

# KKsymbols Package Documentation

Kosei Kawaguchi a.k.a. KKT<sub>E</sub>X

Version 2.2.1 (2026/06/21)

## 目次

1	Outline .....	3
2	Acknowledgements / Credit .....	4
3	Installation .....	4
3.1	Dependencies .....	4
3.2	Loading and Options .....	4
4	Caution .....	5
5	Commands .....	5
5.1	The maru series .....	5
6	Automatic alignment (reference box) .....	6
6.1	Overriding the reference .....	7
7	Rotation .....	7
7.1	Commands .....	7
7.2	Vertical mode .....	8
8	The seihou series .....	9
9	The kakko series .....	10
9.1	Additional Description: \ichimoji and \zenkakuhabafixer .....	11
10	License .....	11
11	Version History .....	11

# 1 Outline

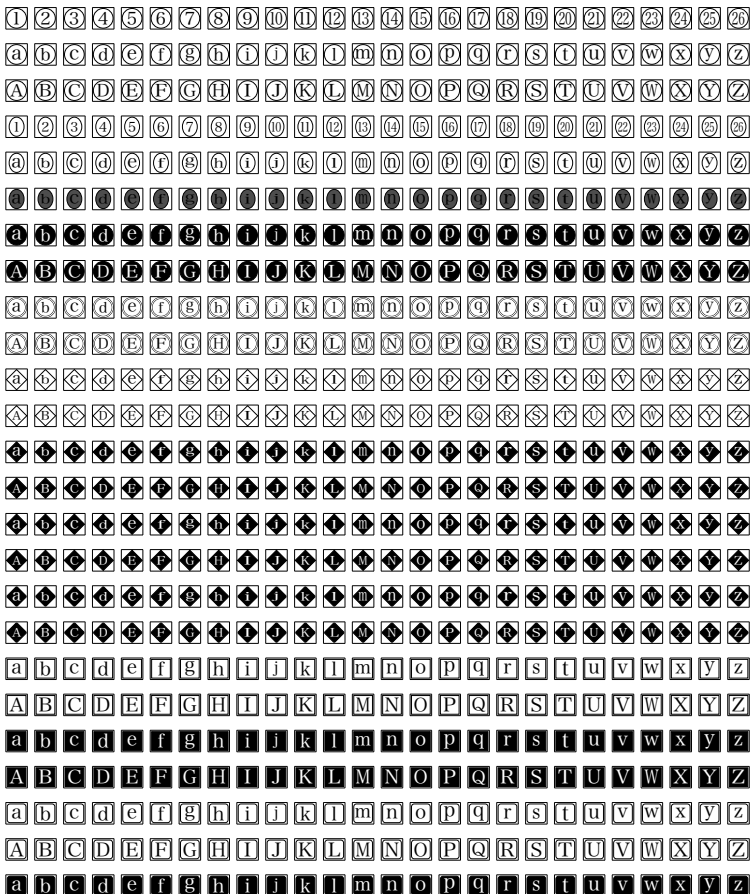
Japanese: このパッケージは、既存の otf フォントに頼ることなく、「任意のフォント、任意の引数で」丸数字などの特殊記号を再現する目的で作成されています。

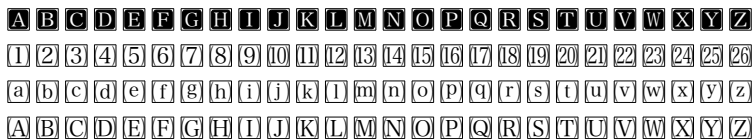
luatexja-otf における `\ajMaru` などは、その設計上いたしかなないデメリットとして、早見表がないと使い物にならないというものがありません。しかし、本パッケージでは、一から特殊文字を設計し直すという取り組みを行なっているため、そのようなデメリットが解消されています。

English: This package is designed to reproduce special characters, such as circled numbers, using arbitrary fonts and parameters without relying on existing otf font sets.

A known disadvantage of commands like `\ajMaru` in `luatexja-otf` is that they are practically unusable without a reference chart. In contrast, this package addresses this issue by re-engineering special characters from the ground up, eliminating the need for complex lookup tables.

あ あ (あ) あ あ





## 2 Acknowledgements / Credit

In developing this package, I made extensive use of the advice I received from Mr. Yusuke Terada.

I recommend you to refer to his article when you develop new-type symbols on  $\LaTeX$ .

<https://doratex.hatenablog.jp/entry/20211205/1638697391>

## 3 Installation

Place `KKsymbols.sty` in a directory where  $\LaTeX$  can find it, e.g., your local `texmf` tree or alongside your document.

### 3.1 Dependencies

This package depends on the following packages.

- `LuaLaTeX-jā`
- `tikz`
- `clac`
- `luacode`
- `kvoptions`

### 3.2 Loading and Options

Load the package: `\usepackage[<options>]{KKsymbols}`

Currently, the following package option is provided.

**`tsumesuji`** To specify whether or not to “shrink” the box width occupied by “1”. By default, `tsumesuji=1` is specified, and the effect is activated. If you deactivate it, `tsumesuji=0` will do.

To clarify, this option is applied when the argument of a command subject to the `tsumesuji` option—after expansion—is a numeric string of two or more digits and contains the digit “1”. The difference is present in the following example:

```

Input
1    % NOT APPLIED
2    \kakko{\vphantom{X}12}

```

```

3
4   % APLIED
5   \kakko{12}

```

Output

(12  
(12

## 4 Caution

Since this package internally calls `\ltjghostbeforejachar` and `\ltjghostafterjachar`, it can be used only in a LuaLaTeX environment.

## 5 Commands

### 5.1 The maru series

This package provides `\maru`, `\kuromaru`, and `\nmaru`. Each of them takes one mandatory argument and no optional arguments. You can pass strings of any length and in any font as arguments.

In most cases, `\maru{argument}` will meet your demands. However, only when you take lower-case alphabet in these commands, you must use star-command just like `\maru*{m}`<sup>1)</sup>.

Input
Mind the star option!

```

1   % Normal Characters
2   \maru{A}\maru{あ}\maru{QED.}
3
4   % Lowercase Alphabetic Characters
5   \maru*{a}\maru*{j}\maru*{z}

```

Output

AあQED.  
a j z

They are used as follows.

---

1) `\jegg` is an exception of this rule. When you use `\jegg`, you don't have to put the star option no matter the argument is lowercase or not. If you do it, the background color of the `\jegg` changes into gray. Only in this case, the effect of the option is different.

表 1: maru series

argument	\maru	\kuromaru	\nmaru	\jegg	\jegg*
1	①	❶	①	①	❶
97	㍻	㍻	㍻	㍻	㍻
だ	㊦	㊦	㊦	㊦	㊦
ぱぱぱ	㊰㊰㊰	㊰㊰㊰	㊰㊰㊰	㊰㊰㊰	㊰㊰㊰
m	㊴	㊴	㊴	㊴	㊴
Qjg	㊶	㊶	㊶	㊶	㊶

They behave as if they were single kanji or hiragana characters:

あいう㊦あいう①②③あいうえお

The spacing between \maru and other characters is adjusted using \ltjghostbeforejachar and \ltjghostafterjachar so that it behaves like hiragana or kanji.

When changing the font size using commands such as \Large, each command is scaled proportionally according to the font size change:



You can also change the current font:



## 6 Automatic alignment (reference box)

By default (since v2.2.0), each enclosure sizes and vertically centers its argument against the current font's reference height (its x-height) rather than against the argument's own bounding box. This keeps short or irregular glyphs—most notably the lowercase roman numerals produced by \Rrnum (from the `KKran` package)—uniform in size and position, and it makes the result consistent across fonts.

Previously you had to insert \vphantom by hand to obtain a consistent reference. Now the same, aligned result is produced automatically:

Input	No more manual vphantom
1 % Before (manual workaround)	
2 \maru{\vphantom{\Rrnum{6}}\Rrnum{1}}\maru{\vphantom{\Rrnum{6}}\Rrnum{8}}	
3	
4 % Now (automatic, identical result)	
5 \maru{\Rrnum{1}}\maru{\Rrnum{8}}	

## Output



Tall content (kanji, kana, uppercase letters, digits) is taller than the reference, so it is unaffected and existing documents keep their appearance.

## 6.1 Overriding the reference

`\kksref{<material>}` Use the height and depth of `<material>` as the reference instead of the automatic one. The effect is scoped like `\RotYoko`, so enclose it in a group. This is handy for list labels:

### Input

```
1  {\kksref{\Rrnum{8}}}%  
2  \maru{\Rrnum{1}}\maru{\Rrnum{4}}\maru{\Rrnum{8}}
```

`\kksreff` Disable the reference box and fall back to the argument's own metrics (the behavior before v2.2.0).

`\kksrefauto` Restore the automatic reference (the default). `\kksrefreset` is an alias.

## 7 Rotation

### 7.1 Commands

This package provides `\RotTate` and `\RotYoko`. The differences are as follows:

**In horizontal mode** You should use `\RotYoko`. The default value is 0. Therefore, if you use `\RotYoko` with no arguments, this is equal to `\RotYoko[0]`

**In vertical mode** You should use `\RotTate`. The default value is 90. Therefore, if you use `\RotTate` with no arguments, this is equal to `\RotTate[90]`

From a technical perspective: Commands like `\maru` provided by this package automatically rotate their arguments based on the “current typesetting direction” Specifically, the package applies a 0-degree rotation for horizontal writing (yoko-gaki) and a 90-degree rotation for vertical writing (tate-gaki).

The commands `\RotYoko` and `\RotTate` redefine this “automatic rotation angle” to the value specified in their arguments. Consequently, the effect is persistent unless localized (similar to how font-size commands like `\small` behave). Therefore, please ensure you use appropriate scoping, such as enclosing the command within curly braces `{...}`, when applying these settings.

Input

```
1 % In horizontal mode
2 {\RotYoko[45]\kakko{あ}\kakko{い}\kakko{う}}\par
3 {\RotYoko[60]\kakko{1}\kakko{2}\kakko{3}}
4
5 % In vertical mode
6 \parbox<t>{5\zw}{% <↳ option requires lljtext package.
7   {\RotTate[45]\kakko{あ}\kakko{い}\kakko{う}}\par
8   {\RotTate[60]\kakko{1}\kakko{2}\kakko{3}}
9 }
```

Output

(あ)(い)(う)  
(あ)(い)(う)  
(1)(あ)  
(2)(い)  
(3)(う)

## 7.2 Vertical mode

When you want to typeset ㊦, for instance, in vertical mode, you should use `\RotTate` command.

As described in the previous subsection, the effect of `\RotTate` lasts, when localized, in a certain group. So when you typeset hiragana or kanji, use it like this:

Input

```
1 % In vertical mode
2 \parbox<t>{5\zw}{%
3   {\RotTate[0]\kakko{あ}\kakko{い}\kakko{う}}\par
4   \kakko{1}\kakko{1}\kakko{3}
5 }
```

Output

(1)(あ)  
(1)(い)  
(3)(う)



# 8 The seihou series

The commands introduced below are used in exactly the same way as the maru series. In most cases, `\seihou{argument}` will meet your demands. However, only when you take lowercase alphabet in these commands, you must use star-command just like `\seihou*{m}`.

表 2: seihou series

argument	<code>\seihou</code>	<code>\kuroseihou</code>	<code>\seimaru</code>	<code>\kuroseimaru</code>
1				
97				
だ				
ばばば				
m				
Qjg				

表 3: hishi series

argument	<code>\hishi</code>	<code>\kurohishi</code>	<code>\maruhishi</code>	<code>\kuromaruhishi</code>
1				
97				
だ				
ばばば				
m				
Qjg				

Input

Mind the star option!

```

1 % Normal Characters
2 \seihou{A}\seihou{あ}\seihou{QED.}
3
4 % Lowercase Alphabetic Characters
5 \seihou*{a}\seihou*{j}\seihou*{z}

```

Output

## 9 The kakko series

The commands introduced below are used in exactly the same way as the maru series.

In most cases, `\kakko{argument}` will meet your demands. However, only when you take lowercase alphabet in these commands, you must use star-command just like `\kakko*{m}`<sup>2)</sup>.

表 4: kakko series ①

argument	<code>\kakko</code>	<code>\sumikakko</code>	<code>\kakukakko</code>	<code>\kikakko</code>	<code>\yakko</code>
1	(1)	【1】	[1]	〔1〕	⟨1⟩
97	(97)	【97】	[97]	〔97〕	⟨97⟩
だ	(だ)	【だ】	[だ]	〔だ〕	⟨だ⟩
ぼぼぼ	(ぼぼぼ)	【ぼぼぼ】	[ぼぼぼ]	〔ぼぼぼ〕	⟨ぼぼぼ⟩
m	(m)	【m】	[m]	〔m〕	⟨m⟩
Qjg	(Qjg)	【Qjg】	[Qjg]	〔Qjg〕	⟨Qjg⟩

表 5: kakko series ②

argument	<code>\nykakko</code>	<code>\namikakko</code>	<code>\kagikakko</code>	<code>\nkagikakko</code>	<code>\ichimoji</code>	<code>\zenkakuhabafixer</code>
1	⟨1⟩	{1}	「1」	『1』	𝐈	l
97	⟨97⟩	{97}	「97」	『97』	97	97
だ	⟨だ⟩	{だ}	「だ」	『だ』	だ	だ
ぼぼぼ	⟨ぼぼぼ⟩	{ぼぼぼ}	「ぼぼぼ」	『ぼぼぼ』	ぼぼぼ	ぼぼぼ
m	⟨m⟩	{m}	「m」	『m』	m	m
Qjg	⟨Qjg⟩	{Qjg}	「Qjg」	『Qjg』	Qjg	Qjg

Input	Mind the star option!
<pre> 1 % Normal Characters 2 \kakko{A}\kakko{あ}\kakko{QED.} 3 4 % Lowercase Alphabetic Characters 5 \kakko*{a}\kakko*{j}\kakko*{z} </pre>	
Output	
<pre> (A)(あ)(QED.) (a)(j)(z) </pre>	

2) `\zenkakuhabafixer` is an exception of this rule. It does not take a star option.

## 9.1 Additional Description: `\ichimoji` and `\zenkakuhabafixer`

The major difference of `\ichimoji` and `\zenkakuhabafixer` is that the former changes the vertical scale to force its totalheight to `\zw`, but the latter doesn't. You can see the difference as follows:

`\ichimoji` ㊦

`\zenkakuhabafixer` ㊦

## 10 License

Released under the MIT License.

## 11 Version History

- **v1.0.0 (2025/10/03)** — Initial public release.
- **v1.0.1–1.0.4** — Added `\ichimoji`; fixed various bugs.
- **v1.1.0 (2025/10/28)** — Unified all commands to `zenkaku` (full-width).
- **v1.1.1 (2025/11/10)** — Refined `\ichimoji` scaling logic.
- **v2.0.0 (2025/12/23)** — Overhauled scaling to match OTF character quality.
- **v2.0.1 (2026/01/08)** — With the update to `luatexja` version 20260107.0, the commands provided by the `KKsymbols` package now behave identically to native Japanese characters. This improvement is due to the bug fixes in `\ltjghostbeforejachar` and `\ltjghostafterjachar`. Previously, when multiple commands from this package were used consecutively, proper glue was not inserted between them; however, this issue has been resolved in this update.
- **v2.0.2 (2026/01/20)** — This update fixes a critical bug where arguments of the `\kakko` command could overflow when using specific fonts (for example, Hiragino fonts with weights W4 and above).
- **v2.1.0 (2026/02/16)** — In this update, the following points are changed.
  - `\period` command was deleted. It have never been used since the significance of existence had been absolutely questionable. I finally decided to delete it.
  - New package option `tsumesuji` was added.
  - New command `\zenkakuhabafixer` was added.
- **v2.1.1 (2026/02/17)** — An emergency update to fix a bug where the arguments of `\kakko`, `\maru` etc. caused expansion errors.
- **v2.1.2 (2026/02/19)** — The effects are applied when the argument of a command subject to the `tsumesuji` option—after expansion—is a numeric string of two or more digits and contains the digit “1”.
- **v2.1.4 (2026/04/17)** — When the arguments provided by this package contain only one-digit-number, the internal resize algorithm is nullified.

- **v2.2.0 (2026/06/21)** — Automatic vertical alignment was introduced. By default, the content of each enclosure is now sized and centered against the current font’s reference height (x-height) instead of the glyph’s own metrics, so that short and irregular glyphs (such as the lowercase roman numerals of `\Rrnum`) line up uniformly without writing `\vphantom` by hand. The new commands `\kksref`, `\kksrefoff`, and `\kksrefauto` (alias `\kksrefreset`) control this behavior. Tall content (kanji, kana, uppercase, digits) is unaffected, so existing documents are unchanged.
- **v2.2.1 (2026/06/21)** — Bug fix: a font-size command inside an argument (e.g. `\maru{\Large A}`) previously caused an infinite loop, because the internal one-digit/`tsumesuji` check expands the argument with `\text_expand:n`, which loops on font-size commands. Such commands are now declared expand-equivalent to empty for that internal check; the visible output size is unaffected. (The hang predated v2.2.0.)