目次

ファイル入力 / 出力の使用 1
ファイル入力 / 出力の使用: イントロダクション 1
基本的ファイル入力/出力概念の理解 2
ファイル 入力/出力ダイアログ ボックスの使用 3
読み込み、または、書き込み用にファイルを開く 4
File Openコードのサンプル 5
読み込み、または、書き込み後、開いたファイルを閉じる 6
File Closeコードのサンプル 7
ファイルにある活字を読み込み 7
Read Characterコードのサンプル 8
ファイルにある行を読み込み 10
Read Lineコードのサンプル 11
サンプルコードが数字を含む先行ゼロへの対処 13
ファイルにあるテキスト ブロック読み込み 15
Read Blockコードのサンプル 16
区切り記号までテキストを読み込み 18
Read Up Toコードのサンプル 20
ファイルに文字を書き込み 21
Write Characterコードのサンプル 22
ファイルに行を書き込み 23
Write Lineコードのサンプル 24
ファイルにテキスト ブロックを書き込み 25
Write Blockコードのサンプル 27
ファイルの冒頭にファイルポインターを位置付け 28
Rewind to Startコードのサンプル 28
ファイルポインターの現在位置保存 30
Save File Positionコードのサンプル 30
保存されたファイルポインターの位置呼び出し 32
Recall File Positionコードのサンプル 33
ファイルのコピー 34

File Copyコードのサンプル 35
ファイルの移動 36
File Moveコードのサンプル 37
ファイル削除 38
File Deleteコードのサンプル 39
ファイルの存在チェック 40
File Existsコードのサンプル 41
ファイル ダイアログ ボックスの表示 42
File Dialogコードのサンプル 43
ファイルの終端、または、ラインの終端をチェック 44
EOF及びEOLコードのサンプル 44

ファイル入力 / 出力の使用

ファイル入力 / 出力の使用: イントロダクション

本章はパーツ・ルーチンからの出力かパーツ・ルーチンへの入力の方法を説明します。利用可能なメニューオプションは読み取りまたは書き込みモードでファイルを開くことができます。データは、フォームから読み取ることができ、ファイルに書き込まれます。ファイルI / Oコマンドはデータに外部ファイルからの一部のルーチンで使用される読み取り可能にします。同様に、測定と公差情報はファイルにバックアップにでき、これらのコマンドを使用して書き込むことができます。これらのコマンドを使用して、他のファイル操作を実行することができます。

本章はこれらのファイルI/O操作を詳しく述べて、各さまざまな操作要素の例が含まれます。これらのサンプルは「フローコントロールを使っての分枝」章と「式及び変数の使用」章の説明された項目を使用します。

この項目で述べられている主なトピックは、次の通り:

- 基本的ファイル入力/出力概念の理解
- ファイル 入力/出力ダイアログ ボックスの使用
- 読み込み、または、書き込み用にファイルを開く
- 読み込み、または、書き込み後、開いたファイルを閉じる
- ファイルにある活字を読み込み
- ファイルにある行を読み込み
- ファイルにあるテキスト ブロック読み込み
- 区切り記号までテキストを読み込み
- ファイルに文字を書き込み
- ファイルに行を書き込み
- ファイルにテキスト ブロックを書き込み
- ファイルの冒頭にファイルポインターを位置付け
- ファイルポインターの現在位置保存

- 保存されたファイルポインターの位置呼び出し
- ファイルのコピー
- ファイルの移動
- ファイル削除
- ファイルの存在チェック
- ファイル ダイアログ ボックスの表示
- ファイルの終端、または、ラインの終端をチェック

基本的ファイル入力/出力概念の理解

ファイルの存在をチェック:
すべてのI / O操作をファイルについて、おそらくファイルの存在を最初にチェックしたいです。チェックば失敗になった場合、これは IF / THENループを入力され、ユーザに通知できます。ファイルに書き込める時には,最初のウィンドウ環境内のファイルを作成します。

「ファイルの存在チェック」を参照してください。

ファイルのオープンとクローズ:
操作は、ファイルの読み取りまたはファイルへの書き込みについて、
最初にシステムのプロセスにオープンする必要があります。ファイル・ポインタと呼ばれる変数にファイルを割り当てることによって、これをします。ファイルを開くとき、ユーザは、ファイルが読み込み、(上書き）または追加のために開かれるかどうか指定できます。開封後は、ファイルにその時から読むか書き込むことができます。ファイルでの作業が終了している場合は、ファイルポインタを閉じる必要があります。このファイルを閉じ、他のシステムプロセスによってアクセスすることができます。既に別のプロセスによって開かれてファイルを開くことはできません。

「読み込み、または、書き込み用にファイルを開く」と「読み込み、または、書き込み後、開いたファイルを閉じる」を参照してください。

ファイル・ポインタと位置:
ファイルポインタ変数は、そのファイルをポイントします。彼らは開かれたファイルの名前と場所を保存して、そのファイルに読み込むか、または書くのに使用されています。一度ファイルを開いてファイルポインタの位置に設定すると、ポインタのカーソルのように振る舞うワードプロセッサは動作します。そこには、現在からの読み込みまたはファイルに書き込むことを示します。

- ファイルに追加していれば、ファイル・ポインターは通常ファイルの端にあります。

- ファイルを読み込む、またはファイルを上書きする場合、ファイルポインタは通常ファイルの開始位置にするべきです。

書き込み、または、読み込み時の区切り記号の使用

データを書き込める時、データの別の作品に区切りを使用のを参照してください。測定ルーチンにデータを読み込めるのを簡単になります。区切りは任意の文字または文字列を指定できます。例えば、2.5,4.3,6.1のX、Y、及び、Zの測定値を持つ、PNT1という点がある、とします。以下に示されたコードのように、コンマの区切り記号によってこれらの値を分離し、データファイル内に書き込みことが容易にできます:

```
FILE/WRITELINE,FPTR,PNT1.X + "," + PNT1.Y + "," + PNT1.Z
```

データを読み込める時に、
指定された区切りに基づく受信データを分離して後の操作の変数にデータを配置します。たとえば、リストされる同じな X、
YとZ値に読み込めるのを仮定します。その値は、テキスト表示の単一行に、このように置かれます:
2.5,4.3,6.1。以下に類似の、一行のコードを用いて、そのテキストをコンマで分離し、それらの値を対応する変数内に配置することができます:

```
V1=FILE/READLINE,FPTR,{ValX}+","+{ValY}+","+{ValZ}
```

測定ルーチンに`ValX`、`ValY`と`ValZ`を通常の変数として使用できます。結果は次のようになります: ValX = 2.5, ValY = 4.3,及びValZ = 6.1。

ファイル 入力/出力ダイアログ ボックスの使用

すべてのファイル I/O コマンドはI / Oのメニューオプション (**挿入|ファイルの入力/出力コマンド**を選択する)の適切なファイル選択して最初の測定ルーチンに挿入されます。コマンドは編集ウィンドウに存在する場合、
コマンドのF9を押してその関連した**ファイルI/O** ダイアログ ボックスにアクセスできます。

ファイル入力/出力ダイアログ ボックスのサンプル

このダイアログボックスは単に *現在* のI / Oコマンドファイルを編集する視覚的な方法を提供します。代わりに、「編集ウインドウの使用」章で説明された手法で、編集ウインドウ内のコマンドを変更できます。

ダイアログ ボックスを使用しなくても *新しい* ファイルI/Oコマンドを挿入できます。適切なメニューオプションを選択して、または、そのコマンドを直接編集ウィンドウ内にタイプ入力して、これを行う必要があります。

読み込み、または、書き込み用にファイルを開く

挿入 | ファイル I/O コマンド | ファイルを開く メニューオプションによって、ユーザーは測定ルーチン実行時にディスクファイルを開くコマンドを編集ウィンドウに挿入できます。

単に、情報の閲覧や、または、情報の追加、保存のために、ファイルを開くことが可能です。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<filepointername> =ファイル/開く,<filename>,<openmode>
```

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
開かれたファイルにアクセスするために使用する、ユーザーが選択したIDとファイルポインタです。このIDは他のファイル入出力コマンドでファイルを開くために参照されます。

<ファイル名>
開く対象のディスクファイル名です。

<開くモード>
これはファイルを開く時のモードです。読み込み、書き出し、または追加のモードの時にファイルを開くことができます。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Openコマンド上に配置して下さい。
3. [F9]を押します。

File Openコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

このコードは、読み取り、書き込み、および追加用のTEST.TXTという名前のファイルを開きます。これは`FPTR`という名前のファイルポインタにファイル名を格納します。

```
FPTR=FILE/OPEN,C:\PCDMISW\TEST.TXT,READ
FPTR=FILE/OPEN,C:\PCDMISW\TEST.TXT,WRITE
FPTR=FILE/OPEN,C:\PCDMISW\TEST.TXT,APPEND
```

コメントを使用して入力としてフルパスを取り、それを`FILE/OPEN`
コマンドに使用することを注意してください。`FILE/DIALOG`
で同じことを行うことができます。以下の例を参考にして下さい:

```
C1=COMMENT/INPUT,ファイルへの絶対パス、及び、ファイル名をタイプ入力して下さい。
V1=FILE/DIALOG, 開くファイルを選択して下さい。
FPTR=FILE/OPEN,C1.INPUT,READ
FPTR=FILE/OPEN,V1,READ
```

「ファイル ダイアログ ボックスの表示」を参照してください。

読み込み、または、書き込み後、開いたファイルを閉じる

挿入 | ファイル I/O コマンド |
ファイルを閉じるメニューオプションを使用すると、測定プログラム実行時に開いたファイルを閉じるコマンドを編集ウィンドウに挿入できます。ファイルを閉じると、ファイルが開いているときに使用されるリソースを解放して、ファイルに行われた変更内容をディスクに保存します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
File/Close, <filepointername>,<closemode>
```

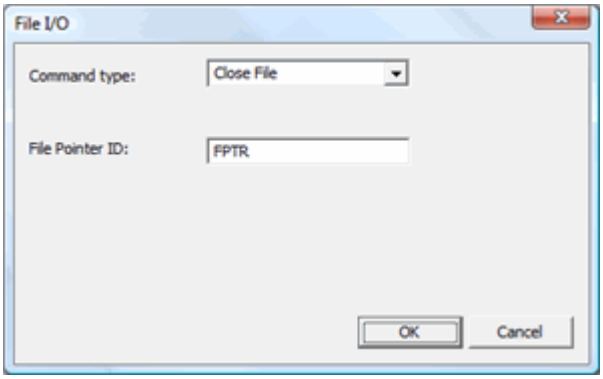
このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルを指定するために使用するIDであり、ファイルを開く時に作成されます。

<閉じるモード>
このパラメータには保持および削除の2つのオプションがあります。保持を使用するとPC-DMIS
は単にファイルポインタで定義されたファイルを閉じます。削除を使用すると、PC-DMISはファイルを閉じてからそれを削除します。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Closeコマンドに置いてください。
3. [F9]を押します。



File Closeコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

このコードはファイルポインタに割り当てられるファイルを簡単に終了します、`FPTR:`

```
FILE/CLOSE,FPTR,KEEP
```

`DELETE`パラメータを用いる、このコードは、`FPTR`に割り当てられたファイルを閉じて削除します:

```
FILE/CLOSE,FPTR,DELETE
```

ファイルにある活字を読み込み

挿入 | ファイル I/O コマンド | 読み込みコマンド|
文字の読み込みメニューオプションはfilepointernameフィールド（以下の構文を参照してください）で指定されたファイルから1つの文字を読み取るおよび変数変数名]フィールドで指定されたその文字を割り当てる編集ウィンドウでコマンドを置きます。それは、その性格を変数名フィールドで指定されている変数に割り当てます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname> = File/ReadCharacter,<filepointername>
```

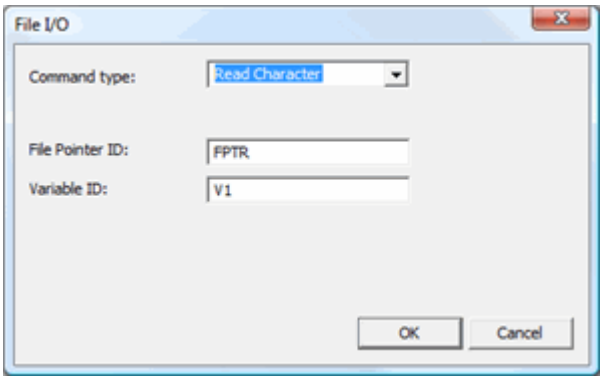
このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルを開くために使用するIDです。

<変数名>
その文字を保持する変数の名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルを文字の読出しコマンド上に配置して下さい。
3. [F9]を押します。



Read Characterコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

ある時にスペースを検出するまでデータファイルの1つの文字からのラインを読み込めるサンプルを参照します。

```
V1=FILE/EXISTS,test.txt
IF/V1<>0
COMMENT/OPER,データ
ファイルから読み込みを行うことができます。続けるには、「OK」をクリックして下さ
い。
ASSIGN/V3=""
FPTR=FILE/OPEN,D:\Program Files\pcdmis35\test.txt,READ
DO/
```

```
V2=FILE/READ_CHARACTER,FPTR
ASSIGN/V3=V3+V2
UNTIL/V2==" "
FILE/CLOSE,FPTR
COMMENT/OPER,「ファイルのテキストの行の第一語は: 」 + V3
END_IF/
ELSE/
COMMENT/OPER、データファイルから読み込めることはできません。ルーチンは現在終了します。
GOTO/END
END_ELSE/
END=LABEL/
ROUTINE/END
```

コードの説明

V1=FILE/EXISTS

このラインでは、指定されたファイルの存在をチェックします。そのファイルは、PC-DMISがこのコードの機能のために作動するディレクトリに位置する必要があり、それ以外の場合には、このファイルを含むラインが、ファイルへの絶対パスをも含んでいなければなりません。V1 は当ファイル
チェックの結果を受け取ります。ファイルが存在する場合は、ゼロ以外の値で、さもないとは、0です。

IF/V1<>0

このラインは V1
の値を取り、ゼロ以外の値を評価されるかどうかを検査します。もしそうなら、コメントは、それが読み取りプロセスを開始する準備ができて意味が表示されます。ゼロに等しい場合、測定ルーチンが終了します。

ASSIGN / V3 = ""

この行は、空の文字列を作成し、
これをV3に割り当てます。このコードは、個々の文字の読出しから文字列を構築するために、この変数を使用しています。空の文字列を作成しない場合には、V3は0の既定値を持ちます。

FPTR=FILE/OPEN

このラインは読込めに指定されるファイルを開いて、デフォルトファイルポインタFPTRに割り当てます。

DO

このラインはDO / UNTIL ループを開始します。それはFILE/READ_CHARACTER
コードの範囲をして、
ある時には文字が継続的に1つの読み取ります。文字の空間で読み取るたびにループは終了します。

V2=FILE/READ_CHARACTER,FPTR
このラインは開いているファイルのファイルポインタ、FPTRに関連付けられてから文字を読み込みます。文字が変数V2に保存されます。
ASSIGN/V3 = V3+V2
このラインは、空白の変数 V3を用いて、文字列V3
をV2に結びつけ、その後、V3にその値を再割り当てします。それで、DO/UNTILループの以降の実行で、V3にもうひとつの文字が追加されます。
UNTIL/V2 == " "
FILE/READ_CHARACTER
コードは開いているファイルから空白文字を検査する場合、このラインはDO / UNTILループを終了します。
FILE/CLOSE,FPTR
このラインは開かれたデータファイルを閉じて、他のシステムプロセスによってアクセスすることができます。コードの残りの部分が実行を完了し、データ
ファイルの第一語を、オペレーターのコメント内に表示します。

ファイルにある行を読み込み

挿入 | ファイル I/O コマンド | 読み込みコマンド | ラインの読み込み
メニューオプションは実行時に指定したファイルからラインを読み取る編集ウィンドウでコマンドを位置します。このコマンドは、変数IDによって特定された変数を 1 (真)、または、 0 (偽) に設定し、成功 (真)、または、失敗 (偽)
を示します。このコマンドが必要な式は読み込みの設定、自動的に変数
を埋める及びデータ内のファイルから読み取りを参照することに使用されます。情報は、入力ファイルから次のキャリッジ リターンの活字まで読み込まれます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname> = File/ReadLine,<filepointername>,<expr>
```

このコマンドのコンポーネントのいくつかについて説明します。

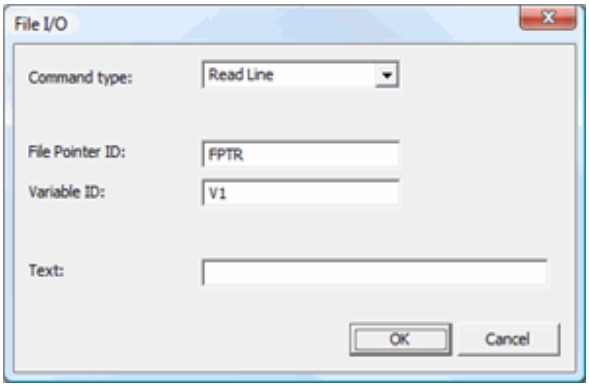
<変数名>
読み込み行コマンドの成功または失敗を示す結果を保持する変数の名前です。OKまたはEOFを返します。

<ファイルポインタ名>
ファイルが開かれた時にファイルポインタを指定する名前です。

<説明>
これは入力データのあて先変数です。入力データはテキストによって区切られ、データの入力行のパージが容易になります。変数および要素のリファレンスは波括弧で囲まれていなくてはなりません。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. 行を読み取りコマンドの上にカーソルを置きます。
3. [F9]を押します。



Read Lineコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

FILE/READ_LINEコマンドが空の行に到達するまでにある時にデータファイルの一行から一行ずつを読み込む例を参照します。測定ルーチンはテキストの結果ブロックを表示して終了します。

```
V1=FILE/EXISTS,test.txt
IF/V1<>0
COMMENT/OPER,データ
ファイルから読み込みを行うことができます。続けるには、「OK」をクリックして下さい。
ASSIGN/V3=""
FPTR=FILE/OPEN,D:\Program Files\pcdmis35\test.txt,READ
DO/
V2=FILE/READLINE,FPTR,{LINE}
ASSIGN/V3=V3+LINE
COMMENT/OPER,「現時点での変数V3の値は:
```

```
,V3
UNTIL/V2=="EOF"
FILE/CLOSE,FPTR
COMMENT/OPER,「テキスト ブロックは、以下のように読まれます: 」
,V3
END_IF/
ELSE/
COMMENT/OPER、データファイルから読み込めることはできません。ルーチンは現在終了します。
GOTO/END
END_ELSE/
END=LABEL/
ROUTINE/END
```

コードの説明

このコードの多くは「 Read Characterのサンプルコード」で説明したのと似ています。ここでは、この例に独自のコードについてのみ、説明を行います。

DO/

このラインはDO / UNTIL ループを開始します。ItそれはFILE/READ_LINE コードの範囲をして、
ある時にはラインが継続的に1つの読み取ります。ループは、ファイルの終端に達した時、閉じられます。

V2=FILE/READLINE,FPTR,{LINE}

キャリッジリターンが発生するまで、この行はすべてのテキストを読み取ります。FILE/READ_CHARACTERのようで、V2でこのテキストを保存することの代わりに、このコードは異なる役割を果たします。

- この場合のV2は2つの値、つまり"OK"あるいは"EOF"のいずれかを返します。読み込む行がまだ存在する場合は"OK"を返します。ファイルの終わりまで達した場合は"EOF"を返します。
- {LINE}
  コードは実際のテキストを保存するユーザーの入力変数です。それは、波状カッコで囲まれ、それは変数であって、区切り文字の一部ではないことを、PC-DMISに告げます。括弧がなければ、PC-DMISが「LINE」という文字列を探して、「LINE」後とキャリッジリターンする前にテキストを返します。

ASSIGN/V3 = V3+LINE

この行は、空白の変数 V3を用いて、文字列V3

をLINEに結びつけ、その後、連結された値をV3に再割り当てします。それで、DO/UNTILループの後続の実行で、V3にもうひとつの行が追加されます。

```
UNTIL/V2 == "EOF"
```

このラインはDO / UNTIL
ループ条件を確認します。FILE/READLINEコードがファイルの終端に出会うと、ループが閉じます。いったんルーチン
フローがループを退くと、コードの残りの部分が実行を完了し、コード
ブロック全体を、オペレーターのコメント内に表示します。

Result = File/ReadLine,F1, "Part ID :" + {V1} - これは"パーツ ID :"テキストの後の読み込み行に表示されるすべてのテキストをV1に割り当てられることに起こします。その行は、ファイルポインター名として、F1を用いて開けられたファイルから読み込まれます。読み込みの結果（成功、または失敗）は、変数結果内に保存されます。

```
File/ReadLine,F1,"Location:"+{VARX}+","+{VARY}+","+{VARZ}+","+{VARI}+"
,"+{VARJ}+","+{VARK}
ASSIGN/CIR1.XYZ=MPOINT(VARX,VARY,VARZ)
ASSIGN/CIR1.IJK=MPOINT(VARI,VARJ,VARK)
```

上記の3つのコマンドラインは、「位置:」文字列の後にカンマ区切りのテキストを読み込み、この値をCIR1のX 、Y 、Z 、およびI、J 、K値にを格納します。

File/ReadLine,F1, "Value # " + loopvar + " : " + {var2} - これはvar2には、コロンの後に表示されるテキストで記入するようになります。この例のloopvar変数は、波状カッコで囲まれておらず、その結果、区切り文字に含まれます。

サンプルコードが数字を含む先行ゼロへの対処

数のあなたが読んでいるファイルが数の系列を含んでいると、そのPC-DMISがキャラクタに全く先行しないことで無視するのに気付きます。例えば、あなたのラインが005450の値を含んでいるなら、厳密にこの値を数と読んで、2つの前のゼロを無視して、5450年の値を化します。これを望むか望まない場合があります。

テキストを外部のバーコードリーダーソフトウェアによって作成されたファイルがあると仮定して、それはデータの２ラインを含まれます：

290291143;582750;0010

291143;5827;0010

いくつかの簡単なコードを使用して、セミコロンの間に番号の値を取得するはずです：

```
ASSIGN/FIRST_VALUE=0
ASSIGN/SECOND_VALUE=0
```

```
     ASSIGN/THIRD_VALUE=0
     ASSIGN/LINENUM=1
FPTR=FILE/OPEN,D:\TEMP\CODES.TXT,READ
     DO/
INLINE=FILE/READLINE,FPTR,{FIRST_VALUE}+";"+{SECOND_VALUE}+";"+{TH
IRD_VALUE}
     COMMENT/OPER,NO,"LINE NUMBER: "+LINENUM
          ,"最初の値: " + FIRST_VALUE
          ,"二番目の値: " + SECOND_VALUE
          ,"三番目の値: "+THIRD_VALUE
     UNTIL/INLINE=="EOF"
     FILE/CLOSE,FPTR,KEEP
```

これはテキスト行を問題なくパージして数値を返しますが、返した値として先行するゼロの値を削除します。つまり、THIRD_VALUE変数には 0010 ではなく 10 が含まれます。

前述のゼロの値を保持するには、テキスト行でセミコロンの位置の場所を見つけて、数の値を得るのに全体のラインをストリングとして扱って、代わりにINDEX、LEFT、およびMID文字列関数を使用する必要があります:

```
FPTR=FILE/OPEN,D:\TEMP\CODES.TXT,READ
     ASSIGN/LINENUM=1
     DO/
LINESTATUS=FILE/READLINE,FPTR,{LINESTR}
     ASSIGN/LINESTR=STR(LINESTR)
     ASSIGN/FIRST_INDEX=INDEX(LINESTR,";")
     ASSIGN/FIRST_VALUE=STR(LEFT(LINESTR,FIRST_INDEX-1))
     ASSIGN/REMAINSTR=STR(MID(LINESTR,(FIRST_INDEX)))
     ASSIGN/SECOND_INDEX=INDEX(REMAINSTR,";")
     ASSIGN/SECOND_VALUE=STR(LEFT(REMAINSTR,SECOND_INDEX-1))
     ASSIGN/THIRD_VALUE=STR(MID(REMAINSTR,SECOND_INDEX))
     COMMENT/OPER,NO,"LINE NUMBER: "+LINENUM
          ,"最初の値: " + FIRST_VALUE
          ,"二番目の値: " + SECOND_VALUE
          ,"三番目の値: "+THIRD_VALUE
     ASSIGN/LINENUM=LINENUM+1
     UNTIL/LINESTATUS=="EOF"
     FILE/CLOSE,FPTR,KEEP
```

コードの説明

このコードの多くが上で説明されることと同様です。説明された文字列関数に特有のコード説明だけがここにリストアップされています。

ASSIGN/FIRST_INDEX=INDEX (LINESTR,";")
この行は行で最初のセミコロンの位置の場所を見つけて、FIRST_INDEX変数に割り当てます。

ASSIGN/FIRST_VALUE=STR(LEFT(LINESTR,FIRST_INDEX-1))
このラインはFIRST_VALUE変数に文字列を割り当て、LINESTR変数内の最初のセミコロンを含まりません。LINESTR は全体のテキスト行を含んでいます。

ASSIGN/REMAINSTR=STR(MID(LINESTR,(FIRST_INDEX)))
ラインの終わりまで、このラインは、REMAINSTR変数（「残っているストリング」を当てはめます）にFIRST_INDEX
位置（最初の半コロンの位置）から始まっている文字列を割り当てます。

ASSIGN/SECOND_INDEX=INDEX (REMAINSTR,";")
別のセミコロンがREMAINSTR変数で探されて、（2番目のセミコロン）SECOND_INDEX変数の位置を割り当てます。

ASSIGN/SECOND_VALUE=STR(LEFT(REMAINSTR,SECOND_INDEX-1))
このラインはSECOND_VALUE変数に文字列を割り当て、REMAINSTR変数内の最初のセミコロンを含まりません（全体のラインの2番目のセミコロン）。

ASSIGN/THIRD_VALUE=STR(MID(REMAINSTR,SECOND_INDEX))
この行は、SECOND_INDEX位置から行末までの文字列のTHIRD_VALUE変数を割り当てます。

ファイルにあるテキスト ブロック読み込み

挿入 | ファイル I/O コマンド | 読み込みコマンド | ブロックの読み込み
メニューオプションは実行時に開いているファイルから文字のブロックの読み込み編集ウィンドウでコマンドを位置します。読み込む活字量は、サイズ パラメータによって指示されます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname>=File/Read_Block,<fptrname>,<size>
```

このコマンドのコンポーネントのいくつかについて説明します。

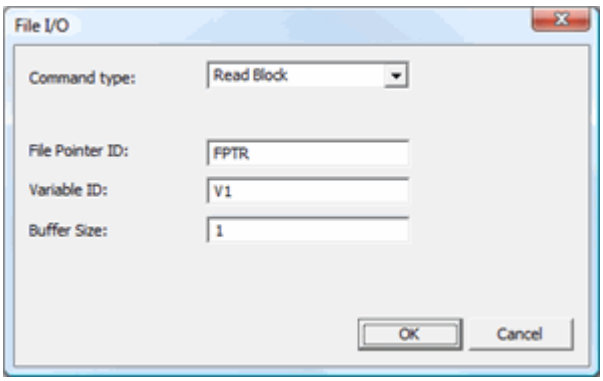
<変数名>
読み込みブロック操作の成功または失敗を示す値を受け取る変数の変数IDです。

<ファイルポインタ名>
ファイルが開かれた時にファイルポインタを指定する名前です。

<サイズ>
読み込む文字数です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. ブロックを読み取りコマンドの上にカーソルを置きます。
3. [F9]を押します。



Read Blockコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

を含む様々な外部のデータファイルがあるのを仮定して、その他の部分のデータと各ファイルがどのようなファイルを指定した最初の数文字を含みます。`File/Read_Block` コマンドを使用して、すべてのラインを読み込むと処理を決定する前に最初の数文字だけを読みます。このコードを参考にして下さい:

```
C3=COMMENT/INPUT,サーチを行うファイル　コードを
タイプ入力して下さい。
ASSIGN/BLOCKSIZE=LEN(C3.INPUT)
ASSIGN/FILECODE=C3.INPUT
DO/
C1=COMMENT/INPUT,ファイルへの絶対パス、
、ファイル名、及び、
```

```
処理したいファイルへの拡張子をタイプ入力して下さい。
、停止するには [Q] をタイプ入力してください。
IF/C1.INPUT=="Q" OR C1.INPUT=="q"
COMMENT/OPER、終了を選択しました。今ルーチンが終了します。
GOTO/END
END_IF/
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
COMMENT/OPER,"データ・ファイル [" + C1.INPUT + "]
が存在します。続けるには、「OK」をクリックして下さい。
FPTR=FILE/OPEN,C1.INPUT,READ
V2=FILE/READ_BLOCK,FPTR,BLOCKSIZE
FILE/CLOSE,FPTR
IF/V2<>FILECODE
COMMENT/OPER,「ファイルの["+ V2 +"]コードが一致しません」
,"["+FILECODE+"]のFILECODE"
END_IF/
UNTIL/V2==FILECODE
COMMENT/OPER," ["+C1.INPUT+"] ファイルは一致です。"
,"ファイルの["+ V2 +"]コードは一致します"
,"["+FILECODE+"]のFILECODE"
COMMENT/OPER、ルーチンはそのファイルを処理します。
END_IF/
ELSE/
COMMENT/OPER,"データ・ファイル[" + C1.INPUT + "]
は、指定場所に存在しません。既存のデータファイルを使用して再試行します。"
GOTO/END
END_ELSE/
END=LABEL/
ROUTINE/END
```

コードの説明

いくつのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

ASSIGN/BLOCKSIZE = LEN(C3.INPUT)
このラインは、BLOCKSIZEというユーザー定義の変数を使用し、それは、C3.INPUTにある文字数を示す整数を含んでいます。これは、読み込む対象となる文字ブロックのサイズとして用いられます。
ASSIGN/FILECODE = C3.INPUT
このラインはFILECODE 変数を作成してまたそれにC3.INPUTの値を与えます。
C1=COMMENT/INPUT
このコマンドはユーザで入力したフルパスをC1.INPUT 変数に保存します。
V1=FILE/EXISTS,C1.INPUT
このラインはC1 コマンドに定義されたファイル名の存在を確認します。
DO/
このラインはDO / UNTIL
ループを開始します。ユーザからの読み取りにファイルを指定できるコードブロックの範囲をします。テキストはFILECODE
変数に割り当てるテキストファイルから読み取ると一致するまでは、ループを続行します。
V2=FILE/READ_BLOCK,FPTR,BLOCKSIZE
このラインは文字の整数のBLOCKSIZE
変数に含まれているに等しい量を読み取ります。テキストはV2 変数に保存されます。
IF/V2FILECODE
この行はV2変数内のテキストが FILECODE
変数に保存されたテキストと一致するかを検査するコードブロックのIF
/端が始まります。それが一致しない場合、ルーチンは動作し続けます。それ以外の場合は、メッセージを2つのコードが一致しないと表示されます。
UNTIL/V2==FILECODE
このラインは、V2変数内のテキストが、FILECODE変数のテキストと一致するかを検査する、DO /
UNTILループの条件をチェックします。そのステートメントの評価が偽と出る場合、DOループが再び実行され、それによって、ユーザーは異なるファイル名を選ぶことができます。そのステートメントの評価が真と出る場合、ループが閉じ、ルーチンは、一致した旨を告げるメッセージを表示します。その後、PC-DMISは、指定されたデータ
ファイルのデータの各行の読み込みを続けます。

区切り記号までテキストを読み込み

挿入 | ファイル I/O コマンド | 読み込みコマンド | 読み込み終了点
メニューオプションは指定されたファイルの実行中にから指定された区切り記号の全体テキストの「終点」を読み込む編集ウィンドウでコマンドを位置します。このコマンドで読み取れらる任意のテキストは指定された先の変数に位置されます。PC-
DMISが次と遭う場合、コマンドはテキストの読み取りを停止します：

- あらかじめ定義された区切り記号

- キャリッジ リターン
- 改行文字

ファイルの終端に達すると、送付先の変数は「EOF」（ファイルの終端）に設定されます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname> = FILE/READ_UPTO,<fptrname>,<delimiters>
```

このコマンドのコンポーネントのいくつかについて説明します。

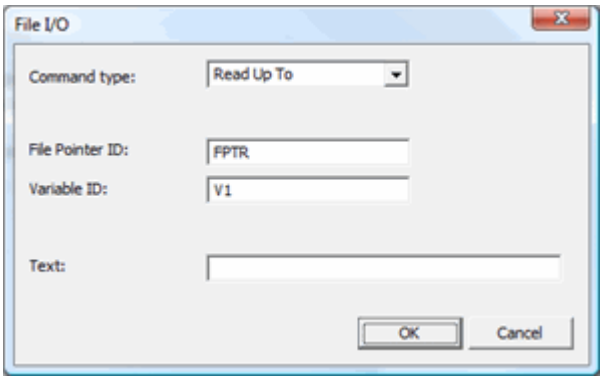
<変数名>
あて先変数の名前です。

<ファイルポインタ名>
ファイルが開かれた時にファイルポインタを指定する名前です。

<区切り文字>
ゼロ個以上の区切り文字を含む文字列です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. **編集ウィンドウ**を選択して編集ウィンドウを開いて下さい。
2. FILE/READ コマンド上にカーソルを配置して下さい
3. [F9]を押します。**ファイル 入力/出力** ダイアログ ボックスが開きます。



ダイアログ ボックスが現れると:

1. **変数ID** ボックス内に、読み込み情報を受け取る変数の変数名をタイプ入力して下さい。
2. **ファイル ポインターID** ボックスにファイル ポインター名を入力します。
3. **テキスト**ボックスに区切り文字を入力します(選んだ区切りの前後に引用符を使用するのを確認します)。
4. **[OK]** をクリックします。

Read Up Toコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

最初の行にこの情報を含むc：\ tempディレクトリに"sample.txt"という名前のテキストファイルがあるこの例を考えてみましょう。

```
root:x:0:0:root:/root:/bin/bash
```

このファイルで、Read Up Toコマンドを使用するには:

1. 編集ウィンドウ内にFILE/OPENコマンドを挿入します。
2. 選択のファイルポインター名を使用して、File Openコマンドを名付けて下さい。この例では、ファイルポインター名として、「サンプル」を使用します。

File Openコマンドは、以下のようになります:

```
SAMPLE =FILE/OPEN,C:\TEMP\SAMPLE.TXT,READ
```

ここで、PC-DMISのRead Up Toコマンドを用い、データの異なる部分を呼び出す、いくつかの変数を定義します。この例では、区切り記号としての「:」（引用符なしで）を探す、以下の変数を使用します。

```
USERNAME =FILE/READ_UPTO,SAMPLE,:
PASSWORD =FILE/READ_UPTO,SAMPLE,:
USER =FILE/READ_UPTO,SAMPLE,:
```

これにより、PC-DMISがこれらのラインを実行すると、以下のように設定されます

ユーザー名 = ルート

パスワード = x

ユーザー = 0

実行中、スクリーン上にこれを表示するには、ここで示されたような、オペレーターコメントを使用することが可能です:

```
COMMENT/OPER、次のテキストはsample.txtから読み込めます。
,Username:
,USERNAME
,Password:
,PASSWORD
,ユーザ:
```

```
,USER
```

ファイルに文字を書き込み

挿入 | ファイル I/O コマンド | 書き込みコマンド | 文字書き込み メニュー
オプションを選ぶと、編集ウィンドウ内にコマンドを挿入でき、そのコマンドを用いると、実行に当たって、単一の文字がディスク ファイルに出力されます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
File/Write_Character,<fptrname>,<expr>
```

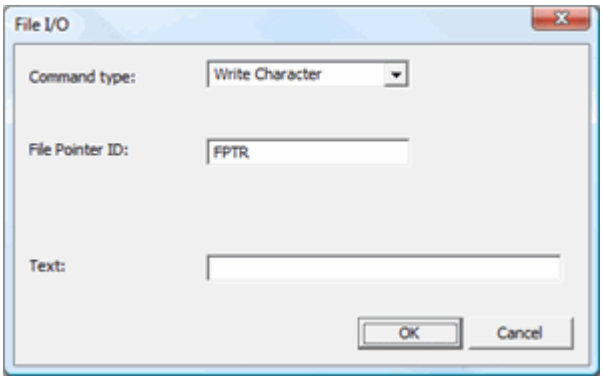
このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルが開かれた時にファイルポインタが指定する名前です。

<式>
ファイルに書き込まれる文字です。式が複数の文字を評価する場合、最初の文字のみが書き込まれます。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルを文字の書き込みコマンド上に配置して下さい。
3. [F9]を押します。



Write Characterコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ
ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

ある時にユーザーがデータファイルの一つの文字に提供した文字列を書き込むコードを参照してください。

```
C1=COMMENT/INPUT,書き込み先のファイル名をタイプ入力して下さい。
、(ファイルの絶対パスを含む)。
FPTR=FILE/OPEN,C1.INPUT,WRITE
C2=COMMENT/INPUT,ファイルに送付する何かをタイプ入力して下さい。
、これは文字列を1つの文字が送信されます
、文字列を送付します。
ASSIGN/COUNT=0
ASSIGN/LENGTH=LEN(C2.INPUT)
DO/
ASSIGN/WRITETHIS=MID(C2.INPUT,COUNT,1)
FILE/WRITE_CHARACTER,FPTR,WRITETHIS
ASSIGN/COUNT=COUNT + 1
UNTIL/COUNT==LENGTH
```

コードの説明

いくつのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FPTR=FILE/OPEN,C1.INPUT,WRITE
このラインはファイル追加のC2のコメントで指定されるを開き、ファイルポインタ、FPTRに割り当てます。このファイル内のすべてのデータはファイルポインタがデータの開始時に始まるように上書き限り取得します。
ASSIGN/COUNT = 0
このラインはユーザが定義変数COUNTにゼロの値を割り当てます。このループの目的文字列を一度に1つの文字を印刷するために使用されます。
ASSIGN/LENGTH = LEN(C2.INPUT)
このラインはLEN( )
関数を使用して文字列の長さを返します。この関数は、1つのパラメータ、文字列を受け取りま

す。それは文字列の文字（スペースを含む）の数を計算その金額の整数値を返します。この例にユーザーに定義された変数、LENGTH はこの値を保持します。

DO/

このラインはDO / UNTIL ループを開始します。DO及びUNTIL
ステートメントの間のコードはループの条件が真に評価されるまでに実行されます。

ASSIGN/WRITETHIS = MID(C2.INPUT,COUNT,1)

このラインはユーザで定義されたWRITETHISという変数を作成してMID( )
関数を使用して、C2.INPUT
文字列からのサブストリングしている文字を返して、またそれをWRITETHISに与えます。
MID( ) 3つのパラメータを受け取ります。

- パラメータ 1: は値の取得元となる文字列です。この場合、C2.INPUT
  が使用されます。
- パラメータ 2:
  は文字列の取得元となる文字列の位置を表します。文字列の最初の文字は位置0
  、2番目は位置2、という具合に続きます。この場合、変数 COUNT
  が使用されます。
- パラメータ 3:
  は取得する2番目のパラメータの位置から開始する文字数を表します。この場合
  、1の値が使用されます(一度に1つの文字のみ書き込まれるため、それ以上取得
  する理由がない)。

FILE/WRITE_CHARACTER,FPTR,WRITETHIS

このラインはファイル ポインター, FPTRで定義されたファイルにWRITETHIS
変数で保存する文字を書き込みます。

ASSIGN/COUNT = COUNT + 1

このラインは現在の**COUNT** 値を取り、 それを1つ増加して、COUNTに新しい値を配置します。

UNTIL/COUNT == LENGTH

このラインはDO / UNTIL ループ条件を確認します。この例で、ループは LENGTH
変数と同じ値になるまで COUNT
変数へのインクリメントを続けます。ループは終了して、ルーチンを終了します。

ファイルに行を書き込み

挿入 | ファイル I/O コマンド | 書き込みコマンド | ブロックを書き込み メニュー
オプションを選ぶと、編集ウィンドウ内にコマンドを挿入でき、そのコマンドを用いると、実行に当たって、テキスト行がディスク
ファイルに出力されます。変数情報、及び、測定ルーチン情報をファイルに出力するには、式シンタックスを使用して下さい。キャリッジ
リターンが、書き出されたテキストに自動的に付け加えられます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
File/WriteLine,<ファイル ポインター名>,<入力式>
```

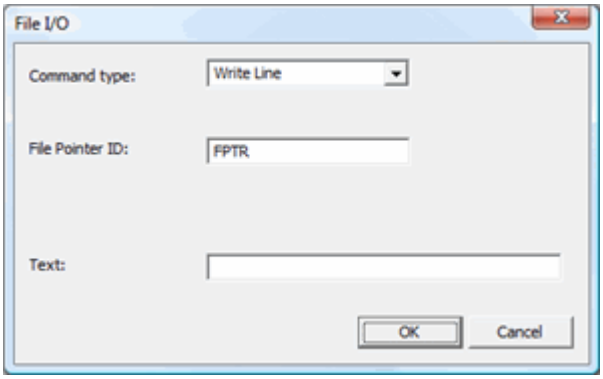
このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルが開かれた時に指定されたファイルリファレンスの名前です。

<式>
ファイルに書き込むテキストです。式は以下のフィールドで使用されます。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. 行を書き込みコマンドの上にカーソルを置きます。
3. [F9]を押します。



Write Lineコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ
ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

データファイルにいくつの測定XYZ
値をエクスポートをしようと仮定します。以下のコードを用いると、フィーチャー
ラベルとデータ
ファイルを入力することができ、さらに、そのフィーチャー用のX、Y、及び、Zのデータをデータ ファイルに送付することができます。

```
C1=COMMENT/INPUT,フィーチャーのラベルをタイプ入力して下さい。
,to use.
C2=COMMENT/INPUT,書き込み先のファイル名をタイプ入力して下さい。
```

```
、(ファイルの絶対パスを含む)。
FPTR=FILE/OPEN,C2.INPUT,APPEND
ASSIGN/FEATNAME=C1.INPUT
ASSIGN/ALLVALS=FEATNAME.X+","+FEATNAME.Y+","+FEATNAME.Z
COMMENT/OPER,「書き込むテキストは: 」 + ALLVALS
FILE/WRITELINE,FPTR,ALLVALS
FILE/CLOSE,FPTR
```

コードの説明

いくつのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FPTR=FILE/OPEN,C2.INPUT,APPEND
このラインはファイル追加のC2のコメントで指定されるを開き、ファイルポインタ、FPTRに割り当てます。代わりにする場合は、APPEND
をWRITEに変更して、データファイルの既存内容は上書きされます。
ASSIGN/FEATNAME = C1.INPUT
このラインはユーザで定義されたFEATNAME変数にC1.INPUTから要素ラベルの文字を割り当てます。
ASSIGN/ALLVALS=FEATNAME.X+","+FEATNAME.Y+","+ FEATNAME.Z
このラインはユーザで定義されたALLVALS
変数FEATNAME.X,FEATNAME.Y,FEATNAME.Zの値、その他の文字にC1入力内容に入力された要素ラベルのX、 YとZ 値を保持します。
FILE/WRITELINE,FPTR,ALLVALS
このラインはファイルポインタ FPTRで指定されたファイルにALLVALS
に含む値を書き込みます。

ファイルにテキスト ブロックを書き込み

挿入 | ファイル I/O コマンド | 書き込みコマンド | 行書き込み メニュー
オプションを選ぶと、編集ウィンドウ内にコマンドを挿入でき、そのコマンドを用いると、実行に当たって、テキスト行がディスク
ファイルに出力されます。変数情報、及び、測定ルーチン情報をファイルに出力するには、式シンタックスを使用して下さい。行の書き込みコマンドをと異なり、ブロックの書き込みは最後に改行を追加*していません*。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
File/WriteBlock,<fptrname>,<expr>
```

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルが開かれた時に指定されたファイルリファレンスの名前です。

<式>
ファイルに書き込むテキストです。式は以下のフィールドで使用されます。

行の書き込みコマンドをと異なり、ブロックの書き込みは最後に改行を追加*していません*。ユーザは、テキストブロック内の新しい行にテキストを配置する必要がある場合は、この例では、ここに示すようにしかし、あなたは、引用符で囲まれた文字列の外にCHR（10）コードを使用して手動でキャリッジリターンとラインフィードを挿入することができます。

```
FILE/WRITEBLOCK,FPTR, 「CHR(10) テキストを挿入... 」＋CHR(10) + 「
...新規行の中に。」
```

これにより、出力ファイル内のこの結果が得られます:

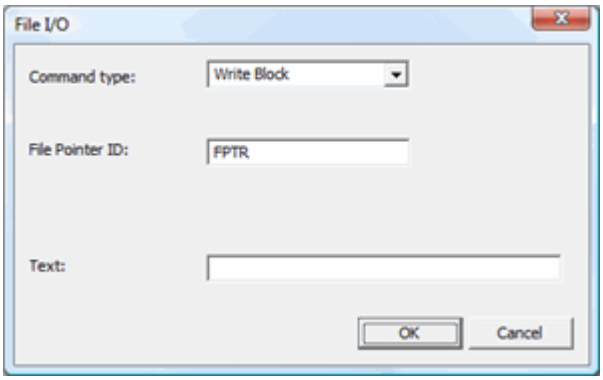
```
CHR(10) inserts text...

...新規の行に
```

CHR(10) は引用符の内側にあると、CHR(10) の実際のテキストがファイルに送られます。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. ブロックを書き込みコマンドの上にカーソルを置きます。
3. [F9]を押します。



Write Blockコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ
ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはどんな入力のコメントにユーザー入力、区切り記号として使用されるコロンを追加するのを書き込みます。

```
C1=COMMENT/INPUT,文字列をタイプ入力して下さい。PC-
DMISは、コロンを付け加え（区切り記号の目的で）、その文字列をご希望のファイルに
書き込みます。

C2=COMMENT/INPUT,書き込み先のファイル名をタイプ入力して下さい。

、(ファイルの絶対パスを含む)。

FPTR=FILE/OPEN,C2.INPUT,APPEND

ASSIGN/WRITETHIS=C1.INPUT+":"

COMMENT/OPER,「書き込むテキストは: 」 + WRITETHIS

FILE/WRITELINE,FPTR,WRITETHIS

FILE/CLOSE,FPTR
```

コードの説明

いくつのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FPTR=FILE/OPEN,C2.INPUT,APPEND
このラインはファイル追加のC2のコメントで指定されるを開き、ファイルポインタ、FPTRに割り当てます。
ASSIGN/WRITETHIS = C1.INPUT + ":"
この行は、C1.INPUTに含まれるテキストにコロンを付け加え、さらに、新規の文字列をユーザーに定義された変数WRITETHISに割り当てます。
FILE/WRITELINE,FPTR,WRITETHIS
このラインはファイルポインタ FPTRで指定されたファイルにWRITETHIS
に含む値を書き込みます。区切り記号としてコロンを使用して後でファイルのテキストから読むこめます。

ファイルの冒頭にファイルポインターを位置付け

挿入 | ファイル I/O コマンド | 位置コマンド | 開始点まで戻る
メニューオプションはコマンドをファイルポインタをファイルストリームの先頭に位置する編集ウィンドウに挿入します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

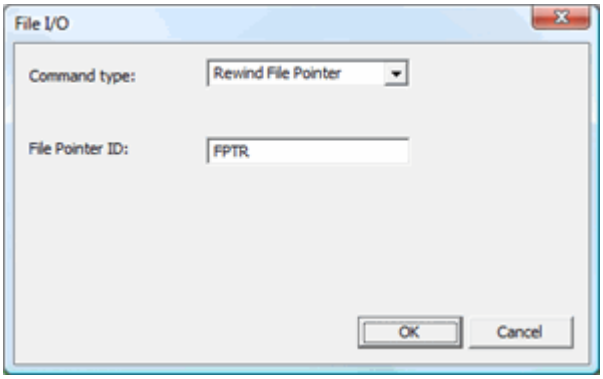
```
File/Rewind,<fptrname>
```

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイルの開始時に再配置するファイルポインタの名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをRewind to Startコマンド上に置きます。
3. [F9]を押します。



Rewind to Startコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

ある時に外部ファイルの１ラインからのデータを読み込めるサンプルを参照します。各行の読み込み後に、はじめに戻り、ファイルの冒頭から読み込みを行うオプションが利用可能です。これはFILE/REWIND コマンドの使用方法を示します。

```
C1=COMMENT/INPUT,読み込みの対象となるファイルをタイプ入力して下さい。
、(ファイルの絶対パスを含む)
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
DO/
FPTR=FILE/OPEN,C1.INPUT,READ
C2=COMMENT/YESNO,冒頭からの読み込みを行いたいですか?
IF/C2.INPUT == "YES"
FILE/REWIND,FPTR
END_IF/
V2=FILE/READLINE,FPTR,{LINE}
COMMENT/OPER,「現時点での行は: 」 + LINE
UNTIL/V2=="EOF"
END_IF/
FILE/CLOSE,FPTR
COMMENT/OPER,ルーチン終了中
```

コードの説明

いくつのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

C2=COMMENT/YESNO
このラインは初めからファイルの読込めを開始するのを確認します。これは YES/NO 応答をC2.INPUT変数に保存します。
IF/C2.INPUT == "YES"
この行はIF/ IFブロックを開始します。それはC2.INPUT にははいの値があるという条件をテストします。条件が真実の場合、 PC-DMISはIF文の後続の行を実行します。条件が偽である場合、PC-DMISはEND_IFステートメントの次のコードを実行します。
FILE/REWIND,FPTR
このラインは、ファイルポインターを、データ ファイルの冒頭に巻き戻します。
END_IF/
このラインは、「IF / END IF」コード ブロックの実行を中止します。

ファイルポインターの現在位置保存

挿入 | ファイル I/O コマンド | 位置コマンド |
ファイル位置を保存メニューオプションはファイルストリームの先頭にファイルポインタの現在の位置を保存するコマンドを編集ウィンドウに挿入します。保存された位置は後で呼び出しファイル位置コマンドに呼び出されます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

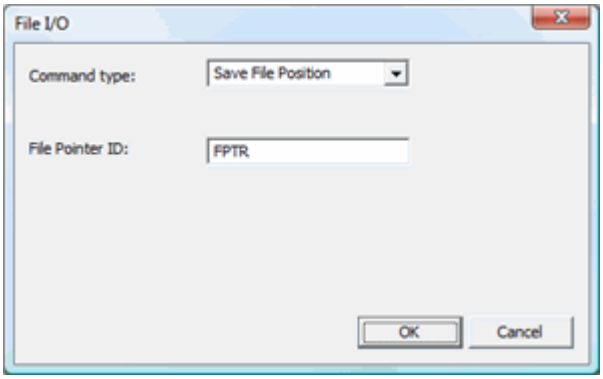
```
File/SavePosition,<fptrname>
```

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
ファイル位置を保存するファイルポインタの名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをSave File Positionコマンド上に移動して下さい。
3. [F9]を押します。



Save File Positionコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

ある時に外部ファイルの１ラインからのデータを読み込めるサンプルを参照します。各行の読み込み後に、今後の呼び出しのために、ファイルの位置付けを保存するオプションが利用可能です。これは`FILE/SAVE_POSITION` コマンドの使用方法を示します。

```
C1=COMMENT/INPUT,読み込みの対象となるファイルをタイプ入力して下さい。
、(ファイルの絶対パスを含む)
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
DO/
FPTR=FILE/OPEN,C1.INPUT,READ
C2=COMMENT/YESNO,ファイルの位置付けを保存し、後に呼び出しを行いますか?
ループは中止されます。
IF/C2.INPUT == "YES"
FILE/SAVE_POSITION,FPTR
GOTO/QUITLOOP
END_IF/
V2=FILE/READLINE,FPTR,{LINE}
COMMENT/OPER,「現時点での行は: 」  + LINE
UNTIL/V2=="EOF"
END_IF/
FILE/CLOSE,FPTR
QUITLOOP=LABEL/
COMMENT/OPER、読んで停止しました。
ROUTINE/END
```

コードの説明

このコードは「巻き戻しの開始ためのサンプルコード」で説明されたのと似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

`C2=COMMENT/YESNO`
このラインは現在のファイル位置を保存してループを終了するのかを確認します。これは YES/NO 応答を`C2.INPUT`変数に保存します。
`FILE/SAVE_POSITION,FPTR`
このラインは、ファイルポインターの位置を、ファイル ストリーム内に保存します。
同一のファイルを、同一のファイルポインターと共に、同一の測定ルーチン内で開いている限り、保存されたファイル位置を呼び出し、読み込みを中断した場所から、読み込みを継続できます。この例を継続するために、「Recall File Positionコードのサンプル」トピックを参照してください。

保存されたファイルポインターの位置呼び出し

挿入 | ファイル I/O コマンド | 位置コマンド |
ファイル位置呼び出し以前に保存したファイルの位置を呼び出す編集ウィンドウにコマンドを挿入します。開いたファイル内に位置付けを保存するには、ファイル位置を保存のコマンドを用いて下さい。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

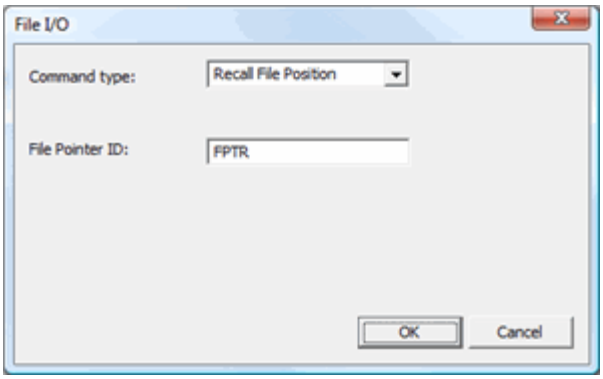
```
File/RecallPosition, <fptrname>
```

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ名>
位置を呼び出すファイルポインタの名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをRecall File Positionコマンド上に配置して下さい。
3. [F9]を押します。



Recall File Positionコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ
ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

このサンプルは以前のファイルポインタを使用して、以前に閉じたファイルを開き、保存されたファイルポインタの保存位置を呼び出されます。その後、その位置からデータを読み込みます。このコードはFILE/RECALL_POSITION
コマンドの使用方法を示します。「ファイルの保存場所のサンプルコード」のトピックに記載されているコードサンプルが続行されます。

```
COMMENT/OPER、ルーチンは保存されたファイルの位置を呼び出します。
FPTR=FILE/OPEN,C1.INPUT,READ
FILE/REWIND,FPTR
COMMENT/OPER,テストのために、ファイルが巻き戻されました。
、巻き戻しをテストするために、第一行が読み込まれます。
V3=FILE/READLINE,FPTR,{LINE}
COMMENT/OPER,第一行は:
、ライン
FILE/REWIND,FPTR
FILE/RECALL_POSITION,FPTR
COMMENT/OPER,以前保存されたファイルの位置付けが、呼び出されました。
、保存された位置付けにある、ライン上のデータが、ここで印刷されます。
V4=FILE/READLINE,FPTR,{STORED}
COMMENT/OPER,保存されたファイル位置でのテキストは:
,STORED
```

コードの説明

このコードは「巻き戻しの開始ためのサンプルコード」で説明されたのと似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FILE/RECALL_POSITION,FPTR
このラインは、FPTRとして指定されたファイルポインターの、ファイル
ストリーム内に保存されている、位置付けを呼び出します。
V4=FILE/READLINE,FPTR,{STORED}
このラインは保存されたファイルポインタの位置の後の次のラインを読込め、ユーザが定義し

たSTOREDの変数に割り当てます。それから、この変数は、次のオペレーター
コメント内に印刷されます。

ファイルのコピー

挿入 | ファイル I/O コマンド | ファイルをコピー メニュー
オプションは、実行時のファイルコピー操作を可能にするコマンドを編集ウィンドウに挿入します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
File/Copy,<srcfilename>,<destfilename>,<replacemode>
```

このコマンドのコンポーネントのいくつかについて説明します。

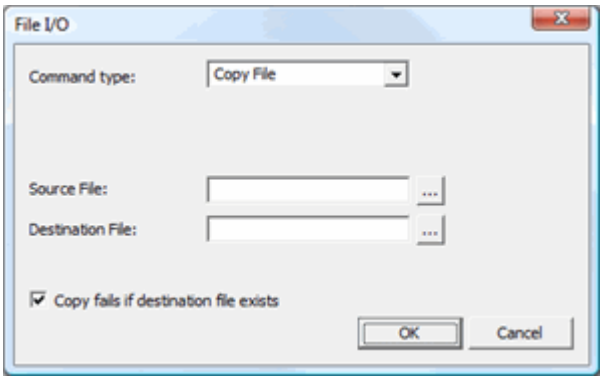
<srcfilename (ソースファイル名)>
ソースファイルの名前です(コピー元のファイル)

<destfilename (あて先ファイル名)>
あて先ファイルの名前です(コピー先のファイル)

<replacemode (置換モード)>
これはあて先ファイルがすでに存在する場合に取られるアクションです。あて先ファイルが存在する場合、上書きおよび失敗の2モードがあります。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Copyコマンド上に置きます。
3. [F9]を押します。



File Copyコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ
ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはファイル名をコピーし、コピー先のディレクトリとコピー先のファイルを要求します。

```
C1=COMMENT/INPUT,コピーしたいファイルの名称をタイプ入力して下さい。
、(完全なファイル経路を含む)
C2=COMMENT/INPUT,コピーの送付先となるファイルの名称をタイプ入力して下さい。
、(完全なファイル経路を含む)
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
COMMENT/OPER,コピー用のファイルが存在します。コピーを開始します。
FILE/COPY,C1.INPUT,C2.INPUT,FAIL_IF_DEST_EXISTS
V2=FILE/EXISTS,C2.INPUT
IF/V2==0
COMMENT/OPER,"ここにファイルは存在しません: " + C2.INPUT
、エンディングにコピーします。
ROUTINE/END
END_IF/
ELSE/
COMMENT/OPER、ファイルのコピーが成功します。
ROUTINE/END
END_ELSE/
END_IF/
COMMENT/OPER,コピーするファイルは存在しません。
```

コードの説明

多くのこのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

C1=COMMENT/INPUT
このラインはコピーするファイルの完全な経路を取り、またそれをC1.INPUT
変数に置きます。
C2=COMMENT/INPUT
このラインは目的ファイルの完全な経路を取り、またそれを C2.INPUT 変数に置きます。
FILE/COPY,C1.INPUT,C2.INPUT,FAIL_IF_DEST_EXISTS
この行は目的ファイルに元のファイルをコピーします。このコマンドは、3個のパラメータを選択することが可能です。

- パラメータ1はC1.INPUT。これはコピーファイルへの絶対パスです。
- パラメータ2は C2.INPUTか、または、送付先ファイルへの絶対パスです。
- パラメータ3は、この場合、同じ送付先ファイル名を持つ既存のファイルに出会ったために、FILE/COPYの手続きを中止します。この設定により、同名の既存ファイルの上書きが行われます。

ファイルの移動

挿入 | ファイル I/O コマンド | ファイルを移動 メニュー
オプションは実行時にファイルの移動操作を行うコマンドを編集ウィンドウに挿入します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

ファイル/削除,<oldfilename>,<newfilename>

このコマンドのコンポーネントのいくつかについて説明します。

<旧ファイル名>
ファイルの場所と名前です。

<新ファイル名>
ファイルの新しい場所と名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Moveコマンドに置きます。
3. [F9]を押します。

File Moveコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードは、移動するファイル名、及び移動する場所のディレクトリとファイル名を要求します。その後、ファイルの移動操作を実行します。

```
C1=COMMENT/INPUT,移動したいファイルの名称をタイプ入力して下さい。
、(完全なファイル経路を含む)
C2=COMMENT/INPUT,コピーの送付先となるファイルの名称をタイプ入力して下さい。
、(完全なファイル経路を含む)
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
COMMENT/OPER,移動用のファイルが存在します。ファイルの移動を開始します。
FILE/MOVE,C1.INPUT,C2.INPUT
V2=FILE/EXISTS,C2.INPUT
IF/V2==0
COMMENT/OPER,"ここにファイルは存在しません: " + C2.INPUT
、移動機能は適切に作動しません。
ROUTINE/END
END_IF/
ELSE/
```

```
COMMENT/OPER、ファイルのコピーが成功します。
ROUTINE/END
END_ELSE/
END_IF/
COMMENT/OPER,オリジナル ファイルは存在しません。もう一度、試みてください。
```

コードの説明

このコードの多くは「ファイルのサンプル」で説明したのと似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FILE/MOVE,C1.INPUT,C2.INPUT
このラインは目的ファイルに元のファイルをコピーします。このコマンドは、最大2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1はC1.INPUT。これは、移動するファイルへの絶対パスです。
- パラメータ2は C2.INPUTか、または、送付先ファイルへの絶対パスです。

ファイル削除

挿入 | ファイル I/O コマンド | ファイルを削除メニュー
オプションは、コマンド実行時にファイルを削除するコマンドを編集ウィンドウに挿入します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

ファイル/削除,<ファイル名>

このコマンドのコンポーネントのいくつかについて説明します。

<filename (ファイル名)>
削除するファイルの名前です。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Deleteコマンド上に置きます。
3. [F9]を押します。

File Deleteコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはファイル名を要求して、またそれを削除します。

```
C1=COMMENT/INPUT,削除したいファイルの名称をタイプ入力して下さい。
、(完全なファイル経路を含む)
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
COMMENT/OPER,ファイルが存在します。削除の用意ができました。
FILE/DELETE,C1.INPUT
V2=FILE/EXISTS,
IF/V2==0
COMMENT/OPER、ファイルが正常に削除します
ROUTINE/END
END_IF/
ELSE/
COMMENT/OPER、ファイルが存在します。
ROUTINE/END
END_ELSE/
END_IF/
```

```
COMMENT/OPER,削除するファイルは存在しません。存在するファイルを選択して下さい
。
```

コードの説明

このコードの多くは「ファイル移動のコードサンプル」で説明したのと似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

FILE/DELETE,C1.INPUT
このラインでは、指定されたファイルが削除されます。このコマンドは、一種のパラメータ、削除するファイルのファイル名を受け取ります。この場合は、C1.INPUTです。

ファイルの存在チェック

挿入 | ファイル I/O コマンド |
ファイルが存在するメニューオプションは、実行時にファイルの存在を確認し、提供される変数に結果を設定するコマンドを編集ウィンドウに挿入します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname> = ファイル/存在,<ファイル名>
```

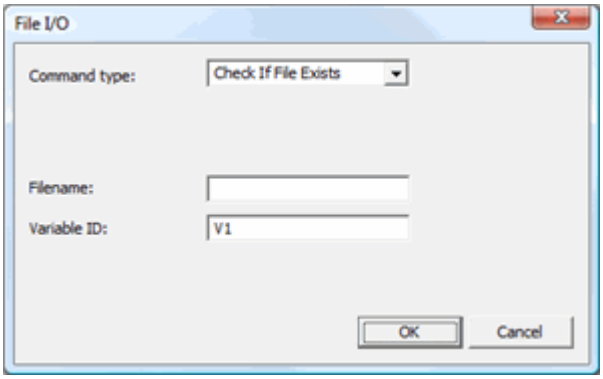
このコマンドのコンポーネントのいくつかについて説明します。

<ファイル名>
ディスクに存在するかどうか確認するためにチェックされるファイルの名前です。

<変数名>
行われる確認の結果に設定される変数の名前です。この変数はファイルが存在する場合は1に、ファイルが存在しない場合は0に設定されます。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Existsコマンド上に置きます。
3. [F9]を押します。



File Existsコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログ ボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはファイル名を要求して、またはファイルの存在をチェックします。

```
C1=COMMENT/INPUT,チェックしたいファイルの名称をタイプ入力して下さい。
V1=FILE/EXISTS,C1.INPUT
IF/V1<>0
    COMMENT/OPER,ファイルが存在します。
END_IF/
ELSE/
    COMMENT/OPER,ファイルは存在しません。
END_ELSE/
```

コードの説明

多くのこのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

V1=FILE/EXISTS,C1.INPUT

この行では、指定されたファイルの存在をチェックします。そのファイルは、PC-DMISがこのコードの機能のために作動するディレクトリに位置する必要があり、それ以外の場合には、このファイルを含むラインが、ファイルへの絶対パスをも含んでいなければなりません。V1 は当ファイル

チェックの結果を受け取ります。ファイルが存在する場合は、ゼロ以外の値で、さもないとは、0です。

ファイル ダイアログ ボックスの表示

挿入 | ファイル I/O コマンド | ファイル ダイアログ メニュー オプションは、実行時に**開く**ダイアログボックスを表示するコマンドを編集ウィンドウに挿入します。これによって、ユーザは実行時にファイル名を選択することができます。選択されたファイル名は指定された変数に保存されます。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

```
<varname> = File/Dialog,<expr>
```

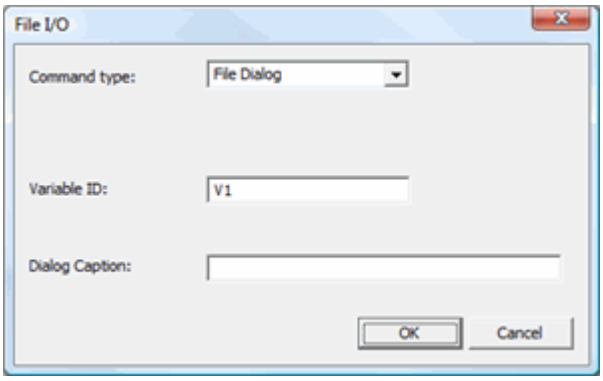
このコマンドのコンポーネントのいくつかについて説明します。

<varname (変数名)>
ファイルダイアログボックスでユーザーが選択したファイル名に設定される変数の名前です。

<expr (説明)>
ファイルダイアログボックスのタイトルバーに表示されるテキストです。

この File I / O コマンドに関連した、ダイアログ ボックスにアクセスするには:

1. 編集ウインドウを開きます。
2. カーソルをFile Dialogコマンド上に置きます。
3. [F9]を押します。



File Dialogコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはファイルの削除を選択するためのダイアログボックスを送信します。

```
V1=FILE/DIALOG,削除するファイルを選択して下さい。
V2=FILE/EXISTS,V1
IF/V2<>0
COMMENT/OPER,ファイルが存在します。削除の用意ができました。
FILE/DELETE,V1
V3=FILE/EXISTS,
IF/V3==0
COMMENT/OPER、ファイルが正常に削除します
ROUTINE/END
END_IF/
ELSE/
COMMENT/OPER、ファイルが存在します。
ROUTINE/END
END_ELSE/
END_IF/
COMMENT/OPER,削除するファイルは存在しません。存在するファイルを選択して下さい
。
```

多くのこのコードは「Read Characterコードのサンプル」または「Sample Read Lineコードのサンプル」の説明に似ています。

ここでは、この例に独自のものについてのみ、説明を行います。

V1=FILE/Dialog,削除するファイルを選択

この行は「削除するファイルを選択して下さい」というタイトルを持つダイアログボックスを表示します。ファイルをブラウザし、**開く**をクリックすると、PC-DMISが、選択されたファイルへの絶対パスをV1に指示します。残りのルーチンが、選択されたそのファイルを削除します。

ファイルの終端、または、ラインの終端をチェック

PC-DMISでは、条件テストにおいて、EOF、または、EOL機能を使用し、ファイルの終端をチェックできます。

EOF ファイルの末尾の略です。この関数は、文字列型のファイルポインターを用います。条件ステートメント内に適切に配置された場合、それは、そのファイルポインターが特定ファイルの終端に達するか否か、をテストします。達する場合、その関数は、真を返します。

EOL EOF ラインの末尾の略です。この関数は文字列型のファイルポインタを受け取ります。分岐条件に適切に配置される場合、それはファイルポインタは指定したラインの末尾に到達したかどうかをテストします。それがある場合、この関数は真を返します。これは、ループ内で最も良く機能します。

編集ウィンドウ内での、このコマンドのシンタックスは、以下のようです:

EOF(<filepointer>) またはEOL(<filepointer>)

このコマンドのコンポーネントのいくつかについて説明します。

<ファイルポインタ>
チェックするファイルポインタの名前です。

EOF及びEOLコードのサンプル

下記のサンプル コードは、**ファイル入力/出力** ダイアログボックス内ではなく、編集ウィンドウのコマンド モード内に入力されるべきです。

次のコードはtest.txtを開き、ファイルを介して読み取ります。ファイルの末尾に達していない限り(コードで指定される、 WHILE/!EOF) で、PC-DMIS文字でファイルの文字を読み込み、V1の文字を割り当てます。

PC-DMISが ファイル内で行の終端に達すると、PC-DMISは、その行の最後の活字を表示します。

PC-DMISがファイルの末尾に達するまでこれが繰り返されます。PC-DMISテキスト「ファイルの末尾に到達...」を示します。

```
FPTR=FILE/OPEN,D:\temp\test.txt,READ
WHILE/!EOF("FPTR")
V1=FILE/READ_CHARACTER,FPTR
IF/EOL("FPTR")
COMMENT/OPER,NO,"行の終端に達しました。最後の活字は:"
,V1
END_IF/
END_WHILE/
COMMENT/OPER、いいえ、「ファイルの末尾に到達した」
```