0)
                  /* 10 Alpha source contains non-hex characters
                  /* 20 Invalid function code
                  /* Open Systems
                  /* 1  Invalid value for RFC parameter
                  /* 2  Invalid function code
                  /* 3  CMBASE64: Overflow, source doesn't fit into
                  /*              target
                  /* 4  CMBASE64: Non-base64 character found in encoded
                  /*              data
                  /* 5  CMBASE64: Out of memory
                  /* 6  CMBASE64: Inalid number of parameters
                  /* 7  CMBASE64: Invalid parameter type
                  /* 8  CMBASE64: Invalid parameter length
                  /* 9  CMBASE64: Invalid function code
                  /* 10 CMBASE64: Unkown return code
1 PARM-ERRTXT     (A72)
```

```
                  /* blank, if ok no error
                  /* else error text
1 PARM-A          (A)   DYNAMIC
                  /* Alpha source input/target output
1 PARM-B          (B)   DYNAMIC
*                 /* Binary source input/target output
1 PARM-RFC        (B1)
                  /* OS only, not used for MF
                  /* 0 - RFC3548; 3 - RFC2045; 4 - NATRPC;
/*
1 #BACKUP-A       (A) DYNAMIC
1 #BACKUP-B       (B) DYNAMIC
END-DEFINE
/*
/*
SET KEY ALL
/*
/* Copycode SAG64BAP and SAG64ABP containing the prototype definition
/* is used at compilation time only in order to determine the type of
/* the return variable for function call reference and to check the
/* parameters, if this is desired. SAG64BAP and SAG64ABP are optional.
INCLUDE SAG64BAP
INCLUDE SAG64ABP
/*
REPEAT
  RESET PARM-A PARM-B
  REDUCE DYNAMIC PARM-A TO 0
  REDUCE DYNAMIC PARM-B TO 0
  FUNCTION := 'BA'
  PARM-B := H'0123456789ABCDEF'
  INPUT (AD=MIL IP=OFF CD=NE) WITH TEXT PARM-ERRTXT
   // 10T 'Base64 Encoding:' (YEI)
    / 10T '-' (19) (YEI) /
    / 10T 'Function (BA,AB) ..' (TU) FUNCTION (AD=T)
    / 10T 'Alpha In/Output ...' (TU) PARM-A (AL=30)
    / 10T 'Binary In/Output ..' (TU) PARM-B (EM=HHHHHHHH)
    / 10T 'Response ..........' (TU) PARM-RC (AD=OD CD=TU)
    / PARM-ERRTXT (AD=OD CD=TU)
  RESET PARM-ERRTXT
  IF *PF-KEY NE 'ENTR'
    ESCAPE BOTTOM
  END-IF
  /*
  RESET #BACKUP-A #BACKUP-B
  REDUCE DYNAMIC #BACKUP-A TO 0
  REDUCE DYNAMIC #BACKUP-B TO 0
  #BACKUP-A := PARM-A
  #BACKUP-B := PARM-B
  /*
  IF FUNCTION = 'BA'
    /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
    /* Parameter PARM-RFC does not apply to mainframe
```

```
  /* PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)
  PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT>)
  /* PARM-A := SAG64BA(<PARM-B,PARM-RC>)
  /* PARM-A := SAG64BA(<PARM-B>)
ELSE
  /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
  /* Parameter PARM-RFC does not apply to mainframe
  /* PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)
  PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT>)
  /* PARM-B := SAG64AB(<PARM-A,PARM-RC>)
  /* PARM-B := SAG64AB(<PARM-A>)
END-IF
/*
IF PARM-RC NE 0 THEN
  WRITE 'Encoding' FUNCTION
  WRITE NOTITLE PARM-ERRTXT
ELSE
  IF FUNCTION = 'BA' THEN
    WRITE 'Binary -> Alpha'
    WRITE '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
      / '=' PARM-A (AL=50)
    RESET PARM-B
    REDUCE DYNAMIC PARM-B TO 0
    FUNCTION := 'AB'
  ELSE
    WRITE 'Alpha -> Binary'
    WRITE '=' PARM-A (AL=50) /
      '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
    RESET PARM-A
    REDUCE DYNAMIC PARM-A TO 0
    FUNCTION := 'BA'
  END-IF
  /*
  IF FUNCTION = 'BA'
    /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
    /* Parameter PARM-RFC does not apply to mainframe
    /* PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT,PARM-RFC>)
    PARM-A := SAG64BA(<PARM-B,PARM-RC,PARM-ERRTXT>)
    /* PARM-A := SAG64BA(<PARM-B,PARM-RC>)
    /* PARM-A := SAG64BA(<PARM-B>)
  ELSE
    /* Parameter PARM-RC, PARM-ERRTXT and PARM-RFC are optional
    /* Parameter PARM-RFC does not apply to mainframe
    /* PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT,PARM-RFC>)
    PARM-B := SAG64AB(<PARM-A,PARM-RC,PARM-ERRTXT>)
    /* PARM-B := SAG64AB(<PARM-A,PARM-RC>)
    /* PARM-B := SAG64AB(<PARM-A>)
  END-IF
  IF PARM-RC NE 0 THEN
    WRITE 'Encoding' FUNCTION
    WRITE NOTITLE PARM-ERRTXT
  ELSE
```

```
      IF FUNCTION = 'BA' THEN
        WRITE 'Binary -> Alpha'
        WRITE '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
          / '=' PARM-A (AL=50)
        IF PARM-A = #BACKUP-A THEN
          WRITE '******** Encoding successful ********'
        ELSE
          WRITE '******** Value changed by encoding ********'
        END-IF
      ELSE
        WRITE 'Alpha -> Binary'
        WRITE '=' PARM-A (AL=50) /
          '=' PARM-B (EM=HHHHHHHHHHHHHHHHHHHHHHHHHH)
        IF PARM-B = #BACKUP-B THEN
          WRITE '******** Encoding successful ********'
        ELSE
          WRITE '******** Value changed by encoding ********'
        END-IF
      END-IF
    END-IF
  END-IF
END-REPEAT
END
```