F＊TRAN＋

V3.0

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＊ＴＲＡＮ＋（エフトラン・プラス）は、汎用機／オフコン／Ｕｎｉｘなどのホストのファイル転送データと、パソコンの標準であるＷｉｎｄｏｗｓファイルとのデータ交換をする汎用性の高いファイル変換ユーティリティです。Ｗｉｎｄｏｗｓファイル間のデータ変換もできます。

Ｗｉｎｄｏｗｓ９８／９５　または　ＷｉｎｄｏｗｓＮＴ／２０００上の３２ビットアプリケーションとして動作します。

ソースプログラム、バイナリファイル、ランダムファイル、プリント形式ファイルなどを変換する基本機能に加え、コンマ区切り（ＣＳＶ）形式対応など、市販ソフトとのデータ交換に適した強力なデータ加工・編集機能を備えています。また、各種漢字コードに対応し、拡張漢字にも本格対応しています。さらに、ＣＯＢＯＬの数値項目等（ゾーン形式、パック形式、２進形式、ＢＣＤ形式）にも対応しています。

Ｆ＊ＴＲＡＮ＋は、さまざまなホストとパソコンの連携利用を強力に支援します。その高い汎用性、高性能、高機能が有効に活用されることを願ってやみません。

Ｆ＊ＴＲＡＮ＋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。コマンド編（本書）の構成はつぎのとおりです。

第１章　操作の基礎
　　操作方法についての基礎的な知識を説明しています。
第２章　コマンド型の実行
　　コマンド別に詳細な機能と操作を説明しています。

－参考－　他のアプリケーションからの利用
　　Visual Basic、Access、Visual C++からの利用法について説明しています。

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、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＊ＴＲＡＮ＋をコマンド型で実行する場合は、

Ｗｉｎｄｏｗｓ９８／９５のＭＳ－ＤＯＳプロンプト
ＷｉｎｄｏｗｓＮＴ／２０００のコマンドプロンプト　　などを利用しますが、

Ｆ＊ＴＲＡＮ＋のディレクトリ以外からＦ＊ＴＲＡＮ＋のコマンドを実行する場合は、下記のＰＡＴＨコマンドを使用します。

■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＊ＴＲＡＮ＋のインストールディレクトリにパス（コマンド検索パス）を通します。これは、Ｆ＊ＴＲＡＮ＋を

どのドライブ、どのディレクトリからでも起動できる

ようにする重要な作業です。

●指定

つぎのように指定します。

Ｃ：￥＞ＰＡＴＨ　％ＰＡＴＨ％；Ｃ：￥ＦＴＲＡＮＰ　↓

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：￥ＦＴＲＡＮＰ　↓

●確認

Ｆ＊ＴＲＡＮ＋をインストールしたディレクトリ以外から

Ｃ：￥＞ＦＰ　↓

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＊ＴＲＡＮ＋のコマンド全体を紹介します。

●コマンドの構成

Ｆ＊ＴＲＡＮ＋の各機能はＦＰ．ＥＸＥ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

ＦＰ．ＥＸＥ　コマンド名		短縮名	機　　能
フ	ＧｅｔＴｅｘｔ	ｇｔ	ホスト→Ｗｉｎテキストファイル変換
ァ	ＧｅｔＤａｔａ	ｇｄ	ホスト→Ｗｉｎデータファイル変換
イ	ＧｅｔＲａｎｄ	ｇｒ	ホスト→Ｗｉｎランダムファイル変換
ル	ＰｕｔＴｅｘｔ	ｐｔ	Ｗｉｎ→ホストテキストファイル変換
変	ＰｕｔＤａｔａ	ｐｄ	Ｗｉｎ→ホストデータファイル変換
換	ＰｕｔＲａｎｄ	ｐｒ	Ｗｉｎ→ホストランダムファイル変換
	ＶｉｒＤｒｉｖｅ	ｖｄ	仮想ドライブの設定
セ	Ａｎｋ	ａｎ	ＡＮＫコードの設定
ッ	Ｋａｎｊｉ	ｋａｎ	漢字変換方式の設定
ト	Ｃｏｍｍｅｎｔ	ｃｏｍ	コメントの変更
Ｕ	ｃＬｏａｄ	ｃｌ	コード変換表の（再）読み込み
Ｐ	ｃＳａｖｅ	ｃｓ	コード変換表の書き込み（保存）

◆注意　----　Ｆ＊ＴＲＡＮⅢにあり、Ｆ＊ＴＲＡＮ＋では機能を持たないコマンド

Ｆ＊ＴＲＡＮⅢにある以下のコマンドは、Ｆ＊ＴＲＡＮ＋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）
ＩＢＭフロッピー関連のコマンド	ｉＬｉｓｔ（ＩＢＭファイル名・属性一覧） ｉＤｅｌｅｔｅ（ＩＢＭファイルの削除） ｉＦｏｒｍａｔ（ＩＢＭディスクの初期化） ｉＥｄｉｔ（ＩＢＭディスクエディタ） ＢｌｏｃｋＭａｐ（ブロック配置モード設定）
ＧＵＩ部でのみ機能を実現したコマンド	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）

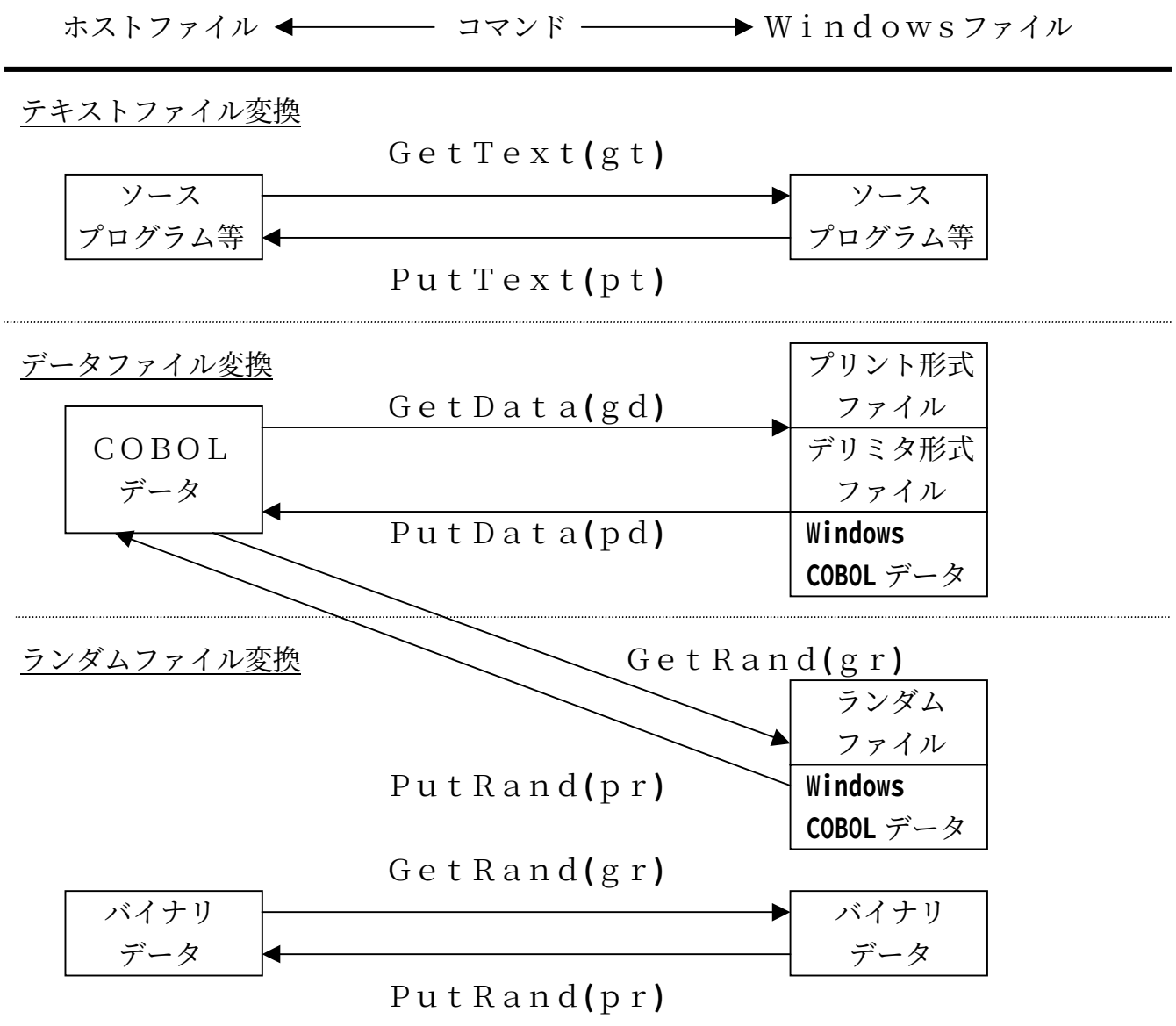
１．３　３つのファイル変換機能

Ｆ＊ＴＲＡＮ＋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。

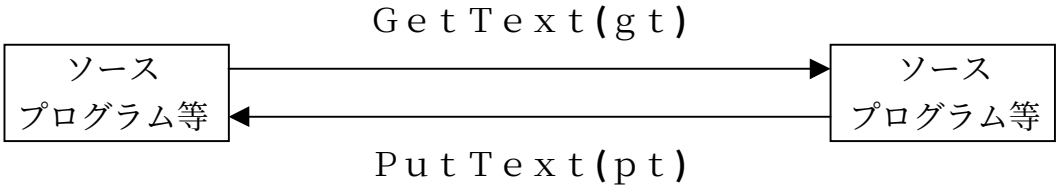
●典型的なデータ形式と使用するコマンド（ホストが汎用機・オフコンの場合）



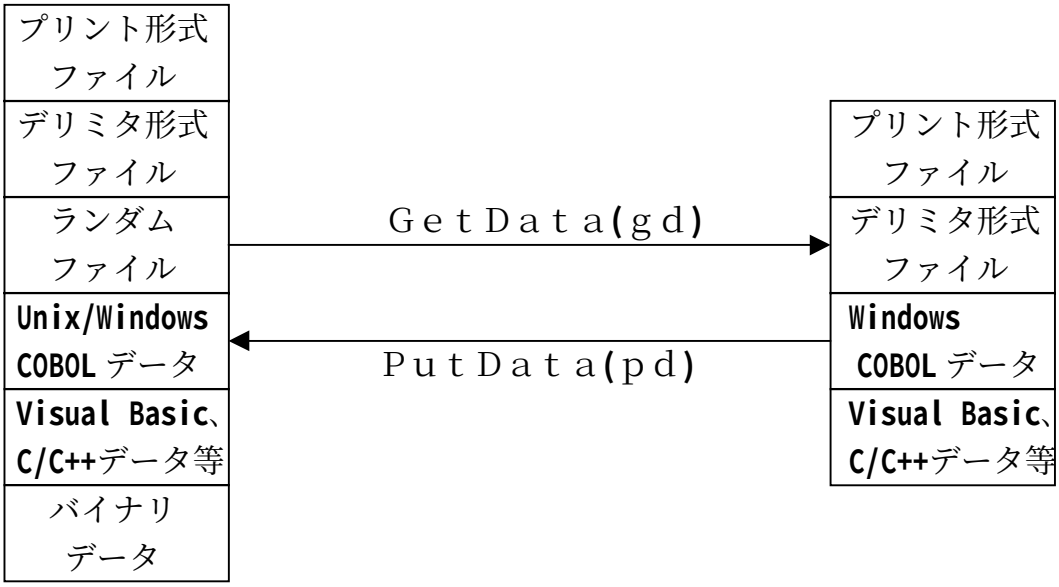
●典型的なデータ形式と使用するコマンド（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

ホストファイル ◀――― コマンド ―――▶ Ｗｉｎｄｏｗｓファイル

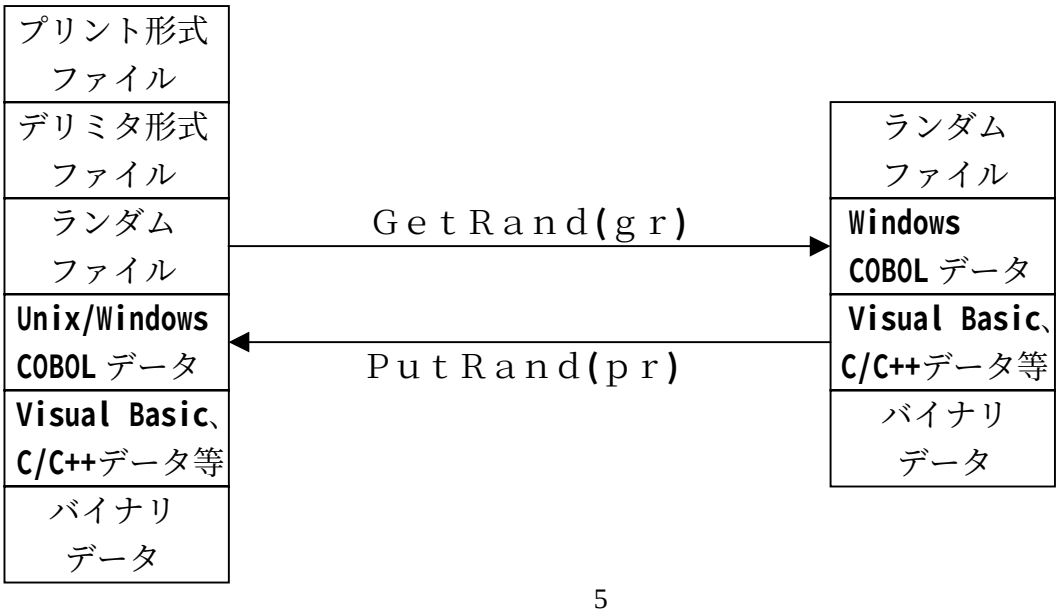
テキストファイル変換



データファイル変換



ランダムファイル変換



１．４　補助機能

Ｆ＊ＴＲＡＮ＋は、つぎの補助機能を提供しています。

●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　----　このコマンドは、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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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。

●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＊ＴＲＡＮ＋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。

つぎの４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ＣＯＢＯＬベンダの設定

ホスト上で使用しているＣＯＢＯＬがどこのベンダかを設定します。この設定は、Ｕｎｉｘ、ＷｉｎｄｏｗｓのＣＯＢＯＬデータと、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬデータの変換を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。

１．６ マルチコマンド

Ｆ＊ＴＲＡＮ＋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に似ている機能です。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の中にある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Ｃ：のホストファイルＥＡＲＴＨ、ＪＵＰＩＴＥＲにデータファイル変換します。

```
C:\>fp pd c:\planet\earth.dat c:* /hs64 ; pd c:\planet\jupiter.dat c:* /hs64 ↓
```

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:\>fp vd x: c:\planet ; pd x:earth.dat c:* /hs64 ; pd x:jupiter.dat c:* /hs64 ↓
          ▲── 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＊ＴＲＡＮ＋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「Ｗｉｎ→ホストデータファイル変換」というコメントをつける

　Ｗｉｎ→ホストデータファイル変換を実行するバッチファイルにコメントをつけます。

＜ＨＥＮＫＡＮ．ＢＡＴ＞

```
fp pd c:\planet\earth.dat c:* /hs64     -- Ｗｉｎ→ホストデータファイル変換 ↓
```

１．８　ファイルの指定

●指定形式

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＊ＴＲＡＮ＋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、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。

●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、ＶｉｄＤｒｉｖｅ(ｖｄ)コマンドで指定していましたが、Ｆ＊ＴＲＡＮ＋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ＶｉｄＤｒｉｖｅ(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、入力側のファイル名を指定する場合、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。

出力側のファイル名を指定する場合、基本ファイル名と拡張子にはふつうの基本ファイル名や拡張子以外に、＊も指定できます。＊は「入力側の基本ファイル名や拡張子を引き継げ」という意味です。ふつうは、

ｄ：＊．ｄａｔ　のように、拡張子を指定します。

こうすると、指定ドライブに

元の基本ファイル名．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

ファイル名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

ＤＯＳのアプリケーションであるＦ＊ＴＲＡＮⅢではロングファイル名の指定はできませんでしたが、Ｗｉｎｄｏｗｓの３２ビットアプリケーションであるＦ＊ＴＲＡＮ＋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、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　オプションとキーワードの指定

Ｆ＊ＴＲＡＮ＋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、ホスト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

１．10　式の指定

Ｆ＊ＴＲＡＮ＋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
ｘ（小文字のｘ）　または　＊	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。ｘ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６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）
~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）

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
２５６ｘ２６	＊	＄％３
１０２４ｘ４	＊－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。

１．11 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。

［u｜s｜b］［0］m［. n ］　（［ ］は省略可の意味）

- n：小数部桁数（１０進数　n＝１～１８）
  - ．nを省略すると整数（小数部なし）
- m：整数部桁数（１０進数　m＝１～１８）
- 0：０をつけると、前ゼロをつけたままにする
  - ０を省略すると、前ゼロをつけない
- u　符号なし（**u:unsigned**）
- s　符号つき（**s:signed**）
- b　符号なしＢＣＤ形式（**b:BCD**）
- 省略すると、ｕ指定（符号なし数値）

●ＣＯＢＯＬをまねた

Ｆ＊ＴＲＡＮ＋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

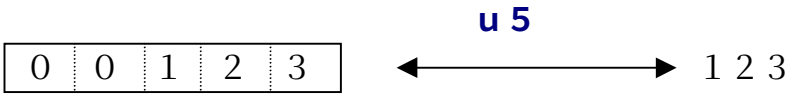
```
PIC   S9(4)V9(1)
```

のように指定します。

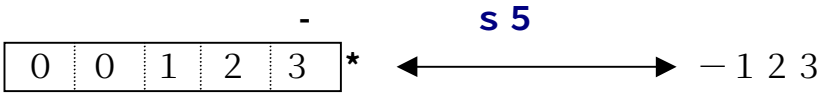
Ｆ＊ＴＲＡＮ＋のピクチャではこれを、　**ｓ４．１**　と指定します。

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

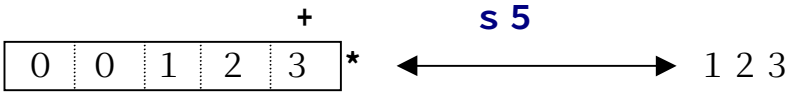


例２）符号つきの、負の整数



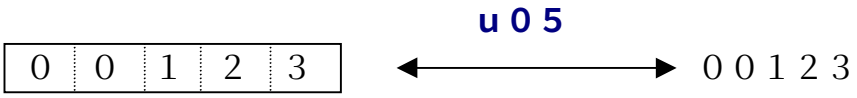
*）数字の上の“ - ”は、負の符号を表す

例３）符号つきの、正の整数

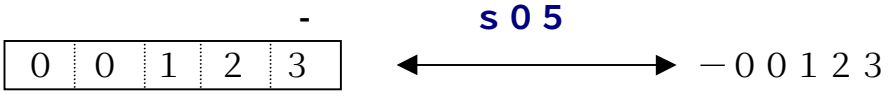


*）数字の上の“ + ”は、正の符号を表す

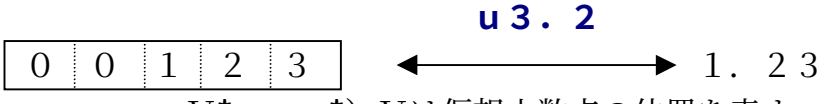
例４）符号なし整数、前ゼロをつけたままにする



例５）符号つき整数、前ゼロをつけたままにする

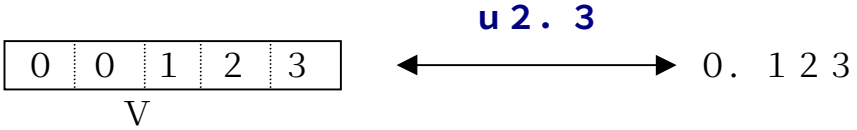


例６）小数部（小数点）がある／その１

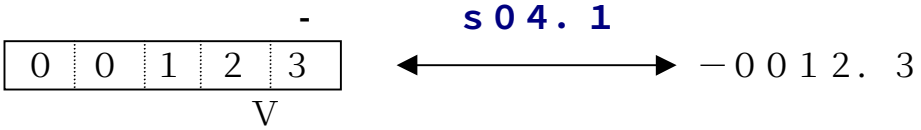


*）Ｖは仮想小数点の位置を表す

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



１．12　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、
  - ホスト側の場合、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－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

１．13　日付データの指定

●日付マスク

Ｆ＊ＴＲＡＮ＋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 *1	1998/12/31 1998-12-31 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 *1	12/31/1998 12-31-1998 12.31.1998
mm－ｄｄ－ｙｙ	12/31/98 12-31-98 12.31.98
mmｄｄｙｙｙｙ	12311998
mmｄｄｙｙ	123198
ｄｄ－mm－ｙｙｙｙ *1	31/12/1998 31-12-1998 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 *1	1998/12 1998-12 1998.12
ｙｙ－mm	98/12 98-12 98.12
ｎｙｙ－mm	H10/12 H10-12 H10.12
ｙｙｙｙmm	199812
ｙｙmm	9812
mm－ｙｙｙｙ *1	12/1998 12-1998 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）の交換を行う場合、Ｆ＊ＴＲＡＮ＋ではウインドウ方式とシフト方式の概念を採用しています。

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。

１．14 パラメータファイルの概要

Ｆ＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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、ふつうインストールディレクトリに置きます。本体ＦＰ．ＥＸＥと同じディレクトリになります。そうすれば、管理・運用が簡単になるように設計されています。

●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：￥＞ｆｐ　［～］＋＋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＊ＴＲＡＮ＋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。

●Ｆ＊ＴＲＡＮⅢのパラメータファイルとの比較

Ｆ＊ＴＲＡＮ＋では、基本的にＦ＊ＴＲＡＮⅢのパラメータをそのまま使用することができるだけでなく、パラメータバッファの大きさが６４Ｋバイトに拡張されていますので、極めて大きなパラメータファイルの記述が可能になっています。

１．15 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、インストールディレクトリを表します。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ＦＰのあとならどこでも参照可能です。

```
Ｃ：￥＞ｆｐ　・・・　＋＋～ ↓
```

この場合、ＦＰ＋＋～のようにＦＰ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。

１．16 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という名前のホストファイルがあり、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	衛生数（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交換　ホスト→Ｗｉｎプリント形式 ↓
fp gd c:planet c:*.pr /hs64 /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交換　ホスト→Ｗｉｎプリント形式 ↓
fp gd c:planet c:*.pr ++pgetpr.p ↓
```

そして、ＰＧＥＴＰＲ．Ｐ（その１）は、つぎのようになります。

```
/hs64 /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。

```
/hostsize 64 ↓
/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　ホスト→Ｗｉｎプリント形式（For GetData）　↓
/hostsize 64              -- ホストレコード長 ↓
/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      -- 衛生数（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 ↓
```

１．17 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、各コマンドに渡されるかを説明します。

●不要部分のスペース化とスペース圧縮

Ｆ＊ＴＲＡＮ＋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「６４Ｋバイト」の大きさの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　ホスト→Ｗｉｎプリント形式（For GetData）　↓
/hostsize 64              -- ホストレコード長 ↓
/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      -- 衛生数（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。

```
/hostsize 64 /map ank 2 kanji 8 ank 10 ank 9 packdisp u4.3 zonedisp u2
 zonedisp s2.1 kanji 20
```

第２章

コマンド型の実行

２．１　ＦＰ．ＥＸＥ　起動と終了

■Ｆ＊ＴＲＡＮ＋（コマンド行方式）の動作環境

Ｆ＊ＴＲＡＮ＋をコマンド行方式で実行するには、

●Ｗｉｎｄｏｗｓ９８／９５　ＭＳ－ＤＯＳプロンプト　または
**　ＷｉｎｄｏｗｓＮＴ／２０００　コマンドプロンプト　からコマンドを入力**
●Ｆ＊ＴＲＡＮ＋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、Access Basic など）からの利用

などがあげられます。

■Ｆ＊ＴＲＡＮ＋（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［ＡＩＴ］］ＦＰ［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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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　ＦＰ　コマンド　～　でなければなりません。ＳＴＡＲＴ／Ｗを指定すると、Ｆ＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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９つがあります。

／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
／ＡｆｔｅｒＷａｉｔ	処理後の待ち時間（秒単位）を指定する
／Ｔｉｔｌｅ	実行ウインドウ等のタイトルを変更する

●Ｆ＊ＴＲＡＮⅢにあり、Ｆ＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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

／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ドライブ？：を指定したとみなし、**インストールディレクトリからコード変換表を読み込みます**。これら、？：や@：はＦ＊ＴＲＡＮ＋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。これ以外のコード変換表を作って使用するときは、ふつう１～３文字ぐらいの短いファイル名にします。

拡張子を省略すると．ＣＣＴとみなします。それ以外でもよいのですが、混乱のもとですから「．ＣＣＴ以外は使うべからず」ぐらいに思ってください。

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：￥＞ｆｐ　コマンド　パラメータ ↓　　デフォルトのＣＣＴを選択
Ｃ：￥＞ｆｐ　／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：￥＞ｆｐ　／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：￥＞ｆｐ　／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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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。そんなときに、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。そんな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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋の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、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、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＊ＴＲＡＮ＋の表示はでませんので、他のアプリケーションからＦ＊ＴＲＡＮ＋を隠して使いたい場合等に指定します。ただし、エラーダイアログ等は表示されます。

／ＮｏＷｉｎｄｏｗＤｉｐｌａｙオプションを指定した場合は、自動的に／ＷｉｎｄｏｗＣｌｏｓｅモードになり、“処理中の実行ウインドウを表示せず、処理後、自動的に実行ウインドウを閉じる”指定になります。／ＮｏＷｉｎｄｏｗＣｌｏｓｅオプションが指定されていても、／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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋を隠して実行する他の方法

／ＮｏＷｉｎｄｏｗＤｉｓｐｌａｙオプション同様にＦ＊ＴＲＡＮ＋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　ＦＰ　／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と、エラーダイアログ等に対処できないことです。

◆参考　----　Ｆ＊ＴＲＡＮ＋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　ＦＰ　／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になります。ただし、エラーダイアログ等に対処できませんので、注意してください。

／ＡｆｔｅｒＷａｉｔ ---- 処理終の待ち時間（秒単位）を指定する

```
／ＡｆｔｅｒＷａｉｔ　秒数（０～３２７６７）
／ＡｆｔｅｒＷａｉｔ　０
```

Ｆ＊ＴＲＡＮ＋のコマンドを実行し、Ｆ＊ＴＲＡＮ＋が終了する前の待ち時間を秒数で指定できます。通常は、／ＷｉｎｄｏｗＣｌｏｓｅオプションとともに指定します。

この指定が必要になるのは、データ変換の対象となるファイルのサイズが小さすぎて、アプリケーションが戻り値を取得する前にＦ＊ＴＲＡＮ＋の処理が終了してしまう場合です。戻り値が取得できない現象が発生したら、

／ＡＷ　５

のように数秒のウエイト指定をするとよいでしょう。

省略値は、０秒ウエイト（待ち時間なし）です。

／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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋”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＊ＴＲＡＮ＋”として出力されます。

●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、

／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**
／ＮｏＤｉｓｋＣｈａｎｇｅ　　ディスク交換はしない
／ＮｏＷｉｎｄｏｗＣｌｏｓｅ　　処理後、自動的に実行ウインドウを閉じない
／ＷｉｎｄｏｗＤｉｓｐｌａｙ　　処理中の実行ウインドウを表示する
／ＡｆｔｅｒＷａｉｔ　０　　処理後の待ち時間なし（０秒）にする
／Ｔｉｔｌｅ　　実行ウインドウ等のタイトルを変更しない

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Ｃ：のあるホストファイルをドライブＣ：のＷｉｎｄｏｗｓファイルに変換するなら、たとえばＧｅｔＤａｔａ(ｇｄ)コマンドを、つぎのように使うとします。

```
C：¥>fp　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、

```
fp　／dc／　getdata　～　↓
fp　／dc　getdata　～　↓
fp　／dc／getdata　～　↓
```

の３つの指定のうち正しいのはどれでしょうか？

```
fp　／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。

```
fp　／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。

```
fp　／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：¥>fp ／dc／cn／ getdata ～ ↓
```

のように指定します。

■Ｆ＊ＴＲＡＮ＋（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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋は終了時に終了コードを返します。

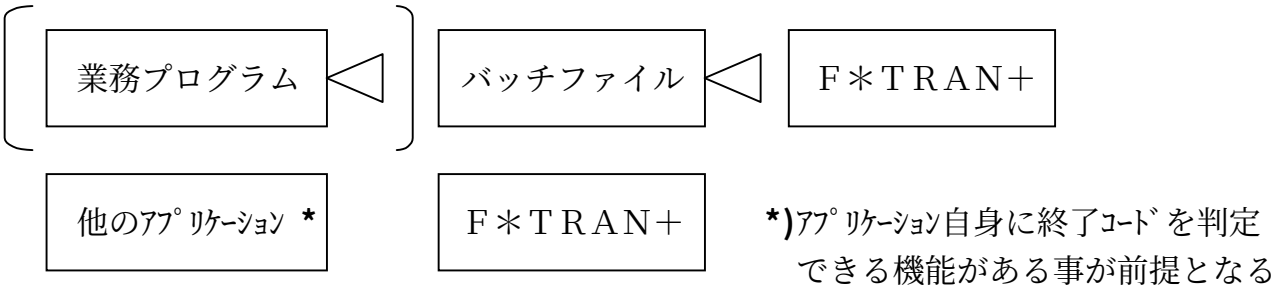
０は正常
１以上はエラー

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＊ＴＲＡＮ＋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 fp ～
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、

```
start /w fp 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）変換が正常であれば、その内容を表示する

バッチファイルで変換が正常に行われたがどうかを判定します。正常であればその内容をコンソールに表示し、正常でなければエラーメッセージを表示します。

＜ＴＥＳＴ．ＢＡＴ＞

```
@echo off ↓
:: 変換が正常に終了したら、その内容を表示する ↓
start /w fp /wc/ getdata planet *.txt ++pngetprn.p ↓
if errorlevel 1 goto onerr ↓
if exist planet.txt type planet.txt ↓
goto end ↓

:onerr ↓
 echo 変換中にエラーが発生しました. ↓
↓
: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（ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイルＦＯＯに変換します。漢字がないテキストファイルを変換します。

```
Ｃ：￥＞ｆｐ　ｐｕｔｔｅｘｔ　ｃ：ｆｏｏ．ｔｘｔ　ｃ：＊ ↓
```

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Ｃ：のホスト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＊ＴＲＡＮ＋を終了させるために／ＷｉｎｄｏｗＣｌｏｓｅオプションを指定します。

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

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

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Ｃ：のホストファイルＦＯＯに変換します。漢字を含むテキストファイルを変換します。

```
Ｃ：￥＞ｆｐ　ｐｕｔｔｅｘｔ　ｃ：ｆｏｏ．ｔｘｔ　ｃ：＊ ↓
```

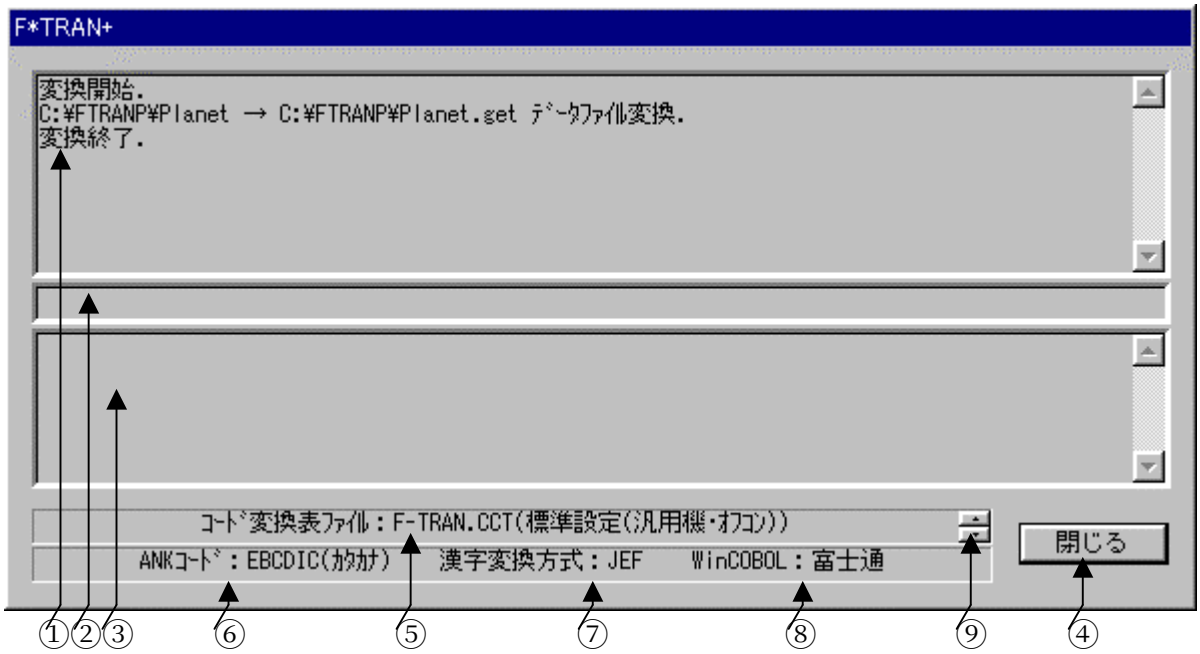
例２）起動オプションつきの、コマンド行方式での起動

ドライブＣ：のすべてのＷｉｎｄｏｗｓファイルを、ドライブＣ：のホストファイルにテキストファイル変換します。コマンド実行後の実行ウインドウを自動的に閉じてＦ＊ＴＲＡＮ＋を終了させるために／ＷｉｎｄｏｗＣｌｏｓｅオプションを指定します。

```
Ｃ：￥＞ｆｐ　／ｗｃ／　ｐｕｔｔｅｘｔ　ｃ：＊．＊　ｃ：＊ ↓
　　（／ＷｉｎｄｏｗＣｌｏｓｅ）
```

２．２　実行ウインドウ

Ｆ＊ＴＲＡＮ＋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。
⑥ホストが汎用機・オフコンの場合、ＡＮＫ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情報です。
⑦漢字変換方式の設定情報（**JEF** 等）を表示しています。漢字変換の基準となる重要な情報です。
⑧Ｗｉｎｄｏｗｓ　ＣＯＢＯＬベンダの設定情報（なし、富士通、日立、**NEC**、ﾒﾗﾝﾄ、**Acucorp**）を表示しています。ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合は、ホストＣＯＢＯＬベンタの設定情報（なし、富士通、日立、**NEC**、ﾒﾗﾝﾄ、**Acucorp**）も表示されます。
⑨コード変換表ファイルの情報表示と、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。

◆参考　----　コマンド実行後の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（ウインドウクローズ）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（ノーウイ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　　ホスト→Ｗｉｎテキストファイル変換
ＧｅｔＤａｔａ(ｇｄ)コマンド　　ホスト→Ｗｉｎデータファイル変換
ＧｅｔＲａｎｄ(ｇｒ)コマンド　　ホスト→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、ホストファイルを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	［ホストファイル　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

●ホストファイル指定

［ｄ：］［パス名指定］基本ファイル名［．拡張子］

ｄ：はドライブ名

入力側のホスト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
ホスト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。

●Ｗｉｎｄｏｗｓファイル指定

［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＊を指定します。＊を指定するとホストファイルの基本ファイル名がＷｉｎｄｏｗｓファイルの基本ファイル名になります。

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。拡張子に＊を指定するとホストファイルの拡張子がＷｉｎｄｏｗｓファイルの拡張子になります。

まとめると、基本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

／ＨｏｓｔＳｉｚｅ（／ＩＳｉｚｅ）	ホストファイルのレコード長を指定する
／ＨｏｓｔＴａｂＥｘｐａｎｄ	タブ拡張する
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＨｏｓｔＥＯＦ	ＥＯＦコードを検査する
／ＨｏｓｔＮｏＥＯＦ	ＥＯＦコードを検査しない
／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
／ＥＯＦ	ＥＯＦコードをつける
／ＮｏＥＯＦ	ＥＯＦコードをつけない

このうち、／ＨｏｓｔＳｉｚｅオプションがとくに重要です。

ＧｅｔＴｅｘｔ(ｇｔ)コマンド固有のオプション

／Ｃｏｄｅ	ＡＮＫ変換か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

／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＨｏｓｔＳＱｕｅｅｚｅ	空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	空行を無視しない
／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

ＧｅｔＲａｎｄ(ｇｒ)コマンド固有のオプション

／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＨｏｓｔＳＱｕｅｅｚｅ	空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	空行を無視しない
／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

以下、Ｇｅｔ系コマンドに共通のオプションを説明します。

／ＨｏｓｔＳｉｚｅ ---- ホストファイルがランダムファイル（固定長）の場合、
（／ＩＳｉｚｅ）　　レコード長を指定する

／ＨｏｓｔＳｉｚｅ　レコード長

ランダムファイル（固定長）のレコード長を１～３２７６７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。

ＧｅｔＴｅｘｔ(ｇｔ)コマンド　　８０バイト
ＧｅｔＤａｔａ(ｇｄ)コマンド　２５６バイト
ＧｅｔＲａｎｄ(ｇｒ)コマンド　２５６バイト

ホストファイルがランダムファイルの場合、レコード長の指定が間違っていると正しいデータ変換が行われませんので、このオプションの指定は極めて重要です。ホストが汎用機・オフコンの場合、この指定が特に重要になります。
ホストファイルがテキスト／データファイル（可変長）の場合は、この指定は意味を持たなくなり、指定しても無視されます。

／ＨｏｓｔＴａｂＥｘｐａｎｄ -------- タブ拡張する
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ ---- タブ拡張しない

／ＨｏｓｔＴａｂＥｘｐａｎｄ〔タブ間隔 8〕 ／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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。

／ＨｏｓｔＴａｂＥｘｐａｎｄ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。
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄオプションを指定すればタブ拡張は行いません。

／ＨｏｓｔＥＯＦ -------- ＥＯＦコードを検査する
／ＨｏｓｔＮｏＥＯＦ ---- ＥＯＦコードを検査しない

／ＨｏｓｔＥＯＦ ／ＨｏｓｔＮｏＥＯＦ

ホストがＷｉｎｄｏｗｓの場合に、ＥＯＦコード（１ＡＨ）を検査するか否かを指定します。

／ＨｏｓｔＥＯＦ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。

／ＨｏｓｔＮｏＥＯＦ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。

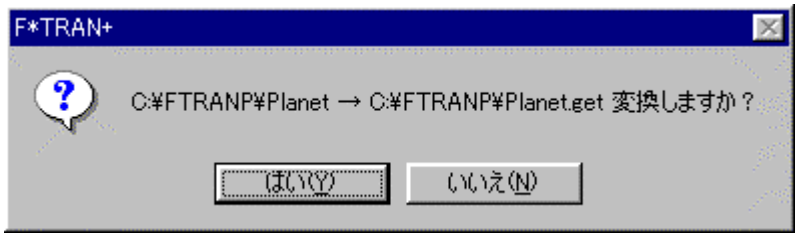
／Ｑｕｅｒｙ -------- 変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動変換する

／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＊ＴＲＡＮ＋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。

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

／ＥＯＦ ／ＮｏＥＯＦ

ホストがＷｉｎｄｏｗｓの場合に、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。

■使用例（ホストが汎用機・オフコンの場合）

例１）一番単純な変換をする

ドライブＣ：のホストファイルＳＡＭＰＬＥを、ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴに変換します。

```
C:\>fp gettext c:sample c:*.txt /hs80 ↓     (/HostSize 80)
```

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２）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Ｃ：にはホスト形式で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:\>fp getext c:* c:*.cbl /hs80 ↓          (/HostSize 80)
```

例３）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Ｃ：にはホスト形式で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:\>fp gettext c:* c:*.for /hs80 /q ↓       (/HostSize 80 /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４）別名をつける

ドライブＣ：のホストファイルＤＡＴＡを、ドライブＣ：のＷｉｎｄｏｗｓファイルＨＥＬＬＯ．Ｃに変換します。

```
C:\>fp gettext c:data c:hello.c /hs80 ↓     (/HostSize 80)
```

例５）項目別に分けてプリント形式に変換する

ドライブＣ：のホストファイル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:\>fp getdata c:pla c:*.prn /hs64 /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６）例５をバッチファイル化する

例５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ファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
fp getdata c:pla c:*.prn /hs64 /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

例７）パラメータファイルを利用する

例６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ファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
:: ↓
fp getdata c:pla c:*.prn ++getpla.p ↓
```

＜ＧＥＴＰＬＡ．Ｐ＞

```
-- For GetData ------------------------------------------------ ↓
-- ＰＬＡファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 -- ↓
------------------------------------------------------------- ↓
/hostsize 64 ↓              -- ホストレコード長 ↓
/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。

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）一番単純な変換をする

ドライブＣ：のホストファイルＳＡＭＰＬＥを、ドライブＣ：のＷｉｎｄｏｗｓファイルＳＡＭＰＬＥ．ＴＸＴに変換します。

```
C:\>fp gettext c: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２）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Ｃ：にはホスト形式で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:\>fp getext c:* c:*.cbl ↓
```

例３）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Ｃ：にはホスト形式で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:\>fp gettext c:* 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４）別名をつける

ドライブＣ：のホストファイルＤＡＴＡを、ドライブＣ：のＷｉｎｄｏｗｓファイルＨＥＬＬＯ．Ｃに変換します。

```
C:\>fp gettext c:data c:hello.c ↓
```

例５）可変長のホストファイルを項目別に分けてプリント形式に変換する

ドライブＣ：のホストファイルＰＬＡは、さまざまな形式の項目が入りまじっている可変長のデータファイルです。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:\>fp getdata c:pla c:*.prn /hfd /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６）例５をバッチファイル化する

例５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ファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
fp getdata c:pla c:*.prn /hfd /map a2 k8 a10 a9 pdu4.3 zdu2 zds2.1 k20 ↓
```

例７）パラメータファイルを利用する

例６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ファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 ↓
:: ↓
fp getdata c:pla c:*.prn ++getpla.p ↓
```

＜ＧＥＴＰＬＡ．Ｐ＞

```
-- For GetData ------------------------------------------------ ↓
-- ＰＬＡファイルのホスト形式→Ｗｉｎｄｏｗｓプリント形式変換 -- ↓
------------------------------------------------------------- ↓
/hostfile data ↓              -- ホストファイル＝可変長データファイル ↓
/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

すでに同じ名前のＷｉｎｄｏｗｓファイルがある場合、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。このとき警告メッセージは出ないので注意してください。

２．４ ＧｅｔＴｅｘｔ（ｇｔ）　ゲット・テキスト

ホスト→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ホストファイルを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。ホストが汎用機・オフコンの場合、ホスト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	［ホストファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ホストファイル、Ｗｉｎｄｏｗｓファイル

ホストファイルと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

／ＨｏｓｔＳｉｚｅ （／ＩＳｉｚｅ）	ホストファイルのレコード長を指定する
／ＨｏｓｔＴａｂＥｘｐａｎｄ	タブ拡張する
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＨｏｓｔＥＯＦ	ＥＯＦコードを検査する
／ＨｏｓｔＮｏＥＯＦ	ＥＯＦコードを検査しない

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

これらのオプションのうち、／Ｃｏｄｅオプション、／ＨｏｓｔＳｉｚｅ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 ---- ホストが汎用機・オフコンの場合、ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する

／Ｃｏｄｅ	Ａｎｋ
	Ｋａｎｊｉｍｉｘ
／Ｃｏｄｅ	Ａｎｋ

ホストが汎用機・オフコンの場合、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、

ホスト側		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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ホスト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

ホストが汎用機・オフコンの場合、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、

／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**
／ＮｏＥＯＦ	**ＥＯＦコードをつけない**
ホストが汎用機・オフコンの場合	
／Ｃｏｄｅ　Ａｎｋ	**ＡＮＫ変換する**
／ＨｏｓｔＳｉｚｅ　８０	**ホストレコード長＝８０バイト**
ホストがＵｎｉｘの場合	
／ＨｏｓｔＴａｂＥｘｐａｎｄ　８	**８桁きざみでタブ拡張する**
ホストがＷｉｎｄｏｗｓの場合	
／ＨｏｓｔＴａｂＥｘｐａｎｄ　８	**８桁きざみでタブ拡張する**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードを検査する**

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（ホストが汎用機・オフコンの場合）

例１）コマンド行方式で実行する

漢字のないホストファイルＵＲＩＡＧＥをコマンド行方式で変換します。同じファイル名にし、拡張子．ＴＸＴをつけます。

```
C:\>fp gettext c:uriage c:*.txt /hs80 ↓          (/HostSize 80)
```

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 ↓
:: ↓
:: 売り上げデータのホスト→Ｗｉｎテキストファイル変換 ↓
:: ↓
fp gettext c:uriage c:*.txt /hs80 ↓
```

例３）標準の８桁きざみでタブ圧縮する

ホストファイルＤＡＴＡがあります。中身は漢字のないＣ言語のソースプログラムです。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:\>fp gettext c:data c:hello.c /hs80 /tc ↓     (/HostSize 80 /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:\>fp gettext c:data c:hello.c /hs80 /tc4 ↓    (/HostSize 80 /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Ｃ：のホストファイルＭＬＩＳＴ００２はＣＯＢＯＬのソースプログラムで、漢字も使っています。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:\>fp gettext c:mlist002 c:*.cbl /hs80 /ck ↓  (/HostSize 80 /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Ｃ：には、ＵＭＡＳｘｘｘｘというパターンの名前のホスト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:\>fp gettext c:umas* c:*.cbl /hs80 /ck ↓     (/HostSize 80 /Code KanjiMix)
```

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）コマンド行方式で実行する

漢字を含むホストファイルＵＲＩＡＧＥをコマンド行方式で変換します。同じファイル名にし、拡張子．ＴＸＴをつけます。

```
C:\>fp gettext c: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 ↓
:: ↓
:: 売り上げデータのホスト→Ｗｉｎテキストファイル変換 ↓
:: ↓
fp gettext c:uriage c:*.txt ↓
```

例３）標準の８桁きざみでタブ圧縮する

ホストファイルＤＡＴＡがあります。中身は漢字を含むＣ言語のソースプログラムです。それを、Ｗｉｎｄｏｗｓファイル名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:\>fp gettext c: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拡張、タブ圧縮しながら変換します。

```
C:\>fp gettext c:data c:hello.c /hte4 /tc4 ↓  (/HostTabExpand 4 /TabCompress 4)
```

例５）ソースプログラムを変換する

ドライブＣ：のホストファイル名ＭＬＩＳＴ００２はＣＯＢＯＬのソースプログラムで、漢字も使っています。それを、ドライブＣ：のＷｉｎｄｏｗｓファイル名ＭＬＩＳＴ００２．ＣＢＬに変換します。

```
C:\>fp gettext c:mlist002 c:*.cbl ↓
```

例６）ソースプログラムを何本もまとめて変換する

ドライブＣ：には、ＵＭＡＳｘｘｘｘというパターンの名前のホスト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:\>fp gettext c:umas* c:*.cbl ↓
```

■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、ホストが汎用機・オフコンの場合、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

ホスト→Ｗｉｎ　データファイル変換

ＧｅｔＤａｔａ(ｇｄ)コマンドは、ホストのデータファイルを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。プリント形式への変換とデリミタ形式への変換の、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	［ホストファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ホストファイル、Ｗｉｎｄｏｗｓファイル

ホストファイルと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

／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＨｏｓｔＳＱｕｅｅｚｅ	空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	空行を無視しない
／Ｄｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

Ｇｅｔ系コマンド共通のオプション

／ＨｏｓｔＳｉｚｅ（／ＩＳｉｚｅ）	**ホストファイルのレコード長を指定する**
／ＨｏｓｔＴａｂＥｘｐａｎｄ	**タブ拡張する**
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	**タブ拡張しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードを検査する**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードを検査しない**
／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**
／ＥＯＦ	**ＥＯＦコードをつける**
／ＮｏＥＯＦ	**ＥＯＦコードをつけない**

これらのオプションのうち、／ＨｏｓｔＦｉｌｅオプション、／ＨｏｓｔＤｅｌｉｍｉｔｅｄオプション、／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄオプション、／Ｄｅｌｉｍｉｔｅｄオプション、／ＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＭＡＰオプション、／ＨｏｓｔＳｉｚｅ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。

／ＨｏｓｔＦｉｌｅ ---- ホストファイルの形式を指定する

```
／ＨｏｓｔＦｉｌｅ　{Ｄａｔａ ｜ Ｒａｎｄｏｍ}
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、ホストファイルがデータファイルかランダムファイルかを指定します。

Ｄａｔａ指定をすると、ホストファイルをデータファイル（プリント／デリミタ形式）として扱います。

Ｒａｎｄｏｍ指定をすると、ホストファイルをランダムファイルとして扱います。

ホストが汎用機・オフコンの場合は、この指定は意味を持たなくなり、ランダムファイルとして扱います。

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ---------- デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ ---- プリント形式のファイルを変換する

```
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ( Ｂｙ　ＣＯＭｍａ
                              Ｂｙ　ＴＡＢ
                              Ｂｙ　ＳＰａｃｅ )
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓで、変換するホストファイルがデータファイルの場合、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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

／ＨｏｓｔＳＱｕｅｅｚｅ -------- 空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ ---- 空行を無視しない

／ＨｏｓｔＳＱｕｅｅｚｅ
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓで、変換するホストファイルがデータファイルの場合、空行を無視するかどうかを指定します。

／ＨｏｓｔＳＱｕｅｅｚｅオプションを指定すると、空行を無視します。

／ＨｏｓｔＮｏＳＱｕｅｅｚｅ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。

／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。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／ＭＡＰ	Ａｎｋ変換 Ｋａｎｊｉ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 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 デリミタ検出／引用符はずし 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／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。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｜＊］［：出力幅］

ＡＮＫ項目を変換します。ホストが汎用機・オフコンの場合、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。

＄　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。

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。
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。

Ｋａｎｊｉ変換

Ｋａｎｊｉ　［入力幅｜＊］［：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。

＄　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。

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。

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ　［入力幅｜＊］［：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)。ホストが汎用機・オフコンの場合、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、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。

＄　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。

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。
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。

＄　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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。

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｜＊］［：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、ホスト→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。

＄　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。

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。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。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。通常は、ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合に使います。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

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。

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のＣＯＢＯＬのゾーン形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。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のＣＯＢＯＬのゾーン形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。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、

入力幅	出力幅
1	3
2	5
3	8
4	１０
5	１３
6	１５
7	１７
8	１８

左記の値を基本として、
符号つきなら＋１、ピクチャ指定の小数部があれば＋１とし、
さらに、ピクチャ指定のほうが大きければ、
整数部桁数に、符号つきなら＋１、小数部があれば＋１＋小数点桁数

の要領で２進キャスト／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、

ＢｉｎＤｉｓｐ　ｉ４ｓ４．１　　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ４．１：１０　　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］
```

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。

出力ピクチャは省略できます。出力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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。

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）の交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。コード変換は、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、／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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、

／ＨＤｅｌｉ／ＭＡＰ　・・・，ＺＤ：Ｕ５，ＰＤ：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（“ ）でくくられていることがあります。その引用符は無視します。また、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。

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。

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。ただし、プリント形式への変換のときは、この引用符くくりの機能は無効になります。

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場合を示すと、つぎのような指定になります。

```
／Ｄｅｌｉ／ＭＡＰ　・・・，ＺＤＵ１０，“Ｋ２０”，ＰＤＵ３，・・・
```

◆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

変換対象にするホスト側（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、

＄　　ホスト側（入力）のレコード長
．　　ホスト側（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、

ホスト側（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＊ＴＲＡＮ＋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

変換対象にするホスト側（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、

$　　ホスト側（入力）のレコード長
.　　ホスト側（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＊ＴＲＡＮ＋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）

変換対象にするホスト側（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、

．　　ホスト側（入力）の現在の項目位置
＄　　ホスト側（入力）の入力全体の項目数
＊　　ホスト側（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　ＺｏｎｅＤｉｓｐＵ８　　と指定すれば、

ホスト側（入力）レコード末尾から２項目に位置づけ、
８バイトＺｏｎｅＤｉｓｐ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＊ＴＲＡＮ＋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

^［n｜1］

ホスト側（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。

＊　ホスト側（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。

●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**
／ＮｏＥＯＦ	**ＥＯＦコードをつけない**
ホストが汎用機・オフコンの場合	
／ＨｏｓｔＳｉｚｅ　２５６	**ホストレコード長＝２５６バイト**
ホストがＵｎｉｘの場合	
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ	**データファイルを変換する**
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	**プリント形式のファイルを変換する**
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	**タブ拡張しない**
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	**空行を無視しない**
ホストがＷｉｎｄｏｗｓの場合	
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ	**データファイルを変換する**
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	**プリント形式のファイルを変換する**
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	**タブ拡張しない**
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	**空行を無視しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードを検査する**

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

●オプションの指定方法

ホスト形式の、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	衛生数（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       -- 衛生数（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 ,      -- 衛生数（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:\>fp getdata c:planet c:test /hs64 /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）＞

```
fp getdata c:planet c:test /hs64 /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、Ｗｉｎファイルエディタ（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 ↓
::          ホスト形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式 ↓
fp getdata c:planet c:*.csv /hs64 /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 ↓
::          ホスト形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式 ↓
fp getdata c: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）          -- ↓
--          ホスト形式→Ｗｉｎｄｏｗｓコンマ区切り（ＣＳＶ）形式  -- ↓
/hostsize 64                -- ホストレコード長 ↓
/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 ,      -- 衛生数（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。

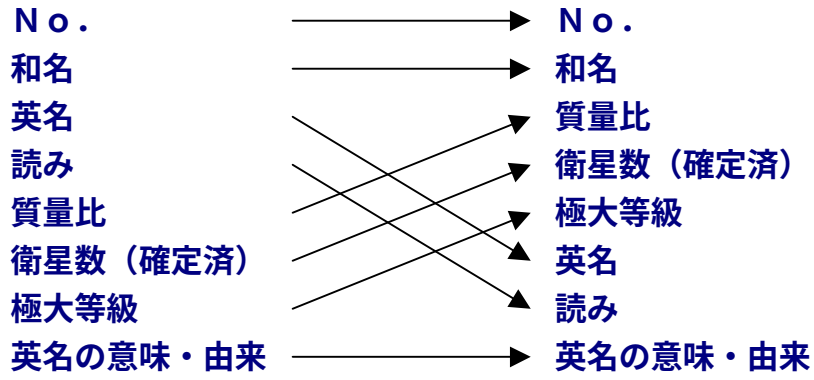
具体的には、レコード長をたとえば１２０バイト（改行コード２バイト含まず）に変換するなら、

```
/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       -- 衛生数（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       -- 衛生数（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      -- 衛生数（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 ,      -- 衛生数（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 ,        -- 衛生数（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 ,        -- 衛生数（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：ホストが汎用機・オフコンの場合）

例１）ＡＮＫコードのみのファイルを変換する

ドライブＣ：のホストファイル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:\>fp getdata c:journal c:*.dat /hs128 ↓          (/HostSize 128)
```

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:\>fp getdata c:journal c:*.dat /hs128 /map km ↓ (/HostSize 128 /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

ホストで作成したホスト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:\>fp getdata c:* c:*.src /hs80 /map a$-8 ↓       (/HostSize 80 /MAP Ank$-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:\>fp getdata c:* c:*.src /hs80 /map km$-8 ↓      (/HostSize 80
                                                    /MAP KanjiMix$-8)
```

例５）レコード長を拡張する

ドライブＣ：にホストファイル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:\>fp getdata c:zaiko?? c:*.dat /hs80 /map a:256-2 ↓   (/HostSize 80
                                                         /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Ｃ：のホストファイル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:\>fp getdata c:data c:shohin.prn /hs168 /map a8 k30 a @:256-2 ↓
                                  (/HostSize 168 /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Ｃ：のいくつかのホストファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子は．ＤＡＴにします。ホストファイルのレコード長は１２８バイトなので、Ｗｉｎｄｏｗｓファイルのレコード長はホスト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:\>fp getdata c:* c:*.dat /hs128 /q ↓      (/HostSize 128 /Query)
```

例８）項目別変換する

ドライブＣ：にホストファイル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:\>fp getdata c:address c:*.db /hs256 /map k16 a34 k40 dyymmdd:yyyy-mm-dd a1 ↓
    (/HostSize 256 /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＞

```
fp getdata c:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
/hostsize 256                       -- ホストレコード長 ↓
/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:\>fp getdata c:address c:*.db /hs256 /map @:1 k16 a34 k40 dyymmdd:yyyy-mm-dd
    @:0 a1 ↓
 (/HostSize 256 /MAP @:1 Kanji16 Ank34 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:\>fp getdata c:address c:*.db ++getaddr.p1 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ１＞

```
/hostsize 256                       -- ホストレコード長 ↓
/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:\>fp getdata c:address c:*.db /hs256 /map @96 a1 @0 k16 a34 k40
    dyymmdd:yyyy-mm-dd ↓
(/HostSize 256 /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 12 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:\>fp getdata c:address c:*.db ++getaddr.p2 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ２＞

```
/hostsize 256                          -- ホストレコード長 ↓
/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＞

```
/hostsize 256                          -- ホストレコード長 ↓
/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：ホストが汎用機・オフコンの場合）

例１）コンマ区切り（ＣＳＶ）形式に変換する

ドライブＣ：にホストファイル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、コンマ区切り（ＣＳＶ）形式にします。

```
C:\>fp getdata c:address c:*.db /hs256 /d /map "k16", "a16", "a12", "a6", "k40",
    dyymmdd:yyyy-mm-dd, a1 ↓
  (/HostSize 256 /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:\>fp getdata c:address c:*.db /hs256 /dbtab /map "k16", "a16", "a12", "a6",
    "k40", dyymmdd:yyyy-mm-dd, a1 ↓
      (/HostSize 256 /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:\>fp getdata c:address c:*.db /hs256 /dbsp /map "k16", "a16", "a12", "a6",
    "k40", dyymmdd:yyyy-mm-dd, a1 ↓
    (/HostSize 256 /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:\>fp getdata c:address c:*.db /hs256 /d /map @96 a1, @0 "k16", "a16", "a12",
    "a6", "k40", dyymmdd:yyyy-mm-dd ↓
      (/HostSize 256 /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:\>fp getdata c:address c:*.db +addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/hostsize 256                          -- ホストレコード長 ↓
/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＞

```
fp getdata c:address c:*.db +addr.p ↓
```

■使用例１（プリント形式：ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）可変長のホストファイルを変換する

ドライブＣ：のホストファイルＪＯＵＲＮＡＬを、ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴに変換します。データには漢字も含まれています。改行コードは自動的につきます。

```
C:\>fp getdata c:journal c:*.dat /hfd /map km↓  (/HostFile Data /MAP KanjiMix)
```

例２）ソースプログラムの行番号を削除する

ホストで作成したホスト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。データには漢字も含まれています。

```
C:\>fp getdata c:* c:*.src /hfd /map km$-8 ↓  (/HostFile Data /MAP KanjiMix$-8)
```

例３）可変長ファイルを固定長ファイルにする

ドライブＣ：にホストファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２があります。レコード長は可変で、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。内容は漢字を含むデータです。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:\>fp getdata c:zaiko?? c:*.dat /hfd /map km:256-2 ↓    (/HostFile Data
                                                          /MAP KanjiMix:256-2)
```

例４）桁移動の機能でレコード長を拡張する

ドライブＣ：のホストファイルＤＡＴＡを、ドライブＣ：のＷｉｎｄｏｗｓファイルＳＨＯＨＩＮ．ＰＲＮに変換します。内容は固定長の商品マスターで、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:\>fp getdata c:data c:shohin.prn /hfr /hs168 /map a8 k30 a @:256-2 ↓
                    (/HostFile Random /HostSize 168 /MAP Ank8 Kanji30 Ank @:256-2)
```

例５）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Ｃ：のいくつかのホストファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子は．ＤＡＴにします。ホストファイルのレコード長は１２８バイトなので、Ｗｉｎｄｏｗｓファイルのレコード長はホスト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)。内容は漢字を含むデータです一応、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:\>fp getdata c:* c:*.dat /hfr /hs128 /map km /q ↓
                        (/HostFile Random /HostSize 128 /Map KanjiMix /Query)
```

例６）項目別変換する

ドライブＣ：にホストファイル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:\>fp getdata c:address c:*.db /hfr /hs256 /map k16 a34 k40 dyymmdd:yyyy-mm-dd
    a1 ↓     (/HostFile Random /HostSize 256 /MAP Kanji16 Ank16+12+6 Kanji40
                                        Date yymmdd:yyyy-mm-dd Ank1)
```

例７）例６をバッチファイル＋パラメータファイル化する

例６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＞

```
fp getdata c:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
/hostfile random                    -- ホストファイル＝ランダムファイル ↓
/hostsize 256                       -- ホストレコード長 ↓
/map ↓
      kanji   16                    -- 氏名 ↓
      ank     16+12+6               -- フリガナ、電話番号、郵便番号 ↓
      kanji   40                    -- 住所 ↓
      date    yymmdd:yyyy-mm-dd     -- 生年月日 ↓
      ank     1                     -- 区分 ↓
```

例８）項目を組み替える（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:\>fp getdata c:address c:*.db /hfr /hs256 /map @:1 k16 a34 k40
    dyymmdd:yyyy-mm-dd @:0 a1 ↓     (/HostFile Random /HostSize 256
                /MAP @:1 Kanji16 Ank34 Kanji40 Date yymmdd:yyyy-mm-dd @:0 Ank1)
```

例９）項目組み替えにパラメータファイルを使う（その１）

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:\>fp getdata c:address c:*.db ++getaddr.p1 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ１＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 256                           -- ホストレコード長 ↓
/map ↓
      @:1  kanji   16                   -- 氏名 ↓
           ank     16+12+6              -- フリガナ、電話番号、郵便番号 ↓
           kanji   40                   -- 住所 ↓
           date    yymmdd:yyyy-mm-dd    -- 生年月日 ↓
      @:0  ank     1                    -- 区分 ↓
```

例 10）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:\>fp getdata c:address c:*.db /hfr /hs256 /map @96 a1 @0 k16 a34 k40
    dyymmdd:yyyy-mm-dd ↓      (/HostFile Random /HostSize 256
            /Map @96 Ank1 @0 Kanji16 Ank16+12+6 Kanji40 Date yymmdd:yyyy-mm-dd)
```

例 11）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:\>fp getdata c:address c:*.db ++getaddr.p2 ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ２＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 256                           -- ホストレコード長 ↓
/map ↓
      @96  ank     1                    -- 区分 ↓
      @0   kanji   16                   -- 氏名 ↓
           ank     16+12+6              -- フリガナ、電話番号、郵便番号 ↓
           kanji   40                   -- 住所 ↓
           date    yymmdd:yyyy-mm-dd    -- 生年月日 ↓
```

例 12）例 11 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＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 256                           -- ホストレコード長 ↓
/map ↓
      @96  ank     1                    -- 区分 ↓
      @0   kanji   16                   -- 氏名 ↓
           ank     16+12+6              -- フリガナ、電話番号、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：ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）コンマ区切り（ＣＳＶ）形式に変換する

ドライブＣ：にホストファイル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、コンマ区切り（ＣＳＶ）形式にします。

```
C:\>fp getdata c:address c:*.db /hfr /hs256 /d /map "k16", "a16", "a12", "a6",
    "k40", dyymmdd:yyyy-mm-dd, a1 ↓          (/HostFile Random /HostSize 256
                          /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:\>fp getdata c:address c:*.db /hfr /hs256 /dbtab /map "k16", "a16", "a12",
    "a6", "k40", dyymmdd:yyyy-mm-dd, a1 ↓    (/HostFile Random /HostSize 256
                    /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:\>fp getdata c:address c:*.db /hfr /hs256 /dbsp /map "k16", "a16", "a12",
    "a6", "k40", dyymmdd:yyyy-mm-dd, a1 ↓    (/HostFile Random /HostSize 256
                        /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:\>fp getdata c:address c:*.db /hfr /hs256 /d /map @96 a1, @0 "k16", "a16",
    "a12", "a6", "k40", dyymmdd:yyyy-mm-dd ↓
(/HostFile Random /HostSize 256 /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:\>fp getdata c:address c:*.db +addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 256                           -- ホストレコード長 ↓
/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＞

```
fp getdata c: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

ホスト→Ｗｉｎ　ランダムファイル変換

ＧｅｔＲａｎｄ(ｇｒ)コマンドは、ホストのデータファイルを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し、オンラインファイル転送と組み合わせて利用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	［ホストファイル　Ｗｉｎｄｏｗｓファイル　［オプション］］

■パラメータの説明

●ホストファイル、Ｗｉｎｄｏｗｓファイル

ホストファイルと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

／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルを変換する
／ＨｏｓｔＳＱｕｅｅｚｅ	空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ	空行を無視しない
／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

Ｇｅｔ系コマンド共通のオプション

／ＨｏｓｔＳｉｚｅ（／ＩＳｉｚｅ）	**ホストファイルのレコード長を指定する**
／ＨｏｓｔＴａｂＥｘｐａｎｄ	**タブ拡張する**
／ＨｏｓｔＮｏＴａｂＥｘｐａｎｄ	**タブ拡張しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードを検査する**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードを検査しない**
／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**

これらのオプションのうち、／ＨｏｓｔＦｉｌｅオプション、／ＨｏｓｔＤｅｌｉｍｉｔｅｄオプション、／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄオプション、／Ｓｉｚｅオプション、／ＭＡＰオプション、／ＨｏｓｔＳｉｚｅ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。

／ＨｏｓｔＦｉｌｅ ---- ホストファイルの形式を指定する

```
／ＨｏｓｔＦｉｌｅ　{Ｄａｔａ ｜ Ｒａｎｄｏｍ}
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、ホストファイルがデータファイルかランダムファイルかを指定します。

Ｄａｔａ指定をすると、ホストファイルをデータファイル（プリント／デリミタ形式）として扱います。

Ｒａｎｄｏｍ指定をすると、ホストファイルをランダムファイルとして扱います。

ホストが汎用機・オフコンの場合は、この指定は意味を持たなくなり、ランダムファイルとして扱います。

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ---------- デリミタ形式のファイルを変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ ---- プリント形式のファイルを変換する

```
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ( Ｂｙ　ＣＯＭｍａ
                              Ｂｙ　ＴＡＢ
                              Ｂｙ　ＳＰａｃｅ )
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓで、変換するホストファイルがデータファイルの場合、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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

／ＨｏｓｔＳＱｕｅｅｚｅ -------- 空行を無視する
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ ---- 空行を無視しない

```
／ＨｏｓｔＳＱｕｅｅｚｅ
／ＨｏｓｔＮｏＳＱｕｅｅｚｅ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓで、変換するホストファイルがデータファイルの場合、空行を無視するかどうかを指定します。

／ＨｏｓｔＳＱｕｅｅｚｅオプションを指定すると、空行を無視します。

／ＨｏｓｔＮｏＳＱｕｅｅｚｅ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。

／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用できます。

＄　　ホスト側（入力）のレコード長

ホスト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、「元のホスト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＝ホスト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。

／ＳｉｚｅオプションでＷｉｎｄｏｗｓレコード長を指定できることは、ＧｅｔＲａｎｄ(ｇｒ)コマンドの性格をよく反映しています。

このコマンドは、おもにＷｉｎｄｏｗｓ　ＣＯＢＯＬの順ファイルやＢＡＳＩＣのランダムファイルに変換することをねらったものである、ということは冒頭で述べたとおりです。そして、ＢＡＳＩＣ側の事情を考慮すると、レコード長を任意に指定できないと困ることが多いのです。

とくに困るのは、Ｎ８８ＢＡＳＩＣコンパイラ／インタプリタを使うときです。Ｎ８８ＢＡＳＩＣコンパイラ／インタプリタでは、１つのプログラム内で複数のレコード長のランダムファイルを扱うことができません。

もう１つの理由は、ゾーン形式とパック形式の変換、日付データの変換をサポートしているため、レコード長を増減する機能がないと困るからです。また、不要な項目がいくつもあるときや、予備領域が多すぎるときはレコード長を縮めたいものです。

◆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 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 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 デリミタ検出／引用符はずし 改行コード挿入／改行コード挿入２ 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。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｜＊］［：出力幅］

ＡＮＫ項目を変換します。ホストが汎用機・オフコンの場合、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。

＄　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。
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。

Ｋａｎｊｉ変換

Ｋａｎｊｉ　［入力幅｜＊］［：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。

＄　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。

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（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

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ　［入力幅｜＊］［：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）。ホストが汎用機・オフコンの場合、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、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。

＄　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。

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。
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。

＄　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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。

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｜＊］［：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。

＄　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。

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。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。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。通常は、ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合に使います。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

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

ＤｉｓｐＰａｃｋ　［入力幅｜１５］：ピクチャ

ホストの文字形式数値項目を、Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパック形式数値項目に変換します。通常は、ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合に使います。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

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。

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のＣＯＢＯＬのゾーン形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。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のＣＯＢＯＬのパック形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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す。

出力ピクチャは省略できます。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。

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のＣＯＢＯＬのゾーン形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。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のＣＯＢＯＬのパック形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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、

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

出力ピクチャは省略できます。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

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。ホストが汎用機・オフコンの場合、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、

入力幅	出力幅
1	3
2	5
3	8
4	１０
5	１３
6	１５
7	１７
8	１８

左記の値を基本として、
符号つきなら＋１、ピクチャ指定の小数部があれば＋１とし、
さらに、ピクチャ指定のほうが大きければ、
整数部桁数に、符号つきなら＋１、小数部があれば＋１＋小数点桁数

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　ｉ４ｓ４．１　　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ４．１：１０　　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］

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

入力ピクチャは省略できます。入力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

入力ピクチャは省略できます。入力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］
```

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。

出力ピクチャは省略できます。出力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］
```

ホストの２進形式整数・小数項目を、Ｗｉｎｄｏｗｓ　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。

出力ピクチャは省略できます。出力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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
```

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、ホスト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。

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）の交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。コード変換は、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、／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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、

／ＨＤｅｌｉ／ＭＡＰ　・・・，ＺＤ：ｕ５，ＰＤ：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（“）でくくられていることがあります。その引用符は無視します。また、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。

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。

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

変換対象にするホスト側（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、

＄　　ホスト側（入力）のレコード長
．　　ホスト側（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、

ホスト側（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。
@：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＊ＴＲＡＮ＋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）

変換対象にするホスト側（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、

．　　ホスト側（入力）の現在の項目位置
＄　　ホスト側（入力）の入力全体の項目数
＊　　ホスト側（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　ＺｏｎｅＤｉｓｐＵ８　　と指定すれば、

ホスト側（入力）レコード末尾から２項目に位置づけ、
８バイトＺｏｎｅＤｉｓｐ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＊ＴＲＡＮ＋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

```
^［n｜1］
```

ホスト側（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。

＊　ホスト側（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

●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、

```
／ＨｏｓｔＳｉｚｅ　２５６　　ホストレコード長＝２５６バイト
／Ｓｉｚｅ　＄　　Ｗｉｎｄｏｗｓレコード長＝ホストレコード長
／ＭＡＰ　Ａｎｋ　　レコード全体をＡＮＫデータ変換、
　　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
ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合
／ＨｏｓｔＦｉｌｅ　Ｒａｎｄｏｍ　　ランダムファイルを変換する
```

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

●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

ホスト形式の、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	衛生数（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      -- 衛生数（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。

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:\>fp getdata c:planet c:test /hs64 /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）＞

```
fp getdata c:planet c:test /hs64 /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、Ｗｉｎファイルエディタ（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 ↓
::                          ホスト形式→Ｗｉｎランダムファイル形式 ↓
fp getrand c:planet c:*.ran /hs64 /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 ↓
::                          ホスト形式→Ｗｉｎランダムファイル形式 ↓
fp getrand c: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）          -- ↓
--                          ホスト形式→Ｗｉｎランダムファイル形式  -- ↓
↓
/hostsize 64                -- ホストレコード長＝６４バイト ↓
/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

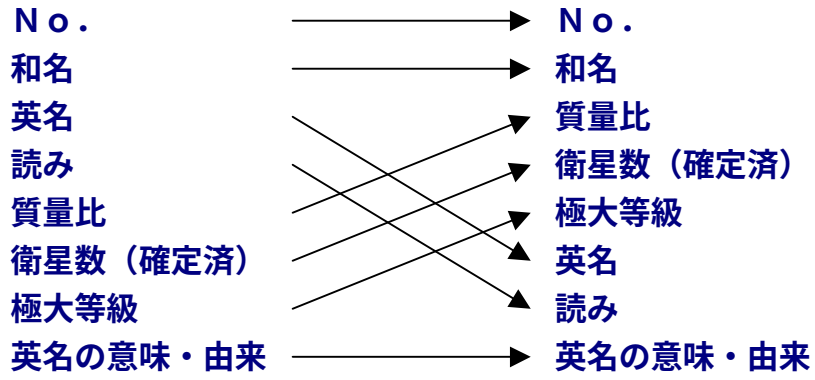
必要に応じて、項目長の増減をするのも簡単です。つぎのように、

```
/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（ホストが汎用機・オフコンの場合）

例１）ＡＮＫのみ、レコード長も変えない

ドライブＣ：のホストファイル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:\>fp getrand c:journal c:*.dat /hs256 ↓          (/HostSize 256)
```

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:\>fp getrand c:journal c:*.dat /hs256 /map a!r /s$+2 ↓
                                   (/HostSize 256 /Map Ank!R /Size $+2)
```

例３）バイナリ変換する

ドライブＣ：のホストファイル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:\>fp getrand c:font24 c:*.pxl /hs256 /map b ↓    (/HostSize 256 /Map Binary)
```

例４）レコード長を増やす

ドライブＣ:にあるレコード長が８０バイトのホストファイル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:\>fp getrand c:zaiko?? c:*.dat /hs80 /s256 ↓     (/HostSize 80 /Size 256)
```

例５）レコード長をそろえる

ドライブＣ：のすべてのホストファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子は．ＤＡＴにします。ホストファイルのほうは、レコード長はすべて２５６バイトです。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:\>fp getrand c:* c:*.dat /hs256 /s512 ↓          (/HostSize 256 /Size 512)
```

例６）項目別変換する

ドライブＣ：にホストファイル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:\>fp getrand c:address c:*.db /hs256 /map k16 a34 k40 dyymmdd:yyyy-mm-dd a1 ↓
    (/HostSize 256 /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:\>fp getrand c:address c:*.db /hs256 /map @:1 k16 a34 k40 dyymmdd:yyyy-mm-dd
    @:0 a1 ↓
 (/HostSize 256 /MAP @:1 Kanji16 Ank24 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:\>fp getrand c:address c:*.db /hs256 /map @90 a1 @0 k16 a34 k40
    dyymmdd:yyyy-mm-dd ↓
(/HostSize 256 /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 ↓
fp getrand c:address c:*.db /hs256 /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 ↓
fp getrand c:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
   .
   .
/hostsize 256                          -- ホストレコード長 ↓
/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Ｃ：のホストファイル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:\>fp getrand c:hanbai c:*.dat /hs256 /map km ↓      (/HostSize 256
                                                       /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。

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）レコード長を変えずに、ランダムファイル変換する

ドライブＣ：のホストファイルＪＯＵＲＮＡＬを、ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴに変換します。データには漢字も含まれています。レコード長は変わらず、２５６バイトになります。

```
C:\>fp getrand c:journal c:*.dat /hfr /hs256 /map km↓
                          (/HostFile Random /HostSize 256 Random /MAP KanjiMix)
```

例２）可変長ファイルを固定長ファイルにして、改行コードをつける

ドライブＣ：のホストファイルＫＡＨＥＮをランダムファイル変換し、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:\>fp getrand c:kahen c:*.dat /hfd /map km!r /s$+2 ↓
                                (/HostFile Data /Map KanjiMix!R /Size 256+2)
```

例３）バイナリ変換する

ドライブＣ：のホストファイル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:\>fp getrand c:font24 c:*.pxl /hfr /hs256 /map b ↓
                                (/HostFile Random /HostSize 256 /Map Binary)
```

例４）レコード長を増やす

ドライブＣ:にあるレコード長が８０バイトのホストファイル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:\>fp getrand c:zaiko?? c:*.dat /hfr /hs80 /s256 /map km ↓
                          (/HostFile Random /HostSize 80 /Size 256 /MAP KanjiMix)
```

例５）レコード長をそろえる

ドライブＣ：のすべてのホストファイルを、ドライブＣ：のＷｉｎｄｏｗｓファイルに変換します。拡張子は．ＤＡＴにします。ホストファイルのほうは、レコード長はすべて２５６バイトです。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。

```
C:\>fp getrand c:* c:*.dat /hs256 /s512 /map km ↓
                    (/HostFile Random /HostSize 256 /Size 512 /MAP KanjiMix)
```

例６）項目別変換する

ドライブＣ：にホストファイル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:\>fp getrand c:address c:*.db /hfr /hs256 /map k16 a34 k40 dyymmdd:yyyy-mm-dd
    a1 ↓        (/HostFile Random /HostSize 256
                 /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:\>fp getrand c:address c:*.db /hfr /hs256 /map @:1 k16 a34 k40
    dyymmdd:yyyy-mm-dd @:0 a1 ↓    (/HostFile Random /HostSize 256
            /MAP @:1 Kanji16 Ank24 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:\>fp getrand c:address c:*.db /hfr /hs256 /map @90 a1 @0 k16 a34 k40
    dyymmdd:yyyy-mm-dd ↓      (/HostFile Random /HostSize 256
          /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 ↓
fp getrand c:address c:*.db /hfr /hs256 /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 ↓
fp getrand c:address c:*.db ++getaddr.p ↓
```

＜ＧＥＴＡＤＤＲ．Ｐ＞

```
   .
   .
/hostfile random                    -- ホストファイル＝ランダムファイル ↓
/hostsize 256                       -- ホストレコード長 ↓
/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 ↓
```

■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→ホストテキストファイル変換
ＰｕｔＤａｔａ(ｐｄ)コマンド　　Ｗｉｎ→ホストデータファイル変換
ＰｕｔＲａｎｄ(ｐｒ)コマンド　　Ｗｉｎ→ホスト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ファイルをホスト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 ｐｔ	［Ｗｉｎｄｏｗｓファイル　ホストファイル　［オプション］］
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

●Ｗｉｎｄｏｗｓファイル指定

［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。

●ホストファイル指定

［ｄ：］［パス名指定］基本ファイル名［．拡張子］

ｄ：はドライブ名

出力側のホスト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＊を指定します。＊を指定するとＷｉｎｄｏｗｓファイルの基本ファイル名がホストファイルの基本ファイル名になります。

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。拡張子に＊を指定するとＷｉｎｄｏｗｓファイルの拡張子がホストファイルの拡張子になります。

まとめると、基本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

ホスト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

Ｐｕｔ系コマンドには各コマンドに共通のオプションと、コマンドごと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

／ＨｏｓｔＳｉｚｅ（／ＩＳｉｚｅ）	ホストファイルのレコード長を指定する
／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
／ＥＯＦ	ＥＯＦコードを検査する
／ＮｏＥＯＦ	ＥＯＦコードを検査しない
／ＨｏｓｔＥＯＦ	ＥＯＦコードをつける
／ＨｏｓｔＮｏＥＯＦ	ＥＯＦコードをつけない

このうち、／ＨｏｓｔＳｉｚｅ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
／ＨｏｓｔＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮する
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮しない

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
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／Ｑｕｅｅｚｅ	空行を無視する
／ＮｏＳＱｕｅｅｚｅ	空行を無視しない
／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

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
／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

以下、Ｐｕｔ系コマンドに共通のオプションを説明します。

／ＨｏｓｔＳｉｚｅ ---- ホストファイルがランダムファイル（固定長）の場合、
（／ＩＳｉｚｅ）　　レコード長を指定する

／ＨｏｓｔＳｉｚｅ　レコード長

ランダムファイル（固定長）のレコード長を１～３２７６７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、

ＰｕｔＴｅｘｔ(ｐｔ)コマンド　　８０バイト
ＰｕｔＤａｔａ(ｐｄ)コマンド　２５６バイト
ＰｕｔＲａｎｄ(ｐｒ)コマンド　２５６バイト

になります。

ホストファイルがランダムファイルの場合、レコード長の指定が間違っていると正しいデータ変換が行われませんので、このオプションの指定は極めて重要です。ホストが汎用機・オフコンの場合、この指定が特に重要になります。
ホストファイルがテキスト／データファイル（可変長）の場合は、この指定は意味を持たなくなり、指定しても無視されます。

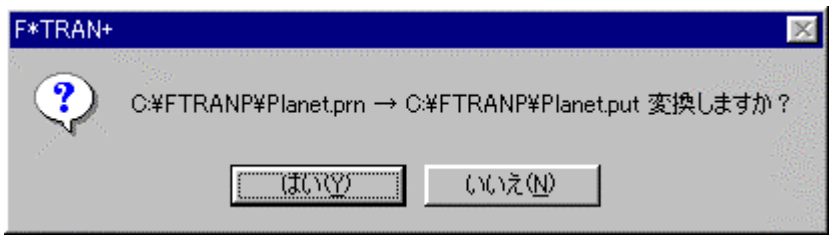
／Ｑｕｅｒｙ -------- 変換するか否かを問い合わせる
／ＮｏＱｕｅｒｙ ---- ファイル名の確認なしで自動変換する

／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＊ＴＲＡＮ＋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。

／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。

／ＨｏｓｔＥＯＦ -------- ＥＯＦコードをつける
／ＨｏｓｔＮｏＥＯＦ ---- ＥＯＦコードをつけない

／ＨｏｓｔＥＯＦ ／ＨｏｓｔＮｏＥＯＦ

ホストがＷｉｎｄｏｗｓの場合に、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ＥＯＦ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。

■使用例（ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイルＳＡＭＰＬＥにＡＮＫ変換します。

```
C:\>fp puttext c:sample.txt c:* /hs80 ↓          (/HostSize 80)
```

例２）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Ｃ：の同じ名前のホストファイルにＡＮＫ変換します。

```
C:\>fp puttext c:*.* c:* /hs80 ↓                 (/HostSize 80)
```

例３）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Ｃ：のホストファイルにＡＮＫ変換します。

```
C:\>fp puttext c:*.txt c:* /hs80 ↓               (/HostSize 80)
```

例４）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ファイルをすべてホストファイルにＡＮＫ変換します。

```
C:\>fp puttext c:umas* c:* /hs80 ↓               (/HostSize 80)
```

例５）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:\>fp puttext c:file02.txt c:* /hs80 /neof ↓    (/HostSize 80 /NoEOF)
```

例６）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:\>fp putrand c:file03.ran c:* /hs256 /eof ↓    (/HostSize 256 /EOF)
```

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

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Ｃ：のホストファイルＳＡＭＰＬＥに変換します。

```
C:\>fp puttext c:sample.txt c:* ↓
```

例２）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Ｃ：の同じ名前のホストファイルに変換します。

```
C:\>fp puttext c:*.* c:* ↓
```

例３）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Ｃ：のホストファイルに変換します。

```
C:\>fp puttext c:*.txt c:* ↓
```

例４）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ファイルをすべてホストファイルに変換します。

```
C:\>fp puttext c:umas* c:* ↓
```

■注意事項

同名ファイルは置換する

すでに同じ名前のホストファイルがある場合、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。このとき警告メッセージは出ないので注意してください。

２．８ ＰｕｔＴｅｘｔ（ｐｔ）　プット・テキスト

Ｗｉｎ→ホスト　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ファイルをホスト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。ホストファイルは、ホストが汎用機・オフコンの場合は固定長になり、ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合は可変長になります。

■コマンド形式

コマンド	パラメータ
ＰｕｔＴｅｘｔ ｐｔ	［Ｗｉｎｄｏｗｓファイル　ホストファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ホストファイル

Ｗｉｎｄｏｗｓファイルとホスト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
／ＨｏｓｔＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮する
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ	タブ圧縮しない

Ｐｕｔ系コマンド共通のオプション

／ＨｏｓｔＳｉｚｅ （／ＩＳｉｚｅ）	ホストファイルのレコード長を指定する
／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

／ＥＯＦ	**ＥＯＦコードを検査する**
／ＮｏＥＯＦ	**ＥＯＦコードを検査しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードをつける**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードをつけない**

これらのオプションのうち、／Ｃｏｄｅオプション、／ＨｏｓｔＳｉｚｅ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 ---- ホストが汎用機・オフコンの場合、ＡＮＫ変換かＡＮＫ・漢字まじり変換かを指定する

／Ｃｏｄｅ	Ａｎｋ Ｋａｎｊｉｍｉｘ
／Ｃｏｄｅ	Ａｎｋ

ホストが汎用機・オフコンの場合、ホスト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。以下に示す、

Ｗｉｎｄｏｗｓ側		ホスト側
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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ホスト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

ホストが汎用機・オフコンの場合、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。

／ＨｏｓｔＴａｂＣｏｍｐｒｅｓｓ -------- タブ圧縮する
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ ---- タブ圧縮しない

```
／ＨｏｓｔＴａｂＣｏｍｐｒｅｓｓ  〔タブ間隔
                                    ８〕
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ
```

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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、

／ＴａｂＥｘｐａｎｄ	**８桁きざみでタブ拡張する**
／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**
ホストが汎用機・オフコンの場合	
／Ｃｏｄｅ　Ａｎｋ	**ＡＮＫ変換する**
／ＨｏｓｔＳｉｚｅ　８０	**ホストレコード長＝８０バイト**
ホストがＵｎｉｘの場合	
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ	**タブ圧縮しない**
ホストがＷｉｎｄｏｗｓの場合	
／ＨｏｓｔＮｏＴａｂＣｏｍｐｒｅｓｓ	**タブ圧縮しない**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードをつけない**

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（ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイルＳＡＭＰＬＥに変換します。内容はＡＮＫのみとし、ＡＮＫ変換します。

```
C:\>fp puttext c:sample.txt c:* /hs80 ↓          (/HostSize 80)
```

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Ｃ：のホストファイルＳＡＭＰＬＥに変換します。漢字が使われています。

```
C:\>fp puttext c:sample.txt c:* /hs80 /ck ↓     (/HostSize 80 /Code KanjiMix)
```

例３）ＫＩ／ＫＯが入るのを見込んで漢字まじり変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＲＥＡＤＭＥ．ＤＯＣを、ドライブＣ：のホストファイル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、ホストファイルのレコード長を１２８バイトにします。

```
C:\>fp puttext c:readme.doc c:* /hs80 /ck ↓     (/HostSize 80 /Code KanjiMix)
```

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Ｃ：のホストファイルＨＥＬＬＯに変換します。

```
C:\>fp puttext c:hello.c c:* /hs80 /te ↓        (/HostSize 80 /TabExpand)
```

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Ｃ：のホストファイルＨＥＬＬＯに変換します。

```
C:\>fp puttext c:hello.c c:* /hs80 /te4 ↓       (/HostSize 80 /TabExpand 4)
```

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

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Ｃ：のホストファイルＳＡＭＰＬＥに変換します。

```
C:\>fp puttext c:sample.txt c:* ↓
```

例２）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Ｃ：のホストファイルＨＥＬＬＯに変換します。

```
C:\>fp puttext c:hello.c c:* /te ↓              (/TabExpand)
```

例３）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Ｃ：のホストファイルＨＥＬＬＯに変換します。

```
C:\>fp puttext c:hello.c c:* /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、ホストが汎用機・オフコンの場合、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、ホスト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、変換後ホストファイルの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、ホストが汎用機・オフコンの場合、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→ホスト　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ホスト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｡項目別に分けて変換できます｡ホストが汎用機・オフコンの場合は、ホスト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　ホストファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ホストファイル

Ｗｉｎｄｏｗｓファイルとホスト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
／ＴａｂＥｘｐａｎｄ	タブ拡張する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／Ｑｕｅｅｚｅ	空行を無視する
／ＮｏＳＱｕｅｅｚｅ	空行を無視しない
／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

Ｐｕｔ系コマンド共通のオプション

／ＨｏｓｔＳｉｚｅ（／ＩＳｉｚｅ）	**ホストファイルのレコード長を指定する**
／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**
／ＥＯＦ	**ＥＯＦコードを検査する**
／ＮｏＥＯＦ	**ＥＯＦコードを検査しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードをつける**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードをつけない**

これらのオプションのうち、／Ｄｅｌｉｍｉｔｅｄオプション、／ＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＨｏｓｔＦｉｌｅオプション、／ＨｏｓｔＤｅｌｉｍｉｔｅｄオプション、／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＭＡＰオプション、／ＨｏｓｔＳｉｚｅ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

```
                  ┌ By  COMma ┐
／Ｄｅｌｉｍｉｔｅｄ │ 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。

／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。

／ＨｏｓｔＦｉｌｅ ---- ホストファイルの形式を指定する

```
／ＨｏｓｔＦｉｌｅ　{Ｄａｔａ | Ｒａｎｄｏｍ}
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、ホストファイルがデータファイルかランダムファイルかを指定します。

Ｄａｔａ指定をすると、ホストファイルをデータファイル（プリント／デリミタ形式）として扱います。

Ｒａｎｄｏｍ指定をすると、ホストファイルをランダムファイルとして扱います。

ホストが汎用機・オフコンの場合は、この指定は意味を持たなくなり、ランダムファイルとして扱います。

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

```
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ［Ｂｙ　ＣＯＭｍａ
                              Ｂｙ　ＴＡＢ
                              Ｂｙ　ＳＰａｃｅ］
                                  ［Ｃｏｍｐｒｅｓｓ
                                    ＮｏＣｏｍｐｒｅｓｓ］
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、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。

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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。

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／MAP	Ａｎｋ変換 Ｋａｎｊｉ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 Ｙｅａｒ設定／ＤａｔｅＤｅｌｉｍ設定 Ｄａｔｅ変換 デリミタ検出／引用符はずし 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／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、ホスト側（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｜＊］［：出力幅］

ＡＮＫ項目を変換します。ホストが汎用機・オフコンの場合、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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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、ホスト側の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。

＊　ホスト側（出力）の残りバイト数

Ｋａｎｊｉ変換

Ｋａｎｊｉ　［入力幅｜＊］［：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、ホスト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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。

＊　ホスト側（出力）の残りバイト数

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ　［入力幅｜＊］［：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)。ホストが汎用機・オフコンの場合、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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。

＊　ホスト側（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。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、ホスト側のユーザー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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。

＊　ホスト側（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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系か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)。

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｜＊］［：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→ホスト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。

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。

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、ホスト側のバイナリ項目のおわりに、ホストが汎用機・オフコンの場合はＮＵＬ（００Ｈ）が、ホストがＵｎｉｘ／Ｗｉｎｄｏｗｓの場合はスペース（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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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）の挿入ができます。通常は、ホストが汎用機・オフコンの場合に指定します。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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」の章で説明しました。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。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。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の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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。

出力ピクチャは省略できます。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の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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。

出力ピクチャは省略できます。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］

Ｗｉｎｄｏｗｓの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。

入力ピクチャは省略できます。入力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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。

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）の交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。コード変換は、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　・・・，ＤＺ：ｕ５，ＤＰ：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（“）でくくられていることがあります。その引用符は無視します。また、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。

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。

デリミタ挿入（コンマ）

,

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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。ＰｕｔＤａｔａ(ｐｄ)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、／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

```
／HDeli／MAP　・・・, DZU5, DZS3．2,・・・
```

引用符くくり

“～”

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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。ただし、プリント形式への変換のときは、この引用符くくりの機能は無効になります。

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場合を示すと、つぎのような指定になります。

```
／HDeli／MAP　・・・, ZDU10,“K20”, PDU3,・・・
```

◆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
```

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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、出力がプリント形式データファイルのときは、この改行コード挿入の機能は無効になります。

出力がデータファイルのときに、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。

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

改行コード挿入２

```
！N
```

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、ホスト側（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、出力がプリント形式データファイルの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Ｗｉｎｄｏｗｓ側（入力）の桁位置や、変換結果を書き込むホスト側（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。

＄　ホスト側（出力）の末尾
．　ホスト側（出力）の現在の桁位置

◆注意　----　先頭を０桁目とする
Ｆ＊ＴＲＡＮ＋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ホスト側（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、

$　　ホスト側（出力）の末尾
.　　ホスト側（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　DispZone:u8,　　と指定すれば、

Windowsレコード末尾の８バイト前に出力先を移動し
ホスト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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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）

```
^［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　・・・　＾＊**　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。

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、ホスト側（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。

●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から変換する
／ＮｏＴａｂＥｘｐａｎｄ	タブ拡張しない
／ＮｏＳＱｕｅｅｚｅ	空行を無視しない
／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
／ＥＯＦ	ＥＯＦコードを検査する
ホストが汎用機・オフコンの場合	
／ＨｏｓｔＳｉｚｅ　２５６	ホストレコード長＝２５６バイト
ホストがＵｎｉｘの場合	
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ	データファイルに変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルに変換する
ホストがＷｉｎｄｏｗｓの場合	
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ	データファイルに変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式のファイルに変換する
／ＨｏｓｔＮｏＥＯＦ	ＥＯＦコードをつけない

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

ホスト形式の､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	衛生数（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、

```
/hs64 /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、

```
/hs64 /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、

```
/hs64 /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)コマンドでホスト形式に変換しても、うまく変換できているかどうか、わかりにくいものです。Ｗｉｎファイルエディタ（ＧＵＩ部）で、できたホスト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）＞

```
fp putdata c:planet.csv c:* /hs64 /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＞

```
fp getdata c:planet c:test /hs64 /dnc /map a2, k8, a10, a9, pdu4.3, zdu2,
 zds2.1, 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 ↓
fp /dc/ putdata c:planet.csv c:* /hs64 /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 ↓
fp /dc/ putdata c:planet.csv c:* ++plaputcs.p ↓
```

つぎのようなパラメータファイルに、

＜ＰＬＡＰＵＴＣＳ．Ｐ＞

```
/hostsize 64                -- ホストレコード長 ↓
/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
```

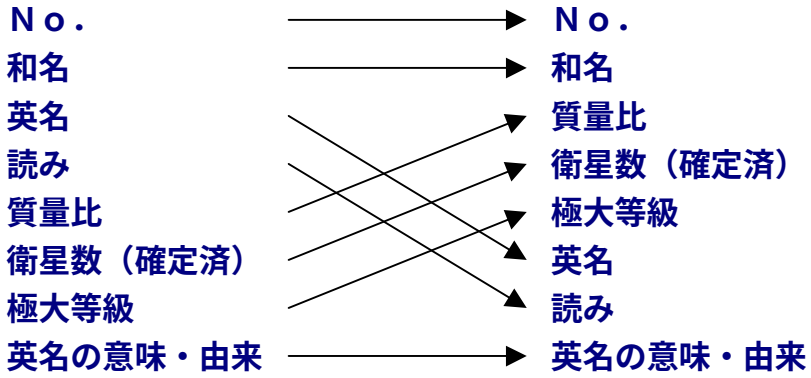
基本的にはコロンのあとに出力幅や出力ピクチャを書いていくだけです。デリミタ形式からの場合でも、

```
/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	衛生数（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	口神）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：ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイルＪＯＵＲＮＡＬに変換します。

```
C:\>fp putdata c:journal.dat c:* /hs128 ↓          (/HostSize 128)
```

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:\>fp putdata c:journal.dat c:* /hs128 /map km ↓ (/HostSize 128 /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。これをホストファイルに変換するとき、同時にレコード長を１７０バイトにします。

```
C:\>fp putdata c:zaiko*.dat c:* /hs170 ↓           (/HostSize 170)
```

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バイトをホストファイルに変換します。

```
C:\>fp putdata c:uriage.256 c:* /hs170 ↓           (/HostSize 170)
```

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Ｃ：のホストファイル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:\>fp putdata c:address.db c:* /hs128 /map k16 a34 k40 dyyyy-mm-dd:yymmdd a1 ↓
    (/HostSize 128 /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:\>fp putdata c:address.db c:* ++putaddr.p ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ＞

```
/hostsize 128                    -- ホストレコード長 ↓
/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。

```
fp putdata c:address.db c:* ++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:\>fp putdata c:address.db c:* /hs128 /map @90 a1 @0 k16 a34 k40
    dyyyy-mm-dd:yymmdd ↓      (/HostSize 128
          /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:\>fp putdata c:address.db c:* ++putaddr.p1 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ１＞

```
/hostsize 128                          -- ホストレコード長 ↓
/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   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:\>fp putdata c:address.db c:* /hs128 /map @:1 k16 a34 k40 dyyyy-mm-dd:yymmdd
    @:0 a1 ↓                  (/HostSize 128
          /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:\>fp putdata c:address.db c:* ++putaddr.p2 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ２＞

```
/hostsize 128                          -- ホストレコード長 ↓
/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：ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイルＡＤＤＲＥＳＳに変換します。

```
C:\>fp putdata c:address.db c:* /hs100 /d /map k:16, a:16, a:12, a:6, k:40,
    dyyyy-mm-dd:yymmdd, a1 ↓          (/HostSize 100 /Delimited
 /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＞

```
fp putdata c:address.db c:* ++putaddr2.p ↓
```

＜ＰＵＴＡＤＤＲ２．Ｐ＞

```
/hostsize 100                          -- ホストレコード長 ↓
/delimited                             -- ＣＳＶ形式から変換 ↓
/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:\>fp putdata c:address.db c:* /hs100 /dbtab /map k:16, a:16, a:12, a:6, k:40,
    dyyyy-mm-dd:yymmdd, a:1 ↓        (/HostSize 100 /Delimited By TAB
 /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:\>fp putdata c:address.db c:* /hs100 /dbsp /map k:16 a:16, a:12, a:6, k:40,
    dyyyy-mm-dd:yymmdd, a:1 ↓        (/HostSize 100 /Delimited By SPace
 /MAP Kanji:16, Ank:16, Ank:12, Ank:6, Kanji:40, Date yyyy-mm-dd:yymmdd, Ank:1)
```

■使用例１（プリント形式：ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）データファイル変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴは、プリント形式のファイルです。それをドライブＣ：のホストファイルＪＯＵＲＮＡＬに変換します。

```
C:\>fp putdata c:journal.dat c:* /hfd /map km ↓ (/HostFile Data /MAP KanjiMix)
```

例２）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バイトです。これをホストファイルに変換するとき、同時にレコード長を１７０バイトにします。

```
C:\>fp putdata c:zaiko*.dat c:* /hfr /hs170 /map km ↓
                                (/HostFile Random /HostSize 170 /MAP KanjiMix)
```

例３）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バイトをホストファイルに変換します。

```
C:\>fp putdata c:uriage.256 c:* /hfr /hs170 /map km ↓
                                (/HostFile Random /HostSize 170 /MAP KanjiMix)
```

例４）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Ｃ：のホストファイル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:\>fp putdata c:address.db c:* /hfr /hs128 /map k16 a34 k40 dyyyy-mm-dd:yymmdd
    a1 ↓                          (/HostFile Random /HostSize 128
                        /MAP Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd Ank1)
```

例５）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:\>fp putdata c:address.db c:* ++putaddr.p ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ＞

```
/hostfile random                -- ホストファイル＝ランダムファイル ↓
/hostsize 128                   -- ホストレコード長 ↓
/NonDelemited                   -- プリント形式からの変換 ↓
/map ↓
      kanji  16                 -- 氏名 ↓
      ank    16+12+6            -- フリガナ、電話番号、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６）バッチファイル化する

例５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。

```
fp putdata c:address.db c:* ++putaddr.p ↓
```

例７）入力桁移動を使って、項目を組み替える

例４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:\>fp putdata c:address.db c:* /hfr /hs128 /map @90 a1 @0 k16 a34 k40
    dyyyy-mm-dd:yymmdd ↓      (/HostFile Random /HostSize 128
          /MAP @96 Ank1 @0 Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd)
```

例８）例８の処理でパラメータファイルを使う

例７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:\>fp putdata c:address.db c:* ++putaddr.p1 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ１＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 128                           -- ホストレコード長 ↓
/map ↓
      @96  ank    1                     -- 区分 ↓
      @0   kanji  16                    -- 氏名 ↓
           ank    16+12+6               -- フリガナ、電話番号、郵便番号 ↓
           kanji  40                    -- 住所 ↓
           date   yyyy-mm-dd:yymmdd     -- 生年月日 ↓
```

例９）出力桁移動を使って、項目を組み替える

例４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:\>fp putdata c:address.db c:* /hfr /hs128 /map @:1 k16 a34 k40
    dyyyy-mm-dd:yymmdd @:0 a1 ↓   (/HostFile Random /HostSize 128
          /MAP @:1 Kanji16 Ank16+12+6 Kanji40 Date yyyy-mm-dd:yymmdd @:0 Ank1)
```

例 10）例９の処理でパラメータファイルを使う

例９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:\>fp putdata c:address.db c:* ++putaddr.p2 ↓
```

＜ＰＵＴＡＤＤＲ．Ｐ２＞

```
/hostfile random                        -- ホストファイル＝ランダムファイル ↓
/hostsize 128                           -- ホストレコード長 ↓
/map ↓
      @:1  kanji   16                   -- 氏名 ↓
           ank     16+12+6              -- フリガナ、電話番号、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：ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

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Ｃ：のホストファイルＡＤＤＲＥＳＳに変換します。

```
C:\>fp putdata c:address.db c:* /hfr /hs100 /d /map k:16, a:16, a:12, a:6,
    k:40, dyyyy-mm-dd:yymmdd, a1 ↓       (/HostFile random /HostSize 100
                                /Delimited /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＞

```
fp putdata c:address.db c:* ++putaddr2.p ↓
```

＜ＰＵＴＡＤＤＲ２．Ｐ＞

```
/hostfile random                          -- ホストファイル＝ランダムファイル ↓
/hostsize 100                             -- ホストレコード長 ↓
/delimited                                -- ＣＳＶ形式から変換 ↓
/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:\>fp putdata c:address.db c:* /hfr /hs100 /dbtab /map k:16, a:16, a:12, a:6,
    k:40, dyyyy-mm-dd:yymmdd, a:1 ↓      (/HostFile random /HostSize 100
                    (/Delimited By TAB /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:\>fp putdata c:address.db c:* /hfr /hs100 /dbsp /map k:16 a:16, a:12, a:6,
    k:40, dyyyy-mm-dd:yymmdd, a:1 ↓      (/HostFile random /HostSize 100
                  (/Delimited By SPace /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。

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→ホスト　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）をホストファイル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　ホストファイル　［オプション］］

■パラメータの説明

●Ｗｉｎｄｏｗｓファイル、ホストファイル

Ｗｉｎｄｏｗｓファイルとホスト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
／ＨｏｓｔＦｉｌｅ	ホストファイルの形式を指定する
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ	デリミタ形式に変換する
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ	プリント形式に変換する
／ＭＡＰ	項目別に変換方法の詳細を指定する

Ｐｕｔ系コマンド共通のオプション

／ＨｏｓｔＳｉｚｅ （／ＩＳｉｚｅ）	ホストファイルのレコード長を指定する
／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

／ＥＯＦ	**ＥＯＦコードを検査する**
／ＮｏＥＯＦ	**ＥＯＦコードを検査しない**
／ＨｏｓｔＥＯＦ	**ＥＯＦコードをつける**
／ＨｏｓｔＮｏＥＯＦ	**ＥＯＦコードをつけない**

これらのオプションのうち、／Ｓｉｚｅオプション、／ＨｏｓｔＦｉｌｅオプション、／ＨｏｓｔＤｅｌｉｍｉｔｅｄオプション、／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄオプション、／ＭＡＰオプション、／ＨｏｓｔＳｉｚｅ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。

＄　ホスト側（出力）のレコード長

ホストファイルのレコード長（／ＨｏｓｔＳｉｚｅ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、「ホスト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＝ホスト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。

／ＳｉｚｅオプションでＷｉｎｄｏｗｓレコード長を指定できることは、ＰｕｔＲａｎｄ(ｐｒ)コマンドの性格をよく反映しています。

このコマンドは、おもにＷｉｎｄｏｗｓ　ＣＯＢＯＬの順ファイルやＢＡＳＩＣのランダムファイルを変換することをねらったものである、ということはすでに述べたとおりです。そして、ＢＡＳＩＣ側の事情を考慮すると、レコード長を任意に指定できないと困ることが多いのです。

とくに困るのは、Ｎ８８ＢＡＳＩＣコンパイラ／インタプリタを使うときです。Ｎ８８ＢＡＳＩＣコンパイラ／インタプリタでは、１つのプログラム内で複数のレコード長のランダムファイルを扱うことができません。

もう１つの理由は、ゾーン形式とパック形式の変換、日付データの変換をサポートしているため、レコード長を増減する機能がないと困るからです。また、不要な項目がいくつもあるときや、予備領域が多すぎるときはレコード長を縮めたいものです。

◆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。

／ＨｏｓｔＦｉｌｅ ---- ホストファイルの形式を指定する

```
／ＨｏｓｔＦｉｌｅ　{Ｄａｔａ | Ｒａｎｄｏｍ}
／ＨｏｓｔＦｉｌｅ　Ｄａｔａ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、ホストファイルがデータファイルかランダムファイルかを指定します。

Ｄａｔａ指定をすると、ホストファイルをデータファイル（プリント／デリミタ形式）として扱います。

Ｒａｎｄｏｍ指定をすると、ホストファイルをランダムファイルとして扱います。

ホストが汎用機・オフコンの場合は、この指定は意味を持たなくなり、ランダムファイルとして扱います。

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

```
／ＨｏｓｔＤｅｌｉｍｉｔｅｄ ［Ｂｙ　ＣＯＭｍａ
                              Ｂｙ　ＴＡＢ
                              Ｂｙ　ＳＰａｃｅ］
                                  ［Ｃｏｍｐｒｅｓｓ
                                    ＮｏＣｏｍｐｒｅｓｓ］
／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ
```

ホストがＵｎｉｘまたはＷｉｎｄｏｗｓの場合、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。

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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

／ＨｏｓｔＮｏｎＤｅｌｉｍｉｔｅｄ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。

／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

／ＭＡＰ ---- 項目別に変換方法の詳細を指定する

／MAP	Ａｎｋ変換 Ｋａｎｊｉ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 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。

◆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｜＊］［：出力幅］

ＡＮＫ項目を変換します。ホストが汎用機・オフコンの場合、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。

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。
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、ホスト側の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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（出力）の残りバイト数

Ｋａｎｊｉ変換

```
Ｋａｎｊｉ　［入力幅｜＊］［：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。

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、ホスト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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（出力）の残りバイト数

ＫａｎｊｉＭｉｘ変換

ＫａｎｊｉＭｉｘ　［入力幅｜＊］［：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）。ホストが汎用機・オフコンの場合、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。

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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。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、ホスト側のユーザー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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系か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)。

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｜＊］［：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。

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。

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。

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、ホスト側のバイナリ項目のおわりに、ホストが汎用機・オフコンの場合はＮＵＬ（００Ｈ）が、ホストがＵｎｉｘ／Ｗｉｎｄｏｗｓの場合はスペース（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。

＄　ホスト側（出力）のレコード長
＊　ホスト側（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。通常は、ホストが汎用機・オフコンの場合に指定します。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければなりません（ふつう、セットアップ時に１回だけ行います）。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければなりません（ふつう、セットアップ時に１回だけ行います）。

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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］

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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］
```

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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。

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の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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。出力ピクチャは省略できます。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の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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。出力ピクチャは省略できます。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のＣＯＢＯＬのパック形式数値項目を、ホストのＣＯＢＯＬのゾーン形式数値項目に変換します。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。出力ピクチャは省略できます。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のＣＯＢＯＬのパック形式数値項目を、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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定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）。

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。出力ピクチャは省略できます。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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います)。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、

入力幅	出力幅
1	3
2	5
3	8
4	１０
5	１３
6	１５
7	１７
8	１８

左記の値を基本として、
符号つきなら＋１、ピクチャ指定の小数部があれば＋１とし、
さらに、ピクチャ指定のほうが大きければ、
整数部桁数に、符号つきなら＋１、小数部があれば＋１＋小数点桁数

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　ｉ４ｓ４．１　　という指定は、
ＢｉｎＤｉｓｐ　ｉ４ｓ４．１：１０　　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］

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。

入力ピクチャは省略できます。入力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］

Ｗｉｎｄｏｗｓ　ＣＯＢＯＬのパック形式数値項目を、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。

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。

入力ピクチャは省略できます。入力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］
```

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。出力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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］

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。ホストが汎用機・オフコンの場合、あらかじめ、変換設定のＡＮＫコードの設定で、ＥＢＣＤＩＣ系かＪＩＳ８／ＡＳＣＩＩ系かの設定をしておかなければいけません（ふつう、セットアップ時に一回だけ行います）。

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。

出力ピクチャは省略できます。出力ピクチャを省略すると「２進キャスト／ピクチャと同じ」とみなされます。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
```

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、ホスト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。

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）の交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。コード変換は、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。

デリミタ挿入（コンマ）

，

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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。ＰｕｔＤａｔａ(ｐｄ)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、／ＨｏｓｔＤｅｌｉｍｉｔｅｄ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。

／ＨＤｅｌｉ／ＭＡＰ　・・・，ＤＺＵ５，ＤＺＳ３．２，・・・

引用符くくり

“～”

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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。ただし、プリント形式への変換のときは、この引用符くくりの機能は無効になります。

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場合を示すと、つぎのような指定になります。

／ＨＤｅｌｉ／ＭＡＰ　・・・，ＺＤＵ１０，“Ｋ２０”，ＰＤＵ３，・・・

◆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
```

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、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、出力がプリント形式データファイルのときは、この改行コード挿入の機能は無効になります。

出力がデータファイルのときに、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。

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

改行コード挿入２

```
！Ｎ
```

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合に、ホスト側（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、出力がプリント形式データファイルのときに改行コードを挿入できることです。

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ホスト側（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。

＄　　ホスト側（出力）のレコード長
．　　ホスト側（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＊ＴＲＡＮ＋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　・・・　^＊**　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、ホスト側（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、そのなかでは、つぎの特殊変数が使えます。

＊　ホスト側（出力）の残りバイト数

●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＝ホストレコード長**
／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**
／ＨｏｓｔＳｉｚｅ　２５６	**ホストレコード長＝２５６バイト**
／ＮｏＱｕｅｒｙ	**ファイル名の確認なしで自動変換する**

ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合

／ＨｏｓｔＦｉｌｅ　Ｒａｎｄｏｍ	**ランダムファイルに変換する**

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オプションの指定の方法

ホスト形式の、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	衛生数（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、

```
/s64 /hs64 /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
ホスト側（出力）レコード長は／ＨｏｓｔＳｉｚｅ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)コマンドでホスト形式に変換しても、うまく変換できているかどうかわかりにくいものです。Ｗｉｎファイルエディタ（ＧＵＩ部）で、できたホスト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）＞

```
fp putrand c:planet.ran c:* /s64 /hs64 /map a2 k8 a10 a9 dp8:u4.3 dz2:u2
 dz5:s2.1 k20 ↓
```

逆変換用バッチファイルを作り、

＜ＰＬＡＧＥＴＣＳ．ＢＡＴ＞

```
fp getdata c:planet test /hs64 /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
fp /dc/ putrand c:planet.ran c:* /s64 /hs64 /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 ↓
fp /dc/ putrand c:planet.ran c:* ++plaputrn.p ↓
```

つぎのようなパラメータファイルに、

＜ＰＬＡＰＵＴＲＮ．Ｐ＞

```
/size 64 ↓
/hostsize 64 ↓
/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

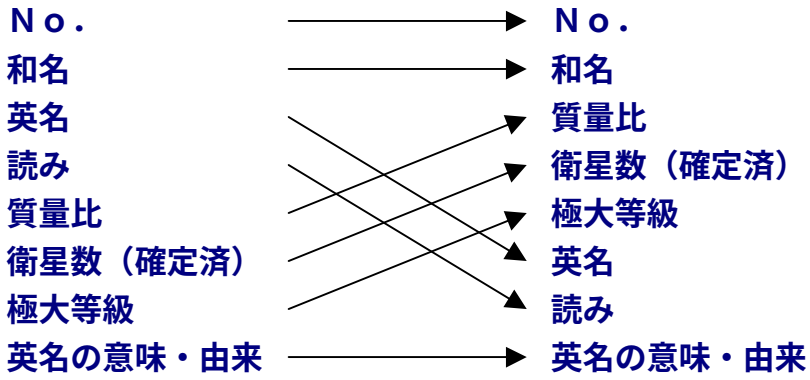
必要に応じて項目長を増減するのも簡単です。つぎのように、

```
/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（ホストが汎用機・オフコンの場合）

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Ｃ：のホストファイル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:\>fp putrand c:journal.dat c:* /hs256 ↓          (/HostSize 256)
```

例２）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Ｃ：のホストファイルＦＯＮＴ２４にバイナリ変換します。

```
C:\>fp putrand c:font24.pxl c:* /hs256 /map b ↓    (/HostSize 256 /MAP Binary)
```

例３）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Ｃ：のホストファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２に変換します。

```
C:\>fp putrand c:zaiko*.dat c:* /s256 /hs80 ↓      (/Size 256 /HostSize 80)
```

例４）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Ｃ：のホストファイル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:\>fp putrand c:address.db c:* /hs256 /map k16 a34 k40 dyyyy-mm-dd:yymmdd a1 ↓
        (/HostSize 256 /MAP Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd Ank1)
```

例５）項目を組み替える（その１）

例４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:\>fp putrand c:address.db c:* /hs256 /map @96 a1 @1 k16 a34 k40
     dyyyy-mm-dd:yymmdd ↓
   (/HostSize 256 /MAP @96 Ank1 @1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd)
```

例６）項目を組み替える（その２）

例４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:\>fp putrand c:address.db c:* /hs256 /map @:1 k16 a34 k40 dyyyy-mm-dd:yymmdd
     @:0 a1 ↓
  (/HostSize 256 /MAP @:1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd @:0 Ank1)
```

例７）パラメータファイルを利用する

例４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:\>fp putrand c:address.db c:* ++addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/hostsize 256                       -- ホストレコード長 ↓
/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 ↓
```

■使用例（ホストがＵｎｉｘ、Ｗｉｎｄｏｗｓの場合）

例１）ランダムファイル変換する

ドライブＣ：のＷｉｎｄｏｗｓファイルＪＯＵＲＮＡＬ．ＤＡＴを、ドライブＣ：のホストファイルＪＯＵＲＮＡＬに変換します。レコード長は２５６バイトです。

```
C:\>fp putrand c:journal.dat c:* /hfr /hs256 /map km ↓
                                    (/HostFile Random /HostSize 256 /MAP KanjiMix)
```

例２）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Ｃ：のホストファイルＦＯＮＴ２４にバイナリ変換します。

```
C:\>fp putrand c:font24.pxl c:* /hfr /hs256 /map b ↓
                                    (/HostFile Random /HostSize 256 /MAP Binary)
```

例３）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。これをドライブＣ：のホストファイルＺＡＩＫＯ０１～ＺＡＩＫＯ１２に変換します。

```
C:\>fp putrand c:zaiko*.dat c:* /hfr /s256 /hs80 /map km ↓
                                    (/HostFile Random /Size 256 /HostSize 80 /MAP KanjiMix)
```

例４）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Ｃ：のホストファイル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:\>fp putrand c:address.db c:* /hfr /hs256 /map k16 a34 k40
    dyyyy-mm-dd:yymmdd a1 ↓    (/HostFile Random /HostSize 256
                        /MAP Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd Ank1)
```

例５）項目を組み替える（その１）

例４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:\>fp putrand c:address.db c:* /hfr /hs256 /map @96 a1 @1 k16 a34 k40
    dyyyy-mm-dd:yymmdd ↓      (/HostFile Random /HostSize 256
                  /MAP @96 Ank1 @1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd)
```

例６）項目を組み替える（その２）

例４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:\>fp putrand c:address.db c:* /hfr /hs256 /map @:1 k16 a34 k40
    dyyyy-mm-dd:yymmdd @:0 a1 ↓    (/HostFile Random /HostSize 256
                 /MAP @:1 Kanji16 Ank34 Kanji40 Date yyyy-mm-dd:yymmdd @:0 Ank1)
```

例７）パラメータファイルを利用する

例４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:\>fp putrand c:address.db c:* ++addr.p ↓
```

＜ＡＤＤＲ．Ｐ＞

```
/hostfile random                    -- ホストファイル＝ランダムファイル ↓
/hostsize 256                       -- ホストレコード長 ↓
/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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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＊ＴＲＡＮ＋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：のホスト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:\>fp 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 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:\>fp ank ebcdickana ; csave . ↓
```

２．13 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：¥>ｆｐ　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：¥>ｆｐ　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:\>fp 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:\>fp /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:\>fp ank ebcdickana ; kanji keis ; csave . ↓
```

２．14 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
コメントは「日立のシステム用」

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:\>fp ank ebcdickana ; kanji keis ; comment 日立のシステム用 ; csave hita ↓
```

２．15　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：］［パス名指定］［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。また、元のコード変換表を読み込んで修正を取り消したいときにはピリオド（.）を指定してください。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。

なお、現在のコード変換表は設定表示バー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:\>fp cload i ; getdata ･･･ （ファイル変換）　↓
```

２．16　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：］［パス名指定］［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。また、元のコード変換表に書き込みたいときは、ピリオド（.）を指定してください。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。

なお、現在のコード変換表は設定表示バーを見ればわかります。

■使用例

例）日立のシステムとのデータ変換

日立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)、コメントを「日立用」に設定します。

```
C:\>fp kanji keis ; ank ek ; comment 日立用 ; csave . ↓
```

―参 考―

他のアプリケーションからの利用

■Visual Basic（Access Basic）から利用するには

Ｖｉｓｕａｌ　Ｂａｓｉｃ（Ａｃｃｅｓｓ　Ｂａｓｉｃ）から直接Ｆ＊ＴＲＡＮ＋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。

●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４以上で動作します（バージョン４の形式で作成されています）。Ｆ＊ＴＲＡＮ＋をインストールしたＶｂｓａｍｐｌｅディレクトリの中につぎの２つのファイルがありますので、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

バージョン５以上のＶｉｓｕａｌ　Ｂａｓｉｃをお使いの場合は、プロジェクトを使用する時にワーニングダイアログが出力されますが、ＯＫボタンをクリックすれば問題ありません。そして、保存することにより現行のバージョンの形式に更新されます。

Ａｃｃｅｓｓ　Ｂａｓｉｃから利用する場合は、 **ｓｍｐｌｃｏｄｅ．ｔｘｔ** の内容をカット＆ペーストして使用してください。

●サンプルプログラムの内容

このサンプルプログラムは、４３ページのＴＥＳＴ．ＢＡＴの内容をＢａｓｉｃ言語で記述し、Ｆ＊ＴＲＡＮ＋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。

①**Form 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 OpenProcess Lib "kernel32" _
  (ByVal dwDesiredAccess As Long, ByVal bInheritHandle As Long, _
   ByVal dsProcessId As Long) As Long
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 CreateProcess Lib "kernel32" Alias "CreateProcessA" _
  (ByVal lpApplicationName As _String, ByVal lpCommandLine As String, _
   ByVal lpProcessAttributes As Long, ByVal lpThreadAttributes As Long, _
   ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long, _
   lpEnvironment As String, _ ByVal lpCurrentDriectory As String, _
   lpStartupInfo As STARTUPINFO, lpProcessInformation As PROCESS_INFORMATION) _
   As Long

Private Type STARTUPINFO
        cb As Long
        lpReserved As String
        lpDesktop As String
        lpTitle As String
        dwX As Long
        dwY As Long
        dwXSize As Long
        dwYSize As Long
        dwXCountChars As Long
        dwYCountChars As Long
        dwFillAttribute As Long
        dwFlags As Long
        wShowWindow As Integer
        cbReserved2 As Integer
        lpReserved2 As Long
        hStdInput As Long
        hStdOutput As Long
        hStdError As Long
End Type

Private Type PROCESS_INFORMATION
        hProcess As Long
        hThread As Long
        dwProcessId As Long
        dwThreadId As Long
End Type

Private Const PROCESS_QUERY_INFORMATION = &H400&
Private Const STILL_ACTIVE = &H103&
```

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 path As String) As Long
    Dim ret As Long
    Dim StartInfo As STARTUPINFO
    Dim ProcInfo As PROCESS_INFORMATION
    Dim cmdLine As String

    cmdLine = path & "\" & command
    StartInfo.cb = Len(StartInfo)
    ret = CreateProcess(vbNullString, cmdLine, 0&, 0&, _
                        0&, 0, vbNullString, path, StartInfo, ProcInfo)

    ProcessExec = ProcInfo.dwProcessId
End Function

Public Function ProcessWait(id As Long) As Long
  Dim hProcess As Long, ExitCode As Long
  Dim ret As Long

  hProcess = OpenProcess(PROCESS_QUERY_INFORMATION, 1, id)
  Do
    ret = GetExitCodeProcess(hProcess, ExitCode)
    DoEvents
  Loop While (ExitCode = STILL_ACTIVE)
  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

```
' 変換が正常に終了したら、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*TRAN+ のインストールディレクトリに設定する
    path = "c:\FtranP"              '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

' カレントディレクトリにあるサンプルファイル「planet」をデータファイル変換
    IDProcess = ProcessExec("fp.exe /nwd /wc/ getdata planet *.txt ++pngetprn.p",
path)
    ExitCode = ProcessWait(IDProcess)
    Open "planet.txt" For Input As #1
    Close #1

    Call Shell("command.com /K type planet.txt", vbNormalFocus)  'NT では cmd.exe

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＊ＴＲＡＮ＋を起動して、バッチ処理を実行することができます。Ｆ＊ＴＲＡＮ＋をインストールしたＶｃｓａｍｐｌｅディレクトリの中に、つぎの２つ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
Ｓｍｐｌｃｏｄｅ．ｄｏｃ	サンプルプログラムのドキュメント

以下、サンプルプログラムのソースの内容です。ドキュメントを参考にしてください。

```
void CFpSampleDlg::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*TRAN のインストールディレクトリに設定する
    _chdir("c:\\ftranp\\");

    // planet を変換して、その結果を表示する
    IdProcess = ProcessExec("fp.exe /nwd /wc/ getdata planet planet.get ++pngetprn.p");
    exitCode = ProcessWait(IdProcess);
    if(exitCode != 0) {
        MessageBox(NULL," 変 換 に 失 敗 し ま し た 。 ","F*TRAN  サ ン プ ル
",MB_OK|MB_ICONEXCLAMATION);
        return(1);
    }

    WinExec("cmd.exe /K type planet.get",SW_SHOW);  // Windows95/98 では command.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:\\ftranp");
    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);
}
```

索　　引

Ｆ＊ＴＲＡＮ＋　Ｖ３．０　　操作説明書・コマンド編

２０００年　１０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　　〒108-8531　東京都港区芝浦 4－15－33 芝浦清水ビル
ﾌﾟﾛﾀﾞｸﾂ&ｻｰﾋﾞｽ事業部　TEL 03－5445－2101
FAX 03－5445－2109
