

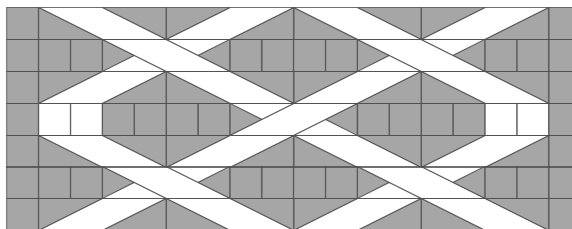
# THE KNITTING PACKAGE

ARIEL BARTON

This package was written to make knitting charts using  $\text{\LaTeX}$  or plain  $\text{\TeX}$ . 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 5. It is copyright Ariel Barton, 2010.

This work may be distributed and/or modified under the conditions of the  $\text{\LaTeX}$  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  $\text{\LaTeX}$  version 2003/06/01 or later.

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

The  $\text{\LaTeX}$  project license above gives the conditions under which you may redistribute or modify the fonts and files listed in Section 5. This license (loosely speaking) allows you to pass around copies of the package provided you redistribute it in its entirety. In addition, any part, no matter how large, 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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  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](mailto:origamist@gmail.com).

## 2. INSTALLATION

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

Most modern  $\TeX$  distributions have a folder, usually named `texmf`, where you can store supporting files for the packages you add yourself.<sup>1</sup> 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`

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}
```

or the lines

```
\pdfmapline{+knitgn\space <knitgn.pfb>
\pdfmapline{+knitwn\space <knitwn.pfb>
\pdfmapline{+knitnn\space <knitnn.pfb>
\pdfmapline{+knitgp\space <knitgp.pfb>
\pdfmapline{+knitwp\space <knitwp.pfb>
```

---

<sup>1</sup>If you're using  $\text{Mac}\TeX$ , this folder should be `Users/username/Library/texmf`. If it isn't there, create it.

If you're using  $\text{MiK}\TeX$ , 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  $\text{MiK}\TeX$ .

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

```

\pdfmapline{+knitnp\space <knitnp.pfb>
\pdfmapline{+knitnl\space <knitnl.pfb>
\pdfmapline{+knitnr\space <knitnr.pfb>
\pdfmapline{+knitgg\space <knitgg.pfb>
\pdfmapline{+knitwg\space <knitwg.pfb>

```

These map lines can go in `t1knit.fd`, `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. Also, the `\pdfmapline` command is part of pdfTeX, and has been since 2004 (and `\pdfmapfile` is even older); any distribution of TeX installed or updated in the last five years will be able to deal with the exact syntax above.


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

With MiKTeX, 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.

### 3. USING THE PACKAGE

In your document, type `\usepackage{knitting}` (L<sup>A</sup>T<sub>E</sub>X) or `\input knitting` (plain TeX). This will define the following macros:

- `\chart`, the command that draws a chart.
- `\textknit`. This command is meant to be used for writing chart keys. It typesets its argument using knitting symbols and puts it in an unbreakable box, which may appear in a table or even in the middle  of a paragraph.

Inside a chart, you mostly just type letters and punctuation, and `knitting` converts them to appropriate knitting symbols. See Section 4.3 for a translation key.

However, there are commands that change overall appearance of the charts, or produce fancier symbols. 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.

These redefined commands do *not* work in concert with `\textknit`.

- `\knitgrid`, `\knitnogrid`, `\knitwide`. These macros let you switch which font you're using: the normal grid font, the normal nogrid font, or the grid font with rectangular (wide) grid cells. These commands should be used outside the chart they are to affect.
- In L<sup>A</sup>T<sub>E</sub>X, 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 `[chartsonly]` (L<sup>A</sup>T<sub>E</sub>X) or the command `\chartsonly` (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}` (L<sup>A</sup>T<sub>E</sub>X) or `\smallpage` and `\endsmallpage` (plain T<sub>E</sub>X).

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` (L<sup>A</sup>T<sub>E</sub>X) or `\hsize` (plain T<sub>E</sub>X).

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-T<sub>E</sub>X; this is a pdfT<sub>E</sub>X command only.

- `fullpages`. Changing page dimensions mid-document in L<sup>A</sup>T<sub>E</sub>X is hard, but a knitting pattern writer might want several pages of instructions with classically large L<sup>A</sup>T<sub>E</sub>X margins followed by several pages of charts with smaller margins. The environment `fullpages` does this. (Changing margins in plain T<sub>E</sub>X is easy enough that `knitting.tex` has no special commands for it.)
- ★ `\rn` prints out the value of the counter `rownumber`, then decreases it by the value of the counter `rownumberskip`.<sup>2</sup> For slightly better-looking 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<sup>A</sup>T<sub>E</sub>X) or `\global \advance \rownumber by -3` (plain T<sub>E</sub>X).

`\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 six words `left`, `right`, `odddleft`, `oddrightright`, `evenleft`, or `evenright`. 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`.

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, `[oddrightright]` or `[odddleft]` for charts which are meant to be worked

---

<sup>2</sup>`\rn` puts the number in a box of width `\rownumberwd`. You can change the width by saying `\renewcommand{\rownumberwd}{1em}`.

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.
- `\purlbackground`. I use a gray background to indicate simple purls, and also the purl version of more complicated stitches:  for slip, slip, knit, and  for purl 2 together. You can get  instead of  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}`.
- ★ `\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 pattern repeats.
- ★ The character | and the command `\|` produce a vertical line suitable for outlining pattern repeats.
- ★ Occasionally, you may need to outline a pattern repeat with lines that bend. The command `\!` produces a vertical line which will overlap the adjacent cells to avoid disrupting the alignment of columns. 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 with just |, then you should use the command `\=`, which takes an argument (the width of the overlining in stitches). You should have one instance of `\=` per column of |s. It is strongly recommended that you not mix short overlines with |s in the grid font; it will look odd.

If you want your horizontal lines to reach the edge of the chart, you should use `\-`, which like `\=` takes an argument.

- `\purlpass`, `\gridpass`, `\mainpass`. `\chart` and `\textknit` compile their argument twice: once in gray, using the purl background font, and then once in black using the foreground font. They then put them on top of each other. (The grid font does 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.)
- `\color{purlgray}` (L<sup>A</sup>T<sub>E</sub>X) or `\purlgray` (plain T<sub>E</sub>X) are used by `\chart`, `\textknit` and `\purlbackground` to change the color to gray.

The L<sup>A</sup>T<sub>E</sub>X package `knitting.sty` loads the `color` package and uses it to define the color. 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 pdfT<sub>E</sub>X 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.

- The colors `knitlinecolor` and `gridcolor` are defined and may be changed similarly. `knitlinecolor` controls the lines produced by `\!`, `|`, `\=`, `\_`, `\-`, `\overline`, and `\underline`; `gridcolor` controls the grid.
- `\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”. They also provide ways to get lots and lots of bizarre symbols if necessary. The first argument is text to appear inside the box; the second is the desired width of the box (in units of one stitch).
- `\wideincrease` and `\widedecrease`. These macros take one argument each (the width, in stitches).
- `\bobble`, `\narrowincrease`, `\narrowdecrease`, `\pnarrowincrease`, and `\pnarrowdecrease`. These macros take one argument each; that argument is typeset in small letters over some symbol I thought was appropriate ⑤.
- ★ Inside a knitting chart (but not after `\textknit`), the shorter commands `\@`, `\<`, `\>`, `\[`, and `\]` are available.
- `\cableleft` and `\cableright` will produce the most general possible cable symbols. While there are simple methods to get  $\frac{1}{1} \frac{1}{1} \frac{1}{1} \frac{1}{1}$  (see Section 4.4), these let you get such obscure symbols as  $\frac{*}{1} \frac{6}{1} \frac{1}{1}$  or even (in concert with `\bobble` and `\knitbox`)  $\frac{3}{\text{knit}} \frac{\text{purl}}{3}$ .
- `\knitlinewd`, `\gridwidth`, `\stitchwd`, `\stitchht`, and `\stitchdp` store most of the dimensional information about the knitting fonts.<sup>3</sup>

It is inadvisable to change any of these (except `\knitlinewd`); a 0.4pt 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 `\knitlinewd` with `\setlength`; however, `\knitlinewd` 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 \knitlinewd` (not `\newlength{\knitlinewd}`) 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` and `\ifchartsonly` are standard  $\TeX$  conditionals: they may be used as

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

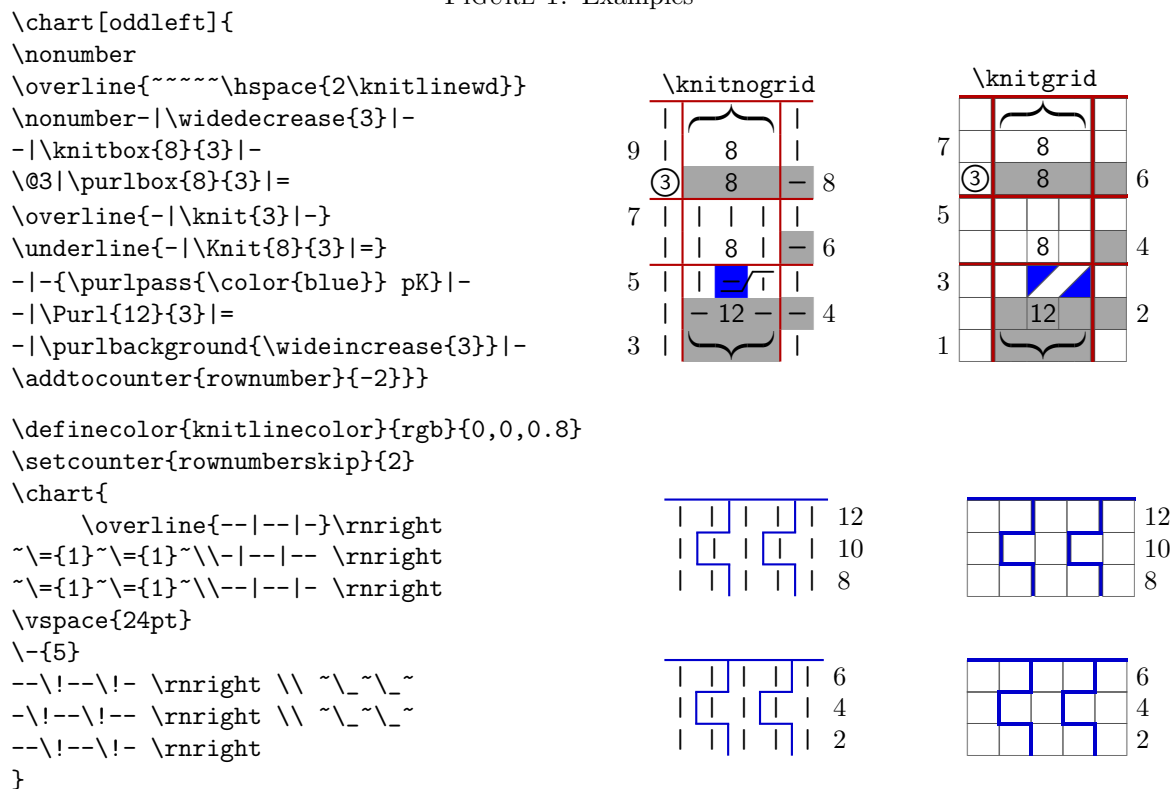
They are useful if you haven’t decided yet how you want to format your document, or if you want to compile it several times with slightly different results.

- In  $\LaTeX$ , you get sans serif text (the font inside the knit boxes) with the usual command `\textsf`. In plain  $\TeX$ , you can get sans serif text with `\knitsf`.

---

<sup>3</sup>Grid cells are designed to be 12pt (or 16.3pt) by 12pt, and extend slightly below the baseline to work gracefully with numbers or other normal text. 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.

FIGURE 1. Examples



#### 4. WHAT GOES IN THE CHARTS

After careful consideration, I decided to depart from  $\TeX$  tradition and set the fonts up so that your input would also look more or less like a chart (rather than defining new commands like `\ssk`).

So here's what you type:

**4.1. Space in the input.** A normal space in the source code in a knit grid, 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 `\return` produces a new paragraph (and therefore a new line), like a double `\return` 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 `\return`s trigger new lines is delicate; specifically, if you put your chart inside another command, it stops working. (It works fine inside environments.) (The bar `|` also stops working; it produces a plain black bar whose width will not change if you redefine `\kmitlinewidth`.)

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

FIGURE 2. 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>	
---	--	--	--

**4.2. Space in the output.** A tilde  $\sim$  produces a gap the width of one stitch. This is meant to be used in charts with ragged edges. It's been designed so that automatically placed rownumbers show up after the gaps left by  $\sim$ , not before. If you really want them to show up earlier, you can use single quotes instead of  $\sim$ s as spacers.

The single quotes ' and ' produce spaces that are half the width of a normal box (white or gray). If for some reason you need an empty, borderless box in the grid font, you can use '' or '''. It is inadvisable to use  $\sim$ s alone on their own line; ''s are much better behaved.

In the nongrid font,  $\square\cdot$  and  $\square,$  both produce empty boxes. (Of different colors;  $\square\cdot$  produces white,  $\square,$  produces gray.) These are meant to be no-stitch markers which are as unobtrusive and non-misleading as possible, and are in fact why the nongrid font has no grid.

**4.3. Normal symbols.** The normal symbols are in Figure 3. The file `knitkey.tex` contains my suggested meanings for all the available symbols.

If these symbols (and the cable symbols in the next section) do not suffice, then you can generate new symbols with `\knitbox:`  $\square a$   $\square$  Cable 6 left  $\square$ . This is the best way to indicate very wide cables in the grid font, since very wide cable symbols do not exist in that font, and if they did, they would slant too steeply to be easily read. A great many strange symbols are available by using math symbols: `\knitbox{\$ \vspace{-1.5pt} \heartsuit $\} {1}`  $\square\heartsuit$ . However, these may look a little incongruous with the rest of the font, so use with care.

**4.4. Ligatures and cables.**  $\TeX$  has a built-in ligature mechanism that lets it get ¿fire-fly? instead of ?'fire--fly? when you type ?'fire--fly?. `knitting` uses this for wide horizontal sequences such as  $\square\equiv\equiv\equiv$ . The ligature mechanism also lets you generate the reverse of  $\mathcal{G}$ : -, =" and "" produce  $\square\mathcal{G}$ ,  $\square\mathcal{G}$ , and  $\mathcal{G}$ , respectively.

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

In the non-grid font, cables look like this:  $\square\lrcorner\square$ . 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.

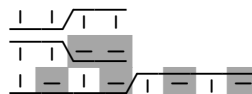
So:



FIGURE 3. The normal symbols

-			t	Q	Q	(	\	\	"	↺	↺
=	■	—	T	Q	Q	)	/	/	""	↻	↻
<	↖	↖	x	Q	Q	i	↗	↗	*	*	*
>	↗	↗	X	Q	Q	I	↗	↗	\@5	⑤	⑤
;	↖	↖	b	ö	ö	j	↖	↖	\<5	5	5
:	↗	↗	B	ö	ö	J	↖	↖	\>5	5	5
L	↖	↖	q	ö	ö	h	↓	↓	\[5	5	5
R	↗	↗	Q	ö	ö	H	↑	↑	\]5	5	5
l	↗	↗	v	V	V	s	→	→	,, ,	■	■
r	↖	↖	V	V	V	S	←	←	,, ,	■	■
A	↖	↖	y	Y	Y	[	↗	↗	111	⌢	⌢
a	↖	↖	u	Y	Y	]	↖	↖	???	⌢	⌢
!	^	^	4	Y	Y	z	⊙	⊙	+++	⌢	⌢
2	^	^	5	Y	Y	Z	⊙	⊙	+/+	⌢	⌢
3	^	^	w	V	V	U	⊙	⊙			
m	m	m	W	V	V	f	⊙	⊙	\!		
M	m	m	E	V	V	F	⊙	⊙		⌢	⌢
0	○	○	Y	V	V	@	●	●		⌢	⌢

kkKK  
 KKpp  
 kpkpKPKP

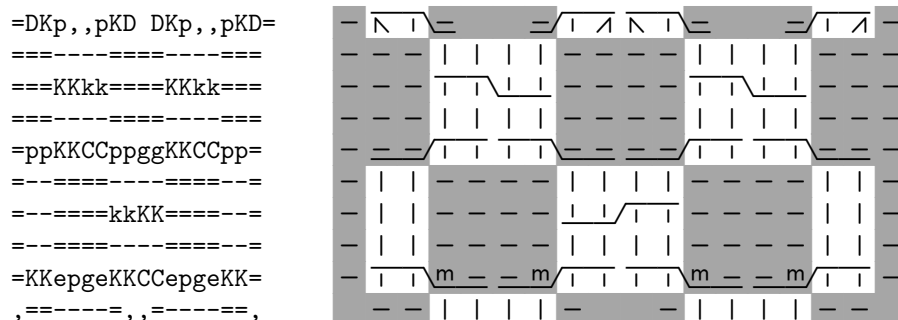


Some allowance for peculiar cables has been made. You can get a front increase or decrease with `N` or `D`, or a back increase or decrease with `n`, `d`, `e`, or `o`: `nedoND m m N N/m A`. All of the symbols present in the font can be used in cables with the help of `\cableleft` and `\cableright`: `\cableleft{AOA}{*} \A O A \_ * _`

Explaining to knitting when one cable starts and another ends can be hard: `pKKKp` 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. (Or you'll get `—| | | |—`, which is not 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.

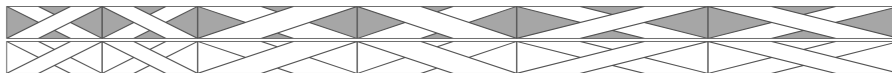
So:





In the grid fonts, cables look like this: . The letters `kpcgKPCG` work for simple cables as in the non-grid font.

Grid 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/Ds`.<sup>4</sup>

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.<sup>5</sup>





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





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

`\setlength{\fboxsep}{0pt}`   
`\textknit{\purlpass[Kk]{\colorbox{lightblue}{\color{blue}{kK}}}`

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.

<sup>4</sup>The 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 .

<sup>5</sup>A `k`, `p`, `K`, `pr` `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.

## 5. 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.

knitting/docs

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

knitting/fonts/afm

- knitgg.afm
- knitgn.afm
- knitgp.afm
- knitnl.afm
- knitnn.afm
- knitnp.afm
- knitnr.afm
- knitwg.afm
- knitwn.afm
- knitwp.afm

knitting/fonts/map

- knitfont.map

knitting/fonts/source

- knit\_dimens.mf
- knit\_grid\_cables.mf
- knit\_nogrid\_cables.mf
- knit\_symbols.mf
- knitgg.mf
- knitgn.mf
- knitgp.mf
- knitnl.mf
- knitnn.mf
- knitnp.mf
- knitnr.mf
- knitwg.mf
- knitwn.mf
- knitwp.mf

knitting/fonts/tfm

- knitgg.tfm
- knitgn.tfm
- knitgp.tfm
- knitnl.tfm
- knitnn.tfm
- knitnp.tfm
- knitnr.tfm
- knitwg.tfm
- knitwn.tfm

- knitwp.tfm

knitting/fonts/type1

- knitgg.pfb
- knitgn.pfb
- knitgp.pfb
- knitnl.pfb
- knitnn.pfb
- knitnp.pfb
- knitnr.pfb
- knitwg.pfb
- knitwn.pfb
- knitwp.pfb

knitting/tex/latex

- knitting.sty
- tlknit.fd

knitting/tex/plain

- knitting.tex