FUJITSU

F*TRAN+ Ver.8.0

操作説明書／マルチレコード編

第１版

株式会社富士通ビー・エス・シー

はじめに

Ｆ＊ＴＲＡＮ＋　Ｖ８．０をお買い上げいただき、ありがとうございます。

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。
発売以来、多くのユーザにささえられている、実績あるソフトウェアです。

Ｆ＊ＴＲＡＮ＋　Ｖ８．０には、動作するＯＳ別に２種類の製品があります。

- ■　Ｆ＊ＴＲＡＮ＋　Ｖ８．０　Ｓｅｒｖｅｒ
  Ｓｅｒｖｅｒ系、およびＤｅｓｋｔｏｐ系ＯＳで動作
- ■　Ｆ＊ＴＲＡＮ＋　Ｖ８．０　Ｄｅｓｋｔｏｐ
  Ｄｅｓｋｔｏｐ系ＯＳで動作

Ｆ＊ＴＲＡＮ＋　Ｖ８．０では、Ｗｉｎｄｏｗｓ１０上での動作をサポートしています。

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

２０１７年１月
株式会社 富士通ビー･エス･シー

Copyright 1997-2017 株式会社 富士通ビー･エス･シー

目　　次

第１章　　Ａｔｌａｓ９８とは？

第２章　　Ａｔｌａｓ９８の文法

第３章　　Ａｔｌａｓ９８の使用法

第４章　　Ａｔｌａｓ９８の利用例

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／B／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 B 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。
a	下線部を入力します。
ａｂｃ↓	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

Ａｔｌａｓ９８とは？

１．１　Ａｔｌａｓ９８とは？

ＣＯＢＯＬのプログラムでは、よくＯＣＣＵＲＳ句を使ってくり返し項目をまとめたり、ＲＥＤＥＦＩＮＥＳ句を使って条件によって項目の定義を変えたりします。順ファイル処理ではごく普通の手法だといってよいでしょう。

従来のＦ＊ＴＲＡＮ＋でこれらに対応するのは大変でした。まず、くり返し項目を記述する方法がなく、必要回数マップの指定をくり返して書いてもらっていました。マルチレコードレイアウトはもっと大変で、レコードタイプ毎に別々に変換してから別プログラムで再合成してもらう、などの方法をとっていました。Ｆ＊ＴＲＡＮ＋だけでは変換できなかったのです。また、一部のレコードだけ抜き出して変換したいとか、逆に一部のレコードをスキップして変換したいなどの要望もありましたが、これも無理でした。

これらの問題を解決すべく生まれたのが、Ａｔｌａｓ９８（アトラス９８）です。これは、今までの ／ＭＡＰオプション（マップ設定）の拡張言語で、くり返しを記述する「Ｌｏｏｐ文」、多分岐を記述する「Ｓｅｌｅｃｔ文」、処理をまとめる「手続き」があります。この３つを利用することで、ほとんどのパターンの順ファイルが変換可能になったといっても過言ではありません。

Ａｔｌａｓ９８は、マップ設定（／ＭＡＰオプション）の拡張言語です。つまり、マップ設定（／ＭＡＰオプション）では、Ａｔｌａｓ９８言語を使わずに、従来のパラメータをそのまま指定することもできます。つまり、従来のマップ設定（／ＭＡＰオプション）との上位互換が保たれています。そして、Ａｔｌａｓ９８を使用することにより、より拡張された機能を使うことができます。

１．２　何ができるようになるか？

Ａｔｌａｓ９８の導入によって、以下のことが実現できるようになります。

■くり返し項目が記述できる。

■集団項目のくり返しや、多重にネストしたくり返しもできる。

■項目の内容に応じて、レイアウトを変えることができる。

■項目の内容に応じ、その項目を変換し分けることができる。

■条件に合ったレコードを抽出して変換することができる。

■条件に合ったレコードを除外して変換することができる。

■特定のレコード／項目で、変換を中断することができる（レコード単位、ファイル単位)。

■ヘッダレコードなど、特定位置のレコードを別レイアウトで変換することができる。

■ヘッダレコード、トレーラレコード、エンドレコードなどを削除することができる。

■レコード番号指定で、特定範囲のレコードだけ変換することができる。

■一定のサイクルで同じパターンのレコードがあるファイルを変換することができる。

■変換結果にヘッダ、トレーラを付加することができる。

■ファイルタイプ別の変換ができる。

第2章

Ａｔｌａｓ９８の文法

２．１　Ａｔｌａｓ９８の構文

■Ａｔｌａｓ９８の構文

Ａｔｌａｓ９８には、つぎの１３の文があります。

Ａｔｌａｓ	--->	Ａｔｌａｓ宣言
Ｐｒｏｃ	--->	手続き開始
ＥｎｄＰｒｏｃ	--->	手続き終了
Ｌｏｏｐ	--->	くり返し開始
ＥｎｄＬｏｏｐ	--->	くり返し終了
Ｓｅｌｅｃｔ	--->	多分岐開始
Ｃａｓｅ	--->	条件指定
ＥｌｓｅＣａｓｅ	--->	その他条件
ＥｎｄＳｅｌｅｃｔ	--->	多分岐終了
Ｃａｌｌ	--->	手続きの呼び出し
Ｒｅｔｕｒｎ	--->	手続きからの復帰
Ｂｒｅａｋ	--->	構造からの抜け出し
Ｑｕｉｔ	--->	処理の中断

Ａｔｌａｓ９８には、くり返しを記述するための「Ｌｏｏｐ文」、多分岐を記述するための「Ｓｅｌｅｃｔ文」、処理をまとめるための「手続き」があります。とくにＳｅｌｅｃｔ文は、Ｐａｓｃａｌ言語のｃａｓｅ文、Ｃ言語のｓｗｉｔｃｈ文、Ｖｉｓｕａｌ　Ｂａｓｉｃのｓｅｌｅｃｔ　ｃａｓｅ文などに似ていますが、はるかに強力な構文として、Ａｔｌａｓ９８の中核機能をなしています。

Ａｔｌａｓ９８は、構造化された言語です。Ａｔｌａｓ９８には、Ｇｏｔｏ文がありません。その代わり、柔軟な脱出機構があります（Ｂｒｅａｋ文、Ｒｅｔｕｒｎ文、Ｑｕｉｔ文)。

Ａｔｌａｓ９８は、ＣＯＢＯＬの仕様である最大１０進１８桁の数値の扱いを保証しています。

◆注意　----　Ａｔｌａｓ構文を指定する時の決まり事（コマンド型での指定）

・Ａｔｌａｓ構文は、Ｍａｐ文との区別のため、ブレース（{と}）でくくる必要があります。
・Ａｔｌａｓ構文のキーワードは短縮できません。
・Ａｔｌａｓ構文のキーワードの大文字、小文字は区別されません。

２．２　Ａｔｌａｓ９８の文法

■Ａｔｌａｓ文（Ａｔｌａｓ宣言）

Ａｔｌａｓ構文の使用を宣言します。

```
{Ａｔｌａｓ　９８}
```

マップ設定（／ＭＡＰオプション）にＡｔｌａｓ構文を指定する場合は、必ずＡｔｌａｓ宣言から始めます。

■Ｐｒｏｃ文（手続き開始）

ＥｎｄＰｒｏｃ文（手続き終了）

Ｐｒｏｃ構造は、複数の文（Ｍａｐ文、Ａｔｌａｓ文）をまとめ、１つのまとまった手続きを構築します。

```
{手続き名：Ｐｒｏｃ}
　　Ｍａｐ文　または　Ａｔｌａｓ文
{ＥｎｄＰｒｏｃ}
```

マップ設定（／ＭＡＰオプション）にＡｔｌａｓ構文を指定する場合は、Ａｔｌａｓ宣言をした後に、１つないし複数の手続きを指定します。手続きは再帰呼び出しができますが、入れ子にはできません。

１番最初の手続きが自動的に主手続きになります。Ａｔｌａｓ指定の実行は、常に主手続きの先頭から始まり、主手続きから復帰時に終了します。

手続き名は、英大文字（Ａ～Ｚ）、英小文字（ａ～ｚ）、数字（０～９）、全角文字で構成されます。ただし、先頭が数字であってはいけません。また、全角スペースを含んではいけません。英大文字と英小文字は区別されません。手続き名の長さは、１６文字以内で指定します。２重定義はできません。手続き名は省略できません。手続き名は任意ですが、主手続きに与える手続き名は習慣的に「ｍａｉｎ」または「メイン」を用います。

ＥｎｄＰｒｏｃ文で、手続きを終了します。

■Ｌｏｏｐ文（くり返し開始）

ＥｎｄＬｏｏｐ文（くり返し終了）

指定回数のくり返し構造を構築します。

```
{[構造名：]Ｌｏｏｐ　回数}
　　Ｍａｐ文　または　Ａｔｌａｓ文
{ＥｎｄＬｏｏｐ}
```

構造名は、英大文字（Ａ～Ｚ）、英小文字（ａ～ｚ）、数字（０～９）、全角文字で構成されます。ただし、先頭が数字であってはいけません。また、全角スペースを含んではいけません。英大文字と英小文字は区別されません。構造名の長さは、１６文字以内で指定します。２重定義はできません。構造名は省略できます。

Ｌｏｏｐ文で、くり返しの回数を指定します。整数値（０～６５５３５）、１６進定数値（００Ｈ～ｆｆｆｆＨ）、またはシステム変数を指定することができます。くり返しの回数は省略できません。

システム変数は、つぎの５つです。

ＳｙｓＰｈａｓｅ	フェーズ（Ｏｐｅｎ＝１、Ｍａｉｎ＝２、Ｃｌｏｓｅ＝８）
ＳｙｓＲｅｃＮｕｍ	レコード番号（０～）
ＳｙｓＲｅｔｕｒｎ	帰り値（Ｒｅｔｕｒｎ文で代入される）
ＳｙｓＢｒｅａｋ	ブレーク値（Ｂｒｅａｋ文で代入される）
ＳｙｓＱｕｉｔ	クイット値（Ｑｕｉｔ文で代入される）

ＥｎｄＬｏｏｐ文で、くり返し指定を終了します。

■Ｓｅｌｅｃｔ文（多分岐開始）

Ｃａｓｅ文（条件指定）

ＥｌｓｅＣａｓｅ文（その他条件）

ＥｎｄＳｅｌｅｃｔ文（多分岐終了）

多分岐構造を構築します。Ｓｅｌｅｃｔ構造の主な使い方は、マルチレイアウト変換への適用と、条件つき変換（抽出変換、除外変換）です。

```
{[構造名：]Ｓｅｌｅｃｔ　選択型}
　　セレクタ（Ｍａｐ文）
　{Ｃａｓｅ　範囲並び１}
　　Ｍａｐ文　または　Ａｔｌａｓ文
　[{Ｃａｓｅ　範囲並び２}
　　Ｍａｐ文　または　Ａｔｌａｓ文]…
　[{ＥｌｓｅＣａｓｅ}
　　Ｍａｐ文　または　Ａｔｌａｓ文]
{ＥｎｄＳｅｌｅｃｔ}
```

構造名は、英大文字（Ａ～Ｚ）、英小文字（ａ～ｚ）、数字（０～９）、全角文字で構成されます。ただし、先頭が数字であってはいけません。また、全角スペースを含んではいけません。英大文字と英小文字は区別されません。構造名の長さは、１６文字以内で指定します。２重定義はできません。構造名は省略できます。

Ｓｅｌｅｃｔ文の選択型の指定は、つぎの４つです

Ｉｎｔ	整数型（Ｉｎｔｅｇｅｒ）
Ｄｅｃ	小数型（Ｄｅｃｉｍａｌ）
Ｓｔｒ	文字列型（Ｓｔｒｉｎｇ）
Ｂｉｎ	バイナリ型（Ｂｉｎａｒｙ）

セレクタとは、多分岐の基準になる項目のことで、各Ｃａｓｅ値と比較されます。有効なＭａｐ文を１つ指定します。

セレクタに指定したＭａｐ文の変換結果は自動的に捨てられ、その変換はなかったことにされます。もし、変換結果も必要なら、同じＭａｐ文をＣａｓｅ文やＥｌｓｅＣａｓｅ文の後に再度、指定します。

選択型と有効なセレクタの組み合わせは、下表の組み合わせになります。

Ｍａｐ文	Ｉｎｔ　／　Ｄｅｃ	Ｓｔｒｉｎｇ	Ｂｉｎａｒｙ
DispZone／ZoneDisp	○	×	×
DispPack／PackDisp	○	×	×
DispBin ／BinDisp	○	×	×
ZonePack／PackZone	○	×	×
ZoneBin ／BinZone	○	×	×
PackBin ／BinPack	○	×	×
ZoneZone／PackPack	○	×	×
BinBin	○	×	×
Ｎｕｍｅｒｉｃ　＊	×	×	×
Ａｎｋ	×	○	×
Ｋａｎｊｉ	×	○	×
ＫａｎｊｉＭｉｘ	×	○	×
Ｂｉｎａｒｙ	×	×	○
ＢｉｎａｒｙＸ	×	×	○
ＢＹ	○	×	×

＊）Ｎｕｍｅｒｉｃ変換は数値比較も文字列比較もできない。

Ｃａｓｅ文は、変換する条件を指定します。Ｓｅｌｅｃｔ文には、少なくとも１つのＣａｓｅ文を含まなければなりません。

```
{Ｃａｓｅ　範囲並び}
```

範囲並びは、Ｃａｓｅ文を実行すべき値、あるいは値の範囲を１つ、または、コンマで区切って複数指定します。範囲はA.. Ｂの形式で示し、両端の値を範囲に含みます。たとえば、「A.. Ｂ，Ｃ，Ｄ」は「AからＢ、またはC、またはD」を意味します。

整数型の範囲並び指定

[-]ｎ	((-)はマイナス符号、ｎは最大１８桁)
０｛Ｘ｜ｘ｝ｈ…	ｈは０～９、Ａ～Ｆ（ａ～ｆ）で最大１６桁（８バイト）
Ｚｅｒｏ	０
Ｍｉｎ	最小値
Ｍａｘ	最大値

小数型の範囲並び指定

[-]ｎ．ｍ	((-)はマイナス符号、ｎとｍは合わせて最大１８桁)
Ｚｅｒｏ	０．０
Ｍｉｎ	最小値
Ｍａｘ	最大値

文字列型の範囲並び指定

’文字列’	文字列は最大２５６文字で、半角(')でくくる
	※(')，(*)，(?)は(¥')，(¥*)，(¥?)で表し、(¥)自身は(¥¥)で表す
Ｓｐａｃｅ	空白文字列（半角、全角の空白からなる文字列)
ＬｏｗＶａｌｕｅ	００Ｈ…の文字列
ＨｉｇｈＶａｌｕｅ	ＦＦＨ…の文字列 *

バイナリ型の範囲並び指定

｛Ｘ｜ｘ｝‘ｈｈ…’	ｈは０～９、Ａ～Ｆ（ａ～ｆ)、２桁で１バイト
	任意のバイト境界に半角スペースを挿入できる
	半角(')でくくる、最大２５６バイト
ＬｏｗＶａｌｕｅ	００Ｈ…の文字列
ＨｉｇｈＶａｌｕｅ	ＦＦＨ…の文字列 *

文字列の比較は、ホスト側文字列であれば、Ｗｉｎ側文字列に変換した文字列の比較を行います。基本的にＪＩＳ８／シフトＪＩＳコード順で大小比較をしますが、連続する空白はＣａｓｅ値、セレクタ値ともに １ 個の空白にまとめてから比較します。文字の大小関係はつぎのとおりです。

LowValue ＜ ANK 制御文字(01～1FH) ＜ 空白 ＜ ANK ＜ 全角文字(空白除く) ＜ HighValue

ＥｌｓｅＣａｓｅ文は、その他の場合の変換方法を指定します。省略できます。
ＥｎｄＳｅｌｅｃｔ文で、多分岐指定を終了します。

*) ＡＮＫ変換表をＦＦ→ＦＦに変更する必要がある。

■ＢＹ変換

Ｍａｐ文の一種として、ＢＹ変換が新設されました。ＢＹ変換は、与えられたＭａｐ式の値をセレクタ値に変換します。Ｍａｐ式の中には「~システム変数名」の形でシステム変数を記述することができます。これによって、システム変数の値によって分岐することができます。

ＢＹ　Ｍａｐ式（システム変数）

指定できるシステム変数は、つぎの５つです。

~ＳｙｓＰｈａｓｅ	フェーズ（Ｏｐｅｎ＝１、Ｍａｉｎ＝２、Ｃｌｏｓｅ＝８）
~ＳｙｓＲｅｃＮｕｍ	レコード番号（０～）
~ＳｙｓＲｅｔｕｒｎ	返り値（Ｒｅｔｕｒｎ文で代入された値）
~ＳｙｓＢｒｅａｋ	ブレーク値（Ｂｒｅａｋ文で代入された値）
~ＳｙｓＱｕｉｔ	クイット値（Ｑｕｉｔ文で代入された値）

指定例　：　ＢＹ　~ＳｙｓＲｅｃＮｕｍ￥￥４　　レコード番号を４で割った余り値
（セレクタ値は０～３の値になる）

■Ｃａｌｌ文（手続きの呼び出し）

手続きを呼び出します。

｛Ｃａｌｌ　手続き名｝

手続き名は、Ｐｒｏｃ文で指定した手続き名を指定します。

■Ｒｅｔｕｒｎ文（手続きからの復帰）

手続きから復帰します。

｛Ｒｅｔｕｒｎ　[整数値｜１６進定数値｜システム変数名]｝

返り値として、整数値（－３２７６８～３２７６７）、１６進定数値（００Ｈ～ｆｆｆｆＨ）または、システム変数を返すことができます。返り値を省略すると、返り値は不定になります。

システム変数は、つぎの５つです。

ＳｙｓＰｈａｓｅ	フェーズ（Ｏｐｅｎ＝１、Ｍａｉｎ＝２、Ｃｌｏｓｅ＝８）
ＳｙｓＲｅｃＮｕｍ	レコード番号（０～）
ＳｙｓＲｅｔｕｒｎ	返り値（Ｒｅｔｕｒｎ文で代入される）
ＳｙｓＢｒｅａｋ	ブレーク値（Ｂｒｅａｋ文で代入される）
ＳｙｓＱｕｉｔ	クイット値（Ｑｕｉｔ文で代入される）

実行時に｛ＥｎｄＰｒｏｃ｝に達した時は、｛Ｒｅｔｕｒｎ　０｝が実行されたとみなされます。

■Ｂｒｅａｋ文（構造からの抜け出し）

Ｌｏｏｐ構造、Ｓｅｌｅｃｔ構造から抜け出します。

{Ｂｒｅａｋ　[構造名][，整数値｜１６進定数値｜システム変数名]}

構造名を省略すると、そのＢｒｅａｋ文が直接属しているＬｏｏｐ構造／Ｓｅｌｅｃｔ構造を抜け出し、構造名を指定すると、指定されたＬｏｏｐ構造／Ｓｅｌｅｃｔ構造を抜け出します。

その時にブレーク値を指定することができます。ブレーク値は、手続きが開始時に自動的に０にセットされます。Ｂｒｅａｋ文では、ブレーク値として整数値（－３２７６８～３２７６７）、１６進定数値（００H～ｆｆｆｆH）、または、システム変数の値を指定します。これを、その後のＳｅｌｅｃｔ文で判定することができます。なお、ブレーク値はＣａｌｌ文の実行に影響されません。

システム変数は、つぎの５つです。

ＳｙｓＰｈａｓｅ	フェーズ（Ｏｐｅｎ＝１、Ｍａｉｎ＝２、Ｃｌｏｓｅ＝８）
ＳｙｓＲｅｃＮｕｍ	レコード番号（０～）
ＳｙｓＲｅｔｕｒｎ	返り値（Ｒｅｔｕｒｎ文で代入される）
ＳｙｓＢｒｅａｋ	ブレーク値（Ｂｒｅａｋ文で代入される）
ＳｙｓＱｕｉｔ	クイット値（Ｑｕｉｔ文で代入される）

■Ｑｕｉｔ文（処理の中断）

処理を中断します。

```
{Ｑｕｉｔ [ＲｅｃＳｋｉｐ｜ＲｅｃＤｏｎｅ]
        [, Ｃｏｎｔｉｎｕｅ｜ＥＯＦ]
        [, 整数値｜１６進定数値｜システム変数名]}　＊
```

その時に、そのレコードの処置、継続／中断、クイット値を指定することができます。

ＲｅｃＳｋｉｐ	処理中のレコードの処理をスキップします。 そのレコードのそこまでの変換結果は放棄されます。
ＲｅｃＤｏｎｅ	処理中のレコードの処理を完了します。 そのレコードのそこまでの変換結果は有効になります。
Ｃｏｎｔｉｎｕｅ	そのファイルの処理を継続します。
ＥＯＦ	そのファイルの処理を中断・完了します。 そのファイルのそこまでの変換結果は有効になります。

*) ＲｅｃＳｋｉｐ／ＲｅｃＤｏｎｅとＣｏｎｔｉｎｕｅ／ＥＯＦは順番を逆に書くこともできます。

クイット値は、全体の開始時（Ｏｐｅｎの直前）に自動的に０にセットされます。Ｑｕｉｔ文では、クイット値として整数値（－３２７６８～３２７６７）、１６進定数値（００Ｈ～ｆｆｆｆＨ）、または、システム変数の値を指定します。これを、その後のＳｅｌｅｃｔ文で判断することができます。

システム変数は、つぎの５つです。

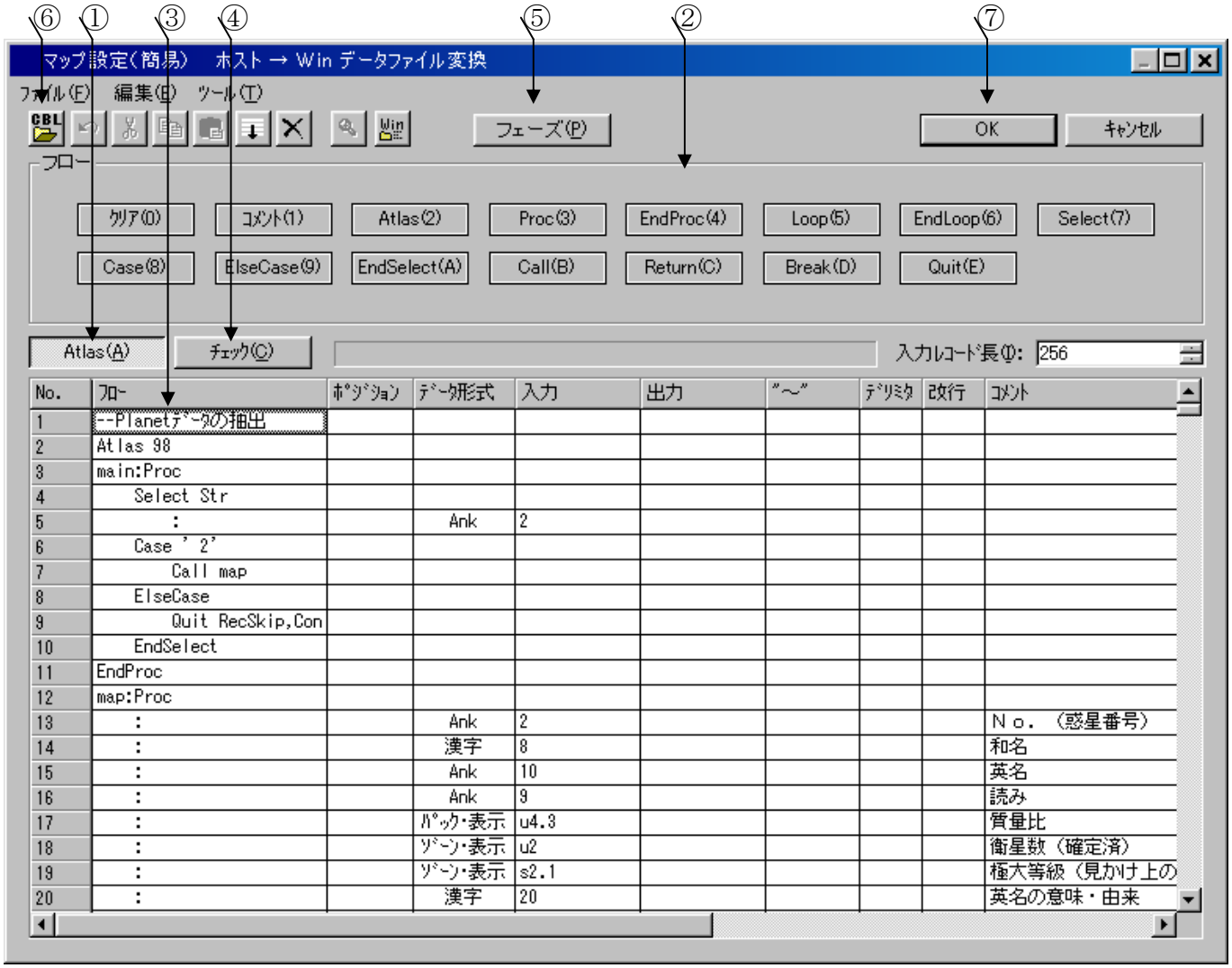
ＳｙｓＰｈａｓｅ	フェーズ（Ｏｐｅｎ＝１、Ｍａｉｎ＝２、Ｃｌｏｓｅ＝８）
ＳｙｓＲｅｃＮｕｍ	レコード番号（０～）
ＳｙｓＲｅｔｕｒｎ	帰り値（Ｒｅｔｕｒｎ文で代入される）
ＳｙｓＢｒｅａｋ	ブレーク値（Ｂｒｅａｋ文で代入される）
ＳｙｓＱｕｉｔ	クイット値（Ｑｕｉｔ文で代入される）

第3章

Ａｔｌａｓ９８の使用法

３．１　Ａｔｌａｓ９８モード

マップ設定（簡易）ウインドウでＡｔｌａｓ９８を設定します。



①Ａｔｌａｓボタンをクリックして、Ａｔｌａｓモードにします。
再度、Ａｔｌａｓボタンをクリックすると、従来互換モードに戻ります。ただし、
フロー欄にＡｔｌａｓ設定がある状態では、従来互換モードに戻ることはできません。
マップ設定（簡易）ウインドウを開いた時にＡｔｌａｓ設定がある場合には、
自動的にＡｔｌａｓモードになります。

②セルポインタがフロー欄にある場合に、Ａｔｌａｓ設定（コメントを含む）選択肢のボタンが表示されます。このボタンをクリック、または、割り付けられたキー入力を行なうと、Ａｔｌａｓ設定ウインドウが開きます。

③Ａｔｌａｓ設定の内容が表示されるフロー欄です。セルをダブルクリックして、Ａｔｌａｓ設定ウインドウが開き、Ａｔｌａｓの構文設定を簡単に行なうことができます。

④このボタンをクリックすると、フロー欄の入力内容を検査します。
　入力されたＡｔｌａｓ設定が正しければ、設定内容を整形して再表示します。
　誤りがあれば、エラーメッセージを出力します。
⑤このボタンをクリックすると、フェーズ設定ウインドウが開きます。
⑥このボタンをクリックすると、ＣＯＢＯＬのＣｏｐｙ句読み込みウインドウが開きます。
⑦ＯＫボタンをクリックした時にも、フロー欄の入力内容を検査します。

＜Ａｔｌａｓ設定ウインドウ＞

　つぎのＡｔｌａｓ設定ウインドウで、Ａｔｌａｓの構文設定を行います。設定する構文のキーワードを選択すると、キーワードに沿った設定項目が右側に表示されます。

●マップ設定（詳細）でのＡｔｌａｓ設定

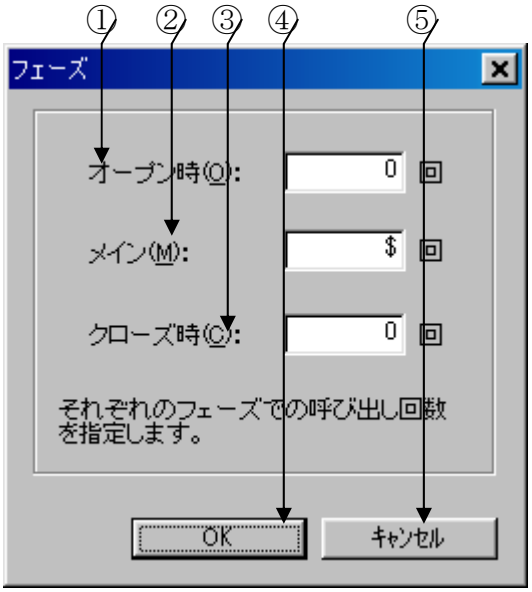
　マップ設定（詳細）でＡｔｌａｓ設定を行う場合は、Ａｔｌａｓ文をブレース（{と}）でくくる記述を行います。Ｆ＊ＴＲＡＮ＋をコマンド型で実行する際の／ＭＡＰ指定の記述、または、その時に使用するパラーメタファイルの記述の仕方も同様です。詳しくは、次章の使用法を参照してください。

＜フェーズ設定ウインドウ＞

変換時にヘッダやトレーラを挿入したいときに、フェーズの設定が必要になります。Ｆ＊ＴＲＡＮ＋には、つぎのフェーズがあります。

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

フェーズ設定ウインドウでは、各フェーズでのＡｔｌａｓ呼び出し回数を指定します。

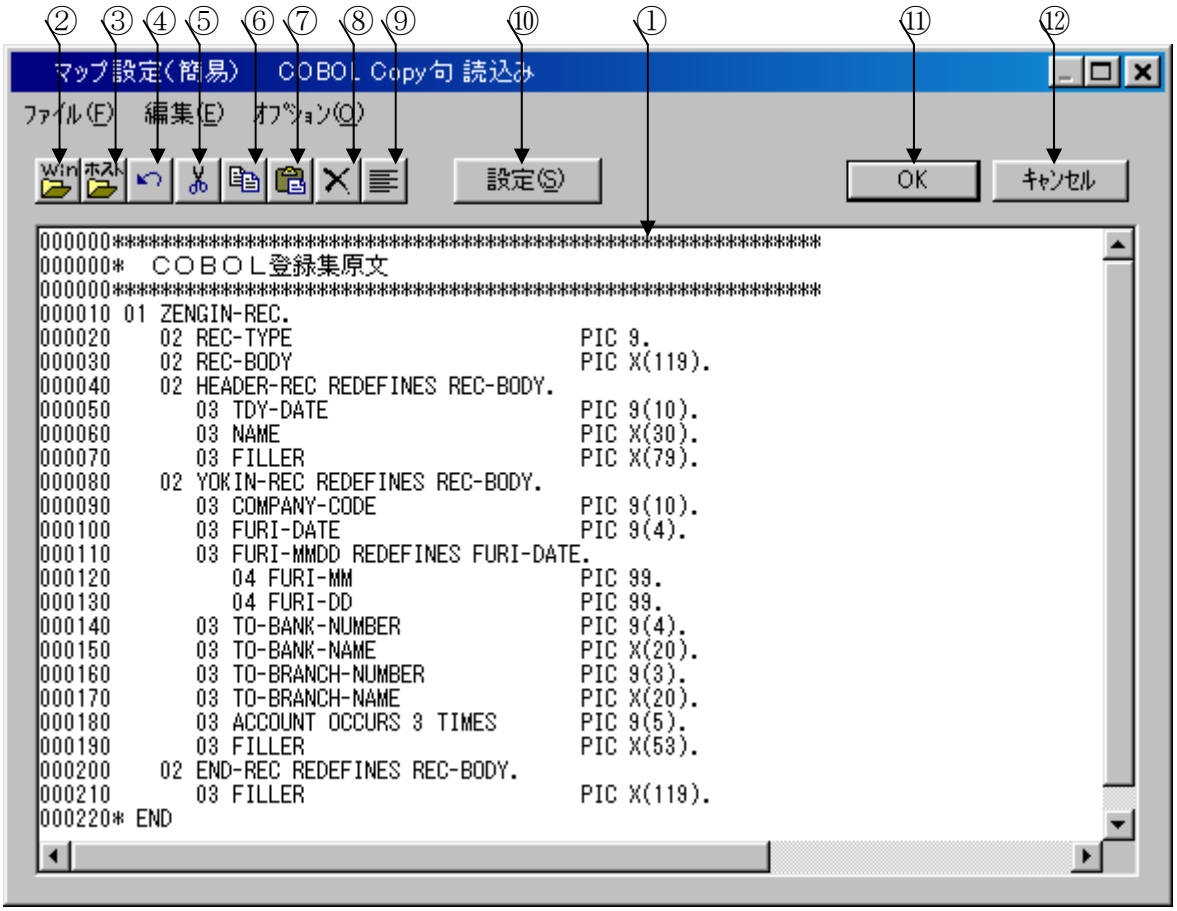


①Ｏｐｅｎフェーズでのａｔｌａｓ呼び出し回数を指定します。
②Ｍａｉｎフェーズでのａｔｌａｓ呼び出し回数を指定します。
③Ｃｌｏｓｅフェースでのａｔｌａｓ呼び出し回数を指定します。
④設定が完了したら、ＯＫボタンをクリックします。
⑤キャンセルボタンをクリックすると、入力／編集作業が無効になります。

回数を０に指定すると、呼び出しは起こりません。初期値はそれぞれ、オープン時が０回、メインが＄（全レコード件数）回、クローズ時が０回となっています。つまり、ヘッダやトレーラは付加せず、本体のみ変換します。

実際のフェーズごとの動作は、Ａｔｌａｓ設定で指定する必要があります。詳しくは、次章の使用法を参照してください。

＜ＣＯＢＯＬのＣｏｐｙ句読み込みウインドウ＞



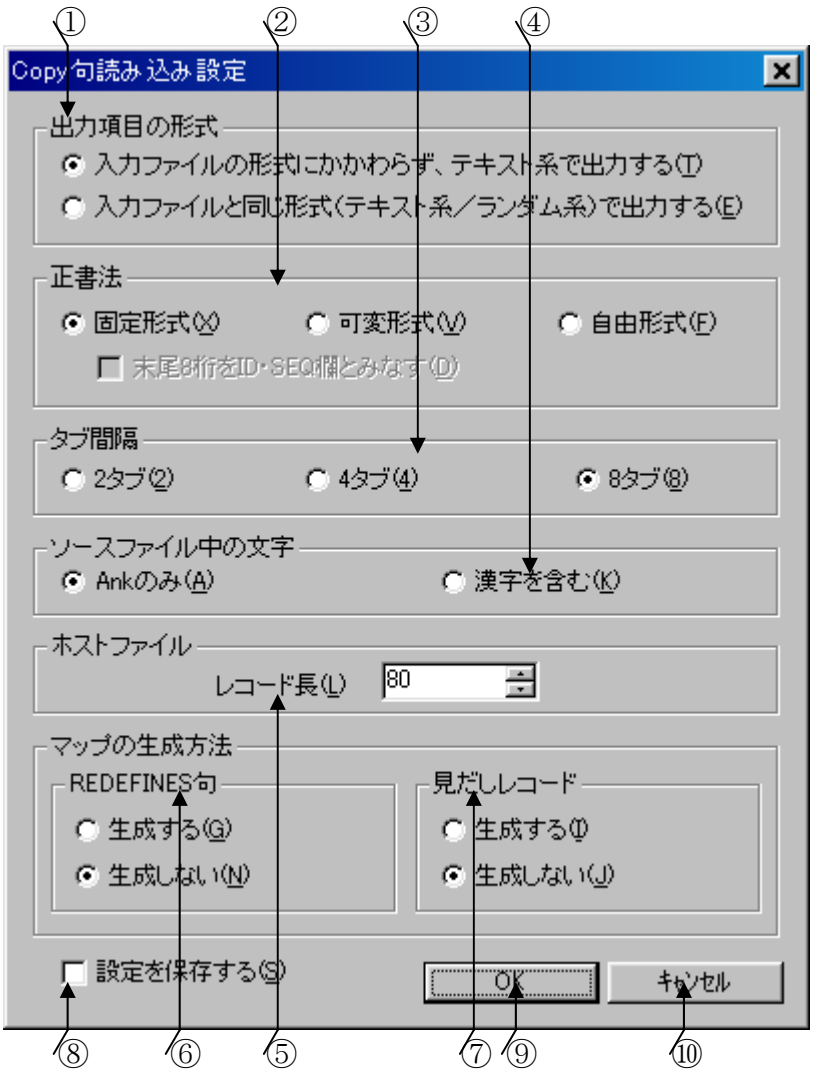
①ＣＯＢＯＬのソースを記述するエディットボックスです。

②～⑩はファイル（Ｆ）、編集（Ｅ）、オプション（Ｏ）のメニューの機能がボタン化されています。
②既存のＣＯＢＯＬソースファイル（Ｗｉｎｄｏｗｓファイル）を読み込みます。
③既存のＣＯＢＯＬソースファイル（ホストファイル）を読み込みます。
④直前の編集作業が無効になり、元に戻ります。
⑤選択した文字列が切り取られ、カットバッファに入ります。
⑥選択した文字列がカットバッファに入ります。
⑦⑤／⑥の操作でカットバッファに入った内容を、カーソルがある位置へ貼りつけます。
カーソルがある位置以降に文字列があれば、挿入になります。
⑧選択した文字列が削除されます.。
⑨①の文字列をすべて選択状態にします。
⑩設定ボタンをクリックすると、Ｃｏｐｙ句読み込み設定ウィンドウが開きます。変換設定ウインドウからも開くことができます。

⑪編集が完了したらＯＫボタンをクリックします。
　マップ設定（簡易）画面に戻り、編集した内容がＭａｐ／Ａｔｌａｓ展開されます。
⑫キャンセルボタンをクリックすると、すべての入力／編集作業が無効になります。

＜Ｃｏｐｙ句読み込み設定ウインドウ＞

　ＣＯＢＯＬのＣｏｐｙ句（登録集）を読み込み、マップ設定に展開するときの設定をします。



①出力項目の形式を指定します。
　テキスト系（Ｔ）を指定すると、入力形式に関わらずテキスト系で出力します。
　入力ファイル形式に準じる（Ｅ）を指定すると、入力形式がテキスト系なら出力もテキスト系にし、入力がランダム系なら出力もランダム系にします。

②ＣＯＢＯＬソースの正書法を指定します。“末尾８桁をＩＤ・ＳＥＱ欄とみなす（Ｉ）”のチェックボックスは、可変形式を指定したときのみ有効です。

［固定形式］１行の初めから６桁目までを一連番号領域、７３桁目から８０桁目をＩＤ・ＳＥＱ欄（プログラム識別番号領域）と見なします。７桁目から７２桁目が有効桁です。

［可変形式］１行の初めから６桁目までを一連番号領域と見なします。７桁目以降が有効桁です。最大２５６桁とします。
　ただし、“末尾８桁をＩＤ・ＳＥＱ欄とみなす（Ｉ）”をチェックすると、有効桁は７桁目から「行末－８」桁目までとなります。

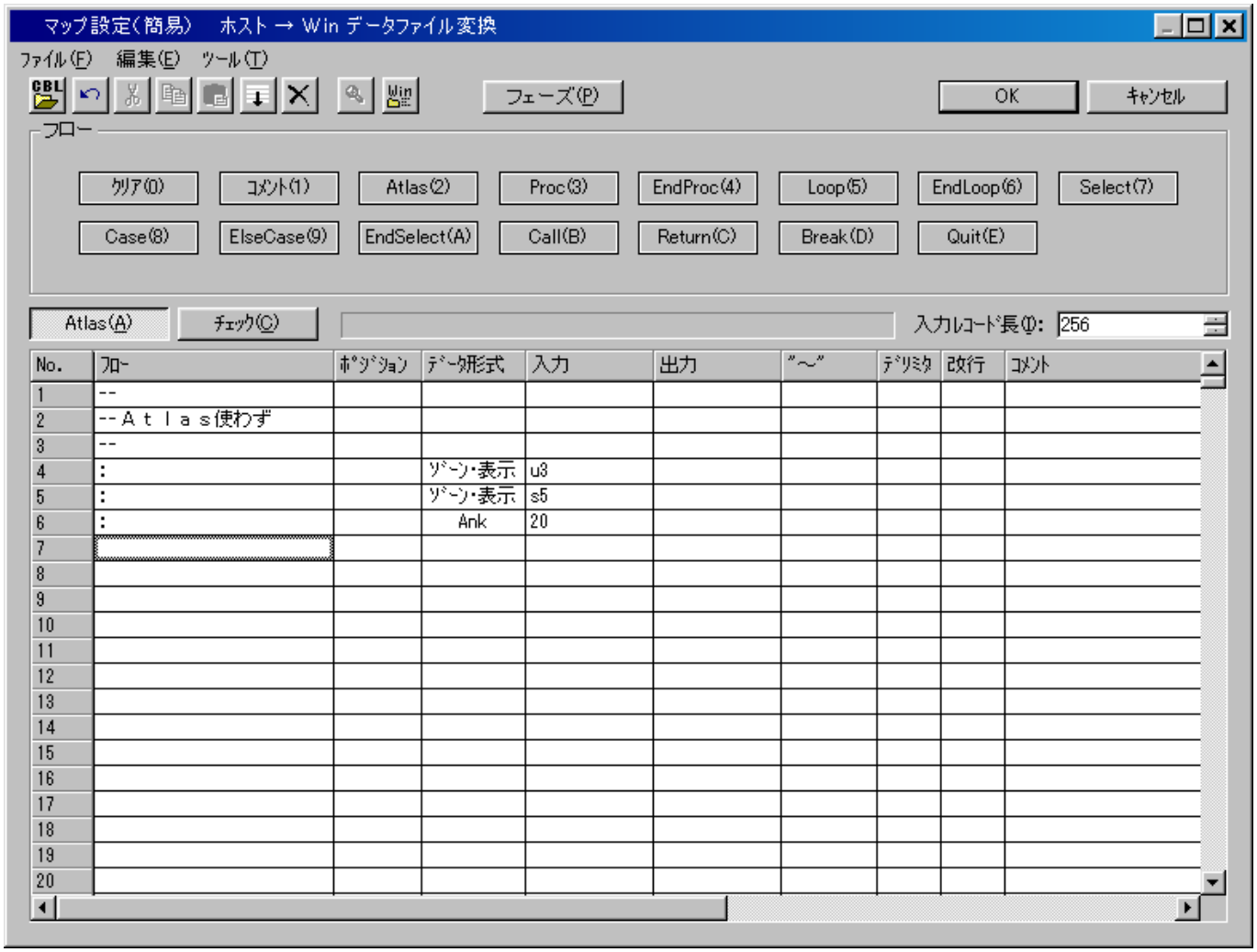
［自由形式］行全体が有効桁です。１行は最大２５６桁とします。

③ＣＯＢＯＬソースで使用しているタブ間隔を指定します。
④ＣＯＢＯＬソースがＡＮＫのみで書かれているか、漢字が含まれているかを指定します。
⑤ＣＯＢＯＬソースがホストファイルのとき、レコード長を指定します。
⑥ＣＯＢＯＬソース中のＲＥＤＥＦＩＮＥＳ句で再定義したデータを無視するか、有効にするかを指定します。
生成する（Ｇ）を指定すると、ＲＥＤＥＦＩＮＥＳ句はコメントとして生成されます。
具体的な使用方法は、次章の使用例を参照してください。
⑦見だしレコードを生成するかを指定します。⑤のＲＥＤＥＦＩＮＥＳ句を生成すると指定したときは、見だしレコードを生成してもあまり意味がないため指定できません。
具体的な使用方法は、次章の使用例を参照して下さい。
⑧このチェックボックスをＯＮにしてＯＫボタンをクリックすると、Ｆ＊ＴＲＡＮ＋終了時にここで設定した内容がコード変換表ファイルに書き込まれます。このウインドウを
変換設定（S）→Ｃｏｐｙ句読み込み設定（Ｐ）で開いたときは、デフォルトでチェックした状態になっていて変更できないようになっています。
⑨ＯＫボタンをクリックすると修正したＣｏｐｙ句読み込み設定が有効になります。
ただし、あくまでもメモリ上の変更であり、コード変換表ファイルに書き込まれるわけではありません。
⑩キャンセルボタンをクリックすると修正したＣｏｐｙ句読み込み設定を無効にし、
Ｃｏｐｙ句読み込み設定のウィンドウを閉じます。

３．２　Ａｔｌａｓ９８の使用法

■Ａｔｌａｓ９８を使わない場合

まず、ＡｔｌａｓモードでＡｔｌａｓ９８を使わない方法を１つ見てみましょう。

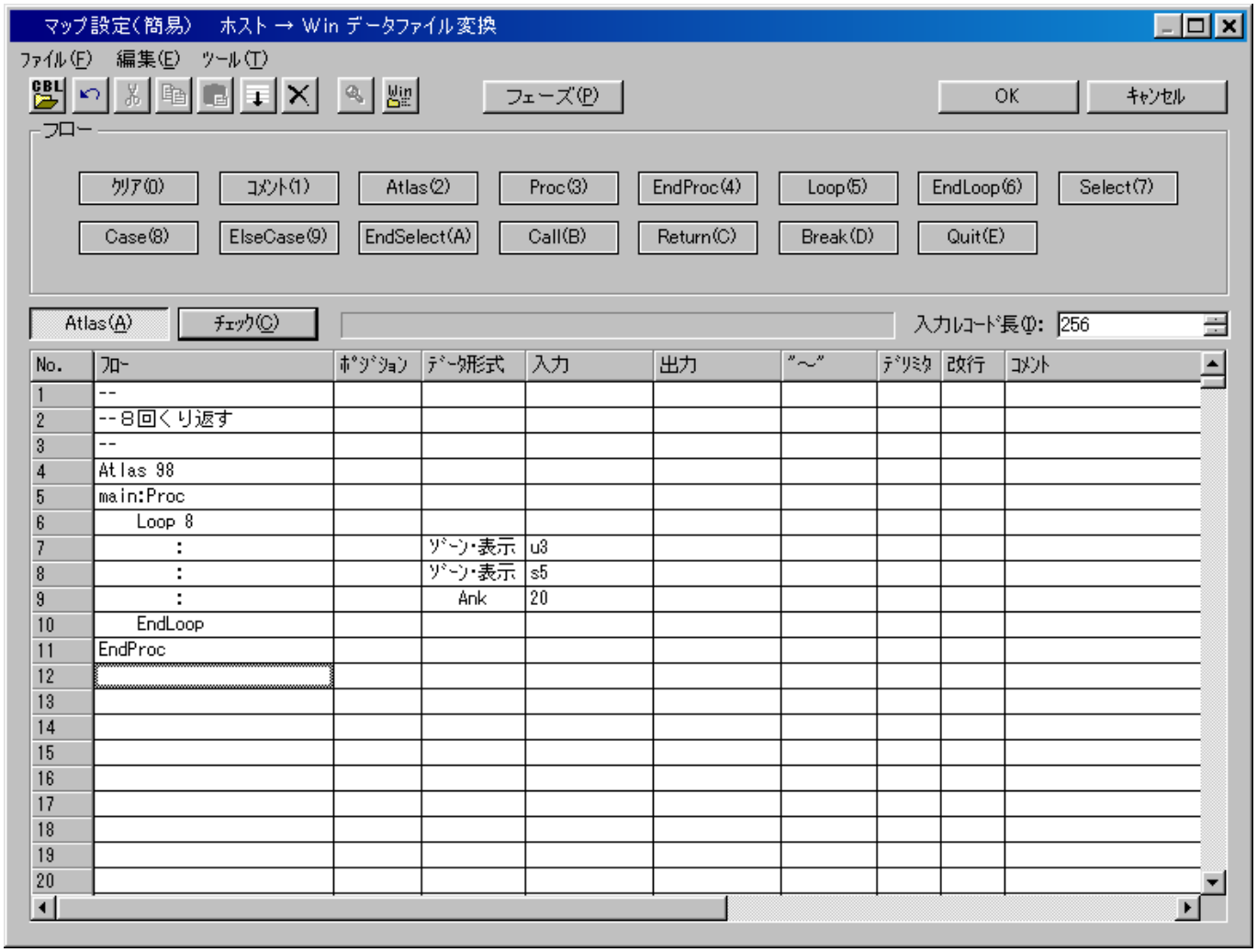


フロー欄というのが先頭にあります。ここにコメントやＡｔｌａｓ９８の処理を書きます（コメントは右端のコメント欄にも書けます)。この例ではマップの設定がある行にコロン（：）が表示されているだけです。このコロンはレベルマークといって、入れ子の状態を分かりやすくする記号です。利用者が入力するものではないことを、最初に断っておきます。

そうすると、これは特に何も指定していないマップで、従来のマップ設定と意味は変わらないことが分かると思います。ここは重要なところです。つまり、**積極的にＡｔｌａｓ９８の機能を使わないかぎり、従来のマップ設定と互換性があるのです。今までの利用者の資産は、１つも無駄にはなりません。**

■Ｌｏｏｐ文を使う方法

では手始めに、Ａｔｌａｓ９８の一番簡単な機能を使ってみましょう。Ａｔｌａｓ９８で一番簡単なのは、Ｌｏｏｐ文です。これは、指定回数だけマップをくり返すものです。最初の例を８回くり返すとしたら、こんなふうになります。



Ｌｏｏｐ　８～ＥｎｄＬｏｏｐというのは、この中を８回くり返せという意味です。

◆注意　----　Ａｔｌａｓ９８のルール

Ａｔｌａｓ９８は、次のルールに従わなければなりません。

コメントや空行を除く実質的な先頭に、Ａｔｌａｓ　９８と書くこと
実行したい部分をＰｒｏｃ～ＥｎｄＰｒｏｃでくくり、手続きという単位にすること

最低限これを守らないと、文法違反になります。Ｐｒｏｃの行の「ｍａｉｎ」は手続き名で、適当な名前を指定してもかまいませんが、習慣的に「ｍａｉｎ」または「メイン」を使います。

■簡単なＳｅｌｅｃｔ文を使う方法

さあ、これでＡｔｌａｓ９８プログラムの最小限の構造は分かりました。次は、簡単なＳｅｌｅｃｔ文の使い方です。

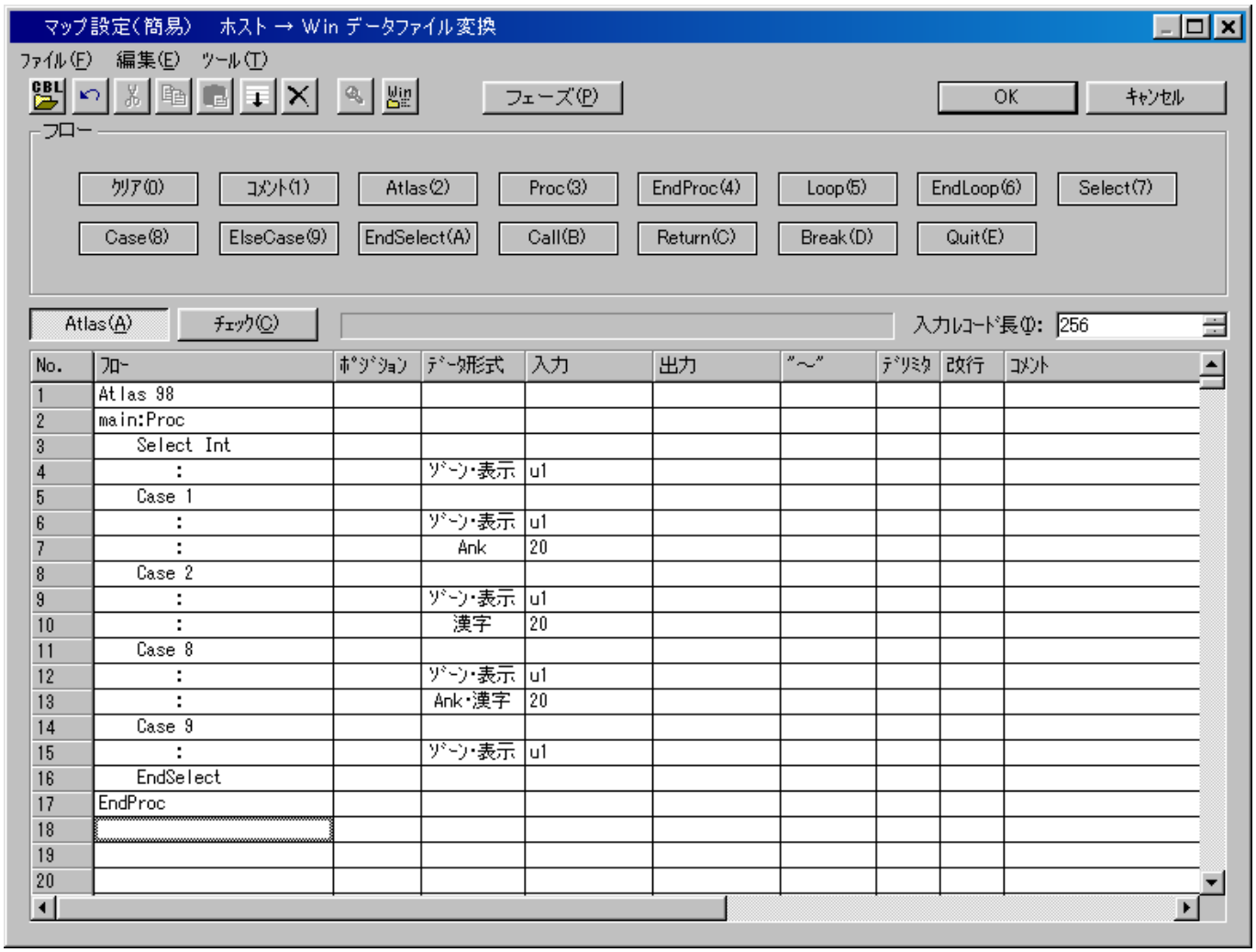
最初に述べたとおり、Ｓｅｌｅｃｔ文は主にマルチレイアウト変換に使います。マルチレイアウトで一番多い例は、レコードの先頭１バイトでレコード区分を示すものです。たとえば、こんなレコード区分になっているファイルを考えます。

’１’　ヘッダレコード
’２’　ボディーレコード
’８’　トレーラレコード
’９’　エンドレコード

このフォーマットのファイルを変換するには、

先頭１バイトを変換してみて、
値が１なら、ヘッダレコード用のマップを実行し、
値が２なら、ボディーレコード用のマップを実行し、
値が８なら、トレーラレコード用のマップを実行し、
値が９なら、エンドレコード用のマップを実行する。

ということが書ければよいことが分かります。実際、この流れのとおり書けるのです。見てみましょう。



簡単ですね。他の高級言語を知っている方なら、とてもやさしいと思います。

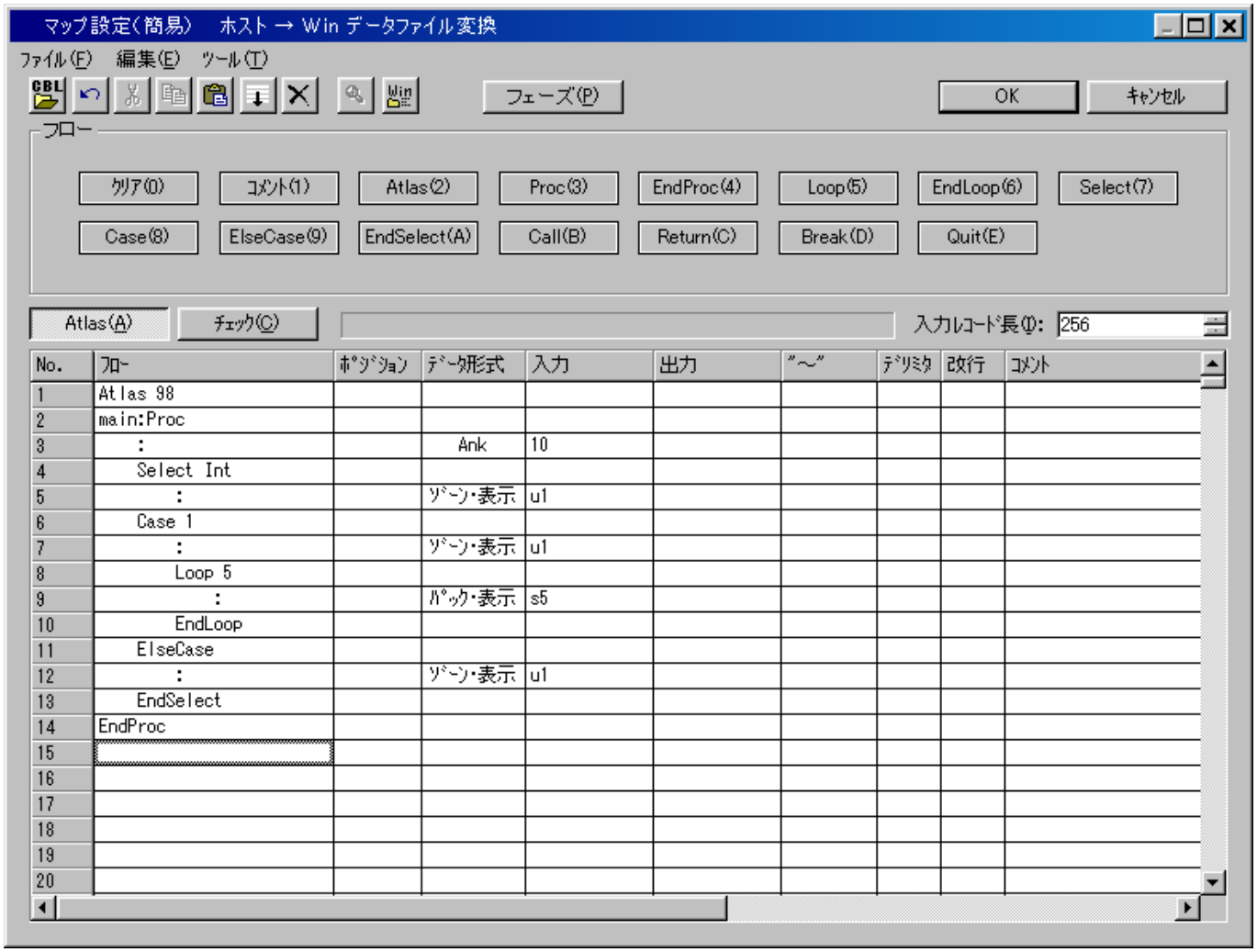
Ｓｅｌｅｃｔ　Ｉｎｔ（Ｓｅｌｅｃｔ　Ｉｎｔｅｇｅｒ）というのは、整数値で場合分けするということです。そして、その次に書いた「ゾーン・表示　ｕ１」が実行されます。といっても、テストのための実行ですから、その項目の値が得られたら実行はなかったことにされます。そして、各Ｃａｓｅの値とつぎつぎに比較していき、Ｃａｓｅの値が同じところを実行するのです。そのＣａｓｅが終わると、ＥｎｄＳｅｌｅｃｔまで飛んで終了ということになります。

■途中からマルチレイアウトになる場合

では途中まで同じレイアウトで、途中の区分項目によって後半のパターンが変化するという、つぎのような例を考えてみましょう。

項　番	レコードタイプ１	レコードタイプ２
１（共通）	Ａｎｋ　　１０	Ａｎｋ　　１０
２（フラグ）	ゾーン・表示　ｕ１（＝１）	ゾーン・表示　ｕ１（１以外の値）
３	パック・表示　ｓ５	なし
４	パック・表示　ｓ５	なし
５	パック・表示　ｓ５	なし
６	パック・表示　ｓ５	なし
７	パック・表示　ｓ５	なし

これは、こんなふうに指定できます（Ｌｏｏｐ文も使ってみましょう）。



何でも入れ子状に書ける様子がよく分かります。「ゾーン・表示　ｕ１」をＣａｓｅ、ＥｌｓｅＣａｓｅのところに再指定しているのに注意してください。そうしないと、そのゾーン形式の項目がＳｅｌｅｃｔに「食われて」しまうと考えてください。

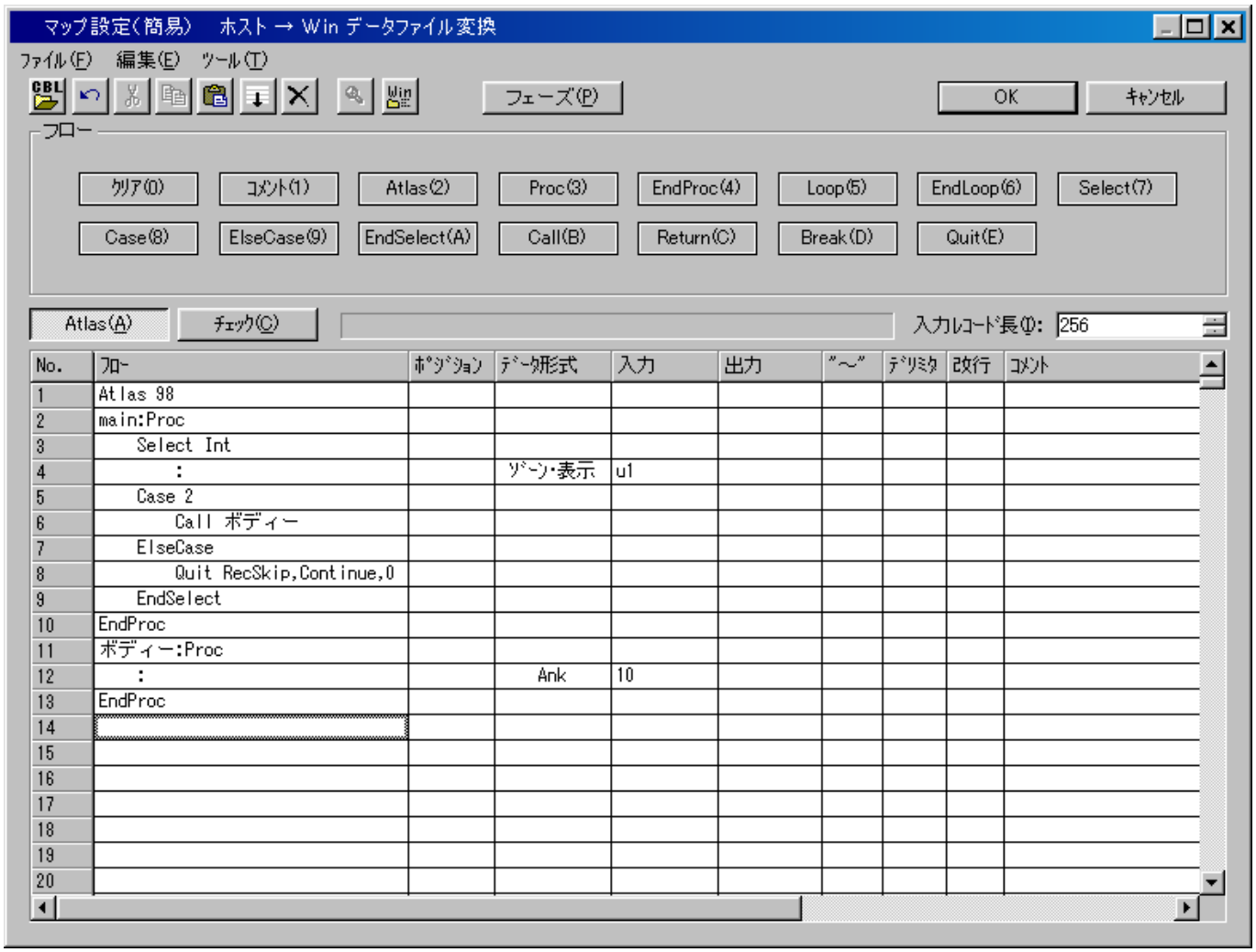
■複数の手続きを使う場合

ところで、ボディーレコード用のマップが何十行もあったらどうすべきでしょうか。そこに書けばいいだけだともいえますが、Ｓｅｌｅｃｔ～ＥｎｄＳｅｌｅｃｔ構造が短くまとまっていたほうが、分かりやすくなります。ボディーレコード用のマップを、外にくくり出してみましょう。そして、それを呼び出すようにします。マップのくくり出しには、Ｐｒｏｃ～ＥｎｄＰｒｏｃを使います。手続きをこんな具合にもう１つ作るのです。

ボディー：Ｐｒｏｃ
　　ボディーレコード用のマップ
ＥｎｄＰｒｏｃ

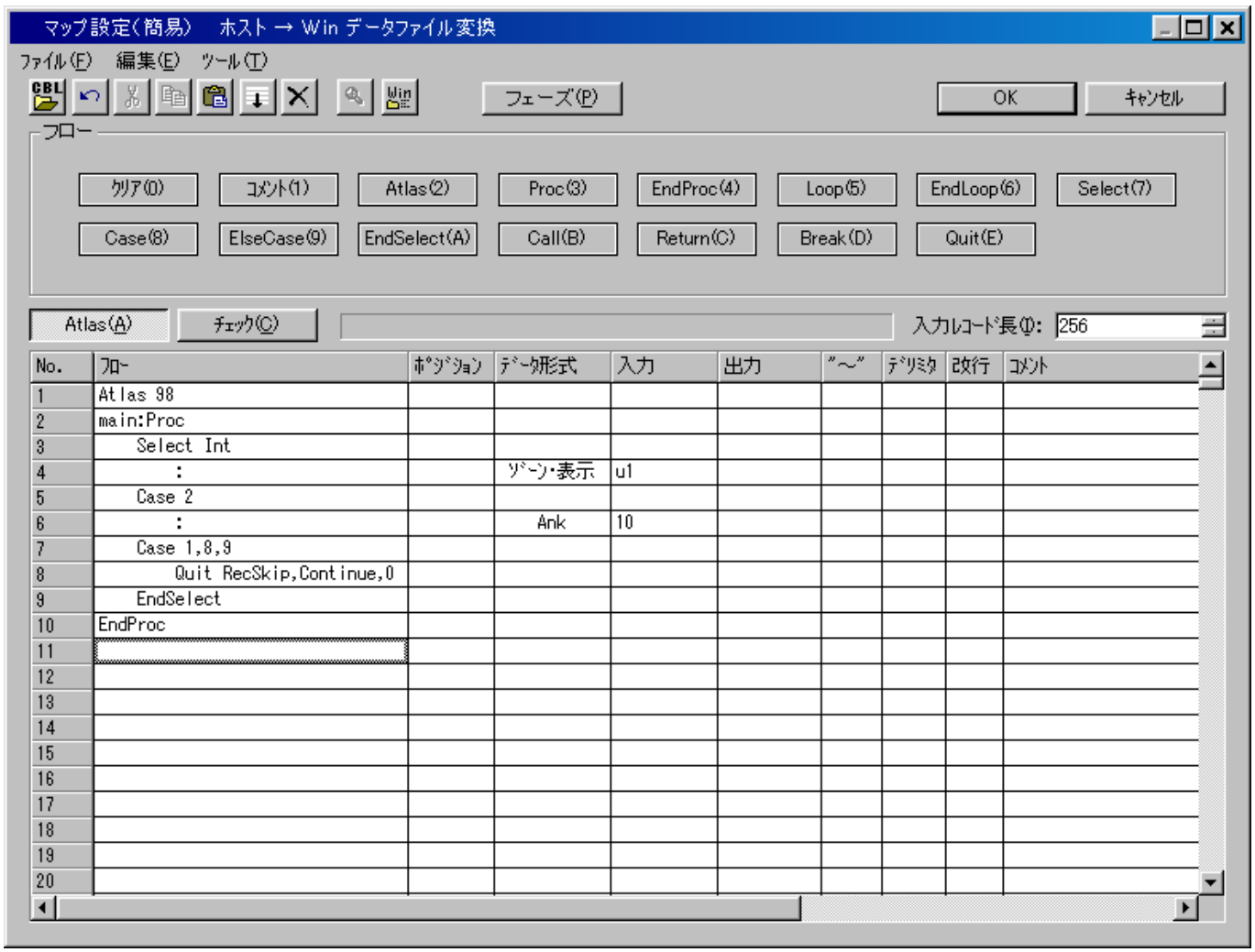
そしてこれを呼び出すには、元のところに

Ｃａｌｌ　ボディー　　と書きます。全体は、こんな具合になります。



■不要なレコードを捨てる場合

さて、最初の例に戻ってボディーレコードしか要らない場合、つまりヘッダレコード、トレーラレコード、エンドレコードは捨てるという処理が必要なら、どう書くのでしょうか。Ｃａｓｅには、どんな場合かをコンマ（,）で区切って列挙できるので、まずこんなふうに書けます。



Ｃａｓｅ　１，８，９は「１、８、または９なら」という意味です。その下に書いたＱｕｉｔ文がポイントです。Ｑｕｉｔ文をデフォルトのまま書くと、「そのレコードを捨てよ」という意味になります。つまり上の例は、「値が１、８、または９のレコードは捨てよ」と指定したことになります。

Ｓｅｌｅｃｔ文には「その他の場合」という指定もできます。具体的には、Ｃａｓｅの代わりにＥｌｓｅＣａｓｅと書いてやればよいのです。すると、前頁の例は、つぎのように書くこともできます。

これで、「その他の場合はそのレコードを捨てよ」と指定したことになります。

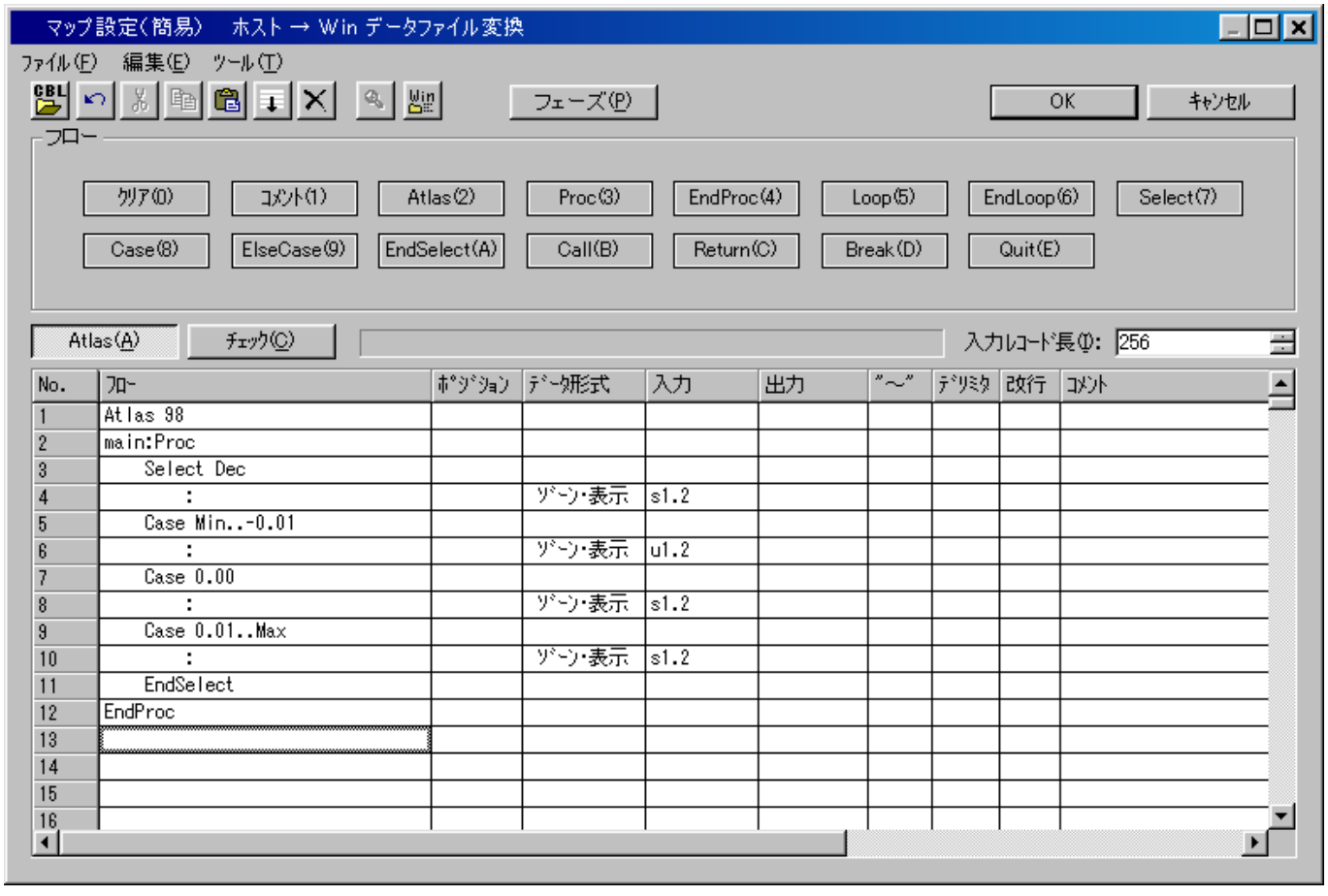
■小数項目による分岐を行う場合

ここまで、整数で分岐させる例を見てきましたが、この他に小数項目の値によって分岐させる方法と、文字列項目の値によって分岐させる方法、そして、バイナリ項目の値によって分岐させる方法があります。

小数項目によって分岐させるには、Ｓｅｌｅｃｔ　Ｄｅｃ（Ｓｅｌｅｃｔ　Ｄｅｃｉｍａｌ）を使います。ＣＯＢＯＬのピクチャがＳ９Ｖ９９のような項目の値によってレイアウトが分かれる、次のようなファイルを考えます。

範　　囲	レコード区分
－９．９９　～　－０．０１	マイナスレコード
０．００	ゼロレコード
０．０１　～　９．９９	プラスレコード

範囲を「ＡからＢまで」と指定するには、「Ａ．．Ｂ」と書きます。また、～以上、～以下という指定をしやすくするＭｉｎ．．と．．Ｍａｘという指定方法があります。これを使ってみましょう。上の例は次のように書けます。ちなみにＭｉｎ、Ｍａｘは整数のときにも使えます。



「Ｍｉｎ．．－０．０１」は「－０．０１以下なら」と読めばよいですし、「０．０１．．Ｍａｘ」は「０．０１以上なら」と読めばよいのです。

■文字列による分岐を行う場合

ではＳｅｌｅｃｔ文を文字列で分岐させる、下表のような例を考えてみましょう。

’Ａ０’	～ ’Ａ９’	タイプＡレコード
’Ｂ０’	～ ’Ｂ９’	タイプＢレコード
’Ｃ０’	～ ’Ｚ９’	タイプＣＺレコード
’ＺＺ’		エンドレコード（途中に現れたらそこで中断）

これは、こんなふうに書けます。

Ｓｅｌｅｃｔ　Ｓｔｒ（Ｓｅｌｅｃｔ　Ｓｔｒｉｎｇ）と書くと、文字列による比較になります。’Ａ０’..’Ａ９’は「Ａ０からＡ９までなら」という意味になります。

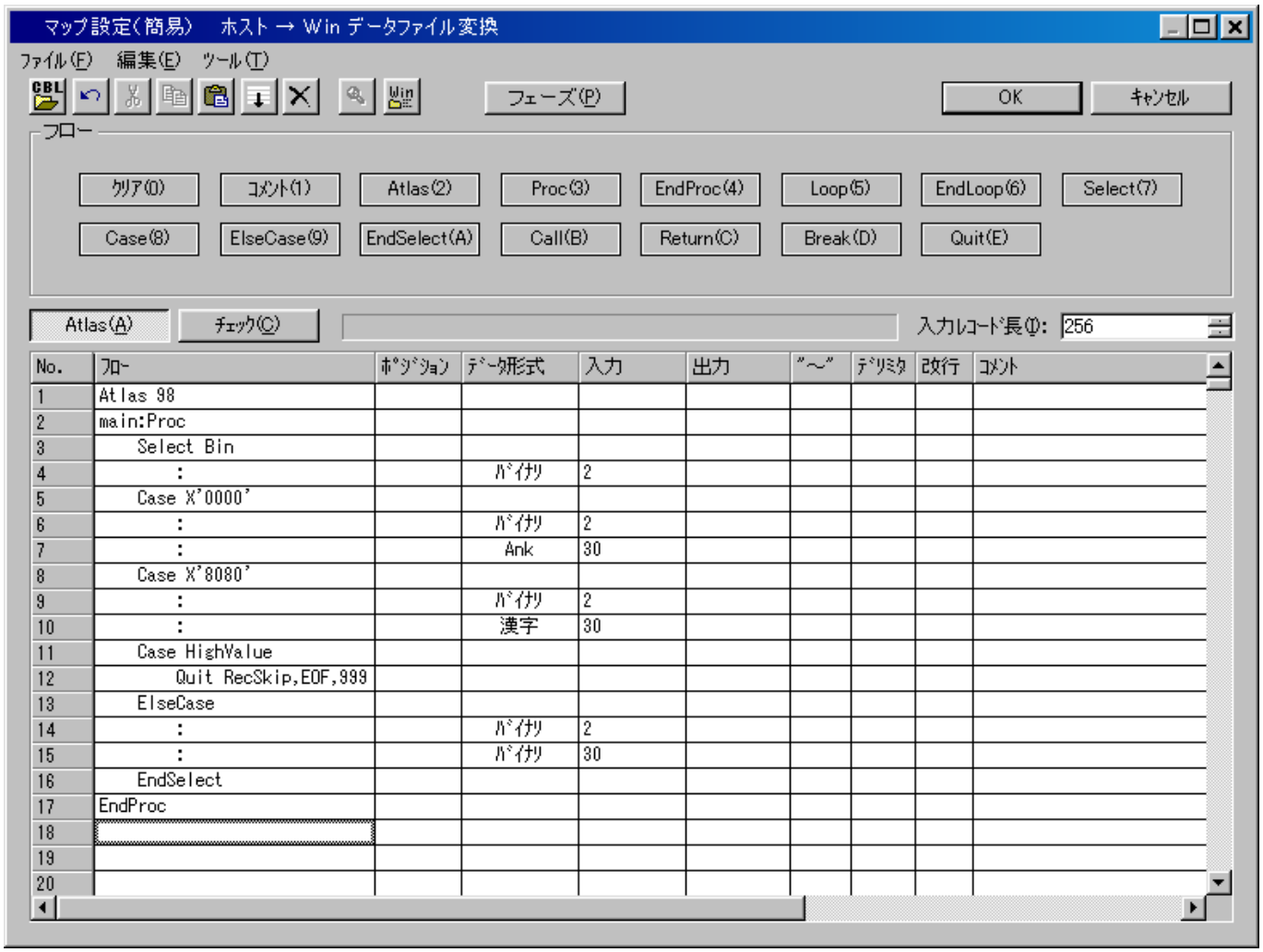
’ＺＺ’の場合の「Ｑｕｉｔ　ＲｅｃＳｋｉｐ，ＥＯＦ」とは、「ＺＺなら、ここでこのファイルの処理を終えよ」という意味です。

■バイナリ項目による分岐を行う場合

Ｓｅｌｅｃｔ文をバイナリ項目によって分岐させる例を考えてみましょう。

Ｘ’００００’	タイプＡレコード
Ｘ’８０８０’	タイプＢレコード
ＨｉｇｈＶａｌｕｅ	エンドレコード（途中に現れたらそこで中断）
上記以外	タイプＣレコード

これは、こんなふうに書けます。



文字列による分岐とほとんど変わりませんが、Ｓｅｌｅｃｔ　Ｂｉｎ（Ｓｅｌｅｃｔ　Ｂｉｎａｒｙ）と書くと、バイナリ項目による比較になります。

ＨｉｇｈＶａｌｕｅはバイナリＦＦ・・・に一致します。「それ以外のどんなバイナリデータよりも大きい」という意味です。

■レコード番号による分岐を行う場合

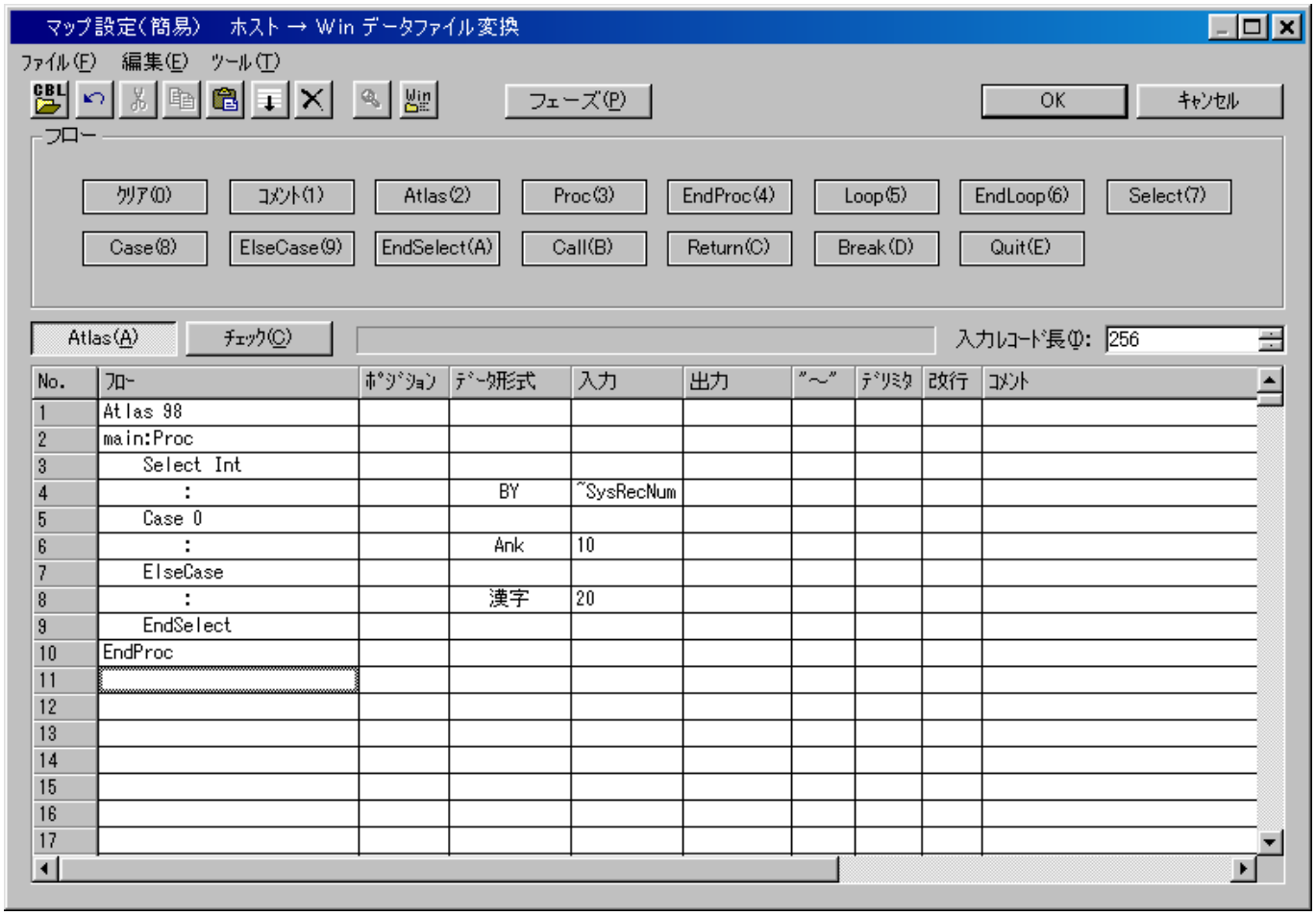
先頭１～数レコードだけ別レイアウトで､その後にはボディーレコードが続くといったファイルもよくあります。Ａｔｌａｓ９８には、レコード番号をもとに変換し分ける方法も用意されています。

「レコード番号が何番か」と聞くには、

```
Ｓｅｌｅｃｔ　Ｉｎｔ
   :                BY　~ＳｙｓＲｅｃＮｕｍ
```

と書きます。このＳｙｓＲｅｃＮｕｍはシステム変数の１つで、Ｆ＊ＴＲＡＮ＋が自動的にレコード番号をセットします。

では、先頭レコードだけ別レイアウトの場合を考えてみましょう。こんなふうになります。



ちなみに、先頭レコードのレコード番号は０です。
一定周期でくり返し同じレイアウトが出てくるようなファイルも、この応用で変換できます。

■脱出文の使用法

条件によって、くり返しを中断したいことがあります。このときは、Ｂｒｅａｋ文やＲｅｔｕｒｎ文を用います。Ｂｒｅａｋ文は、Ｌｏｏｐ構造やＳｅｌｅｃｔ構造から抜け出すための文です。一方、Ｒｅｔｕｒｎ文は、手続きから抜け出すための文です。なお、先に述べたＱｕｉｔ文も脱出文の１つです。

Ｂｒｅａｋ文は、おもにＬｏｏｐ構造からの抜け出しに使います。２重ループからの脱出もできます。脱出条件はＳｅｌｅｃｔ文で判定します。

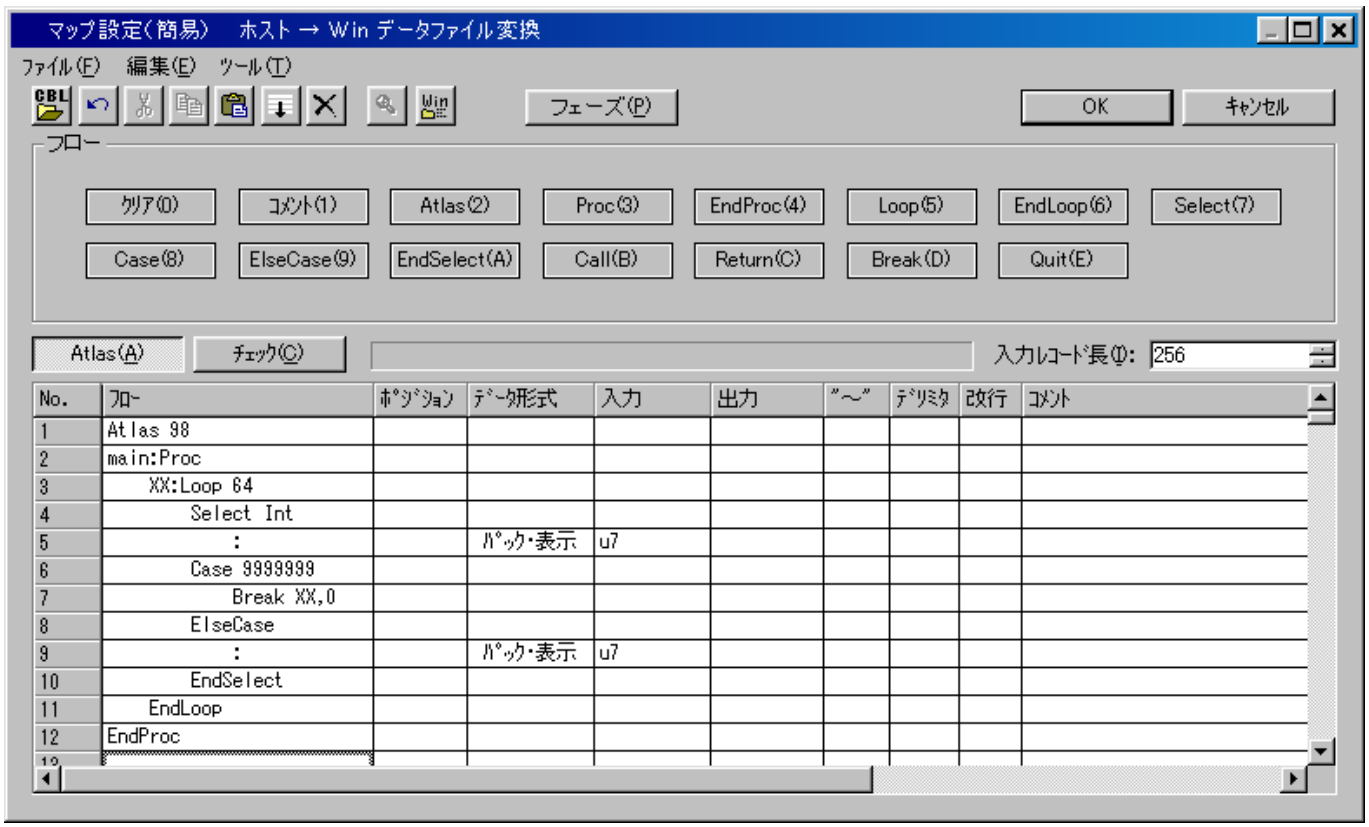
７桁のパック形式項目（４バイト）の配列があるとします。６４エントリで１レコードです。このとき、値が９９９９９９９ならそのレコードはそこまでで変換を中断する例を考えてみます。全体をループにして、

```
ＸＸ：Ｌｏｏｐ　６４
    :
    :　ＸＸループの中身
    :
ＥｎｄＬｏｏｐ
```

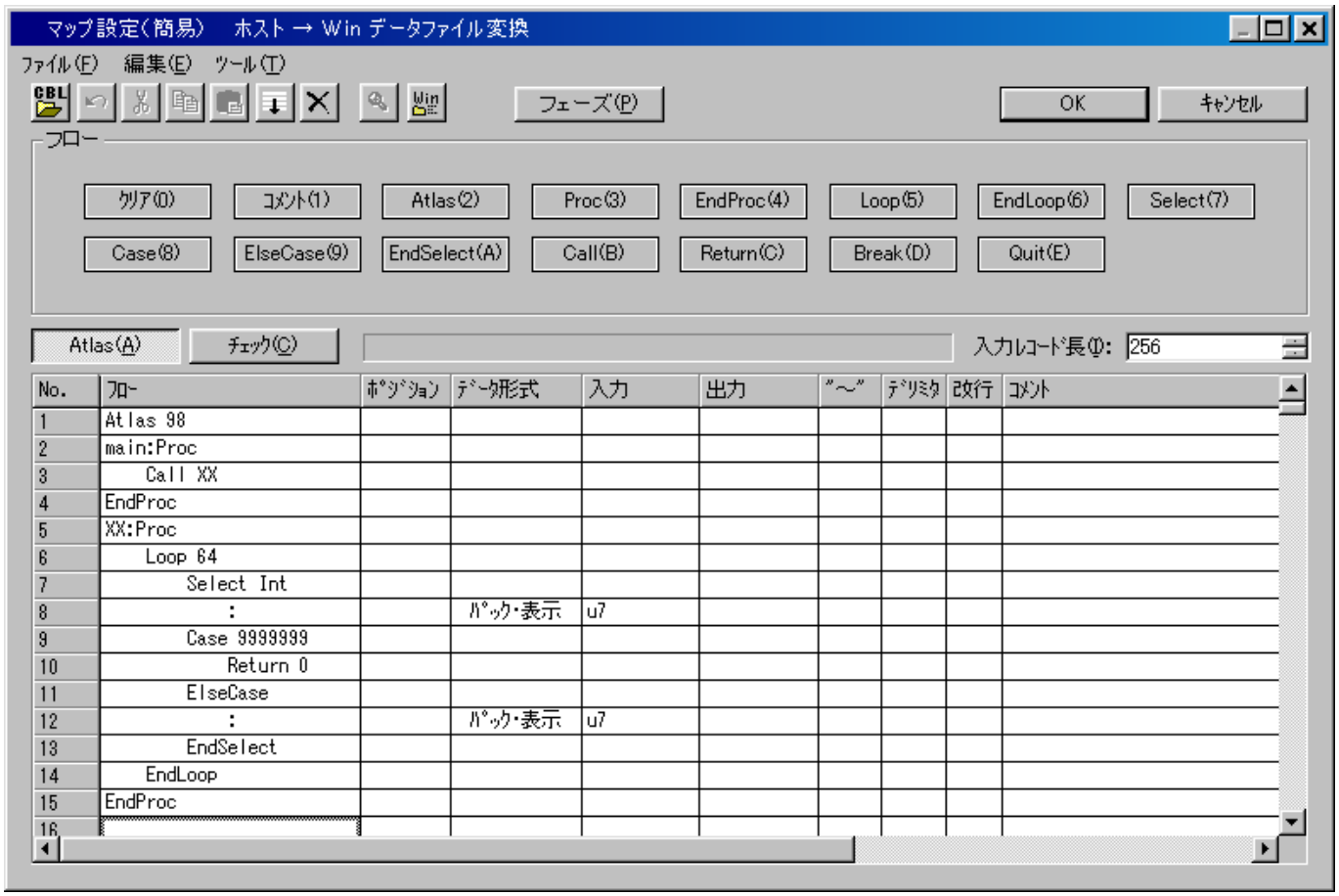
と書きます。このループにＸＸという名前（構造名）を付けていることに注意してください。また、このループから脱出する部分は、

```
Ｓｅｌｅｃｔ　Ｉｎｔ
    :　ＰａｃｋＤｉｓｐ　ｕ７
Ｃａｓｅ　９９９９９９９
   --ＸＸから脱出する
    Ｂｒｅａｋ　ＸＸ
ＥｌｓｅＣａｓｅ
    :　ＰａｃｋＤｉｓｐ　ｕ７
ＥｎｄＳｅｌｅｃｔ
```

と書きます。まとめると、つぎのようになります。



同じことを手続きとＲｅｔｕｒｎ文を使えば、次のようになります。



■ヘッダとトレーラの挿入方法

ボディレコードのみのファイルを変換する際に、ヘッダやトレーラを付けたいという場合がよくあります。Ｆ＊ＴＲＡＮ＋では、ヘッダ／トレーラ挿入の機能をサポートしています。

ヘッダ／トレーラを生成・付加する場合は、Ａｔｌａｓ９８の設定の他に、フェーズの設定を行う必要があります。フェーズには、

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

の３種類があり、フェーズの設定では各フェーズの呼び出し回数を指定します。

それでは、つぎのようなヘッダとトレーラを、それぞれ一行ずつ挿入する場合を考えてみましょう。

“＊＊＊ 今月の売上 ＊＊＊”	生成・付加するヘッダ
：	
：	本体レコード
：	
“合計金額　　￥”	生成・付加するトレーラ

まず、Ａｔｌａｓ９８の設定を行います。Ｓｅｌｅｃｔ文を使って、それぞれのフェーズごとのマップを指定します。

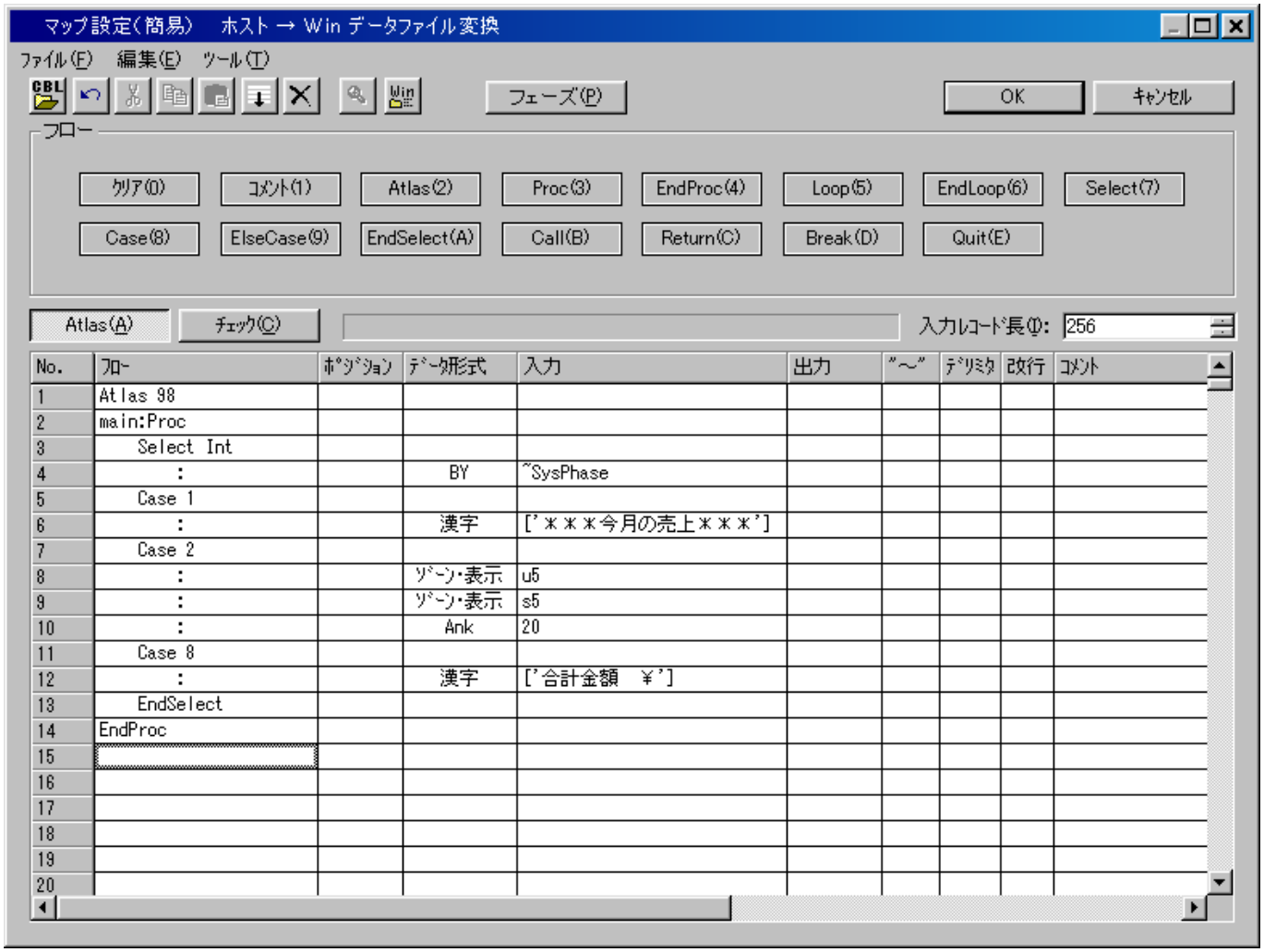
「どのフェーズか」による分岐は、

```
Ｓｅｌｅｃｔ　Ｉｎｔ
　：　　ＢＹ　~ＳｙｓＰｈａｓｅ
```

と書きます。このＳｙｓＰｈａｓｅはシステム変数の１つで、Ｆ＊ＴＲＡＮ＋があらかじめ値を設定しています。
ＳｙｓＰｈａｓｅの値は、つぎのようになっています。

１　＝　Ｏｐｅｎフェーズ
２　＝　Ｍａｉｎフェーズ
８　＝　Ｃｌｏｓｅフェーズ

実際に書いてみると、このようになります。



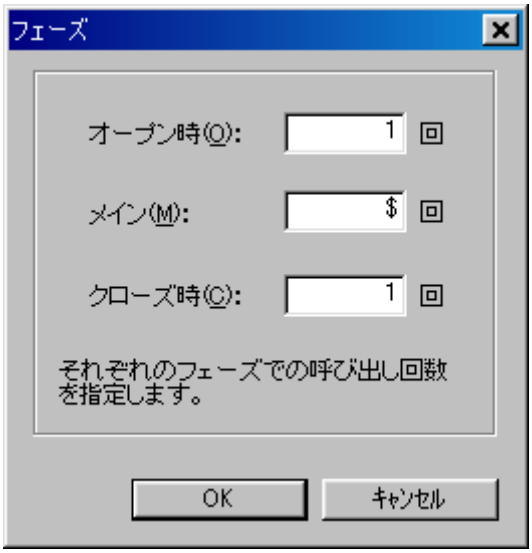
Ｃａｓｅ　１でヘッダ「＊＊＊ 今月の売上 ＊＊＊」を、Ｃａｓｅ　８でトレーラ「合計金額￥」を挿入する指定をしています。以上で、Ａｔｌａｓ９８の設定は終わりです。

つぎに、フェーズの設定を行います。マップ設定（簡易）ウインドウのフェーズボタンをクリックして、フェーズ設定ウインドウを開きます。ここでは、それぞれのフェーズにおいて、Ａｔｌａｓが呼び出される回数を設定します。
呼び出される順番は決まっていて、

Ｏｐｅｎフェーズ → Ｍａｉｎフェーズ → Ｃｌｏｓｅフェーズ

のようになっています。Ｏｐｅｎフェーズから始まり、設定した回数だけＡｔｌａｓを呼び出すとつぎのフェーズに移り、Ｃｌｏｓｅフェーズで終了します。回数を０に指定すると、呼び出しは起こりません。

今回は、Ｏｐｅｎフェーズ、Ｃｌｏｓｅフェーズをそれぞれ「１回」に指定して、ＯＫボタンをクリックします。Ｍａｉｎフェーズは初期値（＄＝全レコード件数）のまま、変更する必要はありません。



以上で設定は終わりです。この設定によって、つぎのように変換します。

１．まず、ＯｐｅｎフェーズでＡｔｌａｓが呼び出され、ＳｙｓＰｈａｓｅ＝１のときの変換を１回行う（ヘッダ挿入)。
２．つぎに、ＭａｉｎフェーズでＡｔｌａｓが呼び出され、ＳｙｓＰｈａｓｅ＝２のときの変換を全レコード件数回行う（ボディレコード変換)。
３．最後に、ＣｌｏｓｅフェーズでＡｔｌａｓが呼び出され、ＳｙｓＰｈａｓｅ＝８のときの変換を１回行い（トレーラ挿入)、Ｆ＊ＴＲＡＮ＋を終了する。

◆注意　----　コマンドで実行する場合

Ｆ＊ＴＲＡＮ＋をコマンド呼び出しする場合、フェーズの設定を行うには、つぎのオプションを付けて実行します。

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

＜ｎ＞は呼び出しの回数で、それぞれのフェーズは順不同です。省略値は

```
／ＰＨａｓｅ　Ｏｐｅｎ０, Ｍａｉｎ＄, Ｃｌｏｓｅ０
```

となっています。

■ＣＯＢＯＬのＣｏｐｙ句を読み込む

ＣＯＢＯＬのＣｏｐｙ句（登録集）などのファイルを直接読み込んで、データ記述項の文をＭａｐ文やＡｔｌａｓ文に自動展開することができます。ファイル中にデータ記述項が複数ある場合には、２つ目以降は無視します。登録集に限らず、ＣＯＢＯＬの原始文中のデータ記述項も読み込み可能です。

データ記述項の中で、Ｍａｐ文やＡｔｌａｓ文に直接関係があるのは、つぎの４つの句です。

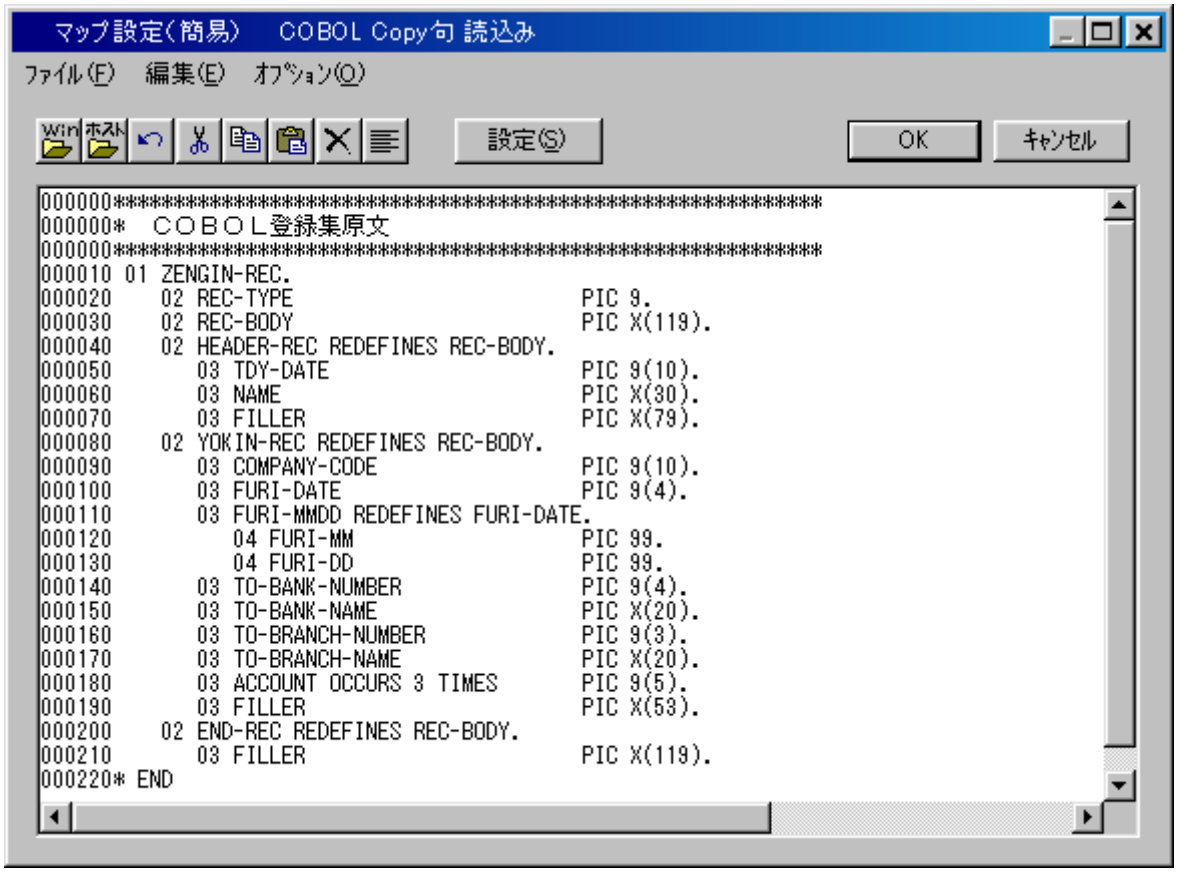
ＰＩＣＴＵＲＥ句	Ｍａｐ文を生成する
ＵＳＡＧＥ句	Ｍａｐ文のデータ形式と入力幅に影響する
ＯＣＣＵＲＥＳ句	Ｌｏｏｐ文を生成する
ＲＥＤＥＦＩＮＥＳ句	Ｓｅｌｅｃｔ文を書くためのコメント文を生成する

その他の句は、全て無視します。

では実際に、どのような手順で行うか見てみましょう。

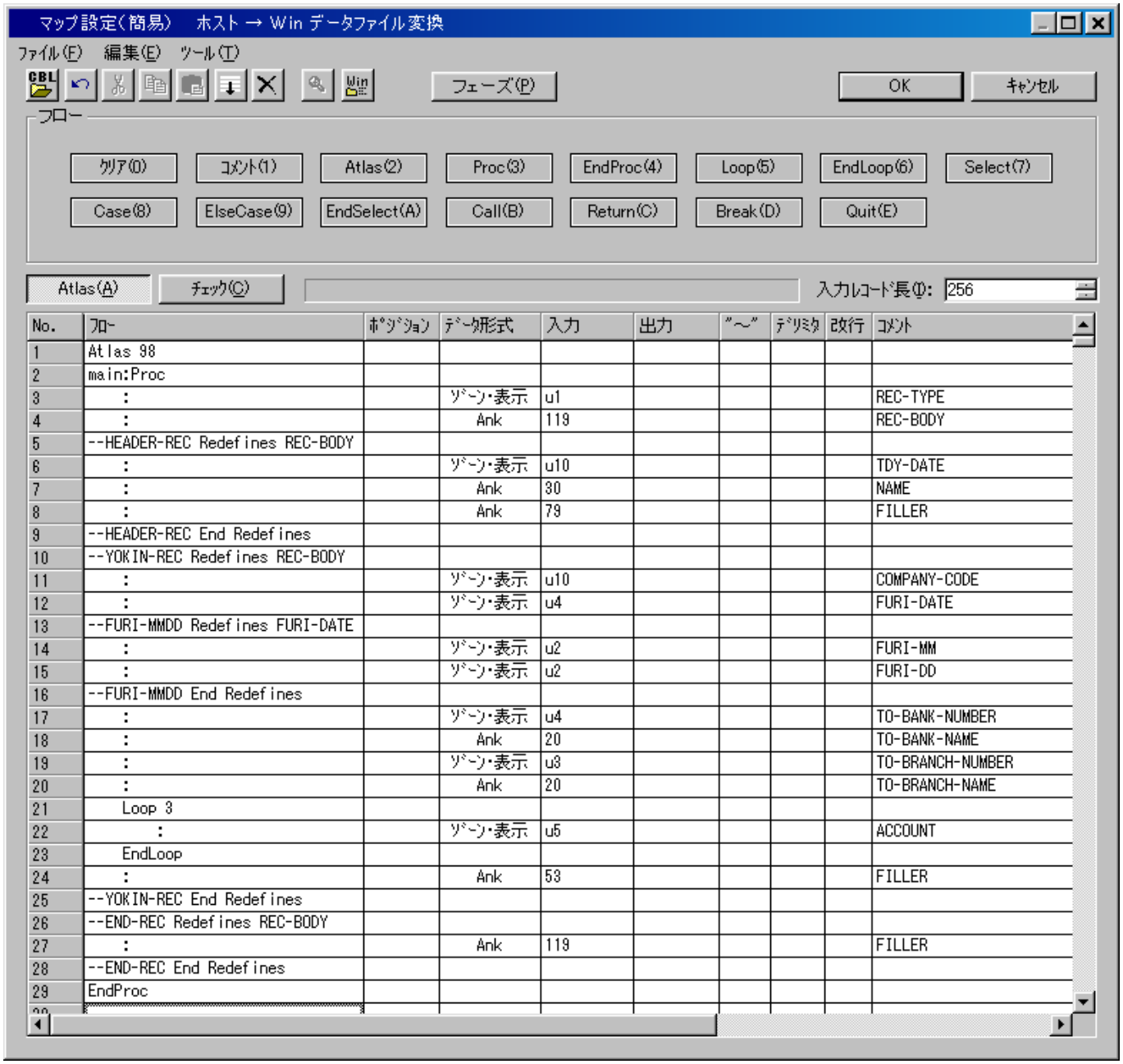
Ｃｏｐｙ句読み込みウインドウのメニューから、ファイル（Ｆ）→Ｗｉｎファイルを開く（Ｏ）（ホストファイルを開くときは、ファイル（Ｆ）→ホストファイルを開く（T））を選択すると、ファイルの参照ウインドウが開きます。

ＣＯＢＯＬのソースファイルを指定してＯＫボタンをクリックすると、Ｃｏｐｙ句読み込みウインドウにファイルの内容が表示されます。



ＣＯＢＯＬの形式や変換方法に合わせて、読み込み設定を行う必要があります。設定ボタンをクリックして、Ｃｏｐｙ句読み込みウインドウから指定してください。詳細は前章のＡｔｌａｓ９８の使用法を参照してください。

内容を確認後（必要があれば編集して)、ＯＫボタンをクリックすると、マップ設定（簡易）ウインドウにデータ記述項の文が展開されます。



No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	:		ゾーン・表示	u1					REC-TYPE
4	:		Ank	119					REC-BODY
5	--HEADER-REC Redefines REC-BODY								
6	:		ゾーン・表示	u10					TDY-DATE
7	:		Ank	30					NAME
8	:		Ank	79					FILLER
9	--HEADER-REC End Redefines								
10	--YOKIN-REC Redefines REC-BODY								
11	:		ゾーン・表示	u10					COMPANY-CODE
12	:		ゾーン・表示	u4					FURI-DATE
13	--FURI-MMDD Redefines FURI-DATE								
14	:		ゾーン・表示	u2					FURI-MM
15	:		ゾーン・表示	u2					FURI-DD
16	--FURI-MMDD End Redefines								
17	:		ゾーン・表示	u4					TO-BANK-NUMBER
18	:		Ank	20					TO-BANK-NAME
19	:		ゾーン・表示	u3					TO-BRANCH-NUMBER
20	:		Ank	20					TO-BRANCH-NAME
21	Loop 3								
22	:		ゾーン・表示	u5					ACCOUNT
23	EndLoop								
24	:		Ank	53					FILLER
25	--YOKIN-REC End Redefines								
26	--END-REC Redefines REC-BODY								
27	:		Ank	119					FILLER
28	--END-REC End Redefines								
29	EndProc								

◆注意　----　データ名に含まれる全角スペースやタブ

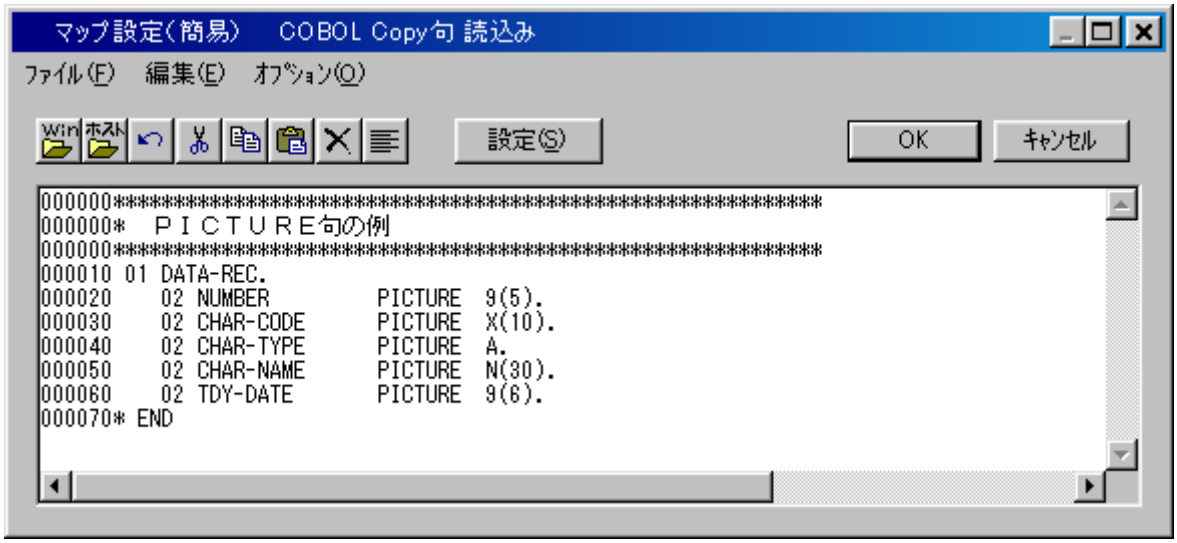
データ名などの利用者語に全角スペースやタブが含まれていた場合、正しく展開できない場合があります。あらかじめ、Ｃｏｐｙ句読み込みウインドウで編集／削除してください。

■Ｃｏｐｙ句読み込みの具体例

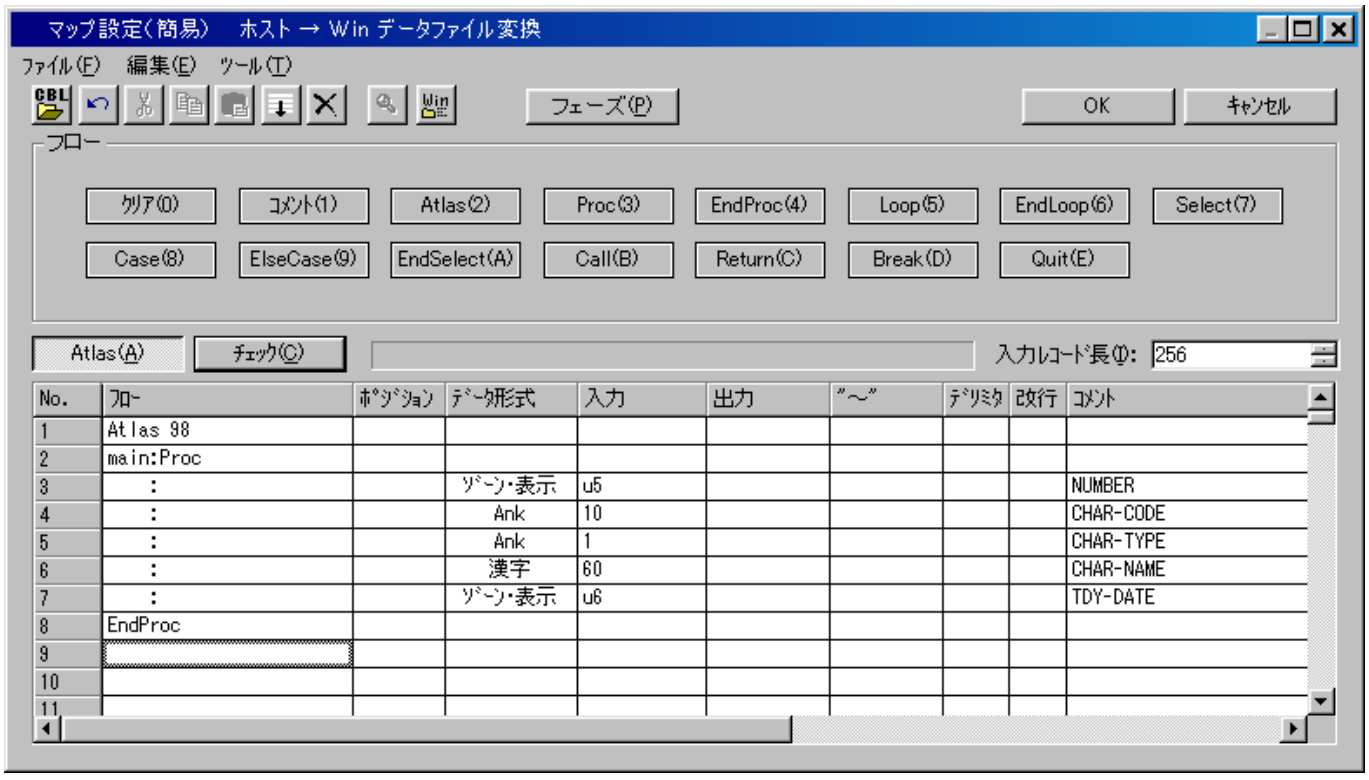
ＣＯＢＯＬの句がどのようにＡｔｌａｓ／Ｍａｐ文に対応しているか、ホスト→Ｗｉｎデータファイル変換を例にとって見てみましょう。

●ＰＩＣＴＵＲＥ句

ＰＩＣＴＵＲＥ句を含む１文が、マップ設定の１項目に対応しています。ＰＩＣＴＵＲＥ句の文字列に応じたデータ形式と入力幅になります。ＰＩＣＴＵＲＥ句でデータ名を指定した場合、マップ設定（簡易）ウインドウのコメント欄に、そのデータ名がセットされます。

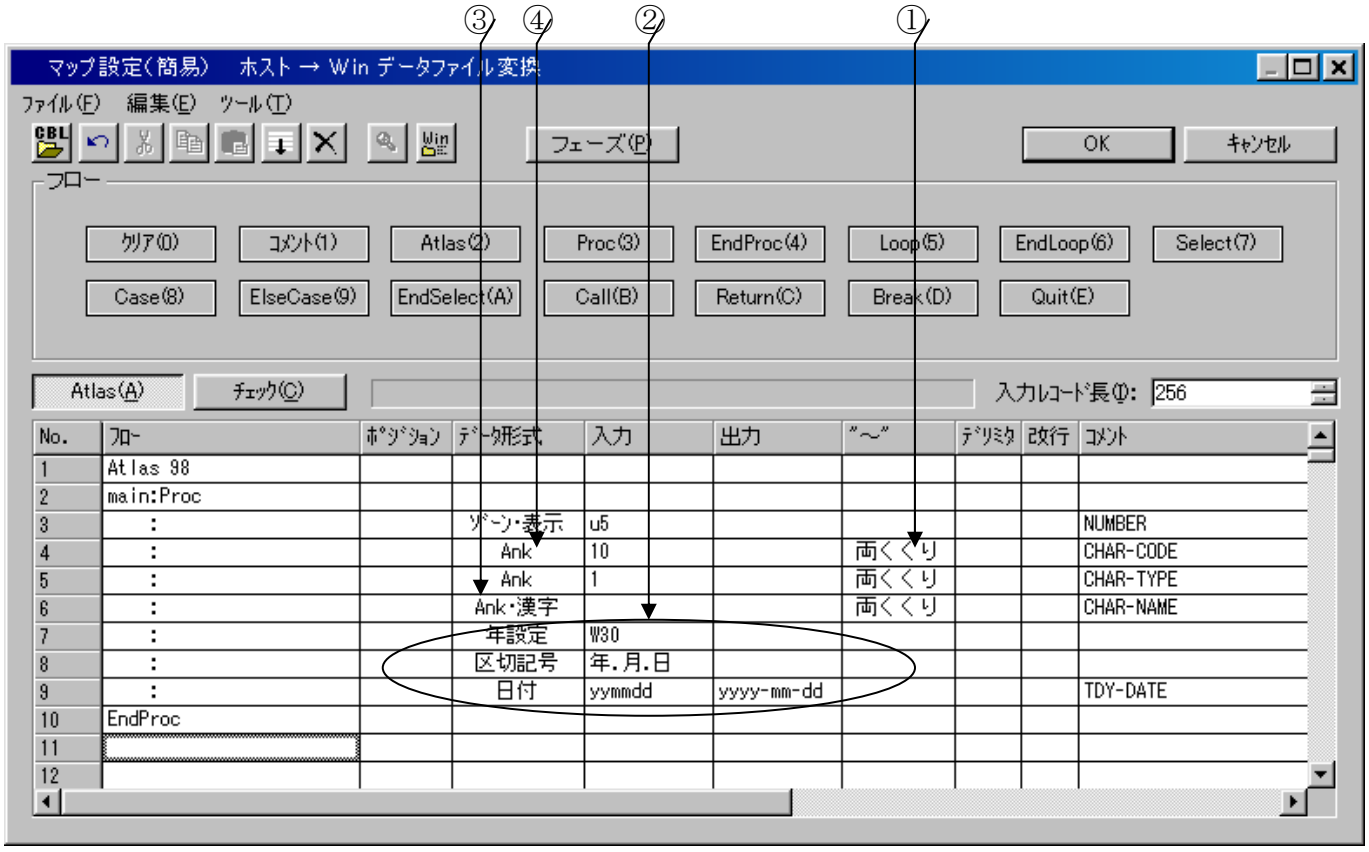


↓



自動展開によって、Ｍａｐ文のデータ形式と入力幅のみ生成されます。必要に応じてその他の設定を手動で行ってください。

一般的に、次のような修正をする場合がよくあります。



①デリミタ形式へ変換するときは、データ形式がＡｎｋ、漢字、Ａｎｋ・漢字などの文字列の項目に、引用符を付けることがよくあります。

②日付データ項目がある場合には、年設定、日付区切り設定、日付設定に変更してください。

③ＰＩＣＴＵＲＥ句の日本語項目“Ｎ”は、漢字項目に自動展開されますが、ＫＩ／ＫＯを含むときはＡｎｋ・漢字項目に変更してください。

④ＰＩＣＴＵＲＥ句の英数字項目“Ｘ”は、Ａｎｋ項目に自動展開されます。
“Ｘ”に日本語文字列を含んでいるときは（おもにＷｉｎｄｏｗｓＣＯＢＯＬの場合）、漢字項目（ＫＩ／ＫＯを含むときはＡｎｋ・漢字項目）に変更してください。

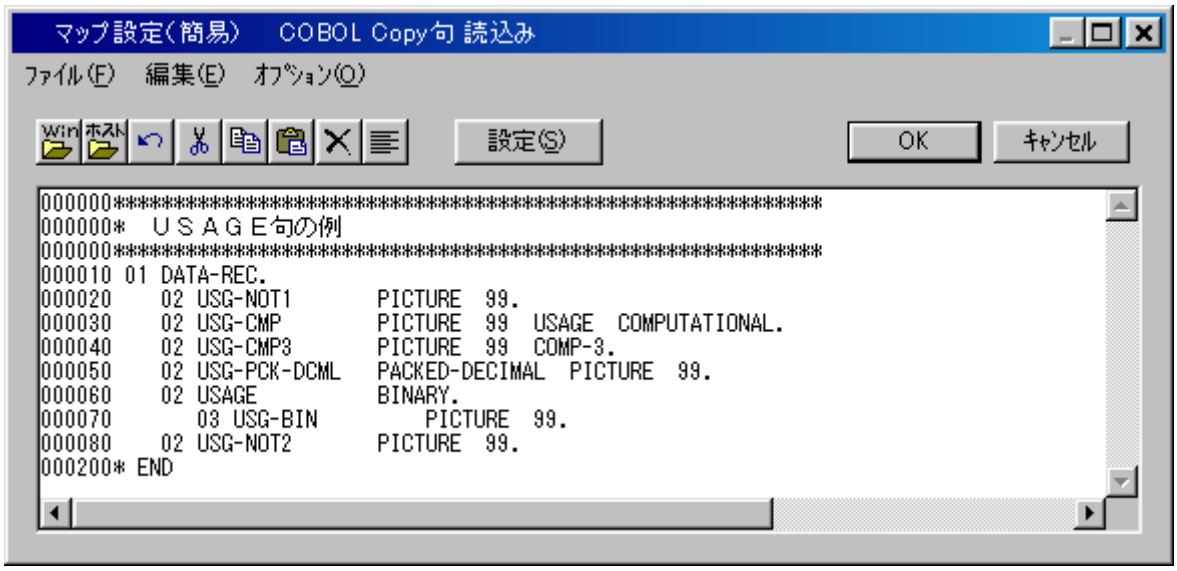
◆注意　----　数字項目の“Ｐ”

ＰＩＣＴＵＲＥ句の数字項目の“Ｐ”はＦ＊ＴＲＡＮ＋では扱えないデータ形式です。ＣＯＢＯＬのソースファイル中に数字項目の“Ｐ”が記述されている場合、数字項目の“９”に置き換えて展開しています。そのため、正しいデータにならないことがあります。注意してください。

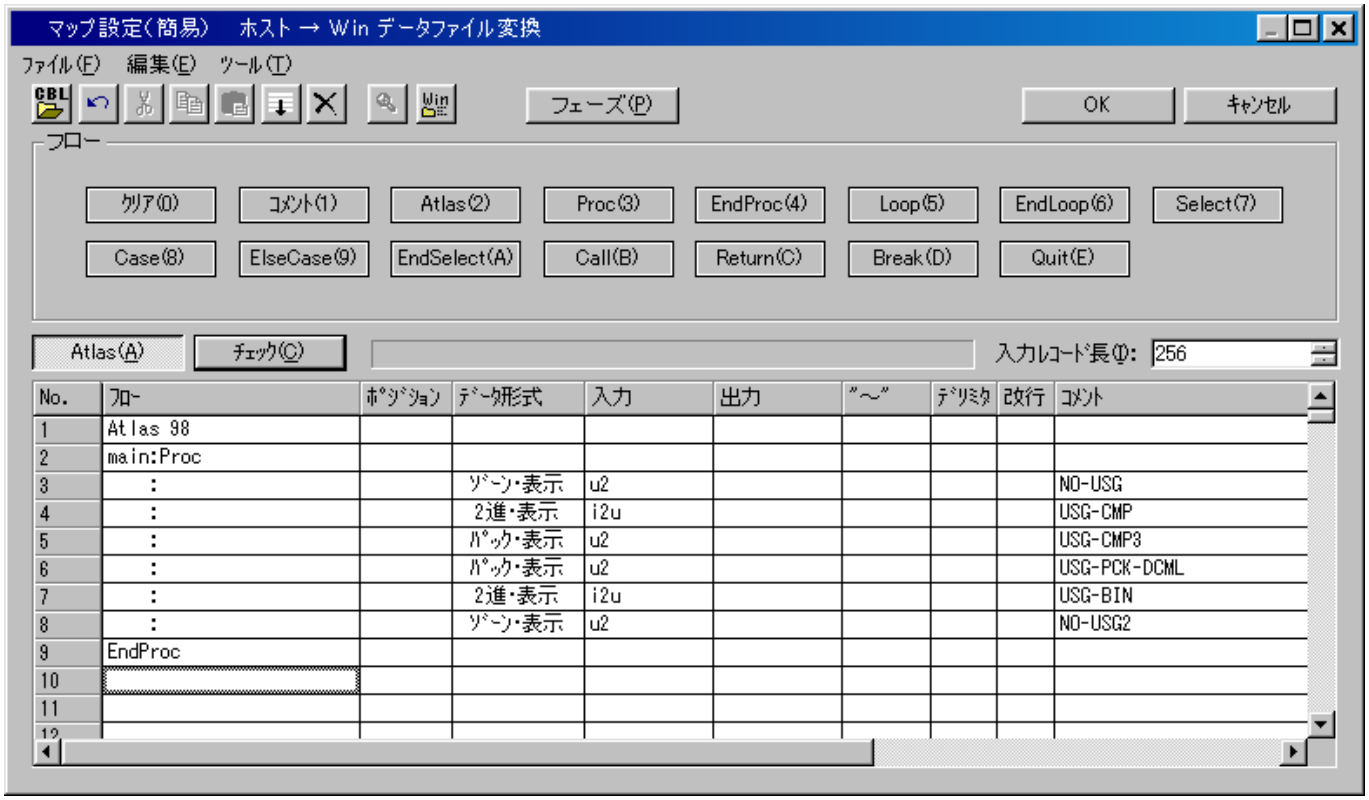
●ＵＳＡＧＥ句

ＰＩＣＴＵＲＥ句から生成されたＭａｐ文のデータ形式と入力幅に、ＵＳＡＧＥ句の指定が反映されます。ＵＳＡＧＥ句を集団項目に指定した場合、その集団に含まれるＰＩＣＴＵＲＥ句のデータ全てに反映されます。

ＰＩＣＴＵＲＥ句を含む文中の“ＵＳＡＧＥ”や“ＩＳ”は省略しても構いません。



↓



＜注意＞　ＵＳＡＧＥ句の指定では、ときどき、意図したものと違うデータ形式と入力幅に展開されることがあります。その場合は、手動で変更してください。

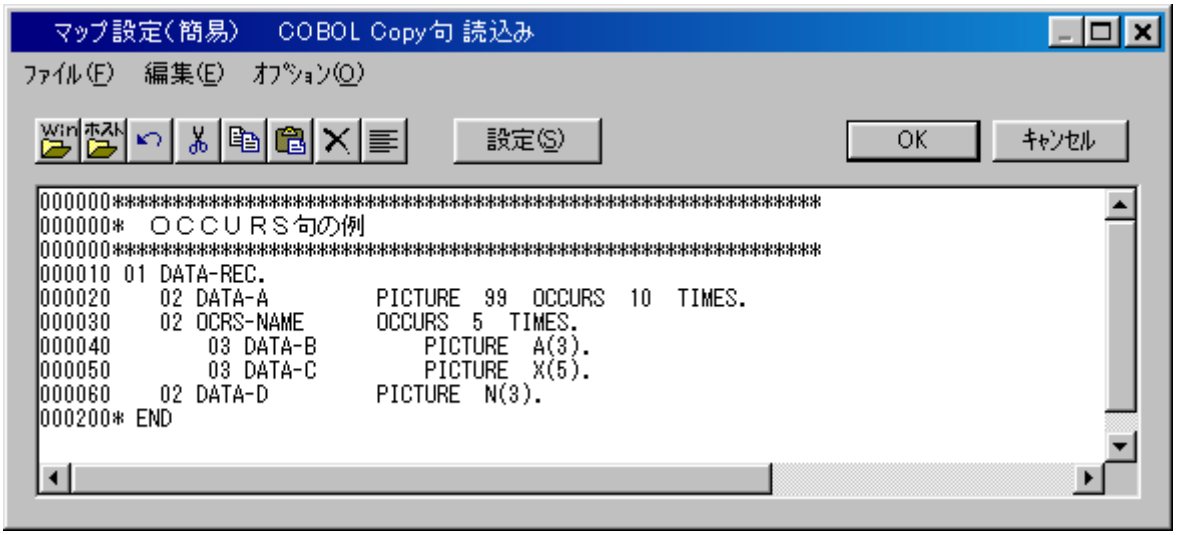
●ＯＣＣＵＲＳ句

Ａｔｌａｓ９８のＬｏｏｐ文とくり返し回数、ＥｎｄＬｏｏｐ文に対応しています。くり返し回数は、

ＯＣＣＵＲＳ　ｎ　ＴＩＭＥＳ
ＯＣＣＵＲＳ　ｍ　ＴＯ　ｎ　ＴＩＭＥＳ

という記述の場合、両方とも同じ＜ｎ＞回になります。

ＯＣＣＵＲＳ句に指定したデータ名は、Ｌｏｏｐ文の構造名となります。ただし、構造名で使用できない文字が含まれている場合、Ａｔｌａｓ文に展開される際に自動的に削除されます。



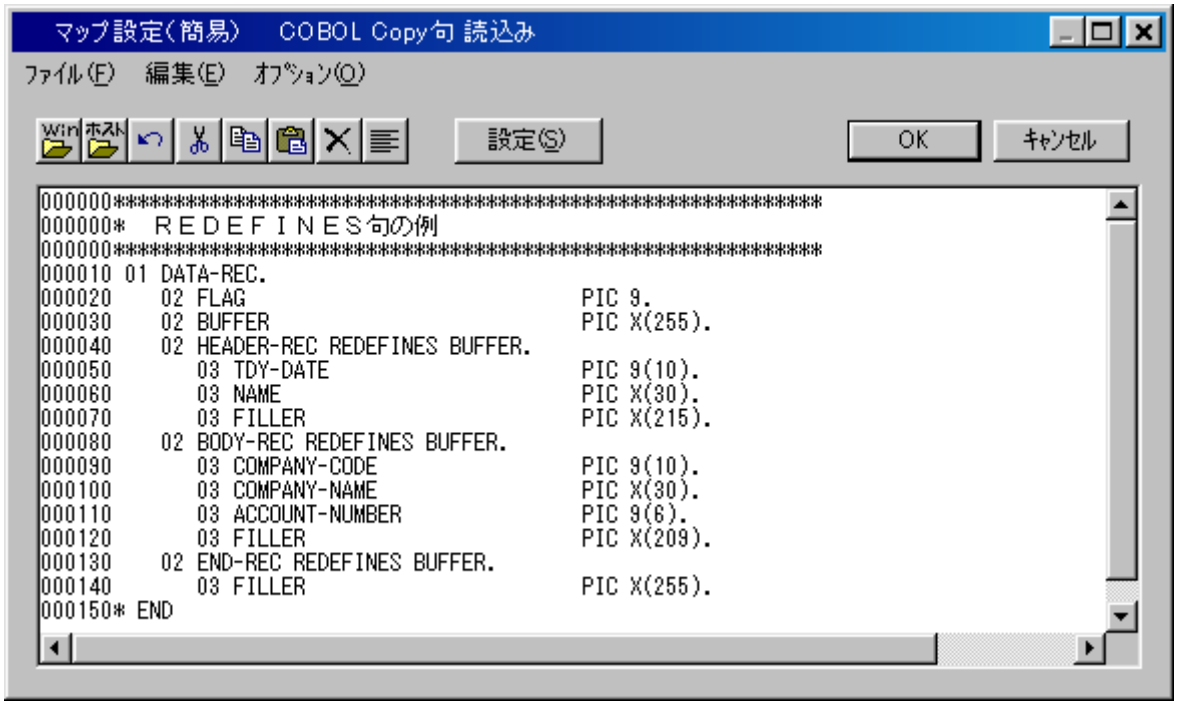
↓

●ＲＥＤＥＦＩＮＥＳ句

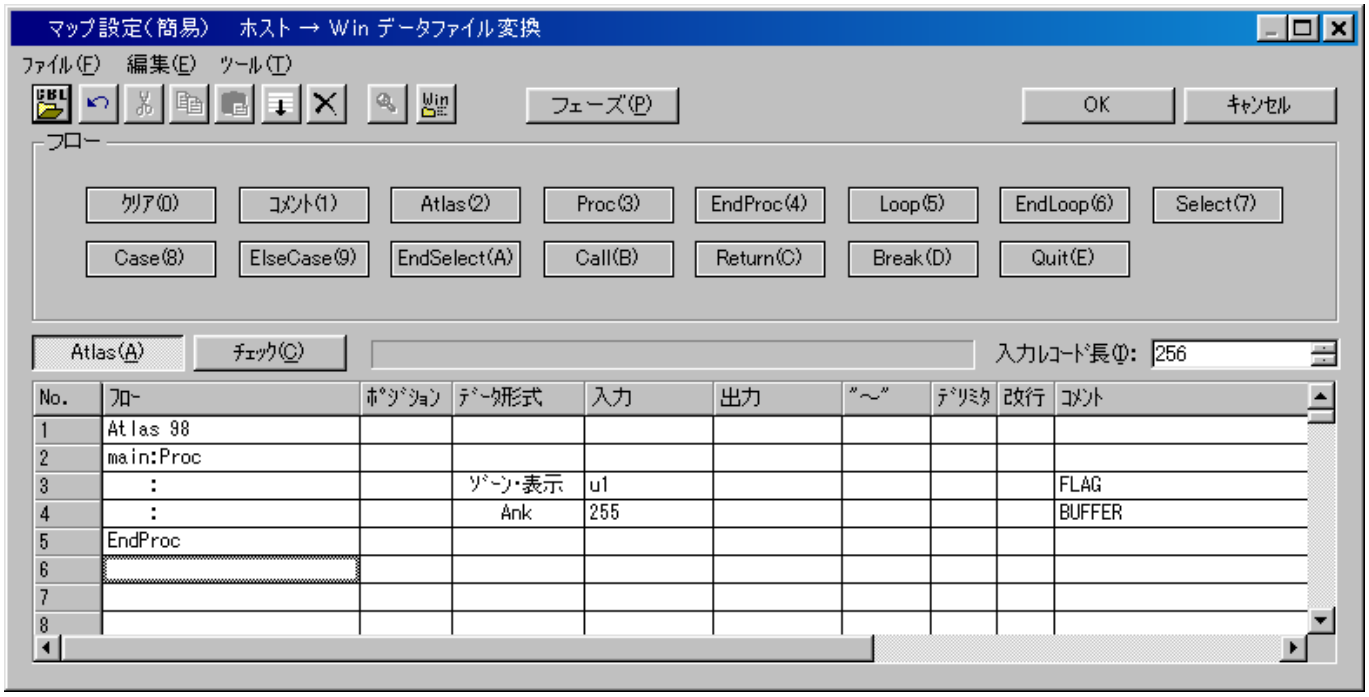
ＲＥＤＥＦＩＮＥＳ句で再定義した項目を無視することも､マップに展開することもできます。Ｃｏｐｙ句読み込み設定で、どちらかを指定してください。

ＲＥＤＥＦＩＮＥＳ句を生成しない場合

ＲＥＤＥＦＩＮＥＳ句で再定義した項目は無視されます。



↓



ＲＥＤＥＦＩＮＥＳ句を生成する場合

ＲＥＤＥＦＩＮＥＳ句で再定義した項目をマップに展開します。再定義した項目の初めと終わりに、コメントが生成されます。たとえば、

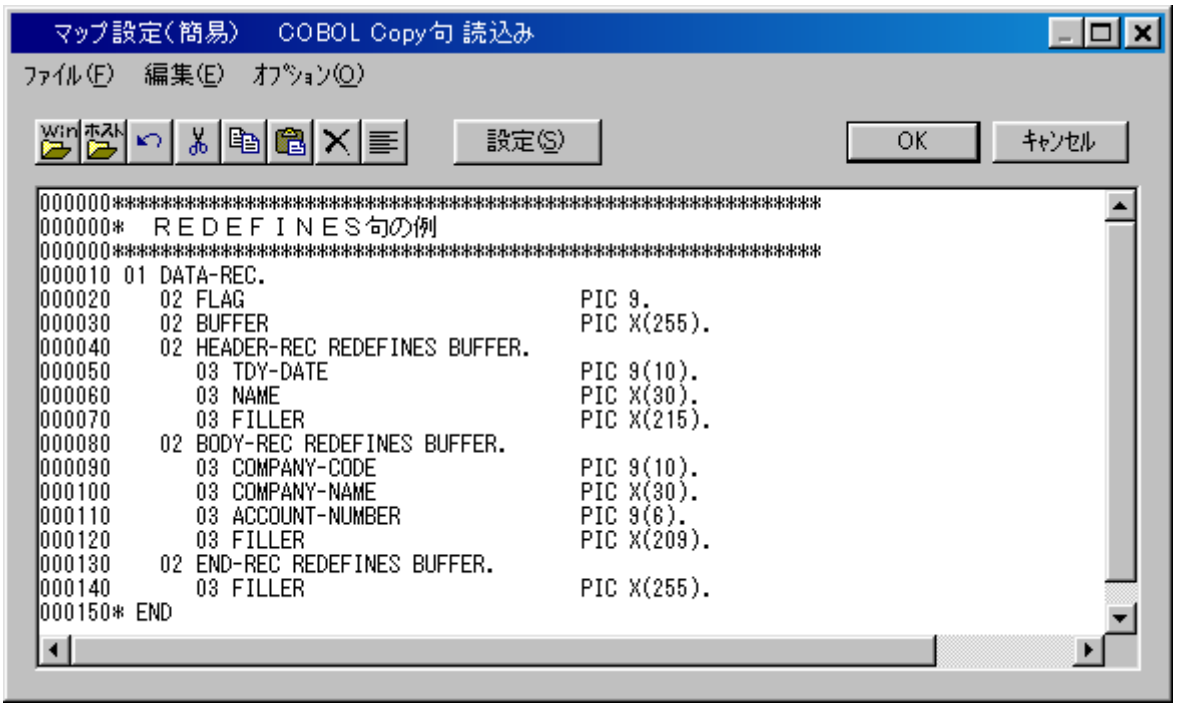
Ａ・・・ＲＥＤＥＦＩＮＥＳ句のデータ名（省略可）
Ｂ・・・再定義される項目のデータ名

とすると、

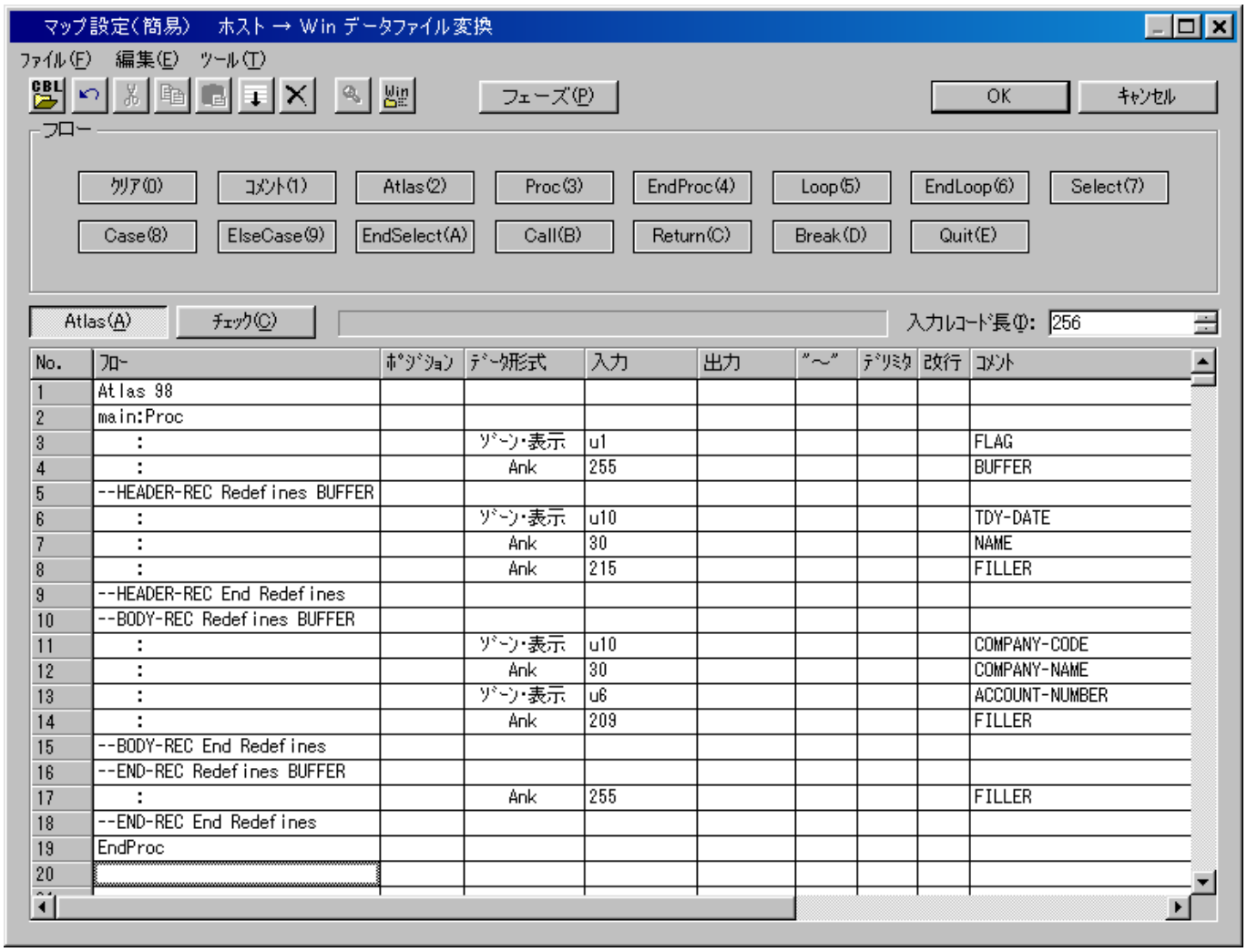
```
-- A Redefines B
                :
                :    再定義した項目のマップ
                :
-- A End Redefines
```

というコメントになります。

それでは、ＲＥＤＥＦＩＮＥＳ句を生成する場合を見てみましょう。例として使用しているＣＯＢＯＬのソースは、ＲＥＤＥＦＩＮＥＳ句を生成しない場合の例と同じものです。



↓



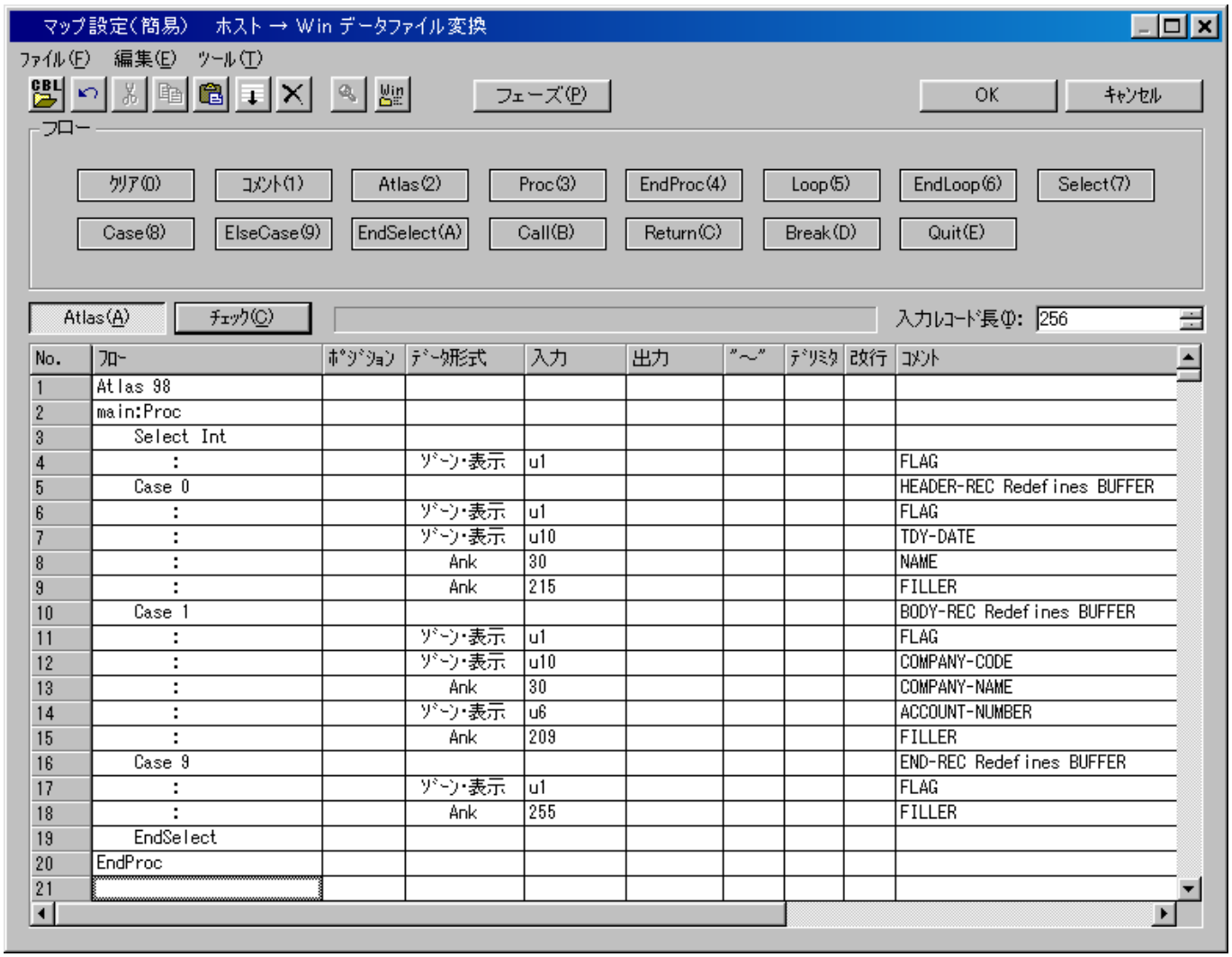
ＲＥＤＥＦＩＮＥＳ句の場合、コメントが生成されるだけなので、このままでは正しい設定にはなりません。このコメントを頼りに、マップ設定（簡易）ウインドウにて、手動で修正をする必要があります。

では実際に、先頭項目の「ＦＬＡＧ」の値を見て、ヘッダレコード／ボディレコード／エンドレコードに分岐するＡｔｌａｓ文にしてみましょう。手順はつぎのとおりです。

①行番号Ｎｏ．２と３の間にＳｅｌｅｃｔ文を追加する。
②Ｎｏ．４の「ＢＵＦＦＥＲ」の行を削除する。（通常、再定義される項目は変換しない。）
③Ｎｏ．５のＲｅｄｅｆｉｎｅｓ句開始のコメント行を消して、変わりにＣａｓｅ文を記述する。
④Ｎｏ．５と６の間にセレクタとして使用した「ＦＬＡＧ」のＭａｐ文を追加する。
⑤Ｎｏ．９のＲｅｄｅｆｉｎｅｓ句終了のコメント行を削除する。
⑥Ｎｏ．１０のＲｅｄｅｆｉｎｅｓ句開始のコメント行を消して、変わりにＣａｓｅ文を記述する。
⑦Ｎｏ．１０と１１の間にセレクタとして使用した「ＦＬＡＧ」のＭａｐ文を追加する。

⑧Ｎｏ．１５のＲｅｄｅｆｉｎｅｓ句終了のコメント行を削除する。
⑨Ｎｏ．１６のＲｅｄｅｆｉｎｅｓ句開始のコメント行を消して、変わりにＣａｓｅ文を記述する。
⑩Ｎｏ．１６と１７の間にセレクタとして使用した「ＦＬＡＧ」のＭａｐ文を追加する。
⑪Ｎｏ．１８のＲｅｄｅｆｉｎｅｓ句終了のコメント行を削除する。
⑫Ｎｏ．１９の前にＥｎｄＳｅｌｅｃｔ文を追加する。

以上の操作を行うと、つぎのようなマップになります。



◆注意　----　データ名の省略

ＲＥＤＥＦＩＮＥＳ句のデータ名は、省略されていても構いません。しかし、ＲＥＤＥＦＩＮＥＳ句が複数あったり、ＲＥＤＥＦＩＮＥＳ句で再定義した項目の中に２重でＲＥＤＥＦＩＮＥＳ句が記述されていたりする場合には、分かりにくくなるので注意してください。

●見だしレコード

見だしレコードとは、いわゆるヘッダのことです。通常、ヘッダを生成・付加する場合には、そのためのマップを手動で書かなくてはなりません。しかし、ＣＯＢＯＬのソースファイルを読み込んでマップに展開する場合には、自動的に生成することができます。

これまでに見てきた例では、見だしレコード用のマップは生成されていません（本体レコード用のマップのみ生成されています)。

見だしレコード用のマップを生成するには、Ｃｏｐｙ句読み込み設定ウインドウで、見だしレコードを生成する（Ｉ）に指定してください。そうすると、

```
{Ｓｅｌｅｃｔ}
    ＢＹ　~ＳｙｓＰｈａｓｅ *1
{Ｃａｓｅ　１}
     :
     :　見だしレコード用マップ
     :
{Ｃａｓｅ　２}
     :
     :　（本体レコード用マップ　*2)
     :
{ＥｎｄＳｅｌｅｃｔ}
```

という文が自動的に追加されます。

見だしレコード用マップには、

Ａｎｋ・漢字　［’ＰＩＣＴＵＲＥ句のデータ名’］

というＭａｐ文が、ＰＩＣＴＵＲＥ句の項目数だけ生成されます。つまり、本体レコード用マップに展開された、すべてのＰＩＣＴＵＲＥ句のデータ名が挿入されます。ただし、データ名が省略されている場合、

Ａｎｋ・漢字　［’’］

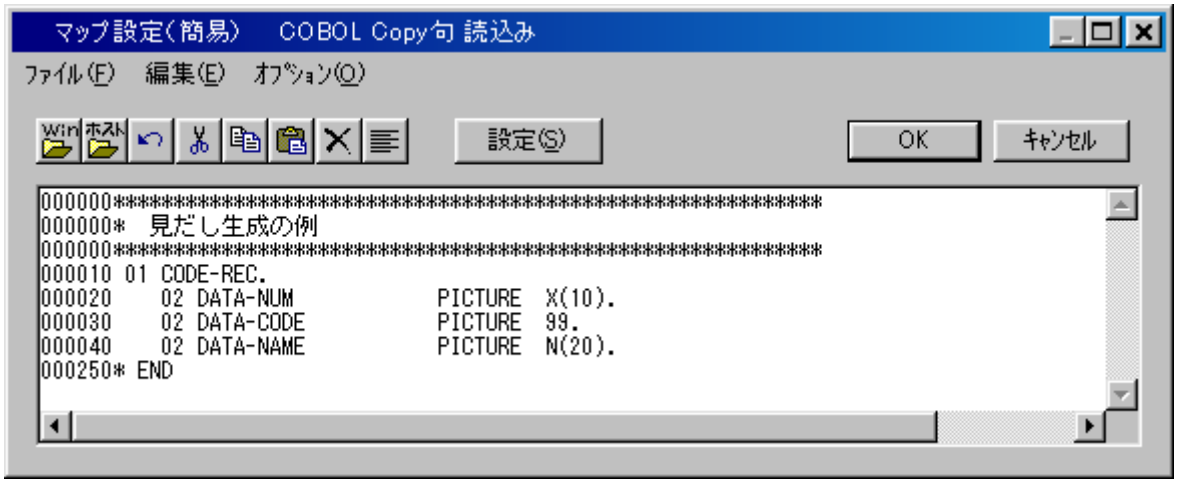
となります。また、ＯＣＣＵＲＳ句でくり返しが指定されている場合は、本体レコード用マップと同様に、Ｌｏｏｐ文が生成されます。

*1）ＳｙｓＰｈａｓｅの使い方についての詳細は、使用法のヘッダとトレーラの挿入方法を参照してください。

*2）本体レコード用マップは、見だしレコードを生成しない場合と共通のマップです。

見だしレコードを生成する場合

それでは実際に、例を見てみましょう。



↓

◆注意　----　ＲＥＤＥＦＩＮＥＳ句と見だしレコード

ＲＥＤＥＦＩＮＥＳ句を生成する場合は、整合性がなくなるため、見だしレコードを生成しても意味がありません。Ｃｏｐｙ句読み込み設定で、ＲＥＤＥＦＩＮＥＳ句を生成する（Ｇ）と指定したときは、見だしレコードを生成する（Ｉ）は指定できないようになっています。

◆注意　----　フェーズの設定

見だしレコードを生成するには、マップ設定の他にフェーズの設定を行う必要があります。見だしレコードは先頭に１行付け加えるのが一般的なため、オープン時の呼び出し回数を自動的に１に変更しています。見出しレコードを複数行付けたいときなどは、Ｃｏｐｙ句読み込み後、手動でフェーズの設定を行ってください。

◆注意　----　出力幅を指定する

見だしレコードを生成した場合、出力幅が省略されていますが、定数挿入の省略時の出力幅は**見た目の文字数×２**になるので、出力コードがＵＣＳ－２以外の場合は明示的に出力幅を指定する必要があります。

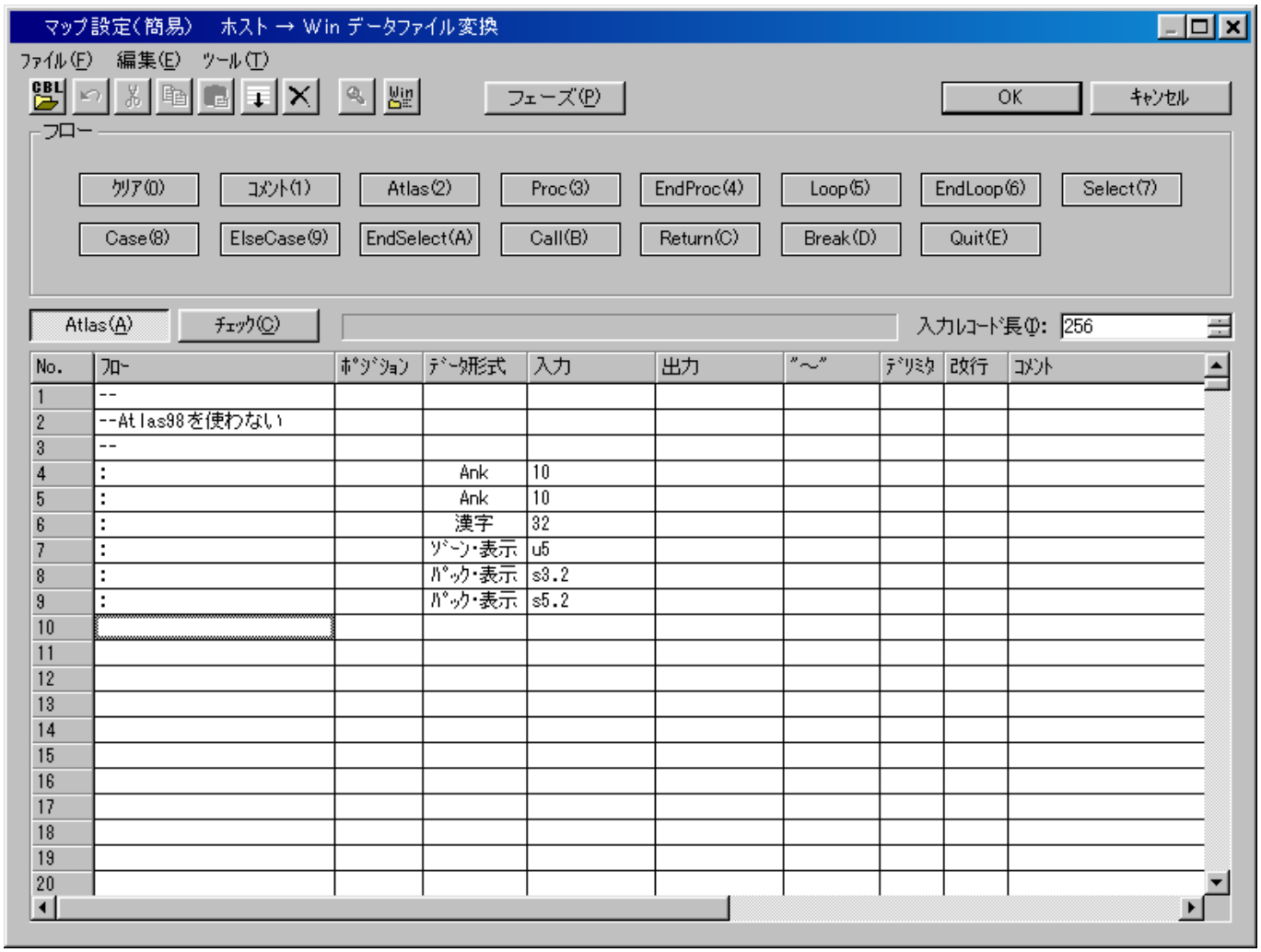
第4章

Ａｔｌａｓ９８の利用例

４．１　Ａｔｌａｓ９８の利用例

■従来互換

Ａｔｌａｓ９８モードで、Ａｔｌａｓ文を使わない例を示します。



＜パラメータファイルの記述＞

```
--
--  Ａｔｌａｓ９８を使わないＭａｐ
--
    ａｎｋ          １０
    ａｎｋ          １０
    ｋａｎｊｉ      ３２
    ｚｏｎｅｄｉｓｐ    ｕ５
    ｐａｃｋｄｉｓｐ    ｓ３．２
    ｐａｃｋｄｉｓｐ    ｓ５．２
```

■Ａｔｌａｓ９８化

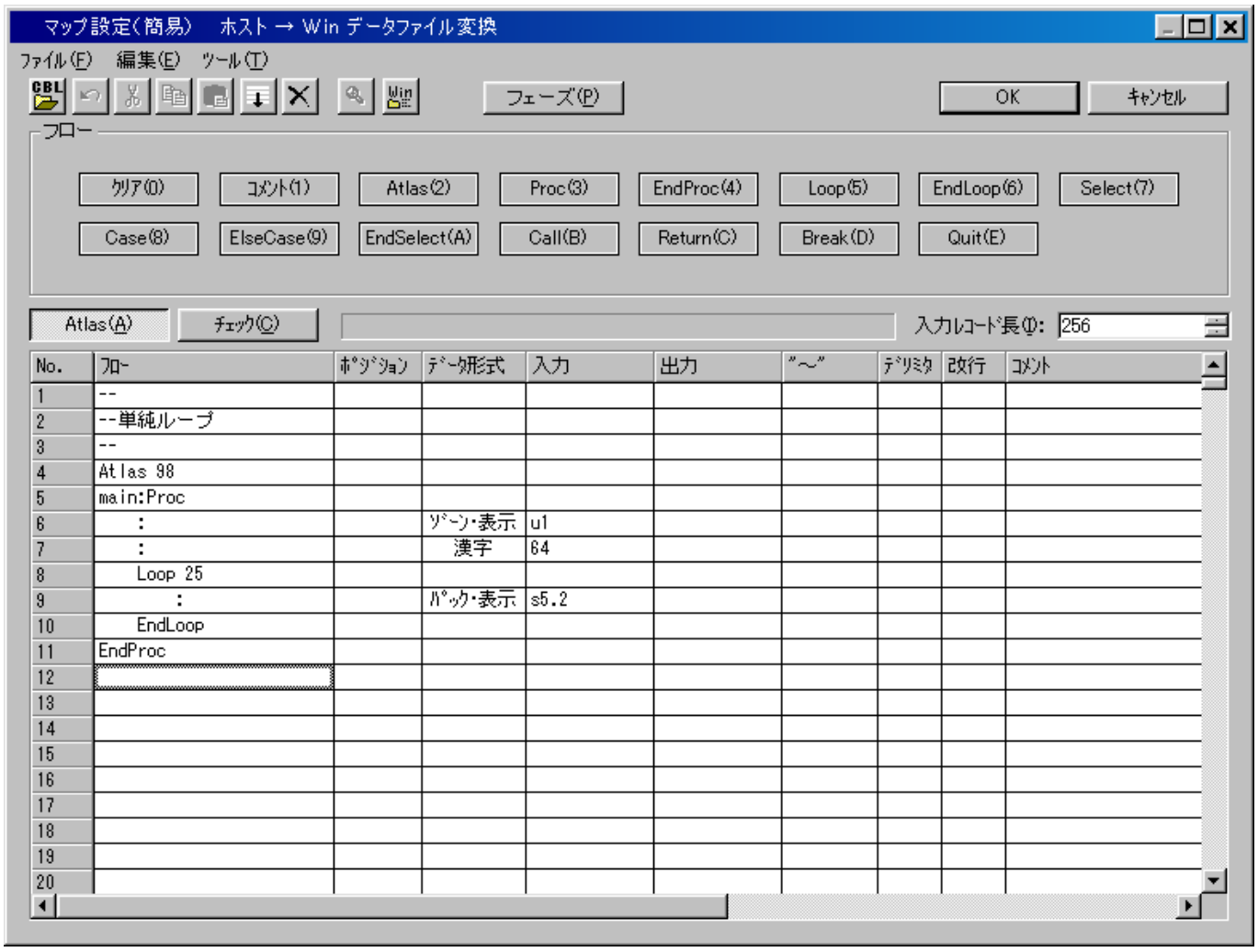
Ａｔｌａｓ９８を、最も単純に使った例です。

＜パラメータファイルの記述＞

```
--
--　Ａｔｌａｓ９８を、単純に使う
--
 {Ａｔｌａｓ　９８}

 {ｍａｉｎ：Ｐｒｏｃ}
    ａｎｋ　　　１０
    ａｎｋ　　　１０
    ｋａｎｊｉ　　３２
    ｚｏｎｅｄｉｓｐ　ｕ５
    ｐａｃｋｄｉｓｐ　ｓ３．２
    ｐａｃｋｄｉｓｐ　ｓ５．２
 {ＥｎｄＰｒｏｃ}
```

■単純なくり返し

レコードの後半が、同じパターンを２５回くり返します。

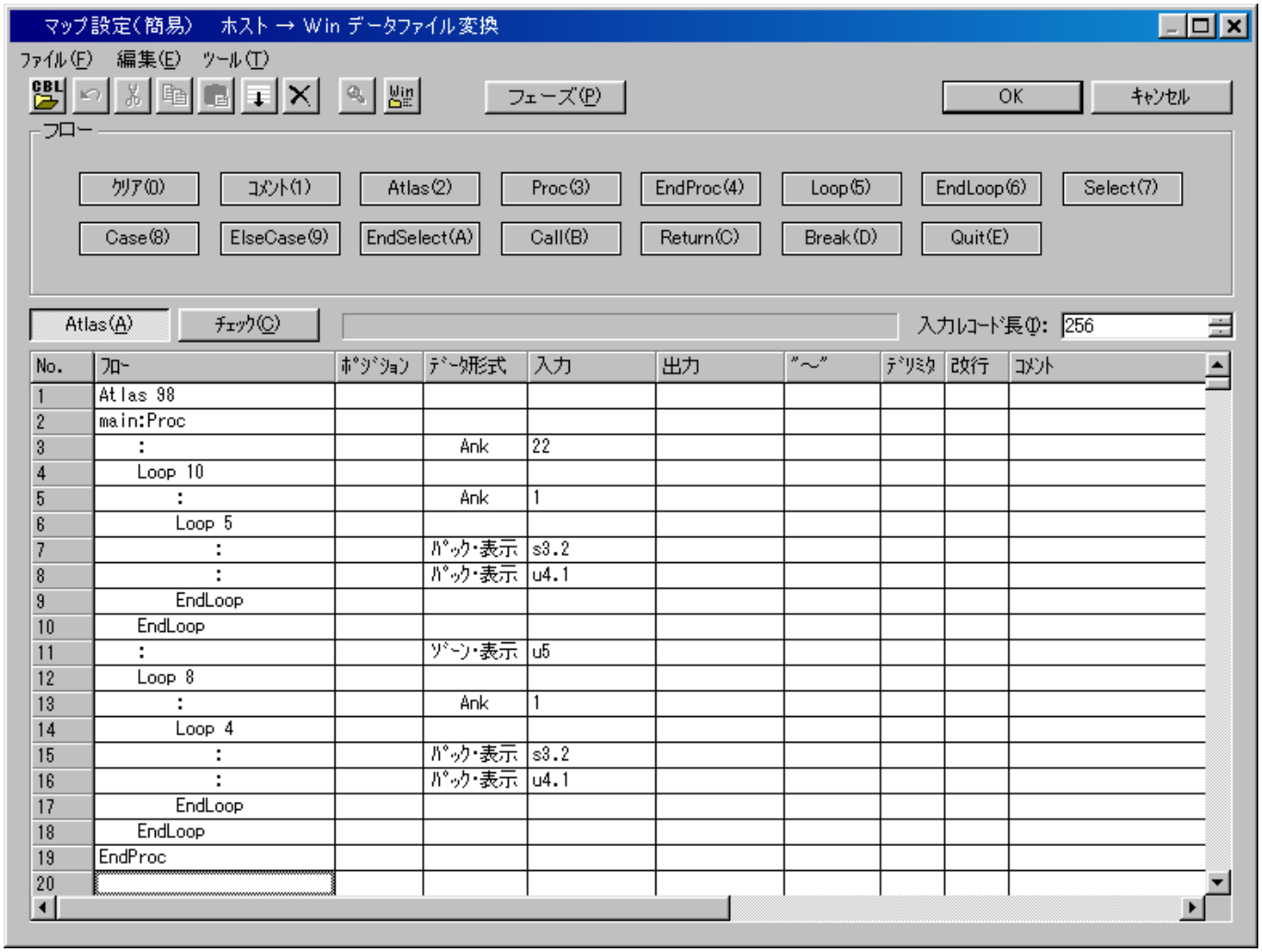


＜パラメータファイルの記述＞

```
--
--単純ループ
--
 {Ａｔｌａｓ　９８}

 {ｍａｉｎ：Ｐｒｏｃ}
    ｚｏｎｅｄｉｓｐ　　ｕ１
    ｋａｎｊｉ　　６４
     {Ｌｏｏｐ　２５}
        ｐａｃｋｄｉｓｐ　　ｓ５．２
     {ＥｎｄＬｏｏｐ}
 {ＥｎｄＰｒｏｃ}
```

■２重ループ

単純な２重ループが２回あります。



＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
  ａｎｋ　　２２
  {Ｌｏｏｐ　１０}
    ａｎｋ　　１
    {Ｌｏｏｐ　５}
      ｐａｃｋｄｉｓｐ　ｓ３．２
      ｐａｃｋｄｉｓｐ　ｕ４．１
    {ＥｎｄＬｏｏｐ}
  {ＥｎｄＬｏｏｐ}
  ｚｏｎｅｄｉｓｐ　ｕ５
  {Ｌｏｏｐ　８}
    ａｎｋ　　１
    {Ｌｏｏｐ　４}
      ｐａｃｋｄｉｓｐ　ｓ３．２
      ｐａｃｋｄｉｓｐ　ｕ４．１
    {ＥｎｄＬｏｏｐ}
  {ＥｎｄＬｏｏｐ}
{ＥｎｄＰｒｏｃ}
```

■マルチレイアウト変換を１つの手続きで設定する

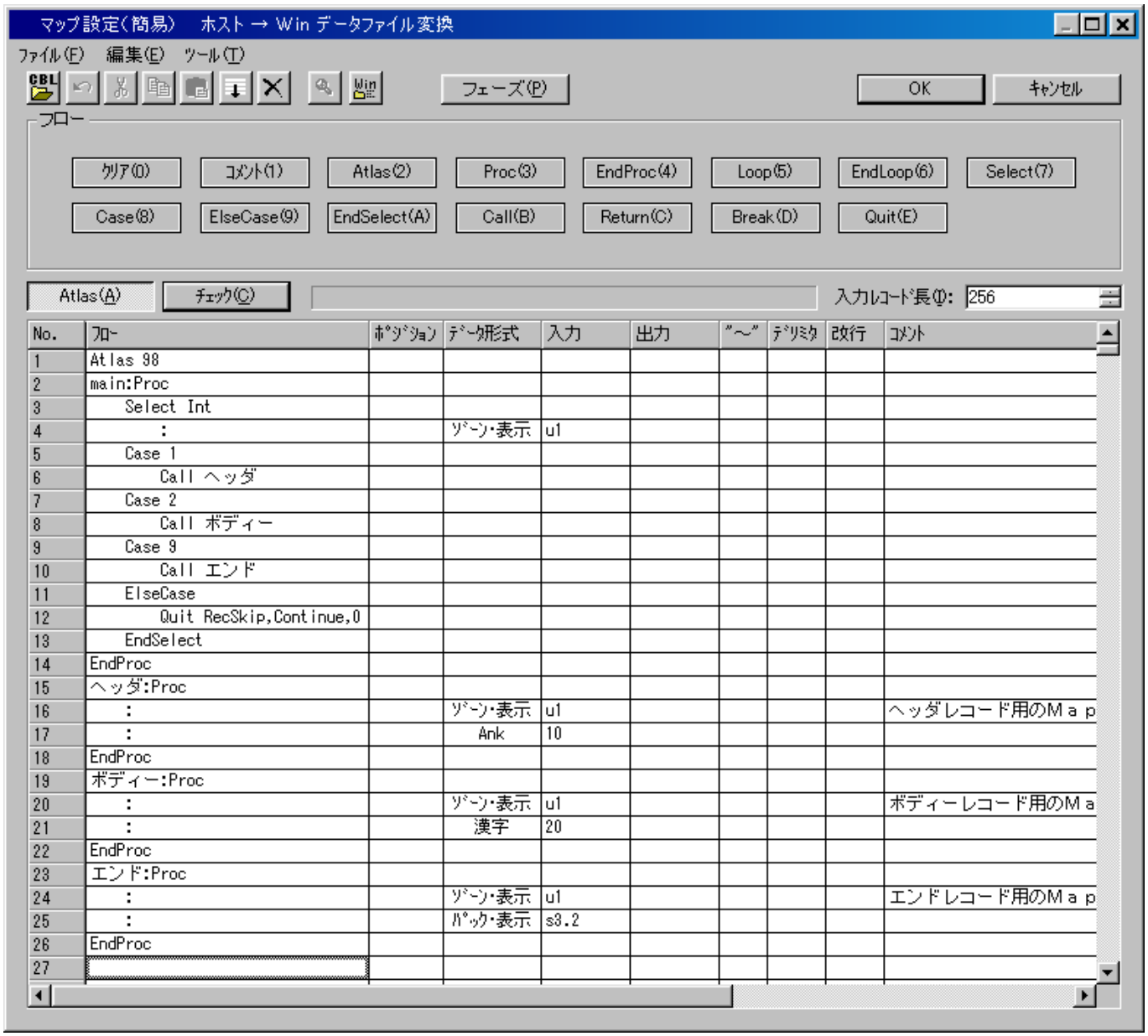
ホストからＷｉｎへ、先頭の項目が’１’のときヘッダレコードとして、’２ ’のときボディーレコードとして、’９ ’のときエンドレコードとして変換します。その他のレコードは無視します。

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
      ｚｏｎｅｄｉｓｐ　　ｕ１
   {Ｃａｓｅ　１}
      ｚｏｎｅｄｉｓｐ　　ｕ１　　--ヘッダレコード用のＭａｐ文
      ａｎｋ　　　　　１０
   {Ｃａｓｅ　２}
      ｚｏｎｅｄｉｓｐ　　ｕ１　　--ボディーレコード用のＭａｐ文
      ｋａｎｊｉ　　　２０
   {Ｃａｓｅ　９}
      ｚｏｎｅｄｉｓｐ　　ｕ１　　--エンドレコード用のＭａｐ文
      ｐａｃｋｄｉｓｐ　　ｓ３．２
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■複数の手続きを活用する

ホストからＷｉｎへ、先頭の項目が’１’のときヘッダレコードとして、’２ ’のときボディーレコードとして、’９ ’のときエンドレコードとして変換します。その他のレコードは無視します。

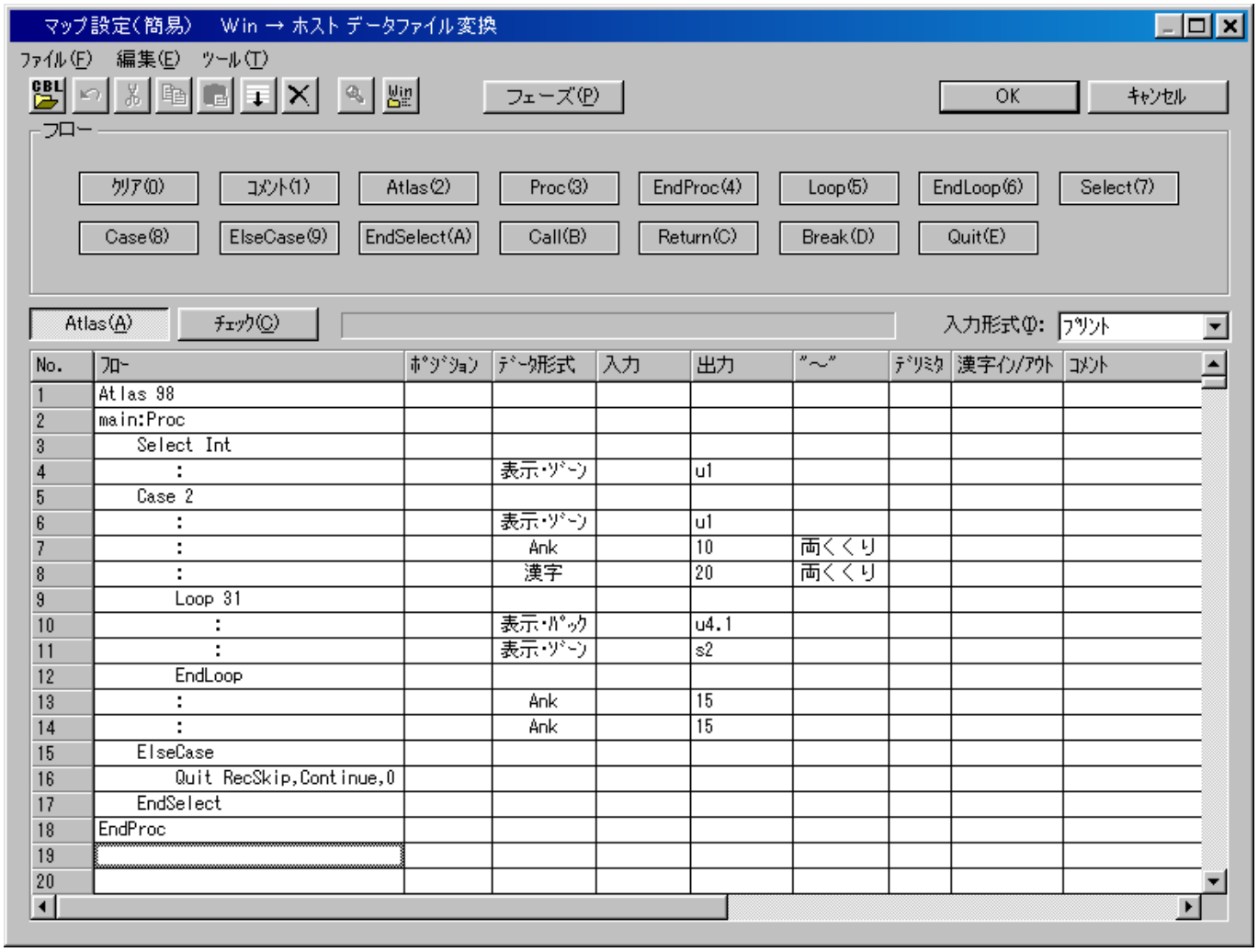


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
      ｚｏｎｅｄｉｓｐ　ｕ１
   {Ｃａｓｅ　１}
      {Ｃａｌｌ　ヘッダ}
   {Ｃａｓｅ　２}
      {Ｃａｌｌ　ボディー}
   {Ｃａｓｅ　９}
      {Ｃａｌｌ　エンド}
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}

{ヘッダ：Ｐｒｏｃ}
   ｚｏｎｅｄｉｓｐ　ｕ１　　--ヘッダレコード用のＭａｐ文
   ａｎｋ　　１０
{ＥｎｄＰｒｏｃ}

{ボディー：Ｐｒｏｃ}
   ｚｏｎｅｄｉｓｐ　ｕ１　　--ボディーレコード用のＭａｐ文
   ｋａｎｊｉ　　２０
{ＥｎｄＰｒｏｃ}

{エンド：Ｐｒｏｃ}
   ｚｏｎｅｄｉｓｐ　ｕ１　　--エンドレコード用のＭａｐ文
   ｐａｃｋｄｉｓｐ　ｓ３．２
{ＥｎｄＰｒｏｃ}
```

■ボディーレコードのみ変換する

Ｗｉｎからホストへ、先頭の項目が’２’のときだけボディーレコードとして変換します。その他のヘッダレコードやエンドレコードは無視します。



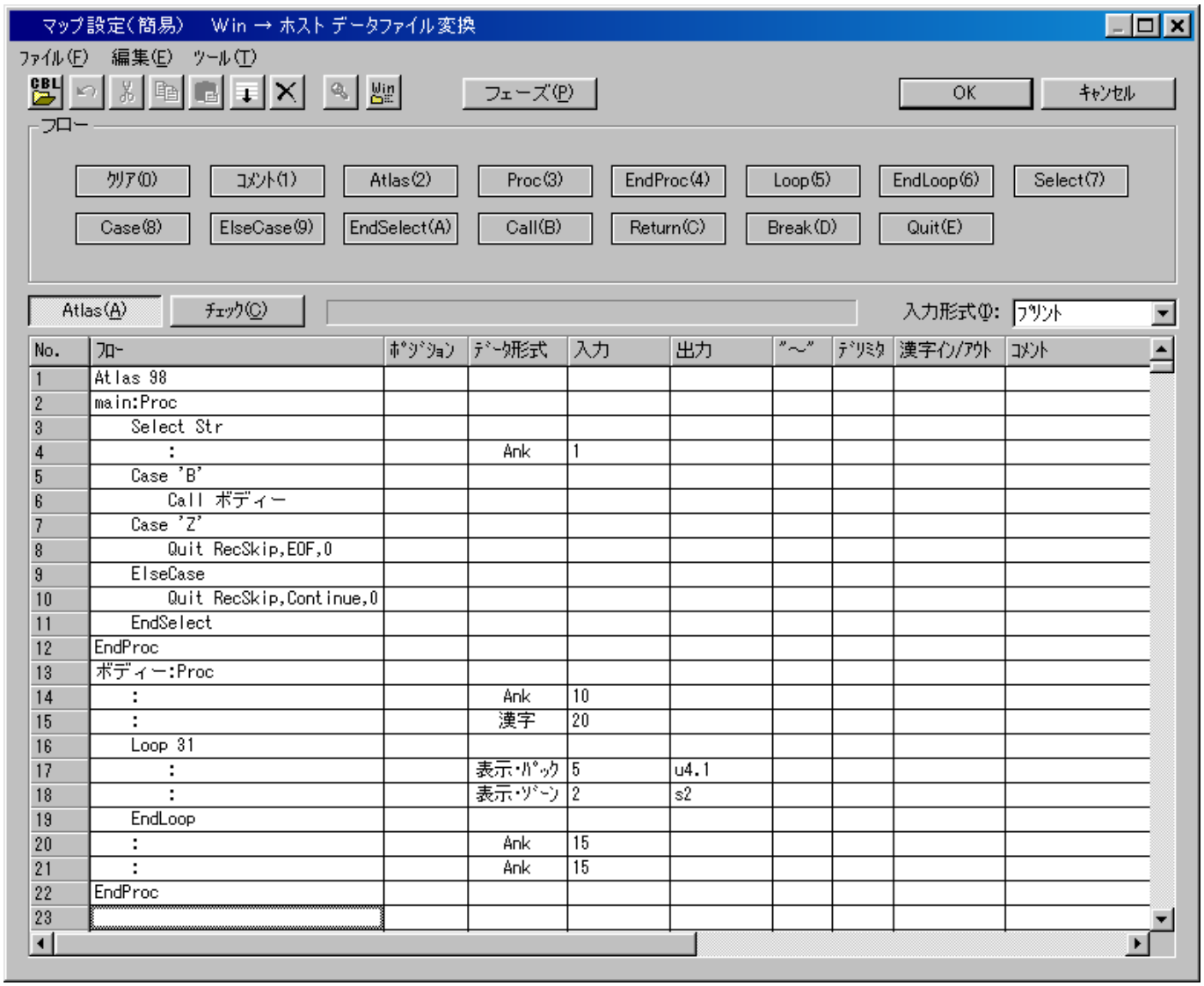
No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	漢字イン/アウト	コメント
1	Atlas 98								
2	main:Proc								
3	Select Int								
4	:		表示・ゾーン		u1				
5	Case 2								
6	:		表示・ゾーン		u1				
7	:		Ank		10	両くくり			
8	:		漢字		20	両くくり			
9	Loop 31								
10	:		表示・パック		u4.1				
11	:		表示・ゾーン		s2				
12	EndLoop								
13	:		Ank		15				
14	:		Ank		15				
15	ElseCase								
16	Quit RecSkip,Continue,0								
17	EndSelect								
18	EndProc								
19									
20									

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
      ｄｉｓｐｚｏｎｅ　　：ｕ１
   {Ｃａｓｅ　２}
      ｄｉｓｐｚｏｎｅ　　：ｕ１
     ”ａｎｋ　　　　：１０”
     ”ｋａｎｊｉ　　　：２０”
      {Ｌｏｏｐ　３１}
        ｄｉｓｐｐａｃｋ　　：ｕ４．１
        ｄｉｓｐｚｏｎｅ　　：ｓ２
      {ＥｎｄＬｏｏｐ}
       ａｎｋ　　　　：１５
       ａｎｋ　　　　：１５
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■変換を中断する

Ｗｉｎからホストへ、先頭の項目が’Ｂ’のときだけボディーレコードとして変換します。ただし、途中で’Ｚ’のレコードが現れたら、そこでこのファイルの変換を中断します。この’Ｚ’のレコードは変換結果に含めません。

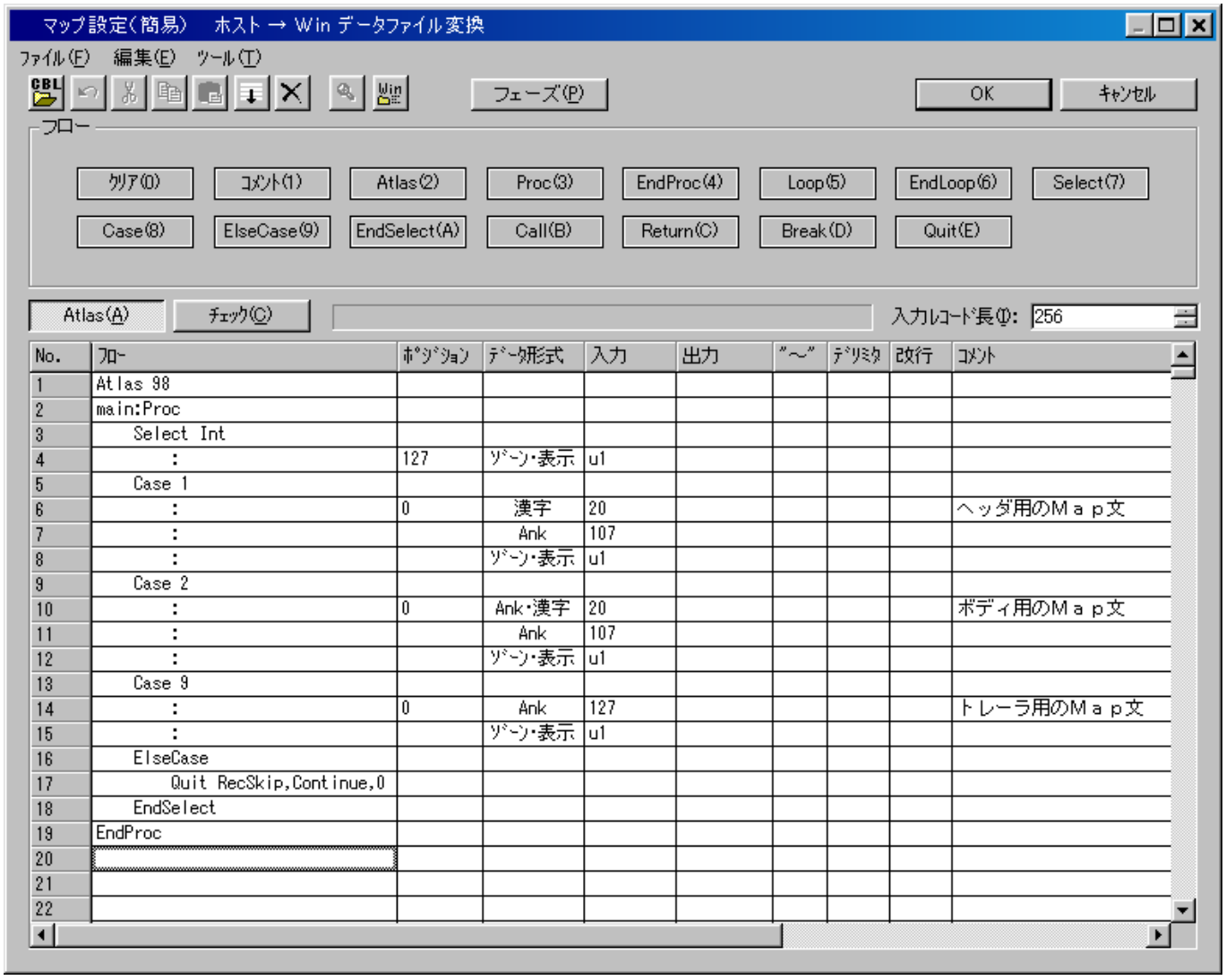


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｓｔｒ}
      ａｎｋ　　１
   {Ｃａｓｅ　’Ｂ’}
      {Ｃａｌｌ　ボディー}
   {Ｃａｓｅ　’Ｚ’}
      {Ｑｉｕｔ　ＲｅｃＳｋｉｐ，ＥＯＦ，　０}
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}

{ボディー：Ｐｒｏｃ}
   ａｎｋ　　１０
   ｋａｎｊｉ　　２０
   {Ｌｏｏｐ　３１}
      ｄｉｓｐｐａｃｋ　５：ｕ４．１
      ｄｉｓｐｚｏｎｅ　２：ｓ２
   {ＥｎｄＬｏｏｐ}
   ａｎｋ　　１５
   ａｎｋ　　１５
{ＥｎｄＰｒｏｃ}
```

■最初にレコード末尾付近の区分項目を見る

ホストからＷｉｎへ、レコードの末尾付近（１２８桁目）にある区分項目が’１’のときヘッダレコードとして、’２’のときボディーレコードとして、’９’のときエンドレコードとして変換します。その他のレコードは無視します。

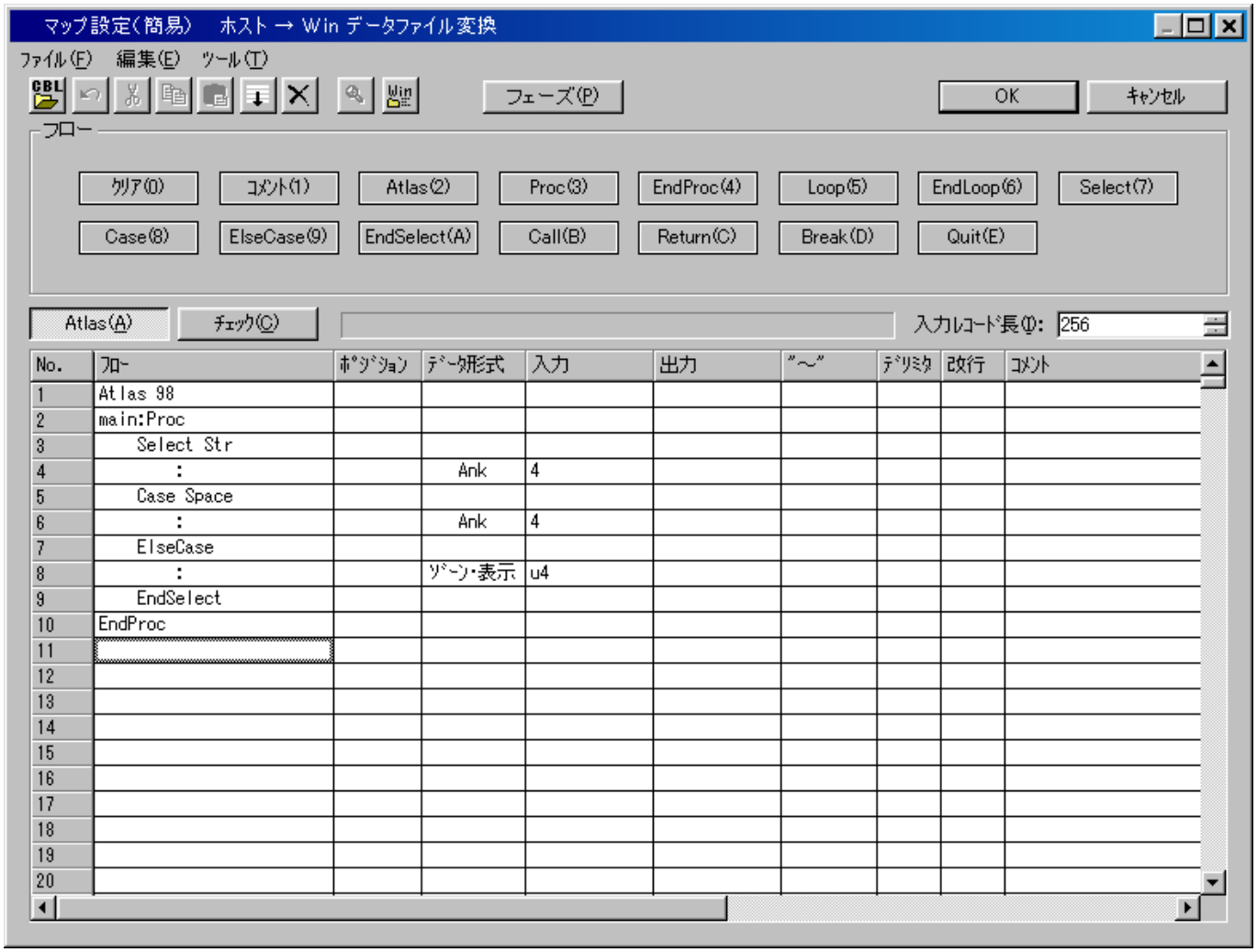


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     @１２７　ｚｏｎｅｄｉｓｐ　　ｕ１
   {Ｃａｓｅ　１}
     @０　　ｋａｎｊｉ　　２０　　--ヘッダ用のＭａｐ文
            ａｎｋ　　１０７
            ｚｏｎｅｄｉｓｐ　　ｕ１
   {Ｃａｓｅ　２}
     @０　　ｋａｎｊｉｍｉｘ　　２０　　--ボディ用のＭａｐ文
            ａｎｋ　　１０７
            ｚｏｎｅｄｉｓｐ　　ｕ１
   {Ｃａｓｅ　９}
     @０　　ａｎｋ　　１２７　　--トレーラ用のＭａｐ文
            ｚｏｎｅｄｉｓｐ　　ｕ１
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■空白とゾーン形式を変換し分ける

空白（空項目）と０を区別して変換します（空項目にゾーン・表示変換をかけると０になってしまうため、文字列型を指定して判定します）。

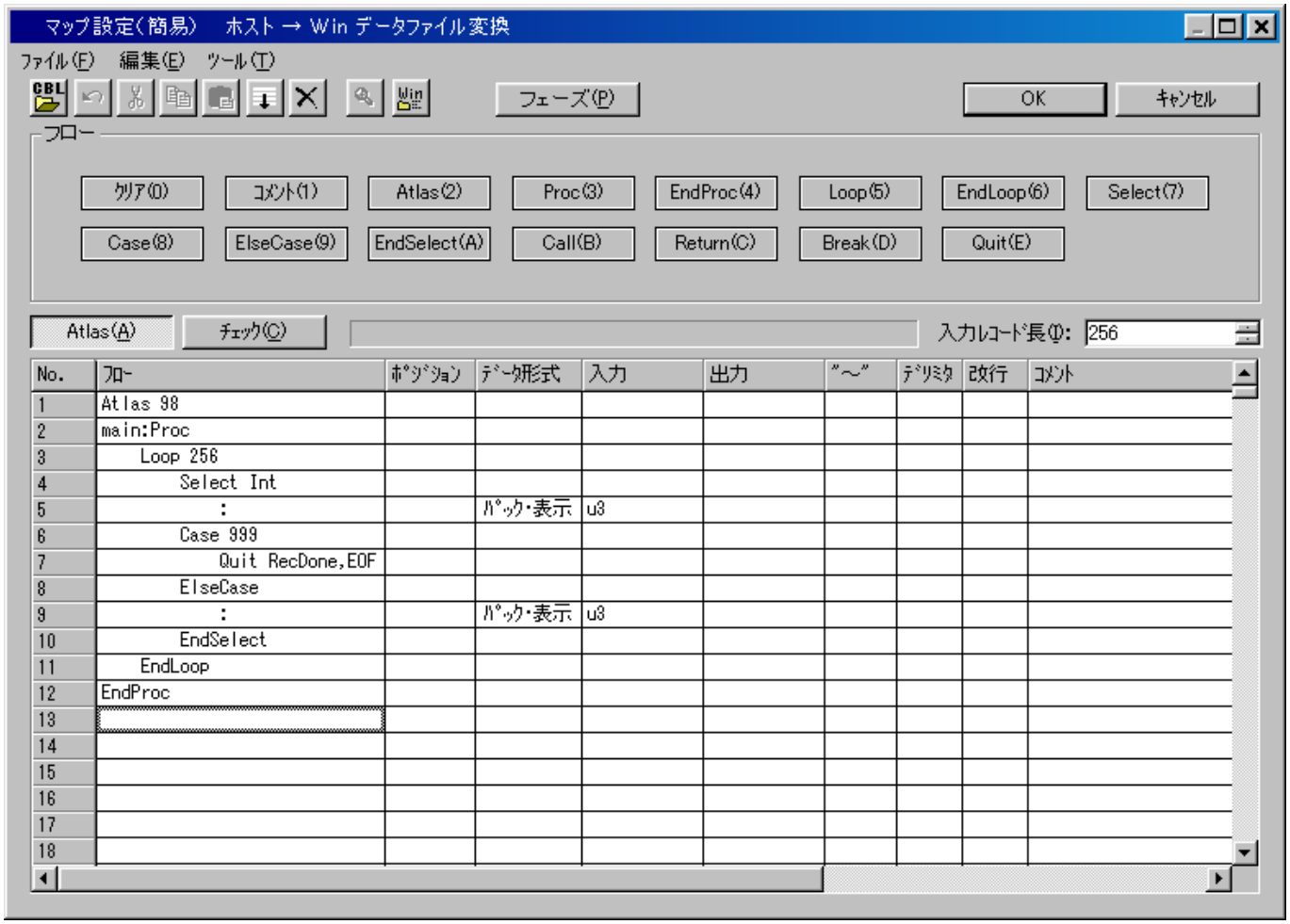


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
    {Ｓｅｌｅｃｔ　Ｓｔｒ}
        ａｎｋ              ４
    {Ｃａｓｅ　Ｓｐａｃｅ}
        ａｎｋ              ４
    {ＥｌｓｅＣａｓｅ}
        ｚｏｎｅｄｉｓｐ    ｕ４
    {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■９９９で変換を止める

パック形式の配列（２５６個／レコード）を変換します。そのとき、値が９９９の項目が見つかったら、その直前まででそのファイルの変換を中断します。

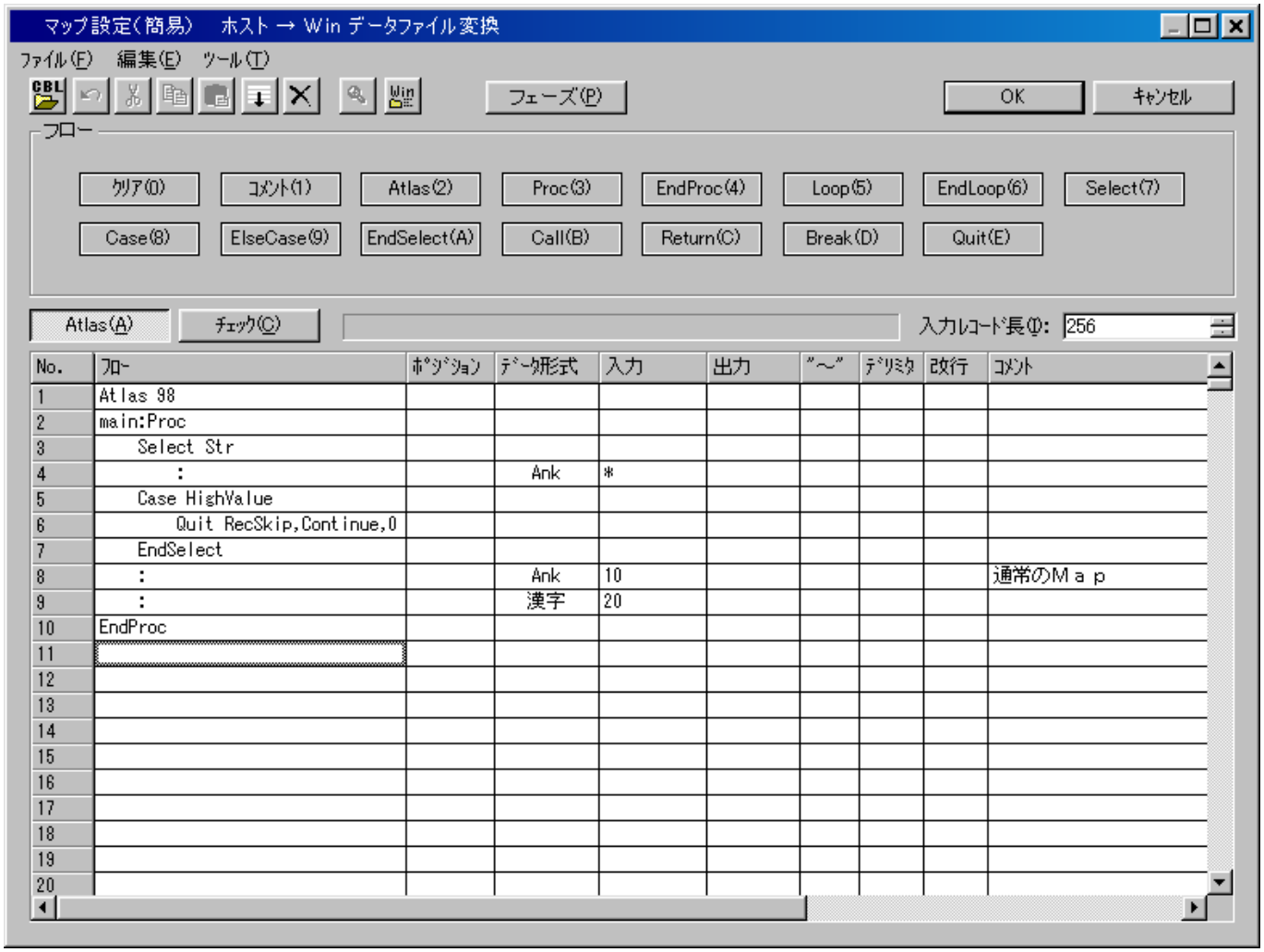


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｌｏｏｐ　２５６}
      {Ｓｅｌｅｃｔ　Ｉｎｔ}
         ｐａｃｋｄｉｓｐ　　ｕ３
      {Ｃａｓｅ　９９９}
         {Ｑｕｉｔ　ＲｅｃＤｏｎｅ，ＥＯＦ，０}
      {ＥｌｓｅＣａｓｅ}
         ｐａｃｋｄｉｓｐ　　ｕ３
      {ＥｎｄＳｅｌｅｃｔ}
   {ＥｎｄＬｏｏｐ}
{ＥｎｄＰｒｏｃ}
```

■ＦＦ…（ハイバリュー）の詰まったレコードをスキップする

オールＦＦ(ハイバリュー)のレコードは削除レコードの意味としているシステムがあります。これをスキップします。



＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｓｔｒ}
      ａｎｋ          ＊
   {Ｃａｓｅ　ＨｉｇｈＶａｌｕｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
      ａｎｋ          １０       --通常のＭａｐ
      ｋａｎｊｉ       ２０
{ＥｎｄＰｒｏｃ}
```

■負の売上を返品とみなす

返品レコードなどの場合、項目分けを変えたいことがあります。Ｍｉｎ、Ｍａｘを利用すると、正負の指定ができます。

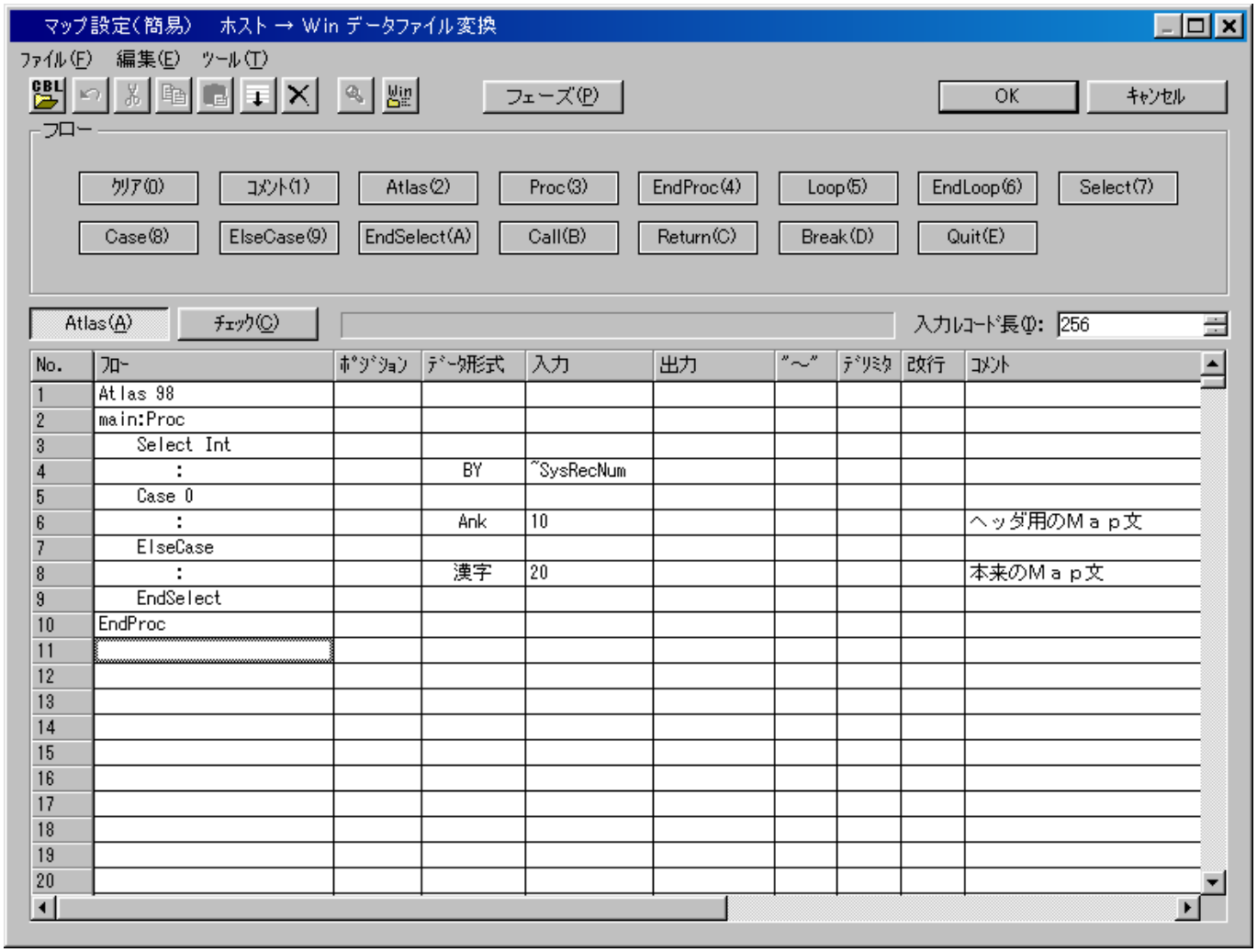
No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	Select Int								
4	:		ゾーン・表示	s10					
5	Case 0..Max								
6	:		ゾーン・表示	s10					売上用Ｍａｐ
7	:		Ank	10					
8	Case Min..-1								
9	:		ゾーン・表示	s10					返品用Ｍａｐ
10	:		漢字	20					
11	EndSelect								
12	EndProc								
13									
14									
15									
16									
17									
18									

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
    {Ｓｅｌｅｃｔ　Ｉｎｔ}
        ｚｏｎｅｄｉｓｐ　　ｓ１０
    {Ｃａｓｅ　０.. Ｍａｘ}
        ｚｏｎｅｄｉｓｐ　　ｓ１０　　--売上用Ｍａｐ
        ａｎｋ　　１０
    {Ｃａｓｅ　Ｍｉｎ.. －１}
        ｚｏｎｅｄｉｓｐ　　ｓ１０　　--返品用Ｍａｐ
        ｋａｎｊｉ　　２０
    {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■先頭レコードだけ別レイアウト

ホストからＷｉｎへ、先頭レコードだけヘッダレコードとして変換し、それ以降は本来のレコードとして変換します。レコードタイプを示す項目はありません。

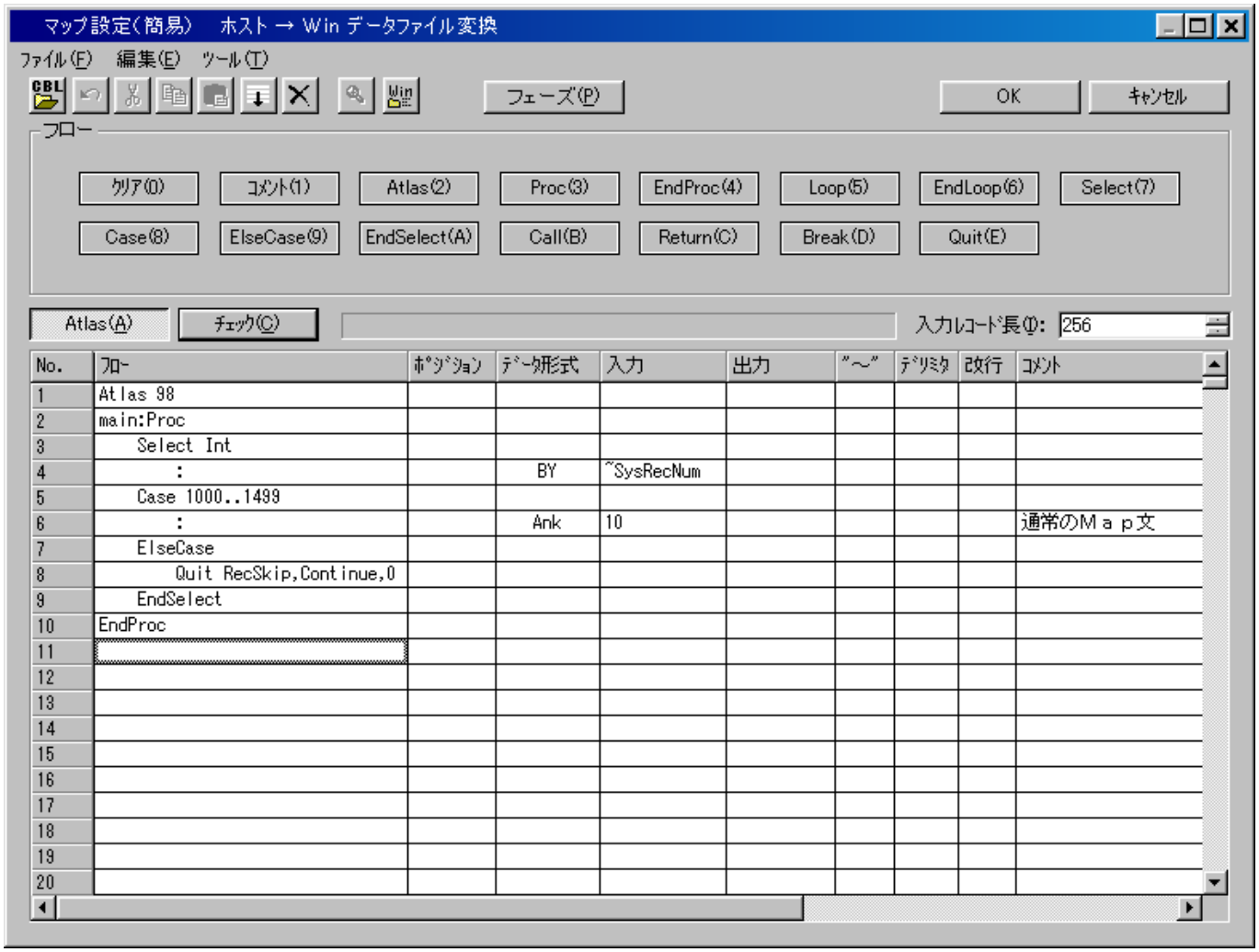


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
    {Ｓｅｌｅｃｔ　Ｉｎｔ}
      ＢＹ              ~ＳｙｓＲｅｃＮｕｍ
    {Ｃａｓｅ　０}
      ａｎｋ            １０        --ヘッダ用のＭａｐ文
    {ＥｌｓｅＣａｓｅ}
      ｋａｎｊｉ        ２０        --本来のＭａｐ文
    {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■範囲指定して変換

ホスト→Ｗｉｎファイル変換で、０から数えて１０００番目のレコードから５００レコードだけを変換します。



＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     ＢＹ　　　~ＳｙｓＲｅｃＮｕｍ
   {Ｃａｓｅ　１０００.. １４９９}
     ａｎｋ　　　１０　　--通常のＭａｐ文
   {ＥｌｓｅＣａｓｅ}
    {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■４レコード毎に同じパターン

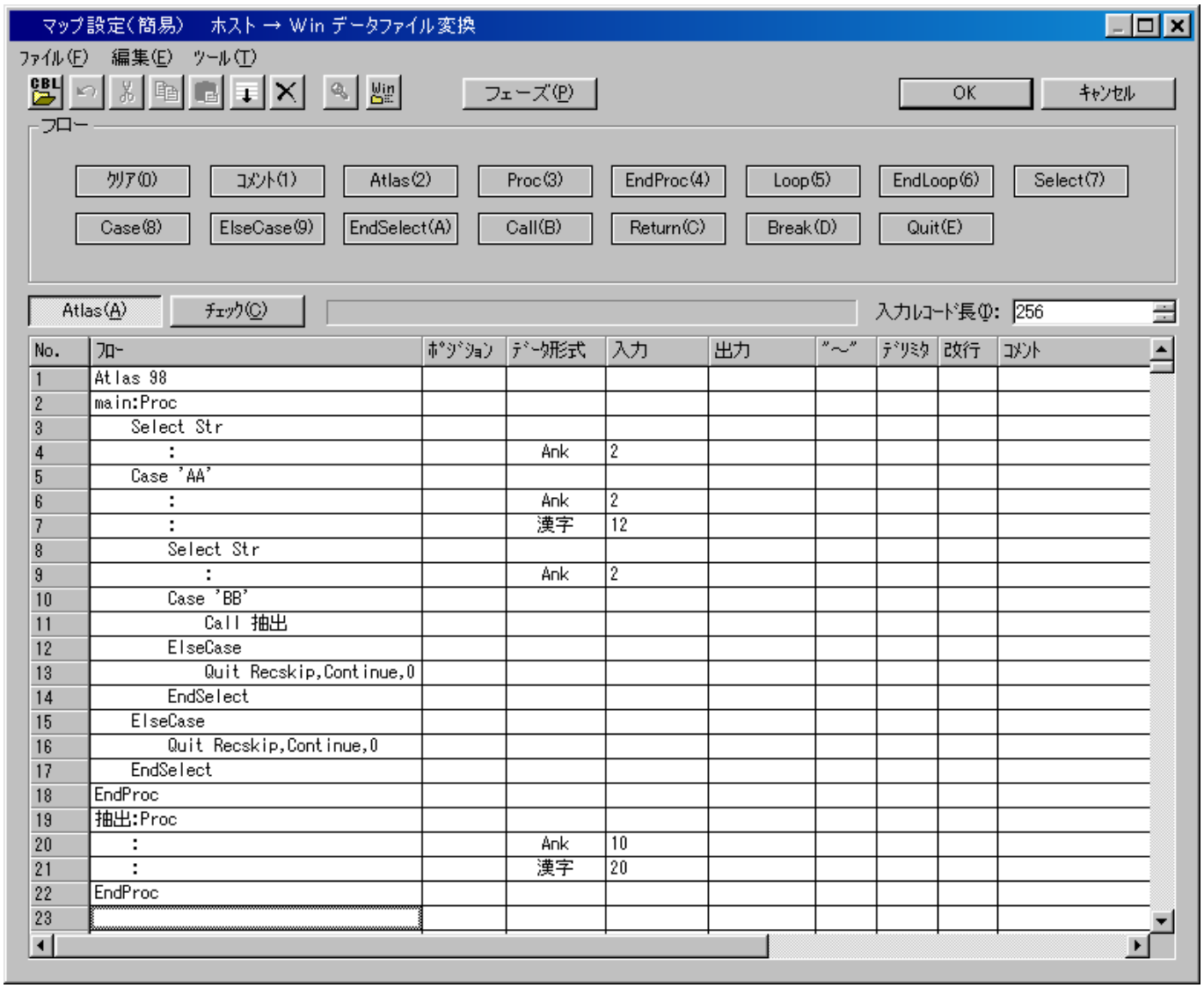
４レコード毎に同じパターンのレコードをくり返すファイルを変換します。

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}
{ｍａｉｎ：Ｐｒｏｃ}
    {Ｓｅｌｅｃｔ　Ｉｎｔ}
      ＢＹ              ~ＳｙｓＲｅｃＮｕｍ￥￥４
    {Ｃａｓｅ　０}
      ａｎｋ            １０          --０番目のレコードのマップ
    {Ｃａｓｅ　１}
      ｋａｎｊｉ        ２０          --１番目のレコードのマップ
    {Ｃａｓｅ　２}
      ｚｏｎｅｄｉｓｐ  ｕ５          --２番目のレコードのマップ
    {Ｃａｓｅ　３}
      ｐａｃｋｄｉｓｐ  ｓ３．２      --３番目のレコードのマップ
    {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■複合条件で抽出

先頭の項目が’ＡＡ’で次々項目が’ＢＢ’のレコードだけ抽出変換します。



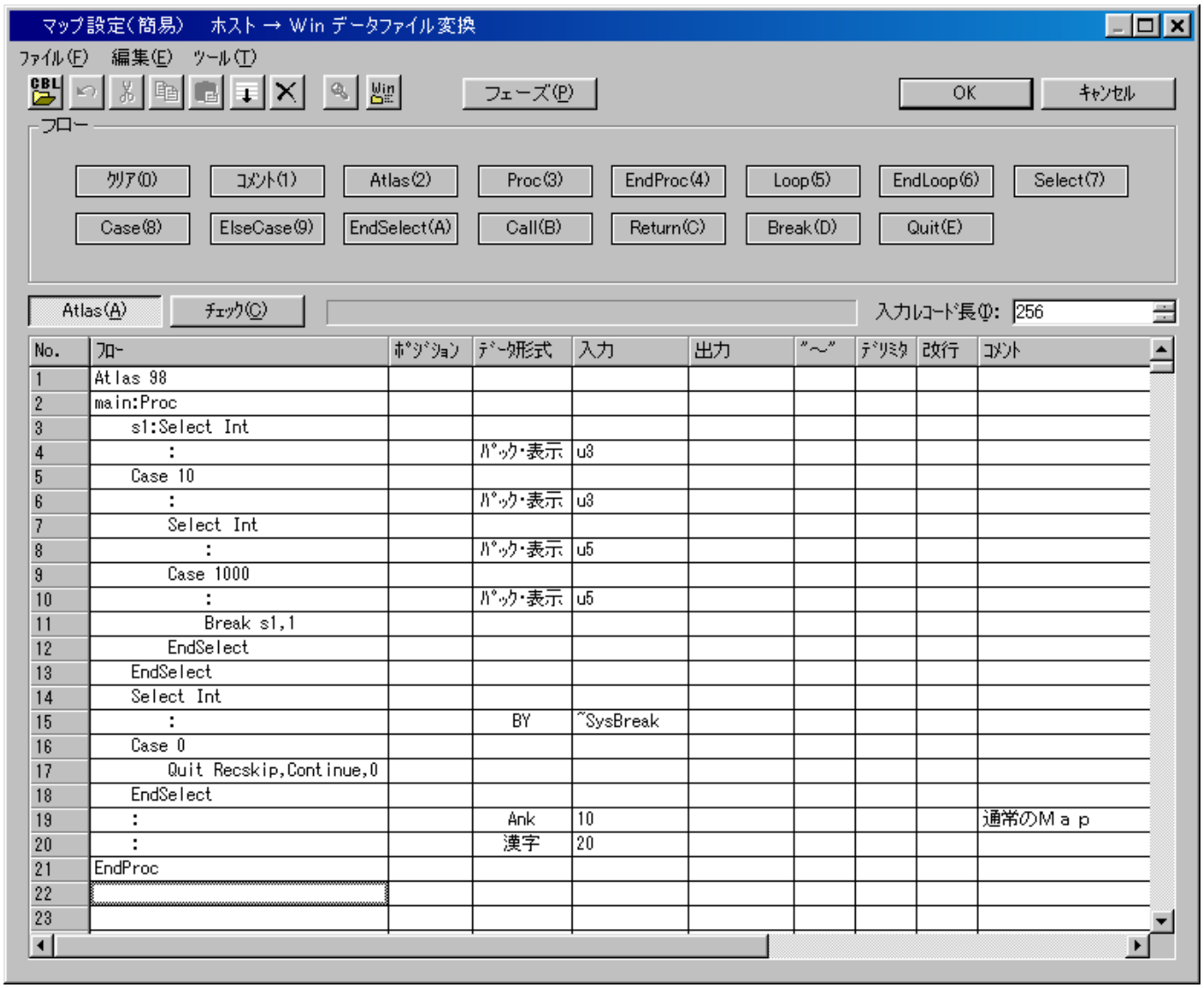
No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	Select Str								
4	:		Ank	2					
5	Case 'AA'								
6	:		Ank	2					
7	:		漢字	12					
8	Select Str								
9	:		Ank	2					
10	Case 'BB'								
11	Call 抽出								
12	ElseCase								
13	Quit Recskip,Continue,0								
14	EndSelect								
15	ElseCase								
16	Quit Recskip,Continue,0								
17	EndSelect								
18	EndProc								
19	抽出:Proc								
20	:		Ank	10					
21	:		漢字	20					
22	EndProc								
23									

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｓｔｒ}
     ａｎｋ          ２
   {Ｃａｓｅ　’ＡＡ’}
     ａｎｋ          ２
     ｋａｎｊｉ        １２
      {Ｓｅｌｅｃｔ　Ｓｔｒ}
        ａｎｋ          ２
      {Ｃａｓｅ　’ＢＢ’}
         {Ｃａｌｌ　抽出}
      {ＥｌｓｅＣａｓｅ}
         {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
      {ＥｎｄＳｅｌｅｃｔ}
   {ＥｌｓｅＣａｓｅ}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}

{抽出：Ｐｒｏｃ}
  ａｎｋ          １０
  ｋａｎｊｉ        ２０
{ＥｎｄＰｒｏｃ}
```

■複合条件と多重脱出

先頭の項目が１０で次の項目が１０００のレコードだけを抽出変換します。



No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	s1:Select Int								
4	:		パック・表示	u3					
5	Case 10								
6	:		パック・表示	u3					
7	Select Int								
8	:		パック・表示	u5					
9	Case 1000								
10	:		パック・表示	u5					
11	Break s1,1								
12	EndSelect								
13	EndSelect								
14	Select Int								
15	:		BY	^SysBreak					
16	Case 0								
17	Quit Recskip,Continue,0								
18	EndSelect								
19	:		Ank	10					通常のＭａｐ
20	:		漢字	20					
21	EndProc								
22									
23									

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {ｓ１：Ｓｅｌｅｃｔ　Ｉｎｔ}
     ｐａｃｋｄｉｓｐ　ｕ３
   {Ｃａｓｅ　１０}
     ｐａｃｋｄｉｓｐ　ｕ３
      {Ｓｅｌｅｃｔ　Ｉｎｔ}
        ｐａｃｋｄｉｓｐ　ｕ５
      {Ｃａｓｅ　１０００}
        ｐａｃｋｄｉｓｐ　ｕ５
         {Ｂｒｅａｋ　ｓ１，１}
      {ＥｎｄＳｅｌｅｃｔ}
   {ＥｎｄＳｅｌｅｃｔ}

   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     ＢＹ　　　　　~ＳｙｓＢｒｅａｋ
   {Ｃａｓｅ　０}
      {Ｑｕｉｔ　ＲｅｃＳｋｉｐ，Ｃｏｎｔｉｎｕｅ，０}
   {ＥｎｄＳｅｌｅｃｔ}
  ａｎｋ　　　１０　　--通常のＭａｐ
  ｋａｎｊｉ　　２０
{ＥｎｄＰｒｏｃ}
```

■ファイル区分別変換

先頭レコードのファイル区分が’ＥＥ’のときと’ＪＪ’のときで、本体レコードを変換し分けます。

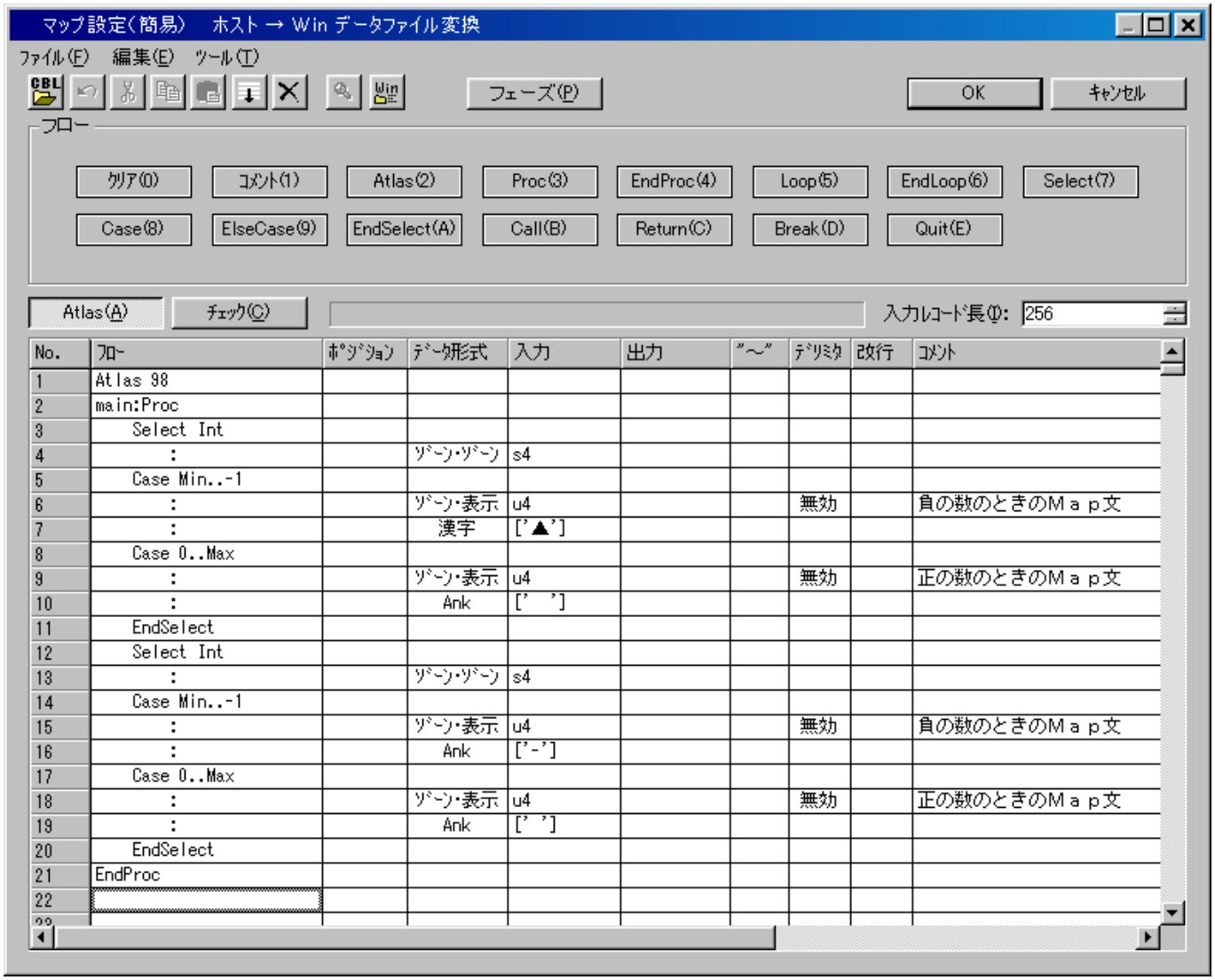
No.	フロー	ポジ	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	Select Int								レコード番号を調べる
4	:		BY	^SysRecNum					
5	Case 0								ヘッダレコードのとき
6	Select Str								ファイル区分を調べる
7	:		Ank	2					
8	Case 'EE'								EE型のとき
9	:		Ank	2		両くくり			
10	Quit RecDone,Continue,1								Quit値＝１でヘッダ変換終了
11	Case 'JJ'								JJ型のとき
12	:		Ank	2		両くくり			
13	Quit RecDone,Continue,2								Quit値＝２でヘッダ変換終了
14	EndSelect								
15	ElseCase								ボディーレコードのとき
16	Select Int								前レコードのQuit値を調べる
17	:		BY	^SysQuit					
18	Case 1								Quit値＝１（EE型）のとき
19	:		Ank	32		両くくり			
20	:		Ank	4		両くくり			
21	:		Ank	*		両くくり			
22	Quit RecDone,Continue,1								Quit値＝１を返し、継続
23	Case 2								Quit値＝２（JJ型）のとき
24	:		漢字	32		両くくり			
25	:		Ank	32		両くくり			
26	:		Ank･漢字	*		両くくり			
27	Quit RecDone,Continue,2								Quit値＝２を返し、継続
28	EndSelect								
29	EndSelect								
30	EndProc								
31									

＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}                    --レコード番号を調べる
     ＢＹ              ~ＳｙｓＲｅｃＮｕｍ
   {Ｃａｓｅ　０}                           --ヘッダレコードのとき
      {Ｓｅｌｅｃｔ　Ｓｔｒ}                 --ファイル区分を調べる
        ａｎｋ        ２
      {Ｃａｓｅ　’ＥＥ’}                    --EE 型のとき
        “ａｎｋ       ２”
        {Ｑｕｉｔ　ＲｅｃＤｏｎｅ，Ｃｏｎｔｉｎｕｅ，１}
                                            --Quit 値＝１でヘッダ変換終了
      {Ｃａｓｅ　’ＪＪ’}                    --JJ 型のとき
        “ａｎｋ       ２”
        {Ｑｕｉｔ　ＲｅｃＤｏｎｅ，Ｃｏｎｔｉｎｕｅ，２}
                                            --Quit 値＝２でヘッダ変換終了
      {ＥｎｄＳｅｌｅｃｔ}

   {ＥｌｓｅＣａｓｅ}                       --ボディーレコードのとき
      {Ｓｅｌｅｃｔ　Ｉｎｔ}                 --前レコードの Quit 値を調べる
        ＢＹ              ~ＳｙｓＱｕｉｔ
      {Ｃａｓｅ　１}                        --Quit 値＝１（EE 型）のとき
        “ａｎｋ       ３２”
        “ａｎｋ       ４”
        “ａｎｋ       ＊”
        {Ｑｕｉｔ　ＲｅｃＤｏｎｅ，Ｃｏｎｔｉｎｕｅ，１}
                                            --Quit 値＝１を返し、継続
      {Ｃａｓｅ　２}                        --Quit 値＝２（JJ 型）のとき
        “ｋａｎｊｉ　３２”
        “ａｎｋ       ３２”
        “ｋａｎｊｉｍｉｘ　＊”
        {Ｑｕｉｔ　ＲｅｃＤｏｎｅ，Ｃｏｎｔｉｎｕｅ，２}
                                            --Quit 値＝２を返し、継続
      {ＥｎｄＳｅｌｅｃｔ}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■負の数のとき“▲”や“－”を後付けする

ホストからＷｉｎへ、値が負の数の項目に“▲”や“－”を後付けして変換します。

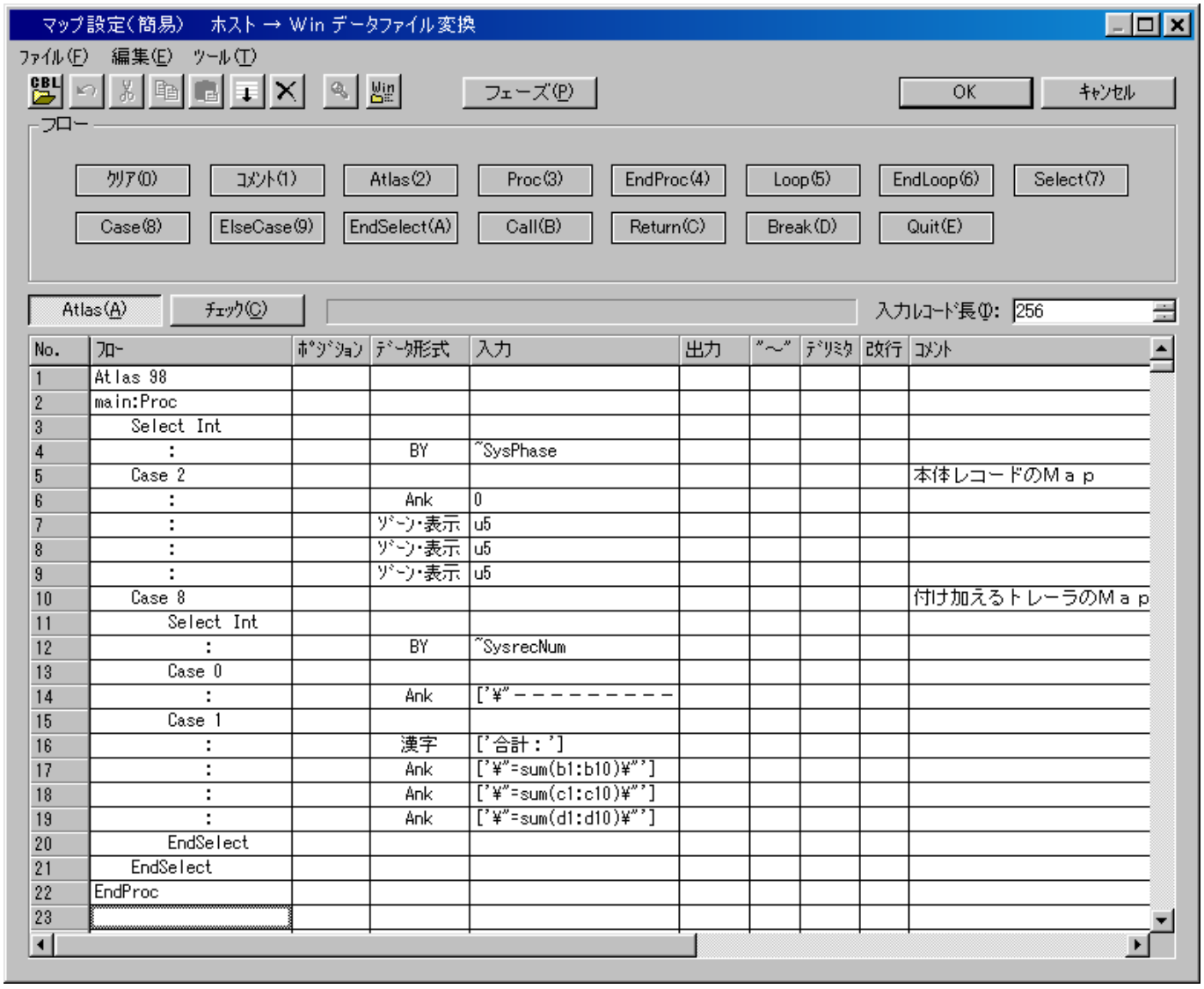


＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     ｚｏｎｅｚｏｎｅ　　ｓ４
   {Ｃａｓｅ　Ｍｉｎ.. －１}
     ｚｏｎｅｄｉｓｐ　　ｕ４　　＆　　--負の数のときのＭａｐ文
     ｋａｎｊｉ　　['▲']　,
   {Ｃａｓｅ　０.. Ｍａｘ}
     ｚｏｎｅｄｉｓｐ　　ｕ４　　＆　　--正の数のときのＭａｐ文
     ａｎｋ　　[' ']　,
   {ＥｎｄＳｅｌｅｃｔ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     ｚｏｎｅｚｏｎｅ　　ｓ４
   {Ｃａｓｅ　Ｍｉｎ.. －１}
     ｚｏｎｅｄｉｓｐ　　ｕ４　　＆　　--負の数のときのＭａｐ文
     ａｎｋ　　['－']　,
   {Ｃａｓｅ　０.. Ｍａｘ}
     ｚｏｎｅｄｉｓｐ　　ｕ４　　＆　　--正の数のときのＭａｐ文
     ａｎｋ　　[' ']　,
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

■Ｅｘｃｅｌの式を埋め込む

ホストからＷｉｎへＣＳＶ形式に変換するときに、Ｅｘｃｅｌの計算式を埋め込みます。トレーラレコードを生成・付加して、そこに合計を算出する式を挿入します。

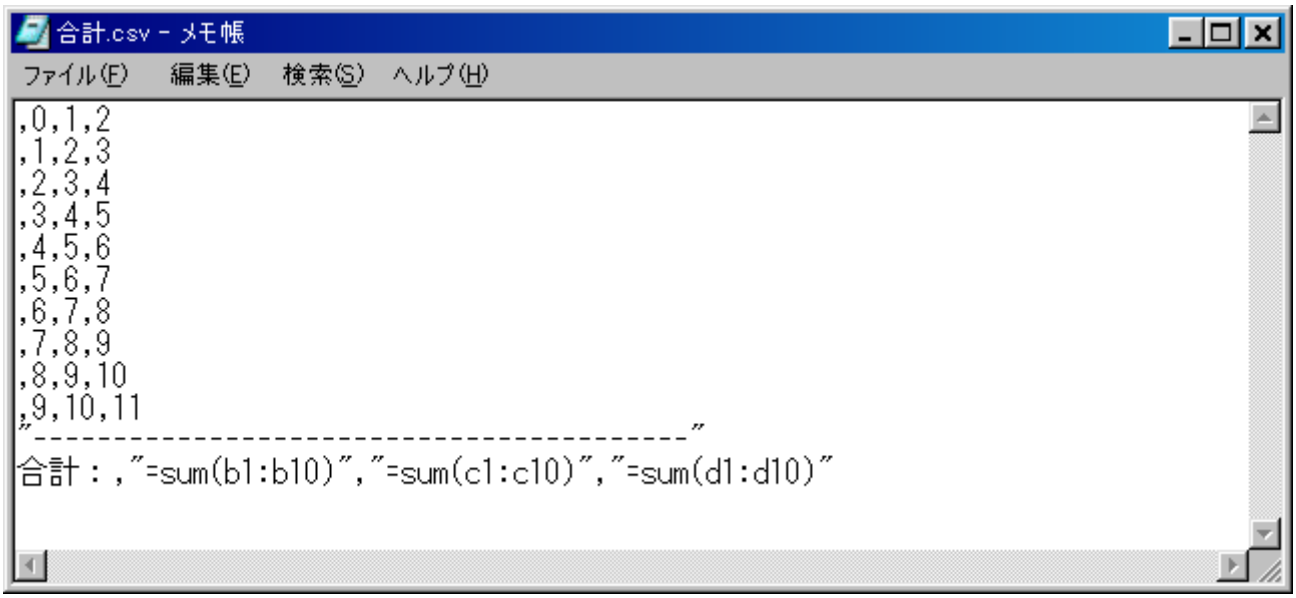


No.	フロー	ポジション	データ形式	入力	出力	"～"	デリミタ	改行	コメント
1	Atlas 98								
2	main:Proc								
3	Select Int								
4	:		BY	~SysPhase					
5	Case 2								本体レコードのＭａｐ
6	:		Ank	0					
7	:		ゾーン・表示	u5					
8	:		ゾーン・表示	u5					
9	:		ゾーン・表示	u5					
10	Case 8								付け加えるトレーラのＭａｐ
11	Select Int								
12	:		BY	~SysrecNum					
13	Case 0								
14	:		Ank	['¥"－－－－－－－－－－					
15	Case 1								
16	:		漢字	['合計：']					
17	:		Ank	['¥"=sum(b1:b10)¥""]					
18	:		Ank	['¥"=sum(c1:c10)¥""]					
19	:		Ank	['¥"=sum(d1:d10)¥""]					
20	EndSelect								
21	EndSelect								
22	EndProc								
23									

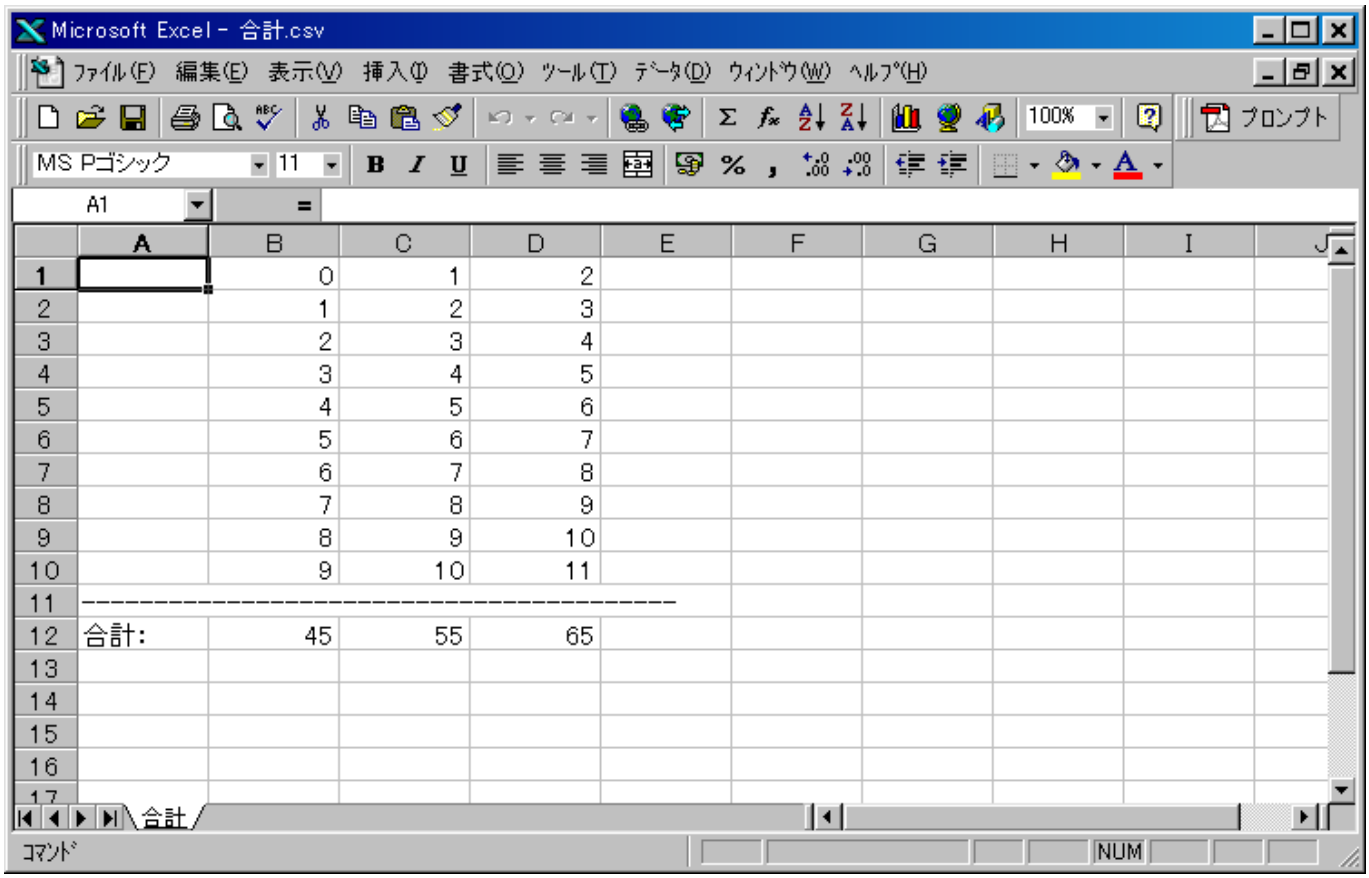
＜パラメータファイルの記述＞

```
{Ａｔｌａｓ　９８}

{ｍａｉｎ：Ｐｒｏｃ}
   {Ｓｅｌｅｃｔ　Ｉｎｔ}
     ＢＹ              ~ＳｙｓＰｈａｓｅ　,
   {Ｃａｓｅ　２}                        --本体レコードのＭａｐ
     ａｎｋ              ０　,
     ｚｏｎｅｄｉｓｐ     ｕ５　,
     ｚｏｎｅｄｉｓｐ     ｕ５　,
     ｚｏｎｅｄｉｓｐ     ｕ５　,
   {Ｃａｓｅ　８}                        --付け加えるトレーラのＭａｐ
      {Ｓｅｌｅｃｔ　Ｉｎｔ}
        ＢＹ              ~ＳｙｓＲｅｃＮｕｍ　,
      {Ｃａｓｅ　０}
        ａｎｋ             [‘¥“－－－－－－－－－－－－－－－¥”’],
      {Ｃａｓｅ　１}
        ｋａｎｊｉ         [‘合計：’],
        ａｎｋ             [‘¥“＝ｓｕｍ（ｂ１：ｂ１０）¥”’],
        ａｎｋ             [‘¥“＝ｓｕｍ（ｃ１：ｃ１０）¥”’],
        ａｎｋ             [‘¥“＝ｓｕｍ（ｄ１：ｄ１０）¥”’],
      {ＥｎｄＳｅｌｅｃｔ}
   {ＥｎｄＳｅｌｅｃｔ}
{ＥｎｄＰｒｏｃ}
```

変換後のファイルをテキストエディタで開くと、このようになります。下の２行が、付け加えたレコードです。



同じファイルをＥｘｃｅｌで開くと、計算された値が表示されます。



Ｆ＊ＴＲＡＮ＋　Ｖ８．０　　操作説明書／マルチレコード編

２０１７年　１月　第１版発行

編集・著作　　株式会社　富士通ビー・エス・シー
所　在　地　　〒135-8300　東京都港区台場 2-3-1 トレードピアお台場

・Windows、MS-DOS、Visual Basic、Access、Visual C++は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。
・Unicode は Unicode コンソーシアムの商標です。
・F*TRAN は富士通ビー・エス・シーの登録商標です。
・会社名および製品名はそれぞれ各社の商標または登録商標です。
・本書およびシステムは、改善のため事前連絡なしに変更することがあります。
・無断複製、および転載を禁じます。