

AJAN

拡張コマンド  
リファレンス

## 目 次

<b>第 1 章</b>	<b>はじめに</b>	<b>8</b>
<b>第 2 章</b>	<b>機能説明</b>	<b>9</b>
2.1	Python 連携の基本 .....	9
2.2	オーディオ入出力の基本 .....	11
2.2.1	オーディオ入出力できない時の確認事項 .....	12
2.2.2	途切れなくオーディオ入出力を行う方法 .....	13
2.3	グローバル共有の基本 .....	16
2.3.1	読み書きを行う変数名の注意とお願い .....	18
2.3.2	配列の読み書きを行う .....	18
2.3.3	連想配列の読み書きを行う .....	20
2.3.4	両端キューに追加・取り出しを行う .....	22
2.3.5	型と存在の確認と削除 .....	24
2.3.6	排他処理をかける .....	25
2.3.7	別コンピュータに対して読み書きする .....	26
2.3.8	通信を暗号化しつつ、別コンピュータに対して読み書きする(ssh 版) .....	27
2.3.9	別プログラムにメッセージを通知・受信する .....	29
2.3.10	構造体を扱う際の注意 .....	31
2.3.11	通信回線切断などのトラブル時の注意 .....	32
2.3.12	データベースの個数を増減したい時 .....	34
2.3.13	グローバル共有のデータを全てクリアしたい .....	34
2.3.14	グローバル共有のデータを別のグローバル共有に複製したい .....	35
2.3.15	複数のサーバを起動したい .....	36
2.3.16	グローバル共有のデータをバックアップする .....	39
2.3.17	グローバル共有モニタの紹介 .....	40
<b>第 3 章</b>	<b>リファレンス</b>	<b>41</b>
3.1	コマンド一覧 .....	41
3.2	数値・文字列に関する関数・命令 .....	47
3.2.1	ARRAY2BINBLK\$ .....	47
3.2.2	ARRAY2STR\$ .....	48
3.2.3	BINBLK2ARRAY .....	49
3.2.4	CBOOL .....	51
3.2.5	CDBL .....	52
3.2.6	CINT .....	52
3.2.7	CLNG .....	53
3.2.8	CSNG .....	53
3.2.9	CLOCK2DATETIMESTR\$ .....	54
3.2.10	CONVBIN .....	55
3.2.11	CONVPHY .....	57
3.2.12	CONVUNIT .....	58

3.2.12.1 <単位変換名一覧> .....	58
3.2.13DATEADD .....	59
3.2.14DATETIMESTR2CLOCK .....	60
3.2.15DAY .....	61
3.2.16FORMAT\$ .....	62
3.2.16.1 <FORMAT\$の数値表示書式表> .....	62
3.2.16.2 <FORMAT\$の日付時刻表示書式表> .....	63
3.2.16.3 <FORMAT\$の文字列表示書式表> .....	63
3.2.17FORMATDATETIME\$ .....	65
3.2.18HASH\$ .....	66
3.2.19HOUR .....	67
3.2.20ICONV\$ .....	68
3.2.21JOIN\$ .....	69
3.2.22LCASE\$ .....	69
3.2.23LEFT\$ .....	70
3.2.24LEFTB\$ .....	70
3.2.25MATH ISEQ .....	71
3.2.26MINUTE .....	72
3.2.27MKUUID\$ .....	72
3.2.28MONTH .....	73
3.2.29PASSWORD_HASH\$ .....	74
3.2.30PASSWORD_VERIFY .....	74
3.2.31RIGHT\$ .....	75
3.2.32RIGHTB\$ .....	75
3.2.33SECOND .....	76
3.2.34STR_BASE64_DECODE\$ .....	77
3.2.35STR_BASE64_ENCODE\$ .....	77
3.2.36STR_DUMP\$ .....	78
3.2.37STR_AND\$ .....	79
3.2.38STR_OR\$ .....	79
3.2.39STR_XOR\$ .....	79
3.2.40STR_IMP\$ .....	80
3.2.41STR_EQV\$ .....	80
3.2.42STR_REPEAT\$ .....	80
3.2.43STR_HAN2ZEN\$ .....	81
3.2.44STR_ZEN2HAN\$ .....	82
3.2.45STR_HIRA2KATA\$ .....	83
3.2.46STR_KATA2HIRA\$ .....	83
3.2.47STR_STARTSWITH .....	84
3.2.48STR_ENDSWITH .....	84
3.2.49STR_REMOVEPREFIX\$ .....	85
3.2.50STR_REMOVESUFFIX\$ .....	85
3.2.51STR_VALIDUTF8 .....	86
3.2.52STR_ISALNUM .....	86
3.2.53STR_ISALPHA .....	87
3.2.54STR_ISCNTRL .....	87

3.2.55	STR_ISDIGIT .....	88
3.2.56	STR_ISGRAPH.....	88
3.2.57	STR_ISLOWER.....	89
3.2.58	STR_ISPRINT.....	89
3.2.59	STR_ISPUNCT.....	90
3.2.60	STR_ISSPACE.....	90
3.2.61	STR_ISUPPER.....	91
3.2.62	STR_ISXDIGIT .....	91
3.2.63	STR_ISASCII.....	92
3.2.64	CHRTYPE.....	93
3.2.64.1	<情報 ID : "type" の値> .....	94
3.2.64.2	<情報 ID : "break_type" の値> .....	94
3.2.65	STR2ARRAY.....	96
3.2.66	STRDELBS.....	97
3.2.67	STRINSBS.....	98
3.2.68	STRREVERSE\$.....	99
3.2.69	STRREVERSEBS.....	99
3.2.70	TRUNC.....	100
3.2.71	UCASE\$.....	100
3.2.72	UUIDCOMP.....	101
3.2.73	YEAR .....	101
3.3	配列演算に関する関数・命令 .....	102
3.3.1	DIMAVG.....	102
3.3.2	DIMMEDIAN .....	102
3.3.3	DIMMAX.....	103
3.3.4	DIMMIN.....	103
3.3.5	DIMSUM.....	104
3.3.6	DIMSTDEVP .....	104
3.3.7	DIMVARP.....	105
3.3.8	DIMVARS.....	105
3.3.9	DIMADD.....	106
3.3.10	DIMADD\$.....	107
3.3.11	DIMSUB.....	108
3.3.12	DIMDIV .....	109
3.3.13	DIMMOD .....	110
3.3.14	DIMMUL .....	111
3.3.15	DIMNOT .....	112
3.3.16	DIMAND.....	113
3.3.17	DIMOR.....	114
3.3.18	DIMXOR.....	115
3.3.19	DIMIMP .....	116
3.3.20	DIMEQV .....	117
3.3.21	DIMSHL.....	118
3.3.22	DIMSHR.....	118
3.3.23	DIMROL .....	119
3.3.24	DIMROR .....	119

3.3.25	DIMEQ .....	120
3.3.26	DIMNE .....	121
3.3.27	DIMLT .....	122
3.3.28	DIMGT .....	123
3.3.29	DIMLTE .....	124
3.3.30	DIMGTE .....	125
3.3.31	DIMIIF .....	126
3.3.32	DIMIIF\$ .....	127
3.3.33	DIMCINT .....	128
3.3.34	DIMCLNG .....	129
3.3.35	DIMCSNG .....	130
3.3.36	DIMCDBL .....	131
3.3.37	DIMCSTR\$ .....	132
3.3.38	DIMFIND .....	133
3.3.39	DIMGET .....	135
3.3.40	DIMGET\$ .....	136
3.3.41	DIMMAP .....	137
3.3.42	DIMMAP\$ .....	138
3.3.43	DIMREDUCE .....	139
3.3.44	DIMSET .....	140
3.3.45	NUM2BIT .....	141
3.3.46	BIT2NUM .....	141
3.3.47	ONEDIM SORT .....	142
3.3.48	ONEDIM SORT\$ .....	142
3.3.49	TWODIM FILTER .....	143
3.3.50	TWODIM FILTER\$ .....	144
3.3.51	TWODIM JOIN .....	145
3.3.52	TWODIM JOIN\$ .....	146
3.3.53	TWODIM EXISTS .....	147
3.3.54	TWODIM EXISTS\$ .....	148
3.3.55	TWODIM NOT EXISTS .....	149
3.3.56	TWODIM NOT EXISTS\$ .....	150
3.3.57	TWODIM SORT .....	151
3.3.58	TWODIM SORT\$ .....	152
3.3.59	TWODIM TRANSPOSE .....	153
3.3.60	TWODIM TRANSPOSE\$ .....	153
3.4	Python 連携に関する関数・命令 .....	154
3.4.1	PYOBJ CREATE CODE .....	154
3.4.2	PYOBJ CALL FUNCTION .....	155
3.4.3	PYOBJ CALL METHOD .....	156
3.4.4	PYOBJ CLOSE .....	158
3.5	オーディオ入出力を使ったアナログ変換に関する関数・命令 .....	159
3.5.1	SNDOOPEN .....	159
3.5.2	SNDCLOSE .....	160
3.5.3	SNDREAD .....	161
3.5.4	SNDREADLEN .....	162

3.5.5	SNDWRITE .....	163
3.5.6	SNDWRITELEN.....	164
3.6	ファイル・フォルダに関する関数・命令 .....	165
3.6.1	CURDIR\$ .....	165
3.7	プロセスに関する関数・命令 .....	166
3.7.1	GETCOMMANDLINEARG\$ .....	166
3.7.2	GETSYSTEMINFO\$ .....	167
3.7.3	ENVIRON\$ .....	168
3.8	グローバル共有に関する関数・命令 .....	169
3.8.1	COMMON OPEN .....	169
3.8.2	COMMON CLOSE .....	171
3.8.3	COMMON FREEFILE .....	171
3.8.4	COMMON ERASE .....	172
3.8.5	COMMON INPUT .....	173
3.8.6	COMMON INPUTR .....	174
3.8.7	COMMON READ .....	175
3.8.8	COMMON PRINT .....	176
3.8.9	COMMON WRITE .....	177
3.8.10	COMMON LOCK .....	178
3.8.11	COMMON UNLOCK .....	179
3.8.12	COMMON NOTIFY .....	180
3.8.13	ON COMMON CALL .....	181
3.8.14	COMMON ON / OFF .....	182
3.8.15	COMMON CDIM .....	183
3.8.16	COMMON GET_DICT_KEY\$ .....	184
3.8.17	COMMON HAS_DICT_KEY .....	185
3.8.18	COMMON LDICT .....	186
3.8.19	COMMON LDIM .....	186
3.8.20	COMMON UBOUND .....	187
3.8.21	COMMON VARTYPE .....	188
3.8.22	COMMON DEQUE PUSH .....	189
3.8.23	COMMON DEQUE LPUSH .....	190
3.8.24	COMMON DEQUE POP .....	191
3.8.25	COMMON DEQUE LPOP .....	192
3.8.26	COMMON DEQUE BACK .....	193
3.8.27	COMMON DEQUE FRONT .....	194
3.8.28	COMMON DEQUE LEN .....	195
3.8.29	COMMON VLIST\$ .....	196
3.8.30	COMMON SWAPDB .....	197
3.8.31	COMMON2 ERASE .....	198
3.8.32	COMMON2 CDIM .....	199
3.8.33	COMMON2 GET_DICT_KEY\$ .....	200
3.8.34	COMMON2 HAS_DICT_KEY .....	201
3.8.35	COMMON2 LDICT .....	202
3.8.36	COMMON2 LDIM .....	202
3.8.37	COMMON2 UBOUND .....	203

---

3.8.38COMMON2 VARTYPE .....	204
3.8.39COMMON2 DEQUE PUSH.....	205
3.8.40COMMON2 DEQUE LPUSH .....	206
3.8.41COMMON2 DEQUE POP.....	207
3.8.42COMMON2 DEQUE LPOP .....	208
3.8.43COMMON2 DEQUE BACK.....	209
3.8.44COMMON2 DEQUE FRONT.....	210
3.8.45COMMON2 DEQUE LEN .....	211
3.8.46COMMON2 VLIST\$ .....	212
3.9     その他に関する関数・命令 .....	213
3.9.1 QRCODE_GET .....	213
3.10    拡張コマンドのサブルーチン集 .....	215
3.10.1GET_OCR\$ .....	216
<b>第 4 章    サンプルプログラム</b> .....	<b>217</b>

---

4.1    サンプルプログラム .....	217
------------------------	-----

---

<b>第 5 章    索引</b> .....	<b>219</b>
<b>第 6 章    重要な情報</b> .....	<b>222</b>

---

## 第1章 はじめに

---

本ドキュメントは、AJANの拡張コマンドの説明を記載しています。

拡張コマンド以外のコマンド(標準コマンド、IO制御コマンドなど)は、別マニュアルを用意しているので、別途参照ください。

本ドキュメントでは、説明で表現している表記として下記のように定義します。

- ・ コマンドの書式の説明において、[ ]内の引数は省略できます。
- ・ 文字の大小について  
コマンドは大文字 / 小文字のどちらでも動作します。  
変数名は大文字 / 小文字も同じものとして扱われます。  
ファイルパス / ファイル名は大文字/小文字で区別されます。



本ドキュメント記載の、AJANはIoT用プログラミング言語です。  
Interface Linux System上でのみ動作可能です。



## 第2章 機能説明

### 2.1 Python連携の基本

AJANからPythonエンジンを内部で呼び出し、Pythonに処理を依頼し、結果を受け取る事ができます。

例えば、以下のコードを実行すると、下図のような円グラフが表示されます。

```
' ↓Pythonコード部
s$='' import sys
import numpy
import matplotlib
matplotlib.use('GTK3Agg')
import matplotlib.pyplot as plt

def pie_show(a, lbl):
    # 円グラフ
    a2 = numpy.array(a)
    plt.pie(a2, labels=lbl)

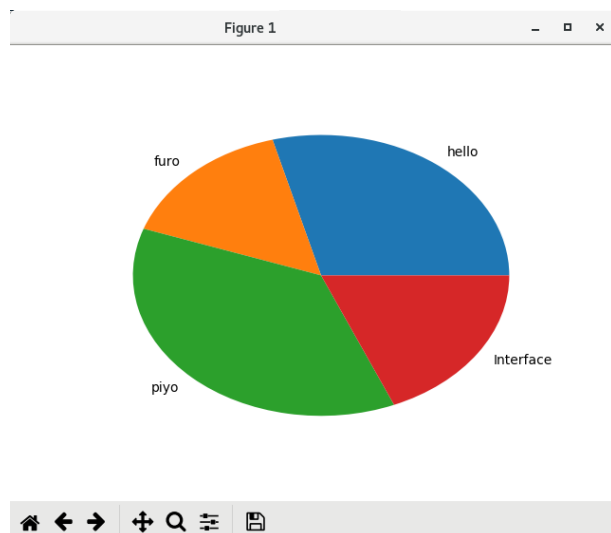
def show_wait(sec):
    # 指定秒数待って閉じる (showだと同期なので対策)
    plt.draw()
    plt.pause(sec)
    plt.close()
'''

' Pythonコードを読み取る
ID = PYOBJ CREATE CODE (S$)

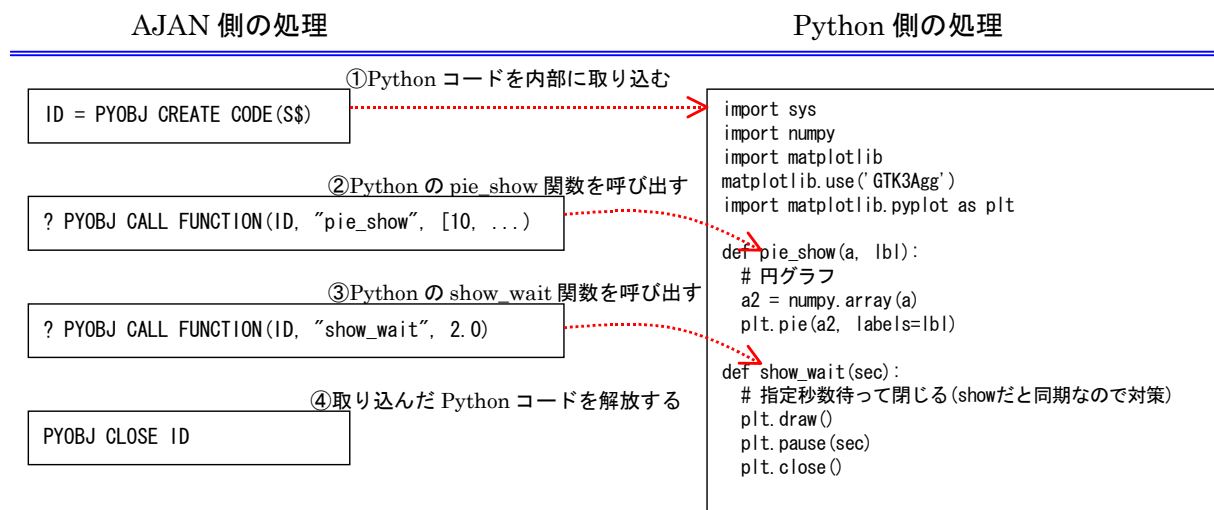
' 円グラフ (pie) の呼び出し
? PYOBJ CALL FUNCTION (ID, "pie_show", [10, 5.3, 12.6, 6.4], ["hello"; "furo"; "piyo"; "Interface"])
? PYOBJ CALL FUNCTION (ID, "show_wait", 2.0)

PYOBJ CLOSE ID
```

実行例：



このコードは、おおよそ以下のように動いています。



このように、Pythonと連携して処理することができます。

## 2.2 オーディオ入出力の基本

製品に搭載されているオーディオ入出力機能(マイク入力、ライン出力端子など)を、アナログ入出力機能に見立てて制御できます。



本機能を使用するには、オーディオ入出力を使って、実際に音が出たり、入力できる必要があります。

例えば、16kHzのサンプリング周波数、-1~+1の範囲のデータ値を、オーディオ入力したい場合、以下のように記述します。

```
' オーディオ入力をオープン(16kHzで入力)
SNDOPEN "?RATE=16000&FORMAT=FLOAT32LE" FOR INPUT AS #1

DIM BUF(511)
BUF = SNDREAD(1, 512) ' オーディオ入力から512件読み取る

' オーディオ入力をクローズ
SNDCLOSE #1
```

また、乱数を使って作ったサンプリングデータを、同じ条件でオーディオ出力したい場合、以下のように記述します。

```
' オーディオ出力をオープン(16kHzで入力)
SNDOPEN "?RATE=16000&FORMAT=FLOAT32LE" FOR OUTPUT AS #1

DIM BUF(511)
' 乱数を使ってサンプリングデータを作る
FOR I=0 TO LDIM(BUF)-1
  BUF(I) = RND()
NEXT I
SNDWRITE #1, BUF ' 作ったサンプリングデータ(512件分)を、オーディオ出力する

' オーディオ出力をクローズ
SNDCLOSE #1
```

## 2.2.1 オーディオ入出力できない時の確認事項

以下のように、オーディオ入出力を動かそうとしてもエラーになる場合、後述する確認事項を試してみてください。

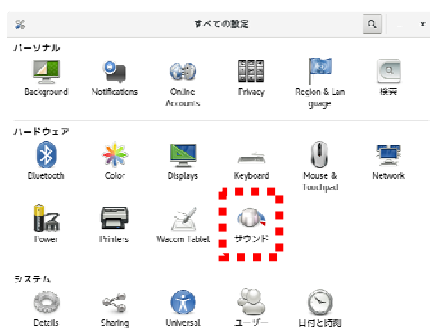
```
$ ./SNDREAD_SNDWRITE
Error! &H01000008: ファイルのオープンに失敗しました(オーディオデバイスを開けません((null), INPUT, 6,
接続拒否)) [/home/user/cajan/src/EXT001/src/samples/SNDREAD_SNDWRITE.AJN:line 9]
```

### ■ 確認① サウンドの入出力が有効か確認する

まず、サウンドの入出力機能が有効になっているか確認しましょう。

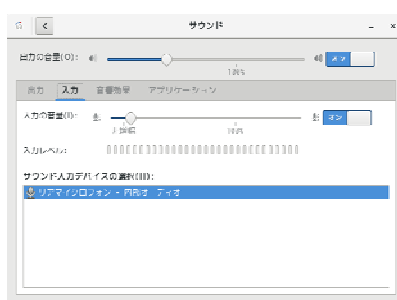
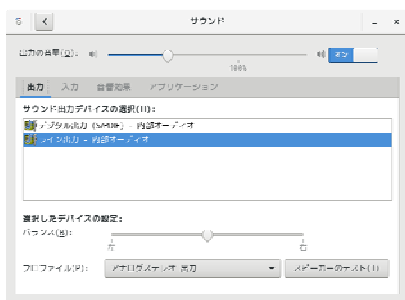
Linuxメニューのアプリケーション→システムツール→Preferences→設定を選択します。

「すべての設定」ダイアログが起動します。ここで、「サウンド」アイコンをクリックすると、サウンドの設定ダイアログが開きます。



「出力」タブがサウンド出力、「入力」タブがサウンド入力の設定です。

ここで、機能が「オン」になっている必要があります。「オフ」の場合「オン」に設定しなおしてください。



サウンド出力の場合、ダイアログ下部に「スピーカーのテスト」ボタンがありますので、これを選択してテストを行い、正しく音が聞こえてくるか確認します。

サウンド入力の場合、マイクなどと繋いで音声入力しつつ、入力レベルのゲージが変化する事を確認します。

## 2.2.2 途切れなくオーディオ入出力を行う方法

途切れなくオーディオ入出力を行うには、十分な入出力バッファを確保し、バッファに蓄えられたデータが途切れないように、入力ないしは出力を行う事が肝要です。

入出力バッファの指定は、「SNDOPEN」時に指定できます。

例えば、オーディオ出力を行う際、出力バッファ長を指定する TLENGTH と、出力開始するサイズ長を指定する PREBUF を用います。

```
' オーディオ出力時、出力バッファ長を20000、出力開始サイズ長を200とする
SNDOPEN "?RATE=1000&FORMAT=FLOAT32LE&TLENGTH=20000&PREBUF=200" FOR OUTPUT AS #1
```

上のコード例では、TLENGTH で指定した 20000バイト ほどの長さの出力バッファを用意し、PREBUF で指定した 200バイト ほどの長さの出力データが溜まる都度、オーディオ出力が行われる事になります。

(指定しない時、システムが自動的にバッファサイズを決定します)

この為、出力バッファにデータが残っている間に、次のデータを「SNDWRITE」で出力バッファにセットする事で、途切れなくオーディオ出力する事が可能となります。

「SNDWRITELEN」で、出力バッファの書き込み可能件数が得られます。

「SNDOPEN」直後、「SNDWRITELEN」で得られる値が、出力バッファが空時の値とみなせるので(正確には、OSが認識できないハードウェアが持つ出力バッファ分が別途ある)、この値にならないように、「SNDWRITE」で出力データを書き込んでいきます。

出力バッファに書き込むデータの余裕があれば、出力バッファにデータをセットした後、「SNDWRITE」は直ぐに戻ります。

しかし、出力バッファに余裕が無い場合、データを出力バッファにセットするまで、待機して戻りません。

これを応用して、データを逐次書き込む例を示します。

```
' オーディオ出力時、出力バッファ長を20000、出力開始サイズ長を200とする
SNDOPEN "?RATE=1000&FORMAT=FLOAT32LE&TLENGTH=20000&PREBUF=200" FOR OUTPUT AS #1

LIST ARY
' 出力する波形データを生成
ARY = CALC_CREATE_SINWAVE(1, 100.0, 400.0, 1000)

' バッファが空時の、出力バッファの書き込み可能件数
SZ_INIT = SNDWRITELEN(1)

' 初回の出力データ書き込み。これより出力開始
SNDWRITE #1, ARY
```

```
SZ_1 = SNDWRITELEN(1)
ASSERT SZ_1 > 0, "書き込み直後にバッファが0になるなら、TLENGTHが少なすぎる"

' 出力バッファが空になる手前のしきい値を決める
SZ_CHK = SZ_INIT - ((SZ_INIT - SZ_1) / 2)

DO WHILE TRUE
    SZ = SNDWRITELEN(1)
    IF SZ > SZ_CHK THEN
        ' 出力バッファにデータを書き込む
        SNDWRITE #1, ARY
    END IF
LOOP
```

オーディオ入力の場合は、入力バッファ長を指定する FRAGSIZE を用います。

```
' オーディオ入力時、入力バッファ長を10000とする
SNDOPEN "?RATE=16000&FORMAT=FLOAT32LE&FRAGSIZE=10000" FOR INPUT AS #1
```

上のコード例では、FRAGSIZE で指定した 10000バイト ほどの長さの入力バッファを用意し、オーディオ入力が行われる事になります。

(指定しない時、システムが自動的にバッファサイズを決定します)

入力バッファに蓄えられた件数は、「SNDREADLEN」で得られます。

「SNDREAD」で指定した件数のデータを抜き取るとき、入力バッファに件数以上のデータがあれば、データを抜き取って、関数は直ぐに戻ります。

入力バッファに件数未満のデータしか無い場合、関数は指定した件数のデータが得られるまで、待機して戻りません。

これを応用して、入力バッファに1000件のデータが溜まる都度、データを抜き取る例を示します。

```
' オーディオ入力時、入力バッファ長を10000とする
SNDOPEN "?RATE=16000&FORMAT=FLOAT32LE&FRAGSIZE=10000" FOR INPUT AS #1

LIST ARY
DO WHILE TRUE
    SZ = SNDREADLEN(1)
    IF SZ > 1000 THEN
        ARY = SNDREAD(1, SZ)      ' SNDREADLEN で得た件数分を抜き取る
    END IF
LOOP
```

「SNDREAD」で抜き取る件数は、「SNDREADLEN」で得られた件数を指定する事を推奨します。得られた件数以外の件数を抜き取ろうとした時、時折 若干長い待機待ちが発生する現象を確認しております。

2022/4 より、「SNDREAD」で抜き取る件数に負数を指定したとき、入力バッファにあるデータのみを抜き取り対象にする事ができるようになりました。

これを応用すると、データの抜き取り例は、以下のようになります。

```
' オーディオ入力時、入力バッファ長を10000とする
SNDOPEN "?RATE=16000&FORMAT=FLOAT32LE&FRAGSIZE=10000" FOR INPUT AS #1

LIST ARY
DO WHILE TRUE
  SZ = SNDREADLEN(1)
  IF SZ > 1000 THEN
    ARY = SNDREAD(1, -1)      ' 入力バッファの一塊分を抜き取る
  END IF
LOOP
```

## 2.3 グローバル共有の基本

グローバル共有は、アプリケーション同士で値をやり取りできる機能です。やり取りする値は一意であり、ネットワークをまたぐ事もできます。

この機能を使って、AJANプログラム同士が、任意の値をやり取りする事ができます。



グローバル共有機能は、内部で Redis( <https://redis.io> ) と呼ばれるソフトウェアが使用されています。

例えば、以下のプログラムを実行すると、CNTの値をグローバル共有に書き込みます。

```
' 簡単な書き込み事例
COMMON OPEN "127.0.0.1" AS #1
' CNTの値を初期化する
CNT = 123
' CNTの値を、グローバル共有に書き込む
COMMON PRINT #1, CNT

COMMON CLOSE #1
```

グローバル共有に書き込んだCNTの値を読み取るには、以下のように書きます。

```
' 簡単な読み込み事例
COMMON OPEN "127.0.0.1" AS #1
' CNTの値を読み取る
IF COMMON INPUTR(1, CNT) = 0 THEN
' 読み取り成功した
PRINT "CNT="; CNT
ELSE
' 読み取れなかった
PRINT "グローバル共有にCNTが存在しません"
END IF

COMMON CLOSE #1
```

グローバル共有に、CNTの値が存在する事が自明ならば、以下のようにも書けます。

```
' 簡単な読み込み事例
COMMON OPEN "127.0.0.1" AS #1
' CNTの値を読み取る
COMMON INPUT #1, CNT
' 読み取った値
PRINT "CNT="; CNT

COMMON CLOSE #1
```

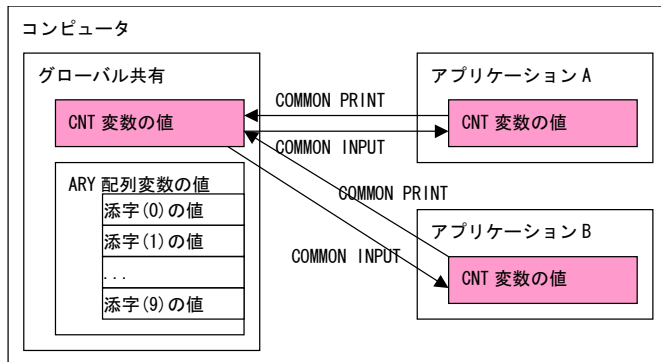
COMMON INPUTR 関数やCOMMON INPUT命令で、グローバル共有からCNT変数の値を取り出し、COMMON PRINT 命令で CNT変数の値をグローバル共有に格納します。

例えていうと、グローバル共有は、コンピュータに1つだけ存在する、データの格納庫のようなものといえます。



イメージを図にすると、以下のようなものです。

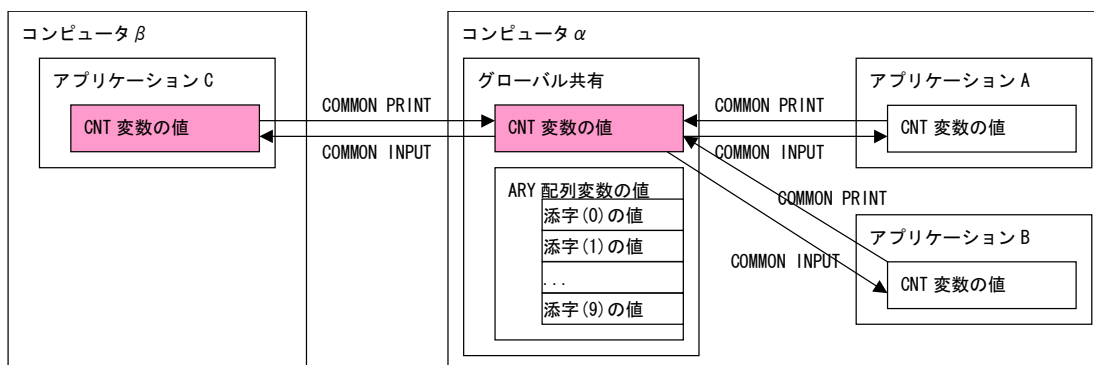
グローバル共有にある CNT変数の値と、自アプリケーションプログラムのCNT変数の値を、COMMON PRINT 命令とCOMMON INPUTR 関数、COMMON INPUT命令を使って、やり取りします。



イメージ的には、CNT変数という変数そのものを、グローバル共有という場所に格納し、取り出すという事です。

グローバル共有は、ネットワークを介して値をやり取りすることもできます。

例えば、以下のように、別のコンピュータにあるCNT変数の値を取り出す事が可能です。



グローバル共有で書き込んだ値は、アプリケーション終了後も残り続けます。

従って、上のプログラムを実行後、再び実行すると、前のCNTの値に対して読み書き操作が行われます。

このように、グローバル共有は、アプリケーション同士を連携させるための糊のような役割を持たせる事が可能です。

### 2.3.1 読み書きを行う変数名の注意とお願い

一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。

これらのグローバル共有の変数名は、弊社から提供するサービス、アプリケーション、ユーティリティ等で使用いたします。

お客様が間違えて該当する変数名を使用して衝突すると、誤動作の可能性があります。

### 2.3.2 配列の読み書きを行う

グローバル共有は、配列の読み書きができます。

以下に、配列の読み書き事例を示します。

```
' 簡単な配列の読み書き事例
COMMON OPEN "127.0.0.1" AS #1

DIM ARY(9)
ARY = [ 1; 2; 3; 4; 5; 6; 7; 8; 9; 10 ] ' 書き込む初期値を設定
' 配列ARYの値を書き込む
COMMON PRINT #1, ARY

ARY = [ 0; 0; 0; 0; 0; 0; 0; 0; 0; 0 ] ' 初期化
' 配列ARYの値を読み取る
COMMON INPUT #1, ARY

PRINT "配列ARY="; ARY

COMMON CLOSE #1
```

配列を扱う際に重要なのは、まず一旦全体をグローバル共有に書き込む事です。

いきなり読もうとしても、グローバル共有に値が書き込まれてないと、エラーとなります。

一旦 値を書き込むと、配列の添字を指定して、グローバル共有に対する部分読み書きが可能となります。

```
' 簡単な配列の部分読み書き事例
COMMON OPEN "127.0.0.1" AS #1

DIM ARY(9)

FOR I=0 TO UBOUND(ARY)
' 配列ARY(I)の値を読み取る
COMMON INPUT #1, ARY(I)

ARY(I) = ARY(I) + 1

' 配列ARY(I)の値を書き込む
```

```
COMMON PRINT #1, ARY(I)
NEXT I

COMMON CLOSE #1
```

配列の部分読み書きは、配列の要素数が大きい時に、グローバル共有との間で、最小限のデータのやり取りで済むメリットがあります。

特にネットワーク経由で、大きなサイズの配列をやり取りする際に、パフォーマンスに大きく影響します。

しかし、指定した添字以外の値は、グローバル共有と不一致になる可能性がある点に留意が必要です。(別のプログラムが添字の要素を書き換えてしまった場合など)

他に、グローバル共有にある配列から値を取り出す前に、配列の情報を得られる関数が幾つか用意されています。

COMMON CDIM	グローバル共有の配列の、次元数が得られます。
COMMON LDIM	グローバル共有の配列の、要素数が得られます。
COMMON UBOUND	グローバル共有の配列の、指定次元の添字最大値を得られます。

例えば、グローバル共有に、ARY配列があると仮定した際、その情報を得るには以下のように書きます。

```
' 簡単な配列の情報読み取り事例
COMMON OPEN "127.0.0.1" AS #1

LIST ARY
' 配列ARYの次元数を得る
JIGEN = COMMON CDIM(1, ARY)
PRINT "次元数="; ARY
' 配列ARYの各次元の要素数を得る
FOR I=1 TO JIGEN
  PRINT "次元"; I; "の要素数="; COMMON LDIM(1, ARY, I)
NEXT I

COMMON CLOSE #1
```

### 2.3.3 連想配列の読み書きを行う

グローバル共有は、連想配列の読み書きができます。  
以下に、連想配列の読み書き事例を示します。

```
' 簡単な連想配列の読み書き事例
COMMON OPEN "127.0.0.1" AS #1

DICT ARY
ARY("hoge") = 123
ARY("fuga") = 456
' 連想配列ARYの値を書き込む
COMMON PRINT #1, ARY

CLEAR_DICT ARY
' 連想配列ARYの値を読み取る
COMMON INPUT #1, ARY

PRINT "連想配列ARY="; ARY

COMMON CLOSE #1
```

連想配列を扱う際に重要なのは、まず一旦全体をグローバル共有に書き込む事です。  
いきなり読もうとしても、グローバル共有に値が書き込まれてないと、エラーとなります。

一旦 値を書き込むと、連想配列のキーを指定して、グローバル共有に対する部分読み書きが可能となります。

```
' 簡単な連想配列の部分読み書き事例
COMMON OPEN "127.0.0.1" AS #1

DICT ARY
' 連想配列ARYの値を読み取る
COMMON INPUT #1, ARY("hoge")

ARY("hoge") = ARY("hoge") + 1

' 連想配列ARYの値を書き込む
COMMON PRINT #1, ARY("hoge")

PRINT "連想配列ARY="; ARY

COMMON CLOSE #1
```

連想配列の部分読み書きは、連想配列のキー数が大きい時に、グローバル共有との間で、最小限のデータのやり取りで済むメリットがあります。  
特にネットワーク経由で、大量のキーを持つ連想配列をやり取りする際に、パフォーマンスに大きく影響します。

しかし、指定したキー以外の値は、グローバル共有と不一致になる可能性がある点に留意が必要です。(別のプログラムがキー以外の要素を書き換えてしまった場合など)

他に、グローバル共有にある連想配列から値を取り出す前に、連想配列の情報を得られる関数が幾つか用意されています。

COMMON GET_DICT_KEYSS	グローバル共有の連想配列の、キー一覧が得られます。
COMMON HAS_DICT_KEY	グローバル共有の連想配列の、指定キーの存在有無を確認できます。
COMMON LDICT	グローバル共有の連想配列の、キー総数を得られます。

## 2.3.4 両端キューに追加・取り出しを行う

グローバル共有に、両端キューを設置して、値の追加と取り出しができます。  
以下に、両端キューを使った、追加と取り出し事例を示します。

’ 両端キューを使った追加・取り出し事例(スタックのような動作)

```
COMMON OPEN "127.0.0.1" AS #1
```

’ 両端キューの末尾に、値を追加

```
LV = 1
```

```
COMMON DEQUE PUSH #1, LV
```

```
LV = 2
```

```
COMMON DEQUE PUSH #1, LV
```

```
LV = 3
```

```
COMMON DEQUE PUSH #1, LV
```

```
PRINT "両端キューLVの個数="; COMMON DEQUE LEN(1, LV)
```

’ 両端キューの末尾から、値を取り出し

```
DO WHILE COMMON DEQUE POP(1, LV) > 0
```

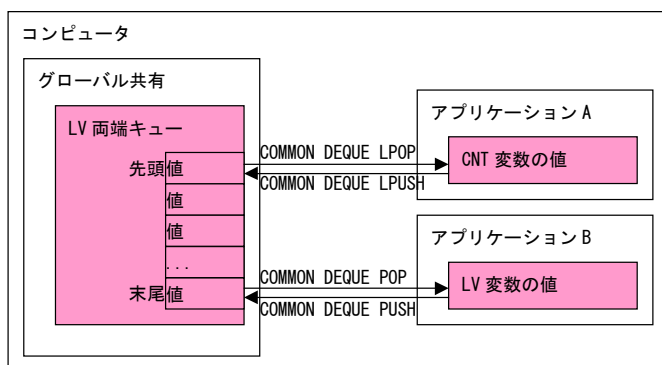
```
  PRINT "LV="; LV      ' 3, 2, 1の順に取り出されます
```

```
LOOP
```

```
PRINT "両端キューLVの個数="; COMMON DEQUE LEN(1, LV)
```

```
COMMON CLOSE #1
```

両端キューとは、先頭または末尾に対して、追加や取り出しが可能な待ち行列のようなものです。  
通常の値読み書きの場合、グローバル共有にある変数名の値は一意ですが、両端キューの場合 幾つも積み重ねる事ができます。  
また、通常の値読み書きは、グローバル共有には必ず値が存在しますが、両端キューの場合、値を取り出していくと、空になる事があります。



両端キューを操作する命令および関数は、以下が用意されています。

COMMON DEQUE LPUSH	グローバル共有の両端キューの先頭に追加
COMMON DEQUE LPOP	グローバル共有の両端キューの先頭から取り出し
COMMON DEQUE PUSH	グローバル共有の両端キューの末尾に追加
COMMON DEQUE POP	グローバル共有の両端キューの末尾から取り出し
COMMON DEQUE LEN	グローバル共有の両端キューの個数を得る

末尾に追加して、末尾から取り出すやり方をすると、スタックのように扱えます。(本を机の上に積んで、上から取り出すのを想像してください)

また、末尾に追加して、先頭から取り出すやり方をすると、キューのように扱えます。(お店のレジに並ぶ姿を想像してください)

### 2.3.5 型と存在の確認と削除

グローバル共有へは、アプリケーションが自由に値を読み書きする事ができます。

この為、読み書きしようとする値が、確かにグローバル共有にあるのか？その値の型は、自分が扱いたい型であるか、判断する方法が必要です。

これらの用途には、COMMON VARTYPE 関数が使用できます。

COMMON VARTYPE 関数の戻り値が負数だと、指定した変数名は無く、0以上の正数だと 値の型情報が得られます。

```
COMMON OPEN "127.0.0.1" AS #1
' CNTの値が有るか？その型は何か調べる。
TYP = COMMON VARTYPE(1, CNT)
IF TYP < 0 THEN
  PRINT "グローバル共有にCNT値はありません"
ELSE
  PRINT "グローバル共有のCNT値の型="; TYP
END IF

COMMON CLOSE #1
```

グローバル共有にある値を削除するには、COMMON ERASE 命令で指定します。

例えば、グローバル共有のCNT変数を削除するには、以下のように呼び出します。

```
COMMON OPEN "127.0.0.1" AS #1
' CNTの値を削除する
COMMON ERASE #1, CNT
COMMON CLOSE #1
```



### 2.3.6 排他処理をかける

複数のプログラムが、同じ変数名に対して読み書きを行うとき、値が順序よく更新されるようにする為に、排他処理をかける必要が出てきます。

排他処理を行うには、COMMON LOCK 関数でロックを取得し、読み書きを行った後に、COMMON UNLOCK 関数でロックを解放します。

以下に、プログラムの事例を示します。

```
' 簡単な排他処理事例
COMMON OPEN "127.0.0.1" AS #1

' "TEST"という名前で、5秒間 ロックを取得する
IF COMMON LOCK(1, "TEST", 5) > 0 THEN
  ' CNTの値を読み取る
  IF COMMON INPUTR(1, CNT) = 0 THEN
    ' 読み取り成功したら 1 を加算
    CNT = CNT + 1
  ELSE
    ' 読み取れなかったら、初期値 0 を与える
    CNT = 0
  END IF
  ' CNTの値を書き込む
  COMMON PRINT #1, CNT
  ' ロックを解放する
  F = COMMON UNLOCK(1, "TEST")
ELSE
  ' ロックが取得できなかった ... 暫くして再挑戦してください
END IF

PRINT "CNT="; CNT

COMMON CLOSE #1
```

COMMON LOCK 関数でロックを取得する際、有効期限を秒単位で指定します。

これは、ロックを取得した後、何がしかの不具合などにより、ロックを解放せずにプログラムを終了してしまうと、誰もロックが取得できなくなるために、有効期限を設定してもらいます。

複数のプログラム同士で排他制御を行うコツは、それぞれが同じ名前でロックを取得して、読み書きした上で、ロックを解放することです。

誰か別のプログラムが、ロックを取得せずに、勝手に読み書きすると、排他制御は意味を持たなくなります。



排他制御の間に、グローバル共有に格納された値を読み書きする対象に対して、同じロック名を指定してください。  
異なるロック名を指定すると、排他制御の意味がありません。

### 2.3.7 別コンピュータに対して読み書きする

別コンピュータの、グローバル共有の値を読み書きするには、COMMON OPEN 命令で、別コンピュータのIPアドレスを指定します。

別コンピュータのIPアドレスが「192.168.1.1」ならば、以下のように記述します。

COMMON OPEN "192.168.1.1" AS #1
---------------------------------

なお、この機能を使うには、別コンピュータの設定ファイルの内容を、一部書き換える必要があります。

Linuxのメニューから端末を開いて、以下の手順で設定ファイルの内容を書き換えた後、コンピュータを再起動してください。

1. Linuxのメニューから、アプリケーション>システムツール>端末を選んで、端末を起動する。
2. 「su」 コマンドを呼び出し、パスワードを入力して、管理者権限を得る。
3. 「nano /etc/redis/redis.conf」を入力し、設定ファイルをエディタで開く。
4. 「bind 127.0.0.1」と書かれている箇所を、「bind 0.0.0.0」と書き換える。
5. Ctrl+Xキーで終了を選び、Yキーで設定ファイルを保存して終了する。
6. 「reboot」 コマンドを呼び出し、コンピュータを再起動する。



設定ファイルの変更例「bind 0.0.0.0」により、どのコンピュータからでも読み書きできるようになります。 アクセス制限を加えたい場合、Redis( <a href="https://redis.io">https://redis.io</a> ) のドキュメンテーションを参照し、読解して各種設定を行ってください。
---

### 2.3.8 通信を暗号化しつつ、別コンピュータに対して読み書きする(ssh版)

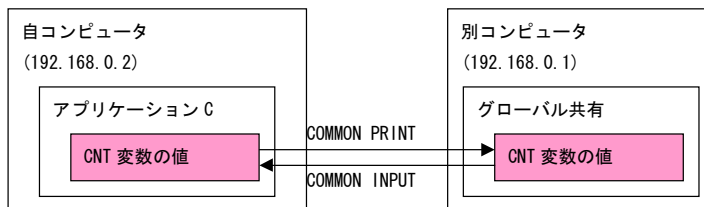
「別コンピュータに対して読み書きする」では、設定ファイルを書き換える事で、別コンピュータからのアクセスを許可するようにして、グローバル共有の値を読み書きできるようにしました。

これとは別の方法で、sshポートフォワーディング という技術を利用すると、設定ファイルを書き換える事なく、別コンピュータに接続でき、更に通信内容を暗号化する事が可能です。



sshポートフォワーディングを利用するには、別コンピュータおよび自コンピュータで、sshサービスを有効にする必要があります。  
sshサービスを有効にするには、Linux端末上から「`sudo systemctl restart sshd`」を入力して Enterキーを押します。

仮に、自コンピュータと別コンピュータの関係が、以下のだとします。



自コンピュータから別コンピュータに対して通信を通すために、まず、sshポートフォワーディングの設定を施す必要があります。

Linuxのメニューから端末を開いて、以下の手順でコマンドを入力します。

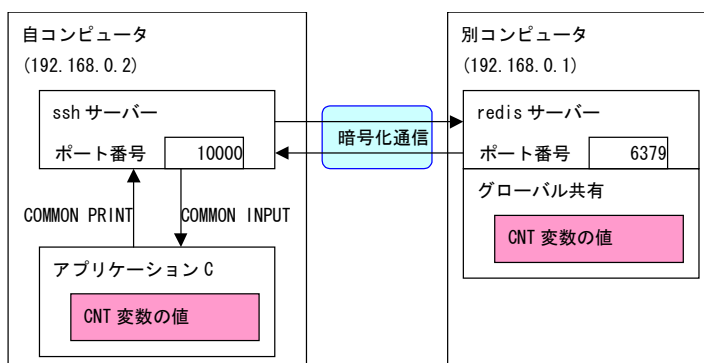
1. Linuxのメニューから、アプリケーション>システムツール>端末を選んで、端末を起動する。
2. 以下のように入力して、Enterキーを押す。

```
$ ssh -NL 10000:192.168.0.1:6379 user@192.168.0.1
```

3. 「user@192.168.0.1's password:」と表示されて、パスワードを求められるので、別コンピュータの user アカウントのパスワードを入力してEnterキーを押す。

これにより、自コンピュータから別コンピュータに対する通信路が開かれます。

構成的には、以下のようになります。



自コンピュータでは、以下のようにプログラムを書きます。

```
' sshポートフォワーディングを使用して、別コンピュータに対して読み書きする例
COMMON OPEN "127.0.0.1:10000" AS #1

' CNT値を初期化する
CNT = 123
' CNTの値をグローバル共有に書き込む
PRINT "CNT="; CNT

COMMON CLOSE #1
```

ここで重要なのは、COMMON OPEN命令のIPアドレスとポート番号の指定で、IPアドレスを自身のアドレスとし、ポート番号を「10000」としている事です。

これを実行すると、自コンピュータの10000ポートに対して接続しようとしします。

すると、Linux端末で ssh コマンドを使って開いた通信路を介して、別コンピュータの6397ポート(グローバル共有を実現するredisの基本ポート番号)が開かれます。

自コンピュータと別コンピュータの間の通信路は、ssh と呼ばれる暗号化された通信データでやり取りされます。

これにより、設定ファイルを変更せず、暗号化しつつ別コンピュータとの読み書きが可能となります。

### 2.3.9 別プログラムにメッセージを通知・受信する

グローバル共有を介して、複数のプログラムに対して、任意のメッセージを通知・受信することができます。



この機能は、Redis の pub / sub機能を用いて実現しています。

メッセージの通知・受信を使うには、COMMON NOTIFY 関数でメッセージを通知し、ON COMMON CALL 命令でメッセージを受信するサブルーチンを定義し、COMMON ON / OFF 命令でメッセージ受信機能のON / OFFを行います。

まず、以下にメッセージを通知する事例を示します。

```
' 簡単な通知事例
COMMON OPEN "127.0.0.1" AS #1

' TESTというトピック名に対して、「Hello AJAN」というメッセージを送ります
CNT = COMMON NOTIFY(1, "TEST", "Hello AJAN")
PRINT "受信者期待数="; CNT

COMMON CLOSE #1
```

次に、メッセージを受信する事例を示します。

```
' 簡単な受信事例
COMMON OPEN "127.0.0.1" AS #1

' トピック名: "TEST"にメッセージが来たら CB_TEST ルーチンにジャンプします
ON COMMON(1, "TEST") CALL CB_TEST
' 受信機能をONにします
COMMON ON #1

' 30秒ほど待機します
ST = CLOCK
DO WHILE (CLOCK - ST) < 30
LOOP

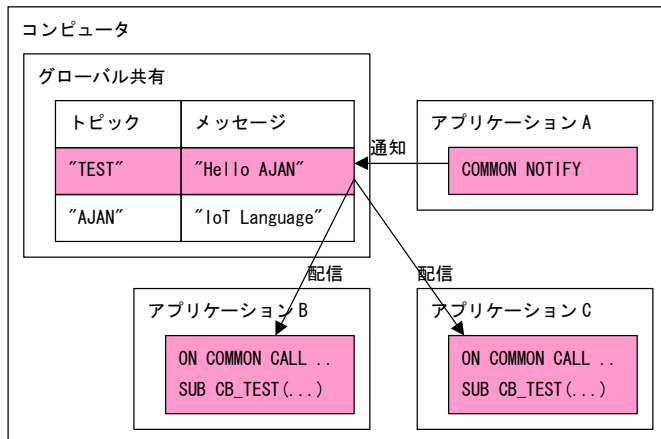
COMMON CLOSE #1

' メッセージを受信したら呼び出されるサブルーチン
SUB CB_TEST(NUM%, TOPIC$, MSG$)
  PRINT "メッセージを受信しました", NUM%, TOPIC$, MSG$
END SUB
```

受信する事例のプログラムを複数起動しておいて、通知するプログラムを実行すると、それぞれの受信プログラムに、「Hello AJAN」というメッセージが表示されます。通知プログラムは、受信した数が表示されます。

イメージを図にすると、以下のようなものです。

グローバル共有内に、メッセージを一旦受け取って、受信したいアプリケーションに対して配信する機能があります。



COMMON NOTIFY 関数で、トピック名とメッセージを、グローバル共有に通知します。

グローバル共有は、ON COMMON CALL 命令で、トピック名が一致するアプリケーションに対して、受け取ったメッセージを配信し、サブルーチンが呼ばれることで、トピック名とメッセージを受信することができます。

メッセージの受信は、トピック名が一致する、複数のアプリケーションが受信できます。

### 2.3.10 構造体を扱う際の注意

構造体を使って値を読み書きする際、グローバル共有との間を構造体まるごとやり取りされる点に注意してください。

例えば、要素100の配列を読み書きする例を考えます。

```
' 要素100の配列の読み書き事例
COMMON OPEN "127.0.0.1" AS #1

DIM ARY(99)      ' 要素100の配列を作る
ARY = [ 1 to 100 ] ' 初期化
' 配列ARYの値を書き込む
COMMON PRINT #1, ARY
' 配列ARYの値を読み取る
COMMON INPUT #1, ARY
PRINT "配列ARY="; ARY

COMMON CLOSE #1
```

構造体も、メンバに配列を定義できるので、似たような読み書き処理が可能です。

```
' 要素100の配列の読み書き事例(構造体版)
COMMON OPEN "127.0.0.1" AS #1

DEFINE STRUCT INFO
  DIM ARY(99)      ' 要素100の配列を作る
END STRUCT
STRUCT INFO V      ' 構造体を宣言
V. ARY = [ 1 to 100 ] ' 初期化

' 構造体Vの値を書き込む
COMMON PRINT #1, V
' 構造体Vの値を読み取る
COMMON INPUT #1, V
PRINT "構造体V="; V

COMMON CLOSE #1
```

ここで注意したいのは、グローバル共有との間で、要素100の配列全体の読み書きが発生している事です。

配列の要素数を1000、10000、と増やしたり、別のコンピュータ間で読み書きを行うようにすると、「たった1コマンド」で、大量の情報量のデータが行き来する事が判ります。

処理上 必要な事なら仕方ありませんが、仮にグローバル共有にある配列の一部を参照したい場合、このようなやり方は「非常に無駄」な事です。

グローバル共有にある配列の一部を参照したい場合、配列や連想配列ならば、以下のようにして一部の値のみを読み取る事ができます。

```
...
DIM ARY(99) ' 要素100の配列を作る
' 配列ARYの一部の値を読み取る
COMMON INPUT #1, ARY(10) ' この例では、添字10の値を読み取っています。
...
```

しかし、構造体では、メンバの配列の一部のみを読み取る事はできません。常に構造体全体を読み取る事しかできません。

状況に応じて、構造体を使うケース、配列や連想配列を使うケースを工夫してください。

### 2.3.11 通信回線切断などのトラブル時の注意

通信回線が断線などの理由で、別コンピュータに対して読み書きに失敗する事態が発生したとします。

通信異常に対して、コマンド発行に対してタイムアウト時間を指定することで、異常事態を検知したい場合は、COMMON OPEN 呼び出し時に「TIMEOUT」パラメータを指定します。

すると、コマンド発行から指定秒数が経過して応答が無い場合に、エラーとなります。

```
' 192.168.1.1のコンピュータに対して接続
' コマンド発行のタイムアウト時間を10秒に設定する
COMMON OPEN [ "192.168.1.1"; "TIMEOUT=10" ] AS #1

' ここで、ケーブル断線などにより、回線異常が発生したとする

COMMON ERASE #1, MSG$
' ↑この呼び出しで、タイムアウト時間を経過すると、以下のようなエラーが表示される
' 「Error! &H01000012: ドライバまたはライブラリの呼び出しに失敗しました(システム共有データへの削除ができません(1, 1, リソースが一時的に利用できません, 184))...」
```

また、COMMON OPEN 呼び出し時に、タイムアウト時間を指定したい場合は、「CONNTIMEOUT」パラメータを指定します。

```
' ここで、ケーブル断線などにより、回線異常が発生したとする

' 192.168.1.1のコンピュータに対して接続
' オープン時の接続タイムアウト時間を10秒に設定する
' コマンド発行のタイムアウト時間を10秒に設定する
COMMON OPEN [ "192.168.1.1"; "CONNTIMEOUT=10"; "TIMEOUT=10" ] AS #1
' ↑この呼び出しで、タイムアウト時間を経過すると、以下のようなエラーが表示される
' 「Error! &H01000012: ドライバまたはライブラリの呼び出しに失敗しました(システム共有データ(192.168.1.1:6379)の接続に失敗(1, ホストへの経路がありません))...」
```



ON ERROR CALL 等で、このエラーを検知して、通信回線が元に戻った後、再度コマンドを発行しようとする時、COMMON OPEN を使って再度接続しなおす必要があります。

この再接続を自動で行いたい場合、COMMON OPEN 呼び出し時に「RECONNECT」パラメータを指定します。

```
' エラーが発生すると、CB_ERR ルーチンを呼び出すように
ON ERROR CALL CB_ERR
ERROR ON

' エラーが発生した際の処理ルーチン
SUB CB_ERR(I_ERR, I_ERMS$, I_ERL)
  G_ERR = I_ERR      ' 発生したエラーを G_ERR変数に代入して戻る
END SUB

' 192.168.1.1のコンピュータに対して接続
' コマンド発行のタイムアウト時間を10秒に設定する
' 前回通信異常によるエラーが起きていた場合、自動再接続してからコマンドを発行させる
COMMON OPEN [ "192.168.1.1"; "CONNTIMEOUT=10"; "TIMEOUT=10"; "RECONNECT=1" ] AS #1

' ここで、ケーブル断線などにより、回線異常が発生したとする

COMMON ERASE #1, MSG$
' ↑この呼び出しで、タイムアウト時間を経過すると、エラーが発生して CB_ERR ルーチンにジャンプ

IF G_ERR <> 0 THEN
  ' 前のCOMMON ERASE 呼び出しでエラーが起きていたので、再度実行する
  G_ERR = 0
  COMMON ERASE #1, MSG$
  ' ↑前のコマンドと同じ処理をリトライする

  IF G_ERR <> 0 THEN
    PRINT "再度コマンド発行したが、またエラーなのでプログラム終了します"
  END
END IF
END IF
```

### 2.3.12 データベースの個数を増減したい時

「COMMON OPEN」のオプション設定で「SELECT」オプションを使うと、異なるデータベース番号にアクセス可能です。これは、数字で指定する、Excelのワークシートのようなものです。

このデータベースの個数はデフォルトが16個で、0から15まで指定可能です。データベースの個数を増やすには、設定ファイルの内容を、一部書き換える必要があります。

Linuxのメニューから端末を開いて、以下の手順で設定ファイルの内容を書き換えた後、コンピュータを再起動してください。

1. Linuxのメニューから、アプリケーション>システムツール>端末を選んで、端末を起動する。
2. 「`su`」コマンドを呼び出し、パスワードを入力して、管理者権限を得る。
3. 「`nano /etc/redis/redis.conf`」を入力し、設定ファイルをエディタで開く。
4. 「`databases 16`」と書かれている箇所の、数値の部分を任意の値に書き換える。  
(データベースの個数を20に増やしたい場合、「16」を「20」に書き換えます)
5. Ctrl+Xキーで終了を選び、Yキーで設定ファイルを保存して終了する。
6. 「reboot」コマンドを呼び出し、コンピュータを再起動する。

### 2.3.13 グローバル共有のデータを全てクリアしたい

グローバル共有のデータを、一旦全てクリアしたい場合、グローバル共有が使用している Redis というソフトウェアが提供している、「`redis-cli`」コマンドの機能を組み合わせて行えます。

1. Linuxのメニューから、アプリケーション>システムツール>端末を選んで、端末を起動する。
2. 「`redis-cli flushdb`」と呼び出す。グローバル共有のデータはクリアされます。  
別のデータベース番号を指定したい場合は、「`-n <db番号>`」を使用します。例えば、データベース番号1のデータをクリアしたい場合は、「`redis-cli -n 1 flushdb`」とします。



データのクリアは、Redis の「`FLUSHDB`」コマンドを使用しています。  
詳しくは、Redis の コマンド説明を参照ください。

### 2.3.14 グローバル共有のデータを別のグローバル共有に複製したい

グローバル共有のデータを、別のグローバル共有や別のデータベース番号に複製したい場合、グローバル共有が使用している Redis というソフトウェアが提供している、「[redis-cli](#)」コマンドの機能を組み合わせて行えます。

例えば、グローバル共有のデータを「1.2.3.4」のIPアドレスのグローバル共有に複製したい場合、Linuxの端末から、以下のように呼び出します。

(Linuxの端末の起動は、Linuxのメニューから、アプリケーション>システムツール>端末 を選びます)

```
$ redis-cli --scan | xargs redis-cli migrate 1.2.3.4 6379 '' 0 5000 copy replace keys
OK
```

上の例では、「[redis-cli --scan](#)」で、グローバル共有の全てのキー名を列挙しつつ、「|」(パイプ記号)」で、次のコマンドに処理を引き渡しています。

次の処理は「[xargs](#)」コマンドで、後続の「[redis-cli](#)」コマンドとパラメータの後に受け取ったキー名を連結して実行します。

「[redis-cli](#)」コマンドの呼び出しでは、Redisの「[MIGRATE](#)」コマンドを呼び出しており、複製先のIPアドレスとポート番号、空文字列、データベース番号、タイムアウト時間と、挙動(copy と replace により、上書きコピーする)および複製するキー名を指定しています。



データの複製は、Redis の「[MIGRATE](#)」コマンドを使用しています。  
詳しくは、Redis の コマンド説明を参照ください。

参考までに、データベース番号0 にあるデータを、データベース番号1 に複製する場合、以下のように呼び出します。

```
$ redis-cli --scan | xargs redis-cli migrate 127.0.0.1 6379 '' 1 0 copy replace keys
(error) IOERR error or timeout reading to target instance
```

コピー先がローカルホスト(127.0.0.1の事)を指定すると、事例のようなエラーメッセージが出ますが、2022年2月時点で既知の問題のようです。動作には支障がありません。

### 2.3.15 複数のサーバを起動したい

グローバル共有を使ってデータを管理するとき、こっちはテスト用、こっちは本番用。という具合にデータを分けて管理したい時があります。

対応としては、「COMMON OPEN」の「SELECT」オプションを使ってデータベース番号を変えて管理する方法や、異なるコンピュータにあるグローバル共有を用いる方法が考えられます。

これとは別に、サーバを複数起動して管理する方法があります。

グローバル共有が使用する Redisサーバは、デフォルトでは ポート番号 : 6379 を使用するよう設定されています。

であれば、別のポート番号を使用するように設定する事で、複数の Redisサーバを起動する事が可能となります。



Redis はインメモリで動作するサーバなので、多くの Redis サーバを起動すると、メモリ不足に陥る可能性があります。  
複数のサーバ起動には、コンピュータのメモリ資源の状況を考慮しつつ、慎重に運用を行ってください。

Interface Linux Systemにインストールされている Redisサーバの基本設定では、サーバの設定ファイルは「[/etc/redis/redis.conf](#)」にあり、systemd と呼ばれるシステム管理下で動くように設定されています。サービス名はデフォルトで「[redis-server.service](#)」です。  
(類似に「[redis-sentinel.service](#)」という分散向けのサービス名もあります)

「[redis-server.service](#)」サービスが稼動中か確認するには、Linux端末から「[systemctl](#)」コマンドを呼び出します。

(Linuxの端末の起動は、Linuxのメニューから、アプリケーション>システムツール>端末 を選びます)

以下に、稼動中確認の呼び出し例を示します。

```
$ systemctl status redis-server.service
● redis-server.service - Advanced key-value store
   Loaded: loaded (/lib/systemd/system/redis-server.service; enabled; vendor pre
   Active: active (running) since Wed 2022-02-02 19:49:15 JST; 1 day 15h ago
     Docs: http://redis.io/documentation,
           man:redis-server(1)
  Main PID: 31530 (redis-server)
    Status: "Ready to accept connections"
     Tasks: 4 (limit: 4915)
    CGroup: /system.slice/redis-server.service
            └─31530 /usr/bin/redis-server 127.0.0.1:6379

lines 1-10/10 (END)
```

複数のRedisサーバを起動するには、別のポート番号向けに設定ファイルを用意し、別の名前のサービスを設定してやれば良いです。

設定方法の詳細に関しては、Redisのコミュニティや Linuxのsystemd などの関連情報を参考にしてください。

設定について少々面倒なので、設定ファイルの作成を支援するツールを用意しました。

サンプルプログラムフォルダ(「/usr/share/interface/AJANPro/samples/EXT/EXT」)内に、「[make\\_multiple\\_service.sh](#)」という、bashスクリプトを用意していますので、Linux端末からroot権限を付与して実行してください。

以下に、呼び出し事例を示します。

```
$ sudo bash make_multiple_service.sh
引数:[make_multiple_service.sh ]
グローバル共有(redis サーバ)の複数起動用の設定ファイルを作ります
新規に作りたいサーバーのポート番号を指定してください
デフォルトはポート番号 6379 です。それ以外を期待します(推奨は 6380 など)
ポート番号:6380
ポート番号:6380 の設定ファイルを作ります
OK?(y/n):y
ポート番号:6380 の設定ファイルを作ります
+ REDIS_CONF=/etc/redis/redis-6380.conf

... <略>

+ systemctl enable redis-server-6380.service
+ set +x
グローバル共有(redis サーバー)の 6380 ポート向けサーバーの設定を作成しました
サーバーを起動するには、「sudo systemctl start redis-server-6380.service」と入力してください
```

事例では、ポート番号 : 6380 を使用するよう、デフォルトの設定ファイルを元に、新たな設定ファイルと、新しいサービス名(ここでは「[redis-server-6380.service](#)」)を作成しています。

サービスを稼動して、動いているか確認している様子を、以下に事例で示します。

```
$ sudo systemctl start redis-server-6380.service
user@InterfaceLinuxSystem:~/AjanProWS/Samples/EXT/EXT$ systemctl status
redis-server-6380.service
● redis-server-6380.service - Advanced key-value store
   Loaded: loaded (/lib/systemd/system/redis-server-6380.service; enabled; vendor
   Active: active (running) since Thu 2022-02-03 20:28:45 JST; 15h ago
     Docs: http://redis.io/documentation,
           man:redis-server(1)
   Main PID: 311 (redis-server)
     Status: "Ready to accept connections"
    CGroup: /system.slice/redis-server-6380.service
            └─311 /usr/bin/redis-server 127.0.0.1:6380
lines 1-9/9 (END)
```

サービスが稼動する事でサーバが起動したら、AJANからは「COMMON OPEN」でIPアドレスとポート番号を明示する事で、新しいサーバのグローバル共有にアクセスできます。

以下に、ポート番号：6380 のグローバル共有をオープンする事例を示します。

' ポート番号：6380 のグローバル共有をオープンする事例 COMMON OPEN "127.0.0.1:6380" AS #1
--

これにより、複数のサーバの稼動および、どのサーバのグローバル共有にアクセスするか選択できるようになります。

### 2.3.16 グローバル共有のデータをバックアップする

グローバル共有のデータを、バックアップしたい場合、データの内容が記録されたファイルと場所の把握と、データの更新を一時停止・再開する必要があります。

グローバル共有が使用している Redis というソフトウェアでは、データのファイルと場所は、設定ファイルに記載されています。

設定ファイル中の どの項目を参照すべきか、以下に事例を示します。

項目	初期値	解説
save	900 1 300 10 60 10000	この項目が "" (空文字列) で無ければ、dir 項目と dbfilename 項目は有効です。
dir	/var/lib/redis	dbfilename 項目のファイルが配置される場所です。
dbfilename	dump.rdb	RDB ファイルと呼ばれるデータファイルが、dir 項目の指すフォルダに配置されます。 このファイルには、ある時点のスナップショット(フルダンプ)が記録されます。
appendonly	no	この項目が yes であれば、appendfilename 項目は有効です。
appendfilename	appendonly.aof	AOF ファイルと呼ばれるデータファイルが、dir 項目の指すフォルダに配置されます。 このファイルは、更新内容を追記ログのように記録されます。

データファイルの場所が把握出来たら、Redisサービスを一旦止めて、データファイルのバックアップを行い、サービスを再開します。

```
$ systemctl stop redis-server.service    # サービスの停止

# ここで redis サービスが停止するので、データファイルのバックアップ等を行えばよい。

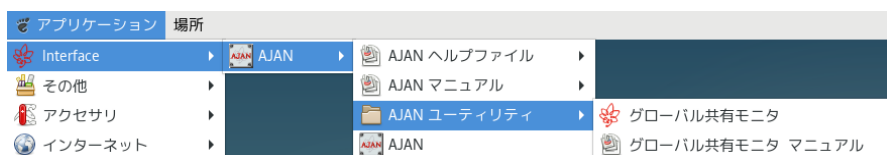
$ systemctl start redis-server.service    # サービスの開始
```

### 2.3.17 グローバル共有モニタの紹介

グローバル共有の読み書きを行うプログラミングを行っているとき、現在 どのような値になっているのか確認したい欲求が出てくると思います。

そういった時に、グローバル共有モニタ と呼ぶツールを使用すると、システム上に どのようなグローバル変数があり、どのような値になっているか簡単に確認できます。

グローバル共有モニタは、デスクトップ画面上部のメニューバーから「アプリケーション」→「Interface」→「AJAN」→「AJANユーティリティ」→「グローバル共有モニタ」をクリックします。



起動すると、グローバル共有モニタが起動します。ここで「OPEN」ボタンをクリックすると、以下のようなグローバル共有の名前と値が対になって表示されます。





メニューに隣接して「グローバル共有モニタ マニュアル」があり、グローバル共有モニタの使い方の説明があります。ご参照ください。



## 第3章 リファレンス

使用できる拡張コマンドの使い方について記載します。

	制限事項については、「注意」に記載しています。
	使用例は動作を保証するものではありません。 実際の使い方は各種サンプルプログラムを参照してください。

### 3.1 コマンド一覧

コマンド名	機能
数値・文字列に関する関数・命令	
ARRAY2BINBLK\$	数値配列を任意ブロック形式などの書式化文字列に変換します。
ARRAY2STR\$	数値配列を指定したバイト単位で文字列に変換します。
BINBLK2ARRAY	任意ブロック形式などの書式化文字列を数値配列に変換します。
CBOOL	式の値を、論理型値に変換します。
CDBL	式の値を、倍精度実数値に変換します。
CINT	式の値を、単精度整数値に変換します。
CLNG	式の値を、倍精度整数値に変換します。
CSNG	式の値を、単精度実数値に変換します。
CLOCK2DATETIMESTR\$	CLOCK関数の値を日時文字列に変換します。
CONVBIN	連続量の値(アナログデータ)からバイナリ値(デジタルデータ)に変換します。
CONVPHY	バイナリ値(デジタルデータ)から連続量の値(アナログデータ)に変換します。
CONVUNIT	値を単位変換します。
DATEADD	日付時刻値に対して、指定した時間間隔に加算した値を返します。
DATETIMESTR2CLOCK	日時文字列を秒単位(CLOCK関数相当)に変換します。
DAY	指定日の日を返します。
FORMAT\$	値を指定した書式に従って文字列変換します。
FORMATDATETIME\$	日付時刻値を指定した書式に従って文字列化します。
HASH\$	文字列に対してハッシュを得ます
HOURL	指定時刻の時を返します。
ICONV\$	文字列に対して、文字コード変換を行います。
JOIN\$	1次元配列の文字列を、挿入記号を間に挟んで、1つの文字列に結合します。
LCASE\$	文字列中の大文字(半角英字)を小文字に変換します。
LEFT\$	文字列の左側から任意の長さの文字列を抜き出します。
LEFTB\$	文字列の左側から任意のバイト数分の文字列を抜き出します。
MATH ISEQ	2つの数値の差分を取って等値比較を行います
MINUTE	指定時刻の分を返します。
MKUUID\$	UUID文字列を生成します。
MONTH	指定日の月を返します。
PASSWORD_HASH\$	受け取ったパスワード文字列からハッシュ文字列を得ます。
PASSWORD_VERIFY	パスワード文字列と、PASSWORD_HASH\$で得たハッシュの内容が一致しているか確認します。

コマンド名	機能
数値・文字列に関する関数・命令	
RIGHT\$	文字列の右側から任意の長さの文字列を抜き出します。
RIGHTB\$	文字列の右側から任意のバイト数分の文字列を抜き出します。
SECOND	指定時刻の秒を返します。
STR_BASE64_DECODE\$	BASE64形式にエンコードされた文字列を、元の文字列にデコード(逆変換)します。
STR_BASE64_ENCODE\$	文字列を、BASE64形式にエンコード(変換)します。
STR_DUMP\$	単一値をダンプ文字列化
STR_AND\$	文字列同士に対して、論理積(AND)を求めた結果を返します。
STR_OR\$	文字列同士に対して、論理和(OR)を求めた結果を返します。
STR_XOR\$	文字列同士に対して、排他的論理和(XOR)を求めた結果を返します。
STR_IMP\$	文字列同士に対して、包含(IMP)を求めた結果を返します。
STR_EQV\$	文字列同士に対して、同値(EQV)を求めた結果を返します。
STR_REPEAT\$	文字列を指定回数繰り返し連結した文字列を返します。
STR_HAN2ZEN\$	文字列の半角文字を全角文字に変換します。
STR_ZEN2HAN\$	文字列の全角文字を半角文字に変換します。
STR_HIRA2KATA\$	文字列のひらがな文字をカタカナ文字に変換します。
STR_KATA2HIRA\$	文字列のカタカナ文字をひらがな文字に変換します。
STR_STARTSWITH	文字列が特定文字列で始まるか判断します。
STR_ENDSWITH	文字列が特定文字列で終わるか判断します。
STR_REMOVEPREFIX\$	文字列の先頭部分が指定文字列と一致すれば、これを削除します。
STR_REMOVESUFFIX\$	文字列の末尾部分が指定文字列と一致すれば、これを削除します。
STR_VALIDUTF8	文字列全てがUTF8文字列として妥当であるか判定します。
STR_ISALNUM	文字列全てが英数文字であるか判定します。
STR_ISALPHA	文字列全てが英文字であるか判定します。
STR_ISCNTRL	文字列全てが制御文字であるか判定します。
STR_ISDIGIT	文字列全てが数文字であるか判定します。
STR_ISGRAPH	文字列全てが表示可能な文字であるか判定します。
STR_ISLOWER	文字列全てが小文字であるか判定します。
STR_ISPRINT	文字列全てが表示可能な文字(空白含む)であるか判定します。
STR_ISPUNCT	文字列全てが表示可能な文字(空白と英数字を含まない)であるか判定します。
STR_ISSPACE	文字列全てが空白、タブ、行区切りであるか判定します。
STR_ISUPPER	文字列全てが大文字であるか判定します。
STR_ISXDIGIT	文字列全てが16進数文字であるか判定します。
STR_ISASCII	文字列全てがASCII文字であるか判定します。
CHRTYPE	文字列の各文字の情報を得ます。
STR2ARRAY	文字列を指定したバイト単位の数値配列に変換します。
STRDEL\$	文字列の任意の位置から指定したバイト長の文字列を削除します。
STRINS\$	文字列の任意の位置に指定したバイトデータ文字列を挿入します。
STRREVERSE\$	文字列の並びを逆にします。
STRREVERSEB\$	バイナリ文字列の並びを逆にします。
TRUNC	指定数値を指定位置で切り捨てます。
UCASE\$	文字列中の小文字(半角英字)を大文字に変換します。
UUIDCOMP	UUID文字列同士が、同じか比較します。
YEAR	指定日の年を返します。
配列演算に関する関数・命令	
DIMAVG	配列の平均を返します。
DIMMEDIAN	配列の中央値を返します。
DIMMAX	配列の最大値を返します。
DIMMIN	配列の最小値を返します。
DIMSUM	配列の合計を返します。
DIMSTDEVP	配列の標準偏差値を返します。

コマンド名	機能
数値・文字列に関する関数・命令	
DIMVARP	配列の分散値を返します。
DIMVARS	配列の不偏分散値を返します。
DIMADD	2つの配列の各要素に対して、加算した結果を返します。
DIMADDS	2つの文字列配列の各要素に対して、連結した結果を返します。
DIMSUB	2つの配列の各要素に対して、減算した結果を返します。
DIMDIV	2つの配列の各要素に対して、割り算した結果を返します。
DIMMOD	2つの配列の各要素に対して、剰余算した結果を返します。
DIMMUL	2つの配列の各要素に対して、掛け算した結果を返します。
DIMNOT	配列の各要素に対して、否定 (NOT) を求めた結果を返します。
DIMAND	2つの配列の各要素に対して、論理積 (AND) を求めた結果を返します。
DIMOR	2つの配列の各要素に対して、論理和 (OR) を求めた結果を返します。
DIMXOR	2つの配列の各要素に対して、排他的論理和 (XOR) を求めた結果を返します。
DIMIMP	2つの配列の各要素に対して、包含 (IMP) を求めた結果を返します。
DIMEQV	2つの配列の各要素に対して、同値 (EQV) を求めた結果を返します。
DIMSHL	配列の各要素に対して、ビット単位に左シフトした結果を返します。
DIMSHR	配列の各要素に対して、ビット単位に右シフトした結果を返します。
DIMROL	配列の各要素に対して、ビット単位に左回転した結果を返します。
DIMROR	配列の各要素に対して、ビット単位に右回転した結果を返します。
DIMEQ	2つの配列の各要素に対して、等しいか比較を求めた結果を返します。
DIMNE	2つの配列の各要素に対して、等しくないか比較を求めた結果を返します。
DIMLT	2つの配列の各要素に対して、小なりか比較を求めた結果を返します。
DIMGT	2つの配列の各要素に対して、大なりか比較を求めた結果を返します。
DIMLTE	2つの配列の各要素に対して、小なりイコールか比較を求めた結果を返します。
DIMGTE	2つの配列の各要素に対して、大なりイコールか比較を求めた結果を返します。
DIMIIF	評価用の配列の各要素に対して、結果が真であれば真用の配列値を、偽であれば偽用の配列値を返します。
DIMIIFS	評価用の配列の各要素に対して、結果が真であれば真用の文字列配列値を、偽であれば偽用の文字列配列値を返します。
DIMCINT	文字列または数値配列から、単精度整数の配列に変換します。変換する対象をマスク用に配列指定できます。
DIMCLNG	文字列または数値配列から、倍精度整数の配列に変換します。変換する対象をマスク用に配列指定できます。
DIMCSNG	文字列または数値配列から、単精度実数の配列に変換します。変換する対象をマスク用に配列指定できます。
DIMCDBL	文字列または数値配列から、倍精度実数の配列に変換します。変換する対象をマスク用に配列指定できます。
DIMCSTR\$	数値配列から、数文字に変換した文字列の配列に変換します。変換する対象をマスク用に配列指定できます。
DIMFIND	配列に対して、検索値と一致する要素の添字番号を求めます。
DIMGET	1次元または2次元の数値配列に対して、指定した範囲を部分配列として取り出します。
DIMGET\$	1次元または2次元の文字列配列に対して、指定した範囲を部分配列として取り出します。
DIMMAP	配列の各要素に対して指定した関数を呼び出して、返された値からなる数値配列を作ります。

コマンド名	機能
<b>数値・文字列に関する関数・命令</b>	
DIMMAP\$	配列の各要素に対して指定した関数を呼び出して、返された値からなる文字列配列を作ります。
DIMREDUCE	配列の各要素に対して指定した関数を呼び出して、一つの値にまとめます。
DIMSET	1次元または2次元の配列に対して、指定領域をコピー元配列で上書きします。
NUM2BIT	数値をビット単位の配列に分解します。
BIT2NUM	ビット単位の配列を数値に結合します。
ONEDIM SORT	数値の1次元配列に対してソートした結果を得ます。
ONEDIM SORT\$	文字列の1次元配列に対してソートした結果を得ます。
TWODIM FILTER	2次元の数値配列の指定された添え字の値に対して、指定した関数を呼び出し、結果が真の行ないしは列からなる配列を作ります。
TWODIM FILTER\$	2次元の文字列配列の指定された添え字の値に対して、指定した関数を呼び出し、結果が真の行ないしは列からなる配列を作ります。
TWODIM JOIN	2つの数値の2次元配列を指定した結合列を基準に、結合した配列を得ます。
TWODIM JOIN\$	2つの文字列の2次元配列を指定した結合列を基準に、結合した配列を得ます。
TWODIM SORT	数値の2次元配列の指定した行／列位置に対してソートした結果を得ます。
TWODIM SORT\$	文字列の2次元配列の指定した行／列位置に対してソートした結果を得ます。
TWODIM EXISTS	2つの数値の2次元配列を指定した評価列を基準に、一致した行を抜き出した配列を得ます。
TWODIM EXISTS\$	2つの文字列の2次元配列を指定した評価列を基準に、一致した行を抜き出した配列を得ます。
TWODIM NOT EXISTS	2つの数値の2次元配列を指定した評価列を基準に、不一致の行を抜き出した配列を得ます。
TWODIM NOT EXISTS\$	2つの文字列の2次元配列を指定した評価列を基準に、不一致の行を抜き出した配列を得ます。
TWODIM TRANSPOSE	2次元の数値配列の行列を入れ替えた配列を得ます。
TWODIM TRANSPOSE\$	2次元の文字列配列の行列を入れ替えた配列を得ます。
<b>Python連携に関する関数・命令</b>	
PYOBJ CREATE CODE	Pythonと連携する為のコード文字列を渡して初期化します。
PYOBJ CALL FUNCTION	Pythonの指定した関数を呼び出します。
PYOBJ CALL METHOD	Pythonの指定のオブジェクトに対するメソッドを呼び出します。
PYOBJ CLOSE	Pythonとの連携処理を閉じます。
<b>オーディオ入出力を使ったアナログ変換に関する関数・命令</b>	
SNDOPEN	オーディオ入出力をオープンします。
SNDCLOSE	オーディオ入出力をクローズします。
SNDREAD	指定件数をオーディオ入力からアナログ変換し、データ配列を得ます。
SNDREADLEN	オーディオ入力の入力バッファの件数を取得します。
SNDWRITE	データ配列のデータを、アナログ変換してオーディオ出力します。
SNDWRITELEN	オーディオ出力の出力バッファの書き込み可能件数を取得します。
<b>ファイル・フォルダに関する関数・命令</b>	
CURDIR\$	現在のカレントディレクトリのパスを取得します。
<b>プロセスに関する関数・命令</b>	
GETCOMMANDLINEARG\$	コマンドライン引数を文字列配列形式で得ます。
GETSYSTEMINFO\$	指定した取得IDに対応するシステムに関連する情報を得ます。
ENVIRON\$	環境変数文字列テーブル内の環境文字列を得ます。
<b>グローバル共有に関する関数・命令</b>	



コマンド名	機能
数値・文字列に関する関数・命令	
COMMON OPEN	グローバル共有機能をオープンします。
COMMON CLOSE	グローバル共有をクローズします。
COMMON FREEFILE	使用可能な、グローバル共有の管理番号を得ます。
COMMON ERASE	グローバル共有に格納された、変数名で指定した内容を削除します。
COMMON INPUT	グローバル共有から、変数名で指定された値を読み取ります。
COMMON INPUTR	グローバル共有から、変数名で指定された値を読み取り、成功可否が得られます。
COMMON READ	グローバル共有から、文字列で指定した変数名で値を読み取って変数に格納し、成功可否が得られます。
COMMON PRINT	変数名で指定した値を、グローバル共有に格納します。
COMMON WRITE	文字列で指定した変数名と指定した値を、グローバル共有に格納します。
COMMON LOCK	グローバル共有に対して、指定した名前ロックを取得します。
COMMON UNLOCK	グローバル共有に対して、指定した名前ロックを解除します。
COMMON NOTIFY	グローバル共有を介して、トピック名にメッセージを通知します。
ON COMMON CALL	グローバル共有を介して、指定したトピック名に合致するメッセージを受信するサブルーチンを定義します。
COMMON ON / OFF	COMMONメッセージ受信の許可、禁止を指定します。
COMMON CDIM	グローバル共有にある、変数名で指定された配列の次元数を返します。
COMMON GET_DICT_KEYSS	グローバル共有にある、変数名で指定された連想配列のキー一覧を文字列配列として得ます。
COMMON HAS_DICT_KEY	グローバル共有にある、変数名で指定した連想配列に対して、指定したキーが存在するか否かを返します。
COMMON LDICT	グローバル共有にある、変数名で指定された連想配列に対して登録されているキーの総数を取得します。
COMMON LDIM	グローバル共有にある、変数名の配列の要素数を返します。
COMMON UBOUND	グローバル共有にある、変数名で指定された配列の指定した次元の添字最大値を得ます。
COMMON VARTYPE	グローバル共有にある、変数名で指定した値の型を調べます。
COMMON DEQUE PUSH	変数名で指定した値を、グローバル共有の両端キューの末尾に追加します。
COMMON DEQUE LPUSH	変数名で指定した値を、グローバル共有の両端キューの先頭に追加します。
COMMON DEQUE POP	グローバル共有から、変数名で指定された両端キューの末尾より値を取り出します。
COMMON DEQUE LPOP	グローバル共有から、変数名で指定された両端キューの先頭より値を取り出します。
COMMON DEQUE BACK	グローバル共有から、変数名で指定された両端キューの末尾より値を参照します。
COMMON DEQUE FRONT	グローバル共有から、変数名で指定された両端キューの先頭より値を参照します。
COMMON DEQUE LEN	グローバル共有から、変数名で指定された両端キューの個数を得ます。
COMMON VLIST\$	グローバル共有の変数名のリストを、文字列配列で得ます。
COMMON SWAPDB	グローバル共有に対して、指定したデータベースの番号を入れ替えます。
COMMON2 ERASE	グローバル共有に格納された、文字列で指定した変数名の内容を削除します。
COMMON2 CDIM	グローバル共有にある、文字列で指定した変数名の、中の配列の次元数を返します。

コマンド名	機能
数値・文字列に関する関数・命令	
COMMON2 GET_DICT_KEYSS	グローバル共有にある、文字列で指定した変数名の、中の連想配列のキー一覧を文字列配列として得ます。
COMMON2 HAS_DICT_KEY	グローバル共有にある、文字列で指定した変数名の、中の連想配列に対して、指定したキーが存在するか否かを得ます。
COMMON2 LDICT	グローバル共有にある、文字列で指定した変数名の、中の連想配列に対して、登録されているキーの総数を取得します。
COMMON2 LDIM	グローバル共有にある、文字列で指定した変数名の、中の配列の要素数を返します。
COMMON2 UBOUND	グローバル共有にある、文字列で指定した変数名の、中の配列の指定した次元の添字最大値を得ます。
COMMON2 VARTYPE	グローバル共有にある、文字列で指定した変数名の、中の値の型を調べます。
COMMON2 DEQUE PUSH	文字列で指定した変数名と指定した値を、グローバル共有の両端キューの末尾に追加します。
COMMON2 DEQUE LPUSH	文字列で指定した変数名と指定した値を、グローバル共有の両端キューの先頭に追加します。
COMMON2 DEQUE POP	グローバル共有から、文字列で指定した変数名で両端キューの末尾より値を取り出します。
COMMON2 DEQUE LPOP	グローバル共有から、文字列で指定した変数名で両端キューの先頭より値を取り出します。
COMMON2 DEQUE BACK	グローバル共有から、文字列で指定された変数名で両端キューの末尾より値を参照します。
COMMON2 DEQUE FRONT	グローバル共有から、文字列で指定された変数名で両端キューの先頭より値を参照します。
COMMON2 DEQUE LEN	グローバル共有から、文字列で指定した変数名で両端キューの個数を得ます。
COMMON2 VLIST\$	グローバル共有の変数名のリストを、文字列配列で得ます。
その他に関する関数・命令	
QRCODE_GET	QRコード作成に必要な情報を得ます。
拡張コマンドのサブルーチン	
GET_OCR\$	画像ファイルに対して、tesseract コマンドを用いて OCR解析によりテキスト情報を取得します。

## 3.2 数値・文字列に関する関数・命令

### 3.2.1 ARRAY2BINBLK\$

関数		
機能	数値配列を任意ブロック形式などの書式化文字列に変換します。	
書式	<(戻り値) 書式化文字列> = ARRAY2BINBLK\$(<①数値配列>, <②変換書式>)	
戻り値	戻り値	<書式化文字列> 文字列
	数値配列を元に、変換書式に従って変換された、書式化された文字列が得られます。	
パラメータ	①	<数値配列> 配列
	1次元の数値配列を指定します。	
	②	<変換書式> 配列
	配列から書式化文字列に変換する際に、どのような変換を行うか指定します。 複数の指定を、配列で与える事ができます。	
	変換書式で "comma" を指定しない時、GPIBやSCPIなどの規格で用いられる、任意ブロック形式(Definite-Length Block Response Data)の文字列に変換します。	
	変換書式	効果
	"int8"	任意ブロック形式文字列の変換時、整数値を 8ビット長のバイナリ値に変換します。 数値の範囲は、-128 から +127 までです。
	"uint8" または "byte"	任意ブロック形式文字列の変換時、整数値を 8ビット長のバイナリ値に変換します。 数値の範囲は、0 から +255 までです。
	"int16" または "short"	任意ブロック形式文字列の変換時、整数値を 16ビット長のバイナリ値に変換します。 数値の範囲は、-32768 から +32767 までです。
	"uint16"	任意ブロック形式文字列の変換時、整数値を 16ビット長のバイナリ値に変換します。 数値の範囲は、-0 から +65535 までです。
備考	"cint" または "int32"	任意ブロック形式文字列の変換時、整数値を 32ビット長のバイナリ値に変換します。
	"clng" または "int64"	任意ブロック形式文字列の変換時、整数値を 64ビット長のバイナリ値に変換します。
使用例	"float" or "csng"	任意ブロック形式文字列の変換時、実数値を 32ビット長のバイナリ値に変換します。
	"double" or "cdbl"	任意ブロック形式文字列の変換時、実数値を 64ビット長のバイナリ値に変換します。
	"comma"	数値をカンマ区切りで繋いだ文字列を得ます。
備考	<ul style="list-style-type: none"> <li>任意ブロック形式の詳細については、SCPIの規格書などを参照ください。</li> <li>任意ブロック形式の文字列から数値配列を得るには、BINBLK2ARRAY 関数を使用します。</li> </ul>	
使用例	PRINT ARRAY2BINBLK\$([ 1; 2; 3; 4; 5 ], "byte") ' 任意ブロック形式の文字列を得ます。	

## 3.2.2 ARRAY2STR\$

関数			
機 能	数値配列を指定したバイト単位で文字列に変換します。		
書 式	<(戻り値)変換文字列> = ARRAY2STR\$ ( <①数値配列> [, <②有効バイト単位> ] )		
戻り値	戻り値	<変換文字列>	文字列
	数値配列から指定したバイト単位で文字列に変換します。		
パラ メータ	①	<数値配列>	配列
	操作対象の数値配列を指定します。		
	②	<有効バイト単位>	数値
数値から文字列に変換するバイト単位数を指定します。 1 / 2 / 4 / 8 のどれかを選択します。省略すると、1 が採用されます。			
備 考	・ 有効バイト数の指定には、「STR2ARRAY」の分割バイト数を指定すると良いです。		
使用例1	LIST ARY ARY = STR2ARRAY("123") PRINT ARRAY2STR\$(ARY) ' "123" が出力されます。		
使用例2	' ? ARRAY2STR\$(配列, 1) を指定した際の動作と同じ処理を記述した例です。 LIST ARY ARY = STR2ARRAY("ABC")     ' 文字列から数値配列へ S\$ = "" FOR I=0 TO UBOUND(ARY) S\$ = S\$ + CHR\$(ARY(I)) NEXT I PRINT S\$		



## 3.2.3 BINBLK2ARRAY

関数			
機能	任意ブロック形式などの書式化文字列を数値配列に変換します。		
書式	<(戻り値)数値配列> = BINBLK2ARRAY(<①書式化文字列>, <②変換書式>)		
戻り値	戻り値	<数値配列>	配列
	書式化文字列で与えられた文字列を元に、変換書式に従って変換された、数値配列が得られます。		
パラメータ	①	<書式化文字列>	文字列
	主に、GPIBやSCPIなどの規格で用いられる、任意ブロック形式(Definite-Length Block Response Data)など、特定の書式に従って作られた文字列を指定します。		
	以下のような書式化文字列をサポートしています。		
	No	項目	説明
	1	任意ブロック形式	任意ブロックは、“#”で始まる文字列です。 例えば、“#512234xxxx”は、最初の“5”が桁数、“12234”が、xxxxの部分のバイト数を表します。 変換時、xxxxの部分がバイナリ値であるとして、変換書式で指定された型単位に、値を取り出して配列化します。
	2	カンマ区切り文字列	“123, 456, 789”という風に、数値を「,(カンマ)」記号で区切った文字列です。 カンマ記号単位に値を取り出して配列化します。
	3	数文字	「,(カンマ)」記号の無い数文字は、配列でない単一の値として扱います。
	カンマ区切り文字列と数文字を扱う時、GPIBやSCPIでよく扱われる、以下のような形式の書式文字列を、数値として扱います。		
	項目	説明	
	2進数数値	“#B”がプレフィクスに付くと、2進数として扱います。 例：“#B0110”は、&B0110として扱います。	
	8進数数値	“#Q”または“#0”がプレフィクスに付くと、8進数として扱います。 例：“#Q57”は、&057として扱います。	
	16進数数値	“#H”がプレフィクスに付くと、16進数として扱います。 例：“#H5A”は、&H5Aとして扱います。	
	ヘッダ付き数値	“NDCV+1.23V”というような文字列を検知すると、“NDCV”のヘッダ部分をスキップして+1.23の数値部分を扱います。	
	②	<変換書式>	配列
	書式化文字列から配列に変換する際に、どのような変換を行うか指定します。 複数の指定を、配列で与える事ができます。		
	変換書式	効果	
	“int8”	入力データは、8ビット長の整数値として扱います。 配列変換時、単精度整数が得られます。 数値の範囲は、-128 から +127 までです。	
	“uint8” または “byte”	入力データは、8ビット長の整数値として扱います。 配列変換時、単精度整数が得られます。 数値の範囲は、0 から +255 までです。	
	“int16” または “short”	入力データは、16ビット長の整数値として扱います。 配列変換時、単精度整数が得られます。 数値の範囲は、-32768 から +32767 までです。	

	"uint16"	入力データは、16ビット長の整数値として扱います。 配列変換時、単精度整数が得られます。 数値の範囲は、0 から +65535 までです。
	"cint" または "int32"	入力データは、32ビット長の整数値として扱います。 配列変換時、単精度整数が得られます。
	"clng" または "int64"	入力データは、64ビット長の整数値として扱います。 配列変換時、倍精度整数が得られます。
	"float" or "csng"	入力データは、32ビット長の実数値として扱います。 配列変換時、単精度実数が得られます。
	"double" or "cdbl"	入力データは、64ビット長の実数値として扱います。 配列変換時、倍精度実数が得られます。
	"big-endian"	入力データから値を読み取る順番を、ビッグエンディアンで行います。(指定しないと、リトルエンディアンで行います)
備 考	<ul style="list-style-type: none"> <li>任意ブロック形式の詳細については、SCPIの規格書などを参照ください。</li> <li>数値配列から任意ブロック形式の文字列を得るには、ARRAY2BINBLK\$ 関数を使用します。</li> </ul>	
使用例	PRINT BINBLK2ARRAY("#1512345", "byte") ' 画面に、[ 49, 50, 51, 52, 53 ] が出力されます。	

### 3.2.4 CBOOL

関数			
機 能	式の値を、論理型値に変換します。		
書 式	<(戻り値) 論理型値> = CBOOL(<①式>)		
戻り値	戻り値	<論理型値>	真偽値
	式の値を、論理型値に変換した値が得られます。		
パラ メータ	①	<式>	式
	数値・文字列 どちらでも指定できます。 文字列で渡す場合、数字として認識できる形式で渡してください。 数字に変換できない場合、エラーとなります。		
使用例1	PRINT CBOOL(1), CBOOL(0) TRUE FALSE		
使用例2	PRINT CDBL("1"), CBOOL("0") TRUE FALSE		

## 3.2.5 CDBL

関数			
機 能	式の値を、倍精度実数値に変換します。		
書 式	<(戻り値)倍精度実数値> = CDBL(<①式>)		
戻り値	戻り値	<倍精度実数値>	数値
	式の値を、倍精度実数値に変換した値が得られます。		
パラ メータ	①	<式>	式
	数値・文字列 どちらでも指定できます。 文字列で渡す場合、数字として認識できる形式で渡してください。 数字に変換できない場合、エラーとなります。		
使用例1	PRINT CDBL(123.45) 123.45		
使用例2	PRINT CDBL("123.45") 123.45		

## 3.2.6 CINT

関数			
機 能	式の値を、単精度整数値に変換します。		
書 式	<(戻り値)単精度整数値> = CINT(<①式>)		
戻り値	戻り値	<単精度整数値>	数値
	式の値を、単精度整数値に変換した値が得られます。		
パラ メータ	①	<式>	式
	数値・文字列 どちらでも指定できます。 文字列で渡す場合、数字として認識できる形式で渡してください。 数字に変換できない場合、エラーとなります。		
使用例1	PRINT CINT(123.45) 123		
使用例2	PRINT CINT("123.45") 123		

## 3.2.7 CLNG

関数			
機 能	式の値を、倍精度整数値に変換します。		
書 式	<(戻り値)倍精度整数値> = CLNG(<①式>)		
戻り値	戻り値	<倍精度整数値>	数値
	式の値を、倍精度整数値に変換した値が得られます。		
パラ メータ	①	<式>	式
	数値・文字列 どちらでも指定できます。 文字列で渡す場合、数字として認識できる形式で渡してください。 数字に変換できない場合、エラーとなります。		
使用例1	PRINT CLNG(123.45) 123		
使用例2	PRINT CLNG("123.45") 123		

## 3.2.8 CSNG

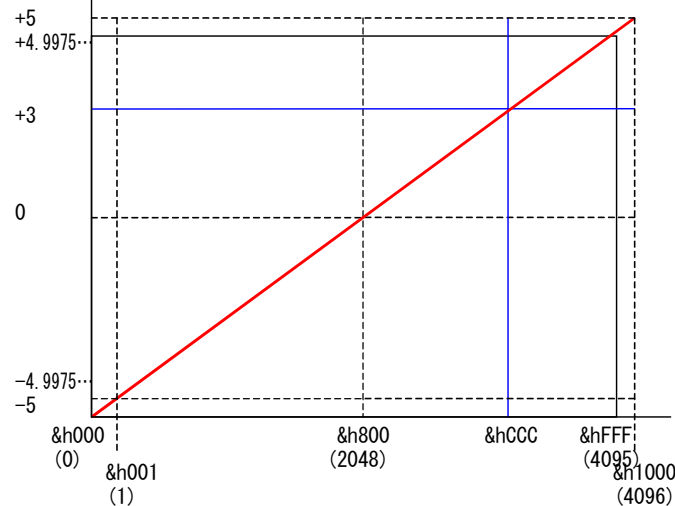
関数			
機 能	式の値を、単精度実数値に変換します。		
書 式	<(戻り値)単精度実数値> = CSNG(<①式>)		
戻り値	戻り値	<単精度実数値>	数値
	式の値を、単精度実数値に変換した値が得られます。		
パラ メータ	①	<式>	式
	数値・文字列 どちらでも指定できます。 文字列で渡す場合、数字として認識できる形式で渡してください。 数字に変換できない場合、エラーとなります。		
使用例1	PRINT CDBL(123.45) 123.45		
使用例2	PRINT CDBL("123.45") 123.45		

## 3.2.9 CLOCK2DATETIMESTR\$

関数													
機 能	CLOCK関数の値を日時文字列に変換します。												
書 式	＜(戻り値) 日時文字列＞ = CLOCK2DATETIMESTR\$( ＜①経過秒＞ [ , ＜②オプション＞ ] )												
戻り値	戻り値	＜日時文字列＞											
	「年/月/日 時:分:秒」の形式で文字列を得ます。												
パラ メータ	①	＜経過秒＞											
	CLOCK関数で得られる、経過秒を引数に与えます。												
	②	＜オプション＞											
	変換時に、ミリ秒、マイクロ秒、ナノ秒の単位まで情報を付加できます。 何も指定しないと、秒のみです。												
	<table><tr><th>設定値</th><th>内容</th></tr><tr><td>"m"</td><td>ミリ秒単位で得られます。</td></tr><tr><td>"u"</td><td>マイクロ秒単位で得られます。</td></tr><tr><td>"n"</td><td>ナノ秒単位で得られます。</td></tr><tr><td>"iso8601"</td><td>ISO8601仕様(基本形式)に変換します。 出力する時刻はローカル標準時に合わせます。</td></tr><tr><td>"iso8601ext"</td><td>ISO8601仕様(拡張形式)に変換します。 出力する時刻はローカル標準時に合わせます。</td></tr></table>		設定値	内容	"m"	ミリ秒単位で得られます。	"u"	マイクロ秒単位で得られます。	"n"	ナノ秒単位で得られます。	"iso8601"	ISO8601仕様(基本形式)に変換します。 出力する時刻はローカル標準時に合わせます。	"iso8601ext"
設定値	内容												
"m"	ミリ秒単位で得られます。												
"u"	マイクロ秒単位で得られます。												
"n"	ナノ秒単位で得られます。												
"iso8601"	ISO8601仕様(基本形式)に変換します。 出力する時刻はローカル標準時に合わせます。												
"iso8601ext"	ISO8601仕様(拡張形式)に変換します。 出力する時刻はローカル標準時に合わせます。												
備 考	・ オプションで、"iso8601"と"iso8601ext"は排他の関係です。 "m"と"u"と"n"も排他の関係です。 ・ オプションで、文字列配列で、"iso8601"と"m"を指定可能です。												
使用例1	? CLOCK2DATETIMESTR\$(CLOCK) ' 「2022/02/01 18:37:09」のような形式で出力が得られます。												
使用例2	? CLOCK2DATETIMESTR\$(CLOCK, [ "iso8601ext"; "m" ]) ' 「2022-02-24T20:37:50.495+09:00」のような形式で出力が得られます。												

3.2.10 CONVBIN

関数									
機 能	連続量の値(アナログデータ)からバイナリ値(デジタルデータ)に変換します。								
書 式	<(戻り値)変換後値> = CONVBIN(<①変換元値>, <②分解能>, <③レンジ上限値>, <④レンジ下限値> [, <⑤範囲外オプション> ] )								
戻り値	戻り値	<変換後値> 配列							
	以下の式で、連続量からバイナリ値に量子化変換した、バイナリ値(単精度整数)の配列が得られます。  <バイナリ値> = (<分解能> * (<変換元値> - <レンジ下限値>)) / (<レンジ上限値> - レンジ下限値)								
パラメータ	①	<変換元値> 配列							
	変換する元となる連続量の値の配列を与えます。								
	②	<分解能> 数値							
	量子化変換を行う際に用いる指標として、バイナリ値のビット分解能を指定します。 例えば、以下のように指定します。 12ビット分解能 : 4096 16ビット分解能 : 65536								
	③	<レンジ上限値> 数値							
	量子化変換を行う際に用いる指標として、連続量のレンジ範囲の上限値を指定します。								
	④	<レンジ下限値> 数値							
	量子化変換を行う際に用いる指標として、連続量のレンジ範囲の下限値を指定します。								
	⑤	<範囲外オプション> 数値							
	上限値と下限値の範囲を超えるような、変換元値が与えられた際の挙動を指定します。 計算式の関係上、変換元値に与えられる上限値は、レンジ上限値 - 1LSB になる為のオプションです。								
<table><tr><td>範囲外オプション値</td><td>挙動</td></tr><tr><td>0</td><td>変換結果が、バイナリ値の上限から下限を超えるとエラーとします。</td></tr><tr><td>1</td><td>変換元値にレンジ上限値を与えた時のみ、バイナリ値の上限値になるように補正します。 省略時、この設定が採用されます。</td></tr><tr><td>2</td><td>変換結果が、バイナリ値の上限を越えると上限値に、バイナリ値の下限を下回ると下限値になるように補正します。</td></tr></table>		範囲外オプション値	挙動	0	変換結果が、バイナリ値の上限から下限を超えるとエラーとします。	1	変換元値にレンジ上限値を与えた時のみ、バイナリ値の上限値になるように補正します。 省略時、この設定が採用されます。	2	変換結果が、バイナリ値の上限を越えると上限値に、バイナリ値の下限を下回ると下限値になるように補正します。
範囲外オプション値	挙動								
0	変換結果が、バイナリ値の上限から下限を超えるとエラーとします。								
1	変換元値にレンジ上限値を与えた時のみ、バイナリ値の上限値になるように補正します。 省略時、この設定が採用されます。								
2	変換結果が、バイナリ値の上限を越えると上限値に、バイナリ値の下限を下回ると下限値になるように補正します。								
備 考	・ CONVPHY で、バイナリ値から連続量を得る事ができます。 ・ 変換結果の値が、指定した分解能の範囲外になるとエラーとなります。 ・								
解 説	コンピュータで電圧や電流などの連続量(アナログデータ)を扱う際、例えば 3Vは バイナリ値(デジタルデータ)で &h0CCC とする。というように量子化して扱います。 関係図で表すと、以下のようになります。								

	<div><p>本関数は、この量子化の計算を補助します。</p></div>
使用例1	<div>分解能4096(12bit)でレンジ範囲 +5~-5の、連続量の値が 3.0 の場合のバイナリ値を得ます。</div> <div>PRINT CONVBIN(3.0, 4096, 5, -5)</div>
使用例2	<div>分解能65536(16bit)でレンジ範囲 +5~-5の、連続量の値が 3.0 の場合のバイナリ値を得ます。</div> <div>PRINT CONVBIN(3.0, 65536, 5, -5)</div>
使用例3	<div>変換元値に配列を与えて、一気に変換させます。</div> <div>PRINT CONVBIN([0; 1; 2; 3; 4], 4096, 5, -5)</div>



## 3.2.11 CONVPHY

関数		
機能	バイナリ値(デジタルデータ)から連続量の値(アナログデータ)に変換します。	
書式	<(戻り値)変換後値> = CONVPHY(<①変換元値>, <②分解能>, <③レンジ上限値>, <④レンジ下限値> )	
戻り値	戻り値	<変換後値> 配列
	<p>以下の式で、バイナリ値から連続量へ量子化変換した、連続量の値(倍精度実数)の配列が得られます。</p> $\text{<アナログ値>} = (\text{<レンジ下限値>} - \text{<レンジ上限値>}) * \text{<変換元値>} / \text{<分解能>} + \text{<レンジ下限値>}$	
パラメータ	①	<変換元値> 配列
	変換する元となるバイナリ値の配列を与えます。	
	②	<分解能> 数値
	<p>量子化変換を行う際に用いる指標として、バイナリ値のビット分解能を指定します。 例えば、以下のように指定します。 12ビット分解能：4096 16ビット分解能：65536</p>	
	③	<レンジ上限値> 数値
	量子化変換を行う際に用いる指標として、連続量のレンジ範囲の上限値を指定します。	
	④	<レンジ下限値> 数値
	量子化変換を行う際に用いる指標として、連続量のレンジ範囲の下限値を指定します。	
備考	<ul style="list-style-type: none"> <li>CONVBIN で、連続量からバイナリ値を得ることができます。</li> <li>変換結果の値が、指定したレンジの範囲外になるとエラーとなります。</li> <li>.</li> </ul>	
使用例1	<p>分解能4096でレンジ範囲 +5～-5の、バイナリ値が &amp;HCCCの場合の連続量の値を得ます。</p> <pre>PRINT CONVPHY(&amp;HCCC, 4096, 5, -5)</pre>	
使用例2	<p>変換元値に配列を与えて、一気に変換します。</p> <pre>PRINT CONVPHY([ &amp;h000; &amp;h333; &amp;h666; &amp;h999; &amp;hCCC; &amp;hFFF ], 4096, 5, -5)</pre>	

### 3.2.12 CONVUNIT

関数			
機 能	値を単位変換します。		
書 式	<(戻り値)変換後値> = CONVUNIT(<①変換元値>, <②変換元単位名>, <③変換後単位名> )		
戻り値	戻り値	<変換後値>	配列
	単位変換した後の値配列が得られます。		
パラメータ	①	<変換元値>	配列
	単位変換する元となる値の配列を与えます。		
	②	<変換元単位名>	文字列
	<変換元値>の単位名を示します。 指定可能な単位名は「<単位変換名一覧>」を、可能な組み合わせは、グループ項同士で可能です。		
	③	<変換後単位名>	文字列
	単位変換する単位名を示します。 指定可能な単位名は「<単位変換名一覧>」を、可能な組み合わせは、グループ項同士で可能です。		
備 考	・		
使用例	45° (度)の値を、ラジアン値に変換します。  PRINT CONVUNIT(45, "deg", "rad")		

#### 3.2.12.1 <単位変換名一覧>

グループ	単位名	項目	説明
A	"rad"	ラジアン	
	"deg"	度(°)	
B	"mm"	ミリメートル	
	"cm"	センチメートル	
	"m"	メートル	
	"km"	キロメートル	
	"in"	インチ	
	"ft"	フィート	
	"yd"	ヤード	
	"mi"	マイル	
C	"kg"	キログラム	
	"lb"	ポンド	
D	"celsius"	摂氏	
	"fahrenheit"	華氏	

変換前単位名と変換後単位名で、指定可能な組み合わせは、同じグループ項のみ可能です。

3. 2. 13 DATEADD

関数																
機 能	日付時刻値に対して、指定した時間間隔に加算した値を返します。															
書 式	<(戻り値) 日付時刻値> = DATEADD(<①指定時間間隔>, <②加算値>, <③日付時刻値> )															
戻り値	戻り値	<日付時刻値> 日付時刻 日付時刻値に対して、加算した結果の値を、日付時刻型の値で得られます。														
パラ メータ	①	<指定時間間隔> 文字列 日付時刻値に対して、加算する対象を文字列で指定します。														
	<table><tr><th>設定値</th><th>内容</th></tr><tr><td>yyyy</td><td>年に対して、加算します。</td></tr><tr><td>m</td><td>月に対して、加算します。</td></tr><tr><td>d</td><td>日に対して、加算します。</td></tr><tr><td>h</td><td>時に対して、加算します。</td></tr><tr><td>n</td><td>分に対して、加算します。</td></tr><tr><td>s</td><td>秒に対して、加算します。</td></tr></table>		設定値	内容	yyyy	年に対して、加算します。	m	月に対して、加算します。	d	日に対して、加算します。	h	時に対して、加算します。	n	分に対して、加算します。	s	秒に対して、加算します。
	設定値	内容														
	yyyy	年に対して、加算します。														
	m	月に対して、加算します。														
	d	日に対して、加算します。														
	h	時に対して、加算します。														
n	分に対して、加算します。															
s	秒に対して、加算します。															
②	<加算値> 数値 指定時間間隔で指定した対象に対して、加算する値を指定します。 負数を指定すると、減算できます。															
③	<日付時刻値> 日付時刻 加算対象とする、日付時刻型の値を指定します。															
使用例	PRINT DATEADD("yyyy", 1, CDATETIME("2020/4/1")) ' 「2021/04/01 00:00:00」が表示されます。  PRINT DATEADD("m", -1, CDATETIME("2020/10/31")) ' 「2020/09/30 00:00:00」が表示されます。  PRINT DATEADD("s", 20, CDATETIME("2020/10/31 01:02:03")) ' 「2020/10/31 01:02:23」が表示されます。															

## 3.2.14 DATETIMESTR2CLOCK

関数			
機 能	日時文字列を秒単位 (CLOCK関数相当) に変換します。		
書 式	<(戻り値) 経過秒> = DATETIMESTR2CLOCK ( <①日時文字列> )		
戻り値	戻り値	<経過秒>	数値
	CLOCK関数と同じ、経過秒の形式で値を得ます。 「年/月/日 時:分:秒」の形式で文字列を得ます。		
パラ メータ	①	<日時文字列>	文字列
	CLOCK2DATETIMESTR\$ で得られた日時文字列を指定します。		
備 考	<ul style="list-style-type: none"> <li>IS08601形式の文字列を指定できます。</li> <li>また、区切り記号に "T" でなく " "(空白) を指定する事も可能です。(RFC3339 でサポートされる形式に相当します)</li> </ul>		
使用例	SEC = DATETIMESTR2CLOCK ("2022/02/01 18:37:09") ' 日時文字列から、秒換算の値が得られます。		

## 3.2.15 DAY

関数		
機 能	指定日（文字列）の日を返します。	
書 式	<(戻り値) 日> = DAY(<①年月日文字列> [, <②エラー送出フラグ> ])	
戻り値	戻り値	数値
	<日> 年月日文字列の日数の値を返します。	
パラ メータ	①	文字列
	<年月日文字列> 日を求める日付(「年/月/日」の形式)を指定します。	
	②	真偽値
	<エラー送出フラグ> TRUEを指定すると、不正な年月日文字列を渡すと、エラーとします。 FALSEを指定すると、不正な年月日文字列の場合に、0を返します。 省略すると、FALSEとします。	
備 考	<年月日文字列>が日付でない場合、0を返します。 閏年ではない年に閏日を入力すると、翌月の初日に繰り越されます。	
使用例1	NUM = DAY("2013/07/15")  入力値 2013/07/15 の日付部分 15 を変数NUMへ代入します。	

## 3. 2. 16 FORMAT\$

関数			
機 能	値を指定した書式に従って文字列変換します。		
書 式	<(戻り値)変換文字列> = FORMAT\$(<①値>, <②変換書式> )		
戻り値	戻り値	<変換文字列>	文字列
	書式に沿って文字列変換された結果を得ます。		
パラメータ	①	<値>	値
	文字列変換する対象となる値を指定します。		
	②	<変換書式>	文字列
	値を文字列変換する際の書式を指定します。 値が数値の場合、「3. 2. 16. 1 <FORMAT\$の数値表示書式表>」を参照ください。 値が日付時刻型の場合、「3. 2. 16. 2 <FORMAT\$の日付時刻表示書式表>」を参照ください。 値が文字列の場合、「3. 2. 16. 3 <FORMAT\$の文字列表示書式表>」を参照ください。		
使用例1	' 1000単位の区切り記号を表示します。 ? FORMAT\$(123456789, "##,##0.0") 123,456,789.0		
使用例2	' パーセント表示します。 ? FORMAT\$(0.25, "0.00%") 25.00%		
使用例3	' 時:分:秒表示します。 ? FORMAT\$(CDATETIME("17:04:23"), "AMPM hh:nn:ss") 午後 05:04:23		
使用例4	' 年/月/日表示します。 ? FORMAT\$(CDATETIME("2019/05/27"), "yy/mm/dd") 19/05/27		
使用例5	' 和暦表示します。 ? FORMAT\$(CDATETIME("2020/05/27"), "ggge m/d") 令和2 5/27		
使用例6	' 大文字表示します。 ? FORMAT\$("hello AJAN", ">") HELLO AJAN		

## 3. 2. 16. 1 &lt;FORMAT\$の数値表示書式表&gt;

文字	効果
なし	指定した数値をそのまま表示
0	桁位置が桁数を指定する時に使う。”0”一つで1桁分の数値を表す。 数値の桁数が満たない場合、”0”が付加される。
#	桁位置が桁数を指定する時に使う。”#”一つで1桁分の数値を表す。 数値の桁数が満たない場合、省略される。
.	“0”と”#”を組み合わせて、小数点の位置を指定する際に使用。
%	数値を100倍して、パーセント記号(“%”)を付ける。
,	1000単位の区切り記号を挿入する。 ”,”の前後に”0”または”#”を付加する事。 なお、整数部の右端に”,”を1つ以上指定すると、1000単位で割った値となる。

¥	すぐ後に続く 1 文字を、そのまま表示
"..."	“(ダブルクォーテーション)で囲った文字列を、そのまま表示

### 3.2.16.2 <FORMAT\$の日付時刻表示書式表>

文字	効果
c	yyyy/mm/dd h:nn:ss と同じ。
d	日付を表示。
dd	日付を表示。但し、1 桁の場合、0 が付加。
ddd	曜日を英語の省略形で表示。
dddd	曜日を英語で表示。
aaa	曜日を日本語の省略形で表示。
aaaa	曜日を日本語で表示。
dddddd	yyyy/mm/dd という風に表示。
dddddd	yyyy 年 mm 月 dd 日 という風に表示。
w	曜日を表す数値で表示。日曜が 1、土曜が 7。
m	月を表示。
mm	月を表示。但し、1 桁の場合、0 が付加。
mmm	月の名前を英語の省略形で表示。
mmmm	月の名前を英語で表示。
q	四半期を表す数値を表示。1-4。
g	元号の頭文字を表示。M, T, S, H, R
gg	元号の先頭 1 文字を表示。明、大、昭、平、令
ggg	元号を表示。明治、大正、昭和、平成、令和
e	元号に基づく和暦の値を表示。
ee	元号に基づく和暦の値を表示。但し、1 桁の場合、0 が付加。
yy	西暦の下 2 桁を表示。
yyyy	西暦の 4 桁全てを表示。
h	時間を表示。 なお、AM/PM 表記になると 0-11、そうでないと 0-23。
hh	時間を表示。但し、1 桁の場合、0 が付加。
n	分を表示。
nn	分を表示。但し、1 桁の場合、0 が付加。
s	秒を表示。
ss	秒を表示。但し、1 桁の場合、0 が付加。
ttttt	h:nn:ss という風に表示。
AM/PM	午前は AM、午後は PM で表示。
am/pm	午前は am、午後は pm で表示。
A/P	午前は A、午後は P で表示。
a/p	午前は a、午後は p で表示。
AMPM	午前は「午前」、午後は「午後」で表示。

### 3.2.16.3 <FORMAT\$の文字列表示書式表>

文字	効果
@	一つの文字またはスペースを表示。
&	一つの文字を表示。
<	小文字にする。

## AJAN 拡張コマンドリファレンス

>	大文字にする。
!	文字を右から左でなく、左から右の順に埋める。



## 3.2.17 FORMATDATETIME\$

関数		
機 能	日付時刻値を指定した書式に従って文字列化します。	
書 式	<(戻り値) 日付時刻文字列> = FORMATDATETIME\$(<①日付時刻値> [, <②変換設定値>] )	
戻り値	戻り値	文字列
	<日付時刻文字列> 書式に沿って文字列変換された結果を得ます。	
パラ メータ	①	日付時刻
	<日付時刻値> 文字列変換する対象となる日付時刻値を指定します。	
	②	数値
	<変換設定値> 値を文字列変換する際の書式を指定します。 以下に対応しています。	
	設定値	変換内容
	1	RFC1123 仕様に変換します。日付の間を“-”で合わせます。 出力する時刻は UTC(協定世界時)に合わせます。
	2	RFC1123 仕様に変換します。日付の間を“-”で合わせます。 出力する時刻はローカル標準時に合わせます。
	3	RFC1123 仕様に変換します。 出力する時刻は UTC(協定世界時)に合わせます。
	4	RFC1123 仕様に変換します。 出力する時刻はローカル標準時に合わせます。
	5	ISO8601 仕様(基本形式)に変換します。 出力する時刻はローカル標準時に合わせます。
	6	ISO8601 仕様(拡張形式)に変換します。 出力する時刻はローカル標準時に合わせます。
	7	年/月/日 時:分:秒 の形式に変換します。
	8	ISO8601 仕様(基本形式)に変換します。 出力する時刻は UTC(協定世界時)に合わせます。
	9	ISO8601 仕様(拡張形式)に変換します。 出力する時刻は UTC(協定世界時)に合わせます。
	15	5 の設定に対し、ミリ秒表記が付加されます。
	16	6 の設定に対し、ミリ秒表記が付加されます。
	17	7 の設定に対し、ミリ秒表記が付加されます。
	18	8 の設定に対し、ミリ秒表記が付加されます。
	19	9 の設定に対し、ミリ秒表記が付加されます。
	省略時、設定値は「1」として扱われます。	
使用例	DATETIME D D = DATE\$ D = D + CDATETIME(TIME\$)  ? FormatDateTime\$(D) ' "Wed, 25-Jul-2018 10:32:49 GMT" のような出力が得られます。	

### 3. 2. 18 HASH\$

関数																										
機 能	文字列に対してハッシュを得ます																									
書 式	<(戻り値)ハッシュ値> = HASH\$( <①ID> , <②元文字列> )																									
戻り値	戻り値	文字列																								
	<ハッシュ値> 元文字列に対してハッシュした結果が、バイナリ文字列形式で得られます。																									
パラ メータ	①	文字列																								
	<ID> ハッシュを得る為のアルゴリズムを、文字列形式で指定します。 以下が指定可能です。																									
	<table><tr><td>ID 引数</td><td>アルゴリズム</td><td>戻り値のサイズ</td></tr><tr><td>SHA512</td><td>SHA-2、512bit 長</td><td>64</td></tr><tr><td>SHA384</td><td>SHA-2、384bit 長</td><td>48</td></tr><tr><td>SHA256</td><td>SHA-2、256bit 長</td><td>32</td></tr><tr><td>SHA224</td><td>SHA-2、224bit 長</td><td>28</td></tr><tr><td>SHA1</td><td>SHA-1、160bit 長</td><td>20</td></tr><tr><td>MD5</td><td>MD5</td><td>16</td></tr><tr><td>WHIRLPOOL</td><td>Whirlpool</td><td>64</td></tr></table>		ID 引数	アルゴリズム	戻り値のサイズ	SHA512	SHA-2、512bit 長	64	SHA384	SHA-2、384bit 長	48	SHA256	SHA-2、256bit 長	32	SHA224	SHA-2、224bit 長	28	SHA1	SHA-1、160bit 長	20	MD5	MD5	16	WHIRLPOOL	Whirlpool	64
	ID 引数	アルゴリズム	戻り値のサイズ																							
SHA512	SHA-2、512bit 長	64																								
SHA384	SHA-2、384bit 長	48																								
SHA256	SHA-2、256bit 長	32																								
SHA224	SHA-2、224bit 長	28																								
SHA1	SHA-1、160bit 長	20																								
MD5	MD5	16																								
WHIRLPOOL	Whirlpool	64																								
	②	文字列																								
	<元文字列> 対象となる文字列を指定します。																									
備 考	・ 得られるハッシュ値を暗号化やパスワードなどに用いる場合、使用するアルゴリズムをよくよく検討してください。 一般に、MD5アルゴリズムは、脆弱性が見つかっている事が知られています。 ・ 得られるハッシュ値はバイナリ文字列形式のため、そのままPRINTすると文字化けします。																									
使用例	A\$ = HASH\$( “SHA512” , “hello Interface world” ) B\$ = HASH\$( “SHA512” , “hello Interface WORLD” ) IF A\$ = B\$ THEN PRINT “ハッシュ同士の比較は一致しました” ELSE PRINT “ハッシュ同士の比較は不一致でした” END IF  SHA-512を使って、文字列「” hello Interface world” 」および「 “hello Interface WORLD” 」のハッシュ値を得ます。 その後、得られたハッシュ値を比較しています。																									

## 3. 2. 19 HOUR

関数		
機 能	指定時刻（文字列）の時を返します。	
書 式	<(戻り値)時> = HOUR(<①時分秒文字列> [, <②エラー送出フラグ> ])	
戻り値	戻り値	数値
	<時> 時分秒文字列の時の値を返します。	
パラ メータ	①	文字列
	<時分秒文字列> 時を求める時刻(「時:分:秒」の形式)を指定します。	
	②	真偽値
	<エラー送出フラグ> TRUEを指定すると、不正な時分秒文字列を渡すと、エラーとします。 FALSEを指定すると、不正な時分秒文字列の場合に、0を返します。 省略すると、FALSEとします。	
備 考	<時分秒文字列>が時刻でない場合、0を返します。 時刻の書式は、24時間表記とします。 閏年ではない年に閏秒を入力すると、翌分の0秒に繰り越されます。	
使用例	NUM=HOUR("14:03:15")  変数NUMに指定時刻「14:03:15」から時の部分「14」を代入します。	

## 3.2.20 ICONV\$

関数		
機能	文字列に対して、文字コード変換を行います。	
書式	<(戻り値)変換後文字列> = ICONV\$(<①変換元文字列>, <②変換先コードセット名>, <③変換元コードセット名> [, <④オプション> ] )	
戻り値	戻り値	<変換後文字列> 変換先コードセット名で指定した、文字コード変換された文字列が得られます。
パラメータ	①	<変換元文字列> 文字コード変換したい元の文字列を指定します。
	②	<変換先コードセット名> 文字コード変換したい先の、コードセット名を指定します。 代表的なコードセット名は、「UTF-8」や「CP932」です。
	③	<変換元コードセット名> 文字コード変換したい元の、コードセット名を指定します。
	④	<オプション> 文字変換時の特殊な動作を指定できます。 省略時は、通常の動作を行います。
	オプション	動作
	"ignore"	文字変換できない時、先頭から変換できない文字をスキップし、次の文字から変換を挑戦します。 全ての文字が変換できないと、空文字列が得られます。
備考	<ul style="list-style-type: none"> <li>指定可能なコードセットは、Linuxの端末上で「iconv --list」コマンドで確認してください。</li> <li>Interface Linux SystemおよびAJANのコードセットは「UTF-8」です。 それ以外のコードセットに変換した文字列をPRINTした場合、得られる表示結果は不定です。 多くの場合、文字化けと呼ばれる文字が判読できない現象を起こします。</li> <li>参考までに、日本語表示されたWindowsで、一般によく使用されるコードセットは「CP932」です。</li> <li>IDE上で UTF-8以外のコードセットに変換した文字列を表示した場合、Linux端末上で実行した結果と異なる事があります。 (IDEの場合、表示自体が行われない場合が観測されています)</li> <li>変換元文字列の末尾が、変換できない文字の場合、それ以上の変換が行われない場合があります。</li> </ul>	
使用例1	<pre>S\$ = ICONV\$("テスト", "CP932", "UTF-8") STR_FWRITEALL "memo.txt", S\$</pre> <p>“テスト”という UTF-8形式の文字列を、CP932形式のエンコード文字列にして、S\$に代入します。 次に、STR_FWRITEALLコマンドで、変換した文字列を“memo.txt”ファイルに書き込みます。このファイルは、日本語環境のWindowsで読めます。</p>	
使用例2	<pre>S\$ = "おはようございます" ' ↓先頭2バイト目から文字列を取り出しているため、不正なUTF-8文字列です。 S2\$ = MIDB\$(S\$, 2) ' 不正なUTF-8文字列を、“ignore”オプションで、正常な文字列を取り出します。 S3\$ = ICONV\$(S2\$, "UTF-8", "UTF-8", "ignore") PRINT S3\$ ' 「はようございます」が表示されます</pre> <p>不正なUTF-8文字列から、正常なUTF-8文字列を取り出す呼び出し事例です。</p>	

## 3. 2. 21 JOIN\$

関数			
機 能	1次元配列の文字列を、挿入記号を間に挟んで、1つの文字列に結合します。		
書 式	<(戻り値) 結合後文字列> = JOIN\$(<①文字列配列> [, <②結合文字列>] )		
戻り値	戻り値	<結合後文字列>	文字列
	1次元配列の文字列を、挿入記号を間に挟んで、1つの文字列に結合した文字列を返します。		
パラ メータ	①	<文字列配列>	配列
	結合対象となる文字列の1次元配列。		
	②	<結合文字列>	文字列
	文字列と文字列の間に挟む結合文字列。 省略すると、1文字の空白(" ")となります。		
使用例1	DIM A\$(3) A\$(0) = "a" A\$(1) = "b" A\$(2) = "c" A\$(3) = "d"  AA\$ = JOIN\$(A\$, "+" ) A\$の配列の各文字列に「+」の文字を追加 「a+b+c+d」		

## 3. 2. 22 LCASE\$

関数			
機 能	文字列中の大文字（半角英字）を小文字に変換します。		
書 式	<(戻り値) 小文字文字列> = LCASE\$(<①文字列>)		
戻り値	戻り値	<小文字文字列>	文字列
	文字列中の大文字（半角英字）を小文字に変換した結果が得られます。		
パラ メータ	①	<文字列>	文字列
	変換対象の文字列を指定します。		
備 考	小文字を大文字に変換するには「UCASE\$」を使用します。		
使用例1	SMP\$=LCASE\$("SAMPLE")  変数SMP\$へ指定値「SAMPLE」を小文字へ変換した値 「sample」を代入します。		
使用例2	SMP\$=LCASE\$("samPLE")  変数SMP\$へ指定値「samPLE」を小文字へ変換した値 「sample」を代入します。		

## 3. 2. 23 LEFT\$

関数			
機 能	文字列の左側から任意の長さの文字列を抜き出します。		
書 式	<(戻り値) 抜き出し後文字列> = LEFT\$(<①文字列>, <②抜き出す文字数>)		
戻り値	戻り値	<抜き出し後文字列>	文字列
	文字列の左側から任意の長さの文字列を抜き出した文字列が得られます。		
パラメータ	①	<文字列>	文字列
	抜き出し対象の文字列を指定します。		
	②	<抜き出す文字数>	数値
使用例1	SMP\$=LEFT\$("SAMPLE", 4)		
	SMP\$へ指定文字列「SAMPLE」を左から順に 4 文字抜き出した値「SAMP」を代入します。		
使用例2	SMP\$=LEFT\$("あいう", 4)		
	指定文字列「あいう」の文字数「3」に対し、抜き出す文字数「4」が上回っているため、文字列「あいう」をそのまま変数SMP\$へ代入します。		

## 3. 2. 24 LEFTB\$

関数			
機 能	文字列の左側から任意のバイト数分の文字列を抜き出します。		
書 式	<(戻り値) 抜き出し後文字列> = LEFTB\$(<①文字列>, <②抜き出すバイト数>)		
戻り値	戻り値	<抜き出し後文字列>	文字列
	文字列の左側から任意のバイト数分の文字列を抜き出した文字列が得られます。		
パラメータ	①	<文字列>	文字列
	抜き出し対象の文字列を指定します。		
	②	<抜き出すバイト数>	数値
備 考	抜き出す文字列の長さを0以上のバイト数単位で指定します。		
	<文字列>の長さより大きい場合、<文字列>をそのまま返します。		
使用例	<ul style="list-style-type: none"> <li>この関数は、文字列のデータをバイトデータとして扱う為に用意されています。</li> <li>得られる値は文字列形式ですが、PRINT文などで表示できない場合があります。(PRINT文で表示するには、UTF-8形式の文字列である必要があります)</li> </ul>		
	SRC\$=CHRB\$(0)+CHRB\$(1)+CHRB\$(2)+CHRB\$(3) SMP\$=LEFTB\$(SRC\$, 2)  SRC\$は、4バイトのバイトデータ(文字列形式)です。 LEFTB\$により、先頭2バイトのバイトデータを抜き出し、SMP\$には「CHRB\$(0)+CHRB\$(1)」のバイトデータが得られます。		

## 3.2.25 MATH ISEQ

関数			
機 能	2つの数値の差分を取って等値比較を行います		
書 式	<(戻り値)判定値> = MATH ISEQ(<①左辺値>, <①右辺値> [, <②許容値>] )		
戻り値	戻り値	<判定値>	真偽値
	<左辺値>と<右辺値>の差分を取って、<許容値>の範囲内にあれば、等値と見なし TRUE を返します。そうでなければ FALSEを返します。		
パラ メータ	①	<左辺値>, <右辺値>	数値
	等値比較を行う値の対象です。		
	②	<許容値>	数値
備 考	等値比較に用いる許容値です。		
	指定しない場合、<左辺値>, <右辺値>の計算機イプシロンが用いられます。		
備 考	<ul style="list-style-type: none"> <li>等値判定を行う際は、許容値の指定に注意してください。 実数演算は、一般的に、大きな値を用いると誤差が大きくなります。</li> <li>計算機イプシロンとは、実数(浮動小数点数)において、1より大きい最小の値と、1との差を指します。</li> <li>左辺値と右辺値に配列を指定すると、配列の各値と等値比較します。 この時、2つの配列の要素数を合わせてください。</li> </ul>		
使用例	? MATH ISEQ(1.0 - 0.9, 0.1) TRUE		

## 3. 2. 26 MINUTE

関数			
機 能	指定時刻（文字列）の分を返します。		
書 式	<(戻り値) 分> = MINUTE(<①時分秒文字列> [, <②エラー送出フラグ> ])		
戻り値	戻り値	<分>	数値
	時分秒文字列の分の値を返します。		
パラ メータ	①	<時分秒文字列>	文字列
	時を求める時刻(「時:分:秒」の形式)を指定します。		
	②	<エラー送出フラグ>	真偽値
	TRUEを指定すると、不正な時分秒文字列を渡すと、エラーとします。 FALSEを指定すると、不正な時分秒文字列の場合に、0を返します。 省略すると、FALSEとします。		
備 考	<文字列>が時刻でない場合、0を返します。 時刻の書式は、24時間表記とします。 閏年ではない年に閏秒を入力すると、翌分の0秒に繰り越されます。		
使用例1	NUM = MINUTE("14:03:15")  変数NUMにMINUTEの実行結果3を代入します。		

## 3. 2. 27 MKUUID\$

関数			
機 能	UUID文字列を生成します。		
書 式	<(戻り値) UUID文字列> = MKUUID\$()		
戻り値	戻り値	<UUID文字列>	文字列
	UUID文字列を生成して得ます。		
備 考	<ul style="list-style-type: none"> <li>• UUID(Universally Unique Identifier)は、ソフトウェア上でオブジェクトを一意に識別する為の識別子です。 128ビット長の数値ですが、この関数では、使用例のような文字列を返します。</li> <li>• UUID値の比較を行うには、UUIDCOMP 関数を使用してください。</li> </ul>		
使用例	? MKUUID\$() 0aeeca8e-eab5-4c2d-8510-6e55cadcdc7d ' ↑上の出力事例は、実際には呼び出し毎に値が変わります。		



## 3. 2. 28 MONTH

関数		
機 能	指定日（文字列）の月を返します。	
書 式	<(戻り値)月> = MONTH (<①年月日文字列> [, <②エラー送出フラグ> ])	
戻り値	戻り値	数値
	<月> 年月日文字列の月の値を返します。	
パラ メータ	①	文字列
	<年月日文字列> 月を求める日付(「年/月/日」の形式)を指定します。	
	②	真偽値
	<エラー送出フラグ> TRUEを指定すると、不正な年月日文字列を渡すと、エラーとします。 FALSEを指定すると、不正な年月日文字列の場合に、0を返します。 省略すると、FALSEとします。	
備 考	<年月日文字列>が日付でない場合、0を返します。 閏年ではない年に閏日を入力すると、翌月の初日に繰り越されます。	
使用例1	NUM = MONTH("2013/07/15")  変数NUMにMONTHの実行結果7を代入します。	

## 3. 2. 29 PASSWORD\_HASH\$

関数		
機 能	受け取ったパスワード文字列からハッシュ文字列を得ます。	
書 式	<(戻り値)ハッシュ文字列> = PASSWORD_HASH\$( <①パスワード文字列> )	
戻り値	戻り値	<ハッシュ文字列> 文字列
	パスワード文字列からハッシュ文字列を得ます。	
パラメータ	①	<パスワード文字列> 文字列
	パスワードとなる文字列を与えてください。	
使用例	? PASSWORD_HASH\$("hello")  パスワード文字列 "hello" のハッシュ文字列が表示されます。 "\$2b\$12\$WB4vvIwbx.oQevVP9Tb/300wfIBg8vGcknkonnsEp0LgHRUKD48zy" のような文字列が得られます。(内容は、実行毎に変化します)	

## 3. 2. 30 PASSWORD\_VERIFY

関数		
機 能	パスワード文字列と、PASSWORD_HASH\$で得たハッシュの内容が一致しているか確認します。	
書 式	<(戻り値)一致結果> = PASSWORD_VERIFY( <①パスワード文字列>, <②ハッシュ文字列> )	
戻り値	戻り値	<一致結果> 数値
	ハッシュ文字列とパスワード文字列の内容が一致すれば1を、不一致ならば0が返ります。	
パラメータ	①	<パスワード文字列> 文字列
	パスワードとなる文字列を与えてください。	
	②	<ハッシュ文字列> 文字列
	PASSWORD_HASH\$ で得たハッシュ文字列を与えてください。	
注 意	・パスワード文字列に、PASSWORD_HASH\$ の値を与えてはいけません。不正な動作を起こします。	
使用例	A\$ = PASSWORD_HASH\$("hello") LINE INPUT "パスワードを入力>"; B\$ ? "一致確認="; PASSWORD_VERIFY(B\$, A\$)  パスワード文字列 "hello" のハッシュ文字列をA\$に取得しておき、 LINE INPUTで、ユーザーにパスワード文字列を入力してもらいます。 その内容が一致しているか、確認して結果を表示します。	

## 3.2.31 RIGHT\$

関数			
機 能	文字列の右側から任意の長さの文字列を抜き出します。		
書 式	<(戻り値) 抜き出し後文字列> = RIGHT\$(<①文字列>, <②抜き出す文字数>)		
戻り値	戻り値	<抜き出し後文字列>	文字列
	文字列の右側から任意の長さの文字列を抜き出した文字列が得られます。		
パラ メータ	①	<文字列>	文字列
	抜き出し対象の文字列を指定します。		
	②	<抜き出す文字数>	数値
	抜き出す文字列の長さを0以上で指定します。 <文字列>の長さより大きい場合、<文字列>をそのまま返します。		
使用例1	SMP\$ = RIGHT\$("SAMPLE", 4)  変数SMP\$に"MPLE" ("SAMPLE"の右側4文字を取り出した文字列)を代入します。		
使用例2	SMP\$ = RIGHT\$("あいうえお", 4)  対象の文字列が全角文字でも使用できます。 変数SMP\$に"いうえお"を代入します。		
使用例3	SMP\$ = RIGHT\$("SAMPLE", 7)  <抜き出す文字数>が<文字列>の字数を上回っているため、"SAMPLE"をそのまま代入します。		

## 3.2.32 RIGHTB\$

関数			
機 能	文字列の右側から任意のバイト数分の文字列を抜き出します。		
書 式	<(戻り値) 抜き出し後文字列> = RIGHTB\$(<①文字列>, <②抜き出すバイト数>)		
戻り値	戻り値	<抜き出し後文字列>	文字列
	文字列の右側から任意のバイト数分の文字列を抜き出した文字列が得られます。		
パラ メータ	①	<文字列>	文字列
	抜き出し対象の文字列を指定します。		
	②	<抜き出すバイト数>	数値
	抜き出す文字列の長さを0以上のバイト数単位で指定します。 <文字列>の長さより大きい場合、<文字列>をそのまま返します。		
備 考	<ul style="list-style-type: none"> <li>この関数は、文字列のデータをバイトデータとして扱う為に用意されています。</li> <li>得られる値は文字列形式ですが、PRINT文などで表示できない場合があります。 (PRINT文で表示するには、UTF-8形式の文字列である必要があります)</li> </ul>		
使用例	SRC\$=CHRB\$(0)+CHRB\$(1)+CHRB\$(2)+CHRB\$(3) SMP\$=RIGHTB\$(SRC\$, 2)  SRC\$は、4バイトのバイトデータ(文字列形式)です。 RIGHTB\$により、後方2バイトのバイトデータを抜き出し、SMP\$には「CHRB\$(2)+CHRB\$(3)」のバイトデータが得られます。		

## 3. 2. 33 SECOND

関数		
機 能	指定時刻（文字列）の秒を返します。	
書 式	<(戻り値) 秒> = SECOND (<①時分秒文字列> [, <②エラー送出フラグ> ])	
戻り値	戻り値	数値
	<秒> 時分秒文字列の秒の値を返します。	
パラ メータ	①	文字列
	<時分秒文字列> 時を求める時刻(「時:分:秒」の形式)を指定します。	
	②	真偽値
	<エラー送出フラグ> TRUEを指定すると、不正な時分秒文字列を渡すと、エラーとします。 FALSEを指定すると、不正な時分秒文字列の場合に、0を返します。 省略すると、FALSEとします。	
備 考	<時分秒文字列>が時刻でない場合、0を返します。 時刻の書式は、24時間表記とします。 閏年ではない年に閏秒を入力すると、翌分の0秒に繰り越されます。	
使用例1	NUM = SECOND("14:03:15")  変数NUMにSECONDの実行結果15を代入します。	

## 3. 2. 34 STR\_BASE64\_DECODE\$

関数		
機 能	BASE64形式にエンコードされた文字列を、元の文字列にデコード(逆変換)します。	
書 式	<(戻り値) 逆変換後文字列> = STR_BASE64_DECODE\$(<①BASE64形式文字列>)	
戻り値	戻り値	<逆変換後文字列> 文字列
	BASE64形式にエンコードされた文字列を、元の文字列にデコード(逆変換)した文字列が得られます。	
パラメータ	①	<BASE64形式文字列> 文字列
	STR_BASE64_ENCODE\$でBASE64形式に変換した文字列を与えます。	
備 考	<ul style="list-style-type: none"> <li>BASE64とは、日本語を含んだ文字列やバイナリ形式の文字列を、6ビットずつ分割して 64種の印字可能な文字列に表現する、文字列表現方式を言います。</li> <li>.</li> </ul>	
使用例	' BASE64形式に文字列「インタフェース」を変換します S\$ = STR_BASE64_ENCODE\$("インタフェース") ' BASE64形式の文字列を、元の文字列に逆変換します M\$ = STR_BASE64_DECODE\$(S\$)	

## 3. 2. 35 STR\_BASE64\_ENCODE\$

関数		
機 能	文字列を、BASE64形式にエンコード(変換)します。	
書 式	<(戻り値) 変換後文字列> = STR_BASE64_ENCODE\$(<①文字列>)	
戻り値	戻り値	<変換後文字列> 文字列
	BASE64形式にエンコード(変換)した文字列が得られます。	
パラメータ	①	<文字列> 文字列
	変換元の文字列を与えます。	
備 考	<ul style="list-style-type: none"> <li>BASE64とは、日本語を含んだ文字列やバイナリ形式の文字列を、6ビットずつ分割して 64種の印字可能な文字列に表現する、文字列表現方式を言います。</li> <li>.</li> </ul>	
使用例	' BASE64形式に文字列「インタフェース」を変換します S\$ = STR_BASE64_ENCODE\$("インタフェース") ' BASE64形式の文字列を、元の文字列に逆変換します M\$ = STR_BASE64_DECODE\$(S\$)	

## 3.2.36 STR\_DUMP\$

関数			
機 能	単一値をダンプ文字列化		
書 式	<(戻り値) ダンプ文字列> = STR_DUMP\$(<①単一値> [, <②オプション> ] )		
戻り値	戻り値	<ダンプ文字列>	文字列
	単一値をダンプ文字列化した値が得られます。		
パラメータ	①	<単一値>	値
	文字列や整数、実数などの単一値を指定します。 値は、パソコン内部のメモリ表現がダンプ形式で得られます。 配列は、最初の添字の値が採用されます。		
	②	<オプション>	真偽値
	TRUE(デフォルト値)を指定すると、ヘッダ付きでダンプ出力されます。 FALSEを指定すると、ヘッダ無しの簡易なダンプ出力が得られます。		
使用例1	<pre> A\$ = "hello"+CHRB\$(0)+"AJAN" PRINT STR_DUMP\$(A\$) dump addr:0x557279735870~0x557279735879 size:10           +0 +1 +2 +3 +4 +5 +6 +7 - +8 +9 +A +B +C +D +E +F   0123456789ABCDEF -----+-----+-----+-----+-----+-----+-----+-----+ 0x0000557279735870 : 68 65 6c 6c 6f 00 41 4a - 41 4e           : hello.AJAN </pre> <p>’ 例えば、上記の様な出力結果が得られます。</p>		
使用例2	<pre> A\$ = "hello"+CHRB\$(0)+"AJAN" PRINT STR_DUMP\$(A\$, FALSE) 68 65 6c 6c 6f 00 41 4a 41 4e </pre> <p>’ スイッチにFALSEを指定すると、上のような簡易的なダンプ出力が得られます。</p>		
使用例3	<pre> PRINT STR_DUMP\$(123%) dump addr:0x7ffef5941c~0x7ffef5941f size:4           +0 +1 +2 +3 +4 +5 +6 +7 - +8 +9 +A +B +C +D +E +F   0123456789ABCDEF -----+-----+-----+-----+-----+-----+-----+ 0x00007ffef59410 :           -           7b 00 00 00 :           {... </pre> <p>’ 数値を指定すると、パソコン内部のメモリ表現が得られます。</p>		

## 3. 2. 37 STR\_AND\$

関数			
機 能	文字列同士に対して、論理積 (AND) を求めた結果を返します。		
書 式	<(戻り値) 結果文字列> = STR_AND\$(<①文字列1>, <①文字列2> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列の1バイト毎に、<文字列1> AND <文字列2> の演算を行った結果が得られます。		
パラ メータ	①	<文字列1>, <文字列2>	文字列
	対象となる文字列を指定します。		
使用例	PRINT STR_DUMP\$(STR_AND\$("abc", "aaa"))		
	文字列同士に AND 演算を行った結果文字列が得られます。		

## 3. 2. 38 STR\_OR\$

関数			
機 能	文字列同士に対して、論理和 (OR) を求めた結果を返します。		
書 式	<(戻り値) 結果文字列> = STR_OR\$(<①文字列1>, <①文字列2> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列の1バイト毎に、<文字列1> OR <文字列2> の演算を行った結果が得られます。		
パラ メータ	①	<文字列1>, <文字列2>	文字列
	対象となる文字列を指定します。		
使用例	PRINT STR_DUMP\$(STR_OR\$("abc", "aaa"))		
	文字列同士に OR 演算を行った結果文字列が得られます。		

## 3. 2. 39 STR\_XOR\$

関数			
機 能	文字列同士に対して、排他的論理和 (XOR) を求めた結果を返します。		
書 式	<(戻り値) 結果文字列> = STR_XOR\$(<①文字列1>, <①文字列2> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列の1バイト毎に、<文字列1> XOR <文字列2> の演算を行った結果が得られます。		
パラ メータ	①	<文字列1>, <文字列2>	文字列
	対象となる文字列を指定します。		
使用例	PRINT STR_DUMP\$(STR_XOR\$("abc", "aaa"))		
	文字列同士に XOR 演算を行った結果文字列が得られます。		

## 3. 2. 40 STR\_IMP\$

関数			
機 能	文字列同士に対して、包含 (IMP) を求めた結果を返します。		
書 式	<(戻り値) 結果文字列> = STR_IMP\$(<①文字列1>, <①文字列2> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列の1バイト毎に、<文字列1> IMP <文字列2> の演算を行った結果が得られます。		
パラ メータ	①	<文字列1>, <文字列2>	文字列
	対象となる文字列を指定します。		
使用例	PRINT STR_DUMP\$(STR_IMP\$("abc", "aaa"))		
	文字列同士に IMP 演算を行った結果文字列が得られます。		

## 3. 2. 41 STR\_EQV\$

関数			
機 能	文字列同士に対して、同値 (EQV) を求めた結果を返します。		
書 式	<(戻り値) 結果文字列> = STR_EQV\$(<①文字列1>, <①文字列2> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列の1バイト毎に、<文字列1> EQV <文字列2> の演算を行った結果が得られます。		
パラ メータ	①	<文字列1>, <文字列2>	文字列
	対象となる文字列を指定します。		
使用例	PRINT STR_DUMP\$(STR_EQV\$("abc", "aaa"))		
	文字列同士に EQV 演算を行った結果文字列が得られます。		

## 3. 2. 42 STR\_REPEAT\$

関数			
機 能	文字列を指定回数繰り返し連結した文字列を返します。		
書 式	<(戻り値) 結果文字列> = STR_REPEAT\$(<①文字列>, <②繰り返し回数> )		
戻り値	戻り値	<結果文字列>	文字列
	文字列に対して、指定回数繰り返し連結した結果が得られます。		
パラ メータ	①	<文字列>	文字列
	対象となる文字列を指定します。		
	②	<繰り返し回数>	数値
	文字列を繰り返す回数を指定します。 1以上の数値を指定してください。		
使用例	PRINT STR_REPEAT\$("abc", 3)		
	"abc"を3回繰り返すので、"abcabcabc"と画面に出力されます。		



### 3. 2. 43 STR\_HAN2ZEN\$

関数								
機 能	文字列の半角文字を全角文字に変換します。							
書 式	<(戻り値) 結果文字列> = STR_HAN2ZEN\$(<①文字列> [, <②除外文字>, <③変換スイッチ文字> ])							
戻り値	<div>戻り値</div> <div>&lt;結果文字列&gt;</div> <div>文字列</div> <p>文字列中の半角文字を全角文字に変換した結果が得られます。</p>							
パラメータ	<div>①</div> <div>&lt;文字列&gt;</div> <div>文字列</div> <p>変換対象となる文字列を指定します。</p>							
	<div>②</div> <div>&lt;除外文字&gt;</div> <div>文字列</div> <p>変換から除外したい文字を列挙します。 省略すると、除外指定はありません。</p>							
	<div>③</div> <div>&lt;変換スイッチ文字&gt;</div> <div>文字列</div> <p>変換対象としたい文字の種別を指定できます。 省略すると、全ての文字の種別を変換対象とします。</p>							
	<table border="1"> <thead> <tr> <th>スイッチ文字</th><th>効果</th></tr> </thead> <tbody> <tr> <td>"A"</td><td>英文字を変換対象とします。</td></tr> <tr> <td>"D"</td><td>数文字を変換対象とします。</td></tr> <tr> <td>"K"</td><td>カタカナを変換対象とします。</td></tr> </tbody> </table>	スイッチ文字	効果	"A"	英文字を変換対象とします。	"D"	数文字を変換対象とします。	"K"
スイッチ文字	効果							
"A"	英文字を変換対象とします。							
"D"	数文字を変換対象とします。							
"K"	カタカナを変換対象とします。							
備 考	<ul style="list-style-type: none"> <li>カタカナの変換は、例えば 半角の「ア」は文字コード(ユニコード)では CHR\$(&amp;HFF71)に相当し、全角に変換すると CHR\$(&amp;H30A2) になります。</li> </ul>							
使用例1	<pre>PRINT STR_HAN2ZEN\$("AJAN!! 2020 テスト") AJAN!! 2020 テスト</pre>							
使用例2	<pre>PRINT STR_HAN2ZEN\$("AJAN!! 2020 テスト", "AJN") AJAN!! 2020 テスト ↑除外文字を指定すると「AJAN」のみ半角のままになります。</pre>							
使用例3	<pre>PRINT STR_HAN2ZEN\$("AJAN!! 2020 テスト", "", "D") AJAN!! 2020 テスト ↑数文字変換のみ指定すると、数文字のみ全角になります。</pre>							

3. 2. 44 STR\_ZEN2HAN\$

関数		
機 能	文字列の全角文字を半角文字に変換します。	
書 式	<(戻り値) 結果文字列> = STR_ZEN2HAN\$(<①文字列> [, <②除外文字>, <③変換スイッチ文字> ])	
戻り値	戻り値	<結果文字列> 文字列中の全角文字を半角文字に変換した結果が得られます。
パラ メータ	①	<文字列> 変換対象となる文字列を指定します。
	②	<除外文字> 変換から除外したい文字を列挙します。 省略すると、除外指定はありません。
	③	<変換スイッチ文字> 変換対象としたい文字の種別を指定できます。 省略すると、全ての文字の種別を変換対象とします。
	スイッチ文字	効果
	"A"	英文字を変換対象とします。
	"D"	数文字を変換対象とします。
	"K"	カタカナを変換対象とします。
備 考	・ カタカナの変換は、例えば 全角の「ア」は文字コード(ユニコード)では CHR\$(&H30A2)に相当し、半角に変換すると CHR\$(&HFF71) になります。	
使用例1	PRINT STR_HAN2ZEN\$("A J A N ! !   2 0 2 0   テスト") AJAN!! 2020 テスト	
使用例2	PRINT STR_HAN2ZEN\$("A J A N ! !   2 0 2 0   テスト", "A J N") A J A N!! 2020 テスト ↑除外文字を指定すると「A J A N」のみ全角のままになります。	
使用例3	PRINT STR_HAN2ZEN\$("A J A N ! !   2 0 2 0   テスト", "", "D") A J A N ! !   2020   テスト ↑数文字変換のみ指定すると、数文字のみ半角になります。	

## 3. 2. 45 STR\_HIRA2KATA\$

関数			
機 能	文字列のひらがな文字をカタカナ文字に変換します。		
書 式	<(戻り値) 結果文字列> = STR_HIRA2KATA\$(<①文字列> [, <②除外文字> ])		
戻り値	戻り値	<結果文字列>	文字列
	文字列中のひらがな文字をカタカナ文字に変換した結果が得られます。		
パラ メータ	①	<文字列>	文字列
	変換対象となる文字列を指定します。		
	②	<除外文字>	文字列
	変換から除外したい文字を列挙します。 省略すると、除外指定はありません。		
備 考	・ カタカナへの変換は、全角のカタカナに変換されます。		
使用例1	PRINT STR_HIRA2KATA\$("いろはニホヘト") イロハニホヘト		
使用例2	PRINT STR_HIRA2KATA\$("いろはニホヘト", "いろ") いろハニホヘト ↑ 除外文字を指定すると「いろ」のみ、ひらがな のままになります。		

## 3. 2. 46 STR\_KATA2HIRA\$

関数			
機 能	文字列のカタカナ文字をひらがな文字に変換します。		
書 式	<(戻り値) 結果文字列> = STR_KATA2HIRA\$(<①文字列> [, <②除外文字> ])		
戻り値	戻り値	<結果文字列>	文字列
	文字列中のカタカナ文字をひらがな文字に変換した結果が得られます。		
パラ メータ	①	<文字列>	文字列
	変換対象となる文字列を指定します。		
	②	<除外文字>	文字列
	変換から除外したい文字を列挙します。 省略すると、除外指定はありません。		
備 考	・ カタカナの変換は、全角のカタカナのみが対象です。		
使用例1	PRINT STR_KATA2HIRA\$("いろはニホヘト") いろはにほへと		
使用例2	PRINT STR_KATA2HIRA\$("いろはニホヘト", "ニホ") いろはニホへと ↑ 除外文字を指定すると「ニホ」のみ、カタカナ のままになります。		

## 3.2.47 STR\_STARTSWITH

関数			
機 能	文字列が特定文字列で始まるか判断します。		
書 式	<(戻り値)一致判定> = STR_STARTSWITH(<①文字列>, <②特定文字列> [, <③オフセット> ] )		
戻り値	戻り値	<一致判断>	真偽値
	文字列が特定文字列で始まっていると TRUE。 そうでなければ FALSEが得られます。		
パラメータ	①	<文字列>	文字列
	調査対象となる文字列を指定します。		
	②	<特定文字列>	文字列
	調査する文字列を指定します。 複数調査する為に、文字列配列で指定できます。		
	③	<オフセット>	数値
	調査を開始する位置を指定できます。 省略すると、先頭から調査します。		
使用例	PRINT STR_STARTSWITH("あいうえお", "あいう") TRUE PRINT STR_STARTSWITH("あいうえお", "あいい") FALSE		

## 3.2.48 STR\_ENDSWITH

関数			
機 能	文字列が特定文字列で終わるか判断します。		
書 式	<(戻り値)一致判定> = STR_ENDSWITH(<①文字列>, <②特定文字列> )		
戻り値	戻り値	<一致判断>	真偽値
	文字列が特定文字列で終わっている場合 TRUE。 そうでなければ FALSEが得られます。		
パラメータ	①	<文字列>	文字列
	調査対象となる文字列を指定します。		
	②	<特定文字列>	文字列
	調査する文字列を指定します。 複数調査する為に、文字列配列で指定できます。		
使用例	PRINT STR_ENDSWITH("あいうえお", "うえお") TRUE PRINT STR_ENDSWITH("あいうえお", "ええお") FALSE		

## 3. 2. 49 STR\_REMOVEPREFIX\$

関数			
機 能	文字列の先頭部分が指定文字列と一致すれば、これを削除します。		
書 式	<(戻り値) 削除後文字列> = STR_REMOVEPREFIX\$(<①文字列>, <②削除文字列> )		
戻り値	戻り値	<削除後文字列>	文字列
	文字列の先頭部分が、削除文字列と一致すれば、文字列から削除文字列部分を削除して得られます。 一致しなければ、元の文字列が得られます。		
パラ メータ	①	<文字列>	文字列
	対象となる文字列を指定します。		
	②	<削除文字列>	文字列
	削除対象となる文字列を指定します。		
使用例	PRINT STR_REMOVEPREFIX\$("abcaabbccabc", "abc") aabbccabc		

## 3. 2. 50 STR\_REMOVESUFFIX\$

関数			
機 能	文字列の末尾部分が指定文字列と一致すれば、これを削除します。		
書 式	<(戻り値) 削除後文字列> = STR_REMOVESUFFIX\$(<①文字列>, <②削除文字列> )		
戻り値	戻り値	<削除後文字列>	文字列
	文字列の末尾部分が、削除文字列と一致すれば、文字列から削除文字列部分を削除して得られます。 一致しなければ、元の文字列が得られます。		
パラ メータ	①	<文字列>	文字列
	対象となる文字列を指定します。		
	②	<削除文字列>	文字列
	削除対象となる文字列を指定します。		
使用例	PRINT STR_REMOVESUFFIX\$("abcaabbccabc", "abc") abcaabbcc		

## 3.2.51 STR\_VALIDUTF8

関数			
機 能	文字列全てがUTF8文字列として妥当であるか判定します。		
書 式	<(戻り値)判定結果> = STR_VALIDUTF8( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てがUTF8文字列として妥当であれば TRUEを、そうでなければ FALSE を返します。		
パラメータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	この関数は、あくまでUTF8形式として正しいかを検証します。 他の文字列エンコーディングでも、UTF8形式としては妥当な場合があります。		
使用例1	PRINT STR_VALIDUTF8("ハローAJAN")  UTF8文字列として正しいので、TRUEが得られます。		
使用例2	PRINT STR_VALIDUTF8("不正"+CHR(80))  UTF8文字列(途中不正なコード値)として正しくないので、FALSEが得られます。		

## 3.2.52 STR\_ISALNUM

関数			
機 能	文字列全てが英数文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISALNUM( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが英数文字であれば TRUEを、そうでなければ FALSE を返します。		
パラメータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISALNUM("123AJAN")  全て英数文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISALNUM("あいいうAJAN")  全て英数文字(一部ひらがな)ではないので、FALSEが得られます。		

## 3. 2. 53 STR\_ISALPHA

関数			
機 能	文字列全てが英文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISALPHA( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが英文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISALPHA("AJANxyz")  全て英文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISALPHA("AJAN123")  全て英文字(一部数字)ではないので、FALSEが得られます。		

## 3. 2. 54 STR\_ISCNTRL

関数			
機 能	文字列全てが制御文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISCNTRL( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが制御文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISCNTRL(CHRB\$(9)+CHRB\$(10))  全て制御文字(タブコードと改行コード)なので、TRUEが得られます。		
使用例2	PRINT STR_ISCNTRL("AJAN")  全て制御文字(英字)ではないので、FALSEが得られます。		

## 3. 2. 55 STR\_ISDIGIT

関数			
機 能	文字列全てが数文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISDIGIT( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが数文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISDIGIT("12345")  全て数文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISDIGIT("123AJAN")  全て数文字(一部英字)ではないので、FALSEが得られます。		

## 3. 2. 56 STR\_ISGRAPH

関数			
機 能	文字列全てが表示可能な文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISGRAPH( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが表示可能な文字であれば TRUEを、 そうでなければ FALSE を返します。(空白、制御文字、書式文字が該当)		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISGRAPH("123AJAN")  全て表示可能な文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISGRAPH("123"+CHR(10))  全て表示可能な文字(一部制御文字)ではないので、FALSEが得られます。		



## 3. 2. 57 STR\_ISLOWER

関数			
機 能	文字列全てが小文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISLOWER( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが小文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISLOWER("helloajan")  全て小文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISLOWER("helloAJAN")  全て小文字(一部大文字)ではないので、FALSEが得られます。		

## 3. 2. 58 STR\_ISPRINT

関数			
機 能	文字列全てが表示可能な文字(空白含む)であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISPRINT( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが表示可能な文字であれば TRUEを、 (STR_ISGRAPHに対して空白も TRUE 判定に含みます) そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISPRINT("123 AJAN")  全て表示可能な文字(空白含む)なので、TRUEが得られます。		
使用例2	PRINT STR_ISPRINT("123 AJAN"+CHRB\$(9))  全て表示可能な文字(一部制御文字)ではないので、FALSEが得られます。		

## 3. 2. 59 STR\_ISPUNCT

関数			
機 能	文字列全てが表示可能な文字(空白と英数字を含まない)であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISPUNCT( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが表示可能な文字(空白と英数字は含みません)であれば TRUEを、そうでなければ FALSE を返します。		
パラメータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISPUNCT("%&!#")		
	全て表示可能(空白と英数字を含まない)な文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISPUNCT("%&!#123")		
	全て表示可能な文字(一部数字)ではないので、FALSEが得られます。		

## 3. 2. 60 STR\_ISSPACE

関数			
機 能	文字列全てが空白、タブ、行区切りであるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISSPACE( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが空白、タブ、行区切り(改行など)であれば TRUEを、そうでなければ FALSE を返します。		
パラメータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISSPACE(" "+CHRB\$(9)+CHRB\$(10)+CHRB\$(13))		
	全て空白、タブ、行区切り文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISSPACE("123 "+CHRB\$(9)+CHRB\$(10)+CHRB\$(13))		
	全て空白、タブ、行区切り文字(一部数字)ではないので、FALSEが得られます。		

## 3. 2. 61 STR\_ISUPPER

関数			
機 能	文字列全てが大文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISUPPER( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが大文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISUPPER("HELLOAJAN")  全て大文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISUPPER("helloAJAN")  全て大文字(一部小文字)ではないので、FALSEが得られます。		

## 3. 2. 62 STR\_ISXDIGIT

関数			
機 能	文字列全てが16進数文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISXDIGIT( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てが16進数文字であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISXDIGIT("123ABCDEF")  全て16進数文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISXDIGIT("123ABCDEFXYZ")  全て16進数文字(16進数以外の英字)ではないので、FALSEが得られます。		

## 3. 2. 63 STR\_ISASCII

関数			
機 能	文字列全てがASCII文字であるか判定します。		
書 式	<(戻り値)判定結果> = STR_ISASCII ( <①文字列> )		
戻り値	戻り値	<判定結果>	真偽値
	文字列全てがASCII文字 (&H00～&H7Fまで)であれば TRUEを、 そうでなければ FALSE を返します。		
パラ メータ	①	<文字列>	文字列
	判定対象となる文字列を指定します。		
備 考	UTF8文字列で構成されていれば文字単位、そうでなければバイト単位で判定します。		
使用例1	PRINT STR_ISASCII ("123ABCxzy")  全てASCII文字なので、TRUEが得られます。		
使用例2	PRINT STR_ISASCII ("123ABCxzy歩")  全てASCII文字(一部漢字)ではないので、FALSEが得られます。		

## 3.2.64 CHRTYPE

関数			
機 能	文字列の各文字の情報を得ます。		
書 式	<(戻り値) 情報配列> = CHRTYPE( <①文字列>, <②情報ID> )		
戻り値	戻り値	<情報配列>	配列
	情報IDで指定した、1文字毎の情報を1次元の数値配列で得られます。		
パラメータ	①	<文字列>	文字列
	情報を得る文字列を指定します。 UTF8として読み取り可能な文字列である必要があります。		
	②	<情報ID>	文字列
	文字列に対して、1文字ずつ得たい情報の種別を指定します。		
	設定値	内容	
	"bytes"	1文字のバイト数を得ます。	
	"isalnum"	英数文字なら1、そうでないなら0を得ます。	
	"isalpha"	英文字なら1、そうでないなら0を得ます。	
	"isctrl"	制御文字なら1、そうでないなら0を得ます。	
	"isdigit"	数文字なら1、そうでないなら0を得ます。	
"isgraph"	非空白出力文字なら1、そうでないなら0を得ます。 (空白、制御文字、書式文字は0です)		
"isprint"	表示可能文字(空白含む)なら1、そうでないなら0を得ます。		
"isspace"	空白、タブ、行区切りなら1、そうでないなら0を得ます。		
"islower"	小文字なら1、そうでないなら0を得ます。		
"isupper"	大文字なら1、そうでないなら0を得ます。		
"isxdigit"	16進数文字なら1、そうでないなら0を得ます。		
"iswide"	ワイド文字なら1、そうでないなら0を得ます。		
"iswide_cjk"	東アジア方面のロケールに従ってワイド文字なら1、そうでないなら0を得ます。		
"iszerowidth"	ゼロ幅文字なら1、そうでないなら0を得ます。		
"type"	Unicode仕様の文字種を得ます。 「<情報ID : "type" の値>」を参照ください。		
"break_type"	Unicode仕様の区切り文字種を得ます。 「<情報ID : "break_type" の値>」を参照ください。		
"todigit"	数文字を数値に変換します。そうでないものは - 1 が得られます。		
"toxdtit"	16進数文字を数値に変換します。そうでないものは - 1 が得られます。		
備 考	・ 文字列がUTF8として読み取り可能かどうかは、STR_VALIDUTF8 で確認できます。		
使用例	? chrtype("123abcDEFZ# あいう", "bytes") ' [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3 ] が得られます ? chrtype("123abcDEFZ# あいう", "isalnum") ' [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 ] が得られます。 ? chrtype("123abcDEFZ# あいう", "isdigit") ' [ 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] が得られます ? chrtype("123abcDEFZ# あいう", "isxdigit") ' [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0 ] が得られます ? chrtype("123abcDEFZ# あいう", "islower") ' [ 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0 ] が得られます		

```
? chrtype("123abcDEFZ# あいう", "isupper")
' [ 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0 ] が得られます
? chrtype("123abcDEFZ# あいう", "iswide_cjk")
' [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1 ] が得られます
? chrtype("123abcDEFZ# あいう", "todigit")
' [ 1, 2, 3, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1 ] が得られます
? chrtype("123abcDEFZ# あいう", "toxdigit")
' [ 1, 2, 3, 10, 11, 12, 13, 14, 15, -1, -1, -1, -1, -1, -1 ] が得られます
```

### 3.2.64.1 <情報ID: "type" の値>

Unicode仕様の文字種の一覧値です。

文字種の各名称と意味は、以下のURLの資料を参考ください。

- Unicode Character Database

[https://www.unicode.org/reports/tr44/#General\\_Category\\_Values](https://www.unicode.org/reports/tr44/#General_Category_Values)

値	説明(文字種)
0	General category "Other Control" (Cc)
1	General category "Other Format" (Cf)
2	General category "Other Not Assigned" (Cn)
3	General category "Other Private Use" (Co)
4	General category "Other Surrogate" (Cs)
5	General category "Letter Lowercase" (Ll)
6	General category "Letter Modifier" (Lm)
7	General category "Letter Other" (Lo)
8	General category "Letter Titlecase" (Lt)
9	General category "Letter Uppercase" (Lu)
10	General category "Mark Spacing" (Mc)
11	General category "Mark Enclosing" (Me)
12	General category "Mark Nonspacing" (Mn)
13	General category "Number Decimal Digit" (Nd)
14	General category "Number Letter" (Nl)
15	General category "Number Other" (No)
16	General category "Punctuation Connector" (Pc)
17	General category "Punctuation Dash" (Pd)
18	General category "Punctuation Close" (Pe)
19	General category "Punctuation Final quote" (Pf)
20	General category "Punctuation Initial quote" (Pi)
21	General category "Punctuation Other" (Po)
22	General category "Punctuation Open" (Ps)
23	General category "Symbol Currency" (Sc)
24	General category "Symbol Modifier" (Sk)
25	General category "Symbol Math" (Sm)
26	General category "Symbol Other" (So)
27	General category "Separator Line" (Zl)
28	General category "Separator Paragraph" (Zp)
29	General category "Separator Space" (Zs)

### 3.2.64.2 <情報ID: "break\_type" の値>

Unicode仕様の区切り文字種の一覧値です。

区切り文字種の各名称と意味は、以下のURLの資料を参考ください。

- Unicode Standard Annex #14 – Unicode line breaking algorithm

<https://www.unicode.org/unicode/reports/tr14/>

値	説明(区切り文字種)
0	Mandatory Break (BK)
1	Carriage Return (CR)
2	Line Feed (LF)
3	Attached Characters and Combining Marks (CM)
4	Surrogates (SG)
5	Zero Width Space (ZW)
6	Inseparable (IN)
7	Non-breaking ("Glue") (GL)
8	Contingent Break Opportunity (CB)
9	Space (SP)
10	Break Opportunity After (BA)
11	Break Opportunity Before (BB)
12	Break Opportunity Before and After (B2)
13	Hyphen (HY)
14	Nonstarter (NS)
15	Opening Punctuation (OP)
16	Closing Punctuation (CL)
17	Ambiguous Quotation (QU)
18	Exclamation/Interrogation (EX)
19	Ideographic (ID)
20	Numeric (NU)
21	Infix Separator (Numeric) (IS)
22	Symbols Allowing Break After (SY)
23	Ordinary Alphabetic and Symbol Characters (AL)
24	Prefix (Numeric) (PR)
25	Postfix (Numeric) (PO)
26	Complex Content Dependent (South East Asian) (SA)
27	Ambiguous (Alphabetic or Ideographic) (AI)
28	Unknown (XX)
29	Next Line (NL)
30	Word Joiner (WJ)
31	Hangul L Jamo (JL)
32	Hangul V Jamo (JV)
33	Hangul T Jamo (JT)
34	Hangul LV Syllable (H2)
35	Hangul LVT Syllable (H3)
36	Closing Parenthesis (CP). Since 2.28
37	Conditional Japanese Starter (CJ). Since: 2.32
38	Hebrew Letter (HL). Since: 2.32
39	Regional Indicator (RI). Since: 2.36
40	Emoji Base (EB). Since: 2.50
41	Emoji Modifier (EM). Since: 2.50
42	Zero Width Joiner (ZWJ). Since: 2.50

## 3. 2. 65 STR2ARRAY

関数			
機 能	文字列を指定したバイト単位の数値配列に変換します。		
書 式	<(戻り値)数値配列> = STR2ARRAY ( <①文字列> [, <②分割バイト単位> ] )		
戻り値	戻り値	<数値配列>	配列
	文字列から指定したバイト単位で数値配列に変換します。		
パラ メータ	①	<文字列>	文字列
	操作対象の文字列を指定します。		
	②	<分割バイト単位>	数値
	文字列から数値に変換するバイト単位数を指定します。 1 / 2 / 4 / 8 のどれかを選択します。省略すると、1 が採用されます。		
備 考	・ 数値配列を元の文字列に変換するには、「ARRAY2STR\$」を使用します。		
使用例1	PRINT STR2ARRAY("123") ' "[ 49, 50, 51 ]" が出力されます。		
使用例2	' ? STR2ARRAY("ABC", 1) を指定した際の動作と同じ処理を記述した例です。 S\$ = "ABC" LIST ARY REDIM ARY(LENB(S\$)-1) FOR I=0 TO LENB(S\$)-1 ARY(I) = ASCB(MIDB\$(S\$, 1+I, 1)) NEXT I PRINT ARY		



## 3. 2. 66 STRDELB\$

関数		
機 能	文字列の任意の位置から指定したバイト長の文字列を削除します。	
書 式	<(戻り値)削除後文字列> = STRDELB\$(<①文字列>, <②開始位置> [, <③削除するバイト数>])	
戻り値	戻り値	<削除後文字列> 文字列 文字列中、開始位置から指定したバイト長の文字列を削除した結果が得られます。
パラ メータ	①	<文字列> 文字列 操作対象の文字列を指定します。
	②	<開始位置> 数値 削除する開始位置をバイト単位(1はじまり)で指定します。 <文字列>の格納バイト数より大きいと格納バイト長と同じになります。
	③	<削除するバイト数> 数値 削除するバイト数を0以上で指定します。 省略した場合や、<開始位置>以降のバイト数より大きい値を指定した場合は、 <開始位置>以降の文字列をすべて削除します。
備 考	<ul style="list-style-type: none"> <li>この関数は、文字列のデータをバイトデータとして扱う為に用意されています。</li> <li>得られる値は文字列形式ですが、PRINT文などで表示できない場合があります。 (PRINT文で表示するには、UTF-8形式の文字列である必要があります)</li> </ul>	
使用例	SRC\$=CHRB\$(0)+CHRB\$(1)+CHRB\$(2)+CHRB\$(3) SMP\$ = STRDELB\$(SRC\$, 2, 2)  SRC\$は、4バイトのバイトデータ(文字列形式)です。 STRDELB\$により、先頭2バイト目から2バイトのバイトデータを削除します。 変数SMP\$には、「CHRB\$(0)+CHRB\$(3)」のバイトデータが得られます。	

## 3. 2. 67 STRINSB\$

関数			
機 能	文字列の任意の位置に指定したバイトデータ文字列を挿入します。		
書 式	<(戻り値)挿入後文字列> = STRINSB\$(<①文字列>, <②開始位置>, <③挿入する文字列>)		
戻り値	戻り値	<挿入後文字列>	文字列
	文字列中、開始位置にバイトデータ文字列を挿入した結果が得られます。		
パラ メータ	①	<文字列>	文字列
	操作対象の文字列を指定します。		
	②	<開始位置>	数値
	挿入する開始位置をバイト単位(1はじまり)で指定します。 <文字列>の格納バイト数より大きいと格納バイト長と同じになります。		
	③	<挿入する文字列>	文字列
	挿入するバイトデータ文字列を指定します。		
備 考	<ul style="list-style-type: none"> <li>この関数は、文字列のデータをバイトデータとして扱う為に用意されています。</li> <li>得られる値は文字列形式ですが、PRINT文などで表示できない場合があります。 (PRINT文で表示するには、UTF-8形式の文字列である必要があります)</li> </ul>		
使用例	SRC\$=CHRB\$(0)+CHRB\$(1)+CHRB\$(2)+CHRB\$(3) SMP\$ = STRINSB\$(SRC\$, 2, CHRB\$(10)+CHRB\$(11))  SRC\$は、4バイトのバイトデータ(文字列形式)です。 STRDELB\$により、先頭2バイト目から2バイトのバイトデータを挿入します。 変数SMP\$には、 「CHRB\$(0)+CHRB\$(10)+CHRB\$(11)+CHRB\$(1)+CHRB\$(2)+CHRB\$(3)」のバイトデータが得られます。		

## 3. 2. 68 STRREVERSE\$

関数			
機 能	文字列の並びを逆にします。		
書 式	<(戻り値) 反転文字列> = STRREVERSE\$( <①文字列> )		
戻り値	戻り値	<反転文字列>	文字列
	指定した文字列の並びを逆にした文字列を得ます。		
パラ メータ	①	<文字列>	文字列
	操作対象の文字列を指定します。		
備 考	・ 指定する文字列は UTF-8 形式の文字列を渡してください。		
使用例1	PRINT STRREVERSE\$("123") ' "321" が出力されます。		
使用例2	PRINT STRREVERSE\$("あいう") ' "ういあ" が出力されます。		

## 3. 2. 69 STRREVERSEB\$

関数			
機 能	バイナリ文字列の並びを逆にします。		
書 式	<(戻り値) 反転文字列> = STRREVERSEB\$( <①文字列> )		
戻り値	戻り値	<反転文字列>	文字列
	指定した文字列の並びをバイト単位で逆にした文字列を得ます。		
パラ メータ	①	<文字列>	文字列
	操作対象のバイナリ文字列を指定します。		
備 考	<ul style="list-style-type: none"> <li>この関数は、文字列のデータをバイトデータとして扱うために用意されています。</li> <li>得られる値は文字列形式ですが、PRINT文などで表示できない場合があります。</li> </ul>		
使用例	S\$ = STRREVERSEB\$(CHR\$(0)+CHR\$(1)+CHR\$(2)) ' S\$ には、CHR\$(2)+CHR\$(1)+CHR\$(0) 相当のバイナリ文字列が得られます。		

## 3. 2. 70 TRUNC

関数			
機 能	指定数値を指定位置で切り捨てます。		
書 式	<(戻り値) 切捨て後数値> = TRUNC(<①数値>[, <②指定位置>])		
戻り値	戻り値	<切捨て後数値>	数値
	数値を指定位置で切り捨てた値が得られます。		
パラ メータ	①	<数値>	数値
	切り捨てる対象の数値を指定します。		
	②	<指定位置>	数値
	正数時、小数点以下の有効桁数を指定します。 負数時、整数部の切捨て位置を指定します。 省略すると小数点以下を切り捨てます。		
備 考	正の値を指定すると小数点以下を、負の値だと小数点以上を切り捨てます。		
使用例1	NUM = TRUNC(123.456)		
	変数NUMに123（123.456の小数部を切り捨てた数値）を代入します。		
使用例2	NUM = TRUNC(123.456, 1)		
	変数NUMに123.4（123.456の小数第2位以下を切り捨てた数値）を代入します。		
使用例3	NUM = TRUNC(123.456, -1)		
	変数NUMに120（123.456の整数部の1桁目以下を切り捨てた数値）を代入します。		

## 3. 2. 71 UCASE\$

関数		
機 能	文字列中の小文字（半角英字）を大文字に変換します。	
書 式	<(戻り値) 大文字文字列> = UCASE\$ (<①文字列>)	
戻り値	戻り値	<大文字文字列>
	文字列中の小文字（半角英字）を大文字に変換した結果が得られます。	
パラ メータ	①	<文字列>
	変換対象の文字列を指定します。	
備 考	大文字を小文字に変換するには「LCASE\$」を使用します。	
使用例1	SMP\$ = UCASE\$("sample")	
	変数SMP\$に"SAMPLE"を代入します。	
使用例2	SMP\$ = UCASE\$("samPLE")	
	大文字が混在する場合も使用できます。 変数SMP\$に"SAMPLE"を代入します。	

## 3. 2. 72 UUIDCOMP

関数		
機 能	UUID文字列同士が、同じか比較します。	
書 式	<(戻り値)比較結果> = UUIDCOMP ( <①UUID文字列1>, <①UUID文字列2> )	
戻り値	戻り値	<比較結果> 数値
	UUID文字列同士を比較した結果を返します。 相互が同じであれば0、そうでなければ非0を返します。	
パラ メータ	①	<UUID文字列1>, <UUID文字列2> 文字列
	比較対象となるUUID文字列です。 MKUUID\$ の戻り値のような形式の文字列を与えてください。	
使用例	A\$ = MKUUID\$() B\$ = MKUUID\$() ? UUIDCOMP (A\$, B\$)    ' 異なるUUIDを比較するので、非0が返ります ? UUIDCOMP (A\$, A\$)    ' 同じUUIDを比較するので、0が返ります。	

## 3. 2. 73 YEAR

関数		
機 能	指定日（文字列）の年を返します。	
書 式	<(戻り値)年> = YEAR ( <①年月日文字列> [, <②エラー送出フラグ> ] )	
戻り値	戻り値	<年> 数値
	年月日文字列の年の値を返します。	
パラ メータ	①	<年月日文字列> 文字列
	年を求める日付(「年/月/日」の形式)を指定します。	
	②	<エラー送出フラグ> 真偽値
備 考	TRUEを指定すると、不正な年月日文字列を渡すと、エラーとします。	
	FALSEを指定すると、不正な年月日文字列の場合に、0を返します。 省略すると、FALSEとします。	
使用例1	<年月日文字列>が日付でない場合、0を返します。 閏年ではない年に閏日を入力すると、翌月の初日に繰り越されます。	
	NUM = YEAR("2013/07/15")  変数NUMに2013を代入します。	

### 3.3 配列演算に関する関数・命令

ここでは、配列に対して評価・演算するためのコマンドおよび関数群を紹介します。

#### 3.3.1 DIMAVG

関数			
機 能	配列の平均を返します。		
書 式	<(戻り値) 平均値> = DIMAVG(<①配列変数>)		
戻り値	戻り値	<平均値>	数値
	配列の平均値が得られます。		
パラメータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> DIM A(100) FOR I = 0 TO 100   A(I) = I NEXT I  ? DIMAVG(A) </pre> 配列Aの平均値を返します。		

#### 3.3.2 DIMMEDIAN

関数			
機 能	配列の中央値を返します。		
書 式	<(戻り値) 中央値> = DIMMEDIAN(<①配列変数>)		
戻り値	戻り値	<中央値>	数値
	配列内のデータの値を小さな順に並べた時、中央に位置する値が得られます。 要素数が奇数の場合、中央の値が得られます。 要素数が偶数の場合、中央に近い2つの値の算術平均が得られます。		
パラメータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> LIST A A = [ 5; 7; 3; 8 ] ? DIMMEDIAN(A)    ' 6 が得られます </pre>		

## 3.3.3 DIMMAX

関数			
機 能	配列の最大値を返します。		
書 式	<(戻り値)最大値> = DIMMAX(<①配列変数>)		
戻り値	戻り値	<最大値>	数値
	配列変数の内、最大値が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> DIM A(10) FOR I = 0 TO 10   A(I) = I - 5 NEXT I  ? A [ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 ] ? DIMMAX(A) 5 </pre> 配列Aの最大値を返します。		

## 3.3.4 DIMMIN

関数			
機 能	配列の最小値を返します。		
書 式	<(戻り値)最小値> = DIMMIN(<①配列変数>)		
戻り値	戻り値	<最小値>	数値
	配列変数の内、最小値が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> DIM A(10) FOR I = 0 TO 10   A(I) = I - 5 NEXT I  ? A [ -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 ] ? DIMMIN(A) -5 </pre> 配列Aの最小値を返します。		

## 3.3.5 DIMSUM

関数			
機 能	配列の合計を返します。		
書 式	<(戻り値) 合計値> = DIMSUM(<①配列変数>)		
戻り値	戻り値	<合計値>	数値
	配列変数の全ての要素をあわせた、合計値が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> DIM A(100) FOR I = 0 TO 100   A(I) = I NEXT I  ? DIMSUM (A) </pre> 配列Aの合計値を返します。		

## 3.3.6 DIMSTDEVP

関数			
機 能	配列の標準偏差値を返します。		
書 式	<(戻り値) 標準偏差値> = DIMSTDEVP(<①配列変数>)		
戻り値	戻り値	<標準偏差値>	数値
	配列変数に対して、標準偏差を求めた結果が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	<pre> LIST ARY ARY = [ 1.2; 2.1; 3.1; 4.1; 5.1 ] PRINT DIMSTDEVP(ARY) ' 1.38621787609308... が表示 </pre>		



## 3.3.7 DIMVARP

関数			
機 能	配列の分散値を返します。		
書 式	<(戻り値) 分散値> = DIMVARP(<①配列変数>)		
戻り値	戻り値	<分散値>	数値
	配列変数に対して、分散(あるいは標本分散)を求めた結果が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	LIST ARY ARY = [ 1.2; 2.1; 3.1; 4.1; 5.1 ] PRINT DIMVARP(ARY)    1.9216が表示		

## 3.3.8 DIMVARS

関数			
機 能	配列の不偏分散値を返します。		
書 式	<(戻り値) 不偏分散値> = DIMVARS(<①配列変数>)		
戻り値	戻り値	<不偏分散値>	数値
	配列変数に対して、不偏分散を求めた結果が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
備 考	与える配列変数は、要素数2以上を与えてください。		
使用例	LIST ARY ARY = [ 1.2; 2.1; 3.1; 4.1; 5.1 ] PRINT DIMVARS(ARY)    2.402が表示		

## 3.3.9 DIMADD

関数			
機 能	2つの配列の各要素に対して、加算した結果を返します。(書式1) または、1つの配列の各要素を、全て加算した結果を返します。(書式2)		
書 式 1	<(戻り値)加算配列値> = DIMADD(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<加算配列値>	配列
	配列変数の各要素に対して、<配列変数1> + <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値)加算結果値> = DIMADD(<②配列変数>)		
戻り値	戻り値	<加算結果値>	数値
	配列変数の各要素に対して、加算の演算を行ったトータル結果が得られます。		
パラ メータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ -2; -1; 0; 1; 2 ] B% = [ 2; 2; 2; 2; 2 ] PRINT DIMADD(A%, B%) [ 0, 1, 2, 3, 4 ]  配列 A%と配列 B% 同士を加算します。		
使用例2	DIM A%(4) A% = [ 1; 2; 3; 4; 5 ] PRINT DIMADD(A%) 15  配列A%の各要素を加算します。		

## 3.3.10 DIMADD\$

関数			
機 能	2つの文字列配列の各要素に対して、連結した結果を返します。		
書 式	<(戻り値)連結配列値> = DIMADD\$(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<連結配列値>	配列
	配列変数の各要素に対して、文字列同士の <配列変数1> + <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 文字列型である必要があります。		
使用例	DIM A\$(4), B\$(4) A\$ = [ "a"; "b"; "c"; "d"; "e" ] B\$ = [ "c"; "c"; "c"; "c"; "c" ] PRINT DIMADD\$(A\$, B\$) [ ac, bc, cc, dc, ec ]  配列 A\$と配列 B\$ 同士を連結します。		

3.3.11 DIMSUB

関数			
機 能	2つの配列の各要素に対して、減算した結果を返します。		
書 式	<(戻り値)減算配列値> = DIMSUB(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<減算配列値>	配列
	配列変数の各要素に対して、<配列変数1> - <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ -2; -1; 0; 1; 2 ] B% = [ 2; 2; 2; 2; 2 ] PRINT DIMSUB(A%, B%) [ -4, -3, -2, -1, 0 ]  配列 A%と配列 B% 同士を減算します。		

## 3.3.12 DIMDIV

関数			
機 能	2つの配列の各要素に対して、割り算した結果を返します。		
書 式	<(戻り値) 割り算配列値> = DIMDIV(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<割り算配列値>	配列
	配列変数の各要素に対して、<配列変数1> ÷ <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ -2; -1; 0; 1; 2 ] B% = [ 2; 2; 2; 2; 2 ] PRINT DIMDIV(A%, B%) [ -1, -0.5, 0, 0.5, 1 ]  配列 A%と配列 B% 同士を割り算します。		

## 3.3.13 DIMMOD

関数			
機 能	2つの配列の各要素に対して、剰余算した結果を返します。		
書 式	<(戻り値) 剰余算配列値> = DIMMOD(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<剰余算配列値>	配列
	配列変数の各要素に対して、<配列変数1> MOD <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ -2; -1; 0; 1; 2 ] B% = [ 2; 2; 2; 2; 2 ] PRINT DIMMOD(A%, B%) [ 0, 0, 0, 0, 1 ]  配列 A%と配列 B% 同士を剰余算します。		

3.3.14 DIMMUL

関数			
機 能	2つの配列の各要素に対して、掛け算した結果を返します。		
書 式	<(戻り値) 掛け算配列値> = DIMMUL(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<掛け算配列値>	配列
	配列変数の各要素に対して、<配列変数1> × <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ -2; -1; 0; 1; 2 ] B% = [ 2; 2; 2; 2; 2 ] PRINT DIMMUL(A%, B%) [ -4, -2, 0, 2, 4 ]  配列 A%と配列 B% 同士を掛け算します。		

## 3.3.15 DIMNOT

関数			
機 能	配列の各要素に対して、否定 (NOT) を求めた結果を返します。		
書 式	<(戻り値) NOT演算配列値> = DIMNOT (<①配列変数>)		
戻り値	戻り値	<NOT演算配列値>	配列
	配列変数の各要素に対して、NOT演算を行った結果が得られます。		
パラ メータ	①	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例	DIM A%(4) A% = [ 1; 2; 3; 4; 5 ] PRINT DIMNOT (A%) [ -2, -3, -4, -5, -6 ]  配列 A%に対して、NOT演算します。		



## 3.3.16 DIMAND

関数			
機 能	2つの配列の各要素に対して、論理積(AND)を求めた結果を返します。(書式1) または、1つの配列の各要素を、全てANDを求めた結果を返します。(書式2)		
書 式 1	<(戻り値)AND演算配列値> = DIMAND(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<AND演算配列値>	配列
	配列変数の各要素に対して、<配列変数1> AND <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値)AND演算結果値> = DIMAND(<②配列変数>)		
戻り値	戻り値	<AND演算結果値>	数値
	配列変数の各要素に対して、AND の演算を行ったトータル結果が得られます。		
パラ メータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 5; 5; 5; 5; 5 ] PRINT DIMAND(A%, B%) [ 1, 0, 1, 4, 5 ]  配列 A%と配列 B% 同士をANDします。		
使用例2	DIM A(2) A = [ 7; 3; 2 ] PRINT DIMAND(A) 2  配列Aの各要素をANDします。		

## 3.3.17 DIMOR

関数			
機 能	2つの配列の各要素に対して、論理和 (OR) を求めた結果を返します。(書式1) または、1つの配列の各要素を、全てORを求めた結果を返します。(書式2)		
書 式 1	<(戻り値)OR演算配列値> = DIMOR(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<OR演算配列値>	配列
	配列変数の各要素に対して、<配列変数1> OR <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値)OR演算結果値> = DIMOR(<②配列変数>)		
戻り値	戻り値	<OR演算結果値>	数値
	配列変数の各要素に対して、OR の演算を行ったトータル結果が得られます。		
パラ メータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 5; 5; 5; 5; 5 ] PRINT DIMOR(A%, B%) [ 5, 7, 7, 5, 5 ]  配列 A%と配列 B% 同士をORします。		
使用例2	DIM A(2) A = [ 7; 3; 2 ] PRINT DIMOR(A) 7  配列Aの各要素をORします。		

## 3.3.18 DIMXOR

関数			
機 能	2つの配列の各要素に対して、排他的論理和(XOR)を求めた結果を返します。(書式1) または、1つの配列の各要素を、全てXORを求めた結果を返します。(書式2)		
書 式 1	<(戻り値) XOR演算配列値> = DIMXOR(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<XOR演算配列値>	配列
	配列変数の各要素に対して、<配列変数1> XOR <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値) XOR演算結果値> = DIMXOR(<②配列変数>)		
戻り値	戻り値	<XOR演算結果値>	数値
	配列変数の各要素に対して、XOR の演算を行ったトータル結果が得られます。		
パラ メータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 5; 5; 5; 5; 5 ] PRINT DIMXOR(A%, B%) [ 4, 7, 6, 1, 0 ]  配列 A%と配列 B% 同士をXORします。		
使用例2	DIM A(2) A = [ 7; 3; 2 ] PRINT DIMXOR(A) 6  配列Aの各要素をXORします。		

## 3.3.19 DIMIMP

関数			
機 能	2つの配列の各要素に対して、包含 (IMP) を求めた結果を返します。(書式1) または、1つの配列の各要素を、全てIMPを求めた結果を返します。(書式2)		
書 式 1	<(戻り値) IMP演算配列値> = DIMIMP(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<IMP演算配列値>	配列
	配列変数の各要素に対して、<配列変数1> IMP <配列変数2> の演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値) IMP演算結果値> = DIMIMP(<②配列変数>)		
戻り値	戻り値	<IMP演算結果値>	数値
	配列変数の各要素に対して、IMP の演算を行ったトータル結果が得られます。		
パラ メータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 5; 5; 5; 5; 5 ] PRINT DIMIMP(A%, B%) [ -1, -3, -3, -1, -1 ]  配列 A%と配列 B% 同士をIMPします。		
使用例2	DIM A(2) A = [ 7; 3; 2 ] PRINT DIMIMP(A) 6  配列Aの各要素をIMPします。		

## 3.3.20 DIMEQV

関数			
機 能	2つの配列の各要素に対して、同値(EQV)を求めた結果を返します。(書式1) または、1つの配列の各要素を、全てEQVを求めた結果を返します。(書式2)		
書 式 1	<(戻り値)EQV演算配列値> = DIMEQV(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<EQV演算配列値>	配列
	配列変数の各要素に対して、<配列変数1> EQV <配列変数2> の演算を行った結果が得られます。		
パラメータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
書 式 2	<(戻り値)EQV演算結果値> = DIMEQV(<②配列変数>)		
戻り値	戻り値	<EQV演算結果値>	数値
	配列変数の各要素に対して、EQV の演算を行ったトータル結果が得られます。		
パラメータ	②	<配列変数>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
使用例1	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 5; 5; 5; 5; 5 ] PRINT DIMEQV(A%, B%) [ -5, -8, -7, -2, -1 ]  配列 A%と配列 B% 同士をEQVします。		
使用例2	DIM A(2) A = [ 7; 3; 2 ] PRINT DIMEQV(A) 6  配列Aの各要素をEQVします。		

## 3.3.21 DIMSHL

関数		
機 能	配列の各要素に対して、ビット単位に左シフトした結果を返します。	
書 式	<(戻り値) 左シフト演算配列値> = DIMSHL(<①元配列>, <②シフト数>)	
戻り値	戻り値	<左シフト演算配列値> 配列
	配列変数の各要素に対して、シフト数で指定した分、左にずらします。 最下位ビットは、0が入ります。 上記の演算を行った結果が、配列形式で得られます。	
パラ メータ	①	<元配列> 配列
	演算対象となる配列変数を指定します。 数値型である必要があります。	
	②	<シフト数> 数値
	左シフトする数を指定します。	
備 考	・ 元配列が実数の場合、戻り値の配列は整数で得られます。	
使用例	? DIMSHL([ 1; 2; 3; 4; 5 ], 2) [ 4, 8, 12, 16, 20 ] ’ 元配列に指定した配列値を、2ほど左シフトします。	

## 3.3.22 DIMSHR

関数		
機 能	配列の各要素に対して、ビット単位に右シフトした結果を返します。	
書 式	<(戻り値) 右シフト演算配列値> = DIMSHR(<①元配列>, <②シフト数>)	
戻り値	戻り値	<右シフト演算配列値> 配列
	配列変数の各要素に対して、シフト数で指定した分、右にずらします。 最上位ビットは、0が入ります。 上記の演算を行った結果が、配列形式で得られます。	
パラ メータ	①	<元配列> 配列
	演算対象となる配列変数を指定します。 数値型である必要があります。	
	②	<シフト数> 数値
	右シフトする数を指定します。	
備 考	・ 元配列が実数の場合、戻り値の配列は整数で得られます。	
使用例	? DIMSHR([ 1; 2; 3; 4; 5 ], 2) [ 0, 0, 0, 1, 1 ] ’ 元配列に指定した配列値を、2ほど右シフトします。	

## 3.3.23 DIMROL

関数		
機 能	配列の各要素に対して、ビット単位に左回転した結果を返します。	
書 式	<(戻り値) 左回転演算配列値> = DIMROL(<①元配列>, <②シフト数>)	
戻り値	戻り値	<左回転演算配列値> 配列
	配列変数の各要素に対して、シフト数で指定した分、左にずらします。 最上位ビットは、最下位ビットに回転移動します。 上記の演算を行った結果が、配列形式で得られます。	
パラ メータ	①	<元配列> 配列
	演算対象となる配列変数を指定します。 数値型である必要があります。	
	②	<シフト数> 数値
	左回転する数を指定します。	
備 考	・元配列が実数の場合、戻り値の配列は整数で得られます。	
使用例	LIST A% A% = [ 1; &H80000000 ] ? DIMROL(A%, 2) [ 4, 2 ]  ’ 元配列に指定した配列値を、2ほど左回転します。 ’ A%は単精度整数変数なので、&H80000000は、最上位ビットに 1が立った状態です ’ 左回転すると、&H80000000 は、最下位ビットに移動するので、2となります。	

## 3.3.24 DIMROR

関数		
機 能	配列の各要素に対して、ビット単位に右回転した結果を返します。	
書 式	<(戻り値) 右回転演算配列値> = DIMROR(<①元配列>, <②シフト数>)	
戻り値	戻り値	<右回転演算配列値> 配列
	配列変数の各要素に対して、シフト数で指定した分、右にずらします。 最下位ビットは、最上位ビットに回転移動します。 上記の演算を行った結果が、配列形式で得られます。	
パラ メータ	①	<元配列> 配列
	演算対象となる配列変数を指定します。 数値型である必要があります。	
	②	<シフト数> 数値
	右回転する数を指定します。	
備 考	・元配列が実数の場合、戻り値の配列は整数で得られます。	
使用例	LIST A% A% = [ 1; 4 ] ? DIMROR(A%, 2) [ 1073741824, 1 ]  ’ 元配列に指定した配列値を、2ほど右回転します。 ’ 右回転すると、最下位ビットにある値が最上位ビットに回転移動するので、 ’ 1073741824 のような大きな値が出ます。	

## 3.3.25 DIMEQ

関数			
機 能	2つの配列の各要素に対して、等しいか比較を求めた結果を返します。(書式1) または、1つの配列の各要素を、全て等しいか比較を求めた結果を返します。(書式2)		
書 式 1	<(戻り値)等しい比較配列値> = DIMEQ(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<等しい比較配列値>	配列
配列変数の各要素に対して、<配列変数1> = <配列変数2> の比較演算を行った結果が得られます。			
パラメータ	①	<配列変数1>, <配列変数2>	配列
対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。			
書 式 2	<(戻り値)等しい比較結果値> = DIMEQ(<②配列変数>)		
戻り値	戻り値	<等しい比較結果値>	数値
配列変数の各要素に対して、全て等しい(=)か、比較演算を行った結果が得られます。			
パラメータ	②	<配列変数>	配列
対象となる配列変数を指定します。 数値型 または 文字列型である必要があります。			
使用例1	<pre> DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMEQ(A%, B%) [ FALSE, FALSE, TRUE, FALSE, FALSE ] </pre>		
使用例2	<pre> DIM A(2) A = [ 5; 5; 5 ] PRINT DIMEQ(A) TRUE </pre> <p>配列Aの各要素を全て等しいか確認します。</p>		



## 3.3.26 DIMNE

関数		
機 能	2つの配列の各要素に対して、等しくないか比較を求めた結果を返します。(書式1) または、1つの配列の各要素を、全て等しくないか比較を求めた結果を返します。(書式2)	
書 式 1	<(戻り値) 等しくない比較配列値> = DIMNE(<①配列変数1>, <①配列変数2>)	
戻り値	戻り値	<等しくない比較配列値> 配列
配列変数の各要素に対して、<配列変数1> <> <配列変数2> の比較演算を行った結果が得られます。		
パラメータ	①	<配列変数1>, <配列変数2> 配列
対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。		
書 式 2	<(戻り値) 等しくない比較結果値> = DIMNE(<②配列変数>)	
戻り値	戻り値	<等しくない比較結果値> 数値
配列変数の各要素に対して、全て等しくない(<>) か、比較演算を行った結果が得られます。		
パラメータ	②	<配列変数> 配列
対象となる配列変数を指定します。 数値型 または 文字列型である必要があります。		
使用例1	<pre> DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMNE(A%, B%) [ TRUE, TRUE, FALSE, TRUE, TRUE ] </pre>	
使用例2	<pre> DIM A(2) A = [ 5; 5; 5 ] PRINT DIMNE(A) FALSE </pre> <p>配列Aの各要素を全て等しくないか確認します。</p>	

3.3.27 DIMLT

関数			
機 能	2つの配列の各要素に対して、小なりか比較を求めた結果を返します。		
書 式	<(戻り値)小なり比較配列値> = DIMLT(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<小なり比較配列値>	配列
	配列変数の各要素に対して、<配列変数1> < <配列変数2> の比較演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMLT(A%, B%) [ TRUE, TRUE, FALSE, FALSE, FALSE ]		

## 3. 3. 28 DIMGT

関数			
機 能	2つの配列の各要素に対して、大なりか比較を求めた結果を返します。		
書 式	<(戻り値) 大なり比較配列値> = DIMGT(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<大なり比較配列値>	配列
	配列変数の各要素に対して、<配列変数1> > <配列変数2> の比較演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMGT(A%, B%) [ FALSE, FALSE, FALSE, TRUE, TRUE ]		

## 3.3.29 DIMLTE

関数			
機 能	2つの配列の各要素に対して、小なりイコールか比較を求めた結果を返します。		
書 式	<(戻り値) 小なりイコール比較配列値> = DIMLTE(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<小なりイコール比較配列値>	配列
	配列変数の各要素に対して、<配列変数1> <= <配列変数2> の比較演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMLTE(A%, B%) [ TRUE, TRUE, TRUE, FALSE, FALSE ]		

### 3.3.30 DIMGTE

関数			
機 能	2つの配列の各要素に対して、大なりイコールか比較を求めた結果を返します。		
書 式	<(戻り値) 大なりイコール比較配列値> = DIMGTE(<①配列変数1>, <①配列変数2>)		
戻り値	戻り値	<大なりイコール比較配列値>	配列
	配列変数の各要素に対して、<配列変数1> >= <配列変数2> の比較演算を行った結果が得られます。		
パラ メータ	①	<配列変数1>, <配列変数2>	配列
	対象となる配列変数を指定します。 双方が数値型、または文字列型である必要があります。		
使用例	DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ 3; 3; 3; 3; 3 ] PRINT DIMGTE(A%, B%) [ FALSE, FALSE, TRUE, TRUE, TRUE ]		

## 3.3.31 DIMIIF

関数			
機 能	評価用の配列の各要素に対して、結果が真であれば真用の配列値を、偽であれば偽用の配列値を返します。		
書 式	<(戻り値)評価結果配列値> = DIMIIF(<①評価用配列変数>, <②真用配列変数>, <②偽用配列変数>)		
戻り値	戻り値	<評価結果配列値>	配列
	評価用配列変数の各要素に対して、 評価結果が真ならば、真用の配列変数の値を、 評価結果が偽であれば、偽用の配列変数の値が、結果用の配列変数に得られます。		
パラ メータ	①	<評価用配列変数>	配列
	評価を行う数値型の配列変数を指定します。		
	②	<真用配列変数>, <偽用配列変数>	配列
	評価が真の場合と偽の場合で採用する配列変数を指定します。 数値型である必要があります。		
使用例	<pre> BOOL EX(4) EX = [ TRUE; FALSE; TRUE; FALSE; TRUE ] DIM A%(4), B%(4) A% = [ 1; 2; 3; 4; 5 ] B% = [ -5; -5; -5; -5; -5 ] PRINT DIMIIF(EX, A%, B%) [ 1, -5, 3, -5, 5 ] </pre> <p>評価式(EX)の真値(A%)と偽値(B%)を渡して結果を得ます。</p>		

3. 3. 32 DIMIIF\$

関数			
機 能	評価用の配列の各要素に対して、結果が真であれば真用の文字列配列値を、偽であれば偽用の文字列配列値を返します。		
書 式	<(戻り値)評価結果配列値> = DIMIIF\$(<①評価用配列変数>, <②真用配列変数>, <②偽用配列変数>)		
戻り値	戻り値	<評価結果配列値>	配列
	評価用配列変数の各要素に対して、 評価結果が真ならば、真用の配列変数の値を、 評価結果が偽であれば、偽用の配列変数の値が、結果用の文字列配列変数に得られます。		
パラ メータ	①	<評価用配列変数>	配列
	評価を行う数値型の配列変数を指定します。		
	②	<真用配列変数>, <偽用配列変数>	配列
使用例	評価が真の場合と偽の場合で採用する配列変数を指定します。 文字列型である必要があります。		
	BOOL EX(4) EX = [ TRUE; FALSE; TRUE; FALSE; TRUE ] DIM A\$(4), B\$(4) A\$ = [ "A"; "B"; "C"; "D"; "E" ] B\$ = [ "-"; "-"; "-"; "-"; "-" ] PRINT DIMIIF\$(EX, A\$, B\$) [ A, -, C, -, E ]  評価式(EX)の真値(A\$)と偽値(B\$)を渡して結果を得ます。		

## 3. 3. 33 DIMCINT

関数		
機 能	文字列または数値配列から、単精度整数の配列に変換します。変換する対象をマスク用に配列指定できます。	
書 式	<(戻り値) 単精度整数配列値> = DIMCINT(<①変換元配列変数> [, <②マスク配列変数>])	
戻り値	戻り値	<p>&lt;単精度整数配列値&gt;</p> <p>配列変数の各要素に対して、単精度整数を得た結果が、配列形式で得られます。</p>
パラメータ	①	<p>&lt;変換元配列変数&gt;</p> <p>変換元となる配列変数を指定します。文字列配列または数値配列が指定できます。文字列を変換指定する時、数文字を与えてください。</p>
	②	<p>&lt;マスク配列変数&gt;</p> <p>指定した配列の添字の値のみ変換するよう、マスク配列変数で指定できます。マスク配列変数の、変換したい配列の添字位置に、TRUEを指定します。省略時、全ての配列要素が変換対象となります。</p>
使用例	<pre> DIM SRC\$(2) SRC\$ = ["123.4"; "456.7"; "hello"] DIM MSK(2) MSK = [TRUE; TRUE; FALSE] PRINT DIMCINT(SRC\$, MSK)  「[ 123, 457, 0] 」と表示されます。 </pre>	



## 3.3.34 DIMCLNG

関数		
機 能	文字列または数値配列から、倍精度整数の配列に変換します。変換する対象をマスク用に配列指定できます。	
書 式	<(戻り値)倍精度整数配列値> = DIMCLNG(<①変換元配列変数> [, <②マスク配列変数>])	
戻り値	戻り値	<倍精度整数配列値> 配列変数の各要素に対して、倍精度整数を得た結果が、配列形式で得られます。
パラメータ	①	<変換元配列変数> 変換元となる配列変数を指定します。文字列配列または数値配列が指定できます。文字列を変換指定する時、数文字を与えてください。
	②	<マスク配列変数> 指定した配列の添字の値のみ変換するよう、マスク配列変数で指定できます。マスク配列変数の、変換したい配列の添字位置に、TRUEを指定します。省略時、全ての配列要素が変換対象となります。
使用例	<pre> DIM SRC\$(2) SRC\$ = ["123.4"; "456.7"; "hello"] DIM MSK(2) MSK = [TRUE; TRUE; FALSE] PRINT DIMCLNG(SRC\$, MSK)  「[ 123, 457, 0] 」と表示されます。 </pre>	

## 3. 3. 35 DIMCSNG

関数		
機 能	文字列または数値配列から、単精度実数の配列に変換します。変換する対象をマスク用に配列指定できます。	
書 式	<(戻り値) 単精度実数配列値> = DIMCSNG(<①変換元配列変数> [, <②マスク配列変数>])	
戻り値	戻り値	<p>&lt;単精度実数配列値&gt;</p> <p>配列変数の各要素に対して、単精度実数を得た結果が、配列形式で得られます。</p>
パラメータ	①	<p>&lt;変換元配列変数&gt;</p> <p>変換元となる配列変数を指定します。文字列配列または数値配列が指定できます。文字列を変換指定する時、数文字を与えてください。</p>
	②	<p>&lt;マスク配列変数&gt;</p> <p>指定した配列の添字の値のみ変換するよう、マスク配列変数で指定できます。マスク配列変数の、変換したい配列の添字位置に、TRUEを指定します。省略時、全ての配列要素が変換対象となります。</p>
使用例	<pre> DIM SRC\$(2) SRC\$ = ["123.4"; "456.7"; "hello"] DIM MSK(2) MSK = [TRUE; TRUE; FALSE] PRINT DIMCSNG(SRC\$, MSK)  「[ 123.4, 456.7, 0] 」と表示されます。 </pre>	

3. 3. 36 DIMCDBL

関数			
機 能	文字列または数値配列から、倍精度実数の配列に変換します。変換する対象をマスク用に配列指定できます。		
書 式	<(戻り値)倍精度実数配列値> = DIMCDBL(<①変換元配列変数> [, <②マスク配列変数>])		
戻り値	戻り値	<倍精度実数配列値>	配列
配列変数の各要素に対して、倍精度実数を得た結果が、配列形式で得られます。			
パラ メータ	①	<変換元配列変数>	配列
	変換元となる配列変数を指定します。文字列配列または数値配列が指定できます。文字列を変換指定する時、数文字を与えてください。		
	②	<マスク配列変数>	配列
	指定した配列の添字の値のみ変換するよう、マスク配列変数で指定できます。マスク配列変数の、変換したい配列の添字位置に、TRUEを指定します。省略時、全ての配列要素が変換対象となります。		
使用例	DIM SRC\$(2) SRC\$ = ["123.4"; "456.7"; "hello"] DIM MSK(2) MSK = [TRUE; TRUE; FALSE] PRINT DIMCDBL(SRC\$, MSK)  「[ 123.4, 456.7, 0] 」と表示されます。		

## 3.3.37 DIMCSTR\$

関数		
機 能	数値配列から、数文字に変換した文字列の配列に変換します。変換する対象をマスク用に配列指定できます。	
書 式	<(戻り値) 文字列配列値> = DIMCSTR\$(<①変換元配列変数> [, <②マスク配列変数>])	
戻り値	戻り値	<文字列配列値> 配列変数の各要素に対して、数文字に変換して得た結果が、配列形式で得られます。
パラメータ	①	<変換元配列変数> 変換元となる配列変数を指定します。数値配列が指定できます。
	②	<マスク配列変数> 指定した配列の添字の値のみ変換するよう、マスク配列変数で指定できます。マスク配列変数の、変換したい配列の添字位置に、TRUEを指定します。省略時、全ての配列要素が変換対象となります。
使用例	<pre> DIM SRC(2) SRC = [123.4; 456.7; -100] DIM MSK(2) MSK = [TRUE; TRUE; FALSE] PRINT DIMCSTR\$(SRC, MSK)  「[ 123.400000, 456.700000, ] 」と表示されます。 </pre>	

3. 3. 38 DIMFIND

関数																																																										
機 能	配列に対して、検索値と一致する要素の添字番号を求めます。																																																									
書 式	<(戻り値) 添字番号配列> = DIMFIND(<①配列>, <②検索値> [, <③モード> [, <④検索開始位置>, <④検索終了位置> ]])																																																									
戻り値	戻り値	<添字番号配列> 配列 配列に対して検索値と一致した要素の添字番号を、配列に得られます。 配列が2次元配列の場合、一致した要素の情報は、n × 2の2次元配列形式が得られます。 配列に対して、一致する要素が無い時、-1 を返します。																																																								
パラ メータ	①	<配列> 配列 検索対象となる配列を指定します。 1次元配列もしくは、2次元配列を指定してください。																																																								
	②	<検索値> 数値 / 文字列 配列に対して、検索を行う値を指定します。																																																								
	③	<モード> 数値 検索方法を指定します。複数のモード値同士を OR する事ができます。 省略時、0とみなします。																																																								
	<table><tr><th>モード値</th><th>内容</th></tr><tr><td>&amp;H1</td><td>配列が文字列配列の時、検索値が部分一致で一致とみなします。 0の時は、完全一致を必要とします。</td></tr><tr><td>&amp;H2</td><td>検索が1つ一致すれば、それ以上の検索を中断します。 0の時は、全要素に対して検索を行います。</td></tr><tr><td>&amp;H4</td><td>部分一致(&amp;H1) と OR すると、前方一致で一致とみなします。</td></tr><tr><td>&amp;H8</td><td>部分一致(&amp;H1) と OR すると、後方一致で一致とみなします。</td></tr></table>		モード値	内容	&H1	配列が文字列配列の時、検索値が部分一致で一致とみなします。 0の時は、完全一致を必要とします。	&H2	検索が1つ一致すれば、それ以上の検索を中断します。 0の時は、全要素に対して検索を行います。	&H4	部分一致(&H1) と OR すると、前方一致で一致とみなします。	&H8	部分一致(&H1) と OR すると、後方一致で一致とみなします。																																														
	モード値	内容																																																								
&H1	配列が文字列配列の時、検索値が部分一致で一致とみなします。 0の時は、完全一致を必要とします。																																																									
&H2	検索が1つ一致すれば、それ以上の検索を中断します。 0の時は、全要素に対して検索を行います。																																																									
&H4	部分一致(&H1) と OR すると、前方一致で一致とみなします。																																																									
&H8	部分一致(&H1) と OR すると、後方一致で一致とみなします。																																																									
④	<検索開始位置>, <検索終了位置> 数値 検索を開始する添字番号と検索を終了する添字番号を指定します。 2次元配列の場合、2要素の1次元配列で指定します。 省略時、配列の全要素を検索するように、開始位置と終了位置が設定されます。																																																									
使用例1	LIST ARY ARY = [ 1; 2; 3; 1; 2; 3; 4; 5 ] PRINT DIMFIND(ARY, 2) ' 1次元配列に対して検索を行います。「[ 1, 4 ]」が表示されます。																																																									
使用例2	DIM ARY(5, 5) FOR Y=0 TO UBOUND(ARY, 1) FOR X=0 TO UBOUND(ARY, 2) ARY(Y, X) = X NEXT X NEXT Y PRINT DIMFIND(ARY, 5, 0, [3;5], [5;5]) ' 2次元配列に対して検索を行います。「[ 3, 5 ], [ 4, 5 ], [ 5, 5 ]」が表示されます。 ' 検索開始位置と検索終了位置は、ARY配列に対して下図の水色部分が該当します。 <table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>←2次元目</td></tr><tr><td>0</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr><tr><td>1</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr><tr><td>2</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr><tr><td>3</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr><tr><td>4</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr><tr><td>5</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td></td></tr></table> ↑1次元目			0	1	2	3	4	5	←2次元目	0	0	1	2	3	4	5		1	0	1	2	3	4	5		2	0	1	2	3	4	5		3	0	1	2	3	4	5		4	0	1	2	3	4	5		5	0	1	2	3	4	5	
	0	1	2	3	4	5	←2次元目																																																			
0	0	1	2	3	4	5																																																				
1	0	1	2	3	4	5																																																				
2	0	1	2	3	4	5																																																				
3	0	1	2	3	4	5																																																				
4	0	1	2	3	4	5																																																				
5	0	1	2	3	4	5																																																				



3.3.39 DIMGET

関数																																	
機 能		1次元または2次元の数値配列に対して、指定した範囲を部分配列として取り出します。																															
書 式		<(戻り値) 部分配列> = DIMGET(<①元配列>, <②開始位置> [, <③終了位置> ])																															
戻り値		戻り値	<部分配列>																														
		元配列に対して、指定した開始位置から終了位置までを部分配列として取得します。																															
パラ メータ		①	<元配列>																														
		取得対象となる配列を指定します。 1次元もしくは、2次元の数値配列を指定してください。																															
		②、③	<開始位置>, <終了位置>																														
		取得する部分配列の開始の添字位置から、終了の添字位置を指定します。 2次元配列の場合、2要素の1次元配列で指定します。 終了位置を省略もしくは、-1 を指定すると、配列の末尾が設定されます。																															
使用例		DIM ARY(3, 3) ARY = [ 0 to 15 ] LIST TMP TMP = DIMGET(ARY, [0; 1], [3; 2]) PRINT TMP ' 2次元配列に対して部分配列を取得します。 ' 「[1,2], [5,6], [9,10], [13,14 ]」が表示されます。 ' 開始位置と検索終了位置は、ARY配列に対して下図の水色部分が該当します。 <table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>←2次元目</td></tr><tr><td>0</td><td>0</td><td>1</td><td>2</td><td>3</td><td></td></tr><tr><td>1</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr><tr><td>2</td><td>8</td><td>9</td><td>10</td><td>11</td><td></td></tr><tr><td>3</td><td>12</td><td>13</td><td>14</td><td>15</td><td></td></tr></table> ↑1次元目			0	1	2	3	←2次元目	0	0	1	2	3		1	4	5	6	7		2	8	9	10	11		3	12	13	14	15	
	0	1	2	3	←2次元目																												
0	0	1	2	3																													
1	4	5	6	7																													
2	8	9	10	11																													
3	12	13	14	15																													

3. 3. 40 DIMGET\$

関数																																	
機 能	1次元または2次元の文字列配列に対して、指定した範囲を部分配列として取り出します。																																
書 式	<(戻り値) 部分配列> = DIMGET\$(<①元配列>, <②開始位置> [, <③終了位置> ])																																
戻り値	戻り値	<部分配列>	配列																														
	元配列に対して、指定した開始位置から終了位置までを部分配列として取得します。																																
パラ メータ	①	<元配列>	配列																														
	取得対象となる配列を指定します。 1次元もしくは、2次元の文字列配列を指定してください。																																
	②、③	<開始位置>, <終了位置>	数値																														
	取得する部分配列の開始の添字位置から、終了の添字位置を指定します。 2次元配列の場合、2要素の1次元配列で指定します。 終了位置を省略もしくは、-1 を指定すると、配列の末尾が設定されます。																																
使用例	DIM ARY\$(3, 3) ARY\$ = SPLIT\$("a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p", ", ") LIST TMP\$ TMP\$ = DIMGET\$(ARY\$, [0; 1], [3; 2]) PRINT TMP\$ ' 2次元配列に対して部分配列を取得します。 ' 「[b, c], [f, g], [j, k], [n, o]」が表示されます。 ' 開始位置と検索終了位置は、ARY配列に対して下図の水色部分が該当します。 <table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>←2次元目</td></tr><tr><td>0</td><td>a</td><td>b</td><td>c</td><td>d</td><td></td></tr><tr><td>1</td><td>e</td><td>f</td><td>g</td><td>h</td><td></td></tr><tr><td>2</td><td>i</td><td>j</td><td>k</td><td>l</td><td></td></tr><tr><td>3</td><td>m</td><td>n</td><td>o</td><td>p</td><td></td></tr></table> ↑1次元目				0	1	2	3	←2次元目	0	a	b	c	d		1	e	f	g	h		2	i	j	k	l		3	m	n	o	p	
	0	1	2	3	←2次元目																												
0	a	b	c	d																													
1	e	f	g	h																													
2	i	j	k	l																													
3	m	n	o	p																													



3.3.41 DIMMAP

関数			
機 能	配列の各要素に対して指定した関数を呼び出して、返された値からなる配列を作ります。		
書 式	〈(戻り値)結果配列〉 = DIMMAP( 〈①呼び出し関数〉, 〈②配列〉, 〈③オプション値〉 )		
戻り値	戻り値	〈結果値〉	配列
	配列の各要素に対して、呼び出し関数を呼び出した戻り値からなる数値配列が得られます。		
パラ メータ	①	〈呼び出し関数〉	関数名
	配列の各要素に対して呼び出す、ユーザー定義関数名を指定します。 ユーザ定義関数名は、以下の定義に従います。		
	<div><pre>' A ... 配列の各要素値です ' OPT ... オプション値が渡されます ' POS ... Aの添え字情報を、次元毎に整理した1次元配列が渡されます FUNCTION 呼び出し関数名(A, OPT, POS AS LIST)     処理内容     呼び出し関数名 = 戻り値 END FUNCTION</pre></div>		
	②	〈配列〉	配列
	対象となる配列変数を指定します。 数値型または文字列型である必要があります。		
	③	〈オプション値〉	値
	呼び出し関数の事例で、OPT に渡すオプション値を指定します。		
備 考	本関数は、以下の仮想コード例のように動作します。 FOR I=0 TO UBOUND(配列) 新配列(I) = 呼び出し関数名(配列(I), オプション値, 添え字情報) NEXT I		
使用例	FUNCTION ADD(A, OPT, POS AS LIST) ADD = A + OPT END FUNCTION  DIM ARY(1, 4) ARY = [ 0; 1; 2; 3; 4; 5; 6; 7; 8; 9 ] PRINT DIMMAP(ADD, ARY, 10)  ' [[ 10, 11, 12, 13, 14 ], [ 15, 16, 17, 18, 19 ]] が表示されます。		

## 3.3.42 DIMMAP\$

関数			
機 能	配列の各要素に対して指定した関数を呼び出して、返された値からなる文字列配列を作ります。		
書 式	<(戻り値) 結果配列> = DIMMAP\$( <①呼び出し関数>, <②配列>, <③オプション値> )		
戻り値	戻り値	<結果値>	配列
	配列の各要素に対して、呼び出し関数を呼び出した戻り値からなる文字列配列が得られます。		
パラ メータ	①	<呼び出し関数>	関数名
	配列の各要素に対して呼び出す、ユーザー定義関数名を指定します。 ユーザ定義関数名は、以下の定義に従います。		
	<pre> ' A ... 配列の各要素値です ' OPT ... オプション値が渡されます ' POS ... Aの添え字情報を、次元毎に整理した1次元配列が渡されます FUNCTION 呼び出し関数名\$(A, OPT, POS AS LIST)     処理内容     呼び出し関数名\$ = 戻り値 END FUNCTION </pre>		
	②	<配列>	配列
	対象となる配列変数を指定します。 数値型または文字列型である必要があります。		
	③	<オプション値>	値
	呼び出し関数の事例で、OPT に渡すオプション値を指定します。		
備 考	本関数は、以下の仮想コード例のように動作します。 FOR I=0 TO UBOUND(配列) 新配列\$(I) = 呼び出し関数名\$(配列(I), オプション値, 添え字情報) NEXT I		
使用例	<pre> FUNCTION ADD\$(A\$, OPT\$, POS AS LIST)     ADD\$ = A\$ + OPT\$ END FUNCTION  DIM ARY\$(1, 4) ARY\$ = [ "0"; "1"; "2"; "3"; "4"; "5"; "6"; "7"; "8"; "9" ] PRINT DIMMAP\$(ADD\$, ARY\$, "X") </pre> ' [[ 0X, 1X, 2X, 3X, 4X ], [ 5X, 6X, 7X, 8X, 9X ]] が表示されます。		

3. 3. 43 DIMREDUCE

関数			
機 能	配列の各要素に対して指定した関数を呼び出して、一つの値にまとめます。		
書 式	<(戻り値) 結果値> = DIMREDUCE( <①呼び出し関数>, <②配列> )		
戻り値	戻り値	<結果値>	数値
	元配列に対して、指定した開始位置から終了位置までを部分配列として取得します。		
パラ メータ	①	<呼び出し関数>	関数名
	配列の各要素に対して呼び出す、ユーザー定義関数名を指定します。 ユーザ定義関数名は、以下の定義に従います。		
	<div>FUNCTION 呼び出し関数名 (A, B)     処理内容     呼び出し関数名 = 戻り値 END FUNCTION</div>		
	②	<配列>	配列
	対象となる配列変数を指定します。 数値型である必要があります。		
備 考	本関数は、以下の仮想コード例のように動作します。 A = 配列(0) FOR I=1 TO UBOUND(配列) A = 呼び出し関数名(A, 配列(I)) NEXT I		
使用例	FUNCTION ADD(A, B) ADD = A + B END FUNCTION  LIST ARY ARY = [ 1; 2; 3; 4; 5; 6; 7; 8; 9 ] PRINT DIMREDUCE(ADD, ARY) ' ARYの各要素に対して ADD 関数が呼ばれた結果、「45」が表示されます。		

### 3.3.44 DIMSET

命令																																															
機 能	1次元または2次元の配列に対して、指定領域をコピー元配列で上書きします。																																														
書 式	DIMSET <①コピー先配列>, <②コピー元配列>, <③開始位置> [, <④終了位置> ]																																														
パラ メータ	①	<div>&lt;コピー先配列&gt;</div> <div>配列</div> <div>コピー元配列から上書きされる配列を指定します。 1次元もしくは、2次元の数値または文字列配列を指定してください。</div>																																													
	②	<div>&lt;コピー元配列&gt;</div> <div>配列</div> <div>上書きする元となる配列を指定します。 コピー先配列と同じ型の、1次元もしくは、2次元の数値または文字列配列を指定してください。</div>																																													
	②、③	<div>&lt;開始位置&gt;, &lt;終了位置&gt;</div> <div>数値</div> <div>コピー元配列からコピー先配列に対して上書きする、開始の添字位置から、終了の添字位置を指定します。 2次元配列の場合、2要素の1次元配列で指定します。 終了位置を省略もしくは、-1 を指定すると、開始位置からコピー元配列の末尾に相当する位置が設定されます。 終了位置が、開始位置からコピー元配列より大きい位置を指定した時、繰り返し上書きされます。</div>																																													
	・ 終了位置がコピー元配列の大きさを超えて指定した場合、コピー元配列の大きさで上書きは終わります。																																														
備 考																																															
使用例1	<div>DIM LV(3, 6), RV(2, 2)</div> <div>LV = [ 1 to 28 ]</div> <div>RV = [ 100 to 108 ]</div> <div>DIMSET LV, RV, [ 2; 1 ]</div> <div>PRINT LV</div> <div>' 「[[ 1, 2, 3, 4, 5, 6, 7 ], [ 8, 9, 10, 11, 12, 13, 14 ], [ 15, 100, 101, 102, 19, 20, 21 ], [ 22, 103, 104, 105, 26, 27, 28 ]]</div> <div>' 開始位置と検索終了位置は、ARY配列に対して下図の水色部分が該当します。</div> <div><table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>←2次元目</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr><tr><td>1</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td></td></tr><tr><td>2</td><td>15</td><td>50</td><td>51</td><td>52</td><td>19</td><td>20</td><td>21</td><td></td></tr><tr><td>3</td><td>22</td><td>53</td><td>54</td><td>55</td><td>26</td><td>27</td><td>28</td><td></td></tr></table><div>↑1次元目</div></div>			0	1	2	3	4	5	6	←2次元目	0	1	2	3	4	5	6	7		1	8	9	10	11	12	13	14		2	15	50	51	52	19	20	21		3	22	53	54	55	26	27	28	
	0	1	2	3	4	5	6	←2次元目																																							
0	1	2	3	4	5	6	7																																								
1	8	9	10	11	12	13	14																																								
2	15	50	51	52	19	20	21																																								
3	22	53	54	55	26	27	28																																								
使用例2	<div>DIM LV(3, 6), RV(2, 2)</div> <div>LV = [ 1 to 28 ]</div> <div>RV = [ 100 to 108 ]</div> <div>DIMSET LV, RV, [ 1; 1 ], [4; 5]</div> <div>PRINT LV</div> <div>' 「[[ 1, 2, 3, 4, 5, 6, 7 ], [ 8, 9, 10, 11, 12, 13, 14 ], [ 15, 100, 101, 102, 19, 20, 21 ], [ 22, 103, 104, 105, 26, 27, 28 ]]</div> <div>' 開始位置と検索終了位置は、ARY配列に対して下図の水色部分が該当します。</div> <div><table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>←2次元目</td></tr><tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td></td></tr><tr><td>1</td><td>8</td><td>50</td><td>51</td><td>52</td><td>50</td><td>51</td><td>14</td><td></td></tr><tr><td>2</td><td>15</td><td>53</td><td>54</td><td>55</td><td>53</td><td>54</td><td>21</td><td></td></tr><tr><td>3</td><td>22</td><td>56</td><td>57</td><td>58</td><td>56</td><td>57</td><td>28</td><td></td></tr></table><div>↑1次元目</div></div>			0	1	2	3	4	5	6	←2次元目	0	1	2	3	4	5	6	7		1	8	50	51	52	50	51	14		2	15	53	54	55	53	54	21		3	22	56	57	58	56	57	28	
	0	1	2	3	4	5	6	←2次元目																																							
0	1	2	3	4	5	6	7																																								
1	8	50	51	52	50	51	14																																								
2	15	53	54	55	53	54	21																																								
3	22	56	57	58	56	57	28																																								

## 3.3.45 NUM2BIT

関数			
機 能	数値をビット単位の配列に分解します。		
書 式	<(戻り値) ビット配列> = NUM2BIT(<①変換元数値>, <②マスクビット数> [, <③最上位ビット開始> ])		
戻り値	戻り値	<ビット配列>	配列
	数値をビット単位の 1 ないしは 0 に分解した配列が得られます。 単精度整数の配列となります。		
パラ メータ	①	<変換元数値>	数値
	変換対象となる元の数値を指定します。 対象となる数値は、単精度整数あるいは倍精度整数です。 実数を与えると、整数に変換して処理します。		
	②	<マスクビット数>	数値
	ビット分解する数値のビット数を指定します。 単精度整数は、1から32まで指定できます。 倍精度整数は、33から64まで指定できます。		
	③	<最上位ビット開始>	真偽値
	ビット分解を開始するのを、最下位ビットから行うか最上位ビットから行うか指定できます。 FALSE：最下位ビットから開始します。(省略時) TRUE：最上位ビットから開始します。		
備考	変換元数値は、配列を指定可能です。		
使用例	PRINT NUM2BIT(&H12, 8)		

## 3.3.46 BIT2NUM

関数			
機 能	ビット単位の配列を数値に結合します。		
書 式	<(戻り値) 結合数値> = BIT2NUM(<①ビット配列>, <②マスクビット数> [, <③最上位ビット開始> ])		
戻り値	戻り値	<結合数値>	配列
	ビット配列を結合して数値に変換した値が得られます。 マスクビット数の指定により、単精度整数の配列か、倍精度整数の配列が得られます。		
パラ メータ	①	<ビット配列>	配列
	変換対象となる元のビット配列を指定します。 「NUM2BIT」関数の戻り値を期待します。		
	②	<マスクビット数>	数値
	ビット結合する数値のビット数を指定します。 単精度整数は、1から32まで指定できます。 倍精度整数は、33から64まで指定できます。		
	③	<最上位ビット開始>	真偽値
	ビット結合を開始するのを、最下位ビットから行うか最上位ビットから行うか指定できます。 FALSE：最下位ビットから開始します。(省略時) TRUE：最上位ビットから開始します。		
備考			
使用例	LIST ARY ARY = NUM2BIT(&H12, 8) PRINT BIT2NUM(ARY, 8)		

## 3. 3. 47 ONEDIM SORT

関数										
機 能	数値の1次元配列に対してソートした結果を得ます。									
書 式	<(戻り値) 結果配列> = ONEDIM SORT(<①元配列> [, <②オプション>])									
戻り値	戻り値	<結果配列>	配列							
	ソートされた、新たな配列変数が得られます。									
パラ メータ	①	<元配列>	配列							
	ソートしたい配列変数を指定します。 数値型の1次元配列を指定して下さい。									
	②	<オプション>	数値							
	ソート方法を指示します。省略すると、0が採用されます。									
	<table><tr><th>オプション</th><th>動作</th></tr><tr><td>0</td><td>昇順にソートします。</td></tr><tr><td>1</td><td>降順にソートします。</td></tr><tr><td>2</td><td>予約値</td></tr></table>			オプション	動作	0	昇順にソートします。	1	降順にソートします。	2
オプション	動作									
0	昇順にソートします。									
1	降順にソートします。									
2	予約値									
使用例	LIST A A = [1;2;3;4;5] ? ONEDIM SORT(A)									

## 3. 3. 48 ONEDIM SORT\$

関数											
機 能	文字列の1次元配列に対してソートした結果を得ます。										
書 式	<(戻り値) 結果配列> = ONEDIM SORT\$(<①元配列> [, <②オプション>])										
戻り値	戻り値	<結果配列>									
	ソートされた、新たな配列変数が得られます。										
パラ メータ	①	<元配列>									
	ソートしたい配列変数を指定します。 文字列型の1次元配列を指定して下さい。										
	②	<オプション>									
	ソート方法を指示します。省略すると、0が採用されます。										
	<table><tr><th>オプション</th><th>動作</th></tr><tr><td>0</td><td>昇順にソートします。</td></tr><tr><td>1</td><td>降順にソートします。</td></tr><tr><td>2</td><td>予約値</td></tr><tr><td>4</td><td>文字列が数値として扱える時、数値として比較します。</td></tr></table>		オプション	動作	0	昇順にソートします。	1	降順にソートします。	2	予約値	4
オプション	動作										
0	昇順にソートします。										
1	降順にソートします。										
2	予約値										
4	文字列が数値として扱える時、数値として比較します。										
備 考	オプション：4を指定しつつ降順ソートする場合、4 or 1 = 5 を指定します。										
使用例	LIST A\$ A\$ = [ "abc"; "123"; "hoge" ] ? ONEDIM SORT\$(A\$)										

3.3.49 TWODIM FILTER

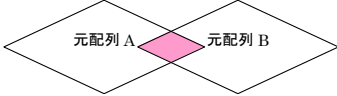
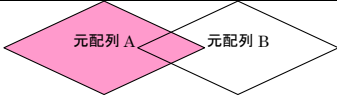
関数			
機 能	2次元の数値配列の指定された添え字の値に対して、指定した関数を呼び出し、結果が真の行ないしは列からなる配列を作ります。		
書 式	<(戻り値)新配列> = TWODIM FILTER <①方向> ( <②呼び出し関数>, <③元配列>, <④インデックス> )		
戻り値	戻り値	<新配列>	配列
	元配列に対して、行列を入れ替えた配列を得ます。		
パラ メータ	①	<方向>	キーワード
	ROW を指定すると、行毎の配列を作ります。 COLUMN を指定すると、列毎の配列を作ります。		
	②	<呼び出し関数>	関数名
	元配列の指定した添え字に対して呼び出す、ユーザー定義関数名を指定します。 ユーザー定義関数名は、以下の定義に従います。		
	<div><pre>' A ... 評価したい配列の要素値です ' POS ... A の添え字情報を、次元毎に整理した 1 次元配列が渡されます ' 戻り値 ... 新配列に抽出したいとき TRUE を返します FUNCTION 呼び出し関数名(A, POS AS LIST) AS BOOL   処理内容   呼び出し関数名 = 戻り値 END FUNCTION</pre></div>		
	③	<元配列>	配列
	評価対象となる配列を指定します。 2次元の数値配列を指定してください。		
	④	<インデックス>	数値
呼び出し関数を呼び出す際に走査する、行ないしは列番号を指定します。 方向引数で ROW を指定した時、列番号を指定します。 方向引数で COLUMN を指定した時、行番号を指定します。			
使用例	<pre>DIM ARY(2, 3) ARY = [ [ 1; 2; 3; 4 ], [ 5; 6; 7; 8 ], [ 9; 10; 11; 12 ] ]  FUNCTION OP_LTE(A, POS AS LIST) AS BOOL   PRINT "OP_LTE=", A, POS   IF A &lt;= 7 THEN OP_LTE = TRUE END FUNCTION  PRINT TWODIM FILTER ROW(OP_LTE, ARY, 2) ' OP_LTE= 3    [ 0, 2 ] ' OP_LTE= 7    [ 1, 2 ] ' OP_LTE= 11   [ 2, 2 ] ' [[ 1, 2, 3, 4 ], [ 5, 6, 7, 8 ]] ' 上記の結果が得られます。</pre>		

## 3.3.50 TWODIM FILTER\$

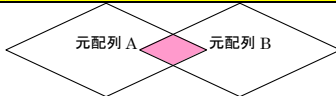
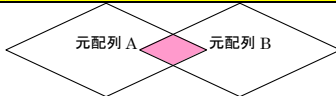
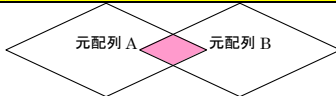
関数			
機 能	2次元の文字列配列の指定された添え字の値に対して、指定した関数を呼び出し、結果が真の行ないしは列からなる配列を作ります。		
書 式	<(戻り値) 新配列> = TWODIM FILTER\$ <①方向> ( <②呼び出し関数>, <③元配列>, <④インデックス> )		
戻り値	戻り値	<新配列>	配列
	元配列に対して、行列を入れ替えた配列を得ます。		
パラメータ	①	<方向>	キーワード
	ROW を指定すると、行毎の配列を作ります。 COLUMN を指定すると、列毎の配列を作ります。		
	②	<呼び出し関数>	関数名
	元配列の指定した添え字に対して呼び出す、ユーザー定義関数名を指定します。 ユーザー定義関数名は、以下の定義に従います。		
	<div><div>' A ... 評価したい配列の要素値です ' POS ... A の添え字情報を、次元毎に整理した 1 次元配列が渡されます ' 戻り値 ... 新配列に抽出したいとき TRUE を返します FUNCTION 呼び出し関数名(A\$, POS AS LIST) AS BOOL     処理内容     呼び出し関数名 = 戻り値 END FUNCTION</div></div>		
	③	<元配列>	配列
評価対象となる配列を指定します。 2次元の文字列配列を指定してください。			
④	<インデックス>	数値	
	呼び出し関数を呼び出す際に走査する、行ないしは列番号を指定します。 方向引数で ROW を指定した時、列番号を指定します。 方向引数で COLUMN を指定した時、行番号を指定します。		
使用例	DIM ARY\$(2, 3) ARY\$ = [ [ "1"; "2"; "3"; "4" ], [ "5"; "6"; "7"; "8" ], [ "9"; "10"; "11"; "12" ] ]  FUNCTION OP_LTE(A\$, POS AS LIST) AS BOOL PRINT "OP_LTE=", A\$, POS IF VAL(A\$) <= 7 THEN OP_LTE = TRUE END FUNCTION  PRINT TWODIM FILTER ROW(OP_LTE, ARY\$, 2) ' OP_LTE= 3 [ 0, 2 ] ' OP_LTE= 7 [ 1, 2 ] ' OP_LTE= 11 [ 2, 2 ] ' [[ 1, 2, 3, 4 ], [ 5, 6, 7, 8 ]] ' 上記の結果が得られます。		



3.3.51 TWODIM JOIN

関数			
機 能	2つの数値の2次元配列を指定した結合列を基準に、結合した配列を得ます。		
書 式	<(戻り値) 新配列> = TWODIM JOIN <①方向> ( <②結合方法>, <③元配列A>, <④結合列番号A>, <③元配列B>, <④結合列番号B> )		
戻り値	戻り値	<新配列>	配列
結合列を基準に、結合した新規配列を返します。			
パラ メータ	①	<方向>	キーワード
	ROW を指定してください。行毎の配列が作られます。		
	②	<結合方法>	数値
	2つの配列の、結合列の各値を比較し、合致した行を、どのように結合させるか、その方法を指示します。		
	結合方法	動作	備考
	1	 両方の配列で合致した行のみが抽出&結合されます。	SQL では、INNER JOIN(内部結合) に相当する処理です。
使用例	2	 元配列Aの全てと、元配列Bの合致した行が抽出&結合されます。	SQL では、LEFT OUTER JOIN(左外部結合) に相当する処理です。
	③	<元配列A>, <元配列B>	配列
	走査対象となる配列を指定します。 2次元の数値配列を指定してください。		
	④	<結合列番号A>, <結合列番号B>	数値
	元配列の評価対象となる結合列の番号を指定します。		
	DIM LV(2, 3) LV = [[ 1; 2; 3; 4 ], [ 5; 6; 7; 8 ], [ 9; 10; 11; 12 ]] DIM RV(2, 2) RV = [[ 10; 2; 12 ], [ 13; 6; 15 ], [ 16; 17; 18 ]]  ' INNTER JOIN の例 PRINT TWODIM JOIN ROW(1, LV, 1, RV, 1) ' [[ 1; 2; 3; 4; 10; 12 ], [ 5; 6; 7; 8; 13; 15 ]] が得られます  ' LEFT OUTER JOIN の例 PRINT TWODIM JOIN ROW(2, LV, 1, RV, 1) ' [[ 1; 2; 3; 4; 10; 12], [ 5; 6; 7; 8; 13; 15 ], [ 9; 10; 11; 12; 0; 0 ]] が得られます		

### 3.3.52 TWODIM JOIN\$

関数												
機能書式	2つの文字列の2次元配列を指定した結合列を基準に、結合した配列を得ます。											
	<(戻り値)新配列> = TWODIM JOIN\$ <①方向> ( <②結合方法>, <③元配列A>, <④結合列番号A>, <③元配列B>, <④結合列番号B> )											
戻り値	戻り値	<新配列>	配列									
	結合列を基準に、結合した新規配列を返します。											
パラメータ	①	<方向>	キーワード									
	ROW を指定してください。行毎の配列が作られます。											
	②	<結合方法>	数値									
	2つの配列の、結合列の各値を比較し、合致した行を、どのように結合させるか、その方法を指示します。											
	<table><tr><th>結合方法</th><th>動作</th><th>備考</th></tr><tr><td>1</td><td><div></div><p>両方の配列で合致した行のみが抽出&amp;結合されます。</p></td><td>SQL では、INNER JOIN(内部結合) に相当する処理です。</td></tr><tr><td>2</td><td><div></div><p>元配列Aの全てと、元配列Bの合致した行が抽出&amp;結合されます。</p></td><td>SQL では、LEFT OUTER JOIN(左外部結合) に相当する処理です。</td></tr></table>			結合方法	動作	備考	1	<div></div> <p>両方の配列で合致した行のみが抽出&amp;結合されます。</p>	SQL では、INNER JOIN(内部結合) に相当する処理です。	2	<div></div> <p>元配列Aの全てと、元配列Bの合致した行が抽出&amp;結合されます。</p>	SQL では、LEFT OUTER JOIN(左外部結合) に相当する処理です。
	結合方法	動作	備考									
1	<div></div> <p>両方の配列で合致した行のみが抽出&amp;結合されます。</p>	SQL では、INNER JOIN(内部結合) に相当する処理です。										
2	<div></div> <p>元配列Aの全てと、元配列Bの合致した行が抽出&amp;結合されます。</p>	SQL では、LEFT OUTER JOIN(左外部結合) に相当する処理です。										
③	<元配列A>, <元配列B>	配列										
	走査対象となる配列を指定します。 2次元の文字列配列を指定してください。											
④	<結合列番号A>, <結合列番号B>	数値										
	元配列の評価対象となる結合列の番号を指定します。											
使用例	<pre>DIM LV\$(2, 3) LV\$ = [[ "1"; "2"; "3"; "4" ], [ "5"; "6"; "7"; "8" ], [ "9"; "10"; "11"; "12" ]] DIM RV\$(2, 2) RV\$ = [[ "10"; "2"; "12" ], [ "13"; "6"; "15" ], [ "16"; "17"; "18" ]]  ' ININTER JOIN の例 PRINT TWODIM JOIN\$ ROW(1, LV\$, 1, RV\$, 1) ' [[ 1; 2; 3; 4; 10; 12 ], [ 5; 6; 7; 8; 13; 15 ]] が得られます  ' LEFT OUTER JOIN の例 PRINT TWODIM JOIN\$ ROW(2, LV\$, 1, RV\$, 1) ' [[ 1; 2; 3; 4; 10; 12], [ 5; 6; 7; 8; 13; 15 ], [ 9; 10; 11; 12; ; ]] が得られます</pre>											

3. 3. 53 TWODIM EXISTS

関数								
機 能	2つの数値の2次元配列を指定した評価列を基準に、一致した行を抜き出した配列を得ます。							
書 式	<(戻り値)新配列> = TWODIM EXISTS <①方向> ( [ <②モード>, ] <③元配列A>, <④評価列番号A>, <③元配列B>, <④評価列番号B> )							
戻り値	戻り値	<新配列> 配列						
	評価列を基準に、一致した行を抜き出した新規配列を返します。 抜き出す対象は、元配列Aです。							
パラ メータ	①	<方向> キーワード						
	ROW を指定してください。行毎の配列が作られます。							
	②	<モード> 数値						
	2つの配列を比較し、抜き出す方法を指定します。							
	<table><tr><td>モード</td><td>動作</td></tr><tr><td>1</td><td>元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。</td></tr><tr><td>2</td><td>元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。</td></tr></table>		モード	動作	1	元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。	2	元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。
	モード	動作						
	1	元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。						
2	元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。							
③	<元配列A>, <元配列B> 配列							
走査対象となる配列を指定します。 2次元の数値配列を指定してください。								
④	<評価列番号A>, <評価列番号B> 数値							
	元配列の評価対象となる結合列の番号を指定します。							
使用例	DIM X(4, 2) X = [[ 1; 1; 100 ], [ 2; 2; 200 ], [ 1; 3; 300 ], [ 2; 4; 400 ]; [ 3; 5; 500 ]] DIM Y(1, 1) Y = [[ 1; 10 ], [ 2; 20 ]]  ? TWODIM EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 1, 1, 100 ], [ 2, 2, 200 ], [ 1, 3, 300 ], [ 2, 4, 400 ]]							

### 3.3.54 TWODIM EXISTS\$

関数								
機 能	2つの文字列の2次元配列を指定した評価列を基準に、一致した行を抜き出した配列を得ます。							
書 式	<(戻り値)新配列> = TWODIM EXISTS\$ <①方向> ( [ <②モード>, ] <③元配列A>, <④評価列番号A>, <③元配列B>, <④評価列番号B> )							
戻り値	戻り値	<新配列> 配列						
	評価列を基準に、一致した行を抜き出した新規配列を返します。 抜き出す対象は、元配列Aです。							
パラ メータ	①	<方向> キーワード						
	ROW を指定してください。行毎の配列が作られます。							
	②	<モード> 数値						
	2つの配列を比較し、抜き出す方法を指定します。							
	<table><tr><td>モード</td><td>動作</td></tr><tr><td>1</td><td>元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。</td></tr><tr><td>2</td><td>元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。</td></tr></table>		モード	動作	1	元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。	2	元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。
	モード	動作						
	1	元配列AとBの指定した評価列の値が一致すればOK。 元配列AとB それぞれに重複があっても、重複削除はしません。 元配列の順序通りで結果が得られます。整列されません。						
2	元配列AとBの指定した評価列の値が一致すればOK。 元配列Bに重複があると、重複削除します。 元配列の順序通りで結果が得られます。整列されません。							
③	<元配列A>, <元配列B> 配列							
走査対象となる配列を指定します。 2次元の文字列配列を指定してください。								
④	<評価列番号A>, <評価列番号B> 数値							
元配列の評価対象となる結合列の番号を指定します。								
使用例	DIM X(4, 2) X = [[ 1; 1; 100 ], [ 2; 2; 200 ], [ 1; 3; 300 ], [ 2; 4; 400 ]; [ 3; 5; 500 ]] DIM Y(1, 1) Y = [[ 1; 10 ], [ 2; 20 ]]  ? TWODIM EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 1, 1, 100 ], [ 2, 2, 200 ], [ 1, 3, 300 ], [ 2, 4, 400 ]]」と表示されます。 ? TWODIM NOT EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 3, 5, 500 ]]」と表示されます。							

## 3.3.55 TWODIM NOT EXISTS

関数			
機 能	2つの数値の2次元配列を指定した評価列を基準に、不一致の行を抜き出した配列を得ます。		
書 式	<(戻り値)新配列> = TWODIM NOT EXISTS <①方向> ( <②元配列A>, <③評価列番号A>, <②元配列B>, <③評価列番号B> )		
戻り値	戻り値	<新配列>	配列
	評価列を基準に、不一致の行を抜き出した新規配列を返します。 抜き出す対象は、元配列Aです。		
パラ メータ	①	<方向>	キーワード
	ROW を指定してください。行毎の配列が作られます。		
	②	<元配列A>, <元配列B>	配列
	走査対象となる配列を指定します。 2次元の数値配列を指定してください。		
	③	<評価列番号A>, <評価列番号B>	数値
	元配列の評価対象となる結合列の番号を指定します。		
備 考	<p>本関数は、以下のように動作します。</p> <ol style="list-style-type: none"> <li>1. 元配列AとBの指定した評価列の値が全て不一致すればOK。 (元配列Aの評価値に対して、元配列Bの全ての評価値が不一致である事)</li> <li>2. 元配列Bに重複があると、重複削除します。</li> <li>3. 元配列の順序通りで結果が得られます。整列されません。</li> </ol>		
使用例	<pre> DIM X(4, 2) X = [[ 1; 1; 100 ], [ 2; 2; 200 ], [ 1; 3; 300 ], [ 2; 4; 400 ]; [ 3; 5; 500 ]] DIM Y(1, 1) Y = [[ 1; 10 ], [ 2; 20 ]]  ? TWODIM EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 1, 1, 100 ], [ 2, 2, 200 ], [ 1, 3, 300 ], [ 2, 4, 400 ]]」と表示されます。 ? TWODIM NOT EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 3, 5, 500 ]]」と表示されます。 </pre>		

## 3.3.56 TWODIM NOT EXISTS\$

関数			
機 能	2つの文字列の2次元配列を指定した評価列を基準に、不一致の行を抜き出した配列を得ます。		
書 式	<(戻り値)新配列> = TWODIM NOT EXISTS\$ <①方向> ( <②元配列A>, <③評価列番号A>, <②元配列B>, <③評価列番号B> )		
戻り値	戻り値	<新配列>	配列
	評価列を基準に、不一致の行を抜き出した新規配列を返します。 抜き出す対象は、元配列Aです。		
パラ メータ	①	<方向>	キーワード
	ROW を指定してください。行毎の配列が作られます。		
	②	<元配列A>, <元配列B>	配列
	走査対象となる配列を指定します。 2次元の文字列配列を指定してください。		
	③	<評価列番号A>, <評価列番号B>	数値
	元配列の評価対象となる結合列の番号を指定します。		
備 考	<p>本関数は、以下のように動作します。</p> <ol style="list-style-type: none"> <li>1. 元配列AとBの指定した評価列の値が全て不一致すればOK。 (元配列Aの評価値に対して、元配列Bの全ての評価値が不一致である事)</li> <li>2. 元配列Bに重複があると、重複削除します。</li> <li>3. 元配列の順序通りで結果が得られます。整列されません。</li> </ol>		
使用例	<pre> DIM X(4, 2) X = [[ 1; 1; 100 ], [ 2; 2; 200 ], [ 1; 3; 300 ], [ 2; 4; 400 ]; [ 3; 5; 500 ]] DIM Y(1, 1) Y = [[ 1; 10 ], [ 2; 20 ]]  ? TWODIM EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 1, 1, 100 ], [ 2, 2, 200 ], [ 1, 3, 300 ], [ 2, 4, 400 ]]」と表示されます。 ? TWODIM NOT EXISTS ROW( X, 0, Y, 0 ) ' 「[[ 3, 5, 500 ]]」と表示されます。 </pre>		

3.3.57 TWODIM SORT

関数											
機 能	数値の2次元配列の指定した行／列位置に対してソートした結果を得ます。										
書 式	<(戻り値)結果配列> = TWODIM SORT <①方向> ( <②元配列>, <③添え字> [ , <④オプション> ] )										
戻り値	戻り値	<結果配列>	配列								
	ソートされた、新たな配列変数が得られます。										
パラ メータ	①	<方向>	キーワード								
	ROW を指定すると、行方向にソートします。 COLUMN を指定すると、列方向にソートします。										
	②	<元配列>	配列								
	ソートしたい配列変数を指定します。 数値型の2次元配列を指定して下さい。										
	③	<インデックス>	数値								
	ソート対象とする位置を指定します。(0始まり) 方向を ROW 指定時、インデックスで指定した列番号をソート対象とします。 方向を COLUMN 指定時、インデックスで指定した行番号をソート対象とします。										
	④	<オプション>	数値								
	ソート方法を指示します。省略すると、0が採用されます。										
	<table><tr><th>オプション</th><th>動作</th></tr><tr><td>0</td><td>昇順にソートします。</td></tr><tr><td>1</td><td>降順にソートします。</td></tr><tr><td>2</td><td>予約値</td></tr></table>			オプション	動作	0	昇順にソートします。	1	降順にソートします。	2	予約値
	オプション	動作									
0	昇順にソートします。										
1	降順にソートします。										
2	予約値										
使用例	DIM ARY(2,2) ARY = [ 1 to 9 ] ? ARY ' [[ 1, 2, 3 ], [ 4, 5, 6 ], [ 7, 8, 9 ]] と表示されます ? TWODIM SORT ROW(ARY, 1, 1) ' [[ 7, 8, 9 ], [ 4, 5, 6 ], [ 1, 2, 3 ]] と表示されます ? TWODIM SORT COLUMN(ARY, 1, 1) ' [[ 3, 2, 1 ], [ 6, 5, 4 ], [ 9, 8, 7 ]] と表示されます										

3. 3. 58 TWODIM SORT\$

関数												
機 能	文字列の2次元配列の指定した行／列位置に対してソートした結果を得ます。											
書 式	<(戻り値)結果配列> = TWODIM SORT\$ <①方向> ( <②元配列>, <③添え字> [ , <④オプション> ] )											
戻り値	戻り値	<結果配列> 配列 ソートされた、新たな配列変数が得られます。										
パラ メータ	①	<方向> キーワード ROW を指定すると、行方向にソートします。 COLUMN を指定すると、列方向にソートします。										
	②	<元配列> 配列 ソートしたい配列変数を指定します。 文字列型の2次元配列を指定して下さい。										
	③	<インデックス> 数値 ソート対象とする位置を指定します。(0始まり) 方向を ROW 指定時、インデックスで指定した列番号をソート対象とします。 方向を COLUMN 指定時、インデックスで指定した行番号をソート対象とします。										
	④	<オプション> 数値 ソート方法を指示します。省略すると、0が採用されます。										
	<table><tr><th>オプション</th><th>動作</th></tr><tr><td>0</td><td>昇順にソートします。</td></tr><tr><td>1</td><td>降順にソートします。</td></tr><tr><td>2</td><td>予約値</td></tr><tr><td>4</td><td>文字列が数値として扱える時、数値として比較します。</td></tr></table>		オプション	動作	0	昇順にソートします。	1	降順にソートします。	2	予約値	4	文字列が数値として扱える時、数値として比較します。
	オプション	動作										
0	昇順にソートします。											
1	降順にソートします。											
2	予約値											
4	文字列が数値として扱える時、数値として比較します。											
備 考	オプション：4を指定しつつ降順ソートする場合、4 or 1 = 5 を指定します。											
使用例	DIM ARY\$(2,2) ARY\$ = SPLIT\$("a,b,c,d,e,f,g,h,i",",") ? ARY\$ ' [[ a, b, c ], [ d, e, f ], [ g, h, i ]] と表示されます ? TWODIM SORT\$ ROW(ARY\$, 1, 1) ' [[ g, h, i ], [ d, e, f ], [ a, b, c ]] と表示されます ? TWODIM SORT\$ COLUMN(ARY\$, 1, 1) ' [[ c, b, a ], [ f, e, d ], [ i, h, g ]] と表示されます											



## 3.3.59 TWODIM TRANSPOSE

関数			
機 能	2次元の数値配列の行列を入れ替えた配列を得ます。		
書 式	<(戻り値) 新配列> = TWODIM TRANSPOSE (<①元配列>)		
戻り値	戻り値	<新配列>	配列
	元配列に対して、行列を入れ替えた配列を得ます。		
パラ メータ	①	<元配列>	配列
	取得対象となる配列を指定します。 2次元の数値配列を指定してください。		
使用例	DIM ARY(2, 2) ARY = [ 0 to 9 ] LIST TMP ' [[ 0, 1, 2 ], [ 3, 4, 5 ], [ 6, 7, 8 ]] と表示されます TMP = TWODIM TRANSPOSE(ARY) PRINT TMP ' [[ 0, 3, 6 ], [ 1, 4, 7 ], [ 2, 5, 8 ]] と表示されます。		

## 3.3.60 TWODIM TRANSPOSE\$

関数			
機 能	2次元の文字列配列の行列を入れ替えた配列を得ます。		
書 式	<(戻り値) 新配列> = TWODIM TRANSPOSE\$ (<①元配列>)		
戻り値	戻り値	<新配列>	配列
	元配列に対して、行列を入れ替えた配列を得ます。		
パラ メータ	①	<元配列>	配列
	取得対象となる配列を指定します。 2次元の文字列配列を指定してください。		
使用例	DIM ARY\$(2, 2) ARY\$ = SPLIT\$("a, b, c, d, e, f, g, h, i", ",") LIST TMP\$ ' [[ a, b, c ], [ d, e, f ], [ g, h, i ]] と表示されます TMP\$ = TWODIM TRANSPOSE\$(ARY\$) PRINT TMP\$ ' [[ a, d, g ], [ b, e, h ], [ c, f, i ]] と表示されます。		

### 3.4 Python連携に関する関数・命令

ここでは、Pythonと連携するためのコマンドおよび関数群を紹介します。

#### 3.4.1 PYOBJ CREATE CODE

関数			
機能	Pythonと連携する為のコード文字列を渡して初期化します。		
書式	<(戻り値) ID値> = PYOBJ CREATE CODE(<①Pythonコード文字列>)		
戻り値	戻り値	<ID値>	数値
	Pythonのコード文字列を受け取って初期化した結果に対する、一意のID値が得られます。 このID値は、その他のPython連携の関数およびコマンドで利用します。		
パラメータ	①	<Pythonコード文字列>	文字列
	Pythonのコード文字列を与えてください。		
備考	<ul style="list-style-type: none"> <li>Python連携で利用できるモジュール群は、python3から利用できるモジュールです。python2のモジュールは使用できません。</li> <li>以下のモジュールは、利用できません。 tkinter</li> </ul>		
使用例	<pre>s\$ = "def hoge(s):" + chr\$(10) s\$ = s\$ + " print(s)" + chr\$(10) s\$ = s\$ + " s = s.upper()" + chr\$(10) s\$ = s\$ + " return s" + chr\$(10)  id = PYOBJ CREATE CODE(s\$) print "pyobj create:"; id  ? PYOBJ CALL FUNCTION(id, "hoge", "hello")  PYOBJ CLOSE id</pre> <p>s\$のPythonコード文字列を渡して、PYOBJ CREATE CODE関数で初期化し、後から呼び出せるようにしています。 Pythonコード文字列内に、「hoge」関数を定義して、PYOBJ CALL FUNCTIONから呼び出しています。 内部では、文字列に対する upperメソッドが呼ばれて、「hello」の表示結果は"HELLO"となります。 最後に、PYOBJ CLOSEで、終了します。</p>		

### 3.4.2 PYOBJ CALL FUNCTION

関数																					
機 能		Pythonの指定した関数を呼び出します。																			
書 式		<(戻り値)結果値> = PYOBJ CALL FUNCTION(<①ID値>, <②関数名>, [ <③引数値>, ... ])																			
戻り値	戻り値	<結果値>	値																		
	Pythonの関数を呼び出した結果値を得ます。 Pythonの戻り値の型とAJANで受け取る型の関係は、以下の通りです。																				
	<table><tr><th>Python の型</th><th>AJAN の型</th></tr><tr><td>int 型</td><td>倍精度整数型</td></tr><tr><td>float 型</td><td>倍精度実数型</td></tr><tr><td>str 型</td><td>文字列型</td></tr><tr><td>bytes 型</td><td>文字列型</td></tr><tr><td>bool 型</td><td>BOOL 型</td></tr><tr><td>list オブジェクト</td><td>(DIM や LIST などの)配列</td></tr><tr><td>dict オブジェクト</td><td>DICT 型</td></tr><tr><td>上記以外</td><td>文字列型 (Python の repr 関数を呼び出したのと類似の結果)</td></tr></table>			Python の型	AJAN の型	int 型	倍精度整数型	float 型	倍精度実数型	str 型	文字列型	bytes 型	文字列型	bool 型	BOOL 型	list オブジェクト	(DIM や LIST などの)配列	dict オブジェクト	DICT 型	上記以外	文字列型 (Python の repr 関数を呼び出したのと類似の結果)
	Python の型	AJAN の型																			
int 型	倍精度整数型																				
float 型	倍精度実数型																				
str 型	文字列型																				
bytes 型	文字列型																				
bool 型	BOOL 型																				
list オブジェクト	(DIM や LIST などの)配列																				
dict オブジェクト	DICT 型																				
上記以外	文字列型 (Python の repr 関数を呼び出したのと類似の結果)																				
パラ メータ	①	<ID値>	数値																		
	PYOBJ CREATE CODEで得られたID値を与えてください。																				
	②	<関数名>	文字列																		
	PYOBJ CREATE CODEで定義されたコード中の関数名を指定してください。																				
	③	<引数値>	値																		
関数に渡す引数値を指定します。 渡す引数値とPythonで受け取る型の関係は、以下の通りです。																					
<table><tr><th>AJAN の型</th><th>Python の型</th></tr><tr><td>単精度整数型</td><td>int 型</td></tr><tr><td>倍精度整数型</td><td>int 型</td></tr><tr><td>単精度実数型</td><td>float 型</td></tr><tr><td>倍精度実数型</td><td>float 型</td></tr><tr><td>文字列型</td><td>str 型</td></tr><tr><td>BOOL 型</td><td>bool 型</td></tr><tr><td>(DIM や LIST などの)配列</td><td>list オブジェクト</td></tr><tr><td>DICT 型</td><td>dict オブジェクト</td></tr></table>				AJAN の型	Python の型	単精度整数型	int 型	倍精度整数型	int 型	単精度実数型	float 型	倍精度実数型	float 型	文字列型	str 型	BOOL 型	bool 型	(DIM や LIST などの)配列	list オブジェクト	DICT 型	dict オブジェクト
AJAN の型	Python の型																				
単精度整数型	int 型																				
倍精度整数型	int 型																				
単精度実数型	float 型																				
倍精度実数型	float 型																				
文字列型	str 型																				
BOOL 型	bool 型																				
(DIM や LIST などの)配列	list オブジェクト																				
DICT 型	dict オブジェクト																				
備 考		・ Ver. 1.00より、Pythonが空のlistオブジェクトを返すと、空の配列となります。 空か否かは、LDIM関数で確認ください。																			
使用例		PYOBJ CREATE CODE関数の使用例を参照ください。																			

## 3.4.3 PYOBJ CALL METHOD

関数																					
機 能	Pythonの指定のオブジェクトに対するメソッドを呼び出します。																				
書 式	<(戻り値)結果値> = PYOBJ CALL METHOD(<①ID値>, <②オブジェクト名>, <③メソッド名>, [ <④引数値>, ... ])																				
戻り値	戻り値	<結果値>	値																		
	Pythonの指定したオブジェクトのメソッドを呼び出した結果値を得ます。 Pythonの戻り値の型とAJANで受け取る型の関係は、以下の通りです。																				
	<table><tr><th>Python の型</th><th>AJAN の型</th></tr><tr><td>int 型</td><td>倍精度整数型</td></tr><tr><td>float 型</td><td>倍精度実数型</td></tr><tr><td>str 型</td><td>文字列型</td></tr><tr><td>bytes 型</td><td>文字列型</td></tr><tr><td>bool 型</td><td>BOOL 型</td></tr><tr><td>list オブジェクト</td><td>(DIM や LIST などの)配列</td></tr><tr><td>dict オブジェクト</td><td>DICT 型</td></tr><tr><td>上記以外</td><td>文字列型 (Python の repr 関数を呼び出したのと類似の結果)</td></tr></table>			Python の型	AJAN の型	int 型	倍精度整数型	float 型	倍精度実数型	str 型	文字列型	bytes 型	文字列型	bool 型	BOOL 型	list オブジェクト	(DIM や LIST などの)配列	dict オブジェクト	DICT 型	上記以外	文字列型 (Python の repr 関数を呼び出したのと類似の結果)
	Python の型	AJAN の型																			
int 型	倍精度整数型																				
float 型	倍精度実数型																				
str 型	文字列型																				
bytes 型	文字列型																				
bool 型	BOOL 型																				
list オブジェクト	(DIM や LIST などの)配列																				
dict オブジェクト	DICT 型																				
上記以外	文字列型 (Python の repr 関数を呼び出したのと類似の結果)																				
パラ メータ	①	<ID値>	数値																		
	PYOBJ CREATE CODEで得られたID値を与えてください。																				
	②	<オブジェクト名>	文字列																		
	PYOBJ CREATE CODEで定義されたコード中のオブジェクト名を指定してください。																				
	③	<メソッド名>	文字列																		
	PYOBJ CREATE CODEで定義されたコード中のオブジェクトに対するメソッド名を指定してください。																				
	④	<引数値>	値																		
	メソッドに渡す引数値を指定します。 渡す引数値とPythonで受け取る型の関係は、以下の通りです。																				
<table><tr><th>AJAN の型</th><th>Python の型</th></tr><tr><td>単精度整数型</td><td>int 型</td></tr><tr><td>倍精度整数型</td><td>int 型</td></tr><tr><td>単精度実数型</td><td>float 型</td></tr><tr><td>倍精度実数型</td><td>float 型</td></tr><tr><td>文字列型</td><td>str 型</td></tr><tr><td>BOOL 型</td><td>bool 型</td></tr><tr><td>(DIM や LIST などの)配列</td><td>list オブジェクト</td></tr><tr><td>DICT 型</td><td>dict オブジェクト</td></tr></table>			AJAN の型	Python の型	単精度整数型	int 型	倍精度整数型	int 型	単精度実数型	float 型	倍精度実数型	float 型	文字列型	str 型	BOOL 型	bool 型	(DIM や LIST などの)配列	list オブジェクト	DICT 型	dict オブジェクト	
AJAN の型	Python の型																				
単精度整数型	int 型																				
倍精度整数型	int 型																				
単精度実数型	float 型																				
倍精度実数型	float 型																				
文字列型	str 型																				
BOOL 型	bool 型																				
(DIM や LIST などの)配列	list オブジェクト																				
DICT 型	dict オブジェクト																				
備 考	・ Ver. 1.00より、Pythonが空のlistオブジェクトを返すと、空の配列となります。 空か否かは、LDIM関数で確認ください。																				
使用例	s\$ = "s = 'hoge, fuga, test'" + chr\$(10)  id = PYOBJ CREATE CODE(s\$) print "pyobj create:"; id																				

```
? PYOBJ CALL METHOD(id, "s", "replace", ",", "+")
```

s\$に、Python文字列を設定して、PYOBJ CREATE CODEで初期化しています。

Python文字列は、内部で「s」変数に” hoge, fuga, test” という文字列を設定します。

PYOBJ CALL METHODは、「s」変数に対して、「replace」メソッドを呼び出して、結果を得ています。

この動きは、Pythonの記述では、おおよそ以下と同じです。

```
s = 'hoge, fuga, test'
print(s.replace(",", "+"))
```

## 3.4.4 PYOBJ CLOSE

命令			
機 能	Pythonとの連携処理を閉じます。		
書 式	PYOBJ CLOSE <①ID値>		
パラ メータ	①	<ID値>	数値
	PYOBJ CREATE CODEで得られたID値を与えてください。		
使用例	PYOBJ CREATE CODE関数の使用例を参照ください。		

## 3.5 オーディオ入出力を使ったアナログ変換に関する関数・命令

### 3.5.1 SNDOPEN

命令																																																			
機能	オーディオ入出力をオープンします。																																																		
書式	SNDOPEN "<①デバイス名および追加パラメータ>" FOR <②入出力方向> AS #<③オーディオ番号>																																																		
パラメータ	①	<デバイス名および追加パラメータ>		文字列																																															
	オーディオ入出力するためのデバイス名および追加パラメータを指定します。 追加パラメータはデバイス名の後に、「?」の後に、「キー1=値1&キー2=値2」のような形式で記述します。 デバイス名、追加パラメータは省略可能であり、省略時 OS の設定が採用されます。																																																		
	追加パラメータのキーと値の組み合わせを、以下に示します。																																																		
	<table><tr><td>キーワード</td><td colspan="3">値</td></tr><tr><td>RATE</td><td colspan="3">サンプリング周波数を、Hz 単位で指定します。 1～192000 の範囲で指定する事が推奨されます。 省略時、8000Hz です。</td></tr><tr><td>FORMAT</td><td colspan="3">入出力時の数値型を指定します。 省略時、S16LE です。<table><tr><td>値</td><td colspan="2">効果</td></tr><tr><td>U8</td><td colspan="2">内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。</td></tr><tr><td>S16LE</td><td colspan="2">内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。</td></tr><tr><td>S32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。</td></tr><tr><td>FLOAT32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。</td></tr></table></td></tr><tr><td>TLENGTH</td><td colspan="3">出力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。</td></tr><tr><td>PREBUF</td><td colspan="3">出力開始するサイズ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。(TLENGTH と同じ値が採用される)</td></tr><tr><td>FRAGSIZE</td><td colspan="3">入力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。</td></tr><tr><td>MAXLENGTH</td><td colspan="3">入出力バッファの最大長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。</td></tr><tr><td>MINREQ</td><td colspan="3">出力時の最小要求長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。</td></tr></table>				キーワード	値			RATE	サンプリング周波数を、Hz 単位で指定します。 1～192000 の範囲で指定する事が推奨されます。 省略時、8000Hz です。			FORMAT	入出力時の数値型を指定します。 省略時、S16LE です。 <table><tr><td>値</td><td colspan="2">効果</td></tr><tr><td>U8</td><td colspan="2">内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。</td></tr><tr><td>S16LE</td><td colspan="2">内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。</td></tr><tr><td>S32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。</td></tr><tr><td>FLOAT32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。</td></tr></table>			値	効果		U8	内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。		S16LE	内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。		S32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。		FLOAT32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。		TLENGTH	出力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。			PREBUF	出力開始するサイズ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。(TLENGTH と同じ値が採用される)			FRAGSIZE	入力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。			MAXLENGTH	入出力バッファの最大長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。			MINREQ	出力時の最小要求長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。		
	キーワード	値																																																	
	RATE	サンプリング周波数を、Hz 単位で指定します。 1～192000 の範囲で指定する事が推奨されます。 省略時、8000Hz です。																																																	
	FORMAT	入出力時の数値型を指定します。 省略時、S16LE です。 <table><tr><td>値</td><td colspan="2">効果</td></tr><tr><td>U8</td><td colspan="2">内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。</td></tr><tr><td>S16LE</td><td colspan="2">内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。</td></tr><tr><td>S32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。</td></tr><tr><td>FLOAT32LE</td><td colspan="2">内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。</td></tr></table>			値	効果		U8	内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。		S16LE	内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。		S32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。		FLOAT32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。																																	
	値	効果																																																	
	U8	内部では、8 ビット単位で入出力します。 得られる範囲は、0～255 の単精度整数です。																																																	
	S16LE	内部では、16 ビット単位で入出力します。 得られる範囲は、-32,768～32,767 の単精度整数です。																																																	
S32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-2,147,483,648～2,147,483,647 の単精度整数です。																																																		
FLOAT32LE	内部では、32 ビット単位で入出力します。 得られる範囲は、-1.0～1.0 の単精度実数です。																																																		
TLENGTH	出力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。																																																		
PREBUF	出力開始するサイズ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。(TLENGTH と同じ値が採用される)																																																		
FRAGSIZE	入力バッファ長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。																																																		
MAXLENGTH	入出力バッファの最大長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。																																																		
MINREQ	出力時の最小要求長を指定します。(バイト単位) 省略時、システムが最適な値を決めます。																																																		
②	<入出力方向>		キーワード																																																
オーディオ入力の場合「INPUT」、オーディオ出力の場合「OUTPUT」を指定します。																																																			
③	<オーディオ番号>		数値																																																
オーディオ入出力に関連付けるオーディオ番号を、1～15の範囲で指定します。																																																			
備考	・オーディオ入出力の操作は、内部でPulseAudioシステムが使用されています。 ・途切れなく入出力を行いたい場合、入力バッファ (FRAGSIZE など) や出力バッファ (TLENGTHおよびPREBUFなど) の長さを調整してください。 ・FRAGSIZEやTLENGTHなどの値を変更した際、SNDREADLEN関数やSNDWRITELEN関数																																																		

	の得られる値に影響はしますが、指定した値と同じになるとは限りません。 ・ FRAGSIZEやTLENGTHなどで指定する際、バッファに用意したい件数から必要なバイト長を計算するには、件数 × FORMATで必要なバイト長 で求めます。
使用例	SNDOPEN "?FORMAT=FLOAT32LE&RATE=8000" FOR INPUT AS #1  8kHzでオーディオ入力としてオープンします。

3.5.2 Sndclose

命令			
機能	オーディオ入出力をクローズします。		
書式	Sndclose #<①オーディオ番号>		
パラメータ	①	<オーディオ番号>	数値
	SNDOPENでオープンしたオーディオ番号を指定します。		
使用例	SNDOPEN "" FOR OUTPUT AS #1 Sndclose #1		



## 3.5.3 SNDREAD

関数			
機 能	指定件数をオーディオ入力からアナログ変換し、データ配列を得ます。		
書 式	<(戻り値)データ配列> = SNDREAD(<①オーディオ番号>, <②入力件数>)		
戻り値	戻り値	<データ配列>	配列
	入力件数で指定した分のオーディオ入力データが、1次元の数値配列で得られます。 数値配列内の個々の値は、SNDOPENのFORMATパラメータで指定した値に準じます。		
パラ メータ	①	<オーディオ番号>	数値
	SNDOPENでオープンしたオーディオ番号を指定します。		
	②	<入力件数>	数値
	オーディオ入力するデータの件数を指定します。 2022/4より、件数に負数を指定すると、入力バッファに有る一塊分のデータを取得できます。(取得できる件数は、下位ライブラリが自動調整するので不定です)		
備 考	<ul style="list-style-type: none"> <li>• SNDOPEN の呼び出し後、SNDREAD または SNDREADLEN を呼び出す事で、オーディオ入力が始まります。</li> <li>• オーディオ入力開始から、入力データは一旦入力バッファに格納されます。 SNDREAD で指定した入力件数分のデータが、入力バッファに有れば、データ配列に取得して即戻ります。 入力件数に対して、入力バッファのデータが不足する場合、指定した入力件数を満たすまで、SNDREAD の呼び出しは待機されます。</li> <li>• 2022/4より、入力件数を負数で指定したとき、入力バッファに有る一塊のデータを取得します。(取得できる件数は、下位ライブラリが自動調整するので不定です) このため、SNDREAD の呼び出しは待機されずに、データを取得できます。</li> </ul>		
使用例	SNDOPEN "?FORMAT=FLOAT32LE" FOR INPUT AS #1 LIST ARY ARY = SNDREAD(1, 256)     ' 256件のオーディオ入力します SNDCLOSE #1		

## 3.5.4 SNDREADLEN

関数			
機 能	オーディオ入力の入力バッファの件数を取得します。		
書 式	<(戻り値)入力バッファ件数> = SNDREADLEN(<①オーディオ番号> )		
戻り値	戻り値	<入力バッファ件数>	数値
	オーディオ入力から入力バッファに蓄えられた、入力バッファ件数が得られます。		
パラ メータ	①	<オーディオ番号>	数値
	SNDOPENでオープンしたオーディオ番号を指定します。		
備 考	・SNDOPEN の呼び出し後、SNDREAD または SNDREADLEN を呼び出す事で、オーディオ入力が始まります。		
使用例	SZ = SNDREADLEN(1)      ’ 入力バッファ件数を得ます LIST ARY ARY = SNDREAD(1, SZ)    ’ 入力バッファ件数分を入力します		

### 3.5.5 SNDWRITE

命令		
機 能	数値配列のデータを、アナログ変換してオーディオ出力します。	
書 式	SNDWRITE #<①オーディオ番号>, <②データ配列>	
パラ メータ	①	<div>&lt;オーディオ番号&gt;</div> <div>数値</div> <div>SNDOPENでオープンしたオーディオ番号を指定します。</div>
	②	<div>&lt;データ配列&gt;</div> <div>配列</div> <div>オーディオ出力するデータを、1次元の数値配列で指定します。 数値配列の要素数分が、出力される件数に一致します。</div>
備 考	<ul style="list-style-type: none"> <li>データ配列に指定する数値型および範囲は、SNDOPENのFORMATパラメータで指定した型になるよう与えてください。</li> <li>SNDWRITEの呼び出しで、一旦出力バッファに格納された後、オーディオ出力されます。出力バッファが空になると、オーディオ出力は停止します。</li> <li>データ配列の件数が、出力バッファの件数を下回る時、SNDWRITE の呼び出しで、出力データを出力バッファに格納した後、即戻ります。 データ配列の件数が、出力バッファの件数を上回る時、出力データが出力バッファに格納されるまで、SNDWRITEの呼び出しは待機されます。</li> </ul>	
使用例	<pre> SNDOPEN "?FORMAT=FLOAT32LE" FOR OUTPUT AS #1 DIM ARY(255) FOR I=0 TO LDIM(ARY)-1   ARY(I) = RND() NEXT I SNDWRITE #1, ARY ' ARY配列(256件)をオーディオ出力します。 SNDCLOSE #1           </pre>	

3.5.6 SNDWRITELEN

関数			
機 能	オーディオ出力の出力バッファの書き込み可能件数を取得します。		
書 式	<(戻り値)出力バッファ可能件数> = SNDWRITELEN(<①オーディオ番号> )		
戻り値	戻り値	<出力バッファ可能件数>	数値
	オーディオ出力の出力バッファの書き込み可能件数が得られます。		
パラ メータ	①	<オーディオ番号>	数値
	SNDOPENでオープンしたオーディオ番号を指定します。		
使用例	SZ = SNDWRITELEN(1)        ’ 出力バッファの書き込み可能件数を得ます LIST ARY REDIM ARY (SZ / 2) SNDWRITE #1, ARY            ’ 出力バッファの1/2まで、出力データを書き込みます		

### 3.6 ファイル・フォルダに関する関数・命令

#### 3.6.1 CURDIR\$

関数			
機 能	現在のカレントディレクトリのパスを取得します。		
書 式	<(戻り値)パス名> = CURDIR\$		
戻り値	戻り値	<パス名>	文字列
	現在のカレントディレクトリのパス名を得ます。		
使用例	PRINT CURDIR\$ 「/home/user/AjanProWS/APP」のような、実行時のカレントフォルダ名が得られます。		

### 3.7 プロセスに関する関数・命令

#### 3.7.1 GETCOMMANDLINEARGS\$

関数			
機 能	コマンドライン引数を文字列配列形式で得ます。		
書 式	<(戻り値) コマンドライン引数配列> = GETCOMMANDLINEARGS\$()		
戻り値	戻り値	<コマンドライン引数配列>	配列
	コマンドライン引数を、文字列配列形式で得ます。		
備 考	<ul style="list-style-type: none"><li>この関数は、IDE上からでなく、プログラムをコンパイルした結果の、実行プログラム自身を外部から呼び出す際の、コマンドライン引数を取得する手段として提供しています。</li><li>コマンドライン引数の概念については、Linuxに関する書籍を参照ください。</li></ul>		
使用例	<pre>LIST ARG\$ ARG\$ = GETCOMMANDLINEARGS\$() PRINT ARG\$</pre> <p>上のコードをコンパイルして、「app1」というプログラム名にしたと仮定します。 次に、Linuxの端末上(IDEではありません)から、以下のようにプログラムを呼び出したと仮定します。 # app1 hoge fuga その際に、得られる結果は、“app1”、“hoge”、“fuga”の一次元文字列配列です。</p>		

### 3.7.2 GETSYSTEMINFO\$

関数			
機 能		指定した取得IDに対応するシステムに関連する情報を得ます。	
書 式		<(戻り値)情報文字列> = GETSYSTEMINFO\$( <①取得ID> )	
戻り値		戻り値	<情報文字列>
		取得IDで指定した、システムに関連する情報を文字列で得ます。	
パラメータ		①	<取得ID>
		取得したい情報を、文字列で指定します。	
使用例		PRINT GETSYSTEMINFO\$("USERNAME") ' ユーザ名を得ます。例えば「user」などが得られます。  PRINT GETSYSTEMINFO\$("HOME_DIR") ' ログインしているユーザのホームディレクトリのパスを得ます。 ' 例えば、「/home/user」などが得られます。	

## 3.7.3 ENVIRON\$

関数			
機 能	環境変数文字列テーブル内の環境文字列を得ます。		
書 式1	<(戻り値) 環境変数値> = ENVIRON\$( <①環境文字列名> )		
戻り値	戻り値	<環境変数値>	文字列
	<環境文字列名>に対応する、環境変数の値文字列が得られます。		
パラメータ	①	<環境文字列名>	文字列
	環境変数文字列テーブルから、取得したい環境変数名を指定します。		
書 式2	<(戻り値) 環境変数名配列> = ENVIRON\$( )		
戻り値	戻り値	<環境変数名配列>	配列
	全環境変数の環境変数名を文字列配列で得られます。		
備 考	<ul style="list-style-type: none"> <li>この関数は、Linuxの端末上から、実行プログラム自身を呼び出す際に、Linux端末のシェルから引き渡される環境変数と呼ばれる情報を、参照する為に用意しています。</li> <li>どのような環境変数情報があるか確認するには、Linux端末上から、「env」コマンドを打鍵して呼び出す事で確認できます。</li> <li>環境変数の概念やシェルについては、Linuxに関する書籍を参照ください。</li> </ul>		
使用例	<pre>PRINT ENVIRON\$("HOME")</pre> <pre>/home/user</pre> <p>環境変数"HOME"の値が得られます。</p>		



## 3.8 グローバル共有に関する関数・命令

### 3.8.1 COMMON OPEN

命令																
機 能	グローバル共有機能をオープンします。															
書 式	COMMON OPEN "<①IPアドレスとポート番号>" AS #<②管理番号>															
パラ メータ	①	<IPアドレスとポート番号> 文字列 グローバル共有にアクセスする為のIPアドレスとポート番号を指定します。 指定は、「IPアドレス：ポート番号」の形式で記述します。 ポート番号を省略して「IPアドレス」の形式でも記述できます。 (省略したとき、ポート番号は、redisの基本ポート番号 6379 が採用されます)  自身のコンピュータを指す場合、IPアドレスは「127.0.0.1」を指定してください。 別のコンピュータを指す場合は、相手先のIPアドレスを指定してください。  <IPアドレスとポート番号>を文字列配列で、添字1からオプション設定を指定できます。 このオプション設定は、グローバル共有が使用している redis の機能に強く関係します。 設定できるオプション設定は、以下の通りです。														
	<table><tr><th>オプション</th><th>挙動</th></tr><tr><td>"SELECT=&lt;数値&gt;"</td><td>redisサーバに対して、使用するデータベースの番号を指定します。何も指定しないと、0 が採用されます。</td></tr><tr><td>"PASSWORD=&lt;文字列&gt;"</td><td>redisサーバがパスワード保護されているとき、認証を受けるためのパスワード文字列を指定できます。</td></tr><tr><td>"OLD=1"</td><td>備考2を、参照ください。</td></tr><tr><td>"CONNTIMEOUT=&lt;数値&gt;"</td><td>COMMON OPEN時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。</td></tr><tr><td>"TIMEOUT=&lt;数値&gt;"</td><td>読み書き時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。</td></tr><tr><td>"RECONNECT=&lt;1 or 0&gt;"</td><td>1を設定すると、自動再接続機能を有効にします。何も指定しないと、0が採用され、自動再接続は行いません。</td></tr></table>		オプション	挙動	"SELECT=<数値>"	redisサーバに対して、使用するデータベースの番号を指定します。何も指定しないと、0 が採用されます。	"PASSWORD=<文字列>"	redisサーバがパスワード保護されているとき、認証を受けるためのパスワード文字列を指定できます。	"OLD=1"	備考2を、参照ください。	"CONNTIMEOUT=<数値>"	COMMON OPEN時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。	"TIMEOUT=<数値>"	読み書き時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。	"RECONNECT=<1 or 0>"	1を設定すると、自動再接続機能を有効にします。何も指定しないと、0が採用され、自動再接続は行いません。
	オプション	挙動														
"SELECT=<数値>"	redisサーバに対して、使用するデータベースの番号を指定します。何も指定しないと、0 が採用されます。															
"PASSWORD=<文字列>"	redisサーバがパスワード保護されているとき、認証を受けるためのパスワード文字列を指定できます。															
"OLD=1"	備考2を、参照ください。															
"CONNTIMEOUT=<数値>"	COMMON OPEN時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。															
"TIMEOUT=<数値>"	読み書き時のタイムアウト時間を秒単位で指定します。何も指定しないと、OSの設定に依存します。															
"RECONNECT=<1 or 0>"	1を設定すると、自動再接続機能を有効にします。何も指定しないと、0が採用され、自動再接続は行いません。															
②	<管理番号> 数値 グローバル共有に関連付ける管理番号を、1～15の範囲で指定します。															
備 考	<ul style="list-style-type: none"><li>グローバル共有機能は、プロセス同士で値をやり取りできる機能です。やり取りする値は一意であり、ネットワークをまたぐ事もできます。</li><li>グローバル共有機能の操作は、内部で redis-server サービスが使用されます。</li></ul>															
備考 2	<ul style="list-style-type: none"><li>Ver. 1.00より、グローバル共有への値の格納方法が変更となりました。この為、2021/6より以前に書き込んだ値を読み取ろうとする場合、「OLD=1」で一旦オープンしてから値を読み取ってください。</li><li>古い形式の値を、「OLD=1」を指定せずに読み取ろうとするとエラーとなります。(例：「古い形式のシリアルライズフォーマットのデータを、新しい形式で読み取ろうとしている可能性があります」のようなメッセージ)</li><li>自動再接続機能は、回線切断などの事由によりコマンドエラーが発生した後、再びコマンドを呼び出した時、COMMON OPEN に相当する処理を自動で行います。再接続時、COMMON OPEN 呼び出し時のパラメータが採用されます。</li><li>TIMEOUTパラメータの適用対象は、COMMON OPEN 以外のグローバル共有コマンドが対</li></ul>															

	<p>象です。</p> <ul style="list-style-type: none"><li>・グローバル共有のサーバがTLS暗号化通信を必要とする設定の時、「CACERT_FILENAME」などのオプションを何も設定しないと、オープン自体は成功するかも知れませんが、実際の読み書き時にはエラーとなります。</li></ul>
使用例1	<p>COMMON OPEN "127.0.0.1" AS #1</p> <p>自身のコンピュータに対して、グローバル共有をオープンします。</p>
使用例2	<p>LIST NM\$ NM\$ = [ "127.0.0.1"; "SELECT=1" ] COMMON OPEN NM\$ AS #1</p> <p>データベース番号1を指定しつつ、自分のコンピュータに対して、グローバル共有をオープンします。</p>

## 3.8.2 COMMON CLOSE

命令			
機 能	グローバル共有をクローズします。		
書 式	COMMON CLOSE #<①管理番号>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
備 考	<ul style="list-style-type: none"> <li>ON COMMON CALL による割り込み処理中に COMMON CLOSE を呼び出すと、割り込み処理は中断されます。 (Ver. 1.00より前は、割り込み処理が完了するまでブロッキングされていました)</li> </ul>		
使用例	COMMON OPEN "127.0.0.1" AS #1 CNT = 123 COMMON PRINT #1, CNT CNT = 0           ' 一旦クリア COMMON INPUT #1, CNT PRINT CNT       ' グローバル共有に格納された値が表示されます COMMON CLOSE #1		

## 3.8.3 COMMON FREEFILE

関数			
機 能	使用可能な、グローバル共有の管理番号を得ます。		
書 式	<(戻り値)管理番号> = COMMON FREEFILE( [ <検索開始番号> ] )		
戻り値	戻り値	<管理番号>	数値
	COMMON OPENで使用可能な、グローバル共有の管理番号が得られます。 全て使用中で空きが無い場合、0が得られます。		
パラ メータ	①	<検索開始番号>	数値
	使用可能なグローバル共有の管理番号を検索開始する番号を指定します。 1から15までを指定すると、指定した番号から昇順に、使用可能か検索します。 -1から-15までを指定すると、末尾の番号(15)から逆順に、使用可能か検索します。 (-1を指定すると15から14, 13, ...の順に。-15を指定すると 1 を検索します) 省略すると、-1 を指定したものとしてみなします。		
使用例	NUM = COMMON FREEFILE() COMMON OPEN "127.0.0.1" AS #NUM  空いている管理番号を求めて、グローバル共有をオープンします。		

## 3.8.4 COMMON ERASE

命令			
機 能	グローバル共有に格納された、変数名で指定した内容を削除します。		
書 式	COMMON ERASE #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有に格納されている変数名を指定します。 ここで指定した変数名が、グローバル共有に格納された名前です。		
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に、変数名で指定した名前の情報が無くとも、エラーにはなりません。</li> <li>・変数名を指定するとき、配列の添字や、連想配列のキーを、同時に指定できません。</li> <li>・変数名を指定するとき、文字列では指定できません。</li> </ul>		
使用例	COMMON OPEN "127.0.0.1" AS #1 CNT = 123 COMMON PRINT #1, CNT   ' グローバル共有にCNTを格納します COMMON ERASE #1, CNT   ' グローバル共有に格納されたCNTを削除します COMMON CLOSE #1		

## 3.8.5 COMMON INPUT

命令			
機 能	グローバル共有から、変数名で指定された値を読み取ります。		
書 式	COMMON INPUT #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
パラ メータ	②	<変数名>	変数名
	<p>グローバル共有から入力する変数を指定します。          ここで指定した変数名が、グローバル共有から読み取る名前です。</p> <p>指定できる変数は、数値または文字列、構造体。および配列と連想配列のみです。          連想配列の配列は、指定できません。</p>		
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。</li> <li>・変数名で指定した名前の値が、グローバル共有にない場合、エラーとなります。</li> <li>・読み取り可否を都度確認したい場合、COMMON INPUTR 関数を使います。</li> <li>・配列や連想配列を指定する時、添字を指定すると、指定した添字の値のみ読み取ります。添字を指定しないと、全体が読み取られます。</li> <li>・配列を指定する時、グローバル共有と指定する変数の次元数と要素数が異なる時、グローバル共有の次元数と要素数の情報が優先されます。</li> <li>・一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。 (参照「読み書きを行う変数名の注意とお願い」)</li> </ul>		
使用例	COMMON OPEN "127.0.0.1" AS #1 COMMON INPUT #1, CNT PRINT CNT COMMON CLOSE #1		

## 3.8.6 COMMON INPUTR

関数			
機 能	グローバル共有から、変数名で指定された値を読み取り、成功可否が得られます。		
書 式	<(戻り値)読み取り結果> = COMMON INPUTR( <①管理番号>, <②変数名> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 0：読み取りできた。 0以外：読み取りできなかった。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有から入力する変数を指定します。 ここで指定した変数名が、グローバル共有から読み取る名前です。  指定できる変数は、数値または文字列、構造体。および配列と連想配列のみです。 連想配列の配列は、指定できません。		
備 考	<ul style="list-style-type: none"> <li>・ グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。</li> <li>・ COMMON INPUT 命令との違いは、読み取り可否が得られる事です。</li> <li>・ 配列や連想配列を指定する時、添字を指定すると、指定した添字の値のみ読み取ります。添字を指定しないと、全体が読み取られます。</li> <li>・ 配列を指定する時、グローバル共有と指定する変数の次元数と要素数が異なる時、グローバル共有の次元数と要素数の情報が優先されます。</li> <li>・ 一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。 (参照「読み書きを行う変数名の注意とお願い」)</li> <li>・ 戻り値が負数の場合例を以下に示します。                -1：変数名が見つからなかった                -2：グローバル共有から情報を正しい情報を取得できなかった                -3：通信切断などの事由で、グローバル共有にアクセスできなかった                -4：変数名と異なる型だった             </li> </ul>		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "CNTの読み取り結果="; COMMON INPUTR(1, CNT) PRINT CNT COMMON CLOSE #1</pre>		

## 3.8.7 COMMON READ

関数		
機 能	グローバル共有から、文字列で指定した変数名で値を読み取って変数に格納し、成功可否が得られます。	
書 式	<(戻り値)読み取り結果> = COMMON READ( <①管理番号>, <②変数名>, <③入力変数> )	
戻り値	戻り値	<読み取り結果> 数値
	グローバル共有から値が読み取りできたか、結果が得られます。 0：読み取りできた。 0以外：読み取りできなかった。	
パラメータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<変数名> 文字列
	グローバル共有から入力する変数名を指定します。 ここで指定した変数名が、グローバル共有から読み取る名前です。	
	③	<入力変数> 変数名
	グローバル共有から、変数名で指定した名前の値を読み取り、ここで指定した入力変数に格納します。  指定できる変数は、数値または文字列、構造体。および配列と連想配列のみです。 連想配列の配列は、指定できません。	
備 考	<ul style="list-style-type: none"> <li>・ グローバル共有に格納されている値の型と、入力変数で指定する型は、同じ型でなければなりません。</li> <li>・ COMMON INPUTR 関数との違いは、任意の名前の変数名を指定できることです。</li> <li>・ 配列や連想配列を指定する時、添字を指定すると、指定した添字の値のみ読み取ります。添字を指定しないと、全体が読み取られます。</li> <li>・ 配列を指定する時、グローバル共有と指定する変数の次元数と要素数が異なる時、グローバル共有の次元数と要素数の情報が優先されます。</li> <li>・ 変数名に、「_(アンダーバー)」を接頭辞に付ける事はできません。</li> <li>・ 一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。 (参照「読み書きを行う変数名の注意とお願い」)</li> <li>・ 戻り値が負数の場合例を以下に示します。               <ul style="list-style-type: none"> <li>-1：変数名が見つからなかった</li> <li>-2：グローバル共有から情報を正しい情報を取得できなかった</li> <li>-3：通信切断などの事由で、グローバル共有にアクセスできなかった</li> <li>-4：変数名と異なる型だった</li> </ul> </li> </ul>	
使用例	<pre>PRINT "読み取り結果="; COMMON READ(1, "カウンター", CNT) ' COMMON INPUTR と異なり、任意の変数名を指定して、値を読み取る事ができます。</pre>	

## 3.8.8 COMMON PRINT

命令			
機 能	変数名で指定した値を、グローバル共有に格納します。		
書 式	COMMON PRINT #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	<p>グローバル共有に格納する変数を指定します。          ここで指定した変数名が、グローバル共有に格納する名前です。</p> <p>指定できる変数は、数値または文字列、構造体。および配列と連想配列のみです。          連想配列の配列は、指定できません。</p>		
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、同じ型でなければなりません。</li> <li>配列や連想配列を指定する時、添字を指定すると、指定した添字の値のみ書き換えられます。添字を指定しないと、全体が書き込まれます。</li> <li>添字を指定せずに、配列や連想配列を指定すると、グローバル共有の情報は一旦消去されて書き込まれます。</li> <li>一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。          (参照「読み書きを行う変数名の注意とお願い」)</li> </ul>		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 CNT = 123 COMMON PRINT #1, CNT COMMON CLOSE #1</pre>		



## 3.8.9 COMMON WRITE

命令		
機 能	文字列で指定した変数名と指定した値を、グローバル共有に格納します。	
書 式	COMMON WRITE #<①管理番号>, <②変数名>, <③出力変数>	
パラ メータ	①	<div>&lt;管理番号&gt;</div> <div>数値</div> <p>COMMON OPENでオープンした管理番号を指定します。</p>
	②	<div>&lt;変数名&gt;</div> <div>文字列</div> <p>グローバル共有に格納する変数名を文字列で指定します。 ここで指定した変数名が、グローバル共有に格納する名前です。</p>
	③	<div>&lt;出力変数&gt;</div> <div>変数名</div> <p>グローバル共有に格納する変数を指定します。 ここで指定した変数の値が、グローバル共有に格納されます。</p> <p>指定できる変数は、数値または文字列、構造体。および配列と連想配列のみです。 連想配列の配列は、指定できません。</p>
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、出力変数は同じ型でなければなりません。</li> <li>COMMON PRINT 命令との違いは、任意の名前の変数名を指定できることです。</li> <li>配列や連想配列を指定する時、添字を指定すると、指定した添字の値のみ書き換えられます。添字を指定しないと、全体が書き込まれます。</li> <li>添字を指定せずに、配列や連想配列を指定すると、グローバル共有の情報は一旦消去されて書き込まれます。</li> <li>変数名に、「_(アンダーバー)」を接頭辞に付ける事はできません。</li> <li>一文字目が「I」で、二文字目が「A」から「Z」を接頭辞に持つ変数名を、グローバル共有の読み書き対象とするのは、避けてください。 (参照「読み書きを行う変数名の注意とお願い」)</li> </ul>	
使用例	<pre>CNT = 123 COMMON WRITE #1, "カウンター", CNT ' COMMON PRINT と異なり、任意の変数名を指定して、値を書き込めます。</pre>	

## 3.8.10 COMMON LOCK

関数		
機 能	グローバル共有に対して、指定した名前でロックを取得します。	
書 式	<(戻り値)ロック可否> = COMMON LOCK( <①管理番号>, <②ロック名> [, <③有効期限秒> ] )	
戻り値	戻り値	<ロック可否> 数値
	ロックが取得できたか、結果が得られます。 1 : ロック取得できた。 0 : ロック取得できなかった。	
パラメータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<ロック名> 文字列
	ロックを取得したい名前を指定します。	
	③	<有効期限秒> 数値
	ロックが有効である期間を秒で指定します。 指定した秒数を超えると、自動的にロックは解除されます。 省略時、10秒が採用されます。	
備 考	<ul style="list-style-type: none"> <li>COMMON LOCKとCOMMON UNLOCKを併用する事で、グローバル共有に格納された値を、読み書きする間、他のプログラムが読み書きできないように、排他処理を作る事ができます。</li> <li>ロックの間に、グローバル共有に格納された値を読み書きする対象に対して、同じロック名を指定してください。異なるロック名を指定すると、ロックの意味がありません。</li> <li>ロック名に、変数名と同じ名前を指定しないでください。</li> </ul>	
使用例	<pre>DO WHILE TRUE   IF COMMON LOCK(1, "TEST", 5) &gt; 0 THEN     ' ロックを取得できた     COMMON INPUT #1, CNT     CNT = CNT + 1     COMMON PRINT #1, CNT     ' ロックを解除する     F = COMMON UNLOCK(1, "TEST")   END IF LOOP</pre> <p>ロックを取得できたら、グローバル共有からCNTの値を取り出してカウントアップして格納する事例です。</p>	

## 3.8.11 COMMON UNLOCK

関数		
機 能	グローバル共有に対して、指定した名前でロックを解除します。	
書 式	<(戻り値)アンロック可否> = COMMON UNLOCK( <①管理番号>, <②ロック名> )	
戻り値	戻り値	<アンロック可否> 数値
	ロックが解除できたか、結果が得られます。 2：有効期限を過ぎたが、ロック解除できた。 1：有効期限内に、ロック解除できた。 0：ロック解除できなかった。 -1：ロック名が見つからなかった。(ロック名の指定ミスか有効期限を過ぎたか)	
パラ メータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<ロック名> 文字列
	ロックを解除したい名前を指定します。 COMMON LOCK で指定したロック名を指定してください。	
備 考	<ul style="list-style-type: none"> <li>・COMMON LOCKとCOMMON UNLOCKを併用する事で、グローバル共有に格納された値を、読み書きする間、他のプログラムが読み書きできないように、排他処理を作る事ができます。</li> <li>・ロックの間に、グローバル共有に格納された値を読み書きする対象に対して、同じロック名を指定してください。異なるロック名を指定すると、ロックの意味がありません。</li> <li>・ロック名に、変数名と同じ名前を指定しないでください。</li> </ul>	
使用例	COMMON LOCK の使用例を参照ください。	

## 3.8.12 COMMON NOTIFY

関数		
機 能	グローバル共有を介して、トピック名にメッセージを通知します。	
書 式	<(戻り値)受信者数> = COMMON NOTIFY( <①管理番号>, <②トピック名>, <③メッセージ> )	
戻り値	戻り値	<受信者数> 数値
	メッセージを受信しようとする相手の数が得られます。	
パラ メータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<トピック名> 文字列
	受信者がメッセージを選択的に受け取る為の、一種のキーワードを文字列で指定します。	
	②	<メッセージ> 文字列
	受信者に向けて送るメッセージを文字列で指定します。	
注 意	<ul style="list-style-type: none"> <li>受信者数は、すなわち ON COMMON CALL で、本コマンドのメッセージを受け取ろうとしている数です。</li> <li>このコマンド発行時に、相手側にメッセージが受け取れているか否かは、通信状態に依存するので、確定できません。</li> </ul>	
使用例	<pre>CNT = COMMON NOTIFY(1, "TEST", "Hello AJAN") PRINT "メッセージを受信したい人の数="; CNT</pre>	

## 3.8.13 ON COMMON CALL

命令			
機能	グローバル共有を介して、指定したトピック名に合致するメッセージを受信するサブルーチンを定義します。		
書式	ON COMMON ( #<①管理番号>, <②トピック名> ) CALL <③サブルーチン名>		
パラメータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<トピック名>	文字列
	<p>メッセージを選択的に受け取る為の、一種のキーワードを文字列で指定します。          ここで指定したトピック名が合致するメッセージのみ、受信できます。          トピック名を指定する時、「*」文字を任意の文字列を意味するワイルドカード文字として指定できます。</p>		
	③	<サブルーチン名>	サブルーチン名
	<p>サブルーチン名を指定します。          サブルーチンは、以下の定義に従います。          サブルーチンで受け取る、トピック名とメッセージは、COMMON NOTIFY で通知された内容と同じです。</p>		
	<pre>SUB サブルーチン名(&lt;管理番号&gt;, &lt;トピック名&gt;, &lt;メッセージ&gt;)   処理内容 END SUB</pre>		
備考	<ul style="list-style-type: none"> <li>・同じ管理番号で本コマンドを呼び出すと、前回の呼び出しの設定が上書きされます。</li> <li>・ON COMMON CALL による割り込み処理中に、別から COMMON NOTIFY が呼ばれた時、現在の割り込み処理が完了してから、続けて 割り込み処理が呼ばれます。</li> <li>・ON COMMON CALL による割り込み処理中に、ON COMMON CALL または COMMON CLOSE を呼び出すと、割り込み処理は中断されます。 (Ver. 1.00より前は、割り込み処理が完了するまでブロッキングされて、COMMON CLOSE などの呼び出しが待たされ続けていました)</li> </ul>		
注意	<ul style="list-style-type: none"> <li>・redis-sentinel デーモンが稼働中の場合(端末から「systemctl status redis-sentinel」で「active(running)」の時に稼働しています)、デーモンは「__sentinel__:hello」をトピック名に定期的にメッセージが発信されます。 COMMON NOTIFY による割り込みメッセージと混同しないよう、ご注意ください。</li> </ul>		
使用例1	<pre>ON COMMON(#1, "TEST") CALL CB_TEST COMMON ON #1  SUB CB_TEST(NUM%, TOPIC\$, MSG\$)   PRINT "メッセージ受信="; TOPIC\$; ", "; MSG\$ END SUB</pre> <p>’ “TEST”というトピック名で指定したメッセージを受け取ると、CB_TEST ルーチンが呼び出されます。</p>		
使用例2	<pre>ON COMMON(#1, "*") CALL CB_TEST ...</pre> <p>’ 使用例1とは異なり、どんなトピック名でもメッセージを受け取ると、CB_TESTルーチンが呼び出されます。</p>		

## 3.8.14 COMMON ON / OFF

命令			
機 能	COMMONメッセージ受信の許可、禁止を指定します。 メッセージ受信の条件、サブルーチンは、ON COMMON CALL で定義します。		
書 式1	COMMON ON #<①管理番号>  メッセージ受信を許可します。これ以降、トピック名に合致したメッセージを受信すると、サブルーチンが呼び出されます。		
書 式2	COMMON OFF #<①管理番号>  メッセージ受信を禁止します。これ以降、トピック名に合致したメッセージを受信しても、サブルーチンは呼び出されません。		
パラ メータ	①	<管理番号>	数値
COMMON OPENでオープンした管理番号を指定します。			
使用例	ON COMMON CALLの使用例を参照ください。		

## 3.8.15 COMMON CDIM

関数			
機 能	グローバル共有にある、変数名で指定された配列の次元数を返します。		
書 式	<(戻り値)次元数> = COMMON CDIM( <①管理番号>, <②配列変数名> )		
戻り値	戻り値	<次元数>	数値
	グローバル共有にある、配列の次元数を返します。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<配列変数名>	変数名
備 考	グローバル共有にある、配列の変数名を指定します。		
	配列でない場合、エラーとなります。		
備 考	・ 変数名を指定するとき、配列の添字を、同時に指定できません。		
	・ 変数名を指定するとき、文字列では指定できません。		
使用例	LIST ARY PRINT "次元数="; COMMON CDIM(1, ARY)		

## 3.8.16 COMMON GET\_DICT\_KEYS\$

関数		
機 能	グローバル共有にある、変数名で指定された連想配列のキー一覧を文字列配列として得ます。	
書 式	〈(戻り値)キー一覧配列〉 = COMMON GET_DICT_KEYS\$ ( 〈①管理番号〉, 〈②連想配列変数名〉 )	
戻り値	戻り値	配列
	グローバル共有にある、連想配列のキー一覧を、1次元の文字列配列にして返します。	
パラ メータ	①	数値
	〈管理番号〉 COMMON OPENでオープンした管理番号を指定します。	
	②	変数名
	〈連想配列変数名〉 グローバル共有にある、連想配列の変数名を指定します。 連想配列でない場合、エラーとなります。	
注 意	<ul style="list-style-type: none"> <li>空の連想配列か否かを判定するには、COMMON LDICT 関数で、0(空)か0以外かで判断してください。</li> <li>空の連想配列に対して、COMMON GET_DICT_KEYS\$ を呼び出した結果は、保証されません。</li> </ul>	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>変数名を指定するとき、文字列では指定できません。得られる文字列配列は、順序保障されません。</li> </ul>	
使用例	<pre> DICT ARY LIST KEYS\$ KEYS\$ = COMMON GET_DICT_KEYS\$(1, ARY) PRINT "連想配列のキー一覧" FOR I=0 TO UBOUND(KEYS\$)   PRINT KEYS\$(I) NEXT I </pre>	



## 3.8.17 COMMON HAS\_DICT\_KEY

関数		
機 能	グローバル共有にある、変数名で指定した連想配列に対して、指定したキーが存在するかどうかを得ます。	
書 式	〈(戻り値)キーの有無〉 = COMMON HAS_DICT_KEY ( 〈①管理番号〉, 〈②連想配列変数名〉, "〈③キー名〉" )	
戻り値	戻り値	真偽値
	グローバル共有にある、連想配列のキー一覧に対して、指定したキー名が存在すれば TRUE、存在しなければ FALSE を返します。	
パラ メータ	①	数値
	〈管理番号〉 COMMON OPEN でオープンした管理番号を指定します。	
	②	変数名
	〈連想配列変数名〉 グローバル共有にある、連想配列の変数名を指定します。 連想配列でない場合、エラーとなります。	
	③	文字列
	〈キー名〉 連想配列に対して探索を行いたいキー名を、文字列形式で与えます。	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>変数名を指定するとき、文字列では指定できません。</li> </ul>	
使用例	DICT ARY PRINT "連想配列に' TEST' キーはあるか?="; COMMON HAS_DICT_KEY (1, ARY, "TEST")	

## 3.8.18 COMMON LDICT

関数		
機 能	グローバル共有にある、変数名で指定された連想配列に対して登録されているキーの総数を取得します。	
書 式	<(戻り値)キーの総数> = COMMON LDICT ( <①管理番号>, <②連想配列変数名> )	
戻り値	戻り値	<キーの総数> 数値
	グローバル共有にある、連想配列に対して登録されているキーの総数を取得します。	
パラメータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<連想配列変数名> 変数名
	グローバル共有にある、連想配列の変数名を指定します。 連想配列でない場合、エラーとなります。	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>変数名を指定するとき、文字列では指定できません。</li> </ul>	
使用例	DICT ARY PRINT "連想配列のキー総数="; COMMON LDICT(1, ARY)	

## 3.8.19 COMMON LDIM

関数		
機 能	グローバル共有にある、変数名の配列の要素数を返します。	
書 式	<(戻り値)配列の要素数> = COMMON LDIM ( <①管理番号>, <②配列変数名>[ , <③次元> ] )	
戻り値	戻り値	<配列の要素数> 数値
	グローバル共有にある、配列の要素数を返します。	
パラメータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<配列変数名> 変数名
	グローバル共有にある、配列の変数名を指定します。 配列でない場合、エラーとなります。	
③	<次元>	数値
	1から始まる、要素数を返す次元を指定します。 指定しない場合、全要素数を返します。	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、配列の添字を、同時に指定できません。</li> <li>変数名を指定するとき、文字列では指定できません。</li> <li>存在しない次元を指定するとエラーが返ります。</li> <li>指定した次元の添字最大値を得たい場合は「COMMON UBOUND」を使用してください。</li> </ul>	
使用例	LIST ARY PRINT "1次元目の要素数="; COMMON LDIM(1, ARY, 1)	

## 3. 8. 20 COMMON UBOUND

関数		
機 能	グローバル共有にある、変数名で指定された配列の指定した次元の添字最大値を得ます。	
書 式	<(戻り値) 添字最大値> = COMMON UBOUND ( <①管理番号>, <②配列変数名> [, <③次元> ] )	
戻り値	戻り値	<添字最大値> 数値 グローバル共有にある、配列の指定した次元の添字最大値が返ります。
パラメータ	①	<管理番号> 数値 COMMON OPENでオープンした管理番号を指定します。
	②	<配列変数名> 変数名 グローバル共有にある、配列の変数名を指定します。 配列でない場合、エラーとなります。
	③	<次元> 数値 1から始まる、添字最大値を返す次元を指定します。 省略すると1が指定されたものとします。
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、配列の添字を、同時に指定できません。</li> <li>変数名を指定するとき、文字列では指定できません。</li> <li>配列変数でない、または存在しない次元を指定するとエラーが返ります。</li> <li>指定した次元の要素数を得たい場合は「COMMON LDIM」を使用してください。</li> </ul>	
使用例	LIST ARY PRINT "1次元目の添字最大値="; COMMON UBOUND (1, ARY, 1) PRINT "2次元目の添字最大値="; COMMON UBOUND (1, ARY, 2)	

## 3.8.21 COMMON VARTYPE

関数			
機 能	グローバル共有にある、変数名で指定した値の型を調べます。		
書 式	<(戻り値)型の値> = COMMON VARTYPE( <①管理番号>, <②変数名> )		
戻り値	戻り値	<型の値>	数値
	グローバル共有にある、値の型情報が、以下の組み合わせ (OR) で得られます。 例えば、倍精度実数型の配列の場合、&H10(倍精度実数型) OR &H8000000(配列) = &H8000010 が得られます。		
	返値	解説	
	&h01	文字列型	
	&h02	単精度整数型	
	&h04	倍精度整数型	
	&h08	単精度実数型	
	&h10	倍精度実数型	
	&h20	論理型	
	&h40	列挙値型	
	&h80	構造体型	
	&h1000	日付時刻型	
	&h4000000	両端キュー	
	&h8000000	配列	
	&h20000000	連想配列	
&h40000000	可変長配列		
もしグローバル共有に、指定した変数名の値が存在しない場合や、情報が取得できなかった場合、負数の値が返ります。			
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
グローバル共有にある、値の変数名を指定します。			
備 考	<ul style="list-style-type: none"><li>変数名を指定するとき、配列の添字や、連想配列のキーを、同時に指定できません。</li><li>変数名を指定するとき、文字列では指定できません。</li><li>戻り値が負数の場合例を以下に示します。<ul style="list-style-type: none"><li>-1：変数名が見つからなかった</li><li>-2：グローバル共有から情報を正しい情報を取得できなかった</li><li>-3：通信切断などの事由で、グローバル共有にアクセスできなかった</li></ul></li><li>戻り値：&amp;h4000000 の両端キューは、COMMON DEQUE PUSH などの両端キューに値を追加した際に有効です。</li></ul>		
使用例	RET = COMMON VARTYPE(1, ARY) IF RET < 0 THEN PRINT "グローバル共有に、ARYの値は存在しません" ELSE PRINT "グローバル共有の、ARYの型情報="; ARY END IF		

## 3.8.22 COMMON DEQUE PUSH

命令			
機 能	変数名で指定した値を、グローバル共有の両端キューの末尾に追加します。		
書 式	COMMON DEQUE PUSH #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	<p>グローバル共有の両端キューの末尾に追加する変数を指定します。          ここで指定した変数名が、グローバル共有の両端キューの名前です。</p> <p>指定できる変数は、単一の数値または文字列、構造体のみです。          配列、連想配列などは、指定できません。</p>		
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、同じ型でなければなりません。</li> <li>例えば、COMMON PRINT 命令で、同じ名前の変数名で値を設定済みの時、両端キューは追加できません。COMMON ERASE 命令で、一旦削除してください。</li> </ul>		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 LV = 123 COMMON DEQUE PUSH #1, LV COMMON CLOSE #1</pre>		

## 3.8.23 COMMON DEQUE LPUSH

命令			
機 能	変数名で指定した値を、グローバル共有の両端キューの先頭に追加します。		
書 式	COMMON DEQUE LPUSH #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
パラ メータ	②	<変数名>	変数名
	<p>グローバル共有の両端キューの先頭に追加する変数を指定します。          ここで指定した変数名が、グローバル共有の両端キューの名前です。</p> <p>指定できる変数は、単一の数値または文字列、構造体のみです。          配列、連想配列などは、指定できません。</p>		
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、同じ型でなければなりません。</li> <li>例えば、COMMON PRINT 命令で、同じ名前の変数名で値を設定済みの時、両端キューは追加できません。COMMON ERASE 命令で、一旦削除してください。</li> </ul>		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 LV = 123 COMMON DEQUE LPUSH #1, LV COMMON CLOSE #1</pre>		

## 3.8.24 COMMON DEQUE POP

関数			
機 能	グローバル共有から、変数名で指定された両端キューの末尾より値を取り出します。		
書 式	<(戻り値)読み取り結果> = COMMON DEQUE POP ( <①管理番号>, <②変数名> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 1 : 読み取りできた。 0 : 両端キューが空だったので読み取れなかった。		
パラメータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有の両端キューの末尾から取り出す変数を指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。		
備 考	・ グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON DEQUE POP(1, LV) PRINT LV COMMON CLOSE #1</pre>		

## 3.8.25 COMMON DEQUE LPOP

関数			
機 能	グローバル共有から、変数名で指定された両端キューの先頭より値を取り出します。		
書 式	<(戻り値)読み取り結果> = COMMON DEQUE LPOP( <①管理番号>, <②変数名> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 1 : 読み取りできた。 0 : 両端キューが空だったので読み取れなかった。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有の両端キューの先頭から取り出す変数を指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。		
備 考	・ グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON DEQUE LPOP(1, LV) PRINT LV COMMON CLOSE #1</pre>		



## 3.8.26 COMMON DEQUE BACK

関数			
機 能	グローバル共有から、変数名で指定された両端キューの末尾より値を参照します。		
書 式	<(戻り値)読み取り結果> = COMMON DEQUE BACK( <①管理番号>, <②変数名> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 1 : 読み取りできた。 0 : 両端キューが空だったので読み取れなかった。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有の両端キューの末尾から取り出す変数を指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。		
備 考	・ グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON DEQUE BACK(1, LV) PRINT LV COMMON CLOSE #1</pre>		

## 3.8.27 COMMON DEQUE FRONT

関数			
機 能	グローバル共有から、変数名で指定された両端キューの先頭より値を参照します。		
書 式	<(戻り値)読み取り結果> = COMMON DEQUE FRONT( <①管理番号>, <②変数名> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 1 : 読み取りできた。 0 : 両端キューが空だったので読み取れなかった。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有の両端キューの先頭から取り出す変数を指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。		
備 考	・ グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON DEQUE FRONT(1, LV) PRINT LV COMMON CLOSE #1</pre>		

## 3.8.28 COMMON DEQUE LEN

関数			
機 能	グローバル共有から、変数名で指定された両端キューの個数を得ます。		
書 式	<(戻り値)両端キューの個数> = COMMON DEQUE LEN( <①管理番号>, <②変数名> )		
戻り値	戻り値	<両端キューの個数>	数値
	グローバル共有の両端キューの個数を得ます。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	変数名
	グローバル共有にある、両端キューの変数名を指定します。		
	両端キューでない場合、エラーとなります。		
備 考	・グローバル共有に格納されている値の型と、変数名で指定する型は、同じ型でなければなりません。		
使用例	COMMON OPEN "127.0.0.1" AS #1 PRINT "両端キューLVの個数="; COMMON DEQUE LEN(1, LV) COMMON CLOSE #1		

3. 8. 29 COMMON VLIST\$

関数			
機 能	グローバル共有の変数名のリストを、文字列配列で得ます。		
書 式	<(戻り値) 変数名の配列> = COMMON VLIST\$( <①管理番号> )		
戻り値	戻り値	<変数名の配列>	文字列
	グローバル共有の変数名のリストを、文字列配列で得られます。		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
使用例	<pre>' AJANの変数名を使った書き込み CNT% = 123 COMMON PRINT #1, CNT% LC&amp; = 456 COMMON PRINT #1, LC&amp; ' 任意の名前を使った書き込み S\$ = "hello" COMMON PRINT #1, S\$ COMMON WRITE #1, "TEST", "AJAN"  PRINT COMMON VLIST\$(1) ' グローバル共有の変数名のリストが配列で得られます。 ' [ TEST\$, LC&amp;, S\$, CNT% ] のような結果が得られます。</pre>		

## 3.8.30 COMMON SWAPDB

命令			
機 能	グローバル共有に対して、指定したデータベースの番号を入れ替えます。		
書 式	COMMON SWAPDB #<①管理番号>, <②DB番号1>, <②DB番号2>		
パラ メータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	①	<DB番号1>, <DB番号2>	数値
	交換したいデータベースの番号を指定します。		
備 考	<ul style="list-style-type: none"> <li>データベースの番号を入れ替えると、データベースを参照していたアプリは、入れ替えられた番号のデータベースを参照する事になります。</li> </ul>		
使用例	COMMON OPEN "127.0.0.1" AS #1 ' inta という名前のユーザを追加 COMMON ADDUSER #1, "inta", "paso" ' 追加したユーザの情報を記録します COMMON SAVEACL #1		

## 3.8.31 COMMON2 ERASE

命令			
機 能	グローバル共有に格納された、文字列で指定した変数名の内容を削除します。		
書 式	COMMON2 ERASE #<①管理番号>, <②変数名>		
パラ メータ	①	<管理番号>	数値
	「COMMON OPEN」でオープンした管理番号を指定します。		
	②	<変数名>	文字列
	グローバル共有に格納されている変数名を、文字列で指定します。 ここで指定した変数名が、グローバル共有に格納された値の名前です。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
備 考	・ グローバル共有に、変数名で指定した名前の情報が無くとも、エラーにはなりません。 ・ 2021/3から、変数名は大文字扱いとします。		
使用例	COMMON OPEN "127.0.0.1" AS #1 COMMON2 PRINT #1, "CNT", 123 ' グローバル共有にCNT名で値を格納します COMMON2 ERASE #1, "CNT" ' グローバル共有に格納されたCNTを削除します COMMON CLOSE #1		

## 3.8.32 COMMON2 CDIM

関数			
機 能	グローバル共有にある、文字列で指定した変数名の、中の配列の次元数を返します。		
書 式	<(戻り値)次元数> = COMMON2 CDIM(<①管理番号>, <②配列変数名>)		
戻り値	戻り値	<次元数>	数値
	グローバル共有にある、配列の次元数を返します。		
パラ メータ	①	<管理番号>	数値
	「COMMON OPEN」でオープンした管理番号を指定します。		
	②	<配列変数名>	文字列
	グローバル共有にある、配列の変数名を、文字列で指定します。		
	指定した変数名の内容が、配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、配列の添字を、同時に指定できません。</li> <li>2021/3から、配列変数名は大文字扱いとします。</li> </ul>		
使用例	PRINT "次元数="; COMMON2 CDIM(1, "ARY") ' グローバル共有に「ARY」という配列変数があり、その変数の次元数を得ます。		

## 3. 8. 33 COMMON2 GET\_DICT\_KEYS\$

関数			
機 能	グローバル共有にある、文字列で指定した変数名の、中の連想配列のキー一覧を文字列配列として得ます。		
書 式	<(戻り値)キー一覧配列> = COMMON2 GET_DICT_KEYS\$(<①管理番号>, <②連想配列変数名>)		
戻り値	戻り値	<キー一覧配列>	配列
	グローバル共有にある、連想配列のキー一覧を、1次元の文字列配列にして返します。		
パラメータ	①	<管理番号>	数値
	「COMMON OPEN」でオープンした管理番号を指定します。		
	②	<連想配列変数名>	文字列
	グローバル共有にある、連想配列の変数名を、文字列で指定します。 指定した変数名の内容が、連想配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
注 意	<ul style="list-style-type: none"> <li>空の連想配列か否かを判定するには、「COMMON LDICT」で、0(空)か0以外かで判断してください。</li> <li>空の連想配列に対して、COMMON2 GET_DICT_KEYS\$ を呼び出した結果は、保証されません。</li> </ul>		
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>得られる文字列配列は、順序保障されません。2021/3から、連想配列変数名は大文字扱いとします。</li> </ul>		
使用例	<pre> LIST KEYS\$ KEYS\$ = COMMON2 GET_DICT_KEYS\$(1, "ARY") PRINT "連想配列のキー一覧" FOR I=0 TO UBOUND(KEYS\$)   PRINT KEYS\$(I) NEXT I ' グローバル共有に「ARY」という連想配列があり、その変数のキー一覧を文字列配列として得ます。 </pre>		



## 3. 8. 34 COMMON2 HAS\_DICT\_KEY

関数		
機 能	グローバル共有にある、文字列で指定した変数名の、中の連想配列に対して、指定したキーが存在するか否かを得ます。	
書 式	〈(戻り値)キーの有無〉 = COMMON2 HAS_DICT_KEY (〈①管理番号〉, 〈②連想配列変数名〉, "〈③キー名〉")	
戻り値	戻り値	真偽値
	グローバル共有にある、連想配列のキー一覧に対して、指定したキー名が存在すればTRUE、存在しなければFALSEを返します。	
パラメータ	①	数値
	「COMMON OPEN」でオープンした管理番号を指定します。	
	②	文字列
	グローバル共有にある、連想配列の変数名を、文字列で指定します。 指定した変数名の内容が、連想配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。	
	③	文字列
	連想配列に対して探索を行いたいキー名を、文字列形式で与えます。	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>2021/3から、連想配列変数名は大文字扱いとします。</li> </ul>	
使用例	PRINT "連想配列に' TEST' キーはあるか?="; COMMON2 HAS_DICT_KEY(1, "ARY", "TEST") ' グローバル共有に「ARY」という連想配列があり、その変数に対して、「TEST」というキーが有るか否か判定します。	

## 3. 8. 35 COMMON2 LDICT

関数			
機 能	グローバル共有にある、文字列で指定した変数名の、中の連想配列に対して、登録されているキーの総数を取得します。		
書 式	<(戻り値)キーの総数> = COMMON2 LDICT(<①管理番号>, <②連想配列変数名>)		
戻り値	戻り値	<キーの総数>	数値
	グローバル共有にある、連想配列に対して登録されているキーの総数を取得します。		
パラメータ	①	<管理番号>	数値
	「COMMON OPEN」でオープンした管理番号を指定します。		
	②	<連想配列変数名>	文字列
	グローバル共有にある、連想配列の変数名を、文字列で指定します。 指定した変数名の内容が、連想配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、連想配列のキーを、同時に指定できません。</li> <li>2021/3から、連想配列変数名は大文字扱いとします。</li> </ul>		
使用例	PRINT "連想配列のキー総数="; COMMON2 LDICT(1, "ARY") ' グローバル共有に「ARY」という連想配列があり、その変数のキー数を得ます。		

## 3. 8. 36 COMMON2 LDIM

関数			
機 能	グローバル共有にある、文字列で指定した変数名の、中の配列の要素数を返します。		
書 式	<(戻り値)配列の要素数> = COMMON2 LDIM(<①管理番号>, <②配列変数名>[, <③次元>])		
戻り値	戻り値	<配列の要素数>	数値
	グローバル共有にある、配列の要素数を返します。		
パラメータ	①	<管理番号>	数値
	「COMMON OPEN」でオープンした管理番号を指定します。		
	②	<配列変数名>	文字列
	グローバル共有にある、配列の変数名を、文字列で指定します。 指定した変数名の内容が、配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
	③	<次元>	数値
	1から始まる、要素数を返す次元を指定します。 指定しない場合、全要素数を返します。		
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、配列の添字を、同時に指定できません。</li> <li>存在しない次元を指定するとエラーが返ります。</li> <li>指定した次元の添字最大値を得たい場合は「COMMON UBOUND」を使用してください。</li> <li>2021/3から、配列変数名は大文字扱いとします。</li> </ul>		
使用例	PRINT "1次元目の要素数="; COMMON2 LDIM(1, "ARY", 1) ' グローバル共有に「ARY」という配列変数があり、その変数の1次元目の要素数を得ます。		

## 3. 8. 37 COMMON2 UBOUND

関数		
機 能	グローバル共有にある、文字列で指定した変数名の、中の配列の指定した次元の添字最大値を得ます。	
書 式	<(戻り値)添字最大値> = COMMON2 UBOUND(<①管理番号>, <②配列変数名> [, <③次元>])	
戻り値	戻り値	<添字最大値> 数値
	グローバル共有にある、配列の指定した次元の添字最大値が返ります。	
パラ メータ	①	<管理番号> 数値
	「COMMON OPEN」でオープンした管理番号を指定します。	
	②	<配列変数名> 文字列
	グローバル共有にある、配列の変数名を、文字列で指定します。 指定した変数名の内容が、配列でない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。	
	③	<次元> 数値
	1から始まる、添字最大値を返す次元を指定します。 省略すると1が指定されたものとします。	
備 考	<ul style="list-style-type: none"> <li>変数名を指定するとき、配列の添字を、同時に指定できません。</li> <li>配列変数でない、または存在しない次元を指定するとエラーが返ります。</li> <li>指定した次元の要素数を得たい場合は「COMMON LDIM」を使用してください。</li> <li>2021/3から、配列変数名は大文字扱いとします。</li> </ul>	
使用例	PRINT "1次元目の添字最大値="; COMMON2 UBOUND(1, "ARY", 1) PRINT "2次元目の添字最大値="; COMMON2 UBOUND(1, "ARY", 2) ' グローバル共有に「ARY」という配列変数があり、その変数の1次元目および2次元目の添字最大値を得ます。	

### 3. 8. 38 COMMON2 VARTYPE

関数				
機 能	グローバル共有にある、文字列で指定した変数名の、中の値の型を調べます。			
書 式	<(戻り値)型の値> = COMMON VARTYPE( <①管理番号>, <②変数名> )			
戻り値	戻り値	<型の値>		数値
	グローバル共有にある、値の型情報が、以下の組み合わせ (OR) で得られます。 例えば、倍精度実数型の配列の場合、&H10(倍精度実数型) OR &H8000000(配列) = &H8000010 が得られます。			
	返値	解説		
	&h01	文字列型		
	&h02	単精度整数型		
	&h04	倍精度整数型		
	&h08	単精度実数型		
	&h10	倍精度実数型		
	&h20	論理型		
	&h40	列挙値型		
	&h80	構造体型		
	&h1000	日付時刻型		
	&h4000000	両端キュー		
	&h8000000	配列		
	&h20000000	連想配列		
&h40000000	可変長配列			
もしグローバル共有に、指定した変数名の値が存在しない場合、負数の値が返ります。				
パラ メータ	①	<管理番号>		数値
	「COMMON OPEN」でオープンした管理番号を指定します。			
	②	<変数名>		変数名
グローバル共有にある、値の変数名を、文字列で指定します。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。				
備 考	・ 変数名を指定するとき、配列の添字や、連想配列のキーを、同時に指定できません。 ・ 2021/3から、変数名は大文字扱いとします。 ・ 戻り値：&h4000000 の両端キューは、COMMON DEQUE PUSH などの両端キューに値を追加した際に有効です。			
使用例	RET = COMMON2 VARTYPE(1, "ARY") IF RET < 0 THEN PRINT "グローバル共有に、ARYの値は存在しません" ELSE PRINT "グローバル共有の、ARYの型情報="; ARY END IF ' グローバル共有に、「ARY」という値があるか否かを調べます。			

## 3. 8. 39 COMMON2 DEQUE PUSH

命令		
機 能	文字列で指定した変数名と指定した値を、グローバル共有の両端キューの末尾に追加します。	
書 式	COMMON2 DEQUE PUSH #<①管理番号>, <②変数名>, <③出力変数>	
パラ メータ	①	<div>&lt;管理番号&gt;</div> <div>数値</div> <div>「COMMON OPEN」でオープンした管理番号を指定します。</div>
	②	<div>&lt;変数名&gt;</div> <div>文字列</div> <div>グローバル共有の両端キューの末尾に追加する変数名を、文字列で指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。</div>
	③	<div>&lt;出力変数&gt;</div> <div>変数名</div> <div>グローバル共有の両端キューの末尾に追加する変数を指定します。 ここで指定した変数の値が、グローバル共有の両端キューに追加格納されます。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。</div>
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、同じ型でなければなりません。</li> <li>例えば、「COMMON WRITE」で、同じ名前の変数名で値を設定済みの時、両端キューは追加できません。「COMMON ERASE」で、一旦削除してください。</li> <li>2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	COMMON OPEN "127.0.0.1" AS #1 COMMON2 DEQUE PUSH #1, "LV", 123 COMMON CLOSE #1 ' グローバル共有の、「LV」という両端キューに対して、123の値を末尾追加します。	

## 3. 8. 40 COMMON2 DEQUE LPUSH

命令		
機 能	文字列で指定した変数名と指定した値を、グローバル共有の両端キューの先頭に追加します。	
書 式	COMMON2 DEQUE LPUSH #<①管理番号>, <②変数名>, <③出力変数>	
パラ メータ	①	<div>&lt;管理番号&gt;</div> <div>数値</div> 「COMMON OPEN」でオープンした管理番号を指定します。
	②	<div>&lt;変数名&gt;</div> <div>文字列</div> グローバル共有の両端キューの先頭に追加する変数名を、文字列で指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。
	③	<div>&lt;出力変数&gt;</div> <div>変数名</div> グローバル共有の両端キューの先頭に追加する変数を指定します。 ここで指定した変数の値が、グローバル共有の両端キューに追加格納されます。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。
備 考	<ul style="list-style-type: none"> <li>変数名で指定した名前が、既にグローバル共有に格納されているとき、同じ型でなければなりません。</li> <li>例えば、「COMMON WRITE」で、同じ名前の変数名で値を設定済みの時、両端キューは追加できません。「COMMON ERASE」で、一旦削除してください。</li> <li>2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	COMMON OPEN "127.0.0.1" AS #1 COMMON2 DEQUE LPUSH #1, "LV", 123 COMMON CLOSE #1 ' グローバル共有の「lv」という両端キューに対して、123の値を先頭追加します。	

## 3.8.41 COMMON2 DEQUE POP

関数		
機 能	グローバル共有から、文字列で指定した変数名で両端キューの末尾より値を取り出します。	
書 式	<(戻り値)読み取り結果> = COMMON2 DEQUE POP(<①管理番号>, <②変数名>, <③入力変数>)	
戻り値	戻り値	<読み取り結果> 数値
	<p>グローバル共有から値が読み取りできたか、結果が得られます。</p> <p>1 : 読み取りできた。</p> <p>0 : 両端キューが空だったので読み取れなかった。</p>	
パラメータ	①	<管理番号> 数値
	「COMMON OPEN」でオープンした管理番号を指定します。	
	②	<変数名> 文字列
	<p>グローバル共有の両端キューの末尾から取り出す変数名を、文字列で指定します。</p> <p>ここで指定した変数名が、グローバル共有の両端キューの名前です。</p> <p>変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。</p>	
	③	<入力変数> 変数名
	<p>グローバル共有の両端キューの末尾から取り出す変数を指定します。</p> <p>ここで指定した変数が、グローバル共有の両端キューから取り出して格納されます。</p>	
	<p>指定できる変数は、単一の数値または文字列、構造体のみです。</p> <p>配列、連想配列などは、指定できません。</p>	
備 考	<ul style="list-style-type: none"> <li>グローバル共有に格納されている値の型と、入力変数で指定する型は、同じ型でなければなりません。</li> <li>2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON2 DEQUE POP(1, "LV", V) PRINT V COMMON CLOSE #1</pre> <p>’ グローバル共有に「LV」という両端キューがあり、末尾のキュー値をV変数に取り出します。</p>	

## 3.8.42 COMMON2 DEQUE LPOP

関数		
機 能	グローバル共有から、文字列で指定した変数名で両端キューの先頭より値を取り出します。	
書 式	<(戻り値)読み取り結果> = COMMON2 DEQUE LPOP(<①管理番号>, <②変数名>, <③入力変数>)	
戻り値	戻り値	<読み取り結果> 数値
	<p>グローバル共有から値が読み取りできたか、結果が得られます。</p> <p>1 : 読み取りできた。</p> <p>0 : 両端キューが空だったので読み取れなかった。</p>	
パラメータ	①	<管理番号> 数値
	「COMMON OPEN」でオープンした管理番号を指定します。	
	②	<変数名> 文字列
	<p>グローバル共有の両端キューの先頭から取り出す変数名を、文字列で指定します。</p> <p>ここで指定した変数名が、グローバル共有の両端キューの名前です。</p> <p>変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。</p>	
	③	<入力変数> 変数名
	<p>グローバル共有の両端キューの先頭から取り出す変数を指定します。</p> <p>ここで指定した変数が、グローバル共有の両端キューから取り出して格納されます。</p>	
	<p>指定できる変数は、単一の数値または文字列、構造体のみです。</p> <p>配列、連想配列などは、指定できません。</p>	
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に格納されている値の型と、入力変数で指定する型は、同じ型でなければなりません。</li> <li>・2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON2 DEQUE LPOP(1, "LV", V) PRINT V COMMON CLOSE #1</pre> <p>’ グローバル共有に「LV」という両端キューがあり、先頭のキュー値をV変数に取り出します。</p>	



## 3.8.43 COMMON2 DEQUE BACK

関数			
機 能	グローバル共有から、文字列で指定された変数名で両端キューの末尾より値を参照します。		
書 式	<(戻り値)読み取り結果> = COMMON2 DEQUE BACK ( <①管理番号>, <②変数名>, <③入力変数> )		
戻り値	戻り値	<読み取り結果>	数値
	グローバル共有から値が読み取りできたか、結果が得られます。 1 : 読み取りできた。 0 : 両端キューが空だったので読み取れなかった。		
パラメータ	①	<管理番号>	数値
	COMMON OPENでオープンした管理番号を指定します。		
	②	<変数名>	文字列
	グローバル共有の両端キューの先頭から取り出す変数名を、文字列で指定します。 ここで指定した変数名が、グローバル共有の両端キューの名前です。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。		
	③	<入力変数>	変数名
	グローバル共有の両端キューの末尾から取り出す変数を指定します。 ここで指定した変数が、グローバル共有の両端キューから取り出して格納されます。  指定できる変数は、単一の数値または文字列、構造体のみです。 配列、連想配列などは、指定できません。		
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に格納されている値の型と、入力変数名で指定する型は、同じ型でなければなりません。</li> <li>・2021/3から、変数名は大文字扱いとします。</li> </ul>		
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON DEQUE BACK(1, "LV", V) PRINT V COMMON CLOSE #1</pre>		

## 3.8.44 COMMON2 DEQUE FRONT

関数		
機 能	グローバル共有から、文字列で指定された変数名で両端キューの先頭より値を参照します。	
書 式	<(戻り値)読み取り結果> = COMMON2 DEQUE FRONT( <①管理番号>, <②変数名>, <③入力変数> )	
戻り値	戻り値	<読み取り結果> 数値
	<p>グローバル共有から値が読み取りできたか、結果が得られます。</p> <p>1 : 読み取りできた。</p> <p>0 : 両端キューが空だったので読み取れなかった。</p>	
パラメータ	①	<管理番号> 数値
	COMMON OPENでオープンした管理番号を指定します。	
	②	<変数名> 文字列
	<p>グローバル共有の両端キューの先頭から取り出す変数名を、文字列で指定します。</p> <p>ここで指定した変数名が、グローバル共有の両端キューの名前です。</p> <p>変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。</p>	
	③	<入力変数> 変数名
	<p>グローバル共有の両端キューの末尾から取り出す変数を指定します。</p> <p>ここで指定した変数が、グローバル共有の両端キューから取り出して格納されます。</p>	
	<p>指定できる変数は、単一の数値または文字列、構造体のみです。</p> <p>配列、連想配列などは、指定できません。</p>	
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に格納されている値の型と、入力変数名で指定する型は、同じ型でなければなりません。</li> <li>・2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	<pre>COMMON OPEN "127.0.0.1" AS #1 PRINT "LVの読み取り結果="; COMMON2 DEQUE FRONT(1, "LV", V) PRINT V COMMON CLOSE #1</pre>	

## 3.8.45 COMMON2 DEQUE LEN

関数		
機 能	グローバル共有から、文字列で指定した変数名で両端キューの個数を得ます。	
書 式	<(戻り値)両端キューの個数> = COMMON2 DEQUE LEN(<①管理番号>, <②変数名>)	
戻り値	戻り値	数値
	<両端キューの個数> グローバル共有の両端キューの個数を得ます。	
パラ メータ	①	数値
	<管理番号> 「COMMON OPEN」でオープンした管理番号を指定します。	
	②	変数名
	<変数名> グローバル共有にある両端キューの変数名を、文字列で指定します。 指定した変数名の内容が、両端キューでない場合、エラーとなります。 変数名は、英字で始まる英数文字を含む1文字以上の文字列で、英字は大文字が必須です。	
備 考	<ul style="list-style-type: none"> <li>・グローバル共有に格納されている値の型が、両端キューでなければなりません。</li> <li>・2021/3から、変数名は大文字扱いとします。</li> </ul>	
使用例	COMMON OPEN "127.0.0.1" AS #1 PRINT "両端キューLVの個数="; COMMON2 DEQUE LEN(1, "LV") COMMON CLOSE #1 ' グローバル共有に、「LV」という両端キューがあり、その両端キューの個数を得ます。	

3. 8. 46 COMMON2 VLIST\$

関数			
機 能	グローバル共有の変数名のリストを、文字列配列で得ます。		
書 式	<(戻り値) 変数名の配列> = COMMON2 VLIST\$( <①管理番号> )		
戻り値	戻り値	<変数名の配列>	文字列
	グローバル共有の変数名のリストを、文字列配列で得られます。		
パラ メータ	①	<管理番号>	数値
	「COMMON OPEN」 でオープンした管理番号を指定します。		
備 考	・ 「COMMON VLIST\$」 と異なり、AJANの変数の型ポストフィクスは、戻り値の変数名に含まれません。		
使用例	<pre>' AJANの変数名を使った書き込み CNT% = 123 COMMON PRINT #1, CNT% LC&amp; = 456 COMMON PRINT #1, LC&amp; S\$ = "hello" COMMON PRINT #1, S\$ ' 任意の名前を使った書き込み COMMON WRITE #1, "TEST", "AJAN"  PRINT COMMON2 VLIST\$(1) ' グローバル共有の変数名のリストが配列で得られます。 ' [ TEST, LC, S, CNT ] のような結果が得られます。</pre>		

## 3.9 その他に関する関数・命令

### 3.9.1 QRCODE\_GET

関数	
機能	QRコード作成に必要な情報を得ます。
書式	<(戻り値)QRコード情報> = QRCODE_GET(<①文字列> [, <②オプション> ] )
戻り値	戻り値
	<QRコード情報> 配列 文字列からQRコード作成に必要な情報を、数値の2次元配列形式で得られます。 行(1次元目)×列(2次元目)の配列形式で、値は1ないしは0が得られます。 1がQRコードの黒点、0が白部分に相当します。
パラメータ	①
	<文字列> 文字列 QRコードを得るための文字列を指定します。
	②
	<オプション> 配列 QRコード情報を得る際のオプションを、文字列の配列で指定できます。
	オプション
	挙動
ECL=<値>	
QRコードの誤り訂正機能のレベルを指定できます。 <値>を以下のように選択できます。 (省略時、レベルLが選択されます) "L" <= レベルL(訂正能力 約 7%) "M" <= レベルM(訂正能力 約 15%) "Q" <= レベルQ(訂正能力 約 25%) "H" <= レベルH(訂正能力 約 30%)	
VER=<値>	
QRコードのバージョンを指定できます。 バージョン 1(21x21 セル)からバージョン 40(177x177 セル)まで指定できます。 (省略時、文字列から自動推定されます)	
MICRO	
マイクロ QR コードを指定できます。 (省略時、通常の QR コードが採用されます)	
UPPER	
文字列中の小文字を大文字にします。 (省略時、大文字・小文字使用可能です)	
備考	<ul style="list-style-type: none"> <li>QRコードは、株式会社デンソーウェーブの登録商標です。</li> <li>本コマンドは、libqrencodeライブラリを使用しています。 このため、QRコードのモデル2に対応し、連結機能はサポートしていません。</li> <li>QRコードについての知識は、書籍およびQRコードドットコム( <a href="https://www.qrcode.com">https://www.qrcode.com</a> ) などを参照ください。</li> </ul>
使用例	<pre> LIST ARY ARY = QRCODE_GET("Hello AJAN")  MARGIN = 4   ' マージン SZ = 6       ' 1点のドットサイズ  FOR Y=0 TO UBOUND(ARY, 1)   FOR X=0 TO UBOUND(ARY, 2)     IF ARY(Y, X) THEN       PX = (MARGIN + X) * SZ       PY = (MARGIN + Y) * SZ       GUGRECTANGLE PIC 10, (PX, PY)-(SZ, SZ), 1, 1     NEXT X   NEXT Y NEXT Y </pre>

	' QRCODE_GET で得たQRコード情報から、GUI コマンドを使って描画するコード断片の例です。
--	--

## 3.10 拡張コマンドのサブルーチン集

ここで紹介しているサブルーチンは、AJANコマンドを組み合わせて作ったサブルーチンです。

サブルーチンは、以下の場所に配置されています。

```
/usr/share/interface/AJANPro/include/EXT001.AJN
```

ここで紹介した命令および関数群は、このサブルーチン(EXT001.AJN)で定義&実装したルーチン群と説明です。

紹介したサブルーチンを使用する際は、プログラムの先頭で、INCLUDE命令で読み込んでから利用ください。以下のように。(赤字部分)

```
INCLUDE "EXT001.AJN"
```

## 3.10.1 GET\_OCR\$

関数		
機 能	画像ファイルに対して、tesseract コマンドを用いて OCR解析によりテキスト情報を取得します。	
書 式	<(戻り値)解析テキスト> = GET_OCR\$( <①画像ファイルパス>, <②オプション> )	
戻り値	戻り値	<解析テキスト> 文字列
	画像ファイルを tesseract コマンドに渡して、OCR解析を行った結果のテキスト情報を得ます。	
パラ メータ	①	<画像ファイルパス> 文字列
	解析対象となる画像ファイルのパスを指定します。	
	②	<オプション> 文字列
	サブルーチンが呼び出す、tesseract コマンドに渡す追加オプションを指定できます。	
	空文字列を渡すと、"-l jpn+eng" が採用されます。	
備 考	<ul style="list-style-type: none"> <li>この関数を呼び出すには、「INCLUDE "EXT001.AJN"」 で、定義を呼び出してから利用ください。</li> <li>本サブルーチンの内部では、tesseract コマンドを呼び出しています。</li> <li>tesseract コマンドを利用する為に、tesseract-ocr パッケージを導入する必要があります。</li> <li>tesseract コマンドが認識できる画像ファイルは、JPEG, PNG, GIF形式などです。</li> </ul>	
使用例	<pre>S\$ = GET_OCR\$("/tmp/1.png", "")</pre> <p>' 画像ファイル(/tmp/1.png) をOCR解析してテキスト情報を得ようとしています。</p>	



## 第4章 サンプルプログラム

サンプルプログラムは「/usr/share/interface/AJANPro/samples/EXT/EXT/」に格納されています。

AJAN統合開発環境を起動すると、左ペインのエクスプローラウィンドウ内の「Samples/EXT/EXT/」に、ファイルが取り込まれて配置されます。

### 4.1 サンプルプログラム

#	ファイル名	内容
1	COMMON_INPUT_PRINT. AJN	グローバル共有機能を用いて、プロセス間で、値のやり取りと、メッセージの通知と受信を行うサンプルプログラムです。 COMMON OPEN でオープンし、COMMON CLOSE でクローズします。 COMMON INPUTR で読み取り、COMMON PRINT で書き込み、COMMON LOCK / UNLOCK でロック処理、COMMON NOTIFY でメッセージ通知と、ON COMMON CALL でメッセージ受信です。
2	CONVBIN_CONVPHY. AJN	CONVBIN 関数で連続量の値をバイナリ値へ、CONVPHY 関数でバイナリ値を連続量の値へ変換するサンプルプログラムです。
3	DIM_CALC. AJN	配列に対して、平均値(DIMAVG 関数)、中央値(DIMMEDIAN 関数)、最大値(DIMMAX 関数)、最小値(DIMMIN 関数)、合計値(DIMSUM 関数)、標準偏差値(DIMSTDEVP 関数)、分散値(DIMVARP 関数)を求めます。
4	DIM_CAST. AJN	文字列配列に対して、数値を取り出し(DIMCINT / DIMCLNG / DIMCSNG / DIMCDBL 関数)、文字列配列に変換(DIMCSTR\$関数)します。
5	DIM_CAST_USE. AJN	CSV 文字列を2次元配列に変換(CSV2ARRAY\$関数)した後、任意の場所の値を取り出したり(DIMCINT 関数)、計算したり(DIMSUM 関数)、文字列配列に戻したり(DIMCSTR\$関数)する応用例です。
6	DIMADD_SUB_DIV_MOD_MUL. AJN	配列の算術演算(DIMADD / DIMSUB / DIMDIV / DIMMOD / DIMMUL 関数)を行います。
7	DIMAND_OR_XOR_IMP_EQV_NOT. AJN	配列の論理演算(DIMAND / DIMOR / DIMXOR / DIMIMP / DIMEQV / DIMNOT 関数)を行います。
8	DIMEQ_NE_LT_GT_LTE_GTE. AJN	配列の関係比較演算(DIMEQ / DIMNE / DIMLT / DIMGT / DIMLTE / DIMGTE 関数)を行います。
9	DIMIIF. AJN	配列に対して、条件判定を使った配列の抽出(DIMIIF / DIMIIF\$関数)を行います。
10	HOURL_MINUTE_SECOND. AJN	今日の時刻から時(HOUR 関数) / 分(MINUTE 関数) / 秒(SECOND 関数)の値を、それぞれ抽出します。
11	JOIN. AJN	文字列をカンマ(,)で分割(SPLIT\$関数)した後、再び結合(JOIN\$関数)します。
12	JSON. AJN	JSON_PARSE@関数 で JSON 型オブジェクトを作り、JSON_DUMP\$関数で文字列に戻す事例です。
13	LCASE_UCASE. AJN	文字列中の大文字を小文字に変換(LCASE\$関数)したり、小文字を大文字に変換(UCASE\$関数)します。
14	LEFT_MID_RIGHT. AJN	文字列に対して、左(LEFT\$関数)から抜き出し、右(RIGHT\$関数)から抜き出し、中(MID\$関数)から抜き出します。 また、バイト単位で、左(LEFTB\$関数)から抜き出し、右(RIGHTB\$関数)から抜き出し、中(MIDB\$関数)から抜き出す

#	ファイル名	内容
		事例も紹介します。
15	OCR. AJN	画面キャプチャ ボタンを押すと、gnome-screenshot を呼び出して、画面全体の スクリーンショットを撮ります。 OCR 処理 ボタンを押すと、スクリーンショット画面の画像データを元に、GET_OCR\$ 関数(中は、tesseract (OCR ソフト)) を呼び出して、解析した文字列を表示します。
16	PYOBJ. AJN	Python 連携機能を使って Python で作ったコードを読み取り (PYOBJ CREATE CODE 関数)、関数 (PYOBJ CALL FUNCTION 関数) やメソッド (PYOBJ CALL METHOD 関数) を呼び出した後に閉じます。 (PYOBJ CLOSE 命令)
17	PYOBJ_SCATTER_GUI. AJN	Python 連携機能を使って Python の matplotlib ライブラリを呼び出して、散布図を GUI に描画します。 この散布図描画処理は、スケールを変更することで、描画する点数を変更できます。
18	QRCODE_GET. AJN	QRCODE_GET 関数 を使って、GUI 上に QR コードを描画するサンプルプログラムです。
19	SNDREAD_SNDWRITE. AJN	オーディオ入力からサンプリング入力 (SNDREAD 関数) 後、オーディオ出力に対して同じデータをサンプリング出力 (SNDWRITE 命令) します。
20	SNDREAD_SNDWRITE_GUI. AJN	SNDREAD_SNDWRITE. AJN サンプルプログラムの GUI 版です。 オーディオ入力からサンプリング入力 (SNDREAD 関数) しつつ、データを波形グラフ表示し、同じデータをサンプリング出力 (SNDWRITE 命令) します。
21	SNDREADLEN_SNDWRITELEN_GUI. AJN	オーディオ入力の入力バッファ件数 (SNDREADLEN) とオーディオ出力の出力バッファ書き込み可能件数 (SNDWRITELEN) を読み取って、入出力バッファにデータがある分を、オーディオ入力 (SNDREAD) してからオーディオ出力 (SNDWRITE) します。 これにより、データの途切れなく入力と出力ができます。
22	STRDELB_STRINSB. AJN	文字列に対して、STRDELB\$関数で指定バイト長の削除、STRINSB\$関数で挿入を行います。
23	TRUNC_ROUND. AJN	数値を指定位置で切り捨て (TRUNC 関数) / 四捨五入 (ROUND 関数) します。
24	YEAR_MONTH_DAY. AJN	今日の日付から年 (YEAR 関数) / 月 (MONTH 関数) / 日 (DAY 関数) の値を、それぞれ抽出します。

## 第5章 索引

<b>A</b>		COMMON VLIST\$	196
ARRAY2BINBLK\$	47	COMMON WRITE	177
ARRAY2STR\$	48	COMMON2 CDIM	199
<b>B</b>		COMMON2 DEQUE LEN	211
BINBLK2ARRAY	49	COMMON2 DEQUE LPOP	208
BIT2NUM	141	COMMON2 DEQUE LPUSH	206
<b>C</b>		COMMON2 DEQUE POP	207
CBOOL	51	COMMON2 DEQUE PUSH	205
CDBL	52	COMMON2 ERASE	198
CHRTYPE	93	COMMON2 GET_DICT_KEYS\$	200
CINT	52	COMMON2 HAS_DICT_KEY	201
CLNG	53	COMMON2 LDICT	202
CLOCK2DATETIMESTR\$	54	COMMON2 LDIM	202
COMMON CDIM	183	COMMON2 UBOUND	203
COMMON CLOSE	171	COMMON2 VARTYPE	204
COMMON DEQUE BACK	193, 209	COMMON2 VLIST\$	212
COMMON DEQUE FRONT	194, 210	CONVBIN	55
COMMON DEQUE LEN	195	CONVPHY	57
COMMON DEQUE LPOP	192	CONVUNIT	58
COMMON DEQUE LPUSH	190	CSNG	53
COMMON DEQUE POP	191	CURDIR\$	165
COMMON DEQUE PUSH	189	<b>D</b>	
COMMON ERASE	172	DATEADD	59
COMMON FREEFILE	171	DATETIMESTR2CLOCK	60
COMMON GET_DICT_KEYS\$	184	DAY	61
COMMON HAS_DICT_KEY	185	DIMADD	106
COMMON INPUT	173	DIMADD\$	107
COMMON INPUTR	174	DIMAND	113
COMMON LDICT	186	DIMAVG	102
COMMON LDIM	186	DIMCDBL	131
COMMON LOCK	178	DIMCINT	128
COMMON NOTIFY	180	DIMCLNG	129
COMMON ON / OFF	182	DIMCSNG	130
COMMON OPEN	169	DIMCSTR\$	132
COMMON PRINT	176	DIMDIV	109
COMMON READ	175	DIMEQ	120
COMMON SWAPDB	197	DIMEQV	117
COMMON UBOUND	187	DIMFIND	133
COMMON UNLOCK	179	DIMGET	135
COMMON VARTYPE	188	DIMGET\$	136
		DIMGT	123
		DIMGTE	125

DIMIIF	126	<i>J</i>	
DIMIIF\$	127	JOIN\$	69
DIMIMP	116		
DIMLT	122	<i>L</i>	
DIMLTE	124	LCASE\$	69
DIMMAP	137	LEFT\$	70
DIMMAP\$	138	LEFTB\$	70
DIMMAX	103		
DIMMEDIAN	102	<i>M</i>	
DIMMIN	103	MATH ISEQ	71
DIMMOD	110	MINUTE	72
DIMMUL	111	MKUUID\$	72
DIMNE	121	MONTH	73
DIMNOT	112	<i>N</i>	
DIMOR	114		
DIMREDUCE	139	NUM2BIT	141
DIMROL	119	<i>O</i>	
DIMROR	119	ON COMMON CALL	181
DIMSET	140	ONEDIM SORT	142
DIMSHL	118	ONEDIM SORT\$	142
DIMSHR	118		
DIMSTDEVP	104	<i>P</i>	
DIMSUB	108	PASSWORD_HASH\$	74
DIMSUM	104	PASSWORD_VERIFY	74
DIMVARP	105	PYOBJ CALL FUNCTION	155
DIMVARS	105	PYOBJ CALL METHOD	156
DIMXOR	115	PYOBJ CLOSE	158
		PYOBJ CREATE CODE	154
<i>E</i>			
ENVIRON\$	168	<i>Q</i>	
		QRCODE_GET	213
<i>F</i>			
FORMAT\$	62	<i>R</i>	
FORMATDATETIME\$	65	RIGHT\$	75
		RIGHTB\$	75
<i>G</i>			
GET_OCR\$	216	<i>S</i>	
GETCOMMANDLINEARGS\$	166	SECOND	76
GETSYSTEMINFO\$	167	SNDCLOSE	160
		SNDOPEN	159
<i>H</i>		SNDREAD	161
HASH\$	66	SNDREADLEN	162
HOURL	67	SNDWRITE	163
		SNDWRITELEN	164
<i>I</i>			
ICONV\$	68	STR_AND\$	79
		STR_BASE64_DECODE\$	77
		STR_BASE64_ENCODE\$	77

STR_DUMP\$	78	STR2ARRAY	96
STR_ENDSWITH	84	STRDELB\$	97
STR_EQV\$	80	STRINSB\$	98
STR_HAN2ZEN\$	81	STRREVERSE\$	99
STR_HIRA2KATA\$	83	STRREVERSEB\$	99
STR_IMP\$	80	<i>T</i>	
STR_ISALNUM	86	TRUNC	100
STR_ISALPHA	87	TWODIM EXISTS	147
STR_ISASCII	92	TWODIM EXISTS\$	148
STR_ISCNTRL	87	TWODIM FILTER	143
STR_ISDIGIT	88	TWODIM FILTER\$	144
STR_ISGRAPH	88	TWODIM JOIN	145
STR_ISLOWER	89	TWODIM JOIN\$	146
STR_ISPRINT	89	TWODIM NOT EXISTS	149
STR_ISPUNCT	90	TWODIM NOT EXISTS\$	150
STR_ISSPACE	90	TWODIM SORT	151
STR_ISUPPER	91	TWODIM SORT\$	152
STR_ISXDIGIT	91	TWODIM TRANSPOSE	153
STR_KATA2HIRA\$	83	TWODIM TRANSPOSE\$	153
STR_OR\$	79	<i>U</i>	
STR_REMOVEPREFIX\$	85	UCASE\$	100
STR_REMOVESUFFIX\$	85	UUIDCOMP	101
STR_REPEAT\$	80	<i>Y</i>	
STR_STARTSWITH	84	YEAR	101
STR_VALIDUTF8	86		
STR_XOR\$	79		
STR_ZEN2HAN\$	82		

## 第6章 重要な情報

### 保証の内容と制限

弊社は本ドキュメントに含まれるソースプログラムの実行が中断しないこと、またはその実行に誤りが無いことを保証していません。

本製品の品質や使用に起因する、性能に起因するいかなるリスクも使用者が負うものとします。

弊社はドキュメント内の情報の正確さに万全を期しています。万一、誤記または誤植などがあつた場合、弊社は予告無く改訂する場合があります。ドキュメントまたはドキュメント内の情報に起因するいかなる損害に対しても弊社は責任を負いません。

ドキュメント内の図や表は説明のためであり、ユーザ個別の応用事例により変化する場合があります。

### 著作権、知的所有権

弊社は本製品に含まれるおよび本製品に対する権利や知的所有権を保持しています。

本製品はコンピュータ ソフトウェア、映像/音声(例えば図、文章、写真など)を含んでいます。

### 医療機器/器具への適用における注意

弊社の製品は人命に関わるような状況下で使用される機器に用いられる事を目的として設計、製造された物では有りません。

弊社の製品は人体の検査などに使用するに適する信頼性を確保する事を意図された部品や検査機器と共に設計された物では有りません。

医療機器、治療器具などの本製品の適用により、製品の故障、ユーザ、設計者の過失などにより、損傷/損害を引き起こす場合が有ります。

### 複製の禁止

弊社の許可なく、本ドキュメントの全て、または一部に関わらず、複製、改変などを行うことはできません。

### 責任の制限

弊社は、弊社または再販売者の予見の有無にかかわらず発生したいかなる特別損害、偶発的損害、間接的な損害、重大な損害について、責任を負いません。

本製品(ハードウェア、ソフトウェア)のシステム組み込み、使用、ならびに本製品から得られる結果に関する一切のリスクについては、本製品の使用者に帰属するものとします。

本製品に含まれる不都合、あるいは本製品の供給(納期遅延)、性能もしくは使用に起因する付帯的損害もしくは間接的損害に対して、弊社に全面的に責がある場合でも、弊社はその製品に対する改良(有償サービスの利用)、代品交換までとし、製品の予防交換並びに、代金減額等、金銭面での賠償の責任は負わないものとします。

本製品は、日本国内仕様です。

### 商標/登録商標

本書に掲載されている会社名、製品名は、それぞれ各社の商標または登録商標です。

## 改訂履歴

Ver.	年 月	改 訂 内 容
0.90	2019年10月	新規作成
1.00	2022年1月	誤記修正 コマンド追加。TWDIM SORT / SORT\$, DIMGET, DIMGET\$, TWODIM TRANSPOSE / TRANSPOSE\$, COMMON DEQUE BACK / FRONT, COMMON READ, COMMON WRITE, COMMON VLIST\$, GET_OCR\$, STRREVERSE\$, STRREVERSEB\$, DIMMAP, DIMMAP\$, DIMREDUCE, TWODIM FILTER / FILTER\$, TWODIM JOIN / JOIN\$, DIMSET サンプル追加。OCR.AJN
1.10	2023年3月	コマンド追加。CLOCK2DATETIMESTR\$, DATETIMESTR2CLOCK, TWODIM EXISTS, TWODIM EXISTS\$, TWODIM NOT EXISTS, TWODIM NOT EXISTS\$, STR_ISUTF8, STR_ISALNUM, STR_ISALPHA, STR_ISCNTRL, STR_ISDIGIT, STR_ISGRAPH, STR_ISLOWER, STR_ISPRINT, STR_ISPUNCT, STR_ISSPACE, STR_ISUPPER, STR_ISXDIGIT, STR_ISASCII, CHRTYPE, GETSYSTEMINFO\$ オプション追加。FORMATDATETIME\$, TWODIM SORT\$, ONEDIM SORT\$

このマニュアルは、製品の改良その他により将来予告なく改訂しますので、予めご了承ください。