FUJITSU

F*TRAN2007 Ver.2.0

操作説明書／コマンド編

第１版

株式会社富士通ビー・エス・シー

は　じ　め　に

Ｆ＊ＴＲＡＮ２００７　Ｖ２．０をお買い上げいただき、誠にありがとうございます。

Ｆ＊ＴＲＡＮ２００７は、汎用機やオフコンなど、ホストコンピュータの標準フロッピーディスク形式であるＩＢＭ形式フロッピー（ＩＢＭファイル）と、パソコンの標準であるＷｉｎｄｏｗｓファイルを相互に変換するためのソフトウェアです。

おもな用途としては、汎用機やオフコンのＣＯＢＯＬデータと、パソコンのＣＳＶ形式ファイルとの交換・プリント形式ファイルとの交換・ＷｉｎｄｏｗｓＣＯＢＯＬデータとの交換などがあります。また、ＣＯＢＯＬのゾーン形式・パック形式をはじめ、豊富なコード変換機能をサポートしています。
日本で販売されたほとんどのホストの漢字をサポートしており、拡張漢字テーブルも標準提供しています。

Ｆ＊ＴＲＡＮ２００７　Ｖ２．０では、Ｗｉｎｄｏｗｓ７上での動作をサポートしています。

Ｆ＊ＴＲＡＮ２００７のマニュアルには、導入編、解説編、コマンド編（本書）、マルチレコード編、プログラム応用編があります。

２０１０年４月
株式会社 富士通ビー・エス・シー

目　　次

第１章　操作の基礎

第２章　コマンド型の実行

本書で用いる表記法

●本文と画面のパラメータ類の表記法

{A｜B｜C}	A、B、またはCのうち、どれか１つを選択します。省略はできません。
A B C	同上。
（A／B／C）	同上。
［A］	Aは省略できます。
［A／<u>B</u>／C］	A、B、またはCのうち、どれか１つを選択します。省略が可能で、その場合は下線を引いたBを選択したものとみなします。
A <u>B</u> C	同上。
（A／［B］／C）	同上。ただし、［　］でくくったBを選択したものとみなします。
X　・・・	X類を　A　B　C　のように列挙します。
n、ｎｎ、＜ｎ＞	１０進数を指定します。 （＜＞は表記上の記号で、入力はしません）
ｘｘH	１６進でｘｘです。Hを省くこともあります。
↓	改行を意味します。リターンキーのシンボルです。
<u>a</u>	下線部を入力します。
<u>ａｂｃ↓</u>	下線部を入力し、リターンキーを押します。
ＣＴＲＬ＋Ａ	コントロール（ＣＴＲＬ）キーを押しながら、Ａキーを押します。コントロールＡと読みます。
^Ａ	同上。
ｄ：	ドライブＡ：やＣ：など、任意のドライブ指定を表します。

◆注意　----　実画面と少し差異がある

本書に示す画面と実際の画面には、若干の差異がある場合があります。あらかじめ、ご了承ください。

第１章

操作の基礎

１．１　動作環境の設定

Ｆ＊ＴＲＡＮ２００７をコマンド型で実行する場合は、各ＯＳの

ＭＳ－ＤＯＳプロンプト　または　コマンドプロンプト

などを利用します。Ｆ＊ＴＲＡＮ２００７のディレクトリ以外からＦ＊ＴＲＡＮ２００７のコマンドを実行する場合には、つぎの２つの方法があります。

ＳＴＡＲＴコマンドで起動する
ＰＡＴＨコマンドを使い、コマンド検索パスを通す

通常は終了を待ち合わせる必要があるため、ＳＴＡＲＴコマンドを使います。

■ＳＴＡＲＴコマンド

●機能

Ｗｉｎｄｏｗｓアプリケーションやドキュメントを起動します。

●解説

ＳＴＡＲＴコマンドを使うと、Ｆ＊ＴＲＡＮ２００７をどのドライブ、どのディレクトリからでも起動できます。Ｆ＊ＴＲＡＮ２００７の終了を待ち合わせることもできます。

●指定

Ｆ＊ＴＲＡＮ２００７の終了を待たなくてよいときは、

Ｃ：￥＞ＳＴＡＲＴ　ＦＴ　↓

のようにして起動し、終了を待ち合わせるときは、

Ｃ：￥＞ＳＴＡＲＴ　／Ｗ　ＦＴ　↓

のようにして起動します。／Ｗ は ／ＷＡＩＴ の略です。

■ＰＡＴＨコマンド

●機能

コマンドの検索パスを指定します。

●解説

ＰＡＴＨコマンドで、Ｆ＊ＴＲＡＮ２００７のインストールディレクトリに、パス（コマンド検索パス）を通す方法もあります。Ｆ＊ＴＲＡＮ２００７を、どのドライブ、どのディレクトリからでも起動できるようになります。ただしこの方法では、Ｆ＊ＴＲＡＮ２００７の終了を待ち合わせることはできません。

●指定

つぎのように指定します。

C:¥>PATH %PATH%;C:¥＜インストールディレクトリ＞¥FTRAN2007↓

ＰＡＴＨコマンドの代わりに、ＳＥＴコマンドを使うこともできます。

C:¥>SET PATH=%PATH%;C:¥＜インストールディレクトリ＞¥FTRAN2007↓

●確認

Ｆ＊ＴＲＡＮ２００７をインストールしたディレクトリ以外から

Ｃ：￥＞ＦＴ　↓

のように入力して、起動できればＯＫです。

１．２　コマンドの紹介

Ｆ＊ＴＲＡＮ２００７のコマンド全体を紹介します。

●コマンドの構成

Ｆ＊ＴＲＡＮ２００７の各機能はＦＴ．ＥＸＥコマンドの内部コマンドとして実現され、構成されています。つぎにそのコマンド一覧を示します。

コマンド一覧

ＦＴ．ＥＸＥ　コマンド名		短縮名	機　　能
ファイル変換	ＧｅｔＴｅｘｔ	ｇｔ	ＩＢＭ→Ｗｉｎテキストファイル変換
	ＧｅｔＤａｔａ	ｇｄ	ＩＢＭ→Ｗｉｎデータファイル変換
	ＧｅｔＲａｎｄ	ｇｒ	ＩＢＭ→Ｗｉｎランダムファイル変換
	ＰｕｔＴｅｘｔ	ｐｔ	Ｗｉｎ→ＩＢＭテキストファイル変換
	ＰｕｔＤａｔａ	ｐｄ	Ｗｉｎ→ＩＢＭデータファイル変換
	ＰｕｔＲａｎｄ	ｐｒ	Ｗｉｎ→ＩＢＭランダムファイル変換
	ＶｉｒＤｒｉｖｅ	ｖｄ	仮想ドライブの設定
補助機能	ｉＬｉｓｔ	ｉｌ	ＩＢＭファイル名・属性一覧
	ｉＤｅｌｅｔｅ	ｉｄｅｌ	ＩＢＭファイルの削除
	ｉＦｏｒｍａｔ	ｉｆｏ	ＩＢＭディスクの初期化
	ｉＤｉｓｋＣｏｐｙ	ｉｄｃ	ＩＢＭディスクの複写
	ｉＲｅｎａｍｅ	ｉｒｅｎ	ＩＢＭディスク内ファイルの改名
	ｉＡｌｌｏｃ	ｉａｌ	ＩＢＭファイル領域のアロケート
	ｉＡｔｔｒ	ｉａｔ	ＩＢＭディスク内のファイル属性変更
	ｉＶｏｌＡｔｔｒ	ｉｖａ	ＩＢＭディスクのボリューム属性変更
	ＩｓＲｅａｄｙ	ｉｒ	フロッピーディスク装着検査
セットアップ	Ａｎｋ	ａｎ	ＡＮＫコードの設定
	Ｋａｎｊｉ	ｋａｎ	漢字変換方式の設定
	ＢｌｏｃｋＭａｐ	ｂｍ	ブロック配置モードの設定
	Ｃｏｍｍｅｎｔ	ｃｏｍ	コメントの変更
	ｃＬｏａｄ	ｃｌ	コード変換表の（再）読み込み
	ｃＳａｖｅ	ｃｓ	コード変換表の書き込み（保存）

◆注意　----　Ｆ＊ＴＲＡＮⅢでのみ有効なコマンド

Ｆ＊ＴＲＡＮⅢにある以下のコマンドは、Ｆ＊ＴＲＡＮ２００７では機能を持たないコマンドとして存在しています。したがって、実行してもエラーにはならず、特別な動作はしません。

メニュー関連のコマンド	Ｃｏｎｖｅｒｔ（ファイル変換メニュー） Ｓｅｔｕｐ（セットアップメニュー）
ＧＵＩ部でのみ機能を実現したコマンド	ｉＥｄｉｔ（ＩＢＭディスクエディタ） ＡｎｋＡｎｋ（ＡＮＫ変換表）
ＤＯＳ関連のコマンド	Ｓｈｅｌｌ（ＤＯＳコマンドの実行） ｄＰｒｏｍｐｔ（ＤＯＳプロンプトへ直行）
実行する意味を持たなくなったコマンド	ｃＴｅｍｐ（コード変換表の一時的修正扱い）

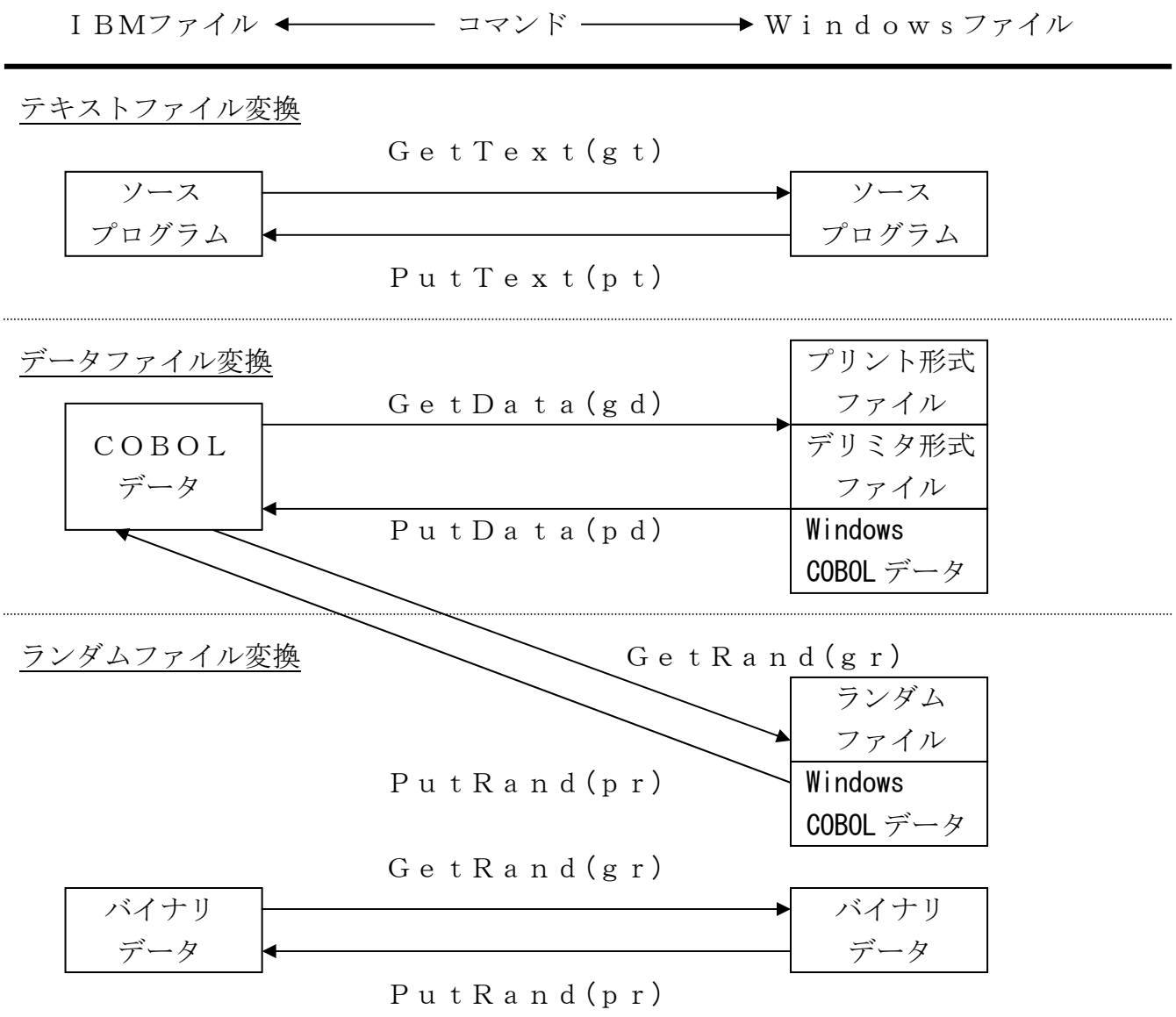
１．３　３つのファイル変換機能

Ｆ＊ＴＲＡＮ２００７では、６つのコマンドで、３つの（実質４つ）のファイル変換機能、

テキストファイル変換
データファイル変換１／プリント形式
データファイル変換２／デリミタ形式
ランダムファイル変換

を使い分けることができます。 **“うまく使い分けなければいけない”** といったほうが、正しいかもしれません。

●典型的なデータ形式と使用するコマンド



１．４　補助機能

Ｆ＊ＴＲＡＮ２００７は、豊富な補助機能を提供しています。おもにＩＢM形式のディスク・ファイルを管理するためのものです。

●ＩＢMファイル名・属性一覧［ｉＬｉｓｔ（ｉｌ）コマンド］
ＩＢMファイルについてのかなり詳しい属性を調べることができます。

●ＩＢMファイルの削除［ｉＤｅｌｅｔｅ（ｉｄｅｌ）コマンド］
ＩＢMファイルを削除できます。

この機能は、ラベルに「削除した」と印をつけるだけなので、うっかり削除したファイルをＩＢMディスクエディタ（ＧＵＩ部）を使って復活させることもできます。

●ＩＢMディスクの初期化［ｉＦｏｒｍａｔ（ｉｆｏ）コマンド］
パソコン上でＩＢM形式にディスクを初期化（フォーマット）することができます。

●ＩＢMディスクの複写［ｉＤｉｓｋＣｏｐｙ（ｉｄｃ）コマンド］
パソコン上でＩＢM形式のディスクコピー（複写）をすることができます。

●ＩＢMディスク内ファイルの改名［ｉＲｅｎａｍｅ（ｉｒｅｎ）コマンド］
パソコン上でＩＢM形式のディスク内ファイルの改名をすることができます。

●ＩＢMディスク内の領域のアロケート［ｉＡｌｌｏｃ（ｉａｌ）コマンド］
パソコン上でＩＢM形式のディスク内のファイル領域をアロケートすることができます。

●ＩＢMディスク内のファイル属性変更［ｉＡｔｔｒ（ｉａｔ）コマンド］
パソコン上でＩＢM形式のディスク内のファイル属性を変更することができます。

●ＩＢMディスクのボリューム属性変更［ｉＶｏｌＡｔｔｒ（ｉｖａ）コマンド］
パソコン上でＩＢM形式のディスクのボリューム属性を変更することができます。

●フロッピーディスクの装着検査［ＩｓＲｅａｄｙ（ｉｒ）コマンド］
フロッピーディスク装置にフロッピーディスクが装着されているかどうかを検査できます。

●仮想ドライブの設定［ＶｉｒＤｒｉｖｅ(ｖｄ)コマンド］

パス名指定（¥やディレクトリ名を使う指定）でＷｉｎｄｏｗｓファイルを指定できますが、この機能を使って実ドライブ･ディレクトリに適当な仮想ドライブ名をつけてアクセスすることもできます。

◆注意　----　ＶｉｒＤｒｉｖｅコマンドは、Ｆ＊ＴＲＡＮⅢで存在したため、Ｆ＊ＴＲＡＮⅢとの継承性を守るために用意しました。

１．５　セットアップ機能

Ｆ＊ＴＲＡＮ２００７は、このセットアップ機能でさまざまなホストへの対応を実現しています。コード変換の方法を設定する機能がその中心です。

●ＡＮＫコードの設定［Ａｎｋ（ａｎ）コマンド］

ホストで採用しているＡＮＫコード系がＥＢＣＤＩＣコードかＪＩＳ８／ＡＳＣＩＩコードかを設定します。

この設定はＡＮＫ変換のときだけでなく、バッファのクリア、ゾーン形式やパック形式の数値の変換、その他でも参照される、とても重要なものです。

ＥＢＣＤＩＣコードのカタカナ版／英小文字版の切り替えもできます。

●漢字変換方式の設定［Ｋａｎｊｉ（ｋａｎ）コマンド］

ホストで採用している漢字体系がどの方式かを設定します。ふつうは、Ｆ＊ＴＲＡＮ２００７が標準提供している漢字変換方式のなかから、どれか１つを選ぶだけですみます。

各漢字変換方式の設定内容はすべて修正可能です（ＧＵＩ部）。たとえば、漢字対応表に定義されない拡張漢字の扱い方の変更などができます。

●ブロック配置モードの設定［ＢｌｏｃｋＭａｐ（ｂｍ）コマンド］

ブロックの配置の方法には、一般の方式（トラック境界をまたぎ、空きセクタはできない）と、日立の方式（トラック境界をまたがず、トラック末尾に空きセクタができる）があります。どちらのモードで動作させるかを設定できます。

●コメントの変更［Ｃｏｍｍｅｎｔ（ｃｏｍ）コマンド］

設定を変更したら、適当なコメントをつけておくことができます。

＜コード変換表＞

以上の設定はファイルに保存できます。それを「コード変換表（ファイル）」と呼んでいます。拡張子は．ＣＣＴ（Code Conversion Table）で、Ｆ＊ＴＲＡＮ２００７の動作を決める重要なファイルです。複数のホストとのデータ交換が必要なときは、ホストの種類に応じてこれを何とおりかにセットアップしておき、使い分けることができます。

●コード変換表の（再）読み込み［ｃＬｏａｄ（ｃｌ）コマンド］

起動時にどのコード変換表を使うかを指定できます。しかし、起動後でも別のコード変換表を読み込んで切り替えることができます。元のコード変換表を読み込み直して、セットアップ前のコード変換表の状態に戻すこともできます。

●コード変換表の書き込み（保存）［ｃＳａｖｅ（ｃｓ）コマンド］

あるコード変換表で起動し、設定を変更して別の名前で書き戻す、といったこともできます。

＜環境設定＞

コード変換表以外にＦ＊ＴＲＡＮ２００７の動作に必要なファイルとして、漢字対応表ファイル（＊．ＫＫＴ）、ＦＴＲＡＮ．ＩＮＩファイルがあります。これらのファイルは初期状態として、標準インストールモードではユーザデータフォルダ配下、２００６互換インストールモードではインストールディレクトリ配下にあるファイルが使用されますが、別のフォルダにあるファイルを使用する事も可能です。この設定は環境設定で行います。ＧＵＩ部から設定可能ですが、起動オプションとしても指定できます。ＧＵＩからの設定は「操作説明書／解説編」を参照してください。起動オプションとしての指定方法は「２．１　ＦＴ．ＥＸＥ 起動と終了」を参照してください。

つぎの６つは、ＧＵＩ部でのみ設定できる機能です。

●漢字対応表の設定

ホストの拡張漢字や外字（ユーザー定義文字）などを、利用者の意図した漢字に割り当てるための漢字対応表を設定します。漢字対応表に登録された内容が変換時の漢字コードに反映されます。ふつうは、標準提供の漢字対応表を利用します。

●Ｗｉｎｄｏｗｓ　ＣＯＢＯＬベンダの設定

Ｗｉｎｄｏｗｓ上で使用しているＣＯＢＯＬがどこのベンダかを設定します。この設定は、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬで使用するデータを変換する場合に重要になります。

●ホストエンディアンの設定

ホストの２進数値項目の格納順序（正順、逆順）を設定します。この設定は、ホストのＣＯＢＯＬの２進数値項目データ等を変換する場合に重要になります。

●ＩＢＭディスクの形式の設定

Ｆ＊ＴＲＡＮ２００７で使用するＩＢＭディスクの形式を設定します。ふつうは、一般のＩＢＭディスク（１．２ＭＢタイプ）か、三菱のＩＢＭディスク（１．４４ＭＢタイプ）のどちらかを選択します。両方とも使用する場合は、優先度の高い順にアクセスするモードに設定することもできます。

●Ｃｏｐｙ句読み込みの設定

ＣＯＢＯＬのＣｏｐｙ句（登録集）を読み込んで、変換のためのパラメータを自動展開するときに、そのファイルの形式（正書法など）や展開のしかた（見だしの生成など）を設定します。

●ホスト種別の設定

ホスト機の中では、IBM社のオフコン（eServer iSeries、AS/400）において、ゾーン形式・パック形式の符号表現が他と若干異なっています。一般的な設定か IBM iSeries、AS/400 かを設定します。

１．６　マルチコマンド

Ｆ＊ＴＲＡＮ２００７をコマンド行方式で使うときは、コマンドをセミコロン（;）で区切りながら、いくつも列挙して指定できます。ＢＡＳＩＣ言語のマルチステートメントのような機能です。パラメータファイルのなかでもこの機能が使えます。

例）Ｃ：￥ＰＬＡＮＥＴの下にある２つのファイルを一度に変換する

ドライブＣ：のＰＬＡＮＥＴディレクトリの中にあるＷｉｎｄｏｗｓファイルＥＡＲＴＨ．ＤＡＴ、ＪＵＰＩＴＥＲ．ＤＡＴを、ドライブＡ：のＩＢＭファイルＥＡＲＴＨ、ＪＵＰＩＴＥＲにデータファイル変換します。

```
C:¥>ft pd c:¥planet¥earth.dat a: ; pd c:¥planet¥jupiter.dat a: ↓
```

仮想ドライブ機能を使えば、下記の記述でも同じになります。

```
C:¥>ft vd x: c:¥planet ; pd x:earth.dat a: ; pd x:jupiter.dat a: ↓
                ↑
                └── X と指定したら C:¥PLANET をアクセスせよ
```

１．７　コメントの付加

Ｆ＊ＴＲＡＮ２００７をコマンド行方式で使うときに、コマンドやパラメータのあとに２つ重ねのハイフン（－－）を書くと、それ以降の行末までをコメントとみなします。ふつうは、バッチファイルのなかでコメントをつけるのに使います。

この機能は、パラメータファイルのなかにコメントを書ける機能と、ほとんど同じものです。

例）バッチファイルに「Ａ：、Ｂ：の内容一覧」というコメントをつける

ドライブＡ：とＢ：のＩＢＭファイル一覧を表示するバッチファイルにコメントをつけます。

＜ＩＬ＿Ａ＿Ｂ．ＢＡＴ＞

```
start /w ft ilist a: ; ilist b:          -- Ａ：、Ｂ：の内容一覧 ↓
```

１．８　Ｗｉｎｄｏｗｓファイルの指定

●指定形式

Ｗｉｎｄｏｗｓファイルはつぎの形式で指定します。

［ｄ：］［パス名指定］［基本ファイル名［．拡張子］］

ｄ：はドライブ名

●ドライブ名（ｄ：）

ドライブ名（ｄ：）には、

Ａ：　～　Ｚ：　　実ドライブ名を指定します。

また、Ｆ＊ＴＲＡＮ２００７内でのみ通用するドライブ名として、つぎの指定ができます。

＠：　　カレントドライブのカレントディレクトリを表す
？：　　インストールディレクトリを表す

ドライブ名は省略可能です。省略するとカレントドライブを指定したものとみなされます。
ただし、

起動時の／Ｃｏｄｅオプション
パラメータファイルの参照（＋＋のあと）
ｃＬｏａｄ（ｃｌ）コマンド
ｃＳａｖｅ（ｃｓ）コマンド

は例外です。これらでドライブ名を省略すると、２００６互換インストールモードではインストールディレクトリ（？：）を指定したものとみなされます。標準インストールモードでは（１）は『¥＜ユーザデータフォルダ＞¥Conf¥』配下を参照します。（２）～（４）は環境フォルダを参照します。

●パス名指定（￥ディレクトリ名￥サブディレクトリ名￥・・・）

パス名指定（¥やディレクトリ名を使う指定）ができます。指定したディレクトリ配下のファイルを扱うことができます。

Ｆ＊ＴＲＡＮⅢではパス名を含むファイル指定ができないために、ＶｉｒＤｒｉｖｅ（ｖｄ）コマンドで指定していましたが、Ｆ＊ＴＲＡＮ２００７ではパス名を含むファイル指定ができるようになっています。また、従来のＶｉｒＤｒｉｖｅ（ｖｄ）コマンドでの指定もできます。

●基本ファイル名と拡張子

大部分のコマンドでは、入力側にＷｉｎｄｏｗｓファイルを指定する場合、基本ファイル名と拡張子にワイルドカード文字（＊、？）が使用できます。

出力側にＷｉｎｄｏｗｓファイルを指定する場合、基本ファイル名にはふつうの基本ファイル名以外に、＊も指定できます。＊は「入力側のファイル名を引き継げ」という意味です。ふつう、入力側のＩＢＭファイル名を引き継ぐことになります。

ドライブ名だけ指定して、基本ファイル名と拡張子を省略すると、入力側のファイル名が引き継がれ、拡張子なしになります。ふつうは、

ｄ：＊．ｄａｔ　　のように、拡張子も指定します。

こうすると、指定ドライブに

元のファイル名．拡張子　　というファイルができます。

なお、できる限り適切な拡張子をつけるように心がけてください。

●拡張子の省略値

Ｗｉｎｄｏｗｓファイルの指定ができるところでも、拡張子の省略値が、

．ＣＣＴ　　コード変換表ファイルの、省略時の拡張子
．Ｐ　　パラメータファイルの、省略時の拡張子

のように決まっているものもあります。

●ロングファイル名が使える

Ｗｉｎｄｏｗｓの３２ビットアプリケーションであるＦ＊ＴＲＡＮ２００７ではロングファイル名の指定ができるようになっています。ファイル名に空白（スペース）が使われている場合は“Ｔｅｓｔ　Ｄａｔａ”のように、ファイル名全体を“”で囲みます。

●デバイスファイルについて

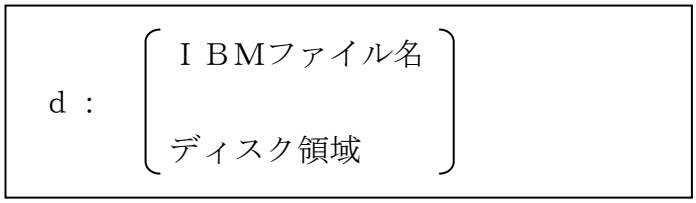
デバイスファイル名はキャラクタ型の周辺装置につけられている名前で、Ｗｉｎｄｏｗｓファイルとして指定できます。指定できるデバイスファイルは、つぎのとおりです。

ＮＵＬ　　ダミーデバイス。入力の場合はただちにＥＯＦになる。
出力の場合は出力データが捨てられる。

１．９　ＩＢＭファイルの指定

●指定形式

ＩＢＭファイルはつぎの形式で指定します。



ｄ：はドライブ名

●Ｗｉｎｄｏｗｓファイル指定との違い

ＩＢＭファイルの指定方法はＷｉｎｄｏｗｓファイルの指定方法とよく似ていますが、

ドライブ名はＡ：～Ｐ：または０：～３：で指定し、省略できない
（０：～３：は１Ｍバイト系フロッピーディスクドライブの物理機番）
ＩＢＭファイル名は基本形式で８文字以内、拡張形式で１７文字以内
（ただし、ふつうは拡張形式でも８文字以内のファイル名にする）
ＩＢＭファイル名は英数字で構成し、先頭は英字。特殊文字は不可
（ただし、＠、＃、￥、＄の使用を許し、英数字扱いすることが多い）
拡張子はない
サブディレクトリの概念はない

などの点が違います。

●ワイルトカード（＊）が使える

大部分のコマンドでは、入力側にＩＢＭファイルを指定するときに、ワイルドカード文字（＊）が使用できます。

多くのコマンドでは出力側にＩＢＭファイルを指定するとき、ＩＢＭファイル名を省略できます。その場合のＩＢＭファイル名の省略値は＊です。

Ｗｉｎ→ＩＢＭファイル変換のときは＊を２個まで指定できます。１個目の＊はＷｉｎｄｏｗｓファイルの基本ファイル名で置き替えられ、２個目の＊はＷｉｎｄｏｗｓファイルの拡張子で置き替えられます。

◆注意　----　「？」は使えない

Ｗｉｎｄｏｗｓファイルの指定ではワイルドカードとして「？」が使えますが、ＩＢＭファイルの指定では使えないのでご注意ください。

●ディスク領域名

ＩＢM形式のディスクはいくつかの領域に区分されています。Ｆ＊ＴＲＡＮ２００７では特別な名前でそれぞれの領域をアクセスすることを許しています。本書ではそれをディスク領域と呼び、アクセスするための名前をディスク領域名と呼びます。

ディスク領域名を使うと、対応するディスク領域はあたかもふつうのＩＢMファイルであるかのように扱われます。

ディスク領域

ディスク領域名	領　域　内　容	レコード長	ブロック長	形式 *1
．ＩＮＤＥＸ	インデックスシリンダの表面 エラーマップラベル ボリュームラベル ファイルラベル	１２８	１２８	拡張 ＦＢ
．ＩＮＤＥＸ２ *2	インデックスシリンダの裏面 ファイルラベル	１２８	128(2S) 256(2HD)	拡張 ＦＢ
．ＤＡＴＡ	データシリンダ ファイルのデータ部	セクタ長	セクタ長	拡張 ＦＢ

*1）拡張：拡張形式、ＦＢ：固定長レコード・ブロック化・非スパン

*2）１Ｓのディスクには存在しない

１．10 オプションとキーワードの指定

Ｆ＊ＴＲＡＮ２００７は､覚えやすいフルスペルでの指定と入力しやすいその短縮形の両方を受けつけます。各所で使うキーワードも同様です。

オプションは､／Ｃｏｄｅのように必ずスラッシュではじまるキーワードです。多くのオプションではパラメータとして１つないし複数のキーワードや数値などを必要とします。本書ではそれをオプションデータと呼ぶことにします。

キーワードは､／ＣｏｄｅとかＫｅｙＷｏｒｄのように単語の一部が大文字で表記されています。これは、

／や大文字の部分は省略できないが、小文字の部分は途中で打ち切ってもよい

ことを表しています。この例ではそれぞれつぎのように短縮できます。

表　記	入　力　例				
／Ｃｏｄｅ	／ＣＯＤＥ	／ＣＯ	／Ｃ		など
ＫｅｙＷｏｒｄ	ＫＥＹＷＯＲＤ	ＫＷＯＲＤ	ＫＥＷ	ＫＷ	など

◆注意　----　空白について

オプションデータつきのオプションの場合､**オプションとオプションデータの間には空白を入れても入れなくてもどちらでもかまいません**。ただし､空白を入れないと指定があいまいになることがあります。そのときは空白でオプションとオプションデータを区切ってください。

たとえば、ＩＢＭファイルのコード系を指定する、つぎのようなオプションがあります。

／Ｃｏｄｅ	Ａｎｋ	
	Ｋａｎｊｉｍｉｘ	

これは、それぞれつぎのように短縮指定できます。

フルスペル		短縮形（例）		最短縮形
／Ｃｏｄｅ　ＡＮＫ	→	／ＣＯＤＡＮ	→	／ＣＡ
／Ｃｏｄｅ　ＫＡＮＪＩＭＩＸ	→	／ＣＯＫＡ	→	／ＣＫ

１．11　式の指定

Ｆ＊ＴＲＡＮ２００７では､オプションデータなどのパラメータを１０進数で指定できるところで、ほとんどの場合、１０進数の代わりに値を「式」で指定することができます。計算を省いたり、レコード長のようなその都度変わるデータに対しても同じ指定ですむようにしたり、相対的な指定を可能にしたりするためです。

●式とは

式といっても、複雑な数式のようなものを指定できるわけではありません。１０進数を、四則演算子とカッコで組み合わせられるだけの単純なものです｡一部のオプションではさらに特殊変数を使うこともできます。

●四則演算子

四則演算子はつぎの５つです。

＋	加算
－	減算
＊　または　ｘ（小文字のｘ）	乗算
／／　または　％	除算
￥￥	剰余（割算の余り）

演算子間の優先順位はありません。＊を＋や－よりも先に計算したりしないということです。カッコを使って計算の順番を明示してください。カッコは何重にも入れ子にできます。

●特殊変数

特殊変数にはつぎの８つがあります。

$	最大値を表す	（ふつうはレコード長を意味する）
.（ﾋﾟﾘｵﾄﾞ）	現在値を表す	（ふつうはレコードの現在の桁位置を意味する）
＊	残りを表す	（ふつうはレコードの残りの長さを意味する）
~SysPhase	フェーズ	（ふつうはマルチレコードの指定時に使用する）
~SysRecNum	レコード番号	（ふつうはマルチレコードの指定時に使用する）
~SysReturn	リターン値	（ふつうはマルチレコードの指定時に使用する）
~SysBreak	ブレーク値	（ふつうはマルチレコードの指定時に使用する）
~SysQuit	クイット値	（ふつうはマルチレコードの指定時に使用する）

式の例を示します。つぎに示すのはいずれも正しい式です。

７	．	＄
（１５）	．－１５	＄ｘ４
４＋（６＋８）	．＋２	＄＋２
２５６＊２６	＊	＄％３
１０２４＊４	＊－２	＄ｘ（（＄％８０）＋１）
＊＊３	＊／／３	＊￥￥３
~SysRecNum	~SysRecNum¥¥3	

◆注意　----　式に空白を入れてはならない

演算子やカッコなどの前後に空白を入れてはいけません。空白はパラメータ類の区切りを意味するからです。

１．12 ピクチャの指定

●ピクチャとは

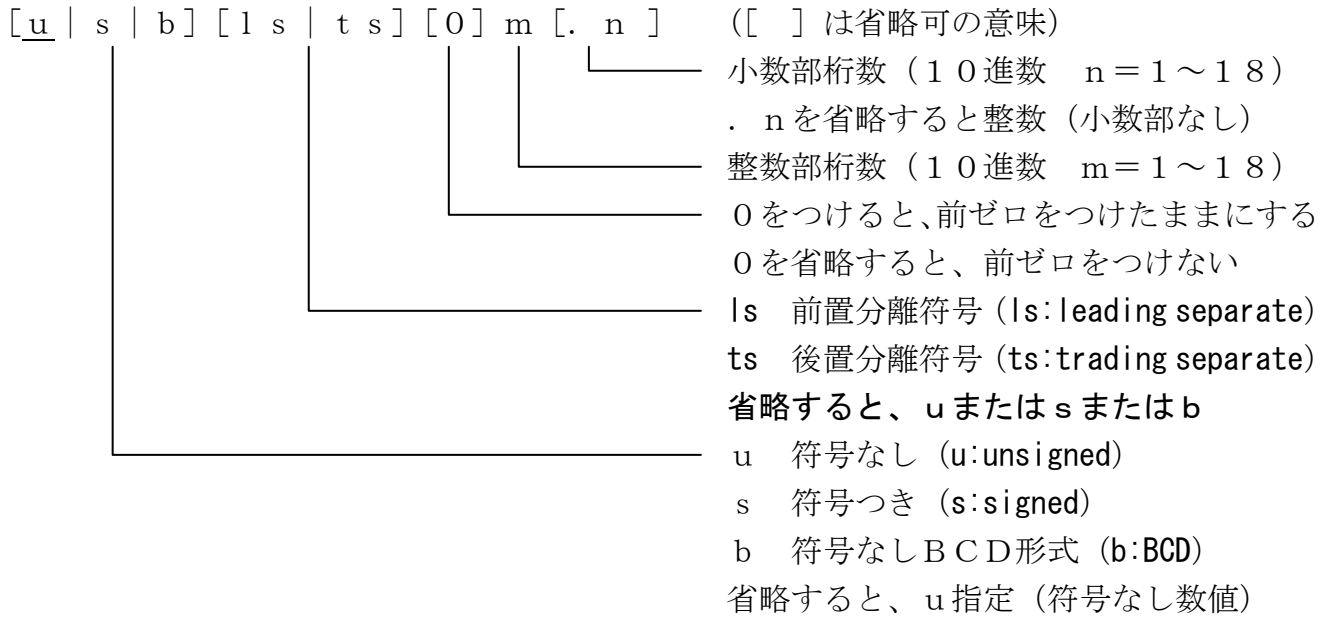
ピクチャとは、ＣＯＢＯＬのゾーン／パック形式や、ＢＣＤ形式の数値項目を変換するとき、

符号の有無
数値の桁数
前ゼロの有無
小数部の有無と桁数

を指定するためのものです。これらは、データ自体には記録されていないので、外からこれらの情報を与える必要があります。

●ピクチャの指定形式

ピクチャはつぎの形式で指定します。



●ＣＯＢＯＬをまねた

Ｆ＊ＴＲＡＮ２００７のピクチャは、ＣＯＢＯＬのピクチャ指定をまねた上で、大幅に簡略化したものです。たとえば、　　－１２．３

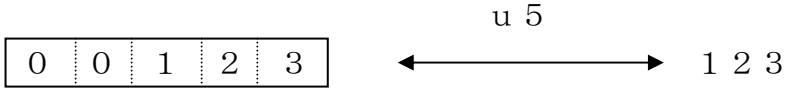
という数字があって、５バイトのゾーン形式の項目に記録してあるとします。
ＣＯＢＯＬのピクチャなら

ＰＩＣ　　Ｓ９（４）Ｖ９（１）

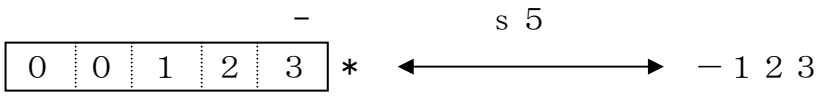
のように指定します。
Ｆ＊ＴＲＡＮ２００７のピクチャではこれを、　　Ｓ４．１　　と指定します。

ピクチャの指定例を示します。以下の図の左側がゾーン形式の項目、右側が文字形式数値、そして矢印の上がピクチャです。

例１）符号なし整数

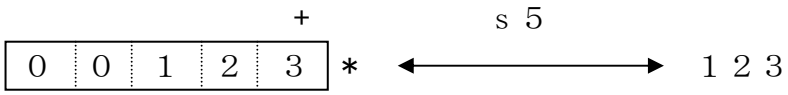


例２）符号つきの、負の整数



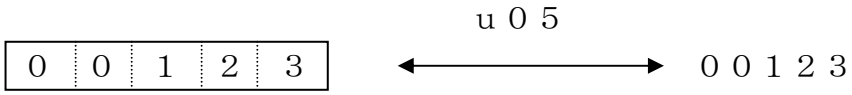
*）数字の上の“-”は、負の符号を表す

例３）符号つきの、正の整数

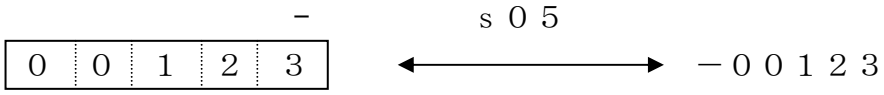


*）数字の上の“+”は、正の符号を表す

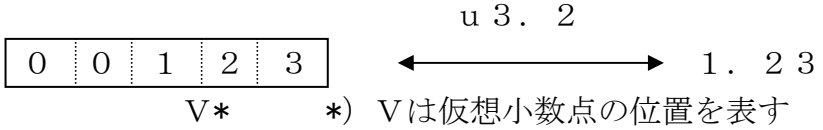
例４）符号なし整数、前ゼロをつけたままにする



例５）符号つき整数、前ゼロをつけたままにする

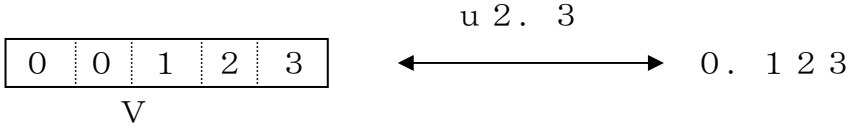


例６）小数部（小数点）がある／その１

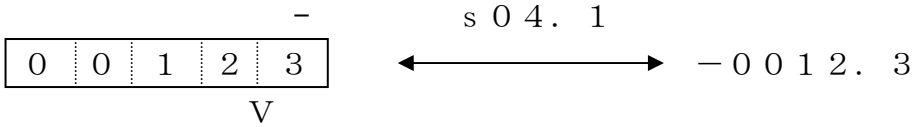


*）Ｖは仮想小数点の位置を表す

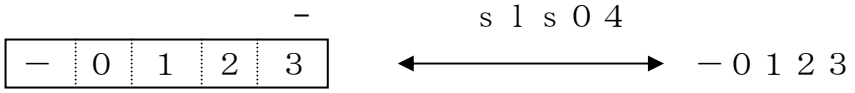
例７）小数部（小数点）がある／その２



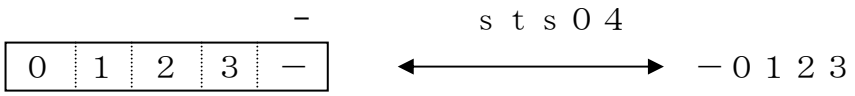
例８）符号も小数部（小数点）もあり、前ゼロもつけたままにする



例９）分離符号(前置)つき整数、前ゼロをつけたままにする



例 10）分離符号(後置)つき整数、前ゼロをつけたままにする



１．13　２進ピクチャの指定

●２進ピクチャとは

２進ピクチャとは、ＣＯＢＯＬ、Ｖｉｓｕａｌ　Ｂａｓｉｃ、Ｃ／Ｃ＋＋などの２進数値項目を変換するとき、

バイト数
格納順
符号の有無
数値の桁数
前ゼロの有無
小数部の有無と桁数

を指定するためのものです。これらは、データ自体には記録されていないので、外からこれらの情報を与える必要があります。

●２進ピクチャの指定形式

２進ピクチャはつぎのどちらかの形式で指定します。

形式１：ｉ＜ｗ＞［ｎ｜ｘ］［ｕ｜ｓ］　（［　］は省略可の意味）

- ｕ　符号なし（u:unsigned）
- ｓ　符号つき（s:signed）
- 省略すると、ｕ指定（符号なし数値）
- ｎ　正順の格納（ビッグエンディアン：汎用機系）
- ｘ　逆順の格納（リトルエンディアン：Intel 系）
- ｎ、ｘともに省略すると、
  - ＩＢＭ側の場合、変換設定のホストエンディアン
  - Ｗｉｎｄｏｗｓ側の場合、逆順のエンディアン
- ＜ｗ＞　バイト数、１～８
- ｉ　整数・小数であることを示す

形式２：ｉ＜ｗ＞［ｎ｜ｘ］［｛ｕ｜ｓ｝［０］＜ｍ＞［．＜ｎ＞］］　（［　］は省略可の意味）

２進キャスト　　ピクチャ

- ０　前ゼロ付加
- ＜ｍ＞　整数部桁数、１～１８
- ＜ｎ＞　小数部桁数、１～１８、省略すると０
- ＜ｍ＞＋＜ｎ＞が１～１８になること
- ＜ｗ＞に応じて、＜ｍ＞＋＜ｎ＞の省略値が定まる

●各言語の型と２進ピクチャの対応

言語	型	２進ピクチャ
ＣＯＢＯＬ	ＢＩＮＡＲＹ	ｉ１ｎｕ～ｉ８ｎｕ
（例）	ＳつきＢＩＮＡＲＹ	ｉ１ｎｓ～ｉ８ｎｓ
	ＣＯＭＰ	ｉ１ｎｕ～ｉ８ｎｕ
	ＳつきＣＯＭＰ	ｉ１ｎｓ～ｉ８ｎｓ
	ＣＯＭＰ－４	ｉ１ｎｕ～ｉ８ｎｕ
	ＳつきＣＯＭＰ－４	ｉ１ｎｓ～ｉ８ｎｓ
	ＣＯＭＰ－５	ｉ１ｘｕ～ｉ８ｘｕ
	ＳつきＣＯＭＰ－５	ｉ１ｘｓ～ｉ８ｘｓ
Ｖｉｓｕａｌ	Ｂｙｔｅ	ｉ１ｕ
Ｂａｓｉｃ	Ｉｎｔｅｇｅｒ	ｉ２ｓ
	Ｌｏｎｇ	ｉ４ｓ
	Ｃｕｒｒｅｎｃｙ	ｉ８ｓ１４．４
Ｃ／Ｃ＋＋	ｓｉｇｎｅｄ　ｃｈａｒ	ｉ１ｓ
	ｕｎｓｉｇｎｅｄ　ｃｈａｒ	ｉ１ｕ
	ｓｉｇｎｅｄ　ｓｈｏｒｔ	ｉ２ｓ
	ｕｎｓｉｇｎｅｄ　ｓｈｏｒｔ	ｉ２ｕ
	ｓｉｇｎｅｄ　ｉｎｔ	ｉ４ｓ
	ｕｎｓｉｇｎｅｄ　ｉｎｔ	ｉ４ｕ
	ｓｉｇｎｅｄ　ｌｏｎｇ	ｉ４ｓ
	ｕｎｓｉｇｎｅｄ　ｌｏｎｇ	ｉ４ｕ
	ｓｉｇｎｅｄ　＿＿ｉｎｔ６４	ｉ８ｓ
	ｕｎｓｉｇｎｅｄ　＿＿ｉｎｔ６４	ｉ８ｕ

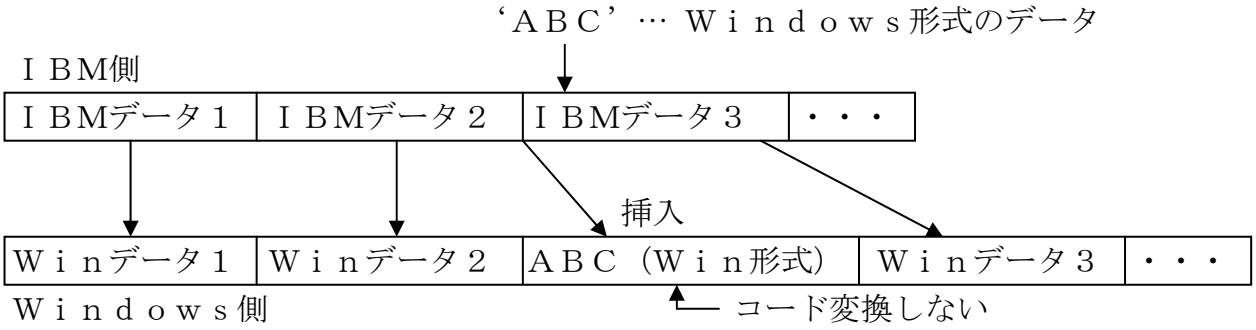
１．14 定数の指定

データファイル変換とランダムファイル変換では、項目別に分けて変換できますが、その際に定数や変数を挿入することができます。

テキストファイル変換の場合は、これに相当する機能はありません。

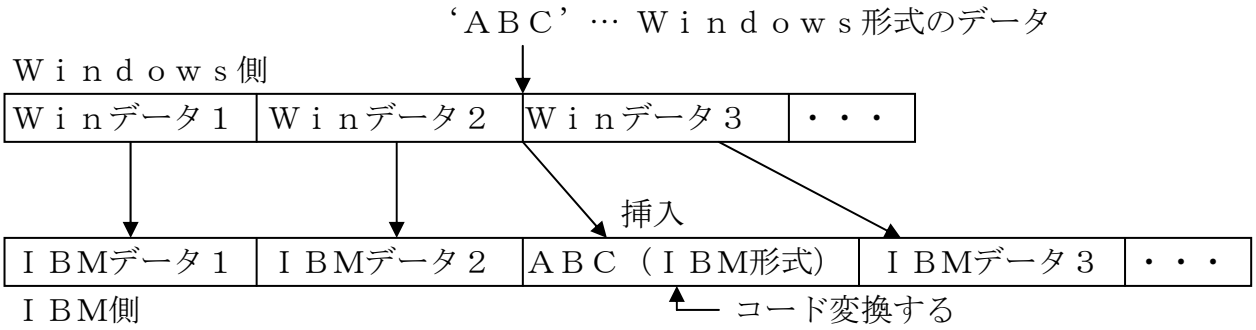
●ＩＢＭ→Ｗｉｎ変換時

指定した定数は、もともとＷｉｎｄｏｗｓ形式のデータなので、コード変換せずにそのままＷｉｎｄｏｗｓ側に挿入・出力されます。



●Ｗｉｎ→ＩＢＭ変換時

指定した定数は、ＩＢＭ形式のデータに変換してから、ＩＢＭ側に挿入・出力されます。



●列定数の種類

文字列定数

‘文字列’	文字列は最大２５６文字で、半角(’)でくくる。 ※ただし(’), (*), (?)は(¥’), (¥*), (¥?)で表し、(¥)自身は(¥¥)で表す。 （例）’ANK’、’漢字’、’ANK&漢字’、”

１６進列定数

{Ｘ｜ｘ}’ｈｈ…’	ｈは０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト 任意のバイト境界に半角スペースを挿入できる。 最大２５６バイト、半角(’)でくくる。 （例）X’1234’、x’00 FF a1a1 ff’、x”

汎用定数

Ｓｐａｃｅ	空白文字列（半角、全角の空白からなる文字列）
ＬｏｗＶａｌｕｅ	００Ｈ…の文字列、または００Ｈ…のバイナリデータ
ＨｉｇｈＶａｌｕｅ	ＦＦＨ…の文字列、またはＦＦＨ…のバイナリデータ
	※ＡＮＫ変換表をＦＦ→ＦＦにする必要がある。

●数値定数の種類

整数定数

［－］ｎ	(-)はマイナス符号、ｎは最大１８桁 （例）0、123、-99

小数定数

［－］ｎ．ｍ	(-)はマイナス符号、ｎとｍは合わせて最大１８桁 （例）0.0、1.23、-99.9

１６進定数

０{Ｘ｜ｘ}ｈ…	ｈは０～９、Ａ～Ｆ（ａ～ｆ）で最大１６桁（８バイト） 途中の空白は不可 （例）0x00、0XFF、0xee、0x1、0xfff

汎用定数

Ｚｅｒｏ	０または０．０
Ｍｉｎ	最小値
Ｍａｘ	最大値

●指定できる定数

文字列変換

変換方法	文字列定数	１６進列定数	Ｓｐａｃｅ ＬｏｗＶａｌｕｅ ＨｉｇｈＶａｌｕｅ
Ａｎｋ変換	○	×	○
漢字変換	○	×	○
Ａｎｋ・漢字混在変換	○	×	○
バイナリ変換	×	○	△ *1
バイナリ反転変換	×	○	×

*1）Ｓｐａｃｅのみ不可

数値変換

変換方法	数値定数	システム変数	Ｚｅｒｏ Ｍｉｎ Ｍａｘ
文字形式へ変換	○	○	×
ゾーン形式へ変換	○	○	○
パック形式へ変換	○	○	○
２進形式へ変換	○	○	○

１．15　日付データの指定

●日付マスク

Ｆ＊ＴＲＡＮ２００７で使用できる日付データの編集指定はつぎのとおりです。

日 付 マ ス ク	データ例
ｙｙｙｙ－ｍｍ－ｄｄ	1998/12/31
	1998-12-31
*1	1998.12.31
ｙｙ－ｍｍ－ｄｄ	98/12/31
	98-12-31
	98.12.31
ｎｙｙ－ｍｍ－ｄｄ	H10/12/31
	H10-12-31
	H10.12.31
ｙｙｙｙｍｍｄｄ	19981231
ｙｙｍｍｄｄ	981231
ｍｍ－ｄｄ－ｙｙｙｙ	12/31/1998
	12-31-1998
*1	12.31.1998
ｍｍ－ｄｄ－ｙｙ	12/31/98
	12-31-98
	12.31.98
ｍｍｄｄｙｙｙｙ	12311998
ｍｍｄｄｙｙ	123198
ｄｄ－ｍｍ－ｙｙｙｙ	31/12/1998
	31-12-1998
*1	31.12.1998
ｄｄ－ｍｍ－ｙｙ	31/12/98
	31-12-98
	31.12.98
ｄｄｍｍｙｙｙｙ	31121998
ｄｄｍｍｙｙ	311298

日付マスク	データ例
ｙｙｙｙ－ｍｍ	1998/12
	1998-12
*1	1998.12
ｙｙ－ｍｍ	98/12
	98-12
	98.12
ｎｙｙ－ｍｍ	H10/12
	H10-12
	H10.12
ｙｙｙｙｍｍ	199812
ｙｙｍｍ	9812
ｍｍ－ｙｙｙｙ	12/1998
	12-1998
*1	12.1998
ｍｍ－ｙｙ	12/98
	12-98
	12.98
ｍｍｙｙｙｙ	121998
ｍｍｙｙ	1298
ｙｙｙｙ　*1	1998
ｙｙ	98
ｎｙｙ	H10
ｇｙｙｍｍｄｄ	4101231
ｇｙｙｍｍ	41012
ｇｙｙ	410

ｎ＝年号
Ｍ（明治）1868-1911
Ｔ（大正）1912-1925
Ｓ（昭和）1926-1988
Ｈ（平成）1989-

ｇ＝元号
１（明治）1868-1911
２（大正）1912-1925
３（昭和）1926-1988
４（平成）1989-

◆注意　----　出力時、和暦の年号／元号の最終年は、次年号／元号の元年(01)になる

実際には、日付マスク分の長さが編集対象になります。たとえば“ｙｙｙｙｍｍｄｄ”と指定すれば、８バイトのデータの編集を行います。

入力側に“ｙｙ－ｍｍ－ｄｄ”のような日付区切りのある指定をした場合は、８バイトの内容が“９８　１２，３１”であっても、“９８－１２－３１”と同等のデータとして扱います。つまり、数字（０～９）以外の文字を日付区切り記号とみなします。

●ウインドウ方式とシフト方式

日付データの年の２桁（ｙｙ）と４桁（ｙｙｙｙ）の変換を行う場合、Ｆ＊ＴＲＡＮ２００７ではウインドウ方式とシフト方式をサポートしています。

ウインドウ方式とは、１９ｘｘ年（基準年）から１００年として扱う方式です。ウインドウ方式で“３０”と指定すれば、実際のデータはつぎのようになります。

	１９３０年		２０００年		２０３０年
データ	３０	９９	００	２９	

シフト方式とは、西暦からｎｎ引いた値の下２桁のデータを扱う方式です。一般には、ｎｎ＝２５（昭和通年方式）、ｎｎ＝８８（平成通年方式）、ｎｎ＝２８（暦一巡方式）などがあります。シフト方式で“３０”と指定すれば、実際のデータはつぎのようになります。

	１９３０年		２０００年		２０３０年
データ	００	６９	７０	９９	

入力側に *1 の日付マスク指定（日付区切りがある４桁の年指定）をし、実際の日付データの年が２桁以下であった場合は、無条件にウインドウ方式による拡張を行います。

●日付区切り記号

日付データを出力する際に指定できる日付区切り記号はつぎのとおりです。

日付区切り記号	デ　ー　タ　例
／（スラッシュ）	１９９８／１２／３１
－（ハイフン）	１９９８－１２－３１
.（ピリオド）	１９９８．１２．３１

●ＭＡＰオプション指定

日付データを変換する手順はつぎのとおりです。

①年設定（日付データ２桁の年の扱い、ウインドウ方式またはシフト方式の指定）
②日付区切り設定（日付データ出力時の日付区切り記号の指定）
③日付項目変換（日付データ変換時の日付マスク指定）

①、②を省略すると、年設定はウインドウ方式で１９３０年より（入力、出力とも）、“／”で日付区切りとなります。

１．16 セクタアドレス ～ＣＣＨＲＲ形式～

●セクタアドレスとは

ＩＢM形式のディスク・ファイルを理解し、操作するには、セクタアドレスというものを理解しないといけません。Ｗｉｎｄｏｗｓの世界では、セクタのアドレスなどどうでもよいことなのですが、ＩＢM形式の世界では、いろいろな理由で物理的なセクタの位置を示す「セクタアドレス」を頻繁に見たり、使ったりします。

●ＣＣＨＲＲ形式とは

セクタアドレスは、シリンダ番号、面番号、セクタ番号を組み合わせて「ＣＣＨＲＲ形式」と呼ぶ形式で表現します。ＣＣＨＲＲ形式は、一見すると、ただの５桁の大きな１０進数に見えますが、たとえば

７４１２６

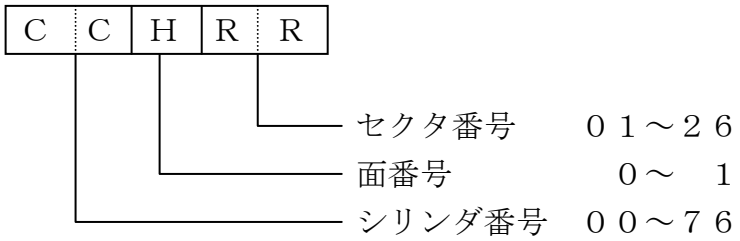
というセクタアドレスは、

７４－１－２６

と区切って読み、

シリンダ　　７４
面（ヘッド）　　１
セクタ　　２６

という物理セクタの位置を表します。



例をあげます。

ＣＣＨＲＲ	内　容
０００００１	インデックスシリンダの先頭セクタ（ＩＰＬ部）
０００００５	エラーマップラベル
０００００７	ボリュームラベル
０００００８	先頭のファイルラベル
０００２６	表面の最終ファイルラベル
００１０１	裏面の先頭ファイルラベル（１枚／２枚）
００１２６	裏面の最終ファイルラベル（１枚／２枚）
０１００１	データ部の先頭セクタ
７４１２６	データ部の最終セクタ（２ＨＤ－２５６）
７３０２６	データ部の最終セクタ（１Ｓ－１２８）

１．17 パラメータファイルの概要

Ｆ＊ＴＲＡＮ２００７の重要な特長の１つに、「パラメータファイル」が使えるということがあります。そのパラメータファイルについて説明します。

●パラメータファイルとは

Ｆ＊ＴＲＡＮ２００７に与えるパラメータの全部もしくは一部を、別のテキストファイルに書き込んでおき、随時それを参照して実行させることができます。そのファイルを、パラメータファイルと呼びます。

パラメータファイルのなかでは、

自由に改行でき、
各行にコメントをつけることができ、
文字数を気にせずパラメータ類を書け

ます。とても自由度が高いのです。

●なぜパラメータファイルが必要か

なぜ、パラメータファイルのようなものが必要なのでしょうか？

コマンドプロンプトレベルで入力できる文字数は、１２７文字（バイト）までです。バッチファイルでも同様に１行１２７文字（バイト）までです。

バッチファイルで１行に１２８文字（バイト）以上書くと、あふれた分が無条件に切り捨てられます。このとき、エラーメッセージは出ず、勝手に実行してしまうので注意してください。

Ｆ＊ＴＲＡＮ２００７では、ファイルを項目別に変換するとき／ＭＡＰオプションというものを使って、項目ごとに必要なパラメータを列挙して与えます。ところが、１行１２７文字以内という制限があると、たかだか２０～３０項目程度しか指定できません。１レコードあたり１００～２００項目ぐらいのファイルがよくあることを考えると、これは非常にきびしい制限です。この問題をパラメータファイルの利用によって解消できます。

もう１つの理由は、１項目ごとにコメントをつけておけないと、管理と修正がやりにくいものになってしまうためです。

●パラメータファイルを置くところ

パラメータファイルは、標準インストールモードでは「¥＜ユーザデータフォルダ＞¥Conf¥」に置く事を推奨しています。２００６互換インストールモードではインストールディレクトリに置く事を推奨しています。パス名を省略してパラメータファイルを指定すると、それぞれのインストールモードでは上記場所を参照するように設計されているからです。

●テキストエディタが必要

パラメータファイル自体はふつうのテキストファイルです。その作成・編集には“メモ帳”等を使用すればよいでしょう。

●パラメータファイルの参照の例

パラメータファイルは、コマンド行方式で実行するなら、

Ｃ：￥＞ｆｔ　［～］＋＋パラメータファイル名　［～］　↓

のようにして参照できます。バッチファイルに組み込むときも、同様です。

つまり、ほとんどどこでも＋＋のあとにパラメータファイル名を入力すれば、そのファイルをパラメータファイルとして認識します。このことは、パラメータファイルにＦ＊ＴＲＡＮ２００７の起動オプションやコマンドを含めてもよいことを意味しています。

１．18 パラメータファイルの参照の方法

パラメータファイルの参照方法について説明します。

●パラメータファイルの参照

パラメータファイル参照の正確な構文は、つぎのようになります。

＋＋ ［？：／ｄ：］［パス名指定］基本ファイル名 ［．Ｐ／．拡張子］

ｄ：はドライブ名

＋＋は、パラメータファイルの開始記号です。
？：は、インストールモードにより意味が変わります。

標準インストールモードの場合：¥<ユーザデータフォルダ>¥Conf¥
２００６互換インストールモードの場合：インストールディレクトリ

を、それぞれ表します。これが省略値で、もっぱらこれを使うのがふつうの運用です。
ｄ：はふつうのドライブ名です。

パス名指定（¥ディレクトリ名¥サブディレクトリ名¥・・・）ができます。指定したディレクトリ配下のパラメータファイルを参照します。

基本ファイル名の部分は、ふつうのファイル名です。業務に密着した名前にしてください。バッチファイルと同じ名前にするのも一つの方法です。

拡張子を省略すると、．Ｐになります。適当な別の拡張子を指定してもかまいません。
拡張子は、．Ｐでよいときでも、なるべく省略しないようにしてください。

●パラメータファイルの参照ができるところ

パラメータファイルの参照ができるところを具体的に示します。
つぎに示すように、コマンドプロンプトにつづくコマンド名ＦＴのあとならどこでも参照可能です。

Ｃ：¥>ｆｔ　・・・　＋＋～ ↓

この場合、ＦＴ＋＋～のようにＦＴと＋＋をくっつけてはいけません。少なくとも１個の空白を入れてください。

◆注意　----　パラメータファイルのネストはできない

パラメータファイルのネスト（入れ子：パラメータファイルのなかからパラメータファイルを参照すること）はできません。

◆注意　----　可変パラメータが渡せない

現在のところ、実行のたびに変えたい可変のパラメータを渡す機能はありません。したがって、ファイルの指定のように頻繁に変更されるものは、パラメータファイルに含めないようにすべきです。

１．19 パラメータファイルの書き方

例題を通して、パラメータファイルの書き方を説明します。

まず､下表のようなレコードレイアウトの､ＰＬＡＮＥＴという名前のＩＢＭファイルがあり、これをプリント形式のＷｉｎｄｏｗｓファイルに変換します。

ＰＬＡＮＥＴのレコードレイアウト

項番	項　　目	桁	幅	デ　ー　タ　形　式	内　容　例
1	Ｎｏ.	0	2	ＡＮＫ	1
2	和名	2	8	漢字	水星
3	英名	10	10	ＡＮＫ	MERCURY
4	読み	20	9	ＡＮＫ	ﾏｰｷｭﾘｰ
5	質量比	29	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
6	衛星数（確定済）	33	2	符号なしゾーン形式 整数部２桁	0
7	極大等級	35	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
8	英名の意味・由来	38	20	漢字	口神）神の使者
--	フィラー	58	6	--	--

これを変換するバッチファイルは、

```
@echo off ↓
:: 太陽系の惑星データの変換　ＩＢＭ→Ｗｉｎプリント形式 ↓
start /w ft getdata a:planet c:*.pr /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

のような内容になります。名前はＰＧＥＴ．ＢＡＴ（その１）としておきます。

／ＭＡＰオプションのところがかなり複雑です。そこで､バッチファイルＰＧＥＴ．ＢＡＴ（その２）と、パラメータファイルＰＧＥＴＰＲ．Ｐ（その１）に分けてみます。

ＰＧＥＴ．ＢＡＴ（その２）は、つぎのようになります。

```
@echo off ↓
:: 太陽系の惑星データの変換　ＩＢＭ→Ｗｉｎプリント形式 ↓
start /w ft getdata a:planet c:*.pr ++pgetpr.p ↓
```

そして、ＰＧＥＴＰＲ．Ｐ（その１）は、つぎのようになります。

```
/map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

これでも動きますが、少し読みやすくします。まず、

パラメータファイルのなかでは自由に改行できる
文字数を気にする必要がない

という特長があるので、キーワードをフルスペルにし、１行１項目にして書いてみると、ＰＧＥＴＰＲ．Ｐ（その２）は、つぎのようになり、だいぶわかりやすくなります。

```
/map ↓
      ank         2 ↓
      kanji       8 ↓
      ank         10 ↓
      ank         9 ↓
      packdisp    u4.3 ↓
      zonedisp    u2 ↓
      zonedisp    s2.1 ↓
      kanji       20 ↓
```

パラメータファイルのなかではコメントをつけることができる
－－から行末までがコメントになる

という特長を活かせば、もっとわかりやすくできます。ＰＧＥＴＰＲ．Ｐ（その３）は、つぎのようになります。これが決定版です。

```
-- 太陽系の惑星データの変換　ＩＢＭ→Ｗｉｎプリント形式（For GetData）↓
/map ↓
      ank         2         -- Ｎｏ．（惑星番号）　↓
      kanji       8         -- 和名　↓
      ank         10        -- 英名　↓
      ank         9         -- 読み　↓
      packdisp    u4.3      -- 質量比　↓
      zonedisp    u2        -- 衛星数（確定済）　↓
      zonedisp    s2.1      -- 極大等数（みかけ上の最大の明るさ）　↓
      kanji       20        -- 英名の意味・由来　↓
```

１．20 パラメータファイルの展開・圧縮

パラメータファイルがどのように展開されて、各コマンドに渡されるかを知っておくとよいでしょう。

●不要部分のスペース化とスペース圧縮

Ｆ＊ＴＲＡＮ２００７の各コマンドは、基本的には「１行」のパラメータ行を見て動作します。
パラメータファイルはその形のまま各コマンドに渡されるわけではありません。つぎのように、

コメントは削除され
改行コードはスペースに置き換えられ
制御コード類（タブも含む）もスペースに置き換えられ
こうしてできた連続するスペースは１個のスペースに置き換えられる

というふうに、スペース化とスペース圧縮がかけられ、１行に展開されます。こうして、各コマンドにパラメータの有効部分だけが渡されます。

●パラメータバッファ

パラメータ行を保持するためにプログラム内に固定領域があります。そこを「パラメータバッファ」と呼んでいます。

パラメータファイルの展開のされかたを、具体的な例で見てみましょう。前節の例の、

```
-- 太陽系の惑星データの変換　ＩＢＭ→Ｗｉｎプリント形式（For GetData）↓
/map ↓
      ank         2        -- Ｎｏ．（惑星番号）　↓
      kanji       8        -- 和名　↓
      ank         10       -- 英名　↓
      ank         9        -- 読み　↓
      packdisp    u4.3     -- 質量比　↓
      zonedisp    u2       -- 衛星数（確定済）　↓
      zonedisp    s2.1     -- 極大等数（みかけ上の最大の明るさ）　↓
      kanji       20       -- 英名の意味・由来　↓
```

という内容のパラメータは、つぎにように展開されます。

```
/map ank 2 kanji 8 ank 10 ank 9 packdisp u4.3 zonedisp u2 zonedisp s2.1 kanji 20
```

１．21 環境について

Ｆ＊ＴＲＡＮ２００７の動作に必要な設定ファイルには以下のものがあります。

動作環境情報ファイル（ＥＮＶ．ＩＮＩ、ファイル名固定）
主設定ファイル（ＦＴＲＡＮ．ＩＮＩ、ファイル名固定）
コード変換表ファイル（拡張子が．ＣＣＴ）
漢字対応表ファイル（拡張子が．ＫＫＴ）

動作環境情報ファイルは、以下の場所にあるファイルを使用しています。

標準インストールモード：%USERPROFILE%¥Application Data¥FTRAN-FD¥
（WindowsVista 以降では「%USERPROFILE%¥AppData¥Roaming¥FTRAN-FD¥」）
２００６互換インストールモード：インストールディレクトリ

それ以外のファイルは、初期状態として以下の場所にあるファイルを使用しています。

標準インストールモード：＜ユーザデータフォルダ＞¥FTRAN-FD¥EnvGroup¥Env¥
２００６互換インストールモード：インストールディレクトリ

これらの動作環境ファイル以外の設定ファイルが集まったフォルダを「**環境**」と呼び、Ｆ＊ＴＲＡＮ２００７ではこの環境を任意に切り替えて実行することが可能となっています。
環境の切り替えは起動オプション「／ＥＮＶｉｒｏｎｍｅｎｔ」で行います。詳細は「２．１ ＦＴ．ＥＸＥ　起動と終了」を参照してください。また、ＧＵＩから変更することもできます。
複数の環境が集まったフォルダを「**環境群フォルダ**」と呼びます。環境群フォルダも切り替えが可能です。環境群フォルダはＧＵＩから変更が可能です。
まず､環境群フォルダを選び､その中から使用する環境を選ぶという順番で環境を指定します。
環境群フォルダ、および環境の省略値はインストールモード別に以下のようになっています。

【標準インストールモード】
環境群フォルダ　　：＜ユーザデータフォルダ＞¥FTRAN-FD¥EnvGroup¥
環境　　：Env
【２００６互換インストールモード】
環境群フォルダ　　：Ｆ＊ＴＲＡＮ２００７のインストールディレクトリ
環境　　：指定なし（Ｆ＊ＴＲＡＮ２００７のインストールディレクトリ）

環境群フォルダ、および環境のＧＵＩからの変更方法は「操作説明書／解説編」を参照してください。

第２章

コマンド型の実行

２．１　ＦＴ．ＥＸＥ　起動と終了

■Ｆ＊ＴＲＡＮ２００７（コマンド行方式）の動作環境

Ｆ＊ＴＲＡＮ２００７をコマンド行方式で実行するには、

- ●各ＯＳの　ＭＳ－ＤＯＳプロンプト または コマンドプロンプト　からコマンドを入力
- ●Ｆ＊ＴＲＡＮ２００７の記述を含むバッチファイルの実行
- ●他のアプリケーション（Visual Basic など）からの利用

などがあげられます。

■Ｆ＊ＴＲＡＮ２００７（コマンド行方式）の起動

コマンド行方式で使うときの起動方法について説明します。

［ＳＴＡＲＴ／Ｗ［Ａ Ｉ Ｔ］］ＦＴ［起動オプション…／△］コマンド パラメータ［；…］

起動オプションのあとに、Ｆ＊ＴＲＡＮ２００７の内部コマンドとそのパラメータをセミコロン（；）で区切りながらいくつも指定できます（マルチコマンド）。
起動オプションのおわりにダミーオプションの／△（スラッシュと空白）をつけるのを忘れないでください。

◆注意　----　ＳＴＡＲＴ指定について

バッチファイルの中でＦ＊ＴＲＡＮ２００７の内部のコマンドを実行し、その結果に対して処理を行う場合、実行するコマンドの記述は、　ＳＴＡＲＴ／Ｗ　ＦＴ　コマンド　～　でなければなりません。ＳＴＡＲＴ／Ｗ　を指定すると、Ｆ＊ＴＲＡＮ２００７の処理が終了するまでつぎの処理がウェイトし、処理が並列で動作することを抑制します。ＳＴＡＲＴ／Ｗ　を指定しないと、バッチファイルの中の処理が正常に動作しない可能性があります。

■起動オプション

Ｆ＊ＴＲＡＮ２００７の起動オプションについて説明します。

●指定できる起動オプション

起動オプションにはつぎの８つがあります。

／Ｏｐｅｎ	スクリプトファイルをダブルクリックした時の動作を指定する
／Ｃｏｄｅ	コード変換表を指定する
／ＤｉｓｋＣｈａｎｇｅ	ディスク変換のタイミングを作る
／ＮｏＤｉｓｋＣｈａｎｇｅ	ディスク交換はしない
／ＷｉｎｄｏｗＣｌｏｓｅ	処理後、自動的に実行ウインドウを閉じる
／ＮｏＷｉｎｄｏｗＣｌｏｓｅ	処理後、自動的に実行ウインドウを閉じない
／ＷｉｎｄｏｗＤｉｓｐｌａｙ	処理中のウインドウを表示する
／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙ	処理中のウインドウを表示しない
／Ｔｉｔｌｅ	実行ウインドウ等のタイトルを変更する
／ＥＮＶｉｒｏｎｍｅｎｔ	環境名を指定する
／ＬｏｇＰａｔｈ	ログ出力先フォルダ指定
／ＥｒｒｏｒＬｏｇＮａｍｅ	エラーログファイル名
／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ	変換漏れログファイル名

●Ｆ＊ＴＲＡＮⅢにあり、Ｆ＊ＴＲＡＮ２００７にない起動オプション

Ｆ＊ＴＲＡＮⅢにある下記の起動オプションは、Ｆ＊ＴＲＡＮ２００７にはありません。

／ＮｏＴｉｔｌｅ	プログラムタイトル行を表示しない
／ＡｄｄｒｅｓｓＭｏｎｉｔｏｒ	セクタアドレスを表示する
／ＮｏＡｄｄｒｅｓｓＭｏｎｉｔｏｒ	セクタアドレスを表示しない

●起動オプションの詳細

／Ｏｐｅｎ ---- スクリプトファイルをダブルクリックした時の動作を指定する

／Ｏｐｅｎ　　［Ｅｄｉｔ｜＊｜Ｒｕｎ］

スクリプトファイルをダブルクリックした時の動作を指定します。

◆注意　----　スクリプトファイルでのみ使用する

このオプションはＧＵＩ部からスクリプトファイルを保存した時に付加される起動オプションです。通常、バッチファイルなどでは使用しません。

Ｅｄｉｔ：	スクリプトファイルのダブルクリック時、Ｆ＊ＴＲＡＮの画面が起動し、ダブルクリックしたスクリプトファイルが読み込まれた状態になる
＊：	環境設定で指定した動作
Ｒｕｎ：	スクリプトファイルのダブルクリック時、スクリプトファイルで指定されダブルクリックしたスクリプトファイルが読み込まれた状態になる

のなかから指定します。
詳細は「操作説明書／解説編」を参照してください。

／Ｃｏｄｅ ---- コード変換表を指定する

／Ｃｏｄｅ［ｄ：／？：］［パス名指定］［コード変換表名／デフォルト］［．拡張子／．ＣＣＴ］

使用するコード変換表ファイルを指定します。

ドライブ名の指定はつぎの、

Ａ：　～　Ｚ：	ふつうのドライブ名
？：	インストールディレクトリを表す仮想ドライブ名（省略値）
＠：	カレントドライブのカレントディレクトリを表す仮想ドライブ名

のなかから指定します。**ふつうはドライブ名を省略します**。ドライブ名が省略されると、**指定されている環境下からコード変換表を読み込みます**。これら、？：や＠：はＦ＊ＴＲＡＮ２００７特有のものです。

パス名指定（￥ディレクトリ名￥サブディレクトリ名￥・・・）ができます。指定したディレクトリ配下のコード変換表ファイルを参照します。

省略時の**コード変換表名**はデフォルト設定してあるコード変換表名です。

拡張子を省略すると．ＣＣＴとみなします。

導入編の「セットアップ」の章で、コード変換表については説明しています。そこで、

```
Ｃ：￥＞ｆｔ　コマンド　パラメータ ↓　　デフォルトのＣＣＴを選択
Ｃ：￥＞ｆｔ　／ｃｉ／　コマンド　パラメータ ↓　　Ｉ．ＣＣＴを選択
```

のように書きました。これは、

```
Ｃ：￥＞ｆｔ　／ｃｏｄｅ　？：デフォルト．ｃｃｔ／　コマンド　パラメータ ↓
                                    デフォルト設定のコード変換表名
Ｃ：￥＞ｆｔ　／ｃｏｄｅ　？：ｉ．ｃｃｔ／　コマンド　パラメータ ↓
```

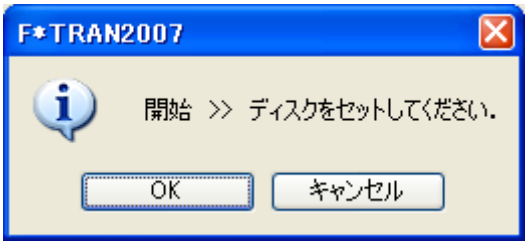
の省略・短縮指定だったのです。

／ＤｉｓｋＣｈａｎｇｅ -------- ディスク交換のタイミングを作る
／ＮｏＤｉｓｋＣｈａｎｇｅ ---- ディスク交換はしない

／ＤｉｓｋＣｈａｎｇｅ
／ＮｏＤｉｓｋＣｈａｎｇｅ

Ｆ＊ＴＲＡＮ２００７の起動・終了時に、ディスク交換のタイミングを作るかどうかを指定します。

／ＤｉｓｋＣｈａｎｇｅオプションを指定すると、ディスクの交換のタイミングを作ります。Ｆ＊ＴＲＡＮ２００７は起動時と終了時に確認ウインドウを開いて、応答待ちになります。



確認ウインドウのメッセージに従ってディスクを交換し、ＯＫボタンをクリックします。

定型的な処理をバッチファイル化したときなどに、ドライブ数が足りなくなってしまうことがあります。そのときに、この／ＤｉｓｋＣｈａｎｇｅオプションを使います。
また、変換処理をバッチファイル化したとき、いきなりファイル変換などがはじまっては面食らってしまいます。そのときも、このオプションを利用できます。
なお、起動時にメインウインドウが開いた時点で、コード変換表の読み込みはおわっています。また、＋＋記号を使ってパラメータファイルを参照しているときは、そのファイルの読み込みもおわっています。

／ＮｏＤｉｓｋＣｈａｎｇｅオプションは、ディスクの交換タイミングを作らないことを示します。これが省略値です。

／ＷｉｎｄｏｗＣｌｏｓｅ -------- 処理後、自動的に実行ウインドウを閉じる
／ＮｏＷｉｎｄｏｗＣｌｏｓｅ ---- 処理後、自動的に実行ウインドウを閉じない

／ＷｉｎｄｏｗＣｌｏｓｅ
／ＮｏＷｉｎｄｏｗＣｌｏｓｅ

Ｆ＊ＴＲＡＮ２００７のコマンドを実行すると、その進行状態を表示する実行ウインドウが開きます。その実行ウインドウを処理後、自動的に閉じるかどうかを指定します。

／ＷｉｎｄｏｗＣｌｏｓｅオプションを指定すると、コマンド処理の終了後、自動的に実行ウインドウを閉じます。既に処理が固定していて、バッチ処理を中断することなく実行する場合に指定します。

／ＮｏＷｉｎｄｏｗＣｌｏｓｅオプションは、コマンド処理の終了後、実行ウインドウを表示したままで“閉じる”ボタンの確認を求めます。コマンド処理の実行状態を確認したうえで、“閉じる”ボタンをクリックします。これが省略値です。

／ＷｉｎｄｏｗＤｉｓｐｌａｙ -------- 処理中の実行ウインドウを表示する
／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙ ---- 処理中の実行ウインドウを表示しない

／ＷｉｎｄｏｗＤｉｓｐｌａｙ
／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙ

Ｆ＊ＴＲＡＮ２００７のエラーダイアログやコマンド処理中の実行ウインドウを表示する／しないを指定します。

／ＷｉｎｄｏｗＤｉｓｐｌａｙオプションは、エラーダイアログやコマンド処理中の実行ウインドウを表示します。これが省略値です。

／ＮｏＷｉｎｄｏｗＤｉｐｌａｙオプションを指定すると、エラーダイアログ、およびコマンド処理中の実行ウインドウを表示しません。タスクバーにもＦ＊ＴＲＡＮ２００７の表示はでませんので、他のアプリケーションからＦ＊ＴＲＡＮ２００７を隠して使いたい場合等に指定します。ただし、この指定に関わらず、マルチボリュームの場合のディスク要求ウインドウ、変換コマンドの／Ｑｕｅｒｙオプションによる応答画面、および起動時の初期エラーダイアログについては表示されます。

／ＮｏＷｉｎｄｏｗＤｉｐｌａｙオプションを指定した場合は、／ＷｉｎｄｏｗＣｌｏｓｅオプション、およびＮｏＷｉｎｄｏｗＣｌｏｓｅオプションは無視されます。

◆重要　----　バッチファイルまたは他のアプリケーションから利用するには

通常、Ｆ＊ＴＲＡＮ２００７をバッチファイルまたは他のアプリケーションからバッチ処理として利用するには、／ＷｉｎｄｏｗＣｌｏｓｅオプションをつけて起動し、処理が中断することなく動作するようにします。

◆参考　----　Ｆ＊ＴＲＡＮ２００７を隠して実行する他の方法

／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙオプション同様にＦ＊ＴＲＡＮ２００７を隠して実行する方法があります。ＳＴＡＲＴコマンドの／ＭＩＮ（最小化実行オプション）です。

ＳＴＡＲＴ　／Ｗ　／ＭＩＮ　ＦＴ　／ＷＣ／　～

と指定します。／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙオプションとの違いは、タスクバーの表示がでてしまうことです。

◆参考　----　Ｆ＊ＴＲＡＮ２００７の内容を含むバッチファイルを隠して実行するには

“ＳＴＡＲＴ　／Ｗ　ＦＴ　／ＮＷＤ／　～”および“ＳＴＡＲＴ　／Ｗ　ＦＴ　／ＷＣ／　～”の内容を含むバッチファイル全体を隠して実行する場合は、つぎの方法を採ります。

作成したバッチファイルのプロパティを開き、プロパティ内プログラム画面の 実行時の大きさの項目を“最小化の状態” プログラム終了時にウインドウを閉じるの□をチェックが入った状態

にします。この設定のバッチファイルを実行するとタスクバーのみの表示となり、エラーダイアログについても表示しません。ただし、マルチボリュームの場合のディスク要求ウインドウおよび、変換コマンドの／Ｑｕｅｒｙオプションによる応答画面については表示されるので、注意してください。

／ＬｏｇＰａｔｈ ---- ログ出力先フォルダを指定する

```
／ＬｏｇＰａｔｈ　｛Ｂｏｔｈ｜Ｃｏｎｖｅｒｔ｜Ｅｒｒｏｒ｝　出力先パス名
／ＬｏｇＰａｔｈ　Ｂｏｔｈ　＊
```

Ｆ＊ＴＲＡＮ２００７が出力するログ（エラーログ、変換漏れログ）の出力先を指定します。以下の指定により一つのオプションでエラーログ､変換漏れログどちらの指定も可能になっています。

◇［Ｂｏｔｈ］指定時
　指定されたパス名はエラーログ、変換漏れログ両方に有効
◇［Ｃｏｎｖｅｒｔ］指定時
　指定されたパス名は変換漏れログのみに有効
◇［Ｅｒｒｏｒ］指定時
　指定されたパス名はエラーログのみに有効

これらの指定は必ずどれかを指定する必要があります。
　オプションの省略時は「／ＬｏｇＰａｔｈ　Ｂｏｔｈ　＊」が指定されたとして動作します。これはエラーログ､変換漏れログともにインストールモード別に以下の場所に出力されるという意味です。

標準インストールモードの場合：¥<ユーザデータフォルダ>¥Log¥
２００６互換インストールモードの場合：インストールディレクトリ

　指定されたパスが存在しなかったり､存在してもディレクトリではない場合も上記場所に出力されます。

／ＥｒｒｏｒＬｏｇＮａｍｅ ---- エラーログのファイル名を指定する

```
／ＥｒｒｏｒＬｏｇＮａｍｅ　エラーログファイル名
／ＥｒｒｏｒＬｏｇＮａｍｅ　＊
```

Ｆ＊ＴＲＡＮ２００７が出力するエラーログのファイル名を指定します。

　オプションの省略時は「／ＥｒｒｏｒＬｏｇＮａｍｅ　＊」が指定されたとして動作します。この場合、“FT.ERR”というファイル名で出力されます。

／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ ---- 変換漏れログのファイル名を指定する

／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ　変換漏れログファイル名 ／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ　＊

Ｆ＊ＴＲＡＮ２００７が出力する変換漏れログのファイル名を指定します。

オプションの省略時は「／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ　＊」が指定されたとして動作します。この場合、変換方向により“Ftglog.txt”（ＩＢＭ→Ｗｉｎ変換時)、“Ftplog.txt”（Ｗｉｎ→ＩＢＭ変換時）というファイル名で出力されます。

／Ｔｉｔｌｅ ---- 実行ウインドウ等のタイトルを変更する

／Ｔｉｔｌｅ　出力文字列（６４バイト以内） ／Ｔｉｔｌｅ

Ｆ＊ＴＲＡＮ２００７のコマンド処理中に出力される実行ウインドウ等のタイトルを変更できます。通常は“Ｆ＊ＴＲＡＮ２００７”というタイトルで出力されますが、これを任意の文字列に置き換えて出力させることができます。たとえば、

／Ｔｉｔｌｅ　変換データ作成

のように指定して他のアプリケーションから起動すれば、あたかも、そのアプリケーションのウインドウであるかのように見せることができます。出力文字列に空白が含まれる場合は、

／Ｔｉｔｌｅ　“変換処理　その１”

のように出力文字列全体を“”で囲みます。
出力文字列を省略すると、“Ｆ＊ＴＲＡＮ２００７”として出力されます。

／ＥＮＶｉｒｏｎｍｅｎｔ ---- 環境名を指定する

```
／ＥＮＶｉｒｏｎｍｅｎｔ［ 環境名
                          ＊ ］
```

使用する環境名を指定します。

環境名は環境群フォルダ配下にあるフォルダ名です。環境群フォルダの省略値、および環境名の省略値はインストールモード別に以下のようになっています。

【標準インストールモード】

環境群フォルダ　　：＜ユーザデータフォルダ＞¥FTRAN-FD¥EnvGroup¥
環境　　：Env

【２００６互換インストールモード】

環境群フォルダ　　：Ｆ＊ＴＲＡＮ２００７のインストールディレクトリ
環境　　：指定なし（Ｆ＊ＴＲＡＮ２００７のインストールディレクトリ）

２００６互換インストールモードの場合、環境名で「＊」が指定できます。「＊」指定の場合は、環境群フォルダそのものが環境として使用されます。標準インストールモードでは「＊」指定はできません。必ず存在する環境名を指定してください。

／Ｃｏｄｅオプションで指定されるコード変換表(.ＣＣＴ)ファイルは、環境名として指定されたフォルダに格納されているものが使用されます。したがって、コード変換表ファイル名が同じでも、ＥＮＶｉｒｏｎｍｅｎｔオプションにて違う環境名を指定することで、別のコード変換表設定でＦ＊ＴＲＡＮ２００７を実行することが可能となっています。

●起動オプション省略時の動作

起動オプションをすべて省略すると、

／Ｏｐｅｎ　＊：デフォルト設定の動作設定に従う	
／Ｃｏｄｅ　？：デフォルト設定のコード変換表名	
	コード変換表はデフォルトのコード変換表
／ＥＮＶｉｒｏｎｍｅｎｔ　Ｅｎｖ	“Ｅｎｖ”ディレクトリを環境として使用
	（標準インストールモードの場合）
／ＥＮＶｉｒｏｎｍｅｎｔ　＊	インストールディレクトリを環境として使用
	（２００６互換インストールモードの場合）
／ＮｏＤｉｓｋＣｈａｎｇｅ	ディスク交換はしない
／ＮｏＷｉｎｄｏｗＣｌｏｓｅ	処理後の実行ウインドウを自動で閉じない
／ＷｉｎｄｏｗＤｉｓｐｌａｙ	処理中の実行ウインドウを表示する
／Ｔｉｔｌｅ	実行ウインドウ等のタイトルを変更しない
／ＬｏｇＰａｔｈ　Ｂｏｔｈ　＊	エラーログ、変換漏れログともにインストール
	ディレクトリに出力
／ＥｒｒｏｒＬｏｇＮａｍｅ　＊	デフォルトのエラーログファイル名で出力
／ＣｏｎｖｅｒｔＬｏｇＮａｍｅ　＊	
	デフォルトの変換漏れログファイル名で出力

と指定したとみなします。

＜起動オプション指定時の注意＞

起動オプションとコマンドの両方を指定するときは、起動オプションのおわりに「ダミーオプションの／△（スラッシュと空白）をつけて、起動オプションを閉じる」のを忘れないでください。そうしないと、コマンド名が起動オプションに渡すパラメータだと解釈されてエラーになります。

例を示します。ドライブＡ：のあるＩＢＭファイルをドライブＣ：のＷｉｎｄｏｗｓファイルに変換するのに、ＧｅｔＤａｔａ（ｇｄ）コマンドを、つぎのように使うとします。

```
Ｃ：￥＞ｆｔ　ｇｅｔｄａｔａ　～　↓
```

その際に、／ＤｉｓｋＣｈａｎｇｅオプションをつけて起動・終了時にディスクの交換をするなら、

```
ｆｔ　／ｄｃ／　ｇｅｔｄａｔａ　～　↓
ｆｔ　／ｄｃ　ｇｅｔｄａｔａ　～　↓
ｆｔ　／ｄｃ／ｇｅｔｄａｔａ　～　↓
```

の３つの指定のうち正しいのはどれでしょうか？

```
ｆｔ　／ｄｃ／　ｇｅｔｄａｔａ　～　↓ ---------- ○
```

これが正解です。きちんと／ＤｉｓｋＣｈａｎｇｅオプションを／と空白で閉じています。ｇｅｔｄａｔａは正しくコマンド名として認識されます。

```
ｆｔ　／ｄｃ　ｇｅｔｄａｔａ　～　↓ ---------- ×
```

これは、よくやりがちなミスです。／ＤｉｓｋＣｈａｎｇｅオプションを閉じる／と空白を忘れています。この場合、／ＤｉｓｋＣｈａｎｇｅオプションにｇｅｔｄａｔａ　～というパラメータをつけたと解釈されます。そして、「オプションに誤りがあります．　→ getdata」というエラーになります。

```
ｆｔ　／ｄｃ／ｇｅｔｄａｔａ　～　↓ ---------- ×
```

この指定は、／ＤｉｓｋＣｈａｎｇｅオプションを／で閉じたのまではよいのですが、空白がないために、／ｇｅｔｄａｔａという起動オプションを指定したものと解釈されます。そして、「オプションはありません．　→ /getdata」というエラーになります。

この例では／ＤｉｓｋＣｈａｎｇｅオプションを取り上げましたが、／Ｃｏｄｅオプションでもまったく同様です。たとえば、起動・終了時にディスク交換をし、同時にＮＥＣ漢字変換用のコード変換表を選択するなら、起動オプションとコマンドを

Ｃ：￥＞ｆｔ　／ｄｃ／ｃｎ／　ｇｅｔｄａｔａ　～　↓

のように指定します。

■Ｆ＊ＴＲＡＮ２００７（コマンド行方式）の終了

コマンド行方式で使うときの、終了の方法について説明します。

Ｆ＊ＴＲＡＮ２００７をコマンド行方式で起動したときは、その内部のコマンドが終了すると、進行状態を示す実行ウインドウが“閉じる”ボタンをクリックすることを求めてきます。“閉じる”ボタンをクリックすると、Ｆ＊ＴＲＡＮ２００７は終了します。
ただし、／ＷｉｎｄｏｗＣｌｏｓｅオプションが指定されている場合は、自動的に実行ウインドウが閉じて、Ｆ＊ＴＲＡＮ２００７は終了します。

●終了コード（エラーレベル）で正常／エラーの区別を返す

Ｆ＊ＴＲＡＮ２００７は終了時に終了コードを返します。

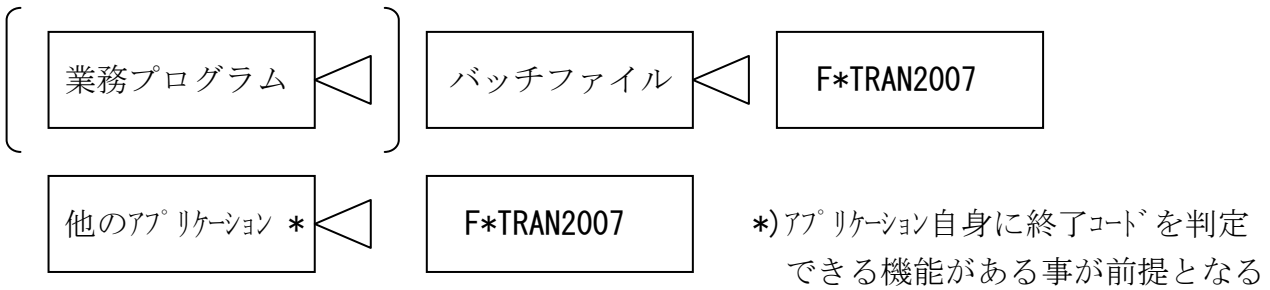
０は正常
１以上はエラー

です。ただし、エラーの詳細な区別は返しません。

起動時にパラメータとして内部のコマンドを指定すると、このコマンドの正常・エラーの区別が返されます。したがって、バッチファイルや他のアプリケーション内で処理結果を終了コードで判定し、そのあとの処理を切り替えることができます。

●終了コードはバッチファイルまたは他のアプリケーションのなかで判定する

組み込み用途の場合、バッチファイルまたは他のアプリケーションで直接Ｆ＊ＴＲＡＮ２００７を呼び出して、終了コードの判定はそのバッチファイルまたは他のアプリケーションのなかで行います。



バッチファイル内での記述は、

```
ｓｔａｒｔ　／ｗ　ｆｔ　～
ｉｆ　ｅｒｒｏｒｌｅｖｅｌ　１　ｇｏｔｏ　エラー処理へ
```

と書くのが常套手段です。たとえば、ｉＬｉｓｔ(ｉｌ)コマンドを使って

```
ｓｔａｒｔ　／ｗ　ｆｔ　ｉｌｉｓｔ　～
ｉｆ　ｅｒｒｏｒｌｅｖｅｌ　１　ｇｏｔｏ　エラー処理へ
```

と書けば、ファイル変換に先立って、あらかじめ

```
ディスクがセットされているかどうかの判定
ＩＢＭ形式のディスクかどうかのおおまかな判定
目的のファイルがあるかどうかの判定
```

などができます。(ディスクがセットされているかどうかの判定は、ＩｓＲｅａｄｙコマンドを使うことを推奨します。)

また、

```
ｓｔａｒｔ　／ｗ　ｆｔ　ｇｅｔｄａｔａ　～
ｉｆ　ｅｒｒｏｒｌｅｖｅｌ　１　ｇｏｔｏ　エラー処理へ
```

と書けば、ファイル変換［この場合はＧｅｔＤａｔａ(ｇｄ)コマンド］が成功したか否かを判定することができます。

終了コードの判定例を示します。

例）ＩＢＭファイルＴＥＳＴがあれば、その内容を表示する

バッチファイルでドライブＡ：にＩＢＭファイルＴＥＳＴがあるかどうかを判定します。あれば、その内容をコンソールに表示し、なければ、エラーメッセージを表示します。

＜ＴＥＳＴ．ＢＡＴ＞

```
@echo off ↓
:: ドライブＡ：にＩＢＭファイルＴＥＳＴがあれば、その内容を表示する ↓
start /w ft /wc/ isready a:  -- ilist a:.index でもチェックできる ↓
if errorlevel 1 goto notready ↓
↓
start /w ft /wc/ ilist a:test ↓
if errorlevel 1 goto notfound ↓
↓
start /w ft /wc/ gettext a:test test ↓
if exist test type test ↓
goto end ↓

:notready ↓
 echo ドライブの準備ができていません. ↓
 goto end ↓
↓
:notfound ↓
 echoＩＢＭファイルＴＥＳＴがありません. ↓
↓
:end ↓
```

◆参考　----　上記の TEST.BAT の内容を他のアプリケーションで実現するには

本書の末尾、　－参考－　を参照してください。

■使用例

例１）コマンド行方式での起動

ドライブＣ：のＷｉｎｄｏｗｓファイルＦＯＯ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＦＯＯに変換します。漢字はなく、ただのテキストファイルです。

```
Ｃ：￥＞ｆｔ　ｐｕｔｔｅｘｔ　ｃ：ｆｏｏ．ｔｘｔ　ａ：↓
```

例２）起動オプションつきの、コマンド行方式での起動

ドライブＣ：のすべてのＷｉｎｄｏｗｓファイルを、ドライブＡ：のＩＢＭファイルにテキストファイル変換します。漢字はＮＥＣ漢字に変換するので、／ｃｎを指定します。また、コマンド実行後の実行ウインドウを自動的に閉じてＦ＊ＴＲＡＮ２００７を終了させるために、／ＷｉｎｄｏｗＣｌｏｓｅオプションを指定します。

```
Ｃ：￥＞ｆｔ　／ｗｃ／ｃｎ／　ｐｕｔｔｅｘｔ　ｃ：＊．＊　ａ：／ｃｋ↓
　　（／ＷｉｎｄｏｗＣｌｏｓｅ　／Ｃｏｄｅ　？：ｎ．ｃｃｔ）
```

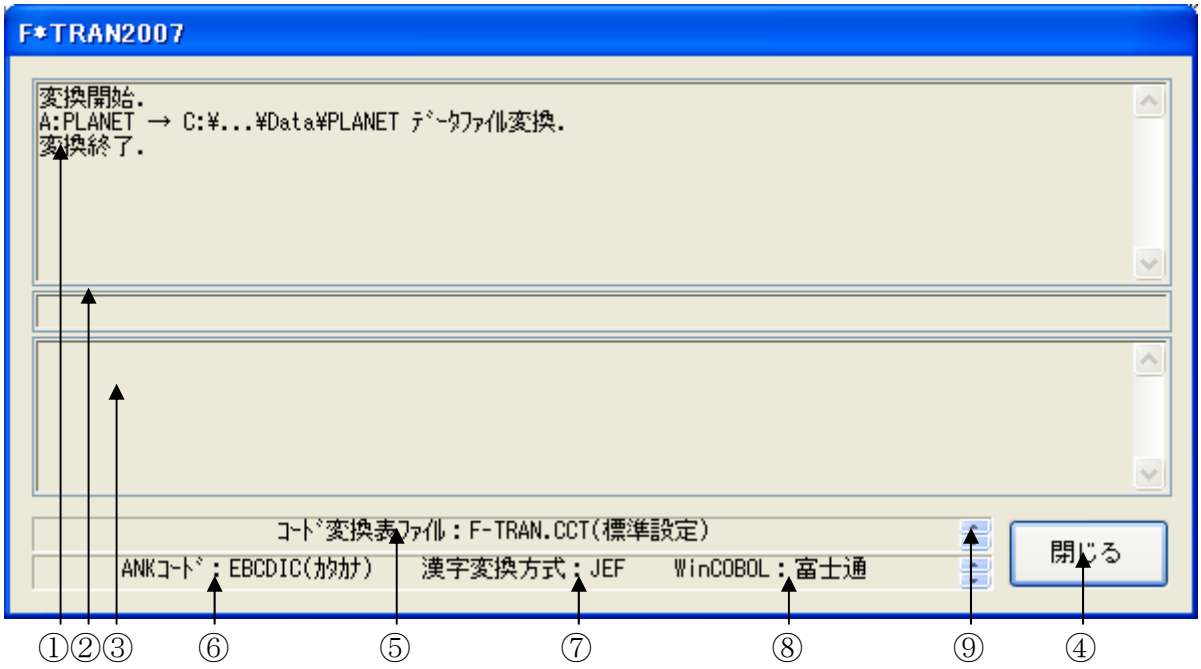
例３）コマンド行方式でｉＬｉｓｔ（ｉｌ）コマンドを使う

コマンド行方式でｉＬｉｓｔ（ｉｌ）コマンドを使って、ドライブＡ：のＩＢＭディスクの内容一覧を表示させます。

```
Ｃ：￥＞ｆｔ　ｉｌｉｓｔ　ａ：↓
```

２．２　実行ウインドウ

Ｆ＊ＴＲＡＮ２００７は、コマンド実行時につぎの実行ウインドウを開きます。



①処理中のメッセージを表示するメッセージフィールドです。
②処理中の進行状況を表示するフィールドです。
③処理中のエラーメッセージを表示するフィールドです。
④処理が終了したら、その処理結果を確認して、“閉じる”ボタンをクリックします。
⑤使用しているコード変換表ファイルの情報を表示しています。⑨上段のボタンをクリックして、インストールディレクトリの情報を表示させることもできます。
⑥ＡＮＫコードの設定情報（EBCDIC（ｶﾀｶﾅ）、EBCDIC（英小）、JIS8･ASCII）を表示しています。ＡＮＫ変換の基準となる重要な設定情報です。
⑦漢字変換方式の設定情報（JEF、JIS ほか）を表示しています。
漢字変換の基準となる重要な設定情報です。
⑧Ｗｉｎｄｏｗｓ　ＣＯＢＯＬベンダの設定情報（なし、富士通、日立、NEC、ﾏｲｸﾛﾌｫｰｶｽ、Acucorp）を表示しています。
⑨上段のボタンは、コード変換表ファイルとインストールディレクトリの情報表示を切り替えるボタンです。下段のボタンは、ＡＮＫコード、漢字変換方式、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのベンダとブロック配置モード（一般ﾓｰﾄﾞ、日立ﾓｰﾄﾞ）、ＩＢＭディスク形式（一般、三菱など）の情報表示を切り替えるボタンです。

◆参考　----　実行ウインドウのコントロール

実行ウインドウの状態は、コマンド実行時の起動オプション（／WC、／NWC、／WD、／NWD）でコントロールすることができます。／WC（ウィンドウクローズ）指定をすることで、コマンド実行後の実行ウインドウを自動的に閉じることができ、／NWD（ノーウィンドウディスプレイ）指定をすることで、コマンド実行中の実行ウインドウを非表示にすることができます。詳しくは、起動オプションの頁を参照してください。

２．３　Ｇｅｔ系コマンド

Ｇｅｔ系のファイル指定と共通オプション

ここでは、Ｇｅｔ系コマンドの共通事項を説明します。頭にＧｅｔがつく３つのコマンド

ＧｅｔＴｅｘｔ（ｇｔ）コマンド　　ＩＢＭ→Ｗｉｎテキストファイル変換
ＧｅｔＤａｔａ（ｇｄ）コマンド　　ＩＢＭ→Ｗｉｎデータファイル変換
ＧｅｔＲａｎｄ（ｇｒ）コマンド　　ＩＢＭ→Ｗｉｎランダムファイル変換

をまとめてＧｅｔ系コマンドと呼び、ＩＢＭファイルをＷｉｎｄｏｗｓファイルに変換するのに使います。この３つのコマンドを用途によって使い分けます。これら３コマンドは、よく似た兄弟です。

■コマンド形式

コマンド	パラメータ
ＧｅｔＴｅｘｔ	［ＩＢＭファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］
ｇｔ	
ＧｅｔＤａｔａ	
ｇｄ	
ＧｅｔＲａｎｄ	
ｇｒ	

Ｇｅｔ系コマンドは、どれも上に示したパラメータ形式を持っています。ファイルの指定方法は３つともまったく同じです。オプションの指定は共通のものもあれば、３つ別々のものもあります。

■パラメータの説明

●ＩＢＭファイル

ｄ：ＩＢＭファイル名

ｄ：はドライブ名

入力側のＩＢＭファイルを指定します。

ドライブ名はＡ：～Ｐ：、または０：～３：で指定します。省略はできません。

ＩＢＭファイル名も省略できません。ＩＢＭファイル名にはワイルドカード文字（＊）を含めることができます。ワイルドカード文字を使うと、指定ドライブのＩＢＭディスクから一致するファイルをすべて検索し、変換の対象にします。

まとめると、１本のファイルだけを指定するなら、

Ａ：ＰＬＡＮＥＴ　　のような指定になり、

全ファイルを指定するなら、

Ａ：＊　　のような指定になります。

●Ｗｉｎｄｏｗｓファイル

```
  ｄ：
［ｄ：］［パス名指定］＊              ［．拡張子］
［ｄ：］［パス名指定］基本ファイル名［．拡張子］
```

ｄ：はドライブ名

出力側のＷｉｎｄｏｗｓファイルを指定します。

ドライブ名はＡ：～Ｚ：、＠：、？：のどれかで指定します。ドライブ名を指定すると、そのドライブにファイルを作ります。
ドライブ名は省略可能です。ドライブ名を省略すると、カレントドライブにファイルを作ります。

パス名指定（￥ディレクトリ名￥サブディレクトリ名￥・・・）ができます。指定したディレクトリ配下にファイルを作ります。パス名指定を省略すると、カレントディレクトリにファイルを作ります。

基本ファイル名の部分には、ふつうの基本ファイル名、または＊（省略値）を指定します。＊を指定すると、ＩＢＭファイル名がＷｉｎｄｏｗｓ側の基本ファイル名になります。複数ファイルの一括変換を行うときは、必ず＊を指定してください。

拡張子を省略すると、拡張子なしのファイルになります。しかし、拡張子をつけたほうがファイルの管理が容易になるので、なるべく適当な拡張子を指定してください。

以上がＷｉｎｄｏｗｓファイルの指定方法です。まとめると、ファイル名を引き継ぐときは

Ｃ：＊．ＤＡＴ　　のような指定になり、

ファイル名をつけ替えるときは

Ｃ：ＮＥＷＮＡＭＥ．ＤＡＴ　　のような指定になります。

◆注意　----　使用できるデバイスファイルについて

Ｗｉｎｄｏｗｓファイルとして使用できるデバイスファイルは、ＮＵＬ（ダミーデバイス）だけです。

●オプション

Ｇｅｔ系コマンドには各コマンドに共通のオプションと、コマンドごとに違うオプションがあります。

Ｇｅｔ系コマンド共通のオプション

／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＥＯＦ	ＥＯＦコードをつける
／ＮｏＥＯＦ	ＥＯＦコードをつけない
／ＲＥＰｌａｃｅ	変換先ファイルの置換設定

ＧｅｔＴｅｘｔ（ｇｔ）コマンド固有のオプション

／Ｃｏｄｅ	ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する
／ＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮する
／ＮｏＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮しない

ＧｅｔＤａｔａ（ｇｄ）コマンド固有のオプション

／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ	引用符くくりの詳細を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

ＧｅｔＲａｎｄ（ｇｒ）コマンド固有のオプション

／Ｓｉｚｅ	Ｗｉｎｄｏｗｓ側のレコード長を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

以下、Ｇｅｔ系コマンドに共通のオプションを説明します。

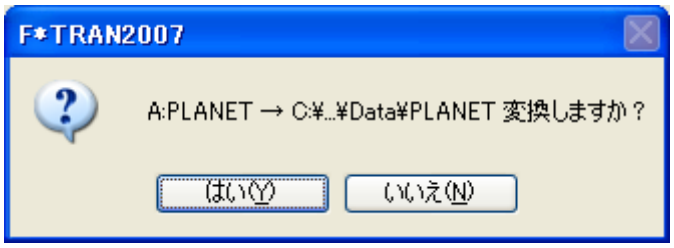
／Ｑｕｅｒｙ -------- 変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動変換する

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

１ファイルごとに処理を問い合わせるか否かを指定します。

／Ｑｕｅｒｙオプションを指定すると、ファイル名を確認しながら変換できます。Ｆ＊ＴＲＡＮ２００７は、１ファイルごとに処理を実行するか問い合わせてきます。
つぎのどれかで応答してください。この機能は、比較的小さいファイルが多数あって、そのうちいくつかを選んで変換したいときなどに便利です。

１ファイルの変換



はい（Ｙ）　　表示中のファイルを変換する
いいえ（Ｎ）　　表示中のファイルは変換しない

２ファイル以上の変換

はい（Ｙ）　　表示中のファイルを変換する
すべて変換（Ａ）　　全ファイル変換に切り替え、以降のファイルをすべて変換する
いいえ（Ｎ）　　表示中のファイルは変換しない
キャンセル　　これ以降の変換処理を中断する

／ＮｏＱｕｅｒｙオプションを指定すると、ファイル名の確認なしで自動的に指定のファイルをすべて変換します。これが省略値です。

／ＭｕｌｔｉＶｏｌｕｍｅ ------ マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ ---- マルチボリューム処理機能を無効にする

／ＭｕｌｔｉＶｏｌｕｍｅ ／ＳｉｎｇｌｅＶｏｌｕｍｅ

マルチボリューム処理機能を有効にするか、無効にするかを指定します。

／ＭｕｌｔｉＶｏｌｕｍｅオプションを指定すると、マルチボリューム処理機能が有効になります。ＩＢＭファイルに「マルチボリュームで、先頭 =″01 C″」と表示されていれば、そのＩＢＭファイルの変換後、ＩＢＭディスクの差し替えメッセージを出してマルチボリューム処理に入ります。こちらが省略値です。

／ＳｉｎｇｌｅＶｏｌｕｍｅオプションを指定すると、マルチボリューム処理機能は抑止されます。ＩＢＭファイルに「マルチボリュームで、何番目 = ″nn C / nn L″ 」と表示されていても無視し、別々の単一ボリュームファイルとして扱います。この機能は、

運用上、１枚ずつフロッピーディスクにバラバラに落としたいとき
途中だけ抜き出して変換したいとき

などに有効です。

／ＥＯＦ -------- ＥＯＦコードをつける
／ＮｏＥＯＦ ---- ＥＯＦコードをつけない

／ＥＯＦ ／ＮｏＥＯＦ

ＥＯＦコード（１ＡＨ）の扱いを指定します。

／ＥＯＦオプションを指定すると、Ｗｉｎｄｏｗｓファイルのおわりにコードをつけます。現在では少なくなりましたが、テキストファイルのおわりにＥＯＦコードがついていないとエラーにするソフトがあります。その場合にも対処するための機能です。

／ＮｏＥＯＦオプションを指定すると、ＥＯＦコードはつけません。これがふつうだと思ってください。こちらが省略値です。

／ＲＥＰｌａｃｅ ---- 変換先ファイルを置き換えるか指定する

／ＲＥＰｌａｃｅ　｛ＹＥＳ｜ＮＯ｜？｝

変換先ファイルと同名のファイルが同一フォルダ上に存在した場合の動作を以下の３通りから指定します。

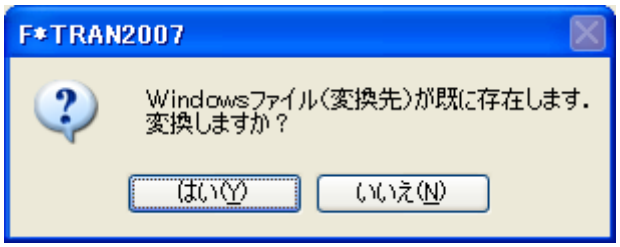
◇［ＹＥＳ］指定時（省略値）
出力先ファイルと同名のファイルが同一フォルダ上に存在した場合は、自動で置き換えます。

◇［ＮＯ］指定時
出力先ファイルと同名のファイルが同一フォルダ上に存在した場合は以下のメッセージをログに出力し、プログラムは正常終了します。（異常終了ではありません）

・ＩＢＭ→Ｗｉｎ変換時のメッセージ
「Ｗｉｎｄｏｗｓファイルが既に存在する為、変換できません.」

◇［？］指定時
出力先ファイルと同名のファイルが同一フォルダ上に存在した場合は､以下の問い合わせダイアログを表示して、ユーザの指示を待ちます。

【ＩＢＭ→Ｗｉｎ変換時の問い合わせ画面】



◆注意　----　／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙとの関係

“／ＲＥＰｌａｃｅ　？”指定で、出力ファイルが既に存在した場合、“／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙ”指定に関わらず問い合わせ画面を表示します。

■使用例

例１）一番単純な変換をする

ドライブＡ：のＩＢＭファイルＳＡＭＰＬＥを、ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥに変換します。拡張子はつけません。

```
C:¥>ft getdata a:sample c: ↓
```

同名で拡張子が不要なときは、出力先のドライブ名だけ指定すれば十分です。

例２）拡張子をつける

例１と同様ですが、変換後のＷｉｎｄｏｗｓファイルに拡張子．ＴＸＴをつけます。

```
C:¥>ft gettext a:sample c:*.txt ↓
```

同名で拡張子をつけるときは、この例のように＊．ＴＸＴの形式で指定します。

例３）ＩＢＭディスクをベリファイ（欠陥検査）する

ドライブＡ：のＩＢＭディスクをベリファイ（欠陥検査）します。ＩＢＭディスクの．ＤＡＴＡ（データ部全体を表すディスク領域名）を指定すれば実現できます。

```
C:¥>ft getdata a:.data test ↓
```

例４）ワイルドカードを使う

ワイルドカードを使って、関連ファイルを一括変換します。ドライブＡ：にはＩＢＭ形式でＣＯＢＯＬのソースプログラムが数本入っています。それらをすべてドライブＣ：のＷｉｎｄｏｗｓファイルに変換し、拡張子は．ＣＢＬにします。

```
C:¥>ft getext a:* c:*.cbl ↓
```

例５）ワイルドカード文字を使う。確認つき

ワイルドカード文字を使って、関連ファイルを一括変換します。／Ｑｕｅｒｙオプションをつけて、ファイル名を確認しながら変換します。

ドライブＡ：にはＩＢＭ形式でＦＯＲＴＲＡＮのソースプログラムが数本入っています。さらに、ほかのファイルも入りまじっています。それらのうち、ＦＯＲＴＲＡＮのソースプログラムだけを選択しながらドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子は、．ＦＯＲにします。

```
C:¥>ft gettext a:* c:*.for /q ↓          (/Query)
```

この例のように、／Ｑｕｅｒｙオプションをつけると、１ファイルごとに「変換しますか？」と聞いてきます。変換するなら、“はい”ボタンをクリックします。変換しないときは、“いいえ”ボタンをクリックするとスキップします。

例６）別名をつける

ドライブＡ：のＩＢＭファイルＤＡＴＡを、ドライブＣ：のＷｉｎｄｏｗｓファイルＨＥＬＬＯ．Ｃに変換します。

```
C:¥>ft gettext a:data c:hello.c ↓
```

ＩＢＭファイルの名前はＤＡＴＡになっていることが多いものです。名前がＤＡＴＡのときは、本当の内容にふさわしい名前につけ替えてください。

例７）項目別に分けてプリント形式に変換する

ドライブＡ：のＩＢＭファイルＰＬＡには、さまざまな形式の項目が入りまじっています。それを、プリント形式に変換します。ＧｅｔＤａｔａ（ｇｄ）コマンドを使い、／ＭＡＰオプションで詳細な指定をします。変換先はドライブＣ：とし、名前はＰＬＡ．ＰＲＮにします。

```
C:¥>ft getdata a:pla c:*.prn /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
      ⋮
C:¥>_
```

ここでは、「／ＭＡＰオプションとはこんなふうに指定するのだ」ぐらいに思ってください。

例８）例７をバッチファイル化する

例７の方法では、／ＭＡＰオプションの指定を入力するのが大変です。そこで、それをバッチファイルにして実行します。バッチファイルの名前はＧＥＴＰＬＡ．ＢＡＴにします。

＜ＧＥＴＰＬＡ．ＢＡＴ＞

```
@echo off ↓
:: ＰＬＡファイルのＩＢＭ形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
start /w ft getdata a:pla c:*.prn /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

例９）パラメータファイルを利用する

例８の方法でバッチファイル化しても、まだ／ＭＡＰオプションの指定がわかりにくい難点があります。そこで、パラメータファイルを利用します。パラメータファイルの名前はＧＥＴＰＬＡ．Ｐとし、インストールディレクトリに置いてあるものとします。

＜ＧＥＴＰＬＡ．ＢＡＴ＞

```
@echo off ↓
:: ↓
:: ＰＬＡファイルのＩＢＭ形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
:: ↓
start /w ft getdata a:pla c:*.prn ++getpla.p ↓
```

＜ＧＥＴＰＬＡ．Ｐ＞

```
-- For GetData ---------------------------------------------- ↓
-- ＰＬＡファイルのＩＢＭ形式→Ｗｉｎｄｏｗｓプリント形式変換 -- ↓
---------------------------------------------------------------- ↓
/map ↓
      ank        2       -- Ｎｏ．（惑星番号） ↓
      kanji      8       -- 和名 ↓
      ank        10      -- 英名 ↓
      ank        9       -- 読み ↓
      packdisp   u4.3    -- 質量比 ↓
      zonedisp   u2      -- 衛星数（確定済） ↓
      zonedisp   s2.1    -- 極大等級（みかけ上の最大の明るさ） ↓
      kanji      20      -- 英名の意味・由来 ↓
```

この例のように、オプションの指定が複雑なときは、パラメータファイルを利用してください。内容がわかりやすく、修正もしやすくなります。

■注意事項

同名ファイルは置換する（省略値）

すでに同じ名前のＷｉｎｄｏｗｓファイルがある場合､デフォルトでは自動的に元のファイルを削除し、新たに変換したファイルで置き替えます。この動作は「／ＲＥＰｌａｃｅ」オプションにより制御可能です。詳細は「／ＲＥＰｌａｃｅ」オプションの説明を参照してください。

差し替えメッセージが出たら

ＩＢＭディスクの差し替えをうながす、

順序番号 nn のＩＢＭディスクに差し替えてください.

というメッセージが出たら、変換中のＩＢＭファイルはマルチボリュームになっていて、つぎのＩＢＭディスクにつづくことを意味しています｡メッセージに従ってＩＢＭディスクを差し替えて、ＯＫボタンをクリックしてください。処理を続行します。詳しくは解説編の「マルチボリューム」の節を参照してください。

２．４　ＧｅｔＴｅｘｔ（ｇｔ）　ゲット・テキスト

ＩＢＭ→Ｗｉｎ　テキストファイル変換

ＧｅｔＴｅｘｔ(ｇｔ)コマンドは、文字データだけからなるＩＢＭファイルをＷｉｎｄｏｗｓのテキストファイルに変換するコマンドです。おもに、ソースプログラムの変換に使用します。ＩＢＭファイル側に漢字があるときは、ＫＩ／ＫＯがついていないといけません。ＫＩ／ＫＯなしのファイルを項目別に変換したいときは、ＧｅｔＤａｔａ(ｇｄ)コマンドを使ってください。

■コマンド形式

コマンド	パラメータ
ＧｅｔＴｅｘｔ ｇｔ	［ＩＢＭファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ＩＢＭファイル、Ｗｉｎｄｏｗｓファイル

ＩＢＭファイルとＷｉｎｄｏｗｓファイルの指定方法は、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＧｅｔＴｅｘｔ(ｇｔ)コマンドには、つぎのオプションが指定できます。

ＧｅｔＴｅｘｔ(ｇｔ)コマンド固有のオプション

／Ｃｏｄｅ	ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する
／ＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮する
／ＮｏＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮しない

Ｇｅｔ系コマンド共通のオプション

／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＥＯＦ	ＥＯＦコードをつける
／ＮｏＥＯＦ	ＥＯＦコードをつけない
／ＲＥＰｌａｃｅ	変換先ファイルの置換設定

これらのオプションのうち、／Ｃｏｄｅオプションがとくに重要です。

Ｇｅｔ系コマンド共通のオプションは、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＧｅｔＴｅｘｔ（ｇｔ）コマンド固有のオプション

ＧｅｔＴｅｘｔ(ｇｔ)コマンドに固有のオプションを説明します。

／Ｃｏｄｅ ---- ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する

／Ｃｏｄｅ	Ａｎｋ Ｋａｎｊｉｍｉｘ
／Ｃｏｄｅ	Ａｎｋ

コード変換の方法を指定します。

Ａｎｋ指定

Ａｎｋ指定すると、すべてＡＮＫデータとして変換します。これが省略値です。以下に示す、

ＩＢＭ側		Ｗｉｎｄｏｗｓ側
ＪＩＳ８／ＡＳＣＩＩ		
ＥＢＣＤＩＣ（カタカナ）	→	ＪＩＳ８／ＡＳＣＩＩ
ＥＢＣＤＩＣ（英小文字）		

の３とおりの変換が可能です。あらかじめ、変換設定のＡＮＫコードの設定で、ＩＢＭファイル側のコード系を設定しておかなければいけません（ふつう、セットアップ時に１回だけ行います)。漢字がまじっているときは、つぎのＫａｎｊｉｍｉｘ指定を使ってください。

Ｋａｎｊｉｍｉｘ指定

ＡＮＫ・漢字混在で、ＫＩ／ＫＯもついているとき、この指定をします。あらかじめ、変換設定の漢字変換方式の設定で、適切な漢字変換方式の割り当てをしておかなければいけません（ふつう、セットアップ時に１回だけ行います)。

／ＴａｂＣｏｍｐｒｅｓｓ -------- タブ圧縮する
／ＮｏＴａｂＣｏｍｐｒｅｓｓ ---- タブ圧縮しない

```
／ＴａｂＣｏｍｐｒｅｓｓ〔タブ間隔〕
                        ８
／ＮｏＴａｂＣｏｍｐｒｅｓｓ
```

タブ圧縮の有無と、タブ圧縮するときのタブ間隔を指定します。タブ圧縮とは、タブ位置の直前までつづく２個以上の連続スペースを、ＴＡＢ（０９Ｈ）に置き替えることです。

／ＴａｂＣｏｍｐｒｅｓｓオプションを指定するとタブ圧縮します。タブ間隔は、２～２５５の範囲で指定し、それがレコードのおわりまで繰り返し適用されます。タブ間隔を省略すると標準のタブ間隔（８桁きざみ）になります。

タブ圧縮は、ソースプログラムなど空白部分が多いファイルの、変換後のファイル容量を減らすのに効果的です。

なお、文字列定数中のスペースまでＴＡＢに変換してしまうことを避けるため、アポストロフィ（’）か引用符（”）が見つかると、その行についてはそこでタブ圧縮を打ち切ります。

／ＮｏＴａｂＣｏｍｐｒｅｓｓオプションを指定すると、タブ圧縮はしません。これが省略値です。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｃｏｄｅ　Ａｎｋ	ＡＮＫ変換する
／ＮｏＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮しない
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＮｏＥＯＦ	ＥＯＦコードをつけない
／ＲＥＰｌａｃｅ　ＹＥＳ	変換先ファイルを置き換える

と指定したものとして、テキストファイル変換を行います。行末の空白類の削除（行末圧縮）と改行コード（ＣＲ／ＬＦ＝０Ｄ０ＡＨ）の付加は無条件に行われます。

■使用例

例１）コマンド行方式で実行する

ＩＢＭファイルＵＲＩＡＧＥをコマンド行方式で変換します。漢字はありません。同じファイル名にし、拡張子を．ＴＸＴにします。

```
C:¥>ft gettext a:uriage c:*.txt ↓
```

例２）例１をバッチファイルに組み込む

例１をバッチファイルＧＥＴＵＲＩ．ＢＡＴに組み込みます。

＜ＧＥＴＵＲＩ．ＢＡＴ＞

```
@echo off ↓
:: ↓
:: 売り上げデータのＩＢＭ→Ｗｉｎテキストファイル変換 ↓
:: ↓
start /w ft gettext a:uriage c:*.txt ↓
```

例３）標準の８桁きざみでタブ圧縮する

ＩＢＭファイルＤＡＴＡがあります。中身はＣ言語のソースプログラムです。漢字はありません。それを、Ｗｉｎｄｏｗｓファイル ＨＥＬＬＯ．Ｃに変換します。このとき、標準の８桁きざみでタブ圧縮も行います。

```
C:¥>ft gettext a:data c:hello.c /tc ↓          (/TabCompress)
```

例４）４桁きざみでタブ圧縮する

例３と同様ですが、タブ間隔を４桁きざみにしてタブ圧縮しながら変換します。

```
C:¥>ft gettext a:data c:hello.c /tc4 ↓         (/TabCompress 4)
```

例５）ＡＮＫ・漢字まじりのソースプログラムを変換する

ドライブＡ：のＩＢＭファイルＭＬＩＳＴ００２はＣＯＢＯＬのソースプログラムで、漢字を使っています。漢字の前後にはＫＩ／ＫＯがついています。それを、ドライブＣ：のＷｉｎｄｏｗｓファイルＭＬＩＳＴ００２．ＣＢＬに変換します。

```
C:¥>ft gettext a:mlist002 c:*.cbl /ck ↓          (/Code KanjiMix)
```

漢字があるので、／ＣＫ指定を忘れずにつけてください。また、あらかじめ適切な漢字変換方式を割り当てておくことも忘れないでください。

例６）ＡＮＫ・漢字まじりのソースプログラムを何本もまとめて変換する

ドライブＡ：には、ＵＭＡＳｘｘｘｘというパターンの名前のＩＢＭファイルが何本も入っています。ＣＯＢＯＬのソースプログラムで、漢字も使っています。それらをまとめて、ドライブＣ：のＵＭＡＳｘｘｘｘ．ＣＢＬという名前のＷｉｎｄｏｗｓファイルに変換します。

```
C:¥>ft gettext a:umas* c:*.cbl /ck ↓          (/Code KanjiMix)
```

■注意事項

漢字があるときは漢字変換方式の割り当てと／ＣＫを忘れずに

漢字が入っているときは、あらかじめ

適当な漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、変換のとき、

／Ｃｏｄｅオプションにｋａｎｊｉｍｉｘ指定（／ＣＫ指定）するのを忘れがち

なので注意してください。

その他の注意事項

「Ｇｅｔ系コマンド」の節を参照してください。

２．５　ＧｅｔＤａｔａ（ｇｄ）　ゲット・データ

ＩＢＭ→Ｗｉｎ　データファイル変換

ＧｅｔＤａｔａ(ｇｄ)コマンドは、ＩＢＭ形式のデータファイル（おもにＣＯＢＯＬデータ）をＷｉｎｄｏｗｓのデータファイル（テキスト形式）に変換するコマンドです。項目別に分けて変換できます。Ｗｉｎｄｏｗｓファイルは常に改行コード（ＣＲ／ＬＦ＝０Ｄ０ＡＨ）つきのテキストファイルになります。プリント形式とデリミタ形式の、２つの変換方法があります。

■コマンド形式

コマンド	パラメータ
ＧｅｔＤａｔａ ｇｄ	［ＩＢＭファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ＩＢＭファイル、Ｗｉｎｄｏｗｓファイル

ＩＢＭファイルとＷｉｎｄｏｗｓファイルの指定方法は、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＧｅｔＤａｔａ(ｇｄ)コマンドには、つぎのオプションが指定できます。

ＧｅｔＤａｔａ(ｇｄ)コマンド固有のオプション

／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ	引用符くくりの詳細を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

Ｇｅｔ系コマンド共通のオプション

／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＥＯＦ	ＥＯＦコードをつける
／ＮｏＥＯＦ	ＥＯＦコードをつけない
／ＲＥＰｌａｃｅ	変換先ファイルの置換設定

このうち、／Ｄｅｌｉｍｉｔｅｄオプション、／ＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＭＡＰオプションがとくに重要です。

Ｇｅｔ系コマンド共通のオプションは、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＧｅｔＤａｔａ（ｇｄ）コマンド固有のオプション

ＧｅｔＤａｔａ（ｇｄ）コマンドに固有のオプションを説明します。

／Ｄｅｌｉｍｉｔｅｄ ---------- デリミタ形式に変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ ---- プリント形式に変換する

```
                  ┌ By  COMma ┐
／Ｄｅｌｉｍｉｔｅｄ │ By  TAB   │
                  └ By  SPace ┘
                    ┌ Compress   ┐
                    └ NoCompress ┘
／ＮｏｎＤｅｌｉｍｉｔｅｄ
```

データとしてのテキストファイルにはいくつもの形式があるので、どの形式にするかを指定します。

／ＮｏｎＤｅｌｉｍｉｔｅｄオプションを指定すると、デリミタ（区切り文字）なしで変換され、プリント形式（固定長のテキストファイル〔ＳＤＦ形式〕）になります。これが省略値です。

◆注意　----　プリント形式のときは、／ＭＡＰでコンマを入れないこと

／ＮｏｎＤｅｌｉｍｉｔｅｄオプションを指定しプリント形式に変換するときは、／ＭＡＰオプションでデリミタ挿入（＝コンマ〔,〕）を使ってはいけません。

／Ｄｅｌｉｍｉｔｅｄオプションを指定すると、デリミタ形式（区切り文字つき）のテキストファイルに変換されます。デリミタの種類によってさらに細かい形式が決まります。デリミタは、Ｂｙ指定で

Ｂｙ　ＣＯＭｍａ	コンマ区切り形式（ＣＳＶ形式、Ｋ３形式）
Ｂｙ　ＴＡＢ	タブ区切り形式（ＴＡＢ＝０９Ｈ）
Ｂｙ　ＳＰａｃｅ	スペース区切り形式（ＳＰ＝２０Ｈ）

の３つのなかから指定できます。／ＭＡＰオプションでデリミタ挿入（＝コンマ〔,〕）を指定したところに、この／Ｄｅｌｉｍｉｔｅｄオプションで指定したデリミタが入ります。省略値はＢｙ　ＣＯＭｍａ、コンマ区切り（ＣＳＶ）形式への変換です。

さらに、変換後に圧縮をかけるか否かを

Ｃｏｍｐｒｅｓｓ　　圧縮をかける（可変長になる。省略値）
ＮｏＣｏｍｐｒｅｓｓ　　圧縮しない（固定長のまま）

で指定できます。圧縮をかけると、本来不要なスペース、つまり

レコードの先頭・末尾のスペース
デリミタの前後のスペース
引用符の前のスペース

が削除されます。圧縮をかける機能があるのは、

変換後のＷｉｎｄｏｗｓファイルの容量を、大幅に減らすことができる
一部の市販ソフトでは、不要なスペースがあるとうまくデータを読み込めない

のような理由があるためです。

◆注意　----　デリミタ形式のときは、／ＭＡＰで項目ごとにコンマを入れること

／Ｄｅｌｉｍｉｔｅｄオプションを指定しデリミタ形式に変換するときは、／ＭＡＰオプションで項目ごとにデリミタ挿入（＝コンマ〔,〕）を入れてください。そこにＢｙで指定したデリミタが入ります。

／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ　引用符くくりの詳細を指定する

```
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ

    ( ＱｕｏｔａｔｉｏｎＭａｒｋ     Ｎｏｎｅ
      ＱｕｏｔａｔｉｏｎＭａｒｋ     Ｓｉｎｇｌｅ
      ＱｕｏｔａｔｉｏｎＭａｒｋ     Ｄｏｕｂｌｅ )

    ( ＡｓＤａｔａ                 Ｅｒａｓｅ
      ＡｓＤａｔａ                 ＴｏＳｐａｃｅ
      ＡｓＤａｔａ                 Ｄｕｐｌｉｃａｔｅ )

    ( Ｋｅｅｐ                     Ｗｉｄｔｈ
      Ｋｅｅｐ                     Ｄａｔａ )
```

変換するＩＢＭファイルがデリミタ形式で、引用符でくくりたい項目があるときに、その引用符によるくくり方を指定します。このオプションは、／Ｄｅｌｉｍｉｔｅｄオプションが指定されているときにのみ指定できます。

／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌオプションではＷｉｎｄｏｗｓファイルの引用符の種類をＱｕｏｔａｔｉｏｎＭａｒｋで指定します。

ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｎｏｎｅ	なし
ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｓｉｎｇｌｅ	単一引用符（’）
ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｄｏｕｂｌｅ	２重引用符（”）

ＱｕｏｔａｔｉｏｎＭａｒｋ　Ｎｏｎｅを指定した場合は、Ｍａｐ文で引用符くくりの指定があっても無視して、引用符をつけずに変換内容を出力します。省略値はＱｕｏｔａｔｉｏｎＭａｒｋ　Ｄｏｕｂｌｅです。

◆注意　----　Ｍａｐ文の引用符記号は、常に二重引用符「”」

ＱｕｏｔａｔｉｏｎＭａｒｋ　Ｓｉｎｇｌｅ指定時も、Ｍａｐ文において引用符くくりに使う記号は、二重引用符「”（半角）」ですので注意してください。

次に、データ内容として引用符が存在した場合の扱いの指定について説明します。

ＡｓＤａｔａ	Ｅｒａｓｅ	削除
ＡｓＤａｔａ	ＴｏＳｐａｃｅ	空白化
ＡｓＤａｔａ	Ｄｕｐｌｉｃａｔｅ	引用符を２連化

ＡｓＤａｔａ　Ｅｒａｓｅを指定した場合、変換項目内にデータ内容として引用符が存在した時はその引用符を削除します。ＡｓＤａｔａ　ＴｏＳｐａｃｅを指定した場合、変換項目内にデータ内容として引用符が存在した時はその引用符を半角スペースに変換します。ＡｓＤａｔａ　Ｄｕｐｌｉｃａｔｅを指定した場合、変換項目内にデータ内容として引用符が存在した時は、その引用符を２つ並べて出力します。その際、データ項目長が変わりますが、それについてはＫｅｅｐで指定します。

◆注意　----　「ＡｓＤａｔａ」のデータ内引用符は「ＱｕｏｔａｔｉｏｎＭａｒｋ」に依存

「ＡｓＤａｔａ」ではデータ内容として引用符が存在した時の扱いを指定しますが、その「引用符」というのは「ＱｕｏｔａｔｉｏｎＭａｒｋ」で指定したものを指します。

「ＱｕｏｔａｔｉｏｎＭａｒｋ」で二重引用符を変換処理で使用する引用符として指定した場合は、一重引用符がデータ中にあっても空白化などの処理の対象になりません。

次に、データ内容として引用符が存在した場合の扱いの指定について説明します。

Ｋｅｅｐ	Ｗｉｄｔｈ	出力幅優先
Ｋｅｅｐ	Ｄａｔａ	データ優先

ＫｅｅｐはＡｓＤａｔａ　Ｄｕｐｌｉｃａｔｅを指定していない場合には意味を持ちません。

「Ｗｉｄｔｈ」を指定すると引用符の２連化によって入力データ長に対する変換後データ長が膨れ上がった場合には出力幅を優先して、データの後方部分を切り捨てます。

「Ｄａｔａ」を指定すると引用符の２連化によって入力データ長に対する変換後データ長が膨れ上がった場合には、変換後の長さそのままを出力します。

◆注意　----　「Ｋｅｅｐ」にてデータ優先をした時の出力幅について

「Ｋｅｅｐ」で「Ｄａｔａ」を指定すると、Ｍａｐ文で出力幅指定を省略した時だけでなく出力幅指定をしていたとしてもそれを無視して変換します。下は変換処理時の実際の出力幅です。

出力幅指定をした場合	…	Ｍａｐ文指定の出力幅＋引用符の長さ
出力幅指定を省略した場合	…	Ｍａｐ文指定の入力幅＋引用符の長さ

「／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」コマンドオプションを省略した場合は

/QuotingContlol QuotationMark Double AsData Duplicate Keep Data

という指定をした扱いになります。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／MAP	Ａｎｋ変換 Ｋａｎｊｉ変換 ＡｎｋｉＺｅ変換 ＫａｎｊｉＺｅ変換 ＫａｎｊｉＭｉｘ変換 ＵｓｅｒＡ／Ｂ変換 Ｎｕｍｅｒｉｃ変換 Ｂｉｎａｒｙ変換 ＺｏｎｅＤｉｓｐ変換（Ｚｏｎｅ変換） ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換） ＤｉｓｐＺｏｎｅ変換 ＺｏｎｅＺｏｎｅ変換 ＰａｃｋＺｏｎｅ変換 ＢｉｎＤｉｓｐ変換 ＢｉｎＺｏｎｅ変換 ＴｏＤｉｓｐ変換 ＴｏＺｏｎｅ変換 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 デリミタ挿入（コンマ）／引用符くくり 改行コード挿入／改行コード挿入２ 桁移動／入力スキップ／出力スキップ	・・・

／ＭＡＰ　　Ａｎｋ

この／ＭＡＰオプションで細かい変換方法を指示します。本来なら、自動的に項目を認識して変換ができると便利です。しかし、ホストの、とくにＣＯＢＯＬのデータには、**データ自身に桁数や小数点位置の判断に必要な情報が含まれていない**、という特性があります。そのため自動変換は原理的に不可能なのです。

◆注意　----　マルチレコードレイアウト等の指定について

／ＭＡＰオプションには、マルチレコードレイアウト等の指定ができるＡｔｌａｓ構文を記述することもできます。詳細は、マルチレコード編のマニュアルを参照してください。

Ａｎｋ変換

Ａｎｋ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ項目を変換します。どのコード変換が行われるかは、変換設定のＡＮＫコードの設定で決まります（ふつう、セットアップ時に一回だけ行います)。

入力幅は、　　ａ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＡｎｋ変換するときに ---- ／ｍａｐ　ａ
最終項目をＡｎｋ変換するときには ------ ／ｍａｐ　・・・　ａ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ａｎｋ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡＮＫ形式、最大２５６文字で、半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ａ［‘ＡＮＫ’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａ２０　　という指定は、
ａ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ項目のおわりにスペース（２０Ｈ）が詰められます。

ＡｎｋｉＺｅ（ＡＮＫ化）変換

ＡｎｋｉＺｅ　［入力幅｜＊］［：出力幅］

ＩＢＭ側（入力）文字列をＡＮＫ（半角）に変換します。入力側文字列は漢字（全角）文字列であると見なして変換します。

入力幅は、　ａｚ２０　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　ホスト側（入力）のレコード長
＊　ホスト側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＡｎｋｉＺｅ変換するときには ---- ／ｍａｐ　ａｚ
最終項目をＡｎｋｉＺｅ変換するときには -------- ／ｍａｐ　・・・　ａｚ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａｚ２０　　という指定は、
ａｚ２０：２０　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ項目のおわりに空白が詰められます。

Ｋａｎｊｉ変換

Ｋａｎｊｉ［入力幅｜＊｜定数］［：出力幅］

漢字項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に一回だけ行います)。

入力幅は、　　ｋ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません)。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉ変換するときには ---- ／ｍａｐ　ｋ
最終項目をＫａｎｊｉ変換するときには -------- ／ｍａｐ　・・・　ｋ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定した漢字形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ｋａｎｊｉ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列は漢字形式、最大２５６文字で、半角(')でくくる
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋ［'漢字'］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋ１６　　という指定は、
ｋ１６：１６　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を大きくしなければならない場合があります。拡張漢字のクエスチョン変換を使っていて、オーバーフローの危険があるときなどです（最大で入力幅の３倍)。そのときは、出力幅を指定します。

項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側の漢字項目のおわりに漢字変換方式で設定されている漢字スペース（２０２０Ｈ／８１４０Ｈ）が詰められます。

ＫａｎｊｉＺｅ（漢字化）変換

ＫａｎｊｉＺｅ　［入力幅｜＊］［：出力幅］

ＩＢＭ側（入力）文字列をできる限り漢字（全角）に変換します。入力側文字列はＡＮＫ文字列であると見なして変換します。

入力幅は、　　ｋｚ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません)。
式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉＺｅ変換するときには ---- ／ｍａｐ　ｋｚ
最終項目をＫａｎｊｉＺｅ変換するときには -------- ／ｍａｐ　・・・　ｋｚ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると入力幅の２倍が指定されます。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を大きくしなければならない場合があります。拡張漢字のクエスチョン変換を使っていて、オーバーフローの危険があるときなどです（最大で入力幅の３倍)。そんなときは、出力幅を指定します。
項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側の漢字項目のおわりに漢字変換方式で設定されている漢字スペースが詰められます。

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ・漢字まじり項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に一回だけ行います)。漢字の前後にＫＩ／ＫＯがついていないと、この変換方法は適用できません。変換後にＫＩ／ＫＯが取れて、その分、左詰めになります。

入力幅は、　　ｋｍ３０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉＭｉｘ変換するときには ---- ／ｍａｐ　ｋｍ
最終項目をＫａｎｊｉＭｉｘ変換するときには -------- ／ｍａｐ　・・・　ｋｍ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ・漢字まじり形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

ＫａｎｊｉＭｉｘ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡｎｋ・漢字まじり形式、最大２５６文字
半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋｍ［‘ＡＮＫ・漢字まじり’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋｍ３２　　という指定は、
ｋｍ３２：３２　　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を変更したい場合があります。ＫＩ／ＫＯが取れて短くなることや、逆に、拡張漢字のクエスチョン変換を使っていてオーバーフローの危険があるときなどです（最大で、入力幅の３倍－ＫＩ／ＫＯのバイト数の合計)。そのときは、出力幅を指定します。

項目長を縮めると、ＡＮＫ・漢字まじり項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ・漢字まじり項目のおわりにスペース（２０Ｈ）が詰められます。

ＵｓｅｒＡ変換
ＵｓｅｒＢ変換

```
ＵｓｅｒＡ［入力幅｜＊］［：出力幅］
ＵｓｅｒＢ［入力幅｜＊］［：出力幅］
```

ＵｓｅｒＡ／Ｂ変換は、利用者独自のバイト単位の変換処理が必要なときに、ＡＮＫ変換表Ｕｓｅｒ－Ａ、Ｕｓｅｒ－Ｂを書き替えて利用します。ＵｓｅｒＡ／Ｂ変換には、Ａｎｋ変換の説明がほとんどそのまま当てはまります。

入力幅は、　　ｕａ１０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＵｓｅｒＡ変換するときには ---- ／ｍａｐ　ｕａ
最終項目をＵｓｅｒＡ変換するときには -------- ／ｍａｐ　・・・　ｕａ

のようにして、入力幅を省略できます。ＵｓｅｒＢ変換でも同様です。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｕａ１０　　という指定は、
ｕａ１０：１０　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ユーザーＡ／Ｂ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のユーザーＡ／Ｂ項目のおわりにスペース（２０Ｈ）が詰められます。

Ｎｕｍｅｒｉｃ変換

Ｎｕｍｅｒｉｃ［入力幅｜１５］［：出力幅］

文字形式の数値項目どうしの変換をします。

Ｎｕｍｅｒｉｃ変換は、Ａｎｋ変換と後述のＺｏｎｅＤｉｓｐ変換の中間的なものです。Ａｎｋ変換と比較すると、

文字形式数値しか通さない
入力幅の省略値が１５桁（バイト）である
右詰めになる

などの点が異なります。

「文字形式数値しか通さない」というのは、具体的には、

＋、－、０～９、ピリオド（.）、Ｅ、ｅ、Ｄ、ｄ

しか変換しないで、これら以外の文字は捨ててしまうということです。たとえば、通貨記号（￥／＄）や位取りのコンマ（,）などは削除されるので、リストファイルから入力データファイルを作るときなどに利用できます。

Ｂｉｎａｒｙ変換

Ｂｉｎａｒｙ　［入力幅｜＊｜定数］［：出力幅］

Ｂｉｎａｒｙ変換は、「コード変換を一切しない」という変換方法です。通常、ＩＢＭ→Ｗｉｎデータファイル変換では使用しません。

入力幅は、　ｂ２０　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　ＩＢＭ側（入力）のレコード長
＊　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＢｉｎａｒｙ変換するときには ---- ／ｍａｐ　ｂ
最終項目をＢｉｎａｒｙ変換するときには -------- ／ｍａｐ　・・・　ｂ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＢｉｎａｒｙ形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ｂｉｎａｒｙ変換では、つぎの定数が使えます。

１６進列定数 ------- ｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト
汎用定数 ---------- ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　ｂ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｂ２０　　という指定は、
ｂ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を縮めると、バイナリ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のバイナリ項目のおわりにスペース（２０Ｈ）が詰められます。

ＺｏｎｅＤｉｓｐ変換（Ｚｏｎｅ変換）

```
ＺｏｎｅＤｉｓｐ　ピクチャ［：出力幅］
（Ｚｏｎｅ　ピクチャ［：出力幅］）
```

ホストのＣＯＢＯＬのゾーン形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　　と指定します。

なお、ピクチャは省略できません。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０とする
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＺｏｎｅＤｉｓｐ　ｓ４．１　　という指定は、
ＺｏｎｅＤｉｓｐ　ｓ４．１：７　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換）

```
ＰａｃｋＤｉｓｐ　ピクチャ［：出力幅］
（Ｐａｃｋ　ピクチャ［：出力幅］）
```

ホストのＣＯＢＯＬのパック形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

パック形式では、整数部桁数＋小数部桁数を奇数にしておくのが通例です。整数部の最上位桁に意味があるのかないのかは、半々の割合です。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０とする
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＰａｃｋＤｉｓｐ　ｓ４．１　という指定は、
ＰａｃｋＤｉｓｐ　ｓ４．１：７　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換）〔ＢＣＤ形式〕

ＰａｃｋＤｉｓｐ　ピクチャ（ｂ指定）[：出力幅]
（Ｐａｃｋ　ピクチャ（ｂ指定）[：出力幅]）

ＰＯＳ端末等で使われているＢＣＤ形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
１２３．４　という数字が３バイトのＢＣＤ形式の項目に記録されているとすれば、ピクチャは

ｂ５．１　　と必ずｂ指定をします。

なお、ピクチャは省略できません。

ＢＣＤ形式では、整数部桁数＋小数部桁数を偶数にしておくのが通例です。整数部の最上位桁に意味があるのかないのかは、半々の割合です。

出力幅は省略できます。出力幅を省略すると、

整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＰａｃｋＤｉｓｐ　ｂ５．１　　という指定は、
ＰａｃｋＤｉｓｐ　ｂ５．１：７　　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、上位桁が切り捨てられるので、注意してください。

ＤｉｓｐＺｏｎｅ変換

ＤｉｓｐＺｏｎｅ　［入力幅｜１５］：ピクチャ

ホストの文字形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的に桁数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの符号つきゾーン形式の項目に記録するとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

例を示すと、

ＤｉｓｐＺｏｎｅ　８：ｓ４．１　のような指定になります。

ＺｏｎｅＺｏｎｅ変換

ＺｏｎｅＺｏｎｅ　入力ピクチャ［：出力ピクチャ］
［入力ピクチャ］：出力ピクチャ

ホストのＣＯＢＯＬのゾーン形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　：ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＰａｃｋＺｏｎｅ変換

```
ＰａｃｋＺｏｎｅ　入力ピクチャ［：出力ピクチャ］
                  ［入力ピクチャ］：出力ピクチャ
```

ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

１２３．４　という数字が３バイトのＢＣＤ形式の項目に記録されているとすれば、入力ピクチャは

ｂ５．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＺｏｎｅ　：ｓ４．１　という指定は、
ＰａｃｋＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＺｏｎｅ　ｓ４．１　という指定は、
ＰａｃｋＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＢｉｎＤｉｓｐ変換

ＢｉｎＤｉｓｐ　２進キャスト［ピクチャ］:［出力幅］

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓの文字形式数値項目に変換します。変換結果は右詰めになります。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

出力幅は省略できます。出力幅を省略すると、ピクチャに従って出力されます。
出力幅、ピクチャ共に省略すると、

入力のバイト数	１	２	３	４	５	６	７	８
出 力 幅	３	５	８	１０	１３	１５	１７	１８

の要領で２進キャストから自動的に計算された値が使われます。例を示すと、

ＢｉｎＤｉｓｐ　ｉ４ｓ　　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ：１０　　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＢｉｎＺｏｎｅ変換

```
ＢｉｎＺｏｎｅ　２進キャスト　ピクチャ：[出力ピクチャ]
　　２進キャスト［ピクチャ］：出力ピクチャ
```

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

　　ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの項目に記録するとすれば、ピクチャは

　　ｓ４．１　　と指定します。

２進キャストにつづくピクチャも出力ピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

```
ＢｉｎＺｏｎｅ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＺｏｎｅ　ｉ４ｓ４．１：ｓ４．１　　という指定と同じです。
```

ＴｏＤｉｓｐ変換

ＴｏＤｉｓｐ｛数値定数｜システム変数｝[：出力幅]

入力した数値定数、またはシステム変数の値を、文字形式数値項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。変換結果は右詰めになります。

入力値として、つぎの数値定数、システム変数が使えます。省略はできません。

数値定数	**整数定数、小数定数、１６進定数**
システム変数	SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　[－１２３．４]　のように、[　]でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

出力幅は省略できます。出力幅を省略すると、入力した値を正確に出力するために必要な、最小限の幅で出力されます。たとえば、

ＴｏＤｉｓｐ　[－１２３．４]　という指定は、
ＴｏＤｉｓｐ　[－１２３．４]：６　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＴｏＺｏｎｅ変換

ＴｏＺｏｎｅ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。

入力値として、つぎの数値定数、システム変数が使えます。省略はできません。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

出力ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、５バイトの符号つきゾーン形式の項目に記録するとすれば、出力ピクチャは　ｓ４．１　と指定します。
入力値、出力ピクチャともに省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、５バイトの符号つきゾーン形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＺｏｎｅ　　［－１２３．４］：ｓ４．１　　のような指定になります。

Ｙｅａｒ設定（年設定）

```
Ｙｅａｒ　　Ｗｎｎ｜Ｓｎｎ
                          ：Ｗｎｎ｜：Ｓｎｎ
              Ｗｎｎ｜Ｓｎｎ　：Ｗｎｎ｜：Ｓｎｎ
```

日付データ項目を変換する際の、年の２桁（ｙｙ）と４桁（ｙｙｙｙ）の変換方式を設定します。Ｗｎｎの“W”はウインドウ方式を、Ｓｎｎの“Ｓ”はシフト方式を意味し、“ｎｎ”は００～９９の数字で指定します。“：”の左側の指定は入力側、“：”の右側の指定は出力側への適用になり、入力側のみ、出力側のみ、入出力両方の３通りの指定ができます。たとえば、

Ｙｅａｒ	Ｗ３０	入力データの年を１９３０～２０２９年とみなす
Ｙｅａｒ	：Ｓ２５	出力データの年下２桁を、－２５する
Ｙｅａｒ	Ｗ３０：Ｓ２５	入力データの年を１９３０～２０２９年とみなし、 出力データの年下２桁を、－２５する

のように指定します。

また、シフト方式（“Ｓｎｎ”指定）では、つぎの特殊指定ができます。

Ｙｅａｒ	ＳＳｈｏｗａ	“Ｓ２５”の指定と同じ（昭和通年方式）
Ｙｅａｒ	ＳＨｅｉｓｅｉ	“Ｓ８８”の指定と同じ（平成通年方式）

Ｙｅａｒ設定はＤａｔｅ変換が実行された時に適用になり、複数のＹｅａｒ設定がなされている場合は、Ｄａｔｅ変換の直前のＹｅａｒ設定が有効になります。

Ｙｅａｒ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

Ｙｅａｒ	Ｗ３０：Ｗ３０	入出力データの年を１９３０～２０２９年とみなす

ＤａｔｅＤｅｌｉｍ設定（日付区切り設定）

```
ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ｜ＨＹＰＨＥＮ｜ＰＥＲＩＯＤ
```

日付データ項目を出力する際の日付区切り記号をつぎの３つの中から設定します。

指定文字	日付区切り記号	データ例
ＳＬＡＳＨ	／（スラッシュ）	１９９８／１２／３１
ＨＹＰＨＥＮ	－（ハイフン）	１９９８－１２－３１
ＰＥＲＩＯＤ	.（ピリオド）	１９９８．１２．３１

ＤａｔｅＤｅｌｉｍ設定はＤａｔｅ変換が実行された時に適用になり、複数のＤａｔｅＤｅｌｉｍ設定がなされている場合は、Ｄａｔｅ変換の直前のＤａｔｅＤｅｌｉｍ設定が有効になります。

ＤａｔｅＤｅｌｉｍ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ　日付区切り記号を“／”にする

Ｄａｔｅ変換

```
Ｄａｔｅ　入力日付マスク：出力日付マスク
```

日付データ項目を変換します。

入力日付マスク、出力日付マスクは必ず指定します。省略はできません。たとえば、

Ｄａｔｅ　ｙｙｍｍｄｄ：ｙｙｙｙ－ｍｍ－ｄｄ

のように指定すると、コード変換後に、入力側６バイトの日付データ項目を、出力側１０バイトの日付データ項目に編集します。その際に、Ｙｅａｒ設定、ＤａｔｅＤｅｌｉｍ設定が適用になります。

デリミタ挿入（コンマ）

,

デリミタ形式に変換するとき、デリミタ（区切り文字）を挿入する位置を指定します。ＧｅｔＤａｔａ(ｇｄ)コマンドで指定できるデリミタには、

コンマ
タブ
スペース

の３つがあります。その指定の方法は、／Ｄｅｌｉｍｉｔｅｄオプションの説明を参照してください。なお、プリント形式への変換のときは、このデリミタ挿入の機能は無効になります。

例をあげます。たとえば、ゾーン形式の数値項目と、パック形式の数値項目をコンマで区切るには、つぎのような指定になります。

／Ｄｅｌｉ／ＭＡＰ　・・・，ＺＤＵ５，ＰＤＳ３．２，・・・

引用符くくり

“～”

デリミタ形式に変換するときは、変換後の文字列を引用符でくくることができます。

市販ソフトの入力用には、文字項目だけを引用符でくくることが多いのですが、引用符を使わないソフトもあれば、すべての項目を引用符でくくるソフトもあります。ここでは文字項目だけを引用符でくくる例を示すと、つぎのような指定になります。

／Ｄｅｌｉ／ＭＡＰ　・・・　ＺＤＵ１０，“Ｋ２０”，ＰＤＵ３　・・・

「ＧｅｔＤａｔａ」コマンドの「引用符くくり」機能において使える引用符としては、二重引用符「”（全角）」と一重引用符「’（半角）」があります。「ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」オプションにて、「ＱｕｏｔａｔｉｏｎＭａｒｋ」を「Ｓｉｎｇｌｅ」または「Ｄｏｕｂｌｅ」にすることで指定します。

◆注意　----　引用符の種類の指定は一回のコマンド処理全体に対して一回のみ

引用符くくりで使用する引用符の種類は、Ｍａｐ文の中で処理に応じて切り替えて指定することは出来ません。

◆注意　----　Ｍａｐ文の引用符記号は、常に二重引用符「”」

「／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」コマンドオプションにて、「ＱｕｏｔａｔｉｏｎＭａｒｋ」を「Ｓｉｎｇｌｅ」に指定した時も、Ｍａｐ文において引用符くくりに使う記号は、二重引用符「”（半角）」ですので注意してください。

◆注意　----　Ｅｃｘｅｌで読み込むためのデータ作成のために変換を行う場合

Ｅｘｃｅｌで取り込むためのデータファイルを変換・作成する際は、「／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」のコマンドオプション指定については、あえて行うことなくデフォルトのままで変換してください。また、作成したデータファイルをＥｘｃｅｌで開く際には、必ずＥｘｃｅｌの「ファイル」メニューの「開く」から行ってください。ダブルクリックで開くと、データ型の指定が出来ないため、数字の先頭の０などが切り捨てられるなどの問題が生じます。

●参考...

たとえば、デリミタ形式のうちＫ３フォーマットと呼ばれるものは、

各項目はコンマで区切る
数値はそのまま（位取りのコンマ不可）
文字列は引用符でくくる　　というルールになっています。

改行コード挿入

```
！Ｒ
```

デリミタ形式に変換するときは、任意のところに改行コードを挿入する（改行コードで項目を区切る）こともできます。ただし、プリント形式への変換のときは、この改行コード挿入の機能は無効になります。

なお、１レコード変換がおわったときに、自動的にレコード末尾に改行コードが付加される機能とは別のものですから、混同しないでください。

例を示します。１項目ずつ改行コードで区切っていくなら、つぎのような指定になります。

```
／Ｄｅｌｉ／ＭＡＰ　Ａ１５！Ｒ　Ａ２０！Ｒ　Ｋ３２！Ｒ　・・・　Ａ
```

改行コード挿入２

```
！Ｎ
```

Ｗｉｎｄｏｗｓ側（出力）レコードに改行コード（ＣＲ／ＬＦ＝０Ｄ０ＡＨ）を挿入します。改行コード挿入との違いは、プリント形式に変換するときに改行コードを挿入できることです。

桁移動（プリント形式）

＠入力桁位置
＠：出力桁位置

変換対象にするＩＢＭ側（入力）の桁位置や、変換結果を書き込むＷｉｎｄｏｗｓ側（出力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

入力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

＄　　ＩＢＭ側（入力）のレコード長
．　　ＩＢＭ側（入力）の現在の桁位置

という特殊変数が使えます。たとえば、

＠．－４０　Ｋａｎｊｉ２０　＾２０　　と指定すれば、

今の入力桁位置から４０バイト戻り、漢字２０バイト変換せよ
そして、元の位置に戻れ

という意味になります。また、

＠＄－８　ＺｏｎｅＤｉｓｐＵ８　　と指定すれば、

ＩＢＭ側（入力）レコード末尾の８バイト前に位置づけ、
８バイトＺｏｎｅＤｉｓｐ変換せよ

という意味になります。

出力桁位置を移動することもできます。ふつう、１０進数で桁位置を指定します。式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

．　　Ｗｉｎｄｏｗｓ側（出力）の現在の桁位置

◆注意　----　先頭を０桁目とする

Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

桁移動（デリミタ形式）

＠入力桁位置

変換対象にするＩＢＭ側（入力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

デリミタ形式への変換の場合、入力桁位置の移動だけが有効です。出力桁位置の指定はできません。

入力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

＄　　ＩＢＭ側（入力）のレコード長
．　　ＩＢＭ側（入力）の現在の桁位置

という特殊変数が使えます。たとえば、

＠４０　ＺｏｎｅＤｉｓｐＵ１０，＠２　ＺｏｎｅＤｉｓｐＵ８，＠０　“Ａｎｋ２”

と指定すれば、

４０桁目から１０バイトＺｏｎｅＤｉｓｐ変換し、デリミタで区切れ
２桁目から８バイトＺｏｎｅＤｉｓｐ変換し、デリミタで区切れ
先頭の２バイトをＡｎｋ変換し、引用符でくくれ

という意味になります。

◆注意　----　先頭を０桁目とする

Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

入力スキップ

```
^［n｜1］
```

ＩＢＭ側（入力）レコードに不要な項目があるとき、それをスキップして変換できます。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

^３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

^３　　は　　^^^　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（入力）の残りバイト数

これを使えば、　　／ＭＡＰ　・・・　^＊　　として「残りは全部スキップせよ」という指定もできます。しかし、何の指定もなかった分は変換対象にならないので、この ^ ＊は冗長です。省いたほうがよいでしょう。

出力スキップ

```
_［n｜1］
```

入力スキップとは逆に、Ｗｉｎｄｏｗｓ側（出力）に何桁か空きを作ることもできます。プリント形式への変換のとき、空項目を作るのがおもな用途です。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

_３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

_３　　は　　_ _ _　　と指定したのと同じです。

／ＰＨａｓｅ　---- ヘッダ・トレーラの生成・付加のタイミングを指定する

```
／ＰＨａｓｅ　　［Ｏｐｅｎ＜ｎ＞］,
　　［Ｍａｉｎ＜ｎ＞］,
　　［Ｃｌｏｓｅ＜ｎ＞］
```

データを変換する際にヘッダやトレーラを付加したいときに、この／ＰＨａｓｅオプションで各フェーズの呼び出しが起きる回数を指定します。フェーズにはつぎの３種類があります。

Ｏｐｅｎフェーズ	ヘッダの生成・付加を行う
Ｍａｉｎフェーズ	本体の変換をする
Ｃｌｏｓｅフェーズ	トレーラやエンドレコードの生成・付加を行う

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数を、＜ｎ＞に１０進数値で指定します。Ｍａｉｎフェーズでは、

＄　全レコード分

という指定もできます。
Ｏｐｅｎフェーズ→Ｍａｉｎフェーズ→Ｃｌｏｓｅフェーズの順に呼び出しを行います。

◆注意　----　フェーズごとの指定について

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数のみ、指定します。各フェーズごとの動作は、Ａｔｌａｓ構文を使って設定する必要があります。詳細はマルチレコード編のマニュアルを参照してください。

●オプション省略時の動作

オプションをすべて省略すると、

／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式ファイルにする
／ＭＡＰ　Ａｎｋ	ＡＮＫデータのみとして変換する
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＮｏＥＯＦ	ＥＯＦコードをつけない
／ＰＨａｓｅ　Ｏｐｅｎ０，Ｍａｉｎ＄，Ｃｌｏｓｅ０	ヘッダ・トレーラをつけないで全レコード変換する
／ＲＥＰｌａｃｅ　ＹＥＳ	変換先ファイルを置き換える

と指定したものとしてデータファイル変換を行います。

■解説

●オプションの指定方法

ＩＢＭ形式の、つぎのようなレコードレイアウトのファイルＰＬＡＮＥＴがあるとします。

ＰＬＡＮＥＴのレコードレイアウト

項番	項　　目	桁	幅	データ形式	内容例
1	Ｎｏ．	0	2	ＡＮＫ	1
2	和名	2	8	漢字	水星
3	英名	10	10	ＡＮＫ	MERCURY
4	読み	20	9	ＡＮＫ	ﾏｰｷｭﾘｰ
5	質量比	29	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
6	衛星数（確定済）	33	2	符号なしゾーン形式 整数部２桁	0
7	極大等級	35	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
8	英名の意味・由来	38	20	漢字	口神）神の使者
--	フィラー	58	6	--	--

プリント形式への変換

これをプリント形式に変換するときのオプションの指定は、コマンド行に書くなら、

```
/map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20
```

のようになり、これをパラメータファイルに書くなら、つぎのようになります。

```
/map ↓
      ank        2       -- Ｎｏ．（惑星番号）　↓
      kanji      8       -- 和名　↓
      ank        10      -- 英名　↓
      ank        9       -- 読み　↓
      packdisp   u4.3    -- 質量比　↓
      zonedisp   u2      -- 衛星数（確定済）　↓
      zonedisp   s2.1    -- 極大等数（みかけ上の最大の明るさ）　↓
      kanji      20      -- 英名の意味・由来　↓
```

デリミタ形式への変換

デリミタ形式のうち、コンマ区切り（ＣＳＶ）形式に変換するときのオプションの指定を、コマンド行に書くなら、つぎのようになります。

```
/d /map "a2", "k8", "a10", "a9", pdu4.3, zdu2, zds2.1, "k20"
```

この例では文字項目だけ引用符でくくってみました。／ｄは／Ｄｅｌｉｍｉｔｅｄオプションの短縮指定です。これをパラメータファイルに書くなら、つぎのようになります。

```
/delimited                  -- コンマ区切り（ＣＳＶ）形式に変換 ↓
/map ↓
      "ank        2",       -- Ｎｏ.（惑星番号） ↓
      "kanji      8",       -- 和名 ↓
      "ank        10",      -- 英名 ↓
      "ank        9",       -- 読み ↓
      packdisp    u4.3,     -- 質量比 ↓
      zonedisp    u2,       -- 衛星数（確定済） ↓
      zonedisp    s2.1,     -- 極大等数（みかけ上の最大の明るさ） ↓
      "kanji      20"       -- 英名の意味・由来 ↓
```

タブ区切り形式に変換したいなら、／ＤｅｌｉｍｉｔｅｄオプションでＢｙ　ＴＡＢ指定すればよいので、コマンド行に書くなら、つぎのようになります。

```
/dbtab /map …
```

同様にこれをパラメータファイルに書くなら、つぎのようになります。

```
/delimited by tab    -- デリミタ形式（タブ区切り）に変換 ↓
/map ↓
  ・
  ・
```

スペース区切り形式でも同様で、／Ｄｅｌｉｍｉｔｅｄ　Ｂｙ　ＴＡＢ（／ｄｂｔａｂ）が／Ｄｅｌｉｍｉｔｅｄ　Ｂｙ　ＳＰａｃｅ（／ｄｂｓｐ）になるだけです。

以上が、基本となるスタイルです。また、／Ｄｅｌｉｍｉｔｅｄオプションと／ＭＡＰオプションのデリミタ挿入（＝コンマ〔,〕）は、常にペアで使うこともわかったかと思います。

●変換テストから本番まで

デリミタ形式への変換を例にとり、変換テストの進め方を説明します。変換テストの段階では、変換結果が簡単にわかるように、

コンマ区切り形式にし
不要スペースの圧縮をはずし

ます。つまり、具体的には

```
C:¥>ft getdata a:planet c:test /dnc /map "a2", "k8", "a10", "a9", pdu4.3, zdu2,
    zds2.1, "k20" ↓
```

のようにします（／ｄｎｃは／Ｄｅｌｉｍｉｔｅｄ　ＮｏＣｏｍｐｒｅｓｓの短縮指定）。しかし、これを直接入力するのは大変です。そこで、実際には、１行の指定ですむ場合でもテスト用のバッチファイルを作ります。つまり、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ（その１）＞

```
start /w ft getdata a:planet c:test /dnc /map "a2", "k8", "a10", "a9", pdu4.3,
 zdu2, zds2.1, "k20" ↓
```

のような内容のバッチファイルを作って、テストランと修正を繰り返していきます。正確なレコードレイアウトが手に入らなければ、ＩＢＭディスクエディタ（ＧＵＩ部）でデータ内容を調査する必要があるかもしれません。

「これでＯＫ」となれば、このテスト用バッチファイルを修正して、本番用のバッチファイルにします。簡単な本番用バッチファイルは、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ（その２）＞

```
@echo off ↓
:: 太陽系の惑星データの変換 ↓
::          ＩＢＭ形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式 ↓
start /w ft getdata a:planet c:*.csv /d /map "a2", "k8", "a10", "a9", pdu4.3,
 zdu2, zds2.1, "k20" ↓
```

のような内容になります。

これをバッチファイルと、オプションを記述したパラメータファイルに分けるなら、バッチファイルのほうは、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ（その３）＞

```
@echo off ↓
:: 太陽系の惑星データの変換 ↓
::          ＩＢＭ形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式 ↓
start /w ft getdata a:planet c:*.csv ++plagetcs.p ↓
```

となり、パラメータファイルのほうは、

＜ＰＬＡＧＥＴＣＳ．Ｐ＞

```
-- 太陽系の惑星データの変換（Ｆｏｒ　ＧｅｔＤａｔａ）          -- ↓
--          ＩＢＭ形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式 -- ↓
↓
/delimited                  -- コンマ区切り（ＣＳＶ）形式に変換 ↓
/map ↓
      "ank        2" ,      -- Ｎｏ．（惑星番号） ↓
      "kanji      8" ,      -- 和名 ↓
      "ank        10" ,     -- 英名 ↓
      "ank        9" ,      -- 読み ↓
      packdisp    u4.3 ,    -- 質量比 ↓
      zonedisp    u2 ,      -- 衛星数（確定済） ↓
      zonedisp    s2.1 ,    -- 極大等数（みかけ上の最大の明るさ） ↓
      "kanji      20"       -- 英名の意味・由来 ↓
```

のようになります。

●いろいろな加工・編集

いらない項目をスキップする

ホストからデータを持ってくると、いらない項目がいくつもあることが多いものです。それをスキップして変換するのは簡単で、

```
^n
```

と書きます。ｎはスキップするバイト数です。たとえば、例題のデータから「英名」「読み」「英名の意味・由来」の項目を変換対象からはずすものとします。オプションの指定は、プリント形式への変換なら

```
/map a2 k8 ^10 ^9 pdu4.3 zdu2 zds2.1
```

となります。同様に、デリミタ形式への変換なら

```
/d /map “a2”, “k8”, ^10 ^9 pdu4.3, zdu2, zds2.1
```

となります。 ^10 と ^9 のあとにはコンマを入れていないことに注意してください。また、「英名の意味・由来」の項目は最後の項目なので、どちらでも ^20 とは書かずに省略しました。

空項目を追加する

空項目を作るのも簡単です。プリント形式への変換なら、

```
_n
```

と書きます。例題のデータに新たに「ボイジャーⅠ通過日」「ボイジャーⅡ通過日」というＹＹＹＭＭＤＤ形式の項目を追加するなら、

```
/map a2 k8 a10 a9 _8 _8 pdu4.3 zdu2 zds2.1 k20
```

とする要領です。この例では英名の「読み」の項目のうしろに、つづけて２つ追加してみました。

デリミタ形式への変換の場合は、

```
, ,
```

と書けば空項目になります。

前の例と同じように、「ボイジャーＩ通過日」「ボイジャーⅡ通過日」というＹＹＹＹMMＤＤ形式の項目を英名の「読み」の項目のあとに追加するには、

```
/d /map “a2″, “k8″, “a10″, “a9″, , , pdu4.3, zdu2, zds2.1, “k20″
```

とします。

項目長を増減する

項目長の増減もしばしば必要になります。プリント形式への変換なら

```
/map a2:3 k8:10 a10 a9 pdu4.3 zdu2 zds2.1:6 k20:30
```

のようになります。同様に、デリミタ形式への変換なら

```
/dnc /map “a2:3″, “k8:10″, “a10″, “a9″, pdu4.3, zdu2, zds2.1:6, “k20:30″
```

のようになります。どちらでも、基本的にはコロン（:）のあとに出力幅を書いていくだけです。

レコード長を指定する

デリミタ形式への変換のときは、レコード長は指定できません。しかし、プリント形式への変換のときは､桁移動の機能を使って強制的に目的のレコード長のテキストファイルを得ることができます。

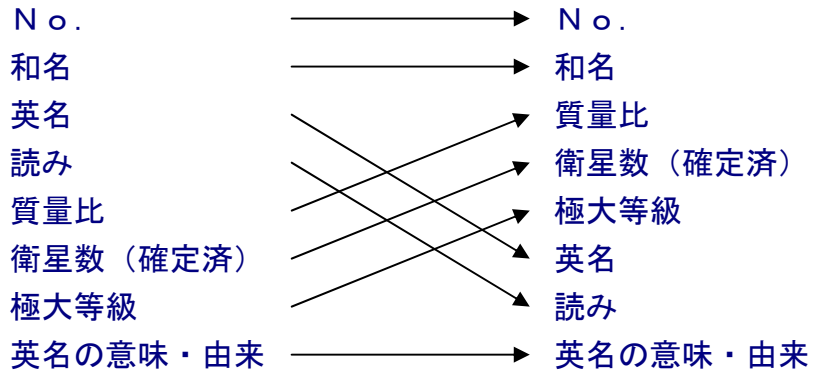
具体的には、レコード長をたとえば１２０バイト（改行コード２バイト含まず）に変換するなら、

```
/map ･･･ @:120
```

のように指定します。

項目を組み替える

項目組み替えは、桁移動の機能を利用するので少し面倒です。例題のデータを、



というふうに組み替えるとします。

プリント形式への変換なら、つぎのようなパラメータファイルを作ります。

```
/map ↓
      ank         2        -- Ｎｏ.（惑星番号）　↓
      kanji       8        -- 和名　↓
      ank         10       -- 英名　↓
      ank         9        -- 読み　↓
      packdisp    u4.3     -- 質量比　↓
      zonedisp    u2       -- 衛星数（確定済）　↓
      zonedisp    s2.1     -- 極大等数（みかけ上の最大の明るさ）　↓
      kanji       20       -- 英名の意味・由来　↓
```

つぎに、各項目に入力桁位置をつけていきます（@0～@38）。

```
 /map ↓
     @0   ank         2        -- Ｎｏ.（惑星番号）　↓
     @2   kanji       8        -- 和名　↓
     @10  ank         10       -- 英名　↓
     @20  ank         9        -- 読み　↓
     @29  packdisp    u4.3     -- 質量比　↓
     @33  zonedisp    u2       -- 衛星数（確定済）　↓
     @35  zonedisp    s2.1     -- 極大等数（みかけ上の最大の明るさ）　↓
     @38  kanji       20       -- 英名の意味・由来　↓
```

こうしておいて、あとはエディタで好きなように順番を入れ替えていきます。たとえば、

```
/map ↓
      @0   ank       2       -- Ｎｏ．（惑星番号）　↓
      @2   kanji     8       -- 和名 ↓
      @29  packdisp  u4.3    -- 質量比 ↓
      @33  zonedisp  u2      -- 衛星数（確定済）　↓
      @35  zonedisp  s2.1    -- 極大等数（みかけ上の最大の明るさ）　↓
      @10  ank       10      -- 英名 ↓
      @20  ank       9       -- 読み ↓
      @38  kanji     20      -- 英名の意味・由来 ↓
```

とします。

デリミタ形式への変換のときも、プリント形式へのときとほとんど同じ要領で項目組み替えができます。つぎのようなパラメータファイルを作ります。

```
/delimited                  -- コンマ区切り（ＣＳＶ）形式に変換
/map ↓
      "ank       2",       -- Ｎｏ．（惑星番号）　↓
      "kanji     8",       -- 和名 ↓
      "ank       10",      -- 英名 ↓
      "ank       9",       -- 読み ↓
      packdisp   u4.3 ,    -- 質量比 ↓
      zonedisp   u2 ,      -- 衛星数（確定済）　↓
      zonedisp   s2.1 ,    -- 極大等数（みかけ上の最大の明るさ）　↓
      "kanji     20"       -- 英名の意味・由来 ↓
```

つぎに、各項目に入力桁位置をつけていきます（@0～@38）。

```
/delimited                          -- コンマ区切り（ＣＳＶ）形式に変換
/map ↓
      @0   "ank        2",          -- Ｎｏ.（惑星番号）　↓
      @2   "kanji      8",          -- 和名　↓
      @10  "ank        10",         -- 英名　↓
      @20  "ank        9",          -- 読み　↓
      @29  packdisp    u4.3 ,       -- 質量比　↓
      @33  zonedisp    u2 ,         -- 衛星数（確定済）　↓
      @35  zonedisp    s2.1 ,       -- 極大等数（みかけ上の最大の明るさ）　↓
      @38  "kanji      20"          -- 英名の意味・由来　↓
```

こうしておいて、あとはエディタで好きなように順番を入れ替えていきます。たとえば、

```
/delimited                          -- コンマ区切り（ＣＳＶ）形式に変換
/map ↓
      @0  "ank         2",          -- Ｎｏ.（惑星番号）　↓
      @2  "kanji       8",          -- 和名　↓
      @29  packdisp    u4.3 ,       -- 質量比　↓
      @33  zonedisp    u2 ,         -- 衛星数（確定済）　↓
      @35  zonedisp    s2.1 ,       -- 極大等数（みかけ上の最大の明るさ）　↓
      @10  "ank        10",         -- 英名　↓
      @20  "ank        9",          -- 読み　↓
      @38  "kanji      20"          -- 英名の意味・由来　↓
```

とします。

この例では最終項目を移動させませんでしたが、最終項目を移動させるときは、あらかじめコンマをおぎなっておくとよいでしょう。また、組み替え後に最終項目になった行のコンマを削除することも忘れてはいけません。

■使用例１（プリント形式）

例１）ＡＮＫコードのみのファイルを変換する

ドライブＡ：のＩＢＭファイルＪＯＵＲＮＡＬを、ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴに変換します。レコードに含まれるのはＡＮＫ文字だけで、それをＪＩＳ８コードに変換します。改行コードは自動的につきます。

```
C:¥>ft getdata a:journal c:*.dat ↓
```

例２）ＡＮＫ・漢字まじりのファイルを変換する

例１と同様ですが、ＡＮＫ項目と漢字項目があります。項目別変換の手間を減らすために、漢字項目の前後にはＫＩ／ＫＯをつけてもらっています。

```
C:¥>ft getdata a:journal c:*.dat /map km ↓          (/MAP KanjiMix)
```

例３）漢字のないソースプログラムの行番号を削除する

ホストで作成したＩＢＭ形式のソースプログラムが数本あり、最後の８桁に行番号がついています。これをＷｉｎｄｏｗｓファイルに変換しますが、行番号は不要でかえってじゃまになるので、それをカットします。漢字は使っていません。ＡＮＫのみです。

```
C:¥>ft getdata a:* c:*.src /map a$-8 ↓          (/MAP Ank$-8)
```

例４）漢字のあるソースプログラムの行番号を削除する

例３と同様ですが、漢字が使われています。

```
C:¥>ft getdata a:* c:*.src /map km$-8 ↓          (/MAP KanjiMix$-8)
```

例５）レコード長を拡張する

ドライブＡ：にＩＢＭファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２があります。レコード長は８０バイトで、これらをすべてドライブＣ：のＷｉｎｄｏｗｓファイルＺＡＩＫＯ０１．ＤＡＴ～ＺＡＩＫＯ１２．ＤＡＴに変換します。内容はＡＮＫデータのみです。変換後のＷｉｎｄｏｗｓファイルをＢＡＳＩＣのランダムファイルとして扱いやすいように、レコード長を２５６バイトに拡張します。２５６バイトのうち、最後の２バイトを改行コードにして、テキストファイルとしても扱えるようにします。

```
C:¥>ft getdata a:zaiko?? c:*.dat /map a:256-2 ↓          (/MAP Ank:256-2)
```

例６）桁移動の機能でレコード長を拡張する

ドライブＡ：のＩＢＭファイルＤＡＴＡを、ドライブＣ：のＷｉｎｄｏｗｓファイルＳＨＯＨＩＮ．ＰＲＮに変換します。内容は商品マスターで、０から数えて８桁目から３０桁（１５文字）の漢字項目（商品名）があります。元のレコード長は１６８バイトですが、将来のために２５６バイトに拡張します。拡張した分のパディング文字はスペースになります。

```
C:¥>ft getdata a:data c:shohin.prn /map a8 k30 a @:256-2 ↓
                                    (/MAP Ank8 Kanji30 Ank @:256-2)
```

例７）問い合わせモードで、必要なファイルだけ変換する

ドライブＡ：のいくつかのＩＢＭファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子はつけません。ＩＢＭファイルのほうは、ファイルによってレコード長が異なるので、Ｗｉｎｄｏｗｓファイルのレコード長はＩＢＭファイルのレコード長に合わせます（実際は改行コードの分２バイト増えます）。データの内容はＡＮＫのみです。一応、全ファイル指定とし、／Ｑｕｅｒｙオプションをつけて問い合わせモードで実行し、変換するファイルだけＯＫボタンで応答します。

```
C:¥>ft getdata a:* c: /q ↓                    (/Query)
```

例８）項目別変換する

ドライブＡ：にＩＢＭファイルＡＤＤＲＥＳＳがあり、内容は住所録で、つぎのように項目が分かれています。

項　目	タイプ	幅	
氏名	漢字	１６	
フリガナ	ＡＮＫ	１６	計３４バイト
電話番号	ＡＮＫ	１２	
郵便番号	ＡＮＫ	６	
住所	漢字	４０	
生年月日	ＹＹＭＭＤＤ	６	
区分	ＡＮＫ	１	

レコード長は２５６バイトです。これをドライブＣ：のＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢに変換します。ＡＮＫ項目と漢字項目が混在しているので項目別変換します。

```
C:¥>ft getdata a:address c:*.db /map k16 a34 k40 dyymmdd:yyyy-mm-dd a1 ↓
          (/MAP Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd Ank1)
```

例９）例８をバッチファイル＋パラメータファイル化する

例８と同じ処理をバッチファイル化し、コマンド行方式で実行できるようにします。／ＭＡＰオプションをわかりやすくするために、パラメータファイルを利用します。

＜ＧＥＴＡＤＤＲ．ＢＡＴ＞

```
start /w ft getdata a:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
/map ↓
      kanji  16                    -- 氏名 ↓
      ank    16                    -- フリガナ ↓
      ank    12                    -- 電話番号 ↓
      ank    6                     -- 郵便番号 ↓
      kanji  40                    -- 住所 ↓
      date   yymmdd:yyyy-mm-dd     -- 生年月日 ↓
      ank    1                     -- 区分 ↓
```

例 10）項目を組み替える（その１）

例８と同様ですが、出力桁移動の機能を利用して、「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft getdata a:address c:*.db /map @:1 k16 a34 k40 dyymmdd:yyyy-mm-dd
    @:0 a1 ↓
    (/MAP @:1 Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd @:0 Ank1)
```

例 11）項目組み替えにパラメータファイルを使う（その１）

例 10 と同じ処理を、パラメータファイルを使って実行します。

```
C:¥>ft getdata a:address c:*.db ++getaddr.p1 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ１＞

```
/map ↓
      @:1  kanji  16                    -- 氏名 ↓
           ank    16                    -- フリガナ ↓
           ank    12                    -- 電話番号 ↓
           ank    6                     -- 郵便番号 ↓
           kanji  40                    -- 住所 ↓
           date   yymmdd:yyyy-mm-dd     -- 生年月日 ↓
      @:0  ank    1                     -- 区分 ↓
```

例 12）項目を組み替える（その２）

例８と同様ですが、入力桁移動の機能を利用して、「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft getdata a:address c:*.db /map @96 a1 @0 k16 a34 k40 dyymmdd:yyyy-mm-dd ↓
          (/MAP @96 Ank1 @0 Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd)
```

例 13）項目組み替えにパラメータファイルを使う（その２）

例 10 と同じ処理を、パラメータファイルを使って実行します。

```
C:¥>ft getdata a:address c:*.db ++getaddr.p2 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ２＞

```
/map ↓
      @96  ank     1                    -- 区分 ↓
      @0   kanji   16                   -- 氏名 ↓
           ank     16                   -- フリガナ ↓
           ank     12                   -- 電話番号 ↓
           ank     6                    -- 郵便番号 ↓
           kanji   40                   -- 住所 ↓
           date    yymmdd:yyyy-mm-dd    -- 生年月日 ↓
```

例 14）例 13 のパラメータファイルに日付項目変換のための年設定等を追加する

日付項目変換のための年設定、日付区切り設定をパラメータファイルに追加します。

＜ＧＥＴＡＤＤＲ．Ｐ２＞

```
/map ↓
      @96  ank        1                   -- 区分 ↓
      @0   kanji      16                  -- 氏名 ↓
           ank        16                  -- フリガナ ↓
           ank        12                  -- 電話番号 ↓
           ank        6                   -- 郵便番号 ↓
           kanji      40                  -- 住所 ↓
           year       sshowa              -- 入力日付年＝昭和 ↓
           datedelim  period              -- 出力日付区切＝ﾋﾟﾘｵﾄﾞ ↓
           date       yymmdd:yyyy-mm-dd   -- 生年月日 ↓
```

■使用例２（デリミタ形式）

例１）コンマ区切り（ＣＳＶ）形式に変換する

ドライブＡ：にＩＢＭファイルＡＤＤＲＥＳＳがあり、内容は住所録で、つぎのように項目が分かれています。

項　目	タイプ	幅	
氏名	漢字	１６	
フリガナ	ＡＮＫ	１６	計３４バイト
電話番号	ＡＮＫ	１２	
郵便番号	ＡＮＫ	６	
住所	漢字	４０	
生年月日	ＹＹＭＭＤＤ	６	
区分	ＡＮＫ	１	

これをドライブＣ：のＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢに変換します。ＡＮＫ項目と漢字項目が混在しているので、項目別変換し、コンマ区切り（ＣＳＶ）形式にします。

```
C:¥>ft getdata a:address c:*.db /d /map "k16", "a16", "a12", "a6", "k40",
    dyymmdd:yyyy-mm-dd, a1 ↓
        (/Delimted  /MAP "Kanji16", "Ank16", "Ank12", "Ank6", "Kanji40",
                                          Date yymmdd:yyyy-mm-dd, Ank1)
```

例２）タブ区切り形式に変換する

例１と同じファイルを、タブ区切り形式に変換します。

```
C:¥>ft getdata a:address c:*.db /dbtab /map "k16", "a16", "a12", "a6", "k40",
    dyymmdd:yyyy-mm-dd, a1 ↓
       (/Delimted By TAB /MAP "Kanji16", "Ank16", "Ank12", "Ank6", "Kanji40",
                                          Date yymmdd:yyyy-mm-dd, Ank1)
```

例３）スペース区切り形式に変換する

例１と同じファイルを、スペース区切り形式に変換します。

```
C:¥>ft getdata a:address c:*.db /dbsp /map "k16", "a16", "a12", "a6", "k40",
    dyymmdd:yyyy-mm-dd, a1 ↓
    (/Delimted By Space /MAP "Kanji16", "Ank16", "Ank12", "Ank6", "Kanji40",
                                          Date yymmdd:yyyy-mm-dd, Ank1)
```

例４）項目を組み替える

例１のファイルの、「区分」の項目を先頭に持ってきます。

```
C:¥>ft getdata a:address c:*.db /d /map @96 a1, @0 "k16", "a16", "a12", "a6",
    "k40", dyymmdd:yyyy-mm-dd ↓
        (/Delimited  /MAP @96 Ank1, @0 "Kanji16", "Ank16", "Ank12", "Ank6",
                                        "Kanji40", Date yymmdd:yyyy-mm-dd)
```

例５）パラメータファイルを使う

例１と同じコンマ区切り(ＣＳＶ)形式への変換を、パラメータファイルを使って実行します。パラメータファイルには、日付項目変換のための年設定、日付区切り設定を追加します。

```
C:¥>ft getdata a:address c:*.db +addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/delimited ↓
/map ↓
      "kanji     16" ,                  -- 氏名 ↓
      "ank       16" ,                  -- フリガナ ↓
      "ank       12" ,                  -- 電話番号 ↓
      "ank       6" ,                   -- 郵便番号 ↓
      "kanji     40" ,                  -- 住所 ↓
      year       sshowa                 -- 入力日付年＝昭和 ↓
      datedelim  period                 -- 出力日付区切＝ﾋﾟﾘｵﾄﾞ ↓
      date       yymmdd:yyyy-mm-dd ,    -- 生年月日 ↓
      ank        1                      -- 区分（親戚、友人、仕事仲間など） ↓
```

例６）バッチファイル化する

例５をさらにバッチファイル化します。パラメータファイルＡＤＤＲ．Ｐは例５と同一です。

＜ＡＤＤＲ．ＢＡＴ＞

```
start /w ft getdata a:address c:*.db +addr.p ↓
```

■注意事項

漢字変換をするときは、漢字変換方式の割り当てを忘れずに

Ｋａｎｊｉ変換やＫａｎｊｉＭｉｘ変換を行うときは、あらかじめ

適当な漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、入力幅、出力幅は漢字データについても

バイト単位で指定

します。漢字の文字数ではないことに注意してください。

その他の注意事項

「Ｇｅｔ系コマンド」の節を参照してください。

２．６　ＧｅｔＲａｎｄ（ｇｒ）　ゲット・ランド

ＩＢＭ→Ｗｉｎ　ランダムファイル変換

ＧｅｔＲａｎｄ(ｇｒ)コマンドは、ＩＢＭ形式のデータファイル(おもにＣＯＢＯＬのデータ)をＷｉｎｄｏｗｓのランダムファイル（固定長ファイル）に変換するコマンドです。Ｗｉｎｄｏｗｓ　ＣＯＢＯＬの順ファイルやＢＡＳＩＣのランダムファイルに変換するのに使います。項目別変換ができます。作成されるＷｉｎｄｏｗｓファイルは固定長で、改行コードもデリミタもないファイルになります。ただし、改行コードの挿入はできます。

そのほかバイナリ変換を使ってアップロードデータを作成すること等もできます。

■コマンド形式

コマンド	パラメータ
ＧｅｔＲａｎｄ ｇｒ	［ＩＢＭファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ＩＢＭファイル、Ｗｉｎｄｏｗｓファイル

ＩＢＭファイルとＷｉｎｄｏｗｓファイルの指定方法は、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＧｅｔＲａｎｄ(ｇｒ)コマンドには、つぎのオプションが指定できます。

Ｇｅｔｒａｎｄ(ｇｒ)コマンド固有のオプション

／Ｓｉｚｅ	Ｗｉｎｄｏｗｓ側のレコード長を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

Ｇｅｔ系コマンド共通のオプション

／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＲＥＰｌａｃｅ	変換先ファイルの置換設定

このうち、／Ｓｉｚｅオプションと／ＭＡＰオプションがとくに重要です。

Ｇｅｔ系コマンド共通のオプションは、「Ｇｅｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＧｅｔＲａｎｄ(ｇｒ)コマンド固有のオプション

ＧｅｔＲａｎｄ(ｇｒ)コマンドに固有のオプションを説明します。

／Ｓｉｚｅ ---- Ｗｉｎｄｏｗｓ側のレコード長を指定する

／Ｓｉｓｅ　式 ／Ｓｉｚｅ　＄

Ｗｉｎｄｏｗｓ側（出力）のレコード長を指定します。Ｗｉｎｄｏｗｓレコード長は、ふつう

　　／ｓｉｚｅ　２５６　　のように１０進数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

　　＄　　ＩＢＭ側（入力）のレコード長

ＩＢＭファイルのレコード長を基準にして、Ｗｉｎｄｏｗｓファイルのレコード長を決めることができます。たとえば、

　　／ｓｉｚｅ　＄＋２

と指定すれば、「元のＩＢＭファイルのレコード長に＋２したものをＷｉｎｄｏｗｓ側のレコード長にせよ」という意味になります。
／Ｓｉｚｅオプションを省略すると、

　　／ｓｉｚｅ　＄

と指定した扱いになり、Ｗｉｎｄｏｗｓレコード長＝ＩＢＭレコード長になります。ですから、比較的単純な変換で、レコード長は元と同じでよい、というときは／Ｓｉｚｅオプションを省略します。

◆注意　----　Ｗｉｎｄｏｗｓ自身には「レコード長」の概念がない

Ｗｉｎｄｏｗｓ自体にはファイルごとに登録されたレコード長というものはありません。そのため、かなり自由が利きます。アプリケーションがファイルを扱う論理的な単位をここではＷｉｎｄｏｗｓ側のレコード長、あるいはＷｉｎｄｏｗｓレコード長と呼んでいます。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／MAP	Ａｎｋ変換 Ｋａｎｊｉ変換 ＡｎｋｉＺｅ変換 ＫａｎｊｉＺｅ変換 ＫａｎｊｉＭｉｘ変換 ＵｓｅｒＡ／Ｂ変換 Ｎｕｍｅｒｉｃ変換 Ｂｉｎａｒｙ変換 ＺｏｎｅＤｉｓｐ変換（Ｚｏｎｅ変換） ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換） ＤｉｓｐＺｏｎｅ変換 ＤｉｓｐＰａｃｋ変換 ＺｏｎｅＺｏｎｅ変換 ＺｏｎｅＰａｃｋ変換 ＰａｃｋＺｏｎｅ変換 ＰａｃｋＰａｃｋ変換 ＤｉｓｐＢｉｎ変換 ＢｉｎＤｉｓｐ変換 ＺｏｎｅＢｉｎ変換 ＰａｃｋＢｉｎ変換 ＢｉｎＺｏｎｅ変換 ＢｉｎＰａｃｋ変換 ＢｉｎＢｉｎ変換 ＢｉｎａｒｙＸ変換 ＴｏＤｉｓｐ変換 ＴｏＺｏｎｅ変換 ＴｏＰａｃｋ変換 ＴｏＢｉｎ変換 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 改行コード挿入／改行コード挿入２ 桁移動／入力スキップ／出力スキップ	・・・

／ＭＡＰ　　Ａｎｋ

この／ＭＡＰオプションで細かい変換方法を指示します。本来なら、自動的に項目を認識して変換ができると便利です。しかし、ホストの、とくにＣＯＢＯＬのデータには、**データ自身に桁数や小数点位置の判断に必要な情報が含まれていない**、という特性があります。そのため自動変換は原理的に不可能なのです。

◆注意　----　マルチレコードレイアウト等の指定について

／ＭＡＰオプションには、マルチレコードレイアウト等の指定ができるＡｔｌａｓ構文を記述することもできます。詳細は、マルチレコード編のマニュアルを参照してください。

Ａｎｋ変換

Ａｎｋ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ項目を変換します。どのコード変換が行われるかは、変換設定のＡＮＫコードの設定で決まります（ふつう、セットアップ時に一回だけ行います)。

入力幅は、　　ａ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＡｎｋ変換するときには ---- ／ｍａｐ　ａ
最終項目をＡｎｋ変換するときには -------- ／ｍａｐ　・・・　ａ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ａｎｋ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡＮＫ形式、最大２５６文字で、半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ａ［‘ＡＮＫ’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａ２０　　という指定は、
ａ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ項目のおわりにスペース（２０Ｈ）が詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

たとえば、／ＭＡＰオプションの最後にａ０：＊と指定すると、「残りはスペースでクリアせよ」という意味になります。

ＡｎｋｉＺｅ（ＡＮＫ化）変換

ＡｎｋｉＺｅ　［入力幅｜＊］［：出力幅］

ＩＢＭ側（入力）文字列をＡＮＫ（半角）に変換します。入力側文字列は漢字（全角）文字列であると見なして変換します。

入力幅は、　ａｚ２０　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　ホスト側（入力）のレコード長
＊　ホスト側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＡｎｋｉＺｅ変換するときには ---- ／ｍａｐ　ａｚ
最終項目をＡｎｋｉＺｅ変換するときには -------- ／ｍａｐ　・・・　ａｚ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａｚ２０　　という指定は、
ａｚ２０：２０　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ項目のおわりに空白が詰められます。

Ｋａｎｊｉ変換

Ｋａｎｊｉ［入力幅｜＊｜定数］［：出力幅］

漢字項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に一回だけ行います)。

入力幅は、　　ｋ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません)。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉ変換するときには ---- ／ｍａｐ　ｋ
最終項目をＫａｎｊｉ変換するときには -------- ／ｍａｐ　・・・　ｋ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定した漢字形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ｋａｎｊｉ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列は漢字形式、最大２５６文字で、半角(')でくくる
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋ［'漢字'］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋ１６　　という指定は、
ｋ１６：１６　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を大きくしなければならない場合があります。拡張漢字のクエスチョン変換を使っていて、オーバーフローの危険があるときなどです（最大で入力幅の３倍)。そのときは、出力幅を指定します。

項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側の漢字項目のおわりに漢字変換方式で設定されている漢字スペース（２０２０Ｈ／８１４０Ｈ）が詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

ＫａｎｊｉＺｅ（漢字化）変換

ＫａｎｊｉＺｅ　［入力幅｜＊］［：出力幅］

ＩＢＭ側（入力）文字列を漢字（全角）に変換します。入力側文字列はＡＮＫ文字列であると見なして変換します。

入力幅は、　　ｋｚ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません)。
式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉＺｅ変換するときには ---- ／ｍａｐ　ｋｚ
最終項目をＫａｎｊｉＺｅ変換するときには -------- ／ｍａｐ　・・・　ｋｚ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると入力幅の２倍が指定されます。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を大きくしなければならない場合があります。拡張漢字のクエスチョン変換を使っていて、オーバーフローの危険があるときなどです（最大で入力幅の３倍)。そんなときは、出力幅を指定します。
項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側の漢字項目のおわりに漢字変換方式で設定されている漢字スペースが詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ・漢字まじり項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に一回だけ行います)。漢字の前後にＫＩ／ＫＯがついていないと、この変換方法は適用できません。変換後にＫＩ／ＫＯが取れて、その分、左詰めになります。

入力幅は、　　ｋｍ３２　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉＭｉｘ変換するときには ---- ／ｍａｐ　ｋｍ
最終項目をＫａｎｊｉＭｉｘ変換するときには -------- ／ｍａｐ　・・・　ｋｍ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ・漢字まじり形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

ＫａｎｊｉＭｉｘ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡｎｋ・漢字まじり形式、最大２５６文字
半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋｍ［‘ＡＮＫ・漢字まじり’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋｍ３２　　という指定は、
ｋｍ３２：３２　　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を変更したい場合があります。ＫＩ／ＫＯが取れて短くなることや、逆に、拡張漢字のクエスチョン変換を使っていてオーバーフローの危険があるときなどです（最大で、入力幅の３倍－ＫＩ／ＫＯのバイト数の合計)。そのときは、出力幅を指定します。

項目長を縮めると、ＡＮＫ・漢字まじり項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のＡＮＫ・漢字まじり項目のおわりにスペース（２０Ｈ）が詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

ＵｓｅｒＡ変換
ＵｓｅｒＢ変換

```
ＵｓｅｒＡ［入力幅｜＊］［：出力幅］
ＵｓｅｒＢ［入力幅｜＊］［：出力幅］
```

ＵｓｅｒＡ／Ｂ変換は、利用者独自のバイト単位の変換処理が必要なときに、ＡＮＫ変換表Ｕｓｅｒ－Ａ、Ｕｓｅｒ－Ｂを書き替えて利用します。ＵｓｅｒＡ／Ｂ変換には、Ａｎｋ変換の説明がほとんどそのまま当てはまります。

入力幅は、　　ｕａ１０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＵｓｅｒＡ変換するときには ---- ／ｍａｐ　ｕａ
最終項目をＵｓｅｒＡ変換するときには -------- ／ｍａｐ　・・・　ｕａ

のようにして、入力幅を省略できます。ＵｓｅｒＢ変換でも同様です。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｕａ１０　　という指定は、
ｕａ１０：１０　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ユーザーＡ／Ｂ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のユーザーＡ／Ｂ項目のおわりにＮＵＬ（００Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

Ｎｕｍｅｒｉｃ変換

Ｎｕｍｅｒｉｃ［入力幅｜１５］［：出力幅］

文字形式の数値項目どうしの変換をします。

Ｎｕｍｅｒｉｃ変換は、Ａｎｋ変換と後述のＺｏｎｅＤｉｓｐ変換の中間的なものです。Ａｎｋ変換と比較すると、

文字形式数値しか通さない
入力幅の省略値が１５桁（バイト）である
右詰めになる

などの点が異なります。

「文字形式数値しか通さない」というのは、具体的には、

＋、－、０～９、ピリオド（.）、Ｅ、ｅ、Ｄ、ｄ

しか変換しないで、これら以外の文字は捨ててしまうということです。たとえば、通貨記号（￥／＄）や位取りのコンマ（,）などは削除されるので、リストファイルから入力データファイルを作るときなどに利用できます。

Ｂｉｎａｒｙ変換

Ｂｉｎａｒｙ［入力幅｜＊｜定数］［：出力幅］

Ｂｉｎａｒｙ変換は、「コード変換を一切しない」という変換方法です。Ｂｉｎａｒｙ変換には、Ａｎｋ変換の説明がほぼそのまま当てはまります。

入力幅は、　　ｂ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（入力）のレコード長
＊　　ＩＢＭ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＢｉｎａｒｙ変換するときには ---- ／ｍａｐ　ｂ
最終項目をＢｉｎａｒｙ変換するときには -------- ／ｍａｐ　・・・　ｂ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したバイナリ形式の定数が、Ｗｉｎｄｏｗｓ側に挿入されることになります。指定した定数は、コード変換されずにそのまま出力されます。

Ｂｉｎａｒｙ変換では、つぎの定数が使えます。

１６進列定数 -------　｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト
汎用定数 ---------- ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｂ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｂ２０　　という指定は、
ｂ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。
項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、バイナリ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、Ｗｉｎｄｏｗｓ側のバイナリ項目のおわりにＮＵＬ（００Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　Ｗｉｎｄｏｗｓ側（出力）のレコード長
＊　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

Ａｎｋ変換の説明が当てはまると書きましたが、実際の運用はかなり違ったものになります。とくに、

```
／ｍａｐ　ｂｉｎａｒｙ
```

などと指定して、レコード全体をＢｉｎａｒｙ変換することが多いことです。このとき、レコード長が変わらないように／Ｓｉｚｅオプションは省略します。一方、

```
／ｍａｐ　・・・　ｂ５　・・・
```

などのようにして、レコードの一部をバイナリ項目として扱うこともあります。

ＺｏｎｅＤｉｓｐ変換（Ｚｏｎｅ変換）

```
ＺｏｎｅＤｉｓｐ　ピクチャ［：出力幅］
（Ｚｏｎｅ　ピクチャ［：出力幅］）
```

ホストのＣＯＢＯＬのゾーン形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＺｏｎｅＤｉｓｐ　ｓ４．１　という指定は、
ＺｏｎｅＤｉｓｐ　ｓ４．１：７　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換）

```
ＰａｃｋＤｉｓｐ　ピクチャ［：出力幅］
（Ｐａｃｋ　ピクチャ［：出力幅］）
```

ホストのＣＯＢＯＬのパック形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

パック形式では、整数部桁数＋小数部桁数を奇数にしておくのが通例です。整数部の最上位桁に意味があるのかないのかは、半々の割合です。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０とする
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＰａｃｋＤｉｓｐ　ｓ４．１　という指定は、
ＰａｃｋＤｉｓｐ　ｓ４．１：７　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＰａｃｋＤｉｓｐ変換（Ｐａｃｋ変換）〔ＢＣＤ形式〕

```
ＰａｃｋＤｉｓｐ　ピクチャ（ｂ指定）[：出力幅]
（Ｐａｃｋ　ピクチャ（ｂ指定）[：出力幅]）
```

ＰＯＳ端末等で使われているＢＣＤ形式数値項目を、文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
１２３．４　という数字が３バイトのＢＣＤ形式の項目に記録されているとすれば、ピクチャは

ｂ５．１　　と必ずｂ指定をします。

なお、ピクチャは省略できません。

ＢＣＤ形式では、整数部桁数＋小数部桁数を偶数にしておくのが通例です。整数部の最上位桁に意味があるのかないのかは、半々の割合です。

出力幅は省略できます。出力幅を省略すると、

整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＰａｃｋＤｉｓｐ　ｂ５．１　　という指定は、
ＰａｃｋＤｉｓｐ　ｂ５．１：７　　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、上位桁が切り捨てられるので、注意してください。

ＤｉｓｐＺｏｎｅ変換

ＤｉｓｐＺｏｎｅ　［入力幅｜１５］：ピクチャ

ホストの文字形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的に桁数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの符号つきゾーン形式の項目に記録するとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

例を示すと、

ＤｉｓｐＺｏｎｅ　８：ｓ４．１　のような指定になります。

ＤｉｓｐＰａｃｋ変換

ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ

ホストの文字形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的に桁数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を３バイトの符号つきパック形式の項目に記録するとすれば、ピクチャは

ｓ４．１　　と指定します。

同様に－１２３．４　という数字をＢＣＤ形式の項目に記録するとすれば、ピクチャは

ｂ４．１　　と指定します。

なお、ピクチャは省略できません。

例を示すと、

ＤｉｓｐＰａｃｋ　８：ｓ４．１　　のような指定になります。

ＺｏｎｅＺｏｎｅ変換

```
ＺｏｎｅＺｏｎｅ　入力ピクチャ［：出力ピクチャ］
                  ［入力ピクチャ］：出力ピクチャ
```

ホストのＣＯＢＯＬのゾーン形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　：ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＺｏｎｅＰａｃｋ変換

```
ＺｏｎｅＰａｃｋ　入力ピクチャ［：出力ピクチャ］
                  ［入力ピクチャ］：出力ピクチャ
```

ホストのＣＯＢＯＬのゾーン形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目およびＢＣＤ項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＰａｃｋ	：ｓ４．１	という指定は、
ＺｏｎｅＰａｃｋ	ｓ４．１：ｓ４．１	という指定と同じであり、
ＺｏｎｅＰａｃｋ	：ｂ４．１	という指定は、
ＺｏｎｅＰａｃｋ	ｂ４．１：ｂ４．１	という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＰａｃｋ	ｓ４．１	という指定は、
ＺｏｎｅＰａｃｋ	ｓ４．１：ｓ４．１	という指定と同じであり、
ＺｏｎｅＰａｃｋ	ｂ４．１	という指定は、
ＺｏｎｅＰａｃｋ	ｂ４．１：ｂ４．１	という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＰａｃｋＺｏｎｅ変換

```
ＰａｃｋＺｏｎｅ　入力ピクチャ［：出力ピクチャ］
                  ［入力ピクチャ］：出力ピクチャ
```

ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

１２３．４　という数字が３バイトのＢＣＤ形式の項目に記録されているとすれば、入力ピクチャは

ｂ５．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＺｏｎｅ　：ｓ４．１　という指定は、
ＰａｃｋＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＺｏｎｅ　ｓ４．１　という指定は、
ＰａｃｋＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＰａｃｋＰａｃｋ変換

```
ＰａｃｋＰａｃｋ　入力ピクチャ［：出力ピクチャ］
                  ［入力ピクチャ］：出力ピクチャ
```

ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

１２３．４　という数字が３バイトのＢＣＤ形式の項目に記録されているとすれば、入力ピクチャは

ｂ３．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＰａｃｋ	：ｓ４．１	という指定は、
ＰａｃｋＰａｃｋ	ｓ４．１：ｓ４．１	という指定と同じであり、
ＰａｃｋＰａｃｋ	：ｂ４．１	という指定は、
ＰａｃｋＰａｃｋ	ｂ４．１：ｂ４．１	という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＰａｃｋＰａｃｋ	ｓ４．１	という指定は、
ＰａｃｋＰａｃｋ	ｓ４．１：ｓ４．１	という指定と同じであり、
ＰａｃｋＰａｃｋ	ｂ４．１	という指定は、
ＰａｃｋＰａｃｋ	ｂ４．１：ｂ４．１	という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＤｉｓｐＢｉｎ変換

ＤｉｓｐＢｉｎ　［入力幅｜１５］：２進キャスト［ピクチャ］

ホストの文字形式数値項目を、Ｗｉｎｄｏｗｓの２進形式整数・小数項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的に桁数を指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

　　ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

例を示すと、

　　ＤｉｓｐＢｉｎ　１０：ｉ４ｓ４．１　　のような指定になります。

ＢｉｎＤｉｓｐ変換

ＢｉｎＤｉｓｐ　２進キャスト［ピクチャ］:［出力幅］

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓの文字形式数値項目に変換します。変換結果は右詰めになります。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

出力幅は省略できます。出力幅を省略すると、ピクチャに従って出力されます。
出力幅、ピクチャ共に省略すると、

入力のバイト数	１	２	３	４	５	６	７	８
出 力 幅	３	５	８	１０	１３	１５	１７	１８

の要領で２進キャストから自動的に計算された値が使われます。例を示すと、

ＢｉｎＤｉｓｐ　ｉ４ｓ　　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ：１０　　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＺｏｎｅＢｉｎ変換

ＺｏｎｅＢｉｎ　入力ピクチャ：２進キャスト［ピクチャ］
［入力ピクチャ］：２進キャスト　ピクチャ

ホストのＣＯＢＯＬのゾーン形式数値項目を、Ｗｉｎｄｏｗｓの２進形式整数・小数項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　　と指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

入力ピクチャも、２進キャストにつづくピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＺｏｎｅＢｉｎ　：ｉ４ｓ４．１　　という指定は、
ＺｏｎｅＢｉｎ　ｓ４．１：ｉ４ｓ４．１　　という指定と同じです。

ＰａｃｋＢｉｎ変換

ＰａｃｋＢｉｎ　入力ピクチャ：２進キャスト［ピクチャ］
［入力ピクチャ］：２進キャスト　ピクチャ

ホストのＣＯＢＯＬのパック形式数値項目を、Ｗｉｎｄｏｗｓの２進形式整数・小数項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　　と指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

入力ピクチャも、２進キャストにつづくピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＰａｃｋＢｉｎ　　：ｉ４ｓ４．１　　という指定は、
ＰａｃｋＢｉｎ　　ｓ４．１：ｉ４ｓ４．１　　という指定と同じです。

ＢｉｎＺｏｎｅ変換

```
ＢｉｎＺｏｎｅ　２進キャスト　ピクチャ：[出力ピクチャ]
　　２進キャスト［ピクチャ］：出力ピクチャ
```

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

　　ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの符号つきゾーン形式の項目に記録するとすれば、ピクチャは

　　ｓ４．１　　と指定します。

２進キャストにつづくピクチャも出力ピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＢｉｎＺｏｎｅ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＺｏｎｅ　ｉ４ｓ４．１：ｓ４．１　　という指定と同じです。

ＢｉｎＰａｃｋ変換

ＢｉｎＰａｃｋ　２進キャスト　ピクチャ：[出力ピクチャ]
２進キャスト［ピクチャ］：出力ピクチャ

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を３バイトの符号つきパック形式の項目に記録するとすれば、ピクチャは

ｓ４．１　　と指定します。

また、ＢＣＤ形式なら

ｂ４．１　　と指定します。

２進キャストにつづくピクチャも出力ピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＢｉｎＰａｃｋ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＰａｃｋ　ｉ４ｓ４．１：ｓ４．１　　という指定と同じです。

ＢｉｎＢｉｎ変換

ＢｉｎＢｉｎ　２進キャスト［ピクチャ］:［２進キャスト［ピクチャ］］
［２進キャスト［ピクチャ］］：２進キャスト［ピクチャ］

ホストの２進形式整数･小数項目を、Ｗｉｎｄｏｗｓの２進形式整数･小数項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目とする場合、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

２進キャスト／ピクチャは入力または出力のどちらかを省略できます。入力を省略すると「出力と同じ」とみなされます。たとえば、

ＢｉｎＢｉｎ　：ｉ４ｓ４．１　　という指定は、
ＢｉｎＢｉｎ　ｉ４ｓ４．１：ｉ４ｓ４．１　　という指定と同じです。

出力を省略すると「入力と同じ」とみなされます。たとえば、

ＢｉｎＢｉｎ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＢｉｎ　ｉ４ｓ４．１：ｉ４ｓ４．１　　という指定と同じです。

なお、入力２進キャスト／ピクチャと出力２進キャスト／ピクチャを同時に省略することはできません。

ＢｉｎａｒｙＸ変換

ＢｉｎａｒｙＸ｛幅｜定数｝

ＢｉｎａｒｙＸ変換は、「コード変換を一切せずに、幅分のデータをバイト単位で左右反転する」という変換方法です。２進数値データ（整数、小数、実数）は、ＩＢＭ側が正順であるのに対して、Ｗｉｎｄｏｗｓ側が逆順であることが多いので、２進数値データの内容をそのままバイト単位で左右反転する場合等に使用します。ＢｉｎＢｉｎ変換のような加工機能はありませんが、その分だけ処理が高速です。

幅は、　ｂｘ８　のように、１０進のバイト数で指定します。

たとえば、　ｂｘ４　のように指定し、１６進表現で入力データが　０１　ＡＢ　ＣＤ　ＥＦ　であれば、出力データは　ＥＦ　ＣＤ　ＡＢ　０１　になります。

また、幅の代わりに定数を指定することもできます。その場合、指定したバイナリ形式の定数は左右反転してから、Ｗｉｎｄｏｗｓ側に挿入されることになります。

ＢｉｎａｒｙＸ変換では、つぎの定数が使えます。

１６進列定数 -------　｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト

定数は、　ｂｘ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

ＴｏＤｉｓｐ変換

ＴｏＤｉｓｐ｛数値定数｜システム変数｝[：出力幅]

入力した数値定数、またはシステム変数の値を、文字形式数値項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。変換結果は右詰めになります。

入力値として、つぎの数値定数、システム変数が使えます。省略はできません。

数値定数　　**整数定数、小数定数、１６進定数**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　[－１２３．４]　のように、[　]でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

出力幅は省略できます。出力幅を省略すると、入力した値を正確に出力するために必要な、最小限の幅で出力されます。たとえば、

ＴｏＤｉｓｐ　[－１２３．４]　　という指定は、
ＴｏＤｉｓｐ　[－１２３．４]：６　　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＴｏＺｏｎｅ変換

ＴｏＺｏｎｅ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、５バイトの符号つきゾーン形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＺｏｎｅ　　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＰａｃｋ変換

ＴｏＰａｃｋ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、３バイトの符号つきパック形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

また、ＢＣＤ形式なら

ｂ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＰａｃｋ　　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＢｉｎ変換

ＴｏＢｉｎ｛数値定数｜システム変数｝：２進キャスト［ピクチャ］

入力した数値定数、またはシステム変数の値を、Ｗｉｎｄｏｗｓの２進形式整数・小数項目に変換して、Ｗｉｎｄｏｗｓ側に挿入します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、４バイトの符号つき２進形式の項目に記録するとすれば、出力ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＢｉｎ　　［－１２３．４］：ｉ４ｓ４．１　　のような指定になります。

Ｙｅａｒ設定（年設定）

```
Ｙｅａｒ　　Ｗｎｎ｜Ｓｎｎ
                          ：Ｗｎｎ｜：Ｓｎｎ
              Ｗｎｎ｜Ｓｎｎ　：Ｗｎｎ｜：Ｓｎｎ
```

日付データ項目を変換する際の、年の２桁（ｙｙ）と４桁（ｙｙｙｙ）の変換方法を設定します。Ｗｎｎの“W”はウインドウ方式を、Ｓｎｎの“Ｓ”はシフト方式を意味し、“ｎｎ”は００～９９の数字で指定します。“：”の左側の指定は入力側、“：”の右側の指定は出力側への適用になり、入力側のみ、出力側のみ、入出力両方の３通りの指定ができます。たとえば、

Ｙｅａｒ	Ｗ３０	入力データの年を１９３０～２０２９年とみなす
Ｙｅａｒ	：Ｓ２５	出力データの年下２桁を、－２５する
Ｙｅａｒ	Ｗ３０：Ｓ２５	入力データの年を１９３０～２０２９年とみなし、 出力データの年下２桁を、－２５する

のように指定します。

また、シフト方式（“Ｓｎｎ”指定）では、つぎの特殊指定ができます。

Ｙｅａｒ	ＳＳｈｏｗａ	“Ｓ２５”の指定と同じ（昭和通年方式）
Ｙｅａｒ	ＳＨｅｉｓｅｉ	“Ｓ８８”の指定と同じ（平成通年方式）

Ｙｅａｒ設定はＤａｔｅ変換が実行された時に適用になり、複数のＹｅａｒ設定がなされている場合は、Ｄａｔｅ変換の直前のＹｅａｒ設定が有効になります。

Ｙｅａｒ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

Ｙｅａｒ	Ｗ３０：Ｗ３０	入出力データの年を１９３０～２０２９年とみなす

ＤａｔｅＤｅｌｉｍ設定（日付区切り設定）

```
ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ｜ＨＹＰＨＥＮ｜ＰＥＲＩＯＤ
```

日付データ項目を出力する際の日付区切り記号をつぎの３つの中から設定します。

指定文字	日付区切り記号	データ例
ＳＬＡＳＨ	／（スラッシュ）	１９９８／１２／３１
ＨＹＰＨＥＮ	－（ハイフン）	１９９８－１２－３１
ＰＥＲＩＯＤ	.（ピリオド）	１９９８．１２．３１

ＤａｔｅＤｅｌｉｍ設定はＤａｔｅ変換が実行された時に適用になり、複数のＤａｔｅＤｅｌｉｍ設定がなされている場合は、Ｄａｔｅ変換の直前のＤａｔｅＤｅｌｉｍ設定が有効になります。

ＤａｔｅＤｅｌｉｍ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ　日付区切り記号を“／”にする

Ｄａｔｅ変換

```
Ｄａｔｅ　入力日付マスク：出力日付マスク
```

日付データ項目を変換します。

入力日付マスク、出力日付マスクは必ず指定します。省略はできません。たとえば、

Ｄａｔｅ　ｙｙｍｍｄｄ：ｙｙｙｙ－ｍｍ－ｄｄ

のように指定すると、コード変換後に、入力側６バイトの日付データ項目を、出力側１０バイトの日付データ項目に編集します。その際に、Ｙｅａｒ設定、ＤａｔｅＤｅｌｉｍ設定が適用になります。

改行コード挿入

```
！Ｒ
```

Ｗｉｎｄｏｗｓ側（出力）レコードに改行コード（ＣＲ／ＬＦ＝０Ｄ０ＡＨ）を挿入します。
ふつう、項目長の増減がない単純な変換のとき／ＭＡＰオプションの最後に指定し、テキストファイル化するのに使います。そのときは、改行コードの２バイト分Ｗｉｎｄｏｗｓレコード長が増えるので、／Ｓｉｚｅオプションでその分を＄＋２と指定して調整するのを忘れないでください。

なお、変換結果のテキストファイル化には、本来はＧｅｔＤａｔａ（ｇｄ）コマンドのプリント形式への変換機能を使うべきです。これなら、変換後のレコード長をほとんど意識する必要がありません。

改行コード挿入２

```
！Ｎ
```

Ｗｉｎｄｏｗｓ側（出力）レコードに改行コード（ＣＲ／ＬＦ＝０Ｄ０ＡＨ）を挿入します。ランダムファイル変換では、改行コード挿入との違いはありません。

桁移動

＠入力桁位置
＠：出力桁位置

変換対象にするＩＢＭ側（入力）の桁位置や、変換結果を書き込むＷｉｎｄｏｗｓ側（出力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

入力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

＄　　ＩＢＭ側（入力）のレコード長
．　　ＩＢＭ側（入力）の現在の桁位置

という特殊変数が使えます。たとえば、

＠．－４０　Ｋａｎｊｉ２０　＾２０　　と指定すれば、

今の入力桁位置から４０バイト戻り、漢字２０バイト変換せよ
そして、元の位置に戻れ

という意味になります。また、

＠＄－８　ＺｏｎｅＵ８　　と指定すれば、

ＩＢＭ側（入力）レコード末尾の８バイト前に位置づけ、
８バイトＺｏｎｅ変換せよ

という意味になります。
＠：ｎの形式で、出力桁位置を移動することもできます。ふつう、１０進数で桁位置を指定します。式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（出力）のレコード長
．　　Ｗｉｎｄｏｗｓ側（出力）の現在の桁位置

◆注意　----　先頭を０桁目とする
Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

入力スキップ

```
^［n｜1］
```

ＩＢＭ側（入力）レコードに不要な項目があるとき、それをスキップして変換できます。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

^３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

^３　　は　　^^^　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（入力）の残りバイト数

これを使えば、　　／ｍａｐ　・・・　^＊　　として「残りは全部スキップせよ」という指定もできます。しかし、何の指定もなかった分は変換対象にならないので、この ^ ＊は冗長です。省いたほうがよいでしょう。

出力スキップ

```
_［n｜1］
```

入力スキップとは逆に、Ｗｉｎｄｏｗｓ側（出力）に何桁か空きを作ることもできます。空項目を作るのがおもな用途です。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

_３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

_３　　は　　＿＿＿　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　Ｗｉｎｄｏｗｓ側（出力）の残りバイト数

／ＰＨａｓｅ　　---- ヘッダ・トレーラの生成・付加のタイミングを指定する

```
／ＰＨａｓｅ　　［Ｏｐｅｎ＜ｎ＞］,
　　［Ｍａｉｎ＜ｎ＞］,
　　［Ｃｌｏｓｅ＜ｎ＞］
```

データを変換する際にヘッダやトレーラを付加したいときに、この／ＰＨａｓｅオプションで各フェーズの呼び出しが起きる回数を指定します。フェーズにはつぎの３種類があります。

Ｏｐｅｎフェーズ	ヘッダの生成・付加を行う
Ｍａｉｎフェーズ	本体の変換をする
Ｃｌｏｓｅフェーズ	トレーラやエンドレコードの生成・付加を行う

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数を、＜ｎ＞に１０進数値で指定します。Ｍａｉｎフェーズでは、

＄　　全レコード分

という指定もできます。
Ｏｐｅｎフェーズ→Ｍａｉｎフェーズ→Ｃｌｏｓｅフェーズの順に呼び出しを行います。

◆注意　----　フェーズごとの指定について

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数のみ、指定します。各フェーズごとの動作は、Ａｔｌａｓ構文を使って設定する必要があります。詳細はマルチレコード編のマニュアルを参照してください。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｓｉｚｅ　＄	Ｗｉｎｄｏｗｓレコード長＝ＩＢＭレコード長
／ＭＡＰ　Ａｎｋ	レコード全体をＡＮＫデータとして変換､改行コードなし
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＮｏＥＯＦ	ＥＯＦコードはつけない
／ＰＨａｓｅ　Ｏｐｅｎ０，Ｍａｉｎ＄，Ｃｌｏｓｅ０	ヘッダ・トレーラをつけないで全レコード変換する
／ＲＥＰｌａｃｅ　ＹＥＳ	変換先ファイルを置き換える

と指定したものとして、ランダムファイル変換を行います。

■解説

●オプションの指定方法

単純な変換

レコード全体で１項目とみなしてよいときは、オプションの指定も単純です。たとえば、Ａｎｋ変換だけですむときは、／ＭＡＰオプションを省略できます。

```
／ＭＡＰ　Ａｎｋ
```

が省略値だからです。また「各レコードの先頭８０バイトだけＡｎｋ変換したい」といったときは、

```
／ｍａｐ　ａｎｋ　＊：＊　／ｓｉｚｅ　８０
```

のように、／Ｓｉｚｅオプションを指定するだけで十分です。

ファイル全体をバイナリ変換するときは、

```
／ｍａｐ　ｂｉｎａｒｙ
```

と指定します。この場合、／Ｓｉｚｅオプションは省略します。レコード長はあまり意味を持ちません。

項目別変換

ＩＢＭ形式の、つぎのようなレコードレイアウトのファイルＰＬＡＮＥＴがあるとします。

ＰＬＡＮＥＴのレコードレイアウト

項番	項　　目	桁	幅	デ　ー　タ　形　式	内　容　例
１	Ｎｏ.	0	2	ＡＮＫ	1
２	和名	2	8	漢字	水星
３	英名	10	10	ＡＮＫ	MERCURY
４	読み	20	9	ＡＮＫ	ﾏｰｷｭﾘｰ
５	質量比	29	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
６	衛星数（確定済）	33	2	符号なしゾーン形式 整数部２桁	0
７	極大等級	35	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
８	英名の意味・由来	38	20	漢字	口神）神の使者
--	フィラー	58	6	--	--

これをランダムファイル変換し、フィラーは切り捨てるものとします。そのときのオプションの指定は、コマンド行に書くなら、

```
/s64 /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20
```

のようになります。／ｓ６４は／Ｓｉｚｅ　６４の短縮指定です。また、これをパラメータファイルに書くなら、つぎのようになります。

```
/size  64                -- Ｗｉｎｄｏｗｓ側レコード長は６４バイト ↓
/map ↓
      ank        2       -- Ｎｏ.（惑星番号） ↓
      kanji      8       -- 和名 ↓
      ank        10      -- 英名 ↓
      ank        9       -- 読み ↓
      packdisp   u4.3    -- 質量比 ↓
      zonedisp   u2      -- 衛星数（確定済） ↓
      zonedisp   s2.1    -- 極大等数（みかけ上の最大の明るさ） ↓
      kanji      20      -- 英名の意味・由来 ↓
```

●変換テストから本番まで

レコードレイアウトが複雑なときに、いきなりＧｅｔＲａｎｄ(ｇｒ)コマンドを使うのは得策ではありません。変換結果が正しいかどうかが確認しにくいからです。兄弟コマンドにあたる、ＧｅｔＤａｔａ(ｇｄ)コマンドのプリント形式への変換を利用して、変換結果を確認してから、バッチファイル、パラメータファイルを書き替えるのがよい方法です。

まず、ＧｅｔＤａｔａ(ｇｄ)コマンドのプリント形式への変換を使って、変換テストをします。

```
C:¥>ft getdata a:planet c:test /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

のようにします。しかし、これを直接入力するのは大変です。そこで、実際には１行の指定ですむ場合でもテスト用のバッチファイルを作ります。つまり、

＜ＰＬＡＧＥＴＲＡ．ＢＡＴ（その１）＞

```
start /w ft getdata a:planet c:test /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

のような内容のバッチファイルを作って、テストランと修正を繰り返していきます。正確なレコードレイアウトが手に入らなければ、ＩＢＭディスクエディタ（ＧＵＩ部）でデータ内容を調査する必要があるかもしれません。

「これでＯＫ」となれば、このテスト用バッチファイルを修正して、本番用のバッチファイルにします。簡単な本番用バッチファイルは、

＜ＰＬＡＧＥＴＲＡ．ＢＡＴ（その２）＞

```
@echo off ↓
:: 太陽系の惑星データの変換 ↓
::                          ＩＢＭ形式→Ｗｉｎランダムファイル形式 ↓
start /w ft getrand a:planet c:*.ran /s64 /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1
 k20 ↓
```

のような内容になります。

これをバッチファイルとオプションを記述したパラメータファイルに分けるなら、バッチファイルのほうは、

＜ＰＬＡＧＥＴＲＡ．ＢＡＴ（その３）＞

```
@echo off ↓
:: 太陽系の惑星データの変換 ↓
::                          ＩＢＭ形式→Ｗｉｎランダムファイル形式 ↓
start /w ft getrand a:planet c:*.ran ++plagetra.p ↓
```

となり、パラメータファイルのほうは、

＜ＰＬＡＧＥＴＲＡ．Ｐ＞

```
--  太陽系の惑星データの変換（Ｆｏｒ　ＧｅｔＲａｎｄ）          -- ↓
--                          ＩＢＭ形式→Ｗｉｎランダムファイル形式  -- ↓
↓
/size  64                   -- Ｗｉｎｄｏｗｓレコード長＝６４バイト ↓
/map ↓
      ank         2         -- Ｎｏ．（惑星番号） ↓
      kanji       8         -- 和名 ↓
      ank         10        -- 英名 ↓
      ank         9         -- 読み ↓
      packdisp    u4.3      -- 質量比 ↓
      zonedisp    u2        -- 衛星数（確定済） ↓
      zonedisp    s2.1      -- 極大等級（みかけ上の最大の明るさ） ↓
      kanji       20        -- 英名の意味・由来 ↓
```

のようになります。

●いろいろな加工・編集

いらない項目をスキップする

ホストからデータを持ってくると、いらない項目（あまり意味のないフィラーなど）がいくつもあることが多いものです。それをスキップして変換するのは簡単で、

```
^n
```

と書きます。ｎはスキップするバイト数です。たとえば、例題のデータから「英名」「読み」「英名の意味・由来」の項目を変換対象からはずすなら、オプションの指定は、

```
/s32 /map a2 k8 ^10 ^9 pdu4.3 zdu2 zds2.1
```

のようになります。「英名の意味・由来」の項目は最後の項目なので、^20 とは書かずに省略しました。

空項目を追加する

空項目を作るのも簡単で、

```
_n
```

と書くだけです。例題のデータに新たに「ボイジャーⅠ通過日」「ボイジャーⅡ通過日」というＹＹＹＹMMＤＤ形式の項目を追加する場合は、

```
/s80 /map a2 k8 a10 a9 _8 _8 pdu4.3 zdu2 zds2.1 k20
```

とする要領です。この例では英名の「読み」の項目のうしろに、つづけて２つ追加してみました。／Ｓｉｚｅオプションでｗｉｎｄｏｗｓ側（出力）レコード長を指定したことにも注目してください。

項目長の増減

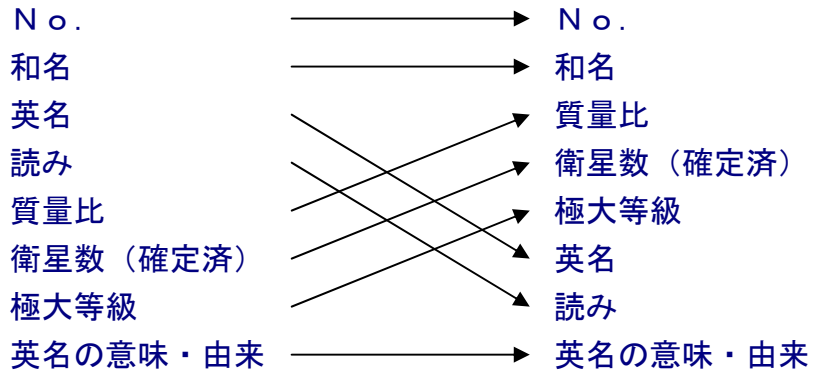
必要に応じて、項目長の増減をするのも簡単です。つぎのように、

```
/s128 /map a2:3 k8:10 a10 a9 pdu4.3 zdu2 zds2.1:6 k20:30
```

とします。基本的にはコロン（：）のあとに新しい項目長（出力幅）を指定するだけです。

項目を組み替える

項目組み替えは、桁移動の機能を利用するので少し面倒です。例題のデータを、



というふうに組み替えるには、つぎのようなパラメータファイルを作ります。

```
/map ↓
      ank         2        -- Ｎｏ.（惑星番号） ↓
      kanji       8        -- 和名 ↓
      ank         10       -- 英名 ↓
      ank         9        -- 読み ↓
      packdisp    u4.3     -- 質量比 ↓
      zonedisp    u2       -- 衛星数（確定済） ↓
      zonedisp    s2.1     -- 極大等級（みかけ上の最大の明るさ） ↓
      kanji       20       -- 英名の意味・由来 ↓
```

つぎに、各項目の桁位置をつけていきます（@0～@38）。

```
/map ↓
      @0   ank         2        -- Ｎｏ.（惑星番号） ↓
      @2   kanji       8        -- 和名 ↓
      @10  ank         10       -- 英名 ↓
      @20  ank         9        -- 読み ↓
      @29  packdisp    u4.3     -- 質量比 ↓
      @33  zonedisp    u2       -- 衛星数（確定済） ↓
      @35  zonedisp    s2.1     -- 極大等級（みかけ上の最大の明るさ） ↓
      @38  kanji       20       -- 英名の意味・由来 ↓
```

こうしておいて、あとはエディタで好きなように順番を入れ替えていきます。たとえば、

```
/map ↓
      @0   ank        2       -- Ｎｏ．（惑星番号）　↓
      @2   kanji      8       -- 和名 ↓
      @29  packdisp   u4.3    -- 質量比 ↓
      @33  zonedisp   u2      -- 衛星数（確定済）　↓
      @35  zonedisp   s2.1    -- 極大等級（みかけ上の最大の明るさ）　↓
      @10  ank        10      -- 英名 ↓
      @20  ank        9       -- 読み ↓
      @38  kanji      20      -- 英名の意味・由来 ↓
```

とします。

■使用例

例１）ＡＮＫのみ、レコード長も変えない

ドライブＡ：のＩＢＭファイルＪＯＵＲＮＡＬを、ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴに変換します。レコードに含まれるのはＡＮＫデータだけです。レコード長は変えません。

```
C:¥>ft getrand a:journal c:*.dat ↓
```

例２）テキストファイル化する

例１と同様のファイルですが、レコード末尾に改行コードをつけ、テキストファイルにします。通常、テキストファイル化にはＧｅｔＤａｔａ（ｇｄ）コマンドを使います。

```
C:¥>ft getrand a:journal c:*.dat /map a!r /s$+2 ↓        (/Map Ank!R /Size $+2)
```

例３）バイナリ変換する

ドライブＡ：のＩＢＭファイルＦＯＮＴ２４を、ドライブＣ：のＷｉｎｄｏｗｓファイルＦＯＮＴ２４．ＰＸＬにバイナリ変換します。

```
C:¥>ft getrand a:font24 c:*.pxl /map b ↓                 (/Map Binary)
```

例４）レコード長を増やす

ドライブA:にあるレコード長が８０バイトのＩＢＭファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２を、ドライブＣ：のＷｉｎｄｏｗｓファイルＺＡＩＫＯ０１．ＤＡＴ～ＺＡＩＫＯ１２．ＤＡＴに変換します。変換後のＷｉｎｄｏｗｓファイルをＢＡＳＩＣのランダムファイルとして扱いやすいように、レコード長を２５６バイトに拡張します。

```
C:¥>ft getrand a:zaiko?? c:*.dat /s256 ↓                 (/Size 256)
```

例５）レコード長をそろえる

ドライブＡ：のすべてのＩＢＭファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子はつけません。ＩＢＭファイルのほうは、ファイルによってレコード長が異なりますが、すべて５１２バイト以内です。Ｗｉｎｄｏｗｓファイルへ変換後のレコード長は、すべて５１２バイトにそろえます。内容はＡＮＫデータだけです。

```
C:¥>ft getrand a:* c: /s512 ↓                            (/Size 512)
```

例６）項目別変換する

ドライブＡ：にＩＢＭファイルＡＤＤＲＥＳＳがあり、内容は住所録で、つぎのように項目が分かれています。

項　目	タイプ	幅	
氏名	漢字	１６	
フリガナ	ＡＮＫ	１６	計３４バイト
電話番号	ＡＮＫ	１２	
郵便番号	ＡＮＫ	６	
住所	漢字	４０	
生年月日	ＹＹＭＭＤＤ	６	
区分	ＡＮＫ	１	

レコード長は２５６バイトです。これをドライブＣ：のＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢに変換します。ＡＮＫ項目と漢字項目が混在しているので項目別変換します。

```
C:¥>ft getrand a:address c:*.db /map k16 a34 k40 dyymmdd:yyyy-mm-dd a1 ↓
          (/MAP Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd Ank1)
```

例７）項目組み替え（その１）

例６と同様ですが、出力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft getrand a:address c:*.db /map @:1 k16 a34 k40 dyymmdd:yyyy-mm-dd
    @:0 a1 ↓
    (/MAP @:1 Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd @:0 Ank1)
```

例８）項目組み替え（その２）

これも例６と同様ですが、入力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft getrand a:address c:*.db /map @90 a1 @0 k16 a34 k40 dyymmdd:yyyy-mm-dd ↓
          (/MAP @90 Ank1 @0 Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd)
```

例９）バッチファイル化する

例６と同じ処理をバッチファイル化して、実行します。

＜ＧＥＴＡＤＤＲ．ＢＡＴ＞

```
@echo off ↓
start /w ft getrand a:address c:*.db /map k16 a34 k40 dyymmdd:yyyy-mm-dd a1 ↓
```

例 10）パラメータファイルを使う

例６と同じ処理を、バッチファイルとパラメータファイルを利用して実行します。パラメータファイルには、日付項目変換のための年設定、日付区切り設定を追加します。

＜ＧＥＴＡＤＤＲ．ＢＡＴ＞

```
@echo off ↓
start /w ft getrand a:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
   ・
   ・
/map ↓
      kanji       16                  -- 氏名 ↓
      ank         16                  -- フリガナ ↓
      ank         12                  -- 電話番号 ↓
      ank         6                   -- 郵便番号 ↓
      kanji       40                  -- 住所 ↓
      year        sshowa              -- 入力日付年＝昭和 ↓
      datedelim   period              -- 出力日付区切＝ﾋﾟﾘｵﾄﾞ ↓
      date        yymmdd:yyyy-mm-dd   -- 生年月日 ↓
      ank         1                   -- 区分 ↓
```

例 11）ＫＩ／ＫＯつきにしてもらったデータファイルを変換する

ドライブＡ：のＩＢＭファイルＨＡＮＢＡＩは、もともとＣＯＢＯＬのデータファイルなのですが、パソコン側の処理を簡単にするためホストで漢字項目の前後にＫＩ／ＫＯをつけてもらいました。それをドライブＣ：のＷｉｎｄｏｗｓファイルＨＡＮＢＡＩ．ＤＡＴに変換します。

```
C:¥>ft getrand a:hanbai c:*.dat /map km ↓          (/MAP KanjiMix)
```

あらかじめ適切な漢字変換方式を割り当てておくことを忘れないでください。

■注意事項

漢字変換をするときは、漢字変換方式の割り当てを忘れずに

Ｋａｎｊｉ変換やＫａｎｊｉＭｉｘ変換を行うときは、あらかじめ

適当な漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、入力幅、出力幅は漢字データについても

バイト単位で指定

します。漢字の文字数ではないことに注意してください。

その他の注意事項

「Ｇｅｔ系コマンド」の節を参照してください。

２．７　Ｐｕｔ系コマンド

Ｐｕｔ系のファイル指定と共通オプション

ここでは、Ｐｕｔ系コマンドの共通事項を説明します。頭にＰｕｔがつく３つのコマンド

ＰｕｔＴｅｘｔ（ｐｔ）コマンド　　Ｗｉｎ→ＩＢＭテキストファイル変換
ＰｕｔＤａｔａ（ｐｄ）コマンド　　Ｗｉｎ→ＩＢＭデータファイル変換
ＰｕｔＲａｎｄ（ｐｒ）コマンド　　Ｗｉｎ→ＩＢＭランダムファイル変換

をまとめてＰｕｔ系コマンドと呼び、Ｗｉｎｄｏｗｓファイルを ＩＢＭファイルに変換するのに使います。この３つのコマンドを用途によって使い分けます。これら３コマンドは、よく似た兄弟です。

■コマンド形式

コマンド	パラメータ
ＰｕｔＴｅｘｔ	［Ｗｉｎｄｏｗｓファイル　ＩＢＭファイル　［オプション］］
ｐｔ	
ＰｕｔＤａｔａ	
ｐｄ	
ＰｕｔＲａｎｄ	
ｐｒ	

Ｐｕｔ系コマンドは、どれも上に示したパラメータ形式を持っています。ファイルの指定方法は３つともまったく同じです。オプションには共通のものもあれば、３つ別々のものもあります。

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル

［ｄ：］［パス名指定］基本ファイル名［．拡張子］

ｄ：はドライブ名

入力側のＷｉｎｄｏｗｓファイルを指定します。

ドライブ名は、Ａ：～Ｚ：、＠：、？：のどれかで指定します。ドライブ名を指定すると、そのドライブを検索します。
ドライブ名は省略可能です。ドライブ名を省略すると、カレントドライブを検索します。

パス名指定（￥ディレクトリ名￥サブディレクトリ名￥・・・）ができます。指定したディレクトリ配下のファイルを検索します。パス名指定を省略すると、カレントディレクトリを検索します。

基本ファイル名と拡張子には、ワイルドカード文字（＊と？）を使うことができます。ワイルドカード文字を使うと、一致するファイルをすべて検索し変換の対象にします。

まとめると、あるディレクトリのファイルをすべて変換したいなら、

Ｃ：＊．＊　　のように指定し、

拡張子が．ＤＡＴのファイルをすべて変換したいなら、

Ｃ：＊．ＤＡＴ　　のような指定になります。

◆注意　----　使用できるデバイスファイルについて

Ｗｉｎｄｏｗｓファイルとして使用できるデバイスファイルは、ＮＵＬ（ダミーデバイス）だけです。

●ＩＢＭファイル

ｄ：はドライブ名

出力側のＩＢＭファイルを指定します。

ドライブ名はＡ：～Ｐ：、または０：～３：で指定します。省略はできません。

ＩＢＭファイル名を省略すると＊を指定したものとみなし、Ｗｉｎｄｏｗｓ側の基本ファイル名がＩＢＭファイル名になります。ふつうはＩＢＭファイル名を省略し、自動的に同じ名前になるようにします。
ＩＢＭファイル名には＊を２個まで含めることができます。１つめの＊はＷｉｎｄｏｗｓ側の基本ファイル名で、２つめの＊はＷｉｎｄｏｗｓ側の拡張子で自動的に置き替えられます。

まとめると、Ｗｉｎｄｏｗｓファイルの基本ファイル名を引き継ぐときは、

Ａ：　　のようにドライブ名だけの指定ですみ、

新しいファイル名をつけるときは、

Ａ：ＮＥＷＦＩＬＥ　　のような指定になります。

また、Ｗｉｎｄｏｗｓファイルの拡張子もＩＢＭファイル名に含ませたいときは、

Ａ：＊＃＊　　のように指定します。

Ｗｉｎｄｏｗｓファイルが、　　ＢＡＳＥＮＡＭ．ＥＸＴ　　であれば、
ＩＢＭファイルは、　　ＢＡＳＥＮＡＭ＃ＥＸＴ　　になります。

●オプション

Ｐｕｔ系コマンドには各コマンドに共通のオプションと、コマンド毎に違うオプションがあります。

Ｐｕｔ系コマンド共通のオプション

／ＮＥＷ	新規ファイルとして作成する
／ＯＬＤ	既存ファイルに出力する
／ＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルは置き替えない
／Ｆｏｒｍ	ＩＢＭファイルの形式を指定する ⇒拡張形式・半拡張形式・基本形式
／ＥｘｔｒａＳｐａｃｅ	新規作成後の予備領域の量を指定する
／ＢｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＢＯＥ境界を指定する
／ＥｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＥＯＥ境界を指定する
／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名を確認せず自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	ボリューム順序番号出力の制御
／ＥＯＦ	ＥＯＦコードを検査する
／ＮｏＥＯＦ	ＥＯＦコードを検査しない

このうち、／Ｆｏｒｍオプションがとくに重要です。

ＰｕｔＴｅｘｔ（ｐｔ）コマンド固有のオプション

／Ｃｏｄｅ	ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない

ＰｕｔＤａｔａ（ｐｄ）コマンド固有のオプション

／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ	引用符はずしの詳細を指定する
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＳＱｕｅｅｚｅ	空行を無視する
／ＮｏＳＱｕｅｅｚｅ	空行を無視しない
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定

ＰｕｔＲａｎｄ（ｐｒ）コマンド固有のオプション

／Ｓｉｚｅ	Ｗｉｎｄｏｗｓ側のレコード長を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

以下、Ｐｕｔ系コマンドに共通のオプションを説明します。

／ＮＥＷ ---- 新規ファイルとして作成する
／ＯＬＤ ---- 既存ファイルに出力する

／ＮＥＷ ／ＯＬＤ

ＩＢＭファイルの作成モードを指定します。

／ＮＥＷオプションを指定すると、ＩＢＭファイルを新規作成します。同じ名前のファイルがすでにあるときは、後述の／ＲＥＰｌａｃｅオプションと組み合わせて使います。こちらが省略値です。

／ＯＬＤオプションは、すでに作成、またはアロケートずみのＩＢＭファイルへの出力（上書き）を指示します。／ＯＬＤオプションを指定したときは、後述の

／ＲＥＰｌａｃｅオプション
／ＮｏＲＥＰｌａｃｅオプション
／Ｆｏｒｍオプション
／ＥｘｔｒａＳｐａｃｅオプション
／ＢｏｅＢｏｕｎｄａｒｙオプション
／ＥｏｅＢｏｕｎｄａｒｙオプション

は指定しても無効になります。これらのオプションは、ＩＢＭファイルを新規作成するときに必要な機能だからです。

なお、／ＯＬＤオプションを使うとＥＯＥ（領域終了アドレス）を超える書き込みはできません。あらかじめ、十分な大きさのＩＢＭファイルをアロケートしておく必要があります。

／ＲＥＰｌａｃｅ -------- 同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ ---- 同名ＩＢＭファイルは置き替えない

／ＲＥＰｌａｃｅ
／ＮｏＲＥＰｌａｃｅ

新規作成するＩＢＭファイルと同名のＩＢＭファイルがすでに存在する場合の処置を指定します。

／ＲＥＰｌａｃｅオプションを指定すると、既存の同じ名前のＩＢＭファイルをいったん削除してから変換処理に入ります。

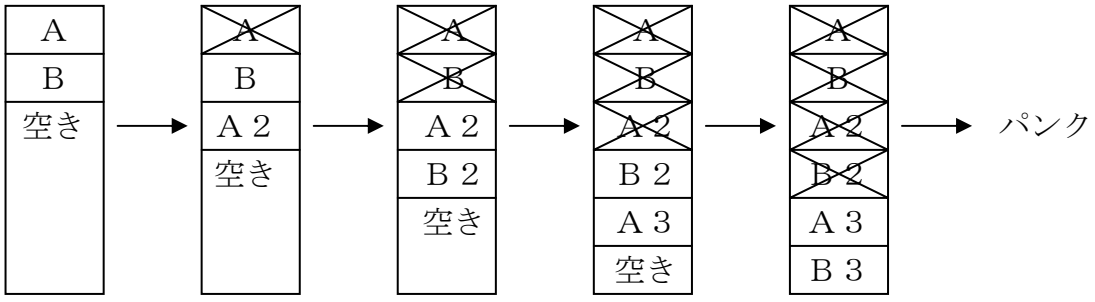
／ＮｏＲＥＰｌａｃｅオプションを指定すると、同じ名前のＩＢＭファイルがあった場合、変換処理を中断します。こちらが省略値です。

◆注意　----　複数ファイルの置換は避けること

複数ファイルに対してワイルドカード文字を使って、

```
Ｃ：￥＞ｆｔ　ｐｕｔｄａｔａ　ｃ：＊．ｄａｔ　ａ：／ｒｅｐ ↓
```

のように／ＲＥＰｌａｃｅオプションで置換してはいけません。そうすると、空き領域がしだいに繰り上がっていくからです。そして、いずれはＩＢＭディスクがパンクします。その様子を下図に示します。



このように複数ファイルの置換をしたいときは、面倒でもあらかじめｉＤｅｌｅｔｅ（ｉｄｅｌ）コマンドを使い、置換したいファイルを削除しておく必要があります。

／Ｆｏｒｍ ---- ＩＢＭファイルの形式を指定する

```
／Ｆｏｒｍ ( Ｅｘｔｅｎｄｅｄ ～
           ＨａｌｆＥｘｔｅｎｄｅｄ ～
           Ｐｒｉｍａｒｙ ～ )
```

この／Ｆｏｒｍオプションは、ＩＢＭファイルの形式を決める、非常に重要なオプションです。

つぎの３つのキーワードで、ＩＢＭファイルの大まかな形式を

Ｅｘｔｅｎｄｅｄ	**拡張形式**
ＨａｌｆＥｘｔｅｎｄｅｄ	**半拡張形式**
Ｐｒｉｍａｒｙ	**基本形式**

のどれにするのかを指定します。これらのキーワードを省略すると、

Ｅｘｔｅｎｄｅｄ	**拡張形式**

とみなします。

これらのキーワードと一緒に、レコード長やブロック長、ブロック化の有無などの指定をします。その指定の方法には、３つとも少しずつ違いがあります。

／Ｆｏｒｍ（拡張形式） ---- ＩＢＭファイルの形式を指定する（拡張形式）

```
／Ｆｏｒｍ　［Ｅｘｔｅｎｄｅｄ］
              ［論理レコード長］
                ┌ ブロック長指定 ┐
                └ Ａｕｔｏ       ┘
                  ┌ レコード形式 ┐
                  └ ＦＢ         ┘
                    ┌ Ｃｌｉｐ   ┐
                    └ ＮｏＣｌｉｐ ┘
```

新規作成するＩＢＭファイルの形式を指定します。拡張形式では論理レコード長、ブロック長、レコード形式、およびブロック長の処理を指定します。

Ｅｘｔｅｎｄｅｄ

拡張形式にすることを指定します。省略もできます。

論理レコード長

論理レコード長を１～９９９９の範囲の１０進数で指定します。この省略値はコマンドによって違い、省略するとそれぞれ、つぎのようになります。

ＰｕｔＴｅｘｔ（ｐｔ）コマンド	８０バイト
ＰｕｔＤａｔａ（ｐｄ）コマンド	１２８バイト
ＰｕｔＲａｎｄ（ｐｒ）コマンド	２５６バイト

ブロック長指定

ブロック長は、以下に示すように、必要に応じていろいろな指定ができます。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｒｅｃｏｒｄｓ＜ｎ＞	＜ｎ＞レコード分（ブロック化係数による指定）
Ｍａｘ	規約上の最大値（＝トラック容量）を割り当てる
Ａｕｔｏ	論理レコード長やレコード形式に応じて、 適当なブロック長を自動的に割り当てる （＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

レコード形式

レコード形式は、

Ｆ	固定長、非ブロック化・非スパン
ＦＢ	固定長、ブロック化・非スパン
ＦＳ	固定長、非ブロック化・スパン
ＦＢＳ	固定長、ブロック化・スパン

のどれかで指定します。省略すると、ＦＢ指定とみなします。

Ｃｌｉｐ、ＮｏＣｌｉｐ　…　ブロック長の処理

非スパン形式（Ｆ、ＦＢ）のとき、ブロック長の切り詰め処理をするかどうかを指定します。

Ｃｌｉｐを指定すると、レコード形式に応じ、

Ｆなら	ブロック長を論理レコード長と同じ長さにする
ＦＢなら	ブロック長を論理レコード長の整数倍に切り詰める

という処理をします。省略すると、このＣｌｉｐ指定になります。

ＮｏＣｌｉｐを指定すると、この処理はしません。指定したブロック長がそのまま有効になります。

ふつうはＣｌｉｐ指定で使います。ＮｏＣｌｉｐ指定を使うのは、特別な場合に限ります。

◆注意　----　ふつうは論理レコード長やブロック長だけ指定すれば十分

Ｅｘｔｅｎｄｅｄの指定、論理レコード長、ブロック長指定、レコード形式、Ｃｌｉｐ、ＮｏＣｌｉｐ指定はどれも省略可能です。また、指定の順序も任意です。空白で区切る必要もありません。ふつうは、

```
／ｆ８０　　（／Ｆｏｒｍ　８０）
／ｆ１００ｓ５００　　（／Ｆｏｒｍ　１００　Ｓｉｚｅ５００）
```

のように、論理レコード長やブロック長だけ指定すれば十分です。

／Ｆｏｒｍ（半拡張形式） ---- ＩＢＭファイルの形式を指定する（半拡張形式）

```
／Ｆｏｒｍ　ＨａｌｆＥｘｔｅｎｄｅｄ
          ［論理レコード長］
            ┌ ブロック長指定 ┐
            └ Ａｕｔｏ       ┘
              ┌ レコード形式 ┐
              └ ＦＢ         ┘
                ┌ Ｃｌｉｐ    ┐
                └ ＮｏＣｌｉｐ ┘
```

新規作成するＩＢＭファイルの形式を指定します。ＨａｌｆＥｘｔｅｎｄｅｄを指定すると、半拡張形式になります。半拡張形式では論理レコード長、ブロック長、レコード形式、および、ブロック長の処理を指定します。

ＨａｌｆＥｘｔｅｎｄｅｄ

半拡張形式にすることを指定します。

論理レコード長

論理レコード長を１～９９９９の範囲の１０進数で指定します。この省略値はコマンドによって違い、省略するとそれぞれ、つぎのようになります。

ＰｕｔＴｅｘｔ（ｐｔ）コマンド	８０バイト
ＰｕｔＤａｔａ（ｐｄ）コマンド	１２８バイト
ＰｕｔＲａｎｄ（ｐｒ）コマンド	２５６バイト

ブロック長指定

ブロック長は、以下に示すように、必要に応じていろいろな指定ができます。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｒｅｃｏｒｄｓ＜ｎ＞	＜ｎ＞レコード分（ブロック化係数による指定）
Ｍａｘ	規約上の最大値（＝トラック容量）を割り当てる
Ａｕｔｏ	論理レコード長やレコード形式に応じて、 適当なブロック長を自動的に割り当てる （＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

レコード形式

レコード形式は、

Ｆ　　固定長、非ブロック化・非スパン
ＦＢ　　固定長、ブロック化・非スパン

のどちらかで指定します。省略すると、ＦＢ指定とみなします。

Ｃｌｉｐ、ＮｏＣｌｉｐ　…　ブロック長の処理

ブロック長の切り詰め処理をするかどうかを指定します。

Ｃｌｉｐを指定すると、レコード形式に応じ、

Ｆなら　　ブロック長を論理レコード長と同じ長さにする
ＦＢなら　　ブロック長を論理レコード長の整数倍に切り詰める

という処理をします。省略すると、このＣｌｉｐ指定になります。

ＮｏＣｌｉｐを指定すると、この処理はしません。指定したブロック長がそのまま有効になります。

ふつうはＣｌｉｐ指定で使います。ＮｏＣｌｉｐ指定を使うのは、特別な場合に限ります。

◆注意　----　ふつうは論理レコード長やブロック長だけ指定すれば十分

論理レコード長、ブロック長指定、レコード形式、Ｃｌｉｐ、ＮｏＣｌｉｐ指定はどれも省略可能です。また、指定の順序も任意です。空白で区切る必要もありません。

ふつうは、

```
／ｆｈｅ８０　　(/Form HalfExtended 80)
／ｆｈｅ１００ｓ５００　　(/Form HalfExtended 100 Size 500)
```

のように、ＨａｌｆＥｘｔｅｎｄｅｄのキーワードと、論理レコード長やブロック長だけ指定すれば十分です。

／Ｆｏｒｍ（基本形式）　----　ＩＢＭファイルの形式を指定する（基本形式）

／Ｆｏｒｍ　Ｐｒｉｍａｒｙ	ブロック長指定 Ａｕｔｏ

新規作成するＩＢＭファイルの形式を指定します。Ｐｒｉｍａｒｙを指定すると基本形式になります。基本形式では「論理レコード」の概念がないので、ブロック長の指定だけが有効です（ブロック＝レコードと考えてください）。

Ｐｒｉｍａｒｙ
基本形式にすることを指定します。

ブロック長指定
ブロック長はつぎのように指定します。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｍａｘ	規約上の最大値（＝セクタ長）を割り当てる
Ａｕｔｏ	セクタ長を割り当てる。結局、Ｍａｘ指定と同じになる
	（＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

◆注意　----　ブロック長をバイト数で指定するときは、Ｓｉｚｅを忘れずに

たとえば全銀フォーマットのＩＢＭファイルを作りたいものとします。その場合、基本形式で、ブロック長（＝レコード長）＝１２０バイトに指定することになります。そのとき、

／ｆｐ１２０　　（／Ｆｏｒｍ　Ｐｒｉｍａｒｙ　１２０）

と指定してしまいがちです。しかし、この場合１２０は論理レコード長にあたり、基本形式のときは無視されてしまいます。基本形式ではブロック長を指定しなければいけないので、この例では

／ｆｐｓ１２０　　（／Ｆｏｒｍ　Ｐｒｉｍａｒｙ　Ｓｉｚｅ１２０）

と指定しなければいけません。

／ＥｘｔｒａＳｐａｃｅ ---- 新規作成後の予備領域の量を指定する

／ＥｘｔｒａＳｐａｃｅ　レコード件数 ／ＥｘｔｒａＳｐａｃｅ　０

新規作成したＩＢＭファイルの予備領域の大きさをレコード数で指定します。

ＩＢＭファイルを新規作成したとき、この／ＥｘｔｒａＳｐａｃｅオプションを使って予備領域をとっておくことができます。省略値は０、すなわち「予備領域はとらない」です。

レコード件数はふつう１０進数で指定しますが、式による指定もできます。式のなかにはつぎの特殊変数が使えます。

＄　　変換ずみのレコード数

たとえば、「１００レコード分余分に領域確保しておきたい」というときは、

／ＥｘｔｒａＳｐａｃｅ　１００

と指定すればよいし、また「全体で２０００件のスペースになるように変換したい」という場合は、

／ＥｘｔｒａＳｐａｃｅ　２０００－＄

などの指定ができます。

なお、この／ＥｘｔｒａＳｐａｃｅオプションを使えば、ＩＢＭファイルのアロケート（領域確保と形式設定）もできます。たとえば、

```
C:¥>ft putxxx nul a:ﾌｧｲﾙ名/extraspace ﾚｺｰﾄﾞ件数/form ﾚｺｰﾄﾞ長 ～ ↓
```

とする要領です。デバイスファイルＮＵＬを０件ファイルとして利用するのです。

／ＢｏｅＢｏｕｎｄａｒｙ ---- ＩＢＭファイルのＢＯＥ境界を指定する

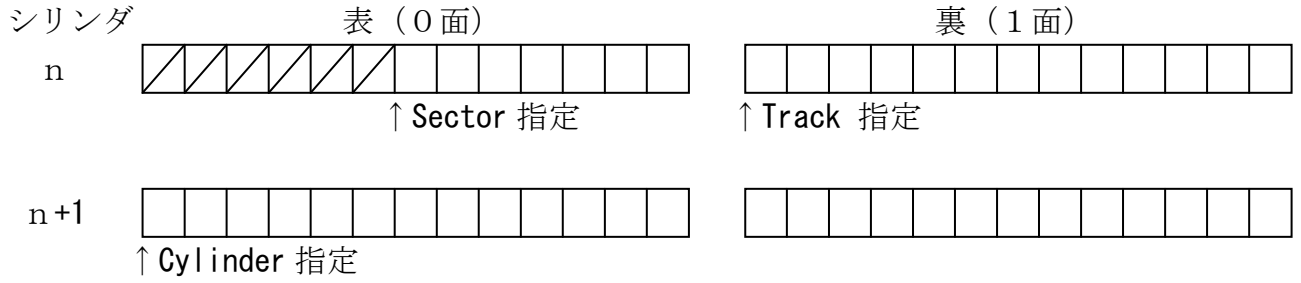
／ＢｏｅＢｏｕｎｄａｒｙ	Ｓｅｃｔｏｒ Ｔｒａｃｋ Ｃｙｌｉｎｄｅｒ

ＩＢＭファイルを新規作成するとき、どんな境界に配置するか（ＢＯＥをどの境界に置くか）を指定します。３つの境界が指定でき、それぞれ、

Ｓｅｃｔｏｒ	前のＩＢＭファイルと隙間を置かないで、セクタ境界で配置
Ｔｒａｃｋ	トラック境界で配置
Ｃｙｌｉｎｄｅｒ	シリンダ境界で配置

という意味になります。／ＢｏｅＢｏｕｎｄａｒｙオプションを省略すると、Ｔｒａｃｋ指定したとみなします。なお、ＢＯＥとは「領域開始アドレス（Beginning Of Extent）」のことです。

言葉ではわかりにくいので、図にしてみます。下図の／の部分まで、直前のＩＢＭファイルで使っているものとします。↑が、それぞれの指定によるＩＢＭファイルの配置される位置です。



Ｓｅｃｔｏｒ指定ならスペース効率がよくなるという利点があります。一方、Ｔｒａｃｋ指定とＣｙｌｉｎｄｅｒ指定は、切りのよい所でファイルがはじまるのでスペースを無駄にしますが、管理しやすい利点があります。また、「ファイルは常にセクタ０１からはじまること」としているシステムもかなりありますが、Ｔｒａｃｋ指定かＣｙｌｉｎｄｅｒ指定でこの要求に対応できます。

なお、日立のシステムではセクタ境界で出力したＩＢＭファイルを扱えません。そこで、Ｆ＊ＴＲＡＮ２００７を日立モードで動かしている場合は、Ｓｅｃｔｏｒ指定しても、自動的にＴｒａｃｋ指定に切り替えられます（エラーにはなりません）。

／ＥｏｅＢｏｕｎｄａｒｙ ---- ＩＢＭファイルのＥＯＥ境界を指定する

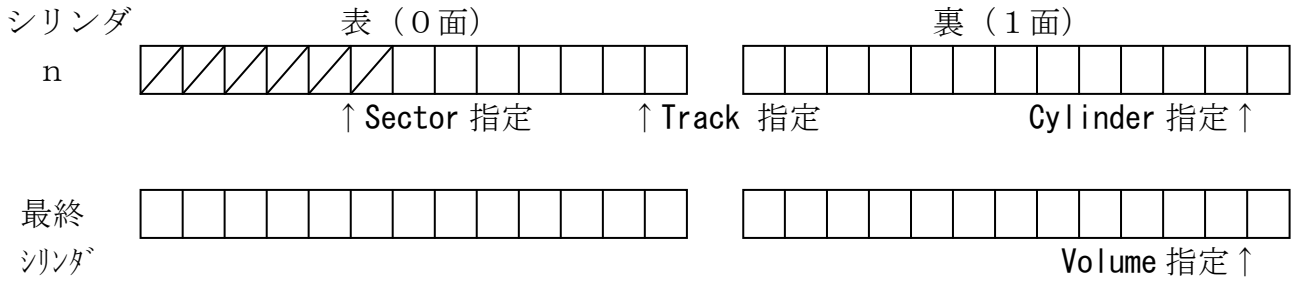
／ＥｏｅＢｏｕｎｄａｒｙ	Ｓｅｃｔｏｒ Ｔｒａｃｋ Ｃｙｌｉｎｄｅｒ Ｖｏｌｕｍｅ ＊

ＩＢＭファイルを新規作成するとき、どんな境界でＩＢＭファイルをおわらせるのか（ＥＯＥをどの境界に置くか）を指定します。４つの境界が指定でき、それぞれ、

Ｓｅｃｔｏｒ	データの書き込みがおわったセクタでＥＯＥにする
Ｔｒａｃｋ	ＥＯＥをトラック境界に切り上げる
Ｃｙｌｉｎｄｅｒ	ＥＯＥをシリンダ境界に切り上げる
Ｖｏｌｕｍｅ	ＥＯＥをディスクのおわりに置く
＊	／ＢｏｅＢｏｕｎｄａｒｙオプションの指定を引き継ぐ

という意味になります。／ＥｏｅＢｏｕｎｄａｒｙオプションを省略すると、＊指定したとみなし、／ＢｏｅＢｏｕｎｄａｒｙオプションの指定を引き継ぎます。なお、ＥＯＥとは「領域終了アドレス（End Of Extent)」のことです。

言葉ではわかりにくいので、図にしてみます。下図の／の部分で、最終ブロックの書き込みがおわっているものとします。↑が、それぞれの指定によるＩＢＭファイルのＥＯＥを置く位置です。



Ｓｅｃｔｏｒ指定なら、つづくＩＢＭファイルもセクタ境界に配置するとスペース効率がよくなるという利点があります。一方、Ｔｒａｃｋ指定やＣｙｌｉｎｄｅｒ指定は無駄なスペースが発生しますが、管理しやすい意味があります。Ｖｏｌｕｍｅ指定は要するに、「ＩＢＭディスクの残りの全領域確保」ということです。

一部に、「ファイルは常にトラック境界でおわること」としているシステムもあります。Ｔｒａｃｋ指定かＣｙｌｉｎｄｅｒ指定で、この要求に対応できます。

／Ｑｕｅｒｙ -------- 変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動変換する

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

１ファイルごとに処理を問い合わせるか否かを指定します。

／Ｑｕｅｒｙオプションを指定すると、Ｆ＊ＴＲＡＮ２００７は１ファイルごとにその処理を実行するか問い合わせます。
つぎのどれかで応答してください。この機能は、比較的小さいファイルが多数あって、そのうちいくつかを選んで変換したいときなどに便利です。

１ファイルの変換

はい（Ｙ）	表示中のファイルを変換する
いいえ（Ｎ）	表示中のファイルは変換しない

２ファイル以上の変換

はい（Ｙ）	表示中のファイルを変換する
すべて変換（Ａ）	全ファイル変換に切り替え、以降のファイルをすべて変換する
いいえ（Ｎ）	表示中のファイルは変換しない
キャンセル	これ以降の変換処理を中断する

／ＮｏＱｕｅｒｙオプションを指定すると、ファイル名の確認なしで自動的に指定のファイルをすべて変換します。こちらが省略値です。

／ＭｕｌｔｉＶｏｌｕｍｅ ---------- マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ -------- マルチボリューム処理機能を無効にする

／ＭｕｌｔｉＶｏｌｕｍｅ ／ＳｉｎｇｌｅＶｏｌｕｍｅ

マルチボリューム処理機能を有効にするか無効にするかを指定します。

／ＭｕｌｔｉＶｏｌｕｍｅオプションを指定すると、マルチボリューム処理機能が有効になります。ＩＢＭディスクがいっぱいになったところで、ＩＢＭディスクの差し替えメッセージを出してマルチボリューム処理に入ります。こちらが省略値です。

／ＳｉｎｇｌｅＶｏｌｕｍｅオプションを指定すると、マルチボリューム処理機能は抑止されます。ＩＢＭディスクがいっぱいになったら、変換がエラー終了します。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ ---- ボリューム順序番号の出力の制御

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	［Ｔｈｅｎ］	Ｂｌａｎｋ ０１

このオプションでは、ファイルラベルのボリューム順序番号の欄をどうするか制御することができます。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋの場合

／ＭｕｌｔｉＶｏｌｕｍｅオプションを指定して、結果的にＩＢＭファイルがシングルボリュームになったときや、／ＳｉｎｇｌｅＶｏｌｕｍｅオプションが指定されたときに、ボリューム順序番号欄を空白にします。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　０１の場合

／ＭｕｌｔｉＶｏｌｕｍｅオプション及び／ＳｉｎｇｌｅＶｏｌｕｍｅのどちらが指定されていた場合でも、ファイルラベルのボリューム順序番号の欄にボリューム順序番号２桁を書き込みます。

／ＥＯＦ -------- ＥＯＦコードを検査する
／ＮｏＥＯＦ ---- ＥＯＦコードを検査しない

```
／ＥＯＦ
／ＮｏＥＯＦ
```

ＥＯＦコード（１ＡＨ）を検査するか否かを指定します。

／ＥＯＦオプションを指定するとＥＯＦコードを検査し、ＥＯＦコードが現れたら変換を終了します。

／ＮｏＥＯＦオプションを指定すると、ＥＯＦコードを検査しません。単なるデータとして扱います。

このオプションは、つぎに示すようにコマンドによって省略値が違います。

コマンド	省略値
ＰｕｔＴｅｘｔ（ｐｔ）	／ＥＯＦ
ＰｕｔＤａｔａ（ｐｄ）	／ＥＯＦ
ＰｕｔＲａｎｄ（ｐｒ）	／ＮｏＥＯＦ

最後のＰｕｔＲａｎｄ（ｐｒ）コマンドだけ省略値が／ＮｏＥＯＦになっているのは、このコマンドではバイナリデータも扱うからです。

■使用例

例１）１ファイル変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＳＡＭＰＬＥにＡＮＫ変換します。

```
C:¥>ft puttext c:sample.txt a: ↓
```

例２）１ファイル変換する。ただし、すでに同じ名前のＩＢＭファイルがある

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＳＡＭＰＬＥにＡＮＫ変換します。ただし、すでにＳＡＭＰＬＥという名前のＩＢＭファイルがあるので、置き替えを指示します。

```
C:¥>ft puttext c:sample.txt a:/rep ↓          (/REPlace)
```

例３）全ファイル変換する

ドライブＣ：のすべてのＷｉｎｄｏｗｓファイルを、ドライブＡ：の同じ名前のＩＢＭファイルにＡＮＫ変換します。

```
C:¥>ft puttext c:*.* a: ↓
```

例４）拡張子が．ＴＸＴのファイルをすべて変換する

ドライブＣ：の、拡張子．ＴＸＴがつくすべてのＷｉｎｄｏｗｓファイルを、ドライブＡ：のＩＢＭファイルにＡＮＫ変換します。

```
C:¥>ft puttext c:*.txt a: ↓
```

例５）拡張子がなく、名前がＵＭＡＳｘｘｘｘのファイルをすべて変換する

ドライブＣ：の、拡張子がなく、ファイル名がＵＭＡＳｘｘｘｘというパターンのＷｉｎｄｏｗｓファイルをすべてＩＢＭファイルにＡＮＫ変換します。

```
C:¥>ft puttext c:umas* a: ↓
```

例６）上書きする

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＤＡＴＡに上書きします。

```
C:¥>ft puttext c:sample.txt a:data/old ↓     (/OLD)
```

例７）拡張形式・非ブロック化ファイルにする

ドライブＣ：のＷｉｎｄｏｗｓファイルＥＲＲＬＯＧ．ＰＲＮを、ドライブＡ：のＩＢＭファイルに変換します。論理レコード長、ブロック長ともに１０００バイトの拡張形式・非ブロック化ファイルにします。

```
C:¥>ft putdata c:errlog.prn a:/f1000f ↓      (/Form 1000 F)
```

例８）拡張形式・ブロック化ファイルにする

ドライブＣ：のＷｉｎｄｏｗｓファイルＥＲＲＬＯＧ．ＰＲＮをドライブＡ：のＩＢＭファイルに変換します。論理レコード長＝１２８、ブロック長＝１０２４バイトの拡張形式・ブロック化ファイルにします。

```
C:¥>ft putdata c:errlog.prn c:/f128s1024 ↓  (/Form 128 Size1024)
```

例９）ブロック化係数を８にする

例８と同じ処理を、ブロック長をブロック化係数で指定する方法で変換します。

```
C:¥>ft putdata c:errlog.prn a:/f128r8 ↓      (/Form 128 Records8)
```

例 10）基本形式のファイルにする

ドライブＣ：のＷｉｎｄｏｗｓファイルＥＲＲＬＯＧ．ＰＲＮをドライブＡ：のＩＢＭファイルに変換します。ブロック長（＝レコード長）が１２８バイトの基本形式のファイルにします。

```
C:¥>ft putdata c:errlog.prn a:/fps128 ↓      (/Form Primary Size128)
```

例 11）ファイルをアロケートする

ドライブＡ：に、レコード長＝２００バイト、拡張形式・非ブロック化で、レコード件数５００件分のＩＢＭファイルＳＨＡＩＮをアロケートします。

```
C:¥>ft putdata nul a:shain /f200f/es500 ↓   (/From 200 F /ExtraSpace 500)
```

例 12）ファイルをセクタ境界に配置し、スペース効率を上げる

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＨＯＲＴｎｎ．ＤＡＴを、ドライブＡ：のＩＢＭファイルＳＨＯＲＴｎｎに変換します。ｎｎは０１からはじまる通し番号です。そのとき、スペース効率を上げるために、ファイルをセクタ境界に配置します。このとき、ＥＯＥ境界もセクタ境界になります。

```
C:¥>ft putdata c:short*.dat a:/bbs ↓         (/BoeBoundary Sector)
```

例 13）ボリューム全体を割り当てる

ドライブＣ：のＷｉｎｄｏｗｓファイルＤＡＴＡ．ＤＡＴを、ドライブＡ：のＩＢＭファイルＤＡＴＡに変換します。元々、そのＩＢＭファイルは空だったものとします。そのとき、このファイルでボリュームがすべて専有されるようにします。ＥＯＥ境界がボリューム境界になります。

```
C:¥>ft putdata c:data.dat a:/ebv ↓          (/EoeBoundary Volume)
```

例 14）空き領域管理ラベルを作る

例１３と似た方法で、空き領域管理ラベルＤＡＴＡを作ります。ＥＯＥ境界がボリューム境界になります。

```
C:¥>ft putdata nul a:data/fp/ebv ↓          (/From Primary/EoeBoundary Volume)
```

例 15）ＩＢＭディスクがパンクしたら、エラーで終了させる

運用上の都合で、１枚のＩＢＭディスクがいっぱいになったらマルチボリューム処理に入るのを禁止し、パンクした時点でエラー終了させます。

```
C:¥>ft putdata c:somedata a:/sv ↓           (/SingleVolume)
```

例 16）テキストファイル変換で、ＥＯＦコードを一般文字扱いさせる

テキストファイル変換やデータファイル変換で、ＥＯＦコード（１ＡＨ）を一般の文字扱いさせたい場合が、ごくまれにあります。

```
C:¥>ft puttext c:file02.txt a:/neof ↓       (/NoEOF)
```

例 17）ランダムファイル変換で、ＥＯＦコードをＥＯＦ扱いさせる

ランダムファイル変換でも、ＥＯＦコード（１ＡＨ）をＥＯＦとして扱いたい場合が、よくあります。

```
C:¥>ft putrand c:file03.ran a:/eof ↓        (/EOF)
```

■注意事項

差し替えメッセージが出たら

ＩＢＭディスクの差し替えをうながす、

> 順序番号 nn のＩＢＭディスクに差し替えてください.

または、

> 次のＩＢＭディスクに差し替えてください.

というメッセージが出たら、今のＩＢＭディスクにはこれ以上書き込めないことを意味しています。メッセージに従ってつぎのＩＢＭディスクに差し替えて、ＯＫボタンをクリックしてください。処理を続行します。詳しくは解説編の「マルチボリューム」の節を参照してください。

ＩＢＭファイル名の重複を避けるには

ワイルドカード文字を使って複数ファイルを変換するときは、ＩＢＭファイル側にドライブ名だけを指定すると、同じ名前のＩＢＭファイルがいくつも生成されることがあります。

```
ＨＡＮＢＡＩ．ＣＢＬ  →  ＨＡＮＢＡＩ
ＨＡＮＢＡＩ．ＤＡＴ  →  ＨＡＮＢＡＩ   } 重複する
ＨＡＮＢＡＩ．ＣＰＹ  →  ＨＡＮＢＡＩ
```

こんなときは、ＩＢＭファイル側をｄ：＊＊やｄ：＊＃＊のような形式で指定してください。そうすれば、拡張子もＩＢＭファイル名に含まれるようになるので、ＩＢＭファイル名が重複することはほとんどなくなります。

```
ＨＡＮＢＡＩ．ＣＢＬ  →  ＨＡＮＢＡＩＣＢＬ
ＨＡＮＢＡＩ．ＤＡＴ  →  ＨＡＮＢＡＩＤＡＴ   } 重複しない
ＨＡＮＢＡＩ．ＣＰＹ  →  ＨＡＮＢＡＩＣＰＹ
```

```
ＳＩＧ．ＭＡＣ      →  ＳＩＧ#ＭＡＣ  ┐ 重複しない
ＳＩＧＭＡ．Ｃ      →  ＳＩＧＭＡ#Ｃ  ┘
```

２．８　ＰｕｔＴｅｘｔ（ｐｔ）　プット・テキスト

Ｗｉｎ→ＩＢＭ　テキストファイル変換

ＰｕｔＴｅｘｔ(ｐｔ)コマンドは、ソースプログラムのようなＷｉｎｄｏｗｓファイルをＩＢＭファイルに変換するコマンドです。Ｗｉｎｄｏｗｓファイルはテキストファイル（改行コードつき）でなければなりません。改行コードでレコードのおわりを検出するので、可変長でかまいません。ＩＢＭファイルは固定長になります。

■コマンド形式

コマンド	パラメータ
ＰｕｔＴｅｘｔ ｐｔ	［Ｗｉｎｄｏｗｓファイル　ＩＢＭファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ＩＢＭファイル

ＷｉｎｄｏｗｓファイルとＩＢＭファイルの指定方法は、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＰｕｔＴｅｘｔ(ｐｔ)コマンドには、つぎのオプションが指定できます。

ＰｕｔＴｅｘｔ(ｐｔ)コマンド固有のオプション

／Ｃｏｄｅ	ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない

Ｐｕｔ系コマンド共通のオプション

／ＮＥＷ	新規ファイルとして作成する
／ＯＬＤ	既存ファイルに出力する
／ＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルは置き替えない
／Ｆｏｒｍ	ＩＢＭファイルの形式を指定する 拡張形式・半拡張形式・基本形式
／ＥｘｔｒａＳｐａｃｅ	新規作成後の予備領域の量を指定する
／ＢｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＢＯＥ境界を指定する
／ＥｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＥＯＥ境界を指定する
／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名を確認せず自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	ボリューム順序番号出力の制御
／ＥＯＦ	ＥＯＦコードを検査する
／ＮｏＥＯＦ	ＥＯＦコードを検査しない

これらのオプションのうち、／Ｃｏｄｅオプション、／Ｆｏｒｍオプションがとくに重要です。

Ｐｕｔ系コマンド共通のオプションは、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＰｕｔＴｅｘｔ（ｐｔ）コマンド固有のオプション

ＰｕｔＴｅｘｔ（ｐｔ）コマンドに固有のオプションを説明します。

／Ｃｏｄｅ ---- ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する

／Ｃｏｄｅ	Ａｎｋ Ｋａｎｊｉｍｉｘ
／Ｃｏｄｅ	Ａｎｋ

ＩＢＭファイル側のコード系を指定します。

Ａｎｋ指定

Ａｎｋ指定すると、すべてＡＮＫデータとして変換します。これが省略値です。

Ｗｉｎｄｏｗｓ側		ＩＢＭ側
ＪＩＳ８／ＡＳＣＩＩ	→	ＪＩＳ８／ＡＳＣＩＩ ＥＢＣＤＩＣ（カタカナ） ＥＢＣＤＩＣ（英小文字）

の３とおりの変換が可能です。あらかじめ、ＩＢＭファイル側のコード系を設定しておかなければいけません（ふつう、セットアップ時に１回だけ行います）。漢字がまじっているときは、つぎのＫａｎｊｉｍｉｘ指定を使ってください。

Ｋａｎｊｉｍｉｘ指定

ＡＮＫ・漢字混在でＫＩ／ＫＯつきに変換するとき、この指定をします。あらかじめ、変換設定の漢字変換方式の設定で、適切な漢字変換方式の割り当てをしておかなければいけません（ふつう、セットアップ時に１回だけ行います）。

／ＴａｂＥｘｐａｎｄ -------- タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ ---- タブ拡張しない

／ＴａｂＥｘｐａｎｄ　〔タブ間隔　８〕 ／ＮｏＴａｂＥｘｐａｎｄ

タブ拡張の有無と、タブ拡張するときのタブ間隔を指定します。タブ拡張とは、ＴＡＢ（０９Ｈ）をつぎのタブ位置の直前までの連続スペースに展開することです。

／ＴａｂＥｘｐａｎｄオプションを指定するとタブ拡張します。タブ間隔は２～２５５の範囲で指定し、それがレコードのおわりまで繰り返し適用されます。タブ間隔を省略すると標準のタブ間隔（８桁きざみ）になります。

／ＮｏＴａｂＥｘｐａｎｄオプションを指定すればタブ拡張は行いません。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｃｏｄｅ　Ａｎｋ	ＡＮＫ変換する
／ＴａｂＥｘｐａｎｄ	８桁きざみでタブ拡張する
／ＮＥＷ	新規ファイルとして作成する
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替えない（中断）
／Ｆｏｒｍ　Ｅｘｔｅｎｄｅｄ　８０　Ａｕｔｏ　Ｃｌｉｐ	レコード長＝８０バイトの、拡張形式で適当にブロック化したファイルを作る
／ＥｘｔｒａＳｐａｃｅ　０	予備領域をとらない
／ＢｏｅＢｏｕｎｄａｒｙ　Ｔｒａｃｋ	ＩＢＭファイルをトラック境界に配置する
／ＥｏｅＢｏｕｎｄａｒｙ　＊	ＥＯＥ境界は　／ＢｏｅＢｏｕｎｄａｒｙオプションの指定に合わせる
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋ	シングルボリューム時、ボリューム順序番号を空白にする
／ＮｏＱｕｅｒｙ	ファイル名を確認せず自動変換する
／ＥＯＦ	ＥＯＦコードを検査する

と指定されたものとして、テキストファイル変換を行います。

■使用例

例１）ＡＮＫ変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＳＡＭＰＬＥに変換します。内容はＡＮＫのみとし、ＡＮＫ変換します。

```
C:¥>ft puttext c:sample.txt a: ↓
```

例２）ＡＮＫ・漢字まじり変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＳＡＭＰＬＥに変換します。漢字が使われています。

```
C:¥>ft puttext c:sample.txt a:/ck ↓          (/Code KanjiMix)
```

例３）ＫＩ／ＫＯが入るのを見込んでＡＮＫ・漢字まじり変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＲＥＡＤＭＥ．ＴＸＴを、ドライブＡ：のＩＢＭファイルＲＥＡＤＭＥに変換します。内容は日本語文書とします。

ＫＩ／ＫＯが挿入されることを見込んで、レコード長が１２８バイトでブロック化された拡張形式のファイルにします。

```
C:¥>ft puttext c:readme.txt a:/ck/f128 ↓     (/Code KanjiMix /Form 128)
```

例４）標準の８桁きざみでタブ拡張する

ドライブＣ:のＣ言語のソースプログラムＨＥＬＬＯ.Ｃを、標準の８桁きざみでタブ拡張し、ドライブＡ：のＩＢＭファイルＨＥＬＬＯに変換します。形式は基本形式で、ブロック長（＝レコード長）を８０バイトにします。

```
C:¥>ft puttext c:hello.c a:/te/fps80 ↓       (/TabExpand /Form Primary Size80)
```

例５）４桁きざみでタブ拡張する

ドライブＣ：のＣ言語のソースプログラムＨＥＬＬＯ．Ｃを、４桁きざみでタブ拡張し、ドライブＡ：のＩＢＭファイルＨＥＬＬＯに変換します。

```
C:¥>ft puttext c:hello.c a:/te4 ↓            (/TabExpand 4)
```

■注意事項

漢字があるときは漢字変換方式の割り当てと／ＣＫを忘れずに

漢字が入っているときは、あらかじめデータ交換の相手のシステムに合わせて、

漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、変換のとき、

／ＣｏｄｅオプションにＫａｎｊｉｍｉｘ指定（／ＣＫ指定）するのを忘れがち

なので、注意してください。

タブ拡張について

ふつう、タブ拡張は必須です。多くの場合、ＩＢＭファイルを受けとるシステムのほうでタブをサポートしていないか、サポートしていてもタブ間隔の設定が異なっています。

あふれた分は捨てられる

Ｗｉｎｄｏｗｓファイルの１行が、変換後ＩＢＭファイルの１レコードに入り切れないときは、あふれた分が切り捨てられます。そのときは、もっと大きなレコード長を指定して再変換してください。とくに、ＡＮＫ・漢字まじり変換をしたとき、ＫＩ／ＫＯが挿入されて、変換後１行の長さが増えることに注意してください。

改行コードがないと、１件だけの変換になる

Ｗｉｎｄｏｗｓファイルの各レコードの末尾に改行コードがついていないと、先頭の１件だけ変換されます。この場合は、ＰｕｔＲａｎｄ（ｐｒ）コマンドを使うべきです。

その他の注意事項

「Ｐｕｔ系コマンド」の節を参照してください。

２．９ ＰｕｔＤａｔａ（ｐｄ）　プット・データ

Ｗｉｎ→ＩＢＭ　データファイル変換

ＰｕｔＤａｔａ(ｐｄ)コマンドは､ＷｉｎｄｏｗｓのデータファイルをＩＢＭファイルに変換するコマンドです｡Ｗｉｎｄｏｗｓファイルはプリント形式のファイルかデリミタ形式のファイルでなければいけません｡ＡＮＫデータだけの単純な変換､項目別の変換などが行えます｡なお､ＩＢＭファイルは必ず固定長・固定欄になります。

■コマンド形式

コマンド	パラメータ
ＰｕｔＤａｔａ ｐｄ	［Ｗｉｎｄｏｗｓファイル　ＩＢＭファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ＩＢＭファイル

ＷｉｎｄｏｗｓファイルとＩＢＭファイルの指定方法は、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＰｕｔＤａｔａ(ｐｄ)コマンドには、つぎのオプションが指定できます。

ＰｕｔＤａｔａ(ｐｄ)コマンド固有のオプション

／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ	引用符はずしの詳細を指定する
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＳＱｕｅｅｚｅ	空行を無視する
／ＮｏＳＱｕｅｅｚｅ	空行を無視しない
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

Ｐｕｔ系コマンド共通のオプション

／ＮＥＷ	新規ファイルとして作成する
／ＯＬＤ	既存ファイルに出力する
／ＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルは置き替えない
／Ｆｏｒｍ	ＩＢＭファイルの形式を指定する 拡張形式・半拡張形式・基本形式
／ＥｘｔｒａＳｐａｃｅ	新規作成後の予備領域の量を指定する
／ＢｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＢＯＥ境界を指定する
／ＥｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＥＯＥ境界を指定する
／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名を確認せず自動変換する
／ＭｕｌｔｉＶｏｌｕｍｅ	マルチボリューム処理機能を有効にする
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	ボリューム順序番号出力の制御
／ＥＯＦ	ＥＯＦコードを検査する
／ＮｏＥＯＦ	ＥＯＦコードを検査しない

これらのオプションのうち、／Ｄｅｌｉｍｉｔｅｄオプション、／ＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＭＡＰオプション、／Ｆｏｒｍオプションがとくに重要です。

Ｐｕｔ系コマンド共通のオプションは、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＰｕｔＤａｔａ(ｐｄ)コマンド固有のオプション

ＰｕｔＤａｔａ(ｐｄ)コマンドに固有のオプションを説明します。

／Ｄｅｌｉｍｉｔｅｄ ---------- デリミタ形式のファイルを変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ ---- プリント形式のファイルを変換する

```
／Ｄｅｌｉｍｉｔｅｄ ( Ｂｙ　ＣＯＭｍａ
                      Ｂｙ　ＴＡＢ
                      Ｂｙ　ＳＰａｃｅ )
／ＮｏｎＤｅｌｉｍｉｔｅｄ
```

変換するＷｉｎｄｏｗｓファイルがプリント形式かデリミタ形式か､デリミタ形式ならコンマ区切り・タブ区切り・スペース区切りのどれなのかを指定します。

プリント形式からの変換

／ＮｏｎＤｅｌｉｍｉｔｅｄオプションを指定すると、デリミタ（区切り文字）なしのプリント形式のファイル（固定長テキストファイル〔ＳＤＦ形式〕。ただし、最終項目だけ可変長でもよい）を変換できます。

◆注意　----　プリント形式のときは、／ＭＡＰでコンマを入れないこと

／ＮｏｎＤｅｌｉｍｉｔｅｄオプションを指定し､プリント形式のファイルを変換するときは、／ＭＡＰオプションでデリミタ検出＝コンマを使ってはいけません。

デリミタ形式からの変換

／Ｄｅｌｉｍｉｔｅｄオプションを指定すると、デリミタ形式（区切り文字つき）のテキストファイルを変換できます。区切り文字の種類によって、さらに細かい形式が決まります。Ｂｙ指定で、

Ｂｙ　ＣＯＭｍａ　　コンマ区切り形式（ＣＳＶ形式、Ｋ３形式）
Ｂｙ　ＴＡＢ　　タブ区切り形式（ＴＡＢ＝０９Ｈ）
Ｂｙ　ＳＰａｃｅ　　スペース区切り形式（ＳＰ＝２０Ｈ）

の３つのなかから指定できます。後述の／ＭＡＰオプションで、デリミタ検出＝コンマ（,）を指定したところに、上記の区切り文字があるとみなされます。

◆注意　----　デリミタ形式のときは、／ＭＡＰで項目ごとにコンマを入れること

／Ｄｅｌｉｍｉｔｅｄオプションを指定し、デリミタ形式のファイルを変換するときは、／ＭＡＰオプションで項目ごとにデリミタ検出＝コンマ（,）を入れてください。コンマが出てくるたびに、Ｂｙで指定した区切り文字を項目の区切りとして判定します。

／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ ---- 引用符はずしの詳細を指定する

```
／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ

    （ＱｕｏｔａｔｉｏｎＭａｒｋ　　Ｎｏｎｅ
      ＱｕｏｔａｔｉｏｎＭａｒｋ　　Ｓｉｎｇｌｅ
      ＱｕｏｔａｔｉｏｎＭａｒｋ　　Ｄｏｕｂｌｅ）

    （ＦｉｅｌｄＧｅｔｔｅｒ　　ＯｌｄＦｔｒａｎ
      ＦｉｅｌｄＧｅｔｔｅｒ　　ＮｅｗＦｔｒａｎ
      ＦｉｅｌｄＧｅｔｔｅｒ　　ＭｓＥｘｃｅｌ
      ＦｉｅｌｄＧｅｔｔｅｒ　　ＭｓＡｃｃｅｓｓ）
```

変換するＷｉｎｄｏｗｓファイルがデリミタ形式で、引用符でくくられた項目があるときにその引用符のはずし方を指定します。このオプションは、／Ｄｅｌｉｍｉｔｅｄオプションが指定されているときにのみ指定できます。

／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌオプションではＷｉｎｄｏｗｓファイルの引用符の種類をＱｕｏｔａｔｉｏｎＭａｒｋで指定します。

ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｎｏｎｅ	なし
ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｓｉｎｇｌｅ	単一引用符（’）
ＱｕｏｔａｔｉｏｎＭａｒｋ	Ｄｏｕｂｌｅ	２重引用符（”）

ＱｕｏｔａｔｉｏｎＭａｒｋ　Ｎｏｎｅを指定したときは、引用符でくくられた項目があっても、引用符をはずさずにそのまま出力します。省略値はＱｕｏｔａｔｉｏｎＭａｒｋ　Ｄｏｕｂｌｅです。

◆注意　----　Ｍａｐ文の引用符記号は、常に二重引用符「”」

ＱｕｏｔａｔｉｏｎＭａｒｋ　Ｓｉｎｇｌｅ指定時も、Ｍａｐ文において引用符はずしに使う記号は、二重引用符「”（半角）」ですので注意してください。

◆注意　----　引用符の種類の指定は一種類のみ

Ｗｉｎｄｏｗｓファイルに単一引用符（’）でくくられた項目と２重引用符（”）でくくられた項目が混在していても、どちらか片方の引用符しか指定することができません。

さらに、引用符つき項目の取得方法について、詳細を

ＦｉｅｌｄＧｅｔｔｅｒ	ＯｌｄＦｔｒａｎ	従来のＦ＊ＴＲＡＮ互換
ＦｉｅｌｄＧｅｔｔｅｒ	ＮｅｗＦｒａｎ	新Ｆ＊ＴＲＡＮ風
ＦｉｅｌｄＧｅｔｔｅｒ	ＭｓＥｘｃｅｌ	ＭＳ－Ｅｘｃｅｌ風
ＦｉｅｌｄＧｅｔｔｅｒ	ＭｓＡｃｃｅｓｓ	ＭＳ－Ａｃｃｅｓｓ風

で指定できます。ＭｓＥｘｃｅｌとＭｓＡｃｃｅｓｓはそれぞれ、ＷｉｎｄｏｗｓファイルがＥｘｃｅｌ、Ａｃｃｅｓｓで出力したコンマ区切り（ＣＳＶ）形式ファイルのときに、指定してください。ＯｌｄＦｔｒａｎ以外は、引用符でくくられた項目の中に、連続した２個の引用符があるときは、それを単なる文字データとして１つの引用符に変換します。省略値はＮｅｗＦｔｒａｎです。

◆注意　----　「ＦｉｅｌｄＧｅｔｔｅｒ」指定の存在について

ＦｉｅｌｄＧｅｔｔｅｒはＰｕｔ変換のコマンドオプションの指定の一つです。従って、ホストからのデータをＥｘｃｅｌやＡｃｃｅｓｓで取り込む際には関わって来ません。

例えば､アプリケーションによりＡｃｃｅｓｓに読み込ませるために書き出したＣＳＶファイルを、なんらかの理由によりホストに持っていこうとした場合に、このＦｉｅｌｄＧｅｔｔｅｒが活躍します。

「／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」コマンドオプションを省略した場合は

```
/QuotingContlol QuotationMark Double FieldGetter NewFtran
```

という指定をした扱いになります。

／ＴａｂＥｘｐａｎｄ -------- タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ ---- タブ拡張しない

```
／ＴａｂＥｘｐａｎｄ　〔タブ間隔
                        ８　〕
／ＮｏＴａｂＥｘｐａｎｄ
```

タブ拡張の有無と、タブ拡張するときのタブ間隔を指定します。タブ拡張とは、ＴＡＢ（０９Ｈ）をつぎのタブ位置の直前までの連続スペースに展開することです。

／ＴａｂＥｘｐａｎｄオプションを指定するとタブ拡張します。タブ間隔は２～２５５の範囲で指定し、それがレコードのおわりまで繰り返し適用されます。タブ間隔を省略すると標準のタブ間隔（８桁きざみ）になります。

／ＮｏＴａｂＥｘｐａｎｄオプションを指定すればタブ拡張は行いません。

／ＳＱｕｅｅｚｅ -------- 空行を無視する
／ＮｏＳＱｕｅｅｚｅ ---- 空行を無視しない

```
／ＳＱｕｅｅｚｅ
／ＮｏＳＱｕｅｅｚｅ
```

Ｗｉｎｄｏｗｓファイルの空行を無視するかどうかを指定します。
／ＳＱｕｅｅｚｅオプションを指定すると、空行を無視します。
／ＮｏＳＱｕｅｅｚｅオプションを指定すると、空行も１レコードとして変換します。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／ＭＡＰ	Ａｎｋ変換 Ｋａｎｊｉ変換 ＡｎｋｉＺｅ変換 ＫａｎｊｉＺｅ変換 ＫａｎｊｉＭｉｘ変換 ＵｓｅｒＡ変換／ＵｓｅｒＢ変換 Ｎｕｍｅｒｉｃ変換 Ｂｉｎａｒｙ変換 漢字イン挿入／漢字アウト挿入 ＤｉｓｐＺｏｎｅ変換（Ｚｏｎｅ変換） ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換） ＺｏｎｅＤｉｓｐ変換 ＺｏｎｅＺｏｎｅ変換 ＺｏｎｅＰａｃｋ変換 ＤｉｓｐＢｉｎ変換 ＺｏｎｅＢｉｎ変換 ＴｏＤｉｓｐ変換 ＴｏＺｏｎｅ変換 ＴｏＰａｃｋ変換 ＴｏＢｉｎ変換 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 デリミタ検出／引用符はずし	・・・

桁移動／項目移動／入力スキップ／出力スキップ

／ＭＡＰ　　Ａｎｋ

この／ＭＡＰオプションで細かい変換方法を指示します。

ＰｕｔＤａｔａ(ｐｄ)コマンドの／ＭＡＰオプションの場合、プリント形式とデリミタ形式では指定の方法と考え方がかなり違うので、この２つを必要に応じて分けて説明することにします。

デリミタ形式の場合には、Ｗｉｎｄｏｗｓ側（入力項目）が可変長で、ＩＢＭ側（出力項目）が固定長だという点に留意してください。

◆注意　----　マルチレコードレイアウト等の指定について

／ＭＡＰオプションには、マルチレコードレイアウト等の指定ができるＡｔｌａｓ構文を記述することもできます。詳細は、マルチレコード編のマニュアルを参照してください。

Ａｎｋ変換

Ａｎｋ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ項目を変換します。どのコード変換が行われるかは、変換設定のＡＮＫコードの設定で決まります（ふつう、セットアップ時に１回だけ行います）。

プリント形式の場合

入力幅は、　　ａ２０　　のように、１０進のバイト数で指定します。また、

レコード全体をＡｎｋ変換するときには ---- ／ｍａｐ　ａ
最終項目をＡｎｋ変換するときには -------- ／ｍａｐ　・・・　ａ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａ２０　　という指定は、
ａ２０：２０　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のＡＮＫ項目のおわりにスペース（２０Ｈ／４０Ｈ）が詰められます。

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

ａ：２０　　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（出力）の残りバイト数

プリント形式・デリミタ形式共通

入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

Ａｎｋ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡＮＫ形式、最大２５６文字で、半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ａ［'ＡＮＫ'］　のように、［　］でくくります。

出力幅は省略できます。出力幅を省略すると、「正確な出力に必要な最小限の幅」とみなします。出力幅の指定は、プリント形式の場合、デリミタ形式の場合を参照してください。

ＡｎｋｉＺｅ（ＡＮＫ化）変換

ＡｎｋｉＺｅ　［入力幅｜＊］［：出力幅］

Ｗｉｎｄｏｗｓ側（入力）文字列をできる限りＡＮＫ（半角）に変換します。どのコード変換が行われるかは、変換設定のＡＮＫコードの設定で決まります（ふつう、セットアップ時に１回だけ行います）。

プリント形式の場合

入力幅は、　ａｚ２０　のように、１０進のバイト数で指定します。また、

レコード全体をＡｎｋｉＺｅ変換するときには ---- ／ｍａｐ　ａｚ
最終項目をＡｎｋｉＺｅ変換するときには -------- ／ｍａｐ　・・・　ａｚ

のようにして、入力幅を省略できます。
式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａｚ２０　という指定は、
ａｚ２０：２０　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のＡＮＫ項目のおわりにスペースが詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　ＩＢＭ側（出力）のレコード長
＊　ＩＢＭ側（出力）の残りバイト数

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

ａｚ：２０　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（出力）の残りバイト数

Ｋａｎｊｉ変換

Ｋａｎｊｉ［入力幅｜＊｜定数］［：出力幅］

漢字項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に１回だけ行います)。

プリント形式の場合

入力幅は、　ｋ１６　のように、１０進のバイト数で指定します（漢字の文字数ではありません)。また、

レコード全体をＫａｎｊｉ変換するときは ---- ／ｍａｐ　ｋ
最終項目をＫａｎｊｉ変換するときは -------- ／ｍａｐ　・・・　ｋ

のようにして、入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋ１６　　という指定は、
ｋ１６：１６　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません)。逆に、項目長を伸ばすと、ＩＢＭ側の漢字項目のおわりに漢字変換方式に設定されている漢字スペースが詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　ＩＢＭ側（出力）のレコード長
＊　ＩＢＭ側（出力）の残りバイト数

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

ｋ：２０　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　ＩＢＭ側（出力）の残りバイト数

プリント形式・デリミタ形式共通

入力幅の代わりに定数を指定することもできます。その場合、指定した漢字形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

Ｋａｎｊｉ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列は漢字形式、最大２５６文字で、半角(')でくくる
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　**ｋ［'漢字'］**　のように、［　］でくくります。

出力幅は省略できます。出力幅を省略すると、「正確な出力に必要な最小限の幅」とみなします。出力幅の指定は、プリント形式の場合、デリミタ形式の場合を参照してください。

ＫａｎｊｉＺｅ（漢字化）変換

ＫａｎｊｉＺｅ　［入力幅｜＊］［：出力幅］

Ｗｉｎｄｏｗｓ側（入力）文字列をできる限り漢字（全角）に変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に１回だけ行います）。

プリント形式の場合

入力幅は、　ｋｚ１６　のように、１０進のバイト数で指定します（漢字の文字数ではありません）。また、

レコード全体をＫａｎｊｉＺｅ変換するときは ---- ／ｍａｐ　ｋｚ
最終項目をＫａｎｊｉＺｅ変換するときは -------- ／ｍａｐ　・・・　ｋｚ

のようにして、入力幅を省略できます。
式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。
出力幅も省略できます。出力幅を省略すると入力幅の２倍が指定されます。

という指定になります。ただし、入力幅に指定された値より小さくなることはありません。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません）。逆に、項目長を伸ばすと、ＩＢＭ側の漢字項目のおわりに漢字変換方式に設定されている漢字スペースが詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

　　ｋｚ：２０　　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

　　＊　　ＩＢＭ側（出力）の残りバイト数

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ・漢字まじり項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に１回だけ行います)。変換後、漢字文字列の前後にはＫＩ／ＫＯが挿入されます。この、ＫＩ／ＫＯ挿入のタイミングも漢字変換方式の設定で決まります。そのため、オーバーフローの危険性があるので、出力幅の指定には注意しなければいけません。

プリント形式の場合

入力幅は、　　ｋｍ３０　　のように、１０進のバイト数で指定します。また、

レコード全体をＫａｎｊｉＭｉｘ変換するときは ---- ／ｍａｐ　ｋｍ
最終項目をＫａｎｊｉＭｉｘ変換するときは -------- ／ｍａｐ　・・・　ｋｍ

のようにして入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋｍ３２　　という指定は、
ｋｍ３２：３２　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
ふつうは、ＫＩ／ＫＯが挿入されてデータ長が長くなる分を加算した出力幅を指定します。オーバーフローすると、ＡＮＫ・漢字まじり項目のおわりのほうが切り捨てられます。ただし、漢字モードのままおわることはありません。また、ＫＯが途中で切れることもありません。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

ｋｍ：２０　　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（出力）の残りバイト数

プリント形式・デリミタ形式共通

入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ・漢字まじり形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

ＫａｎｊｉＭｉｘ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡＮＫ形式、最大２５６文字で、半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　**ｋｍ［'ＡＮＫ・漢字'］**　のように、［　］でくくります。

出力幅は省略できます。出力幅を省略すると、「正確な出力に必要な最小限の幅」とみなします。出力幅の指定は、プリント形式の場合、デリミタ形式の場合を参照してください。

ＵｓｅｒＡ変換
ＵｓｅｒＢ変換

```
ＵｓｅｒＡ［入力幅｜＊］［：出力幅］
ＵｓｅｒＢ［入力幅｜＊］［：出力幅］
```

ＵｓｅｒＡ／Ｂ変換は、利用者独自のバイト単位の変換処理が必要なときに、ＡＮＫ変換表Ｕｓｅｒ－Ａ、Ｕｓｅｒ－Ｂを書き替えて利用します。なお、ＵｓｅｒＡ／Ｂ変換には、Ａｎｋ変換の説明がほとんどそのまま当てはまります。

プリント形式の場合

入力幅は、　　ｕａ２０　　のように、１０進のバイト数で指定します。また、

レコード全体をＵｓｅｒＡ変換するときは ---- ／ｍａｐ　ｕａ
最終項目をＵｓｅｒＡ変換するときは -------- ／ｍａｐ　・・・　ｕａ

のようにして入力幅を省略できます。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｕａ２０　　という指定は、
ｕａ２０：２０　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ユーザーＡ／Ｂ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のユーザーＡ／Ｂ項目のおわりにスペース（２０Ｈ／４０Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

デリミタ形式の場合

ふつう、入力幅は省略し、出力幅だけを１０進のバイト数で

ｕａ：２０　　のように指定します。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（出力）の残りバイト数

Ｎｕｍｅｒｉｃ変換

Ｎｕｍｅｒｉｃ［入力幅｜１５］［：出力幅］

文字形式の数値項目どうしの変換をします。

Ｎｕｍｅｒｉｃ変換は、Ａｎｋ変換と後述のＤｉｓｐＺｏｎｅ変換の中間的なものです。Ａｎｋ変換と比較すると、

文字形式数値しか通さない
入力幅の省略値が１５桁（バイト）である
右詰めになる

などの点が異なります。

「文字形式数値しか通さない」というのは、具体的にいい替えれば、

＋、－、０～９、ピリオド（.）、Ｅ、ｅ、Ｄ、ｄ

しか変換しないで、これら以外の文字は捨ててしまうということです。たとえば、通貨記号（￥／＄）や位取りのコンマ（,）などは削除されるので、リストファイルから入力データファイルを作るときなどに利用できます。

Ｂｉｎａｒｙ変換

Ｂｉｎａｒｙ　［入力幅｜＊｜定数］［：出力幅］

Ｂｉｎａｒｙ変換は、「コード変換を一切しない」という変換方法です。通常、Ｗｉｎ→ＩＢＭデータファイル変換では使用しません。

入力幅は、　ｂ２０　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＢｉｎａｒｙ変換するときには ---- ／ｍａｐ　ｂ
最終項目をＢｉｎａｒｙ変換するときには -------- ／ｍａｐ　・・・　ｂ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したバイナリ形式の定数が、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、コード変換されずにそのまま出力されます。

Ｂｉｎａｒｙ変換では、つぎの定数が使えます。

１６進列定数 -------　｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト
汎用定数 ---------- ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　ｂ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｂ２０　　という指定は、
ｂ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。
項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、バイナリ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のバイナリ項目のおわりにＮＵＬ（００Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

漢字イン挿入
漢字アウト挿入

！Ｉ［ｎ］	（英字の“アイ”）
！Ｏ	（英字の“オー”）

！Ｉ、！Ｏでそれぞれ、漢字イン（ＫＩ）、漢字アウト（ＫＯ）を固定位置に挿入できます。

漢字項目の前後で、

！Ｉ　Ｋａｎｊｉ４０　！Ｏ　と指定するのが、ふつうの使い方です。

この指定とＫａｎｊｉＭｉｘ変換は､似ているところもありますが別物です｡ご注意ください。

！Ｉのあとに、０～２５５の範囲の１０進数を指定することもできます。東芝方式のＡＮＫ・漢字まじり項目化するために「長さバイト」を付加する場合に指定します。

！ｉ４０　ｋ４０　のように指定します。

ＤｉｓｐＺｏｎｅ変換（Ｚｏｎｅ変換）

ＤｉｓｐＺｏｎｅ　［入力幅｜１５］：ピクチャ
（Ｚｏｎｅ　［入力幅｜１５］：ピクチャ）

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、プリント形式からの変換の場合は、ふつうは明示的に桁数を指定します。デリミタ形式からの変換の場合は、数値項目が１５バイトを超えることは少ないので、省略するほうがふつうです。

ピクチャの指定方法については、「操作の基礎」の章で説明したとおりです。たとえば、
－１２３．４　という数字を５バイトの項目に記録するとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

まとめると、プリント形式からの変換の場合は

ＤｉｓｐＺｏｎｅ　８：ｓ４．１　のような指定になり、

デリミタ形式からの変換の場合は

ＤｉｓｐＺｏｎｅ　：ｓ４．１　のような指定になるのがふつうです。

ただし、入力幅が１５バイトを超えるときは、

ＤｉｓｐＺｏｎｅ　２０：ｕ１５．２　のように、ダミーの入力幅を指定します。

ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換）

ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ
（Ｐａｃｋ　［入力幅｜１５］：ピクチャ）

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストのＣＯＢＯＬのパック形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、プリント形式からの変換の場合は、ふつうは明示的に桁数を指定します。デリミタ形式からの変換の場合は、数値項目が１５バイトを超えることは少ないので、省略するほうがふつうです。

ピクチャの指定方法については、「操作の基礎」の章で説明したとおりです。たとえば、
－１２３．４　という数字を３バイトの項目に記録するとすれば、ピクチャは

ｓ４．１　と指定します。

なお、ピクチャは省略できません。

まとめると、プリント形式からの変換の場合は

ＤｉｓｐＰａｃｋ　８：ｓ４．１　のような指定になり、

デリミタ形式からの変換の場合は

ＤｉｓｐＰａｃｋ　：ｓ４．１　のような指定になるのがふつうです。

ただし、入力幅が１５バイトを超えるときは、

ＤｉｓｐＰａｃｋ　２０：ｕ１５．２　のように、ダミーの入力幅を指定します。

ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換）〔ＢＣＤ形式〕

ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ（ｂ指定）
（Ｐａｃｋ　［入力幅｜１５］：ピクチャ（ｂ指定））

Ｗｉｎｄｏｗｓの文字形式数値項目を、ＰＯＳ端末等で使われているＢＣＤ形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、プリント形式からの変換の場合は、ふつうは明示的に桁数を指定します。デリミタ形式からの変換の場合は、数値項目が１５バイトを超えることは少ないので、省略するほうがふつうです。

ピクチャの指定方法については、「操作の基礎」の章で説明したとおりです。たとえば、
１２３．４　という数字を３バイトの項目に記録するとすれば、ピクチャは

ｂ５．１　と必ずｂ指定をします。

なお、ピクチャは省略できません。

まとめると、プリント形式からの変換の場合は

ＤｉｓｐＰａｃｋ　８：ｂ５．１　のような指定になり、

デリミタ形式からの変換の場合は

ＤｉｓｐＰａｃｋ　：ｂ５．１　のような指定になるのがふつうです。

ただし、入力幅が１５バイトを超えるときは、

ＤｉｓｐＰａｃｋ　２０：ｂ１５．２　のように、ダミーの入力幅を指定します。

ＺｏｎｅＤｉｓｐ変換

```
ＺｏｎｅＤｉｓｐ　ピクチャ［：出力幅］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　　と指定します。

なお、ピクチャは省略できません。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０とする
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＺｏｎｅＤｉｓｐ　ｓ４．１　　という指定は、
ＺｏｎｅＤｉｓｐ　ｓ４．１：７　　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＺｏｎｅＺｏｎｅ変換

ＺｏｎｅＺｏｎｅ　［入力ピクチャ］：出力ピクチャ 入力ピクチャ：［出力ピクチャ］

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、出力ピクチャは

ｓ４．１　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　：ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＺｏｎｅ　ｓ４．１　という指定は、
ＺｏｎｅＺｏｎｅ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＺｏｎｅＰａｃｋ変換

ＺｏｎｅＰａｃｋ　［入力ピクチャ］：出力ピクチャ
　　入力ピクチャ：［出力ピクチャ］

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、出力ピクチャは

ｓ４．１　（ＢＣＤ形式なら、ｂ４．１）　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＰａｃｋ　：ｓ４．１　という指定は、
ＺｏｎｅＰａｃｋ　ｓ４．１：ｓ４．１　という指定と同じです。

出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。たとえば、

ＺｏｎｅＰａｃｋ　ｓ４．１　という指定は、
ＺｏｎｅＰａｃｋ　ｓ４．１：ｓ４．１　という指定と同じです。

なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＤｉｓｐＢｉｎ変換

ＤｉｓｐＢｉｎ　［入力幅｜１５］：２進キャスト［ピクチャ］

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストの２進形式整数・小数項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、プリント形式からの変換の場合は、ふつうは明示的に桁数を指定します。デリミタ形式からの変換の場合は、数値項目が１５バイトを超えることは少ないので、省略するほうがふつうです。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

　　ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

まとめると、プリント形式からの変換の場合は

　　ＤｉｓｐＢｉｎ　１０：ｉ４ｓ４．１　　のような指定になり、

デリミタ形式からの変換の場合は

　　ＤｉｓｐＢｉｎ　：ｉ４ｓ４．１　　のような指定になるのがふつうです。

ただし、入力幅が１５バイトを超えるときは、

　　ＤｉｓｐＢｉｎ　２０：ｉ８ｕ１５．２　　のように、ダミーの入力幅を指定します。

ＺｏｎｅＢｉｎ変換

ＺｏｎｅＢｉｎ　入力ピクチャ：２進キャスト［ピクチャ］
［入力ピクチャ］：２進キャスト　ピクチャ

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストの２進形式整数・小数項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

なお、２進キャストは省略できません。

入力ピクチャも、２進キャストにつづくピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＺｏｎｅＢｉｎ　：ｉ４ｓ４．１　という指定は、
ＺｏｎｅＢｉｎ　ｓ４．１：ｉ４ｓ４．１　という指定と同じです。

ＴｏＤｉｓｐ変換

ＴｏＤｉｓｐ｛数値定数｜システム変数｝[：出力幅]

入力した数値定数、またはシステム変数の値を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

入力値として、つぎの数値定数、システム変数が使えます。省略はできません。

数値定数　　**整数定数、小数定数、１６進定数**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　[－１２３．４]　のように、[　]でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

出力幅は省略できます。出力幅を省略すると、入力した値を正確に出力するために必要な、最小限の幅で出力されます。たとえば、

ＴｏＤｉｓｐ　[－１２３．４]　　という指定は、
ＴｏＤｉｓｐ　[－１２３．４]：６　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＴｏＺｏｎｅ変換

ＴｏＺｏｎｅ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、５バイトの符号つきゾーン形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＺｏｎｅ　　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＰａｃｋ変換

ＴｏＰａｃｋ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、ホストのＣＯＢＯＬのパック形式数値項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、３バイトの符号つきパック形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＰａｃｋ　　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＢｉｎ変換

ＴｏＢｉｎ｛数値定数｜システム変数｝：２進キャスト［ピクチャ］

入力した数値定数、またはシステム変数の値を、ホストの２進形式整数・小数項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、４バイトの符号つき２進形式の項目に記録するとすれば、出力ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＢｉｎ　　［－１２３．４］：ｉ４ｓ４．１　　のような指定になります。

Ｙｅａｒ設定（年設定）

```
Ｙｅａｒ　　Ｗｎｎ｜Ｓｎｎ
                        ：Ｗｎｎ｜：Ｓｎｎ
          Ｗｎｎ｜Ｓｎｎ　：Ｗｎｎ｜：Ｓｎｎ
```

日付データ項目を変換する際の、年の２桁（ｙｙ）と４桁（ｙｙｙｙ）の変換方式を設定します。Ｗｎｎの“W”はウインドウ方式を、Ｓｎｎの“Ｓ”はシフト方式を意味し、“ｎｎ”は００～９９の数字で指定します。“：”の左側の指定は入力側、“：”の右側の指定は出力側への適用になり、入力側のみ、出力側のみ、入出力両方の３通りの指定ができます。たとえば、

Ｙｅａｒ	Ｗ３０	入力データの年を１９３０～２０２９年とみなす
Ｙｅａｒ	：Ｓ２５	出力データの年下２桁を、－２５する
Ｙｅａｒ	Ｗ３０：Ｓ２５	入力データの年を１９３０～２０２９年とみなし、 出力データの年下２桁を、－２５する

のように指定します。

また、シフト方式（“Ｓｎｎ”指定）では、つぎの特殊指定ができます。

Ｙｅａｒ	ＳＳｈｏｗａ	“Ｓ２５”の指定と同じ（昭和通年方式）
Ｙｅａｒ	ＳＨｅｉｓｅｉ	“Ｓ８８”の指定と同じ（平成通年方式）

Ｙｅａｒ設定はＤａｔｅ変換が実行された時に適用になり、複数のＹｅａｒ設定がなされている場合は、Ｄａｔｅ変換の直前のＹｅａｒ設定が有効になります。

Ｙｅａｒ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

Ｙｅａｒ	Ｗ３０：Ｗ３０	入出力データの年を１９３０～２０２９年とみなす

ＤａｔｅＤｅｌｉｍ設定（日付区切り設定）

```
ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ｜ＨＹＰＨＥＮ｜ＰＥＲＩＯＤ
```

日付データ項目を出力する際の日付区切り記号をつぎの３つの中から設定します。

指定文字	日付区切り記号	データ例
ＳＬＡＳＨ	／（スラッシュ）	１９９８／１２／３１
ＨＹＰＨＥＮ	－（ハイフン）	１９９８－１２－３１
ＰＥＲＩＯＤ	.（ピリオド）	１９９８．１２．３１

ＤａｔｅＤｅｌｉｍ設定はＤａｔｅ変換が実行された時に適用になり、複数のＤａｔｅＤｅｌｉｍ設定がなされている場合は、Ｄａｔｅ変換の直前のＤａｔｅＤｅｌｉｍ設定が有効になります。

ＤａｔｅＤｅｌｉｍ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ　日付区切り記号を“／”にする

Ｄａｔｅ変換

```
Ｄａｔｅ　入力日付マスク：出力日付マスク
```

日付データ項目を変換します。

入力日付マスク、出力日付マスクは必ず指定します。省略はできません。たとえば、

Ｄａｔｅ　ｙｙｙｙ－ｍｍ－ｄｄ：ｙｙｍｍｄｄ

のように指定すると、コード変換後に、入力側１０バイトの日付データ項目を、出力側６バイトの日付データ項目に編集します。その際に、Ｙｅａｒ設定、ＤａｔｅＤｅｌｉｍ設定が適用になります。

デリミタ検出（コンマ）

,

デリミタ形式から変換するときは、なんらかの方法で項目分けをしなければいけません。コンマを使うと行頭からデリミタ、デリミタからデリミタ、デリミタから行末までを項目として認識します。なお、引用符でくくられた項目は、そのなかにデリミタがあっても単なる文字データとして扱われます。

デリミタとしてはコンマ、タブ、スペースの３つをサポートしています。その指定方法については、／Ｄｅｌｉｍｉｔｅｄオプションの説明を参照してください。なお、プリント形式からの変換のときは、このデリミタ検出の機能は無効になります。

例をあげます。たとえば、ゾーン形式の数値項目と、パック形式の数値項目を区切るには、

／Ｄｅｌｉ　／ＭＡＰ　・・・　Ｚｏｎｅ：ｕ５，Ｐａｃｋ：ｓ３．２　・・・

のような指定になります。

引用符はずし

“～”

デリミタ形式からの変換の場合には、項目が引用符（“ ”や‘ ’）でくくられていることがあります。引用符はずしの指定があった場合は、引用符でくくったなかにデリミタがあっても、ただの文字データとして扱います。その代わり、引用符は文字データとして扱われませんので、変換後には消滅しています。

引用符でくくられていることがわかっている項目に対しては、読みやすさのために、

／ＭＡＰ　・・・，“Ａｎｋ：８”，・・・　　のような指定ができます。

これは、

／ＭＡＰ　・・・，Ａｎｋ：８，・・・　　と指定したのとまったく同じ働きをします。

「ＰｕｔＤａｔａ」コマンドの「引用符はずし」機能において使える引用符としては、二重引用符「”（全角）」と一重引用符「’（半角）」があります。「ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」オプションにて、「ＱｕｏｔａｔｉｏｎＭａｒｋ」を「Ｓｉｎｇｌｅ」または「Ｄｏｕｂｌｅ」にすることで指定します。

◆注意　----　引用符の種類の指定は一回のコマンド処理全体に対して一回のみ

引用符はずしで使用する引用符の種類は、Ｍａｐ文の中で処理に応じて切り替えて指定することは出来ません。

◆注意　----　Ｍａｐ文の引用符記号は、常に二重引用符「”」

「／ＱｕｏｔｉｎｇＣｏｎｔｒｏｌ」コマンドオプションにて、「ＱｕｏｔａｔｉｏｎＭａｒｋ」を「Ｓｉｎｇｌｅ」に指定した時も、Ｍａｐ文において引用符くくりに使う記号は、二重引用符「”（半角）」ですので注意してください。

桁移動（プリント形式）

＠入力桁位置
＠：出力桁位置

変換対象にするＷｉｎｄｏｗｓ側（入力）の桁位置や、変換結果を書き込むＩＢＭ側（出力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

入力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

．　　Ｗｉｎｄｏｗｓ（入力）側の現在の桁位置

という特殊変数が使えます。たとえば、

＠．－４０　Ｋａｎｊｉ２０　＾２０　　と指定すれば、

現在の入力桁位置から４０バイト戻り、漢字２０バイト変換せよ
そして、元の位置に戻れ

という意味になります。

出力桁位置を移動することもできます。ふつう１０進数で桁位置を指定します。式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）の末尾
．　　ＩＢＭ側（出力）の現在の桁位置

◆注意　----　先頭を０桁目とする
Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

桁移動（デリミタ形式）

＠：出力桁位置

変換結果を書き込むＩＢＭ側（出力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

デリミタ形式からの変換の場合、出力桁位置の移動だけが有効です。入力桁位置の指定はできません。

出力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

＄　　ＩＢＭ側（出力）の末尾
．　　ＩＢＭ側（出力）の現在の桁位置

という特殊変数が使えます。たとえば、

，＠：＄－８　Ｚｏｎｅ：ｕ８，　　と指定すれば、

ＩＢＭレコード末尾の８バイト前に出力先を移動し
Ｗｉｎｄｏｗｓ側の今の数値項目を８桁のゾーン形式に変換せよ

という意味になります。

◆注意　----　先頭を０桁目とする

Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

項目移動（デリミタ形式）

＠＠項目位置（０～最大項目数－１）

変換対象にするＷｉｎｄｏｗｓ側（入力）データのデリミタ形式の項目位置を、別の任意の項目位置に移動できます。現在、処理対象にしている項目位置を、この機能で強制的に変更できます。この機能を利用すると、デリミタ形式の項目の組み替えなどが簡単に実現できます。

項目位置は、ふつう１０進数で何番目（０起点）の項目に移動するかを指定します。式による指定もでき、そのなかでは、

．　Ｗｉｎｄｏｗｓ側（入力）の現在の項目位置
＄　Ｗｉｎｄｏｗｓ側（入力）の入力全体の項目数
＊　Ｗｉｎｄｏｗｓ側（入力）の残りの項目数

という特殊変数が使えます。たとえば、

＠＠．－２　Ｋａｎｊｉ２０　＠＠．＋１　と指定すれば、

今の入力項目から２項目戻り、漢字２０バイト変換せよ
そして、元の位置に戻れ

という意味になります。また、

＠＠＄－２　ＤｉｓｐＺｏｎｅＵ８　と指定すれば、

Ｗｉｎｄｏｗｓ側（入力）レコード末尾から２項目に位置づけ、
８バイトＤｉｓｐＺｏｎｅ変換せよ

という意味になります。

◆注意　----　先頭を０項目目とする

Ｆ＊ＴＲＡＮ２００７では、レコードの先頭項目を０項目目として数えます。

入力スキップ（プリント形式）

```
ˆ［ｎ｜１］
```

Ｗｉｎｄｏｗｓ側（入力）レコードに不要な項目があるとき、それをスキップして変換できます。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

ˆ３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

ˆ３　　は　　ˆˆˆ　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

これを使えば、　　／ＭＡＰ　・・・　ˆ＊　　として「残りは全部スキップせよ」という指定もできます。しかし、何の指定もなかった分は変換対象にならないので、この ˆ ＊は冗長です。省いたほうがよいでしょう。

入力スキップ（デリミタ形式）

```
，　，　・・・
```

デリミタ形式の場合には、コンマを２つ連ねて書くと１項目スキップできます。同様に、コンマをｎ個連ねるとｎ－１項目スキップできます。

ただし、レコードの先頭項目をスキップするには、コンマをｎ個連ねて、ｎ項目のスキップにします。例をあげると、先頭の２項目スキップしたいなら、

／ＭＡＰ　,,　ＤｉｓｐＺｏｎｅ：×××，ＤｉｓｐＺｏｎｅ：×××

のような指定になり、途中の２項目スキップしたいなら、

／ＭＡＰ　ＤｉｓｐＺｏｎｅ：×××,,，ＤｉｓｐＺｏｎｅ：×××

のような指定になります。

出力スキップ

＿［ｎ｜１］

入力スキップ（プリント形式）とは逆に、ＩＢＭ側（出力）に何桁かスペースを作ることもできます。空項目を作るのがおもな用途です。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

＿３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

＿３　　は　　＿ ＿ ＿　　と指定したのと同じです。

／ＰＨａｓｅ　　---- ヘッダ・トレーラの生成・付加のタイミングを指定する

```
／ＰＨａｓｅ　　［Ｏｐｅｎ＜ｎ＞］,
　　［Ｍａｉｎ＜ｎ＞］,
　　［Ｃｌｏｓｅ＜ｎ＞］
```

データを変換する際にヘッダやトレーラを付加したいときに、この／ＰＨａｓｅオプションで各フェーズの呼び出しが起きる回数を指定します。フェーズにはつぎの３種類があります。

Ｏｐｅｎフェーズ	ヘッダの生成・付加を行う
Ｍａｉｎフェーズ	本体の変換をする
Ｃｌｏｓｅフェーズ	トレーラやエンドレコードの生成・付加を行う

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数を、＜ｎ＞に１０進数値で指定します。Ｍａｉｎフェーズでは、

＄　　全レコード分

という指定もできます。
Ｏｐｅｎフェーズ→Ｍａｉｎフェーズ→Ｃｌｏｓｅフェーズの順に呼び出しを行います。

◆注意　----　フェーズごとの指定について

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数のみ、指定します。各フェーズごとの動作は、Ａｔｌａｓ構文を使って設定する必要があります。詳細はマルチレコード編のマニュアルを参照してください。

●オプション省略時の動作

オプションをすべて省略すると、

／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式ファイルから変換する
／ＭＡＰ　Ａｎｋ	ＡＮＫデータのみとして変換する
／ＮＥＷ	新規ファイルとして作成する
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替えない（中断）
／Ｆｏｒｍ　Ｅｘｔｅｎｄｅｄ　１２８　Ａｕｔｏ　ＦＢ　Ｃｌｉｐ	レコード長＝１２８バイトの、拡張形式で適当にブロック化したファイルを作る
／ＥｘｔｒａＳｐａｃｅ　０	予備領域をとらない
／ＢｏｅＢｏｕｎｄａｒｙ　Ｔｒａｃｋ	ＩＢＭファイルをトラック境界ではじめる
／ＥｏｅＢｏｕｎｄａｒｙ　＊	ＢｏｅＢｏｕｎｄａｒｙオプションに従う
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋ	シングルボリューム時､ボリューム順序番号を空白にする
／ＥＯＦ	ＥＯＦコードを検査する
／ＰＨａｓｅ　Ｏｐｅｎ０，Ｍａｉｎ＄，Ｃｌｏｓｅ０	ヘッダ・トレーラをつけないで全レコード変換する

と指定したものとしてデータファイル変換を行います。

■解説

●／ＭＡＰオプションと／Ｄｅｌｉｍｉｔｅｄオプションの指定方法

ＩＢＭ形式の、つぎのようなレコードレイアウトのファイルＰＬＡＮＥＴを作成するものとします。レコード長は６４バイトです。プリント形式からの変換と、デリミタ形式からの変換では、／ＭＡＰオプションの指定がまったく違うので、気をつけてください。

ＰＬＡＮＥＴのレコードレイアウト

項番	項　　目	桁	幅	デ　ー　タ　形　式	内　容　例
１	Ｎｏ．	0	2	ＡＮＫ	1
２	和名	2	8	漢字	水星
３	英名	10	10	ＡＮＫ	MERCURY
４	読み	20	9	ＡＮＫ	ﾏｰｷｭﾘｰ
５	質量比	29	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
６	衛星数（確定済）	33	2	符号なしゾーン形式 整数部２桁	0
７	極大等級	35	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
８	英名の意味・由来	38	20	漢字	口神）神の使者
--	フィラー	58	6	--	--

プリント形式からの変換

これをプリント形式から変換するときのオプションの指定は、

```
/f64 /map a2 k8 a10 a9 dp8:u4.3 dz2:u2 dz5:s2.1 k20
```

のようになります。

デリミタ形式からの変換

デリミタ形式のうち、コンマ区切り（ＣＳＶ）形式から変換するなら、

```
/f64 /d /map a:2, k:8, a:10, a:9, dp:u4.3, dz:u2, dz:s2.1, k:20
```

のようになります。入力幅を省くのがポイントです。この例では引用符を省いてみましたが、読みやすさのために引用符を使ってもかまいません。

／ｄは／Ｄｅｌｉｍｉｔｅｄオプションの短縮指定です。タブ区切り形式から変換したいなら、／Ｄｅｌｉｍｉｔｅｄオプションで　Ｂｙ　ＴＡＢ　指定すればよいので、

```
/f64 /dbtab /map a:2, k:8, a:10, a:9, dp:u4.3, dz:u2, dz:s2.1, k:20
```

という指定になります。スペース区切り形式からでも同様で、／ｄｂｔａｂが／ｄｂｓｐになるだけです。

以上が、バッチファイルに組み込むときの基本スタイルです。また、／ＭＡＰオプションのデリミタ検出（＝コンマ〔,〕）と／Ｄｅｌｉｍｉｔｅｄオプションは、常にペアで使うこともわかったかと思います。

デリミタ形式からの変換のときは、「出力側」を指定しなければいけないことも、わかったと思います。

●変換テストから本番まで

デリミタ形式からの変換を例にとり、変換テストの進め方を説明します。

ＰｕｔＤａｔａ(ｐｄ)コマンドでＩＢＭ形式に変換しても、うまく変換できているかどうか、わかりにくいものです。ＩＢＭディスクエディタ（ＧＵＩ部）で、できたＩＢＭファイルを直接見る方法はやや専門的な知識を必要とします。簡単なのは、ＧｅｔＤａｔａ(ｇｄ)コマンドを使って逆方向の変換をしてみることです。

つぎのような変換用バッチファイルと、

＜ＰＬＡＰＵＴＣＳ．ＢＡＴ（その１）＞

```
start /w ft putdata c:planet.csv a: /f64 /rep /d /map a:2, k:8, a:10, a:9,
 dp:u4.3, dz:u2, dz:s2.1, k:20 ↓
```

逆変換用バッチファイルを作り、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ＞

```
start /w ft getdata a:planet c:test /dnc /map a2, k8, a10, a9, dpu4.3, dzu2,
 dzs2.1, k20 ↓
```

テストランと修正を繰り返しながら変換テストを進めていきます。

変換テストの段階では、変換結果が簡単にわかるように、

圧縮なしのコンマ区切り（ＣＳＶ）形式

にしていることに注意してください。

「これでＯＫ」となれば、このテスト用バッチファイルを修正して、本番用のバッチファイルにします。簡単な本番用バッチファイルは、

＜ＰＬＡＰＵＴＣＳ．ＢＡＴ（その２）＞

```
@echo off ↓
start /w ft /dc/ putdata c:planet.csv a: /f64 /rep /d /map a:2, k:8, a:10, a:9,
 dp:u4.3, dz:u2, dz:s2.1, k:20 ↓
```

のようなスタイルになります。

●パラメータファイルを使う

パラメータファイルを利用すると、さらに運用と保守・管理が簡単になります。上の例は、つぎのようなバッチファイルと、

＜ＰＬＡＰＵＴＣＳ．ＢＡＴ（その３）＞

```
@echo off ↓
start /w ft /dc/ putdata c:planet.csv a: ++plaputcs.p ↓
```

つぎのようなパラメータファイルに、

＜ＰＬＡＰＵＴＣＳ．Ｐ＞

```
/form 64                    -- 適当にブロッキング ↓
/replace ↓
/delimited                  -- ＣＳＶ形式から変換 ↓
/map ↓
      ank       :2 ,        -- Ｎｏ．（惑星番号） ↓
      kanji     :8 ,        -- 和名 ↓
      ank       :10 ,       -- 英名 ↓
      ank       :9 ,        -- 読み ↓
      disppack  :u4.3 ,     -- 質量比 ↓
      dispzone  :u2 ,       -- 衛星数（確定済） ↓
      dispzone  :s2.1 ,     -- 極大等級（みかけ上の最大の明るさ） ↓
      kanji     :20         -- 英名の意味・由来 ↓
```

分けて記述できます。これで大幅にわかりやすくなりました。

●いろいろな加工・編集の方法

いらない項目をスキップする

ホストからデータを持ってくると、いらない項目がいくつもあることが多いものです。それをスキップして変換するのは簡単で、

プリント形式からの変換の場合は、　^n
デリミタ形式からの変換の場合は、　,,

と書きます。
プリント形式からの場合のｎはスキップするバイト数です。たとえば、例題のデータから「英名」「読み」「英名の意味・由来」の項目を変換対象からはずすなら、／ＭＡＰオプションの指定は、

```
/map a2 k8 ^10 ^9 dp8:u4.3 dz2:u2 dz5:s2.1
```

となります。「英名の意味・由来」の項目は最後の項目なので、^20 とは書かずに省略しました。同様にデリミタ形式からの場合は、

```
/map a:2, k:8, , , dp:u4.3, dz:u2, dz:s2.1
```

となります。この場合も「英名の意味・由来」の項目は最後の項目なので、なにも書かずに省略しました。

空項目を追加する

空項目を作るのも簡単です。プリント形式からの変換でもデリミタ形式からの変換でも、

```
_n
```

と書くだけです。例題のデータに新たに「ボイジャーⅠ通過日」「ボイジャーⅡ通過日」というＹＹＭＭＤＤ形式の項目を追加するには、プリント形式からの場合は、

```
/map a2 k8 a10 a9 _6 _6 dp8:u4.3 dz2:u2 dz5:s2.1 k20
```

となり、デリミタ形式からの場合は、

```
/d /map a:2, k:8, a:10, a:9 _6 _6, dp:u4.3, dz:u2, dz:s2.1, k:20
```

となります。この例では「読み」の項目のうしろに、つづけて２つ追加してみました。

項目長を増減する

項目長の増減もよく必要になります。プリント形式からの場合はつぎのように、

```
/map a2:3 k8:10 a10 a9 dp10:u4.3 dz2:u3 dz6:s2.1 k20:30
```

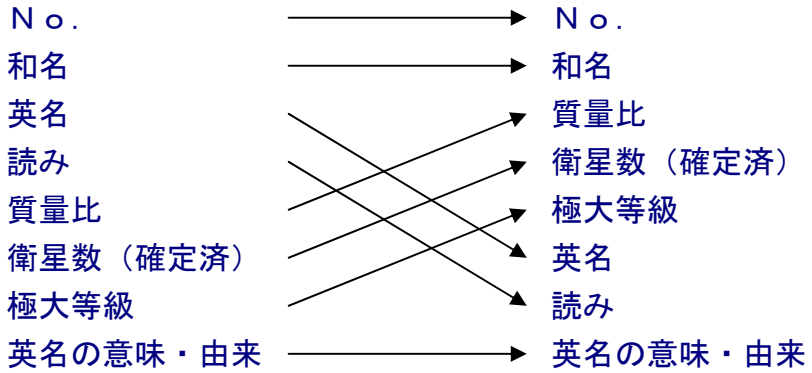
基本的にはコロンのあとに出力幅や出力ピクチャを書いていくだけです。デリミタ形式からの場合でも、

```
/d /map a:3, k:10, a:10, a:9, dp:u4.3, dz:u3, dz:s2.1, k:30
```

のように、出力幅や出力ピクチャを変えるだけですみます。

項目を組み替える

項目組み替えは、桁移動の機能を利用するので少し面倒です。例題のデータを、



というふうに組み替える場合を例にとりましょう。

プリント形式からの項目組み替えは、つぎのようなパラメータファイルを作ることからはじめます。

```
/map ↓
      ank         2          -- Ｎｏ.（惑星番号）　↓
      kanji       8          -- 和名 ↓
      ank         10         -- 英名 ↓
      ank         9          -- 読み ↓
      disppack    8:u4.3     -- 質量比 ↓
      dispzone    2:u2       -- 衛星数（確定済）　↓
      dispzone    5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      kanji       20         -- 英名の意味・由来 ↓
```

つぎに、各項目の「入力」桁位置をつけていきます（@0～@44）。

```
/map ↓
      @0   ank        2          -- Ｎｏ．（惑星番号）　↓
      @2   kanji      8          -- 和名 ↓
      @10  ank        10         -- 英名 ↓
      @20  ank        9          -- 読み ↓
      @29  disppack   8:u4.3     -- 質量比 ↓
      @37  dispzone   2:u2       -- 衛星数（確定済）　↓
      @39  dispzone   5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      @44  kanji      20         -- 英名の意味・由来 ↓
```

こうしておいて、あとはエディタで好きなように順番を入れ替えていきます。たとえば、

```
/map ↓
      @0   ank        2          -- Ｎｏ．（惑星番号）　↓
      @2   kanji      8          -- 和名 ↓
      @29  disppack   8:u4.3     -- 質量比 ↓
      @37  dispzone   2:u2       -- 衛星数（確定済）　↓
      @39  dispzone   5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      @10  ank        10         -- 英名 ↓
      @20  ank        9          -- 読み ↓
      @44  kanji      20         -- 英名の意味・由来 ↓
```

とします。

デリミタ形式からの項目組み替えは､つぎのようなパラメータファイルを作ることからはじめます。

```
/delimited                  -- ＣＳＶ形式から変換 ↓
/map ↓
      ank        :2 ,       -- Ｎｏ．（惑星番号） ↓
      kanji      :8 ,       -- 和名 ↓
      ank        :10 ,      -- 英名 ↓
      ank        :9 ,       -- 読み ↓
      disppack   :u4.3 ,    -- 質量比 ↓
      dispzone   :u2 ,      -- 衛星数（確定済） ↓
      dispzone   :s2.1 ,    -- 極大等級（みかけ上の最大の明るさ） ↓
      kanji      :20        -- 英名の意味・由来 ↓
```

つぎに、新しいレコードレイアウト図を起こします。

ＰＬＡＮＥＴのレコードレイアウト（２）

項番	項　　目	桁	幅	デ　ー　タ　形　式	内　容　例
１	Ｎｏ．	0	2	ＡＮＫ	1
２	和名	2	8	漢字	水星
３	質量比	10	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
４	衛星数（確定済）	14	2	符号なしゾーン形式 整数部２桁	0
５	極大等級	16	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
６	英名	19	10	ＡＮＫ	MERCURY
７	読み	29	9	ＡＮＫ	ﾏｰｷｭﾘｰ
８	英名の意味・由来	38	20	漢字	ロ神）神の使者
--	フィラー	58	6	--	--

こうしておいて、あとはエディタで「出力桁位置」を指定していきます。

たとえば、

```
/delimited                          -- ＣＳＶ形式から変換 ↓
/map ↓
      @:0   ank        :2 ,         -- Ｎｏ．（惑星番号） ↓
      @:2   kanji      :8 ,         -- 和名 ↓
      @:19  ank        :10 ,        -- 英名 ↓
      @:29  ank        :9 ,         -- 読み ↓
      @:10  disppack   :u4.3 ,      -- 質量比 ↓
      @:14  dispzone   :u2 ,        -- 衛星数（確定済） ↓
      @:16  dispzone   :s2.1 ,      -- 極大等級（みかけ上の最大の明るさ） ↓
      @:38  kanji      :20          -- 英名の意味・由来 ↓
```

とします。

■使用例１（プリント形式）

例１）ＡＮＫコードのみのファイルを変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴは、ＡＮＫデータだけのプリント形式のファイルです。それをドライブＡ：のＩＢＭファイルＪＯＵＲＮＡＬに変換します。論理レコード長は１２８バイトにするので、／Ｆｏｒｍオプションを省略します。

```
C:¥>ft putdata c:journal.dat a:* ↓
```

例２）ＡＮＫ・漢字まじりのファイルを変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴは、ＡＮＫ項目と漢字項目が入りまじっています。しかし、漢字項目内のスペースは全角スペース（８１４０Ｈ）に統一されています。これをＡＮＫ・漢字まじり変換します。

```
C:¥>ft putdata c:journal.dat a:/map km ↓        (/MAP KanjiMix)
```

この変換をするときは、変換設定の漢字変換方式設定（ＧＵＩ部）の「シフト節約度」を「弱」に設定しておかなければいけません。

例３）レコード長を拡大する

ドライブＣ：にＷｉｎｄｏｗｓファイルＺＡＩＫＯｎｎ．ＤＡＴが数本あります。レコード長は１５０バイトで、内容はＡＮＫのみです。これをＩＢＭ形式に変換するとき、同時にレコード長を１７０バイトに、ブロック長を５１０バイトにします。

```
C:¥>ft putdata c:zaiko*.dat a:/f170s510 ↓       (/Form 170 Size510)
```

例４）レコード長を縮小する

ドライブＣ：のＷｉｎｄｏｗｓファイルＵＲＩＡＧＥ．２５６は、レコード長が２５６バイトで、後半の１００バイト前後がフィラーです。先頭から１７０バイトをＩＢＭ形式に変換し、ブロック化係数６でブロック化し、ブロック長を１０２０バイトにします。

```
C:¥>ft putdata c:uriage.256 a:/f170r6 ↓         (/Form 170 Records6)
```

例５）項目別変換する

ドライブＣ：にＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢがあり、内容は住所録でつぎのように項目が分かれています。

項　目	タイプ	幅	
氏名	漢字	１６	
フリガナ	ＡＮＫ	１６	計３４バイト
電話番号	ＡＮＫ	１２	
郵便番号	ＡＮＫ	６	
住所	漢字	４０	
生年月日	ＹＹＹＹ－ＭＭ－ＤＤ	１０	
区分	ＡＮＫ	１	

レコード長は１０１バイト（改行コード含まず）です。これをドライブＡ：のＩＢＭファイルＡＤＤＲＥＳＳに変換し、ＡＮＫ項目や漢字項目が混在しているので項目別変換にします。レコード長は省略値の１２８バイトにし、余った部分はフィラーにします。

```
C:¥>ft putdata c:address.db a:/map k16 a34 k40 dyyyy-mm-dd:yymmdd a1 ↓
          (/MAP Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd Ank1)
```

例６）パラメータファイルを使う

例５のファイルを、パラメータファイルを使って変換します。パラメータファイルには、日付項目変換のための年設定を追加します。

```
C:¥>ft putdata c:address.db a: ++putaddr.p ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ＞

```
/NonDelemited                    -- プリント形式からの変換 ↓
/map ↓
      kanji  16                  -- 氏名 ↓
      ank    16                  -- フリガナ ↓
      ank    12                  -- 電話番号 ↓
      ank    6                   -- 郵便番号 ↓
      kanji  40                  -- 住所 ↓
      year   :sshowa             -- 出力日付年＝昭和 ↓
      date   yyyy-mm-dd:yymmdd   -- 生年月日 ↓
      ank    1                   -- 区分 ↓
```

例７）バッチファイル化する

例６の処理をバッチファイル化します。パラメータファイルＰＵＴＡＤＤＲ．Ｐは例６と同一です。

```
start /w ft putdata c:address.db a: ++putaddr.p ↓
```

例８）入力桁移動を使って、項目を組み替える

例５と同様ですが、入力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft putdata c:address.db a:/map @100 a1 @0 k16 a34 k40 dyyyy-mm-dd:yymmdd ↓
        (/MAP @96 Ank1 @0 Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd)
```

例９）例８の処理でパラメータファイルを使う

例８と同じ処理を、パラメータファイルを使って実行します。

```
C:¥>ft putdata c:address.db a: ++putaddr.p1 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ１＞

```
/map ↓
      @100  ank     1                    -- 区分 ↓
      @0    kanji   16                   -- 氏名 ↓
            ank     16                   -- フリガナ ↓
            ank     12                   -- 電話番号 ↓
            ank     6                    -- 郵便番号 ↓
            kanji   40                   -- 住所 ↓
            date    yyyy-mm-dd:yymmdd    -- 生年月日 ↓
```

例 10）出力桁移動を使って、項目を組み替える

例５と同様ですが、出力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft putdata c:address.db a:/map @:1 k16 a34 k40 dyyyy-mm-dd:yymmdd @:0 a1 ↓
        (/MAP @:1 Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd @:0 Ank1)
```

例 11）例 10 の処理でパラメータファイルを使う

例 10 と同じ処理を、パラメータファイルを使って実行します。

```
C:¥>ft putdata c:address.db a: ++putaddr.p2 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ２＞

```
/map ↓
      @:1  kanji   16                    -- 氏名 ↓
           ank     16                    -- フリガナ ↓
           ank     12                    -- 電話番号 ↓
           ank     6                     -- 郵便番号 ↓
           kanji   40                    -- 住所 ↓
           date    yyyy-mm-dd:yymmdd     -- 生年月日 ↓
      @:0  ank     1                     -- 区分 ↓
```

■使用例２（デリミタ形式）

例１）コンマ区切り（ＣＳＶ）形式のファイルを変換する

ドライブＣ：にＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢがあり、内容は住所録でつぎのようにコンマ（,）で区切られて項目が分かれています。

項　目	タイプ	幅　（最大）
氏名	漢字	１６
フリガナ	ＡＮＫ	１６
電話番号	ＡＮＫ	１２
郵便番号	ＡＮＫ	６
住所	漢字	４０
生年月日	ＹＹＹＹ－ＭＭ－ＤＤ	１０
区分	ＡＮＫ	１

レコード長は可変ですが、固定長・固定欄に変換しても１００バイトに収まります。これをドライブＡ:のＩＢＭファイルＡＤＤＲＥＳＳに変換します。ブロック長は５００バイトにします。

```
C:¥>ft putdata c:address.db a: /d /f100s500 /map k:16, a:16, a:12, a:6, k:40,
     dyyyy-mm-dd:yymmdd, a1 ↓
  (/Delimited /Form 100 Size500 /MAP Kanji:16, Ank:16, Ank:12, Ank:6, Kanji:40,
    Date yyyy-mm-dd:yymmdd, Ank:1)
```

例２）バッチファイルとパラメータファイルを利用する。項目組み替えもする

例１と同様ですが、出力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。バッチファイルとパラメータファイルを使います。パラメータファイルには、日付項目変換のための年設定を追加します。

＜ＰＵＴＡＤＤＲ２．ＢＡＴ＞

```
start /w ft putdata c:address.db a: ++putaddr2.p ↓
```

＜ＰＵＴＡＤＤＲ２．Ｐ＞

```
/delimited                                  -- ＣＳＶ形式から変換 ↓
/form 100 size 500                          -- ＲＬ＝１００，Ｂ／Ｆ＝５ ↓
/map ↓
      @:1  kanji   :16                      -- 氏名 ↓
           ank     :16                      -- フリガナ ↓
           ank     :12                      -- 電話番号 ↓
           ank     :6                       -- 郵便番号 ↓
           kanji   :40                      -- 住所 ↓
           year    :sshowa                  -- 出力日付年＝昭和 ↓
           date    yyyy-mm-dd:yymmdd        -- 生年月日 ↓
      @:0  ank     :1                       -- 区分 ↓
```

例３）タブ区切り形式から変換する

例１と同様の内容の、タブ区切り形式のＷｉｎｄｏｗｓファイルを変換します。

```
C:¥>ft putdata c:address.db a: /dbtab /f100s500 /map k:16, a:16, a:12, a:6,
    k:40, dyyyy-mm-dd:yymmdd, a:1 ↓
    (/Delimited By TAB /Form 100 Size500 /MAP Kanji:16, Ank:16, Ank:12, Ank:6,
      Kanji:40, Date yyyy-mm-dd:yymmdd, Ank:1)
```

例４）スペース区切り形式から変換する

例１と同様の内容の、スペース区切り形式のＷｉｎｄｏｗｓファイルを変換します。

```
C:¥>ft putdata c:address.db a: /dbsp /f100s500 /map k:16 a:16, a:12, a:6,
    k:40, dyyyy-mm-dd:yymmdd, a:1 ↓
   (/Delimited By SPace /Form 100 Size500 /MAP Kanji:16, Ank:16, Ank:12, Ank:6,
     Kanji:40, Date yyyy-mm-dd:yymmdd, Ank:1)
```

■注意事項

漢字変換をするときは、漢字変換方式の割り当てを忘れずに

Ｋａｎｊｉ変換やＫａｎｊｉＭｉｘ変換を行うときは、あらかじめ

適当な漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、入力幅や出力幅は漢字データについても

バイト単位で指定

します。漢字の文字数ではないことに注意してください。

改行コードがないと、１件だけの変換になる

Ｗｉｎｄｏｗｓファイルの各レコードの末尾に改行コードがついていないと、先頭の１件だけ変換されます。この場合は、ＰｕｔＲａｎｄ（ｐｒ）コマンドを使うべきです。

その他の注意事項

「Ｐｕｔ系コマンド」の節を参照してください。

２．１０　ＰｕｔＲａｎｄ（ｐｒ）　プット・ランド

Ｗｉｎ→ＩＢＭ　ランダムファイル変換

ＰｕｔＲａｎｄ(ｐｒ)コマンドは、Ｗｉｎｄｏｗｓのランダムファイル（固定長ファイル）をＩＢＭ形式に変換するコマンドです。固定長レコードの単純な変換、固定長・固定欄形式のファイルの項目別変換などが行えます。バイナリ変換もできます。

固定長・固定欄なら、改行コードのついているテキストファイルも扱えますが、その場合にはＰｕｔＤａｔａ(ｐｄ)コマンドのプリント形式からの変換機能を使ったほうがずっと簡単です。

■コマンド形式

コマンド	パラメータ
ＰｕｔＲａｎｄ ｐｒ	［Ｗｉｎｄｏｗｓファイル　ＩＢＭファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ＩＢＭファイル

ＷｉｎｄｏｗｓファイルとＩＢＭファイルの指定方法は、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●指定できるオプション

ＰｕｔＲａｎｄ(ｐｒ)コマンドには、つぎのオプションが指定できます。

ＰｕｔＲａｎｄ(ｐｒ)コマンド固有のオプション

／Ｓｉｚｅ	Ｗｉｎｄｏｗｓ側のレコード長を指定する
／ＭＡＰ	項目別に変換方法の詳細を指定する
／ＰＨａｓｅ	ヘッダやトレーラの生成・付加のタイミングを指定する

Ｐｕｔ系コマンド共通のオプション

／ＮＥＷ	新規ファイルとして作成する
／ＯＬＤ	既存ファイルに出力する
／ＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルは置き替えない
／Ｆｏｒｍ	ＩＢＭファイルの形式を指定する 拡張形式・半拡張形式・基本形式
／ＥｘｔｒａＳｐａｃｅ	新規作成後の予備領域の量を指定する
／ＢｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＢＯＥ境界を指定する
／ＥｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＥＯＥ境界を指定する
／Ｑｕｅｒｙ	変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	ファイル名を確認せず自動変換する
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	ボリューム順序番号出力の制御
／ＳｉｎｇｌｅＶｏｌｕｍｅ	マルチボリューム処理機能を無効にする
／ＥＯＦ	ＥＯＦコードを検査する
／ＮｏＥＯＦ	ＥＯＦコードを検査しない

これらのオプションのうち、／Ｓｉｚｅオプション、／ＭＡＰオプション、／Ｆｏｒｍオプションがとくに重要です。

Ｐｕｔ系コマンド共通のオプションは、「Ｐｕｔ系コマンド」の節ですでに詳しく説明しました。そちらを参照してください。

●ＰｕｔＲａｎｄ（ｐｒ）コマンド固有のオプション

ＰｕｔＲａｎｄ（ｐｒ）コマンドに固有のオプションを説明します。

／Ｓｉｚｅ ---- Ｗｉｎｄｏｗｓ側のレコード長を指定する

```
／Ｓｉｚｅ　式
／Ｓｉｚｅ　＄
```

／Ｓｉｚｅオプションで、Ｗｉｎｄｏｗｓ側のレコード長を指定します。ふつう、

／Ｓｉｚｅ　２５６　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長

ＩＢＭファイルのレコード長（ふつう、／Ｆｏｒｍオプションで指定）を基準にして、Ｗｉｎｄｏｗｓファイルのレコード長を決めることができます。たとえば、

／Ｓｉｚｅ　＄＋２

と指定すれば、「ＩＢＭファイルのレコード長に＋２したものをＷｉｎｄｏｗｓ側のレコード長にせよ」という意味になります。

／Ｓｉｚｅオプションを省略すると、

／Ｓｉｚｅ　＄

と指定した扱いになり、Ｗｉｎｄｏｗｓレコード長＝ＩＢＭレコード長になります。ですから、比較的単純な変換で、レコード長は元と同じでよい、というときは／Ｓｉｚｅオプションを省略します。

◆注意　----　Ｗｉｎｄｏｗｓ自身には「レコード長」の概念がない

Ｗｉｎｄｏｗｓ自体にはファイルごとに登録されたレコード長というものはありません。そのため、かなり自由が利きます。アプリケーションがファイルを扱う論理的な単位を、ここではＷｉｎｄｏｗｓ側のレコード長、あるいはＷｉｎｄｏｗｓレコード長と呼んでいます。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／ＭＡＰ	Ａｎｋ変換 Ｋａｎｊｉ変換　　・・・ ＡｎｋｉＺｅ変換 ＫａｎｊｉＺｅ変換 ＫａｎｊｉＭｉｘ変換 ＵｓｅｒＡ変換／ＵｓｅｒＢ変換 Ｎｕｍｅｒｉｃ変換 Ｂｉｎａｒｙ変換 漢字イン挿入／漢字アウト挿入 ＤｉｓｐＺｏｎｅ変換（Ｚｏｎｅ変換） ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換） ＺｏｎｅＤｉｓｐ変換 ＰａｃｋＤｉｓｐ変換 ＺｏｎｅＺｏｎｅ変換 ＺｏｎｅＰａｃｋ変換 ＰａｃｋＺｏｎｅ変換 ＰａｃｋＰａｃｋ変換 ＤｉｓｐＢｉｎ変換 ＢｉｎＤｉｓｐ変換 ＺｏｎｅＢｉｎ変換 ＰａｃｋＢｉｎ変換 ＢｉｎＺｏｎｅ変換 ＢｉｎＰａｃｋ変換 ＢｉｎＢｉｎ変換 ＢｉｎａｒｙＸ変換 ＴｏＤｉｓｐ変換 ＴｏＺｏｎｅ変換 ＴｏＰａｃｋ変換 ＴｏＢｉｎ変換 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 桁移動／入力スキップ／出力スキップ	

／ＭＡＰ　　Ａｎｋ

この／ＭＡＰオプションで細かい変換方法を指示します。本来なら、自動的に項目を認識して変換ができると便利です。しかし、ホストの、とくにＣＯＢＯＬのデータには、**データ自身に桁数や小数点位置の判断に必要な情報が含まれていない**、という特性があります。そのため自動変換は原理的に不可能なのです。

◆注意　----　マルチレコードレイアウト等の指定について

／ＭＡＰオプションには、マルチレコードレイアウト等の指定ができるＡｔｌａｓ構文を記述することもできます。詳細は、マルチレコード編のマニュアルを参照してください。

Ａｎｋ変換

Ａｎｋ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ項目を変換します。どのコード変換が行われるかは、変換設定のＡＮＫコードの設定で決まります（ふつう、セットアップ時に１回だけ行います)。

入力幅は、　　ａ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＡｎｋ変換するときには ---- ／ｍａｐ　ａ
最終項目をＡｎｋ変換するときには -------- ／ｍａｐ　・・・　ａ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

Ａｎｋ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡＮＫ形式、最大２５６文字で、半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ａ［‘ＡＮＫ’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ａ２０　　という指定は、
ａ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のＡＮＫ項目のおわりにスペース（２０Ｈ／４０Ｈ）が詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

ＡｎｋｉＺｅ（ＡＮＫ化）変換

ＡｎｋｉＺｅ　［入力幅｜＊］［：出力幅］

Ｗｉｎｄｏｗｓ側（入力）文字列をできる限りＡＮＫ（半角）に変換します。

入力幅は、　　ａｚ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。
出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ＡＮＫ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のＡＮＫ項目のおわりにスペースが詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

Ｋａｎｊｉ変換

Ｋａｎｊｉ［入力幅｜＊｜定数］［：出力幅］

漢字項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に１回だけ行います）。

入力幅は、　　ｋ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません）。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉ変換するときには ---- ／ｍａｐ　ｋ
最終項目をＫａｎｊｉ変換するときには -------- ／ｍａｐ　・・・　ｋ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定した漢字形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

Ｋａｎｊｉ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列は漢字形式、最大２５６文字で、半角(')でくくる
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋ［‘漢字’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋ１６　　という指定は、
ｋ１６：１６　　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません）。逆に、項目長を伸ばすと、ＩＢＭ側の漢字項目のおわりに漢字変換方式に設定されている漢字スペースが詰められます。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

ＫａｎｊｉＺｅ（漢字化）変換

```
ＫａｎｊｉＺｅ　［入力幅｜＊］［：出力幅］
```

Ｗｉｎｄｏｗｓ側（入力）文字列をできる限り漢字（全角）に変換します。

入力幅は、　　ｋｚ１６　　のように、１０進のバイト数で指定します（漢字の文字数ではありません）。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます
項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、漢字項目のおわりのほうが切り捨てられます（漢字の中央で切れることはありません）。逆に、項目長を伸ばすと、ＩＢＭ側の漢字項目のおわりに漢字変換方式に設定されている漢字スペースが詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ［入力幅｜＊｜定数］［：出力幅］

ＡＮＫ・漢字まじり項目を変換します。どのコード変換が行われるかは、変換設定の漢字変換方式の設定で決まります（ふつう、セットアップ時に１回だけ行います)。変換後、漢字文字列の前後にはＫＩ／ＫＯが挿入されます。この、ＫＩ／ＫＯ挿入のタイミングも漢字変換方式の設定で決まります。そのため、オーバーフローの危険性があるので、出力幅の指定には注意しなければいけません。

入力幅は、　　ｋｍ３０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＫａｎｊｉＭｉｘ変換するときには ---- ／ｍａｐ　ｋｍ
最終項目をＫａｎｊｉＭｉｘ変換するときには -------- ／ｍａｐ　・・・　ｋｍ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＡＮＫ・漢字まじり形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後コード変換されます。

ＫａｎｊｉＭｉｘ変換では、つぎの定数が使えます。

文字列定数 -------- 文字列はＡｎｋ・漢字まじり形式、最大２５６文字
半角(')でくくる
※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
汎用定数 ---------- Ｓｐａｃｅ、ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｋｍ［‘ＡＮＫ・漢字まじり’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｋｍ３２　　という指定は、
ｋｍ３２：３２　　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。

ふつうは、ＫＩ／ＫＯが挿入されてデータ長が長くなる分を加算した出力幅を指定します。オーバーフローすると、ＡＮＫ・漢字まじり項目のおわりのほうが切り捨てられます。ただし、漢字モードのままおわることはありません。また、ＫＯが途中で切れることもありません。

出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

ＵｓｅｒＡ変換
ＵｓｅｒＢ変換

```
ＵｓｅｒＡ［入力幅｜＊］［：出力幅］
ＵｓｅｒＢ［入力幅｜＊］［：出力幅］
```

ＵｓｅｒＡ／Ｂ変換は、利用者独自のバイト単位の変換処理が必要なときに、ＡＮＫ変換表ＵｓｅｒーＡ、ＵｓｅｒーＢを書き替えて利用します。なお、ＵｓｅｒＡ／Ｂ変換には、Ａｎｋ変換の説明がほとんどそのまま当てはまります。

入力幅は、　　ｕａ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＵｓｅｒＡ変換するときには ----- ／ｍａｐ　ｕａ
最終項目をＵｓｅｒＡ変換するときには --------- ／ｍａｐ　・・・　ｕａ

のようにして、入力幅を省略できます。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｕａ２０　　という指定は、
ｕａ２０：２０　　という指定と同じです。

項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、ユーザーＡ／Ｂ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のユーザーＡ／Ｂ項目のおわりにＮＵＬ（００Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

Ｎｕｍｅｒｉｃ変換

Ｎｕｍｅｒｉｃ［入力幅｜１５］［：出力幅］

文字形式の数値項目どうしの変換をします。

Ｎｕｍｅｒｉｃ変換は、Ａｎｋ変換と後述のＤｉｓｐＺｏｎｅ変換の中間的なものです。Ａｎｋ変換と比較すると、

文字形式数値しか通さない
入力幅の省略値が１５桁（バイト）である
右詰めになる

などの点が異なります。

「文字形式数値しか通さない」というのは、具体的には、

＋、－、０～９、ピリオド（.）、Ｅ、ｅ、Ｄ、ｄ

しか変換しないで、これら以外の文字は捨ててしまうということです。たとえば、通貨記号（￥／＄）や位取りのコンマ（,）などは削除されるので、リストファイルから入力データファイルを作るときなどに利用できます。

Ｂｉｎａｒｙ変換

Ｂｉｎａｒｙ［入力幅｜＊｜定数］［：出力幅］

Ｂｉｎａｒｙ変換は、「コード変換を一切しない」という変換方法です。ダウンロードデータをＩＢＭ形式に変換するときに多く用いられます。

入力幅は、　　ｂ２０　　のように、１０進のバイト数で指定します。

式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

入力幅の省略値は＊です。いい替えれば、

レコード全体をＢｉｎａｒｙ変換するときには ---- ／ｍａｐ　ｂ
最終項目をＢｉｎａｒｙ変換するときには -------- ／ｍａｐ　・・・　ｂ

のようにして、入力幅を省略できます。

また、入力幅の代わりに定数を指定することもできます。その場合、指定したＢｉｎａｒｙ形式の定数が、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、コード変換されずにそのまま出力されます。

Ｂｉｎａｒｙ変換では、つぎの定数が使えます。

１６進列定数 -------　｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト
汎用定数 ---------- ＬｏｗＶａｌｕｅ、ＨｉｇｈＶａｌｕｅ

定数は、　　ｂ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

出力幅も省略できます。出力幅を省略すると「入力幅と同じ」とみなされます。たとえば、

ｂ２０　　という指定は、
ｂ２０：２０　という指定と同じです。

入力幅の代わりに定数を指定したときは、出力幅の省略値は、「正確な出力に必要な最小限の幅」となります。
項目長を縮めるか、伸ばす場合は、出力幅を１０進のバイト数で指定します。
項目長を縮めると、バイナリ項目のおわりのほうが切り捨てられます。逆に、項目長を伸ばすと、ＩＢＭ側のバイナリ項目のおわりにＮＵＬ（００Ｈ）が詰められます。
出力幅は式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
＊　　ＩＢＭ側（出力）の残りバイト数

漢字イン挿入
漢字アウト挿入

！Ｉ［ｎ］	（英字の“アイ”）
！Ｏ	（英字の“オー”）

！Ｉ、！Ｏでそれぞれ、漢字イン（ＫＩ）、漢字アウト（ＫＯ）を固定位置に挿入できます。

漢字項目の前後で、

！Ｉ　Ｋａｎｊｉ４０　！Ｏ　のように指定するのが、ふつうの使い方です。

この指定とＫａｎｊｉＭｉｘ変換は､似ているところもありますが別物です｡ご注意ください。

！Ｉのあとに、０～２５５の範囲の１０進数を指定することもできます。東芝方式のＡＮＫ・漢字まじり項目化するために「長さバイト」を付加する場合に指定します。

！ｉ４０　ｋ４０　のように指定します。

ＤｉｓｐＺｏｎｅ変換（Ｚｏｎｅ変換）

```
ＤｉｓｐＺｏｎｅ　［入力幅｜１５］：ピクチャ
（Ｚｏｎｅ　［入力幅｜１５］：ピクチャ）
```

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的にバイト数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの符号つきゾーン形式の項目に記録するとすれば、ピクチャは　　ｓ４．１　　と指定します。なお、ピクチャは省略できません。

ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換）

```
ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ
（Ｐａｃｋ　［入力幅｜１５］：ピクチャ）
```

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストのＣＯＢＯＬのパック形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的にバイト数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を３バイトの符号つきパック形式の項目に記録するとすれば、ピクチャは　　ｓ４．１　　と指定します。なお、ピクチャは省略できません。

ＤｉｓｐＰａｃｋ変換（Ｐａｃｋ変換）〔ＢＣＤ形式〕

ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ（ｂ指定）
（Ｐａｃｋ　［入力幅｜１５］：ピクチャ（ｂ指定））

Ｗｉｎｄｏｗｓの文字形式数値項目を、ＰＯＳ端末等で使われているＢＣＤ形式数値項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的にバイト数を指定します。

ピクチャの指定方法については、「操作の基礎」の章で説明したとおりです。たとえば、
１２３．４　という数字を３バイトのＢＣＤ形式の項目に記録するとすれば、ピクチャは

ｂ５．１　　と必ずｂ指定をします。

なお、ピクチャは省略できません。

ＺｏｎｅＤｉｓｐ変換

```
ＺｏｎｅＤｉｓｐ　ピクチャ［：出力幅］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　　と指定します。

なお、ピクチャは省略できません。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＺｏｎｅＤｉｓｐ　ｓ４．１　　という指定は、
ＺｏｎｅＤｉｓｐ　ｓ４．１：７　　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＰａｃｋＤｉｓｐ変換

```
ＰａｃｋＤｉｓｐ　ピクチャ［：出力幅］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、ピクチャは

ｓ４．１　と指定します。

また、ＢＣＤ形式なら

ｂ４．１　と指定します。

なお、ピクチャは省略できません。

パック形式では、整数部桁数＋小数部桁数を奇数にしておくのが通例です。整数部の最上位桁に意味があるのかないのかは、半々の割合です。

出力幅は省略できます。出力幅を省略すると、

符号つきなら１、符号なしなら０とする
＋整数部桁数
＋１＋小数部桁数（小数部があれば）

の要領でピクチャから自動的に計算された値が使われます。例を示すと、

ＰａｃｋＤｉｓｐ　ｓ４．１　という指定は、
ＰａｃｋＤｉｓｐ　ｓ４．１：７　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＺｏｎｅＺｏｎｅ変換

```
ＺｏｎｅＺｏｎｅ　［入力ピクチャ］：出力ピクチャ
                  入力ピクチャ：［出力ピクチャ］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、出力ピクチャは　　ｓ４．１　　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＺｏｎｅＰａｃｋ変換

```
ＺｏｎｅＰａｃｋ　［入力ピクチャ］：出力ピクチャ
                  入力ピクチャ：［出力ピクチャ］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、出力ピクチャは　　ｓ４．１　　（ＢＣＤ形式なら、ｂ４．１）　　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＰａｃｋＺｏｎｅ変換

```
ＰａｃｋＺｏｎｅ　［入力ピクチャ］：出力ピクチャ
                  入力ピクチャ：［出力ピクチャ］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、出力ピクチャは　　ｓ４．１　　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＰａｃｋＰａｃｋ変換

```
ＰａｃｋＰａｃｋ　［入力ピクチャ］：出力ピクチャ
                  入力ピクチャ：［出力ピクチャ］
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目を、ホストのＣＯＢＯＬのパック形式数値項目、ＢＣＤ形式数値項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、出力ピクチャは　　ｓ４．１　　（ＢＣＤ形式なら、ｂ４．１）　　と指定します。

入力ピクチャは省略できます。入力ピクチャを省略すると「出力ピクチャと同じ」とみなされます。出力ピクチャも省略できます。出力ピクチャを省略すると「入力ピクチャと同じ」とみなされます。なお、入力ピクチャと出力ピクチャを同時に省略することはできません。

ＤｉｓｐＢｉｎ変換

ＤｉｓｐＢｉｎ　［入力幅｜１５］：２進キャスト［ピクチャ］

Ｗｉｎｄｏｗｓの文字形式数値項目を、ホストの２進形式整数・小数項目に変換します。

入力幅は、バイト数で指定します。省略すると１５バイトとみなされるので、ふつうは明示的に桁数を指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

例を示すと、

ＤｉｓｐＢｉｎ　１０：ｉ４ｓ４．１　　のような指定になります。

ＢｉｎＤｉｓｐ変換

ＢｉｎＤｉｓｐ　２進キャスト［ピクチャ］:［出力幅］

Ｗｉｎｄｏｗｓの２進形式整数・小数項目を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、 －１２３．４ という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

なお、２進キャストは省略できません。

出力幅は省略できます。出力幅を省略すると、ピクチャに従って出力されます。
出力幅、ピクチャ共に省略すると、

入力のバイト数	１	２	３	４	５	６	７	８
出 力 幅	３	５	８	１０	１３	１５	１７	１８

の要領で２進キャストから自動的に計算された値が使われます。例を示すと、

ＢｉｎＤｉｓｐ　ｉ４ｓ　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ：１０　という指定と同じです。

出力幅を明示的に指定するときは､オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＺｏｎｅＢｉｎ変換

ＺｏｎｅＢｉｎ　入力ピクチャ：２進キャスト［ピクチャ］
［入力ピクチャ］：２進キャスト　ピクチャ

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのゾーン形式数値項目を、ホストの２進形式整数・小数項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が５バイトの符号つきゾーン形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

なお、２進キャストは省略できません。

入力ピクチャも、２進キャストにつづくピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＺｏｎｅＢｉｎ　：ｉ４ｓ４．１　という指定は、
ＺｏｎｅＢｉｎ　ｓ４．１：ｉ４ｓ４．１　という指定と同じです。

ＰａｃｋＢｉｎ変換

```
ＰａｃｋＢｉｎ　入力ピクチャ：２進キャスト［ピクチャ］
　　［入力ピクチャ］：２進キャスト　ピクチャ
```

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパックおよびＢＣＤ形式数値項目を、ホストの２進形式整数・小数項目に変換します。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字が３バイトの符号つきパック形式の項目に記録されているとすれば、入力ピクチャは

ｓ４．１　と指定します。

また、ＢＣＤ形式なら

ｂ４．１　と指定します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、 －１２３．４　という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

なお、２進キャストは省略できません。

入力ピクチャも、２進キャストにつづくピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＰａｃｋＢｉｎ　：ｉ４ｓ４．１　という指定は、
ＰａｃｋＢｉｎ　ｓ４．１：ｉ４ｓ４．１　という指定と同じです。

ＢｉｎＺｏｎｅ変換

ＢｉｎＺｏｎｅ　２進キャスト　ピクチャ：[出力ピクチャ]
２進キャスト［ピクチャ］：出力ピクチャ

Ｗｉｎｄｏｗｓの２進形式整数・小数項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を５バイトの符号つきゾーン形式の項目に記録するとすれば、ピクチャは

ｓ４．１　　と指定します。

２進キャストにつづくピクチャも出力ピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＢｉｎＺｏｎｅ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＺｏｎｅ　ｉ４ｓ４．１：ｓ４．１　　という指定と同じです。

ＢｉｎＰａｃｋ変換

ＢｉｎＰａｃｋ　２進キャスト　ピクチャ：[出力ピクチャ]
２進キャスト［ピクチャ］：出力ピクチャ

Ｗｉｎｄｏｗｓの２進形式整数・小数項目を、ホストのＣＯＢＯＬのパック形式数値項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、　－１２３．４　という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、２進キャストは省略できません。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という数字を３バイトの符号つきパック形式の項目に記録するとすれば、ピクチャは

ｓ４．１　　と指定します。

２進キャストにつづくピクチャも出力ピクチャも省略できますが、両方同時には省略できません。一方を省略すると、指定した方の値で補って実行されます。たとえば、

ＢｉｎＰａｃｋ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＰａｃｋ　ｉ４ｓ４．１：ｓ４．１　　という指定と同じです。

ＢｉｎＢｉｎ変換

ＢｉｎＢｉｎ　２進キャスト［ピクチャ］:［２進キャスト［ピクチャ］］
［２進キャスト［ピクチャ］］：２進キャスト［ピクチャ］

Ｗｉｎｄｏｗｓの２進形式整数･小数項目を、ホストの２進形式整数･小数項目に変換します。

２進キャスト／ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、 －１２３．４ という数字が４バイトの符号つき２進形式の項目に記録されているとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

－１２３．４ という数字を４バイトの符号つき２進形式の項目に記録するとすれば、２進キャスト／ピクチャは

ｉ４ｓ４．１　と指定します。

２進キャスト／ピクチャは入力または出力のどちらかを省略できます。入力を省略すると「出力と同じ」とみなされます。たとえば、

ＢｉｎＢｉｎ　：ｉ４ｓ４．１　という指定は、
ＢｉｎＢｉｎ　ｉ４ｓ４．１：ｉ４ｓ４．１　という指定と同じです。

出力を省略すると「入力と同じ」とみなされます。たとえば、

ＢｉｎＢｉｎ　ｉ４ｓ４．１　という指定は、
ＢｉｎＢｉｎ　ｉ４ｓ４．１：ｉ４ｓ４．１　という指定と同じです。

なお、入力２進キャスト／ピクチャと出力２進キャスト／ピクチャを同時に省略することはできません。

ＢｉｎａｒｙＸ変換

ＢｉｎａｒｙＸ｛幅｜定数｝

ＢｉｎａｒｙＸ変換は、「コード変換を一切せずに、幅分のデータをバイト単位で左右反転する」という変換方法です。２進数値データ（整数、小数、実数）は、ＩＢＭ側が正順であるのに対して、Ｗｉｎｄｏｗｓ側が逆順であることが多いので、２進数値データの内容をそのままバイト単位で左右反転する場合等に使用します。ＢｉｎＢｉｎ変換のような加工機能はありませんが、その分だけ処理が高速です。

幅は、　　ｂｘ８　　のように、１０進のバイト数で指定します。

たとえば、　ｂｘ４　のように指定し、１６進表現で入力データが　０１　ＡＢ　ＣＤ　ＥＦ　であれば、出力データは　ＥＦ　ＣＤ　ＡＢ　０１　になります。

また、幅の代わりに定数を指定することもできます。その場合、指定したバイナリ形式の定数は、Ｗｉｎｄｏｗｓ側（入力）レコードにあったかのようにみなされ、その後左右反転されます。

ＢｉｎａｒｙＸ変換では、つぎの定数が使えます。

１６進列定数 -------　｛Ｘ｜ｘ｝‘１６進列’のように半角(')でくくる
１６進列は０～９、Ａ～Ｆ（ａ～ｆ）、２桁で１バイト
任意のバイト境界に半角スペースを挿入できる
最大２５６バイト

定数は、　　ｂｘ［Ｘ‘０Ａ　Ｆ１’］　のように、［　］でくくります。

ＴｏＤｉｓｐ変換

ＴｏＤｉｓｐ｛数値定数｜システム変数｝[：出力幅]

入力した数値定数、またはシステム変数の値を、ホストの文字形式数値項目に変換します。変換結果は右詰めになります。

入力値として、つぎの数値定数、システム変数が使えます。省略はできません。

数値定数　　**整数定数、小数定数、１６進定数**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　[－１２３．４]　のように、[　]でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

出力幅は省略できます。出力幅を省略すると、入力した値を正確に出力するために必要な、最小限の幅で出力されます。たとえば、

ＴｏＤｉｓｐ　[－１２３．４]　　という指定は、
ＴｏＤｉｓｐ　[－１２３．４]：６　という指定と同じです。

出力幅を明示的に指定するときは、オーバーフローに注意しながら１０進のバイト数で指定します。オーバーフローすると、符号や上位桁が切り捨てられるので、注意してください。

ＴｏＺｏｎｅ変換

ＴｏＺｏｎｅ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、５バイトの符号つきゾーン形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＺｏｎｅ　　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＰａｃｋ変換

ＴｏＰａｃｋ｛数値定数｜システム変数｝：出力ピクチャ

入力した数値定数、またはシステム変数の値を、ホストのＣＯＢＯＬのパック形式数値項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、３バイトの符号つきパック形式の項目に記録するとすれば、出力ピクチャは

ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＰａｃｋ　［－１２３．４］：ｓ４．１　　のような指定になります。

ＴｏＢｉｎ変換

ＴｏＢｉｎ｛数値定数｜システム変数｝：２進キャスト［ピクチャ］

入力した数値定数、またはシステム変数の値を、ホストの２進形式整数・小数項目に変換します。

入力値として、つぎの数値定数、システム変数が使えます。

数値定数　　**整数定数、小数定数、１６進定数、汎用定数（Ｚｅｒｏ、Ｍｉｎ、Ｍａｘ）**
システム変数　　SysPhase、SysRecNum、SysBreak、SysReturn、SysQuit

数値定数、またはシステム変数は、　［－１２３．４］　のように、［　］でくくります。システム変数の詳細は、マニュアルのマルチレコード編を参照してください。

ピクチャの指定方法については、「操作の基礎」の章で説明しました。たとえば、
－１２３．４　という定数を入力値として指定して、４バイトの符号つき２進形式の項目に記録するとすれば、出力ピクチャは

ｉ４ｓ４．１　　と指定します。

なお、入力値、出力ピクチャともに省略できません。

例を示すと、

ＴｏＢｉｎ　　［－１２３．４］：ｉ４ｓ４．１　　のような指定になります。

Ｙｅａｒ設定（年設定）

```
Ｙｅａｒ　　Ｗｎｎ｜Ｓｎｎ
                          ：Ｗｎｎ｜：Ｓｎｎ
              Ｗｎｎ｜Ｓｎｎ　：Ｗｎｎ｜：Ｓｎｎ
```

日付データ項目を変換する際の、年の２桁（ｙｙ）と４桁（ｙｙｙｙ）の変換方式を設定します。Ｗｎｎの“W”はウインドウ方式を、Ｓｎｎの“Ｓ”はシフト方式を意味し、“ｎｎ”は００～９９の数字で指定します。“：”の左側の指定は入力側、“：”の右側の指定は出力側への適用になり、入力側のみ、出力側のみ、入出力両方の３通りの指定ができます。たとえば、

Ｙｅａｒ	Ｗ３０	入力データの年を１９３０～２０２９年とみなす
Ｙｅａｒ	：Ｓ２５	出力データの年下２桁を、－２５する
Ｙｅａｒ	Ｗ３０：Ｓ２５	入力データの年を１９３０～２０２９年とみなし、 出力データの年下２桁を、－２５する

のように指定します。

また、シフト方式（“Ｓｎｎ”指定）では、つぎの特殊指定ができます。

Ｙｅａｒ	ＳＳｈｏｗａ	“Ｓ２５”の指定と同じ（昭和通年方式）
Ｙｅａｒ	ＳＨｅｉｓｅｉ	“Ｓ８８”の指定と同じ（平成通年方式）

Ｙｅａｒ設定はＤａｔｅ変換が実行された時に適用になり、複数のＹｅａｒ設定がなされている場合は、Ｄａｔｅ変換の直前のＹｅａｒ設定が有効になります。

Ｙｅａｒ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

Ｙｅａｒ	Ｗ３０：Ｗ３０	入出力データの年を１９３０～２０２９年とみなす

ＤａｔｅＤｅｌｉｍ設定（日付区切り設定）

```
ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ｜ＨＹＰＨＥＮ｜ＰＥＲＩＯＤ
```

日付データ項目を出力する際の日付区切り記号をつぎの３つの中から設定します。

指定文字	日付区切り記号	データ例
ＳＬＡＳＨ	／（スラッシュ）	１９９８／１２／３１
ＨＹＰＨＥＮ	－（ハイフン）	１９９８－１２－３１
ＰＥＲＩＯＤ	.（ピリオド）	１９９８．１２．３１

ＤａｔｅＤｅｌｉｍ設定はＤａｔｅ変換が実行された時に適用になり、複数のＤａｔｅＤｅｌｉｍ設定がなされている場合は、Ｄａｔｅ変換の直前のＤａｔｅＤｅｌｉｍ設定が有効になります。

ＤａｔｅＤｅｌｉｍ設定がない場合のＤａｔｅ変換のデフォルトは、つぎの指定になります。

ＤａｔｅＤｅｌｉｍ　ＳＬＡＳＨ　日付区切り記号を“／”にする

Ｄａｔｅ変換

```
Ｄａｔｅ　入力日付マスク：出力日付マスク
```

日付データ項目を変換します。

入力日付マスク、出力日付マスクは必ず指定します。省略はできません。たとえば、

Ｄａｔｅ　ｙｙｙｙ－ｍｍ－ｄｄ：ｙｙｍｍｄｄ

のように指定すると、コード変換後に、入力側１０バイトの日付データ項目を、出力側６バイトの日付データ項目に編集します。その際に、Ｙｅａｒ設定、ＤａｔｅＤｅｌｉｍ設定が適用になります。

桁移動

＠入力桁位置
＠：出力桁位置

変換対象にするＷｉｎｄｏｗｓ側（入力）の桁位置や、変換結果を書き込むＩＢＭ側（出力）の桁位置を、別の任意の位置に移動できます。現在、処理対象にしている桁位置を、この機能で強制的に変更できます。この機能を利用すると、項目の組み替えなどが簡単に実現できます。

入力桁位置は、ふつう１０進数で指定します。式による指定もでき、そのなかでは、

＄　　Ｗｉｎｄｏｗｓ側（入力）のレコード長
．　　Ｗｉｎｄｏｗｓ側（入力）の現在の桁位置

という特殊変数が使えます。たとえば、

＠．－４０　Ｋａｎｊｉ２０　＾２０　　と指定すれば、

現在の入力桁位置から４０バイト戻り、漢字２０バイト変換せよ
そして、元の位置に戻れ

という意味になります。

出力桁位置を移動することもできます。ふつう１０進数で桁位置を指定します。式による指定もでき、そのなかでは、つぎの特殊変数が使えます。

＄　　ＩＢＭ側（出力）のレコード長
．　　ＩＢＭ側（出力）の現在の桁位置

たとえば、

＠：．－４０　　と指定すれば、

現在の出力桁位置から４０バイトバックせよ

という意味になります。

◆注意　----　先頭を０桁目とする
Ｆ＊ＴＲＡＮ２００７では、レコードの先頭を０桁目として数えます。

入力スキップ

```
^［n｜1］
```

Ｗｉｎｄｏｗｓ側（入力）レコードに不要な項目があるとき、それをスキップして変換できます。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

^３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

^３　　は　　^^^　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかで、つぎの特殊変数が使えます。

＊　　Ｗｉｎｄｏｗｓ側（入力）の残りバイト数

これを使えば、　　／ｍａｐ　・・・　^＊　　として「残りは全部スキップせよ」という指定もできます。しかし、何の指定もなかった分は変換対象にならないので、この ^ ＊は冗長です。省いたほうがよいでしょう。

出力スキップ

```
_［n｜1］
```

入力スキップとは逆に、ＩＢＭ側（出力）に何桁かの空きを作ることもできます。空項目を作るのがおもな用途です。スキップする幅は、バイト数で指定します。たとえば、３バイト分スキップしたいなら、

_３　　と指定します。

バイト数は省略でき、省略すると１バイトとみなされるので、

_３　　は　　_ _ _　　と指定したのと同じです。

スキップする幅は式による指定もでき、そのなかで、つぎの特殊変数が使えます。

＊　　ＩＢＭ側（出力）の残りバイト数

／ＰＨａｓｅ　　---- ヘッダ・トレーラの生成・付加のタイミングを指定する

```
／ＰＨａｓｅ　　［Ｏｐｅｎ＜ｎ＞］,
　　［Ｍａｉｎ＜ｎ＞］,
　　［Ｃｌｏｓｅ＜ｎ＞］
```

データを変換する際にヘッダやトレーラを付加したいときに、この／ＰＨａｓｅオプションで各フェーズの呼び出しが起きる回数を指定します。フェーズにはつぎの３種類があります。

Ｏｐｅｎフェーズ	ヘッダの生成・付加を行う
Ｍａｉｎフェーズ	本体の変換をする
Ｃｌｏｓｅフェーズ	トレーラやエンドレコードの生成・付加を行う

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数を、＜ｎ＞に１０進数値で指定します。Ｍａｉｎフェーズでは、

＄　　全レコード分

という指定もできます。
Ｏｐｅｎフェーズ→Ｍａｉｎフェーズ→Ｃｌｏｓｅフェーズの順に呼び出しを行います。

◆注意　----　フェーズごとの指定について

／ＰＨａｓｅオプションでは、各フェーズでのＡｔｌａｓ呼び出し回数のみ、指定します。各フェーズごとの動作は、Ａｔｌａｓ構文を使って設定する必要があります。詳細はマルチレコード編のマニュアルを参照してください。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｓｉｚｅ　＄	Ｗｉｎｄｏｗｓレコード長＝ＩＢＭレコード長
／ＭＡＰ　Ａｎｋ	レコード全体をＡＮＫデータとして変換、 改行コードなし
／ＮＥＷ	新規ファイルとして作成する
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替えない（中断）
／Ｆｏｒｍ　Ｅｘｔｅｎｄｅｄ　２５６　Ａｕｔｏ　ＦＢ　Ｃｌｉｐ	レコード長＝２５６バイトの、拡張形式で適当に ブロック化したファイルを作る
／ＥｘｔｒａＳｐａｃｅ　０	予備領域をとらない
／ＢｏｅＢｏｕｎｄａｒｙ　Ｔｒａｃｋ	ＩＢＭファイルをトラック境界ではじめる
／ＥｏｅＢｏｕｎｄａｒｙ　＊	ＩＢＭファイル領域のおわり方は、 ＢｏｅＢｏｕｎｄａｒｙオプションに従う
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動変換する
／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋ	シングルボリューム時、ボリューム順序番号 空白にする
／ＮｏＥＯＦ	ＥＯＦコードを検査しない
／ＰＨａｓｅ　Ｏｐｅｎ０，Ｍａｉｎ＄，Ｃｌｏｓｅ０	ヘッダ・トレーラをつけないで全レコード変換する

と指定したものとして、ランダムファイル変換を行います。

■解説

●／ＭＡＰオプションと／Ｓｉｚｅオプションの指定方法

ＩＢＭ形式の、つぎのようなレコードレイアウトのファイルＰＬＡＮＥＴを作成するものとします。

ＰＬＡＮＥＴのレコードレイアウト

項番	項　　目	桁	幅	データ形式	内容例
1	Ｎｏ.	0	2	ＡＮＫ	1
2	和名	2	8	漢字	水星
3	英名	10	10	ＡＮＫ	MERCURY
4	読み	20	9	ＡＮＫ	ﾏｰｷｭﾘｰ
5	質量比	29	4	符号なしパック形式 整数部４桁、小数部３桁	0.055
6	衛星数（確定済）	33	2	符号なしゾーン形式 整数部２桁	0
7	極大等級	35	3	符号つきゾーン形式 整数部２桁、小数部１桁	-2.4
8	英名の意味・由来	38	20	漢字	口神）神の使者
--	フィラー	58	6	--	--

これをランダムファイル変換するときのオプションの指定は、

```
/f64 /s64 /map a2 k8 a10 a9 dp8:u4.3 dz2:u2 dz5:s2.1 k20
```

のようになります。このように、

Ｗｉｎｄｏｗｓ側（入力）レコード長は／Ｓｉｚｅオプションで
ＩＢＭ側（出力）レコード長は／Ｆｏｒｍオプションで

決めます（この例では、偶然両者の値が同じになってしまいました）。

●変換テストから本番まで

ＰｕｔＲａｎｄ(ｐｒ)コマンドでＩＢＭ形式に変換しても、うまく変換できているかどうかわかりにくいものです。ＩＢＭディスクエディタ（ＧＵＩ部）で、できたＩＢＭファイルを直接見る方法はやや専門的な知識を必要とします。簡単な方法は、ＧｅｔＤａｔａ(ｇｄ)コマンドを使って逆方向の変換をしてみることです。

つぎのような変換用バッチファイルと、

＜ＰＬＡＰＵＴＲＮ．ＢＡＴ（その１）＞

```
start /w ft putrand c:planet.ran a:/map a2 k8 a10 a9 dp8:u4.3 dz2:u2 dz5:s2.1
 k20 /f64/s64 ↓
```

逆変換用バッチファイルを作り、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ＞

```
start /w ft getdata c:planet test /dnc /map a2, k8, a10, a9, dpu4.3, dzu2,
 dzs2.1, k20 ↓
```

テストランと修正を繰り返しながら変換テストを進めていきます。

変換テストの段階では、変換結果が簡単にわかるように、

圧縮なしのコンマ区切り（ＣＳＶ）形式にし

ていることに注意してください。

「これでＯＫ」となれば、このテスト用バッチファイルを修正して、本番用のバッチファイルにします。簡単な本番用バッチファイルは、

＜ＰＬＡＰＵＴＲＮ．ＢＡＴ（その２）＞

```
@echo off
start /w ft /dc/ putrand c:planet.ran a:/f64 /rep /s64 /map a2 k8 a10 a9
 dp8:u4.3 dz2:u2 dz5:s2.1 k20 ↓
```

のようなスタイルになります。

●パラメータファイルを使う

パラメータファイルを利用すると、さらに運用と保守・管理が簡単になります。上の例は、つぎのようなバッチファイルと、

＜ＰＬＡＰＵＴＲＮ．ＢＡＴ（その３）＞

```
@echo off ↓
start /w ft /dc/ putrand c:planet.ran a: ++plaputrn.p ↓
```

つぎのようなパラメータファイルに、

＜ＰＬＡＰＵＴＲＮ．Ｐ＞

```
/size 64 ↓
/form 64 ↓
/replace ↓
/map ↓
      ank        2          -- Ｎｏ．（惑星番号）　↓
      kanji      8          -- 和名 ↓
      ank        10         -- 英名 ↓
      ank        9          -- 読み ↓
      disppack   8:u4.3     -- 質量比 ↓
      dispzone   2:u2       -- 衛星数（確定済）　↓
      dispzone   5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      kanji      20         -- 英名の意味・由来 ↓
```

分けて記述できます。これで、大幅にわかりやすくなりました。

●いろいろな加工・編集の方法

いらない項目をスキップする

ホストにデータを持っていくと、いらない項目がいくつもあることが多いものです。それをスキップして変換するのは簡単で、

```
^n
```

と書けばよいのです。ｎはスキップするバイト数です。たとえば、例題のデータから「英名」「読み」「英名の意味・由来」の項目を変換対象からはずすなら、／ＭＡＰオプションの指定は、

```
/map a2 k8 ^10 ^9 dp8:u4.3 dz2:u2 dz5:s2.1
```

となります。「英名の意味・由来」の項目は最後の項目なので、^20 とは書かずに省略しました。

空項目を追加する

空項目を作るのも簡単で、

```
_n
```

と書くだけです。例題のデータに新たに「ボイジャーⅠ通過日」「ボイジャーⅡ通過日」というＹＹＭＭＤＤ形式の項目を追加する場合は、

```
/map a2 k8 a10 a9 _6 _6 dp8:u4.3 dz2:u2 dz5:s2.1 k20
```

とする要領です。この例では英名の「読み」の項目のうしろに、つづけて２つ追加してみました。

項目長の増減

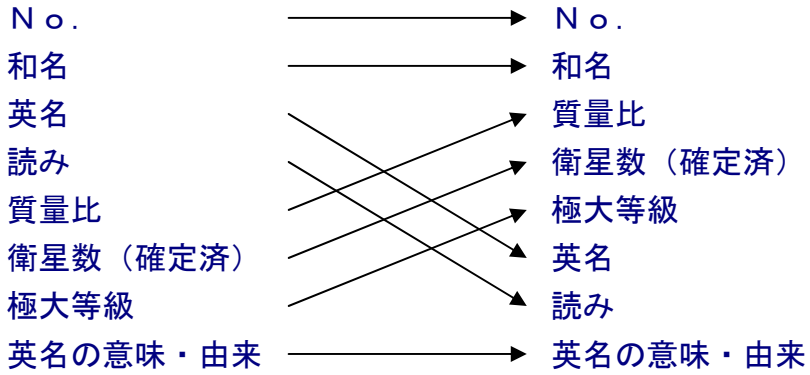
必要に応じて項目長を増減するのも簡単です。つぎのように、

```
/map a2:3 k8:10 a10 a9 dp8:u5.3 dz2:u3 dz5:s2.1 k20:30
```

とします。基本的にはコロン(:)のあとに新しい出力幅や出力ピクチャを指定するだけです。

項目を組み替える

項目組み替えは、桁移動の機能を利用するので少し面倒です。例題のデータを、



というふうに組み替えるには、つぎのようなパラメータファイルを作ります。

```
/map ↓
      ank         2          -- Ｎｏ．（惑星番号）　↓
      kanji       8          -- 和名 ↓
      ank         10         -- 英名 ↓
      ank         9          -- 読み ↓
      disppack    8:u4.3     -- 質量比 ↓
      dispzone    2:u2       -- 衛星数（確定済）　↓
      dispzone    5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      kanji       20         -- 英名の意味・由来 ↓
```

つぎに、各項目の桁位置をつけていきます（@0～@44）。

```
/map ↓
      @0   ank         2          -- Ｎｏ．（惑星番号）　↓
      @2   kanji       8          -- 和名 ↓
      @10  ank         10         -- 英名 ↓
      @20  ank         9          -- 読み ↓
      @29  disppack    8:u4.3     -- 質量比 ↓
      @37  dispzone    2:u2       -- 衛星数（確定済）↓
      @39  dispzone    5:s2.1     -- 極大等級（みかけ上の最大の明るさ）↓
      @44  kanji       20         -- 英名の意味・由来 ↓
```

こうしておいて、あとはエディタで好きなように順番を入れ替えていきます。たとえば、

```
/map ↓
      @0   ank         2          -- Ｎｏ．（惑星番号）　↓
      @2   kanji       8          -- 和名 ↓
      @29  disppack    8:u4.3     -- 質量比 ↓
      @37  dispzone    2:u2       -- 衛星数（確定済）　↓
      @39  dispzone    5:s2.1     -- 極大等級（みかけ上の最大の明るさ）　↓
      @10  ank         10         -- 英名 ↓
      @20  ank         9          -- 読み ↓
      @44  kanji       20         -- 英名の意味・由来 ↓
```

とします。

■使用例

例１）ＡＮＫコードのみのファイルを変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴを、ドライブＡ：のＩＢＭファイルＪＯＵＲＮＡＬに変換します。レコードに含まれるのはＡＮＫデータだけで、それをホストのコードに変換します。レコード長は２５６バイトです。

```
C:¥>ft putrand c:journal.dat a:* ↓
```

例２）ＡＮＫコードのみのファイルを変換し、適当にブロック化する

ドライブＣ：のＷｉｎｄｏｗｓファイルＬＯＧ．ＤＡＴを、ドライブＡ：のＩＢＭファイルＬＯＧに変換します。内容はＡＮＫデータだけで、それをホストのコードに変換します。レコード長は６４バイトです。ＩＢＭファイルは拡張形式で、レコード長を同じく６４バイトにし、適当にブロック化します。

```
C:¥>ft putrand c:log.dat a:/f64 ↓              (/Form 64)
```

例３）バイナリ変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＦＯＮＴ２４．ＰＸＬを、ドライブＡ：のＩＢＭファイルＦＯＮＴ２４にバイナリ変換します。ＩＢＭファイルは基本形式にし、同名のＩＢＭファイルがあれば強制的に置き替えます。

```
C:¥>ft putrand c:font24.pxl a:/map b /fp /rep ↓
                              (/MAP Binary /Form Primary /REPlace)
```

例４）有効データだけ変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＺＡＩＫＯ０１．ＤＡＴ～ＺＡＩＫＯ１２．ＤＡＴはレコード長が２５６バイトで、有効なデータは先頭の８０バイトだけです。そして、その内容は、ＡＮＫデータだけです。これをドライブＡ：のＩＢＭファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２に変換します。

```
C:¥>ft putrand c:zaiko*.dat a:/s256 /f80 ↓     (/Size 256 /Form 80)
```

例５）項目別変換する

ドライブＣ：にＷｉｎｄｏｗｓファイルＡＤＤＲＥＳＳ．ＤＢがあり、内容は住所録でつぎのように項目が分かれています。

項　目	タイプ	幅	
氏名	漢字	１６	
フリガナ	ＡＮＫ	１６	計３４バイト
電話番号	ＡＮＫ	１２	
郵便番号	ＡＮＫ	６	
住所	漢字	４０	
生年月日	ＹＹＹＹ－ＭＭ－ＤＤ	１０	
区分	ＡＮＫ	１	

レコード長は２５６バイトです。これをドライブＡ：のＩＢＭファイルＡＤＤＲＥＳＳに変換します。ＡＮＫ項目、漢字項目、日付項目が混在しているので、項目別変換します。

```
C:¥>ft putrand c:address.db a:/map k16 a34 k40 dyyyy-mm-dd:yymmdd a1 ↓
                (/MAP Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd Ank1)
```

例６）項目を組み替える（その１）

例５と同様ですが、入力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft putrand c:address.db a:/map @100 a1 @1 k16 a34 k40 dyyyy-mm-dd:yymmdd ↓
                (/MAP @100 Ank1 @1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd)
```

例７）項目を組み替える（その２）

例５と同様ですが、出力桁移動の機能を利用して「区分」の項目をレコードの先頭に持ってきます。ほかの項目は順にうしろにずらします。

```
C:¥>ft putrand c:address.db a:/map @:1 k16 a34 k40 dyyyy-mm-dd:yymmdd @:0 a1 ↓
                (/MAP @:1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd @:0 Ank1)
```

例８）パラメータファイルを利用する

例５と同じ処理を、パラメータファイルＡＤＤＲ．Ｐを使って変換します。このパラメータファイルはインストールディレクトリに置きます。パラメータファイルには、日付項目変換のための年設定を追加します。

```
C:¥>ft putrand c:address.db a: ++addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/map ↓
      kanji  16                    -- 氏名 ↓
      ank    16                    -- フリガナ ↓
      ank    12                    -- 電話番号 ↓
      ank    6                     -- 郵便番号 ↓
      kanji  40                    -- 住所 ↓
      year   :sshowa               -- 出力日付年＝昭和 ↓
      date   yyyy-mm-dd:yymmdd     -- 生年月日 ↓
      ank    1                     -- 区分 ↓
```

■注意事項

漢字変換をするときは、漢字変換方式の割り当てを忘れずに

Ｋａｎｊｉ変換やＫａｎｊｉＭｉｘ変換を行うときは、あらかじめ

適切な漢字変換方式を割り当てておく（通常、セットアップ時に行う）

のを忘れないでください。また、入力幅や出力幅は漢字データについても

バイト単位で指定

します。漢字の文字数ではないことに注意してください。

その他の注意事項

「Ｐｕｔ系コマンド」の節を参照してください。

２．11 ＶｉｒＤｒｉｖｅ（ｖｄ）　バーチャル・ドライブ

仮想ドライブの設定

ＶｉｒＤｒｉｖｅ(ｖｄ)は､Ｗｉｎｄｏｗｓ側の各ドライブの特定ディレクトリを簡易的にアクセスできるようにするためのコマンドです。

このコマンドを使うと、アクセスしたい実ドライブ・ディレクトリに適当な仮想ドライブ名をつけることができます。以後、その仮想ドライブ名で目的の実ドライブ・ディレクトリをアクセスできます。設定はＦ＊ＴＲＡＮ２００７を終了するまで有効です。

■コマンド形式

コマンド	パラメータ
ＶｉｒＤｒｉｖｅ ｖｄ	仮想ドライブ名　［実ドライブ・ディレクトリ］］

■パラメータの説明

●仮想ドライブ名

```
ｄ：
```

ｄ：はドライブ名

仮想ドライブ名として、つぎのドライブ名が指定できます。

Ａ：　～　Ｚ：　　自由に仮想ドライブ名として使えます。
仮に実ドライブがあっても使えます。

＠：　　Ｆ＊ＴＲＡＮ２００７特有のものです。カレントドライブのカレントディレクトリを表すので、原則として使いません。

？：　　Ｆ＊ＴＲＡＮ２００７特有のものです。インストールディレクトリを表すので、原則として使いません。

●実ドライブ・ディレクトリ

```
ｄ：
ｄ：￥
ｄ：￥ｄｉｒ
ｄ：￥ｓｕｂ￥ｄｉｒ
ｄ：ｄｉｒ　　など
```

ｄ：は実ドライブ名

仮想ドライブに対応づける実ドライブ・ディレクトリを指定します。**ふつうは、目的のディレクトリを、ルートディレクトリからの絶対パス名で指定します。実ドライブ名もつけます。**

■使用例

例）Ｃ：→Ｃ：￥ＵＳＲ￥ＤＡＴとしたあと、ファイル変換する

ドライブＡ：のＩＢＭファイルを、すべてドライブＣ：のディレクトリ￥ＵＳＲ￥ＤＡＴのなかにデータファイル変換するように指定します。元のファイル名を引き継ぎ、拡張子は．ＤＡＴにします。

```
C:¥>ft virdrive c: c:¥usr¥dat ; getdata a:* c:*.dat ↓
```

■注意事項

＠：と？：は特別扱い

＠：と？：は起動時にすでに設定されています。

実ドライブ・ディレクトリは検査しない

ＶｉｒＤｒｉｖｅ（ｖｄ）コマンドで設定をした時点では、対応する実ドライブやディレクトリがあるかどうかは検査されません。ほかのコマンドで仮想ドライブにアクセスしたとき、はじめて検査されます。

未設定は実ドライブへのアクセスに

実ドライブ・ディレクトリが設定されていないドライブをほかのコマンドで指定したときは、実ドライブへのアクセスとみなされます。

２．12　ｉＬｉｓｔ（ｉｌ）　アイ・リスト

ＩＢＭファイル名・属性一覧

ｉＬｉｓｔ(ｉｌ)は、ＩＢＭディスクのインデックスシリンダ（ディレクトリに相当）の内容を表示するコマンドです。表示内容はボリューム情報とＩＢＭファイル名・属性（レコード長、ブロック長、レコード形式など）の一覧です。

■コマンド形式

コマンド	パラメータ
ｉＬｉｓｔ ｉｌ	ＩＢＭファイル　［オプション］

■パラメータの説明

●ＩＢＭファイル

ｄ：［ＩＢＭファイル名
＊］

ｄ：はドライブ名

表示するＩＢＭディスク・ファイルを指定します。

ドライブ名はＡ：～Ｐ：、または０：～３：で指定します。省略はできません。

ＩＢＭファイル名にはワイルドカード文字（＊）が使えます。省略もできます。ＩＢＭファイル名が省略されると＊を指定したものとみなし､指定したドライブのＩＢＭディスクのＩＢＭファイル情報をすべて表示します。

ｉＬｉｓｔ(ｉｌ)コマンドを使うときは、

Ｃ：￥＞ｆｔ　ｉｌｉｓｔ　ａ：↓

のようにドライブ名だけを指定するのがふつうです。

●指定できるオプション

ｉＬｉｓｔ(ｉｌ)コマンドには、

／Ｌｏｎｇ	ロング形式で表示する
／Ｓｈｏｒｔ	ショート形式で表示する
／Ｗ	／Ｓｈｏｒｔと同じ

というオプションがあります。どれも表示方法を指定するオプションです。

●オプションの詳細

／Ｌｏｎｇ ---------- ロング形式で表示する
／Ｓｈｏｒｔ　ｏｒ　／Ｗ ---- ショート形式で表示する

／Ｌｏｎｇ
／Ｓｈｏｒｔ
／Ｗ

ＩＢＭファイル名・属性の表示形式を指定します。

／Ｌｏｎｇオプションを指定すると、１行に１ファイルの形式（ロング形式）で詳しい情報を表示します。

／Ｓｈｏｒｔオプションを指定すると、４ファイル／１行の横並びの形式（ショート形式）でファイル名だけを表示します。／Ｗは／Ｓｈｏｒｔと同じです。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｌｏｎｇ	ロング形式（１ファイル／１行、属性も表示）

と指定したものとみなします。ふつう、これで十分です。

◆注意　----　Ｆ＊ＴＲＡＮⅢにあった／Ｐａｕｓｅオプションはない

Ｆ＊ＴＲＡＮⅢにあった／Ｐａｕｓｅ、／ＮｏＰａｕｓｅオプションは、機能的な意味を持たないため、なくなりました。

■解説

●ボリューム情報

ｉＬｉｓｔ(ｉｌ)コマンドを実行すると、ロング形式でもショート形式でも最初の１行にそのＩＢＭディスクのボリューム全体に関する情報を、

```
ﾀｲﾌﾟ: 一般(360rpm)  媒体種別: 2HD-256  ﾎﾞﾘｭｰﾑ ID: FTRAN  所有者名: FUJITSU BSC
```

のように表示します。これで、

タイプ：	一般(360rpm)：一般のＩＢＭ形式（1.2MB ﾌｫｰﾏｯﾄ、EBCDIC ﾗﾍﾞﾙ）
	東芝(360rpm)：東芝形式（1.2MB ﾌｫｰﾏｯﾄ、JIS8/ASCII ﾗﾍﾞﾙ）
	三菱(300rpm)：三菱形式（1.44MB ﾌｫｰﾏｯﾄ、EBCDIC ﾗﾍﾞﾙ）
媒体種別：	ＩＢＭディスクの種別（1S-128～2HD-1024）
ボリュームＩＤ：	ボリュームの識別コード（ボリューム通番）
所有者名：	所有者の名前（つけないことが多い）

を知ることができます。これらは、ｉＦｏｒｍａｔ(ｉｆｏ)コマンドでディスクを初期化するときに指定できます。また、ＩＢＭファイラのボリュームＩＤ変更機能で、ボリュームＩＤと所有者名の修正ができます。それらの解説も参考にしてください。

●ロング形式の表示とその内容

ロング形式の場合、ボリューム情報につづけてＩＢＭファイル名と各種属性が、

```
ＩＢＭファイル名   順番 ﾚｺｰﾄﾞﾌﾞﾛｯｸ 形式 ﾌﾗｸﾞ  BOE    EOE    EOD    作成  ﾚｺｰﾄﾞ数 満了
-------- -------- -- - ---- ----- ---- pbvd cchrr-cchrr cchrr yymmdd ------ yymmdd
SOLARSYSTEM            500   500 eF   .... 01001-01008 01009 960704      4
SUN                    256   256 prim .... 01101-01126 01112 960704     11
PLANET                  64   256 eFB  .... 02001-02026 02004 960918     10
EARTH                   80   240 eFB  .... 15001-22126 22107 961003   1110
JUPITER           01 C  80   240 eFB  .... 67001-74126 75001 961003   1248
```

のように表示されます。それぞれの欄とその内容は、

ＩＢＭファイル名	ＩＢＭファイルの名前
順番	ボリューム順序番号、マルチボリュームフラグ
レコード	論理レコード長
ブロック	ブロック長
形式	データ交換形式とレコード形式
フラグ	各種の管理用フラグ
ＢＯＥ	領域開始アドレス（ＣＣＨＲＲ形式）
ＥＯＥ	領域終了アドレス（ＣＣＨＲＲ形式） このセクタ（含む）までがファイル領域 ＥＯＥ＝ＥＯＤ－１になり得る
ＥＯＤ	データ終了アドレス（ＣＣＨＲＲ形式） データの入っている最終ブロックの、つぎのセクタのアドレス
作成	ファイル作成日付（西暦ＹＹＭＭＤＤ形式）
レコード数	レコード件数
満了	ファイル満了日付（西暦ＹＹＭＭＤＤ形式）

という意味です。つぎのページにさらに詳しい説明を載せます。

◆参考...

ＢＯＥ、ＥＯＥ、ＥＯＤはそれぞれ、

ＢＯＥ　Ｂｅｇｉｎｎｉｎｇ　Ｏｆ　Ｅｘｔｅｎｔ
ＥＯＥ　Ｅｎｄ　Ｏｆ　Ｅｘｔｅｎｔ
ＥＯＤ　Ｅｎｄ　Ｏｆ　Ｄａｔａ

の略です。

◆参考...

ｉＬｉｓｔ(ｉｌ)コマンドには、ロング形式でもファイルごとのブロック数・セクタ数を表示する機能はありません。

ファイル情報の詳細

欄	内　　容
IBM ﾌｧｲﾙ名	ＩＢＭファイルの名前（左詰め、空白不可、ふつう８文字以内）
順　番	ボリューム順序番号とマルチボリュームフラグ 空　白　　単一ボリューム ０１　　〃　　（大部分のシステムで） ｎｎ　Ｃ　　nn 番目で、次のﾎﾞﾘｭｰﾑ につづく（nn が空白なら順不定） ｎｎ　Ｌ　　nn 番目で、最終ﾎﾞﾘｭｰﾑ である　（nn が空白なら順不定）
レコード	論理レコード長（ふつう、ブロック長と同じか、それ以下）
ブロック	ブロック長（基本：セクタ長以下／拡張・半拡張：トラック容量以下）
形　式	データ交換形式とレコード形式 ｐｒｉｍ　基本形式 ｅＦ　拡張形式、固定長レコード・非ブロック化・非スパン ｅＦＢ　〃　、固定長レコード・ブロック化・非スパン ｅＦＳ　〃　、固定長レコード・非ブロック化・スパン ｅＦＢＳ　〃　、固定長レコード・ブロック化・スパン ｈｅＦ　半拡張形式、固定長レコード・非ブロック化　*1 ｈｅＦＢ　〃　、固定長レコード・ブロック化　*1 ？？？　形式不明（処理不可）
フラグ	各種の管理用フラグ
ｐの桁	プロテクトフラグ　．　書き込み可 Ｐ　書き込み禁止
ｂの桁	バイパスフラグ　．　処理対象にする　*2 Ｂ　処理対象からはずす
ｖの桁	ﾍﾞﾘﾌｧｲ･ｺﾋﾟｰﾌﾗｸﾞ　．　ファイル作成時（書き込み時）のまま　*2 Ｖ　ベリファイずみ Ｃ　コピーずみ／転送ずみ
ｄの桁	ファイル編成　．　順編成ファイルである　*2 Ｓ　〃 Ｄ　順次再配置（不良セクタを詰める）は不可
ＢＯＥ	領域開始アドレス（ＣＣＨＲＲ形式）
ＥＯＥ	領域終了アドレス（ＣＣＨＲＲ形式） （このセクタ（含む）までがファイル領域。EOE=EOD-1 になり得る）
ＥＯＤ	データ終了アドレス（ＣＣＨＲＲ形式） （データの入っている最終ブロックの、つぎのセクタのアドレス）
作　成	ファイルの作成日付（西暦ＹＹＭＭＤＤ形式）
ﾚｺｰﾄﾞ数	レコード件数
満　了	ファイルの満了日付（西暦ＹＹＭＭＤＤ形式）　*3 （この日（含む）まで更新／削除禁止。 空白＝いつでも更新／削除可。999999＝永久に更新／削除禁止）

*1）本来は規約外の形式だが、「半拡張形式」としてサポートしている
*2）Ｆ＊ＴＲＡＮ２００７では管理していない。設定状況を見ることができるだけである
*3）Ｆ＊ＴＲＡＮ２００７は満了日付をチェックしない。いつでも更新／削除可能

●ショート形式の表示とその内容

／Ｓｈｏｒｔか／Wオプションをつけてショート形式で表示すると、ボリューム情報につづけて、

```
SOLARSYSTEM     SUN         PLANET      MERCURY
VENUS           EARTH       MARS        ASTEROID
JUPITER         SATURN      URANUS      NEPTUNE
PLUTO
```

のように、ＩＢＭファイル名だけを１行に４ファイルずつ横並びの形式で表示します。

●ｉＬｉｓｔ（ｉｌ）コマンドの応用

バッチファイルにＦ＊ＴＲＡＮ２００７を組み込んでファイル変換するときは、

ディスクが正しくセットされているかどうか
空のディスクかどうか
目的のＩＢＭファイルはあるかどうか

などを、あらかじめ検査する必要性が高いものです。これらの検査にｉＬｉｓｔ（ｉｌ）コマンドを利用できます。このコマンドが、終了コードとして、指定したファイルがあるときは０、ないときは１以上を返すことを利用します。

ディスクが正しくセットされているかどうかを検査する

```
:start
    start /w ft /wc/ ilist a:.index
    if errorlevel 1 goto seterr
        ⋮
    本来の処理
        ⋮
    goto end

:seterr
    echo ドライブ　a：に、正しくＩＢＭ形式のＦＤをセットしてください。
    pause
    goto start

:end
```

．ＩＮＤＥＸというファイルはインデックスシリンダの表面（０面）を表すディスク領域名で、どの種別のＩＢＭディスクにも必ずあります。つまり、それが見つかったということは、「ディスクは正しくセットされていた」ということです。

◆参考　----　ディスク装着検査コマンド（ＩｓＲｅａｄｙ）

ディスクがセットされているかどうかの判定は、ＩｓＲｅａｄｙコマンドを使うことを推奨します。

空のディスクかどうかを検査する

```
start /w ft /wc/ ilist a:*
if not errorlevel 1 goto notblank
        ┆
```

ファイル名として＊、つまり「全ファイル」を指定すると、１つでもファイルがあれば終了コードで０（正常）を返します。ということは、逆に考えれば１以上が返ってくれば「空のディスクである」と判断してよいことになります。

一見、“if errorlevel 0 ～” と書けばよいように思いがちです。しかし、その書き方では０以上はすべて～、つまり無条件に～せよという意味になってしまいます。注意してください。

ファイルＸＸＸがあるかどうかを検査する

```
:start
        ┆
    まず、ディスクが正しくセットされているかどうかを検査する
        ┆
:convert
    start /w ft /wc/ ilist a:XXX
    if errorlevel 1 goto notfound
        ┆
    本来の変換処理
        ┆
    goto end

:notfound
    echo ＩＢＭファイルＸＸＸがありません。
    echo 正しいＦＤをセットしてください。
    pause
    goto start

:end
```

ｉＬｉｓｔ（ｉｌ）コマンドは、ワイルドカード文字を使わないふつうのファイル名を指定すると、そのファイルがあれば終了コードで０を返し、なければ１以上を返します。このことを、目的のファイルがあるかどうかの判定に利用できます。

◆参考...

業務ごとにＩＢＭディスクを使い分けるときは、各ＩＢＭディスクの先頭に、業務に則した名前の小さいキーファイルを作っておくと便利です（内容は何でもよいでしょう）。そのキーファイルがあるかどうかをｉＬｉｓｔ(ｉｌ)コマンドで検査すれば、ＩＢＭディスクの使い分けの間違いを未然に防ぐことができます。ちょっとした運用上のテクニックです。

■使用例

例１）全ファイルについて詳しく表示する

ドライブＡ：の全ＩＢＭファイルを詳しく表示します。

```
C:¥>ft ilist a: ↓
```

例２）先頭がＡＢのファイルをショート形式表示する

ドライブＡ：の、名前がＡＢではじまるすべてのＩＢＭファイルを、ＩＢＭファイル名だけの横並びの形式で表示します。

```
C:¥>ft ilist a:ab* /s ↓              (/Short)
```

２．１３　ｉＤｅｌｅｔｅ（ｉｄｅｌ）　アイ・デリート

ＩＢＭファイルの削除

ｉＤｅｌｅｔｅ（ｉｄｅｌ）は、ＩＢＭディスクから指定したＩＢＭファイルを削除するコマンドです。

操作ミスなどで必要なファイルを削除してしまっても、その直後なら簡単に復活できるようになっています。

なお、ＩＢＭ形式に初期化ずみで販売されているディスクには、ＤＡＴＡという名前の空ファイルが作ってあることがあります。このファイルはディスクのデータ部全体をファイル領域として割りつけてあるので、ｉＤｅｌｅｔｅ（ｉｄｅｌ）コマンドでそれを削除しないと、Ｆ＊ＴＲＡＮ２００７で使えるようにはなりません。

新品ディスクを利用するときは、ｉＬｉｓｔ（ｉｌ）コマンドで確認　が原則です。

■コマンド形式

コマンド	パラメータ
ｉＤｅｌｅｔｅ ｉｄｅｌ	ＩＢＭファイル　［オプション］

■パラメータの説明

●ＩＢＭファイル

ｄ：ＩＢＭファイル名

ｄ：はドライブ名

削除するＩＢＭファイルを指定します。
ドライブ名はＡ：～Ｐ：、または０：～３：で指定します。省略はできません。
ＩＢＭファイル名にはワイルドカード文字（＊）も使えます。たとえば、

```
Ｃ：￥＞ｆｔ　ｉｄｅｌｅｔｅ　ａ：＊ ↓
```

と指定すればドライブＡ:のすべてのＩＢＭファイルを削除できます。１ファイルしか入っていないときもこの方法を使うと便利です。なお、ＩＢＭファイル名の省略はできません。

●指定できるオプション

ｉＤｅｌｅｔｅ(ｉｄｅｌ)コマンドには、

／Ｑｕｅｒｙ	ファイル名を確認しながら削除する
／ＮｏＱｕｅｒｙ	ファイル名の確認なしで自動削除する

の２つのオプションがあります。

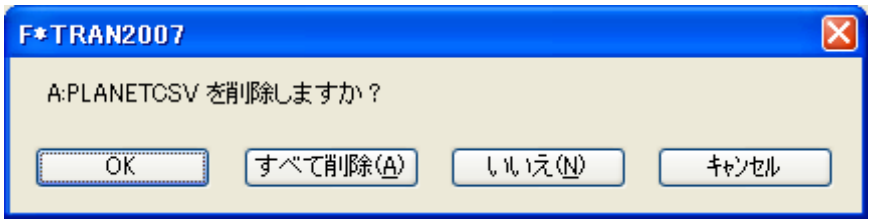
●オプションの詳細

／Ｑｕｅｒｙ -------- ファイル名を確認しながら削除する
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動削除する

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

ＩＢＭファイルを削除するとき、問い合わせするかどうかを指定します。

／Ｑｕｅｒｙオプションを指定すると、Ｆ＊ＴＲＡＮ２００７は１ファイルごとに処理を問い合わせます。



つぎのどれかで応答してください。

ＯＫ	表示中のファイルを削除する
すべて削除（Ａ）	全削除へ切り替え、以降のファイルをすべて削除する
いいえ（Ｎ）	削除処理を中断する
キャンセル	表示中のファイルは削除しない

／ＮｏＱｕｅｒｙオプションを指定すると、確認なしで指定ファイルを削除します。こちらが省略値です。

●オプション省略時の動作

オプションを省略すると、

／ＮｏＱｕｅｒｙ　　指定ファイルを無条件に削除する

と指定したとみなします。

■解説

●削除とは

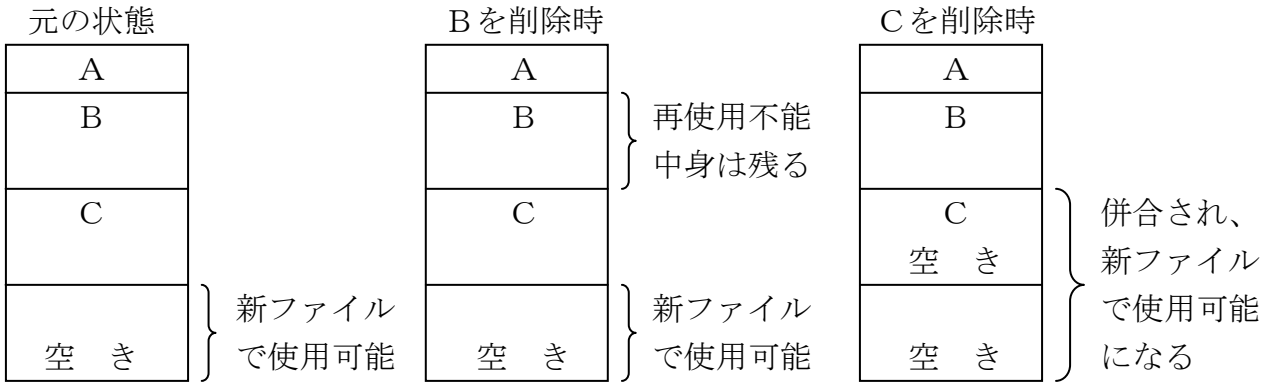
ｉＤｅｌｅｔｅ（ｉｄｅｌ）コマンドによるファイルの「削除」とは、インデックスシリンダにあるファイルラベルに「削除した」という印をつけることです。具体的には、

ファイルラベルの先頭１文字をHからDに書き替えている

だけです。ファイルラベルの残りの部分は元のままです。データ領域の中身もそっくりそのまま残しておきます。

ファイルを削除すると、そのファイルラベルは「空きファイルラベル」になり、使っていたデータ領域は「再使用不能」になります。ただし、そこがＩＢＭディスク上の最後のファイルならば、そのうしろの空き領域と併合されます。

今、Ａ、Ｂ、Ｃの順にデータ領域を使用している３つのファイルがあり、そのうしろは空き領域だとします（ファイルラベルもＡ、Ｂ、Ｃの順に並んでいるとは限りません）。最初のＡか途中のＢを削除したら、そこは再使用されません。最後のＣを削除したときは、そのうしろの元の空き領域と併合されて、新たにより広い空き領域になります。



●削除の直後なら簡単に復活できる

うっかり必要なファイルを削除してしまってもあわてないでください。削除の直後なら簡単に復活できます。しかし、新しいファイルを作ってしまったら復活は不可能、もしくはかなり困難な作業になります。注意してください。

ファイルの復活にはＩＢＭディスクエディタ（ＧＵＩ部）でインデックスシリンダ（ラベル部）を表示・修正する機能を使います。インデックスシリンダのなかから削除したファイルのファイルラベルを見つけ、その先頭１文字をDからHに書き替えてディスクに書き戻します。これだけでファイルを復活できます。なお、復活専用のコマンドはありません。

■使用例

例１）新品ディスクの空ファイルＤＡＴＡを削除する

新品のディスクによくある空ファイルＤＡＴＡを削除し、Ｆ＊ＴＲＡＮ２００７で使えるようにします。

```
C:¥>ft idelete a:data ↓
```

例２）ＴＥＭＰではじまる一時ファイルをすべて削除する

ドライブＡ：のＴＥＭＰで始まる名前のＩＢＭファイルをすべて削除します。

```
C:¥>ft idelete a:temp* ↓
```

例３）全ファイルを削除する

ドライブＡ：の全ＩＢＭファイルを削除します。

```
C:¥>ft idelete a:* ↓
```

例４）ファイル名を確かめながら削除する

ファイル名を確認しながら問い合わせモードで削除します。

```
C:¥>ft idelete a:* /q ↓              (/Query)
```

■注意事項

書き込み禁止は処理を中断

書き込み禁止のファイルがあるとそこで処理を中断し、エラー扱いにします。

満了日付は無視する

満了日付が来ていないファイルに対しても、削除を実行してしまいます。満了日付の管理はしていません。

２．14　ｉＦｏｒｍａｔ（ｉｆｏ）　アイ・フォーマット

ＩＢＭディスクの初期化

ｉＦｏｒｍａｔ(ｉｆｏ)は、フロッピーディスクを初期化（イニシャライズ／フォーマット）し、ＩＢＭ形式のディスクとして使えるようにするコマンドです。

これは危険なコマンドです。ｉＦｏｒｍａｔ(ｉｆｏ)コマンドでディスクを初期化すると、元のデータは完全に失われます。それを回復する方法はありません。使用にあたっては細心の注意が必要です。運用上可能であれば新品ディスクを使い回すようにし、このコマンドの使用を避けるべきです。

■コマンド形式

コマンド	パラメータ
ｉＦｏｒｍａｔ ｉｆｏ	ドライブ名　［オプション］

■パラメータの説明

●ドライブ名

ｄ：

ｄ：はドライブ名

初期化するディスクを挿入するドライブを、Ａ：～Ｐ：、または０：～３：で指定します。うっかり別のディスクを初期化してしまわないように、あらかじめｉＬｉｓｔ(ｉｌ)コマンドを使って、

目的のドライブのアクセスランプが点灯する

ことを確かめておいてください。

●指定できるオプション

ｉＦｏｒｍａｔ(ｉｆｏ)コマンドには、

／Ｔｙｐｅ	一般のＩＢＭ形式、東芝形式、三菱形式を指定する
／Ｍｅｄｉａ	媒体種別を指定する
／ＶｏｌｕｍｅＩＤ	ボリュームＩＤを指定する
／ＮｏＶｏｌｕｍｅＩＤ	ボリュームＩＤをつけない
／Ｏｗｎｅｒ	所有者名を指定する
／ＮＯＯｗｎｅｒ	所有者名をつけない
／ＳＵｒｅｌｙ	初期化実行直前に確認をとる／とらない
／ＶＥＲＩｆｙ	ベリファイを行う
／ＮｏＶＥＲＩｆｙ	ベリファイを行わない
／Ｑｕｉｃｋ	初期化をクイックフォーマットとして行う
／ＮｏＱｕｉｃｋ	初期化の際、物理フォーマット部分も行う

という多くのオプションがあります。これらのオプションでディスクの初期化に必要なパラメータを与えます。どのオプションでも、

初期化設定ウインドウを開き、パラメータを指定する
初期化設定ウインドウを開かず、オプションにパラメータを直接指定する

のどちらにするか、利用者が自由に指定できます。初期化設定ウインドウを開かず、初期化の実行直前の初期化確認ウインドウすら省いて実行させることもできます。

●オプションの詳細

／Ｔｙｐｅ ---- 一般のＩＢＭ形式、東芝形式、三菱形式を指定する

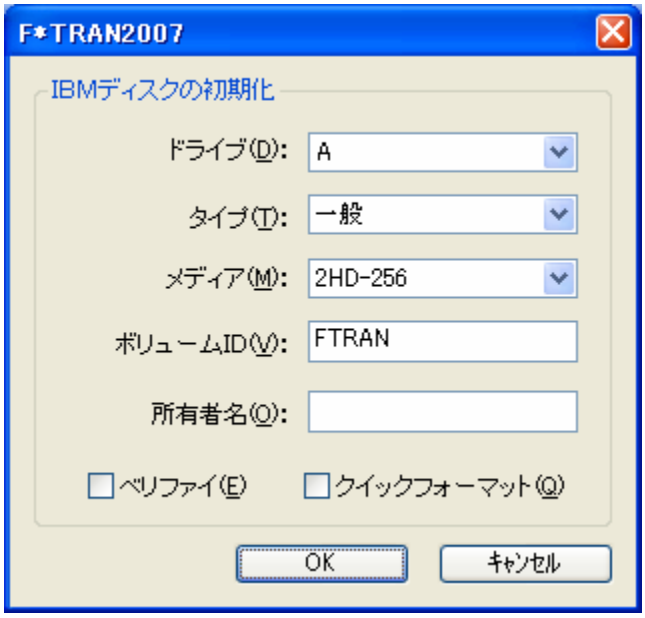
```
／Ｔｙｐｅ　［？｜Ｇｅｎｅｒａｌ｜Ｔｏｓｈｉｂａ｜Ｍｉｔｓｕｂｉｓｈｉ］
／Ｔｙｐｅ　Ｇｅｎｅｒａｌ
```

一般のＩＢＭ形式、東芝形式、三菱形式のどれにするかを指定します。つぎに示す、

？、または省略	どの形式にするか初期化設定ウインドウを開く
Ｇｅｎｅｒａｌ	一般のＩＢＭ形式にする（1.2MB ﾌｫｰﾏｯﾄ、EBCDIC ﾗﾍﾞﾙ）
Ｔｏｓｈｉｂａ	東芝形式にする（1.2MB ﾌｫｰﾏｯﾄ、JIS8/ASCII ﾗﾍﾞﾙ）
Ｍｉｔｓｕｂｉｓｈｉ	三菱形式にする（1.44MB ﾌｫｰﾏｯﾄ、EBCDIC ﾗﾍﾞﾙ）

の３とおりの指定ができます。

／Ｔｙｐｅオプションに何もつけないか、？指定したときは、初期化設定ウインドウが開きます。



「一般」は一般のＩＢＭ形式にする指定です。
「東芝」は東芝形式にする指定です。
「三菱」は三菱形式にする指定です。
一般、東芝、三菱のどれかを選択してください。

＜一般のＩＢＭ形式、東芝形式、三菱形式＞

一般に、ＩＢＭ形式といえばラベル類をＥＢＣＤＩＣコードで記録することになっています。しかし、東芝のシステムでは、ラベル類をＪＩＳ８／ＡＳＣＩＩコードで記録するのが標準です（実際はシステムによって少しずつ事情が違います)。また、三菱のシステムの３．５インチディスクは、ラベル類はＥＢＣＤＩＣコードで記録しますが、１．４４ＭＢフォーマットになっています（通常は、１．２ＭＢフォーマット)。

Ｆ＊ＴＲＡＮ２００７は３タイプとも扱うことができるので、区別する必要がないときはまとめて「ＩＢＭ形式」と呼びます。区別する必要があるときは、それぞれ「一般のＩＢＭ形式」、「東芝形式」、「三菱形式」と呼び分けます。

／Ｍｅｄｉａ ---- 媒体種別を指定する

```
                ┌ ？
                │ １Ｓ［－１２８｜－２５６｜－５１２］
／Ｍｅｄｉａ     │ ２Ｓ［－１２８｜－２５６｜－５１２］
                └ ２ＨＤ［－２５６｜－５１２｜－１０２４］
／Ｍｅｄｉａ　？
```

ｉＦｏｒｍａｔ（ｉｆｏ）コマンドがサポートしている９種類の物理フォーマットのうち、どれにするかを指定します。つぎに示す、

？、または省略			初期化設定ウインドウを開く
２ＨＤ－ｎｎｎ	ｏｒ	２ＨＤ	５ｲﾝﾁ／３．５ｲﾝﾁ　２ＨＤ－ｎｎｎにする
１Ｓ－ｎｎｎ	ｏｒ	１Ｓ	８インチ互換１Ｓ－ｎｎｎにする
２Ｓ－ｎｎｎ	ｏｒ	２Ｓ	８インチ互換２Ｓ－ｎｎｎにする

のどれかで指定します。

／Ｍｅｄｉａオプションに何もつけないか、？指定したときは、初期化設定ウインドウが開きます。媒体種別を選択してください。

なお、当然のことながら、

媒体種別の指定はフロッピーディスクのサイズ・面数・密度に合った指定に

しなければいけません。絶対です。これを間違えた場合、結果は保証できません。

＜媒体種別＞

ｉＦｏｒｍａｔ(ｉｆｏ)コマンドで初期化可能なディスク・フォーマットは、

８インチ互換１Ｓ－１２８／２５６／５１２
８インチ互換２Ｓ－１２８／２５６／５１２
５インチ２ＨＤ－２５６／５１２／１０２４
３．５インチ２ＨＤ－２５６／５１２／１０２４　} ２つまとめて扱う

の９種類です。全部で９種類あるフォーマットをすべてサポートしています。なお“－”のあとの数字は、セクタ長を意味しています。

◆参考　----　８インチ互換モードについて

８インチディスク互換モードとは３．５インチおよび５インチのディスクを８インチの１Ｓや２Ｓと同じ形式でアクセスする機能です。初期化時に８インチディスク互換モード形式で初期化を行った場合、変換処理のアクセスもこのモードで行われます。

／ＶｏｌｕｍｅＩＤ -------- ボリュームＩＤを指定する
／ＮｏＶｏｌｕｍｅＩＤ ---- ボリュームＩＤをつけない

```
／ＶｏｌｕｍｅＩＤ　［？｜－｜ボリュームＩＤ］
／ＶｏｌｕｍｅＩＤ　ＦＴＲＡＮ
／ＮｏＶｏｌｕｍｅＩＤ
```

ボリュームＩＤ（ボリューム通番）を指定します。

／ＶｏｌｕｍｅＩＤオプションには、つぎのように、

？、または省略	初期化設定ウインドウを開く
－	空白＝ボリュームＩＤなし
ボリュームＩＤの文字列	ボリュームＩＤを直接指定する

の２とおりの指定ができます。

／ＶｏｌｕｍｅＩＤオプションに何もつけないか、？指定したときは、初期化設定ウインドウが開きます。ボリュームＩＤを入力してください。ボリュームＩＤは省略可能です。

／ＮｏＶｏｌｕｍｅＩＤオプションを指定すると、はじめからボリュームＩＤなし（空白）になります。

＜ボリュームＩＤ＞
ボリュームＩＤとは、ＩＢＭディスクにつける識別名です。というよりは、識別コードのようなものと思ってください。つぎの規則に従ってつけます。

英大文字（Ａ～Ｚ）と数字（０～９）を組み合わせ、６文字以内で指定する
英小文字で指定してもよいが、英大文字に変換される
数字ではじまってもよい（たとえば“００７”のような指定も許される）
途中に空白を置いてはいけない
特殊文字、半角カタカナ、漢字などは使えない
ただし、特殊文字のうち　@、＃、￥、＄　は使えるシステムが多い

ボリュームＩＤは適当につけてもよいし、つけなくてもよいことが多いのですが、相手システムの事情にもよります。データ交換の当事者間の合意による、指定された特定のボリュームＩＤをつけなければならないこともあります。

／Ｏｗｎｅｒ -------- 所有者名を指定する
／ＮＯＯｗｎｅｒ ---- 所有者名をつけない

／Ｏｗｎｅｒ　［？｜－｜所有者名］ ／ＮＯＯｗｎｅｒ

所有者名を指定します。

／Ｏｗｎｅｒオプションには、つぎのように、

？、または省略	初期化設定ウインドウを開く
－	空白＝所有者名なし
所有者名の文字列	所有者名を直接指定する

の２とおりの指定ができます。

／Ｏｗｎｅｒオプションに何もつけないか、？指定したときは、初期化設定ウインドウが開きます。所有者名を入力してください。所有者名は省略可能です。

／ＮＯＯｗｎｅｒオプションを指定すると、はじめから所有者名なし（空白）になります。

＜所有者名＞
所有者名とはＩＢＭディスクに補助的に「だれのものか」を書き込んでおくもの、と思ってください。つけるときは、ふつう担当者の名前や会社名にします。
所有者名は制御には使用されません。ですから、省略してもまったく悪影響はありません。つけるならつぎの規則に従ってください。

英大文字（Ａ～Ｚ）と数字（０～９）で構成し、途中に空白を入れてもよい
英小文字で指定しても英大文字に変換される
１４文字以内で指定し、先頭は英字に限る
特殊文字、半角カタカナ、漢字などは使えない
ただし、特殊文字のうち　＠、＃、￥、＄　は使えるシステムが多い

ただし、／Ｏｗｎｅｒオプションに所有者名を直接指定し、途中に空白を入れたいときは、空白の代わりに、たとえば /Owner Fujitsu_BSC のように、下線（_）を使ってください。初期化設定ウインドウ上では、空白は空白そのもので指定してかまいません。

／ＳＵｒｅｌｙ ---- 初期化実行直前に確認をとる／とらない

／ＳＵｒｅｌｙ　［？｜Ｙｅｓ｜Ｎｏ］

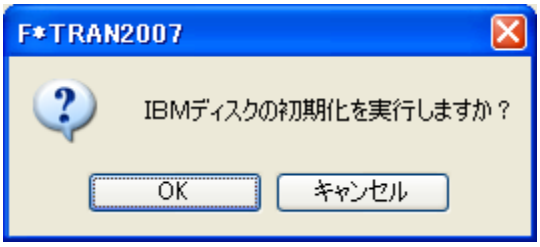
ディスクの初期化の直前に初期化確認ウインドウを開くか、初期化確認ウインドウを開かずにいきなり実行させるかを指定します。

／ＳＵｒｅｌｙオプションには、つぎのように、

？、または省略	初期化確認ウインドウを開く
Ｙｅｓ	確認をとらずに実行する
Ｎｏ	初期化パラメータの確認だけする。実行はしない

の３とおりの指定ができます。

／ＳＵｒｅｌｙオプションに何もつけないか、？指定したときは、初期化確認ウインドウが開きます。



ＯＫ	初期化を実行する
キャンセル	初期化を中止する

のどちらかで答えてください。

／ＳＵｒｅｌｙオプションは、ほかのオプションと際立った違いが１つあります。それは、／ＳＵｒｅｌｙ　Ｙｅｓ と指定したときでも、初期化設定ウインドウが開いたときには、／ＳＵｒｅｌｙ　？ の指定に切り替わり、初期化確認ウインドウが開くことです。もし、初期化設定ウインドウに答えたとたんに初期化がはじまってしまうとしたら、とても使えたものではありません。それが強制的に？指定に切り替える理由です。

／ＶＥＲＩｆｙ　　ベリファイを行う
／ＮｏＶＥＲＩｆｙ　　ベリファイを行わない

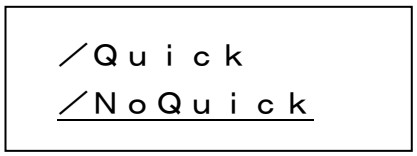
このオプションをつけることでIBMディスクのフォーマットの際にベリファイを行うかどうかを指定できるようになります。

／ＶＥＲＩｆｙオプションを指定すると、フォーマット後、ディスクの全領域が正常にフォーマットされたかを検証します。このオプションは／Ｑｕｉｃｋオプションと同時に指定することはできません。

／ＮｏＶＥＲＩｆｙオプションは、ベリファイ処理を行いません。

どちらも指定しない場合は、／ＮｏＶＥＲＩｆｙが指定されたものとして動作します。

／Ｑｕｉｃｋ　　初期化をクイックフォーマットとして行う
／ＮｏＱｕｉｃｋ　　初期化の際、物理フォーマット部分も行う



このオプションをつけることでIBMディスクのフォーマットをクイックフォーマットを行うかどうかを指定できるようになります。

／Ｑｕｉｃｋオプションを指定すると、物理フォーマットを行わずに論理フォーマットのみを行います。このオプションは／ＶＥＲＩｆｙオプションと同時に指定することはできません。

／ＮｏＱｕｉｃｋオプションを指定すると、物理フォーマットも含めたフォーマットを行います。

どちらも指定しない場合は、／ＮｏＱｉｃｋオプションが指定されたものとして動作します。

●オプション省略時の動作

オプションをすべて省略すると、

／Ｔｙｐｅ　Ｇｅｎｅｒａｌ	一般のＩＢＭ形式にする
／Ｍｅｄｉａ　？	初期化設定ウインドウを開く
／ＶｏｌｕｍｅＩＤ　ＦＴＲＡＮ	ボリュームＩＤ“ＦＴＲＡＮ”をつける
／ＮＯＯｗｎｅｒ	所有者名はつけない
／ＳＵｒｅｌｙ　？	初期化確認ウインドウを開く
／ＮｏＶＥＲＩｆｙ	ベリファイを行わない
／ＮｏＱｕｉｃｋ	初期化の際、物理フォーマット部分も行う

と指定したものとみなします。

■解説

●初期化とは　――　物理フォーマットと論理フォーマット

ディスクの初期化は、つぎの２つのステップに分けることができます。

物理フォーマット　　各トラック上のセクタを仕切り直すこと
論理フォーマット　　インデックスシリンダに初期データ（ラベル）を書き込み、
ＩＢＭディスクとして使えるようにすること

物理フォーマットを行うと、元のデータは完全に失われます。回復する方法はありません。

●フォーマット済み新品ディスク利用のすすめ

「標準的なＩＢＭ形式のディスクが必要だ」というだけなら、新品ディスクの利用をおすすめします。

しかしながら、店頭で入手できる３．５インチ２ＨＤのディスクにおいても、ＩＢＭ形式でフォーマット済みのものは最近では希少になっています。ましてや、５インチのディスクについては店頭での入手はかなわないでしょう。使いたいものが店頭にないときは、メーカーのサプライ品を利用するのも１つの方法です。

標準的なＩＢＭ形式に初期化ずみのものと、それ以外のものの見分け方を説明します。標準的なＩＢＭ形式に初期化ずみのものは、ケースやラベルに

５インチ２ＨＤ－２５６　　なら　　“ＭＤ／２ＨＤ　２５６”
３．５インチ２ＨＤ－２５６　　なら　　“ＭＦ／２ＨＤ　２５６”

などと書いてあるので判別できます。下線を引いた数字のところがポイントです。この数字は、

データ部のセクタ長がｎｎｎの、一般のＩＢＭ形式に初期化ずみ

という意味です。

■使用例

例１）ＩＢＭ形式の初期化を行う

初期化設定ウインドウ／初期化確認ウインドウを開いて、ドライブＡ：の３．５インチ２ＨＤディスクをＩＢＭ形式に初期化します。

```
C:¥>ft iformat a: ↓
```

例２）ボリュームＩＤも所有者名もつけない

ドライブＡ：の３．５インチ２ＨＤディスクを一般のＩＢＭ形式に初期化します。ボリュームＩＤも所有者名もつけません。

```
C:¥>ft iformat a: /nvid ↓          (/NoVolumeID)
```

例３）すべてのパラメータをはじめからオプションで指定する

パラメータをすべてオプションで直接、指定します。

```
C:¥>ft iformat a:/tt /m2hd-1024 /vid 000000 /o nanashi_gonbei ↓
      (/Type Toshiba /Media 2hd-1024 /VolumeID 000000 /Owner nanashi_gonbei)
```

例４）バッチファイル化し、うまく初期化できたかを確認する

初期化を自動化し、さらに不良ディスクの検査までするバッチファイルは、つぎのように書けます。

＜ＩＦＯＶ．ＢＡＴ＞

```
@echo off ↓
start /w ft /wc/ iformat a: /m 2hd-256 /vid ↓
if errorlevel 1 goto ifoerr1 ↓
start /w ft /wc/ ilist a:.index ↓
if errorlevel 1 goto ifoerr2 ↓
echo インデックスシリンダはＯＫ．　↓
echo データシリンダの検査に１～２分かかります。　↓
start /w ft /wc/ getrand a:.data nul /map binary ↓
if errorlevel 1 goto ifoerr3 ↓
echo データシリンダもＯＫ．Ｇｏ！　↓
     ┆
ファイル変換
     ┆
goto end ↓

:ifoerr1 ↓
 echo ドライブ A: のＩＢＭディスクの初期化に失敗しました．　↓
 echo ディスクのセットからやり直してください．　↓
 goto end ↓

:ifoerr2 ↓
 echo インデックスシリンダが不良です．ディスクは本当に２ＨＤですか？　↓
 echo このディスクは廃棄し、別のディスクでやり直してください．　↓
 goto end ↓

:ifoerr3 ↓
 echo データシリンダに不良箇所があります．　↓
 echo このディスクは廃棄し、別のディスクでやり直してください．　↓

:end ↓
```

■注意事項

媒体種別（ディスクの種別）に注意

媒体種別の指定には十分に注意してください。誤った指定をすると、

規約はずれの形式になる
異常終了する
ハングアップする

のどれかになります。

空ファイルＤＡＴＡは作らない

一部の規約では、初期化したディスクにはＤＡＴＡという名前の、データ領域全体を割りつけた空ファイルを作ることになっています。しかし、ｉＦｏｒｍａｔ（ｉｆｏ）コマンドはこのファイルを作りません。その理由は、このファイルがあると通常の使用方法では使用前にこのファイルＤＡＴＡを削除するという余計な処理が必要になるからです。

不良ドライブは使わないで

入出力エラーを起こしやすい、動作不良または調整の狂ったドライブは、初期化には絶対に使わないでください。不良ドライブで初期化をすると、初期化自体は大体正常終了しますが、使いはじめてから入出力エラーが多発します。

不良ディスクは使わないで

指紋やキズなどによる不良箇所のあることがわかっているディスクは使わないでください。不良箇所があっても、初期化自体は正常終了することが多く、使いはじめてから入出力エラーに悩まされます。なお、フロッピーディスク不良について初期化時に／VREIｆｙオプションを併せて指定することでチェック可能です。

２．１５　ｉＤｉｓｋＣｏｐｙ（ｉｄｃ）　アイ・ディスク・コピー

ＩＢＭディスクの複写

ｉＤｉｓｋＣｏｐｙ（ｉｄｃ）は、ＩＢＭディスクをまるごと複写（コピー）するコマンドです。フロッピードライブが１基であっても、フロッピーディスクを差し替えて複写できるようになっています。

ディスクコピーには、最小コピーモードと完全コピーモードがあります。最小コピーモードは、ＩＢＭファイルの有効セクタ（ＢＯＥ～ＥＯＤ－１）を基にコピーするため、短時間でディスクを複写できます。完全コピーモードは、データシリンダ全体をまるごとコピーします。

また、コピー先フロッピーディスクの初期化指定をして、コピーすることもできます。

■コマンド形式

コマンド	パラメータ
ｉＤｉｓｋＣｏｐｙ ｉｄｃ	コピー元ドライブ名　コピー先ドライブ名　［オプション］

■パラメータの説明

●コピー元ドライブ名、コピー先ドライブ名

ｄ：　ｓ：

ｄ：，ｓ：はドライブ名

複写の対象となるディスクを挿入するコピー元ドライブ、コピー先ドライブを、Ａ：～Ｐ：、または０：～３：で指定します。省略はできません。

たとえば、

Ｃ：￥＞ｆｔ　ｉｄｉｓｋｃｏｐｙ　ａ：　ａ：↓

のように指定します。

●指定できるオプション

ｉＤｉｓｋＣｏｐｙ(ｉｄｃ)コマンドには、

／Ｐａｕｓｅ	対話モードでディスクコピーを行う
／ＮｏＰａｕｓｅ	即時モードでディスクコピーを行う
／Ｆｏｒｍａｔ	コピー時にコピー先ディスクを初期化する
／ＮｏＦｏｒｍａｔ	コピー先ディスクを初期化済みとして扱う
／Ｔｙｐｅ	初期化タイプ（1.2MB､1.44MB）を指定する
／ＭＩＮ	最小コピーモードでディスクコピーを行う
／ＡＬＬ	完全コピーモードでディスクコピーを行う

というオプションがあります。

●オプションの詳細

／Ｐａｕｓｅ -------- 対話モードでディスクコピーを行う
／ＮｏＰａｕｓｅ ---- 即時モードでディスクコピーを行う

／Ｐａｕｓｅ ／ＮｏＰａｕｓｅ

対話モードで動作させるか、即時モードで動作させるかを指定します。

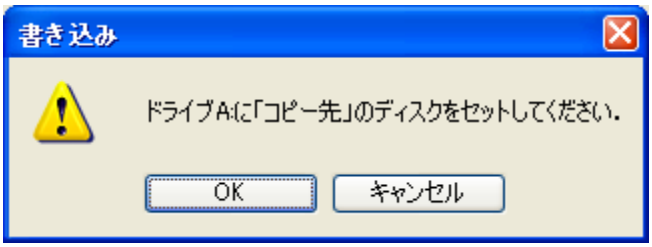
／Ｐａｕｓｅオプションを指定して実行すると、つぎのようになります。

①コピー元のディスク挿入の確認ウインドウが開きます。

ディスクセット後の“ＯＫ”ボタンＯＮで、読み込み処理開始
“キャンセル”ボタンＯＮで、ディスクコピー処理中止

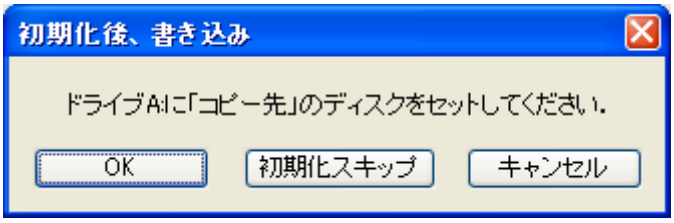
②コピー元のディスクの読み込みが終了すると、コピー先のディスク挿入の確認ウインドウが開きます。後述のフォーマット指定によって、ウインドウの形態が異なります。

＜フォーマット指定なしの場合＞



ディスクセット後の“ＯＫ”ボタンＯＮで、書き込み処理開始
“キャンセル”ボタンＯＮで、ディスクコピー処理中止

＜フォーマット指定ありの場合＞



ディスクセット後の“ＯＫ”ボタンＯＮで、初期化処理・書き込み処理開始
“初期化スキップ”ボタンＯＮで、書き込み処理開始
“キャンセル”ボタンＯＮで、ディスクコピー処理中止

この確認ウインドウの指示に従ってディスクコピーを行うモードが、対話モードです。これが、省略値です。

／ＮｏＰａｕｓｅオプションを指定すると、確認ウインドウなしでディスクコピーをします。ただし、／ＮｏＰａｕｓｅオプションを指定しても、コピー元ドライブとコピー先ドライブが同一である場合は、自動的に対話モードになります。

つまり、最低２ドライブを搭載しているパソコンにコピー元ディスクとコピー先ディスクを同時にセットし、確認なしでディスクコピーを行う場合に使用するのが即時モードです。

／Ｆｏｒｍａｔ -------- コピー時にコピー先ディスクを初期化する
／ＮｏＦｏｒｍａｔ ---- コピー先ディスクを初期化済みとして扱う

```
／Ｆｏｒｍａｔ　［？｜Ｙｅｓ］
／ＮｏＦｏｒｍａｔ
```

コピー先ディスクへの書き込み時に、コピー先ディスクを初期化するかどうかを指定します。

／Ｆｏｒｍａｔオプションで、？指定すると、対話モードのところで説明した初期化・書き込み確認ウインドウが“即時モード”であっても無条件に開き、初期化・書き込み処理を行います。／Ｆｏｒｍａｔオプションの省略値はこちらです。

／Ｆｏｒｍａｔオプションで、Ｙｅｓ指定すると、初期化・書き込み処理を行います。確認ウインドウは、対話モードと即時モードの指定に依存します。

／ＮｏＦｏｒｍａｔオプションを指定すると、コピー先を初期化済みディスクとして扱い、書き込み処理のみを行います。こちらが、省略値です。

／Ｔｙｐｅ ---- 初期化タイプ（1.2MB、1.44MB）を指定する

```
／Ｔｙｐｅ　［Ｉｎｐｕｔ｜１．２ｍｂ｜１．４４ｍｂ］
／Ｔｙｐｅ　　Ｉｎｐｕｔ
```

初期化処理のときのフォーマットタイプを指定します。つぎに示す、

Ｉｎｐｕｔ	コピー元のディスクと同一のフォーマットにする
１．２ｍｂ	１．２ＭＢフォーマット（一般形式、東芝形式）にする
１．４４ｍｂ	１．４４ＭＢフォーマット（三菱形式）にする

の３とおりの指定ができます。省略値は、Ｉｎｐｕｔです。

／ＭＩＮ ---- 最小コピーモードでディスクコピーを行う
／ＡＬＬ ---- 完全コピーモードでディスクコピーを行う

／ＭＩＮ ／ＡＬＬ

最小コピーモードか、完全コピーモードかを指定します。

／ＭＩＮオプションを指定すると、最小コピーモードになり、ＩＢＭファイルの有効セクタ（ＢＯＥ～ＥＯＤ－１）のみをコピーします。少量のデータしかないときには、コピー時間を大幅に節約することができます。こちらが、省略値です。

／ＡＬＬオプションを指定すると、完全コピーモードになり、データ部の全セクタがコピーされます。

どちらのモードでも、インデックスシリンダの内容は完全にコピーされます。

●オプション省略時の動作

オプションを省略すると、

／Ｐａｕｓｅ	対話モード
／ＮｏＦｏｒｍａｔ	コピー先ディスクは初期化済みとして扱う
／ＭＩＮ	最小コピーモード

と指定したものとみなして、ＩＢＭディスクコピーを行います。

■使用例

例１）新品のフロッピーディスクを使って、バックアップディスクを作成する

新品のフロッピーディスクを一枚用意し、通常のバックアップ作業を行います。

```
C:¥>ft idiskcopy a: a: /format ↓
```

例２）２ドライブ構成のパソコンで、即時モードでバックアップディスクを作成する

２ドライブのフロッピーディスク装置があるパソコンで、確認なしでフルバックアップ作業を行います。

```
C:¥>ft idiskcopy a: b: /nopause /format yes /all ↓
```

例３）ディスク形式を一般から三菱に変更して、バックアップディスクを作成する

一般形式ディスクを三菱形式ディスクに変更して、バックアップ作業を行います。

```
C:¥>ft idiskcopy a: a: /format /type 1.44mb ↓
```

■注意事項

コピー元ディスクを書き込み禁止状態にしてから、ディスクコピーを実行する

ＩＢＭディスクコピーを行う前に、コピー元のディスクを書き込み禁止状態にすることをお勧めします。特に、即時モードで実行した場合のオペレーションミスを防止することができます。

欠陥があるＩＢＭディスクはディスクコピーできない

コピー元のＩＢＭディスク読み込み時に、欠陥（欠陥シリンダ、欠陥セクタ）が検出されたら、ディスクコピー処理を中断します。

削除セクタは削除セクタとしてコピーする

コピー元のＩＢＭディスク読み込み時に、削除セクタが検出されたら、コピー先のＩＢＭディスクに、削除セクタとして書き込みを行います。

２．16　ｉＲｅｎａｍｅ（ｉｒｅｎ）　アイ・リネーム

ＩＢＭディスク内ファイルの改名

ｉＲｅｎａｍｅ(ｉｒｅｎ)は、ＩＢＭディスク内ファイルを改名する機能です。

■コマンド形式

コマンド	パラメータ
ｉＲｅｎａｍｅ	ドライブ名：ＩＢＭファイル名 新しいファイル名 ［オプション］

■パラメータの説明

●ドライブ名　ファイル名　新しいファイル名

Ａ：ＩＢＭファイル名　新しいファイル名

Ａ：はドライブ名

改名したいファイルの存在するディスクを挿入するドライブ名＋コロンの後に改名したいＩＢＭファイルの現在の名前と付け替えたい名前を指定します。省略はできません。

たとえば、

Ｃ：￥＞ｆｔ　ｉＲｅｎａｍｅ　Ａ：ｐｌａｎｅｔ　ｆｉｘｅｄ↓

のように指定します。

●指定できるオプション

ｉＲｅｎａｍｅ(ｉｒｅｎ)コマンドには、

／Ｑｕｅｒｙ	**改名して良いかの確認ダイアログを出す。**
／ＮｏＱｕｅｒｙ	**改名して良いかの確認ダイアログを出さない。**

というオプションがあります。

●オプションの詳細

／Ｑｕｅｒｙ　　改名して良いかの確認ダイアログを出す。
／ＮｏＱｕｅｒｙ　　改名して良いかの確認ダイアログを出さない。

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

改名して良いかの確認ダイアログを出すかを指定します。

／Ｑｕｅｒｙオプションを指定して実行すると、確認ダイアログが開きます。

“ＯＫ”ボタン押下で、改名処理が実行されます。
“キャンセル”ボタン押下で、改名処理の実行を中止します。

／ＮｏＱｕｅｒｙオプションを指定して実行すると、確認なしで改名処理を行います。こちらが省略値です。

●オプション省略時の動作

オプションを省略すると、

／ＮｏＱｕｅｒｙ　　指定ファイルを無条件に改名する

と指定したものとみなして、改名処理を行います。

２．17　ｉＡｌｌｏｃ（ｉａｌ）　アイ・アロック

ＩＢＭディスク内領域のアロケート

ｉＡｌｌｏｃ(ｉａｌ)は、ＩＢＭファイルの領域確保と形式設定をする機能です。

■コマンド形式

コマンド	パラメータ
ｉＡｌｌｏｃ ｉａｌ	ＩＢＭドライブ名：ＩＢＭファイル名　［オプション］

■パラメータの説明

●アロケート対象ドライブ名、アロケートするファイル名

Ａ：ＩＢＭファイル名

Ａ：はドライブ名

アロケートの対象となるディスクを挿入するドライブを、Ａ：～Ｐ：、または０：～３：で指定し、その領域のファイル名を指定します。省略はできません。

●指定できるオプション

ｉＡｌｌｏｃ(ｉａｌ)コマンドには、

／Ｆｏｒｍ	ＩＢＭファイルの形式を指定する
／ＳＰａｃｅ	レコード件数による容量指定
／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅ	ボリューム順番範囲指定
／ＦＩＬＬ	初期データ指定
／ＮｏＦＩＬＬ	初期データなし
／ＲＥＰｌａｃｅ	同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ	同名ＩＢＭファイルは置き替えない
／ＢｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＢＯＥ境界を指定する
／ＥｏｅＢｏｕｎｄａｒｙ	ＩＢＭファイルのＥＯＥ境界を指定する

というオプションがあります。

●オプションの詳細

／ＳＰａｃｅ レコード件数
／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅ ボリューム順序番号範囲 ［Ｃｏｎｔｉｎｕｅ ｜ Ｌａｓｔ］

／ＳＰａｃｅ　　レコード件数	
／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅ　開始番号　［Ｔｏ　終了番号］	Ｌａｓｔ Ｃｏｎｔｉｎｕｅ
／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅ　０１　Ｌａｓｔ	

確保する領域の大きさ指定を２通りの指定方法から選択します。総レコード件数による指定（／ＳＰａｃｅオプション）、もしくはディスク単位の指定（／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅオプション）ができます。

／ＳＰａｃｅオプションでは、ＩＢＭファイル全体の容量を総レコード件数で指定します。１～２，１４７，４８３，６４８の整数の範囲で指定できます。先頭（01/C）から順番にアロケートしてゆきます。

／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅオプションでは、ボリューム順序番号の範囲を、

ｎｎ	ｎｎ番のみ
Ｔｏ　ｎｎ	０１番からｎｎ番まで
ｍｍ　Ｔｏ　ｎｎ	ｍｍ番からｎｎ番まで

（ただし、１≦ｍｍ≦９９、１≦ｎｎ≦９９、ｍｍ＜ｎｎ）

という形式で指定し、ディスク単位でアロケートができます。

ｎｎ番のディスクがまだ途中のボリュームのときはＣｏｎｔｉｎｕｅを指定します。ｎｎ番のディスクが最後の１枚のときはＬａｓｔを指定します。なお、単一ボリュームのときもＬａｓｔ指定でかまいません。

省略値は、

／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅ　０１　Ｌａｓｔ

となります。

／ＦＩＬＬ　　１バイトコード（１６進数）　｜　２バイトコード（１６進数）
／ＮｏＦＩＬＬ

／ＦＩＬＬ	１６進数２個 １６進数４個
／ＮｏＦＩＬＬ	

／ＦＩＬＬオプションを指定すると、アロケートの際に初期データとして埋め込むコードを１６進で指定できます。ただし／ＦＩＬＬオプションを使うと、データ部への書き込みが発生するため、アロケート容量あたり、Ｐｕｔ変換時にディスクに書き込んでいる時と同じくらいの時間がかかります。

たとえば以下のように指定します。

／ＦＩＬＬ　ｆｆ　　アロケート領域全てに”１”のビットで埋め尽くします。
／ＦＩＬＬ　００　　アロケート領域全てを”０”のビットで埋め尽くします。

／ＮｏＦＩＬＬオプションを指定すればデータ部への書き込みが発生しないため、非常に高速に処理できます。こちらが省略値です。

／ＲＥＰｌａｃｅ -------- 同名ＩＢＭファイルを置き替える
／ＮｏＲＥＰｌａｃｅ ---- 同名ＩＢＭファイルは置き替えない

／ＲＥＰｌａｃｅ ／ＮｏＲＥＰｌａｃｅ

アロケートするＩＢＭファイルと同名のＩＢＭファイルがすでに存在する場合の処置を指定します。

／ＲＥＰｌａｃｅオプションを指定すると、既存の同じ名前のＩＢＭファイルをいったん削除してからアロケート処理に入ります。

／ＮｏＲＥＰｌａｃｅオプションを指定すると、同じ名前のＩＢＭファイルがあった場合、アロケート処理を中断します。こちらが省略値です。

／Ｆｏｒｍ ---- ＩＢＭファイルの形式を指定する

```
／Ｆｏｒｍ ( Ｅｘｔｅｎｄｅｄ ～
             ＨａｌｆＥｘｔｅｎｄｅｄ ～
             Ｐｒｉｍａｒｙ ～ )
```

この／Ｆｏｒｍオプションは、ＩＢＭファイルの形式を決める、非常に重要なオプションです。

つぎの３つのキーワードで、ＩＢＭファイルの大まかな形式を

Ｅｘｔｅｎｄｅｄ	**拡張形式**
ＨａｌｆＥｘｔｅｎｄｅｄ	**半拡張形式**
Ｐｒｉｍａｒｙ	**基本形式**

のどれにするのかを指定します。これらのキーワードを省略すると、

Ｅｘｔｅｎｄｅｄ	**拡張形式**

とみなします。

これらのキーワードと一緒に、レコード長やブロック長、ブロック化の有無などの指定をします。その指定の方法には、３つとも少しずつ違いがあります。

／Ｆｏｒｍ（拡張形式） ---- ＩＢＭファイルの形式を指定する（拡張形式）

```
／Ｆｏｒｍ　［Ｅｘｔｅｎｄｅｄ］
              ［論理レコード長］
                ┌ ブロック長指定 ┐
                └ Ａｕｔｏ       ┘
                  ┌ レコード形式 ┐
                  └ ＦＢ         ┘
                    ┌ Ｃｌｉｐ   ┐
                    └ ＮｏＣｌｉｐ ┘
```

新規作成するＩＢＭファイルの形式を指定します。拡張形式では論理レコード長、ブロック長、レコード形式、およびブロック長の処理を指定します。

Ｅｘｔｅｎｄｅｄ

拡張形式にすることを指定します。省略もできます。

論理レコード長

論理レコード長を１～９９９９の範囲の１０進数で指定します。この省略値は「２５６」です。

ブロック長指定

ブロック長は、以下に示すように、必要に応じていろいろな指定ができます。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｒｅｃｏｒｄｓ＜ｎ＞	＜ｎ＞レコード分（ブロック化係数による指定）
Ｍａｘ	規約上の最大値（＝トラック容量）を割り当てる
Ａｕｔｏ	論理レコード長やレコード形式に応じて、 適当なブロック長を自動的に割り当てる

（＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

レコード形式

レコード形式は、

Ｆ	固定長、非ブロック化・非スパン
ＦＢ	固定長、ブロック化・非スパン
ＦＳ	固定長、非ブロック化・スパン
ＦＢＳ	固定長、ブロック化・スパン

のどれかで指定します。省略すると、ＦＢ指定とみなします。

Ｃｌｉｐ、ＮｏＣｌｉｐ　…　ブロック長の処理

非スパン形式（Ｆ、ＦＢ）のとき、ブロック長の切り詰め処理をするかどうかを指定します。

Ｃｌｉｐを指定すると、レコード形式に応じ、

Ｆなら	ブロック長を論理レコード長と同じ長さにする
ＦＢなら	ブロック長を論理レコード長の整数倍に切り詰める

という処理をします。省略すると、このＣｌｉｐ指定になります。

ＮｏＣｌｉｐを指定すると、この処理はしません。指定したブロック長がそのまま有効になります。

ふつうはＣｌｉｐ指定で使います。ＮｏＣｌｉｐ指定を使うのは、特別な場合に限ります。

◆注意　----　ふつうは論理レコード長やブロック長だけ指定すれば十分

Ｅｘｔｅｎｄｅｄの指定、論理レコード長、ブロック長指定、レコード形式、Ｃｌｉｐ、ＮｏＣｌｉｐ指定はどれも省略可能です。また、指定の順序も任意です。空白で区切る必要もありません。ふつうは、

```
／ｆ８０　　（／Ｆｏｒｍ　８０）
／ｆ１００ｓ５００　　（／Ｆｏｒｍ　１００　Ｓｉｚｅ５００）
```

のように、論理レコード長やブロック長だけ指定すれば十分です。

／Ｆｏｒｍ（半拡張形式） ---- ＩＢＭファイルの形式を指定する（半拡張形式）

```
／Ｆｏｒｍ　ＨａｌｆＥｘｔｅｎｄｅｄ
              ［論理レコード長］
                ｛ ブロック長指定
                  Ａｕｔｏ          ｝
                    ｛ レコード形式
                      ＦＢ          ｝
                      ｛ Ｃｌｉｐ
                        ＮｏＣｌｉｐ ｝
```

新規作成するＩＢＭファイルの形式を指定します。ＨａｌｆＥｘｔｅｎｄｅｄを指定すると、半拡張形式になります。半拡張形式では論理レコード長、ブロック長、レコード形式、および、ブロック長の処理を指定します。

ＨａｌｆＥｘｔｅｎｄｅｄ

半拡張形式にすることを指定します。

論理レコード長

論理レコード長を１～９９９９の範囲の１０進数で指定します。この省略値は「２５６」です。

ブロック長指定

ブロック長は、以下に示すように、必要に応じていろいろな指定ができます。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｒｅｃｏｒｄｓ＜ｎ＞	＜ｎ＞レコード分（ブロック化係数による指定）
Ｍａｘ	規約上の最大値（＝トラック容量）を割り当てる
Ａｕｔｏ	論理レコード長やレコード形式に応じて、 適当なブロック長を自動的に割り当てる （＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

レコード形式

レコード形式は、

Ｆ　　固定長、非ブロック化・非スパン
ＦＢ　　固定長、ブロック化・非スパン

のどちらかで指定します。省略すると、ＦＢ指定とみなします。

Ｃｌｉｐ、ＮｏＣｌｉｐ　…　ブロック長の処理

ブロック長の切り詰め処理をするかどうかを指定します。

Ｃｌｉｐを指定すると、レコード形式に応じ、

Ｆなら　　ブロック長を論理レコード長と同じ長さにする
ＦＢなら　　ブロック長を論理レコード長の整数倍に切り詰める

という処理をします。省略すると、このＣｌｉｐ指定になります。

ＮｏＣｌｉｐを指定すると、この処理はしません。指定したブロック長がそのまま有効になります。

ふつうはＣｌｉｐ指定で使います。ＮｏＣｌｉｐ指定を使うのは、特別な場合に限ります。

◆注意　----　ふつうは論理レコード長やブロック長だけ指定すれば十分

論理レコード長、ブロック長指定、レコード形式、Ｃｌｉｐ、ＮｏＣｌｉｐ指定はどれも省略可能です。また、指定の順序も任意です。空白で区切る必要もありません。

ふつうは、

```
／ｆｈｅ８０　　(/Form HalfExtended 80)
／ｆｈｅ１００ｓ５００　　(/Form HalfExtended 100 Size 500)
```

のように、ＨａｌｆＥｘｔｅｎｄｅｄのキーワードと、論理レコード長やブロック長だけ指定すれば十分です。

／Ｆｏｒｍ（基本形式）　----　ＩＢＭファイルの形式を指定する（基本形式）

／Ｆｏｒｍ　Ｐｒｉｍａｒｙ	ブロック長指定 Ａｕｔｏ

新規作成するＩＢＭファイルの形式を指定します。Ｐｒｉｍａｒｙを指定すると基本形式になります。基本形式では「論理レコード」の概念がないので、ブロック長の指定だけが有効です（ブロック＝レコードと考えてください）。

Ｐｒｉｍａｒｙ
基本形式にすることを指定します。

ブロック長指定
ブロック長はつぎのように指定します。

Ｓｉｚｅ＜ｎ＞	＜ｎ＞バイト
Ｍａｘ	規約上の最大値（＝セクタ長）を割り当てる
Ａｕｔｏ	セクタ長を割り当てる。結局、Ｍａｘ指定と同じになる
	（＜＞は表記上の記号で、入力はしません）

省略するとＡｕｔｏ指定とみなします。

◆注意　----　ブロック長をバイト数で指定するときは、Ｓｉｚｅを忘れずに

たとえば全銀フォーマットのＩＢＭファイルを作りたいものとします。その場合、基本形式で、ブロック長（＝レコード長）＝１２０バイトに指定することになります。そのとき、

／ｆｐ１２０　　（／Ｆｏｒｍ　Ｐｒｉｍａｒｙ　１２０）

と指定してしまいがちです。しかし、この場合１２０は論理レコード長にあたり、基本形式のときは無視されてしまいます。基本形式ではブロック長を指定しなければいけないので、この例では

／ｆｐｓ１２０　　（／Ｆｏｒｍ　Ｐｒｉｍａｒｙ　Ｓｉｚｅ１２０）

と指定しなければいけません。

／ＢｏｅＢｏｕｎｄａｒｙ ---- ＩＢＭファイルのＢＯＥ境界を指定する

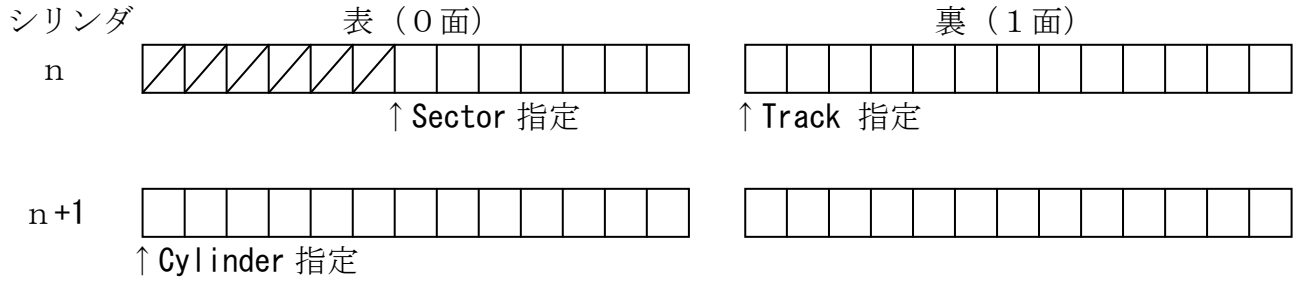
／ＢｏｅＢｏｕｎｄａｒｙ	Ｓｅｃｔｏｒ Ｔｒａｃｋ Ｃｙｌｉｎｄｅｒ

ＩＢＭファイルを新規作成するとき、どんな境界に配置するか（ＢＯＥをどの境界に置くか）を指定します。３つの境界が指定でき、それぞれ、

Ｓｅｃｔｏｒ	前のＩＢＭファイルと隙間を置かないで、セクタ境界で配置
Ｔｒａｃｋ	トラック境界で配置
Ｃｙｌｉｎｄｅｒ	シリンダ境界で配置

という意味になります。／ＢｏｅＢｏｕｎｄａｒｙオプションを省略すると、Ｔｒａｃｋ指定したとみなします。なお、ＢＯＥとは「領域開始アドレス（Beginning Of Extent）」のことです。

言葉ではわかりにくいので、図にしてみます。下図の／の部分まで、直前のＩＢＭファイルで使っているものとします。↑が、それぞれの指定によるＩＢＭファイルの配置される位置です。



Ｓｅｃｔｏｒ指定ならスペース効率がよくなるという利点があります。一方、Ｔｒａｃｋ指定とＣｙｌｉｎｄｅｒ指定は、切りのよい所でファイルがはじまるのでスペースを無駄にしますが、管理しやすい利点があります。また、「ファイルは常にセクタ０１からはじまること」としているシステムもかなりありますが、Ｔｒａｃｋ指定かＣｙｌｉｎｄｅｒ指定でこの要求に対応できます。

なお、日立のシステムではセクタ境界で出力したＩＢＭファイルを扱えません。そこで、Ｆ＊ＴＲＡＮ２００７を日立モードで動かしている場合は、Ｓｅｃｔｏｒ指定しても、自動的にＴｒａｃｋ指定に切り替えられます（エラーにはなりません）。

／ＥｏｅＢｏｕｎｄａｒｙ ---- ＩＢＭファイルのＥＯＥ境界を指定する

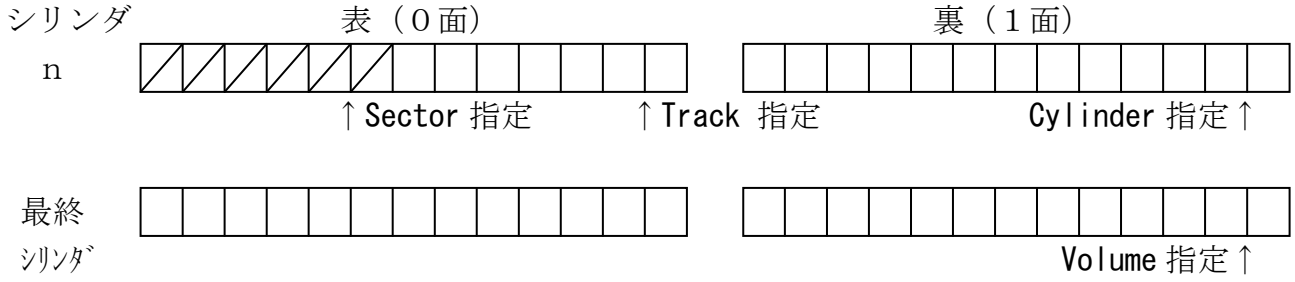
／ＥｏｅＢｏｕｎｄａｒｙ	Ｓｅｃｔｏｒ Ｔｒａｃｋ Ｃｙｌｉｎｄｅｒ Ｖｏｌｕｍｅ ＊

ＩＢＭファイルを新規作成するとき、どんな境界でＩＢＭファイルをおわらせるのか（ＥＯＥをどの境界に置くか）を指定します。４つの境界が指定でき、それぞれ、

Ｓｅｃｔｏｒ	データの書き込みがおわったセクタでＥＯＥにする
Ｔｒａｃｋ	ＥＯＥをトラック境界に切り上げる
Ｃｙｌｉｎｄｅｒ	ＥＯＥをシリンダ境界に切り上げる
Ｖｏｌｕｍｅ	ＥＯＥをディスクのおわりに置く
＊	／ＢｏｅＢｏｕｎｄａｒｙオプションの指定を引き継ぐ

という意味になります。／ＥｏｅＢｏｕｎｄａｒｙオプションを省略すると、＊指定したとみなし、／ＢｏｅＢｏｕｎｄａｒｙオプションの指定を引き継ぎます。なお、ＥＯＥとは「領域終了アドレス（End Of Extent)」のことです。

言葉ではわかりにくいので、図にしてみます。下図の／の部分で、最終ブロックの書き込みがおわっているものとします。↑が、それぞれの指定によるＩＢＭファイルのＥＯＥを置く位置です。



Ｓｅｃｔｏｒ指定なら､つづくＩＢＭファイルもセクタ境界に配置するとスペース効率がよくなるという利点があります。一方、Ｔｒａｃｋ指定やＣｙｌｉｎｄｅｒ指定は無駄なスペースが発生しますが、管理しやすい意味があります。Ｖｏｌｕｍｅ指定は要するに、「ＩＢＭディスクの残りの全領域確保」ということです。

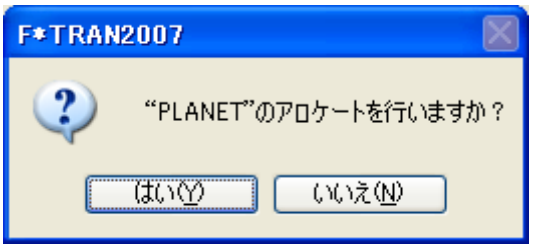
一部に、「ファイルは常にトラック境界でおわること」としているシステムもあります。Ｔｒａｃｋ指定かＣｙｌｉｎｄｅｒ指定で、この要求に対応できます。

／Ｑｕｅｒｙ -------- アロケートするか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動アロケートする

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

処理を問い合わせるか否かを指定します。

／Ｑｕｅｒｙオプションを指定すると、Ｆ＊ＴＲＡＮ２００７は処理を実行するか問い合わせます。こちらが省略値です。
つぎのどれかで応答してください。



はい（Ｙ）	表示中のファイル名でアロケートする
いいえ（Ｎ）	アロケートしない

／ＮｏＱｕｅｒｙオプションを指定すると、確認なしで自動的にアロケートします。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ ---- ボリューム順序番号の出力の制御

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ	［Ｔｈｅｎ］	Ｂｌａｎｋ ０１

このオプションでは､ファイルラベルのボリューム順序番号の欄をどうするか制御することができます。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋの場合

確保した領域が結果的にフロッピー１枚に収まる場合で、ボリューム順序番号が「01」にあたる場合、ファイルラベルのボリューム順序番号の欄を空白にします。フロッピー１枚に収まらない領域を確保した場合は、指定された順番にボリューム順序番号を書き込みます。

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　０１の場合

確保した領域が結果的にフロッピー１枚に収まる場合で、ボリューム順序番号が「０１」にあたる領域を確保した時に、ボリューム順序番号の欄に「01」を書き込みます。

省略値は、

／ＩｆＳｉｎｇｌｅＶｏｌｕｍｅ　Ｔｈｅｎ　Ｂｌａｎｋ

です。

■使用例

例１）フロッピーディスクをまたいだ領域をアロケートする。

フロッピーディスクをまたいだ領域をアロケートします。（「ＰＬＡＮＥＴ」はファイル名）

```
C:¥>ft iAlloc A:PLANET /VS 1 To 2 Last ↓
```

例２）１０キロバイト分の領域を指定してアロケートする。

Ｈ型ＩＢＭ形式フロッピーディスクに、総レコード件数指定で 10 キロバイト分をアロケートします。（２５６バイト＊４０件＝１０キロバイト）

```
C:¥>ft iAlloc a:PLANET /SP 40↓
```

２．１８　ｉＡｔｔｒ（ｉａｔ）　アイ・アター

ＩＢＭディスク内ファイルの属性変更

ｉＡｔｔｒ(ｉａｔ)は、ＩＢＭディスク内ファイルの属性を変更する機能です。

■コマンド形式

コマンド	パラメータ
ｉＡｔｔｒ ｉａｔ	ドライブ名：ＩＢＭファイル名　［オプション］

■パラメータの説明

●ドライブ名：　ＩＢＭファイル名　［オプション］

Ａ：ＩＢＭファイル名　オプション

Ａ：はドライブ名

対象となるディスクを挿入するドライブ名とＩＢＭファイル名を入力します。省略はできません。続いて任意のオプションを入力します。
ＩＢＭファイル名にはワイルドカード文字（＊）も使えます。

たとえば、

Ｃ：￥＞ｆｔ　ｉＡｔｔｒ　ａ：＊　／ＣｒｅａｔｉｏｎＤａｔｅ　０６０４０１↓

のように指定します。共通の属性を全てのファイルに反映したい時などに便利です。

●指定できるオプション

iAttr(iat)コマンドには、

／Mode	モード
／DataExchangeType	データ交換タイプ
／BlockSize	ブロック長
／RecordLength	レコード長
／RecordBlockForm	レコード／ブロック形式
／RecordAttributes	レコード属性
／VolumeSequenceNumber	ボリューム順序番号
／MultiVolumeFlag	マルチボリュームフラグ
／BOE	領域開始アドレス
／EOE	領域終了アドレス
／EOD	データ終了アドレス
／Offset	オフセット
／CreationDate	作成日付
／ExpirationDate	満了日付
／BypassFlag	バイパスフラグ
／ProtectionFlag	書込禁止フラグ
／VerifyCopyFlag	ベリファイ／コピーフラグ
／FileOrganization	ファイル編成
／SectorSize	セクタ長
／Query	変更するか否かを問い合わせる
／NoQuery	確認せず変更する

というオプションがあります。

◆注意　----　使用には十分注意する

iAttrコマンドでは、かなり自由な変更が可能となっています。そのため、誤った属性を指定すると、ファイルとして認識できなくなる恐れがあります。IBM形式ファイルの規約を十分理解した上でご利用下さい。

●オプションの詳細

／Ｍｏｄｅ　　モード

／Ｍｏｄｅ	Ｓａｆｅｔｙ
	Ｆｒｅｅ
	Ｒｉｓｋｙ

ファイル属性変更ではかなり自由な値を指定することができます。そのため、誤った属性を指定すると、ファイルとして認識できなくなる場合があります。そのため、動作モードとして、

Ｓａｆｅｔｙ	安全モード
Ｆｒｅｅ	自由モード
Ｒｉｓｋｙ	危険モード

という３つのモードを用意しました。

安全モード

この範囲で使っているかぎり基本的に問題は起きない安全なモードです。

自由モード

自由モードではかなり自由な変更が出来てしまうため、問題を引き起こすことがあります。

危険モード

危険モードでは、サポートしているすべての項目に、任意の英数字を設定できるようになります。規格外れのＩＢＭディスク・ファイルを扱うために存在します。

したがって、自由モードや危険モードを使うには、ＩＢＭディスク・ファイルの管理方式を良く理解して、十分な注意を払う必要があります。

以下は各モードが指定された際にｉＡｔｔｒコマンドの他のオプションの指定の可否を表した表です。

オプション	安全モード	自由モード	危険モード
／ＤａｔａＥｘｃｈａｎｇｅＴｙｐｅ	×	○	○
／ＢｌｏｃｋＳｉｚｅ	×	○	○
／ＲｅｃｏｒｄＬｅｎｇｔｈ	×	○	○
／ＲｅｃｏｒｄＢｌｏｃｋＦｏｒｍ	×	○	○
／ＲｅｃｏｒｄＡｔｔｒｉｂｕｔｅｓ	×	○	○
／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅＮｕｍｂｅｒ	○	○	○
／ＭｕｌｔｉＶｏｌｕｍｅＦｌａｇ	○	○	○
／ＢＯＥ	×	○	○
／ＥＯＥ	×	○	○
／ＥＯＤ	○	○	○
／Ｏｆｆｓｅｔ	○	○	○
／ＣｒｅａｔｉｏｎＤａｔｅ	○	○	○
／ＥｘｐｉｒａｔｉｏｎＤａｔｅ	○	○	○
／ＢｙｐａｓｓＦｌａｇ	○	○	○
／ＰｒｏｔｅｃｔｉｏｎＦｌａｇ	○	○	○
／ＶｅｒｉｆｙＣｏｐｙＦｌａｇ	○	○	○
／ＦｉｌｅＯｒｇａｎｉｚａｔｉｏｎ	○	○	○
／ＳｅｃｔｏｒＳｉｚｅ	×	○	○

※　○は指定内容反映。×は指定しても処理の際に無視

／ＤａｔａＥｘｃｈａｎｇｅＴｙｐｅ　　データ交換タイプ

／ＤａｔａＥｘｃｈａｎｇｅＴｙｐｅ	＊ － Ｈ Ｅ 英数字

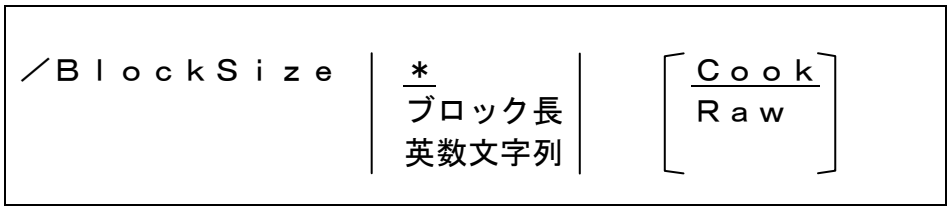
データ交換タイプには、

＊	元のまま （自由／危険モード）
－	空白＝標準データ交換 （基本形式、１Ｓ／２Ｓ－１２８のみに適用。自由／危険モード）
Ｈ	Ｈ型データ交換 （基本形式、２ＨＤ－２５６のみに適用。自由／危険モード）
Ｅ	Ｅ型データ交換 （拡張形式、全媒体に適用。自由／危険モード）
英数字	その他の英数字 （危険モード）

が設定できます。

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／Ｈ／Ｅが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／ＢｌｏｃｋＳｉｚｅ　　ブロック長



ブロック長を指定します。以下の指定が可能です。

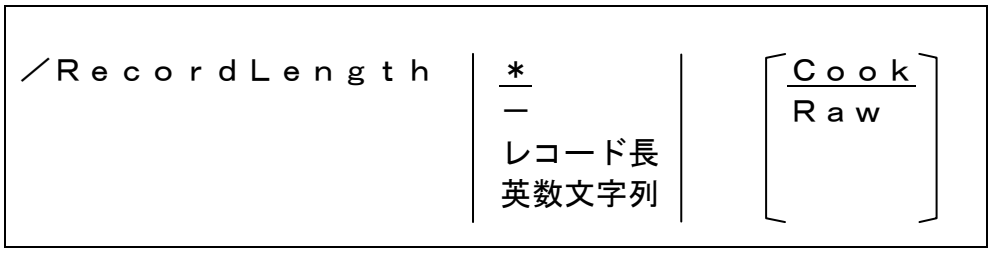
＊	元のブロック長のまま（自由／危険モード）
ブロック長	１～３２７６７の範囲でブロック長（自由／危険モード）
英数文字列	任意の５桁の英数文字列（危険モード）

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたブロック長に、前ゼロをつける
Ｒａｗ	指定されたブロック長文字列を、そのまま加工せず右詰にする

安全モードでは、このオプションは無視されます。
自由モードでは、＊／１０進数が設定できます。
危険モードでは、任意の英数文字列が設定できます。また、加工パラメータ設定の有無は無視されます。

／ＲｅｃｏｒｄＬｅｎｇｔｈ　　レコード長



レコード長を設定します。以下の指定が可能です。

＊	元のレコード長のまま（自由／危険モード）
－	空白＝ブロック長をレコード長とみなす（自由／危険モード）
レコード長	１～９９９９の範囲（自由／危険モード）
英数文字列	任意の４桁の英数文字列（危険モード）

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたレコード長に、前ゼロをつけて設定
Ｒａｗ	指定されたレコード長文字列を、そのまま加工せず右詰で設定

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／１０進数が設定できます。
危険モードでは、さらに任意の４桁の英数文字列が設定できます。また、加工パラメータ設定の有無は無視されます。

／ＲｅｃｏｒｄＢｌｏｃｋＦｏｒｍ　　レコード／ブロック形式

／ＲｅｃｏｒｄＢｌｏｃｋＦｏｒｍ	＊ － Ｆ 英数文字列

レコード／ブロックの形式を設定します。以下の指定が可能です。

＊	元のまま（自由／危険モード）
－	空白＝固定長（自由／危険モード）
Ｆ	固定長（自由／危険モード）
英数字	その他の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／Ｆが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／ＲｅｃｏｒｄＡｔｔｒｉｂｕｔｅｓ　　レコード属性

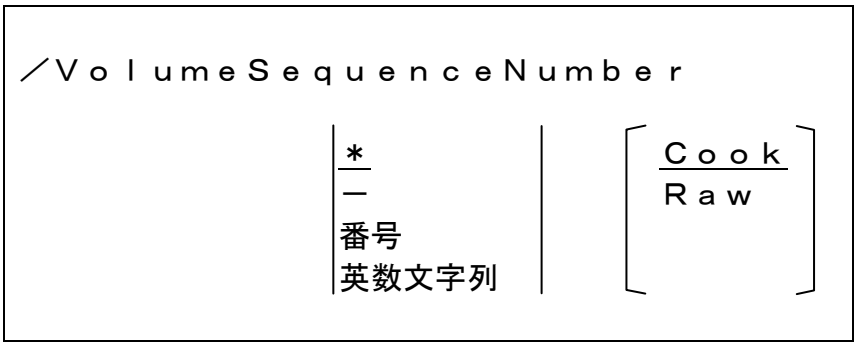
／ＲｅｃｏｒｄＡｔｔｒｉｂｕｔｅｓ	＊ － Ｂ Ｓ Ｒ 英数字

ブロック化とスパン化の有無を設定します。以下の指定が可能です。

＊	元のまま（自由／危険モード）
－	空白（非ブロック化・非スパン（Ｆ））を設定（自由／危険モード）
Ｂ	ブロック化・非スパン（ＦＢ）を設定（自由／危険モード）
Ｓ	非ブロック化・スパン（ＦＳ）を設定（自由／危険モード）
Ｒ	ブロック化・スパン（ＦＢＳ）を設定（自由／危険モード）
英数字	その他の英数字を設定

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／Ｂ／Ｓ／Ｒが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／ＶｏｌｕｍｅＳｅｑｕｅｎｃｅＮｕｍｂｅｒ　　ボリューム順序番号



ボリューム順序番号を設定します。以下の指定が可能です。

＊	元のボリューム順序番号のまま
－	空白＝ボリューム順序番号なし
番号	０１～９９の範囲でボリューム順序番号
英数文字列	任意の２桁の英数文字列

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたボリューム順序番号に、前ゼロをつけて設定
Ｒａｗ	指定されたボリューム順序番号文字列をそのまま右詰で設定

安全モードと自由モードでは、＊／－／１０進数が設定できます。

危険モードでは、任意の２桁の英数文字列が設定できます。また、加工パラメータ設定の有無は無視されます。

／ＭｕｌｔｉＶｏｌｕｍｅＦｌａｇ　　マルチボリュームフラグ

／ＭｕｌｔｉＶｏｌｕｍｅＦｌａｇ	＊ － Ｃ Ｌ 英数字

マルチボリュームフラグを設定します。以下の指定が可能です。

＊	元のまま(安全モード／自由モード／危険モード)
－	空白＝シングルボリューム(安全モード／自由モード／危険モード)
Ｃ	マルチボリュームの途中のボリューム (安全モード／自由モード／危険モード)
Ｌ	マルチボリュームの最終ボリューム (安全モード／自由モード／危険モード)
英数字	その他の英数字を設定（危険モード）

安全モードと自由モードでは、＊／－／Ｃ／Ｌが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／ＢＯＥ　　ＢＯＥ（領域開始アドレス）

ＢＯＥ（Beginning of Extent＝領域開始アドレス）を設定します。以下の指定が可能です。

＊	元のＢＯＥのまま（自由モード／危険モード）
ｃｃｈｒｒ	ＢＯＥをＣＣＨＲＲ形式で設定（自由モード／危険モード）
英数文字列	任意の５桁の英数文字列を設定（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／ｃｃｈｒｒが設定できます。
危険モードでは、さらに任意の５桁の英数文字列が設定できます。

このオプションは、１ファイル指定時にのみ有効になります。複数ファイルが指定された場合、このオプションが指定されても無視されます。

／ＥＯＥ　　ＥＯＥ（領域終了アドレス）

ＥＯＥ（End of Extent＝領域終了アドレス）を設定します。以下の指定が可能です。

＊	元のＥＯＥのまま（自由モード／危険モード）
ｃｃｈｒｒ	ＥＯＥをＣＣＨＲＲ形式で設定（自由モード／危険モード）
英数文字列	任意の５桁の英数文字列を設定（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／ｃｃｈｒｒが設定できます。
危険モードでは、さらに任意の５桁の英数文字列が設定できます。

このオプションは、１ファイル指定時にのみ有効になります。複数ファイルが指定された場合、このオプションが指定されても無視されます。

／ＥＯＤ　　ＥＯＤ（データ終了アドレス）

ＥＯＤ（End of Data＝データ終了アドレス）を設定します。以下の指定が可能です。

＊	元のＥＯＤのまま(安全モード／自由モード／危険モード)
ｃｃｈｒｒ	ＣＣＨＲＲ形式（安全モード／自由モード／危険モード)
英数文字列	任意の５桁の英数文字列(危険モード)

安全モードと自由モードでは、＊／ｃｃｈｒｒが設定できます。
危険モードでは、さらに任意の５桁の英数文字列が設定できます。

このオプションは、１ファイル指定時にのみ有効になります。複数ファイルが指定された場合、このオプションが指定されても無視されます。

／Ｏｆｆｓｅｔ　　オフセット

／Ｏｆｆｓｅｔ	＊ － １０進数 英数文字列	[Ｃｏｏｋ Ｒａｗ]

オフセット（最終ブロック内残り長）を設定します。以下の指定が可能です。

＊	元のオフセットのまま(安全モード／自由モード／危険モード)
－	空白を設定(安全モード／自由モード／危険モード)
１０進数	１～３２７６７の範囲の１０進数を設定 (安全モード／自由モード／危険モード)
英数文字列	任意の５桁の英数文字列(危険モード)

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたオフセットに前ゼロをつけ、値が０なら空白にする
Ｒａｗ	指定されたオフセット文字列を、そのまま加工せず右詰で設定

安全モードと自由モードでは、＊／－／１０進数が設定できます。
危険モードでは、任意の英数文字列が設定できます。また、加工パラメータ設定の有無は無視されます。

このオプションは、１ファイル指定時にのみ有効になります。複数ファイルが指定された場合、このオプションが指定されても無視されます。

／ＣｒｅａｔｉｏｎＤａｔｅ　　作成日付

／ＣｒｅａｔｉｏｎＤａｔｅ	＊ － ｙｙｍｍｄｄ 英数文字列	

作成日付を設定します。以下の指定が可能です。

＊	元の作成日付のまま（安全モード／自由モード／危険モード）
－	空白（安全モード／自由モード／危険モード）
ｙｙｍｍｄｄ	西暦ＹＹＭＭＤＤ形式で作成日付 （安全モード／自由モード／危険モード）
英数文字列	任意の６桁の英数文字列（危険モード）

安全モードと自由モードでは、＊／－／ｙｙｍｍｄｄが設定できます。
危険モードでは、さらに任意の６桁の英数文字列が設定できます。

／ＥｘｐｉｒａｔｉｏｎＤａｔｅ　　満了日付

／ＥｘｐｉｒａｔｉｏｎＤａｔｅ	＊ － ｙｙｍｍｄｄ ９９９９９９

満了日付を設定します。以下の指定が可能です。

＊	元の満了日付のまま（安全モード／自由モード／危険モード）
－	空白＝満了日付を使用しない＝すでに満了済み （安全モード／自由モード／危険モード）
ｙｙｍｍｄｄ	西暦ＹＹＭＭＤＤ形式で設定 （安全モード／自由モード／危険モード）
９９９９９９	永久に満了しないように設定 （安全モード／自由モード／危険モード）
英数文字列	任意の６桁の英数文字列（危険モード）

安全モードと自由モードでは、＊／－／ｙｙｍｍｄｄ／９９９９９９が設定できます。
危険モードでは、任意の６桁の英数文字列が設定できます。

◆注意　----　満了日付とＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７がファイルに対して行う変換などの処理は、ファイルの満了日付は無視して行われます。また、満了日になる前のファイルをＦ＊ＴＲＡＮによって改変させないように設定することも出来ません。

／ＢｙｐａｓｓＦｌａｇ　　バイパスフラグ

／ＢｙｐａｓｓＦｌａｇ	＊ － Ｂ 英数字

バイパスフラグを設定します。以下の指定が可能です。

＊	元の値のまま（安全モード／自由モード／危険モード）
－	空白＝処理対象とする（安全モード／自由モード／危険モード）
Ｂ	処理対象外（安全モード／自由モード／危険モード）
英数字	その他の英数字（危険モード）

安全モードと自由モードでは、＊／－／Ｂが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

◆注意　----　バイパスフラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７がファイルに対して行う変換などの処理は、バイパスフラグの内容は無視して行われます。

／ＰｒｏｔｅｃｔｉｏｎＦｌａｇ　　書込禁止フラグ

／ＰｒｏｔｅｃｔｉｏｎＦｌａｇ	＊ － Ｐ 英数字

書込禁止フラグを設定します。以下の指定が可能です。

＊	元の値のまま（安全モード／自由モード／危険モード）
－	空白＝書込禁止にしない（安全モード／自由モード／危険モード）
Ｐ	書込禁止（安全モード／自由モード／危険モード）
英数字	その他の英数字（危険モード）

安全モードと自由モードでは、＊／－／Ｐが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／ＶｅｒｉｆｙＣｏｐｙＦｌａｇ　　ベリファイ／コピーフラグ

／ＶｅｒｉｆｙＣｏｐｙＦｌａｇ	＊ － Ｖ Ｃ 英数字

ベリファイ／コピーフラグを設定します。以下の指定が可能です。

＊	元の値のまま（安全モード／自由モード／危険モード）
－	空白＝ファイル作成時のまま（安全モード／自由モード／危険モード）
Ｖ	ベリファイ済み（安全モード／自由モード／危険モード）
Ｃ	コピー済み（安全モード／自由モード／危険モード）
英数字	その他の英数字（危険モード）

安全モードと自由モードでは、＊／－／Ｖ／Ｃが設定できます。
危険モードでは、さらに任意の英数字が設定できます。

◆注意　----　ベリファイ／コピーフラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７がファイルに対して行う変換などの処理は、ベリファイ／コピーフラグの内容は無視して行われます。

／ＦｉｌｅＯｒｇａｎｉｚａｔｉｏｎ　　ファイル編成

／ＦｉｌｅＯｒｇａｎｉｚａｔｉｏｎ	＊ － Ｓ Ｄ 英数字

ファイル編成を設定します。以下の指定が可能です。

＊	元のまま（安全モード／自由モード／危険モード）
－	空白＝順編成・逐次再配置可（安全モード／自由モード／危険モード）
Ｓ	順編成・逐次再配置可（安全モード／自由モード／危険モード）
Ｄ	逐次再配置不可（安全モード／自由モード／危険モード）
英数字	その他の英数字（危険モード）

安全モードと自由モードでは、＊／－／Ｓ／Ｄが設定できる。
危険モードでは、さらに任意の英数字が設定できる。

◆注意　----　ファイル編成とＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７がファイルに対して行う変換などの処理は、ファイル編成設定の内容は無視して行われます。

／ＳｅｃｔｏｒＳｉｚｅ　　セクタ長

／ＳｅｃｔｏｒＳｉｚｅ	＊ － １ ２ ３ 英数字

セクタ長を設定します。以下の指定が可能です。

＊	元のまま（自由モード／危険モード）
－	１２８バイト（自由モード／危険モード）
１	２５６バイト（自由モード／危険モード）
２	５１２バイト（自由モード／危険モード）
３	１０２４バイト（自由モード／危険モード）
英数字	その他の英数字（危険モード）

安全モードではこのオプションは無視されます。
自由モードでは、＊／－／１／２／３が設定できます。
危険モードでは、さらに任意の英数字が設定できます。

／Ｑｕｅｒｙ -------- 属性変更するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- 確認なしで属性変更する

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

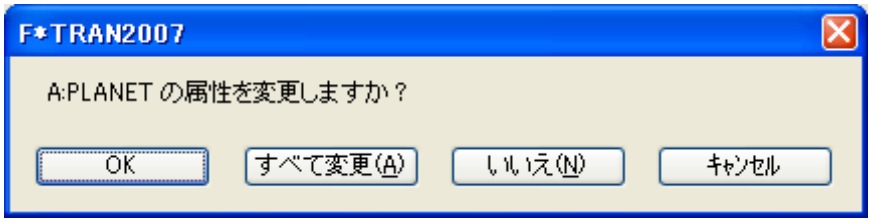
１ファイルごとに処理を問い合わせるか否かを指定します。

／Ｑｕｅｒｙオプションを指定すると、Ｆ＊ＴＲＡＮ２００７は１ファイルごとにその処理を実行するか問い合わせます。
つぎのどれかで応答してください。

１ファイルの変更

はい（Ｙ）	表示中のファイルの属性を変更する
いいえ（Ｎ）	表示中のファイルの属性は変更しない

２ファイル以上の変更



ＯＫ	表示中のファイルの属性を変更する
すべて変更（Ａ）	全ファイル変更に切り替え、以降のファイルの属性をすべて変更する
いいえ（Ｎ）	表示中のファイルの属性は変更しない
キャンセル	これ以降の処理を中断する

／ＮｏＱｕｅｒｙオプションを指定すると、ファイル名の確認なしで自動的に指定のファイルの属性をすべて変更します。こちらが省略値です。

●オプション省略時の動作

オプションを省略すると、何も変更しません。

２．１９　ｉＶｏｌＡｔｔｒ（ｉｖａ）　アイ・ボル・アター

ＩＢＭディスクのボリュームラベルを編集

ｉＶｏｌＡｔｔｒ(ｉｖａ)は、ＩＢＭディスクのボリューム属性を変更する機能です。

■コマンド形式

コマンド	パラメータ
ｉＶｏｌＡｔｔｒ ｉｖａ	ドライブ名：　［オプション］

■パラメータの説明

●ドライブ名

Ａ：

Ａ：はドライブ名

ボリューム属性を変更するフロッピーが挿入するドライブを、Ａ：～Ｐ：、または０：～３：で指定します。省略はできません。

たとえば、

Ｃ：￥＞ｆｔ　ｉＶｏｌＡｔｔｒ　ａ：　／Ｏｗｎｅｒ　ＢＳＣ↓

のように指定します。

●指定できるオプション

ｉＶｏｌＡｔｔｒ(ｉｖａ)コマンドには、

／Ｍｏｄｅ	モード
／ＶｏｌｕｍｅＩＤ	ボリュームＩＤ
／Ｏｗｎｅｒ	所有者名
／ＳｅｃｔｏｒＩｎｔｅｒｌｅａｖｅ	セクタ順番
／ＶｏｌｕｍｅＳｉｄｅＲｅｃｏｇｎｉｔｉｏｎ	ボリューム面認識
／ＳｅｃｔｏｒＳｉｚｅ	セクタ長
／ＬａｂｅｌＥｘｔｅｎｔｉｏｎＦｌａｇ	ラベル拡張フラグ
／ＥｘｔｅｎｔＡｌｌｏｃａｔｉｏｎＦｌａｇ	領域配置フラグ
／ＳｐｅｃｉａｌＲｅｑｕｉｒｅｍｅｎｔＦｌａｇ	特殊要求フラグ
／ＬａｂｅｌＳｔａｎｄａｒｄｉｚａｔｉｏｎＬｅｖｅｌ	ラベル標準化水準
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＯｎｅ	欠陥シリンダフラグ１
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＯｎｅ	欠陥シリンダ番号１
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＴｗｏ	欠陥シリンダフラグ２
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＴｗｏ	欠陥シリンダ番号２
／ＤｅｆｅｃｔｉｖｅＳｅｃｔｏｒＦｌａｇ	欠陥セクタフラグ
／Ｑｕｅｒｙ	変更するか否かを問い合わせる
／ＮｏＱｕｅｒｙ	確認せず変更する

というオプションがあります。

◆注意　----　使用には十分注意する

ｉＶｏｌＡｔｔｒコマンドでは、かなり自由な変更が可能となっています。そのため、誤った属性を指定すると、正常なフロッピーとして認識できなくなる恐れがあります。ＩＢＭフォーマットの規約を十分理解した上でご利用下さい。

●オプションの詳細

／Ｍｏｄｅ　　モード

／Ｍｏｄｅ	Ｓａｆｅｔｙ Ｆｒｅｅ Ｒｉｓｋｙ

ボリューム属性変更ではかなり自由な値を指定することができます。そのため、誤った属性を指定すると、正常なフロッピーとして認識できなくなる場合があります。そのため、動作モードとして、

Ｓａｆｅｔｙ	安全モード
Ｆｒｅｅ	自由モード
Ｒｉｓｋｙ	危険モード

という３つのモードを用意しました。

安全モード

この範囲で使っているかぎり基本的に問題は起きない安全なモードです。

自由モード

自由モードではかなり自由な変更が出来てしまうため、問題を引き起こすことがあります。

危険モード

危険モードでは、サポートしているすべての項目に、任意の英数字を設定できるようになります。規格外れのＩＢＭディスク・ファイルを扱うために存在します。

したがって、自由モードや危険モードを使うには、ＩＢＭディスク・ファイルの管理方式を良く理解して、十分な注意を払う必要があります。

以下は各モードが指定された際にｉＶｏｌＡｔｔｒコマンドの他のオプションの指定の可否を表した表です。

オプション	安全モード	自由モード	危険モード
／ＶｏｌｕｍｅＩＤ	○	○	○
／Ｏｗｎｅｒ	○	○	○
／ＳｅｃｔｏｒＩｎｔｅｒｌｅａｖｅ	○	○	○
／ＶｏｌｕｍｅＳｉｄｅＲｅｃｏｇｎｉｔｉｏｎ	×	○	○
／ＳｅｃｔｏｒＳｉｚｅ	×	○	○
／ＬａｂｅｌＥｘｔｅｎｔｉｏｎＦｌａｇ	×	○	○
／ＥｘｔｅｎｔＡｌｌｏｃａｔｉｏｎＦｌａｇ	○	○	○
／ＳｐｅｃｉａｌＲｅｑｕｉｒｅｍｅｎｔＦｌａｇ	○	○	○
／ＬａｂｅｌＳｔａｎｄａｒｄｉｚａｔｉｏｎＬｅｖｅｌ	×	○	○
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＯｎｅ	×	○	○
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＯｎｅ	×	○	○
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＴｗｏ	×	○	○
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＴｗｏ	×	○	○
／ＤｅｆｅｃｔｉｖｅＳｅｃｔｏｒＦｌａｇ	×	○	○

※　○は指定内容反映。×は指定しても処理の際に無視

／ＶｏｌｕｍｅＩＤ　　ボリュームＩＤ

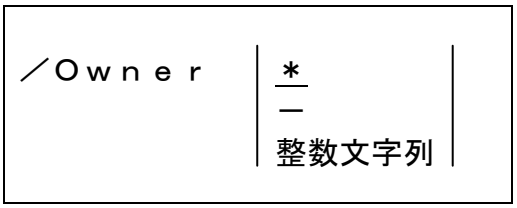
／ＶｏｌｕｍｅＩＤ	＊ － 英数文字列

ボリュームＩＤを設定します。以下の指定が可能です。

＊	元のボリュームＩＤのまま （安全モード／自由モード／危険モード）
－	空白＝ボリュームＩＤなし （安全モード／自由モード／危険モード）
英数文字列	６桁の英数文字列（安全モード／自由モード／危険モード）

安全モード、自由モード、危険モードの全てのモードにて全パラメータを指定できます。

／Ｏｗｎｅｒ



所有者名を設定します。以下の指定が可能です。

＊	元の所有者名のまま（安全モード／自由モード／危険モード）
－	空白＝所有者名なし（安全モード／自由モード／危険モード）
英数文字列	所有者名（安全モード／自由モード／危険モード）

安全モード、自由モード、危険モードの全てのモードにて全パラメータを指定できます。

／ＳｅｃｔｏｒＩｎｔｅｒｌｅａｖｅ　　セクタ順番

／ＳｅｃｔｏｒＩｎｔｅｒｌｅａｖｅ	＊ － 係数 英数文字列	［Ｃｏｏｋ Ｒａｗ］

物理セクタの並び順を示す係数を設定します。以下の指定が可能です。

＊	元の係数のまま（安全モード／自由モード／危険モード）
－	空白＝連続配置（安全モード／自由モード／危険モード）
係数	０１～１３で並び順を表す係数 （安全モード／自由モード／危険モード）
英数文字列	任意の２桁の英数文字列（危険モード）

安全モードと自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定された値に、前ゼロをつける
Ｒａｗ	指定された値を、そのまま加工せず右詰にする

安全モードと自由モードでは、＊／－／１０進数が指定できます。
危険モードでは、さらに任意の２桁の英数文字列が指定できます。

◆注意　----　セクタ順番とＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７は、変換処理等において、セクタ順番情報を参照しません。

／ＶｏｌｕｍｅＳｉｄｅＲｅｃｏｇｎｉｔｉｏｎ　　ボリューム面認識

／ＶｏｌｕｍｅＳｉｄｅＲｅｃｏｇｎｉｔｉｏｎ	＊ － ２ Ｍ 英数字

１Ｓ、２Ｓ、２ＨＤの区別を示すボリューム面認識を設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝１Ｓ＝片面単密度（自由モード／危険モード）
２	２Ｓ＝両面単密度（自由モード／危険モード）
Ｍ	２ＨＤ＝両面高密度（自由モード／危険モード）
英数字	任意の英数字を設定（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／２／Ｍが指定できます。
危険モードでは、さらに任意の英数字が指定できます。

／ＳｅｃｔｏｒＳｉｚｅ　　セクタ長

／ＳｅｃｔｏｒＳｉｚｅ	＊
	－
	１
	２
	３
	英数字

セクタ長を設定します。以下の指定が可能です。

＊	元のセクタ長のまま（自由モード／危険モード）
－	空白＝１２８バイト（自由モード／危険モード）
１	２５６バイト（自由モード／危険モード）
２	５１２バイト（自由モード／危険モード）
３	１０２４バイト（自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／１／２／３が指定できます。
危険モードでは、任意の英数字が指定できます。

／LabelExtentionFlag　　ラベル拡張フラグ

／LabelExtentionFlag	＊ － １～９ 英数字

ラベル拡張フラグを設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝ラベル拡張なし（自由モード／危険モード）
１～９	シリンダ１から指定数のシリンダをファイルラベル領域として設定（自由モード／危険モード）
英数字	任意の英数字を設定

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／１～９が指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　ラベル拡張フラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７の変換処理等では、ラベル拡張フラグを参照しません。

／ＥｘｔｅｎｔＡｌｌｏｃａｔｉｏｎＦｌａｇ　　領域配置フラグ

／ＥｘｔｅｎｔＡｌｌｏｃａｔｉｏｎＦｌａｇ	＊ － Ｐ 英数字

領域配置フラグを設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝領域配置に制約なし（自由モード／危険モード）
Ｐ	領域の配置に特殊制約あり（自由モード／危険モード）
英数字	任意の英数字を設定（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／Ｐが指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　領域配置フラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７の変換処理等においては、領域配置フラグは参照致しません。

／SpecialRequirementFlag　　特殊要求フラグ

／SpecialRequirementFlag	＊ － R 英数字

特殊要求フラグを設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝特殊な要求なし（自由モード／危険モード）
R	順編成でないファイルがディスク内に存在（自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／？が指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　特殊要求フラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮにおける変換処理等では、特殊要求フラグによって処理を変えるような制御は出来ません。

／LabelStandardizationLevel　　ラベル標準化水準

／LabelStandardizationLevel	＊ W 英数字

ラベル標準化水準を設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
W	ラベルとファイルの構成が標準に合致している（自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／Wが指定できます。
危険モードでは、さらに任意の英数字が指定できます。

／DefectiveCylinderFlagOne　　欠陥シリンダフラグ１

／DefectiveCylinderFlagOne	＊ － ０ 英数字

欠陥シリンダ１が使用されていると設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝欠陥シリンダなし（自由モード／危険モード）
０	欠陥シリンダあり（自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／０が指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　欠陥シリンダフラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７は､欠陥シリンダフラグ情報をもとに欠陥シリンダを避けて読み書きを行うことが出来ません｡ホストとのデータ交換にはエラーセクタの無いディスクをお使いください｡このフラグが空白以外に設定されていると、ｉＶｏｌＡｔｔｒ以外の処理ではエラーとなります。

／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＯｎｅ　欠陥シリンダ番号１

```
／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＯｎｅ
      ｜ ＊          ｜ ［Ｃｏｏｋ］
      ｜ －          ｜ ［Ｒａｗ］
      ｜ シリンダ番号  ｜
      ｜ 英数文字列    ｜
```

低位の欠陥シリンダの番号を設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝欠陥シリンダなし（自由モード／危険モード）
シリンダ番号	欠陥シリンダ１の番号（自由モード／危険モード）
英数文字列	任意の２桁の英数文字列（危険モード）

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたシリンダ番号に、前ゼロをつける
Ｒａｗ	指定されたシリンダ番号文字列を、そのまま加工せず右詰で

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／１０進数が指定できます。
危険モードでは、さらに任意の２桁の英数文字列が指定できます。

◆注意　----　欠陥シリンダ番号とＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７は､欠陥シリンダ番号情報をもとに欠陥シリンダを避けて読み書きを行うことが出来ません｡ホストとのデータ交換にはエラーセクタの無いディスクをお使いください。このフラグが空白以外に設定されていると､ｉＶｏｌＡｔｔｒ以外の処理ではエラーとなります。

／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＴｗｏ　　欠陥シリンダフラグ２

／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＦｌａｇＴｗｏ	＊ － １ 英数字

欠陥シリンダが２個あると設定します。以下の指定が可能です。

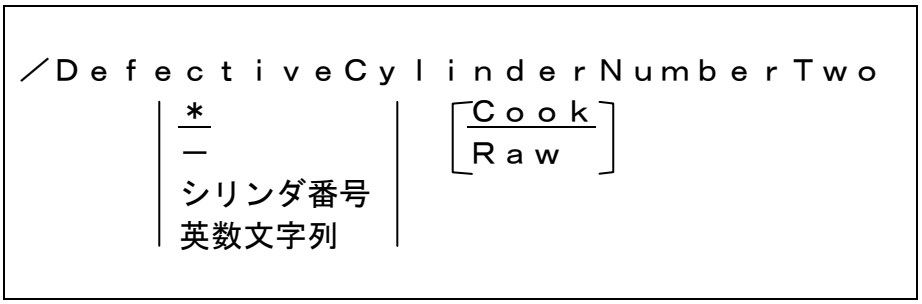
＊	元の値のまま（自由モード／危険モード）
－	空白＝欠陥シリンダが１個以下（自由モード／危険モード）
０	欠陥シリンダが２個ある＝余剰シリンダが０個ある （自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／０が指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　欠陥シリンダフラグとＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７は､欠陥シリンダフラグ情報をもとに欠陥シリンダを避けて読み書きを行うことが出来ません｡ホストとのデータ交換にはエラーセクタの無いディスクをお使いください｡このフラグが空白以外に設定されていると、ｉＶｏｌＡｔｔｒ以外の処理ではエラーとなります。

／ＤｅｆｅｃｔｉｖｅＣｙｌｉｎｄｅｒＮｕｍｂｅｒＴｗｏ　欠陥シリンダ番号２



高位の欠陥シリンダの番号を設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝欠陥シリンダが１個以下（自由モード／危険モード）
シリンダ番号	高位の欠陥シリンダの番号（自由モード／危険モード）
英数文字列	任意の２桁の英数文字列（危険モード）

自由モードで数値が指定された場合、任意に以下の指定ができます。

Ｃｏｏｋ	指定されたシリンダ番号に、前ゼロをつける
Ｒａｗ	指定されたシリンダ番号文字列を、そのまま加工せず右詰で

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／１０進数が指定できます。
危険モードでは、さらに任意の２桁の英数文字列が指定できます。

◆注意　----　欠陥シリンダ番号とＦ＊ＴＲＡＮの変換処理の関係について

Ｆ＊ＴＲＡＮ２００７は､欠陥シリンダ番号情報をもとに欠陥シリンダを避けて読み書きを行うことが出来ません｡ホストとのデータ交換にはエラーセクタの無いディスクをお使いください。このフラグが空白以外に設定されていると､ｉＶｏｌＡｔｔｒ以外の処理ではエラーとなります。

／DefectiveSectorFlag　　欠陥セクタフラグ

／DefectiveSectorFlag	＊ － D 英数字

欠陥セクタがあると設定します。以下の指定が可能です。

＊	元の値のまま（自由モード／危険モード）
－	空白＝欠陥セクタなし（自由モード／危険モード）
D	欠陥セクタがある（自由モード／危険モード）
英数字	任意の英数字（危険モード）

安全モードでは、このオプションは無視されます。
自由モードでは、＊／－／Dが指定できます。
危険モードでは、さらに任意の英数字が指定できます。

◆注意　----　欠陥セクタフラグとＦ＊ＴＲＡＮの変換処理の関係について

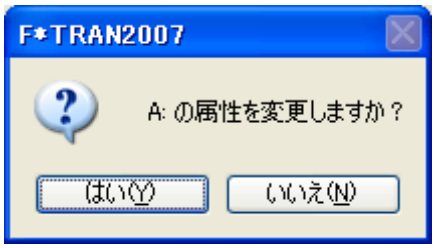
Ｆ＊ＴＲＡＮ２００７は、欠陥セクタフラグ情報をもとに欠陥セクタを避けて読み書きを行うことが出来ますが、あくまで欠陥セクタを含むブロックを避けての読み書きとなります。また、Ｆ＊ＴＲＡＮ２００７には、欠陥セクタがタディスク内のどのアドレスにあるのかを検出する機能がありません。また、欠陥セクタのアドレスを登録することもｉＶｏｌＡｔｔｒコマンドからは出来ません。

／Ｑｕｅｒｙ -------- 属性変更するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- 確認なしで属性変更する

／Ｑｕｅｒｙ ／ＮｏＱｕｅｒｙ

処理を問い合わせるか否かを指定します。

／Ｑｕｅｒｙオプションを指定すると、Ｆ＊ＴＲＡＮ２００７はその処理を実行するか問い合わせます。
つぎのどれかで応答してください。



はい（Ｙ）　　表示中のフロッピーの属性を変更する
いいえ（Ｎ）　　表示中のフロッピーの属性は変更しない

／ＮｏＱｕｅｒｙオプションを指定すると、確認なしで自動的に指定の属性をすべて変更します。こちらが省略値です。

●オプション省略時の動作

オプションを省略しますと、何も致しません。

２．20 ＩｓＲｅａｄｙ（ｉｒ）　イズ・レディ

フロッピーディスクの装着検査

ＩｓＲｅａｄｙ(ｉｒ)は､フロッピーディスク装置にフロッピーディスクが装着されているかどうかを検査するためのコマンドです。

■コマンド形式

コマンド	パラメータ
ＩｓＲｅａｄｙ ｉｒ	ドライブ名

■パラメータの説明

●ドライブ名

ｄ：

ｄ：はドライブ名

フロッピーディスク装置のドライブ名を、Ａ：～Ｐ：、または０：～３：で指定します。

■使用例

例）ＩＢＭファイルＴＥＳＴがあれば、その内容を表示する

バッチファイルでドライブＡ：にＩＢＭファイルＴＥＳＴがあるかどうかを判定します。あれば、その内容をコンソールに表示し、なければ、エラーメッセージを表示します。

＜ＴＥＳＴ．ＢＡＴ＞

```
@echo off ↓
:: ドライブＡ：にＩＢＭファイルＴＥＳＴがあれば、その内容を表示する ↓
start /w ft /wc/ isready a:  -- ilist a:.index でもチェックできる ↓
if errorlevel 1 goto notready ↓
↓
start /w ft /wc/ ilist a:test ↓
if errorlevel 1 goto notfound ↓
↓
start /w ft /wc/ gettext a:test test ↓
if exist test type test ↓
goto end ↓

:notready ↓
 echo ドライブの準備ができていません. ↓
 goto end ↓
↓
:notfound ↓
 echoＩＢＭファイルＴＥＳＴがありません. ↓
↓
:end ↓
```

２．21　Ａｎｋ（ａｎ）　アンク

ＡＮＫコードの設定

ホストシステムには、ＪＩＳ８／ＡＳＣＩＩ系のシステムと、ＥＢＣＤＩＣ系のシステムがあります。さらに、ＥＢＣＤＩＣコードには、カタカナ版と英小文字版があります。Ａｎｋ（ａｎ）コマンドは、それらのどれに合わせるかを設定します。

この設定はとても重要です。というのは、ＪＩＳ８／ＡＳＣＩＩ系にするかＥＢＣＤＩＣ系にするかで、単純なＡＮＫ変換にとどまらず、バッファをクリアするコードや、ゾーン形式、パック形式の変換などにも影響を及ぼすからです。

■コマンド形式

コマンド	パラメータ
Ａｎｋ ａｎ	[Jis ｜ Ascii ｜ Ebcdic ｜ EbcdicKana ｜ EbcdicSmall]

Ｊｉｓ　または　Ａｓｃｉｉ	ホストはＪＩＳ８／ＡＳＣＩＩ系
Ｅｂｃｄｉｃ	ホストはＥＢＣＤＩＣ系
ＥｂｃｄｉｃＫａｎａ	ＥＢＣＤＩＣ（カタカナ版）を使う
ＥｂｃｄｉｃＳｍａｌｌ	ＥＢＣＤＩＣ（英小文字版）を使う

■使用例

例）ＥＢＣＤＩＣ（カタカナ版）に設定し、自動的に保存する

ＥＢＣＤＩＣ（カタカナ版）に設定します。そして、修正のかかったコード変換表を自動的に保存するようにします。

```
C:¥>ft ank ebcdickana ; csave .  ↓
```

２．22 Ｋａｎｊｉ（ｋａｎ）　カンジ

漢字変換方式の設定

Ｋａｎｊｉ(ｋａｎ)コマンドは、漢字変換方式を割り当てます。

■コマンド形式

コマンド	パラメータ
Ｋａｎｊｉ ｋａｎ	［漢字変換方式の名前］

Ｋａｎｊｉ(ｋａｎ)コマンドは、パラメータとして漢字変換方式の名前（ＪＥＦ、ＪＩＳ等）を付けて実行します。通常は、

Ｃ：￥＞ｆｔ　ｋａｎｊｉ　ｊｅｆ　；　ｇｅｔｄａｔａ　～ ↓

のようにファイル変換の前で実行し、ホストの漢字コードの体系を指定します。

パラメータを指定せずに実行すると､デフォルトの漢字変換方式を指定したものとみなします。たとえば、

Ｃ：￥＞ｆｔ　ｋａｎｊｉ　；　ｇｅｔｄａｔａ　～ ↓

のように指定した場合は、デフォルトの漢字変換方式でファイル変換を実行します。

■使用例

例１）ＡＮＫをＥＢＣＤＩＣ、漢字をＪＥＦにしてからファイル変換を実行する

あるファイル変換に先立って、ＡＮＫの設定はＥＢＣＤＩＣに、漢字変換方式は富士通のＪＥＦの方式に設定します。

```
C:¥>ft ank ebcdic ; kanji jef ; getdata ･･･（ファイル変換）　↓
```

例２）コード変換表Ｎ．ＣＣＴの漢字変換方式ＩＮＴＮＬ＿Ｅ（内部コード(Ｅ)）を使う

ＮＥＣの内部コード（Ｅ）方式の漢字変換をしたいので、起動時に／Ｃｏｄｅ起動オプションでＮ．ＣＣＴを選択し、さらにＫａｎｊｉ(ｋａｎ)コマンドで漢字変換方式ＩＮＴＮＬ＿Ｅを即時割り当てします。

```
C:¥>ft /cn/ kanji intnl_e ; getdata ～ /map ～ k40 ～ ↓
```

例３）日立対応にカスタマイズする

マルチコマンド機能を使い、ＡＮＫはＥＢＣＤＩＣ（カタカナ版)、漢字変換方式は日立ＫＥＩＳ、ブロック配置は日立モードに一度で設定します。コード変換表は自動的に保存されるようにします。

```
C:¥>ft ank ebcdickana ; kanji keis ; blockmap hitachi ; csave . ↓
```

２．23　ＢｌｏｃｋＭａｐ（ｂｍ）　ブロック・マップ

ブロック配置モードの設定

詳しくは、「■解説」で述べますが、ＩＢＭファイル／ブロックの配置方法には「一般のＩＢＭ形式」のモードと「日立のＩＢＭ形式」のモードがあります。ＢｌｏｃｋＭａｐ(ｂｍ)コマンドで、どちらのモードで動作させるかを指定します。

■コマンド形式

コマンド	パラメータ
ＢｌｏｃｋＭａｐ ｂｍ	［Ｇｅｎｅｒａｌ｜Ｈｉｔａｃｈｉ］

■解説

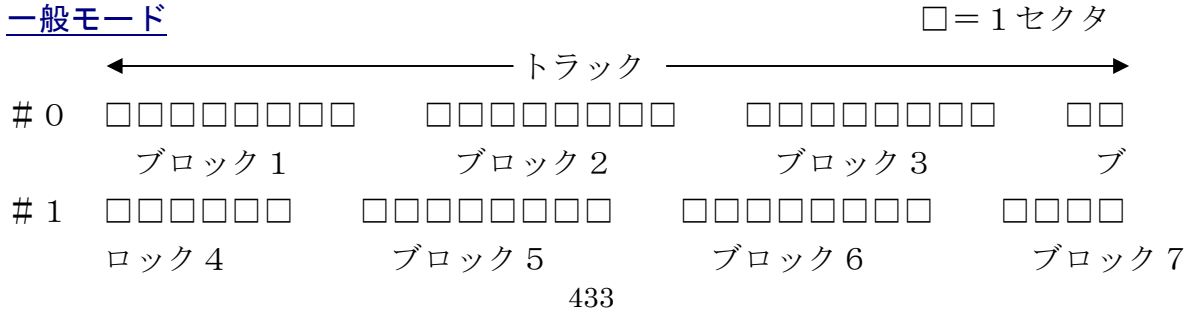
日立のＩＢＭ形式では、ＩＢＭファイル、ブロックの配置に関して、

①ＢＯＥ（領域開始のアドレス）は常にトラック境界（セクタ０１）にする
②ブロックが、２つのトラックにまたがらないようにする

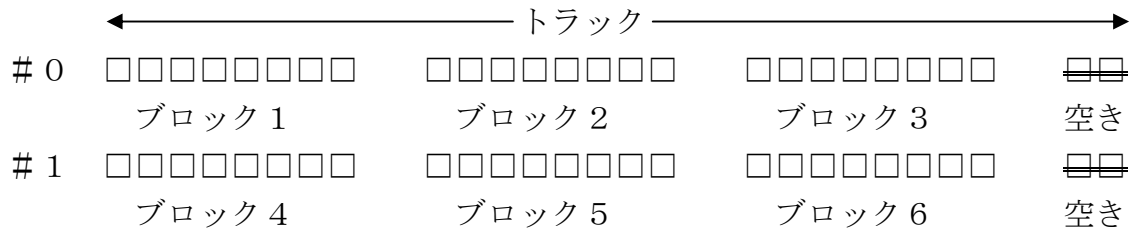
という条件を課しています。①の条件はよく見られるもので日立のシステムに限らないものですが、②の条件によってブロック長によっては（とくに、大きなブロック長のとき）「トラック末尾に１ブロック書き込めるだけのセクタがない」という状況が生じます。このとき日立のＩＢＭ形式では、つぎのトラックにまたがるような書き込みをしないで、トラック末尾を強制的に「空きセクタ」にします。そして、そのブロックはつぎのトラックの先頭セクタから書き込みます。

一般のＩＢＭ形式では、このようなことをせずに、ブロックが２つのトラックにまたがることを許しています。ですから、空きセクタはできません。

図にしてみましょう。条件は、①２ＨＤ－２５６、２６セクタ／トラックのフロッピーディスクを使用、②ブロック長は２０４８バイト、とします。



日立モード



なお、日立のシステムではさらに、

③ＥＯＥ（領域終了アドレス）もトラック境界にする

という条件を課すこともあります。

これらの条件を考慮し、Ｐｕｔ系のコマンドの／ＢｏｅＢｏｕｎｄａｒｙオプションにＳｅｃｔｏｒ指定しても、Ｔｒａｃｋ指定に切り替えられるようにしてあります（Ｓｅｃｔｏｒ指定してもエラーにはなりません）。

■使用例

例１）ブロック配置モードを日立モードにしてからファイル変換を実行する

あるファイル変換に先立って、ブロック配置モードを日立モードに設定します。

```
C:¥>ft blockmap hitachi ; putdata ･･･ （ファイル変換） ↓
```

例２）バッチファイルで日立対応のファイル変換をする

バッチファイルを使うときは、標準提供されるコード変換表Ｈ．ＣＣＴを使えば十分です。

```
@echo off ↓
:: パテントデータ取得 ↓
ft /ch/ vd p: c:¥pat ; gd a:* p:*.pat /map ～ ↓
```

■注意事項

混乱を避ける

１枚のＩＢＭディスクに一般モードのＩＢＭファイルと日立モードのＩＢＭファイルを混在させるのは避けてください。混乱のもとになります。

２．24 Ｃｏｍｍｅｎｔ（ｃｏｍ）　コメント

コメントの変更

Ｃｏｍｍｅｎｔ(ｃｏｍ)コマンドを使って､コード変換表ごとにつけるコメントを自分なりに変更できます。

■コマンド形式

コマンド	パラメータ
Ｃｏｍｍｅｎｔ ｃｏｍ	［コメント文字列］

コメント文字列を入力するには、つぎの点に注意してください。

ＡＮＫ４０文字（漢字２０文字）以内で指定する
セミコロン（；）は使えない（マルチコマンドになってしまう）
全体を引用符（“）でくくることはできない
引用符でくくると、その引用符自体もコメントになってしまう
コメント文字列を省略すると、デフォルトのコメントのままになる

■使用例

例）ほかのコマンドと組み合わせてコード変換表をカスタマイズする

日立製のシステムとのデータ交換用に、コード変換表を

ＡＮＫコードはＥＢＣＤＩＣ
そのカタカナ版を使用
漢字変換方式は日立ＫＥＩＳ
ブロック配置モードは日立モード
コメントは「日立製のシステム用」

とカスタマイズし、インストールディレクトリにＨＩＴＡ．ＣＣＴという名前で保存します。

```
C:¥>ft ank ebcdickana ; kanji keis ; blockmap hitachi ;
    comment日立製のシステム用 ; csave hita ↓
```

２．25　ｃＬｏａｄ（ｃｌ）　シー・ロード

コード変換表の（再）読み込み

ｃＬｏａｄ(ｃｌ)は、現在のコード変換表ファイルの再読み込みや、別のコード変換表ファイルの読み込みをするコマンドです。コード変換表の修正を取り消したり、別のコード変換表に切り替えたりするのに使います。

■コマンド形式

コマンド	パラメータ
ｃＬｏａｄ ｃｌ	［　［ｄ：］[ファイルまでのパス］ファイル名［．拡張子］　］

ｄ：はドライブ名

ｃＬｏａｄ(ｃｌ)コマンドは、パラメータとしてコード変換表ファイルを指定します。パラメータを省略すると、デフォルトのコード変換表が指定されたことになります。

ドライブ名及びパス名を省略した場合は、現在指定されている環境名のフォルダ（出荷状態時はインストールフォルダ）内のコード変換表ファイルを読み込みます。

なお、現在のコード変換表は設定表示バー（ＧＵＩ部）を見ればわかります。

■使用例

例）コード変換表をＩ．ＣＣＴにしてからファイル変換を実行する

あるファイル変換に先立って、コード変換表をＩ．ＣＣＴ（ＩＢＭ方式用）に設定します。

```
C:¥>ft cload i ; getdata ･･･ （ファイル変換） ↓
```

２．26　ｃＳａｖｅ（ｃｓ）　シー・セーブ

コード変換表の書き込み（保存）

ｃＳａｖｅ(ｃｓ)は、現在のメモリ上のコード変換表を元のコード変換表ファイルに保存、または別のコード変換表ファイルとしてディスクに書き込むコマンドです。

■コマンド形式

コマンド	パラメータ
ｃＳａｖｅ ｃｓ	［ｄ：］［ファイルまでのパス］ファイル名［．拡張子］

ｄ：はドライブ名

ｃＳａｖｅ(ｃｓ)コマンドは、パラメータとしてコード変換表ファイルを指定します。パラメータを省略すると、ディスクには書き込まれません。

ドライブ名及びパス名を省略した場合は、現在指定されている環境名のフォルダ（出荷状態時はインストールフォルダ）内のコード変換表ファイルを書き込み対象に指定します。

なお、現在のコード変換表は設定表示バー（ＧＵＩ部）を見ればわかります。

■使用例

例）日立製のシステムとのデータ変換

日立製のシステムとのデータ交換用に、漢字変換方式をＫＥＩＳ、ＡＮＫ変換をＥＢＣＤＩＣ（カタカナ)、ブロック配置モードを日立方式、コメントを「日立用」に設定します。

```
C:¥>ft kanji keis ; ank ek ; blockmap hita ; comment日立用 ; csave H.CCT↓
```

ー参 考ー

他のアプリケーションからの利用

■Visual Basic から利用するには

Ｖｉｓｕａｌ　Ｂａｓｉｃから直接Ｆ＊ＴＲＡＮ２００７を起動して、バッチ処理を実行することができます。その例を添付のサンプルプログラムを使って説明します。

◆注意　----　Ｆ＊ＴＲＡＮが起動しない場合

“CreateProcess()”を使用した場合、Ｆ＊ＴＲＡＮ２００７が起動しない場合があります。その時はサンプルフォルダに格納されている“FT.EXE.manifest”ファイルをＦ＊ＴＲＡＮ２００７のインストールフォルダにコピーしてください。

●サンプルプログラムの読み込み

このサンプルプログラムは、Ｖｉｓｕａｌ　Ｂａｓｉｃのバージョン６以上で動作します。Ｆ＊ＴＲＡＮ２００７をインストールした、Ｂａｓｉｃディレクトリの中につぎの２つのファイルが入っていますので、プロジェクトを開いた後に、フォームモジュールの追加をしてください。

Ｐｒｏｊｅｃｔ１．ｖｂｐ
Ｆｏｒｍ１　　．ｆｒｍ

●サンプルプログラムの内容

このサンプルプログラムは、５６ページのＴＥＳＴ．ＢＡＴの内容をＢａｓｉｃ言語で記述し、Ｆ＊ＴＲＡＮ２００７を非表示モードで起動しています。このサンプルプログラムを基にしてプログラムを作成する場合は、通常、③の部分のみを変更します。

①Formのジェネラルプロシージャのdeclarationに以下の宣言を記述する

```
Private Declare Function SHGetSpecialFolderPath Lib "shell32.dll" _
    Alias "SHGetSpecialFolderPathA" _
    (ByVal hwndOwner As Long, ByVal lpszPath As String, _
     ByVal nFolder As Long, ByVal fCreate As Long) As Long

Private Const CSIDL_PERSONAL = &H5

Private Declare Function CloseHandle Lib "kernel32" _
  (ByVal hObject As Long) As Long

Private Declare Function GetExitCodeProcess Lib "kernel32" _
  (ByVal hProcess As Long, lpExitCode As Long) As Long
```

```
Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal
dwMilliseconds As Long) As Long

Private Const SYNCHRONIZE = &H100000
Private Const INFINITE = &HFFFF ' Infinite timeout

Private Declare Function ShellExecuteEX Lib "shell32.dll" Alias "ShellExecuteEx" (SEI As
SHELLEXECUTEINFO) As Long

Private Type SHELLEXECUTEINFO
    cbSize As Long
    fMask As Long
    hwnd As Long
    lpVerb As String
    lpFile As String
    lpParameters As String
    lpDirectory As String
    nShow As Long
    hInstApp As Long
    lpIDList As Long
    lpClass As String
    hkeyClass As Long
    dwHotKey As Long
    hIcon As Long
    hProcess As Long
End Type

Private Const SEE_MASK_NOCLOSEPROCESS = &H40
Private Const SW_SHOWNORMAL = 1
```

②Form内のジェネラルプロシージャに以下の関数を記述する

```
Public Function ProcessExec(ByRef command As String, ByRef param As String, ByRef path
As String) As Long

    Dim sdtSEXI As SHELLEXECUTEINFO
    Dim ret As Long

    With sdtSEXI
        .cbSize = Len(sdtSEXI)
        .fMask = SEE_MASK_NOCLOSEPROCESS
        .hwnd = Me.hwnd
        .lpVerb = "open"
        .lpFile = path & "¥" & command
        .lpParameters = param
        .lpDirectory = vbNullString
        .nShow = SW_SHOWNORMAL
        .hInstApp = 0
        .lpIDList = 0
    End With

    ret = ShellExecuteEX(sdtSEXI)

    ProcessExec = sdtSEXI.hProcess
End Function

Public Function ProcessWait(hProcess As Long) As Long
    Dim ExitCode As Long
    Dim ret As Long

    ret = WaitForSingleObject(hProcess, INFINITE)
    ret = GetExitCodeProcess(hProcess, ExitCode)

    ret = CloseHandle(hProcess)

    ProcessWait = ExitCode
End Function
```

③Form内の処理を割り当てるオブジェクトに以下のプログラムを記述する

```
' ドライブA:にIBMファイル"TEST"があればその内容を表示する
Private Sub Command1_Click()

    On Error GoTo Err_Command1_Click
    Dim IDProcess As Long, ExitCode As Long
    Dim path As String

' カレントディレクトリをF*TRAN2007のインストールディレクトリに設定する
    path = "c:¥Program Files¥FujitsuBSC¥ftran2007" 'インストールディレクトリ＜確認！＞
    ChDrive "c"                 'インストールドライブ     ＜確認！＞
    ChDir path

' ドライブA:にIBMファイル"TEST"があればその内容を表示する
    IDProcess = ProcessExec("ft.exe", "/nwd /wc/ isready a:", path)
    ExitCode = ProcessWait(IDProcess)
    If ExitCode <> 0 Then
        MsgBox "ドライブの準備ができていません。", vbExclamation
        GoTo Exit_Command1_Click
    End If

    IDProcess = ProcessExec("ft.exe", "/nwd /wc/ ilist a:TEST", path)
    ExitCode = ProcessWait(IDProcess)
    If ExitCode <> 0 Then
        MsgBox "IBMファイルTESTがありません。", vbExclamation
        GoTo Exit_Command1_Click
    End If

    IDProcess = ProcessExec("ft.exe", "/nwd /wc/ gettext a:TEST test", path)
    ExitCode = ProcessWait(IDProcess)
    Open "test" For Input As #1
    Close #1

    Call Shell("notepad.exe test", vbNormalFocus)     ' メモ帳で変換結果を表示

 Exit_Command1_Click:
Exit Sub
```

```
Err_Command1_Click:
    MsgBox Err.Description
    Resume Exit_Command1_Click
End Sub

Private Sub Command2_Click()
    End
End Sub
```

■Visual C++から利用するには

Ｖｉｓｕａｌ　Ｃ＋＋から直接Ｆ＊ＴＲＡＮ２００７を起動して、バッチ処理を実行することができます。Ｆ＊ＴＲＡＮ２００７をインストールしたＣディレクトリの中に、つぎのファイルがあります。

Ｓｍｐｌｃｏｄｅ．ｔｘｔ　　サンプルプログラムのソース

以下、サンプルプログラムのソースの内容です。

```
void CFtSampleDlg::OnExec()
{
    UINT  lFunc(LPVOID);

    AfxBeginThread(lFunc,0,0,0,0,0);
}
```

```
UINT lFunc(LPVOID pParam)
{
    long            IdProcess;
    unsigned long   exitCode;

    // カレントディレクトリをF*TRAN2007のインストールディレクトリに設定する
    _chdir("c:¥¥Program Files¥¥FujitsubSC¥¥ftran2007¥¥");    // ＜確認！＞

    // ドライブA:にIBMファイル"TEST"があれば変換してその内容を表示する
    IdProcess = ProcessExec("ft.exe /nwd /wc/ isready a:");
    exitCode = ProcessWait(IdProcess);
    if(exitCode != 0) {
        MessageBox(NULL,"ドライブの準備ができていません。","F*TRAN2007サンプル",
MB_OK|MB_ICONEXCLAMATION);
        return(1);
    }

    IdProcess = ProcessExec("ft.exe /nwd /wc/ ilist a:TEST");
    exitCode = ProcessWait(IdProcess);
    if(exitCode != 0) {
        MessageBox(NULL,"IBMファイルTESTがありません。","F*TRAN2007サンプル",
MB_OK|MB_ICONEXCLAMATION);
        return(1);
    }

    IdProcess = ProcessExec("ft.exe /nwd /wc/ gettext a:TEST test");
    exitCode = ProcessWait(IdProcess);
    if(exitCode != 0) {
        MessageBox(NULL,"変換に失敗しました。","F*TRAN2007サンプル",
MB_OK|MB_ICONEXCLAMATION);
        return(1);
    }

    WinExec("command.exe /K type test",SW_SHOW); // Windows2000,XPではcmd.com

    return(0);
}
```

```
long ProcessExec(char *command)
{
    long                ret;
    STARTUPINFO         stInfo;
    PROCESS_INFORMATION procInfo;
    char                path[256];
    char                cmdline[512];

    memset(path,0x00,sizeof(path));
    memset(cmdline,0x00,sizeof(cmdline));

    strcpy(path," c:¥¥Program Files¥¥FujitsubSC¥¥ftran2007¥¥");    // ＜確認！＞
    sprintf(cmdline,"%s¥¥%s",path,command);

    stInfo.cb = sizeof(STARTUPINFO);
    stInfo.lpReserved = NULL;
    stInfo.lpDesktop = NULL;
    stInfo.lpTitle = NULL;
    stInfo.dwFlags = STARTF_USESHOWWINDOW;
    stInfo.cbReserved2 = 0;
    stInfo.lpReserved2 = NULL;
    stInfo.wShowWindow = SW_SHOWNORMAL;
    ret = CreateProcess(NULL,cmdline,NULL,NULL,FALSE,0,NULL,path,&stInfo,&procInfo);

    return(procInfo.dwProcessId);
}

unsigned long ProcessWait(long id)
{
    unsigned long  exitCode;
    HANDLE         hProcess;
    long           ret;

    hProcess = OpenProcess(PROCESS_QUERY_INFORMATION,1,id);
    do {
        ret = GetExitCodeProcess(hProcess,&exitCode);
    } while(exitCode == STILL_ACTIVE);

    return(exitCode);
}
```

Ｆ＊ＴＲＡＮ２００７ Ｖ２．０　　操作説明書／コマンド編

２０１０年　４月　　第１版発行

編集・著作　　株式会社　富士通ビー・エス・シー
所　在　地　　〒135-8300　東京都港区台場 2-3-1 トレードピアお台場
