目次

式及び変数の使用 1
式及び変数の使用: イントロダクション 1
測定ルーチンでの式の使用 1
式の値を閲覧 2
式の値のみを保持 2
分枝を持つ式の使用 2
ファイル入力 / 出力を伴う、式の使用 2
式ビルダによる、複数の式作成 2
式を持つ変数の使用 9
式の構成要素の理解 10
リテラル 11
参照 11
変数 18
ストラクチャー 21
ポインター 23
配列 24
式用の演算子 33
優先順位 34
関数 35
関数一覧 35
スキャン映像から作成された直線フィーチャーの例 73
被演算子型変換 74

ID式 77
レポートのオブジェクト属性にアクセス 80
構築されたｽｷｬﾝ最小円からのｱｸｾｽ情報 83

式及び変数の使用

式及び変数の使用: イントロダクション

式はPC-DMISのフローコントロールコマンドと共に使用されるユーザーによって定義された条件です。フローコントロールステートメントを用いて、これらの条件をお客様の測定ルーチン内でテストすることができ、条件が満たされるか否かに応じて、PC-DMISの対応を決めることができます。

PC-DMISに、お客様特有の作業を行わせる時、式は重要です。フローコントロールコマンドとの連結で式を使用すると、PC-DMISの強力な機能をさらに解き放ち、使用することが可能になります。

本章では、PC-DMISの編集ウィンドウ内での式の作成方法、及び、使用方法について説明します。式を利用する場合、PC-DMISの編集ウィンドウをコマンドモードに配置するべきです。これによって、編集ウィンドウのコードを、直接閲覧できます。

次の主要なトピックは、本章で説明されています。

- 測定ルーチンでの式の使用
- 式ビルダによる、複数の式作成
- 式を持つ変数の使用
- 式の構成要素について
- レポートオブジェクトのプロパティへのアクセス
- 構築されたスキャン最小円から情報へのアクセス

あなたが式をレポート上の情報をお探しの場合は、「レポート測定結果」章の「レポート式の詳細」を参照してください。

測定ルーチンでの式の使用

PC-DMIS編集ウインドウはを編集可能フィールドのほとんどの式を許可します。編集可能なフィールドは通常コマンドモードであなたが編集ウィンドウの中のTABを押す時黄色でハイライトされたフィールドです。フィーチャーのタイプを変えるフィールドは式を許可しません。

例:
自動要素のタイプ、すなわち面上点、自動円、自動丸溝など自動要素のタイプを特定する自動要素のボックスは式の入力を認めません。

この項目の下にある、サブ項目には、利用可能な式についての詳細に渡った説明があります。

式の値を閲覧

式の値を見るためには、マウスのカーソルを式の上に位置付け、少なくとも一秒そこに留まってください。その式の評価が行われ、式とその時点での式の値を表示する、小さい黄色のポップアップウィンドウが、マウス カーソルの下に表示されます。

式の値のみを保持

編集ウィンドウ内の式を即座に解き、これにより、値のみを保持するには:

1. 編集ウィンドウ内で式テキストを選択して下さい。
2. ` (抑音アクセント)符号を、その式テキストに先行して使用して下さい。

例: 式「**`1/7**」を数値フィールドに入力する、とします。直ちに式の解が出され、その値 (**0.143**) のみがフィールド内に配置されます。

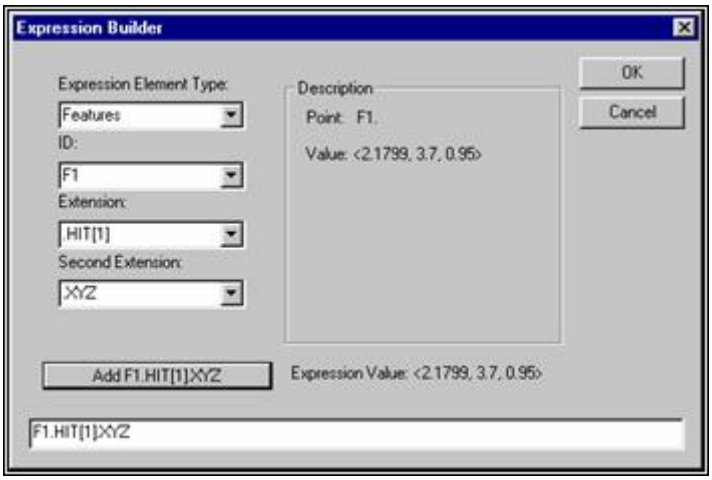
分枝を持つ式の使用

フローコントロール
コマンドはルーチン実行のフローを決めるために式を使用します。分枝の発生について詳しくは、「フローコントロールを使っての分枝」の章を参照して下さい。

ファイル入力 / 出力を伴う、式の使用

外部データ ファイルへのデータの書き込み、または、外部データ
ファイルからのデータの読み込みを行うには、そのデータを効果的に管理保存、または、表示するために、しばしば、変数とその他の式が使用されます。より詳しい説明については、「ファイル入力 / 出力の使用」の章を参照して下さい。

式ビルダによる、複数の式作成



「式ビルダ」ダイアログ ボックス

PC-DMISは、編集ウィンドウに単に入力するか、または **式ビルダ** ダイアログボックスのより使いやすいインタフェースを使用して、式を作成して追加します。**編集 | 式** メニューオプションは**式ビルダ**ダイアログボックスを表示します。

このダイアログボックスでは、式を作成し、それを編集可能なフィールドに挿入することができます。カーソルが式を受け入れるフィールド上にある時、F2キーを押すと**式ビルダ**ダイアログボックスが表示されます。

式ビルダ
ダイアログには、式で利用可能なすべてのタイプの演算子と関数の一覧が表示されます。

タイプ入力による式作成

編集ウィンドウ内に直接タイプ入力して、式を作成するには:

1. 編集ウィンドウを開いて下さい (**ビュー | 編集ウィンドウ**)。
2. 編集ウィンドウをコマンド モードにして下さい。
3. 式を挿入する予定の、編集可能なフィールドにカーソルを移動するには、タブキーを押して下さい。黄色でハイライト表示されたフィールドが、「編集可能」とみなされます。
4. 式をタイプ入力して下さい。

式ビルダによる式作成

注記:式オプションを有効にするには、コマンドモードにある必要があります。

式ビルダダイアログ ボックスを用いて、式を入力するには(**編集 | 式**):

1. 編集ウィンドウを開いて下さい (**ビュー | 編集ウィンドウ**)。
2. 編集ウィンドウをコマンド モードにして下さい (**ビュー | コマンド モード**）。
3. 式を挿入する予定の、編集可能なフィールドにカーソルを移動して下さい。
4. カーソルが式を受け入れるフィールド上にある間にF2キーを押して下さい。**式ビルダ**ダイアログ ボックスが開きます。**式ビルダ** ダイアログ ボックス
   には、演算子、被演算子、および関数の一覧が表示されます。以下は、このダイアログ ボックスを通して参照が可能です:
   - 利用可能な式のタイプ
   - 変数
   - フィーチャー
   - 測定結果
   - パーツ配置
   - コメント
5. 最初のドロップダウン一覧から、式要素のタイプを選択して下さい。お客様の選択次第では、その他のコンボ ボックスが表示されます。
6. **ID** ドロップダウン一覧から、ご希望のIDを選択して下さい。
7. **拡張子** ドロップダウン一覧から、拡張子を選択して下さい。
8. **第二拡張子**ドロップダウン一覧から、もうひとつの拡張子を選択して下さい。その式が使用可能な場合、**追加**ボタンが利用可能になります。
9. **追加**ボタンをクリックして下さい。その式が、編集ボックス内に表示されます。
10. **OK**ボタンをクリックします。その式が、編集ボックス内のカーソルの所在場所に表示されます。

注記: また、以下の、その他のダイアログ ボックスから**式ビルダ** ダイアログ ボックスを開くことができます:

- **If Expression** ダイアログ ボックス - **挿入 | フローコントロール | If Goto**を選択して下さい。**式** ボタンをクリックして下さい。

- **割り当て** ダイアログ ボックス -**挿入 |**
  割り当てを選択して下さい。**代入先**、または、**代入元**ボタンをクリックして下さい 。

いったん入力式が作成されると、PC-DMISは、編集ウィンドウ内の次の正当な位置に、自動的に入力式を挿入します。

入力式の正確性をチェック

式を追加したフィールドからカーソルが離れると、PC-DMISは式の正確さチェックしようとします。式に問題があれば、無効な番号を示すエラーメッセージを表示するか式のテキストが赤色に変わります。また、また、存在しないオブジェクトを参照する式も赤文字で表示されます。

フィールドを離れる時に式の正確性のテストが発生したので、存在しないオブジェクトの参照のためにフィールドが赤くなり（例：CIRCLE1.X）、新しいオブジェクトは（例：CIRCLE1）追加されても赤いままになります。式が正しくに再テストされるまで、フィールドは赤いままです。

式の正確性を再検査するには:

1. その式のフィールドにカーソルを移動して下さい。
2. F2 キーを押して下さい。**式ビルダ**ダイアログ
   ボックスが、再び開きます。その式に対する変更すべてが、編集ボックス内に表示されます。
3. 「入力」キーを押して、当ダイアログ ボックスを閉じて下さい。

式要素のタイプ

式ビルダダイアログボックス (**編集 |**
式)にある式要素のタイプドロップダウン一覧は、式に配置できる、各種の式要素のタイプを一覧表示します。以下が含まれます。

- 関数
- 演算子
- パーツ配置
- コメント
- 測定寸法
- フィーチャー

- 変数

ID

式ビルダダイアログ ボックス (**編集 | 式**)にあるIDドロップダウン一覧には、式要素タイプドロップダウン一覧で選択された式要素タイプに基づき、利用可能な一連の項目が一覧表示されています。

例:

- **関数及び演算子**が**式要素タイプ** ドロップダウン一覧から選択された場合、**ID** ドロップダウン一覧は、利用可能な関数と演算子の一覧を含みます。
- 要素が**式要素タイプ**ドロップダウン一覧から選択された場合、**ID**ドロップダウン一覧は、測定ルーチン内の要素IDを表示します。

拡張子

式ビルダーダイアログボックス (**編集 | 式**)の**[拡張機能]**ドロップダウンリストは「**ID**」ドロップダウンリストで選択したアイテムが使用可能な式エレメントを形成するために、拡張機能の追加を必要とするときに利用可能になります。**拡張子**ドロップダウンリストは、IDドロップダウン一覧で選択された項目に基づいて、利用可能な拡張子を表示します。

例:
IDドロップダウンリストから要素を選択すると仮定します。そのフィーチャーの参照データ(例えば、「X」、「Y」、「Z」、「Diam」、「Length」、等)に使用されうる拡張子は、**拡張子**ドロップダウン一覧内に表示されます。

可能な拡張子には、以下の測定された、または、理論的データ タイプを含まれます:

- すべて – 要素のすべての値は変数に割り当てられます。以下の例を参照してください。
- X – 測定されたヒットのX値
- Y – 測定されたヒットのY値
- Z – 測定されたヒットのZ値
- XYZ – 測定されたヒットの XYZ 値
- I – 測定された ヒットのI 値

- J – 測定されたヒットのJ 値
- K – 測定されたヒットのK 値
- IJK – 測定されたヒットのIJK 値

- TX – 理論的なヒットのX 値
- TY – 理論的なヒットのY 値
- TZ – 理論的なヒットのZ 値
- TXYZ – 理論的なヒットのXYZ 値
- TI – 理論的なヒットのI 値
- TJ – 理論的なヒットのJ 値
- TK – 理論的なヒットのK値
- TIJK – 理論的なヒットのIJK値

例:

```
F1=GENERIC/POINT,DEPENDENT,CARTESIAN,$
      NOM/XYZ,<8,9,10>,$
      MEAS/XYZ,<7.98,8.98,9.98>,$
      NOM/IJK,<1,0,0>,$
      MEAS/IJK,<1,0,0>
      ASSIGN/MYFEATURE = F1.ALL
      ASSIGN/V1 = MYFEATURE.X
      ASSIGN/V2 = MYFEATURE.TX
```

V1 は7.98の値を含みます。

V2 は8の値を含みます。

第二拡張子

拡張子ドロップダウン一覧で選択された項目が、使用可能な式要素を構成するために、第二拡張子の追加を必要とする時のみ、**第二拡張子**ドロップダウン一覧が利用可能になります。

例: 「D1」という名前の測定のX軸の位置の公称値を参照することを仮定します。
ユーザを次のステップをします：

1. **ユーザーはID ドロップダウン一覧からD1を選択します。**
2. **拡張子ドロップダウン一覧から、Xを選択して下さい。**
3. **二番目の拡張子ドロップダウン一覧から、Nomを選択して下さい。**

追加ボタン

ドロップダウンリストを用いて、使用可能な、または、完成した式要素を選択する時には、常に、**追加**
ボタンが利用可能になります。このボタンをクリックすると、その式に追加するテキストが表示されます。

例えば、以下が選択された場合:

- **式要素のタイプ**一覧にある測定結果
- **ID** リストからD1
- **延長端子** リストからX
- 第二延長端子リストから公称値

その後、**追加**ボタンが有効化され、以下のテキストを持ちます: **Add D1.X.NOM**。

追加 ボタンをクリックすると、その後、そのテキストがダイアログ
ボックスの底部にある編集ボックス内に表示されます。

注記:
OKボタンをクリックすると、編集ボックスにあるテキストが編集ウィンドウの、カーソルが位置する式フィールドに追加されます。編集ウィンドウの式フィールドから項目を選び、追加されるテキストが丸かっこを含む場合、選択された項目は、追加テキストの丸かっこ内に配置されます。

編集ボックス

式ビルダ ダイアログ
ボックスの一番下には、その時点での式を表示する編集ボックスがあります。式を、直接このボックスに入力するか、または、**追加**ボタンを用いて入力することが可能です。

内容説明エリア

式ビルダダイアログボックス(**編集 | 式**)には、**説明**
エリアも含まれ、それを用いて、ドロップダウン一覧から選択された項目についての情報を入手できます。**追加**ボタンに隣接したフィールドは、式のその時点での値も表示します。

注記: 無効の式は、0の値を持ちます。

式を持つ変数の使用

変数は、値を持つオブジェクトです。変数とは、整数型、実数型、文字列型、または、ポイント型被演算子のことです。変数は、式の使用に不可欠です。変数は、名称と値を持ちます。その名称は、変数の値にアクセスするために使用されます。変数名は不変であり、変数値は変更可能です。ASSIGN/ コマンドを用いて、変数に値を割り当てます。

例えば、ステートメントASSIGN/V1 = 2は、V1という名と、値2を持つ変数を作成します。ASSIGN/V2 = V1 + 2はV1の値にアクセスします。実行されたときV1がまだ2の数値を持つならば、V2は4の数値を持ちます。

変数に関する、より詳しい説明については、「変数」を参照して下さい。

「割り当て」ダイアログ ボックス

挿入 | 代入 メニューオプションが
値の代入ダイアログボックスを表示します。このダイアログボックスを用いて、要素、寸法、またはアラインメントの変数またはデータの要素に値を代入できます。代入コマンドの使用には、PC-DMISでの式の基本的理解が要求されます。

> **割り当て先**ボタン -
> このボタンは、**割り当て元**ボックスから算出された値を受信している変数を指定するために使用される。情**代入先**ボタンを使用して選択した情報が**代入先**
> ボックスに配置されます。この値は変数名、あるいは要素、測定結果、あるいはアラインメントのデータ要素の参照先になります。
>
> *「評価を出す」という語句は、数学式の結果となる値を出すことを意味します。*

割り当て元ボタン -
このボタンは、**割り当て元**ボックスに割り当てられた値を格納することに使用されます。このボックスが式を含む場合、実行時にその式の評価が出され、その計算結果や数値が、**割り当て先** ボックスで指定されたオブジェクトに割り当てられます。

挿入ボタン - このボタンは**割り当て**ダイアログ
ボックスが開いているときに測定ルーチンに割り当てコマンドを挿入します。割り当て一連のコマンドは、ダイアログボックスを閉じずに挿入することができます。

式の構成要素の理解

式は、以下のタイプの被演算子を持ちます:

- 整数
- 実数
- 文字列
- 点
- フィーチャー ポインター
- 配列
- 関数

以下の項目が、下記で詳しく記述されています。

被演算子のタイプ

被演算子は、以下の形式で存在する可能性があります:

- リテラル
- 参照
- 変数
- 構造
- ポインタ
- 配列

リテラル

***整数:** 1, -6, 209

実数: 1, -6, 2.4, -0.1, 345.6789

文字列: "Hello World", "47", "CIRCLE 1"

点:
ポイントについては、リテラル表示はありません。ただし、ポイントは、MPOINT関数を用いて、他のリテラルから作ることが可能です: 例として`MPOINT(0,0,1)`, `MPOINT(2.2, 3.1, 4.0)`があります。

ポインター: 中括弧に囲まれた要素名: `{CIR1}, {LIN2}, {F3}`

配列:
配列については、リテラル表示はありません。しかし、配列はARRAY関数を用いて、他のリテラルから作ることが可能です: 例として `ARRAY(3, 5, 6)`, `ARRAY("Hello", 2.3, 9)`などです。これらの関数は最初の例と文字列の要素 「Hello」に整数 要素3、5、6 で3つの要素配列、2個の要素2.3、2番目の例の整数要素9を作成します。

関数: 関数については、リテラルの例示はありません。関数は、`FUNCTION`
というキーワードを用いて定義され、変数IDを通してアクセスされます。例えば、`ASSIGN/Add2 = FUNCTION((X), X+2)`
は、ひとつの引数を取り、それに2を加える関数を定義します。変数Add2が、その関数に割り当てられます。次のように、当関数は変数Add2を用いて呼び出しが可能です。`ASSIGN/Result = Add2(5)`。結果には7の値が割り当てられます。

注記:
演算子や関数は整数の使用に暗示しない限り、数値リテラルは実際の数値として解釈されます。たとえば、式10/8は1の代わりに1.25まで評価します。
また、その不連続除数もオペランド被演算子型強制で可能であることに注意してください。 式INT(10) / INT(8)は1と評価を出します。

参照

参照は測定ルーチンで他のオブジェクトのデータメンバーを示します。参照はドットおよびオブジェクトのデータメンバーを参照する拡張子の前の測定ルーチンでオブジェクトのIDを使用します。

例: CIRCLE1 が測定ルーチン内の測定された円の名称である場合、`CIRCLE1.X`
はCIRCLE1のX構成要素の測定値を参照します。すべての参照はパーツ座標の現在のアラインメントの相対要素に評価されます。

ダブル型の参照

以下の参照式が利用可能です:

ダブル型のフィーチャー参照のための、有効な拡張子の例

形式: <要素ID>.<拡張子> -> `CIRCLE1.X`

CIRCLE1.X CIRCLE1のXの測定値
CIRCLE1.Y CIRCLE1のYの測定値
CIRCLE1.Z CIRCLE1のZの測定値

CIRCLE1.TX CIRCLE1のXの理論値(公称値)
CIRCLE1.TY CIRCLE1のYの理論値(公称値)
CIRCLE1.TZ CIRCLE1のZの理論値(公称値)

LINE1.SX LINE1の開始点の測定されたX値
LINE1.SY
LINE1.SZ

LINE1.TSX LINE1の開始点の理論的X値
LINE1.TSY
LINE1.TSZ

LINE1.EX LINE1の終了点の測定されたX値
LINE1.EY
LINE1.EZ

LINE1.TEX LINE1の終了点の理論的X値
LINE1.TEY
LINE1.TEZ

POINT.I 測定値 I POINT用のベクトル構成要素
POINT.J
POINT.K

POINT.TI 理論値 I 点のベクトル構成要素
POINT.TJ
POINT.TK

FEAT1.TYP 要素のタイプ
(円、溝、円錐など)。これを用いて、一般要素のタイプを変更することが可能です (Assign/Gen1.TYP = Feat1.TYP)。

FEAT1.ALL
すべての要素を参照します。これは、一般要素用に情報をコピーする場合、役に立ちます
。(Assign/ Gen1.ALL = Feat1.ALL)

平面ベクトル

EDGE.SURFI

EDGE.SURFJ

EDGE.SURFK

EDGE.TSURFI

EDGE.TSURFJ

EDGE.TSURFK

角度ベクトル

CIR.ANGI

CIR.ANGJ

CIR.ANGK

CIR.TANGI

CIR.TANGJ

CIR.TANGK

半径

CIRCLE1.R

CIRCLE1.TR

CIRCLE1.RAD

CIRCLE1.TRAD

CIRCLE1.RADIUS

CIRCLE1.PR – 極半径

CIRCLE1.TPR – 理論的な極半径

CIRCLE1.TRADIUS (第一文字のみ重要)

直径

CIRCLE1.D

CIRCLE1.TD

CIRCLE1.DIAM

CIRCLE1.TDIAM

CIRCLE1.DIAMETER

CIRCLE1.TDIAMETER (第一文字のみ重要)

角度

CONE.A

CONE.TA

CONE.ANG

CONE.TANG

CONE.ANGLE

CONE.TANGLE

CONE.PA – 極角度

CONE.TPA – 理論的な極角度(最初の一文字のみ重要)

長さ

LINE.L

LINE.TL

LINE.LEN

LINE.TLEN

LINE.LENGTH

LINE.TLENGTH (最初の1文字のみ重要)

高さ

CYLINDER.PH – 極高さ

CYLINDER.TPH – 理論的な極高さ

半径、角度、高さ

POINT.RAH – 測定された半径、角度、および高さを伴う点

POINT.TRAH – 理論的な半径、角度、および高さを伴う点

領域

BLOB1.AREA - Blob要素の測定された領域の値を返します。

BLOB1.TAREA - Blob要素の理論的な領域の値を返します。

例えば、`ASSIGN/V1=BLOB1.AREA`
はBLOB1要素の測定された領域の値を返し、V1変数に割り当てます。

現在、Blob要素はこれらの領域の拡張子のみで機能します。Blob要素に関する情報は、「PC-DMIS Vision」文書の「Vision Blob」トピックを参照してください。

ダブル型の寸法参照のための、有効な拡張子の例

フォーマット: <測定結果 ID>.<軸>.<測定結果要素> -> `DIM1.X.NOM`

DIM1.X.NOM	DIM1のX 軸位置の公称値
DIM1.X.MEAS	DIM1のX 軸位置の測定値
DIM1.X.MAX	DIM1のX軸の位置の最大偏差
DIM1.X.MIN	DIM1のX軸の位置の最小偏差
DIM1.X.PTOL	DIM1のX軸の位置の正公差値
DIM1.X.MTOL	DIM1のX軸の位置の負公差値
DIM1.X.DEV	DIM1のX軸の位置の偏差
DIM1.X.OUTTOL	DIM1のX軸の位置の公差域外値
DIM1.Y.NOM	DIM1のY軸の位置の名目値
DIM1.Z.DEV	DIM1のZ軸の位置の偏差
DIM3.PA.MEAS	DIM3の極角位置の測定値
DIM4.M.PTOL	DIM4のM軸の正公差値
*DIM4.PTOL	DIM4のM軸の正公差値 (下記の「有効な軸」の下にある注記を参照して下さい)。
DIM5.BTOL	DIM5がポジションであるボーナス許容値。

有効な軸:

X, Y, Z, D, R, A, T, V, L, PR, PA, M, PD, RS, RT, S, H, DD, DF, TP

注記: * 定義によって、軸をひとつしか持たない測定結果
(つまり、真円度、偏心度等)は、軸限定子を省略できます。軸定子が使用される場合、これらのタイプの全寸法（軸をひとつしか持たない）は、M軸限定子を用いますが、A軸限定子を用いる2D、及び、3D角度は除くことに、注意して下さい。

ダブル型のパーツ配置参照のための、有効な拡張子の例:

フォーマット:
<アラインメントID>.<アラインメント軸または原点>.<アラインメントまたは原点の要素> ->
A1.ORIGIN.X

A1.ORIGIN.X	パーツ配置 A1 の測定された原点のX要素
A2.ORIGIN.Y	パーツ配置 A2 の測定された原点のY要素
A1.ORIGIN.Z	パーツ配置 A1 の測定された原点のZ要素
A1.XAXIS.I	パーツ配置 A1 の測定されたX軸の I 要素
A1.YAXIS.J	パーツ配置 A1 の測定されたY軸の J 要素
A1.ZAXIS.K	パーツ配置 A1 の測定されたZ軸の K 要素
A1.CORIGIN.X	理論的（CはCADを示す）データに基づいた、パーツX配置 A1の原点のX要素
A1.CXAXIS.J	理論的（CはCADを示す）データに基づいた、パーツX配置 A1の原点のJ要素

ポイント型の参照

以下の参照式が利用可能です:

ポイント型のフィーチャー参照のための、有効な拡張子の例

フォーマット: <要素ID>.<拡張子> -> CIRCLE1.XYZ

CIRCLE1.XYZ	CIRCLE1の測定された重心
CIRCLE1.TXYZ	CIRCLE1の理論的重心
LINE1.SXYZ	LINE1の測定された開始点

LINE1.TSXYZ	LINE1の理論的開始点
LINE1.EXYZ	LINE1の測定された終了点
LINE1.TEXYZ	LINE1の理論的終了点
CIRCLE1.IJK	Circle1の測定されたベクトル
CIRCLE1.TIJK	CIRCLE1の理論的ベクトル
EDGE.SURFIJK	EDGEの測定された平面ベクトル
EDGE.TSURFIJK	EDGEの理論的平面ベクトル
AUTOCIR1.ANGIJK	AUTOCIR1の測定された角度ベクトル
AUTOCIR1.TANGIJK	AUTOCIR1の理論的角度ベクトル

ポイント型のパーツ配置参照のための、有効な拡張子の例

フォーマット: <アラインメントID>.<アラインメント軸または原点> -> A1.XAXIS

A1.ORIGIN	パーツ配置 A1 の測定された原点
A1.XAXIS	パーツ配置 A1 の測定されたX軸
A1.YAXIS	パーツ配置 A1 の測定されたY軸
A1.ZAXIS	パーツ配置 A1 の測定されたZ軸
A1.CORIGIN	パーツ配置 A1 の理論的原点
A1.CXAXIS	パーツ配置 A1 の理論的X軸
A1.CYAXIS	パーツ配置 A1 の理論的Y軸
A1.CZAXIS	パーツ配置 A1 の理論的Z軸

文字列型の参照

コメントへの参照はタイプ文字列である唯一のオブジェクトタイプです。インプットまたはYES/NOコメントだけは参照によって参照することができます。これらのコメントのタイプにコメントを識別するのに使用することができるIDがあります。

フォーマット: <コメントID>.INPUT -> `C1.INPUT`

C1.INPUT - コメントC1用の（オペレーターからの）入力値

YES/NO コメント タイプは、その時点でのPC-DMISの言語に基づいた、適切な「はい」、または、「いいえ」の文字列を、その入力内容に設定します。英語版のPC-DMISにおいては、オペレーターが「はい」ボタンを押すと、文字列が「はい」に設定され、オペレーターが「いいえ」ボタンを押すと、文字列が「いいえ」に設定されます。「はい」、または、「いいえ」の文字列を検査比較する場合、その比較は大文字と小文字の区別を行います。したがって、YES/NOコメント入力が大文字の「はい」、または、「いいえ」に設定されている場合、小文字の「はい」、または、「いいえ」の比較は、常に失敗似終わります。

変数

変数は、7つのオペランドタイプ、整数、実数、文字列、点、要素ポインター、配列または関数のどれを取ることも可能です。変数は`ASSIGN`ステートメントを通して生成され、値とタイプを受け取ります。変数IDは、数値の文字で始まらない任意の英数字を取ることができます。最初の文字でない限り、変数IDの中で下線も使用できます。

変数値は測定ルーチン実行中に保存されます。これは、ルーチン実行が停止して再始動すると、実行停止時の変数値がプログラム再始動時の値になることを意味します。

注記:
編集ウィンドウがアクティブな状態である場合、カーソルがそのフィールドに配置されると、常にPC-DMISが、変数のその時点での値を表示します。パーツプログラム実行中、変数値は、実行フローに基づいて変化します。その時点での変数値を知るには、ご希望の変数上に、マウスを置いて下さい。

```
ASSIGN/ V1 = 2.2+2
```
変数 V1 は、値 4.2を持つ実数です。

```
ASSIGN/ VAR1 = CIRCLE1.X
```
変数 VAR1 は実数であり、割り当ての時点でのCIRCLE1.X の測定値に等しい値を持ちます。

```
ASSIGN/ MYVAR = LINE1.XYZ
```
変数 MYVAR は点であり、割り当ての時点での LINE1 の測定された重心と同一の値を持ちます。

```
ASSIGN/SVAR = "Hello World"
```
変数SVARは「Hello World」という値を持つ文字列です。

これらの例では、 変数
は、割り当てられた値です。いったん変数の値が割り当てられると、その変数は、どのような式フィールドにおいても、被演算子として使用が可能です。

数値フィールドで使用される V1 の例:

```
ASSIGN/V1 = 1/3
PREHIT / V1
```

ここで、 V1 は、prehit コマンドのヒット前の値として用いられます。

注記:
式は、ほとんどの編集可能なフィールドで使用が可能なので、以下の式はまた、正当でもあり、以下と同様の効果を持ちます: PREHIT / 1/3。

ポイント型変数の要素は、参照に使用される、ドット拡張子を用いて、個々に参照が可能です。

```
ASSIGN/ V1 = MPOINT(3, 4, 5)
```
V1は3、4、5の値を備えた点のタイプです。

```
ASSIGN/ XVAR = V1.X
```
XVARは3の値を持つ倍精度型です。

```
ASSIGN/ YVAR = V1.Y
```
YVARは4の値を持つ倍精度型です。

```
ASSIGN/ IVAR = V1.I
```
IVARは3の値を持つ倍精度型です。

```
ASSIGN/ REDUNVAR = V1.XYZ
```
REDUNVARは3、4、5の値を持つポイント型です。

以下の拡張子は互いに同等です。両方とも測定ルーチン内の式の意味を明確化するために付いています。

V1 がポイント型である、とします。

V1.XはV1.Iと同じです。
V1.YはV1.Jと同じです。
V1.ZはV1.Kと同じです。

V1.XYZはV1.IJKおよび拡張子の付かないV1と同じです。

文字列型の変数が、フィーチャーID名、寸法ID名、パーツ配置ID名と等しい文字列値を持つ場合、その変数を参照のオブジェクトとして使用することが可能です。

例:

```
ASSIGN/ V1 = "CIRCLE1"
```

CIRCLE1という名称のフィーチャーが存在する場合、以下の被演算子は可能で有効です。

V1.X - CIRCLE1の測定されたX値
V1.TX - CIRCLE1の測定されたX値
V1.Diameter - CIRCLE1の測定された直径
V1.Radius - CIRCLE1の測定された半径

文字列変数で利用可能な、この種の間接的方法は、ひとつのレベルにおいてのみ利用可能です。以下は機能しません。

```
ASSIGN/ V1 = "CIRCLE1"
ASSIGN/ V2 = "V1"
```

V2.X - これはCIRCLE1.Xの現在の測定値の代わりに、0に評価されます。

注記: 参照 V2.X
は、これより上の式によって文字列型と設定されたとしても、エラーとして赤テキストでフラグ*されません*。エラーとしてフラグを立てられない理由は、実行時間になるまでは測定ルーチンの実行フローが分からないからです。エラーとしてフラグを立てられない理由は、実行時間になるまでは測定ルーチンの実行フローが分からないからです。

しかし、波状カッコを用いると、以下は機能します:

```
ASSIGN/ V1 = {CIRCLE1}
ASSIGN/ V2 = {V1}
```

V2.X - これは、CIRCLE1.Xの値を指定します。

以下の例を参考にして下さい:

```
ASSIGN/ V1 = "CIRCLE1"
ASSIGN/ V2 = "V1"
IF / CIRCLE1.X > CIRCLE1.TX, GOTO, L2
L1 = LABEL /
ASSIGN/ V3 = V2.X
GOTO / LABEL, L3
L2 = LABEL /
ASSIGN/ V2 = MPOINT( 2, 5, 7)
GOTO / LABEL, L1
L3 = LABEL /
```

測定ルーチン実行中、CIRCLE1.X の値がCIRCLE1.TXの値より大きい場合、式 V2.X
が有効で、それは2に評価します。それ以外の場合には、V3の「割り当て」の時点での V2

の値が文字列「V1」であるため、式 V2.X
は0と評価されます。パーツプログラマーはこれらの場合式が期待通り確実に実行されるようにする責任があります。

割り当てステートメントに関するその他の注記: ほぼすべての参照要素は、要素の測定されたデータまたは理論的なデータのメンバーに数値を入れるために、割り当てステートメントの左側で使用されます。唯一の例外は単一I、J、Kのベクトルコンポーネントです。ベクトルに割り当てるためには、完成したベクトルはポイントに評価する式をすぐに使用して割り当てられなければなりません。ベクトルデータは要素のベクトルデータメンバーに入ると同時に正規化されます。

例:

```
ASSIGN/ CIRCLE1.I = 2 - 無効
ASSIGN/ CIRCLE1.IJK = MPOINT(2, 0, 0)-有効 (ベクトルは 1, 0, 0 に正規化)
```

寸法測定での変数の使用に関する情報については、「要素の寸法測定」章の「変数の寸法測定」トピックを参照して下さい。

ストラクチャー

ユーザは、その変数のサブエレメントを識別するために変数に拡張子を配置する*構造*と呼ばれる変数型を使用することができます。この例を参考にして下さい:

例:

```
ASSIGN/V1.HEIGHT = 6
ASSIGN/V1.WIDTH = 4.3
ASSIGN/V1.MODE = "CIRCULAR"
ASSIGN/V1.POINT = MPOINT(100.3, 37.5, 63.1)
```

この例では、

- V1が構造です。
- HEIGHT、WIDTH、MODE、およびPOINT はストラクチャーのサブ要素です。

ストラクチャー用のルール

- 変数と同じように、ストラクチャーでは実行命令を設定する必要はありません。
- ストラクチャーのサブ要素は、以下の変数タイプのうち、いずれでもありえます:
    - 整数
    - ダブル
    - 点

- 要素ポインター
- 関数
- 配列
- 構造

例えば、配列であるストラクチャー要素、及び、ストラクチャーである配列要素を持つことが可能です。これは、以下の式の見本を有効なものにします:

例

```
ASSIGN/CAR.LEFTSIDE.DOOR[2].QUADRANT[3].JOINT[5].HIT [4] = MPOINT(558.89,
910.12, 42.45)

COMMENT/OPER,"現在の Z 位置: " +
CAR.LEFTSIDE.DOOR[2].QUADRANT[3].JOINT[5].HIT[4].Z

ASSIGN/CURRENTJOINT = LEFTSIDE.DOOR[2].QUADRANT[3].JOINT[5]

COMMENT/OPER,"Next Hit: " + CURRENTJOINT.HIT[4]
```

ポイント型の変数を伴うストラクチャー

変数がポイント型である場合、ユーザーは、依然 .X、.Y、.Z、.I、.J、及び、.K
拡張子を用いて、そのポイントの個々の項目を知ることが可能です。また、ユーザーは、この例で使用されている拡張子を、ポイント要素としての使用を強制されることなく、それらのストラクチャー内で用いることができます。

例:

```
ASSIGN/V1.X = "Some string"
ASSIGN/V1.Y = ARRAY(1,3,5,9,7)
ASSIGN/V1.Z = MPOINT(3,5,7)
```

`COMMENT/REPT,V1.X`	出力内容は「`Some string`」です。
`COMMENT/REPT,V1.Y[2]`	出力内容は 3 で、これは、配列の2番目の要素です。
`COMMENT/REPT,V1.Z.Y`	出力内容は 5で、これは、MPOINTのY値です。

PC-
DMIS式言語の関数能力とストラクチャーを組み合わせることにより、ここで表示されたような、動的なストラクチャーへの参照を行うことが可能です:

(上記の例の続き)

```
ASSIGN/DYNAMICSTRUCT = FUNCTION((X,Y), X.Y)
C1 =COMMENT/INPUT,Please enter in item
ASSIGN/TESTSTR = C1.INPUT
ASSIGN/FRONT = LEFT(TESTSTR, INDEX(TESTSTR, ".")-1)
ASSIGN/BACK = MID(TESTSTR, INDEX(TESTSTR, "."))
ASSIGN/RESULT = DYNAMICSTRUCT(FRONT, BACK)
```

例のこの部分では、変数の参照を入力し、その参照を最初の「DYNAMICSTRUCT」関数を使用して基準のRESULTに割り当てます。

つまり、C1.INPUT変数のためにV1.Y[4]が入力されると、RESULTは、9の値を持つことになります (V1.Yに割り当てられた、配列の第四要素)。

学習時の式評価は、ストラクチャー、または、配列の全要素を正確に表示するために、強化されています。

ポインター

*ポインターは、また、「フィーチャー
ポインター」としても知られます。より詳しい情報については、用語集の用語「フィーチャー
ポインター」を参照して下さい。*

ポインタは簡単な方法を提供して、変数を介する機能を参照でき、またはコールサブコマンドを使用してオブジェクトをパスできます。ポインタは文字列名で間接指定と同様です。ただし、ポインタを使用する利点はサブルーチンです。文字列と異なり、ポインタはサブルーチンの引数としてパスした場合に、サブルーチンで指されるオブジェクトの直接の修正を考慮してください。ポインタは複雑な
入力式で使用されていません。それが複雑な入力式において使われるならば、ポインターはゼロまで評価します。

以下の例を参考にして下さい:

ポインター使用例:

- ASSIGN/ V1 = {CIR1}

  ここで、V1 は、CIR1を指すポインターです。

- ASSIGN/ DIST = DOUBLE(V1.XYZ)

  Dist = 原点から CIR1 への距離。

また、フィーチャー
ポインターを得るために、波状カッコ内に式を入れることもできます。ここでの、以下の例は、すべて、フィーチャーCIR1にポインターを得る、正当な方法です:

- `ASSIGN/FEATCOUNT = 1`

  `ASSIGN/V1={"CIR" + FEATCOUNT}`

  式「CIR1」をV1に割り当てます。

- `ASSIGN/V2="CIR1"`

  `ASSIGN/V3={V2}`

  変数V2からの式「CIR1」を変数V3に割り当てます。

- `C1=COMMENT/INPUT, 要素名を入力して下さい。`

  `ASSIGN/V4={C1.INPUT}`

  これは、要素名C1.INPUTを取り、それを変数V4内に置きます。

サブルーチンの例:

呼び出しルーチンでは:

```
CS1 = CALLSUB/SUB.PRG,CHANGEX, {CIR1}
```

サブルーチンでは:

```
GEN1 = GENERIC/FEATURE
SUBROUTINE/CHANGEX, ARG1 = {GEN1}
```

(CIR1 が渡された時、それは、GEN1に置き換えられます)

```
ARG1.X = 5
```

(CIR1の測定されたX値を、5に設定)

```
END/SUBROUTINE
```

複雑な式の例

`ASSIGN/ V1 = {CIR1} + 2`

{CIR1}はゼロと評価されるので、この式全体では、2と評価が出ます。

配列

3つのタイプの配列が利用可能です: フィーチャーの配列、ヒットの配列、及び、変数の配列。

重要事項:
当ソフトウェア内で、多次元配列が複数次元として表示されていても、その配列の前にARRAY INDICESコマンドを配置するまでは、単一次元配列としてのみ本当の使用が可能です
(「配列インデックスオブジェクト:」の項目を参照して下さい)。

要素の配列

フィーチャーがルーチン実行中に、二度以上測られるとき、フィーチャー配列は自動的につくられます。フィーチャー配列の要素の数は、フィーチャーが実行された回数と等しくなります。

例:
測定円オブジェクトが5回実行できるループの中に位置していた場合、5個の測定円の配列が存在します。測定円のidが「Circle1」である場合には、その配列の式が測定円オブジェクトの個々のインスタンスへのアクセスに使用されます。目的のインスタンスを示すために角カッコが使用されます。

```
Assign/V1 = Circle1[3].X
```

V1 は、円1の3番目のインスタンスの測定されたX値に割り当てられます。

注記:
与えられた要素について、要素配列が存在する場合、配列の表記は、要素への参照では使用されず、直前の実例は使用されます。上記の例では、5番目のインスタンスがそのオブジェクトの最も最近のインスタンスなので、`Circle1.x` への参照は `Circle1[5].x` への参照と同様の意味を持ちます。

配列式の角カッコ内で、式を使用することができます。したがって、`Circle1[3].x` は、`Circle1[2+1].x` に相当します。以下の例では、上記の例の5つの円の測定された重心を印刷するために、ループを使用します。

例:

```
Assign/V1 = 1
While/V1 < 6
Comment/Rept, "Centroid of instance #" + V1 + " of Circle1: " +
Circle1[V1].XYZ
Assign/V1 = V1 + 1
End/While
```

上記の例からの、可能な出力内容:

Centroid of instance #1 of Circle1: 3.4, 2.6, 1.43

Centroid of instance #2 of Circle1: 4.4, 3.6, 2.43

Circle1の事例 #3 の重心: 5.4, 4.6, 3.43

Circle1の事例 #4 の重心: 6.4, 5.6, 4.43

Centroid of instance #5 of Circle1: 7.5, 6.6, 5.43

パーツプログラム実行中に、複数回実行された測定結果とパーツ配置にも、配列が存在します。測定結果「Dim1」が少なくとも2回実行され、パーツ配置「Align1」が少なくとも4回実行され、これにより、`Dim1[2].Nom`と`Align1[4].Origin`が利用可能になります。

要素配列への参照が、範囲外である場合、(例えば、ユーザーが`Circle1[2.5]`か、`> Circle1["Hello, World"]`を請求)
上限か下限の項目が返されます。Circle1が、3つのインスタンスを持つ場合、`Circle1[4]`以上については、`Circle1[3]`と`Circle[0]`が返され、それより以下については`Circle1[1]`が返されます。角カッコ間のすべての式は、強制的に整数に変換され、したがって、2.5 は 2 に、そして、「Hello World」は 0に変換されます。

配列インデックスオブジェクト

デフォルトによって、フィーチャー配列は、常時、1次元の配列です。フィーチャーの配列を複数次元配列として扱ったほうが便利な場合、配列誘導オブジェクトを用いて、これを行うことができます。

配列誘導オブジェクトを用いると、配列の複数次元に上下限を特定できます。

- 第一次元の上下限の設定によって、第一次元の制限があり第二次元の制限がないところに、2次元配列を作成することができます。
- 配列の最初の2つの次元の上下限の設定によって、3次元配列を作成することができます。最後の次元は常時、無制限のままです。

例:

フィーチャーF1は、ネスト化された WHILE ループの内側にあります。内側の WHILE ループは5回実行され、外側の WHILE ループは3回実行されます。実行の完了時には、F1は15回実行され、その結果、15個のF1が存在します。

以下の測定ルーチンの一部の例を参考にして下さい:

```
ARRAY_INDICES/1..5,..
ASSIGN/V1 = 1
WHILE/V1<=3
ASSIGN/V2 = 1
```

```
WHILE/V2<=5
F1=FEAT/POINT,RECT
THEO/V2,V1,0,0,0,1
ACTL/1,1,0,0,0,1
MEAS/POINT,1
HIT/BASIC,V2,V1,0,0,0,1,1,1,0
ENDMEAS/
ASSIGN/V2 = V2+1
COMMENT/REPT,"Location of F1[" + V2 + "," + V1 + "] :" +
F1[V2,V1].XYZ
END_WHILE/
ASSIGN/V1 = V1+1
END_WHILE/
```

このコードは、15の測定ポイントからなる、3 X 5 の格子を作成します。

配列誘導コマンドは、フィーチャーの第一次元を、初めと終りを含んで1と5の間に制限します。これによりインスペクション レポートでは、F1[1] – F1[15]として表示されるかわりに、オブジェクトは、F1[1, 1] – F1[5, 3]として表示され、フィーチャーのレイアウトにより適合した形となります。コメントが、2次元配列シンタックスを用いて、フィーチャー配列を参照していることに注意して下さい。

配列インデックスオブジェクトを測定ルーチンに挿入するには:

1. キーボードを用い、編集ウィンドウ内の空白行に「**配列**」と入力して下さい。
2. キーボードのタブキーを押します。

注記:要素配列用にカッコ表示 チェック ボックスのチェックが消去されている場合、要素名は、カッコと共に表示されません。「優先設定」の章にある「要素配列用にカッコ表示」 を参照して下さい。

ヒットの配列:

与えられた要素のヒットは配列として利用可能であり、構文形式 <FeatID>.Hit[<配列式>].<配列式> の配列構文、または、構文形式 <FeatID>.RawHit[<配列式>].<式>を通してアクセスすることができます。プローブ補整が作動している時には、ヒットはプローブ補整データを返します。RawHit は常に、補整されていないデータを返します。有効な拡張子は、X、Y、Z、I、J、K、TX、TY、TZ、TI、TJ、TK、XYZ、TXYZ、IJK、及び、TIJKです。

「`Circle1.Hit[1].XYZ`
「円1」のヒット1の測定された重心（補整されたプローブ）

`Circle1.Hit[2].IJK`
「円1」のヒット2の測定されたベクトル

ヒットのデータは、実際のヒットが編集ウィンドウ内に表示されているか否かに関係なく、ヒットが行われたすべてのオブジェクトについて、入手可能です。したがって、ヒットのデータは、スキャンや自動フィーチャーについても、入手できます。

以下の項目は、いくつかの追加の配列関数について記述しており、それらは、スキャン内の最低点、または、最高点を検索する時、役に立ちます:

ヒット範囲を配列に割り当て

また、この構文を用いて、一続きのヒットを配列に割り当てることもできます:

<Feature Id>.<Hittype>[<Startnum>..<Endnum>].<Extension>

この説明

<Feature Id> は、フィーチャーの名称です。

<Hittype>
は、補整済みのデータについては「HIT」という語、または、補整されていないデータについては「RAWHIT」という語のどちらかが入ります。プローブ補整がオフになっている場合、戻り値は常に補整されていない値となります。

<Startnum> は、一続きのヒットの、最初のインデックス値を識別する式です。

<Endnum> は、一続きのヒットの、2番目のインデックス値を識別する式です。

<Extension>
は、データのタイプを識別します。可能な拡張子には、以下の測定された、または、理論的データ タイプを含まれます:

- X – 測定されたヒットのX値
- Y – 測定されたヒットのY値
- Z – 測定されたヒットのZ値
- XYZ – 測定されたヒットの XYZ 値
- I – 測定された ヒットのI 値
- J – 測定されたヒットのJ 値
- K – 測定されたヒットのK 値

- IJK – 測定されたヒットのIJK 値

- TX – 理論的なヒットのX 値
- TY – 理論的なヒットのY 値
- TZ – 理論的なヒットのZ 値
- TXYZ – 理論的なヒットのXYZ 値
- TI – 理論的なヒットのI 値
- TJ – 理論的なヒットのJ 値
- TK – 理論的なヒットのK値
- TIJK – 理論的なヒットのIJK値

例えば、

```
ASSIGN/V1 = SCAN1.HIT[1..10].X
```

V1
には、SCAN1の最初の10箇所のヒットから得られた、Xの測定値である10個の値の配列が割り当てられます。

```
ASSIGN/V2 = SCAN1.HIT[1..SCAN1.NUMHITS].XYZ
```

V２
には、そのスキャンに行われたヒットの重心それぞれからのポイントの配列が割り当てられます。

配列の並べ替え

PC-DMISでは、昇順、あるいは、降順で配列を並べ替えることができます。以下の2つの式は配列を取り、並べ替えられた配列を戻します:

昇順 で並べ替えるために使用するのは:
`SORTUP(<配列>)`を使用

降順 で並べ替えるために使用するのは:
`SORTDOWN(<配列>)`

例えば、

```
ASSIGN/V1 = ARRAY(5,8,3,9,2,6,1,7)
```
V1 に「５,８,３,９,２,６,１,７」の配列を割り当てます。

```
ASSIGN/V2 = SORTUP(V1)
```
V2 は昇順で並べ替えられた値の配列を持ちます: 「1,2,3,5,6,7,8,9」

```
ASSIGN/V3 = SORTDOWN(V1)
```
V3 は降順で並べ替えられた値の配列を持ちます: 「9,8,7,6,5,3,2,1」

配列から、最大指標値、あるいは、最小指標値を返すには:

これらの関数を使用することによって、配列を関数内に入力し、要素の最大、または、最小の、指標番号を戻すことができます:

最大 値を持つ要素の指標値を返すには、以下を使用して下さい:

```
MAXINDEX(<配列>)
```

最小 値を持つ要素の指標値を返すには、以下を使用して下さい:

```
MININDEX(<配列>)
```

例えば、

```
ASSIGN/V1 = ARRAY(5,8,3,9,2,6,1,7)
```
V1 に「５,８,３,９,２,６,１,７」の配列を割り当てます。

```
ASSIGN/V2 = MAXINDEX(V1)
```
V2は、配列インデックス値の4を保持します。配列要素の実際の数値は9です。

```
ASSIGN/V3 = MININDEX(V1)
```
V3は、配列インデックス値の７を保持します。配列の要素の実際の数値は1です。

その後、返された指標値を用いて、その配列要素の実績値を得ることができます。

配列から、並べ替えられた指標値を返すには

これらの関数を使用することによって、配列を関数内に入力し、配列値を昇順、あるいは、降順に並べ替え、その指標番号を戻すことができます:

最大使用頻度から最小使用頻度へと、配列の指標値を並べ替えるために、指標値の位置を返すには:
```
MAXINDICES(<配列>)
```

最大使用頻度から最小使用頻度へと、配列の指標値を並べ替えるために、指標値の位置を返すには:
```
MININDICES(<配列>)
```

例えば、

```
ASSIGN/V1 = ARRAY(4,8,2,9,5,7)
```
V1 に「4,8,2,9,5,7」の配列を割り当てます。

```
ASSIGN/V2 = MAXINDICES(V1)
```
V2は、配列に「4,2,6,5,1,3」の値を保持します。

```
ASSIGN/V3 = MININDICES(V1)
```
V3 は配列に「3,1,5,6,2,4」の値を保持します。

Exa配列関数を使用してスキャンの最小および最大ポイントを検索する例

上記で述べられたヒット配列関数の主な目的は、スキャン内の最小点、及び、最大点の検索を容易に行うことです。

SCAN1において、測定されたX値のうち最大のものを持つ、ポイント測定を行うには、以下の式を使用することが可能です:

```
ASSIGN/MAXPTINDEX = MAXINDEX(SCAN1.HIT[1..SCAN1.NUMHITS].X)

D1 = LOCATION OF FEATURE SCAN1.HIT[MAXPTINDEX]
```

SCAN2のZ軸にある3つの最高点を検索するには、以下の式を使用することが可能です:

```
ASSIGN/MI = MAXINDICES(SCAN2.HIT[1..SCAN2.NUMHITS].Z)

ASSIGN/THREEPOINTS = ARRAY(SCAN2.HIT[MI[1]].XYZ,
SCAN2.HIT(MI[2]].XYZ, SCAN2.HIT[MI[3]].XYZ)
```

変数の配列

変数の配列は、宣言を行う必要がありません。割り当てステートメントの右手側にある式が配列を評価する時、または、割り当てステートメントの左手側が変数配列内の要素を参照する時、変数配列は割り当てステートメントを通して作成されます。

```
Assign/V1 = Array(3, 4, 5, 6, 7)
```
5個の要素からなる配列を作成し、それをV1 に割り当てます。

```
Assign/V2 = V1[3]
```
配列 V1 : 5の第三要素の値 をV2に割り当てます

```
Assign/V1[4] = 23
```
配列 V1 の第4要素に23を割り当てます。

配列が作成され、動的に割り当てられました。このように、割り当てステートメントの左手側の配列参照を使用して、配列を作成することができます。

```
Assign/V3[5] = 8
```
5番目の要素を8にイコールする配列を動的に作成します。

今まで値を受け取ったことのない配列要素を参照すると、配列式は、0と評価されます。

```
Assign/V3[5] = 8
```

```
Assign/V4 = V3[5]
```
V4 の値は 8に設定されます。

```
Assign/V5 = V3[6]
```
V3の6番目の要素がまだ設定されていない場合、V5 は 0に設定されます。

他の配列タイプと同じように、式は、角カッコ内での使用が可能です。

```
Assign/V3[5] = 8
```

```
Assign/V4 = V3[2+3]
```
V4 の値は 8に設定されます。

変数配列は、複数の次元を持つことができます。

```
Assign/V6 = Array( Array(4, 7, 2), Array(9, 2, 6))
```
V6 は、2 掛ける 3の次元的配列として設定され、V6[1, 1] は 4、V6[1, 2] は 7、V6[1, 3] は 2、 V6[2, 1] は 9、 V6[2,2] は 2、及び、 V6[2,3] は 6の値を持ちます。

```
Assign/V7 = V6[2,1]
```
V7 は 9 に設定されます。

変数配列は、負の指数を持つことができます:

```
Assign/V8[-3] = 5
```
配列V8の – 3番目の指数は、5に設定されます。

配列の割り当ては、それ以前の値を上書きします:

```
Assign/V8 = "Hello"
```
変数 V8 は、「Hello」という値を持つ文字列です。

```
Assign/V8[2] = 5
```
V8 はもはや文字列型ではなく、配列タイプであり、その2番目の要素 5 の値を持ちます。

```
Assign/V8 = 9
```
V8 はもはや配列タイプではなく、整数 9 の値を持ちます。

配列は、複数の型から構成されることが可能です:

```
Assign/V9 = Array("Hello", 3, 2.9, {FEAT1})
```

４要素を持つ配列 V9
を作成します。第一要素は文字列であり、第二要素は整数、第三要素は実数、そして、第四要素は FEAT1を指すポインターです。

配列がもっと多くの要素を含むためにサイズで増やされることができます:

```
ASSIGN/V10=ARRAY(3,1,5)
```

```
ASSIGN/V10[LEN(V1)+1]=7
```

最初の文は、3つの要素（3、1および5）で、最初の配列V10を生成します。そして、二番目のステートメントは、1つの要素によってV10の中の配列を構成し、最終要素に7の値を与えます。

式用の演算子

以下の基本的演算子が、PC-DMIS内で利用可能です:

+ 加算: <*式*> + <*式*>
2つの式を一つに加算します。文字列の場合、 文字列が結びつけられます。

- 減算: <*式*> - <*式*>
第一の式を、第二の式から引きます。

***** 乗算: <*式*> * <*式*>
2つの式を掛けます。

/ 除算: <*式*> / <*式*>
第一の式を、第二の式で割ります。

^ 累乗: <*式*> ^ <*式*>
「第一の式」を、「第二の式」で累乗します。

% 剰余: <*式*> % <式>
ひとつの式を、もうひとつの式で割った余りを返します。

- 反数-<*式*>
式の反数を返します。

! 論理非:*!*<*式*>
式の論理非を返します。

== 等しい: <*式*> == <*式*>
式 が等しい場合、1と評価します。それ以外の場合は0と評価します。(2つの等号は割り当て操作=割り当てステートメントに区別に使用されます)。

<> 等しくない: <*式*> <> <*式*>
式が等しくない場合、1と評価します。それ以外の場合は0と評価します。

> より大きい : *:* <*式*> > <*式*>
第一の式が第二の式より大きい場合、1と評価します。それ以外の場合は0と評価します。

>= より大きい、または、等しい: <*式*> >= <*式*>
第一の式が第二の式より大きいか、または、等しい場合、1と評価します。それ以外の場合は0と評価します。

< より小さい: <*式*> < <*式*>
第一の式が第二の式より小さい場合、1と評価します。それ以外の場合は0と評価します。

<= 以下: <*式*> <= <*式*>
第一の式が第二の式より小さいか、または、等しい場合、1と評価します。それ以外の場合は0と評価します。

AND Logical And: <*式*> *AND* <*式*>
両方の式 が0と評価する場合、1と評価します。それ以外の場合は0と評価します。

OR 論理和: <*式*> *OR* <*式*> いずれかの式
が0と評価しない場合には、1と評価します。それ以外の場合は0と評価します。

() 丸かっこ: *(* <*式*> *)*
丸かっこ内にある式に優先度を与えます。

優先順位

式は、下記に表示された優先順位で、評価されます
(最高の優先順位から、最低の優先順位へと一覧に表示)。

最高の優先順位

- 被演算子
- (単項マイナス), !, (), 関数functions (such as ABS, COS, STR, LEN, CROSS, など)
- ^
- *, /, %

- +, -
- ==, <>, <, <=, >, >=
- AND
- または

最低の優先順位

関数

関数は、PC-DMISの特定の式かパラメーターを取って結果を戻るユーザーの定義された式です。式が評価される前に、パラメーターは式に代替されます。

関数一覧

以下のアルファベット順の一覧には、PC-DMISの式言語に利用可能な、すべての関数が表示されています。

- ABS (数学)
- ACOS (数学)
- ANGLEBETWEEN (ポイント)
- ARCSEGMENTENDINDEX (その他)
- ARCSEGMENTSTARTINDEX (その他)
- ARRAY (配列)
- ASIN (数学)
- ATAN (数学)
- CHR (文字列)
- COS (数学)
- CROSS (ポイント)
- DEG2RAD (数学)
- DELTA (ポイント)

- DIST2D (ポインタ)
- DIST3D (ポインタ)
- DOT (ポイント)
- ELEMENT (文字列)
- EOF (その他)
- EOL (その他)
- EQUAL (配列)
- EQUAL (文字列)
- EXP (数学)
- FORMAT (文字列)
- FUNCTION (関数)
- GETCOMMAND (ポインタ)
- GETSETTING (文字列)
- GETTEXT (文字列)
- IF (その他)
- INDEX (文字列)
- ISIOCHANNELSET (その他)
- LEFT (文字列)
- LEN (配列)
- LEN (ポインタ)
- LEN (文字列)
- LINESEGMENTENDINDEX (その他)
- LINESEGMENTSTARTINDEX (その他)
- LN (数学)

- LOG (数学)
- LOWERCASE (文字列)
- MAX (配列)
- MID (文字列)
- MIN (配列)
- MPOINT (ポイント)
- ORD (文字列)
- PCDMISAPPLICATIONPATH (文字列)
- PCDMISUSERHIDDENDATAPATH (文字列)
- PCDMISUSERVISIBLEDATAPATH (文字列)
- PCDMISSYSTEMHIDDENDATAPATH (文字列)
- PCDMISSYSTEMVISIBLEDATAPATH (文字列)
- PCDMISSYSTEMREPORTINGPATH (文字列)
- PROBEDATA (その他)
- QUALTOOLDATA (その他)
- RAD2DEG (数学)
- RIGHT (文字列)
- ROUND (数学)
- SIN (数学)
- SQRT (数学)
- SYSTEMDATE (文字列)
- SYSTEMTIME (文字列)
- SYSTIME (文字列)
- TAN (数学)

- TUTORELEMENT (その他)
- UNIT (ポイント)
- UPPERCASE (文字列)

文字列関数

以下の関数は、テキスト文字列と共に使用されます。

CHR

CHR
文字変換: *CHR(<整数>)*

ASCII 10進値に対応する文字から構成された文字列を返します。

ELEMENT

ELEMENT
サブ文字列の位置に範囲設定: *ELEMENT(<整数>, <文字列1>, <文字列2>)*

"文字列2の要素を分けて区切るテキストとして文字列1を使用して文字列2からｎ番目サブストリング(要素)を返します。例えば、もし文字列2
が「6、12、8、4、5」であり、そして文字列1 が "" 、""
であるなら、要素コマンドという状態で、５つの要素は「6」、「12」、「8」、「4」と「5」という個々に検索されることができます。

EQUAL

EQUAL
大文字と小文字を区別せずに文字列比較: *EQUAL(<文字列>, <文字列>)*

2つの文字列が全く同一であるか否か、を決めるために、それらを (文字格は無視)
比較します。それらの文字列が同一である場合、整数1を返し、そうでない場合には、0を返します。

FORMAT

FORMAT
フォーマット: FORMAT(<文字列>,<整数,倍精度,またはポイント>)

C++内での *sprintf*
関数と類似した、この関数は、2つの式を取り、書式整形された文字列を返します。

- 式1は、*文字列*型であるべきであり、1個か3個の書式指定子を含みます。式1が異なる型の場合、式評価機は、それを文字列への強制変換を試みます。式2が整数型

、または、ダブル型である場合、文字列は、*ひとつ*
の書式指定子を持つべきであり、式2がポイント型である場合、*3つ*
の書式指定子を持つべきです(下記の記述を参照して下さい)。

- 式2
  は、*整数*、*ダブル*、または、*ポイント*型であることが予期されます。異なるタイプが使用されている場合、その式の値は 0になります。

FORMAT関数用の書式指定子:

当書式指定子は、C++プログラミング言語で使用される *sprintf*
関数で用いられる書式指定子と同様の構文を持つべきです。

書式指定子は、随意フィールドと要求フィールドからなり、以下の構文を持ちます:

%[フラッグ] [幅] [.精度] タイプ

書式指定子の各フィールドは、特定の書式整形オプションを意味する、単一の文字または数字からなります。最も簡単な書式指定子は、パーセント記号と、タイプを示す文字のみを使用します
(例えば、%d)。パーセント記号の後に、書式整形フィールドとしての意味を持たない文字が続く場合、その文字はSTDOUTにコピーされます。例えば、パーセント記号を印刷するには、％％を用いて下さい。

随意のフラッグ、幅、及び、精度フィールドは、タイプを示す文字の前に表示されますが、これらは、書式整形のその他の側面をコントロールします。これらは、下記に述べられています:

フラグ
これらの *オプション文字* 出力内容調整、及び、記号、空白、小数点、及び、8進 / 16進接頭辞の印刷をコントロールします。複数のフラグを、書式指定子に表示することが可能です。

以下は、利用可能なフラッグです:

–
意味: 所定のフィールドの幅内で結果を左揃えに配置します。
ﾃﾞﾌｫﾙﾄ: 右割り当てる。

+
意味: 出力値が符号付のタイプの場合、出力値に符号(+ あるいは–
)の接頭辞を付けます。
デフォルト:負の値のみ符号 (–) が表示されます。

0
意味:
幅の前に 0 が付いている場合、最小幅に達するまで、ゼロが付け加えられます。0 と – が表示されると、0は無視されます。0が整数形式 (i, u, x, X, o, d)に指定される場合、0は無視されます。
デフォルト: スペースはありません。

ブランク(' ')
意味:
出力値が符号付きで正の値の場合、この出力値の前にブランクを付けます；ブランクと+フラグが表示されている場合はこのブランクは無視されます。
デフォルト: ブランクは表示されません。

#
意味1: o、 x、または X タイプで使用している場合、
＃フラグがそれぞれに0、 0x、または0Xに非零出力値でも前に置きます。
デフォルト1: 接頭辞が表示されません。

意味2: e、 E、または f
タイプを使用している場合、＃フラグが出力値をすべての場合に小数点を含むこと強制します。
デフォルト2: 数字が続く場合は小数点が表示されます。

意味3: g または G
形式で使用される場合、＃フラグが出力値をすべての場合に小数点を含むこと強制にして、末尾のゼロの切捨てを防ぎます。

デフォルト3:桁が続く場合のみ小数点が現れます。最後のゼロは切り捨てられます。
注記: d、 i、 またはuと共に使用されると、無視されます。

幅
この第2オプションのフィールドまたは引数は、印刷される文字の最小数を制御します。それは負ではない十進数の整数です。

- 出力値の文字数が、指定幅以下である場合、 — 「–」フラッグ (左配置用の) が — 指定されているか否か次第で –
最低幅に達するまで、その値の左か右に空白が追加されます。

- 幅の前に 0 が付いている場合、最小幅に達するまで、ゼロが付け加えられます（左配置の数字では役に立ちません）。

- 幅の特定機能が、値の切り捨てを行うことは決してありません。出力値の文字数が指定幅以上の場合、または、幅の指定がない場合、出力値の文字すべてが印刷されます(下記の一覧に表示された、精度の特定機能の影響を受けます)。

精度

この第三の任意のフィールド、または、引数は、印刷文字数、小数点以下の表示位、あるいは、有効数字を指定します。幅の指定と異なり、精度特定機能を用いると、出力値の切り捨て、または、浮動小数点数の切り捨てが可能です。これは、負でない、10進数の整数で、ピリオド (.) が前に置かれます。

タイプ

この必須文字は、関連の引数が整数、倍精度、または、ポイントのいずれであるか、を決定します。利用可能なタイプの一覧には、以下が含まれます:

d - 正負の符号表記のある、10進数の整数

i - 正負の符号表記のある、10進数の整数

o - 正負の符号表記のない、8進数の整数

u - 正負の符号表記のない、 10進数の整数

x - 正負の符号表記のない 16進数の整数で、小文字「abcdef」表記

X - 正負の符号表記のない 16進数の整数で、大文字「ABCDEF」表記

e - 指数形式 [-]d.dddd e [符号]ddd の倍精度

E - 指数を導入するために、Eを使用することを除いて、e と同じ。

f - [-]dddd.dddd 形式の倍精度

g - eか、または、f タイプのうち、より簡潔な方にフォーマット

G - 指数を導入するために、Eを使用することを除いて、 g と同じ。

FORMAT 例 :

この例では、測定ルーチン内でFORMAT関数を使用したいくつかのステートメントを示します:

`ASSIGN/V1 = PROBEDATA("OFFSET")`	V1はその時点でのプローブのオフセットを表すポイント型になります。この例で用いられている測定ルーチンからの値を使用すると、V1は以下のよう になります: <-1.8898, 1.8898, 5.704>。
`ASSIGN/V3 = FORMAT("%.5f, %.5f, %.5f", V1)`	V3 は文字列型になります。文字列は、変数V1のポイントオブジェクトを用いてフォーマットされます。ここでV3は、以下のような値を持ちます: -1.88976, 1.88976, 5.70403
`ASSIGN/V4 = 1.123456789`	V4 はダブル型になります。
`ASSIGN/V5 = FORMAT("%.5f ", V4) + FORMAT("%.6f ", V4) + FORMAT("%.7f ", V4) + FORMAT("%.8f", V4)`	V5は、この値を持つ文字列型になります: 1.12346 1.123457 1.1234568 1.12345679
`ASSIGN/V6A = "V4の値は: " + FORMAT("%.8f", V4)`	V6Aは、次の値を持つ文字列型になります: V4の値は: 1.12345679
`ASSIGN/V6B = FORMAT("V4 の値は: %.8f", V4)`	式の結果は、上記のV6Aと同じで、変化はありません。
`ASSIGN/V7 = 4444`	すべての数字がダブル型であると、推測されるので、整数に強制変換されないかぎり、V7はダブル型になります。
`ASSIGN/V8 = FORMAT("%o", INT(V7))`	V8は、この値を持つ文字列型になります: 10534
`ASSIGN/V9 = FORMAT("%u", INT(-1))`	V9は、この値を持つ文字列型になります: 4294967295
`ASSIGN/V10 = FORMAT("%x", INT(2143))`	V10は、この値を持つ文字列型になります: 85f
`ASSIGN/V11 = FORMAT("%X",`	V11は、この値を持つ文字列型になります: 255F

`INT(9567))`	
`ASSIGN/V12 = FORMAT("%e", 0.0005432)`	V12は、この値を持つ文字列型になります: 5.432000e-004
`ASSIGN/V13 = FORMAT("%E", 145.3421)`	V13は、この値を持つ文字列型になります: 1.453421E+002
`ASSIGN/V14 = FORMAT(",%6d,", INT(1))`	V14は、この値を持つ文字列型になります: , 1,
`ASSIGN/V15 = FORMAT(",%-6d,", INT(1))`	V15は、この値を持つ文字列型になります: ,1 ,

GETSETTING

GETSETTING
これを用いると、挿入された文字列パラメータ次第で、PC-DMISの各種の設定を返すことができます。

GETSETTING(<*文字列*>)

以下の文字列パラメータを使用することができます:

- 「DCC Mode」– PC-DMISがDCCモードにある場合、1を返し、それ以外の場合には、0を返します。
- 「手動モード」 – PC-DMISが手動モードにある場合、1を返し、それ以外の場合には、0を返します。
- 「現在のアラインメント」 – その時点でのアラインメントの文字列を返します。
- 「現在の作業平面」 – その時点での作業平面の文字列を返します。
- 「作業平面値」 – その時点での作業平面の数値を返します。
- 「プレヒット」 – その時点でのヒット前の値を、倍精度数として返します。
- 「撤回」 – その時点での撤回値を、倍精度数として返します。

- 「チェック」 – その時点でのチェック値を、倍精度数として返します。
- 「接触速度」 – その時点での接触速度値を、倍精度数として返します。
- 「移動速度」 – その時点での移動速度値を、倍精度数として返します。
- 「フライモード」 – PC-DMISがフライモードにある場合、1を返し、それ以外の場合には、0を返します。
- 「Ph9 の存在」 – Ph9/Ph10が存在する場合、1を返し、それ以外の場合には、0を返します。
- 「手動 CMM」 – CMMが手動CMMである場合、1を返し、それ以外の場合には、0を返します。
- "LangStr(<数字または ID>)" – リソースID番号から、または以下のIDのうちの一つから、現在の言語でPC-DMISのリソースから文字列を返します:
  "Yes", "No", "Oper", "Rept", "Input", "Doc", "YesNo", "Readout", "Internal", "External", "Rect ", "Polr ", "Out", "In", "Least_Sqr", "Min_Sep", "Max_Insc", "Min_CircSc", "Fixed_Rad", "Workplane", "Xaxis", "YAxis", "ZAxis", "Xplus", "Xminus", "YPlus", "YMinus", "ZPlus", "ZMinus", "Point", "Plane", "Line", "Circle", "Sphere", "Cylinder", "Round_Slot", "Square_slot", "Cone",または"None"。

  お客様の値が正の数である場合、PC-DMISは、そのresource.dll ファイルから文字列を引き出します。負数を使用する場合、PC-DMISは、そのstrings.dll ファイル (文字列表)から文字列を引き出します。
- 「展開の板金」–「**拡大された板金測定オプション**を表示」チェックボックスが**セットアップ オプション** ダイアログボックス内で選択された場合、1を返し、それ以外の場合には、0を返します。
- 「LastHitMove(X)」 – 最も最近の HIT /BASIC、または、MOVE/POINT コマンドのX値を返します。これが作動するには、PC-DMISがDCCモードになければなりません。
- 「LastHitMove(Y)」 – 最も最近の HIT /BASIC、または、MOVE/POINT コマンドのY値を返します。これが作動するには、PC-DMISがDCCモードになければなりません。
- 「LastHitMove(Z)」 – 最も最近の HIT /BASIC、または、MOVE/POINT コマンドのZ値を返します。これが作動するには、PC-DMISがDCCモードになければなりません。

PC-DMISを手動モードとDCCモードのどちらに置くか、決めるには、GETSETTING関数を用いた、以下の例を参考にして下さい:

例:
例:`ASSIGN/DCCMODEVAR = GETSETTING("DCC モード")` - PC-DMISがDCCモードにある場合、変数CCMODEVARに1の値を指定し、そうでなければ0を与えます。

`ASSIGN/MANMODEVAR = GETSETTING ("手動モード")` - PC-DMISが手動モードの場合、変数MANMODEVARに1の値を指定し、そうでなければ0を与えます。

その時点での作業平面を決めるには、以下の例を参考にして下さい:

例:
`ASSIGN/WORKPLANE_ID = GETSETTING("現在の作業平面")` - 変数WORKPLANE_IDに現在の作業平面の文字列値(ZPLUS, ZMINUS など)を与えます。

`ASSIGN/WORKPLANE_VALUE = GETSETTING("作業平面の値")` - 変数WORKPLANE_VALUEに、作業平面を表わす数値を与えます。作業平面がそれらに関連付けられる値があります: ZPLUS = 0、ZMINUS = 3、XPLUS = 1、 XMINUS = 4、YPLUS = 2、またはYMINUS = 5。

GETTEXT

GETTEXT
特定されたデータ フィールドから、その時点でのテキストを返します:
`GETTEXT(<文字列または整数>, <整数>, <ポインター>)`

この関数には、3つのフィールドがあります。

最初のフィールド—データ フィールドの番号、あるいは、内容説明

最初のフィールドは下図の項目(A)に示すようにデータフィールドの文字列の説明、または下図の項目(C)に示すようにデータフィールド番号のいずれかです。

注記:
下図の項目(B)はこの機能では使用されませんが、自動化やレポート式では時々使用されます。

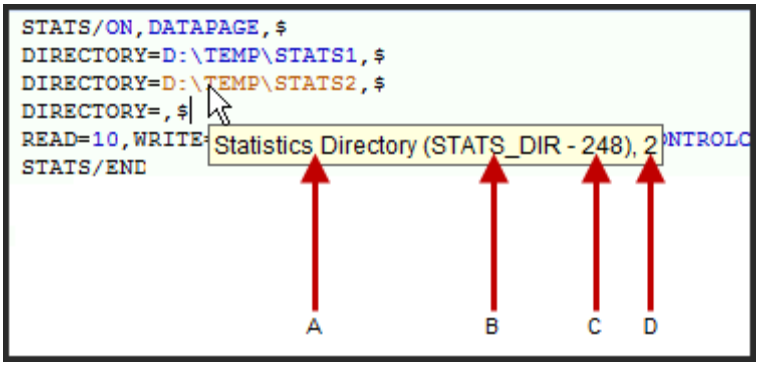
これらの値を得るには:

1. PC-DMISをコマンド
   モードにして下さい。編集ウィンドウ上の任意のところで右クリックして下さい。ショートカットメニューが現れます。

2. ショートカット メニューから、**ポップアップ表示の変更**
   を選択し、それから**データタイプ情報**を選択して下さい。

3. 編集ウィンドウのデータフィールドの上にマウスを置きます。そのデータ項目に対するタイプ名、タイプ番号、及びタイプインデックスが表示されます。

注記：タイプ名は、異なる言語によって違う場合があります。測定ルーチンが異なる言語のPC- DMISバージョンで実行されている場合は、代わりにタイプ番号を使用します。



(A)タイプ名、(B)タイプ文字列識別子、(C)タイプ番号、および(D)タイプインデックスを示すデータタイプ情報の例

第二フィールド—タイプ インデックス

2番目のフィールドは、上図(D)の示すようにタイプ
インデックスです。上図に示すように複数のDIRECTORYフィールドのような同一コマンドの中に同じタイプのフィールドのインスタンスが多くない限り、このフィールドは通常ゼロとなります。このフィールドの正しい値は、最初のフィールドと同じ要領で、得ることが可能です。

第三フィールド—コマンド ポインター

3番目のフィールドは、コマンド
ポインタです。これは、テキストの入手先であるフィールドにあるコマンドを指します。このフィールドは、コマンド ポインター表記
(つまり、{F15})か、または、この例に表示されているように、GetCommand式を使用して、特定することができます。

例:

```
ASSIGN/V1 = GETTEXT("最適化数学型", 0, {F15})-
```
このコマンドはV1を関数F15の最適化数学型トグルの現在の値に割り当てます。

```
ASSIGN/V2 = GETCOMMAND("コメント", "TOP", 1)-V2
```
はポインタを測定ルーチンの一番上から最初のコメントに割り当てます。

```
ASSIGN/V3 = GETTEXT("Comment Type", 1, V2)-V3
```
がコメントタイプのトグルフィールドの値を割り当てられます。測定ルーチンの最初のコメントはオペレーターに表示されるコメントである場合、V3の値は文字列「OPER」になります。

コマンドへ向けたポインタの設定に使用するGETCOMMAND式の情報については「ポインタ関数」を参照して下さい。

GETPROGRAMINFO

GETPROGRAMINFO
は以下で渡されるパラメータに基づいて測定ルーチンの情報を返します：
```
GETPROGRAMINFO(<String>, <Optional String>)
```

この関数は最大2つのパラメータを取ります。ほとんどのアイテムについては、最初のパラメータだけが必要になります。文字列フィールドでは、大文字と小文字は区別されません。

はじめのフィールド—文字列

最初のフィールドはどんな情報を返すかを詳述するストリング入力です。

DATE - 現在の日付を返します

TIME - 現在の時刻を返します。

PARTNAME - 測定ルーチンのヘッダで定義されるパーツ名を返します。

PARTPATH - フルパスを測定ルーチンファイルに返します

REVISION - 見出しに定義されるような改訂番号を返します。

DRAWING - REVISIONと同様、見出しに定義されるような改訂番号を返します。

SERIALNUM - ヘッダーに定義されるようなシリアル番号を返します。

STATSCOUNT - 現在の統計数を返します。

SEQNUM - STATSCOUNTと同様、現在の統計数を返します。

SHRINK - グローバルスケール係数を返します。

NUMMEAS - 実行された測定結果の数を返します。

NUMOOT - 実行された測定結果の公差範囲外の数を返します。

ELAPSEDTIME - 実行開始から経過した時間を返します

FILENAME - 測定ルーチンのファイル名 (.prg) を返します。

REPORTNAME - 現在の出力ファイル名を返します

TEMP -
オプションの第二の入力文字列の温度を返します。下記の「第2フィールド—オプションのストリング」を参照してください。

PRGVERSION - 測定ルーチンファイルの PC-DMIS
バージョン番号の文字列値を返します。測定ルーチンファイルを特定のバージョンと互換性のあるバージョンで保存できます。詳しくは「基本的なファイルオプションの使用」の章にある「名前を付けて保存」を参照して下さい。

PRGSCHEMA - 測定ルーチンファイルのPC-DMISスキーマ番号を整数で返します。これは、シリアライズされるコマンドとオプションを示すためにPC-DMISによって使用される内部値です。

PCDMISVERSION - 実際にインストールされたPC-DMISソフトウェアのバージョンを文字列で返します。

第2のフィールド—オプションのストリング

第2のフィールドはオプションのストリング入力です。それは単にTEMPが最初の入力フィールドの中で使用される場合には、必要です。下記の可能なストリングは温度補償コマンドから来ます。詳しくは「優先設定」の章の「温度の補償」を参照してください。

TEMPP - パーツセンサーの温度を返します

TEMPX - X軸センサーの温度を返します

TEMPY - Y軸センサーの温度を返します

TEMPZ - Z軸センサーの温度を返します

REF_TEMP - 基準温度を返します

HIGH_THRESHOLD - 温度の最高閾値を返します

LOW_THRESHOLD - 温度の最低閾値を返します

例

`$ $ NO` 、このコードサンプルは、全寸法数と許容範囲外の寸法数を表示します。

```
ASSIGN/V1 = GETPROGRAMINFO("NUMMEAS")

ASSIGN/V2 = GETPROGRAMINFO("NUMOOT")

COMMENT/REPT
```

```
「全部寸法: 」 + V1
「公差範囲外: 」 + V2
```

```
$ $ NO 、このサンプルコードは、 Zセンサ軸の温度を返します。
ASSIGN/V3 = GETPROGRAMINFO("TEMP", "TEMPZ")
COMMENT/REPT
「Z 軸の温度:」 + V3
```

ELAPSEDEXECUTIONTIME

ELAPSEDEXECUTIONTIME
フォーマットされた実行経過時間: *ELAPSEDEXECUTIONTIME()*

この機能は、測定ルーチンまたはミニルーチンが実行を開始してからの経過時間を返します。実行経過時間、実行のDCC部分の実行時間であり、それは、ユーザが注意に必要とする一時停止の時間を追加していません。（例えば、コメントが実行されたときに、実行は一時停止されて、メッセージが表示されたか、またはエラーが表示され、実行が停止することがあります）時間は「HH：MM：SS 」の書式で返されます。

次のように、関数を変数に割り当てることによって、測定ルーチンまたはミニルーチン内の任意の時点で実行経過の時間を記録することができます：

ASSIGN/V1=ELAPSEDEXECUTIONTIME()

INDEX

INDEX
サブストリングの位置*: INDEX(<文字列>, <文字列>)*

最初の文字列の中にある2番目の文字列の位置を返します。文字列の最初の文字は1です。ゼロの返す値は、サブ文字列が文字列に存在しないことを示します。

この関数の例については、「ファイルインプット、アウトプットを使用」章に「レッドラインのサンプルコード」というトピックをご覧ください。

LASTEXECUTIONTIME

LASTEXECUTIONTIME
書式が設定された最後の実行時間: *LASTEXECUTIONTIME()*

この関数は、 PC-DMISが<*測定ルーチン名*>.MiniRoutines.xmlファイルに記録して、保存された最後の実行時

間を返します。最後の実行時間は、**実行**ダイアログボックスに表示されます。時間は「HH：MM：SS 」の書式で返されます。

LEFT

LEFT
文字列の左の n 文字: *LEFT(<文字列>, <文字列>)*

最初の式で指定された文字列から2番目の式によって指定される、左からn文字を含む文字列を返します。第一の式は文字列型に強制変換され、第二文字列は整数型に強制変換されます。

この関数の例については、「ファイルインプット、アウトプットを使用」章に「レッドラインのサンプルコード」というトピックをご覧ください。

LEN

LEN
文字列の長さ: *LEN(<文字列>)*

文字列内の文字数を返します。

LOWERCASE

LOWERCASE
小文字の文字列を作ります: *LOWERCASE(<文字列>)*

小文字の文字列を返します。

MID

MID
文字列の中央のn文字です: *MID(<文字列>, <整数>, <任意の整数>)*

第一パラメータで指定された、文字列にある文字で構成されたサブ文字列を返します。
その開始位置は第二パラメータによって指定され、そのn個の文字数の長さは、第三パラメータによって指定されます。第三パラメータが存在しない場合、文字列の残りの部分が返されます。

この関数の例については、「ファイルインプット、アウトプットを使用」章に「レッドラインのサンプルコード」というトピックをご覧ください。

ORD

ORD
序数変換: *ORD(<文字列>)*

文字列の第一文字の整数 ASCII値を返します(0～255)。

PCDMISAPPLICATIONPATH

PCDMISAPPLICATIONPATH
完全パスの表示: PCDMISAPPLICATIONPPATH*()*

PC-
DMISがインストールされているアプリケーションディレクトリへの完全パスを含む文字列の値を返します。このディレクトリは、PC-
DMISを実行するために主な実行可能で他の必要なプログラムファイルを含んでいます。

PCDMISUSERHIDDENDATAPATH

PCDMISUSERHIDDENDATAPATH
完全パスの表示: PCDMISUSERHIDDENDATAPATH*()*

PC-
DMISによって使用される隠れたユーザーデータディレクトリへの完全パスを含む文字列の値を返します。このディレクトリに含まれるファイルに関しては
「ファイルの場所について」を参照してください。

PCDMISUSERVISIBLEDATAPATH

PCDMISUSERVISIBLEDATAPATH
完全パスの表示: PCDMISUSERHIDDENDATAPATH*()*

PC-
DMISによって使用される表示されたユーザーデータディレクトリへの完全パスを含む文字列の値を返します。このディレクトリに含まれるファイルに関しては
「ファイルの場所について」を参照してください。

PCDMISSYSTEMHIDDENDATAPATH

PCDMISSYSTEMHIDDENDATAPATH
完全パスの表示: PCDMISSYSTEMHIDDENDATAPATH*()*

PC-
DMISによって使用される隠れたシステムデータディレクトリへの完全パスを含む文字列の値を返します。このディレクトリに含まれるファイルに関しては
「ファイルの場所について」を参照してください。

PCDMISSYSTEMVISIBLEDATAPATH

PCDMISSYSTEMVISIBLEDATAPATH
完全パスの表示: PCDMISSYSTEMVISIBLEDATAPATH*()*

PC-
DMISによって使用される表示されたシステムデータディレクトリへの完全パスを含む文字列の値を返します。このディレクトリに含まれるファイルに関しては「ファイルの場所について」を参照してください。

PCDMISSYSTEMREPORTINGPATH

PCDMISSYSTEMREPORTINGPATH
完全パスの表示: PCDMISSYSTEMREPORTINGPATH*()*

PC-
DMISによって使用されるレポートディレクトリへの完全パスを含む文字列の値を返します。このディレクトリには、レポート･ウィンドウで使用されるレポートとラベルのテンプレートが含まれています。

RIGHT

RIGHT
文字列の右からn文字: *RIGHT(<文字列S>, <整数>)*

文字列から、整数で特定された右からn個の文字から成る文字列を返します。

SYSTEMDATE

SYSTEMDATE
システムの日付: SYSTEMDATE(<日付の形式の文字列>)

その時点での日付の詳細が記入された、日付フォーマットの文字列を返します。例えば、現在の日付が2014年2月12日の場合、コマンド
SYSTEMDATE("MM'/'dd'/'yy")は、文字列「02/12/14」を返します。

日付の文字列を作るためには、以下の文字列要素を使用して下さい。要素は、以下の表示と同じ文字格でなければなりません（mmのかわりにMM）。日付フォーマットの文字列要素の合間にある日付以外の活字（スペースのような）は、入力文字列と同一の位置にある、出力文字列内に表示されます。単一引用符によって範囲設定された、入力文字列内の文字は、単一引用符にない出力文字列と同一の位置に表示されます。

d - 月の日付を数字で表示します。一桁の日付については先頭にゼロを付けません。

dd – 日付を数字で表示します。一桁の日付では、先頭にゼロが付きます。

ddd – 週の曜日を3文字の略字で表示します。

dddd – 週の現在の曜日の完全な名前です。

M – 月を数字で表示、一桁の月の場合は先頭にゼロを付けません。

MM – 月を数字で表示、一桁の月の場合は先頭にゼロを付けます。

MMM – 月名を3文字の略字で表示します。

MMMM – 月を全名で表示します。

y – 年を数字で表示し、一桁の年は先頭にゼロを付けません。

yy –年を２桁数字で表示、一桁の年には先頭にゼロを付けます。

yyyy – 年を4桁数字で表示します。

SYSTEMTIME

SYSTEMTIME
書式設定されたシステム時間: *SYSTEMTIME(<時間書式の文字列>)*

その時点での時間の詳細が記入された、時間書式の文字列を返します。例えば、コマンドSYSTEMTIME("hh:mm:ss tt")は"11:29:40 PM"
のように書式設定された文字列で時刻を返します。

時間の文字列を作るためには、以下の文字列要素を使用して下さい。要素は、以下の表示と同じ文字格でなければなりません
(**TT**ではなく**tt**)。時間フォーマットの文字列要素の合間にある時間以外の活字（スペースのような）は、入力文字列と同一の位置にある、出力文字列内に表示されます。単一引用符によって範囲設定された、入力文字列内の文字は、単一引用符にない出力文字列と同一の位置に表示されます。

h - 12時間表示の時間、一桁の時間の場合は先頭にゼロを付けない

hh - 12時間表示の時間、一桁の時間の場合は先頭にゼロを付ける

H - 24時間表示の時間、一桁の時間の場合は先頭にゼロを付けない

HH - 24時間表示の時間、一桁の時間の場合は先頭にゼロを付ける

m - 分を桁通りに表示、一桁の場合は先頭にゼロを付けない

mm - 分を桁通りに表示、一桁の場合は先頭にゼロを付ける

s - 秒を桁通りに表示、一桁の場合は先頭にゼロを付けない

ss - 秒を桁通りに表示、一桁の場合は先頭にゼロを付ける

t - A や Pのような一文字の時間表示文字列

tt - AM や PMのような複数文字の時間表示文字列

SYSTIME

SYSTIME
システム時刻: *SYSTIME()*

現在のシステム時刻を文字列で返します。この関数は、上記で説明したSYSTEMTIMEとは異なります。これは、日付、曜日、時間、その後に年を自動的に返します。

例: "Wed February 12 13:50:21 2014"

> **注記:**
> その時点のシステム時刻を表示し、返された文字列は、現地の時間帯の設定に調整されています。

UPPERCASE

UPPERCASE
大文字の文字列を作成する: *UPPERCASE(<String>)*

大文字の文字列を返します。

数学関数

ABS

ABS 絶対値: *ABS(<倍精度>)*
入力の絶対値を返します。

EXP

EXP 指数関数: *EXP(<倍精度>)*
式の指数関数を返します。

LOG

LOG 10底対数関数: *LOG(<倍精度>)*
式の10底対数関数を返します。

LN

LN 自然対数関数: *LN(<倍精度>)*
式の自然対数関数を返します。

ROUND

ROUND 四捨五入: *ROUND(<倍精度>)*
入力内容を最も近い整数に四捨五入して返します。

SQRT

SQRT 平方根: *SQRT(<倍精度>)*
入力内容の平方根を返します。

三角法関数

重要事項:
各々の三角法関数は、デフォルトによって、ラジアンを取り、ラジアンを返します。度の値を希望する場合には、下記に記述のある、RAD2DEG関数を用いて下さい。

ACOS

ACOS 逆余弦: *ACOS(<倍精度>)*
式の逆余弦を返します。例えば、 ACOS(5.0)は、0 を返します。一般に、ACOS(<式>)は、式の値の余弦逆関数を返します。

ASIN

ASIN 正弦逆関数: *ASIN(<倍精度>)*
入力内容の正弦逆関数を返します。

ATAN

ATAN 正接逆関数: *ATAN(<倍精度>)*
入力内容の正接逆関数を返します。

COS

COS 余弦: *COS(<倍精度>)*
入力内容の余弦を返します。

DEG2RAD

DEG2RAD 度をラジアンに変換: *DEG2RAD(<倍精度>)*
入力内容を360で割り、2πで掛けて返します。度をラジアンに変換します。

RAD2DEG

RAD2DEG ラジアンを度に変換: *RAD2DEG(<倍精度>)*
入力内容を360で掛け、2πで割って返します。ラジアンを度に変換します。

SIN

SIN 正弦: *SIN(<倍精度>)*
入力内容の正弦を返します。

TAN

TAN 正接: *TAN(<倍精度>)*
入力内容の正接を返します。

注記: 入力内容が範囲外である関数は、(例えば、ACOS, ASIN, LOG, LN, SQRT等では、コンピューターのクラッシュの原因になります) 0を返します。

ポイント関数

ANGLEBETWEEN

ANGLEBETWEEN

間の角度: *ANGLEBETWEEN(<ベクトル>, <ベクトル>)*

2つのベクトル間の角度を計算します。結果は度で返されます。2つのパラメータは、ベクトル・タイプに評価する式でなければなりません。要素からベクトルを得るためには、たとえば、.IJK拡張子が続く要素IDを使う必要があります。以下の例を参照してください。

例:

```
F1 =GENERIC/POINT,DEPENDENT,CARTESIAN,$

NOM/XYZ,<3,3,3>,$

MEAS/XYZ,<3,3,3>,$

NOM/IJK,<1,0,0>,$

MEAS/IJK,<1,0,0>

F2 =GENERIC/POINT,DEPENDENT,CARTESIAN,$

NOM/XYZ,<10,10,10>,$

MEAS/XYZ,<10,10,10>,$

NOM/IJK,<0,0,1>,$

MEAS/IJK,<0,0,1>

ASSIGN/V1=F1.IJK

ASSIGN/V2=F2.IJK

ASSIGN/V3=ANGLEBETWEEN(V1,V2)

COMMENT/OPER,NO,FULL SCREEN=NO,AUTO-CONTINUE=NO,
```

`V1 + 「 と 」 + V2 + 「 の間の角度は: 」 + V3`

CROSS

CROSS
外積: *CROSS(<点>, <点>)*

戻り値はポイント型であり、第一と第二の式の外積です。

DELTA

DELTA
ベクトルオフセット*: DELTA(<点>, <点>, <倍精度>)*

この関数は、最初の式 (ポイント) を取り、そして、2番目の式 (ベクトル)
の方向に新規のポイントを、3番目の式のオフセットにおいて、計算します。例えば、
`DELTA(MPOINT(0,0,0), MPOINT(1,0,0), 10)` は、ポイント 10,0,0を返します。

DOT

DOT
点乗積: *DOT(<点>, <点>)*

2つのポイント(ベクトル)の点乗積を返します。

UNIT

UNIT
単位ベクトル: *UNIT(<点>)*

その長さによって分割された点を返します。例えば、`UNIT(MPOINT(0,0,0))`
は、ポイント0,0,1を返します。

MPOINT

MPOINT
ポイント強制: *MPOINT(<式1>, <式2>, <式3>)*

3つの式をポイントタイプに強制型変換します。

例:
```
ASSIGN/V1 = MPOINT(2.5,3.6,4)
```

X は2.5の値を持ちます。
V1.Y は3.6の値を持ちます。
V1.Z は4.0の値を持ちます。

「点の強制型変換」を参照してください。

ポインター関数

DIST2D

DIST2D
2次元距離: DIST2D(<FEAT1>, <FEAT2>, <FEAT3>)

> 要素は、内部のブレースでなければなりません。

DIST2D: 2d Distance: DIST2D(<FEAT1>, <FEAT2>, <FEAT3>)
これはコマンド（Feat1とFeat2）の最初の2つの引数の間に、3番目の引数（Feat3）に垂直する距離を計算できます。

- 番目の引数は平面である場合、最初の2つの引数の間の距離は平面に垂直な計算されます。
- 3番目の引数はラインまたは円柱である場合、最初の2つの引数の間の距離はアクティブワークプレーン3番目の引数に垂直な計算されます。

たとえば、XY平面を3番目の引数とする場合、それはZ+ ベクトル(0,0,1) とレポートされた距離はZ axis軸にあります。

例

```
ASSIGN/V3=DIST2D({CIR1},{CIR2},{PLN1})

COMMENT/OPER,NO,FULL SCREEN=NO,AUTO-CONTINUE=NO,

V3
```

DIST3D

DIST3D
3次元距離: DIST3D(<FEAT1>, <FEAT2>)

要素1および要素2の間の3次元距離を計算します。

> **注記:** 要素は中括弧の中になくてはなりません。

例

```
ASSIGN/V3=DIST3D({CIR1},{CIR2})

COMMENT/OPER,NO,FULL SCREEN=NO,AUTO-CONTINUE=NO,

V3
```

GETCOMMAND

GETCOMMAND
パラメータによって特定されたコマンドに、ポインターを与えます:
`GETCOMMAND(<整数または文字列>, <文字列>, <整数>`

最初のパラメータコマンド情報フィールド

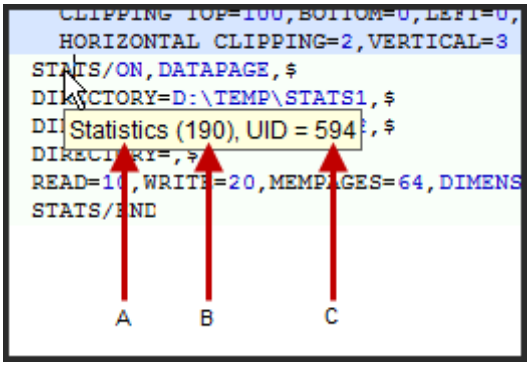
第一パラメータは、コマンド情報フィールドです。これは、サーチの対象となるコマンドのタイプを指定します。以下を渡すことが可能です:

- コマンド内容説明の文字列。下記の図(A)を参照して下さい:
- コマンド タイプの番号。下記の図(B)を参照して下さい:
- 独自番号識別子。下記の図(C)を参照して下さい:

コマンドの独自のIDが渡される場合、その他の引数は必要ありません。

コマンド内容説明の文字列、コマンドタイプ番号、及び、コマンドの独自番号識別子を入手するには:

1. 編集ウィンドウ上を右クリックして下さい。
2. **ポップアップ表示の変更 | コマンド情報** を選択して下さい (PC-DMISは、コマンド モードでなければなりません)。
3. ご希望のコマンド上に、マウスを位置付けて下さい。当コマンドの内容説明の文字列、コマンドタイプ番号、及び、ストリングの独自番号識別子が、ポップアップに表示されます。



(A)コマンド説明文、(B)コマンドタイプ番号、および(C)一意の番号識別子 (UID) を示したコマンド情報の例。

2番目のパラメータの検索方向

第二パラメータでは、サーチの指示が行われます。正当な値は、以下の通りです:

値	説明
UP	この値は、サーチが、その時点でのコマンドから開始され、 上方に向けて進むことを意味します。
DOWN	この値は、サーチが、その時点でのコマンドから開始され、 下方に向けて進むことを意味します。
TOP	この値はサーチが測定ルーチンの冒頭から開始され、 下方に向けて進むことを意味します。
下面	この値はサーチが測定ルーチンの最後のオブジェクトから開始され、 上方に向けて進むことを意味します。

第3パラメータ - どのインスタンスが検索します

3番目のパラメータは測定ルーチン内に同じコマンドの複数のインスタンスがる場合、コマンドが見つかるはずという例を示しています。

例:
測定ルーチンに、STATS/ONコマンドのインスタンスが2つあり、頂上から2番目のインスタンスへのポインターを得たい場合、「2」が第三パラメータとして渡され、「TOP」が第二パラメータとして渡されます。

```
ASSIGN/V1 = GETCOMMAND("Statistics", "TOP", 2)。
```

GETCOMMAND関数は、GETTEXT文字列関数に、第三パラメータを与えるために使用することができます。GETTEXTに関する情報については、「文字列関数」を参照して下さい。

LEN

LEN
ポインターのループ数カウント: *LEN(<POINTER>)*

ポインターがループを行った回数を返します。例えば、要素CIR1が、10回反復するループ内にある場合、CIR1のこれまでの測定回数を、以下のようなASSIGNステートメントを用いて、変数内に保存することが可能です: `ASSIGN/V1=LEN({CIR1})`

配列関数

ARRAY

ARRAY:配列を作成: *ARRAY(<EXPRESSION1>, <EXPRESSION2>, <EXPRESSION3>, …)*
式パラメータによって指定される配列要素を持つ配列オブジェクトを作成します。配列要素は、ベース指数1で番号付けられています。

EQUAL

EQUAL: 要素の配列の比較による要素: EQUAL(<ARRAY>, <ARRAY**>)** *要素で要素の2つの配列を比較して配列は同じ* 要素
を保持するかどうかを決定します。2つの配列のサイズが同じでない場合、または、一方の配列の要素がもう一方の配列の対応する要素と一致しない場合、その関数は0を返します。それ以外の場合には、その関数は1を返します。

LEN

LEN: 配列要素の総数: *LEN(<ARRAY>)*
配列内の要素の数を返します。

MAX

MAX: 最大配列要素: *MAX(<ARRAY>)*
配列内の最大要素を返します。配列内の項目は、数値的に、または、アルファベット順で比較されます。

MIN

MIN: 最小配列要素: *MIN(<ARRAY>)*
配列内の最小要素を返します。配列内の項目は、数値的に、または、アルファベット順で比較されます。

その他の関数

ARCSEGMENTENDINDEX

ARCSEGMENTENDINDEX
:
これは、スキャン内の、特定のアークの終点のインデックス番号を返します:ARCSEGMENTENDINDEX*(<ID>, <index>,<tol1>,<tol2>)*

<ID> –
最初のパラメータは、スキャンIDの文字列値であり、この関数は、円弧の開始点のインデックス番号をここから引き出します。これは、引用符内のIDであるか、または、文字列型に強制されると、スキャンのIDを表示する式、のどちらかとなります。

<*インデックス*> –
第二パラメータは、終了点番号を得る、円弧のインデックス番号です。これは、ひとつに基づいた

値です。例えば、スキャン内の3番目の円弧の終了点番号を得たい場合、円弧インデックス番号は 3 になります。

<tol1> –
三番目のパラメータは、一般要素の公差です。これは、スキャン情報を、線や円弧に分割するために用いられる、最大形状誤差です。

<tol2> –
第四パラメータは、正確な公差です。一般に、セグメントの形状誤差がこの公差範囲内にある限り、要素のいずれのか終端から点をドロップするために使用されます。

いったん、円弧の開始インデックスと終了インデックスが得られると、分離した円弧フィーチャーを組み立てるために、組み立てられたフィーチャー内でこれらの点を使用することができます。類似の例については、「スキャンから作成された直線フィーチャーの例」を参照して下さい。

ARCSEGMENTSTARTINDEX

ARCSEGMENTSTARTINDEX
:これは、スキャンからの特定のアークの始点のインデックス番号を返します:
ARCSEGMENTSTARTINDEX*(<ID>, <インデックス>,<tol1>,<tol2>)*。

<ID> –
最初のパラメータは、スキャンIDの文字列値であり、この関数は、円弧の開始点のインデックス番号をここから引き出します。これは、引用符内のIDであるか、または、文字列型に強制されると、スキャンのIDを表示する式、のどちらかとなります。

<index> –
第二パラメータは、開始点番号を取得する円弧のインデックス番号です。これは、ひとつに基づいた値です。例えば、スキャン内の3番目の円弧の開始点番号を得たい場合、円弧インデックス番号は 3 になります。

<tol1> –
三番目のパラメータは、一般要素の公差です。これは、スキャン情報を、線や円弧に分割するために用いられる、最大形状誤差です。

<tol2> –
第四パラメータは、正確な公差です。一般に、セグメントの形状誤差がこの公差範囲内にある限り、要素のいずれのか終端から点をドロップするために使用されます。

2つの追加のパラメータがあり、それらは、スキャン内の識別された円弧が許容可能か否かをコントロールします。これらは、PC-DMIS設定エディタを用いてのみ変更できます。その半径が`MinimumArcSegmentRadiusInMM`以下である円弧はすべて拒否されます。このパラメータのデフォルト設定値は 2 mmです。同様に、その半径が `MaximumArcSegmentRadiusInMM`

以上である円弧はすべて拒否されます。このパラメータのデフォルト設定値は 2000 mm
(この値の変更は不必要であるべきです)。

いったん、円弧の開始インデックスと終了インデックスが得られると、分離した円弧フィーチャーを組み立てるために、組み立てられたフィーチャー内でこれらの点を使用することができます。類似の例については、「スキャンから作成された直線フィーチャーの例」を参照して下さい。

EOF および EOL

EOF 及び**EOL**
: これらの関数に関する情報については、「ファイル入力 /
出力の使用」の章にある「ファイルの終端、または行の終端をチェック」を参照して下さい。

FUNCTION

FUNCTION
関数: FUNCTION*((<PARAM1>, <PARAM2>…), <EXPRESSION>)を作成します。*

パラメーター一覧に表示されたパラメータ数を取り、それらを式に置き換える、関数を作成します。

- `FUNCTION` キーワードを使用時の、最初の項目は、パラメーター一覧です。
- この一覧は、コンマによって区切られた、パラメータ名で構成されています。
- パラメーター一覧はまた、丸かっこによって囲まれています。
- 第二項目は、式です。
- 式には、パラメータ名があり、それらのパラメータは、関数が呼ばれた時、置き換えられます。

この例については、「包括的関数の例」の項目を参照して下さい。

GETROTABDATA

GETROTABDATA
これは、現在の回転表からのデータを返します: GETROTABDATA*(<パラメータ>)*

<パラメータ> - これは「中心」あるいは「角度」です。

「中心」- テーブルのXYZ中心を返します。

「角度」- テーブルの現在の角度を返します。

例:

`ASSIGN/V1 = GETROTABDATA("CENTER")`	V1 の値はXYZセンター回転表の値に設定されます。
`ASSIGN/V2 = GETROTABDATA("ANGLE")`	V2 の値は現在の角度回転表の値に設定されます。

IF

IF
: 条件的式評価: IF*(<式1>, <式2>, <式3>)*

式1の評価が真(ゼロ以外)の場合、この関数は式2の値を返します;
それ以外の場合には、この関数は式３の値を返します。

ISIOCHANNELSET

ISIOCHANNELSET
: この式は、2つのパラメータを取ります。最初のパラメータは、I / Oチャネル使用可能な番号（範囲チェックされるマシンを使用しているに基づいている）ことを示します。第2のパラメータは、ソフトウェアがアーム1またはアーム2の測定機にクエリするかどうかを決定します。第2パラメータが1に設定されると、アーム2コントローラにクエリします。2番目のパラメータが存在しない(または、ゼロに設定される)場合、IOチャネルアーム1コントローラにクエリします。複数のアームモードでない場合、アーム1コントローラが唯一のオプションとなります。

> **注記:**先端チップID、プローブ
> ファイル名、または、チャネル番号など、無効なタイプのプローブ
> データが与えられると、その式は 0 と評価します。

例:

`ASSIGN/V4 = ISIOCHANNELSET(3, 0)`	チャネルが設定されている場合、V4 の値は 1 であり(真と評価)、それ以外の場合には、その値は 0 です(偽と評価)。

LINESEGMENTENDINDEX

LINESEGMENTENDINDEX
:これは、スキャンからの、特定の線分の終了点のインデックス番号を返します:
LINESEGMENTENDINDEX*(<ID>, <index>,<tol1>,<tol2>)*。

<ID> –
最初のパラメータは、スキャンIDの文字列値であり、この関数は、線分の終了点のインデックス番号をここから引き出します。これは、引用符内のIDであるか、または、文字列型に強制されると、スキャンのIDを表示する式、のどちらかとなります。

<インデックス> –
第二パラメータは、終了点の番号を取得する線分からのインデックス番号です。これは、ひとつに基づいた値です。例えば、スキャン内の3番目の直線の終了点番号を得たい場合、線分インデックス番号は3になります。

<tol1> –
三番目のパラメータは、一般要素の公差です。これは、スキャン情報を、線や円弧に分割するために用いられる、最大形状誤差です。

<tol2> –
第四パラメータは、正確な公差です。一般に、セグメントの形状誤差がこの公差範囲内にある限り、要素のいずれのか終端から点をドロップするために使用されます。

いったん、線分の開始インデックスと終了インデックスが得られると、分離した直線フィーチャーを組み立てるために、組み立てられたフィーチャー内でこれらの点を使用することができます。この例については、「スキャンから作成された直線フィーチャーの例」を参照して下さい。

LINESEGMENTSTARTINDEX

LINESEGMENTSTARTINDEX
これは、スキャンからの、特定の線分の開始点のインデックス番号を返します:
LINESEGMENTSTARTINDEX*(<ID>, <index>,<tol1>,<tol2>).*

<ID> –
最初のパラメータは、スキャンIDの文字列値であり、この関数は、線分の開始点のインデックス番号をここから引き出します。これは、引用符内のIDであるか、または、文字列型に強制されると、スキャンのIDを表示する式、のどちらかとなります。

<インデックス> –
第二パラメータは、開始点の番号を取得する線分からのインデックス番号です。これは、ひとつに基づいた値です。例えば、スキャン内の3番目の直線の開始点番号を得たい場合、線分インデックス番号は 3 になります。

<tol1> –
三番目のパラメータは、一般要素の公差です。これは、スキャン情報を、線や円弧に分割するために用いられる、最大形状誤差です。

<tol2> –
第四パラメータは、正確な公差です。一般に、セグメントの形状誤差がこの公差範囲内にある限り、要素のいずれのか終端から点をドロップするために使用されます。

追加のパラメータがあり、それは、スキャン内の識別された線分が許容可能か否かをコントロールします。これは、PC-DMIS設定エディタを用いてのみ変更できます。その長さが、

`MinimumLineSegmentLengthInMM`以下である線分はすべて拒否されます。このパラメータのデフォルト設定値は 2 mmです。

いったん、線分の開始インデックスと終了インデックスが得られると、分離した直線フィーチャーを組み立てるために、組み立てられたフィーチャー内でこれらの点を使用することができます。この例については、「スキャンから作成された直線フィーチャーの例」を参照して下さい。

PROBEDATA

PROBEDATA
現在または指定のプローブに関するデータを返します : PROBEDATA*(<OPTPROBEDATATYPE>, <OPTTIPID>, <OPTPROBEFILENAME>)*

この関数は、最大3個までのオプションを随意に選択することが可能です。パラメータ間にコンマを置くだけで、複数のパラメータを使用することができます。空のパラメータ間のコンマは必要ありません。例えば、その時点でのプローブの直径を知るためには、単に`ASSIGN/V1 = PROBEDATA("DIAM")`を使用して下さい。

OPTPROBEDATATYPE: 式がどのプローブ
データを返すべきか指定する、選択可能なパラメータです。このパラメータが存在しない場合、その時点での先端チップIDが返されます。このパラメータは、文字列型です。有効な文字列式を評価する、式すべては、最初の式スロット内に配置することができます。最初のパラメータ用の、有効な文字列式 (大文字と小文字の区別を行わず)
には、以下のものを含みます。これらは文字列式であり、二重引用符内に置かれるべきです:

> *"オフセット" -*
> 測定された先端チップのX、Y、Zオフセットです。ポイント型を返します。
>
> *「ベクトル」* – 先端チップのベクトル。ポイント型を返します。
>
> *「**A**」* - A角度の先端チップ。倍精度型を返します。
>
> *「**B**」* - B角度の先端チップ。倍精度型を返します。
>
> *「**Diam**」 または「直径」 -*
> チップ直径の測定値。最初の4文字「Diam」が最低必要とされますが、"Diameter"のフル ネームになるまでの文字を加えることが可能です。倍精度型を返します。
>
> *「厚さ」または「**Thickness**」 –*
> 測定された先端チップの厚さ。最初の5文字「Thick」が最低必要とされますが、フルネームの「Thickness」になるまでの文字を加えることが可能です。倍精度型を返します。
>
> *「日付」* – 先端チップが最後に校正された日付です。文字列型を返します。

「時間」 – 先端チップが最後に校正された時間。文字列型を返します。

*「**ID**」* – 先端チップIDです。デフォルトのパラメータ。文字列型を返します。

"PrbRdv" – プローブ半径偏差。倍精度型を返します。

*"**Standarddeviation**"* - プローブの標準偏差。倍精度型を返します。

*"**C**"* - CW43 照明プローブ ヘッドのC角度。整数型を返します。

> **注記:**
> 「**Offset**」、「**Diameter**」、または、「**Thickness**」の前に「**T**」を付け加えると、理論値が返されます (つまり、TOFFSET、TDIAMETER、及び、TTHICKNESS)。

OPTTIPID -
この随意のパラメータは、最初の式で指定されたプローブのデータを得る時に、使用する先端チップを特定します。特定されない場合、その時点での先端チップIDが使用されます。このパラメータは、文字列型であるべきです。

OPTPROBEFILENAME:
この随意のパラメータは、プローブのデータを得る時に、使用するプローブのファイル名を特定します。特定されない場合、その時点でのプローブ ファイルが使用されます。

例:

`ASSIGN/V1 = PROBEDATA()`	V1は、その時点での先端チップIDに設定されます (つまり、「T1A0B0」)
`ASSIGN/V2 = PROBEDATA("TOFFSET", "T1A45B0")`	V2 は、先端チップT1A45B0の、理論的プローブ オフセットに設定されます。
`ASSIGN/V3 = PROBEDATA("Date", "T1A90B90", "MYPROB")`	V3 は、プローブ ファイル MYPROBの先端チップT1A90B90が、最後に校正された 日付を表示する文字列に設定されています。

TUTORELEMENT

TUTORELEMENT: この関数は、数か文字列のいずれかの引数をひとつ取ります(文字列は、要素のIDです)。

この関数は、*ストラクチャー*というタイプの変数と共に機能します。ストラクチャーとサブ要素に関する説明については「ストラクチャー」を参照して下さい。

例:

`ASSIGN/E = TUTORELEMENT(1)`	単一のTutor Element ストラクチャーを作成します。
`ASSIGN/WM = TUTORELEMENT(n)`	1以上のどのような数でも、その数に応じて、n個のTutor Element ストラクチャーの配列を作成します。
`ASSIGN/CIR1E = TUTORELEMENT("CIR1")`	フィーチャー CIR1 からのデータをTutor Element ストラクチャー内に、コピーします。

TutorElement ストラクチャーは現在、以下のサブ要素を持ちます:

サブ要素	説明
ID	フィーチャーIDの文字列
タイプ	整数(FTYPE)
X、Y、Z	X、Y、及び、Z座標値
PR	極半径
PA	極角度
CX	I
CY	J
CZ	K
DM	直径 1
DM2	直径 2
DS	原点からの距離
A	角度
AXY	XY 平面での角度
AYZ	YZ 平面での角度
AZX	ZX 平面での角度
F	フォーム エラー

SDEV	標準偏差
TP	位置付け

QUALTOOLDATA

QUALTOOLDATA: この関数は、その時点での、または、特定のキャリブレーション ツールに関するデータを返します。それは、以下の構文を持ちます:

QUALTOOLDATA(<TOOLINFO>, <TOOLID>, <FACENUMBER>)

この関数は、最大3個までのオプションを随意に選択することが可能です。何かのデータを返すには、少なくともひとつのパラメータが必要です:

最初のパラメータ ***<TOOLINFO>*** は *文字列* であり、キャリブレーション ツールに関する、返す情報のタイプを指定します。このパラメータを渡さない場合、この関数は、その時点での、または、特定のキャリブレーション ツール名を返します。

- ・「**DIAM**」 – これは、ツールの直径をダブル値として返します。
- 「***ID***」 - これは、ツールの名称を文字列値として返します。
- 「***LENGTH***」 – これは、「DIAM」と同じように機能します。これはまた、ツールの直径を、ダブル値として返します。
- ・「**OVERRIDEIJK**」 – これは、サーチ優先 IJK ベクトルをポイント値として返します。
- ・「**POLYDIAM**」 – これは、指定された多面体表面の直径を、倍精度値として返します。
- ・「**POLYIJK**」 – これは、指定された多面体表面のIJKベクトルを、ポイント値として返します。
- ・「**POLYXYZ**」 – これは、指定された多面体表面のXYZの中心点をポイント値として返します。
- ・「***SHANKIJK***」 – これは、シャンクのIJKベクトルを、ポイント値として返します。
- 「***TYPE***」 – これは、ツールのタイプを整数値として返します(球には0、アム2 球には1、多面体には2、アム2多面体には3)。

- 「***WIDTH***」 – このパラメータは、もはや使用されておりません。
- 「***XYZ***」 – これは、ツールのXYZ位置をポイント値として返します。

第二のパラメータ*<TOOLID>*は *文字列*
であり、ユーザーが情報を要求している、キャリブレーション
ツールの名称を特定します。このパラメータを渡さない場合、PC-
DMISは、その時点でのキャリブレーション
ツールの情報が要求されている、と推測します。当文字列は、大文字と小文字の区別をしません。

第三のパラメータ*<FACENUMBER>*は、多面体キャリブレーション
ツールと共に使用される時、及び、
第一のパラメータが「POLYXYZ」、「POLYIJK」、または、「POLYDIAM」である場合にのみ必要です。これは、データ入手のために使用する多面体ツールの表面を特定する、整数値です。

例:

`ASSIGN/VDIAM = QUALTOOLDATA("DIAM","SPHERE_1_IN")`	変数`VDIAM`に、ツールSPHERE_1_INの直径を与えます。
`ASSIGN/VID = QUALTOOLDATA("ID")`	変数`VID`に、その時点でのツール名を与えます。
`ASSIGN/VTYPE = QUALTOOLDATA("TYPE")`	変数`VTYPE`に、その時点でのツールのタイプを与えます。
`ASSIGN/VPOLYDIAM = QUALTOOLDATA("POLYDIAM","POLYTEST",3)`	変数`VPOLYDIAM`に、多面体ツールPOLYTESTの表面3の直径を与えます。

関数の例

下記は、いくつかの異なる関数の例であり、これらは、お客様ご自身の関数を作成し使用する時に役に立ちます:

- 汎用関数の例
- 変数として渡された関数の例
- 複数のパラメータを用いた関数の例
- 他の関数を作成する関数の例

- 配列要素としての関数の例
- 再帰的に定義される関数の例

包括的関数の例

```
Assign/MYFUNC = FUNCTION((X,Y,Z), X*3 + Y*2 + Z)
```
ユーザーに定義された関数を作成し、それを変数
MYFUNCに割り当てます。その関数は、3種のパラメータ、X、Y、及びZを取ります。

X の値は、3で掛けられます。

Y の値は、2で掛けられます。

Z は、単に渡された値を持ちます。

X + Y + Z の合計が返されます。

```
Assign/V1 = MYFUNC(7,2,5)
```
関数MYFUNC(7、2、5)にパスされたパラメタの評価によって、V1が値30を持つよう割り当てます。７はパラメータであり、関数定義の式の部分にXが発生した位置に置換されます。これにより、X*3 は 7*3、または、21になります。

2 はY が発生するところで置き換えられ、これにより、Y*2 が 2*2、または、4になります。

5 はZ が発生したところで置き換えられます。

その後、それらの値はすべて加算され(21 + 4 + 5)、V1に渡されます。

変数として渡された関数の例

関数は、変数として渡されることが可能です。以下の例は、上記の汎用関数の例で用いられたものを更に使用しています:

```
Assign/NEWFUNC = MYFUNC
```
変数 NEWFUNCを変数 MYFUNCと同じの関数を持つように設定します。

```
Assign/V3 = NEWFUNC(12,2,3)
```
V3を、関数(36 + 4 + 3)内の評価された式からの値43を持つように割り当てます。

複数のパラメータを用いた関数の例

関数は複数のパラメータを持つことができます:

```
Assign/ADDANDDOUBLE = FUNCTION((A,B), 2*(A+B))
```
関数を作成し、これを関数ADDANDDOUBLEに割り当てます。この関数は、2つのパラメータを取り、それらを加え、その後、その結果を2倍に倍増します。

```
Assign/V2 = ADDANDDOUBLE(4, 5)
```
V2に値18を割り当てます。パラメータ 4 とパラメータ 5
は、その関数の式部分で置き換えられ、これにより、2*(4+5)となります。

他の関数を作る関数の例

関数は、他の関数を作ることができます。

```
Assign/COMPOSE = FUNCTION((F, G), FUNCTION((X), G(F(X)) ))
```
COMPOSEを、2つの関数をパラメータとして取る関数となるよう割り当て、その2つの関数を用いて、新しい関数を作成します。

```
Assign/ADD2 = FUNCTION((X), X+2)
```
ADD2を、渡されたパラメータに2を付け加える関数となるように割り当てます。

```
Assign/ADD3 = FUNCTION((X), X+2)ADD3
```
を、渡されたパラメータに3を追加する関数となるように割り当てます。

Assign/ADD5 = COMPOSE(ADD2, ADD3)
ADD5を、関数ADD2と関数ADD3からなる関数として割り当てます。

```
Assign/V5 = ADD5(3)
```
V5がV8の値を持つよう割り当てます。

配列の要素としての関数の例

関数は、配列の要素となることができます。

```
Assign/ANARRAY = ARRAY(3, FACTORIAL, "Hello World", ADD5)
```
ANARRAYを 4 要素の配列となるよう割り当てます : 数字 (3)、関数
(FACTORIAL)、文字列、("Hello World")、及び、関数 (Add5)。

```
Assign/V6 = ANARRAY[2](4)
```
ANARRAYの第二の要素は、関数FACTORIALです。パラメータ 4
がこの関数に渡され、その結果である 24 が V6に割り当てられます。

```
Assign/V7 = ANARRAY[2](ANARRAY[4] (ANARRAY[1]))
```
内側から外側へ: ANARRAY (3) の最初の要素が、4番目の配列要素である関数
(Add5)に渡されます。その結果である 8 は、2番目の配列要素である関数 (FACTORIAL)
に渡され、V7に割り当てられます。V7 は、40320の値を受け取ります。

再帰的に定義される関数の例

関数は、再帰的に定義、つまり、関数がそれ自身を呼び出すよう、定義することができます。

```
Assign/FACTORIAL = FUNCTION((X), IF(X<=1, 1, X*FACTORIAL(X-1))
```

1つのパラメーターをとる階乗と呼ばれる関数を作成します。パラメータが、1以下の場合は1と評価し、それ以外の場合には、XをX-1の階乗によって倍増した値と評価を出します。

```
Assign/V4 = FACTORIAL(5)
```

V4に120 (5*4*3*2*1)の値を割り当てます。

スキャン映像から作成された直線フィーチャーの例

"このトピックは、PC-DMIS式言語（特にラインセグメント機能）をどのように使用して、スキャンの範囲内でラインセグメントのスタートと終点番号を輸出し、それから組み立てられたフィーチャーの範囲内で引き抜かれたポイントを用いてあなた自分のラインフィーチャーをつくる方法の例を提供します。この例でカバーされた同じ原理を同様にスキャンからの弧部分を作ることに使うことができます。

お客様の測定ルーチンが、以下のように見える、SCN1という名のスキャンフィーチャーを持つとします:

```
SCN1=FEAT/SCAN,LINEAROPEN,SHOW HITS=NO,SHOWALLPARAMS=YES

     EXEC MODE=RELEARN, NOMS MODE=FIND NOMS,CLEARPLANE=NO,SINGLE
     POINT=NO,THICKNESS=0

     FINDNOMS=5,SELECTEDONLY=NO,USEBESTFIT=NO,PROBECOMP=YES,AVOIDANCE
     MOVE=NO,DISTANCE=0,CAD Compensation=NO

     DIR1=VARIABLE,

     HITTYPE=VECTOR

     INITVEC=0,-1,0

     DIRVEC=1,0,0

     CUTVEC=0,0,1

     ENDVEC=0,-1,0

     PLANEVEC=-1,0,0

     POINT1=100,0,-5

     POINT2=70,0,-5

     MEAS/SCAN

     BASICSCAN/LINE,SHOW HITS=NO,SHOWALLPARAMS=YES

     <100,0,-5>,<70,0,-5>,CutVec=0,0,1,DirVec=1,0,0
```

```
InitVec=0,-1,0,EndVec=0,-1,0,THICKNESS=0

FILTER/NULLFILTER,

EXEC MODE=RELEARN

BOUNDARY/PLANE,<70,0,-5>,PlaneVec=-1,0,0,Crossings=2

HITTYPE/VECTOR

NOMS MODE=FINDNOMS,5

ENDSCAN

ENDMEAS/
```

このスキャンから直線を作るには、LINESEGMENTSTARTINDEX
関数、及び、LINESEGMENTENDINDEX
関数を用いて、以下のようなデータを引き出すことが必要です:

```
ASSIGN/LINESTARTINDEX = LINESEGMENTSTARTINDEX("SCN1", 1, 0.4,0.1)

ASSIGN/LINEENDINDEX = LINESEGMENTENDINDEX("SCN1", 1, 0.4, 0.1)
```

これは、「SCN1」と呼ばれるスキャン、その最初のラインセグメントから定義された公差の範囲内入ったスタートとエンド指数値を引き抜くようにPC-
DMISに指示します。そして、それらの指数値をラインスタートインデックスとラインエンドインデックスと呼ばれた変数に割り当てます。

いったん線分の開始インデックス値と終了インデックス値が変数に割り当てられると、以下のように、それらの変数を、組み立てられた直線内で使用することが可能です。

```
LIN4=FEAT/LINE,RECT,UNBND

THEO/100.225,0,-5.011,1,0,0

ACTL/100.225,-0.005,-5.011,1,-0.0000388,0

CONSTR/LINE,BF,2D,SCN1.HIT[LINESTARTINDEX..LINEENDINDEX],,

OUTLIER_REMOVAL/OFF,3

FILTER/OFF,WAVELENGTH=0
```

上記の直線要素のハイライト表示されたコードにおいて、PC-
DMISが、スキャンからプルされた、開始数と終了数を使用して当要素を作成することに注意して下さい: `SCN1.HIT[LINESTARTINDEX..LINEENDINDEX]`

被演算子型変換

いずれの型強制演算子を用いてでも、被演算子をその他のタイプに強制変換することができます:

整数型強制

INT(<式>) - 式の値を整数型に強制します。

INT(4)	4 と評価
INT(4.5)	4 と評価
INT("Hello World")	0 と評価
INT("2")	2 と評価
INT("2.2")	2 と評価
INT("3 Blind Mice")	3 と評価
INT("The 3 Blind Mice")	0 と評価
INT("3, 4, 5")	3 と評価
INT(MPOINT(0, 0, 1))	原点からポイントへの距離を評価として出します。 この場合は、1。
INT(MPOINT(3, 4, 5))	距離は7.0711と評価されるので、この式は、7と評価が出ます。

ダブル型強制

DOUBLE(<式>) - 式の値をダブル型に強制します。

DOUBLE(4)	4.0 と評価
DOUBLE(4.5)	4.5 と評価
DOUBLE("文字列")	0.0 と評価
DOUBLE("3.5")	3.5 と評価
DOUBLE("3.5インチ")	3.5 と評価

DOUBLE("円の直径が3.5インチと測定される")	0.0 と評価
DOUBLE(MPOINT(0,0,1))	1.0 と評価
DOUBLE(MPOINT(3,4,5))	7.0711 と評価

文字列型変換

STR(<式>) - 式の値を文字列型に強制変換します。

STR(4)	4 と評価
STR(4.5)	4.5 と評価
STR("Hello World")	「Hello World」と評価
STR(MPOINT(3,4,5))	「3, 4, 5」と評価

ポイント型強制

MPOINT(<式1>, <式2>, <式3>) -
各々の式をダブル型に強制した後で、式の値を、ポイント型に強制します。

MPOINT(1, 1, 1)	ポイント 1.0,1.0,1.0と評価
MPOINT(1.1, 1.1, 1.1)	ポイント 1.1, 1.1, 1.1と評価
MPOINT("1", "1", "1")	ポイント 1.0,1.0,1.0と評価
MPOINT(3, 4.5, "5.6")	ポイント 3.0, 4.5, 5.6と評価
MPOINT(MPOINT(1, 0, 0), MPOINT(0,1,0), MPOINT(3,4,5))	1.0, 1.0, 7.0711と評価

被演算子型強制と混合型式

式評価機は、変数を自動的に混合型の式に強制変換します。自動的型強制のために、式の結果が予測と異なる場合、一部の場合には、型強制演算子を用いると、ご希望の結果が出るようになります。以下は、混合型式における、自動的な型強制の例です。

"CIR" + 1
「CIR1」に評価されます。

"2" + 2
4 と評価されます。

「2+2の値」は + 2 + 2
「2+2の値は 22」と評価されます（式が左から右に評価されるため)

「2+2の値」は+(2+2)
は「2+2の値は4である」と評価されます。

LINE1.XYZ > 2
原点から、LINE1の重心への距離が、2以上である場合、1と評価されます。

LINE1.XYZ > LINE2.XYZ
原点からLINE1の重心への距離が、LINE2の重心からの距離がLINE1への距離より大きい場合、1と評価されます。

LINE1.XYZ = LINE2.XYZ
LINE1とLINE2の重心が同じである場合、1と評価します (この場合、型強制は行われず)

DOUBLE(LINE1.XYZ) = DOUBLE(LINE2.XYZ)
二つの重心が、原点から同じ距離にある場合、1と評価します

11% 3.12
2と評価します（%は整数で機能するよう設計された剰余演算子。不連続な除数の余りを返します。11%3 = 2.)

CIRCLE1.HIT [3.2].X
円1の第3ヒットの測定されたXの数値を評価します。引数3.2は自動的に3という数値の整数に強制的に変換されます。

ID式

PC-DMISコマンドの多くは、フィーチャーIDを
パラメータとして用います。例えば、組み立てられるフィーチャーは、IDを用いて、どのフィーチャーが、組み立ての入力情報として使用されるかを指示します。ID式を用いると、ユーザーは、フィーチャーの特定の事例、類似の名称を持つフィーチャーのグループ、サブルーチンへの呼び出し内のフィーチャーの事例、、または、外部測定ルーチン内のフィーチャーを参照することができます。

フィーチャー配列のID

要素の特定のインスタンスを参照する場合、または、要素の一連のインスタンスを参照する場合、要素配列IDを用いて下さい。例えば、要素「Circle1」が、5度実行された while
ループ内にある場合、そのループを退出時には、その円の5つのインスタンスが存在することになります。「Circle1」の5つのインスタンスのうちの、個々のインスタンスを参照する場合、「要素配列:」下に記述された、要素配列シンタックスを使用して下さい。
それを用いると、「Circle1[1]」が最初のインスタンスを参照し、「Circle1[2]」が2番目のインスタンスを参照、等、となります。

インスタンスの範囲を参照するには、**..**
表記を用いてください。「円1[1..3]」は円1の最初から３番目までのインスタンスを参照します。「円[3..5]」は円1の３番目から５番目までのインスタンスを参照します。円[1..5]」は円1の最初から５番目までのインスタンスを参照します。一続きの要素が参照される場合はセットとして扱われ、構築されたセットして機能します。

IDワイルドカード

似たような名前の要素を参照するにはID
ワイルドカードを使用します。二つのワイルドカード文字は「*****」と「**?**」です。
詳しくは「CAD表示セクションの編集」セクションで「要素を選択してメタキャラクターマッチングを使う」をご覧ください。

星印
'*'の活字は、どのような活字でも、0個以上の事例を参照するために使われます。「CIR」という文字で始まる一連のフィーチャーすべてを参照するには、式 ID
として「CIR*」を使用して下さい。この構文は、「CIRCLE1」、「CIRCLE2」、「CIR3」、または、「CIR」のような、「CIR」で始まるIDを持つフィーチャーすべてを含む、一連のフィーチャーを作成します。

注記:
CIR3が複数回の実行を行った場合、最も最近の測定結果のみが使用されます。異なる実行のインスタンスを使用するには、以下の式を使用して下さい: CIR?[1..3]

疑問符「? 」文字は任意の文字の単一のインスタンスを参照するのに使用されます。

例:
「MY???1」というID式は、例えば、「MYCIR1」、「MYCON1」、「MYLIN1」、または、「MYFT21」のように、「MY」で始まり「1」で終り、長さが6文字である、一連の要素を作成します。

サブルーチン、BASICスクリプト、または、外部ルーチン内のフィーチャー用ID

サブルーチンは現在の測定ルーチン内または外部測定ルーチンに存在することができます。サブルーチンがサブルーチンの呼び出しと同じルーチンにある場合、「要素の配列:」で説明されている

要素配列ID構文を用いて、そのサブルーチン内に作成された要素の個々のインスタンスを参照することができます。しかし、サブルーチンが外部測定ルーチンにある場合、次の構文を使用して、サブルーチンで作成された任意の要素を参照することができます: "<Call Sub ID>:<FeatID>"。たとえば、「F1」という名前の要素が id「CS1」で Call Subコマンドから呼び出さた外部サブルーチンにある場合、ID 式「CS1:F1」を使用してその要素を参照することがきます。

例: この例は、CS1.F1 という構文の使用を説明しているに過ぎず、実際の使用を意図するものではありません。

ルーチン 1: PLUS1.PRG

```
SUBROUTINE/PLUS1, A1 = 0, A2 = 0, A3 = 0
F1 =FEAT/POINT,RECT
THEO /A1+1,A2+1,A3+1,0,0,1
ACTL/3,1,1,0,0,1
MEAS/POINT,1
HIT /BASIC,A1+1,A2+1,A3+1,0,0,1,0,0,0
ENDMEAS/
ENDSUB/
```

ルーチン 2: TEST.PRG

```
CS1 =CALLSUB/PLUS1,D:\V30\WINDEBUG\PLUS1.PRG: 3,3,3,,
DIM D1= LOCATION OF POINT CS1:F1 UNITS=IN,$
GRAPH=OFF TEXT=OFF MULT=10.00 OUTPUT=BOTH
AX NOMINAL +TOL -TOL MEAS MAX MIN DEV OUTTOL
X 3.0000 0.0000 0.0000 3.0000 3.0000 3.0000 0.0000 0.0000
----#----
END OF DIMENSION D1
```

BASICスクリプトは、オブジェクトを動的に作成、及び、削除します。BASICスクリプトで作成されたフィーチャーに参照するには、構文「<Basic Script ID>:<Feat ID>」を用いて下さい。例えば、「BS1」を持つBASICスクリプトが、ID 「F2」を持つフィーチャーを作成する場合、そのフィーチャーに参照するには、ID の式「BS1:F2」を使用して下さい。

添付コマンドを使用してPC-DMISに外部ルーチンを添付することができます。添付されたルーチンの要素を参照するには、以下の構文を使用してください: "<Attach Routine ID>:<Feat ID>"。ID "GEAR1"で添付された測定ルーチンの要素"F3"を参照するにはこの式を使用してください、"GEAR1:F3"。(さらに詳しい説明については、「外部要素の追加」の項の「外部測定ルーチンの添付」 を参照してください。)

IDの式の組み合わせ

IDの式配列、 ID
式ワイルドカード、外部サブルーチン、BASICスクリプト、及び、外部測定ルーチンのID式は、組み合わせて使用することができます。例えば、ID「BOLTPAT」の添付した、外部測定ルーチンにある「CIR」という文字で始まる全要素のうち、3番目のインスタンスに参照するには、ID式「`BOLTPAT:CIR*[3]`」を使用して下さい。

また、ID式は、通常の式においても使用することができます。したがって、上記の一連の要素の測定された重心は、以下の式を用いて、変数に割り当てられます:

```
ASSIGN/V1 = BOLTPAT:CIR*[3].XYZ
```

また、ID式は、通常の式においても使用することができます。したがって、上記の一連の要素の測定された重心は、以下の式を用いて、変数に割り当てられます:

```
ASSIGN/V1 = BOLTPAT:CIR*[3].XYZ
```

レポートのオブジェクト属性にアクセス

独自のカスタムレポートとラベルテンプレートを作成することができます。PC-DMISはこれらを使用してレポートウィンドウ内にレポートデータを表示します（**[ビュー | レポートウィンドウ]**を参照してください）。これらのテンプレートは、Visual Basicのようなインターフェースを利用する、テンプレートエディタを用いて作成され、それによって、「オブジェクト」と呼ばれる特別の構成要素を挿入、配置し直し、及び、サイズ変更を行うことができます。

各オブジェクトは、「属性」で構成されており、それによって、その表示方法と収容する情報内容が決まります。これらの属性の一部は、その他のオブジェクトすべてと共通したものであり、他の一部は、関連したオブジェクトにのみ共通したものであり、そして、それ以外の属性はその特定のオブジェクト独自のものです。

PC-DMIS式言語は、その時点でロードされているレポートを検査し、特定オブジェクトの属性値を変数内に保存します。以下のシンタックスを用いて、文字列タイプ、整数タイプ、及び、実績タイプの値を得ることができます:

属性検査シンタックス

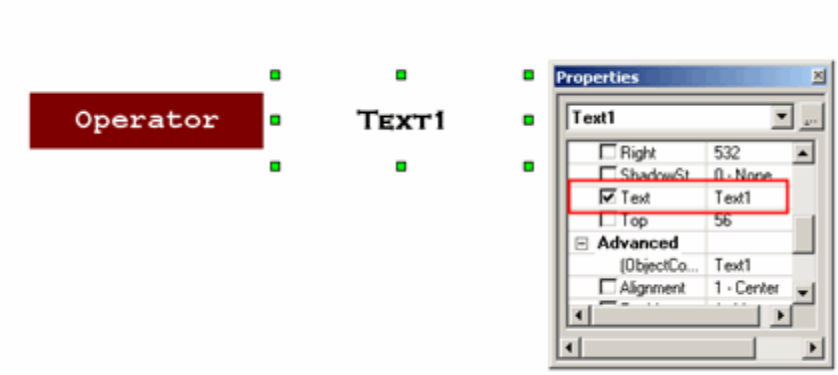
```
Assign/V1 = Report.<オブジェクト名>.<属性名>
```

レポートは、その時点で呼び出されているレポートを指しています。<オブジェクト名>は、そのオブジェクトの独自の名称であり、そして、<属性名>は、そのオブジェクト用の有効な属性名です。

例

オペレータの名前を表示するのにあなたのレポートのテンプレートに最終報告でご使用したいテキストオブジェクトがあり、それを「テキスト１」と呼ばれると仮定します。オペレータの名前を表

す実際の文字列はオブジェクトの
テキストプロパティに格納されます。デフォルトでは、テキストプロパティ(表示されたテキスト)は最初、「テキスト１」の値を持ちます(下の図を参照)。これはユーザーに割り当てられたプロパティであるため、オペレータが実行中に名前を入力すると変わります。



選択されたオブジェクトと検査される属性を表示する、「属性」ダイアログ ボックス

式言語コードを用いて、このテキスト
オブジェクトの「テキスト」属性を検査し、キー入力されたデータを得るためには、以下のコマンドを使用します:

```
Assign/V1 = Report.Text1.Text
```

このコードには：

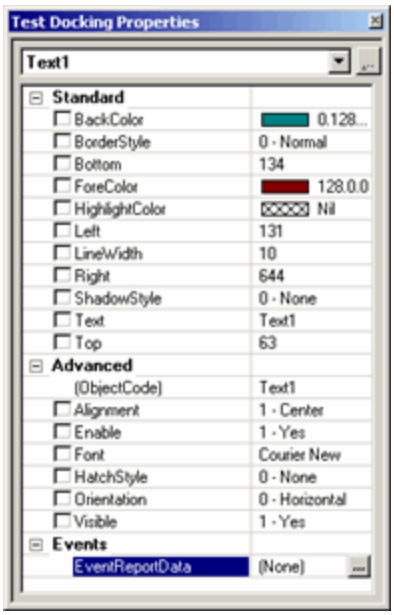
「レポート」は、レポート
ウィンドウ内に呼び出されたレポートを検査するよう、コードに指示します。

「Text1」は「Text1」という名のオブジェクトを探すよう、それに指示します。

「Text」は、そのオブジェクト内の「テキスト」属性を探すよう指示します。「テキスト」属性の値は、その後、変数V1内に渡され、それによって、PC-
DMIS式言語を用いて、さらに、操作や表示を行うことができます。

属性検索

特定のオブジェクトに関連した属性を検索できます。 それを行うには、レポート テンプレート エディタ内のレポート テンプレートにアクセスし(**ファイル | レポート | 編集 | レポート テンプレート**)、オブジェクトを選択し、その後、当オブジェクトを右クリックして、プロパティ シートを表示することができます。



テキスト オブジェクト用の属性シート

属性シートには、2つの縦列があります。左列は属性名を表示し、そして、右列はその時点での値を表示します。式コードでは、正確な属性名の使用を確認して下さい。

重要:
プロパティの値をクエリする際に、一部のプロパティが一見役に立たない数値を返すことに、お気付きになるかも知れません。一般に、これが起こるのは、そのプロパティが利用可能なオプションの一覧を持ち、PC-DMISが、表示のプロパティと関連のない、選択されたプロパティの内部値を返す場合です。
例えば、**テキスト**オブジェクトは、これらの値を備える**方向**プロパティを持っています。

- 0-水平
- 1-垂直に上へ
- 2-垂直に下へ

PC-DMIS式言語を利用して値を取得する場合は、ソフトウェアが代わりに、次を返します：

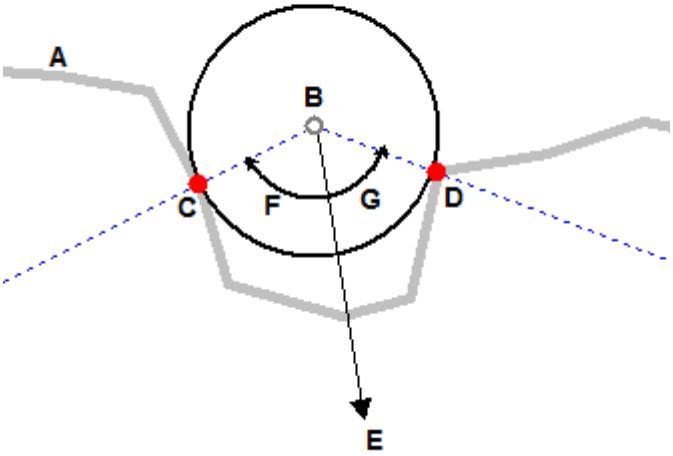
- 0 (水平)
- 900 (垂直に上へ)
- -900 (垂直に下へ)

これは、どの値がプロパティシートに表示された値に対応して返されるかを決定することに、いくつかの試行やエラーを必要とします。

構築されたｽｷｬﾝ最小円からのｱｸｾｽ情報

PC-DMIS式を使用して、リニアスキャンに沿って最小の時点で指定した半径で構成された円の機能から情報を引き出すことができます。もっと詳しい情報については、既存の要素から新要素の作成」項の「 スキャンの最小ポイントで円の作成」を参照してください。

スキャン最小サークルフィーチャーを組み立てると、サークルは究極的に（下方きのベクトルと呼ばれる）ベクトルを用いてスキャンラインを組み立てます。それは、コンタクトポイント（コンタクトポイント1とコンタクトポイント2）と呼ばれている2つの場所で、ラインと連絡するだけです。PC-DMISは、下方きのベクトルからこれらのコンタクトポイント（コンタクトポイント1とコンタクトポイント2）まで角度を測定するために、これらのポイントを使うことができます。たとえば、この図を参照してください：



A - 構築された円のスキャン線。
B - 円重心の最終のXYZ 位置。
C - 下方ベクトルの左の接点。CONTACTPOINT1と呼ばれます。
D - 下方ベクトルの右の接点。CONTACTPOINT2と呼ばれます。
E - 下方ベクトル。
F - 下方ベクトルからCONTACTPOINT1の角度。CONTACTANGLE1と呼ばれます。
G - 下方ベクトルからCONTACTPOINT2の角度。CONTACTANGLE2と呼ばれます。

下記に詳述される式は、この種類の構築円要素のみで機能します。第2接点で等しい情報を返すために、下記のシンタックスで代わりにCONTACTPOINT2を使うこともできます。

`ASSIGN/V1=CIR1.CONTACTPOINT1. XYZ`
線と円の最初の接点のXYZ情報を返します CONTACTPOINT1。

`ASSIGN/V1=CIR1.CONTACTPOINT1.X`
CONTACTPOINT1 のX情報を返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.Y`
CONTACTPOINT1のY情報を返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.Z`
CONTACTPOINT1のZ情報を返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.IJK`
CONTACTPOINT1から円の重心へのIJKベクトルを返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.I`
CONTACTPOINT1 IJK ベクトルより上からのI値を返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.J`
CONTACTPOINT1 IJK ベクトルより上からのJ値を返します。

`ASSIGN/V1=CIR1.CONTACTPOINT1.K`
CONTACTPOINT1 IJK ベクトルより上からのK値を返します。

`ASSIGN/V1=CIR1.CONTACTANGLE1`
下向きベクトルからCONTACTPOINT1への角度を返します。

`ASSIGN/V1=CIR1.CONTACTANGLE2`
下向きベクトルからCONTACTPOINT2への角度を返します。

`ASSIGN/V1=CIR1.CONTACTANGLE`
CONTACTANGLE1およびCONTACTANGLE2の間の絶対値の合計を返します。180度より大きくはならないはずです。