Coal

言語仕様書編

第　５．７　版

２０１３年9月１７日

改 訂 履 歴

Ver	改訂年月日	改訂理由	作成者	承認者
5.7	2012.01.27	新規	小林	
	2013.03.11	[f]putline()の出力内容をsayコマンドと同様にした。	小林	
	2013.03.12	10進少数点数変換においてカンマを無視するオプションを追加した。	小林	
	2013.03.13	10進少数点数表示で３桁毎のカンマを付けるオプションを追加した。	小林	
	2013.08.11	DO WHILE,UNTIL機能を追加した	小林	

目　　次

1. はじめに....7
2. 全般規定....8
2.1. 構文の表記法....8
2.2. スクリプトの形式....10
2.2.1. 構造....10
2.2.2. 手続き、関数、クラス....10
2.2.3. 文....11
2.2.4. 語....12
2.2.5. コメント....12
2.2.6. 引用符....13
2.2.7. 改行の扱い....13
2.2.8. エスケープ文字....13
2.2.9. 入力構文の文字コード....13
2.2.10. プリプロセッサ機能....14
2.3. コマンド一覧....15
3. コマンド....17
3.1. ＰＲＯＣ　または　ＳＵＢ....17
3.2. ＥＸＥＣ....18
3.3. ＲＥＴＵＲＮ....19
3.4. ＬＯＯＰ....20
3.5. ＢＲＥＡＫ....23
3.6. ＣＯＮＴＩＮＵＥ....23
3.7. ＩＦ....24
3.8. ＢＥＸＰ　（Ｂｉｎｏｍｉｎａｌ　ＥｘＰｒｅｓｓｉｏｎ）....25
3.9. ＳＱＬ....26
3.10. ＲＥＡＤ....28
3.11. ＯＵＴＰＵＴ....29
3.12. ＯＮ....31
3.13. ＳＬＥＥＰ....33
3.14. ＬＥＡＶＥ....34
3.15. ＬＥＴ....35
3.15.1. 代入演算....35
3.15.2. パラメータ・シフト....36
3.15.3. ループ継続....36
3.15.4. 関数実行....36
3.15.5. オプション設定....37
3.15.6. スクリプト実行終了....37
3.15.7. ＮＵＬＬ行....37
3.15.8. ＬＰＲＩＮＴ....38
3.15.9. ＰＲＩＮＴ....40
3.15.10. ＥＣＨＯ　または　ＳＡＹ....40
3.15.11. ＤＵＭＰ....41
3.15.12. ＩＮＴＥＲＡＣＴＩＶＥ....42
3.15.13. ＬＯＧＰＡＲＭ....44
3.16. ＤＥＦＩＮＥ....45
3.16.1. 変数の修飾子....45
3.16.2. 変数のデータ型指定....46
3.16.3. 変数名の式指定....47
3.16.4. 変数定義....48
3.16.5. マップド配列変数定義....49
3.16.6. 配列変数定義....50
3.16.7. 手続き内定義....51
3.16.8. オプション設定....51

3.16.9. 型定義 52
3.17. ＲＥＤＥＦＩＮＥ 53
3.17.1. マップド配列変数再定義 53
3.17.2. 内部および外部配列変数再定義 53
3.17.3. 内部および外部変数再定義 53
3.18. ＵＮＤＥＦＩＮＥ 54
3.19. ＭＥＳＳＡＧＥ 55
3.19.1. MESSAGE SLEEP 55
3.19.2. MESSAGE SQL 55
3.19.3. MESSAGE SEND 55
3.20. ＳＷＩＴＣＨ 56
3.21. ＦＵＮＣＴＩＯＮ 57
3.22. ＩＭＰＯＲＴ 58
3.23. ＴＲＹ 59
3.24. ＴＨＲＯＷ または ＲＡＩＳＥ 60
3.25. ＤＯ 60
3.26. ＬＡＢＥＬ 61
3.27. ＣＬＡＳＳ 62
4. 式 63
4.1. 一般形式 63
4.2. 論理式 68
4.3. 比較式 68
4.4. 算術式 68
4.5. 文字列式 68
4.6. 範囲式 73
4.7. 式の並び 73
4.7.1. 式の並びについて 73
4.7.2. 式の並びの扱い 73
4.8. ドット式 75
4.8.1. システム関数の実行 75
4.8.2. ユーザ定義関数の実行 75
4.8.3. クラス関数の実行 75
4.8.4. クラス変数へのアクセス 75
4.8.5. 構造体メンバ要素へのアクセス 75
4.8.6. 文字列の切り出し 75
4.9. 集合演算 76
4.10. データ指定単項演算子 77
4.11. インスタンス生成式 77
5. 変数 78
5.1. パラメータ変数 78
5.2. 検索結果読み込み変数 78
5.3. 内部変数 78
5.4. 外部変数 78
5.5. システム変数 79
5.6. 変数の配列 80
5.7. 構造体変数 81
6. 定数 82
6.1. 文字列定数 82
6.2. 数値定数 83
7. データ属性 84
7.1. データ属性一覧 84
7.2. データ・リスト 84
8. 組み込み関数 85
8.1. 関数一覧 85
8.2. 全般規則 90
8.3. ＡＢＳ 90

8.4. ARRAYCPY 90
8.5. ARRAYCLR 91
8.6. ARRAYCMP 91
8.7. ARRAYBXP 92
8.8. ARRAYMAP 92
8.9. ASC 93
8.10. ATAN 93
8.11. ATANH 93
8.12. CBIN 94
8.13. CBRT 94
8.14. CBULK 94
8.15. CCHAR 94
8.16. CDEC 95
8.17. CDOUBLE (CDBL) 95
8.18. CEIL 95
8.19. CFLOAT (CFLT) 95
8.20. CHANNEL 96
8.21. CHR 97
8.22. CINT 98
8.23. CLONG (CLNG) 98
8.24. CLOSEDIR 98
8.25. CONCAT 98
8.26. CONDAS 99
8.27. CONS 99
8.28. COS 99
8.29. COSH 99
8.30. COUNTV 100
8.31. CSTRING (CSTR) 100
8.32. EDIT 100
8.33. ELREAD1 101
8.34. ELWRITE 101
8.35. ELWRITE1 101
8.36. EVAL 101
8.37. EXIT 102
8.38. EXP 102
8.39. FCLOSE 102
8.40. FELREAD1 103
8.41. FELWRITE 104
8.42. FELWRITE1 105
8.43. FF 106
8.44. FGETLINE 106
8.45. FIRST 107
8.46. FL 107
8.47. FLOOR 108
8.48. FOPEN 108
8.49. [FP]STAT 109
8.50. FPUTLINE 110
8.51. GETARGS 111
8.52. GETENV 112
8.53. GETLINE 112
8.54. GETLOGPARM 112
8.55. GETMEMUSED 113
8.56. GETTIME 113
8.57. GETVAL 114
8.58. GETWD 114
8.59. iIN 116

8.60. iLIKE 116
8.61. IN 116
8.62. INDEXA 117
8.63. INiRSTR 117
8.64. INiSTR 117
8.65. INLIKE 118
8.66. INRSTR 118
8.67. INSTR 119
8.68. iREGEX 119
8.69. IS 119
8.70. LEFT 120
8.71. LEFTB 120
8.72. LENG 120
8.73. LENB 121
8.74. LIKE 121
8.75. LIST 121
8.76. LIST_REF 122
8.77. LOG 122
8.78. LOG10 122
8.79. LOGOUT 122
8.80. MAX 123
8.81. MIN 123
8.82. MOD 123
8.83. NDEF 123
8.84. NEW 124
8.85. NVAL 124
8.86. NOFREE 124
8.87. OPENDIR 124
8.88. PCLOSE 125
8.89. POPEN 125
8.90. POWER 125
8.91. PUTENV 126
8.92. PUTLINE 126
8.93. RAND1(DRAND48) 126
8.94. READDIR 127
8.95. REGEX 127
8.96. REP 128
8.97. REPLIKE 128
8.98. RESLOGPARM 130
8.99. REST 130
8.100. RIGHT 131
8.101. RIGHTB 131
8.102. RINT 132
8.103. ROUND 132
8.104. SETARRAY 133
8.105. SETENV 133
8.106. SETLOGPARM 134
8.107. SHELL 134
8.108. SHUTCTL 134
8.109. SIN 135
8.110. SINH 135
8.111. SKIP_OPT 135
8.112. SORT 136
8.113. SQRT 138
8.114. SRAND1(SRAND48) 138
8.115. STRINGS 138

8.116. ＳＵＢＳＴＲ（ＭＩＤ）....139
8.117. ＳＵＢＳＴＲＢ（ＭＩＤＢ）....139
8.118. ＳＹＳＬＯＧ....139
8.119. ＴＡＮ....140
8.120. ＴＡＮＨ....140
8.121. ＴＩＭＥＳ....140
8.122. ＴＯ....140
8.123. ＴＯ＿ＢＵＬＫ....141
8.124. ＴＯ＿ＢＵＬＫＳ....141
8.125. ＴＯ＿ＮＵＭＢＥＲ....142
8.126. ＴＲＩＭ....142
8.127. ＵＮＬＩＮＫ....142
8.128. ＵＮＳＥＴＥＮＶ....143
8.129. ＸＨＡＳＨ....143
9. 付録....145
9.1. ＳＱＬコマンドの処理番号....145
9.2. 実行時オプション....146
9.3. データ属性の値....148
9.4. 演算子の順位....149
9.5. 変数、手続きまたは関数のサーチ順....150
9.6. デバッグ形式での出力....150
9.7. 例外の種類....152
9.7.1. 例外番号の構成....152
9.7.2. 区分....152
9.7.3. 機能....152
9.8. スクリプトの例....154
9.8.1. Hellow World....154
9.8.2. クロージャ....155
9.8.3. 構造体を使った擬似クラス....156

1. はじめに

本書は、Coal（COmmon platform Application Language）の言語仕様を記述したものである。

2. 全般規定

2.1. 構文の表記法

表記法は以降の記述で現れるステートメントの形式を記述するために用いる方法であり、一般形式の明解な表現を与えるものである。

一般形式中の日本語およびその後ろに「－」（ハイフン）を伴って続く数字で記述された部分は表記法上の変数であり、ステートメントを記述する場合には、それに該当する名標、定数等に置換なければならない。

＜例＞

コマンド名△パラメーター１△パラメーター２

上記例でコマンド名は表記上の変数であり、このステートメントを記述する場合には、名標を記述しなければならない。

全角文字で表記されている名票、記号は、全角文字、または、半角文字で記述することができる。

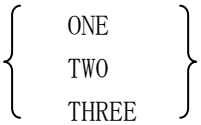
全角文字で記述できる部分は、モードによって変わる。(2.2.8を参照)

次に構文の定義に用いられる記号について説明する。

（1）中カッコ（｛｝）

縦に複数の項目が並んでいる場合、そのうちの１つを選択することを示す。

＜例＞



上記例では、ONE、TWO、THREE のいずれか１つを選択することを示している。

（2）縦棒（|）

この縦棒に区切られた複数項目の内、１つを選択することを示す。

縦棒を項目として含める場合は、″縦棒（|）″と表記する。

＜例＞

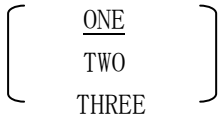
{ ONE | TWO | THREE }

上記例では（1）の例と全く同じ意味を示す。

（3）大カッコ（［］）

この大カッコで囲まれた項目が、省略可能であることを示す。この大カッコ中に縦に複数の項目がある場合には、そのうちの１つを選択するか全てを省略しなければならない。省略した場合、この項目に下線があれば、それが指定されたものとみなされる。

＜例＞



上記例では、ONE、TWO、THREEの内いずれか１つを選択するか、省略するかを示している。この例では、省略するとONEが指定されたものとみなされる。また、この例では、次のような記述と等価である。

[ONE | TWO | THREE]

（4）省略記号（・・・）

項目が複数個連続することを示す。

＜例＞

列１［, 列２　］・・・

上記例では、列１から列ｎまで指定できることを示している。

（5）空白（△）

△は、１文字以上の全角スペースまたは半角スペースまたはタブまたは区切り文字となる改行を示す。改行の扱いについては、「2.2.7 改行の扱い」を参照。

全角文字で記述できる部分は、モードによって変わる。

（6）太字

太字は、少なくても、その文字数までの指定でよいことを示す。

＜例＞

〔 **INT**EGER
FLOAT
DECIMAL 〕

2.2. スクリプトの形式

2.2.1. 構造

スクリプトの構造を図２．２－１に示す。

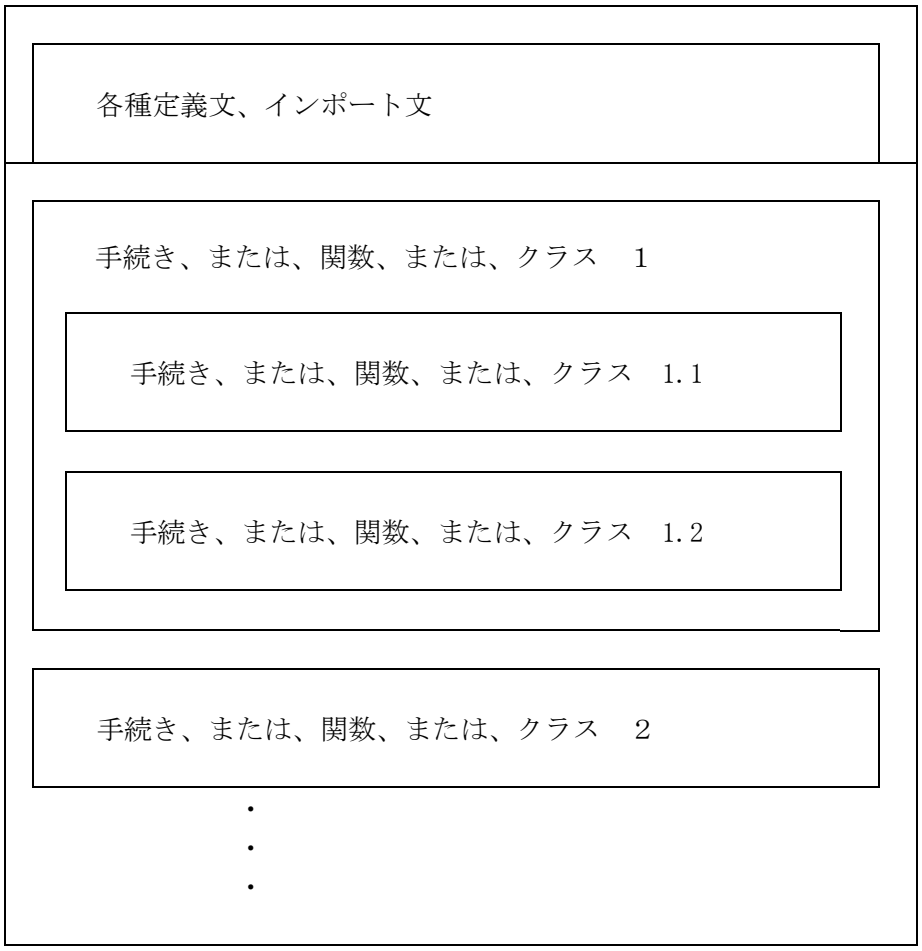


図２．２－１　スクリプトの構造

2.2.2. 手続き、関数、クラス

手続きの形式を図２．２－２に示す。詳細はコマンドを参照。
最初に実行される手続きの名称は、’ｍａｉｎ’である。

```
ＰＲＯＣ △ 手続き名（引数，・・・）；

    文の並び

ＥＮＤＰＲＯＣ ；
```

図２．２－２　手続きの形式

関数の形式を図２．２－３に示す。詳細はコマンドを参照。

```
ＦＵＮＣＴＩＯＮ △ 関数名（引数，・・・）；

    文の並び

ＥＮＤＦＵＮＣ；
```

図２．２－３　関数の形式

クラスの形式を図２．２－４に示す。詳細はコマンドを参照。

```
ＣＬＡＳＳ △ クラス名；

    文の並び

ＥＮＤＣＬＡＳＳ；
```

図２．２－４　クラスの形式

2.2.3. 文

文はコマンドである。
コマンドには、実行用コマンドと制御用コマンドがあり、それぞれ形式１と形式２に対応する。
１つの文は、コマンド名で始まり、セミコロン（；）で終わる。
文の前後および文中には、任意個の空白または改行を入れることができる。
改行の扱いについては、「2.2.7 改行の扱い」を参照。

（1）形式１

```
コマンド名 △ パラメータ・リスト；
```

・パラメータ・リストは空白で区切られたパラメータの並びである。
・パラメータには、改行コードを除く文字列を指定できる。
・パラメータの前後を引用符で囲むことによって、空白等の文字列を明示的にパラメータとして指定することができる。
・２重引用符を２個続けて指定したものをＮＵＬＬパラメータと呼ぶ。途中のパラメータを省略するときは、これを指定する。

（2）形式２

```
コマンド名 △ パラメータ・リスト；

        文の並び

制御ブロック終了コマンド名；
```

2.2.4. 語

語はシステム語とユーザ語に分けることができる。

（1）システム語

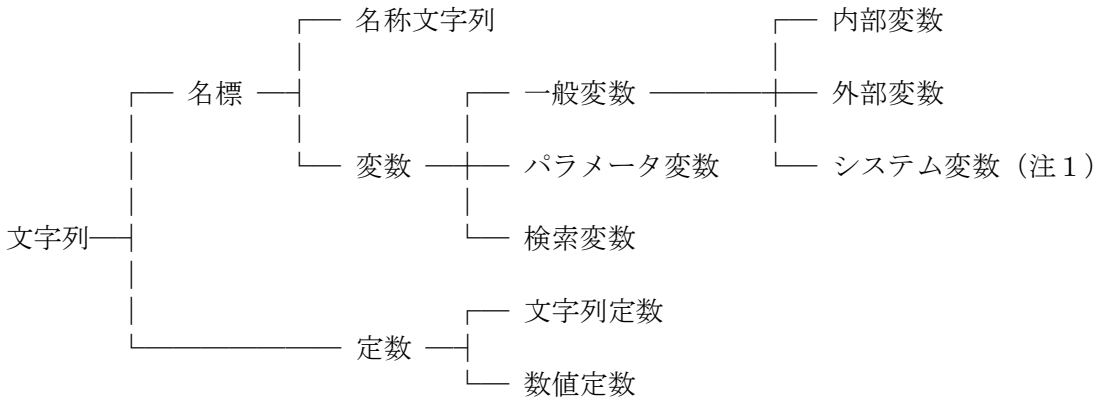
システム語はキーワードであり、コマンド名、パラメータの一部、システム変数名、記号がある。

システム変数名の英字には、大文字を使用する。その他のシステム語は、大文字と小文字を区別しない。

（2）ユーザ語

ユーザ語は文字列であり、パラメータで使われる。ユーザ語は以下のように分けられる。

ユーザ語は、大文字と小文字が区別される。



（注１）システム語に属する

コマンドの一般形式においてパラメータに対して特に指定がない場合には、変数、定数などの区別はなく、単なる文字列として解釈される。名称文字列とは、変数または定数に属さない文字列を指す。各種名称に使用する名称文字列は、全角文字(注２)、英字、下線で始まる全角文字(注２)、英数字、下線で構成される。

（注２）全角文字が半角文字に変換されるモードのときは、記号を除く。

2.2.5. コメント

スクリプトの中には、コメントを入れることができる。

（1）単価記号（@）

単価記号（@）以降、行の終わり（改行コード）までをコメントとする。

パラメータを示す最初の２重引用符以降にあっても、コメントとなる。

（2）スラッシュ記号（／）

連続する２個のスラッシュ記号（／／）以降、行の終わり（改行コード）までをコメントとする。

引用符で囲まれた部分では、コメントにならない。

本コメントを使用した以降では、引用符の扱い、文字数の単位等は新仕様となる。

（3）スラッシュ記号（／）とアスタリスク記号（＊）

″/*″と″*/″で囲まれた部分をコメントとする。ネストも可能。

引用符で囲まれた部分では、コメントにならない。

（注）引用符とは、１重引用符または２重引用符をさす。

2.2.6. 引用符

（1）旧仕様

２重引用符で囲まれた文字列がパラメータとなる。２重引用符は含まれない。
１重引用符は、どこにあっても文字として扱われる。

（2）新仕様

１重引用符または２重引用符の最初に現れた方が引用符記号として使われる。
１重引用符で囲まれた文字列中の連続する２個の１重引用符は、１個の１重引用符文字となる。
２重引用符で囲まれた文字列中に文字として２重引用符を入れるときは、エスケープ文字を使用する。
１重引用符で囲まれた文字列は、前後の１重引用符を含めてパラメータとなる。
２重引用符で囲まれた文字列は、前後の２重引用符を除いてパラメータとなる。

2.2.7. 改行の扱い

スクリプト実行時の改行の扱いは以下となる。

（1）旧仕様

改行は無視され、ないものとして扱われる。

（2）新仕様

（A）１重引用符または２重引用符の外側
改行は、区切り文字として扱われる。

（B）１重引用符（文字列定数のとき）または２重引用符の内側
改行は、改行コードとして扱われる。

2.2.8. エスケープ文字

円記号（¥）をエスケープ文字とする。
エスケープ文字は、機能を持つ文字（エスケープ文字、引用符、単価記号等）の前に付けて、その文字の機能を無効にする。
ただし、旧仕様と新仕様では、以下となる。

（1）旧仕様

エスケープ文字は、スクリプトを読み込んだ時点で除かれる。したがって、文字列定数中にエスケープ文字を入れるときは、２個続けて指定する。
ただし、改行の直前にある場合は、エスケープ文字は無視される。

（2）新仕様

エスケープ文字の後ろが改行の場合は、改行と共には除かれ、ないものとして扱われる。
エスケープ文字の後ろが改行でない場合は、以下となる。

（A）１重引用符外
エスケープ文字として機能し、スクリプトを読み込んだ時点で除かれる。

（B）１重引用符内（文字列定数のとき）
エスケープ文字と見なされず、実行時にエスケープ処理される。

2.2.9. 入力構文の文字コード

以下のモードがある。モードは実行時オプションで決まる。

No.	モード	文字コード
1	デフォルトモード	以下の部分では、英数記号は、全角文字で記述できる。（半角文字に変換される） ・引用符の外側 ・引用符で囲まれた部分の最初の引用符 ・コメントの始まり(@,／＊,／／)。
2	全変換	全てを全角文字で記述できる。（全ての全角英数記号は、半角文字に変換される）
3	無変換	システム語および記号は、半角で記述しなければならない。

2.2.10. プリプロセッサ機能

文字列の置換、ソースの入力条件を指示する文をプリプロセッサ文と呼ぶ。
プリプロセッサ文は、シャープ記号（＃）＋特殊記号（＋－．＞＠＊＝のどれか１つ）で始まる。
プリプロセッサ文を次行に継続するときは、行末に円記号（￥）を指定する。
キーは、空白文字で区切り、空白文字、引用符(’または″)、括弧(())以外の文字を使用できる。
以下の種類がある。
if、ifdef、ifndefは入れ子にできる。

No.	文	意味
1	define キー［定義値］	・キーを定義し、値を設定する。ソース中の引用符(’または″)で囲まれた部分を除いて、キーに一致する文字列を定義値に置換する。 ・定義値の中に、当該ソース行より前に定義されたキーがあれば、それも置換される。 ・置換によって、キー値がループした場合は、エラーとなる。 ・定義値は、改行で終了する。次行に継続するときは、改行の直前に円記号（￥）を指定する。 ・キーが重複したときは、エラーとなる。
2	define キー(a,b,...)［定義値］	・上記同様に、″キー(a,b,...)″の形式に一致する文字列を定義値に置換する。その際に、″(a,b,...)″のa,b,...(仮引数)に一致する定義値の中のa,b,...は、ソース中で指定された()内の値(実引数)に置換される。ただし、定義値の中で、１重引用符(’)で囲まれた部分は置換されない。定義値の中の２重引用符(″)は、単なる文字として扱われる。 ・仮引数には、空白文字、引用符(’または″)、括弧(())、カンマ(,)以外の文字を使用できる。 ・実引数には、引用符で囲まれた任意の文字列と、引用符で囲まれない空白文字、引用符、括弧、カンマ以外の文字列を使用できる。
3	undef キー	キー定義を取り消す。
4	if 式	・式の値が0以外または真のときに、以降のソースを読み込む。 ・式には、符号あり／なし数字列、または、″defined(キー)″、″||″、″&&″、の組み合わせによる論理式を指定できる。論理式は、左から右に評価される。″||″と″&&″には、順位はない。
5	elif 式	直前までのif式またはelif式が全て0または偽の場合、 式の値が0以外または真のときに、以降のソースを読み込む。
6	else	直前までのif式等が全て0または偽の場合、以降のソースを読み込む。
7	endif	if式等の終了。
8	ifdef キー	キーが定義されているときに、以降のソースを読み込む。
9	ifndef キー	キーが定義されていないときに、以降のソースを読み込む。
10	elifdef キー	直前までのif式等が全て0または偽の場合、 キーが定義されているときに、以降のソースを読み込む。
11	elifndef キー	直前までのif式等が全て0または偽の場合、 キーが定義されていないときに、以降のソースを読み込む。
12	include ファイル名	指定された位置にソースを読み込む。ファイル名は、１重引用符または２重引用符で囲っても良い。ファイル名の指定方法は、ＥＸＥＣコマンドのスクリプトと同様。

#*define/#*undefの定義では、キーの比較時に大文字小文字を区別しない。

2.3. コマンド一覧

コマンドの一覧を表２－１に示す。

表２－１　コマンド一覧（1/2）

項番	コマンド名	機　　能
１	ＰＲＯＣ　または ＳＵＢ	手続きの始まりを示す
２	ＥＮＤＰＲＯＣ または ＥＮＤＳＵＢ	手続きの終わりを示す
３	ＲＥＴＵＲＮ	手続きまたは関数の呼び出し元に復帰する
４	ＥＸＥＣ	手続きを呼び出す
５	ＬＯＯＰ	繰り返しブロックの開始を示す
６	ＢＲＥＡＫ	繰り返しブロックから抜ける
７	ＣＯＮＴＩＮＵＥ	繰り返しブロックを継続する。
８	ＥＮＤＬＯＯＰ	繰り返しブロックの終わりを示す
９	ＩＦ	条件分岐ブロックの開始および条件が真のときに実行するブロックの開始を示す
１０	ＥＬＳＥ	条件が偽のときに実行するブロックの開始を示す
１１	ＥＬＳＥＩＦ 又は ＥＬＳＩＦ	条件が偽のときの条件分岐ブロックの開始を示す
１２	ＥＮＤＩＦ	条件分岐ブロックの終わりを示す
１３	ＢＥＸＰ	拡張２項演算結果を変数に代入する
１４	ＳＱＬ	ＳＱＬ文をＤＢ管理に発行し、結果を受け取る
１５	ＲＥＡＤ	検索結果からタプル単位でデータを読み込む
１６	ＯＵＴＰＵＴ	表示用データをパッケットに出力する
１７	ＯＮ　　**（定義）**	ＯＮ条件を定義する
１８	ＳＬＥＥＰ	処理を休止する。
１９	ＬＥＡＶＥ	コマンド転送
２０	ＬＥＴ 又は ＳＥＴ	式の演算結果を変数に代入する。その他、サブコマンドを実行する。
２１	ＤＥＦＩＮＥ　**（定義）**	変数、配列、構造体を定義する
２２	ＲＥＤＥＦＩＮＥ	配列を再定義する
２３	ＵＮＤＥＦＩＮＥ	内部変数または外部変数を未定義にする。
２４	ＭＥＳＳＡＧＥ	メッセージを送信し、結果を受信する。
２５	ＳＷＩＴＣＨ	ＳＷＩＴＣＨ条件ブロックの開始を示す
２６	ＣＡＳＥ	条件が真のときに実行するブロックの開始を示す
２７	ＤＥＦＡＵＬＴ	どのＣＡＳＥ文でも真にならなかったとき実行するブロックの開始を示す
２８	ＥＮＤＳＷ	ＳＷＩＴＣＨ条件ブロックの終わりを示す
２９	ＦＵＮＣ　又は ＦＵＮＣＴＩＯＮ	関数の始まりを示す
３０	ＥＮＤＦＵＮＣ	関数の終わりを示す

（定義）定義文部分でのみ使用可能。

表２－１　コマンド一覧（2/2）

項番	コマンド名	機　　能
３１	ＩＭＰＯＲＴ　**（定義）**	スクリプトをインポートする
３２	ＴＲＹ	例外処理ブロックの開始
３３	ＣＡＴＣＨ	例外の捕捉
３４	ＦＩＮＡＬＬＹ	例外後処理の開始
３５	ＥＮＤＴＲＹ	例外処理ブロックの終了
３６	ＴＨＲＯＷ　または ＲＡＩＳＥ	例外を起こす
３７	ＤＯ	ブロックまたは繰り返しブロックの開始を示す
３８	ＥＮＤＤＯ	ブロックまたは繰り返しブロックの終わりを示す
３９	ＬＡＢＥＬ	ラベル
４０	ＣＬＡＳＳ	クラスの始まりを示す
４１	ＥＮＤＣＬＡＳＳ	クラスの終わりを示す

3. コマンド

3.1. ＰＲＯＣ　または　ＳＵＢ

（１）機能

手続きの開始を宣言する 。

（２）一般形式

```
{ＰＲＯＣ}
{ＳＵＢ }  △ 手続き名［△［変数名－１］{,|△}［変数名－２］］
              ［［△］（［△］［データ型］△［仮引数名］
                                  ［ △ ＡＳ △ データ型 ］［△］，・・・）］；

                文の並び

ＥＮＤ[△] {ＰＲＯＣ};
          {ＳＵＢ }
```

（３）構文規則

（Ａ）手続き名は、空白文字を除く３２バイト以内の文字列とする。
（Ｂ）変数名－１、変数名－２、仮引数名、・・・には、"$"または"%"を付けない。
（Ｃ）変数名を指定しない等で不要なカンマは、省略できる。
（Ｄ）仮引数にデータ型を指定する場合は、仮引数の前か後ろ(ＡＳ指定)の一方にのみ指定可能。

（４）一般規則

（Ａ）手続きは入れ子にすることができる。（オプション指定）
（Ｂ）手続きから呼出元へ復帰するためには、ＲＥＴＵＲＮコマンドを使用するが、
ＲＥＴＵＲＮコマンドがなくＥＮＤＰＲＯＣに達したときは、リターン値は前のままで
リターンする。
（Ｃ）変数名－１は、本手続き内の変数となり、手続き実行コマンドで指定したパラメータ(実引数)の数が入る。$変数名－１で参照する。
（Ｄ）変数名－２は、１番目の実引数からマップされた本手続き内のMAPPEDARRAY変数となる。
%変数名－２[インデックス番号]で参照する。
%変数名－２[0]が、１番目のパラメータに対応する。
手続き実行コマンドでのＮＵＬＬパラメータに対応する要素は、ＮＵＬＬパラメータ属性を持つＮＵＬＬ値（これもＮＵＬＬパラメータと呼ぶ）となる。
実引数の個数を超えたインデックス番号の指定は、範囲外のエラーとなる。ただし、"main"では、エラーとならず、ＮＵＬＬパラメータとなる。
ＮＵＬＬパラメータは、NDEF()関数でチェックできる。
（Ｅ）括弧内は、仮引数を示し、実引数に順に対応する。$仮引数名で参照する。
省略された実引数に対応する仮引数は、ＮＵＬＬパラメータとなる。
仮引数は、実引数の個数以上の個数は指定できない。ただし、"main"では、実引数の個数を超えて、仮引数を指定できる。このときの仮引数は、ＮＵＬＬパラメータとなる。
（Ｆ）ｍａｉｎ手続きには、スクリプトへのパラメータが渡される。
（Ｇ）手続きの実行中にエラーがあった場合は、システム変数$ERRORにエラーコードが設定され、手続きからは正常にリターンする。
（Ｈ）仮引数にデータ型が指定されているときは、実引数の値を指定のデータ型に変換する。
指定のデータ型がＣＨＡＲまたはＢＵＬＫでデータ長の指定がないときは、実引数の値が指定のデータ型に変換され、変換されたデータ長のまま仮引数の値となる。

3.2. ＥＸＥＣ

（１）機能

手続きを呼び出す。

（２）一般形式

ＥＸＥＣ △ 手続き種別 △ 手続き名〔 △ パラメータ・リスト 〕；

手続き種別は

ＩＰ	ＩＰ：本スクリプト内の手続き
ＥＰ	ＥＰ：システムで用意されている外部手続き
ＳＣ	ＳＣ：スクリプト
ＳＭ	ＳＭ：メモリー内のスクリプト

手続き名は

式
名称文字列

パラメータリストには、式を指定できる。

（３）構文規則

（Ａ）手続き名は、空白文字を除く３２バイト以内の文字列とする。

手続き種別がスクリプトのときは、手続き名にスクリプト・ファイル名を指定する。

ファイルの拡張子、サーチパス等は、「Ｃｏａｌ操作マニュアル」を参照。

ファイルにディレクトリ指定が含まれるときは、ディレクトリ指定部分は手続き名長の制限は適用されない。また、″内部手続き名：スクリプト名″の形式でスクリプト内部の手続き名を指定することができる。

（４）一般規則

（Ａ）ＯＮ条件で手続きまたはスクリプトが呼び出されたときは、パラメータ変数には所定のデータが設定される。

（Ｂ）パラメータでは、値を呼出手続きに引き渡す事はできるが、返却させることはできない。

ただし、配列名または構造体名を指定することによって、その要素の値を返すことができる。

呼出手続き内で、配列名または構造体名に対応するパラメータを変数に代入することによっても、その変数を呼び出し元の配列または構造体として使用できる。

（Ｃ）手続きの実行に入る前にエラーがあった場合は、本コマンドを含む手続きからエラーリターンする。

手続きの実行に入った時点で、システム変数$ERRORは、ゼロになる。

（Ｄ）手続きの実行中にエラーがあった場合は、システム変数$ERRORにエラーコードが設定され、手続きからは正常にリターンする。

（Ｅ）パラメータの″%*″は、スクリプトパラメータの全てが指定されたことになる。

（Ｆ）手続き種別がスクリプトで、″内部手続き名：スクリプト名″を指定したときは、スクリプト内部の手続き名を直接実行する。

（Ｇ）手続き種別がメモリー内スクリプトのときは、手続き名にスクリプトを格納した変数か文字列定数を指定する。

（Ｈ）手続きまたは関数の入れ子をサポートするオプションを指定したときは、ピリオドで手続きまたは関数をつなげて、実行する手続きへのパスを指定できる。

3.3. ＲＥＴＵＲＮ

（１）機能

呼び出した元へ復帰する。

（２）一般形式

```
ＲＥＴＵＲＮ 〔 △ 式 〕 ；
```

（３）構文規則

なし。

（４）一般規則

（Ａ）ＰＲＯＣの場合

（ａ）式は、リターン値を示し、この値はシステム変数＄ＥＲＲＯＲに設定される。
リターン値がないときは、リターン値は０となる。

（ｂ）リターン値は、数値型または文字型で指定することができる。文字型のときは、整数値に変換される。

（Ｂ）ＦＵＮＣＴＩＯＮの場合

（ａ）式は、リターン値を示し、この値は関数の戻り値に設定される。
リターン値がないときは、関数の戻り値は、データ未設定となる。
~~戻り値は、当該関数を呼び出した手続きまたは関数内の関数名と同じ名前のローカル変数にも設定される。~~
ＦＵＮＣＴＩＯＮ定義に返却値のデータ型指定(ＡＳ)があるときは、そのデータ型に、自動変換される。詳細はＦＵＮＣＴＩＯＮを参照。
このとき、リターン値がない場合は、以下となる。
CHAR: NULL値
INT,DOUBLE,DEC: それぞれのデータ型のゼロ値
BULK: 長さゼロ

（Ｃ）ＴＲＹ、ＣＡＴＣＨ、ＦＩＮＡＬＬＹブロック内のときは、終了する手続き内にある全てのＦＩＮＡＬＬＹブロックを内側から順に実行する。
ただし、ＦＩＮＡＬＬＹブロック内のとき、自ＦＩＮＡＬＬＹブロックは除く。
実行したＦＩＮＡＬＬＹブロック内で例外が発生したときは、ＲＥＴＵＲＮ処理を中断し、例外処理が実行される。

3.4. ＬＯＯＰ

（１）機能

繰り返しを制御する。

（２）一般形式

```
ＬＯＯＰ［△ 繰り返し数 ］ { ； | △ ＤＯ }
        文の並び
ＥＮＤ[△]{ ＬＯＯＰ | ＤＯ } ；
```

または

```
{ [ＬＯＯＰ △] { { ＷＨＩＬＥ | ＵＮＴＩＬ } △ 条件式
                 ＦＯＲ { [ △ 初期設定式 ][ △ 条件式 ][ △ 増分式 ]
                        ( [ 初期設定式 ]; [ 条件式 ]; [ 増分式 ] )
                          △ 初期設定式 △ ＴＯ △ 終端値式
                                        [ △ ＳＴＥＰ △ 増分値式 ] } } { ； | ＤＯ }
  { ＬＯＯＰ △ | ＦＯＲ △ } ＥＡＣＨ △ 要素用変数 △ ＩＮ △ 対象データ
        文の並び
{ ＥＮＤ[△] { ＬＯＯＰ | ＷＨＩＬＥ | ＵＮＴＩＬ | ＦＯＲ | ＤＯ } ；
  ＮＥＸＴ }
```

または

```
{ ＤＯ ；
  ＬＯＯＰ { ； | ＤＯ } }
        文の並び
ＥＮＤ[△]{ ＷＨＩＬＥ | ＵＮＴＩＬ } △ 条件式 ；
```

繰り返し数、条件式、終端値式、増分値式は

式

初期設定式および増分式は

代入式 [, 代入式] . . .

要素用変数は

内部スカラー変数名

対象データは

［（］［△］式－１［［△］,［△］式－２］・・・［）］

（３）構文規則

（Ａ）ＬＯＯＰは入れ子にすることができる。

（Ｂ）ＬＯＯＰ　ＦＯＲの(;;)形式でない場合は、初期設定式、条件式および増分式は、それぞれ１つのパラメータとして記述しなければならない。

（Ｃ）開始と終了は以下の組み合わせが可能。

また、ループ開始行が、DOで終わるときは、ENDDOも指定可能。

1）すべての開始 と ENDLOOP

2）[LOOP]WHILE と ENDWHILE

3）[LOOP]UNTIL と ENDUNTIL

4）[LOOP]FOR ループ条件 または FOR EACH と NEXT または ENDFOR

（Ｄ）対象データにおけるかっこは、複数の式の並びを囲む形で任意に指定することができる。

（４）一般規則

（Ａ）繰り返し数は、ＬＯＯＰ処理開始時に、ＭＡＸ値として保存され、繰り返しのチェックでは、これを使用する。

したがって、ＬＯＯＰの中で、繰り返し数に指定された変数の値が変わっても、ＬＯＯＰ回数は変わらない。

（Ｂ）繰り返し数は、式の値が整数値型に変換される。NULL値のときは、ゼロと見なされる。

繰り返し数を省略したときは、外部変数の$MAX_LOOP_WHILE回ループする。

（Ｃ）ＷＨＩＬＥまたはＦＯＲの場合は、条件式の値が真かまたは、ループ回数が外部変数の$MAX_LOOP_WHILE以下の間ループする。以下の条件のとき、真となる。

・ 条件式の値が数値型の場合　ゼロでないとき。

・ 条件式の値が数値型以外の場合　ヌルでないとき。

（Ｄ）ＥＡＣＨの場合は、対象データの式の値には、スカラー変数、配列、連想配列、リスト、式の並び、範囲指定が指定できる。

対象データに式の並びを指定したときは、各式の値が要素用変数の値となる。ループ数は、式の並び数となる。

その他のループ数は、表3.4-1の設定値か外部変数$MAX_LOOP_WHILEの小さい方となる。

要素用変数は、ループ開始時に、以下のLOCAL 構造体変数として、初期化され、ループ終了時に削除される。メンバー名は大文字小文字を問わない。

要素用変数名.Index：対象データのインデックスまたは順序番号が入る。

要素用変数名.Value：対象データの値が入る。

表 3.4-1 対象データ別の設定値

設定値	スカラー変数	配列	連想配列	リスト
要素用変数.Value	スカラー変数	1)データ型指定なし配列 設定済みの要素 2)データ型指定配列 定義要素の全て 3)MAPPED配列($) 設定済みの要素 4)MAPPED配列(%) 有効な要素（注１） 5)MAPPED配列(#) 有効な要素（注１）	設定済みの要素	リスト要素
ループ数	１	上記要素の数	上記要素の数	リスト数
要素用変数.Index	０	配列を一元としたときのインデックス（０～）	連想配列のインデックス	リストの順序番号（先頭が１）

（注１）指定されたパラメータ数または検索されたカラム数とMAPPED配列の定義範囲が重なった部分が有効な要素となる。

（E）ＦＯＲ ＴＯ ＳＴＥＰ の場合は、増分指定の有無、増分値式の最初の符号によって、ループ終了判定条件が変わる。初期設定式を Ｘ＝ｎ としたとき、
・増分指定なし、または、符号なし、または、'+'のとき、Ｘ <= (終端値)。
・'-'のとき、Ｘ >= (終端値)。

（F）ＦＯＲの場合、条件式が省略されたときは、真と見なされる。

（G）ＵＮＴＩＬの場合は、条件式の値の判定がＷＨＩＬＥとは、逆転する。

（H）DO(LOOP)...END WHILE(UNTIL)条件式 は、先にDO(LOOP)ブロックの中を実行し、その後で、条件式の結果で判定することを除いて、WHILE(UNTIL)条件式...END WHILE(UNTIL) と同じ。

3.5. ＢＲＥＡＫ

（１）機能
　　ＬＯＯＰまたはＤＯブロックを終了する。

（２）一般形式

ＢＲＥＡＫ　〔△式〕　；

（３）構文規則
　　なし。
（４）一般規則
（Ａ）ＬＯＯＰまたはＤＯ内部以外では、使用できない。
（Ｂ）式の値には、実行を終了するＬＯＯＰまたはＤＯのレベルを指定する。

No.	レベル	動作
１	指定なし または、１	本コマンドが指定されたＬＯＯＰまたはＤＯを終了する。
２	２以上	本コマンドが指定されたＬＯＯＰまたはＤＯを１つ目と数えて、外側のレベル個目のＬＯＯＰまたはＤＯを終了する。
３	０	何もしない

（Ｃ）ＴＲＹ、ＣＡＴＣＨ、ＦＩＮＡＬＬＹブロック内のときは、終了するＬＯＯＰまたはＤＯ内にある全てのＦＩＮＡＬＬＹブロックを内側から順に実行する。
　ただし、ＦＩＮＡＬＬＹブロック内のとき、自ＦＩＮＡＬＬＹブロックは除く。
　実行したＦＩＮＡＬＬＹブロック内で例外が発生したときは、ＢＲＥＡＫ処理を中断し、例外処理が実行される。

3.6. ＣＯＮＴＩＮＵＥ

（１）機能
　　ＬＯＯＰの先頭に戻る。または、ＤＯを終了する。

（２）一般形式

ＣＯＮＴＩＮＵＥ　〔△式〕；

（３）構文規則
　　なし。
（４）一般規則
（Ａ）ＬＯＯＰまたはＤＯ内部以外では、使用できない。
（Ｂ）式の値には、先頭に戻るＬＯＯＰまたは終了するＤＯのレベルを指定する。
　　ＢＲＥＡＫと同様に、指定されたレベルのＬＯＯＰの先頭に戻る。または、ＤＯを終了する。
（Ｃ）ＴＲＹ、ＣＡＴＣＨ、ＦＩＮＡＬＬＹブロック内のときは、ＢＲＥＡＫと同様になる。

3.7. ＩＦ

（1） 機能

処理の選択を制御する。

（2） 一般形式

（A）複数文形式

```
    ＩＦ △ 式 {;| △ ＴＨＥＮ }
[     文の並び     ]
[ ＥＬＳ[Ｅ]ＩＦ △ 式 {;| △ ＴＨＥＮ }]
[     文の並び     ]
[ ＥＬＳＥ [;| △ ] ]
[     文の並び     ]
    ＥＮＤ[△]ＩＦ ;
```

（B）単文形式

```
    ＩＦ △ 式 △ THENL △ １つの文
```

または

```
    ＩＦ △ 式 {;| △ ＴＨＥＮ }
[     文の並び     ]
    ELSEL △ １つの文
```

（3） 構文規則

（A）ＥＬＳＥＩＦ文は、複数置くことができる。
（B）ＩＦ文は、入れ子にすることができる。
（C）ＴＨＥＮ、ＥＬＳＥの直後には、区切り文字が必要。
（D）単文形式の１つの文には、複数文形式または単文形式のＩＦ文を使用できる。

（4） 一般規則

（A）式の値が０またはＮＵＬＬ値なら偽、それ以外なら真とする。
（B）式の値が真のときは、そのＩＦ文またはＥＬＳＥＩＦ文に続く文の並び、または、１つの文が実行される。

3.8. ＢＥＸＰ　（Ｂｉｎｏｍｉｎａｌ　ＥｘＰｒｅｓｓｉｏｎ）

（１）機能

拡張２項演算を行い、結果を変数に代入する。

（２）一般形式

```
ＢＥＸＰ△ 変数式 △ 代入演算子 △ 第一項 ［ △ 二項演算子 △ 第二項
                                          ［ △ 第三項 ．．．］］；
```

変数式は

```
式
```

二項演算子は

```
｛ 文字列演算子
  算術演算子
  比較演算子
  論理演算子 ｝
```

各項は

```
式
```

（３）構文規則

（Ａ）変数式と各項は、１つのパラメータとしなければならない。

（Ｂ）二項演算子が、文字列演算子のＣＯＮＣＡＴとＳＵＢＳＴＲ以外のときは、第三項以降は無視される。

（４）一般規則

（Ａ）項の指定が、第二項以下のときは、ＬＥＴの代入演算と同じ。

（Ｂ）ＣＯＮＣＡＴのときは、第三項以降も順次連結する。

（Ｃ）ＳＵＢＳＴＲのときは、第三項は、切出し文字数となる。このとき、第二項中に指定された切出し文字数は、無視される。第四項以降は無視される。

（Ｄ）第三項以降が指定されたときは、変数および各項は、それぞれ式単位で評価された後で、二項演算と代入演算が行われる。

（Ｅ）第一項および第二項において、式が引用符から始まる場合は、文字定数が指定されたと見なし、旧仕様での扱いとなる。これは、スクリプトが旧仕様であり、以下の例のための仕様である

【例】

```
BEXP $1 = %1 CONDAS ’//null/’%q’’;
```

（Ｆ）変数式の式は、式の評価結果が変数でなければならない。

（Ｇ）ＩＮまたはｉＩＮのときは、第三項以降も比較する。

3.9. ＳＱＬ

（1） 機能

ＳＱＬ文をＤＢ管理に発行する。

（2） 一般形式

ＳＱＬ △ ホストＩＤ △ 項目のデータ属性 △

{ ”ＳＱＬ文” / 処理番号 } 〔 △ データ格納エリアＩＤ〔 △ タイムアウト時間 〕〕 ；

ホストＩＤとデータ格納エリアＩＤは

{ 名称文字 / 式 }

項目のデータ属性は、データ属性指定文字の並びである。

{ 名称文字 / 式 }

内容は、
ＣまたはＣ：文字属性
ＤまたはＤ：数値属性
ＸまたはＸ：ＢＵＬＫ属性
△：文字属性

タイムアウト時間(秒)は

式

（3）構文規則
データ格納エリアＩＤを省略し、タイムアウト時間を指定するときは、データ格納エリアＩＤをNULLパラメータをすること。

（4）一般規則
（A）データ格納エリアＩＤは、英字で始まる英数字または下線からなる。
データ格納エリアの指定が省略されたとき、または、ＮＵＬＬパラメータのときは、システムで用意した一意のエリアとなる。

（B）処理番号が指定されたときは、ＤＢ管理内にあらかじめ設定されている処理番号に対応するＳＱＬ文が実行される。（付録参照）

（C）項目のデータ属性は、ＳＱＬ文が、’ｓｅｌｅｃｔ’のときのみ有効とする。
本指定は、検索結果を本来の内部形式で受け取るために使用される。
データ属性指定文字の並びは、ｓｅｌｅｃｔ項目の順番で対応する。
（a）指定文字が少ないときは、’ｃ’が指定されたものとする。

（ｂ）指定文字が多いときは、多い部分は無視される。

（D）ホストＩＤがＮＵＬＬパラメータかＮＵＬＬ値のときは、自ホストが指定されたものとする。

（E）データ格納エリアＩＤは、スクリプト実行中は一意に識別されるため、呼び出す手続きの中で、有効範囲が重なる場合は重複しない様にしなければならない。

（F）ＳＱＬ文中には、変数を指定することができる。
変数のデータ属性が数値の時は、文字列に変換する。

（G）変数を変換後のＳＱＬ文の先頭が数字のときは、処理番号となる。
処理番号には、０～１００を指定できる。

（H）実行の成否は、＄ＥＲＲＯＲに設定される。
①＄ＥＲＲＯＲ＝０のとき・・・正常終了
②＄ＥＲＲＯＲ≠０のとき・・・エラー終了。ＤＢアクセス時のエラーの時は、エラーの原因がエラーコードに反映される（付録参照）

（Ｉ）ｓｅｌｅｃｔ文が正常終了したときは、＄ＴＵＰＬＥと＄ＣＯＬＵＭＮに検索件数、カラム数が設定される。エラー終了の時は、共にゼロが設定される。

（Ｊ）ＳＱＬ文中では（｀）（バックコーテーション）はエスケープ文字として扱われる。従って、バックコーテーションを文字として使用するときはその前にバックコーテーションを付加しなければならない。
また、バックコーテーションを前に付加することによって、％、＄、＃を文字として使用することができる。

（K）ＤＢアクセス中以外でエラーが起きたときは、本コマンドを含む手続きからエラーリターンする。

（L）タイムアウト時間は、秒で指定する。ミリ秒(少数点以下３桁)未満は切り捨てられる。
タイムアウト時間省略時は、送信メッセージのタイムアウト値(MSG_TIMEOUT)となる。

3.10. ＲＥＡＤ

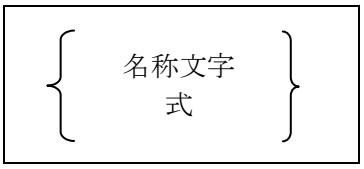
（1） 機能

ＳＱＬコマンドで検索した検索結果からタプル単位でデータを読み込む。

（2） 一般形式

ＲＥＡＤ 〔 △ データ格納エリアＩＤ〔 △ タプル番号 〕〕 ；

データ格納エリアＩＤは



タプル番号は

（3） 構文規則

（A） データ格納エリアを省略し、タプル番号を指定するときは、データ格納エリアは、ＮＵＬＬパラメータとすること。

（4） 一般規則

（A） データは、検索結果読み込み用変数に、ｓｅｌｅｃｔ項目で指定された順に設定される。また、項目のデータ属性で指定された属性で設定される。

（B） タプル番号が指定されたときは、該当タプルのデータを読み込む。

（C） タプル番号が省略されたときは、１つ前にＲＥＡＤされた次のタプルのデータを読み込む。

（D）タプル番号にゼロが指定されたときは、読み込みタプル位置を先頭に設定する。検索結果にｓｅｌｅｃｔ項目リストが添付されているときは、それが検索変数に設定される。添付されていないときは、第一タプルのデータが検索変数に設定される。

（E） 本コマンドを実行すると、＄ＴＵＰＬＥと＄ＣＯＬＵＭＮには、指定されたデータ格納エリアＩＤで検索されたときの値が設定される。

（F） タプル番号が文字型のときは数値に変換される。

（G） データ格納エリアＩＤはＳＱＬコマンドと同様に指定する。

（H） ＳＱＬコマンドで指定したデータ格納エリアＩＤを本コマンドで指定することにより、ＳＱＬコマンドで検索した結果を読み込むことができる。ＳＱＬコマンドでデータ検索エリアＩＤを省略したときは、本コマンドでも省略できる。

3.11. ＯＵＴＰＵＴ

（1） 機能

表示用データをパッケットまたはPRINT_LOGに出力する 。

（2） 一般形式

ＯＵＴＰＵＴ △ { フォーマット用データの指定 / グループ・ノード・フィールド用データの指定 / データ・ノード・フィールド用データの指定 } ;

フォーマット用データの指定は

ＦＭ △ フォーマット・データ

グループ・ノード・フィールド用データの指定は

ＧＲ △ グループ・ノードのフィールド番号 △ タプル番号 △

データ・フィールドの数 △ データ・フィールド番号リスト

各番号と数は

式

データ・ノード・フィールド用データの指定は

ＤＴ △ タプル番号 △ データの個数 △ データ・リスト

または

{ ＴＸ / ＴＮ / ＴＦ / ＡＣ / ＰＲ } △ データ・リスト

タプル番号、データの個数およびデータ・リストの各データは

式

（3） 構文規則

なし。

（４） 一般規則

（Ａ）グループ・ノードのフィールド番号、タプル番号、データ・フィールドの数および
データの個数には、０～９９９９９を指定可能。

（Ｂ）OUTPUT DT の前には、通常、OUTPUT GR が必要であるが、タプル番号に、９００００
以上を指定したときは不要となる。
現在、以下が、使用できる。

項番	タプル番号	用　　途
1	９９９９９	フィールド以外のデータを出力するときに使用する
2	９８８８８	PRINT_LOG(注)にデータ・リスト分のデータを以下の形式で出力する（データの個数は、無視される）。 [coal] Print: 式の値,式の値,・・・ 詳細は、ＬＥＴ　ＰＲＩＮＴを参照。 式がエラーのときは、"##ERROR##"を出力し、処理を継続する。

(注)ログ番号PRINT_LOGで示される媒体に出力する。デフォルトでは、標準出力が
設定されている。詳細は、Ｃｏａｌ実行手順書を参照

（Ｃ）グループフィールドのフィールド番号に９００００以上指定したときは、ＧＲのデータは
出力されない。

（Ｄ）OUTPUT DTにおいて、データの個数とデータリストのデータ数が合わないときは、データの
個数をデータリストのデータ数にする。

（Ｅ）ＴＸまたはＴＮのときは、データリストに指定された式の値を文字列に変換して出力する。
ＴＮのときは、データリスト出力後に改行コード('¥n')を出力する。

（Ｆ）ＴＦのときは、データリストに指定された式の値をファイル名とするファイルの内容を出力
する。ファイル内容が空でないときは、ファイル内容出力後改行コード('¥n')を出力する。

（Ｇ）ＡＣのときは、スクリプト実行開始後にパケットに出力した全てをクリアする。

（Ｈ）ＰＲのときは、OUTPUT DT 98888 0 と同じ。

3.12. ＯＮ

（１）機能

ＯＮ条件を定義する。

（２）一般形式

```
ＯＮ △ { ＧＲ △ グループ・ノードのフィールド番号
       { < { 〔キーワード〕 } >
       {   { ?            }
       {   { !            }

                △ 手続き種別－１ △ 手続き名－１
                △ 手続き種別－２ △ 手続き名－２
              〔 △ 手続き種別－３ △ 手続き名－３ 〕 ；
```

（３）構文規則

（Ａ）手続き名は、空白文字を除く３２バイト以内の文字列とする。

（４）一般規則

（Ａ）ＧＲは保存用データのデータベースへの格納方法を示すために用いられる。

（ａ）手続き名－１は、保存用データにグループ・ノード・フレームが現れたときに呼び出される。

パラメータには、グループのフィールド番号以降が順次設定される。データ属性は全て数値で渡される。

（ｂ）手続き名－２は、保存用データにデータ・ノード・フレームが現れたときに呼び出される。

パラメータには、タプル番号以降が順次設定される。

タプル番号とデータ数の属性は数値で渡される。

（ｃ）手続き名－３は、保存用データが終わったときか、次のグループ・ノード・フレームが現れたときに呼び出される。

パラメータには、フィールド番号とタプル番号が数値属性で渡される。

（ｄ）呼び出し可能な手続きは内部手続きである。

（ｅ）グループ・ノードのフィールド番号は数値定数で指定する。

（Ｂ）＜＞はＸＭＬデータを解析、実行する方法を示すために用いられる。

（ａ）＜＞は、ＸＭＬ要素に対する処理を指定する。

（ｂ）＜？＞は、ＸＭＬ宣言または処理命令に対する処理を指定する。

（ｃ）＜！＞は、ＤＴＤ，コメント、ＣＤＡＴＡに対する処理を指定する。

（C）XML要素に対する処理
（a）手続き名－１は、XML要素の開始タグが現れたときに呼び出される。
パラメータには、以下が渡される。
第１パラメータ　　：グループ・ノードのフィールド番号＝９９００３
第２パラメータ　　：タプル番号＝０
第３パラメータ　　：第４パラメータ以降のパラメータの数
第４パラメータ　　：タグのネームスペース名
第５パラメータ　　：タグ名
第６パラメータ以降：以下の３項目が続く
アトリビュート名のネームスペース名
アトリビュート名
アトリビュート値

（b）手続き名－２は、XML要素のデータが現れたときに呼び出される。
パラメータには、以下が数値で渡される。
第１パラメータ　　：タプル番号＝０
第２パラメータ　　：データの数＝１
第３パラメータ　　：データ

（c）手続き名－３は、XML要素の終了タグが現れたときに呼び出される。
パラメータには、以下が渡される。
第１パラメータ　　：グループ・ノードのフィールド番号＝９９００９
第二パラメータ以降：開始タグと同じ

（D）XML宣言または処理命令に対する処理
（a）手続き名－１は、XML宣言または処理命令が現れたときに呼び出される。
パラメータには、以下が渡される。
第１パラメータ　　：グループ・ノードのフィールド番号＝９９００１
第２パラメータ　　：タプル番号＝０
第３パラメータ　　：第４パラメータ以降のパラメータの数
第４パラメータ　　：ＮＵＬＬ
第５パラメータ　　：ｘｍｌまたは処理命令
第６パラメータ以降：XML要素と同様

（b）手続き名－２と手続き名－３は、使用されない。

（E）ＤＴＤ，コメント、ＣＤＡＴＡに対する処理
（a）手続き名－１は、ＤＴＤ，コメント、ＣＤＡＴＡが現れたときに呼び出される。
パラメータには、以下が渡される。
第１パラメータ　　：グループ・ノードのフィールド番号＝９９００２
第２パラメータ　　：タプル番号＝０
第３パラメータ　　：第４パラメータ以降のパラメータの数
第４パラメータ　　：ＮＵＬＬ
第５パラメータ　　：ＤＴＤタグ名、'--', CDATA
第６パラメータ以降：XML要素と同様

（b）手続き名－２と手続き名－３は、使用されない。

3.13. ＳＬＥＥＰ

（１）　機能

処理を休止する。

（２）　一般形式

```
ＳＬＥＥＰ［ △ 式 ］;
```

（３）構文規則

なし。

（４）一般規則

（A）式は、ＳＬＥＥＰ値を示し、秒で指定する。ミリ秒(少数点以下３桁)未満は切り捨てられる。ＳＬＥＥＰ値がないときは、ＳＬＥＥＰ値は１となる。

（B）ＳＬＥＥＰ値は、正の数値型または文字型で指定することができる。文字型のときは、正の数値文字列でなければならない。

3.14. ＬＥＡＶＥ

（１） 機能

コマンドを指定ホストに対して転送する。

（２） 一般形式

ＬＥＡＶＥ　　転送ホスト;

転送ホストは { 変　数 / 文字定数 } とする。

（３） 構文規則

なし。

（４） 一般規則

（A）本コマンドが実行された時点で、本コマンドを含むスクリプトが起動される原因となったコマンドが転送される。

転送されたときには、実行中の処理は消滅する。

転送されたホストでは、端末からのコマンドは、最初から実行される。

（B）転送ホストが自ホスト、NULLパラメータ、NULL値の場合は何もせずに正常リターンする。

（C）本コマンドを実行する処理において、現在実行中のスクリプトがトランザクション実行中の場合には、$ERROR（-11060）でエラーリターンする。

（D）指定したホストＩＤに対応するホストＩＤが見つからない場合には、スクリプト実行エラーで、スクリプトの実行を終了する。

（E）転送されてきたコマンドを更に、他のホストへ転送しようとした場合には、$ERROR（-11061）でエラーリターンする。

（F）一連のスクリプト実行中に、LEAVEコマンドの実行は複数回できるが、他ホストへの転送は、最後の１回のみ実行する。

（５）例

（A）コマンド中に文書IDがあるとき

```
@%1:host_cod + dcid_nmb

   BEXP $1 = %1 SUBSTR '1,4';

   LEAVE $1;  @他ホスト転送コマンド
```

（B）コマンド実行中に文書のホストが決まるとき

ホストが決まった時点で転送コマンドが実行される様に記述する。

3.15. ＬＥＴ

3.15.1. 代入演算

（１）機能

式の値を変数に代入する。

（２）一般形式

［ ＬＥＴ ］［ △ 修飾子 ］［ △ データ型 ］△ { 式 / 代入式 } ；

修飾子およびデータ型は、ＤＥＦＩＮＥコマンドを参照。

代入式は

［前置代入演算子］[△］変数式［[△］後置代入演算子］［[△]代入演算子［[△］式 ］］

（注）式には、代入式も含まれる。

変数式は

式

代入演算子は

＝
＋＝
－＝
＊＝
％＝　または　ＭＯＤ＝
／＝
＆＝
｜＝
＾＝
～＝
＆＋＝　または　｜＋＝　または　ＣＯＮＣＡＴ＝

前置代入演算子、または、後置代入演算子は

＋＋
－－
！！
～～

（３）構文規則

（Ａ）代入演算子が、～＝ のときは、右辺の式は指定できない。

（４）一般規則

（Ａ）外部変数に対する演算は常に実行される。

（Ｂ）後置代入演算子は、式または代入式の全体が実行された後に実行される。

（Ｃ）変数式の式は、式の評価結果が変数でなければならない。

（Ｄ）代入式の右辺が、配列名または構造体名のときは、それぞれのアドレスが代入され、'*' を変数名の前に付けるとデータ実体が代入される。

3.15.2. パラメータ・シフト

（1）機能

パラメータを前に１つシフトする。

（2）一般形式

```
［ ＬＥＴ △ ］ＳＨＩＦＴ［ △ ＬＯＣＡＬ ];
```

（3）構文規則

なし。

（4）一般規則

（A）パラメータがないときは、何もしない。

（B）ＬＯＣＡＬが指定されたときは、内部手続きのパラメータをシフトする。
シフトするのは、パラメータにマップされた本手続き内のMAPPEDARRAY変数のみであり、仮引数の変数値は変わらない。

3.15.3. ループ継続

（1）機能

ループの先頭に戻る。

（2）一般形式

```
［ ＬＥＴ △ ］ＣＯＮＴＩＮＵＥ［ △ 式 ];
```

（3）構文規則

なし。

（4）一般規則

（A）ＣＯＮＴＩＮＵＥコマンドと同じ。

3.15.4. 関数実行

（1）機能

関数を実行する。

（2）一般形式

```
［ ＬＥＴ △ ］関数式 ；
```

（3）構文規則

なし。

（4）一般規則

なし。

3.15.5. オプション設定

（1）機能

オプションを設定する。

（2）一般形式

```
[ＬＥＴ△]｛ＯＰＴＩＯＮＳ　［△ オプション値－１］［△ オプション値－２］・・・｝;
[DEFINE]△｛ＯＰＴＩＯＮ　　オプション番号 △ オプション値　　　　　　　　　　｝
```

（3）構文規則

（A）ＯＰＴＩＯＮＳのときは、オプション番号１のオプション値から順に指定する。
　　オプションの途中をスキップするときは、NULLパラメータまたはNULL値を指定する。

（B）オプション番号と値は、数値式で指定する。式の値が文字型のときは数値に変換される。

（C）DEFINEは、定義部で使用するときに使用する。

（4）一般規則

（A）オプションは、セッション間で有効となる。

（B）オプションの内容は、付録を参照。

3.15.6. スクリプト実行終了

（1）機能

即座にスクリプト実行を中断し、セッションを終了する。

（2）一般形式

```
［ＬＥＴ △］｛ＥＸＩＴ｝［△ 式］；
            ｛ＱＵＩＴ｝
            ｛ＢＹＥ　｝
```

（3）構文規則

なし。

（4）一般規則

（A）式が指定されているときは、その値がリターン値となる。

（B）ＴＲＹ、ＣＡＴＣＨ、ＦＩＮＡＬＬＹブロックの延長内のときは、終了する全スクリプト内にある全てのＦＩＮＡＬＬＹブロックを内側から順に実行する。
　ただし、ＦＩＮＡＬＬＹブロック内のとき、自ＦＩＮＡＬＬＹブロックは除く。
　実行したＦＩＮＡＬＬＹブロック内で例外が発生したときは、ＥＸＩＴ処理を中断し、例外処理が実行される。

3.15.7. ＮＵＬＬ行

（1）機能

何もしない。

（2）一般形式

```
［ＬＥＴ △］ＮＵＬＬ；
```

（3）構文規則

なし。

（4）一般規則

なし。

3.15.8. ＬＰＲＩＮＴ

（１）機能

式の値をPRINT_LOG(注1)またはDEBUG_LOG(注2)に出力する。

(注1)ログ番号PRINT_LOGで示される媒体に出力する。デフォルトでは、標準出力とファイル出力が設定されている。詳細は、Ｃｏａｌ操作マニュアルを参照。
(注2)ログ番号DEBUG_LOGで示される媒体に出力する。デフォルトでは、出力なしが設定されている。詳細は、Ｃｏａｌ操作マニュアルを参照。

（２）一般形式

```
［ ＬＥＴ △ ］ＬＰＲＩＮＴ［ △ オプション ］［ △ 式 ］・・・ ；
```

（３）構文規則

オプションは、式と式の間にも指定できる。

（４）一般規則

（Ａ）以下の形式でPRINT_LOGに出力する。

式が、文字定数のとき、または、数値で少数点と数字のみのときは、"式="を出力しない。

標準出力形式

```
[coal] 式=式の値, 式=式の値, ・・・
```

ファイル出力形式

```
yyyy/mm/dd hh:mm:ss coal:  式=式の値, 式=式の値, ・・・
```

・数値属性：１０進表示
・文字属性：二重引用符で囲んで表示
・ＢＵＬＫ：１６進ダンプ表示(始めの512バイトまで)
　　(式の値の部分には、"**BULK**"を出力する)
・関数名　："関数名()"
・配列名　：アクセス属性 データ属性 配列名 配列属性
　　配列属性は、
　　MAPPEDのとき： (MAPPED INDEX)[次元１,次元２,次元３]
　　HASH　のとき： Hash(配列の大きさの初期値)
　　その他のとき：［次元１,次元２,次元３］
・構造体名、構造体定義名：アクセス属性 構造体名={要素名=属性,・・・}
・*配列名　：通常配列：値,・・・
　　連想配列：[キー値]=値,・・・
・*構造体名：{要素名=値,・・・}
・範囲値の展開：値,・・・
・クラス名："Class クラス名"
・インスタンス名："Instance as クラス名"

（Ｂ）式がエラーのときは、"##ERROR##"を出力し、処理を継続する

（C）オプションは、以下の形式で指定する。

オプションは、指定した以降のデータ出力時の２重引用符とカンマの出力、デバッグ形式出力を制御する。

一般形式 | /制御文字[+|-] |　　制御文字 | {c|q|d|x|e}

c: カンマの制御指定。
q: ２重引用符の制御指定。
d: デバッグ形式(情報構造体(システム内部での管理情報)の内容出力)での出力制御指定。
x: 整数値を１６進表示する。(%08x)
e: ERROROUT()を使用してエラーログに出力する。
m: １０進小数点で３桁毎にカンマを出力する。(+,- 以外のときは、実行時オプション)
+: 制御対象を出力する。
-: 制御対象を出力しない。
+,- 省略時: 制御対象を出力する。

（D）デバッグ形式での出力時は、以下の形式でDEBUG_LOGに出力する。

出力内容の詳細は、付録を参照。

（a）式が「式の並び」でないとき（標準出力）

```
[coal] DUMP1= pInfo= ・・・
[coal] DUMP2= pInfo= ・・・
  ・・・
[coal] 式=#DUMP1#, 式=#DUMP2#, ・・・
```

（b）式が「式の並び」のとき（標準出力）

```
[coal] DUMP1= pInfo= ・・・
[coal] PARMINFO2: pInfo= ・・・
[coal] DUMP1-1= pInfo= ・・・
[coal] DUMP1-2= pInfo= ・・・
  ・・・
[coal] 式=#DUMP1#->#DUMP1-1#,#DUMP1-2#, ・・・
```

（c）ファイル出力形式

″[coal]″の代わりに以下を出力する。

```
yyyy/mm/dd hh:mm:ss coal/ソースファイル名(ソース行数):
```

3.15.9. ＰＲＩＮＴ

（1）機能

式の値を標準出力に出力する。

（2）一般形式

［ ＬＥＴ △ ］ＰＲＩＮＴ［ △ オプション ］［ △ 式 ］・・・ ；

（3）構文規則

ＬＰＲＩＮＴと同じ。

（4）一般規則

以下を除き　ＬＰＲＩＮＴ　と同じ。

（A）″[coal]″は出力しない。

3.15.10. ＥＣＨＯ　または　ＳＡＹ

（1）機能

式の値を標準出力に出力する。

（2）一般形式

［ ＬＥＴ △ ］｛ ＥＣＨＯ｜ＳＡＹ ｝［ △ オプション ］［ △ 式 ］・・・ ；

（3）構文規則

ＰＲＩＮＴと同じ。

（4）一般規則

以下を除き　ＰＲＩＮＴ　と同じ。

（A）式=″は出力しない。

（B）文字属性データは、２重引用符で囲まない。

（C）各出力データの間には、1個の半角スペースを出力する。

3.15.11. ＤＵＭＰ

（１）機能

式の値をデバッグ形式でDEBUG_LOGに出力する。

（２）一般形式

```
［ ＬＥＴ △ ］ＤＵＭＰ ［ △ 式 ］・・・ ；
```

（３）構文規則

オプションを除き、ＰＲＩＮＴと同じ。

（４）一般規則

以下の形式で出力する。

出力内容の詳細は、付録を参照。

（ａ）式が「式の並び」でないとき

```
[coal] 式= pInfo= ・・・
[coal] 式= pInfo= ・・・
  ・・・
```

（ｂ）式が「式の並び」のとき

```
[coal] 式= pInfo= ・・・
[coal] PARMINFO2: pInfo= ・・・
[coal] 式= pInfo= ・・・
[coal] PARMINFO2: pInfo= ・・・
  ・・・
```

（ｃ）ファイル出力形式

″[coal]″の代わりに以下を出力する。

```
yyyy/mm/dd hh:mm:ss coal/ソースファイル名(ソース行数):
```

3.15.12. ＩＮＴＥＲＡＣＴＩＶＥ

（１）機能

対話モードでコマンドを実行する。

（２）一般形式

```
［ＬＥＴ △］ＩＮＴＥＲ［ＡＣＴＩＶＥ］ ；
```

（３）構文規則

なし。

（４）一般規則

（Ａ）対話モードでは、手続き、または、関数内で実行できるコマンドを実行できる。

（Ｂ）対話モードは、ＲＥＴＵＲＮ、または、ＬＥＴ　ＥＸＩＴコマンドで終了する。

（５）入力規則

（Ａ）対話モードでの入力方法は以下の通り。

・対話モードに入ると、"COAL>"のプロンプトが出力される。
・プロンプトに続いて、文を入力する。文はフリーフォーマットで入力することできる。
・入力された文は、解析され、コマンドtreeに追加される。
・改行で区切って複数行の文を入力することできる。">"のプロンプトが出力される。
・１回の文の入力は、改行のみの入力で終了する。
・入力の終了で、コマンドtreeが実行される。実行後、コマンドtreeは削除される。

【例】

```
coal>array a 10;
>for (i=0;i<5;i++);
>    a[i]=i+1;
>next;
>for (i=0;i<5;i++);
>    print a[i];
>next;
>(改行)
1
2
3
4
5
coal>
```

（Ｂ）文の保存と編集

・入力された行は、行番号付きで入力履歴域に保存される。
・行番号なしで入力された行は、自動で採番され入力履歴域の末尾に追加される。
(採番のデフォルト値は、初期値=10、増分値=10)
行は、コマンドtreeに追加される。
・行番号付きで入力された行は、その番号で入力履歴域に上書きまたは挿入・追加される。
行は、コマンドtreeに追加されない。
・行番号のみか、行番号付き空白行のときは、その行が入力履歴域から削除される。
・行番号と文の間は、１個以上の空白文字を開ける。行番号の後、２文字目以降がそのまま入力履歴域に保存される。
・行番号が、ゼロ以下のときは、入力行は無視される。

（C）サブコマンド

No	サブコマンド	機能
1	/	入力履歴域に保存された文を実行する。
2	/**l**ist［**t**ree］	″tree″なしのときは、入力履歴域に保存された文を表示する。 ″tree″のときは、生成済みコマンドtreeのみを表示する。
3	/**r**enum ［初期値][,|△][増分値］	採番のデフォルト値を変更し、行番号を振り直す。 値を省略するか、ゼロ以下のときは、デフォルト値は変更されない。
4	/**d**el［**t**ree］	″tree″なしのときは、入力履歴域に保存された文と生成済みコマンドtreeを全て削除する。 ″tree″のときは、生成済みコマンドtreeのみを削除する。
5	/**t**ree	入力履歴域に保存された文からコマンドtreeを生成し直す。
6	/**q**uit	終了する。

（F）スクリプト例

test_inter.cl

```
//
proc main;
    option 7 2;   // エラーを無視して処理を継続する
    option 8 2;   // 手続き／関数の入れ子をサポートする
    inter;
    return ERROR;
end proc;
```

3.15.13. ＬＯＧＰＡＲＭ

（１）機能

ログパラメータを設定する。

（２）一般形式

```
［ ＬＥＴ △ ］ＬＯＧＰＡＲＭ △ log_no △ flag［ △ level
                          △ size_max △ file_max △ option △ log_file ］；
```

（３）構文規則

log_no、flag、level、size_max、file_max、option、log_fileには、式を指定できる。

（４）一般規則

（A）パラメータの内容を以下に示す。

No.	パラメータ	内容
1	log_no	0:エラーログ　　または　'ERROR_LOG' 1:プリントログ　または　'PRINT_LOG' 2:デバッグログ　または　'DEBUG_LOG' 3:統計情報ログ　または　'STATI_LOG' 4:トレースログ　または　'TRACE_LOG'
2	flag	0x0001:標準出力に出力する 0x0002:ファイルに出力する 0x0004:標準エラー出力に出力する(標準出力なし) 0x0008:SYSLOGに出力する 0x0010:ソースファイル名を出力しない 0x0020:プロセス名を出力しない 0x0040:日時を出力しない 0x0080:プライオリティを出力する 0x0100:ログファイルをクリアする 0x0200:ログ出力情報を出力する 0x0400:プライオリティをチェックする 0x1000:LFを出力しない 0x2000:CRを出力する 0x4000:出力メッセージ中のLFを変換対象とする 0x8000:ログを出力せずにスタックする
3	level	出力レベル
4	size_max	ローテーション・ファイル・サイズ(Kbyte) >0のとき，ローテーションする。
5	file_max	ローテーション・ファイル数
6	option	ローテーション・オプション 0x01ビット: 0のとき，退避ファイル名をローテーションする。 1のとき，出力ファイル名をローテーションする。
7	log_file	出力ファイル名 実行ディレクトリからの相対パスか絶対パスで指定する

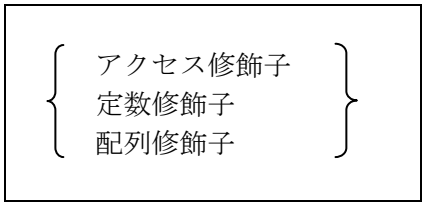
3.16. ＤＥＦＩＮＥ

3.16.1. 変数の修飾子

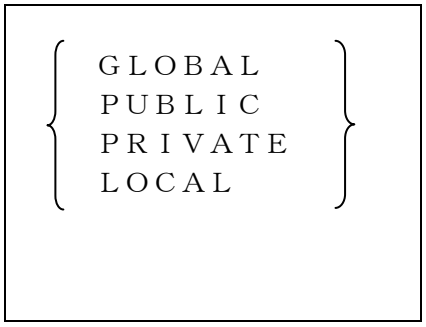
（１）機能

変数の属性を指定する。

（２）一般形式



アクセス修飾子は、



定数修飾子は

配列修飾子は

（３）構文規則

（Ａ）各修飾子は、順不同で指定できる。

（Ｂ）ＬＯＣＡＬは、ＤＥＦＩＮＥコマンドでは、指定できない。

（４）一般規則

（Ａ）アクセス修飾子は、以下を定義する。

No.	修飾子	変数の種別	スコープ
1	ＧＬＯＢＡＬ または ＰＵＢＬＩＣ	外部変数	セッション内(スクリプト間)
2	ＰＲＩＶＡＴＥ	PRIVATE変数	スクリプト内
3	ＬＯＣＡＬ	LOCAL変数	手続き内または関数内

（Ｂ）アクセス修飾子を省略したときは、以下となる。

No.	コマンド	省略時のアクセス修飾子
1	ＤＥＦＩＮＥ	ＰＲＩＶＡＴＥ
2	ＬＥＴ	旧仕様：ＰＲＩＶＡＴＥ 新仕様：ＬＯＣＡＬ (オプションで変更可能)

（Ｃ）ＣＯＮＳＴを指定すると定数となり、値を変更することができなくなる。

（Ｄ）ＡＲＲＡＹは、配列変数を定義するときに指定する。

ただし、配列定義を指定したときは、省略可能。

3.16.2. 変数のデータ型指定

（１）機能

変数のデータ型を指定する。

（２）一般形式

```
{ BINARY     |  INTEGER
  FLOAT | FLT | DOUBLE | DBL
  DECIMAL[(精度[, 位取り])]
  CHAR[(長さ)] | STRING[(長さ)]
  BULK[(長さ)]
  VARIANT                      }
```

ＤＥＣＩＭＡＬの制度と位取り、ＣＨＡＲ・ＳＴＲＩＮＧ・ＢＵＬＫの長さは

式

（３）構文規則

~~（Ａ）″ＤＥＣ（精度,位取り）″、″ＣＨＡＲ（長さ）″、″ＢＵＬＫ（長さ）″は１パラメータとすること。~~

（４）一般規則

（Ａ）ＢＩＮＡＲＹまたはＩＮＴＥＧＥＲは整数型を示す。

（Ｂ）ＦＬＯＡＴ、ＦＬＴまたはＤＯＵＢＬＥ、ＤＢＬは２進浮動少数点数型を示す。

（Ｃ）ＤＥＣＩＭＡＬは１０進浮動少数点数型を示す。
精度を指定したときは、１０進固定少数点数型となる。
精度は、１～５１、位取りは、－１２８～１２７を指定することができる。
精度は、数字部分の桁数を示す。
位取りは、正のときは、少数点以下の桁数を、負のときは、少数点以上の丸め桁数を示す。

（Ｄ）ＣＨＡＲまたはＳＴＲＩＮＧは文字型を、ＢＵＬＫはバルク型を示す。長さはバイト単位で指定する。
設定するデータの長さが定義長を超えた部分は捨てられる。逆に、満たない場合は、その要素のデータ長は、設定データ長となる。
長さを指定しなかったときは、可変長となる。

（Ｅ）ＶＡＲＩＡＮＴは、データ型が固定されない型で、代入されたデータの型でデータが保存される。データ型を省略したときは、ＶＡＲＩＡＮＴ型となる。

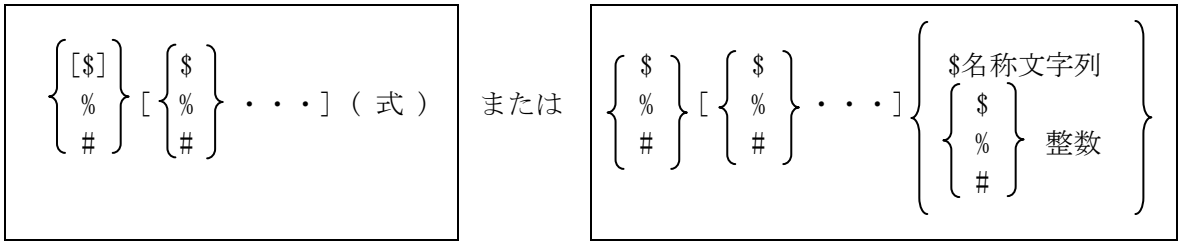
（Ｆ）データ型を指定すると指定された型でデータが保存される。
変数に値が代入されるとき、型が合わないときは定義された型に自動変換される。

3.16.3. 変数名の式指定

（1）機能

変数名を式で指定する。

（2）一般形式



（3）構文規則

（A）変数名の先頭には、それぞれの変数種別を示す以下の記号を付ける。

ただし、先頭の％と＃は、マップド配列変数定義のときのみ指定できる。

＄：内部番号変数（省略可能）

％：パラメータ変数

＃：検索変数

（4）一般規則

（A）変数名を式で指定するときは、先頭の記号を除いた部分の評価結果が名称文字列となる。

″（式）″のみのときは、式の値が名称文字列となる。

3.16.4. 変数定義

（1）機能

変数を定義する。

（2）一般形式

形式1

```
［ＤＥＦＩＮＥ］［ △ 修飾子 ］［ △ データ型 ］△ 変数名 ［ △ ＝ △ 式 ］ ；
```

形式2

```
｛［ＤＥＦＩＮＥ］△ ＶＡＲ｝［ △ 修飾子 ］△ 変数名-1［ △ 配列の定義-1 ］
 ＤＩＭ                    ［［, ］ △ 変数名-2［ △ 配列の定義-2 ］
                           [, ］・・・ ］

                  ［ △ ＡＳ △ ｛ 構造体型名
                                 配列型名
                                 データ型 ｝］

    ［［△］＝ △［（ ］［△］式－１［［△］, ［△］式－２ ］・・・［ ）］］；
```

（3）構文規則

（A）修飾子に配列修飾子は指定できない。

（B）形式2では、変数の型指定(AS ...)が構造体でも配列でもないときは、″[]″を使った配列定義を指定することできる。

（4）一般規則

（A）代入式を指定して、初期値を設定することができる。
ただし、形式2では、最後の変数が初期化対象となる。また、構造体の初期化はできない。

（B）外部変数においては、複数のスクリプトにおいて、同じ変数に対する初期値設定が定義されているときは、最初に実行された設定が有効となる。

~~（C）DIMは手続き内または関数内で指定できる。~~

（D）配列型名にマップド配列の型名を指定するときは、変数名の先頭に、'$'、'%'、'#'を付けて、変数の種類を指定する。ただし、'$'は省略できる。

（E）変数の型指定を省略したときは、バリアント型となる。

3.16.5. マップド配列変数定義

（1）機能

内部番号変数、パラメータ変数、検索変数にマップされた配列変数を定義する。

（2）一般形式

［ＤＥＦＩＮＥ］△ ＭＡＰＰＥＤＡＲＲＡＹ △ 変数名

（ △ インデックス［△ 次元１［△ 次元２［△ 次元３］］］ ）

（ [インデックス［, 次元１［, 次元２［, 次元３］］］] ）

［［△］＝ △［（ ］［△］式－１［［△］,［△］式－２ ］・・・［ ）］］］；

(注) ’**[**’,’**]**’は、そのまま指定する。’**[**’,’**]**’の代わりに’**(**’,’**)**’も使用可能。

インデックスは

式

次元１～３は

［ 式 ｜ 範囲式 ］

（3）構文規則

（A）変数名の先頭には、それぞれの変数種別を示す以下の記号を付ける。

＄：内部番号変数（省略可能）

％：パラメータ変数

＃：検索変数

（B）インデックスおよび配列の定義で、″[]″を使用しないでインデックスまたは各次元に式を指定するときは、式内は詰めて指定し、インデックスおよび各次元間のみが空白で区切られるようにすること。また、このとき、途中を省略するときは、NULL文字を指定する(NULLパラメータは無視される)。

（4）一般規則

（A）インデックスは、マップする配列の番号を指定する。
省略時、または、指定値がNULL値のときは、１と見なされる。

（B）次元を省略したとき、または、指定値がNULL値のときは、次元1は10、他は1と見なされる。

（C）次元の配置は、Ｃ言語と同様である。オプション１５の指定により、FORTRAN形式となる。

（D）範囲式では、下限値、上限値の順に指定する。
下限値または上限値が整数でないときは、整数に変換される。

（E）内部番号変数のときは、代入式を指定し、式の値を配列変数の先頭要素から順に設定することができる。

（F）配列変数は、PRIVATE変数として定義される。

3.16.6. 配列変数定義

（1）機能

配列変数を定義する。

（2）一般形式

```
［ＤＥＦＩＮＥ］［ △ 修飾子 ］［ △ データ型 ］△ 変数名［ △ 配列の定義 ］

      ［［△］＝ △［（ ］［△］式－１［［△］，［△］式－２ ］・・・［ ）］］；
```

配列の定義は

```
｛  次元１〔 △ 次元２〔 △ 次元３ 〕〕                    ｝
｛ [ 次元１〔 ，  次元２〔 ，  次元３ 〕〕]               ｝
｛ ＨＡＳＨ｛ △ 配列の大きさの初期値                    ｝｝
｛          ｛ [  配列の大きさの初期値 ]                ｝｝
```

(注)
'['，']'は、そのまま指定する。
データ型が指定されているときは、'['，']'の代わりに'('，')'も使用可能。

配列の大きさの初期値は

```
式
```

次元１～３は

```
［ 式 ｜ 範囲式 ］
```

（3）構文規則

（A）代入式におけるかっこは、複数の式の並びを囲む形で任意に指定することができる。

（B）配列の定義で、"[]"または"()"を使用しないで各次元に式を指定するときは、式内は詰めて指定し、各次元間のみが空白で区切られるようにすること。

（4）一般規則

（A）次元を省略したとき、または、指定値がNULL値のときは、次元1は10、他は1と見なされる。

（B）次元の配置は、Ｃ言語と同様である。オプション１５の指定により、FORTRAN形式となる。

（C）ＨＡＳＨは連想配列を示す。配列の大きさは、ＩＮＴの最大値まで自動拡張される。
配列の大きさの初期値を省略したとき、または、指定値がNULL値のときは、１０が取られる。

（D）代入式を指定したときは、式の値が配列の先頭要素から順に設定される。
連想配列のときは、式の値が、順に、キー値、データ値となる。

（E）範囲式では、下限値、上限値の順に指定する。
下限値または上限値が整数でないときは、整数に変換される。

3.16.7. 手続き内定義

（１）機能

手続き内または関数内に指定して、以下を定義する。

・スカラー変数
・マップド配列変数
・内部配列変数
・外部配列変数
・型定義変数

（２）一般形式

（Ａ）DEFINEの代わりにLETを使用する。他は、上記変数定義と同じ。

（Ｂ）定義済み配列に対しては、配列修飾子を指定し、配列定義を省略することで、
配列定義時と同様に、代入式によって先頭の要素から順に値を設定することができる。
以下のＡＲＲＡＹとアクセス修飾子は、順不同。
マップド配列のときは、″ＡＲＲＡＹ″でも″ＭＡＰＰＥＤＡＲＲＡＹ″でも良い。

［ ＬＥＴ ］△｛ ＡＲＲＡＹ ［ △ アクセス修飾子 ］ ｜ ＭＡＰＰＥＤＡＲＲＡＹ ｝ △ 変数名

［ △ ＝ △［（ ］［△］式－１［［△］，［△］式－２ ］・・・［ ）］］；

変数名を式で指定するときは、配列変数定義と同じ。

（Ｃ）手続き内または関数内定義では、LOCALを指定できる。

（３）構文規則

上記変数定義と同じ。

（４）一般規則

（Ａ）外部変数と外部配列変数は、DEFINEと同じ。ただし、変数への代入は常に行われる。

（Ｂ）マップド配列変数と内部配列変数は、変数が手続き内変数となる以外は、DEFINEと同じ。

（Ｃ）マップド配列変数をパラメータにマップしたときは、手続きのパラメータにマップされる。

（Ｄ）修飾子とデータ型の両方の指定がないか、または、アクセス修飾子のみのときは、'['と']'の指定は、配列要素と見なされる。

【例】
```
x[10,2] = 100;          // 配列要素
local a[0] = 3;         // 配列要素
array x[10,2] = 100;    // 配列定義
char(5) a[3] = 3;       // 配列定義
```

（Ｅ）スカラー変数では、定義済み変数であっても、データ型が指定された場合は、常にそのデータ型で初期化される。

3.16.8. オプション設定

ＬＥＴコマンドを参照。

3.16.9. 型定義

（1）機能

構造体、配列、マップド配列の型を定義する。

（2）一般形式

```
[DEFINE] △ TYPE △ STRUCT △ 構造体型名
    ┌ [ △ データ型 ] △ 変数名 〔 △ 配列の定義 〕〔 [△] , 〕 ┐
                    ・  ・  ・
      [ △ データ型 ] △ 変数名 〔 △ 配列の定義〕 ;
```

```
[DEFINE] △ TYPE △ { ARRAY [ △ データ型 ] | MAPPEDARRAY } △ 配列型名
                  [ △ 配列の定義 ] ;
```

構造体型名、変数名、配列型名を式で指定するときは

```
[ $ ]( 式 )
```

データ型および配列の定義は、内部配列変数と同じ。
構造体内の変数名をメンバ変数名、変数をメンバ[変数]と呼ぶ。

（3）構文規則

（C）を除き内部配列変数と同じ

（4）一般規則

（A）次元は内部配列変数と同じ。
（B）データ型は内部配列変数と同じ。
（C）ＨＡＳＨは内部配列変数と同じ。

3.17. ＲＥＤＥＦＩＮＥ

3.17.1. マップド配列変数再定義

（１）機能

手続き外または手続き内で定義されたマップド配列変数を再定義する。

（２）一般形式

ＤＥＦＩＮＥコマンドをＲＥＤＥＦＩＮＥに置き換えたものと同じ。

（３）構文規則

ＤＥＦＩＮＥコマンドと同じ。

（４）一般規則

（Ａ）インデックスおよび次元１～３を変更可能。

3.17.2. 内部および外部配列変数再定義

（１）機能

手続き外または手続き内で定義された内部および外部配列変数を再定義する。

（２）一般形式

ＤＥＦＩＮＥコマンドをＲＥＤＥＦＩＮＥに置き換えたものと同じ。

（３）構文規則

ＤＥＦＩＮＥコマンドと同じ。

（４）一般規則

（Ａ）以下を変更可能。

①データ型がＣＨＡＲ、または、ＢＵＬＫのときの長さ。

②データ型がＤＥＣの精度と位取り。

③次元１～３。

（Ｂ）データ型がＣＨＡＲ、または、ＢＵＬＫのときの長さが短くなったときは、データは再定義長に切り捨てられる。ＤＥＣのときは、指定の精度と位取りとなる。

（Ｃ）ＨＡＳＨ配列は、再定義できない。

3.17.3. 内部および外部変数再定義

（１）機能

手続き外または手続き内で定義された内部および外部変数を再定義する。

（２）一般形式

ＤＥＦＩＮＥコマンドをＲＥＤＥＦＩＮＥに置き換えたものと同じ。

（３）構文規則

ＤＥＦＩＮＥコマンドと同じ。

（４）一般規則

（Ａ）以下を変更可能。

①データ型がＣＨＡＲ、または、ＢＵＬＫのときの長さ。

②データ型がＤＥＣの精度と位取り。

（Ｂ）データ型がＣＨＡＲ、または、ＢＵＬＫのときの長さが短くなったときは、データは再定義長に切り捨てられる。ＤＥＣのときは、指定の精度と位取りとなる。

3.18. ＵＮＤＥＦＩＮＥ

（１）機能
内部(配列)変数または外部(配列)変数を未定義にする。

（２）一般形式

ＵＮＤＥＦＩＮＥ △ 変数名－１［ △ 変数名－２ ・・・ ］ ；

（３）構文規則
なし。

（４）一般規則
（A）

3.19. ＭＥＳＳＡＧＥ

3.19.1. MESSAGE SLEEP

ＳＬＥＥＰコマンドと同じ。

3.19.2. MESSAGE SQL

ＳＱＬコマンドと同じ。

3.19.3. MESSAGE SEND

（１）機能

メッセージを送信し、結果を受信する。

（２）一般形式

```
ＭＥＳＳＡＧＥ △ ＳＥＮＤ △ { ［ホスト名］ △［プロセス番号］ } △［クラス番号］
                             { ＣＨＡＮＮＥＬ △［チャネル番号］ }

        △［メッセージ］△［ファイル名］△［タイムアウト値(msec)］
      ［ △ ＲＥＣＶ［返信メッセージ］］ ；
```

ホスト名、プロセス番号、チャネル番号、クラス番号、メッセージ、ファイル名、
タイムアウト値は

式

返信メッセージは

{ 配列名 内部番号配列の要素番号 }

（３）構文規則

なし。

（４）一般規則

（Ａ）ＣＨＡＮＮＥＬが指定されたときは、チャネル番号のチャネルを使って通信する。

（Ｂ）ＲＥＣＶが指定されたときは、返信を待つ。

（Ｃ）結果は返信メッセージに指定された配列の先頭から順に以下の順番で設定される。

メッセージ以降の数、メッセージ、ファイル名－１、ファイル名－２、・・・

配列には、連想配列は指定できない。

（Ｄ）パラメータ省略時は以下となる。

No.	パラメータ	デフォルト値
1	ホスト名	自ホスト
2	プロセス番号	0
3	チャネル番号	0
4	クラス番号	0
5	メッセージ	なし
6	ファイル名	なし
7	タイムアウト値	規定のメッセージ・タイムアウト値
8	返信メッセージ	返信を待たない

3.20. ＳＷＩＴＣＨ

（１）機能

処理の選択を制御する。

（２）一般形式

```
   ＳＷＩＴＣＨ △ 式１ ;
[ ＣＡＳＥ △ 式２ [ , △ 式２ ] ・・・ ; ]
[     文の並び     ]
[ ＤＥＦＡＵＬＴ [;] ]
[     文の並び     ]
   ＥＮＤ[△]ＳＷ[ＩＴＣＨ];
```

（３）構文規則

（Ａ）ＣＡＳＥ文は、複数置くことができる。
（Ｂ）ＳＷＩＴＣＨ文は、入れ子にすることができる。

（４） 一般規則

（Ａ）式１と式２のどれかの値が等しいとき、ＣＡＳＥ文に続く文の並びを順次実行する。
　　実行する範囲は、当該ＣＡＳＥ文と同じレベルの次のＣＡＳＥ文、ＤＥＦＡＵＬＴ文、または、ＥＮＤＳＷ文の間。
　　１つのＣＡＳＥ文で一致したときは、他のＣＡＳＥ文で一致する式２があっても実行しない。
（Ｂ）式２の先頭に比較子があるときは、それで比較し、ないときは、等号で比較する。
（Ｃ）ＤＥＦＡＵＬＴ文とＥＮＤＳＷ文の間の文は、どのＣＡＳＥ文にも一致しなときに実行される。

3.21. ＦＵＮＣＴＩＯＮ

（１）機能

関数の開始を宣言する 。

（２）一般形式

```
ＦＵＮＣ[ＴＩＯＮ] △ 関数名 [ △ [変数名－１] {,|△} [変数名－２] ]
                  [ [△] ([データ型] △ [仮引数名－１]
                                  [ △ ＡＳ △ データ型 ] [△] , ・・・ ) ]
                   [ △ ＡＳ データ型 ] ；

            文の並び

ＥＮＤ[△]ＦＵＮＣ[ＴＩＯＮ]；
```

（３）構文規則

（Ａ）関数名は、空白文字を除く３２バイト以内の文字列とする。
（Ｂ）変数については、ＰＲＯＣと同じ。
（Ｃ）仮引数のデータ型指定は、ＰＲＯＣと同じ。

（４）一般規則

（Ａ）関数は入れ子にすることができる。(オプション番号 ８の指定)
（Ｂ）関数から呼出元へ復帰するためには、ＲＥＴＵＲＮコマンドを使用する。
ＲＥＴＵＲＮコマンドがなくＥＮＤＦＵＮＣに達したときは、リターン値は、ＲＥＴＵＲＮコマンドで、リターン値指定なしのときと同じになる。
（Ｃ）パラメータおよび仮引数については、ＰＲＯＣと同じ。
（Ｄ）関数の実行中にエラーがあった場合は、本関数を含む式の実行がエラーとなる。
~~（Ｅ）リターン値は、本関数を呼び出した手続きまたは関数内のローカル変数にも設定される。ローカル変数名は、関数名と同じ名前となる。~~
（Ｅ）″ＡＳ データ型″が指定されているときは、関数の戻り値を指定のデータ型に変換する。
指定のデータ型がＣＨＡＲまたはＢＵＬＫでデータ長の指定がないときは、返却値が指定のデータ型に変換され、変換されたデータ長のまま関数の戻り値となる。

3.22. ＩＭＰＯＲＴ

（１）機能

スクリプトをインポートする 。

（２）一般形式

ＩＭＰＯＲＴ △ スクリプト名［ △ インポートオプション ］・・・ ；

スクリプト名および
インポートオプションは
{ 定数 / 変数 / 名称文字列 }

インポートオプションの種類
{ ＨＥＡＤ / ＴＡＩＬ / ＵＳＥ＿ＭＡＩＮ }

（３）構文規則

（Ａ）スクリプト名は、空白文字を除く３２バイト以内の文字列とする。
ただし、ディレクトリ指定を含んでもよい。ディレクトリ指定部分はスクリプト名長の制限は適用されない。

（４）一般規則

（Ａ）ＩＭＰＯＲＴコマンドは、スクリプトの先頭で、ＰＲＯＣまたはＦＵＮＣＴＩＯＮの前に、なければならない。
（Ｂ）インポートしたスクリプト内にも、ＩＭＰＯＲＴコマンドを指定することができる。
（Ｃ）ＩＭＰＯＲＴコマンドは、出現した順に実行される。
（Ｄ）インポートしたスクリプトが挿入される位置は、オプションによって決まる。
（ａ）オプション番号６（コマンドラインの-o、または、ＬＥＴコマンドで指定する）
ＬＥＴコマンドのオプション設定を参照。
（ｂ）インポートオプション
"HEAD"が指定されたときは、本スクリプトは、先頭に挿入される。
"TAIL"が指定されたときは、末尾に追加される。
本指定は、オプション番号６より優先する。
（Ｅ）同じスクリプトを２度以上インポートした場合には、２度目以降はスキップされる。
（Ｆ）インポートしたスクリプト内に、main 手続きがある場合には、インポートオプションによって扱いが変わる。
インポートオプションに、"USE_MAIN"がないときは、当該 main は無効になる。
"USE_MAIN"があるときは、当該 main が有効になる。ただし、実行される main は、スクリプトの先頭からサーチして、最初に現れた有効な main である。
（Ｇ）インポートしたスクリプト内にあるＯＮおよびＤＥＦＩＮＥコマンドは、インポート時には実行されない。全てのインポートが終了した後に、スクリプト内にある全てのＯＮおよびＤＥＦＩＮＥコマンドが、先頭から順に実行される。

3.23. ＴＲＹ

（１）機能

例外処理を制御する。

（２）一般形式

```
    ＴＲＹ［;］
      通常処理部分(ＴＲＹブロックと呼ぶ)
    ＣＡＴＣＨ［△式］,［△式］・・・ ;
[     例外処理部分(ＣＡＴＣＨブロックと呼ぶ)  ]
[  ＦＩＮＡＬＬＹ［;］ ]
[     例外が発生しかどうかにかかわらず常に実行される部分
                                  (ＦＩＮＡＬＬＹブロックと呼ぶ) ]
    ＥＮＤ[△]ＴＲＹ;
```

（３）構文規則

（Ａ）ＴＲＹ文は、入れ子にすることができる。
（Ｂ）ＣＡＴＣＨ文は、１つ以上複数置くことができる。
（Ｃ）ＣＡＴＣＨ文の式を省略するときは、カンマも省略する。
（Ｄ）各ブロックには、複数の文を置くことができる。

（４）一般規則

（Ａ）例外が発生したときは、例外値は、システム変数$EXCEPTIONにセットされる。
　　例外値の種類については、付録を参照。
（Ｂ）ＴＲＹブロック内で例外が発生したときは、ＣＡＴＣＨ文が評価される。
　　式に例外の種類にある名称のみを指定したときは、指定した名称の部分までが比較される。
　　名称のみでないときは、指定した式の値が比較される。
　　例外値とＣＡＴＣＨ文の式のどれかの値が等しいとき、ＣＡＴＣＨブロックの文の並びを順次実行する。
　　実行する範囲は、当該ＣＡＴＣＨ文と次のＣＡＴＣＨ文、ＦＩＮＡＬＬＹ文、または、ＥＮＤＴＲＹ文の間。
　　１つのＣＡＴＣＨ文で一致したときは、他のＣＡＴＣＨ文で一致する式があっても実行しない。
　　式の先頭に比較子があるときは、それで比較し、ないときは、等号で比較する。
　　式がないときは、例外値のすべてに一致する。
（Ｃ）ＴＲＹブロック以外で例外が発生したときは、当該手続き、または、関数から例外状態のままリターンする。
（Ｄ）ＴＲＹブロック内の延長(呼び出された手続きまたは関数の中)で、例外が発生したときは、システム関数の実行時エラーも例外として扱われる。($ERRORにエラーコードがセットされるが、正常終了はしない)
（Ｅ）例外がどのＣＡＴＣＨ文とも一致しないときは、例外処理は保留され、ＦＩＮＡＬＬＹブロック実行後に、例外状態のまま呼び出し元に正常リターンする。

3.24. ＴＨＲＯＷ または ＲＡＩＳＥ

（１）機能

例外を発生させる。

（２）一般形式

```
ＴＨＲＯＷ △ 式 ；
```

（３）構文規則

なし。

（４）一般規則

（Ａ）式の値が０以外のとき、絶対値がユーザ例外値となり、例外を発生させる。
例外値はシステム変数$EXCEPTIONに設定される。

3.25. ＤＯ

（１）機能

ＤＯブロックを制御する。

（２）一般形式

```
ＤＯ；
      文の並び

{ ＥＮＤ[△]ＤＯ ；                          }
{ ＥＮＤ { ＷＨＩＬＥ } △ 条件式 ；          }
         { ＵＮＴＩＬ }
```

（３）構文規則

（Ａ）ＤＯは入れ子にすることができる。

（４）一般規則

（Ａ）ブロック内を１回実行する。

（Ｂ）ブロック内では、ＢＲＥＡＫまたはＣＯＮＴＩＮＵＥを使用できる。

（Ｃ）ＷＨＩＬＥ、ＵＮＴＩＬについては、ＬＯＯＰと同じ。

3.26. ＬＡＢＥＬ

（１）機能

ラベル。

（２）一般形式

形式１

```
ＬＡＢＥＬ［ △ ラベル名［ △ 任意の文字列 ］］;
```

形式２

```
ラベル名 :
```

（３）構文規則

（A）ラベルは、手続きまたは関数の内部でのみ使用できる。

（B）ラベル名には、名称文字列を指定する。

（４）一般規則

（A）ラベルは、何もしない。

3.27. ＣＬＡＳＳ

（１）機能
クラスの定義を開始する。

（２）一般形式

```
ＣＬＡＳＳ △ クラス名 [△ STATIC ];
    クラス変数の定義
    コンストラクタ関数の定義
    メソッド関数の定義
ＥＮＤ[△]ＣＬＡＳＳ；
```

（３）構文規則
（Ａ）クラス変数の定義は、手続き内定義と同じ。
（Ｂ）コンストラクタ関数の定義は、関数定義と同じ。関数名はクラス名と同じにする。
引数の数を変えて複数定義できる。
（Ｃ）メソッド関数の定義は、関数定義と同じ。
（Ｄ）"STATIC"は、大文字、小文字の混合でよい。

（４）一般規則
（Ａ）クラス変数の定義とコンストラクタ関数の定義は、インスタンス生成時に一回だけ実行される。コンストラクタ関数は、インスタンス生成時に指定する実引数の数と同じ仮引数の関数が実行される。
（Ｂ）クラス内では、クラス変数は、その前に"My."を付けても明示的に参照できる。
（Ｃ）"STATIC"は、静的クラスを示し、インスタンスを生成しなくても、クラス名を指定して、使用できる。
静的クラスにおいては、最初に使用されたときに、暗黙のインスタンスが生成され、セッション終了まで保持される。
（Ｄ）メソッド関数は、指定する実引数の数と同じ仮引数の関数が実行される。

4. 式

4.1. 一般形式

ここでの表記は、必ずしも厳密ではなく、一般的な形式を示している。括弧の挿入、左右の辺の交換等は適宜可能である。
なお、スクリプトの定義部分では、ユーザ定義関数は指定できない。

式は

［キャスト演算子］ { 代入式 / 論理式 / 比較式 / 算術式 / 文字列式 / ３項演算式 / 範囲式 / ドット式 / ｲﾝｽﾀﾝｽ生成式 }

（注）代入式については、ＬＥＴを参照。
（注）算術式には、ビット式を含むものとする。
（注）式を指定できる所には、式をカンマで区切って、複数個指定できる。このときには、最後の式が評価対象となる。
（注）データ要素以外の式にキャスト演算子を付けるときは、式全体を括弧で囲む。

論理式は

{ ［論理単項演算子］（ 論理式 ） / 比較式 [論理演算子 （ 論理式 ）] }

比較式は

算術式 [比較演算子 （ 比較式 ）]

または

文字列式 [比較演算子 （ 比較式 ）]

算術式は

{ ［算術単項演算子］（ 算術式 ） / ［算術単項演算子］ データ要素 [算術演算子 （ 算術式 ）] }

文字列式は

データ要素 [文字列演算子 （ 文字列式 ）]

３項演算式は

論理式 ？ 式 ： 式

範囲式は

```
式 . . 式
```

(注)第一番目の式が下限値、第二番目の式が上限値。

ドット式は

```
式１ . 式２
```

構造体メンバ、クラス変数にアクセスするため。または、クラス関数を実行するために使用する。

インスタンス生成式は

```
ＮＥＷ クラス名 [ ( 式〔 , 式 , ・・・ 〕) ]
```

データ要素は

```
{ [ アクセス修飾子 △ ] [*]変数
  定数
  関数式
  データ・リスト式
  [*](式)
}
```

(注)式中の文字定数は、スクリプトが旧仕様でも、新仕様で処理される。
　アクセス修飾子は、内部名称変数または外部変数が直接指定されているときにのみ指定できる。
　'*'は、データ指定単項演算子である。

関数式は

```
[{手続き名|関数名}.・・・・.] 関数名 ( [ 式 ] , [ 式 ] , ・・・ )
```

関数名は

```
{ 名称文字列
  式(値は関数名 または ″{}″で囲った関数本体の文字列を関数化した値)
  関数化関数(FF)の戻り値(内容は関数名)
}
```

変数は、配列変数

```
変数名 [ [ 式 ] , [ 式 ] , ・・・ ]
```

(注)'[',']'は、そのまま指定する。

または、スカラー変数

```
{ [$]   { 整数定数
  %       名称文字列
  #       変数
          ( 式 )
}       }
```

(注) $, %, #がネストする場合は、$は省略できない。

データ・リスト式は

左中括弧({) [式] , [式] , ・・・ 右中括弧(})

論理演算子は

{ AND | OR }
または
{ && }
{ || }

比較演算子は

{ EQ | NE | GT | LT | GE | LE }
または
{ == | != | NOT= | <> | >< | > | < | >= | => | <= | =< }

各記号は、以下を意味する

No	記号	意味
1	EQ、==	＝
2	NE、!=、NOT=、<>、><	≠
3	GT、>	＞
4	LT、<	＜
5	GE、>=、=>	≧
6	LE、<=、=<	≦

特殊な比較演算子は

```
{ ＬＩＫＥ｜ｉＬＩＫＥ｜ｉＥＱ｜ｉＮＥ｜
  ＩＮＳＴＲ｜ＩＮｉＳＴＲ｜ＩＮＲＳＴＲ｜ＩＮｉＲＳＴＲ
  ＩＮ｜ｉＩＮ }
```

各記号は、以下を意味する。詳細は、対応する関数を参照。

No	記号	意味
1	ＬＩＫＥ	パターンマッチング
2	ｉＬＩＫＥ	大文字、小文字を区別しない パターンマッチング
3	ｉＥＱ	大文字、小文字を区別しない＝
4	ｉＮＥ	大文字、小文字を区別しない≠
5	ＩＮＳＴＲ	文字列サーチ
6	ＩＮｉＳＴＲ	大文字、小文字を区別しない 文字列サーチ
7	ＩＮＲＳＴＲ	後ろからの文字列サーチ
8	ＩＮｉＲＳＴＲ	大文字、小文字を区別しない 後ろからの文字列サーチ
9	ＩＮ	複数文字列比較
10	ｉＩＮ	大文字、小文字を区別しない 複数文字列比較

算術演算子は

```
{ ＋ ｜ － ｜ ＊ ｜ ／ ｜ ％ ｜ ＭＯＤ ｜ ＊＊ }
```

または、以下のビット演算子

```
{ ＆ ｜ 縦棒(｜) ｜ ＾ ｜ ＜＜ ｜ ＞＞ }
```

文字列演算子は

```
{ ＣＯＮＣＡＴ ｜ ＆[＋] ｜ 縦棒(｜)[＋]
  ＳＵＢＳＴＲ
  ＲＥＰ
  ＩＳ
  ＣＯＮＤＡＳ
  ＴＯ                                  }
```

算術単項演算子は

```
{ ＋ ｜ － ｜ ～ }
```

論理単項演算子は

{ ! | NOT }

データ指定
単項演算子は

*

配列名、構造体名または範囲値に付け、データ指定属性を付与する。

３項演算子は

? :

キャスト演算子は

(データ型名)

キャストされたデータ型を以下の型に変換する。長さ省略時は、変換後のデータ長となる。

データ型名	データ型
CCHAR[(長さ)] または **CSTR**ING[(長さ)]	文字
CBIN または CINT	整数
CLONG または CLNG	(倍精度)整数
CFLOAT または **CDOU**BLE または CFLT または CDBL	倍精度２進浮動小数点
CDEC[(精度[, 位取り])]	１０進浮動小数点 または １０進固定小数点
CBULK[(長さ)]	バルク
CVARIANT	バリアント
FUNC	関数

(注)長さ、精度、位取りは、式。倍精度整数は未サポート、短精度になる。

その他演算子は

一部の関数の 関数名

関数名を二項演算の演算子として指定したときは、第一項、第二項を関数の第一引数、第二引数としたときと同じになる。引数が１の関数のときは第一項が使われ、第二項は無視される。

以下の関数群が対応する。（関数一覧を参照）

- 文字列演算子と同じ関数
- TO_XXXX関連
- 文字列演算関連
- 比較・マッチング関連

4.2. 論理式

（１）一般規則

（A）論理演算は、論理値で演算する。演算の両辺は数値属性でなければならない。

（B）両辺の値は、０なら偽、０でないなら真とみなす。

（C）論理演算の結果は、真なら１、偽なら０となる。

4.3. 比較式

（１）一般規則

（A）比較の両辺は同じ属性でなければならない。

両辺が数値属性の場合で、データの型が異なる場合は、順位の高い方に変換される。変換の順位は以下となる。

整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数（順位が高い）

（B）比較演算の結果は、真なら１、偽なら０となる。

（D）文字列の比較は、辞書順の大小で行う。比較は、１文字ごとのコードの大小比較となる。

［例］

```
( i)   ’a’      ＜ ’b’
( ii)  ’abc’    ＜ ’abcd’
(iii)  ’abc’    ＜ ’b’
```

4.4. 算術式

（１）一般規則

（A）定数又は変数は、数値属性でも文字属性でも良い。文字属性のときは、数値に変換する。

（B）演算において、どちらか一方が配列名のときは、他方は、配列の先頭からの相対位置と解釈され、’+’または’-’の結果は、その位置を先頭とする１次元配列となる。

（C）演算において、どちらか一方が数値属性のときは、他方も数値属性に変換される。

変換の順位は以下となる。

整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数（順位が高い）

（D）ビット演算は、ビット毎に演算する。演算の両辺は整数値に変換される。

（E）べき乗演算においては、演算の両辺は２進倍精度浮動小数点数に変換される。

4.5. 文字列式

（１）一般規則

（A）ＩＳ演算子の一部機能を除き、定数又は変数が数値属性のときは、数字文字列に変換する。

変換は、ゼロサプレスされた左詰めであり、負符号は先頭につく。

ＢＵＬＫ属性のときは、１６進表記文字列に変換する。変換はデータの１バイトを０～９、ａ～ｆの２バイトで表わす。

（2）文字列演算子

（A）文字列演算子の一覧

No.	文字列演算子	機能概要
1	ＣＯＮＣＡＴ ＆[＋]、｜[＋]	第一項の文字列と第二項の文字列を連結する。
2	ＳＵＢＳＴＲ	第一項の文字列から第二項で示される文字列を切り出す。
3	ＲＥＰ	第一項の文字列中の置換指定を第二項の文字列と置き換える。
4	ＩＳ	第一項または第一項を文字列に変換したものを第二項の指定で調べ、結果を返す。
5	ＣＯＮＤＡＳ	第一項の値に対して第二項で与えられる条件式を実行する。
6	ＴＯ	第一項の値を第二項で与えられる条件で変換する。

(注)文字列演算子は、関数名としても登録されているため、他の関数名と同じ扱いになる。

（B）文字列演算子の機能

（a）文字列の連結（演算子は、**ＣＯＮＣＡＴ　または、＆[＋]　または、｜[＋]**）

第一項の文字列と第二項の文字列を連結する。

第一項と第二項が文字列のときは、＆または｜でもよい。

（b）バイト単位の文字列の切出し（演算子は、**ＳＵＢＳＴＲ**）

第一項の文字列から第二項で示される文字列を切り出す。

第二項は、以下の形式の文字列で指定する。

’切出し開始位置［，切出し文字数］’

（ｉ）　切出し開始位置は先頭を１として数える。

（ii）　切出し文字数を省略したときは、切出し開始位置以降を切出す。

(iii)　切出し開始位置が範囲外のときはヌル文字列となる。

（iv）　切出し文字数が負または実際の文字数を越えるときは、省略時と同じ扱いとなる。

第二項が数値属性のときは、切出し開始位置のみが指定されたものとして扱う。

（c）文字列の置換（演算子は、**ＲＥＰ**）

第一項の文字列中の置換指定を第二項の文字列と置き換える。

置換指定は、以下の形式で指定する。

＠ｎ＠　　ｎは０以上の整数で第二項の置換文字列の順序を示す。（注１）

第二項は、置換指定の順序に対応する文字列を単価記号（@）２個（注１）で区切って指定する。

第一項文字列中に同じ置換指定があったときは、その全てが置き変わる。

また、第二項に対応するものがないとき、

（ｉ）［第１項］が@で指定されている場合

そのまま指定を残す。

(ii)［第１項］が＆で指定されている場合

ヌル文字列と置き換わる

＜例１＞

［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１

〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下＆＆1g＆＆105°＆＆1時間 ――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（1g，105°，1時間）となる。
〔＄３　――――　0.5%以下（1g，105°，1時間）〕

＜例２＞
［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１
〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下＆＆　――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（，＠２＠，＠３＠）となる。
〔＄３　――――　0.5%以下（，，）　〕

＜例３＞
［第１項］　＠０＠（＠１＠，＠２＠，＠３＠）――――　＄１
〔＆０＆（＆１＆，＆２＆，＆３＆）〕

［第２項］　0.5%以下　――――　＄２

＄３　＝　＄１　ＲＥＰ　＄２；
＄３　――――　0.5%以下（＠１＠，＠２＠，＠３＠）となる。
〔＄３　――――　0.5%以下（，，）　〕

（注１）アンパサント（&）２個を使用してもよい。

（ｄ）文字列の調査（演算子は、**ＩＳ**）
第一項または第一項を文字列に変換したものを第二項の指定で調べ、結果を返す。
第二項は先頭の文字列で判定する。

No	調査項目	第二項の指定 (小文字も同じ)	結果と例
1	文字列長	L	旧仕様：文字列バイト長 新仕様：文字列文字数 ＜例＞ $1 = 'abcd' IS 'length'; $1-->4
	文字列バイト長	ＬＢ	文字列バイト長
2	数字列	Ｎ、Ｄ、Ｆ	整数属性または整数文字列のとき１、 ２進浮動少数点属性または２進浮動少数点文字列のとき２、 １０進少数点属性または１０進少数点文字列のとき３、 その他のときは０となる(数値定数を参照)。 文字列のときは、前後のブランクは無視される。 ＜例＞ $1 = '-12' IS 'numelic'; $1-->1 $2 = '-12.34' IS 'numelic'; $2-->2 $3 = '-12A' IS 'numelic'; $3-->0 $4 = '-12.34E+01' IS 'numelic'; $4-->2
3	半角ブランク	Ｂ、Ｓ	半角ブランクのとき、１、その他のとき、０となる。 ＜例＞ $1 = ″' '″ IS 'blank'; $1-->1
4	１６進文字列	Ｘ	ＢＵＬＫ属性のとき、または文字列が半角ブランク、小数点、符号、数字、'a'～'f'、'A'～'F'の組み合わせのとき、１となり、その他のときは０となる。
5	全角文字	Ｚ	先頭文字が全角文字のとき、１となり、その他のときは０となる。
6	半角文字	Ｈ	先頭文字が半角文字のとき、１となり、その他のときは０となる。
7	ＡＮＫ文字	Ａ	先頭文字がＡＮＫ文字のとき、１となり、その他のときは０となる。
8	文字のバイト数	Ｍ	先頭文字のバイト数を返す。
9	属性	Ｔ	属性値（第一項は変換しない）
10	データＩＤ	Ｉ	データＩＤ（第一項は変換しない）

（ｅ）条件式（演算子は、**ＣＯＮＤＡＳ**）

第一項の値に対して第二項で与えられる条件式を実行する。

条件式の形式は、

区切り文字　比較文字列　区切り文字　真時代入文字列　区切り文字　偽時代入文字列

(i)　区切り文字は、任意の半角文字を使用し条件式の先頭１文字が区切り文字となる。

(ii)　比較文字列は、第一項と比較される文字列であり等しいかどうか比較される。

(iii) 比較の結果が真のとき、真時代入文字列が代入され、偽のとき、偽時代入文字列が代入される。

(iv)　代入文字列は、Ｃ言語の書式指定と同じ形式で指定し、%sが第一項の文字列に置換される。

%qを指定したときは、第一項の文字列中の引用符（’）１個が２個に変換された後に、%sと同じ様に置換される。

(v)　文字列が省略されたときは、ヌル文字列とみなされる。

＜例＞

区切り文字にスラッシュ（/）を使用する場合

```
$3 = $1 CONDAS '//null/''%s''';
```

$1がヌルのときに、$3にはnullという文字列が入り、ヌルでないときには、$1の内容の前後に引用符（’）が付いた文字列が入る。

（ｆ）変換（演算子は、**ＴＯ**）

第一項の値を第二項で与えられる条件で変換する。

第二項が変数データ型を示す文字列のときは、対応する型に変換する。

このときには、データ型には、制度、桁数、長さ、VARIANTは指定できない。

第二項が変数データ型を示す文字列でないときは、以下で変換する。

文字列中を変換する処理では、第一項は、文字列に変換される。

項番	第二項の文字列の先頭	処理
1	Ｚ，ｚ	文字列中の半角英数字記号を全角に変換する
2	Ｈ，ｈ	文字列中の全角英数字記号を半角に変換する
3	Ｕ，ｕ	文字列中の半角英字を大文字に変換する
4	Ｌ，ｌ	文字列中の半角英字を小文字に変換する
5	Ｉ，ｉ	整数に変換する
6	Ｆ，ｆ，Ｄ，ｄ	２進倍精度浮動小数点に変換する
7	Ｘ，ｘ ［２文字目：Ｓ，ｓ］	１６進文字列に変換する。ａ～ｆは、ｘのときは小文字、Ｘのときは大文字になる。第一項が整数で、第二項２文字目がＳまたはｓのときは、ゼロサプレスする
8	Ｃ，ｃ，Ｐ，ｐ	文字列中の半角英単語の先頭文字を大文字に変換する Ｐ，ｐのときは、２文字目以降を小文字に変換する

4.6. 範囲式

式(第一番目が下限値)と式(第二番目が上限値)を範囲指定子(..)でつなげたものを範囲式という。
式の値には、文字と数値が指定できる。
下限値と上限値が共に文字のときは、先頭１バイトが、範囲として使われる。
下限値と上限値のどちらかが数値のときは、以下の順位で変換され、同じ属性に合わされる。
整数 --> １０進浮動(固定)小数点数 --> ２進倍精度浮動小数点数
以下の場所では、関数の引数および配列のインデックスを除き、増分値＝１で展開される。
~~・ＰＲＩＮＴのデータ~~
・ＦＯＲ ＥＡＣＨの対象データ
・配列への初期値設定式および代入式
展開されない場合は、範囲値（範囲式の評価結果）が使われる。
バリアント型または可変長文字型の変数に範囲式を代入したときは、範囲値として保存される。
変数の範囲値は、式の中では展開されない。

4.7. 式の並び

4.7.1. 式の並びについて

式(代入式を含む)をカンマで区切って並べたものを"式の並び"と言う。（関数名、クラス名の後ろを除く）
特に、式の並びを
①かっこ(())で囲ったものを、"データ並び式"
②中かっこ({})で囲ったものをデータ・リスト式
と言う。
一つのデータ・リスト式は、一つのデータ要素となり、一般の式として扱われる。

4.7.2. 式の並びの扱い

式を指定できるヶ所には、式の並びを指定できる。ただし、データ・リスト式を除き、そのか所によって扱いが異なる。
一般の式の途中にカンマがあるときは、カンマの前後は、別の式として扱われるため、一つの式の中で複数の式を処理させたいときは、データ並びを使用するとよい。

（１）ＦＯＲ ＥＡＣＨの対象データ
①式の並び　　：各式の値が、対象要素となる。
②データ並び式　　：かっこの入れ子状態に係わらず、各式の値が、対象要素となる。
③データ・リスト(式)：第一レベルのリスト要素が、対象要素となる。

（２）ＦＯＲ （；；）の各部分
（Ａ）初期設定部と増分部
①式の並び　　：各式が全て前から順に評価される。
②データ並び式　　：同上。
③データ・リスト(式)：同上。
（Ｂ）条件部
①式の並び　　：各式が全て前から順に評価され、最後の式の値が条件に使われる。
②データ並び式　　：同上。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが条件に使われる。

（３）配列の初期値設定式および代入式
①式の並び　　：各式が全て前から順に評価され、式の値が前から順に配列要素に代入される。
②データ並び式　　：同上。
③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが最初の要素に代入される。

（4）代入式（=）

①式の並び　　：カンマで区切られた各代入式が全て前から順に評価される。

②データ並び式　　：左辺の場合：各式が全て前から順に評価され、各評価結果の全てに、右辺の値が代入される。

右辺の場合：各式が全て前から順に評価され、左辺に代入される

・左辺がデータ並び式の場合は、左右の先頭から順に対応する要素間で代入される。右辺の要素が足りないときは、最後の要素が代入される。

・左辺がデータ並び式でない場合は、最後の要素が代入される。

③データ・リスト(式)：左辺の場合：指定できない。

右辺の場合：各式が全て前から順に評価され、左辺に代入される。

（5）その他

①式の並び　　：各式が全て前から順に評価され、最後の式の値が使われる。

②データ並び式　　：同上。

③データ・リスト(式)：各式が全て前から順に評価され、データ・リストが使われる。

4.8. ドット式

4.8.1. システム関数の実行

式に付けてシステム関数を実行する。
式には、データ値、配列名を結果とする式を指定可能。

```
式 . 関数名([引数リスト])
```

（注）式を第一パラメータ、引数リストを第二パラメータ以降とする関数式と同じ。
静的クラス名に"System"を指定する方法でも実行可能。

4.8.2. ユーザ定義関数の実行

関数へのパスを付けてユーザ定義関数を実行する。
手続きまたは関数をつなげて、実行する関数名へのパスを指定する。

```
関数へのパス . 関数名([引数リスト])
```

（注）手続きまたは関数の入れ子をサポートするオプションを指定したときのみ指定可能。

4.8.3. クラス関数の実行

インスタンス名または静的クラス名を付けて、実行する関数名へのパスを指定する。

```
{ インスタンス名 | 静的クラス名 } . 関数名([引数リスト])
```

（注）システム関数は、静的クラス名に"System"を指定する。

4.8.4. クラス変数へのアクセス

インスタンス名または静的クラス名を付けて、クラス変数へのパスを指定する。

```
{ インスタンス名 | 静的クラス名 } . クラス変数名
```

4.8.5. 構造体メンバ要素へのアクセス

構造体変数を参照。

4.8.6. 文字列の切り出し

指定された定数または変数値から切り出し指定の文字列を切り出す。

```
{ 一般データ変数 | 定数 } . 切り出し指定
```

(注)一般データ変数値は、文字列に変換される。
切り出し指定は、全体を文字定数または変数で使用し、内容は以下を指定できる。

```
{ 切り出し開始位置
  切り出し開始位置. . 切り出し終了位置
  切り出し開始位置, 切り出し文字数 }
```

(注)切り出し規則は、ＳＵＢＳＴＲと同じ。

4.9. 集合演算

配列またはデータ・リストに対して以下の集合演算を行うことができる。
配列は、配列名の前にアスタリスク(*)を付けて指定する。
データ・リストまたは一般変数は、変数名または定数をそのまま指定する。
第一項または第二項には、式を指定することができる。

表4.7-1 演算可能なデータ属性の組み合わせ

		第二項			
		一般配列	連想配列	データ・リスト	一般変数
第一項	一般配列	一般配列	一般配列	×	一般配列
	連想配列	一般配列	連想配列(キー比較)	×	一般配列
	データ・リスト	×	×	データ・リスト	データ・リスト
	一般変数	一般配列	一般配列	データ・リスト	×

(注)各交点の×以外が可能な組み合わせであり、演算結果の属性を示す。

表4.7-2 演算の種類

No.	演算子	機能
1	＋	第一項のデータ要素と第二項のデータ要素を連結する
2	－	第一項のデータ要素から第二項にあるデータ要素のみを削除する
3	&	第一項と第二項の両方にあるデータ要素を連結する
4	|	第一項のデータ要素と第二項のデータ要素を重複がないように連結する
5	^	第一項と第二項の共通でないデータ要素を連結する

(注)データ・リストの場合は、第一レベルのデータが比較される。
第一レベルのデータがデータ・リストのときは、その定義情報のみが比較されるため、全て異なるデータとなる。

表4.7-3 演算結果

No.	結果のデータ型	演算	要素のインデックス	要素の順序	配列のサイズ
1	一般配列	＋	保存される	保存される	nm1 + nm2
		－	保存される	保存される	max(nm1, nm2)
		&	前詰め	保存される	max(nm1, nm2)
		|	前詰め	保存される	nm1 + nm2
		^	前詰め	保存される	nm1 + nm2
2	連想配列	全て	－	－	－
3	データ・リスト	全て	前詰め	保存される	－

(注)nm1,nm2は、それぞれの項の配列定義サイズ。連想配列のときは、拡張済みエントリ数。
連想配列において同じキーのデータが残る場合は、第一項のデータが残される。

4.10. データ指定単項演算子

配列名、構造体名または範囲値に付け、データ指定属性を付与する。付与されたデータ要素は、それが使用される場所でデータ要素に対する操作がデータ要素属性により変わる。

No	操作	配列名	構造体名	範囲値
1	代入	データ実体がコピーされ、代入される。	データ実体がコピーされ、代入される。	データ指定属性付で、そのまま代入される。
2	PRINT	配列要素の内容が出力される。	構造体要素の内容が出力される。	下限値から上限値を超えない値まで１づつ増えながら出力される。
3	FOR EACH	配列要素の内容が対象データとなる。	使用できない。	下限値から上限値を超えない値まで１づつ増えながら要素データとなる。
4	集合演算	配列要素の内容が演算対象となる。	使用できない。	集合演算とならない。

4.11. インスタンス生成式

クラス名で指定したクラスのインスタンスを生成する。
クラス名の後ろには、実行したいコンストラクタと同じ数のパラメータを指定する。
機能は、ＮＥＷ関数と同じ。

5. 変数

変数には、パラメータ変数、検索結果読み込み変数、内部変数、外部変数、システム変数がある。
システム変数を除く変数には、表７－１のデータを格納することができる。
外部変数とシステム変数を除く変数は、スクリプト内でのみ有効である。

5.1. パラメータ変数

・手続き、または、関数に引き渡されたデータを扱う。
・パーセント記号（%）で始まる整数で表す。
・呼出し側で指定されたパラメータの順に、％１から順に対応する。
・％０は、パラメータの個数を示す。
・未定義変数値は、ＮＵＬＬ値となる。
・マップド配列は、パーセント記号（%）で始まる名称文字列で表す。

5.2. 検索結果読み込み変数

・ＲＥＡＤコマンド実行時に検索結果が格納される。
・シャープ記号（#）で始まる整数で表す。
・タプル単位にｓｅｌｅｃｔ項目の順に、＃１から順に対応する。
・＃０は、検索カラム数を示す。（$ＣＯＬＩＭＮと同じ）
・マップド配列は、シャープ記号（#）で始まる名称文字列で表す。

5.3. 内部変数

スクリプトの内部のデータを扱う。

（1）内部番号変数
・ドル記号（$）で始まる整数で表す。
・＄１から始まる。

（2）内部名称変数
・ドル記号（$）で始まる名称文字列で表す。ドル記号は省略できる。
・スカラー変数、配列変数、マップド配列変数がある。
・システム変数と同じ名称は使用できない。
・スクリプト内部(PRIVATE)変数と手続き内部(LOCAL)変数がある。
LOCAL変数は、手続き内でのみ有効である。
PRIVATE変数は、スクリプト内でのみ有効である。
デフォルトの変数種別は、LOCALである。デフォルトはオプションで変更できる。

5.4. 外部変数

・スクリプト間で共有されるデータを扱う。セッション内で有効である。
・ドル記号（$）で始まる名称文字列で表す。ドル記号は省略できる。
・スカラー変数と配列変数がある。
・システム変数と同じ名称は使用できない。

表５－２　定義済み外部変数一覧

項番	変　数　名	内　　容	データ型	初期値
1	$MAX_LOOP_WHILE	LOOP WHILEの最大ループ回数	整数	100000
2	$TRNSHID	Start Transactionを発行したホストのホストＩＤが規定バイト単位で連続して格納される。（重複はなし）	文字	ＮＵＬＬ値

5.5. システム変数

ドル記号（$）で始まる英大文字で表す。ドル記号は省略できる。

表５－３　システム変数一覧

項番	変　数　名	内　　容	データ型
１	$ＥＲＲＯＲ	直前に実行されたコマンドのリターン値	整数
２	$ＥＲＲＭＳＧ	直前に実行されたコマンドのエラーメッセージ	文字
３	$ＥＲＲＮＯ	システムエラー時のエラー番号	整数
４	$ＳＴＲＥＲＲＯＲ	上記のエラーメッセージ	文字
５	$ＴＵＰＬＥ	検索タプル数	整数
６	$ＣＯＬＵＭＮ	検索カラム数	整数
７	$ＵＳＥＲＩＤ	コマンドヘッダのユーザーコード	整数
８	$ＤＡＴＥ	ＹＹＹＹＭＭＤＤ	文字
９	$ＴＩＭＥ	ＨＨＭＩＳＳ	文字
１０	$ＤＡＴＥＴＩＭＥ	YYYY/MM/DD HH:MI:SS	文字
１１	$ＨＩＤ	自ホストＩＤ	文字
１２	$ＵＳＥＲＩＮＦ	コマンドヘッダのユーザ情報 （内容はインタフェース仕様書「３．２ 端末－ＷＳ間インタフェース」を参照）	文字
１３	$ＶＥＲＳＩＯＮ	本モジュールのバージョン番号が格納される。 書式は以下の通り。 Ｖ／Ｒ＝バージョン番号 例）Ｖ／Ｒ＝３．１	文字
１４	$ＭＡＫＥＤＡＴＥ	本モジュールの作成日が格納される。 書式は以下の通り。 ＤＡＴＥ＝作成日 例）DATE=Wed Jan 18 14:27:32 JST 1995	文字
１５	$ＳＴＤＩＮ	標準入力へのファイルポインタ	整数
１６	$ＳＴＤＯＵＴ	標準出力へのファイルポインタ	整数
１７	$ＳＴＤＥＲＲ	標準エラー出力へのファイルポインタ	整数
１８	$ＳＣＲＩＰＴＮＡＭＥ	実行スクリプト名	文字
１９	$ＰＲＯＣＮＡＭＥ	実行手続き名または関数名	文字
２０	$ＳＣＲＩＰＴＬＩＮＥ	実行スクリプト行数	整数
２１	$**ＣＨＡ**Ｒ	文字属性値　（１）	整数
２２	$**ＢＩＮ**ＡＲＹ	整数属性値　（２）	整数
２３	$**ＦＬＯ**ＡＴ($FLT)	倍精度浮動小数点属性値　（３）	整数
２４	$**ＤＥＣ**ＩＭＡＬ	ＤＥＣＩＭＡＬ属性値　（４）	整数
２５	$**ＢＵＬ**Ｋ	ＢＵＬＫ属性値　（５）	整数
２６	$**ＰＯＩ**ＮＴＥＲ	ポインタ属性値　（６）	整数
２７	$**ＶＡＲ**ＩＡＮＴ	バリアント属性値　（７）	整数
２８	$**ＩＮＴ**ＥＧＥＲ	整数属性値　（２）	整数
２９	$**ＤＯＵ**ＢＬＥ($DBL)	倍精度浮動小数点属性値　（３）	整数
３０	$ＮＵＬＬ	NULL	文字
３１	$ＰＩ	円周率π	浮動小数
３２	$Ｍ＿Ｅ	ｅ	浮動小数

5.6. 変数の配列

（１）番号変数

パラメータ変数、検索結果読み込み変数、内部番号変数は、その種類毎に配列として定義されており、変数種類を指す文字（%、＃、$）の後に付ける整数文字列がインデックスを示している。

このインデックス部分を変数で指定することによって変数を間接的にアクセスすることができる。この指定はネストすることができる。

インデックス部分は、カッコ（()）で囲むことができる。これにより、ＳＱＬ文中の連続する文字列の一部に変数を指定することが可能となる。

変数種類を指す文字（%、＃、$）の後にカッコ（()）を付けたものは、それぞれの変数の先頭からマップしたマップド配列変数名として使うことができる。

~~/＊ 配列を指定するときは、カッコ（()）を使用してネストさせる。＊/~~

＜例１＞

```
＄１　＝　２；
＄２　＝　’ＡＢＣ’；
＄３　＝　４；
＄＄３＝　＄＄１；
```

この結果＄４には’ＡＢＣ’が入る。

＜例２＞

```
＄１　＝　２；
＄２　＝　＄（１）；
```

＄２には、２がはいる。

＜例３＞

例１の４行目は、次の様にも書ける。

```
＄（＄（３））　＝　＄（＄（１））；
```

＜例４＞

```
＄１　＝　’Ａ’；
＄２　＝　２；

ＳＱＬ　”” 　”” 　”ｓｅｌｅｃｔ　＊　ｆｒｏｍ　＄（１）ＢＣ
　　ｗｈｅｒｅ　ａ＄（２）１　＝　’ｘｙ’””；
```

以下となる

```
ＳＱＬ　”” 　”” 　”ｓｅｌｅｃｔ　＊　ｆｒｏｍ　ＡＢＣ
　　ｗｈｅｒｅ　ａ２１　＝　’ｘｙ’””；
```

＜例５＞

インデックス部分に配列を使用する。

```
$a[0] = 1;
$($a[0]) = 'A';
$2 = 2;

SQL ""  ""  ""select * from $($a[0])BC where a$(2)1 = 'xy';
```

（2）マップド配列変数
内部番号変数のあるインデックス以降にマップした変数である。
定義方法は、DEFINE MAPPEDARRAY コマンドを参照。

（3）配列変数
一般的な配列を持つ変数である。
定義方法は、DEFINE［GLOBAL］ARRAY コマンドを参照。

（4）インデックスの指定方法
配列のインデックスは、かぎ括弧でくくって指定する。
次元は、カンマ（,）で区切って指定する。
インデックスには、式を指定可能。
（A）一般の配列のインデックス
数値で指定する。インデックスの開始値は、０である。
インデックス開始値は、オプション１５の指定により１となる。
定義した次元より大きい次元を指定できるが、指定できる値は、開始値である。
省略した次元は、インデックスに開始値が指定されたものと見なす。
（B）連想配列のインデックス
文字で指定する。
省略した次元は、インデックスにnull値が指定されたものと見なす。

（5）連想配列
インデックスに文字列を指定できる配列である。
インデックスに数値が指定されたときは、文字列に変換される。
次元数に制限はなく、任意の次元を使用可能である。
指定されたインデックスは、全てがバッククォート＋カンマ(`,)で連結され、ハッシュで管理される。
インデックス中に、バッククォート＋カンマ(`,)またはバッククォート＋バッククォート(``)以外でバッククォート(`)があるときは、その前にバッククォート(`)が付加される。
インデックス中のバッククォート＋カンマ(`,)は、インデックスを区切るカンマ(,)と同値となる。
未設定の配列要素を参照すると、ＮＵＬＬパラメータが設定される。(オプションで変わる)
［例］

```
define array $成績 hash;

$成績['小林明人','１学期','数学'] = 100;
```

5.7. 構造体変数

構造体型を指定して定義した変数を構造体と言う。
構造体のメンバ変数には、全てのデータ型を格納できる。
構造体のメンバ変数は、以下の形式でアクセスする。

```
構造体変数名.メンバのデータ要素[.メンバのデータ要素 ・・・]
```

メンバのデータ要素の内容が構造体変数名のときは、さらに、その構造体変数のメンバを指定できる。

6. 定数

定数には、文字列定数と数値定数とがある。

6.1. 文字列定数

１重引用符（’）で囲まれた文字列を文字列定数とする。
文字列が空の定数（’’）はＮＵＬＬ値を示す。

（１）旧仕様

文字列中に１重引用符を入れるときは、そのまま指定する。
文字列中に空白文字を入れるときは、１重引用符で囲まれた文字列を更に２重引用符で囲む。
＜例＞
① ’ａｂｃ’ｄ’は、ａｂｃ’ｄとなる。
② ”’ａｂ　ｄ’”は、ａｂ　ｄとなる

（２）新仕様

文字列中に１重引用符を入れるときは、２個続けて指定する。
文字列中に空白文字を入れるときは、そのまま指定する。
＜例＞
① ’ａｂｃ’’ｄ’は、ａｂｃ’ｄとなる。
② ’ａｂ　ｄ’は、ａｂ　ｄとなる。

（３）エスケープ処理

エスケープ文字の次の文字を定数文字とする。
ただし、文字列がエスケープ文字の１文字の場合、または、エスケープ処理の結果、末尾にエスケープ文字が残った場合は、エスケープ文字をそのまま定数文字とする。
また、特殊文字は、以下の形式で指定する。

No	指定形式	コード	名称
1	’¥n’	0x0a	ＬＦ
2	’¥t’	0x09	ＨＴ　タブ
3	’¥r’	0x0d	ＣＲ
4	’¥b’	0x08	ＢＳ　バックスペース
5	’¥f’	0x0c	ＦＦ　フォームフィード
6	’¥DDD’	Ｄは０～７	８進数
7	’¥xXX’	Ｘは０～９、Ａ～Ｆ、ａ～ｆ	１６進数

（４）空白文字

半角スペースまたはタブを空白文字と言う。

6.2. 数値定数

（１）整数値定数

２進、８進、１０進、１６進がある。

指定形式は、

［０　型指定文字］　数字列

型指定文字は、

Ｂ又はｂは	２進
Ｏ又はｏは	８進
~~Ｄ又はｄは~~	~~１０進~~
Ｘ又はｘは	１６進

１０進のときは、符号を含む数字列のみとする。

＜例＞

以下はすべて整数の１０６を表す。

２進	0b01101010
８進	0o152
~~１０進~~	~~0d106~~
１６進	0x6a

（２）２進倍精度浮動小数点定数（内部的に２進数で保持）

以下の形式で指定する。

少数点のみの指定はできない。

｛　［＋｜－］［整数］［．［整数］］｛Ｆ｜ｆ｜Ｄ｜ｄ｝
　［＋｜－］［整数］［．［整数］］｛｛Ｅ｜ｅ｜Ｄ｜ｄ｝［＋｜－］整数］　｝

（３）１０進浮動小数点定数（内部的に１０進数で保持）

以下の形式で指定する。

指数部の″＋－″は、数字列が文字列定数の時のみ有効となる。

少数点のみの指定はできない。

｛　［＋｜－］［整数］［．［整数］］
　［＋｜－］［整数］［．［整数］］｛｛＋｜－｝整数｝　｝

7. データ属性

7.1. データ属性一覧

使用できるデータの属性を表７－１に示す。

表７－１　データ属性

項番	データＩＤ	データ型	内　容
１	一般データ	文字	０ｘ０１～０ｘＦＦのコードを収容可能。ただし、スクリプト内においては非表示文字コードは設定できない。データ長は０～（４バイト整数の最大値）。データ長ゼロは、特別にＮＵＬＬ値として扱われる。
２	一般データ	整数	４バイトの整数値
３	一般データ	２進倍精度浮動小数点数	２進倍精度浮動小数点数値（内部的に２進保持）
４	一般データ	１０進小数点数	１０進小数点数値（内部的に１０進保持）。 精度は４２桁。１０の－３２７０１乗～３２７００乗。
５	一般データ	ｂｕｌｋ	０ｘ００～０ｘＦＦのコードを収容可能。データ長は１～（４バイト整数の最大値）。データ長０は文字型となる。
６	配列名	ｂｕｌｋ	配列定義した配列の名前
７	リスト	ｂｕｌｋ	データ・リスト
８	構造体名	ｂｕｌｋ	構造体の情報
９	構造体定義名	ｂｕｌｋ	構造体の定義情報
１０	関数名	文字	関数の情報
１１	クラス名	文字	クラスの情報
１２	インスタンス名	文字	インスタンスの情報
１３	メソッド名	文字	メソッドの情報
１４	手続き名	文字	手続きの情報

データＩＤ、データ型およびデータ長の値は、付録を参照。

7.2. データ・リスト

（１）データ・リストには、全てのデータ型を任意の組み合わせで格納できる。
　　データ・リストも格納できる。

（２）以下の変数には、データ・リスト作成関数、または、データ・リスト式を使用して、
　　データ・リストを設定できる。
　　・内部変数
　　・外部変数
　　ただし、データ型指定の配列の要素を除く。

8. 組み込み関数

8.1. 関数一覧

表8.1-1 関数一覧（文字列関連）

No	関数名	機能	OP	戻り値	属性
1	ＩＳ	文字列の調査	○	調査結果	数値
2	ＴＯ	文字列の変換	○	変換結果	注１
3	ＲＥＰ	文字列の置換	○	置換結果	文字
4	ＣＯＮＤＡＳ	文字列の条件式	○	条件式の実行結果	文字
5	ＳＵＢＳＴＲ ＭＩＤ	文字列を切出す	○	切出し文字列	文字
6	ＣＯＮＣＡＴ	文字列を連結する	○	連結文字列	文字
7	ＧＥＴＡＲＧＳ	文字列を指定文字で分解する	○	分解数	数値
8	ＴＲＩＭ	文字列の前後の空白文字を削除する	○	削除済み文字列	文字
9	ＳＴＲＩＮＧＳ	文字列を繰り返す	○	繰り返し文字列	文字
10	ＬＥＦＴ	文字列の左側を取り出す	○	切出し文字列	文字
11	ＲＩＧＨＴ	文字列の右側を取り出す	○	切出し文字列	文字
12	ＥＤＩＴ	データのフォーマッティング	○	編集結果	文字
13	ＲＥＰＬＩＫＥ	文字列のＬＩＫＥ置換	○	置換結果	文字
14	ＬＥＮＧ	データの長さまたは文字数を返す		長さまたは文字数	数値
15	ＬＥＮＢ	バイト単位で長さを返す		バイト長	文字
16	ＳＵＢＳＴＲＢ ＭＩＤＢ	バイト単位で文字列を切出す	○	切出し文字列	文字
17	ＬＥＦＴＢ	バイト単位で文字列の左側を取り出す	○	切出し文字列	文字
18	ＲＩＧＨＴＢ	バイト単位で文字列の右側を取り出す	○	切出し文字列	文字
19	ＧＥＴＷＤ	ワード取り出し		取出し文字列	文字

（注１）変換数値の属性。（注）OP欄の"○"は演算子でも使用可を示す。

表8.1-1 関数一覧（演算関連　1/3）

No	関数名	機能	OP	戻り値	属性
1	ＡＳＣ	文字をコードに変換する	○	文字コード	数値
2	ＣＨＲ	コードを文字型に変換する	○	変換文字列	文字
3	ＴＯ＿ＢＵＬＫ	各型のデータをそのままバルク型に変換し、バイト単位で切り出す。	○	バルク・データ	BULK
4	ＴＯ＿ＢＵＬＫＳ	各型のデータをそのままバルク型に変換し、連結する。	○	バルク・データ	BULK
5	ＴＯ＿ＮＵＭＢＥＲ	数値への変換	○	変換数値	注１
6	ＡＢＳ	絶対値を求める		絶対値	数値
7	ＭＡＸ	最大値を求める	○	最大値	注２
8	ＭＩＮ	最小値を求める	○	最小値	注２
9	ＭＯＤ	余りを求める	○	余り	数値

（注２）第一項のデータ要素の属性。（注）OP欄の"○"は演算子でも使用可を示す。

表8.1-1 関数一覧（演算関連　2/3）

No	関数名	機能	OP	戻り値	属性
10	ＦＦ	関数名と同じ文字列を関数指定扱いにする		関数名	関数
11	ＥＶＡＬ	式を評価する		式の値	注３
12	ＮＶＡＬ	ＮＵＬＬ値を置き換える		式の値	注３
13	ＮＤＥＦ	未定義値を置き換える		式の値	注３
14	ＬＩＫＥ	パターン・マッチング	○	マッチング結果	数値
15	ｉＬＩＫＥ	大文字、小文字を区別しないパターン・マッチング	○	マッチング結果	数値
16	ＲＥＧＥＸ	正規表現パターン・マッチング	○	マッチング結果	数値
17	ｉＲＥＧＥＸ	大文字、小文字を区別しない正規表現パターン・マッチング	○	マッチング結果	数値
18	ＩＮ	文字列比較	○	比較結果	数値
19	ｉＩＮ	大文字、小文字を区別しない文字列比較	○	比較結果	数値
20	ＩＮＳＴＲ	文字列サーチ	○	文字列の位置	数値
21	ＩＮｉＳＴＲ	大文字、小文字を区別しない文字列サーチ	○	文字列の位置	数値
22	ＩＮＲＳＴＲ	後ろからの文字列サーチ	○	文字列の位置	数値
23	ＩＮｉＲＳＴＲ	後ろからの大文字、小文字を区別しない文字列サーチ	○	文字列の位置	数値
24	ＩＮＬＩＫＥ	文字列のＬＩＫＥ一致ヶ所数と位置を返す。	○	一致ヶ所数	数値
25	ＸＨＡＳＨ	ハッシュ処理		ハッシュ表へのポインタ、または、ハッシュ・インデックス	数値
26	ＳＥＴＡＲＲＡＹ	配列に値を設定する		設定数	数値
27	ＡＲＲＡＹＣＰＹ	配列をコピーする		コピー数	数値
28	ＡＲＲＡＹＣＬＲ	配列をクリアする		クリア数	数値
29	ＡＲＲＡＹＣＭＰ	配列どうしを比較する		比較結果	数値
30	ＡＲＲＡＹＢＸＰ	配列どうしを演算する		演算数	数値
31	ＣＯＵＮＴＶ	配列またはデータ・リストの要素数		定義済み要素数	数値
32	ＩＮＤＥＸＡ	配列の１次元要素位置を求める		１次元要素位置	数値
33	ＦＬ	データ・リストを操作する		データ・リスト	注３
34	ＬＩＳＴ	データ・リストを作成する		データ・リスト	注３
35	ＦＩＲＳＴ	データ・リストの最初の要素を参照する		データ・リスト	注３
36	ＲＥＳＴ	データ・リストの最初の要素を除くリスト参照する		データ・リスト	注３
37	ＣＯＮＳ	データ・リストを連結する		データ・リスト	注３
38	ＬＩＳＴ＿ＲＥＦ	データ・リストの指定番目の要素を参照する。		データ・リスト	注３
39	ＳＯＲＴ	配列等のデータをソートする		データ個数	数値

（注３）データ要素の属性

表8.1-1 関数一覧（演算関連　3/3）

No	関数名	機能	OP	戻り値	属性
40	ＲＯＵＮＤ	数値を丸める		丸め結果	注３
41	ＮＥＷ	インスタンスを生成する		インスタンス	
42	ＡＲＲＡＹＭＡＰ	指定インデックス位置にマップする配列を作成する。		配列	
43	ＣＣＨＡＲ、ＣＳＴＲ[ＩＮＧ]	文字型に変換する		変換データ	注３
44	ＣＩＮＴ、ＣＢＩＮ	整数型に変換する		変換データ	注３
45	ＣＦＬＯＡＴ、ＣＤＯＵＢＬＥ、ＣＦＬＴ、ＣＤＢＬ	倍精度２進浮動小数点型に変換する		変換データ	注３
46	ＣＤＥＣ[ＩＭＡＬ]	１０進小数点型に変換する		変換データ	注３
47	ＣＢＵＬＫ	ＢＵＬＫ型に変換する		変換データ	注３
48	ＣＶＡＲＩ[ＡＮＴ]	バリアント型に変換する		変換データ	
49	ＳＫＩＰ＿ＯＰＴ	文字列をスキップする	○	文字列の位置	数値

表8.1-1 関数一覧（ファイル関連）

No	関数名	機能	戻り値	属性
1	ＦＯＰＥＮ	ファイルをオープンする	ファイル・ポインタ	数値
2	ＦＣＬＯＳＥ	ファイルをクローズする	リターン・コード	数値
3	ＦＧＥＴＬＩＮＥ	ファイルから１行読み込む	１行文字列	文字
4	ＦＰＵＴＬＩＮＥ	ファイルに複数個の文字列を書き込む	文字列長の合計	数値
5	ＦＥＬＲＥＡＤ１	ファイルから１データ要素を読み込む	データ要素	注２
6	ＦＥＬＷＲＩＴＥ	ファイルに複数個のデータ要素を書き込む	出力データ長の合計	数値
7	ＦＥＬＷＲＩＴＥ１	ファイルに１データ要素を書き込む	出力データ長	数値
8	ＵＮＬＩＮＫ	ファイルを削除する	リターン・コード	数値
9	ＧＥＴＬＩＮＥ	標準入力から１行読み込む	１行文字列	文字
10	ＰＵＴＬＩＮＥ	標準出力に複数個の文字列を書き込む	文字列長の合計	数値
11	ＥＬＲＥＡＤ１	標準入力から１データ要素を読み込む	データ要素	注２
12	ＥＬＷＲＩＴＥ	標準出力に複数個のデータ要素を書き込む	出力データ長の合計	数値
13	ＥＬＷＲＩＴＥ１	標準出力に１データ要素を書き込む	出力データ長	数値
14	ＯＰＥＮＤＩＲ	ディレクトリをオープンする	ディレクトリ・ポインタ	数値
15	ＲＥＡＤＤＩＲ	ディレクトリ内を読む	ファイル名	文字
16	ＣＬＯＳＥＤＩＲ	ディレクトリをクローズする	リターン・コード	数値
17	ＳＴＡＴ	ファイル属性を取得する	リターン・コード	数値
18	ＦＰＳＴＡＴ	ファイル属性を取得する	リターン・コード	数値
19	ＬＰＲＩＮＴ	ＬＰＲＩＮＴコマンドと同様	リターン・コード	数値
20	ＰＲＩＮＴ	ＰＲＩＮＴコマンドと同様	リターン・コード	数値
21	ＥＣＨＯ	ＥＣＨＯコマンドと同様	リターン・コード	数値
22	ＳＡＹ	ＥＣＨＯと同じ	リターン・コード	数値

表8.1-1 関数一覧（シェル実行関連）

No	関数名	機能	戻り値	属性
1	ＳＨＥＬＬ	システムのコマンド・ラインを実行する	リターン・コード	数値
2	ＰＯＰＥＮ	パイプをオープンしてコマンド・ラインを実行する	ファイル・ポインタ	数値
3	ＰＣＬＯＳＥ	パイプをクローズする	リターン・コード	数値

表8.1-1 関数一覧（ログ関連）

No	関数名	機能	戻り値	属性
1	ＳＹＳＬＯＧ	syslogにログを出力する	リターン・コード	数値
2	ＬＯＧＯＵＴ	ログを出力する	リターン・コード	数値
3	ＧＥＴＬＯＧＰＡＲＭ	ログパラメータを取得する	取得項目数	数値
4	ＳＥＴＬＯＧＰＡＲＭ	ログパラメータを設定する	設定項目数	数値
5	ＲＥＳＬＯＧＰＡＲＭ	取得済ログパラメータをリストアする	設定項目数	数値

表8.1-1 関数一覧（メモリ管理関連）

No	関数名	機能	戻り値	属性
1	ＮＯＦＲＥＥ	未開放メモリのアドレスをファイルに出力する	未開放メモリ数	数値
2	ＧＥＴＭＥＭＵＳＥＤ	各種のメモリ使用量を取得する	メモリ使用量	数値

表8.1-1 関数一覧（実行制御関連）

No	関数名	機能	戻り値	属性
1	ＳＨＵＴＣＴＬ	ｓｈｕｔを制御する	リターン・コード	数値
2	ＥＸＩＴ	スクリプトの実行を終了する	リターン・コード	数値
3	ＳＥＴＥＮＶ	環境変数の値を設定する	リターン・コード	数値
4	ＵＮＳＥＴＥＮＶ	環境変数を削除する	リターン・コード	数値
5	ＰＵＴＥＮＶ	環境変数の値を設定する(name=value形式)	リターン・コード	数値
6	ＧＥＴＥＮＶ	環境変数の値を取得する	環境変数の値	文字
7	ＧＥＴＴＩＭＥ	セッション開始からの経過時間を返す	経過時間(ミリ秒まで)	数値
8	ＴＩＭＥＳ	関数を指定回数繰り返す	関数の戻り値	
9	ＧＥＴＶＡＬ	各種設定値の値を取得する	取得した値	

表8.1-1 関数一覧（通信関連）

No	関数名	機能	戻り値	属性
1	ＣＨＡＮＮＥＬ	チャネルを制御する	リターン・コード	数値

表8.1-1 関数一覧（数値計算関連）

No	関数名	機能	戻り値	属性
1	ＳＱＲＴ	平方根	関数値	２進浮動小数
2	ＳＩＮ	sin(x)	関数値	２進浮動小数
3	ＣＯＳ	cos(x)	関数値	２進浮動小数
4	ＴＡＮ	tan(x)	関数値	２進浮動小数
5	ＡＴＡＮ	atan(x)	関数値	２進浮動小数
6	ＬＯＧ	自然対数	関数値	２進浮動小数
7	ＬＯＧ１０	１０の対数	関数値	２進浮動小数
8	ＥＸＰ	eのx乗	関数値	２進浮動小数
9	ＰＯＷＥＲ	xのy乗	関数値	２進浮動小数
10	ＳＲＡＮＤ１ ＳＲＡＮＤ４８	乱数のSEED(種)を設定する	NULL値	文字
11	ＲＡＮＤ１ ＤＲＡＮＤ４８	0.0<=、<1.0の間の乱数を求める	関数値	２進浮動小数
12	ＣＢＲＴ	立方根	関数値	２進浮動小数
13	ＳＩＮＨ	sinh(x)	関数値	２進浮動小数
14	ＣＯＳＨ	cosh(x)	関数値	２進浮動小数
15	ＴＡＮＨ	tanh(x)	関数値	２進浮動小数
16	ＡＴＡＮＨ	atanh(x)	関数値	２進浮動小数
17	ＣＥＩＬ	ceil(x)	関数値	２進浮動小数
18	ＦＬＯＯＲ	floor(x)	関数値	２進浮動小数
19	ＲＩＮＴ	rint(x)	関数値	２進浮動小数

8.2. 全般規則

（１）関数の引数には、式を指定できる。

8.3. ＡＢＳ

(1)機能

絶対値を求める。

(2)一般形式

```
result = ABS(var) ;
```

(3)構文規則

なし

(4)一般規則

(A)varが数値でないときは、数値に変換される。

(B)varまたは変換結果が、整数のときは、整数の絶対値を、２進浮動小数点のときは、２進浮動小数点の絶対値を、１０進小数点のときは、１０進小数点の絶対値求める。

8.4. ＡＲＲＡＹＣＰＹ

(1)機能

配列をコピーする。

(2)一般形式

```
n  = ARRAYCPY(array_to, [start_to], array_from, [start_from], [num]) ;
```

(3)構文規則

(A)array_toまたはarray_fromには、配列名または数値を指定する。

(B)array_toまたはarray_fromに文字定数が指定されたときは、数値に変換される。

(C)array_toには、連想配列は指定できない。

(D)配列が連想配列のときは、両配列共に連想配列でなければならない。

(E)配列が連想配列のときは、startは指定できない。

(4)一般規則

(A)array_toまたはarray_fromに配列名を指定したときは、配列名[start]の要素から最大で、配列のサイズまたはnum個までコピーされる。

(B)array_toまたはarray_fromに数値が指定されたときは、内部番号変数の$(指定数値+start)の要素から、最大で配列のサイズまたはnum個までコピーされる。

(C)コピー元要素が未設定のときは、コピー先要素も未設定となる。

(D)返却値nには、コピーされた個数が返る。

(E)配列が連想配列のときは、定義済みキーのデータがコピーされる。

numは定義済み要素が対象となる。

8.5. ＡＲＲＡＹＣＬＲ

(1)機能

配列をクリアする。

(2)一般形式

```
n  = ARRAYCLR(array, [start], [var], [num]) ;
```

(3)構文規則

(A)arrayには、配列名または数値を指定する。
(B)arrayに文字定数が指定されたときは、数値に変換される。
(C)配列が連想配列名のときは、start、var、numは指定できない。

(4)一般規則

(A)arrayに配列名を指定したときは、配列名[start]の要素から最大で、配列のサイズまたはnum個までvarが格納される。
(B)arrayに数値が指定されたときは、内部番号変数の$(指定数値+start)の要素から、最大で配列のサイズまたはnum個までvarが格納される。
(C)返却値nには、実際に設定された個数が返る。
(D)配列が連想配列名のときは、要素数がゼロとなる
(E)デフォルト値は、start=0, var='', num=配列の最大サイズ。

8.6. ＡＲＲＡＹＣＭＰ

(1)機能

配列どうしを比較する。

(2)一般形式

```
n  = ARRAYCMP(array1, [start1], array2, [start2], [num], [cmp]) ;
```

(3)構文規則

(A)array1またはarray2には、配列名または数値を指定する。
(B)array1またはarray2に文字定数が指定されたときは、数値に変換される。
(C)配列が連想配列のときは、両方が連想配列でなければならない。このときは、start1とstart2は指定できない。

(4)一般規則

(A)array1またはarray2に配列名を指定したときは、配列名[start]の要素から最大で、配列のサイズまたはnum個まで比較される。
(B)array1またはarray2に数値が指定されたときは、内部番号変数の$(指定数値+start)の要素から、最大で配列のサイズまたはnum個まで比較される。
(C)比較は要素単位に行われ、偽となった時点で終了する。
(D)返却値nには、真の要素数が返る。
(E)cmpには、比較演算子を文字属性で指定する。省略時は、'='で比較する。
(F)配列が連想配列のときは、同じキーどうしを比較する。numは同じキーどうしが対象となる。

8.7. ＡＲＲＡＹＢＸＰ

(1)機能

配列どうしを演算する。array_to = array1 演算子 array2

(2)一般形式

```
n = ARRAYBXP(array_to, [start_to], array1, [start1], array2, [start2], [num], [bxp]) ;
```

(3)構文規則

(A)array_to、array1、array2には、配列名または数値を指定する。
(B)array_to、array1、array2に文字定数が指定されたときは、数値に変換される。
(C)array_toには、連想配列は指定できない。
(D)配列が連想配列のときは、３つの配列共に連想配列でなければならない。
(E)配列が連想配列のときは、startは指定できない。

(4)一般規則

(A)array_to、array1、array2に配列名を指定したときは、配列名[start]の要素から最大で、配列のサイズまたはnum個まで比較される。
(B)array_to、array1、array2に数値が指定されたときは、内部番号変数の$(指定数値+start)の要素から、最大で配列のサイズまたはnum個まで比較される。
(C)返却値nには、演算した要素数が返る。
(D)bxpには、演算子を文字属性で指定する。省略時は、'+'が指定されたものと見なす。
(F)配列が連想配列のときは、同じキーどうしを演算する。numは同じキーどうしが対象となる。

8.8. ＡＲＲＡＹＭＡＰ

(1)機能

指定インデックス位置を先頭とするマップド配列を返す。

(2)一般形式

```
array_m = ARRAYMAP(array, [i], [j], [k]) ;
```

(3)構文規則

(A)arrayには、配列名を指定する。
(B)i,j,kには、結果が数値となる式を指定する。

(4)一般規則

(A)i,j,kには、先頭としたい、各次元の位置を指定する。
(B)各次元の大きさ、または、範囲の上限は、指定位置からの相対的大きさに調整される。
なお、下限は変わらない。

8.9. ＡＳＣ

(1)機能

文字列の１文字目をアスキーコードに変換する

(2)一般形式

```
result  = ASC(var) ;
```

(3)構文規則

(4)一般規則

(A)文字は４バイト以下とする。文字コードを４バイト整数で表した値が返される。

(B)データが文字型以外のときは、文字型に変換された後でアスキーコードに変換される。

8.10. ＡＴＡＮ

(1)機能

atanを求める。

(2)一般形式

```
ret  = ATAN(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

(B)リターン値はラジアンである。

8.11. ＡＴＡＮＨ

(1)機能

atanhを求める。

(2)一般形式

```
ret  = ATANH(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

(B)リターン値はラジアンである。

8.12. ＣＢＩＮ

(1)機能

整数型に変換する。

(2)一般形式

```
ret  = CBIN(x) ;
```

(3)構文規則

なし。

(4)一般規則

なし。

8.13. ＣＢＲＴ

(1)機能

立方根を求める。

(2)一般形式

```
ret  = CBRT(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

8.14. ＣＢＵＬＫ

(1)機能

ＢＵＬＫ型に変換する。

(2)一般形式

```
ret  = CBULK(x [,size]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)sizeを指定すると固定長に、省略すると可変長となる。

8.15. ＣＣＨＡＲ

(1)機能

文字型に変換する。

(2)一般形式

```
ret  = CCHAR(x [,size]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)sizeを指定すると固定長に、省略すると可変長となる。

8.16. ＣＤＥＣ

(1)機能

１０進小数点型に変換する。

(2)一般形式

```
ret  = CDEC(x [,size [,scale]]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)sizeは精度を、scaleは位取りを示す。詳細は、変数のデータ型を参照。

8.17. ＣＤＯＵＢＬＥ（ＣＤＢＬ）

(1)機能

倍精度２進浮動小数点型に変換する。

(2)一般形式

```
ret  = CDOUBLE(x) ;
```

(3)構文規則

なし。

(4)一般規則

なし。

8.18. ＣＥＩＬ

(1)機能

を求める。

(2)一般形式

```
ret  = CEIL(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

8.19. ＣＦＬＯＡＴ（ＣＦＬＴ）

(1)機能

倍精度２進浮動小数点型に変換する。

(2)一般形式

```
ret  = CFLOAT(x) ;
```

(3)構文規則

なし。

(4)一般規則

なし。

8.20. ＣＨＡＮＮＥＬ

(1)機能

チャネルを制御する

(2)一般形式

```
cha = CHANNEL('open', Host, Service, HeadLen, Options, HeadCheck, Exception) ;

ret = CHANNEL('shut', cha, option) ;

ret = CHANNEL('resume', cha, option) ;

ret = CHANNEL('close', cha) ;
```

(3)構文規則

(4)一般規則

(A)

8.21. ＣＨＲ

(1)機能

コードを文字型に変換する。

(2)一般形式

```
result  = CHR(var1[, var2, ...]) ;
```

(3)構文規則

(4)一般規則

(A)コードは、整数値を0xFFでマスクした値が使われる。

(B)データが整数値以外のときは、整数値に変換された後で文字型に変換される。

(C)指定されたコードは、全てが１バイト文字として連結される。

8.22. ＣＩＮＴ

(1)機能

整数型に変換する。

(2)一般形式

```
ret  = CINT(x) ;
```

(3)構文規則

なし。

(4)一般規則

なし。

8.23. ＣＬＯＮＧ（ＣＬＮＧ）

(1)機能

整数型に変換する。

(2)一般形式

```
ret  = CLONG(x) ;
```

(3)構文規則

なし。

(4)一般規則

なし。

8.24. ＣＬＯＳＥＤＩＲ

(1)機能

ディレクトリをクローズする。

(2)一般形式

```
ret  = CLOSEDIR(dp) ;
```

(3)構文規則

(4)一般規則

(A)ディレクトリ以外は、クローズできない。

8.25. ＣＯＮＣＡＴ

（１）機能

文字列の連結

（２）一般形式

```
result = CONCAT(var, [v1], [v2], ･･･ ) ;
```

（３）構文規則

文字列演算式の第一項以降がそれぞれ第１パラメータ以降に対応する。

（４）一般規則

文字列演算子を参照。

8.26. ＣＯＮＤＡＳ

（１）機能

文字列の条件式

（２）一般形式

```
result = CONDAS(var [, start [, len]], condas) ;
```

（３）構文規則

(A)文字列演算式の第一項と第二項がそれぞれ第１パラメータと第２パラメータに対応する。

(B)start、lenについては、INSTR()と同じ。

（４）一般規則

文字列演算子を参照。

8.27. ＣＯＮＳ

(1)機能

データ・リストまたは要素を連結してリストを作る。

(2)一般形式

```
var = CONS([var1], [var2], ...) ;
```

(3)構文規則

ＦＬと同じ。

(4)一般規則

ＦＬと同じ。

8.28. ＣＯＳ

(1)機能

cosを求める。

(2)一般形式

```
ret  = COS(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.29. ＣＯＳＨ

(1)機能

coshを求める。

(2)一般形式

```
ret  = COSH(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.30. ＣＯＵＮＴＶ

(1)機能

スカラー、配列またはデータ・リストの定義済み要素数を数える。

(2)一般形式

```
n = COUNTV(var);
```

(3)構文規則

なし

(4)一般規則

(A)オプション番号１の影響は受けない。

(B)スカラー変数が未定義のときは、エラーとなる。定義済みのときは、要素数は、１となる。

8.31. ＣＳＴＲＩＮＧ（ＣＳＴＲ）

(1)機能

文字型に変換する。

(2)一般形式

```
ret  = CSTRING(x [,size]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)sizeを指定すると固定長に、省略すると可変長となる。

8.32. ＥＤＩＴ

(1)機能

変数の値を指定したフォーマットへ変換する。

(2)一般形式

```
result = EDIT(format [, var1, var2, var3 ・・・]) ;
```

(3)構文規則

(A)

(4)一般規則

(A)formatは、Ｃ言語のprintfでの指定と同様($,*,#,lは不可)。

(B)変換子に対応する変数がないときは、"(N/A)"を出力する。

(C)変換できないときは、"(ERR)"を出力する。

(D)データの編集指定子による編集規則を以下に示す。

	文字	整数	浮動少数点数	ＢＵＬＫ
s	文字列	文字列に変換	文字列に変換	文字列に変換
c	先頭１バイト	最下位１バイト	整数に変換し、最下位１バイト	先頭１バイト
d,i,o,u,x	整数に変換	数値	整数に変換	アドレス
f,e,g,a	浮動少数点数に変換	浮動少数点数に変換	数値	(ERR)
その他	アドレス	アドレス	アドレス	アドレス

（注）編集指定子は大文字でも同様。

8.33. ＥＬＲＥＡＤ１

(1)機能

標準入力から１データ要素を読み込む

(2)一般形式

```
var  = ELREAD1(attr, [size], [scale]) ;
```

以下、ＦＥＬＲＥＡＤ１と同じ。

8.34. ＥＬＷＲＩＴＥ

(1)機能

標準出力に複数個のデータ要素を書き込む。

(2)一般形式

```
ret  = ELWRITE(v1, v2, ...) ;
```

以下、ＦＥＬＷＲＩＴＥと同じ。

8.35. ＥＬＷＲＩＴＥ１

(1)機能

標準出力に１データ要素を書き込む。

(2)一般形式

```
ret  = ELWRITE1(fp, v1, [size], [scale]) ;
```

以下、ＦＥＬＷＲＩＴＥ１と同じ。

8.36. ＥＶＡＬ

(1)機能

文字列で与えられた式を評価し結果を返却する。

(2)一般形式

```
result  = EVAL(var[,opt]) ;
```

(3)構文規則

(4)一般規則

(A)varの内容が空のときはエラー。

(B)optは、varの実行でエラーが発生したときの処理を指定する。

=0 または省略: エラーメッセージを出力する。本プロックからエラーリターンする。

=1: エラーメッセージを出力する。エラーコードをERRORに設定し、正常終了する。NULL値を返却する。

=2: エラーメッセージを出力しない。他は、1 と同じ。

8.37. ＥＸＩＴ

(1)機能

スクリプト実行を中断し、セッションを終了する。

(2)一般形式

```
ret  = exit([error_code]) ;
```

(3)構文規則

(4)一般規則

(A)error_codeが指定されているときは、その値がセッションのリターン値となり、その値を返却する。
省略されているときは、ゼロを返却する。

(B)本関数を含む文の実行が終了した時点で、セッションが終了する。

(C)その他は、ＬＥＴ　ＥＸＩＴと同じ。

8.38. ＥＸＰ

(1)機能

eのx乗を求める。

(2)一般形式

```
ret  = EXP(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

8.39. ＦＣＬＯＳＥ

(1)機能

ファイルをクローズする。

(2)一般形式

```
ret  = FCLOSE(fp) ;
```

(3)構文規則

(4)一般規則

(A)標準入力、標準出力、標準エラー出力は、クローズできない。

8.40. ＦＥＬＲＥＡＤ１

(1)機能

ファイルから１データ要素を読み込む

(2)一般形式

```
var  = FELREAD1(fp, attr, [size], [scale]) ;
```

(3)構文規則

(A)attrは、以下の値を指定する。

文字列　　：1
整数　　：2
２進浮動小数点数　：3
１０進浮動小数点数：4（ファイル上のデータは、パック１０進と見なす）
バルク　　：5

(4)一般規則

(A)sizeは、読み取るファイル上の領域の長さ(バイト)を示す。

ただし、１０進浮動小数点数の場合は、(size/2+1)バイトのパック１０進として読み込む。
sizeがファイルのサイズより大きいときは、ファイルのサイズ分読み込む。
size省略時またはゼロ以下のときは、以下が指定されたと見なす。

・文字列　　：1
・整数　　：4
・２進浮動小数点数　：8
・１０進浮動小数点数：15
・バルク　　：1

２進浮動小数点数において、sizeが４未満のときは、エラーとなる。

(B)scale省略時は、ゼロと見なす。

１０進浮動小数点数のときは、
　少数点の右側(正のとき)または左側(負のとき)の桁数
その他のときは、
　scaleは、その大きさでデータとして実際に読み込む長さ示し、正負で位置を示す。
　scaleが正またはゼロのときは、以下を示し、負のときは、その逆となる。
　・文字列　　：前詰(左詰)
　・整数　　：後詰(右詰)
　・２進浮動小数点数　：後詰(右詰)
　・バルク　　：前詰(左詰)
　scaleがゼロのときは、sizeが設定される。
　size < |scale| のときは、scaleには、sizeの値が設定される。
scaleの大きさにより、データとして実際に読み込む長さおよびデータの型は以下である。

No.	attr	|scale|の値	データ長	データ型	備考
1	文字列	－	scaleバイト	文字列	
2	整数	1	1	char	読込み後intに変換
		2 ～ 3	2	short	同上
		4 ～ 7	4	int	同上
		8 以上	8	int64	同上
3	２進浮動小数点数	1 ～ 7	4	float	読込み後doubleに変換
		8 以上	8	double	
4	１０進浮動小数点数	少数点桁数	size/2 + 1	パック10進	
5	バルク	－	scaleバイト	バルク	

8.41. ＦＥＬＷＲＩＴＥ

(1)機能

ファイルに複数個のデータ要素を書き込む。

(2)一般形式

```
ret  = FELWRITE(fp, v1, v2, ...) ;
```

(3)構文規則

(4)一般規則

(A)変数または定数のデータ長で出力する。

- 文字　　：データ長
- 整数　　：4 バイト
- ２進浮動小数点数　：8 バイト
- １０進浮動小数点数：有効桁数(ファイル上のデータは、(有効桁数/2+1)バイトのパック１０進)
- バルク　　：データ長

(B)retには、出力データ長の合計が返る。

8.42. ＦＥＬＷＲＩＴＥ１

(1)機能

ファイルに１データ要素を書き込む。

(2)一般形式

```
ret  = FELWRITE1(fp, v1, [size], [scale]) ;
```

(3)構文規則

(4)一般規則

(A)sizeは、書き込むファイル上の領域の長さ(バイト)を示す。

ただし、１０進浮動小数点数の場合は、(size/2+1)バイトのパック１０進として書き込む。

size省略時またはゼロ以下のときは、以下が指定されたと見なす。

- ・文字　　：データ長
- ・整数　　：4 バイト
- ・２進浮動小数点数　：8 バイト
- ・１０進浮動小数点数: 有効桁数
- ・バルク　　：データ長

２進浮動小数点数において、sizeが４未満のときは、エラーとなる。

１０進浮動小数点数において、sizeが有効桁数より小さいときは、エラーとなる。

(B)scaleは、以下となる。

１０進浮動小数点数のときは、

省略時は、データのまま。

設定されたときは、少数点の右側(正のとき)または左側(負のとき)の桁数。

その他のときは、

scaleは、その大きさでデータとして実際に読み込む長さ示し、正負で位置を示す。

scale省略時またはゼロのときは、以下と見なす。

- ・文字　　：size
- ・整数　　：size (size>0), 4 バイト(size<=0)
- ・２進浮動小数点数　：size (size>0), 8 バイト(size<=0)
- ・バルク　　：size

size > |scale| のときは、余った部分には、文字ならスペース、その他はゼロが埋められる。

size < |scale| のときは、scaleには、sizeの値が設定される。

文字とバルクにおいて、|scale|>データ長のときは、|scale|バイト内で前詰めされる。

余った部分には、文字はスペース、バルクはゼロが埋められる。

データとして実際に書き込む長さおよび位置は、FELREAD1()と同じ。

8.43. ＦＦ

(1)機能

関数名と同じ文字列、または、関数定義文字列を関数指定の扱いにする。

(2)一般形式

```
FF(var)[(本関数の戻り値となった関数名の関数引数)]
```

(3)構文規則

(A)関数の直後に、関数引数を与える'('がなければならない。

(4)一般規則

(A)関数の戻り値は、関数の関数名にのみ使用できる。

(5)例

```
$x = FF('ABS');
print $x(-1);
```

8.44. ＦＧＥＴＬＩＮＥ

(1)機能

ファイルから１行読み込む。

(2)一般形式

```
line = FGETLINE(fp, [size], [opt]) ;
```

(3)構文規則

(4)一般規則

(A)改行コードまでを１行として読み込む。'¥r','¥r¥n','¥n'を改行コードと見なす。

(B)sizeは、１行の有効長を指定する。2以上をバイト単位で指定する。

１行の最初のsize-1バイトが読み込まれ、それより後ろは捨てられる。

デフォルトは、2048。

(C)optは、

0x01 ： ONのとき、'¥r','¥r¥n'は'¥n'に変換される。

0x02 ： ONのとき、行末の改行コードを削除する。

0x04 ： ONのとき、ＣＳＶ形式で読み込む(２重引用符中の改行コード以降も読み込む)。

0x08 ： ONのとき、ＣＳＶ形式で読み込み、２重引用符中の改行コードを削除する。

デフォルトは、0x02 + 0x04

8.45. ＦＩＲＳＴ

(1)機能

データ・リストの最初の要素を参照する。

(2)一般形式

```
var = FIRST(var1) ;
```

(3)構文規則

ＦＬと同じ。

(4)一般規則

ＦＬと同じ。

8.46. ＦＬ

(1)機能

データ・リストについて、以下の操作を行なう。

(A)LIST　・・・・リストを作成する

(B)FIRST ・・・・リストの最初の要素を参照する。

(C)REST　・・・・リストの最初のvar2個の要素を除くリストを参照する。(var2>=0)

var2省略時は、var2=1と同じ。

(D)CONS　・・・・リストまたは要素を連結して１つのリストを作る。

(E)LIST_REF　・・リストのvar2番目の要素を参照する。(先頭は０番目)

var2省略時は、var2=0と同じ。

(2)一般形式

```
var = FL('LIST', [var1], [var2], ...) ;
var = FL('FIRST', var1) ;
var = FL('REST',  var1 [,var2]) ;
var = FL('CONS', [var1], [var2], ...) ;
var = FL('LIST_REF',  var1 [,var2]) ;
```

(3)構文規則

(A)第一パラメータには、機能に示す操作指示文字列を指定する。大文字と小文字は区別されない。

(4)一般規則

(A)全般

①パラメータが省略されたときは、ＮＵＬＬ値が追加される。

②データ要素がゼロのときの扱いは、オプション番号５によって異なる。

(B)LIST

①パラメータに指定された変数がデータ・リストのときは、そのデータ・リストが作成されるリストにそのまま追加される。

8.47. ＦＬＯＯＲ

(1)機能

　する

(2)一般形式

```
ret  = FLOOR(x) ;
```

(3)構文規則

(4)一般規則

8.48. ＦＯＰＥＮ

(1)機能

　ファイルをオープンする

(2)一般形式

```
fp   = FOPEN(fname [,mode]) ;
```

(3)構文規則

(4)一般規則

(A)mode省略時は、'r'。

8.49. ［ＦＰ］ＳＴＡＴ

(1)機能

ファイルの属性を取得する。

(2)一般形式

```
ret = STAT(file [, map_index [, maxval] ]) ;
ret = FPSTAT(fp [, map_index [, maxval] ]) ;
```

(3)構文規則

(A)map_indexには、配列名または数値を指定する。

(B)map_indexに文字定数が指定されたときは、数値に変換される。

(4)一般規則

(A)属性値は、以下の順で指定された配列に返却される。

```
st_val[ 0] = f_type;
                    =0 : それ以外
                    =1 : 名前付きパイプ
                    =2 : キャラクタ・ディバイス
                    =3 : ディレクトリ
                    =4 : ブロック・ディバイス
                    =5 : レギュラ・ファイル
                    =6 : シンボリック・リンク
st_val[ 1] = tStat.st_dev;
st_val[ 2] = tStat.st_ino;
st_val[ 3] = tStat.st_mode;
st_val[ 4] = tStat.st_nlink;
st_val[ 5] = tStat.st_uid;
st_val[ 6] = tStat.st_gid;
st_val[ 7] = tStat.st_rdev;
st_val[ 8] = tStat.st_size;          /* 属性は、DECIMAL */
st_val[ 9] = tStat.st_atim.tv_sec;
st_val[10] = tStat.st_atim.tv_nsec;
st_val[11] = tStat.st_mtim.tv_sec;
st_val[12] = tStat.st_mtim.tv_nsec;
st_val[13] = tStat.st_ctim.tv_sec;
st_val[14] = tStat.st_ctim.tv_nsec;
st_val[15] = tStat.st_blksize;
st_val[16] = tStat.st_blocks;
```

(B)map_indexに配列名を指定したときは、属性値は、配列の先頭から最大で、配列のサイズまで格納される。

maxvalの値が、配列のサイズより大きいときは、配列のサイズまで格納される。

(C)map_indexに数値が指定されたときは、内部番号変数の当該位置から、最大でmaxval個格納される。

maxvalが、内部番号変数の最後の変数を越えるような場合は、エラーとなる。

(D)map_indexには、連想配列名は指定できない。

8.50. ＦＰＵＴＬＩＮＥ

(1)機能

ファイルに複数個の文字列を書き込む。

(2)一般形式

```
ret  = FPUTLINE(fp, v1, v2, ...) ;
```

(3)構文規則

(4)一般規則

(A)変数または定数のデータ長で出力する。

１つの引数データの出力形式は、以下を除き、SAYコマンドと同じ。

(a)BULKデータは、そのデータ長まで出力する。

(B)末尾には、改行コード('¥n')を出力する。これは、オプション３で変更できる。

(C)retには、出力データ長の合計が返る。

8.51. ＧＥＴＡＲＧＳ

(1)機能

文字列をカンマまたは空白文字または指定文字で分割する。分割の最大値は、128。

(2)一般形式

```
n    = GETARGS(line, map_index, [max_args], [opt], [sep_char]) ;
```

(3)構文規則

(A)map_indexには、配列名または数値を指定する。

(B)map_indexに文字定数が指定されたときは、数値に変換される。

(C)max_argsのデフォルト値は、128。

(4)一般規則

(A)区切り文字は、空白文字（スペースまたはタブ）またはカンマ。引用符内のときは、区切りとならない。区切られた文字列の前後の空白文字は削除される。

(B)map_indexに配列名を指定したときは、アーギュメントは、配列の先頭から

最大で、配列のサイズまで格納される。

max_argsの値が、配列のサイズより大きいときは、配列のサイズまで格納される。

(C)map_indexに数値が指定されたときは、内部番号変数の当該位置から、最大でmax_args個格納される。

max_argsが、内部番号変数の最後の変数を越えるような場合は、エラーとなる。

(D)sep_charが指定されたときは、sep_charで文字列を分割する。

これが、カンマ(,)であり、optが未指定のときは、自動でCSV形式の処理になる。

(E)max_argsが指定され分割数がこれに満たない場合は、残りには、null値が設定される。

(F)map_indexには、連想配列名は指定できない。

(G)optは、以下の通り。

No.	opt	機能	備考
1	0x01	'#'以降を無視する	デフォルト
2	0x02	２ワード目が、'='のときは無視する １ワード目のときは、ＮＵＬＬ値とみなす	デフォルト
3	0x04	','も区切りとする。 ','が現れた時点で0x12=ONの指定を無効にする	デフォルト
4	0x10	':'or':='もワード区切りとする １ワード目のときは、ＮＵＬＬ値とみなす	
5	0x20	'['と']'を引用符と同様に扱う(IPV6対応)	
6	0x40	0x12=1のとき、'='or':'or':='が１ワード目のときには、１ワード目として設定する	
7	0x80	0x04=1のとき、','が現れた時点で0x12=ONの指定を無効にしない	
8	0x010000	前後の引用符を残す。また、引用符の中の連続する２つの引用符を１つにしない	
9	0x040000	ＣＳＶ形式でのワード処理を行う	sep_charがカンマ(,)のときは、デフォルト

8.52. ＧＥＴＥＮＶ

(1)機能
　環境変数の値を取得する。

(2)一般形式

```
ret = getenv(name) ;
```

(3)構文規則
　なし。
(4)一般規則
(5)戻り値
　環境変数の値。環境変数が未設定のときは、NULL値が返される。

8.53. ＧＥＴＬＩＮＥ

(1)機能
　標準入力から１行読み込む。

(2)一般形式

```
line = GETLINE([size], [opt]) ;
```

以下、ＦＧＥＴＬＩＮＥと同じ。

8.54. ＧＥＴＬＯＧＰＡＲＭ

(1)機能
　ログパラメータを取得する。

(2)一般形式

```
ret = GETLOGPARM(logno, map_index, [max_args]) ;
```

(3)構文規則
(A)map_indexは、GETARGS()と同じ。
(B)max_argsのデフォルト値は、7。

(4)一般規則
(A)lognoは、番号またはログ名称で指定する。ＬＥＴ　ＬＯＧＰＡＲＭコマンドを参照。
(B)map_indexは、GETARGS()と同じ。
(C)取得されるログパラメータは以下の通り。詳細は、ＬＥＴ　ＬＯＧＰＡＲＭコマンドを参照。

```
LOGNO:ログ番号
FLAG:出力フラグ
LEVEL:出力レベル
SIZE_MAX:ローテーション・ファイル・サイズ(Kbyte)
FILE_MAX:ローテーション・ファイル数
OPTION:ローテーション・オプション
FILE:出力ファイル名
```

(5)戻り値
　取得パラメータ数。

8.55. ＧＥＴMEMUＳＥＤ

(1)機能

各種のメモリ使用量を取得する。

(2)一般形式

```
ret = GETMEMUSED(map_index, [max_args]) ;
```

(3)構文規則

(A)map_indexは、GETARGS()と同じ。

(B)max_argsのデフォルト値は、5。

(4)一般規則

(A)map_indexは、GETARGS()と同じ。

(B)以下のメモリ使用量を取得する。

・配列要素の１番目：当該スクリプトのソースおよびコンパイル情報。
・配列要素の２番目：当該セッションで使用している定数情報。
・配列要素の３番目：当該スクリプト実行で使用している定数情報。
・配列要素の４番目：現在行の実行で使用しているワーク情報。
・配列要素の５番目：当該セッション実行中に保存している情報。
　スクリプト、手続き、関数実行中のみ保存している情報を含む。

(5)戻り値

取得したメモリ使用量の合計。

8.56. ＧＥＴＴＩME

(1)機能

セッション開始またはシステム開始からの経過時間を秒単位ミリ秒の精度で返す。

(2)一般形式

```
ret = gettime([opt]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)opt

省略または0：セッション開始から
0以外の数値：システム開始から

(5)戻り値

１０進浮動少数点数で返す。

8.57. ＧＥＴＶＡＬ

(1)機能

各種設定値の値を取得する。

(2)一般形式

```
ret = getval(取得名 [, 要素名または番号]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)取得名と要素名または番号は以下。

(a)OPTION:オプションの設定値。

要素番号：オプション番号

(5)戻り値

取得した値。

8.58. ＧＥＴWD

(1)機能

指定された区切り文字で区切ってワードを取り出す。区切り文字もワードとして取り出す。
ただし、ブランク、タブ、改行コード('¥n','¥r')はスキップする。（opt指定）
また、引用符、２重引用符は、引用符、２重引用符処理を行う（opt指定）。

(2)一般形式

```
len  = getwd(line, map_index, [sep], [opt], [wdlen]) ;
```

(3)構文規則

(A)map_indexは、GETARGS()と同じ。

(4)一般規則

(A)map_indexには、以下が設定される。

配列の１番目：ワードのサーチ開始位置。文字列の先頭を１文字目と数える。
サーチ開始位置は、取り出されたワードの次の位置に自動的に更新される。
(注)初回呼び出し時には、設定しなければならない。

配列の２番目：取り出したワード。

配列の３番目：ワード種別（ワードが区切り文字のときは、区切り文字に0x80をＯＲしたもの）。
１：名票、数字等のワード
５：引用符ではじまる
６：２重引用符ではじまる

配列の４番目：２バイト文字有りフラグ。1/0=あり／なし。

配列の５番目：0x01 不完全〔２重〕引用符のとき、１。
0x02 ワード長オーバーのとき、１。

配列の６番目：[２重]引用符中に連続する[２重]引用符ありのとき、１

配列の個数が６未満のときは、エラーとなる。

(B)sepには、区切り文字を指定する。デフォルトは、以下。
△¥t,;()[]{}<>='″!^*/　　(注)△は、半角スペース

(C)optは、以下の通り。

No.	opt	機能	デフォルト
1	0x01	1：先行ブランクとタブをスキップする。 0：スキップしない。	1
2	0x02	1：引用符、２重引用符をはずす。 0：はずさない（不完全エラーあり）。	1
3	0x04	1：引用符、２重引用符の中の連続する２つのそれらを１つにする。 0：１つにしない。	1
4	0x08	1：引用符、２重引用符の処理をしない。 0：処理をする。引用符、２重引用符もワードの区切りとなる。	0
5	0x10	1：区切り文字以外が現れるまでスキップする。 0：左記処理をしない。	0
6	0x20	未使用。	―
7	0x40	1：'¥n'および'¥r'を終端としない。 (引用符内のときのみ有効。ブランクと同じ扱いする) 0：'¥n'または'¥r'を終端とする。	0
8	0x80	未使用。	―
9	0x0100	1：引用符の処理をしない。	0
10	0x0200	1：２重引用符の処理をしない。	0

(D)wdlenには、取り出すワードの最大文字数を指定する。
>0: 指定文字数まで格納する。
=<0または省略: 255文字。

(5)戻り値
以下を返す。
>=0: 取り出したワードの文字数を返す。
<0: 文字列の終端。

8.59. ｉＩＮ

(1)機能
　大文字、小文字を区別しない複数文字列との比較結果を返す。

(2)一般形式

```
result  = iIN(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則
(A)start、lenについては、IN()と同じ。

(4)一般規則
(A)半角について大文字、小文字を区別しない。その他は、IN()と同じ。

8.60. ｉＬＩＫＥ

(1)機能
　大文字、小文字を区別しないパターン・マッチング結果を返す。

(2)一般形式

```
result  = iLIKE(var [, start [, len]], pat1 [, pat2, pat3, ...]) ;
```

(3)構文規則
(A)start、lenについては、INSTR()と同じ。

(4)一般規則
(A)半角について大文字、小文字を区別しない。その他は、ＬＩＫＥと同じ。

8.61. ＩＮ

(1)機能
　複数文字列との比較結果を返す。

(2)一般形式

```
result  = IN(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則
(A)第２、第３引数(start,len)に数値属性が指定されたときは、比較開始位置、文字数と見なす。
　比較開始位置は、比較対象文字列(var)の先頭を１文字目として文字単位で数える。
(B)第２引数(start)に範囲指定が指定されたときは、比較開始位置、終了位置と見なす。
(4)一般規則
(A)複数文字列(str1,str2,...)のどれかと一致したとき、１を返す。
　どれとも一致しなかったとき、０を返す。

8.62. ＩＮＤＥＸＡ

(1)機能

配列の１次元要素位置を求める。

(2)一般形式

```
index = INDEXA(array, index1, [index2], [index3]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)連想配列でないとき、省略された次元のindexは、ゼロと見なされる。

(B)連想配列のときは、キー値のハッシュインデックスが返される。

8.63. ＩＮｉＲＳＴＲ

(1)機能

後ろからの大文字、小文字を区別しない文字列サーチ結果を返す。

(2)一般形式

```
result  = INiRSTR(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則

ＩＮＳＴＲと同じ。

(4)一般規則

(A)後ろからサーチする以外は、ＩＮｉＳＴＲと同じ。

8.64. ＩＮｉＳＴＲ

(1)機能

大文字、小文字を区別しない文字列サーチ結果を返す。

(2)一般形式

```
result  = INiSTR(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則

ＩＮＳＴＲと同じ。

(4)一般規則

(A)半角について大文字、小文字を区別しない。その他は、ＩＮＳＴＲと同じ。

8.65. ＩＮＬＩＫＥ

(1)機能

文字列(var)のＬＩＫＥ一致ヶ所数と位置を返す。

(2)一般形式

```
result = INLIKE(map_index, npos, var [, start [, len]], pat_str, [opt_str], [escape]);
```

(3)構文規則

(A)map_indexについては、GETARGS()と同じ。

(B)第４、第５引数(start,len)に数値属性が指定されたときは、サーチ開始位置、文字数と見なす。
サーチ開始位置は、サーチ対象文字列(var)の先頭を１文字目として文字単位で数える。

(C)第４引数(start)に範囲指定が指定されたときは、サーチ開始位置、終了位置と見なす。

(4)一般規則

(A)resultには、一致したヶ所数を返す。

(B)map_indexには、先頭から順に、一致した位置と長さをnpos組返す。
位置と長さの単位は文字数。位置はサーチ対象文字列の先頭を１文字目と数える。
位置は、オプションによって、サーチ開始位置が１文字目に変わる。(INSTR()と同じ)
nposがゼロのときには、map_indexには結果を返さない。
nposを省略したときは、最大でmap_indexの配列数/2組の結果をかえす。

(C)pat_str,opt_str,escapeは、REPLIKE()と同じ。

(D)サーチ方法は、INSTR()と同様。

8.66. ＩＮＲＳＴＲ

(1)機能

後ろからの文字列サーチ結果を返す。

(2)一般形式

```
result  = INRSTR(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則

ＩＮＳＴＲと同じ。

(4)一般規則

(A)一致した文字列中で最も後ろの先頭位置を返す。どれとも一致しなかったとき、０を返す。

(B)後ろからサーチする以外は、上記を除きＩＮＳＴＲと同じ。

8.67. ＩＮＳＴＲ

(1)機能

文字列サーチ結果を返す。

(2)一般形式

```
result  = INSTR(var [, start [, len]], str1 [, str2, ...]) ;
```

(3)構文規則

(A)第２、第３引数(start,len)に数値属性が指定されたときは、サーチ開始位置、文字数と見なす。
サーチ開始位置は、サーチ対象文字列(var)の先頭を１文字目として文字単位で数える。

(B)第２引数(start)に範囲指定が指定されたときは、サーチ開始位置、終了位置と見なす。

(4)一般規則

(A)一致した文字列中で最も手前の先頭位置を返す。どれとも一致しなかったとき、０を返す。

(B)サーチは文字単位で行い、一致したときの位置は、サーチ対象文字列の先頭を１文字目として文字単位で数える。一致位置は、オプションによって、サーチ開始位置が１文字目に変わる。

(C)スクリプトの仕様で、返却する一致文字列の位置を数える単位が変わる。
旧仕様：バイト単位
新仕様：文字単位

8.68. ｉＲＥＧＥＸ

(1)機能

大文字、小文字を区別しない正規表現のパターン・マッチング結果を返す。

(2)一般形式

```
result  = iREGEX(var [, start [, len]], pat1 [, pat2, pat3, ...]) ;
```

(3)構文規則

(A)REGEX()と同じ。

(4)一般規則

(A)半角について大文字、小文字を区別しない。その他は、ＲＥＧＥＸと同じ。

8.69. ＩＳ

（１）機能

文字列を調査する。

（２）一般形式

```
ret = IS(var [, start [, len]], check) ;
```

（３）構文規則

（Ａ）文字列演算式の第一項と第二項がそれぞれ第１引数と第２引数に対応する。

（Ｂ）第２引数(start)に数値属性が指定されたときは、サーチ開始位置と見なす。
サーチ開始位置は、サーチ対象文字列の先頭を１文字目として文字単位で数える。

（４）一般規則

（Ａ）第１引数が文字列に変換されない場合は、サーチ開始位置情報は使用されない。

（Ｂ）その他は、文字列演算子を参照。

8.70. ＬＥＦＴ

(1)機能

文字列の左側を取り出す

(2)一般形式

```
result  = left(var, len) ;
```

(3)構文規則

なし

(4)一般規則

(A)varがNULL値のとき、または、lenが０以下のときは、NULL値を返す。

(B)lenがvarの文字列長より大きいときは、varをそのまま返す。

(C)varが文字属性のときは、スクリプトの仕様で、lenの単位が変わる。

旧仕様：バイト単位

新仕様：文字単位

8.71. ＬＥＦＴＢ

(1)機能

バイト単位で文字列の左側を取り出す

(2)一般形式

```
result  = leftb(var, bytes) ;
```

(3)構文規則

なし

(4)一般規則

(A)varがNULL値のとき、または、bytesが０以下のときは、NULL値を返す。

(B)bytesがvarの文字列長(バイト)より大きいときは、varをそのまま返す。

8.72. ＬＥＮＧ

(1)機能

データの長さまたは文字数を返す。

(2)一般形式

```
len = LENG(var) ;
```

(3)構文規則

なし。

(4)一般規則

(A)varの属性により返す値は以下となる。

(a)一般変数

No.	属性	返す値
1	文字	スクリプト仕様により長さを数える単位が変わる。 旧仕様：バイト単位。 新仕様：文字単位。
2	整数	4
3	２進浮動小数点	8
4	１０進小数点	64
5	ＢＵＬＫ	データ長(バイト)

(B)その他の変数

３２を返す。

8.73. ＬＥＮＢ

(1)機能

バイト単位で長さを返す。

(2)一般形式

```
len = LENB(var) ;
```

(3)構文規則

なし

(4)一般規則

(A)varが文字属性のとき、バイト単位で長さを返す。その他のときは、LENG()と同じ。

8.74. ＬＩＫＥ

(1)機能

パターン・マッチング結果を返す。

(2)一般形式

```
result  = LIKE(var [, start [, len]], pat1 [, pat2, ...]) ;
```

(3)構文規則

(A)start、lenについては、INSTR()と同じ。

(4)一般規則

(A)パターンは、ＳＱＬのＬＩＫＥと同様に指定する。

'%': ＮＵＬＬ値を含む任意の文字列に一致する。

'_': ＮＵＬＬ値を含む任意の１文字に一致する。

'[',']': これで囲まれた文字列中のどれか１文字と一致する。

この文字列中では、ハイフン('-')で繋げて文字の範囲を複数指定することができる。

範囲指定の文字には、半角と全角の組合わせを指定することができる。

ただし、左側の文字コード≦右側の文字コードであること。

'¥': パターン文字に'%'、'_'、'['、']'を含めるとき、それらの前に付ける。

文字定数中で指定するときは、'¥'が文字定数でのエスケープ文字となるため、'¥¥'と指定しなければならない。

(B)パターンのどれかにマッチしたとき、１を返す。どれともマッチしなかったとき、０を返す。

(C)パターンのどれかがＮＵＬＬ値のときは、無条件にマッチする。

8.75. ＬＩＳＴ

(1)機能

データ・リストを作成する

(2)一般形式

```
var = LIST([var1], [var2], ...) ;
```

(3)構文規則

ＦＬと同じ。

(4)一般規則

ＦＬと同じ。

8.76. ＬＩＳＴ＿ＲＥＦ

(1)機能

リストのvar2番目の要素を参照する。(先頭は０番目)

var2省略時は、var2=0と同じ。

(2)一般形式

```
var = LIST_REF(var1 [,var2]) ;
```

(3)構文規則

Ｌと同じ。

(4)一般規則

Ｌと同じ。

8.77. ＬＯＧ

(1)機能

自然対数を求める。

(2)一般形式

```
ret  = LOG(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xには、1.0E-6より大きい値を指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.78. ＬＯＧ１０

(1)機能

１０を底とする対数を求める。

(2)一般形式

```
ret  = LOG10(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xには、1.0E-6より大きい値を指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.79. ＬＯＧＯＵＴ

(1)機能

ログを出力する

(2)一般形式

```
ret = LOGOUT(logno,format[,var1,var2,...,var5]);
```

(3)構文規則

(4)一般規則

(A)lognoは、ＬＥＴ　ＬＯＧＰＡＲＭコマンドと同じ。

8.80. ＭＡＸ

(1)機能

最大値を求める。

(2)一般形式

result = MAX(var1[,var2,…]) ;

(3)構文規則

なし

(4)一般規則

(A)var1が数値のときは、Var2以降はvar1の属性に変換されて比較される。

(B)var1が文字のときは、Var2以降は文字に変換されて比較される。

8.81. ＭＩＮ

(1)機能

最小値を求める。

(2)一般形式

result = MIN(var1[,var2,…]) ;

(3)構文規則

なし

(4)一般規則

(A)var1が数値のときは、Var2以降はvar1の属性に変換されて比較される。

(B)var1が文字のときは、Var2以降は文字に変換されて比較される。

8.82. ＭＯＤ

(1)機能

varをradで割った余りを求める。

(2)一般形式

result = MOD(var, rad) ;

(3)構文規則

なし

(4)一般規則

(A)varおよびradが整数値でないときは、整数値に変換される。

8.83. ＮＤＥＦ

(1)機能

文字列で与えられた式を評価し、変数が未定義、変数値が未設定、ＮＵＬＬパラメータ属性を持つＮＵＬＬ値のとき、指定された値に置き換える

(2)一般形式

result = NDEF(var [,rep]) ;

(3)構文規則

varには、評価したい式を文字列で表した式を指定する。【例】$xを調べたいときは、'$x'とする。

(4)一般規則

(A)varの内容が空のときはエラー。

(B)repが省略されたときは、NULL値が使用される。

(C)opt=0x01のときは、評価結果(1:変数が未定義等/0: 変数が定義済み)を返す。

8.84. ＮＥＷ

(1)機能

クラスのインスタンスを生成する。

(2)一般形式

```
inst = NEW({ クラス名 | var } [,パラメータリスト]) ;
```

(3)構文規則

なし

(4)一般規則

(A)パラメータリストには、実行したいコンストラクタと同じ数のパラメータを指定する。
(B)varには、クラス名をデータとする文字定数か文字変数を指定する。

8.85. ＮＶＡＬ

(1)機能

式を評価し、ＮＵＬＬ値のときは、指定された値に置き換える

(2)一般形式

```
result  = NVAL(var [, rep]) ;
```

(3)構文規則

(4)一般規則

(A)repが省略されたときは、NULL値が使用される。
(B)式の値が未定義またはＮＵＬＬパラメータのときはエラーとなる。

8.86. ＮＯＦＲＥＥ

(1)機能

未開放メモリのアドレスをファイルに出力する

(2)一般形式

```
ret = NOFREE(var);
```

(3)構文規則

(4)一般規則

(A)

8.87. ＯＰＥＮＤＩＲ

(1)機能

ディレクトリをオープンする。

(2)一般形式

```
dp = OPENDIR(dirname) ;
```

(3)構文規則

(4)一般規則

(A)ディレクトリへのポインタを返す。

8.88. ＰＣＬＯＳＥ

(1)機能

パイプをクローズする。

(2)一般形式

```
ret  = PCLOSE(fp) ;
```

(3)構文規則

(4)一般規則

(A)パイプ以外は、クローズできない。

8.89. ＰＯＰＥＮ

(1)機能

パイプをオープンしてコマンド・ラインを実行する

(2)一般形式

```
fp   = POPEN(line [,mode]) ;
```

(3)構文規則

(4)一般規則

(A)mode省略時は、'r'。

(3)構文規則

8.90. ＰＯＷＥＲ

(1)機能

xのy乗を求める。

(2)一般形式

```
ret  = POWER(x,y) ;
```

(3)構文規則

なし。

(4)一般規則

(A)x,yが浮動小数でないときは、浮動小数に変換される。

8.91. ＰＵＴＥＮＶ

(1)機能

環境変数の値を設定する(name=value形式)。

(2)一般形式

```
ret  = PUTENV(name_value) ;
```

(3)構文規則

なし。

(4)一般規則

(A)本設定は、当該セッションの間のみ有効となり、変更された値は、セッション終了時に元に戻る。

(5)戻り値

0: 正常。-1: エラー。

8.92. ＰＵＴＬＩＮＥ

(1)機能

標準出力に複数個の文字列を書き込む。

(2)一般形式

```
ret  = PUTLINE(v1, v2, ...) ;
```

以下、ＦＰＵＴＬＩＮＥと同じ。

8.93. ＲＡＮＤ１(ＤＲＡＮＤ４８)

(1)機能

0.0<=、<1.0の間の乱数を求める。

(2)一般形式

```
ret  = RAND1() ;
```

(3)構文規則

なし。

(4)一般規則

8.94. ＲＥＡＤＤＩＲ

(1)機能

ディレクトリの１エントリを読み込み、ファイル名を返す。

(2)一般形式

```
file = READDIR(dp) ;
```

(3)構文規則

なし

(4)一般規則

(A)エントリを読み切ると、ＮＵＬＬ値を返し、$ERROR=-1となる。

8.95. ＲＥＧＥＸ

(1)機能

正規表現のパターン・マッチング結果を返す。

(2)一般形式

```
result  = REGEX(var [, start [, len]], pat1 [, pat2, pat3, ...]) ;
```

(3)構文規則

(A)start、lenについては、INSTR()と同じ。

(4)一般規則

(A)パターンは、ＵＮＩＸのＲＥＧＥＸ(拡張仕様)と同様に指定する。

'*':

'?':

'[',']': これで囲まれた文字列中のどれか１文字と一致する。

'¥': パターン文字に'*'、'?'、'['、']'を含めるとき、それらの前に付ける。

文字定数中で指定するときは、'¥'が文字定数でのエスケープ文字となるため、'¥¥'と指定しなければならない。

(B)パターンのどれかにマッチしたとき、１を返す。どれともマッチしなかったとき、０を返す。

(C)マッチするパターンより前にＮＵＬＬ値のパターンがあるときは、エラーとなる。

(D)パターンのどれかが'()'のときは、無条件にマッチする。

8.96. ＲＥＰ

（１）機能

文字列の置換

（２）一般形式

```
result = REP(var [, start [, len]], rep_parm) ;
```

（３）構文規則

(A)文字列演算式の第一項と第二項がそれぞれ第１パラメータと第２パラメータに対応する。

(B)start、lenについては、INSTR()と同じ。

（４）一般規則

文字列演算子を参照

8.97. ＲＥＰＬＩＫＥ

(1)機能

文字列のＬＩＫＥ置換

rep_patに一致する文字列をrep_strに置換する。

resultには、置換対象文字列(var)の先頭以降の置換された結果の文字列を返す。

オプションによって、サーチ開始以降の文字列を返す。

(2)一般形式

```
result = REPLIKE(var [, start [, len]], rep_pat, rep_str, opt_str, escape);
```

(3)構文規則

(A)start、lenについては、INSTR()と同じ。

(4)一般規則

(A)opt_strに'i'または'I'が含まれるときは、半角英字の大文字、小文字は無視される。

(B)opt_strに'g'または'G'が含まれるときは、rep_patに一致する全ての文字列をrep_strに置換する。

(C)rep_patがNULL値のときは、何も置換しない。

(D)'^'は、先頭に一致することを示す。逆に、'$'は、末尾に一致することを示す。

(E)rep_patの指定方法は、ＬＩＫＥと同様である。ただし、先頭または末尾に'^'、'$'、'%'がないときは、'%'があるものと見なされる。
以下に、例を示す。

(F)opt_strに'o'または'0'が含まれるときは、rep_patに一致する文字列を抜き出す。

(G)escapeが指定されたときは、それをエスケープ文字とする。escapeが、ＮＵＬＬ値のときは、エスケープ処理を行わない。
デフォルトは、'¥'である。使用方法は、LIKE関数を参照のこと。

(H)rep_patが１文字であり、エスケープ文字のときは、エスケープ文字とは見なさない。

No.	rep_pat	置換動作	'g'指定時
1	'ABC'	初めの'ABC'がrep_strに置換される	全ての'ABC'が置換される
2	'^'	先頭にrep_strが挿入される	同左
3	'$'	末尾にrep_strが追加される	同左
4	'^$'	先頭と末尾にrep_strが挿入される	同左
5	'^ABC'	先頭の'ABC'がrep_strに置換される	同左
6	'ABC$'	末尾の'ABC'がrep_strに置換される	同左
7	'^ABC$'	varが'ABC'のときのみ置換される	同左
8	'A%C'	初めの'AC'または'A...C'が置換される	全ての'AC'または'A...C'が置換される
9	'%'	varがrep_strに置換される	同左
10	'^%ABC'	'ABC'と同じ	同左
11	'ABC%$'	'ABC'と同じ	同左
12	'_B_'	初めの真ん中が'B'の３バイトが置換される	全ての左記３バイトが置換される
13	'^%'	'%'と同じ	同左
14	'%$'	'%'と同じ	同左
15	'$^'	varが'$^'のときのみ置換される	同左
16	'$%'	先頭の'$'が置換される	同左
17	'%^'	末尾の'^'が置換される	同左
18			

8.98. ＲＥＳＬＯＧＰＡＲＭ

(1)機能

取得済ログパラメータをリストアする。

(2)一般形式

```
ret = RESLOGPARM(map_index, [max_args]) ;
```

(3)構文規則

(A)map_indexは、GETARGS()と同じ。

(B)max_argsのデフォルト値は、7。

(4)一般規則

(A)map_indexは、GETARGS()と同じ。

(B)リストアされるログパラメータは、ＧＥＴＬＯＧＰＡＲＭコマンドを参照。

(5)戻り値

設定パラメータ数。

8.99. ＲＥＳＴ

(1)機能

データ・リストの最初のvar2個の要素を除くリストを参照する。(var2>=0)

var2省略時は、var2=1と同じ。

(2)一般形式

```
var = REST(var1 [,var2]) ;
```

(3)構文規則

ＦＬと同じ。

(4)一般規則

ＦＬと同じ。

8.100. ＲＩＧＨＴ

(1)機能

文字列の右側を取り出す

(2)一般形式

```
result  = RIGHT(var, len) ;
```

(3)構文規則

なし

(4)一般規則

(A)varがNULL値のとき、または、lenが０以下のときは、NULL値を返す。

(B)lenがvarの文字列長より大きいときは、varをそのまま返す。

(C)varが文字属性のときは、スクリプトの仕様で、lenの単位が変わる。

旧仕様：バイト単位

新仕様：文字単位

8.101. ＲＩＧＨＴＢ

(1)機能

バイト単位で文字列の右側を取り出す

(2)一般形式

```
result  = RIGHTB(var, bytes) ;
```

(3)構文規則

(4)一般規則

(A)varがNULL値のとき、または、bytesが０以下のときは、NULL値を返す。

(B)bytesがvarの文字列長(バイト)より大きいときは、varをそのまま返す。

8.102. ＲＩＮＴ

(1)機能

を求める。

(2)一般形式

```
ret  = RINT(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

8.103. ＲＯＵＮＤ

(1)機能

数値を丸める。

(2)一般形式

```
result = ROUND(var, [scale], [opt]) ;
```

(3)構文規則

(4)一般規則

(A)scale桁に丸める。省略時は、ゼロと見なす。

scale >= 0 : 小数点以下scale桁に丸める。

scale < 0 : 小数点以上-scale桁目を丸める。

【例】sacle=-1 opt=0 var=125 ==> result=130

(B)optの下２ビットで丸め方を指定する。省略時は、ゼロと見なす。

opt = 0x00 : 四捨五入。

opt = 0x01 ビットON : 切捨て。(0x02ビットONでも優先する)

opt = 0x02 ビットON : 切上げ。

(C)varが文字列のときは、数値に変換される。

(D)数値の属性は保存される。

8.104. ＳＥＴＡＲＲＡＹ

(1)機能

配列にデータを設定する。

(2)一般形式

```
n  = SETARRAY(array, start, [var1], [var2], ...) ;
```

(3)構文規則

(A)arrayには、配列名または数値を指定する。
(B)arrayに文字定数が指定されたときは、数値に変換される。
(C)arrayに連想配列名を指定したときは、varには、キー、データの順に指定する。

(4)一般規則

(A)arrayに配列名を指定したときは、配列名[start]の要素から最大で、配列のサイズまで格納される。
(B)arrayに数値が指定されたときは、内部番号変数の$(指定数値+start)の要素から、最大で配列のサイズまで格納される。
(C)設定データが省略された位置に対応する要素への設定はスキップされる。
(D)返却値nには、実際に設定されたデータ個数が返る。
(E)連想配列で、キー省略されたデータはスキップされる。同じキーのときは、後が有効となる。

8.105. ＳＥＴＥＮＶ

(1)機能

環境変数の値を設定する。

(2)一般形式

```
ret = SETENV(name, value [,overwrite]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)overwriteが0のときは、既存の環境変数の値を上書きしない。0以外のときは、上書きする。
overwrite省略時は、1と見なされる。
(B)本設定は、当該セッションの間のみ有効となり、変更された値は、セッション終了時に元に戻る。

(5)戻り値

0: 正常。-1: エラー。

8.106. ＳＥＴＬＯＧＰＡＲＭ

(1)機能

ログパラメータを設定する。

(2)一般形式

```
ret  = SETLOGPARM(log_no,flag,[level],[size_max],[file_max],[option],[log_file]) ;
```

(3)構文規則

(A)後続の引数を省略するときは、不要なカンマも省略する。

(4)一般規則

(A)引数については、ＬＥＴ　ＬＯＧＰＡＲＡＭコマンドを参照。

8.107. ＳＨＥＬＬ

(1)機能

システムのコマンド・ラインを実行する。

(2)一般形式

```
ret  = SHELL(line) ;
```

(3)構文規則

(4)一般規則

(A)

8.108. ＳＨＵＴＣＴＬ

(1)機能

ｓｈｕｔ状態を返却する。

(2)一般形式

```
ret  = SHUTCTL([mode,var]) ;
```

(3)構文規則

(4)一般規則

(A)ｓｈｕｔ状態であり、ｓｈｕｔが保留されていないとき、０以外を返す。

8.109. ＳＩＮ

(1)機能

sinを求める。

(2)一般形式

```
ret  = SIN(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.110. ＳＩＮＨ

(1)機能

sinhを求める。

(2)一般形式

```
ret  = SINH(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.111. ＳＫＩＰ＿ＯＰＴ

(1)機能

文字列をサーチし、スキップした文字数を返す。

(2)一般形式

```
pos = SKIP_OPT(str, pat [, opt]) ;
```

(3)構文規則

なし。

(4)一般規則

(A)optは以下を組み合わせて指定する。

```
0x01;   /* ignore case */
0x02;   /* reverse */
0x08;   /* 0/1=in/to */
```

(B)str中でpat内のどれかの文字を対称として、optのスキップ指定に合致するスキップ文字数を返す。

(C)サーチは文字単位で行い、一致したときの位置は、サーチ対象文字列の先頭を１文字目として文字単位で数える。

(D)スクリプトの仕様で、返却する一致文字列の位置を数える単位が変わる。

旧仕様：バイト単位

新仕様：文字単位

8.112. ＳＯＲＴ

(1)機能

配列等のデータをソートする。

(2)一般形式

ret = SORT(array_d, array_s, number, [option], [ソート位置指定のリスト]) ;

(3)構文規則

(A)array_d および array_s には、配列名または数値を指定する。

文字データが指定されたときは、数値に変換される。

(B)option

数値または文字列で指定する。

指定形式	値	意味
数値指定	0x01ビット	0:バイト単位での範囲指定(省略時) 1:区切り文字で区切られた項目指定
	0x02ビット	0:データをソート対象とする 1:連想配列のキー値＋データ値をソート対象とする
	0x04ビット	1:連想配列のキー値のみをソート対象とする
文字列指定	**F**ixed または **R**anged	バイト単位での範囲指定(省略時)
	Item[=X] または **C**olumn[=X]	区切り文字で区切られた項目指定 X:半角１バイトの区切り文字 省略時の区切り文字は、空白文字またはカンマ
	Option=opt	区切り文字に関するオプション
	Key[=**O**nly]	連想配列のキー値とデータ値を区切り文字で連結したものをソート対象とする。 **O**nlyが指定されたときは､キー値のみをソート対象とする。

(注)文字列指定時は、先頭文字のみで判定する。上記以外のときは数値に変換する。

区切り文字に関するオプション(opt)
0x01=1:'#'以降を無視する
0x02=1:２ワード目が、'='のときは無視する。１ワード目のときは、ＮＵＬＬ値とみなす。
0x04=1:','も区切りとする。','が現れた時点で0x12=ONの指定を無効にする。
0x10=1:２ワード目が、':'or':='のときは無視する。１ワード目のときは、ＮＵＬＬ値とみなす。
0x20=1:'['と']'を引用符と同様に扱う(IPV6対応)
0x40=1:0x12=1のとき、'='or':'or':='が１ワード目のときには、１ワード目として設定する。
0x80=1:0x04=1のとき、','が現れた時点で0x12=ONの指定を無効にしない
0x010000=1:引用符の中の連続する２つの引用符を１つにしない

(注)区切り文字が指定されたときは、0x02,0x04,0x10,0x40,0x80 は、0となる。

(C)ソート位置指定のリスト

範囲指定: pos1, len1, order1, pos2, len2, order2, ...

項目指定: col1, order1, col2, order1, ...

posは先頭を１バイト目と数える。

lenはバイト単位で数える。

len省略時は、posから末尾までが範囲となる。

colは先頭を１項目目と数える。

order は、数値または文字列で指定する。

文字列のときは、各キーワードを空白文字で区切って指定する。

指定形式	値	意味
数値指定	0x01ビットが、0	昇順(省略時)
	0x01ビットが、1	降順
	0x02ビットが、0	文字列として比較する(省略時)
	0x02ビットが、1	数値として比較する
文字列指定	**A**sc	昇順(省略時)
	Desc	降順
	Char または **S**tring	文字列として比較する(省略時)
	Number または **I**nteger	数値として比較する Iのときは、整数値として比較する

(注)文字列指定時は、先頭文字のみで判定する。上記以外のときは数値に変換する。

(4)一般規則

(A)array_dに配列名を指定したときは、配列の先頭要素から、ソート結果が格納される。
数値が指定されたときは、内部番号変数の$(指定数値)の要素から、ソート結果が格納される。
配列には、パラメータ変数または検索変数へのMAPPEDARRAYは、指定できない。

(B)array_sに配列名を指定したときは、配列の先頭要素からが、ソート対象データとなる。
数値が指定されたときは、内部番号変数の$(指定数値)の要素からが、ソート対象データとなる。

(C)option以降が省略された場合は、全データが昇順のソート対象となる。

(D)数値変換は、先頭空白文字をスキップし、整数または浮動小数点形式以外の文字が現れた時点で終了し、それまでの変換結果が数値として扱われる。変換エラーのときは、０と見なされる。

(E)ソート対象データの属性が数値であり、ソート位置が先頭のみで数値で比較するときは、数値データがそのままソートに使われる。その他のときは一旦文字列に変換され、ソート位置指定にしたがって変換されソートに使われる。
ただし、ソート結果には元のデータが返される。

8.113. ＳＱＲＴ

(1)機能

平方根を求める。

(2)一般形式

```
ret  = SQRT(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが浮動小数でないときは、浮動小数に変換される。

(B)xが負のときは、エラー。

8.114. ＳＲＡＮＤ１（ＳＲＡＮＤ４８）

(1)機能

乱数のSEED(種)を設定する。

(2)一般形式

```
ret  = SRAND1(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xが整数型でないときは、整数型に変換される。

8.115. ＳＴＲＩＮＧＳ

(1)機能

文字列を繰り返す

(2)一般形式

```
result  = STRINGS(var, repeat) ;
```

(3)構文規則

なし

(4)一般規則

(A)varがNULL値のとき、または、repeatが０以下のときは、NULL値を返す。

8.116. ＳＵＢＳＴＲ（ＭＩＤ）

(1)機能

文字列を切り出す。

(2)一般形式

result = SUBSTR(var, [pos], [len]) ;

(3)構文規則

文字列演算式の第一項～第三項がそれぞれ第１パラメータ～第３パラメータに対応する。

(4)一般規則

(A)varが文字属性のときは、スクリプトの仕様で、posとlenの単位が変わる。

旧仕様：バイト単位

新仕様：文字単位

(B)その他は、文字列演算子を参照。

8.117. ＳＵＢＳＴＲＢ（ＭＩＤＢ）

(1)機能

バイト単位で文字列の切り出し。

(2)一般形式

result = SUBSTRB(var, [pos], [len]) ;

(3)構文規則

文字列演算式の第一項～第三項がそれぞれ第１パラメータ～第３パラメータに対応する。

(4)一般規則

文字列演算子を参照。

8.118. ＳＹＳＬＯＧ

(1)機能

syslogにログを出力する

(2)一般形式

ret = SYSLOG(syspri,format[,var1,var2,...,var5]);

(3)構文規則

(4)一般規則

(A)

8.119. ＴＡＮ

(1)機能

tanを求める。

(2)一般形式

```
ret  = TAN(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.120. ＴＡＮＨ

(1)機能

tanhを求める。

(2)一般形式

```
ret  = TANH(x) ;
```

(3)構文規則

なし。

(4)一般規則

(A)xはラジアンで指定する。

(B)xが浮動小数でないときは、浮動小数に変換される。

8.121. ＴＩＭＥＳ

（１）機能

関数または式文字列を指定回数実行する。関数の返却値または式文字列実行結果を返す。

（２）一般形式

```
result = TIMES([num], [func, p1, p2, ...]) ;
result = TIMES([num], [式文字列 [,opt] ]) ;
```

（３）構文規則

関数(func)のパラメータ(p1,p2,...)は、関数に合わせて指定する。

（４）一般規則

（Ａ）回数(num)省略時は、$MAX_LOOP_WHILE回繰り返す。

（Ｂ）関数または式文字列を省略した場合は、何もしないで、NULL文字を返す。

（Ｃ）式文字列を指定した場合は、EVAL()関数と同じ。

8.122. ＴＯ

（１）機能

文字列の変換

（２）一般形式

```
result = TO(var [,start [, len]], trans) ;
```

（３）構文規則

(A)文字列演算式の第一項と第二項がそれぞれ第１パラメータと第２パラメータに対応する。

(B)start、lenについては、INSTR()と同じ。

（４）一般規則

文字列演算子を参照

8.123. ＴＯ＿ＢＵＬＫ

(1)機能

各型のデータをそのままバルク型に変換し、バイト単位で切り出す。

(2)一般形式

```
result  = TO_BULK(var, [pos], [len], [opt]) ;
```

(3)構文規則

カンマ以降のパラメータを全て省略する場合は、カンマを省略可能。

(4)一般規則

(A)切り出しの規則は、SUBSTRBと同じ。

(B)以下のように変換される。

データ型	opt	変換
数値	0 または、省略	ネットワーク・バイト・オーダーとなる
	1	ホスト・バイト・オーダーとなる
文字	0 または、省略	変換しない
	1	１６進文字列とみなし、２バイトづつ数値に変換する
ＢＵＬＫ	無視	変換しない

8.124. ＴＯ＿ＢＵＬＫＳ

(1)機能

各型のデータをそのままバルク型に変換し、連結する

(2)一般形式

```
result  = TO_BULKS(var1[, var2, ...]) ;
```

(3)構文規則

(4)一般規則

(A)数値データは、ネットワーク・バイト・オーダーとなる。

(B)文字型とバルク型はそのまま使われる。

8.125. ＴＯ＿ＮＵＭＢＥＲ

(1)機能

数値への変換

(2)一般形式

```
num  = TO_NUMBER(var, attr, [size], [scale], [opt]) ;
```

(3)構文規則

(A)attrは、以下の値を指定する。

整数　　：２（$BIN[ARY],$INT[EGER]）

２進浮動小数点数　：３（$FLO[AT],$DOU[BLE]）

１０進浮動小数点数：４（$DEC[IMAL]）

(4)一般規則

(A)size,scaleは、１０進浮動小数点数のときのみ有効。

sizeを指定したときは、１０進固定小数点数に変換する。

(B)optは、

0x01：数字列の後ろに数字(.+EDを含む)以外があってもエラーとしない。

0x02：数字列中のカンマを無視する。（１０進浮動小数点数のみ）

8.126. ＴＲＩＭ

(1)機能

文字列の前後の空白文字を削除する。

(2)一般形式

```
result  = TRIM(var [,opt]) ;
```

(3)構文規則

(4)一般規則

(A)optを省略したときは、文字列の前後の空白文字を削除する。

(B)opt<=1を指定したときは、文字列の後の空白文字を削除する。

(C)opt>=1を指定したときは、文字列の前の空白文字を削除する。

8.127. ＵＮＬＩＮＫ

(1)機能

ファイルを削除する。

(2)一般形式

```
ret  = UNLINK(fname) ;
```

(3)構文規則

(4)一般規則

8.128. ＵＮＳＥＴＥＮＶ

(1)機能

環境変数を削除する。

(2)一般形式

```
ret  = unsetenv(name) ;
```

(3)構文規則

なし。

(4)一般規則

(A)本設定は、当該セッションの間のみ有効となり、変更された値は、セッション終了時に元に戻る。

(5)戻り値

0: 正常。-1: エラー。

8.129. ＸＨＡＳＨ

(1)機能

ハッシュ処理を行う。

(2)一般形式

(A)新規にハッシュ表を作成する

```
hp    = XHASH('New', $sKeyLen, $lMaxReg, [$lPreReg], [$DataFlag]) ;
    hp       : = 0 : エラー
               <>0 : ハッシュ表へのポインタ
    $sKeyLen : キー定義長（バイト）
               >0：指定の固定長キー。内容は任意。キー定義長は 1 ～ 32,759 バイト。
                   キーのデータ型が文字とＢＵＬＫ以外は、データ長以上を指定すること。
                   データ長については、付録「データ属性の値」を参照。
               =0：NULL終端文字列のキー。
    $lMaxReg : 最初に取られるエントリ数。(>2)
    $lPreReg : 第一割り当て数。0 か、省略時は自動計算。
    $DataFlag: >0 のとき、データを保存。<=0 か、省略のとき、データを保存しない。
```

(B)ハッシュ表を削除する

```
ret   = XHASH($hp, 'Free') ;
    ret      : 常に 0
    $hp      : ハッシュ表へのポインタ
```

(C)登録/検索/削除

```
index = XHASH($hp, $Func, $cKey, [$Data]) ;
    index    : < 0 : エラー
               = 0 : 空きなし
               > 0 : ハッシュされたエントリへのインデックス。
    $hp      : ハッシュ表へのポインタ
    $Func    : コマンド
               'S' : 登録
               'R' : 検索
               'D' : 削除
    $cKey    : ハッシュ・キー
```

キーには、一般データを指定できる。
固定長キーで、
a)ハッシュ・キー長＞キー定義長のとき、
・文字、ＢＵＬＫ属性：余った部分は切り捨てられる。
・その他のデータ属性：エラーとなる。
b)ハッシュ・キー長＜キー定義長のとき、不足部分は半角スペースが入る。

$Data : 登録のとき、指定されたデータを登録する。
検索、削除のとき、以下の指定で変数にデータを返却する
整数値 : 整数値を番号とする内部番号変数が対象
配列変数名 : 配列変数名[0]が対象

(D)チェック

```
ret   = XHASH($hp, 'K', $index, [$cKey], [$Data]) ;
```

ret : ＜ 0 : エラー
= 0 : 未使用
＞ 0 : 使用中（ハッシュされたエントリへのインデックス）
$hp : ハッシュ表へのポインタ
$index : チェックするエントリへのインデックス
$cKeyRet : ハッシュ・キーを返却する以下の変数指定
整数値 : 整数値を番号とする内部番号変数が対象
配列変数名 : 配列変数名[0]が対象
$DataRet : データを返却する変数指定
指定方法は、$cKeyRetと同じ

(E) 登録数/登録インデックスの最大値

```
ret   = XHASH($hp, $Func) ;
```

ret : ＜ 0 : エラー
その他 : 登録数/登録インデックスの最大値
$hp : ハッシュ表へのポインタ
$Func : コマンド
'U' : 登録数
'M' : 登録インデックスの最大値

(3)構文規則

(4)一般規則

(A)

9. 付録

9.1. ＳＱＬコマンドの処理番号

表１１．１－１に示す。

表１１．１－１　ＳＱＬコマンドの処理番号

項番	処理番号	処　　理　　内　　容
1	0	何もしない
2	1	START TRANSACTION（更新開始）　を発行する
3	2	COMMIT WORK　（実更新）　を発行する
4	3	ROLLBACK WORK　（更新の取消）　を発行する
5	4	何もしない
6	5	START TRANSACTION（更新開始）　を強制発行する
7	6	COMMIT WORK　（実更新）　を強制発行する
8	7	ROLLBACK WORK　（更新の取消）　を強制発行する

9.2. 実行時オプション

表１１．３－１　実行時オプション（１／２）

オプション番号	オプション値の意味(デフォルト値は、0x00)
1	変数値未設定時の処理 0x01 ： 配列要素のとき、NULL値とする。 0x02 ： 配列以外の変数のとき、NULL値とする。 これを設定すると、自動で0x01が設定される。 0x04 ： 連想配列要素を未設定にする。
2	名称に使用できる文字と文字列定数の引用符に使用する文字 ~~0x01 ： スペースと非表示文字以外を使用可能とする。~~ ~~0x02 ： 数字で始まる変数名を許す。~~ 0x04 ： １重引用符(')と２重引用符(")の機能を交換する。
3	テキスト出力時の改行コードを制御する 0x01 ： LFを出力しない 0x02 ： CRを出力する 0x04 ： テキスト中の改行コードも制御する
4	式がないときの処理 0x00 ： ワーニング終了(ret=100)、エラーメッセージあり 0x01 ： エラーメッセージなしビット 0x02 ： 正常終了ビット 0x04 ： エラー終了ビット
5	要素なしのリストの扱い 0x00 ： ＮＵＬＬ値とする 0x01 ： NULLリストとする
6	インポートするスクリプトの追加位置を指定する 0x00 ： 末尾に追加する 0x01 ： 先頭に挿入する
7	エラー時の処理 0x00 ： 当該手続きをエラー終了する 0x01 ： エラー行出力を行わない 0x02 ： エラーを無視して処理を継続する 0x04 ： 当該セッションをエラー終了する
8	ユーザ定義関数のサーチ順位 0x00 ： 定義済み関数より先にサーチする。 0x01 ： 定義済み関数の後にサーチする。 0x02 ： 手続き／関数の入れ子をサポートする。 クラスを使用するときは、自動で設定される。
9	入力構文の文字コード 0x00:引用符外側の英数記号の倍角文字は半角文字に変換する。 0x01:全ての英数記号の倍角文字は半角文字に変換する。 0x02:全て変換しない。
１０	デフォルトの変数種別 0x00 ： ローカル変数とする。 0x01 ： スクリプト変数とする。

表１１．３－１　実行時オプション（２／２）

オプション番号	オプション値の意味(デフォルト値は、0x00)
１１	１０進浮動(固定)小数点数の０表示 0x00 ： 少数点以下がないときに少数点以下を表示しない。 0x01 ： １未満のときに最初の０を表示しない。 0x02 ： 少数点以下がないときに少数点を表示する。 0x04 ： 少数点以下がないときに少数第１位の０を表示する。 0x20 ： ３桁毎にカンマを付ける。
１２	１０進固定小数点数の位取り未満の処理 0x00 ： 四捨五入 0x01 ： 切捨て 0x02 ： 切上げ
１３	ＰＲＩＮＴ文での配列、構造体出力形式 0x00 ： 詳細情報を出力しない 0x01 ： 詳細情報を出力する
１４	メッセージの言語種別 ０ ： 日本語 １ ： 英語 ２ ： その他
１５	配列の開始インデックスとデータの配置 0x00 ： 開始インデックスは、０ データの配置は、C言語形式 0x01 ： 開始インデックスは、１ 0x02 ： データの配置は、FORTRAN形式
１６	１０進浮動(固定)小数点数のオーバーフロー、アンダーフロー時の処理 0x00 ： オーバーフロー時、エラー終了。メッセージを出力する。 アンダーフロー時、続行。 0x01 ： オーバーフロー時、続行。 0x02 ： オーバーフロー時、メッセージを出力する。 0x04 ： アンダーフロー時、エラー終了。 0x08 ： アンダーフロー時、メッセージを出力する。
１７	１０進浮動小数点数になる場合の数字列の変換 0x00 ： １０進浮動小数点数に変換 0x01 ： ２進浮動小数点数に変換 0x10 ： 数字列の後ろに数字以外があってもエラーとしない。 0x20 ： カンマを無視する。
１８	サーチ開始位置指定時の返却値 0x00 ： INSTR,INLIKE,REPLIKEで、指定文字列の先頭を基準とする。 0x01 ： INSTRにおいて、サーチ開始位置を基準とする。 0x02 ： INLIKEにおいて、サーチ開始位置を基準とする。 0x04 ： REPLIKEにおいて、サーチ開始位置以降を返却する。

9.3. データ属性の値

データＩＤおよびデータ型の値を以下に示す。

表11.4-1　データＩＤ

項番	データＩＤ	値	データ長(バイト)
1	一般データ	’ ’	テータ型による
2	配列名	一般配列:’R’、MAPPED配列:’A’	３２
3	リスト	’L’	３２
4	構造体名	’T’	３２
5	構造体定義名	’P’	３２
6	関数名	’F’	３２
7	クラス名	’C’	３２
8	インスタンス名	’I’	３２
9	メソッド名	’M’	３２
１０	手続き名	’0’	３２

表11.4-2　一般データのデータ型とデータ長

項番	データ型	値	データ長(バイト)
1	文字	1	実データ長
2	整数	2	4
3	２進倍精度浮動小数点数	3	8
4	１０進小数点数	4	６４
5	ｂｕｌｋ	5	実データ長
6	ポインタ(未サポート)	6	4
7	バリアント(注)	7	0

(注)データ未設定時のみ存在し、データ設定後は、そのデータ型になる。

9.4. 演算子の順位

表１１．５－１　演算子の順位

順位	演算子
１	．（　）　［　］　｛　｝　後置演算＋＋　－－
２	＊＊
３	単項演算子　！　～　－　＋　＋＋　－－　＊
４	キャスト
５	＊　／　％
６	＋　－
７	＜＜　＞＞
８	文字列演算子　CONCAT　SUBSTR　REP　CONDAS　TO　IS
９	＜　＞　＜＝　＜＝
１０	＝＝　！＝
１１	＆
１２	＾
１３	｜
１４	＆＆
１５	｜｜
１６	？：
１７	＝ += -= *= /= %= <<= >>= &= ^= |= &+=
１８	,

（注）ピリオドはドット式で使われるとき。

9.5. 変数、手続きまたは関数のサーチ順

（1）変数のサーチ順

スコープ指定がないときは、以下の順にサーチする。

ローカル変数 → スクリプト変数 → 外部変数

手続きまたは関数の入れ子をサポートするときは、ローカル変数は、以下でサーチする。

実行済みの手続きまたは関数を実行の逆順に構造上の上位に向かってサーチする。、

（2）手続きまたは関数のサーチ順

（A）手続きまたは関数の入れ子をサポートしないとき

スクリプトの先頭から、第一レベルにある手続きまたは関数をサーチする。

（B）手続きまたは関数の入れ子をサポートするとき

手続きまたは関数を呼び出している手続きまたは関数の中 →上位の手続きまたは関数の中 → その上位の手続きまたは関数の中 ・・・

9.6. デバッグ形式での出力

（1）一般形式

（A）内容

ログヘッダ	出力元が設定したデータ	情報構造体の内容

データＩＤまたはデータ型による個別情報出力

（B）ログヘッダ

標準出力形式

```
[coal]
```

ファイル出力形式

```
yyyy/mm/dd hh:mm:ss coal/<ソースファイル名>(<ソース行数>):
```

(注)<>は、これで囲まれた情報が出力されることを示す。

（C）出力元が設定したデータ

出力対象となった最上位の変数のとき

```
式=
```

（2）個別情報

（A）式の並びまたはデータ並び式

並びの最後の式の値を一般形式で出力し、次に、並びの情報を" PARMINFO2:"を「出力元が設定したデータ」として一般形式で出力する。

（B）配列

情報構造体の内容

```
pInfo=%08x id=%c attr=%d scale=0x%02x dlen=%d len(pScCT)=%08x hlen(gid)=%d
 aux=0x%02x 0x%02x auxlen=%04x %08x data=%08x
```

配列の情報

```
(VarIndex=%08x Attr=%d %d %d %d size=%d index=%d[%d,%d,%d] xhp=%08x
  (_xhash Id=%c%c keylen=%d max=%d pre=%d ha=%08x next=%08x datlen=%d dreg=%08x))
 varnam=[配列名]
```

(注)"(_xhash ・・・ dreg=%08x)"は、連想配列のとき出力する。

（C）その他

「情報構造体の内容」は共通で、その後に個別情報が出力される。

情報構造体の内容

```
pInfo=%08x id=0x%02x[%c] attr=%d scale=0x%02x code=%d dlen=%d len(gid)=%d hlen=%d pos=%d(%08x) aux=0x%02x 0x%02x auxlen=%04x %08x data=%08x
```

（a）構造体または構造体定義

ヘッダ

```
size=%d data=0x%08x ntype=%d vname=0x%08x pType=0x%08x varnam=[<オブジェクト名>]
```

メンバー情報（メンバー数分出力する）

```
"vnlen=<メンバー名長> <メンバー名>: "を「出力元が設定したデータ」として、一般形式で出力
```

（b）リスト

```
 rb_bfsize=%d rb_max=%d rb_num=%d rb_used=%d rb_pos=%d rb_raddr=0x%08x rb_waddr=0x%08x rb_wpriv=0x%08x rb_cur=0x%08x
```

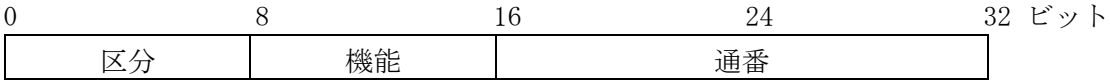
（c）その他

データ型	単純	範囲
データへのポインタがNULL	NULL	なし
CHAR	[%s]	[%s]..[%s]
BIN	%d	%d..%d
FLOAT	%e	%e..%e
DEC	10進表示	10進表示..10進表示
BULK または PARMINFO2の２要素目	先頭32バイトの16進表示	なし
その他	**INVALID**	なし

9.7. 例外の種類

9.7.1. 例外番号の構成

（1）値の構成



（2）名称の構成

区分名称__[機能名称__[個別名称__]]ＥＸＣＥＰＴＩＯＮ

9.7.2. 区分

No.	区分		内容
	名　　称	値	
1	ＩＳ	1	検査
2	ＴＯ	2	変換
3	ＣＭＰＲ	3	比較
4	ＳＴＲＩＮＧ	4	文字列操作
5	ＭＡＴＨ	5	数値演算
6	ＦＩＬＥ	6	ファイル
7	ＬＯＧ	7	ログ
8	ＳＱＬ	8	ＳＱＬ
9	ＣＯＭＭ	9	通信
10	ＵＳＥＲ	0x7c	ユーザ
11	ＳＹＳＴＥＭ	0x7d	システム
12	ＥＴＣ	0x7e	その他
13	ＡＬＬ	0x80	全ての例外を表す。 例外の値は、これとのＯＲを取った値となる。

9.7.3. 機能

（5）ＭＡＴＨ

No.	機能		個別		内容
	名　　称	値	名　　称	値	
	ＣＯＭＰ	1	［ ＥＲＲＯＲ ］	0	演算エラー全般
			ＤＥＶＩＤＥ	1	０割
	ＥＴＣ	254	［ ＥＲＲＯＲ ］	0	その他エラー全般

（6）ＦＩＬＥ

No.	機能		個別		内容
	名　　称	値	名　　称	値	
	ＯＰＥＮ	1	［ ＥＲＲＯＲ ］	0	オープン・エラー全般
			ＮＯＴＦＯＵＮＤ	2	file not found
	ＣＬＯＳＥ	2	［ ＥＲＲＯＲ ］	0	クローズ・エラー全般
	ＲＥＡＤ	3	［ ＥＲＲＯＲ ］	0	入力エラー全般
			ＥＮＤ	1	end of file
	ＷＲＩＴＥ	4	［ ＥＲＲＯＲ ］	0	出力エラー全般
	ＥＴＣ	254	［ ＥＲＲＯＲ ］	0	その他エラー全般

9.8. スクリプトの例

9.8.1. Hellow World

```
proc main;
        print 'Hellow World.';
        return 0;
end proc;
```

9.8.2. クロージャ

test_class3.cl

```
//
Class Counter;
        int i=0;
        f = _count;
        func Counter;
                return f;       // クラス変数を使って関数を返す。
        end func;
        func Counter(n);
                i = n;
                return f;
        end func;
        func Counter(x,y);
                i = x + y;
                return My._count;       // Myは、本クラスを示す。
        end func;
        func _count();
                return ++i;
        end func;
end class;

proc main;
        c = new(Counter);
        print c();
        print c();
        d = new(Counter,10);
        print d();
        print d();
        print c();
        x = new(Counter,100,1);
        print x();
        print x();
        return ERROR;
end proc;
```

実行結果

```
145:/home/coal/test>coal test_class3
c()=1
c()=2
d()=11
d()=12
c()=3
x()=102
x()=103
```

9.8.3. 構造体を使った擬似クラス

test_class3.cl

```
//
define type struct Greeter
         variant salute
        ,variant name           // variantは、データ型指定なしと同じ。
;
define var g as Greeter;        // define var g,h as Greeter;
define var h as Greeter;        // gとhは、まとめて定義可能。

func _new (g,name);             // gには、構造体への参照が渡される。
        g.name = name;
//      g.salute = _salute;             // 直接関数名を指定しても同じ。
        g.salute = (FUNC)'{func _salute;// {}で囲った文字列を関数化して指定可能。
        print My.name&'' World!!'';     // 文字定数中の引用符は２つ続けて指定する。
        return 0;
end func;}';
        return 0;
end func;
/*
func _salute;
        print My.name&' World!!';       // Myは、構造体で修飾したときのみその構造体を示す。
        return 0;
end func;
*/
proc main;
        _new(g,'Hellow');
        g.salute();
        _new(h,'Good night');
        h.salute();
        return ERROR;
end proc;
```

実行結果

```
146:/home/coal/test>coal test_struct_class
My.name&' World!!'="Hellow World!!"
My.name&' World!!'="Good night World!!"
```

(注)printは、対象となった式=結果の形式で出力する。(ただし、定数は値のみを出力する。)