pTeX 系列の文字列化における和文欧文の区別

北川 弘典

2019–10–12

~~TeXConf 2019~~ 中止

本発表の話題は次の現象に関連した諸々

入力

```
\documentclass{minimal}%                    UTF-8 入力
\begin{document} ſ \end{document}%    ſ: U+017F
```

エラーメッセージ

```
! Package inputenc Error: Unicode character 顛 (U+C4CF)
(inputenc)                not set up for use with LaTeX.
```

“Inconsistent error message”, JulienPalard, 2019-05-27, platex/#84

前提など

- **入出力は UTF-8 と想定**
- 主に**内部コード EUC の pTEX** を扱う

記法

hoge hoge という名前の制御綴

8 ビット値 0xXY
もしくはその文字コードの欧文文字トークン

16 ビット値 0xABCD
もしくはその文字コードの和文文字トークン

■導入：TEX とトークン

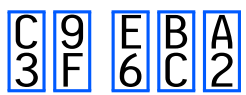
■和文・欧文の区別が消えるとき

■現行の実装と改善案

pTEX のトークン

(p)TEX は入力を**トークン**単位で処理する.

入力例 \hbox{ß 漢\hskip3pt}



前処理 ptexenc により，内部コードへ変換

\hbox{^^c3^^9f漢\hskip3pt}

トークン列

hbox { C3 9F 漢 hskip 3 p t }

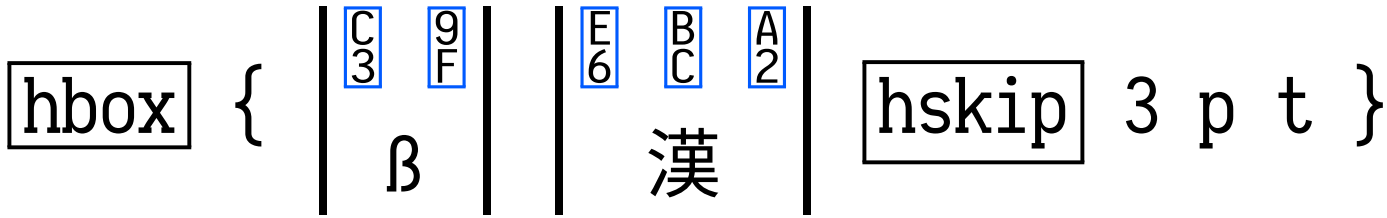
：和文文字トークン　無：欧文文字トークン

トークンの補足

^^ 記法　間接的な入力法，**常に欧文扱い**[1]

- ^^**pq**　（**p**, **q**: 0–9, a–f）
- ^^**X**　（**X**: ASCII で 0–9, a–f 以外）

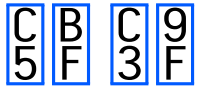
upTEX　\kcatcode で欧文扱いと和文扱い切り替え



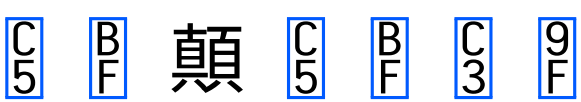
[1] pTEX p3.1.4 以降．p3.1.8 で制御綴名内の ^^ 記法の扱い改善．

想定通りの例

```
\font\TT=ec-lmtt10 \TT %     256文字あるフォント
\def\fuga{^^c5^^bf顛ſ ß }
```



- \fuga の展開結果のトークン列

- \fuga の結果 (DVI)

  Å£顛Å£Ã§ (*)

- ソース中で直に ^^c5^^bf顛ſß としても結果は (*)

導入：TEX とトークン

和文・欧文の区別が消えるとき

- plain pTEX の例
- pLATEX での例
- upTEX での状況

現行の実装と改善案

\meaning 他の挙動

```
\font\TT=ec-lmtt10 \TT
\def\fuga{^^c5^^bf顛ſß}
```

	C5BF
UTF-8	ſ
EUC-JP	顛

- \writeN{\fuga} 顛顛顛Å¿^^9f
- \meaning\fuga macro:->顛顛顛Ã§
- \detokenize\expandafter{\fuga} 顛顛顛Ã§
- \message{\fuga} 顛顛顛Å¿^^9f

後ろの３命令は「文字列化」を行う命令

制御綴名と \string

```
\catcode"C5=11 \catcode"BF=11
\def\顛{P} \def\ſ{Q}
```

X	\X	\csname X\endcsname	\string\X
顛	Q	Q	\顛
ſ	Q	Q	\顛
^^c5^^bf	Q	Q	\顛

“Inconsistent error message” 再掲

JulienPalard, 2019-05-27, platex/#84

入力

```
\documentclass{minimal}%                    UTF-8 入力
\begin{document} ſ \end{document}
```

エラーメッセージ

```
! Package inputenc Error: Unicode character 顛 (U+C4CF)
(inputenc)                not set up for use with LaTeX.
```

u8:ſ (= u8:C5BF) の \string の結果から作られる

実際には \ u 8 : 顛

“Non-ASCII symbols in labels”

aminophen, 2019−07−13, platex/#86

入力

元々は LaTeX 2ε 2019−10−01 関係

```
\documentclass{article}
\UseRawInputEncoding%                    "raw" encoding
\begin{document}
\section{a}\label{abß}\ref{abß}
\end{document}
```

警告（消えない）

```
LaTeX Warning: Reference `abC3^^9f' on page 1 undefined
on input line 4.
```

“Non-ASCII symbols in labels” 解析

aminophen, 2019-07-13, platex/#86

aux ファイル 3 行目

```
\newlabel{ab[C3]^^9f}{{1}{1}}
```

→読み込み時に ptexenc は次のように変換：

```
\newlabel{ab[A2AF]^^9f}{{1}{1}}
```

→ LaTeX はラベル「ab[A2AF][9F]」が定義されたと認識

亜種：違うラベル名だが……？

入力

```
\documentclass{article}
\UseRawInputEncoding %                    % `raw' encoding
\begin{document}
\section{}\label{ſ}\label{顛}
\end{document}
```

警告メッセージ（消えない）

```
LaTeX Warning: Label `顛' multiply defined.
```

理由

aux ファイル 3, 4 行目が同一（`\newlabel{顛}{{1}{1}}`）

upTEX での状況

- （実質的には）ptexenc によるコード変換なし
  → platex/#84, #86 は upLATEX では影響なし
- **文字列化後の再トークン化は \kcatcode 依存**

```
\font\TT=ec-lmtt10 \TT
\def\fuga{^^e3^^81^^82あß}
\kcatcode"3042=16
\fuga%                                          ãĄĆあÃ§
\meaning\fuga%                                macro:->あ あÃ§
\expandafter\string\csname\fuga\endcsname
%                                               \あ あÃ§
```

upTEX での状況

- （実質的には）ptexenc によるコード変換なし
  → platex/#84, #86 は upLATEX では影響なし
- **文字列化後の再トークン化は \kcatcode 依存**

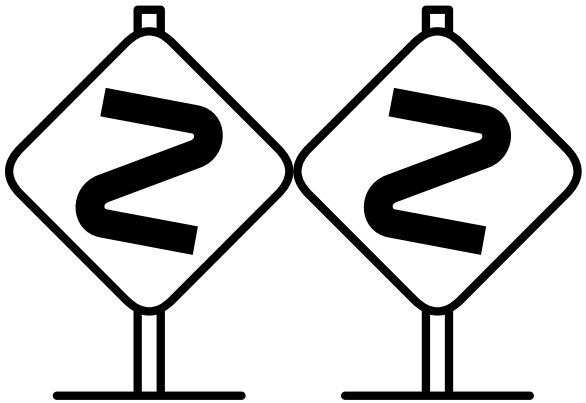
```
\font\TT=ec-lmtt10 \TT
\def\fuga{^^e3^^81^^82あß}
\kcatcode"3042=15
\fuga%                                          ãĄĆあÃ§
\meaning\fuga%                          macro:->ãĄĆãĄĆÃ§
\expandafter\string\csname\fuga\endcsname
%                                               \ãĄĆãĄĆÃ§
```

■導入：TEX とトークン

■和文・欧文の区別が消えるとき

■現行の実装と改善案

- 文字列化
- 端末・ファイルへの出力
- 端末・ファイルからの入力



改善案 tex-jp-build 個人フォーク[2]で実験中

[2] https://github.com/h-kitagawa/tex-jp-build/tree/printkanji_16bit

文字列化

（途中の）結果を一旦配列 *str_pool* に格納する処理

- *str_pool* にはバイト列として格納される
  →和文由来か欧文由来かの情報は消える
- プリミティブ達
  - \the, \detokenize, \number, \string, \meaning, \euc 他
    →結果を文字列化したものを再トークン化
  - \message, \errmessage, \special, \write18
    →文字列化したものを出力（実行）
  - \scantokens
    →文字列化したものを疑似ファイルと扱う

「\the, \detokenize, ... , \euc 他」の一覧

\the, \detokenize 以外は command code が *convert*

TeX82	\the, \number, \romannumeral, \string, \meaning, \fontname, \jobname
ε-TeX	\detokenize, \eTeXrevision
(u)pTeX	\kansuji, \euc, \sjis, \jis, \kuten, \ucs, \ptexrevision, \uptexrevision
pdfTeX	\pdfstrcmp, \pdfcreationdate, \pdffilemoddate, \pdffilesize, \pdfmdfivesum, \pdffiledump, \pdfuniformdeviate, \pdfnormaldeviate

文字列化

（途中の）結果を一旦配列 *str_pool* に格納する処理

現行　*str_pool* の各要素は 0 .. 255 を格納

改善案　*str_pool* の各要素は 0 .. 511 を格納[3]

0–255　欧文由来バイト

256–511　和文由来バイト


改造量を考え，「1 要素 1 文字」にはしなかった


\pdf... などで XeTeX のコード流用可能
（XeTeX では *str_pool* の各要素は 0 .. 65535 を格納）

[3] WEB ソース中では 0 .. 32768 (unsigned short[] にするため).

余談：fmt ファイルの容量

```
pLaTeX2e <2019-10-01> (based on LaTeX2e <2019-10-01>)
```

[KiB]	英語のみ			全部		
（切り上げ）	現行	**改善案**		現行	**改善案**	
pLaTeX	898	**994**	+96	4449	**4584**	+136
upLaTeX	894	**990**	+96	4445	**4580**	+136

- 多数のハイフネーションパターンの下では影響薄
- 容量が気になれば XeTeX, LuaTeX のように gzip 圧縮？
  （TeX Live の pTeX 系列は SyncTeX のために zlib リンク済）

例：\meaning (pTEX)


```
\def\fuga{^^c5^^bf顛ß\cr}\meaning\fuga
```

1. \fuga の中身を表すトークン列

   C5 BF 顛 C3 9F cr

2. 文字列化の結果（*str_pool* の中身）

   m a c r o : - > C5 BF C5 BF C3 9F \ c r ␣

3. 再トークン化

   m a c r o : - > 顛 顛 C3 9F \ c r ␣

和文文字と解釈できるバイト列は和文文字トークンに

例：\meaning (pTEX)


```
\def\fuga{^^c5^^bf顛ß\cr}\meaning\fuga
```

1. \fuga の中身を表すトークン列

   C5 BF 顛 C3 9F cr

2. 文字列化の結果（*str_pool* の中身）

   m a c r o : - > C5 BF 01 C5 01 BF C3 9F \ c r ␣

3. 再トークン化

   m a c r o : - > C5 BF 顛 C3 9F \ c r ␣

フラグあり (01 00–01 FF) の列からのみ和文文字トークンに

例：\meaning (upTEX)


```
\def\fuga{^^e3^^81^^82あß\cr}\meaning\fuga
```

1. \fuga の中身を表すトークン列

   E3 81 82 顛 C3 9F cr

2. 文字列化の結果（*str_pool* の中身）

   m a c r o : - > E3 81 82 E3 81 82 C3 9F \ c r ␣

3. 再トークン化

   m a c r o : - > あ あ C3 9F \ c r ␣

   和文文字トークンにするかは \kcatcode 依存

例：\meaning (upTEX)

改善案 2/2

```
\def\fuga{^^e3^^81^^82あß\cr}\meaning\fuga
```

1. \fuga の中身を表すトークン列

   E3 81 82 顛 C3 9F cr

2. 文字列化の結果（*str_pool* の中身）

   m a c r o : - > E3 81 82 01 E3 01 81 01 82 C3 9F \ c r ␣

3. 再トークン化

   m a c r o : - > E3 81 82 あ C3 9F \ c r ␣

フラグあり (0100–01FF) の列からのみ和文文字トークンに

制御綴名と \string

制御綴名も配列 *str_pool* に格納される

→ここでも和文由来・欧文由来の区別はない

例

X	\X	\csname X\endcsname	\string\X
顛	内部では 		\顛
ſ			\顛
^^c5^^bf			\顛

制御綴名と \string

制御綴名も配列 *str_pool* に格納される

→ここでも和文由来・欧文由来の区別を**つける**

例

X	\X	\csname X\endcsname	\string\X
顛	内部では 01 C5 01 BF		\顛
ſ	内部では C5 BF		\ C5 BF
^^c5^^bf			\ C5 BF

制御綴名と \string (upTEX)

\kcatcode 依存なのは**制御綴名の再トークン化時**か？
制御綴名の内部表現か？

```
                          % 現行                  改善案
 \font\TT=ec-lmtt10 \TT
 \def\あ{hoge}\def\B{\あ}
 \catcode"E3=11 \catcode"81=11 \catcode"82=11
 {\kcatcode"3042=15
  \あ                     % hoge                  ! Undefined
  \meaning\B}             % macro:->\ãĄĆ  macro:->\あ
\bye
```

端末・ファイルへの出力

pTEX 系列における **1 バイト**の出力は複数段階

(ptexenc)

print ⟶ *print_char* ⟶ *putc2* ⟶ 実際の出力

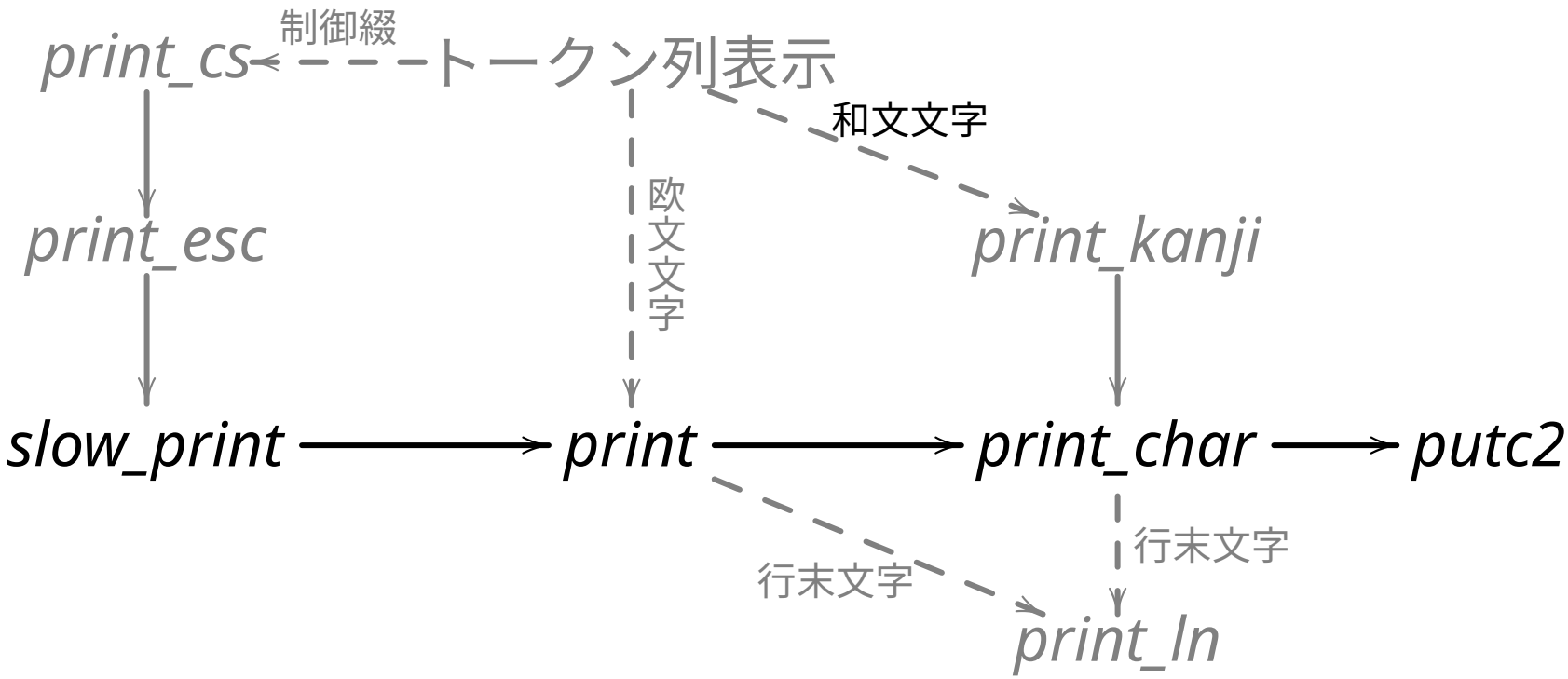
9F	9F	^ ^ 9 f	^ ^ 9 f	下記以外
a	a	a	a	ASCII
C5	C5	C5	（次を待つ）	和文先頭
C5 A [4]	C5 A	C5 A	C5 A	
(C5 BF)[5]	C5 BF	C5 BF	顛	

putc2 はバイトが和文由来か欧文由来か知らない

[4] 実際は *print*(0xC5); *print*(0x41) という 2 回の呼び出し．

[5] 和文文字トークンの出力時には *print* は経由しない．

print 系プロシージャ



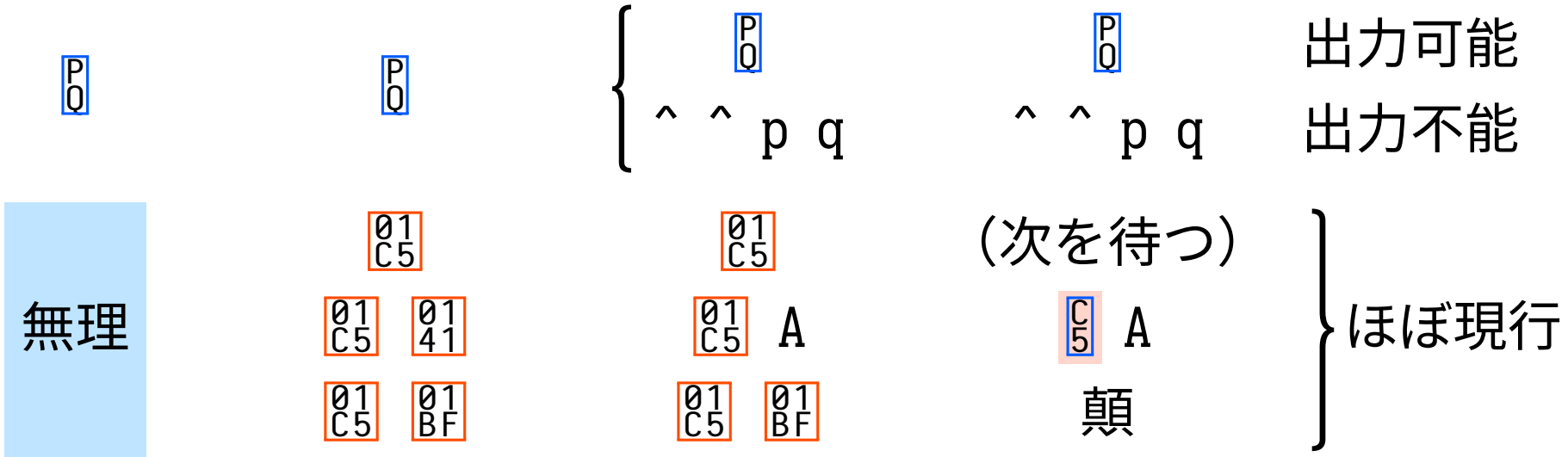
	文字列	1 バイト	
汎用	*slow_print*	*print*	（必要なら ^^ 記法）
高速	*print*	*print_char*	

端末・ファイルへの出力

改善案

str_pool で 0100–01FF を和文由来にしたことと合わせる

print ⟶ *print_char* ⟶ *putc2* (ptexenc) ⟶ 実際の出力



1 文字 *c* を出力する目的の *print*(*c*) は変更が必要

(非常に素直だが)

1 文字を出力する *print*

$c \geq 256$ のとき *print*(c) は「c 番の**文字列**」を出力

→**改善案**では場合わけが必要

例：関数 *slow_print*(s)　現行（と同じ動作のコード）

出力不能な文字を含みうる，s 番の文字列を出力

```
while j < str_start[s + 1] do begin
    c ← so(str_pool[j]);
    print(c);
    incr(j);
    end;
```

1 文字を出力する *print*

$c \geq 256$ のとき *print*(c) は「c 番の**文字列**」を出力

→改善案では場合わけが必要

例：関数 *slow_print*(s)　改善案

出力不能な文字を含みうる，s 番の文字列を出力

```
while j < str_start[s + 1] do begin
    c ← so(str_pool[j]);
    if c ≥ 256 then print_char(c) else print(c);
    incr(j);
    end;
```

出力可能なバイト

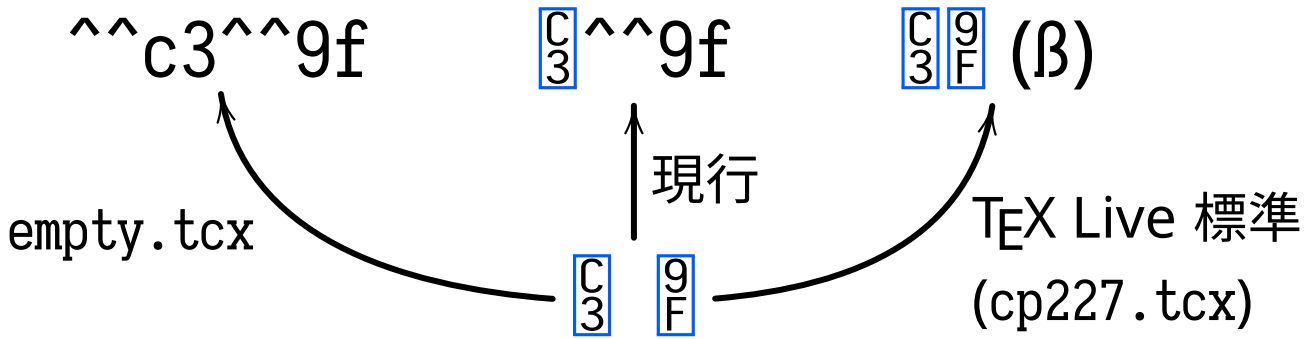
20 (␣)–7E (~) 以外に……

現行

和文の内部コードで現れるバイトは必ず出力可

改善案

内部コードと離れて再考可能．cf. TCX ファイル



TCX ファイル 1

TEX Character Translation

- 8 ビットエンコーディング用の入出力変換機構
- 出力可能性では「何も変換しない」TCX 利用

欧文 TEX での例

```
\message{\string\^^A.\string\^^K.^^9f.^^c1.^^f7}
```

empty.tcx	\^^A.\^^K.^^9f.^^c1.^^f7	
cp8bit.tcx	\^^A.\^^K.9f.c1.f7	80–ff も出力可
cp227.tcx	\^^A.\0b.9f.c1.f7	↑と HT, LF, VT
natural.tcx	\01.\0b.9f.c1.f7	00–ff出力可

TCX ファイル 2

pTEX 系列の TCX サポートは限定的

- そもそも「変換する」TCX の利用価値なし
  →ここでは出力可能性のみ話題に
- 出力可能性がフォーマット作成時 (inipTEX) に固定されてしまう

```
eptex -ini -etex -translate-file=HOGE platex.ini
%                                  →出力可能性について効力あり
platex -translate-file=FUGA ... %                  効力なし
```

例：\write


```
\write1{^^c5^^bf顛abß}
```

1. 出力すべきトークン列

   C5 BF 顛 a b C3 9F

2. ptexenc へ渡るもの

   C5 BF C5 BF a b C3 ^ ^ 9 f

3. 出力

   顛顛abC3^^9f

   \message は文字列化を挟むがほぼ同じ

例：\write


```
\write1{^^c5^^bf顛abß}
```

TCX: empty.tcx

1. 出力すべきトークン列

   C5 BF 顛 a b C3 9F

2. ptexenc へ渡るもの

   C5 BF 01 C5 01 BF a b ^ ^ c 3 ^ ^ 9 f

3. 出力

   ^^c5^^bf顛ab^^c3^^9f

   \message は文字列化を挟むがほぼ同じ

例：\write


```
\write1{^^c5^^bf顛abß}
```

TCX: cp227.tcx

1. 出力すべきトークン列

   C5 BF 顛 a b C3 9F

2. ptexenc へ渡るもの

   C5 BF 01 C5 01 BF a b C3 9F

3. 出力

   C5 BF 顛ab C3 9F　(= ſ顛abß)

\message は文字列化を挟むがほぼ同じ

端末・ファイルからの入力 (pTEX)

各行は ptexenc がコード変換し，配列 *buffer* に格納

- JIS X 0208 範囲外の Unicode 文字
  → ^^ 記法に変換
- 1 バイト目に現れないバイト (80–C1, F5–FF) 単独
  →変換せずそのまま
- 80–BFが決まった数だけ続かないもの

  (C2–F4)　　(E0–F4) (80–DF)　　(F0–F4) (80–DF) (80–DF)

  → TL2019 以前では A2AF (EUC), 81AD (SJIS) に変換
  これが platex/#86 の一要因

端末・ファイルからの入力 (pTEX)

各行は ptexenc がコード変換し，配列 *buffer* に格納

- JIS X 0208 範囲外の Unicode 文字
  → ^^ 記法に変換
- 1 バイト目に現れないバイト (80–C1, F5–FF) 単独
  →変換せずそのまま
- 80–BFが決まった数だけ続かないもの

  (C2–F4)　　(E0–F4) (80–DF)　　(F0–F4) (80–DF) (80–DF)

  → TL2019 以前では A2AF (EUC), 81AD (SJIS) に変換
  TL2020 では ^^ 記法に変換 (r52071)

端末・ファイルからの入力 (upTEX)

各行は ptexenc がコード変換し，配列 *buffer* に格納

- 1 バイト目に現れないバイト (80–C1, F5–FF) 単独
  →変換せずそのまま
- 80–BFが決まった数だけ続かないもの
  → ^^ 記法に変換（安全）

UTF-8 入力（補足）

- A2 AF (EUC), 81 AD (SJIS) は何者？
  → JIS X 0208 で最初に未割り当て」なコード

  UTF-8 列 ⟶ スカラー値 ⟶ JIS X 0208 ⟶ 内部コード
  C9 「0」 「2 区 15 点」 A2AF (EUC)

- 冗長な UTF-8 バイト列の扱い

  pTeX 特に問題にしない
  upTeX 最短のバイト数での表現に変換
  但し U+0000–U+007F については
  00 1F のように前に 00 が付加

配列 *buffer*

入力の各行を格納する配列

制御綴名読み取りの際，*buffer* 内で ^^ 記法を変換

前　a^^c3^^9f.\  ^^c5^^bf^^c3^^9f.

後　a^^c3^^9f.\  .

仮定：80–FFの catcode は 11

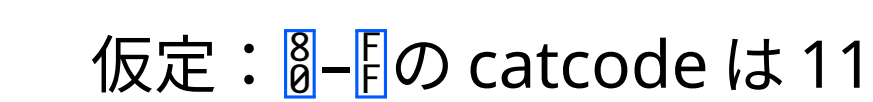
配列 *buffer*

入力の各行を格納する配列

制御綴名読み取りの際，*buffer* 内で ^^ 記法を変換

前　a^^c3^^9f.\^^c5^^bf^^c3^^9f.

後　a^^c3^^9f.\  .



制御綴名の和文・欧文の区別も簡単に実装したい

→和文由来バイトのフラグを別配列に格納

buffer は 8 bit のまま（ptexenc は他プログラムでも使う）