05DE-4EB0341

Pro-face

リアルタイム品質管理支援ツール

Esyface-spc

Statistical Process Control

ルール（TCL）説明書

はじめに

このたびは、「Esyface-spc Ver.1.00」をご採用いただき、誠にありがとうございます。
この製品を正しくご使用いただくために、マニュアル類をよくお読みください。
また、マニュアル類は必ずご利用になる場所のお手元に保管し、いつでもご覧いただけるようにしておいてください。

おことわり

1．「Esyface-spc Ver.1.00」（以下本製品といいます）のプログラムおよびマニュアル類は、すべて（株）デジタルの著作物であり、(株)デジタルがユーザーに対し使用権を許諾したものです。当該内容に反する行為は、日本国内外の法令により禁止されています。
2．本書の内容については万全を期して作成しておりますが、万一お気づきの点がありましたら、担当営業または担当 SE までご連絡ください。
3．前項にかかわらず、本製品を運用した結果の影響および第三者のいかなる請求にも、(株)デジタルは一切責任を負いません。
4．本製品が記録・表示する情報の中に、(株)デジタルまたは第三者が権利を有する無体財産権、知的所有権に関わる内容を含む場合がありますが、これは(株)デジタルがこれらの権利の利用について、ユーザーまたはその他の第三者に、何らかの保証や許諾を与えるものではありません。また本製品に記録・表示された情報を使用したことにより第三者の知的所有権などの権利に関わる問題が生じた場合、(株)デジタルはその責任を負いませんのであらかじめご了承ください。

「spc」は、Statistical Process Control(統計的工程管理）の頭文字です。

目 次

表記のルール

本書は、以下のルールで表記します。

パソコンや Windows そのものに関することは、パソコンをお買い上げの販売店、メーカにお問い合わせください。

安全に関する注意表記

本製品のご使用上、安全に関して重要な説明には、以下の表示を添えています。

表示	意味内容
⚠警告	この表示を無視して誤った取り扱いをすると、人が死亡または重症を負う可能性が想定される内容を示します。
⚠注意	この表示を無視して誤った取り扱いをすると、人が傷害を負ったり、物的損害の発生が想定される内容を示します。
ⓘ	必ず実施していただきたい操作、作業などを表します。
⊘	決して行ってはならない操作、作業などを表します。

商標権などについて

本書に記載の社名、商品名は、各社の商号、商標(登録商標を含む)またはサービスマークです。

本製品の表示・記述の中では、これらの権利に関する個別の表示は省略しております。

商標等	権利者
Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows 98 SE, Windows Me, Windows NT, Windows 2000, Windows XP	米国Microsoft社
Intel, Pentium, MMX	米国Intel社
Pro-face	(株)デジタル
PC/AT	米国IBM社

なお、上記商号･商標類で、本書での表記と正式な表記が異なるものは以下の通りです。

本書での表記	正式な表記
Windows 2000	Microsoft(R) Windows(R) 2000 オペレーティングシステム
Windows XP	Microsoft(R) Windows(R) XP オペレーティングシステム

使用上の注意

本製品の使用について

誤動作や事故の原因となりますので、以下の点にご注意ください。

⚠警告

⏻ **タッチパネル　　スイッチやパソコンからのオペレーションは非常停止用スイッチとして使えません。産業用ロボットほか、労働大臣が指定する産業機械設備の非常停止用スイッチとしては、必ず人間が直接操作するスイッチを設置することが関係法令で義務づけられています。また、これ以外の装置設備でも、安全確保のため、必ず同様のスイッチを設置してください。**

⚠注意

⃠　・テキストエディタなどを使用して、本製品のファイルの中身を変更しないでください。

■ディスクの取り扱いについて

ディスクの破損・故障を防ぐため、以下の点にご注意ください。

- CD-ROM の記録面に手を触れないでくだい。
- 極端な高温や低温、湿気やホコリの多い場所にディスクを置かないでくだい。

1　概要

本説明書は、Esyface-spc の管理図にプロットされた内容から、「警報ルール」に従ってアラーム検出する条件を記述するルール（TCL）について記載しています。汎用のスクリプト言語である TCL を使用し、ユーザ側で任意のアラーム判定関数を定義する事ができます。また工程管理図で有効な関数については Esyface-spc 本体側でサポート関数として準備しています。スクリプト側でこれらのサポート関数を使用し、ルールを構築する事ができます。
ルールの判定については、工程管理図および p／np 管理図にて実施します。

2　アラーム表示概要

①アラームが発生した場合、管理図ではアラームに該当するデータのマーカ色を変更します。
②アラーム一覧に、発生時刻・任意に設定されたアラーム内容を表示します。

3　アラーム設定方法

①アラームの監視内容については、ユーザ側でスクリプト（Esyface-spc で準備）を使用し、設定を行います。ただし、基本的な JIS の８つのルールについてはスクリプト内に標準で準備しています。
②アラーム内容については、ユーザ側でスクリプトを変更することにより、任意のルールを設定する事が可能です。

4　データのアラーム判定を行うタイミング

①CSV データ
　CSV を読込んだ際にチェックを行います。
②DB データ
　DB からデータを読込んだ際にチェックを行います。
③Esyface データ

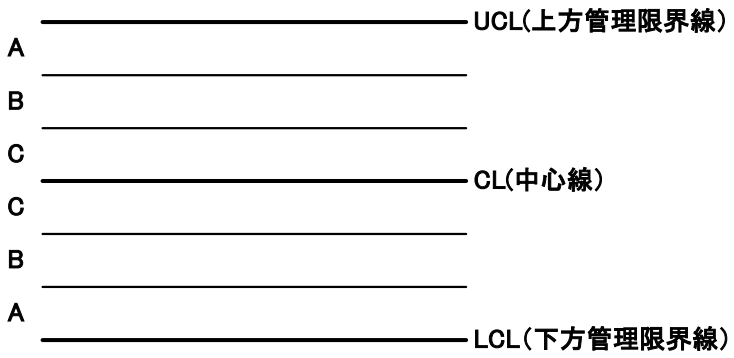
Esyface サーバからデータを受信した際にチェックを行います。

④AddData 関数

AddData 関数によりデータを与えられたタイミングでチェックを行います。

5 管理限界線の考え方

上下の管理限界線は中心線よりそれぞれ３σの距離にあり、１σずつに分割して上よりＡ，Ｂ，Ｃ，Ｃ，Ｂ，Ａとします。



6 Esyface-spc のサポート関数

※ｉはデータの系列を示します。

共通関数

関数名	内容
JSNi i	(i)番目の郡内データ数
JSTi i	(i)番目の時間

Xbar 管理図専用関数

関　数　名	内　　容
JSXi　i	(i)番目のデータ値（群内の Xbar）
JSXij　i　j	(i)番目の群内 j 番目のデータ
JSAV　i	Xbar の値
JSUAi　i	(i)番目の上限 A ライン（上限３σ） ※群内の数が一定の場合は UCL の値 ※群内の数が一定でない場合は各群の UCL の値
JSUBi　i	(i)番目の上限 B ライン（上限２σ） ※群内の数が一定の場合は UCL の 2／3 の値 ※群内の数が一定でない場合は各群の UCL の 2／3 の値
JSUCi　i	(i)番目の上限 C ライン（上限１σ） ※群内の数が一定の場合は UCL の 1／3 の値 ※群内の数が一定でない場合は各群の UCL の 1／3 の値
JSLAi　i	(i)番目の下限 A ライン（下限３σ） ※群内の数が一定の場合は LCL の値 ※群内の数が一定でない場合は各群の LCL の値
JSLBi　i	(i)番目の下限 B ライン（下限２σ） ※群内の数が一定の場合は LCL の 2／3 の値 ※群内の数が一定でない場合は各群の LCL の 2／3 の値
JSLCi　i	(i)番目の下限 C ライン（下限１σ） ※群内の数が一定の場合は LCL の 1／3 の値 ※群内の数が一定でない場合は各群の LCL の 1／3 の値
JSFL1	任意線１の値
JSFL2	任意線２の値
JSFL3	任意線３の値
JSFL4	任意線４の値
JSFL5	任意線５の値
JSBetweenOver　i　n　min　max	(i)番目から前後 n 個のデータで min から max の範囲を超えたデータの数 ※ルール１，５，６，８で使用
JSBetweenInside　i　n　min　max	(i)番目から前後 n 個のデータで min から max の範囲内にあるデータの数 ※ルール２，７で使用
JSDirction　i	(i)番目の前後データの推移が連続して同じ傾きである場合のデータ数 ※ルール３で使用
JSRepetition　i	(i)番目以降の連続した領域のデータで交互に増減しているデータ数 ※ルール４で使用

上下限の UCL および LCL の値は、Xbar-R または Xbar-S の表示状態により、表示選択された管理図の算出方法に基づいた Xbar 管理図の値となります。

R（Rs）管理図およびS管理図の専用関数

関　数　名	内　　容
JSUi　i	(i)番目のデータ値
JSUAV　i	(i)番目の平均線
JSUUAi　i	(i)番目の上限Aライン（上限３σ）
JSUUBi　i	(i)番目の上限Bライン（上限２σ）
JSUUCi　i	(i)番目の上限Cライン（上限１σ）
JSULAi　i	(i)番目の下限Aライン（下限３σ）
JSULBi　i	(i)番目の下限Bライン（下限２σ）
JSULCi　i	(i)番目の下限Cライン（下限１σ）
JSUFL1	任意線１の値
JSUFL2	任意線２の値
JSUFL3	任意線３の値
JSUFL4	任意線４の値
JSUFL5	任意線５の値
JSUBetweenOver　i　n　min max	(i)番目から前後ｎ個のデータでminからmaxの範囲を超えたデータの数 ※ルール１，５，６，８で使用
JSUBetweenInside　i　n　min max	(i)番目から前後ｎ個のデータでminからmaxの範囲内にあるデータの数 ※ルール２，７で使用
JSUDirction　i	(i)番目の前後データの推移が連続して同じ傾きである場合のデータ数 ※ルール３で使用
JSURepetition　i	(i)番目以降の連続した領域のデータで交互に増減しているデータ数 ※ルール４で使用

※上下限の３σの値はXbar管理図と同様の算出方法により算出した、R管理図もしくはS管理図の値となります。

p 管理図および np 管理図の専用関数

関　数　名	内　　容
JSVAL i	(i)番目のデータ値
JSAV　i	(i)番目の平均線
JSAU　i	(i)番目の上限 A ライン（上限３σ）
JSBU　i	(i)番目の上限 B ライン（上限２σ）
JSCU　i	(i)番目の上限 C ライン（上限１σ）
JSAL　i	(i)番目の下限 A ライン（下限３σ）
JSBL　i	(i)番目の下限 B ライン（下限２σ）
JSCL　i	(i)番目の下限 C ライン（下限１σ）
JSBetweenOver　i　n　min　max	(i)番目から前後 n 個のデータで min から max の範囲を超えたデータの数 ※ルール１，５，６，８で使用
JSBetweenInside　i　n　min　max	(i)番目から前後 n 個のデータで min から max の範囲内にあるデータの数 ※ルール２，７で使用
JSDirction　i	(i)番目の前後データの推移が連続して同じ傾きである場合のデータ数 ※ルール３で使用
JSRepetition　i	(i)番目以降の連続した領域のデータで交互に増減しているデータ数 ※ルール４で使用

※上下限の UCL および LCL の値は、ｐ管理図または np 管理図の表示状態により、表示選択された管理図の算出方法に基づいた Xbar 管理図の値となります。

7　ＪＩＳの８つのルール

ルール１：　１点が領域Ａを超えている。

ルール２：　９点が中心線に対して同じ側にある。

ルール３：　６点が増加、または減少している。

ルール４：　１４の点が交互に増減している。

ルール５：　連続する３点中、２点が領域Ａまたはそれを超えた領域にある。

ルール６：　連続する５点中、４点が領域Ｂまたはそれを超えた領域にある。

ルール７：　連続する１５点が領域Ｃにある。

ルール８：　連続する８点が領域Ｃを超えた領域にある。

8　標準ルール解説

8.1　Esyface-spc が実行するルール

```
proc Func {i} {
}
proc Func2 {i} {
}
```

・Esyface-spc は Xbar のルールとして Func{i}の内容を実行し、R 管理図または S 管理図のルールとして Func2{i}の内容を実行します。
・各ルールは Func 以降の”　{　}　”で囲まれた内容を実行します。
・Func の引数”　i　”は判定を行う群（データ元のレコード）の Index です。
・TCL の文法および固有の関数については”9　TCL 文法”を参照して下さい。

8.2　Func 内で定義しているルールの内容

```
set A [ rule1 $i ]
if { $A != 0 } {
   return $A
}
set A [ rule21 $i ]
if {$A != 0} {
   return $A
}
```

・rule1 関数（ユーザが作成した TCL 関数）を呼出します。
関数の内容を実行し、変数"A"へ結果を代入します。
・変数"A"（rule1 の結果）が"0"でなければ"A"の値を今回の群の判定結果として Esyface-spc へ返します。
・変数"A"（rule1 の結果）が"0"であれば、次のルールである rule21 関数を呼出し、関数の内容を実行します。　以降の動作については、上記の内容を繰り返します。
・全てのルールに該当しない場合は”0”を Esyface-spc へ返します。
・１つの群に対して、Esyface-spc に返すことができる戻り値は、１つだけです。
・標準で準備しているルールでは rule1 が最もアラームレベルが重く、次に rule2 が重くなる仕様となっています。

・標準で作成している rule1～rule8 は JIS の８つのルール１～８の内容を TCL で実装したものです。　各ルールの関数内容は以下の通りとなっています。

・各ルール関数の引数”ｉ”は判定を行う群（データ元のレコード）の Index です。

8.3　各ルール関数の説明

・各ルール関数は戻り値を返します。

・戻り値の値はユーザ側で任意に設定を行いますが、整数型の値のみとします。
　また、この戻り値が Esyface-spc 本体の警報設定画面の rule No.に該当します。

・戻り値の値は各ルール関数と重複して使用しても可能です。各ルールをアラームレベルに分けて、rule No.でルールレベルを識別するイメージです。

8.3.1 Rule1（１点が領域Ａ（３σ）を超えている）

```
proc rule1 {i} {
      set B [ JSUAi $i ]
      set C [ JSLAi $i ]
      set Q [ JSBetweenOver $i 1 $B $C ]
      if { -1 == $Q } {
        return 0
      } else {
        if { 1 == $Q } {
          return 1
        } else {
          return 0
        }
      }
}
```

・”JSUAi”、”JSLAi”、”JSBetweenOver”は Esyface-spc が標準で準備する関数です。各関数の内容は”1 概要”の Esyface-spc のサポート関数を参照して下さい。

・JSBetweenOver 関数で求めた結果を変数”Q”へ格納します。

・関数の実行結果がエラーの場合は、”Ｑ ＝ －１”となり rule1 は戻り値”0”を返します。

・関数の実行結果が”Ｑ ＝ １”の場合（領域Ａを超えている場合）は、rule1 は戻り値”１”を返します。

・領域Ａを超えていない場合は、rule1 は戻り値”0”を返します。

8.3.2 Rule21、Rule22（９点が中心線に対して同じ側にある）

```
proc rule21 {i} {
      set B 9999
      set C [ JSAV $i ]
      set Q [ JSBetweenInside $i 9 $B $C ]
      if { -1 == $Q } {
        return 0
      } else {
        if { 9 == $Q } {
          return 2
        } else {
          return 0
        }
      }
}
proc rule22 {i} {
      set B [ JSAV $i ]
      set C -9999
      set Q [ JSBetweenInside $i 9 $B $C ]
      if { -1 == $Q } {
        return 0
      } else {
        if { 9 == $Q } {
          return 2
        } else {
          return 0
        }
      }
}
```

・”JSAV” 、”JSBetweenInside”は Esyface-spc が標準で準備する関数です。
　各関数の内容は”1 概要”の Esyface-spc のサポート関数を参照して下さい。
・JSBetweenInside 関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果がエラーの場合は、”Q ＝ －１”となり rule2 は戻り値”0”を返します。
・関数の実行結果が”Q＝ 9”の場合（９点が中心線に対して同じ側にある）は、rule21 および rule22 は戻り値”２”を返します。

・ ９点が中心線に対して同じ側にない場合は、rule21（rule22）は戻り値”0”を返します。

8.3.3 Rule3（６点が増加、または減少している）

```
proc rule3 {i} {
        set Q [ JSDirction $i ]
        if { 6 <= $Q } {
          return 3
        } else {
          return 0
        }
}
```

・” JSDirction”はEsyface-spcが標準で準備する関数です。
各関数の内容は”1 概要”のEsyface-spcのサポート関数を参照して下さい。
・JSDirction関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果が”Q”が6以上の場合（６点が増加、または減少している）は、rule3は戻り値”３”を返します。
・６点が増加、または減少していない場合は、rule3は戻り値”0”を返します。

8.3.4 Rule4（１４点が交互に増減している）

```
proc rule4 {i} {
        set Q [ JSRepetition $i ]
        if { 14 <= $Q } {
          return 4
        } else {
          return 0
        }
}
```

・”JSRepetition”はEsyface-spcが標準で準備する関数です。
各関数の内容は”1 概要”のEsyface-spcのサポート関数を参照して下さい。
・JSRepetition関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果が”Q”が14以上の場合（１４点が交互に増減している）は、rule4は戻り値”4”を返します。

・１４点が交互に増減していない場合は、rule4 は戻り値”0”を返します。

8.3.5 Rule5（連続する３点中、２点が領域Ａまたはそれを超えた領域にある）

```
proc rule5 {i} {
        set B [ JSUBi $i ]
        set C [ JSLBi $i ]
        set Q [ JSBetweenOver $i 3 $B $C ]
        if { -1 == $Q } {
          return 0
        } else {
          if { 2 <= $Q } {
            return 5
          } else {
            return 0
          }
        }
}
```

・” JSUBi”、” JSLBi”、” JSBetweenOver”は Esyface-spc が標準で準備する関数です。
各関数の内容は”1 概要”の Esyface-spc のサポート関数を参照して下さい。
・JSBetweenOver 関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果がエラーの場合は、”Q ＝ －１”となり rule5 は戻り値”0”を返します。
・関数の実行結果が”Q”が 2 以上の場合（連続する３点中、２点が領域Ａまたはそれを超えた領域にある）は、rule5 は戻り値”５”を返します。
・連続する３点中、２点が領域Ａまたはそれを超えた領域にない場合は、rule5 は戻り値”0”を返します。

8.3.6 Rule6（連続する５点中、４点が領域Ｂまたはそれを超えた領域にある）

```
proc rule6 {i} {
        set B [ JSUCi $i ]
        set C [ JSLCi $i ]
        set Q [ JSBetweenOver $i 5 $B $C ]
        if { -1 == $Q } {
          return 0
        } else {
          if { 4 <= $Q } {
            return 6
          } else {
            return 0
          }
        }
}
```

・” JSUCi”、” JSLCi”、” JSBetweenOver”は Esyface-spc が標準で準備する関数です。
各関数の内容は”1 概要”の Esyface-spc のサポート関数を参照して下さい。
・JSBetweenOver 関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果がエラーの場合は、”Q ＝ －１”となり rule6 は戻り値”0”を返します。
・関数の実行結果が”Q”が４以上の場合（連続する５点中、４点が領域Ｂまたはそれを超えた領域にある）は、rule6 は戻り値”６”を返します。
・連続する５点中、４点が領域Ｂまたはそれを超えた領域にない場合は、rule6 は戻り値”0”を返します。

8.3.7 Rule7（連続する１５点が領域Ｃにある）

```
proc rule7 {i} {
        set B [ JSUCi $i ]
        set C [ JSLCi $i ]
        set Q [ JSBetweenInside $i 15 $B $C ]
        if { -1 == $Q } {
          return 0
        } else {
          if { 15 == $Q } {
            return 7
          } else {
            return 0
          }
        }
}
```

・”JSUCi”、”JSLCi”、”JSBetweenInside”はEsyface-spcが標準で準備する関数です。
各関数の内容は”1 概要”のEsyface-spcのサポート関数を参照して下さい。
・JSBetweenInside関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果がエラーの場合は、”Q ＝ －１”となりrule6は戻り値”0”を返します。
・関数の実行結果が”Q＝ 15”の場合（連続する１５点が領域Ｃにある）は、rule7は戻り値”７”を返します。
・連続する１５点が領域Ｃにない場合は、rule7は戻り値”0”を返します。

8.3.8 Rule8（連続する８点が領域Ｃを超えた領域にある）

```
proc rule8 {i} {
        set N [ JSUCi $i ]
        set M [ JSLAi $i ]
        set Q [ JSBetweenOver $i 8 $N $M ]
        if { -1 == $Q } {
          return 0
        } else {
          if { 8 == $Q } {
            return 8
          } else {
            return 0
          }
        }
}
```

・”JSUCi”、”JSLCi”、”JSBetweenOver”は Esyface-spc が標準で準備する関数です。
各関数の内容は”1 概要”の Esyface-spc のサポート関数を参照して下さい。
・JSBetweenOver 関数で求めた結果を変数”Q”へ格納します。
・関数の実行結果がエラーの場合は、”Q ＝ －１”となり rule6 は戻り値”0”を返します。
・関数の実行結果が”Q ＝ ８”の場合（連続する８点が領域Ｃを超えた領域にある）は、rule8 は戻り値”８”を返します。
・連続する８点が領域Ｃを超えた領域にない場合は、rule8 は戻り値”0”を返します。

8.3.9 任意規格線を使用したルールの作成例

X 管理図の任意線を超える物についてエラーとする場合

①下記のように Func {i} の中に新たなルール（rule9）を追加します。
基本的にはルール名称（rule9）以外は、他のルールと同じにします。

```
proc Func {i} {
  set A [ rule9 $i ]
  if {$A != 0} {
      return $A
  }
   ・・・・・・
```

②次に追加するルールの内容を実装します。
使用する関数等は、本説明書の関数を参考にして下さい。
以下の内容は、-9999～任意線１の値以外の物については、戻り値（return）として
“9”を代入し、それ以外につては、“０”を代入しています。

```
proc rule9 {i} {
  set N -9999
  set M [ JSFL1 ]
  set Q [ JSBetweenOver $i 1 $N $M ]
  if { -1 == $Q } {
    return 0
  } else {
    if { 1 == $Q } {
      return 9
    } else {
      return 0
    }
  }
}
```

9 TCL 文法

9.1 共通文法

TCL（Tool Command Language）の文法は、コマンドと引数をスペースで区切って並べるコーディングを基本としています。

例

```
command   val1 val2 val3 …
```

一行に複数のコマンドを並べる場合

一行に複数のコマンドを並べる場合には、コマンド間をセミコロンで区切ります。

例

```
command val1 val2 val3 … ;  command val1 val2 val3 …
```

9.2 一行が長くなる場合

一行が長くなる場合には、行末にバックスラッシュ（￥）を置くことで、改行を行いコマンドを継続することができます。

例

```
command val1 ¥
        val2 val3 …
```

9.3 変数

TCL の変数には型がありません。 型を意識せずに文字列や整数を代入できます。

9.3.1 set コマンド

set コマンドは、変数に値を代入します。 Set コマンドで値を省略するか、変数名の頭に“＄”を付けると変数の値を参照することができます。

9.3.2 unset コマンド

例

```
set val 123
val ⇒ 123
set val
val ⇒ 123
set val abc
val ⇒ abc
unset val
```

unset コマンドは、変数を削除します。

9.4　定数

定数

通常の数字(0-9)	10 進数 (例: 123)
'0'で始まる数字(0-7)	8 進数 (例: 077)
'0x'で始まる数字(0-7, a-f)	16 進数 (例: 0xabc)
'.'のある数字(0-9)	浮動少数点数 (例: 7.91e+16)

Boolean 型

True/false yes/no on/off　　真理値

9.5　算術演算

expr コマンドは、整数や浮動小数点数の演算や比較を行います。　三角関数や乱数等も使用することができます。

例

```
expr 4 / 2
⇒ 2
expr 10 + 10
⇒ 20
expr 2.0 ＊ asin(1.0)
⇒ 3.14159265359
set i 1
i ⇒ 1
incr i
⇒ 2
```

9.5.1 演算子と数学関数一覧

演算子

− + ~ !	負符号、正符号、補数、否定
* / %	乗算、除算、剰余
+ −	加算、減算
<< >>	左シフト、右シフト
< > <= >=	左不等、右不等、等価左不等、等価右不等
== !=	等価、非等価
&	ビット積(AND)
^	ビット差(XOR)
|	ビット和(OR)
&&	積結合
||	和結合
x?y:z	条件

関数

acos cos hypot sinhasin() cosh() log() sqrt() atan() exp() log10() tan() atan2() floor() pow() tanh() ceil() fmod() sin()	数学関数
abs(arg)	絶対値
double(arg)	倍精度値
int(arg)	整数値
rand()	乱数値
round(arg)	整数への丸め値
srand(arg)	乱数の初期値

9.6 コマンドの置換

大括弧は、大括弧内のコマンドをコマンドの出力結果で置換します。

例

```
set val [expr 2.0 * 10]
⇒ 20
set val
⇒ 20
```

9.7　ダブルクォートと中括弧

ダブルクォートと中括弧は、複数の文字列を一塊にまとめます。　ただし、ダブルクォート内で変数とコマンドは置換されますが、中括弧内では置換できません。
バックスラッシュを使用すると‘[’と‘$’の置換効果を無効にできます。

例

```
set val 123
⇒ 123
puts "result = $val"
⇒ result = 123
puts {result = $val}
⇒ result = $val
puts "result = ¥$val"
⇒ result = $val
```

```
set val abc
⇒ abc
puts "result = [val]"
⇒ result = abc
puts {result = [val]}
⇒ result = [val]
puts "result = ¥[val]"
⇒ result = [val]
```

9.8　フォーマット

scan と format コマンドは、ANSIC 言語の scan と printf 関数と同様の処理が行えます。　“%”の書式によって文字列をフォーマットします。

例

```
scan "123.456" "%d.%d" a b
⇒ 2
set a
⇒ 123
set b
⇒ 456
format "%d.%d" $a $b
⇒ 123.456
```

9.9　プロシジャ

proc コマンドは、0 個以上の引数を取る関数（プロシジャ）を定義できます。　プロシジャ内で定義された変数は、ローカル変数となり、プロシジャ内でのみ参照できます。
プロシジャ内でグローバル変数を参照するには、global 宣言が必要になります。

例

```
# 普通の引数
set val 3
⇒ 3
proc add {a b} {
    global val
    return [expr $a + $b + $val]
}
add 1 2
⇒ 6
```

9.10　コメント

“#”で始まる行は、コメント行を意味します。　ステートメントの途中からコメントを記述するには、“; #”となる点に注意してください。

例

```
# コメント 1
 # コメント 2 ¥
    コメント 2 の続き
puts {Hello World} ;#コメント 3
⇒ Hello World
```

9.11 制御文

if, for, foreach, while, switch コマンドは、制御文として使えます。 if, for, while の条件式は、expr コマンドと同じ式が使えます。

9.11.1 if, elseif, else 文

例

```
set val 壱
⇒ 壱
if {$val == "壱"} {
    set data 1
}
⇒ 1
```

```
set val 弐
⇒ 弐
if {$val == "壱"} {
    set data 1
} else {
    set data 2
}
⇒ 2
```

```
set val 参
⇒ 参
if {$val == "壱"} {
    set data 1
} elseif {$val == "弐"} {
    set data 2
} else {
    set data 3
}
⇒ 3
```

9.11.2 for，foreach，while 文

for，foreach，while 内では、continue，break コマンドが使用できます。

for 文例

```
for {set i 1} {$i <= 3} {incr i} {
    set data $i
}
⇒ 1
⇒ 2
⇒ 3
```

```
for {set i 1} {$i <= 5} {incr i} {
    if {$i > 3} {
        break
    }
    set data $i
}
⇒ 1
⇒ 2
⇒ 3
```

```
for {set i 1} {$i <= 5} {incr i} {
    if {$i < 3} {
        continue
    }
    set data $i
}
⇒ 3
⇒ 4
⇒ 5
```

foreach 文例

```
foreach i {A B C} {
      set data $i
}
⇒ A
⇒ B
⇒ C
```

```
foreach {i j} {A B C D E F} {
      set data "$i $j"
}
=> A B
=> C D
=> E F
```

switch 文例

```
switch err1 {
err1      {puts 1}
err2      {puts 2}
err3      {puts 3}
default   {puts 不明}
}
⇒ 1
```

```
switch err1 {
err1
          {puts 1 }
err2
          {puts 2}
err3
          {puts 3 }
default
          {puts 不明}
}
⇒ 1
```

※ TCL 参考 URL http://www.interq.or.jp/japan/s-imai/tcltk/

株式会社デジタル 国内販売事業部 ＥＳＳ部

東日本ブロック ： 東京都台東区鳥越1-8-2 鳥越Fビル6Ｆ
TEL.03-5821-1108　FAX.03-5821-2595
中部ブロック ： 名古屋市東区葵3-15-31 住友生命千種ニュータワービル6Ｆ
TEL.052-932-6610　FAX.052-932-6802
近畿ブロック: 大阪市住之江区南港東8-2-52
TEL.06-6613-1741　FAX.06-6613-5888
中四国･九州ブロック: 福岡市博多区博多駅東2-15-19 KS･T駅東ビル6Ｆ
TEL.092-441-5236　FAX.092-441-6032

E-Mail:ess@proface.co.jp　ＵＲＬ:www.proface.co.jp/ess