

ϵ -pT_EX 取扱説明書

北川 弘典 (H7K)*

2010 年 4 月 20 日, build 100420

1 はじめに

ϵ -pT_EX は, 東京大学理学部数学科 3 年生対象の 2007 年度の授業「計算数学 II」^{*1}において北川が作成したプログラムである. もともとは pT_EX 3.1.10 を基盤として, ϵ -T_EX 2.2 相当の機能や 10 進 21 桁の浮動小数点演算を追加したものであった.

しかし, 本版においては, 黒木氏などからのアドバイスもあり, ϵ -T_EX をベースにして pT_EX 拡張を組み込むという実装をとっていて, また, 実装の中途半端さから, 浮動小数点演算は削除^{*2}している.

製作の動機や作業過程などについては, 詳しくは [1] を参照して欲しいけれども, 大雑把に言うとも, 動機は以下のように要約できる.

- pT_EX は, T_EX が持っている「レジスタ 1 種類につき 256 個まで」という制限をひきずっており, 現状でも非常に多数のパッケージを読み込ませたりすると制限にぶち当たってしまう.
- 一方, ϵ -T_EX 拡張ではこれが「レジスタ 1 種類につき 32768 個まで」と緩和されており, 欧文で標準となっている pdfT_EX やその後継の LuaT_EX, 及び X_YT_EX でも ϵ -T_EX の機能が取り込まれている.
- そうすると, pT_EX だけが制限をレジスタ制限を引きずっているのは世界から取り残されることになるのではないか.

インストールについては, ϵ -pT_EX のアーカイブ内にある INSTALL.txt を参照してほしい.

* <http://sourceforge.jp/projects/eptex/wiki/>, e-mail: h_kitagawa2001(at)yahoo.co.jp

*1 <http://ks.ms.u-tokyo.ac.jp/>

*2 primitive で浮動小数点が扱えることが重要であることは認めます. しかし, 今までの 95 bit の実装では, 10 進でしかも機械に依存しない形で実装されているとはいえ, ϵ -pT_EX の中だけで終わってしまう (つまり, 他拡張でも使われる可能性はない) ように思います.

T_EX 一族に浮動小数点演算が実装されるのであれば, 「float 型専用のレジスタ」とか, IEEE754 にあるような 128 bit 精度のフォーマットの採用とか, もっとしっかりとした実装であるべきだと僕は考えています. そういう浮動小数点演算の実装が出ればもちろん採用するつもりですし, 僕の方でも (もし気力 + 時間 + 体力が許せば) ϵ -pT_EX の枠外でまた 0 から書いていきたいな, と思っています.

以下の理由から，コンパイルは $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ Live 2009 対応の ptexlive (特に ptexlive-20100322) に限っている：

- 前版 ($\varepsilon\text{-p}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ -100201) では ptetex3, (TL2008 対応の) ptexlive でのコンパイルに対応させているが，これとの変更点はスクリプトなど周辺部分のみ
- $\mathrm{up}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ も開発が ptexlive ベースに移行するようである

二つ目の理由により， $\mathrm{up}\mathrm{T}_{\mathrm{E}}\mathrm{X}$, $\varepsilon\text{-}\mathrm{up}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の自動インストールは本版では見送っている． $\mathrm{up}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は ptexenc, dvipdfmx, pxdvi などパッチするが，公式ではどのような対応をとるのか現時点ではわからないためである．

2 $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張について

前に述べたように， $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ は $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の拡張の一つである． $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ のマニュアル [5] には，開発目的が以下のように述べられている．

The $\mathcal{N}\mathcal{T}\mathcal{S}$ project intends to develop an ‘New Typesetting System’ ($\mathcal{N}\mathcal{T}\mathcal{S}$) that will eventually replace today’s $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 3. The $\mathcal{N}\mathcal{T}\mathcal{S}$ program will include many features missing in $\mathrm{T}_{\mathrm{E}}\mathrm{X}$, but there will also exist a mode of operation that is 100% compatible with $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 3. It will, necessarily, require quite some time to develop $\mathcal{N}\mathcal{T}\mathcal{S}$ to maturity and make it widely available.

Meanwhile $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ intends to fill the gap between $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 3 and the future $\mathcal{N}\mathcal{T}\mathcal{S}$. It consists of a series of features extending the capabilities of $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 3.

$\mathcal{N}\mathcal{T}\mathcal{S}$ がどうなったのか僕は知らない．しかし，少なくとも $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張自体は実用的な物であり，そのせいか \aleph (Aleph), pdf $\mathrm{T}_{\mathrm{E}}\mathrm{X}$, X $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ などの他の拡張にもマージされており，かなりの人が $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張を使うことができるようになっている．

$\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張で追加される機能について，詳しくは [5] を参照して欲しい．しかしそうやって丸投げするのはよろしくなさそうな気もするので，ひとまず [1] 中の 4.2 節「 $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ の機能」を引用することにする（一部改変）：

$\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ には Compatibility mode と Extended mode の 2 つが存在し，前者では $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 特有の拡張は無効になるのでつまらない．後者がおもしろい．

拡張機能を使うにはファイル名を渡すときに * をつけるかコマンドラインオプションとして `-etex` スイッチをつければいいが， $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張に関わる追加マクロは当然ながらそれだけでは駄目である．「plain マクロ for $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 」(`etex.fmt` というのが一番マシかな) では自動的に追加マクロである `etex.src` が呼ばれる． $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 下ではちょうど `etex.src` に対応した `etex` パッケージを読み込む必要がある．

レジスタの増加

最初に述べたように， $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ では 6 種類のレジスタが各 256 個ずつ利用できる．それぞれ

のレジスタには\dimen75 などのように 0 ~ 255 の番号で指定できる他、予め別名の定義をしておけばそれによって指定することもできる。これらのいくつかは特殊な用途に用いられる（例えば\count0 はページ番号などのように）ことになっているので、さらに user が使えるレジスタは減少する。

ϵ -T_EX では、追加のレジスタとして番号で言うと 256 ~ 32767 が使用できるようになった。上の pdf によると最初の 0 ~ 255 と違って若干の制限はあるようだが、それは些細な話である。追加された（各種類あたり）32768 - 256 = 32512 個のレジスタは、メモリの効率を重視するため sparse register として、つまり、必要な時に始めてツリー構造の中で確保されるようになっている。

式が使用可能に

T_EX における数量の計算は充実しているとは言い難い。例えば、

\dimen123 ← (\dimen42 + \@tempdima)/2

という計算を元々の T_EX で書こうとすると、

\dimen123=\dimen42\advance\dimen123by\@tempdima\dimen123=0.5\@tempdima

のように書かないといけない。代入、加算代入、乗算代入、除算代入ぐらいしか演算が用意されていない状態になっている（上のコードのように、 $d_2 += 0.8d_1$ というような定数倍を冠することは平気）。

ϵ -T_EX では、そのレジスタの演算に、他のプログラミング言語で使われているような数式の表現が使えるようになった。上の PDF では実例として

\ifdim \dimexpr (2pt-5pt)*\numexpr 3-3*13/5\relax + 34pt/2<\wd20

が書かれている。これは、

$$32 \text{ pt} = (2 \text{ pt} - 5 \text{ pt}) (3 - \text{div}(3 \cdot 13, 5)) + \frac{34 \text{ pt}}{2} < \text{\box20 の幅}$$

が真か偽かを判定していることになる。

`\middle primitive`

T_EX に\left, \right という primitive があり、それを使えば括弧の大きさが自動調整されるのはよく知られている。 ϵ -T_EX では、さらに\middle primitive が追加された。

具体例を述べる。

$$\left\{ n + \frac{1}{2} \middle| n \in \omega \right\} \left\{ n + \frac{1}{2} \middle| n \in \omega \right\}$$

これは以下の source で出力したものである：

```
\def\set#1#2{\setbox0=\hbox{\$ \displaystyle #1,#2$}%
\left\{\, \, \vphantom{\copy0}\#1 \,,\right|!\left.\,, \,
\vphantom{\copy0}\#2 \,,\right\}}
\def\eset#1#2{\left\{\, \, #1 \,,\middle|\,, \, #2 \,,\right\}}
\[\set{n+\frac{1}{2}}{n\in \omega} \set{n+\frac{1}{2}}{n\in \omega} \]
```

両方とも集合の表記を行うコマンドである． $\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 流の`\set` では 2 つの`\left`, `\right` の組で実現させなければならない，そのために | の左側と右側に入る式の最大寸法を測定するという面倒な方法を使っている．その上，この定義では`\textstyle` 以下の数式（文章中の数式とか）ではそのまま使えず，それにも対応させようとすると面倒になる．一方， $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 流の`\eset` では，何も考えずに`\left`, `\middle`, `\right` だけで実現できる．

$\mathrm{T}_{\mathrm{E}}\mathrm{X}\text{--}\mathrm{X}_{\mathrm{L}}\mathrm{T}$ ($\mathrm{T}_{\mathrm{E}}\mathrm{X}\text{--}\mathrm{X}_{\mathrm{e}}\mathrm{T}$)

`left-to-right` と `right-to-left` を混植できるという機能であるらしい．ヘブライ語あたりの組版に使えるらしいが，よく知らない．ここでの `RtoL` は `LtoR` に組んだものを逆順にしているだけのような気がする．

とりあえず一目につきそうな拡張機能といたらこれぐらいだろうか．他にも `tracing` 機能や条件判断文の強化などあるが，そこら辺はパツとしないのでここで紹介するのは省略することにしよう．

$\varepsilon\text{-p}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ ではここに述べた代表的な機能を含め，ほとんどすべての機能を実装しているつもりである．ただ， $\mathrm{T}_{\mathrm{E}}\mathrm{X}\text{--}\mathrm{X}_{\mathrm{L}}\mathrm{T}$ を和文で使うと約物の位置がずれたり空白がおかしかったりするけれども，その修正は大変に思えるし，苦勞して実装する意味があるのか疑問なので放置している．

`\lastnodetype` と `\currentiflevel` の挙動については，[1] にもあるが， $\mathrm{p}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 拡張のため，（それらで追加された部分に関しては） $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 本来の挙動では起こり得ない値をとることがある．まず `\lastnodetype` については，以下のような値をとる．

−1: none (empty list)	6: adjust node	13: penalty node
0: char node	7: ligature node	14: unset node
1: hlist node	8: disc node	15: math mode nodes
2: vlist node	9: whatsit node	16: direction node
3: rule node	10: math node	17: displacement node
4: ins node	11: glue node	
5: mark node	12: kern node	

次に，`\currentiflevel` における条件判断文とそれを表す数字との対応は，以下のようになっている．

1: <code>\if</code>	8: <code>\ifmmode</code>	15: <code>\iftrue</code>	22: <code>\ifydir</code>
2: <code>\ifcat</code>	9: <code>\ifinner</code>	16: <code>\iffalse</code>	23: <code>\ifmdir</code>
3: <code>\ifnum</code>	10: <code>\ifvoid</code>	17: <code>\ifcase</code>	24: <code>\iftbox</code>
4: <code>\ifdim</code>	11: <code>\ifhbox</code>	18: <code>\ifdefined</code>	25: <code>\ifybox</code>
5: <code>\ifodd</code>	12: <code>\ifvbox</code>	19: <code>\ifcsname</code>	
6: <code>\ifvmode</code>	13: <code>\ifx</code>	20: <code>\iffontchar</code>	
7: <code>\ifhmode</code>	14: <code>\ifeof</code>	21: <code>\iftdir</code>	

3 「FAM256」パッチについて

FAM256 パッチは、掲示板 T_EX Q & A の山本氏の書き込み [2] に刺激されて作ったものであり、以下に説明する Ω の一部機能^{*3}を使えるようにするパッチである（この名称は便宜的なもの）。本パッチは ε -T_EX 拡張とは関係はありません。

本版も含め、091003 版以降では、FAM256 パッチは標準で有効になる。 ε -T_EX 拡張とは違い、FAM256 パッチを有効にしてバイナリを作った場合、追加機能は extendend mode でなくても有効になっている。ただし、後に述べるように、本パッチではレジスタが 65536 個まで使えるようにしているが、それについてだけは extended mode の時に限り有効になる。

3.1 機能解説

本ドキュメントの最後のページ^{*4}にちょっとしたサンプルを載せてある。

数式フォント制限の緩和

Ω の大きな特徴としては、T_EX 内部のデータ構造を倍の領域を用いるように改変し^{*5}、T_EX に従来から存在していた「256 個制限」を 2^{16} 個にまで緩和したことが挙げられる。同様に、 Ω では ([2] にもあるように) 数式フォントを同時に 256 個まで用いることができ、各フォントも 65536 文字まで許されるようになっている。

FAM256 パッチでは、中途半端だが、数式フォント 1 つあたりの使用可能文字数は 256 個のままで、同時に数式フォントを 256 個まで使えるようにしている。基本的には Ω と同様の方法を用いているが、内部でのデータ構造に違いがある（数字はすべて bit 幅）：

	category	family	char	math code	delimiter code
T _E X	3	4	8	$3 + 4 + 8 = 15$	$3 + 2 \cdot (4 + 8) = 27$
Ω	3	8	16	$3 + 8 + 16 = 27$	$(3 + 8 + 16, 8 + 16) = (27, 24)$
FAM256	3	8	8	$3 + 8 + 8 = 21$	$(3 + 8 + 8, 8 + 8) = (19, 16)$

T_EX に既存のコマンド類は互換性維持のために同じ動作とする必要があるので、16 番から 255 番のフォントを利用する際には別のコマンドが必要となる。（実装自体に Ω の流儀を使っているから）FAM256 では、 Ω のコマンド類を流用することにした。すなわち、FAM256 では、以下の primitive が追加されている^{*6}。

- $\backslash\mathrm{omathcode} \langle 8\text{-bit number} \rangle = \langle 27\text{-bit number} \rangle$
- $\backslash\mathrm{omathcode} \langle 8\text{-bit number} \rangle$

^{*3} LuaT_EX ではこれらの機能は使えるはずである。見た感じだと Xe_LT_EX では使えないようだ。

^{*4} ただし、ソースファイルで言えば fam256d.tex（本文）と fam256p.tex（preamble 部）に対応する。

^{*5} 詳しい話は texk/web2c/texmfmem.h 中の共用体 memoryword の定義を参照。大雑把に言うと、1 つの「メモリ要素」に 2 つの 32 bit 整数を同時に格納できるようになっている。

^{*6} Ω では $\langle 8\text{-bit number} \rangle$ のところが $\langle 16\text{-bit number} \rangle$ になっている。

- `\omathchar` $\langle 27\text{-bit number} \rangle$
- `\omathaccent` $\langle 27\text{-bit number} \rangle$
- `\omathchardef` $\langle control\text{-}sequence \rangle = \langle 27\text{-bit number} \rangle$
- `\odelcode` $\langle 8\text{-bit number} \rangle = \langle 27\text{-bit number} \rangle \langle 24\text{-bit number} \rangle$
- `\odelimiter` $\langle 27\text{-bit number} \rangle \langle 24\text{-bit number} \rangle$
- `\oradical` $\langle 27\text{-bit number} \rangle \langle 24\text{-bit number} \rangle$

ここで、27 bit とか 24 bit の自然数の意味については、上の表の Ω の行を参照して欲しい。上に書いた FAM256 での実装から、character code の指定に使われる 16 bit の数値で、実際に使われるのは下位 8 bit であり、上位 8 bit は無視される。なお、`\odelcode` $\langle 8\text{-bit number} \rangle$ として delimiter code を取得しようとしても、現時点のパッチでは、うまく動作しない*⁷。

当然ながら、 \LaTeX において数式 fam を 16 個以上使うには、`\omathchar` などのコマンドに対応したマクロを使う必要がある。実験的と書かれてはいるが、山本氏による「最低限のパッケージ」[4] が手っ取り早いような気がする。

無限のレベル

\TeX では、glue の伸縮量に 3 つの無限大のレベルが存在した：`fil`, `fill`, `fillll` であり、1 が多いほど無限大のオーダーが高い。 Ω では、「inter-letter spacing のために」`fi` という、有限と `fil` の中間にあたる無限大のレベルが付け加えられ、`\hfi`, `\vfi` という 2 つの primitive も追加された。 \FAM256 パッチでは、この無限大レベル `fi` も採用することにした。

実装方法は、大まかには Ω で `fi` の実装を行っている change file `omfi.ch` の通りであるのだが、これに \pTeX や $\varepsilon\text{-TeX}$ に伴う少々の変更を行っている。

- コマンド `\pagefistretch` を新たに定義している。
- `\gluestretchorder`, `\glueshrinkorder` の動作を $\varepsilon\text{-TeX}$ のそれと合わせた。具体的には、ある適当な glue `\someglue` の stretch 幅を $\langle stretch \rangle$ とおくと、

$$\backslash\gluestretchorder\backslashsomeglue = \begin{cases} 0 & \langle stretch \rangle \text{ が高々 } fi \text{ レベルの量} \\ 1 & \langle stretch \rangle \text{ がちょうど } fil \text{ レベルの無限量} \\ 2 & \langle stretch \rangle \text{ がちょうど } fill \text{ レベルの無限量} \\ 3 & \langle stretch \rangle \text{ がちょうど } fillll \text{ レベルの無限量} \end{cases}$$

となっている。内部では `fi` レベルが 1, `fil` レベルが 2,として処理している。

レジスタについて

Ω では（前にも書いたが）データ構造の変更が行われ、それによってレジスタが各種類あたり 65536 個使えるようになっている。

一方、 $\varepsilon\text{-TeX}$ では、256 番以降のレジスタを専用の sparse tree に格納することにより、32767 番までのレジスタの使用を可能にしていた。この tree 構造を分析してみると、65536 個までレジ

*⁷ 51 bit 自然数を返さないといけないうえからねえ。やる気があれば検討してみます。

スタを拡張するのはさほど難しくないことのように思われた．具体的には，`tree` の階層を 1 つ増やしてみた（だから，おそらく各種類あたり $16 \cdot 32768 = 524288$ 個まで使えるとは思うが，これはきりが悪い）．そこで，FAM256 パッチでは ε -T_EX 流の方法を用いながらも，レジスタをさらに 65536 個まで増やしている．

4 互換性

ε -T_EX, pT_EX との互換性をはかるのに有効な手段としては，まず TRIP test がある．[1] にも書いたが，これは T_EX のソースの全行を実行するようなテストソースであり，TRIP test の実行結果が違えば，どこかの動作が違っていることが分かるという仕掛けである．

幸いにも， ε -pT_EX の（compatibility mode における）TRIP test の実行結果は，pT_EX のそれとほとんど同じであり，違う点は以下のみであった：

- Memory usage の違い：

```
Memory usage before: $x$&$y$; after: $z$&$w$; still untouched: $u$
```

のところで， x, z が 4 多く， u が 4 少ない．

- (FAM256 パッチのおかげで) 数式フォントが 256 個同時に扱えるようになったことによる，エラーの未発生やエラーメッセージの違い．
- ログの最後に出てくる，メモリ総使用量．

また，extended mode と compatibility mode による出力の違いを検証すると，Memory usage の違いやメモリ総使用量の他には，e-TRIP の説明書 p. 4 の項目 4. にある違いしか見受けられなかった，

次に， ε -T_EX には， ε -T_EX によって拡張された部分を調べる，同様の e-TRIP test が存在する．pT_EX 系列では，`\showbox` などでいちいち組方向を表示するので，それを

```
$ alias sep='sed "s/, yoko direction//;s/yoko direction, //;\n\ns/yoko(math) direction, //"'
```

という alias を利用することで log file から除去し，それと ε -T_EX での e-TRIP test の出力と比べた結果，

- Memory Usage の違い（略）
- レジスタの使用可能個数を 65536 個/種類としたことによるエラーの未発生
- (和文文字のため) haracter code に 256 以上も許すことによるエラーの未発生．
- `last_node_type` の検証の 3 箇所での失敗．

という違いを得た．最後の点が気になる（ずっと前から放置）が，今はひとまず放置する．

すると、最後に、 $\mathrm{pT_E X}$ 特有の拡張部分のテストをしたくなるわけだが、北川の知る限りにおいては、そのようなテストソースは公式には存在しないようである。しかし、やはり気になる問題であり（さる方面から圧力もきたので）、個人的に作ってみたものが `ptex-qtrip` である。詳細な説明はそちらのドキュメントに譲る。ともかく、これを動作させて `log file` を比べてみると、メモリ関連や `FAM256` 関連以外は

```
-% split2 to -0.00293,6.94444 p=-10000
+% split2 to -0.00299,6.94444 p=-10000
```

以外の違いしかなかった。わずか 0.00006 の違い（単位がもし `point` だとしたら $4\mathrm{sp}$ ）だが、どこか気になるところである。

上の箇所もあるし、また `ptex-qtrip` 自体がまだ完全なものではなのだが、 $\mathrm{pT_E X}$ と $\varepsilon\text{-}\mathrm{pT_E X}$ は実用上においては互換と言えそうな気もしなくもない。

5 最後に

[1] にも書いたような気がするが、 $\varepsilon\text{-}\mathrm{pT_E X}$ は単につなぎでしかない、と思っている。 $\mathrm{upT_E X}$, $\varepsilon\text{-}\mathrm{pT_E X}$ によってある程度の新機能が利用可能になっている間に、 $\mathrm{X_T E X}$ や $\mathrm{LuaT_E X}$ といった次世代の $\mathrm{T_E X}$ に移行する準備を整えていくのが良いのではないだろうか？

参考文献

- [1] 北川 弘典, 「計算数学 II 作業記録」, 2008.
<https://sourceforge.jp/projects/eptex/document/resume/ja/1/resume.pdf> 他
- [2] 山本 和義, 「数式 fam の制限と `luatex`」, 掲示板「 $\mathrm{T_E X}$ Q & A」52744 番書き込み, 2009.2.12,
<http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/52744.html>
- [3] 山本 和義, 「Re: 数式 fam の制限と `luatex`」, 掲示板「 $\mathrm{T_E X}$ Q & A」52767 番書き込み,
2009.2.16, <http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/52767.html>
- [4] 山本 和義, 「数式 fam 拡張マクロ for $\varepsilon\text{-}\mathrm{pT_E X}$ 等」, 掲示板「 $\mathrm{T_E X}$ Q & A」52799 番書き込み,
2009.2.21, <http://oku.edu.mie-u.ac.jp/~okumura/texfaq/qa/52799.html>
- [5] The $\mathcal{N}\mathcal{T}\mathcal{S}$ Team. *The $\varepsilon\text{-}\mathrm{T_E X}$ manual*.
http://www.tug.org/texlive/Contents/live/texmf-dist/doc/etex/base/etex_man.pdf
- [6] J. Plaice, Y. Haralambous. *Draft documentation for the Ω system*, 1999.
http://www.tug.org/texlive/Contents/live/texmf-dist/doc/omega/base/doc_1.8.tex

Test source for FAM256 patch

本ソースは山本和義氏による「数式 fam の制限と luatex」(qa:52744) 中のコードをベースにしたものである。

More than 16 math font families.

ABCDEFGHIJKL fam = 19

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAoAp fam35

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAoAp fam51

AaAbAcAdAeAfAgAhAiAjAkAlAmAnAoAp fam67

Aa fam68 Ab fam69 Ac fam70 Ad fam71 roman

\omathchar etc.

\mathchar"7F25 : %, \omathchar"7420125 : %

meaning of \langle: macro:->\delimiter "426830A ,

meaning of \lx: macro:->\odelimiter "4450068"030001

\lx:h, \bigl\lx:), \Bigl\lx:)

\the\mathcode'\f: 7166, \the\omathcode'\f: 7010066 (どちらも 16 進に変換した)
t>>>))) e e

$$\sqrt{\cdot}, \sqrt[n]{\cdot}, \sqrt[n]{\int_V f d\mu}, \sqrt{\int_V f d\mu}$$

\odelcode primitive による delimiter code の取得はうまく動かない:

\the\delcode'\|: 618, \the\odelcode'\|: -1 (どちらも 10 進)

Infinite level “fi”

(fi)	(fil)	(fill)	(filll)
(fi)	(fil)	(fill)	
(fi)	(fil)		
(fi)	(fi)	(fi)	

65536 registers

fuga! a 漢字仮名 sdf T_EX ほげほげ T_EX

589