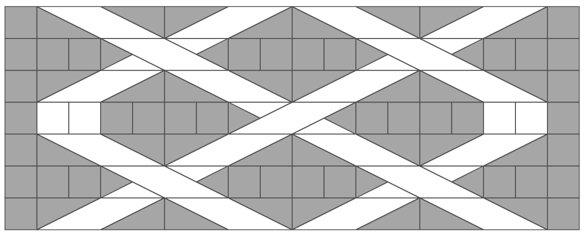
THE KNITTING PACKAGE

ARIEL BARTON

This package was written to make knitting charts using LaTeX or plain 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

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 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.[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`

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 pdfTEX produces) don't know about the fonts themselves.

There's a simple way to tell pdfTEX 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}
```

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

If you're using MiKTEX, 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 MiKTeX.

Any time you add supporting files to a local MiKTeX 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 `uknit.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`.

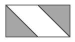
With MacTeX, I can cause `dvips`, `dvipdfm`, and pdfTeX to know about these fonts by opening a Terminal window (command prompt) and typing `updmap --enable Map=knitfont.map`.

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.

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

In your document, type `\usepackage{knitting}` (LaTeX) 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 4.1 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 $\star$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`.

3.1. **Commands affecting overall appearance.**

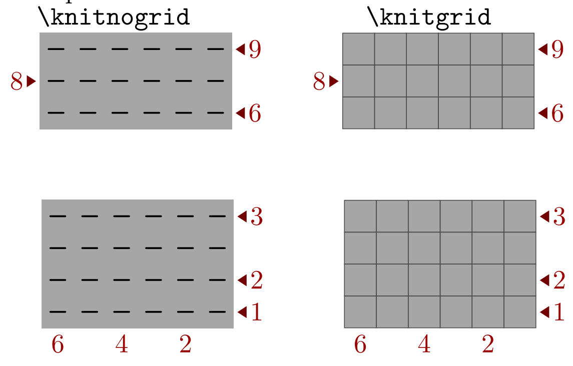
- `\knitgrid`, `\knitnogrid`, `\knitwide`. These macros let you switch which font you're using: the normal grid font, the normal nogrid font, or the grid font with rectangular (wide) grid cells. These commands should be used outside the chart they are to affect.
- In LaTeX, you can change the sizes of 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),

FIGURE 1. Counting examples

```
\chart{
         ======\rnright
  \rnleft======\addtocounter{rownumber}{-1}
         ======\rnright
  \addtocounter{rownumber}{-5}}
```

```
\chart[right]{
  ======
  ====== \nonumber
  ======
  ======
  \numberrow{6}{2}{2}}
```



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` (LaTeX) 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` (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.)

3.2. **Counting stitches and rows.**

⋆ `\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}` (LaTeX) or `\global \advance \rownumber by -3` (plain TeX).

`\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`, `oddleft`, `oddright`, `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 5 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

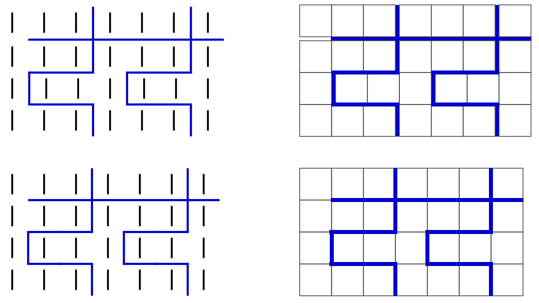
`{{\color{rncolor}\textnormal{#1}}}`,
`{\knitleftarrowhead{\color{rncolor}\textnormal{#1}}}` and
`{{\color{rncolor}\textnormal{#1}}\knitrightarrowhead}`.

`\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.

3.3. **Repeat outlines.**

Figure 2. Outline examples

```
\definecolor{knitlinecolor}{rgb}{0,0,0.8}
\setcounter{rownumberskip}{2}
\chart{
                 ---|---|-
       -\overline{--|---|-}
~\={2}~\={2}~ \\ -|---|---
~\={2}~\={2}~ \\ ---|---|-
\vspace{12pt}
                ---\!---\!-
    -\overline*{--\!---\!-}
~\_\_~\_\_~ \\ -\!---\!---
~\_\_~\_\_~ \\ ---\!---\!-
}
```



⋆ `\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.

⋆ 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 with `\|` or `|`, 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.

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

3.4. **Colors.**

- `\color{purlgray}` (LaTeX) or `\purlgray` (plain TeX) are used by `\chart`, `\textknit` and `\purlbackground` to change the color to gray.

  The LaTeX 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.

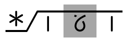
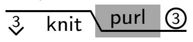
  `knitting.tex` defines `\purlgray` itself, using syntax that works for pdfTeX and `dvips`, but possibly not other drivers; if you insist on using plain TEX and another driver, you are assumed to know enough to edit `knitting.tex` to compensate.

- 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.
- Finally, the color `rncolor` is used by `\printrownumber` (and `\print-stitchcount`) so that the row numbers and stitch counts cannot be mistaken for parts of the chart proper. (The color `rnarrowcolor` is used for just the small arrows in left and right row numbers.)

3.5. Stitch symbols produced or modified with commands.

- `\wideincrease` and `\widedecrease`. These macros take one argument each (the width, in stitches) and produce wide symbols:
  `\textknit{\wideincrease{3}}`
- `\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:
  `\textknit{\bobble{5}}` 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 (see Section 4.4), these let you get such obscure symbols as  or even (in concert with `\bobble` and `\knitbox`) .
- `\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}`.
- `\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).

  They look like this:

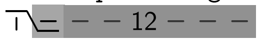
`\Knit{12}{5}`	☐ ☐ 12 ☐ ☐	`\knitbox{12}{5}`	12
`\Purl{12}{5}`	▨ ▨ 12 ▨ ▨	`\purlbox{12}{5}`	12
`\Knit{12}{5}`	| | 12 | |	`\knitbox{12}{5}`	12
`\Purl{12}{5}`	– – 12 – –	`\purlbox{12}{5}`	12

  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`).

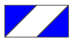
```
\definecolor{purlboxbg}{gray}{0.57}
\definecolor{purlboxfg}{gray}{0.2}
\renewcommand\purlboxbackground{\color{purlboxbg}}
\renewcommand\purlboxforeground{\color{purlboxfg}}
\textknit{Kp\Purl[-1]{12}{6}}
```



3.6. Miscellaneous commands.

- `\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 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.) These let you produce a variety of effects:

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

  
- `\knitlinewd`, `\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 `\knitlinewd`); 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 `\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.

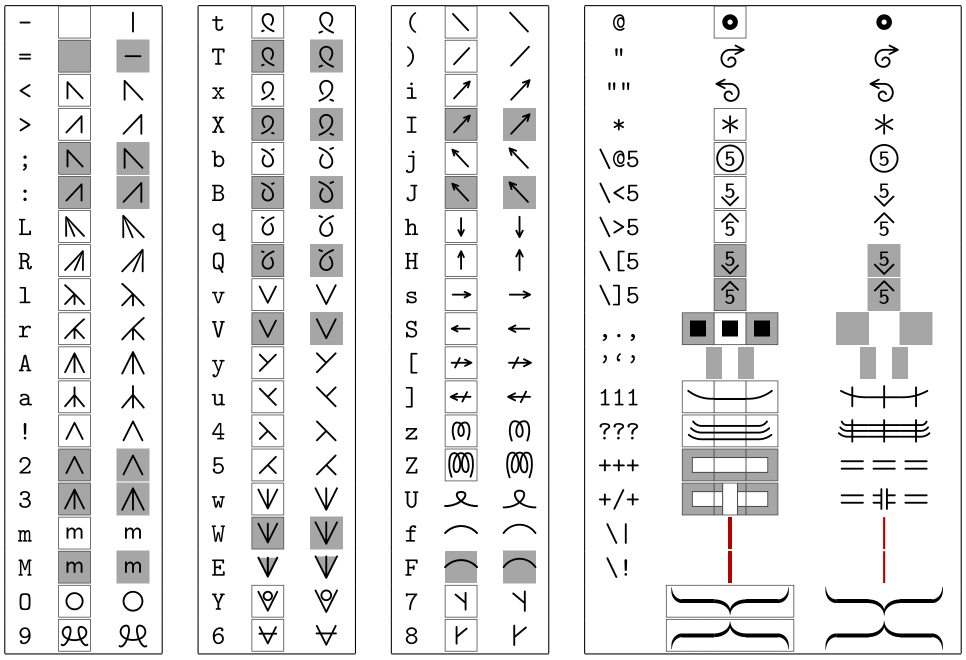
  If you use one of these parameters outside of a `\chart` or `\textknit`, you may get error messages about undefined fonts. To fix them, use `\knitgrid`, `\knitnogrid` or `\knitwide` again.
- The boolean variables `\ifgrid` and `\ifchartsonly` are standard TeX conditionals: they may be used as

  `\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) 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`.

[2]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 3. The normal symbols


- = < > ; : L R l r A a ! 2 3 m M O 9
t T x X b B q Q v V y u 4 5 w W E Y 6
( ) i I j J h H s S [ ] z Z U f F 7 8
@ " "" * \@5 \<5 \>5 \[5 \]5 ,., ’‘’ 111 ??? +++ +/+ \| \!

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`).

4.1. 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 Section 4.4) do not suffice, then you can generate new symbols with `\knitbox`: $\boxed{\text{a}}$ $\boxed{\text{Cable 6 left}}$. 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}` $\boxed{\heartsuit}$. However, these may look a little incongruous with the rest of the font, so use with care.

4.2. Space in the input.

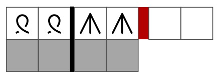
A normal space in the source code in a knitting chart, like a space in math mode, is ignored. In a knitting chart, unlike in normal text, you want to prescribe all your line breaks. For convenience, knitting charts have the macro `\obeylines` built in, so that a single ⟨return⟩ produces a new paragraph (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 `%`.

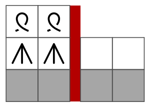
Unfortunately, the trick I use to make ⟨return⟩s trigger new lines is delicate; specifically, if you put your chart inside another command, it stops working. (It

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

```
\newdimen\knitlinewd
\setlength{\knitlinewd}{4pt}
\parbox{77pt}{
\chart{
tt|
AA\|%
--\\
==|==\\}
}
```



```
\newdimen\knitlinewd
\setlength{\knitlinewd}{4pt}
\begin{center}
\chart{
tt|
AA\|%
--\\
==|==\\}
\end{center}
```



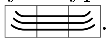
works fine inside environments.) (The bar `|` also stops working; it produces a plain black bar whose width will not change if you redefine `\knitlinewd`.)

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).

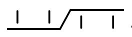
4.3. **Space in the output.** A tilde ∼ produces a gap the width of one stitch. This is meant to be used in charts with ragged edges. It's been designed so that automatically placed rownumbers show up after the gaps left by ∼, not before. If you really want them to show up earlier, you can use single quotes instead of ∼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 ∼s alone on their own line; `‘‘`s are much better behaved.

In the nongrid font, $\boxed{.}$ and $\boxed{,}$ both produce empty boxes. (Of different colors; $\boxed{.}$ produces white, $\boxed{,}$ 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.

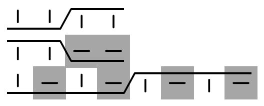
4.4. **Ligatures and cables.** TEX has a built-in ligature mechanism that lets it get ¿fire–fly? instead of ?‘fire--fly? when you type `?‘fire--fly?`. knitting uses this for wide horizontal sequences such as . The ligature mechanism also lets you generate the reverse of ↻: `-"`, `="` and `""` produce ☐↺, ▨↺, and ↺, respectively.

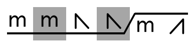
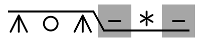
However, the ligature mechanism really comes into its own when making cable symbols.

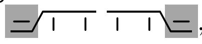
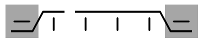
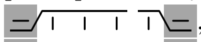
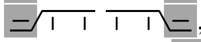
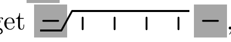
In the non-grid font, cables look like this: . The keys `k`, `p`, `K`, and `P` produce raised or lowered knit and purl symbols, and the ligature mechanism adds in the underbars or slant connectors automatically.

So:

```
kkKK
KKpp
kpkpKPKP
```



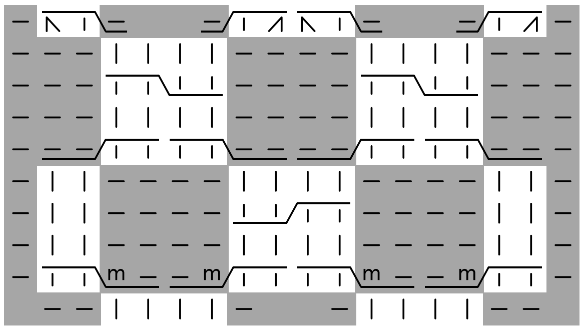
Some allowance for peculiar cables has been made. You can get a front increase or decrease with `N` or `D`, or a back increase or decrease with `n`, `d`, `e`, or `o`: `nedoND` . All of the symbols present in the font can be used in cables with the help of `\cableleft` and `\cableright`: `\cableleft{AOA}{=*=}` 

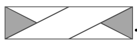
Explaining to knitting when one cable starts and another ends can be hard: `pKKKKp` probably means , but it could mean  or , and the ligature mechanism isn't smart enough to default to , let alone read your mind. So you have to tell it what you want. (Or you'll get , which is not what you want.)

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:

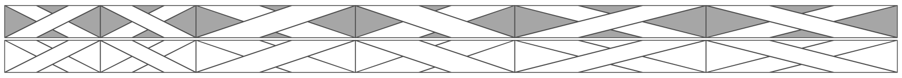
```
=DKp,,pKD DKp,,pKD=
===----=======----===
===KKkk====KKkk===
===----=======----===
=ppKKCCppggKKCCpp=
=--====----=====--=
=--====kkKK=====--=
=--====----=====--=
=KKepgeKKCCepgeKK=
,==----=,,=----==,
```



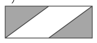
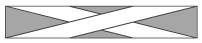
In the grid fonts, cables look like this: . The letters `kpcgKPCG` work for simple cables as in the non-grid font.

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:



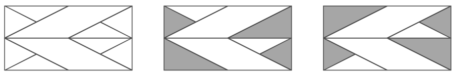
You get these by putting `p`s (not `g`s) between the `k`s and `K`s.[3]

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.

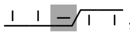
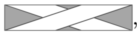
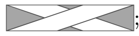
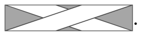
The keys `N`, `n`, `e`, `o`, `d` and `D` have a different function here.

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]

```
kkDD kkNN ppNN
KKdd KKnn KKoo
```



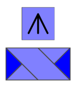
`e` and `o` have the same effect.

[3]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 `c`s and `g`s. 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 `K`s will produce .

[4]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.

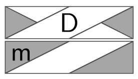
Some special effects are possible. You can get fancy with the colors of these symbols:[5]

```
\colorbox{lightblue}{A}
\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.

5. 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.

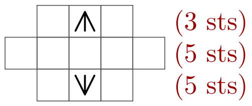
If used inside a knitting chart, the command `\countstitches` takes one argument and sets the counters `stitchcountout` and `stitchcountin` to reflect how many stitches that sequence of stitches would produce or consume, assuming that all the symbols have the meanings given in knitkey.tex.

So, for example:

```
\chart{\countstitches{->-}}
stitchcountout: \thestitchcountout              stitchcountout: 3
stitchcountin: \thestitchcountin                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-~}}
```



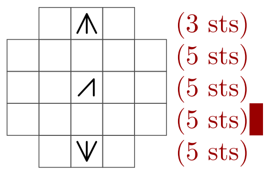
(A more complicated example is in `knitexamples.tex`. Repeated patterns affect stitch count in strange ways; that example should show you how to cope.)

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:

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

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

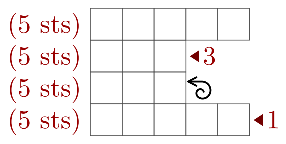


The heavy bar beside Row 2 indicates that Row 3 uses six stitches, but Row 2 produces only five. `\stitchcountwarningbar` is built using the same code as TEX's usual overfull rules; thus, it will only appear (in LATEX) if the `draft` document option is enabled.

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:

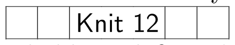
```
\rnoddonly
\stitchcountchart[right]{
   -----
   ---\adjuststitchcount{2}
   ---"\adjuststitchcount{2}
   -----
   }
```



The only way to customize this is the optional argument (which places row numbers as usual) and by redefining the commands `\printleftstitchcount` and `\printrightstitchcount`. The default (LATEX) values are

```
\newcommand{\printleftstitchcount}{{\color{rncolor}%
   \textnormal{(\thestitchcountout\ sts) }}}
\newcommand{\printrightstitchcount}{}
```

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.

In particular, `\Knit`, `\Purl`, `\knitbox` and `\purlbox` count out as many stitches as their printed chart width. So `\Knit{Knit 12}{7}` 

will be counted as 7 stitches. The best way to deal with this is probably to define a `\myknit` command:

```
\newcommand{\myknit}[2]{\Knit{Knit #1}{#2}%
    \adjuststitchcount{-#2}\adjuststitchcount{#1}}
```

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{}`.)

6. Revision history

August 2010. I've added the symbols $\boxed{\diagdown}$, $\boxed{\diagup}$, $\boxed{\forall}$, $\boxed{\curlyvee}$, $\boxed{\curlyvee}$, and $\boxed{\varrho\varrho}$ 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`.

7. 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`

- `knitg_sc_in.afm`
- `knitg_sc_out.afm`
- `knitgg.afm`
- `knitgn.afm`
- `knitgp.afm`
- `knitn_sc_in.afm`
- `knitn_sc_out.afm`
- `knitnl.afm`
- `knitnn.afm`
- `knitnp.afm`
- `knitnr.afm`
- `knitw_sc_in.afm`
- `knitw_sc_out.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
- knitg_sc_in.mf
- knitg_sc_out.mf
- knitgg.mf
- knitgn.mf
- knitgp.mf
- knitn_sc_in.mf
- knitn_sc_out.mf
- knitnl.mf
- knitnn.mf
- knitnp.mf
- knitnr.mf
- knitw_sc_in.mf
- knitw_sc_out.mf
- knitwg.mf
- knitwn.mf
- knitwp.mf

knitting/fonts/tfm

- knitg_sc_in.tfm
- knitg_sc_out.tfm
- knitgg.tfm
- knitgn.tfm
- knitgp.tfm
- knitn_sc_in.tfm
- knitn_sc_out.tfm
- knitnl.tfm
- knitnn.tfm
- knitnp.tfm
- knitnr.tfm
- knitw_sc_in.tfm
- knitw_sc_out.tfm
- knitwg.tfm
- knitwn.tfm
- knitwp.tfm

knitting/fonts/type1

- knitg_sc_in.pfb
- knitg_sc_out.pfb
- knitgg.pfb
- knitgn.pfb
- knitgp.pfb
- knitn_sc_in.pfb
- knitn_sc_out.pfb
- knitnl.pfb
- knitnn.pfb
- knitnp.pfb
- knitnr.pfb

- knitw_sc_in.pfb
- knitw_sc_out.pfb
- knitwg.pfb
- knitwn.pfb
- knitwp.pfb

knitting/tex/latex

- knitting.sty
- uknit.fd

knitting/tex/plain

- knitting.tex