

Plain T_EX quick reference



Conventions used in this document

The major sections are keyed to the corresponding chapter of Don Knuth's *The T_EXbook*.

<cs>	Control sequence, e.g., \mymacro, \somedimen
<N>	An integer from 0 to 255
<file>	An integer from 0 to 15 (file number), or -1 to read terminal
<int>	An integer
[]	Optional item
...	One or more of the preceding item

Chapter 2: Book printing vs. ordinary typing

-	Hyphen (-)
--	En-dash (–)
---	Em-dash (—)
\thinspace	Thin space:

Chapter 3: Controlling T_EX

\input <file>	Go read <file>
\endinput	Stop reading this file
_	Control space (where _ is a space character)
\TeX	The T _E X logo

Chapter 4: Fonts of type

\rm	Roman (the default)
\it	<i>Italic</i>
\/	Italic correction; use for italic→non-italic transition
\sl	<i>Slanted</i>
\tt	Typewriter
\bf	Bold extended
\tenpoint	Or \ninepoint, etc.
\tenrm	Or \ninesl, etc.
\font<cs> [scaled <int>]	Load a font; <int>=1000 for normal size
\magstephalf	Use after scaled to magnify by 1.095%
\magstep1	Use after scaled to magnify by 1.2

<code>\magstep2</code>	Use after scaled to magnify by 1.44
<code>\magstep3</code>	Use after scaled to magnify by 1.728%
<code>\magstep4</code>	Use after scaled to magnify by 2.0746%
<code>\magstep5</code>	Use after scaled to magnify by 2.488%

Chapter 5: Grouping

<code>{ ... }</code>	Localize all changes between the <code>{ ... }</code>
<code>\centerline{<text>}</code>	Center <code><text></code>
<code>\begingroup... \endgroup</code>	Functionally like <code>{ ... }</code>

Chapter 6: Running T_EX

<code>\relax</code>	Do nothing
<code>\hsize=<dimen></code>	Set horizontal size
<code>\tolerance=<int></code>	Allow bad interword spacing
<code>\raggedright</code>	Allow right margin to vary

Chapter 7: How T_EX reads what you type

To change the way character `<char>` is interpreted:

```
\catcode' \<char>=<category-number>
```

Here are the category codes, and an example of each:

0 (<code>\</code>)	Escape character
1 (<code>{</code>)	Begin group
2 (<code>}</code>)	End group
3 (<code>\$</code>)	Math shift
4 (<code>&</code>)	Tab separator
5 (<code>LF</code>)	End of line (newline)
6 (<code>#</code>)	Parameter
7 (<code>^</code>)	Superscript
8 (<code>_</code>)	Subscript
9 (<code>NUL</code>)	Ignored character
10 (<code>␣</code>)	Space
11 (<code>A-Z, a-z</code>)	Letter
12	Other character
13 (<code>~</code>)	Active character
14 (<code>%</code>)	Comment
15 (<code>DEL</code>)	Invalid character

To produce certain characters you'll need these control sequences:

```
\$                                            $
```

<code>\%</code>	<code>%</code>
<code>\&</code>	<code>&</code>
<code>_</code>	<code>_</code>
<code>\string<cs></code>	E.g., <code>\string\TeX</code> yields <code>\, T, e, X</code> , all in category 12
<code>\csname<token>...\endcsname</code>	Convert tokens to a control sequence
<code>\number<int></code>	Yields the digits of <code><int></code> as a token list
<code>\romannumeral<int></code>	E.g., <code>\romannumeral 23</code> yields <code>x, x, i, i, i</code>
<code>\uppercase{<token>...}</code>	Convert lowercase to uppercase
<code>\lowercase{<token>...}</code>	Convert uppercase to lowercase

Chapter 8: The characters you type

<code>\char<int></code>	Produces the character with decimal code <code><int></code>
<code>\char'<octal></code>	Produces the character with octal code <code><octal></code>
<code>\char"<hex></code>	Produces the character with hex code <code><hex></code>
<code>\chardef<cs>=<int></code>	Associate the character whose code is <code><int></code> with control sequence <code><cs></code>
<code>^^@</code>	The NUL character
<code>^^A</code>	Control A
<code>^^<char></code>	In general, for <code><char></code> less than 64, this produces character <code><char>+64</code> , otherwise it produces <code><char>-64</code>

Chapter 9: T_EX's Roman fonts

Use	To get	
<code>\'e</code>	è	Grave accent
<code>\'e</code>	é	Acute accent
<code>\^o</code>	ô	Circumflex
<code>\"o</code>	ö	Umlaut
<code>\i</code>	ı	Dotless i, for diacriticals over letter i
<code>\j</code>		Dotless j
<code>\~n</code>	ñ	Tilde
<code>\=a</code>	ā	Long mark
<code>\.s</code>	š	Dot accent
<code>\u a</code>	ǎ	Short mark
<code>\d n</code>	ṅ	Dot under
<code>\v c</code>	č	Hacek
<code>\H o</code>	ø	Long umlaut
<code>\t ii</code>	î	Tie-after
<code>\b o</code>	o̅	Bar under
<code>\c c</code>	ç	Cedilla
<code>\aa</code>	å	Circle-a
<code>\L</code>	Ł	Polish l
<code>\ss</code>	ß	Es-zet
<code>\o</code>	ø	Slash-o
<code>\dag</code>	†	Dagger

<code>\ddag</code>	‡	Double dagger
<code>\S</code>	§	Section symbol
<code>\P</code>	¶	Paragraph symbol
<code>\ae</code>	æ	Ligatures
<code>\AE</code>	Æ	
<code>\oe</code>	œ	
<code>\OE</code>	Œ	

Chapter 10: Dimensions

A *<dimen>* can be expressed as a number followed by one of these units:

<code>pt</code>	Point, 1/72.27"
<code>pc</code>	Pica = 12 pt
<code>in</code>	Inch
<code>bp</code>	Big pt., 1/72"
<code>cm</code>	Centimeter
<code>mm</code>	Millimeter
<code>dd</code>	1157 Didot pts. = 1238 pt
<code>cc</code>	Cicero = 12 dd
<code>sp</code>	Scaled point = 1/65536 pt

Use `\magnification=<int>` to change the overall magnification of the document, where unity is 1000.

Use `true <dimen>` to get a dimension that is not affected by any `\magnification` that may be in effect.

Chapter 11: Boxes

<code>\hbox{<text>}</code>	Horizontal box: align baselines
<code>\vbox{<text>}</code>	Vertical box: align reference points

Chapter 12: Glue

A *<glue>* has the form:

<dimen> [plus *<dimen>*] [minus *<dimen>*]

Glue-related control sequences:

<code>\smallskip</code>	3pt plus 1pt minus 1pt
<code>\medskip</code>	Twice <code>\smallskip</code>
<code>\bigskip</code>	Twice <code>\medskip</code>
<code>\vfil, \vfill</code>	Vertical fill, two strengths
<code>\hfil, \hfill</code>	Horizontal fill, two strengths
<code>\hss, \vss</code>	Infinite glue
<code>\line{<text>}</code>	<code>\hbox</code> to <code>hsize{...}</code>

<code>\hbox to <dimen>{...}</code>	Sized hbox; also works with <code>\vbox</code>
<code>\hbox spread <dimen>{...}</code>	Natural size plus <code><dimen></code> ; also <code>\vbox</code>
<code>\baselineskip=<glue></code>	Baseline-to-baseline distance
<code>\lineskip=<glue></code>	Use if baselines closer than <code>\lineskiplimit</code>
<code>\lineskiplimit=<glue></code>	See previous item
<code>\prevdepth</code>	Depth of last box on main vertical list
<code>\nointerlineskip</code>	Suppress next interline glue
<code>\vtop</code>	Likcs <code>\vbox</code> but use top, not bottom, baseline
<code>\strut</code>	An invisible rule with 8.5 points of height and 3.5 points of depth
<code>\llap{<text>}</code>	Set to left of current position, hide width
<code>\rlap{<text>}</code>	Same as <code>\llap</code> , but to right of current position

Chapter 13: Modes

T_EX operates in one of six modes at any given time:

vertical	Building the main vertical list
internal vertical	Constructing a <code>\vbox</code>
horizontal	Building a paragraph
restricted horizontal	Constructing an <code>\hbox</code>
math	<code>\$...\$</code>
display math	<code>\$\$...\$\$</code>

Chapter 14: How T_EX breaks paragraphs into lines

Things that can appear in a horizontal list are placed in two categories:

non-discardable	box, discretionary, whatsit, vertical material
discardable	glue, kern, penalty, math-on

Discardable items disappear at a line break.

<code>\slash</code>	Like <code>/</code> , but allow a line break after it
<code>~</code>	Tie (non-break space)
<code>\obeylines</code>	Treat newline as <code>\par</code>
<code>\break</code>	Break the line here; use <code>\hfil\break</code> to avoid stretching
<code>\discretionary{<pre>}{<post>}{<no>}</code>	Discretionary item: <code><pre></code> is the pre-break text, <code><post></code> is the post-break text, and <code><no></code> is used if there is no break
<code>\-</code>	Discretionary hyphen
<code>\nobreak</code>	Don't break the line here!
<code>\parindent=<glue></code>	Set the paragraph indentation
<code>\parfillskip=<glue></code>	Placed after last line in a paragraph
<code>\leftskip=<glue></code>	Left margin skip

<code>\rightskip=<glue></code>	Right margin skip
<code>\narrower</code>	Reduce <code>\leftskip</code> and <code>\rightskip</code> by the size of <code>\parindent</code>
<code>\parskip=<glue></code>	Inter-paragraph vertical spacing
<code>\parshape=<i>n i₁ l₁ i₂ l₂ ...</i></code>	Special paragraph shape
<code>\hangindent=<dimen></code>	Set width of hanging indentation
<code>\hangafter=<int></code>	If <code><int></code> is negative, indent the first <code>-<int></code> lines, else indent all lines after the first <code><int></code>
<code>\item{1.}</code>	Itemization, first level. . .
<code>\itemitem{a.}</code>	. . .and second
<code>\prevgraf</code>	Line count in previous/current paragraph
<code>\looseness=<int></code>	Stretch (– for shrink) next paragraph <code><int></code> lines
<code>\interlinepenalty=<int></code>	Penalties for: interline breaks. . .
<code>\clubpenalty=<int></code>	. . .break after the first line of a paragraph. . .
<code>\widowpenalty=<int></code>	. . .break before the last line of a paragraph. . .
<code>\brokenpenalty=<int></code>	. . .and break at a discretionary
<code>\vadjust{...}</code>	Place material after the current line in the vertical list
<code>\everypar=<token>...</code>	Insert token list before each paragraph

Chapter 15: How T_EX makes lines into pages

Things that can appear in a horizontal list are placed in two categories:

non-discardable	box, whatsit, mark, insertion
discardable	glue, kern, penalty

Discardable items disappear at a page break.

<code>\eject</code>	Break page; use <code>\vfill\eject</code> to avoid stretching
<code>\smallbreak</code>	Like <code>\smallskip</code> but also a good page break
<code>\medbreak</code>	Same for <code>\medskip</code> . . .
<code>\bigbreak</code>	. . .and <code>\bigskip</code>
<code>\filbreak</code>	Keep text down to the next <code>\filbreak</code> on the same page
<code>\raggedbottom</code>	Don't stretch each page to <code>\vsize</code>
<code>\topinsert... \endinsert</code>	Top-of-page insertion
<code>\pageinsert... \endinsert</code>	Full-page insertion
<code>\midinsert... \endinsert</code>	Drop it here if you can
<code>\supereject</code>	Like <code>\eject</code> but also flush all insertions
<code>\footnote{<symbol>}{<text>}</code>	Attach a footnote to this point
<code>\count<N>=<int></code>	Count registers, signed 32-bit number
<code>\dimen<N>=<dimen></code>	Dimension registers
<code>\skip<N>=<glue></code>	Skip registers
<code>\muskip<N>=<muglue></code>	Math unit skip registers
<code>\advance<reg> by <int></code>	Add or subtract
<code>\multiply<reg> by <int></code>	Multiply
<code>\divide<reg> by <int></code>	Divide

<code>\global</code>	Prefix to transcend current scope
<code>\setbox<N>=<text></code>	Box registers
<code>\box<N></code>	Copy (destructive; set box <N> to null)
<code>\copy<N></code>	Non-destructive copy of box <N>
<code>\unhbox<N></code>	Unset glue and copy (destructive)
<code>\unhcopy<N></code>	Unset glue and copy (non-destructive)
<code>\unvbox, \unvcopy</code>	Vertical versions of <code>\unhbox</code> and <code>\unhcopy</code>
<code>\showthe<item></code>	Show count, dimen, or skip
<code>\showbox<box></code>	Show the contents of a box
<code>\newcount<cs></code>	Allocate a new count register*
<code>\newdimen</code>	Allocate a new dimension register*
<code>\newskip</code>	Allocate a new skip register*
<code>\newmuskip</code>	Allocate a new math unit skip register*
<code>\newbox</code>	Allocate other item*

***Note:** Box 255, and count/dimen/skip/muskip registers 0–9 are special; none of the `\new<thing>` commands will allocate 0–9.

<code>\insert<N><vert. material></code>	Insertion of type <N>
<code>\newinsert<cs></code>	Allocate a new insertion type. Each insertion type <N> is tied to:
<code>\box<N></code>	Where the material appears upon output
<code>\count<N></code>	Magnification factor: 1000=unity, 500 for double-column
<code>\dimen<N></code>	Maximum insert size per page
<code>\skip<N></code>	Extra space allocated on the page for this insert type
<code>\vsplit<N> to <dimen></code>	Split off the first <dimen> of box <N>. Uses <code>\splitmaxdepth</code> , <code>\splittopskip</code>

Chapter 20: Definitions (also called Macros)

<code>\def<cs>[<parm>...]{<replacement text>}</code>	Define a macro, where <parm> consists of parameter identifiers of the form #<int> optionally separated by delimiter strings
<code>\long\def...</code>	Allow <code>\par</code> tokens in the arguments
<code>\outer\def...</code>	Allow only at outer level (not in arguments, preambles, ...)
<code>\global\def...</code>	Definition transcends current group
<code>\gdef...</code>	Same as <code>\global\def</code>
<code>\let<cs>=<token></code>	Give the current meaning of <token> to <cs>
<code>\futurelet<cs><token1><token2></code>	Equivalent to <code>\let<cs>=<token2></code> followed by <token1><token2>
<code>\if<condition><>true-text>\else<>false-text>\fi</code>	General form of <code>\if</code>
<code>\ifnum<int1><relation><int2></code>	Compare numbers, where the <relation> can be any of <, =, or >

<code>\ifdim<dim1><relation><dim2></code>	Compare dimensions
<code>\ifodd<int></code>	True if <code><int></code> is odd
<code>\ifvmode</code>	If in vertical or internal vertical mode
<code>\ifhmode</code>	If in horizontal or restricted horizontal mode
<code>\ifmmode</code>	If in math or display math mode
<code>\ifinner</code>	If in internal vertical, restricted horizontal, or non-display math mode
<code>\if<token1><token2></code>	Compare character codes (ignore categories)
<code>\ifcat<token1><token2></code>	Compare category codes (ignore character codes)
<code>\ifx<token1><token2></code>	Compare unexpanded tokens to see if they have the same function
<code>\ifvoid<N></code>	True if box <code>N</code> is empty
<code>\ifhbox<N></code>	True if box <code>N</code> contains an hbox...
<code>\ifvbox<N></code>	...or vbox
<code>\ifeof<int></code>	True if input file <code><int></code> is at end-of-file
<code>\iftrue</code>	Always take the true branch
<code>\iffalse</code>	Always take the false branch
<code>\ifcase<int><case0-text>[\or<case1-text>]...[\else<default-text>]\fi</code>	Expand the case corresponding to <code><int></code>
<code>\newif<cs></code>	Declare a Boolean switch <code><cs></code>
<code>\<cs>true</code>	Set a Boolean switch
<code>\<cs>>false</code>	Clear a Boolean switch
<code>\newtoks<cs></code>	Declare a token list register
<code>\jobname</code>	Unqualified filename, e.g., "book" for "book.dvi"
<code>\fontname</code>	Expands to the name of the font file. Note: Use <code>\font</code> for the current <code></code> .
<code>\meaning<token></code>	Same as <code>"\let\test=<token> \show\test"</code>
<code>\expandafter<token><text></code>	Saves <code><token></code> unexpanded; expands <code><text></code> including any arguments; then puts <code><token></code> back at the front of the input
<code>\noexpand<token></code>	Expansion is <code><token></code> , treated as <code>\relax</code>
<code>\the<cs></code>	Expand internal quantity to token list
<code>\the<codename></code>	Display the code number for <code><codename></code> , where
<code><codename> ::= \catcode \mathcode \lccode \uccode \sfcode \delcode</code>	
<code>\the\fontdimen<int></code>	Produces a dimension characteristic of that font, and selected by <code><int></code> , e.g., 6 for the "em" dimension
<code>\showthe<cs></code>	Same as <code>\the</code> , but display on the terminal
<code>\message<token-list></code>	Write message on terminal
<code>\edef<cs>...</code>	Expanded definition—expand arguments now
<code>\xdef<cs>...</code>	Same as <code>\global\edef</code> . Use <code>\noexpand</code> to control partial expansion within <code>\edef</code> .
<code>\openin<file>=<filename></code>	Open an input stream
<code>\newread<cs></code>	Open the next input stream
<code>\closein<file></code>	Close an input stream
<code>\read<file> to<cs></code>	Input file to a token list register; use -1 for the terminal
<code>\loop<prequel>\if<condition><sequel>\repeat</code>	

Expands *<prequel>*, then evaluates the *<condition>*; if true, expands *<sequel>* and then returns to *<prequel>*

Chapter 21: Making boxes

<code>\vrule <dimitem></code>	Solid black box in horizontal mode. The default width is 0.4pt.
<code>\hrule <dimitem></code>	Solid black box in vertical mode. The default height is 0.4pt.
<code>\lastbox</code>	Remove the last box on the current horizontal or vertical list
<code>\unskip</code>	Remove the last item if it is glue or leaders
<code>\leaders<fillitem><glue></code>	Acts like <i><glue></i> but fills with <i><fillitem></i> where <i><fillitem></i> is a box or a rule. For tables of contents, use: <pre>\def\leaderfill\leaders\hbox to 1em{\hss.\hss\hfill}</pre> then use this for each line:
<code>\line{<text>\leaderfill <pageno>}</code>	
<code>\cleaders</code>	Like <i>\leaders</i> but centered in the space, and not aligned
<code>\xleaders</code>	Like <i>\leaders</i> but expanded to fill the space
<code>\openout<file>=<filename></code>	Open an output file
<code>\closeout<file></code>	Close an output file
<code>\write<file>{<token list>}</code>	Write a token string (deferred until <i>\shipout</i> time)
<code>\immediate\write...</code>	Write it now, don't defer it until the output routine. Note: Use 16 for the <i><file></i> to write on the terminal.
<code>\special{<keyword>[<arg>]}</code>	Image insertion, for example

Written by John W. Shipman (tcc-doc@nmt.edu). This version printed 2003-06-30. Copyright © 2001 by the New Mexico Institute of Mining and Technology.