

THE KNITTING PACKAGE

ARIEL BARTON

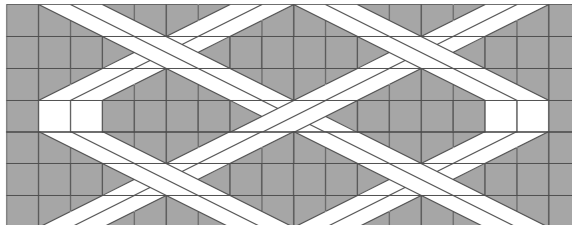
CONTENTS

1. License	2
2. Installation	2
3. Using the package	3
3.1. Stitch symbols produced or modified with commands	5
3.2. The four knitting fonts and other commands affecting overall appearance	6
3.3. Repeat outlines	9
3.4. Colors and miscellaneous commands	10
3.5. Spaces in the input	11
4. Ligatures and cables	11
4.1. Symbolic cables	12
4.2. Stylized cables	13
5. Counting rows	14
6. Counting stitches	15
7. Revision history	18
8. List of files that are considered part of this package	19

This package was written to make knitting charts using L^AT_EX or plain T_EX. It consists of several PostScript fonts of knitting symbols, font-support documents, and packages of commands.

Here's an example of the code and output:

```
\chart{  
=CCppggKKCCppggKK=  
===KKkk===KKkk===  
=ggKKCCppggKKCCpp=  
=-----kkKK=====--=  
=CCppggKKCCppggKK=  
===KKkk===KKkk===  
=ggKKCCppggKKCCpp=  
}
```



More examples may be found in the file `knitexamples.tex` and in later sections of this document.

1. LICENSE

This work (the knitting package) consists of all files listed in Section 8. It is copyright Ariel Barton, 2011.

This work may be distributed and/or modified under the conditions of the L^AT_EX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of the license is in

<http://www.latex-project.org/lppl.txt>

and version 1.3 or later is part of all distributions of L^AT_EX version 2003/06/01 or later.

This work has the LPPL maintenance status “author-maintained”.

The L^AT_EX project license above gives the conditions under which you may re-distribute or modify the fonts and files listed in Section 8. This license (loosely speaking) allows you to pass around copies of the package provided you re-distribute it in its entirety. In addition, any part, no matter how large or small, of the files `knitkey.tex` and `knitexamples.tex`, *and these two files only*, may be freely copied, verbatim or modified, into any document you write, without restriction.

As per the conditions of the LPPL, no restrictions are placed on running this package (i.e., compiling L^AT_EX documents that use this package). In particular, no restrictions are placed by the package author on selling or distributing patterns typeset using this package. Not all L^AT_EX packages may be used in commercial products; if you use other packages to produce a PDF or paper document, you must check their documentation to see if you are allowed to sell the result.

Suggestions and questions may be sent to the package author at origamist@gmail.com.

2. INSTALLATION

This package involves many supporting files. They should be put in appropriate places. Your distribution of T_EX may be able to do this for you. If it can't, you'll have to place them yourself.

Most modern T_EX distributions have a folder, usually named `texmf`, where you can store supporting files for the packages you add yourself.¹ All supporting files should be sorted into specific subfolders of `texmf`. The sorting rules are:

- `.fd` and `.sty` files go in `texmf/tex/latex`
- `.tex` files go in `texmf/tex/plain`
- `.tfm` files go in `texmf/fonts/tfm`
- `.pfb` files go in `texmf/fonts/type1`
- `.map` files go in `texmf/fonts/map`
- `.mf` files go in `texmf/fonts/source`
- `.afm` files go in `texmf/fonts/afm`

¹If you're using MacT_EX, this folder should be `Users/username/Library/texmf`. If it isn't there, create it.

If you're using MiK_TE_X, it is possible to designate any folder you like as the root of your local tree, that is, the place where you store supporting files. Instructions may be found at <http://docs.miktex.org/manual/localadditions.html#id573803> or through the manual which should have come with MiK_TE_X.

Any time you add supporting files to a local MiK_TE_X root, you have to refresh the file name database; see <http://docs.miktex.org/manual/configuring.html#fndbupdate>.

In all cases they can go in sub-subfolders; for example, `.tfm` files may be put into `texmf/fonts/tfm/knit` and not `texmf/fonts/tfm`.

If you're using some other distribution, you may have some entirely different place where you can put these files. Your distribution's documentation should tell you where.

If you really can't figure out where to put the files, if you're in a hurry, or if you're using someone else's computer and don't want to mess with their `texmf` folder, just dump every file you think you might need into the same folder as the document that uses the package. (This is probably all the `.tfm`, `.pfb`, and `.map` files, plus the `.sty` and `.fd` files if you are using \LaTeX or the `.tex` file if you are using plain \TeX .)

You aren't done! \TeX now knows everything it needs to do its job and *arrange* the characters in the font, and so your document will compile, but the postprocessing software (your DVI viewer, your printer, or the PDF files that `pdf \TeX` produces) don't know about the fonts themselves.

There's a simple way to tell `pdf \TeX` about the fonts: use the command

```
\pdfmapfile{+knitfont.map}
```

This map line can go in `knitting.sty` or the file that uses the package.

The advantage to this is that it involves nothing outside of the document you are compiling. On the other hand, it can be annoying to have to say that everywhere. And this doesn't work at all if you decide you want to produce DVI files and use a postprocessor such as `dvips`.

With `Mac \TeX` , I can cause `dvips`, `dvipdfm`, and `pdf \TeX` to know about these fonts by opening a Terminal window (command prompt) and typing `updmap --enable Map=knitfont.map`.

With `MiK \TeX` , I need to say `initexmf --edit-config-file updmap` and then add the line `Map knitfont.map` to the file which opens, then run `updmap` from the command line.

Something similar should work with most distributions; it is always wise to check and see what your distribution's documentation says about `updmap` before using it.

3. USING THE PACKAGE

Here is a minimal \LaTeX document that will produce a knitting pattern with a chart:

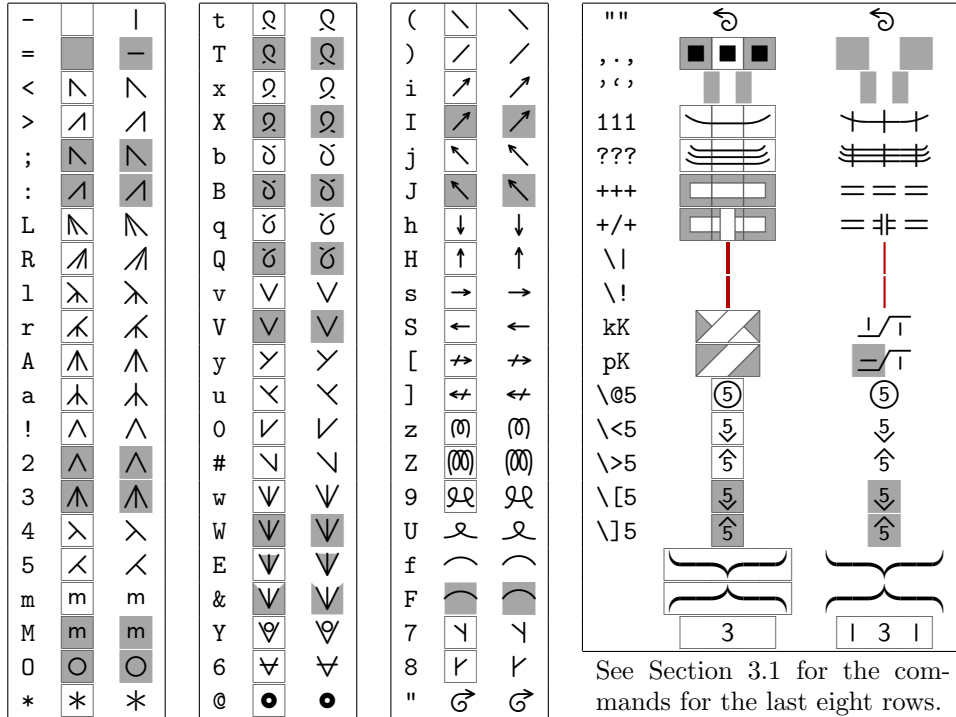
```
\documentclass{article}
\usepackage{knitting}
\begin{document}
Here is my chart:
\chart{
--0A0--
<0---0>
-<0-0>-
}
\end{document}
```

		○	∧	○		
∧	○				○	∧
	∧	○	○	∧		

Here is my chart:

Here is a minimal plain \TeX document that will do the same thing:

FIGURE 1. The normal symbols

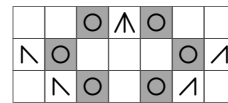


```

\input knitting
Here is my chart:
\chart{
--OAO--
<0---0>
-<0-0>-
}
\bye

```

Here is my chart:



To draw your own chart, you first need access to various knitting symbols. They are produced inside charts using normal letters: **A** produces \blacktriangleup , **O** produces \bigcirc , and so on. An almost-complete typing key is provided in Figure 1. The file `knitkey.tex` contains my suggested meanings for all the available symbols.

Ways to modify these symbols are described in Section 3.1.

Some commands to change the overall appearance of the charts or the document are given in Section 3.2.

Sometimes a part of a knitting chart (usually a repeat) should be outlined; the machinery to do so is described in Section 3.3.

Some miscellaneous commands, including control over the various grays and colors used in knitting charts, are described in Section 3.4. The behavior of spaces and carriage returns in the input is described in Section 3.5.

Cable symbols such as \boxtimes or \boxplus are complicated; they are described in Section 4.

Finally, `knitting` will count rows and stitches for you automatically; the machinery to do so is described in Section 5 and Section 6.

These sections describe the commands defined by `knitting`. Many of these commands (indicated with `*`s and not bullets) only work inside of a knitting chart, to avoid conflicts: `\overline`, for example, already has a meaning \overline{xy} in math mode.

The command `\chart` is fairly complicated, and as such, may be prone to strange behavior when used with other commands. An alternative command, `\textknit`, may be used inside of, for example, a tabular environment with less risk of unexpected behavior. This is useful for writing chart keys.

Most of the commands labeled with `*` do *not* work in concert with `\textknit`. Also, `\N` and `\V` must be accessed with `\textknit{\#}` and `\textknit{\&}`, not `\textknit{\#}` and `\textknit{\&}`. (Unadorned `#` and `&` may be used with `\chart`.)

3.1. Stitch symbols produced or modified with commands. You can generate new symbols with the command `\knitbox`: `a` `Cable 6 left`. A great many strange symbols are available by using math symbols:

`\knitbox{\$ \vspace{-1.5pt} \heartsuit $}{1}` `\heartsuit`.

However, these may look a little incongruous with the rest of the font, so use with care.

Here are some other commands that produce particular symbols:

Command	<code>\chart</code> -only shorthand	Symbol
<code>\wideincrease{3}</code>		
<code>\widedecrease{3}</code>		
<code>\bobble{3}</code>	<code>\@</code>	
<code>\narrowincrease{3}</code>	<code>\<</code>	
<code>\narrowdecrease{3}</code>	<code>\></code>	
<code>\pnarrowincrease{3}</code>	<code>\[</code>	
<code>\pnarrowdecrease{3}</code>	<code>\]</code>	
	<code>~</code>	Empty space

The arguments to `\wideincrease` and `\widedecrease` must be numbers; they are the width (in stitches). The arguments to `\bobble` and so on may be any text (although you will probably use numbers). The `\chart`-only shorthand may be used inside `\charts` but not inside `\textknit`.

The `~` should be used on the edges of non-rectangular charts:


====
 ~====
 ~~==
 ~~~=  
 ~~~=



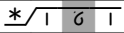



You can also use `~` inside a chart, although `'`, `'` or `.` may be better choices. (Do not use `~` on its own line; if you need a blank line in a cable chart, use `'`.)

Here are some commands that modify or rearrange existing symbols in odd ways:

- `\purlbackground`. I use a gray background to indicate simple purls, and also the purl version of more complicated stitches: `\N` for slip, slip, knit, and `\N` for purl 2 together. You can get `\N` instead of `\N` by typing 3

instead of A, but I didn't have space to provide a purl version of every symbol; so the only way to get  is with `\purlbackground{r}`.

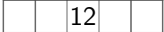
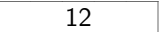


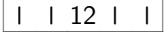
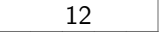
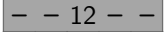
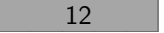
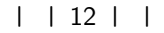
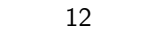


- `\widesymbol` places a symbol or sequence of symbols in a box of a given width: `\widesymbol{W}{2}` . In other words, `\widesymbol` behaves like `\knitbox`, but typesets its first argument using the knit symbol font rather than a Roman font.
- `\cableleft` and `\cableright` will produce the most general possible cable symbols. While there are simple methods to get  (see Section 4), these let you get such obscure symbols as  or even (in concert with `\bobble` and `\knitbox`) . `\cableforeground` and `\cablebackground` may be used for similar effects.

And here are some commands designed to deal with long sequences of knits and purls in a chart, although `\knitbox` can be used to generate other symbols:

- `\knit` and `\purl`. These macros take one argument each and type out that many plain knit or plain purl symbols.
- `\Knit`, `\Purl`, `\knitbox`, `\purlbox`. These macros were designed to produce appropriate shorthand for “Knit or purl 12 stitches, regardless of how many are actually shown”. The first argument is text to appear inside the box; the second is the desired width of the box (in units of one stitch).

They look like this:

```

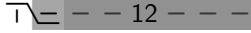
\knitgrid
\Knit{12}{5}  \knitbox{12}{5} 
\Purl{12}{5}  \purlbox{12}{5} 
\knitmixed
\Knit{12}{5}  \knitbox{12}{5} 
\Purl{12}{5}  \purlbox{12}{5} 
\knitnograd
\Knit{12}{5}  \knitbox{12}{5} 
\Purl{12}{5}  \purlbox{12}{5} 

```

Normally, the first argument is centered inside the box. However, you can offset the label with an optional first argument. This is the displacement to the right in half-stitches; to get a displacement to the left, use negative numbers.

If you want to make these characters more obvious, you can change the colors for the knit and purl boxes (and the generated `—`s and `|`s) by redefining the commands `\knitboxforeground`, `\purlboxforeground`, `\knitboxbackground`, `\purlboxbackground`; by default, they are blank and `\color{purlgray}` (or `\purlgray`).

```

\definecolor{purlboxbg}{gray}{0.57}
\definecolor{purlboxfg}{gray}{0.2}
\renewcommand\purlboxbackground{\color{purlboxbg}}
\renewcommand\purlboxforeground{\color{purlboxfg}}
\textknit{Kp\Purl[-1]{12}{6}} 

```

3.2. The four knitting fonts and other commands affecting overall appearance.

FIGURE 2. Suggested use of the fonts

| Font | Example | Suggested use |
|--------------------------|---------|--|
| <code>\knitgrid</code> | | Most simple patterns |
| <code>\knitwide</code> | | Colorwork |
| <code>\knitmixed</code> | | Patterns with complicated cables; patterns that use <code>\Knit</code> and <code>\Purl</code> to deal with altered stitch counts |
| <code>\knitnograd</code> | | Patterns that use empty squares (“no stitch” symbols) to deal with altered stitch counts |

- There are four available knitting fonts. They are selected with the commands `\knitgrid`, `\knitnograd`, `\knitwide`, `\knitmixed`. These commands should be used outside the chart they are to affect. See Figure 2 or the end of this section for a discussion of when each font should be used.
- In \LaTeX , you can change the sizes of the chart symbols with the usual commands `\small`, `\large`, etc.

In plain \TeX , use the command `\changeknitsize`, which takes one argument (the desired size). After `\changeknitsize{10pt}` (the default size), five lines of a knitting chart will take up as much vertical space as five lines of 10-point text. This means that chart cells are 12 (not 10) points tall.

- The package option `chartonly` (\LaTeX) or the command `\chartonly` (plain \TeX) causes charts to be typeset in small PDF files which can be easily included in other documents, or converted to image files for use on a webpage.

If you would also like to bundle your chart key into a little PDF, you can do it with the environment `smallpage` (\LaTeX) or `\smallpage` and `\endsmallpage` (plain \TeX).

The charts will automatically be the right width. The pages you generate with `smallpage` will be the natural width of their contents; this is usually `\textwidth` (\LaTeX) or `\hsize` (plain \TeX).

For some reason, `smallpage` won't work if your small page only has one line. Also, your PDF viewer may cut off a few pixels around the edges.

This command doesn't work with `dvi-TeX`; this is a `pdfTeX` command only.

- **fullpages.** Changing page dimensions mid-document in \LaTeX is hard, but a knitting pattern writer might want several pages of instructions with the usual large \LaTeX margins followed by several pages of charts with smaller margins. The environment `fullpages` does this. (Changing margins in plain \TeX is easy enough that `knitting.tex` has no special commands for it.)

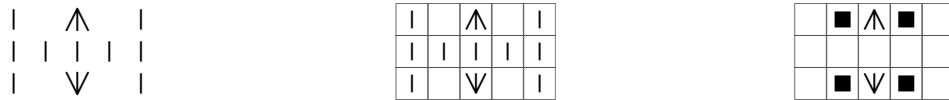
The `geometry` package now has the ability to change page dimensions mid-document, which affords much more flexibility than the `fullpages` environment; the `fullpages` environment is retained only for reasons of backwards compatibility.

The suggested uses of the four knitting fonts are summarized in Figure 2. I prefer to draw most charts with `\knitgrid`. However, there are some situations where each of the other four fonts is more useful.

The cells in the `\knitwide` fonts are the approximate dimensions of a knit stitch, so a colorwork chart done using the `\knitwide` font will look more like the finished piece than a colorwork chart done using `\knitgrid`. However, if a chart does not display wrong-side rows, the chart will not have the same proportions as the finished result. Also, lace is often blocked severely, and cables pull in. Thus, such charts will not have the same proportions as the finished piece, even if drawn with `\knitwide`; `\knitgrid` is probably a better choice.

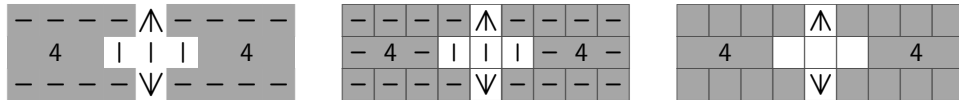
The `\knitgrid` cables look better than the `\knitnogrid` cables; however, wide cables, as well as cables involving increases, decreases, ribbing, and so on, can be expressed much more clearly with `\knitnogrid` or `\knitmixed`.

Finally, many knitting patterns exist that have more stitches in some rows than in others. This is usually expressed by using special “no stitch” symbols:



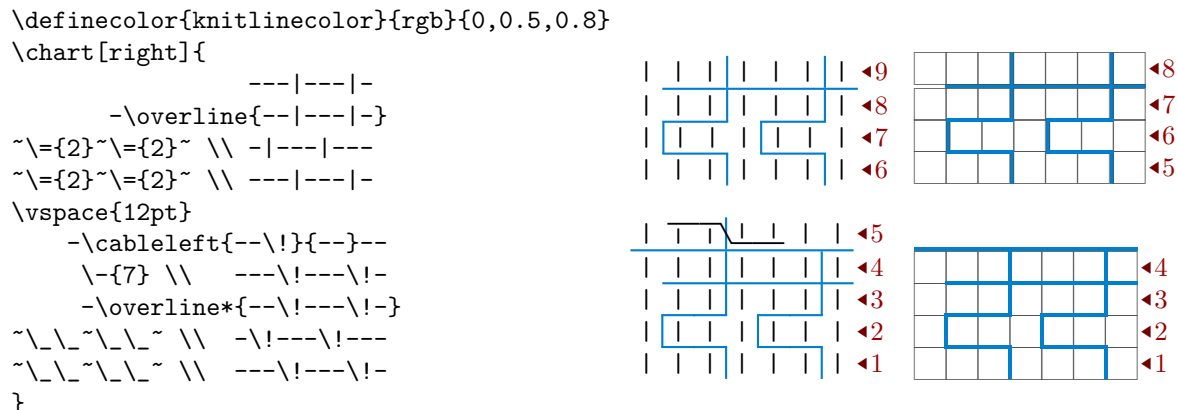
It is my feeling that the empty gaps of `\knitnogrid` are the clearest way to express this, and thus `\knitnogrid` should be preferred in charts that involve “no stitch” blocks.

However, in some patterns, we can omit knit or purl symbols to keep the chart the same width. This is best done using `\purlbox` or `\Purl`:



You may feel that such things look better if a grid is involved. (I do.) This is a way to avoid “no stitch” boxes with `\knitgrid` or `\knitmixed`; again, the presence of complicated cables probably determines which you should use.

FIGURE 3. Outline examples



3.3. Repeat outlines. As illustrated in Figure 3, there are ways to outline part of a knitting chart.

- * `\overline` and `\underline` take one argument and typeset it, then put a colored bar over or under it. This is designed to provide a way to outline vertical pattern repeats.

These commands have starred and unstarred forms. `\overline` spreads the chart rows to make room for the line; it should be used for lines that go all the way across the chart. `\overline*` does not (the lines overlap the chart a bit) and so should be used for lines that go only partway across the chart.

- * A line that goes across the chart can be specified with `\-`, which takes an argument (the width of the overlining in stitches). This has starred forms and unstarred forms, like `\overline`.
- * The character `|` and the command `\|` produce a vertical line suitable for outlining horizontal pattern repeats. The command `\!` produces a slightly different vertical line: like `\overline*`, it will overlap the adjacent cells to avoid disrupting the alignment of columns. Thus, `|` and `\|` should be used with `\overline`, while `\!` should be used with `\overline*`.
- * You may prefer to specify horizontal lines on their own, without interleaving them with the chart symbols using `\overline`. This may be done, but implementation is complicated.

You can get short horizontal lines to go with `\!` by using `_`; it should be positioned using tildes `~`.

If you want to use short horizontal lines between two `\|`s, you should use the command `\=`, which like `\-` takes an argument. (This will look wrong if you use it at the edge of a chart; it's better to use `\overline` or `\underline`.)

3.4. Colors and miscellaneous commands. `knitting` uses several colors and shades of gray to draw charts. You can customize these.


The \LaTeX package `knitting.sty` loads the `color` package and uses it to define colors; so the color of purl stitches is accessed via `\color{purlgray}`. If you want a different purl color, you can use `color`'s `\definecolor` syntax to change `{purlgray}`. This is especially useful if you want to write two-color colorwork charts.

`knitting.tex` defines `\purlgray` itself, using syntax that works for `pdfTeX` and `dvips`, but possibly not other drivers. If you insist on using plain `TeX` and another driver, you are assumed to know enough to edit `knitting.tex` to compensate. Redefining colors in plain `TeX` is complicated and driver-dependent; read `knitting.tex` itself to get an idea of the color definition syntax.

- `purlgray` is the color of the purl background.
- `gridcolor` is the color of the grid lines in the grid fonts.
- `knitlinecolor` controls the lines produced by `\!`, `|`, `\=`, `_`, `\-`, `\overline`, and `\underline`.
- `rncolor` is used by `\printrownumber` and `\printstitchcount` so that the row numbers and stitch counts cannot be mistaken for parts of the chart proper. The color `rnnarrowcolor` is used for just the small arrows in left and right row numbers. See Section 5 and Section 6.

There are also some commands you might want to use:

- `\purlpass`, `\gridpass`, `\mainpass`. `\chart` and `\textknit` compile their argument twice: once in gray, for the purl background, and then once in black for the foreground. They then put them on top of each other. (The grid fonts do a third pass, in the middle, for just the grid; this lets us have grid lines that are gray rather than black.) `\purlpass` takes one mandatory argument (something to do only during the purl pass) and one optional first argument (something to do during the other two passes.) These let you produce a variety of effects:

`\purlpass{\color{blue}}` pK 

- `\knitlinewidth`, `\gridwidth`, `\stitchwd`, `\stitchht`, and `\stitchdp` store most of the dimensional information about the knitting fonts.²

It is inadvisable to change any of these (except `\knitlinewidth`); a 0.3pt grid is built into the fonts, and changing `\gridwidth` won't change it, just mess up any code that relies on `\gridwidth`.

You can change `\knitlinewidth` with `\setlength`; however, `\knitlinewidth` is defined by `knitting` in a complicated way so as to change gracefully with changing knit sizes, and so it is probably best to say `\newdimen \knitlinewidth` (not `\newlength{\knitlinewidth}`) first.

If you use one of these parameters outside of a `\chart` or `\textknit`, you may get error messages about undefined fonts. To fix them, use `\knitgrid`, `\knitngrid` or `\knitwide` again.

- The boolean variables `\ifgrid`, `\ifknitsymbol` and `\ifchartsonly` are standard `TeX` conditionals: they may be used as

```
\ifgrid Grid code \else Nongrid code \fi
```

²Grid cells are designed to be 12pt (or 16.3pt) by 12pt, and extend slightly below the baseline. In \LaTeX , `\stitchht` is 12pt. In plain `TeX`, `\stitchht` is 12pt-`\stitchdp`. This is because the plain `TeX` `\raise` macro and the \LaTeX `\raisebox` macro work differently.

They are useful if you haven't decided yet how you want to format your document, or if you want to compile it several times with slightly different results. (`\ifgrid` distinguishes between `\knitngrid` and the other three options; `\ifknitsymbol` distinguishes between `\knitngrid/\knitmixed` and `\knitgrid/\knitwide`.)

- In \LaTeX , you get sans serif text (the font inside the knit boxes) and roman text (the row number/stitch count font) with the usual commands `\textsf` and `\textnormal`. In plain \TeX , you can get these fonts with `\knitsf` and `\knitrm`.

3.5. Spaces in the input. A normal space in the source code in a knitting chart, like a space in math mode, is ignored. In a knitting chart, unlike in normal text, you want to prescribe all your line breaks. For convenience, knitting charts have the macro `\obeylines` built in, so that a single `\par` produces a new paragraph (and therefore a new line), like a double `\par` usually does.

If you want two or more lines in the source code to produce one line on the chart, end all but the last with the comment character `%`.

Unfortunately, the trick I use to make `\par`s trigger new lines is delicate; specifically, if you put your chart inside another command, it stops working. (It works fine inside environments.) (Similarly, `|` and `#` are redefined inside knitting charts, so that `|` produces a colored bar `|` and `#` produces a symbol `⊞`. If your chart ends up inside a command, `|` will produce a plain black bar and `#` will produce an error; use `\|` and `\#` instead.)

So if you want to embed your chart inside another command, you have to end each line with a `\par`, `\|`, or a double `\par` (blank line).

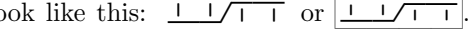
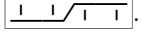
FIGURE 4. The effects of embedding charts in commands and environments

| | | | |
|--|--|---|--|
| <pre> \newdimen\knitlinewidth \setlength{\knitlinewidth}{4pt} \parbox{77pt}{ \chart{ tt AA\ % --\ == ==\ } </pre> | | <pre> \newdimen\knitlinewidth \setlength{\knitlinewidth}{4pt} \begin{center} \chart{ tt AA\ % --\ == ==\ } \end{center} </pre> | |
|--|--|---|--|

4. LIGATURES AND CABLES

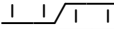


\TeX has a built-in ligature mechanism that lets it get `ıfire-fly?` instead of `?fire-fly?` when you type `?fire--fly?`. `knitting` uses this for wide horizontal sequences such as `⊞`. The ligature mechanism also lets you generate the reverse of `⊞`: `-`, `=` and `"` produce `⊞`, `⊞`, and `⊞`, respectively.

However, the ligature mechanism really comes into its own when making cable symbols.


4.1. **Symbolic cables.** In the fonts which represent “knit 1” by a vertical bar symbol (`\knitnograd/\knitmixed`), cables look like this:  or . The symbols are a stylized representation of what you must do to produce the cables.

The keys `k`, `p`, `K`, and `P` produce raised or lowered knit and purl symbols, and the ligature mechanism adds in the underbars or slant connectors automatically.

So:

| | |
|-----------------------|--|
| <code>kkKK</code> |  |
| <code>KKpp</code> |  |
| <code>kpkpKPKP</code> |  |

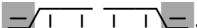
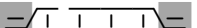
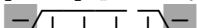
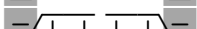
Some allowance for peculiar cables has been made. You can get a front increase or decrease with `N` or `D`, or a back increase or decrease with `n`, `d`, `e`, or `o`: `nedoND`
`m m N N m A`.

All of the symbols present in the font can be used in cables with the help of `\cableleft` and `\cableright`: `\cableleft{AOA}{==}`  `* -`.

If you have a lot of cables involving the same obscure symbol, you may wish to use the commands `\cableforeground` and `\cablebackground` instead:

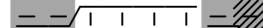
```
\renewcommand\${\cableforeground{b}}
\renewcommand\#{\cablebackground{b}}
\chart{
    -\$\$\#\#-          | ̈́ ̈́ ̈́ ̈́ |
    -\$\$\$kk-         | ̈́ ̈́ ̈́ ̈́ |
    \$\#--\#\}$}      ̈́ ̈́ | | ̈́ ̈́
```

Be aware that the implementation of `\cablebackground` is such that only cable connectors are drawn in the main pass; the symbols are drawn during the purl pass, and so `\cablebackground` interacts in unexpected ways with color effects and with `\purlpass`.

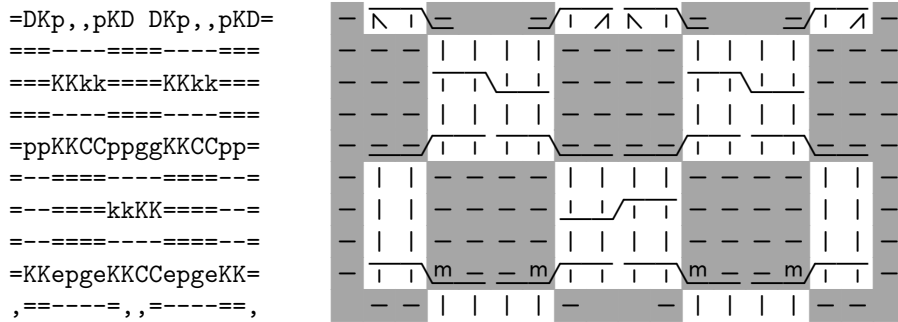
Explaining to knitting when one cable starts and another ends can be hard: `pKKKKp` probably means  but it could mean  or  and the ligature mechanism isn't smart enough to default to  let alone read your mind. So you have to tell it what you want.


There are two ways to do this. You can put a space in: `ppKK KKpp`. You can also use the characters `c`, `g`, `C`, and `G`: these behave just like `k`, `p`, `K`, and `P`, except that they are only allowed to show up in the left part of a cable.

Some limited debugging mechanisms are available. A sequence of background cables `kpk` which is not paired with a sequence of foreground stitches is not a cable; it is an orphaned half-cable. If for some reason you want an orphaned half-cable `- -`, use `\cableleft` or `\cableright` with an empty argument.

Orphaned half-cables produced by other methods (unaccompanied `pp`) are probably due to a typographical error which must be fixed. So they are made obvious. If you have an orphaned half-cable somewhere, the last square will be shaded: `ppKKKKpp` produces  since the terminal `ps` are not part of a full cable.

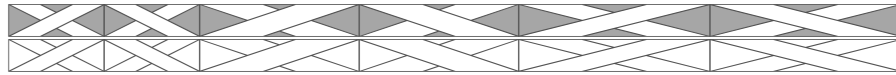
So:





4.2. **Stylized cables.** In the fonts which represent “knit 1” by an empty box (`\knitgrid/\knitwide`), cables look like this: . The symbols are a stylized sketch of how the cables actually look.

The letters `kpcgKPCG` work for simple cables as in the non-grid font. These cables are fairly limited. You can cross 1, 2, or 3 stitches over 1, 2, or 3 stitches, going left or right. The letters `kpcgKPCG` will let you draw purl-over-purl, knit-over-purl, or knit-over-knit cables.

Twelve special cable symbols are also possible:



You get these by putting `ps` (not `gs`) between the `ks` and `Ks`.³

In the wide-cell grid font, for technical reasons no symbols more than five cells wide are available; so while  is available, a wider version of  is not.

The keys `N`, `n`, `e`, `o`, `d` and `D` have a different function here.

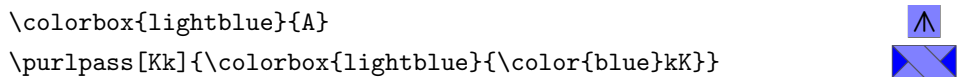
If a knit-over-knit cable ends with a `d` or `D` instead of `k` or `K`, the result will have a solid white background instead of a gray one. The letters `n`, `N`, `e` and `o` produce symbols that are hybrids of twist and cable symbols.⁴


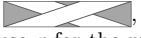




`e` and `o` have the same effect.

Again, orphaned half-cables are indicated with diagonal hash marks .

Some special effects are possible. You can get fancy with the colors of these symbols.⁵



³The rules for these cables are actually more complicated. The only way to get , the nogrid equivalent of , is by typing `kkpKK` or some equivalent with `cs` and `gs`. In the grid font, you can't use `g` for the middle purl stitch. More importantly, the fonts don't check to make sure you typed exactly `kkpKK`. The only grid cable that can start with `kkp` is ; so `kkp` followed by any number of `Ks` will produce .

⁴A `k`, `p`, `K`, or `P` after a `n`, `e`, `o`, `d`, `N`, or `D` starts a new cable; you don't need to use `C`, `G`, `c`, `g` or spaces to separate them.

⁵If you use `\colorbox` with `\textknit` instead of `\chart`, you will need to say `\setlength{\fboxsep}{0pt}` at some point to make this work; this statement is built into `\chart`.

You can also indicate unusual cables by superimposing other symbols:

```
\mainpass{\rlap{\knitbox{D}{4}}} kkKK
\mainpass{\rlap{\knitbox{m\vspace{2pt}}{1}}} ppKK
```



This is enough for most cable patterns.

5. COUNTING ROWS

This section describes how to get and modify the little red numbers shown here.

```
\chart{
  \rnleft =====
  \rnleft =====\addtocounter{rownumber}{-1}
  \rnleft =====
  \addtocounter{rownumber}{-5}}
\chart[right]{
  =====
  ===== \nonumber
  =====
  =====
  \numberrow{6}{2}{2}}
```

- ★ `\rn` prints out the value of the counter `rownumber`, then decreases it by the value of the counter `rownumberskip`. For better results, use `\rnleft` on the left edge and `\rnright` on the right edge.

If you want to skip a few row numbers, you can say `\addtocounter{rownumber}{-3}` (L^AT_EX) or `\global \advance \rownumber by -3` (plain T_EX).

`\chart` will usually automatically arrange things so that the last `\rn` produces a 1. If you want numbers in different charts to be numbered consecutively (e.g., if they are pieces of one big chart), you can turn this behavior off with `\resetrnfalse` and back on with `\resetrntrue`. You can then reset the row numbers with `\setcounter{rownumber}{20}`; all future charts will count down from there.

This may be necessary if you have very long charts, since charts do not break across pages. (You may need to put a `\par\nointerlineskip\par` between the pieces of charts.)

- `\chart` has an optional first argument that places row numbers automatically. It should be one of the seven words `left`, `right`, `odddleft`, `oddrightright`, `evenleft`, `evenright`, or `both`. This will automatically place numbers down the left edge, the right edge, or put the odd numbers on one side and the even numbers on the other side.

If you want to show only even or only odd numbers, you can do it with the commands `\rnoddonly` or `\rnevenonly`, and can restore normal behavior with the command `\rnnormal`. Alternatively, you can redefine `\printrightrownumber` to only print if the counter `rownumber` is odd; effects like this are why the `[both]` option exists.

I suggest using `\setcounter{rownumberskip}{2} \chart[right]` for charts which show only right-side rows. For charts which show all rows,

I suggest using `[right]` for charts which are meant to be worked in the round, `[oddright]` or `[oddleft]` for charts which are meant to be worked flat (back and forth), and `\rnevenonly` or `\rnoddonly` with `[right]` for charts which may be used either flat or in the round.

- ★ Inside an auto-numbered chart, `\nonumber` skips the next row number.
- ★ To number the stitches (by producing a *row* of stitch numbers), you can use the command `\numberrow`. It takes three arguments: the first number to be printed, the countdown (how often to print intermediate numbers), and the last number to be printed.
- ★ If you don't like any of my automatic countdown options, you can use `\rnbox{8}`, `\rnboxleft{12}`, `\rnboxright{3}` to do your own row-number boxes.
- Two commands exist to let you get \TeX to count stitches for you: `\stitchcountchart` and `\countstitches`. The command `\adjuststitchcount` is also useful here. See Section 6 for more details.

You can adjust the appearance of the row numbers in a few ways. `\rn` puts the number in a box of width `\rownumberwd`; you can change the width by saying `\renewcommand{\rownumberwd}{1em}`. You can change the appearance of the rownumbers by renewing the commands `\printrownumber`, `\printleftrownumber` and `\printrightrownumber`; the default values (in \LaTeX) are

```
{\color{rncolor}\textnormal{#1}},
{\knitleftarrowhead{\color{rncolor}\textnormal{#1}}} and
{\color{rncolor}\textnormal{#1}}\knitrightarrowhead}.
```

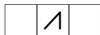
`\printrownumber` controls the appearance of both `\rn` and `\rnbox`. You can also change the appearance of `\rn`, but not `\rnbox`, by redefining `\therownumber`. `\roman{rownumber}` (\LaTeX) or `\romannumeral rownumber` (plain \TeX), for example, will number rows with Roman numerals.

6. COUNTING STITCHES

When knitting a complicated pattern, it can be helpful if the chart indicates the expected stitch count after each row. When writing a complicated chart, and especially when designing a new stitch pattern, it can be *very* helpful to have some way of checking to see that each row uses exactly as many stitches as the previous row generated. `knitting` provides a mechanism for counting stitches and for comparing them from row to row.

If used inside a knitting chart, the command `\countstitches` takes one argument and sets the counters `stitchcountout` and `stitchcountin` to reflect how many stitches that sequence of stitches would produce or consume, assuming that all the symbols have the meanings given in `knitkey.tex`.

So, for example:

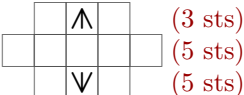
| | |
|--|--|
| <code>\chart{\countstitches{->-} ->-}</code> |  |
| <code>stitchcountout: \thestitchcountout</code> | stitchcountout: 3 |
| <code>stitchcountin: \thestitchcountin</code> | stitchcountin: 4 |

This can be used to automatically label each row with its stitch count.

```

\newcommand{\mystitchcount}[1]{\countstitches{#1}#1
  \mainpass{\color{rncolor}%
    \textnormal{(\thestitchcountout\ sts)}}}
\chart{
  \mystitchcount{~-A-~}
  \mystitchcount{-----}
  \mystitchcount{~-w-~}

```



(A more complicated example is in `knitexamples.tex`. Repeated patterns affect stitch count in strange ways; that example should show you how to cope.)

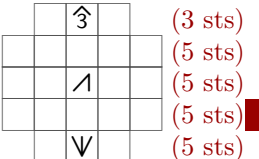
You can adjust the stitch count using the command `\adjuststitchcount`. It takes a mandatory argument (how many stitches to add) and an optional argument (a different number of stitches to add to the incount.)

By using the command `\stitchcountwarningbar`, we can check to see that stitch counts are consistent from row to row:

```

\newcommand{\mystitchcount}[1]{\countstitches{#1}#1%
  \textnormal{\color{rncolor}
    \mainpass{(\thestitchcountout\ sts)\stitchcountwarningbar}}
\chart{
  \setcounter{stitchcountin}{-100}
  \mystitchcount{~-A-~}
  \mystitchcount{-----}
  \mystitchcount{-->--}
  \mystitchcount{-----}
  \mystitchcount{~-w-~}

```



The heavy bar beside Row 2 indicates that Row 3 uses six stitches, but Row 2 produces only five. `\stitchcountwarningbar` is built using the same code as `TEX`'s usual overfull rules; thus, it will only appear (in `LATEX`) if the `draft` document option is enabled.

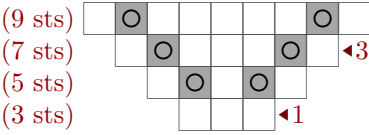
The initial `\setcounter{stitchcountin}{-100}` prevents a warning bar from showing up on the very first row. A `\setcounter{stitchcountin}{3}` would work in this case as well, since the top row produces 3 stitches.

All of this gives you a lot of control over the stitch-counting mechanism in your chart. However, it requires a lot of work to set up. `knitting` has a quick command for generating charts with stitch counts:

```

\rnoddonly
\stitchcountchart[right]{
  -0-----0-
  ~-0---0-
  ^^~0-0-
  ~~~----
}

```



You can customize this with the optional argument (which places row numbers as usual), the command `\knitdebug` (which shows the stitch counts both before and after knitting the row), and by redefining the commands `\printleftstitchcount` and `\printrightstitchcount`. The default (`LATEX`) values are


```

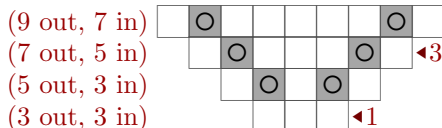
\newcommand{\printleftstitchcount}{\color{rncolor}%
  \textnormal{(\thestitchcountout\ sts) }}
\newcommand{\printrightstitchcount}{}
\knitdebug redefines \printleftstitchcount to
\renewcommand \printleftstitchcount{\color{rncolor}%
  \textnormal{(\thestitchcountout\ out,
  \thestitchcountin\ in) }}

```

```

\rnoddonly
\knitdebug
\stitchcountchart[right]{
  -0-----0-
  ~-0---0-
  ^^~0-0-
  ~~~---
}

```



(9 out, 7 in)
(7 out, 5 in)
(5 out, 3 in)
(3 out, 3 in)

The commands `\narrowincrease` and so on, when used with the stitch-counting mechanism, do require that their arguments be numbers: if your chart involves the symbol \boxtimes , that symbol cannot be counted.

The commands `\Knit`, `\Purl`, `\knitbox` and `\purlbox` by default are assumed to be as many stitches wide as they take up stitches on the chart: so `\knitbox{A}{2}` is assumed to be two stitches wide.

This means that `\Knit{4}{3}` is assumed to be three, not four, stitches wide. You can change this behavior by redefining the commands `\Knitstitchcount`, `\Purlstitchcount`, `\knitboxstitchcount`, `\purlboxstitchcount`:

```
\renewcommand{\Knitstitchcount}[2]{#1}
```

Redefining `\Knitstitchcount` does mean that you must give `\Knit` a number as its first argument.

Similarly, `\widesymbol` normally is assumed to be as many stitches as its contents: \boxminus represents one in-stitch and three out-stitches, just like \boxplus . If you prefer for \boxtimes to represent three stitches, redefine `widesymbolspacer`:

```
\renewcommand{\widesymbolspacer}[2]{\hspace #2\stitchwd}
```

The arguments of `\widesymbolspacer` are the two arguments of `\widesymbol`; the default value is just

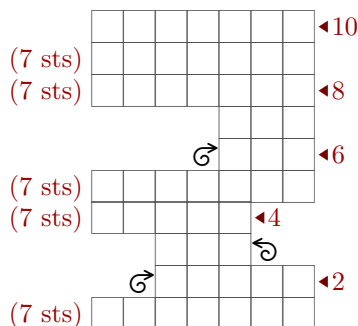
```
\newcommand{\widesymbolspacer}[2]{#1}.
```

There is a way to deal with a series of short rows: put the rows inside the command `\shortrows`, or use the command `\nostitchcount` to omit a stitch count.

```

\rnevenonly
\stitchcountchart[right]{
  \nostitchcount-----
      -----
      \shortrows{~~~~---
          ~~~"----}
      -----
      \shortrows{-----
          ~~~---"
          ~"-----}
          -----
}

```



Warnings. The stitch-counting machinery cannot, of course, actually know what meanings you assign to symbols. It is designed for the meanings in `knitkey.tex`. If you deviate from these, it will get the wrong answer.

Also, `\stitchcountchart` is delicate; it must not be used inside commands, the closing `}` must be on its own line, and it often has trouble if strange things are put at the start of a line. (You can fix this by starting the line with a `\noindent` or `\leavevmode` or `\mbox{}`.)

7. REVISION HISTORY

August 2010. I've added the symbols `\`, `/`, `∇`, `∨`, `∩`, and `⊔` to the fonts for the benefit of people who want to use the symbols suggested by the Craft Yarn Council of America, and updated `knitkey.tex` to include these symbols (and a few others that can be generated by judicious use of `\knitbox` and `\purlpass`).

I've added the commands `\printrightrownumber`, `\printleftrownumber` and `\printrownumber` to make row numbers easier to customize. I've redefined the left and right row number macros to include little arrows.

I've added the starred forms of `\overline` and `\underline` and the optional argument to `\Knit`, `\Purl`, `\knitbox` and `\purlbox` to offset the labels, and have rewritten those commands to use `\knitboxbackground` and `\purlboxbackground` in order to make them easier to customize.

I've added the stitch-counting mechanism. This has entailed minor revisions to a number of existing commands, writing the stitch-counting macros themselves, and also creating the supporting fonts `knitn_sc_in`, `knitn_sc_out`, `knitg_sc_in`, `knitg_sc_out`, `knitw_sc_in`, and `knitw_sc_out`.

April 5, 2019. The fonts `knitw_sc_in` and `knitw_sc_out` have been removed, because I realized that `knitg_sc_in` and `knitg_sc_out` could be used instead.

The commands `\|` and `\!` now work with `\textknit`.

The following have been added:

- The characters `∩`, `∇` and `∨`.
- The commands `\Knitstitchcount`, `\Purlstitchcount`, `\knitboxstitchcount`, `\purlboxstitchcount`;

```
\cableforeground, \cablebackground;
\widesymbol.
```

- The \knitmixed fonts.
- Support for counting stitches in short rows.
- Hash marks to show errors in cables.

8. LIST OF FILES THAT ARE CONSIDERED PART OF THIS PACKAGE

This package should have come with all the following files, organized into the directories listed.

README

knitting/docs

- knitexamples.pdf
- knitexamples.tex
- knitkey.pdf
- knitkey.tex
- knitting-doc.pdf
- knitting-doc.tex

knitting/fonts/afm

- knitg_sc_in.afm
- knitg_sc_out.afm
- knitgg.afm
- knitgn.afm
- knitgp.afm
- knitmg.afm
- knitml.afm
- knitmn.afm
- knitmp.afm
- knitmr.afm
- knitn_sc_in.afm
- knitn_sc_out.afm
- knitnl.afm
- knitnn.afm
- knitnp.afm
- knitnr.afm
- knitwg.afm
- knitwn.afm
- knitwp.afm

knitting/fonts/map

- knitfont.map

knitting/fonts/source

- knit_grid_cables.mf
- knit_nogrid_cables.mf
- knit_symbols.mf
- knitg_sc_in.mf
- knitg_sc_out.mf
- knitgg.mf
- knitgn.mf

- knitgp.mf
- knitmg.mf
- knitml.mf
- knitmn.mf
- knitmp.mf
- knitmr.mf
- knitn_sc_in.mf
- knitn_sc_out.mf
- knitnl.mf
- knitnn.mf
- knitnp.mf
- knitnr.mf
- knitwg.mf
- knitwn.mf
- knitwp.mf

knitting/fonts/tfm

- knitg_sc_in.tfm
- knitg_sc_out.tfm
- knitgg.tfm
- knitgn.tfm
- knitgp.tfm
- knitmg.tfm
- knitml.tfm
- knitmn.tfm
- knitmp.tfm
- knitmr.tfm
- knitn_sc_in.tfm
- knitn_sc_out.tfm
- knitnl.tfm
- knitnn.tfm
- knitnp.tfm
- knitnr.tfm
- knitwg.tfm
- knitwn.tfm
- knitwp.tfm

knitting/fonts/type1

- knitg_sc_in.pfb
- knitg_sc_out.pfb
- knitgg.pfb
- knitgn.pfb
- knitgp.pfb
- knitmg.pfb
- knitml.pfb
- knitmn.pfb
- knitmp.pfb
- knitmr.pfb
- knitn_sc_in.pfb
- knitn_sc_out.pfb

- knitnl.pfb
 - knitnn.pfb
 - knitnp.pfb
 - knitnr.pfb
 - knitwg.pfb
 - knitwn.pfb
 - knitwp.pfb
- knitting/tex/latex
- knitting.sty
 - uknit.fd
- knitting/tex/plain
- knitting.tex