一時的な片手でのキーボードの利用に関する考察

平成 15 年 2 月 10 日

情報電子工学科 竹野研究室
柴 信一郎

目 次

1 はじめに **1**

2 片手でのキーボードの利用に関する話 **1**
2.1 背景 1
2.2 一時的な片手での利用 1
2.3 市販されているものについて 2
2.4 UNIX X Window xmodmap 2

3 xmodmap での変更仕組み **3**
3.1 キーボードの仕組み 3
3.2 キーマップの構成 4
3.3 変更方法 6

4 変更例 **7**
4.1 変更例 1 7
4.2 変更例 2 8
4.3 変更例 3 8
4.4 変更例 4 9
4.5 変更例 5 9

5 実験と考察 **9**
5.1 第 4 章の変更例を試しての感想 10
5.2 実験結果 11
5.2.1 “Fig.4.1 変更例 1 , Fig.4.2 変更例 2” について 11
5.2.2 “Fig.4.3 変更例 3 , Fig.4.4 変更例 4” について 11
5.2.3 “Fig.4.5 変更例 5” について 12
5.3 考察 12
5.3.1 Fig.4.1 変更例 1 , Fig.4.2 変更例 2 について 12
5.3.2 Fig.4.3 変更例 3 , Fig.4.4 変更例 4 について 12
5.3.3 Fig.4.5 変更例 5 について 12

6 まとめ **13**

参考文献 **18**

概要

現在、コンピュータで広く使われているキーボードを、片手で扱うことができれば、片手が塞がっている場合、あるいはなんらかの障害で片手が使えない場合などに役に立つと考えられる。現在、片手で扱うためのキーボードは既に市販されたものがあるが、本研究では一時的な事情で片手で扱えなくなったときのことを考慮している。UNIX 上の X Window System では、xmodmap という仕組みがあり容易にキーの配置をカスタマイズでき、カスタマイズしたものは、任意に通常の状態に戻すことができる。また、カスタマイズするにあたり、キーボードの構成について調べ、どのようにしてキーボードを片手で扱うように変更するかということを考案し、それを実際にキーボードに割り当て、実験し考察する。

1　はじめに

コンピュータで広く使われているキーボードは、両手で扱うことは否めない。しかし、何らかの事情で片手が使えなくなった場合に、キーボードを片手で扱うことになる。そのような状況になった場合、キー範囲が広いため片手によるタイピングは辛く、文字キー等と他のキーを組み合わせて使用することも多いので不便性が考えられる。そのため、キーボードを片手で扱えるようにするために、他にも研究、製品等といったものがないか探し、それらを参考にし、実験して利便性を考えキーボード上のキーマップを一時的に変更して利用できるように考える。

2　片手でのキーボードの利用に関する話

2.1　背景

突然の事故等で障害を負ったことによって、片手が使用できなくなった場合や一時的に片手が塞がることによって、片手でキーボードを使用することになる。

コンピュータで使われているキーボードを利用している人にとっては、片手が使えないというのは非常に困難であると思われる。片手によるキー操作となれば、キー操作に慣れていようが慣れていまいが不便であり、ストレスさえ感じるのは必至である。だとして、市販されている “片手キーボード” を使えば良いかもしれないが、かと言って、一時的な事で片手が使えないのならば、新ためて “片手キーボード” を購入する必要もないであろう。

そのために、

“一時的な片手でのキーボード利用”

を考察する。

元々、“片手キーボード” のユーザならば問題は無い。

2.2　一時的な片手での利用

一時的にキーボードを片手で利用するならば、キーボード上のキーマップを変更することによって、片手でのキーボード操作が可能になるのである。

そこで、キーマップをどのように変更するかというと片手で扱うために、キー数はできるだけ少ない形が良いと思われるので、

- 文字列の左側
- 文字列の右側
- テンキーでの操作

等のキーマップ変更パターンを考える。
変更方法等は、後に示す。

2.3 市販されているものについて

2.1 で示した、市販されている “片手キーボード” について述べる。

- CUT Key Pocket

この製品は、MISAWA ホームで開発されたもので

> “進化しつづけるパソコン本体に対し、キーボードのカタチはほとんど変化していませんでした。もっと正確に、もっと効率のよい文字入力を ···”

というコンセプトを掲げている。

> “コンパクトで機能性の高い小型キーボード”

として登場。
ローマ字入力における文字の使用頻度を考慮して、文字や記号、数字を 12 個の文字キーに配置し、キースイッチ数全 31 個。機能キーとの組合せでフルキーボードと同等の機能を発揮し、また、入力モードを切り替える専用キーを設置。一目で入力モードがわかる表示ランプを搭載している。
以下に文字キーの配列を示す。
日本語を構成している 5 つの母音と、各行の子音を 50 音順にわかりやすくレイアウトしてあり、テンキーや電卓、携帯電話、リモコンと同じ馴染みやすい 3 列 4 段の配列となっている。
基本入力はローマ字入力で、英語入力も併用している。
入力方式は、各キーの 1 番目の文字例えば、図を参考にすると S は 1 回、Z は 2 回、J は 3 回打って入力という設定になっている。

2.4 UNIX X Window xmodmap

X Window は、1980 年代中頃から MIT (マサチューセッツ工科大学) で開発された UNIX ワークステーション上のウィンドウシステムである。これは、

- ソースコードが無料で公開されている
- システム中でハードウェアに依存する部分が明確に切り分けられているので、UNIX ワークステーションであれば、容易に移植することが可能

等の理由から急速に普及し、UNIX 環境上の最も一般的なウィンドウシステムとなっている。
特徴を列挙すると、以下のようになる。

- サーバクライアント · モデルに基づくシステム構成

- ネットワーク透過性
- 数多くのプラットフォーム (ワークステーション環境) で動作可能
- 豊富なカスタマイズ機能
- 種々の GUI スタイルをサポート
- 日本語入力を含む国際化機能

本研究では、豊富なカスタマイズ機能から X Window のクライアントの 1 つである xmodmap を使用することによって、ユーザ毎にキーボード上のキーマップを自由に使いやすい配置にカスタマイズすることが可能である。これを利用することによって、キーボード上のキーマップを片手で扱うように変更できる。

3 xmodmap での変更仕組み

3.1 キーボードの仕組み

xmodmap での変更の前に、キーボードについて知る必要がある。キーボードは、

“キーコード (keycode)”
“キーシム (keysym)”

という 2 階層のモデルでキーを構成している。
以下に詳しく示す。

キーコード キーボード上の全てのキーに、キーの持つ意味とは関係なく機械的に番号を割り当てたものだが、キーコードを用いればキーボード上のすべてのキーを認識できるが、キーボードを作成したメーカーによって異なるため基本的にメーカー間で互換性はなく、サーバ依存である。例えば、 x 社製のキーボードは、 Ctrl キーを押したとき 83 というキーコードを発生させるが、 y 社製のキーボードは 38 というキーコードを発生させる。つまり、キーコードとそのキーコードを発生するキーに印刷されている文字は何の因果関係もない。

キーシム キーボード上のキーに表示されている言語、数字、記号等、キーの持つ意味をもとに決めており、全てのサーバで共通なキーコードである。よってキーシムはすべてのキーボードの持つキーの共通な性質を持ち、必ずしも全てのキーに対してキーコードが割り当てられていない場合もある。キーコードが割り当てられているものは、基本的にキーに印刷されている文字と対応している。

その他に、X Window で割り当てられている修飾キー名で、

“MODIFIER”

という文字や数字、記号キーと一緒に用いられるキーが存在する。
　例えば、 shift キーは大文字を入力する場合に文字キーと一緒に用いられる修飾キーである。
X Window では、このような修飾キー名として shift, lock, control, mod1, mod2, mod3, mod4, mod5 の 8 つをサポートしている。
修飾キーにはどのようなキーが登録されているかは、

```
% xmodmap -pm
```

として、以下のように設定されている。

```
shift       Shift_L,  Shift_R
lock        Caps_Lock
control     Control_L
mod1        Alt_L
mod2        Mode_switch
mod3        Num_Lock
mod4        Meta_L, Meta_R
mod5
```

上記は左側に修飾キー名が表示され、右側にその修飾キーに登録されているキーのキーシム名が表示されている。

3.2　キーマップの構成

　キーマップの構成を見るには、

```
% xmodmap -pke
```

とすると、使用中のキーマップのキーコード及びキーシムが以下のように表示される。

```
KeyCode     Keysym (Keysym)
Value       Value   (Name)
.
.
.
keycode  37 = 1 exclam kana_NU
keycode  38 = 2 quotedbl kana_FU
keycode  39 = 3 numbersign kana_A kana_a
```

```
keycode  40 = 4 dollar kana_U kana_u
keycode  41 = 5 percent kana_E kana_e
.
.
.
keycode  61 = q Q kana_TA
keycode  62 = w W kana_TE
keycode  63 = e E kana_I kana_i
keycode  64 = r R kana_SU
keycode  65 = t T kana_KA
keycode  66 = y Y kana_N
.
.
.

keycode  98 = Left F30 KP_4
keycode  99 = F31 F31 KP_5
keycode 100 = Right F32 KP_6
keycode 101 = KP_Insert KP_Insert KP_0
.
.
.
```

以上のように 1 〜 4 つのカラムから成り立っている。

- 1 つ目のカラムは、1 〜 0 の数字や小文字のアルファベット、記号で修飾キーを押さないときのキーシム値である。

- 2 つ目のカラムは、1 〜 0 の数字や小文字のアルファベット、記号で shift 修飾キーを押したときのキーシム値である。

- 3 つ目のカラムは、1 〜 0 の数字や小文字のアルファベット、記号で mod2 修飾キーの Mode_switch を押したときのキーシム値である。ただし、keycode 98 〜101 のようなテンキーの場合は、 mod3 修飾キーの Num_Lock を押したときのキーシム値である。

- 4 つ目のカラムは、1 〜 0 の数字や小文字のアルファベット、記号で shift 修飾キー及び mod2 修飾キーを押したときのキーシム値である。

3.3 変更方法

xmodmap による キーマップ及び MODIFIER の変更として、

```
% xmodmap -e 'keycode NUMBER = KEYSYM'
```

及び

```
% xmodmap -e 'keysym KEYSYM = KEYSYM'
```

とすると、キーを変更することができる。
この変更式は MODIFIER にも使えるが、クオートのなかの変更式が多少異なる。
以下に示す。

構文	機能
add MODIFIER = KEYSYM	指定したキーを修飾キーに登録
remove MODIFIER = KEYSYM	指定したキーを修飾キーから削除
clear MODIFIER	修飾キーを全てクリア

以上の図が MODIFIER の変更式の説明である。例として、

```
% xmodmap -e 'remove shift = Shift_R'
% xmodmap -e 'add control = Shift_R'
```

この変更式は、Shift_R が Control キーとして働くことになった。
別の形の変更式として以下のような設定で変更が可能である。

```
!!
!!
!! keymap change
!! .Xmodmap
!!
!!  37  (1): 1 exclam kana_NU
!!  42  (6): 6 ampersand kana_O kana_o
!!
keycode 37 = 1 6 kana_NU
keycode 42 = exclam ampersand kana_O kana_o
!!
!!
```

以上の図はキーマップ変更式で、標準の状態であれば 'Shift + 1' で出力される '!' が、数字の '6' が表示されることになり、'6' のキーを打つと '!' が表示されるように割り当てられている。

```
!!
!!
!! 105 (Num_Lock) : Num_Lock
!!  50 (Backspace): BackSpace
!!
remove mod3 = Num_Lock
keycode 105 = BackSpace
keycode  50 = Num_Lock
add mod3    = Num_Lock
!!
!!
!!
```

以上の図は MODIFIER の NumLock と BackSpace を入れ換えた変更式である。

このように、キーマップを変更していくことによって、片手で扱えるようなキーボードが出来るのである。

ワークステーション終了時、変更したキーマップは元に戻されるが、キーマップを元に戻すファイルを作成しておくと、ワークステーションを終了させることなく、動的にキーマップの再設定が可能である。

これまでに示して来たキーマッピングの変更式を、 UNIX ワークステーションの使用時に常に有効にしておきたい場合には、

- .xsession ファイル

に、変更式を張り付け、指定しておけばよい。

4 変更例

4.1 変更例 1

Fig.4.1 は、標準時のテンキーと xmodmap による変更式によって割り当てられたキーマップである。

特徴 MISAWA ホームの CUT Key Pocket をモデルにしてテンキーに割り当てて構成した。矢印キーの Right を Space に割り当てた。変更時の右にある文字は、Shift と同時押しで出力される。

長所 CUT Key Pocket のように、コンパクトにまとまっていて、配列も比較的憶えやすいように、子音の清音と濁音を一緒にしている。使用頻度が少い文字は、母音や子音の清音だけの部分に当てはめた。

短所 数字を入力する場合は、BackSpace が NumLock になっているので、BackSpace を押さないと入力できない。記号に関しては全く考慮していないのでテンキーによる入力は不可能。BackSpace の、押したままでの削除ができなくなり、1 文字毎に削除することになる。

4.2 変更例 2

Fig.4.2 も、標準時のテンキーと xmodmap による変更式によって割り当てられたキーマップである。

特徴 変更例 1 とは殆ど変化はないが、自分なりに少々変更したものである。矢印キーの Right を Space に割り当てた。変更時の右にある文字は、Shift と同時押しで出力される。

長所 変更例 1 とは殆ど変化はないが、比較的、子音部を順序通りに並べたことで多少は扱いやすい。

短所 数字を入力する場合は、NumLock が BackSpace になっているので、テンキーでの数字入力が不可能である。記号に関しては全く考慮していないのでテンキーによる入力は不可能。BackSpace の、押したままでの削除ができなくなり、1 文字毎に削除することになる。

4.3 変更例 3

特徴 左手で使うことを考慮し、文字列を折り畳んだ形で左側部分に合わせたものである。Tab キーを Return キーに割り当て、確定キーを BackSpace キーに割り当てた。標準状態であれば shift + 文字キーで大文字もしくは、1〜5 の記号は出力されるが、この変更例では変更時の右側に表示されている文字が出力される。ただし、G と V は標準と変わっていない。

長所 QWERT 配列のままで、特別にキーを並べ変えたということでもなく、文字列をを折り畳んだ状態のようなものなので、文字、数字入力は比較的扱いやすい。

短所 記号に関しては全く考慮していない。
BackSpace の、押したままでの削除ができなくなり、1 文字毎に削除することになる。

4.4 変更例 4

特徴 左手で使うことを考慮し、文字列を左側部分に合わせスライドさせた形である。

Tab キーを Return に割り当て、確定キーを BackSpace に割り当てた。

標準状態であれば shift + 文字キーで大文字もしくは、1〜5 の記号は出力されるが、この変更例では変更時の右側に表示されている文字が出力される。ただし、G と V は標準のままの状態である。

長所 QWERT 配列のままで、特別に並べ変えたということでもなく、文字列をスライドさせた状態なので、文字、数字入力は比較的扱いやすい。

shift キーを押した状態での入力になるが、右手での使用も悪くはない。

短所 記号に関しては全く考慮していない。

BackSpace の、押したままでの削除ができなくなり、1 文字毎に削除することになる。

4.5 変更例 5

特徴 右手で使うことを考慮し、右側部分の文字列に合わせスライドさせた形である。

長所 記号キーが横にあるので、左手で使うこと考慮したものよりは使いやすい。

短所 「,」、「.」、「/」、「;」の記号と G, C, V, B を入れ換えたので、多少、記号の扱いが不便になる。

今までに表した変更図はこの研究で、自分で創り出した構成である。

5 実験と考察

本研究では、

“一時的な片手でのキーボードの利用”

というコンセプトを掲げているので、実際に使えることを考慮している。そのために、第 4 章で挙げた変更例を被験者に試してもらい感想を述べてもらった。

5.1 第 4 章の変更例を試しての感想

Fig.4.1 変更例 1 , Fig.4.2 変更例 2

- キーボードに不慣れな人の場合、ローマ字入力する場合に、母音部が目立って分かりやすくなっているので扱いやすい。
- BackSpace の押したままでの削除ができず、 1 文字毎の削除になるので扱いにくい。
- 小指で shift キーを押しながらのキー操作には、多少、不便性を感じるところがあるので、 Enter と入れ換えたら良いのではないか。
- Meta キーが設定されていない。
- キーボードに不慣れな人の場合、変更例 2 は比較的、変更例 1 より、50 音順なので多少扱いやすい。

被験者の提案として、図に示したようなキーマップを求められた。
濁音は、子音の清音に割り当てられ、半濁音は使用頻度を考慮し、それぞれ当てはめる。
更に、付け加えとして

- shift + H で、NumLock とすることにより、テンキーでの操作が一層楽になる
- 他に、設定したいキーとして、「Ctrl」、「Meta」、「,」、「.」、「–」、「?」、「Space」

が挙げられた。

Fig.4.3 変更例 3 , Fig.4.4 変更例 4

- 変更例 3 の場合、キーボードに慣れている人は、文字キー B は標準の状態のままで良い。
- BackSpace キーは、確定キーではなくて、¥ キーに配置したほうが良い。
- 確定キーを変換キーに配置する。
- 数字の並びは、変更例 3 と変更例 4 どちらでも覚えてしまえば、使いやすさは一緒。
- “ー” や “?” は、日本語入力で使用頻度が高いので、配置をしたほうが良い。
- 日本語入力の場合、 O, U, I が shift + でないと出力されないので辛い。
- 変更例 4 の場合、左部分の文字を shift + で出力できるようにすれば、右手での操作が楽になる。

Fig.4.5 変更例 5

- 数字の並びはよいが、文字キーに関しては、通常入力と shift + による入力は逆でも良い。
- 左側部分で構成した変更例よりは、記号キーや、使用頻度の高いキーがまとまっているので、左右どちらの手でも扱いやすい。
- G, C, V, B と変更した記号キーは、その変更したキーと shift + で出力できたほうが良い。

5.2 実験結果

5.2.1 “Fig.4.1 変更例 1 , Fig.4.2 変更例 2” について

テンキーはコンパクトにまとまっていて、片手で使うには非常に使い勝手が良いように思われ、日本語入力する場合に、文字列の母音と子音の設定は非常に良いと思われる。だとしても、以上のことはあまりキーボードに慣れてはいない人達にとっては良いかも知れないが、既に、現在のコンピュータで使われているキーボードに慣れた人達にとっては苦痛であると思われる。何故かと言うと、キーボードに慣れた人達はキーボードの配列を指が憶えてしまっていて、また新たにキーボードの配置を憶える気はしないであろう。Ctrl + 文字キーによるショートカットキー中には多少は使えるキーもあるだろうが、 shift + で出力される文字でのショートカットは、さらに Ctrl を打たなくてはならないので扱い辛い。だとすれば、 4.3 〜 4.5 の変更例が使いやすいのではないだろうか。

この 2 つの変更例に設定してある、 BackSpace が押したままの状態で削除することができない。何故そういうことになるのか不明である。

数字は、 NumLock が BackSpace と入れ換えてしまっているため扱いづらい。

これらテンキーの、変更例は全く記号が使えないのが問題である。

5.2.2 “Fig.4.3 変更例 3 , Fig.4.4 変更例 4” について

これら変更例は、左手で扱うことを考慮していたが、実際に実験して、これはキーボードの扱いに慣れた人達に限るのかもしれないが、右手でも扱えるようである。キーボードを使い慣れている人達はキー配置を完全に記憶している。配列的には QWERT 配列のままで特にキーを変えているわけではないが、変更例 3 は、文字列を折り畳んだ状態の配置にしてしまっているので扱いづらいかもしれない。変更例 4 は、右文字列を左文字列にスライドさせた形なので、変更例 3 よりかは左手、右手でも扱いやすいかもしれない。キーボードの扱いに慣れていない人達も慣れることによって問題はないであろう。

これらの変更例もまた、 BackSpace が押したままの状態で削除することができないのである。

5.2.3 “Fig.4.5 変更例 5” について

右手で使うことを考慮していたが、実際に実験して、左手でも比較的扱いやすいと感じた。

「,」、「.」、「/」、「;」に、それぞれ G, C, V, B を配置したために、文字入力が多少、不便になってしまっている。

Fig.4.3 変更例 3 , Fig.4.4 変更例 4 と比べると、この変更式が一番良いのではないかという結果になった。

5.3 考察

5.3.1 Fig.4.1 変更例 1 , Fig.4.2 変更例 2 について

変更したことによって BackSpace が、何故、1 文字毎の削除になるかということが明らかになれば、BackSpace を押したままの状態で削除できるのである。

数字の入力は、被験者の提案のようなキーマップにすれば扱いやすくなり、テンキー全体の向上に繋がる。

記号入力に関しては難しい問題である。

5.3.2 Fig.4.3 変更例 3 , Fig.4.4 変更例 4 について

これらの変更例も先に述べたように、変更したことによって BackSpace が、何故、1 文字毎の削除になるかということが明らかになれば、BackSpace を押したままの状態で削除できるのである。

BackSpace は、確定キーに配置するよりは ¥ キーに配置し、確定キーには変換キーを配置したほうが扱いやすいようである。

日本語の入力を考えた場合に、句読点や “ー” の使用頻度が高いと言うことでどこか扱いやすい場所に配置したほうが良く、 U, I, O が shift + でないと出力されないので、これも扱いやすい配置に変更する。

変更例 4 の左部分の文字を shift + で出力できるようにすれば、右手でより扱いやすくなる。

5.3.3 Fig.4.5 変更例 5 について

この変更式はこのままでもよいが、通常入力と shift + 入力を逆にしても扱いやすいと感じ、右手での扱いが一層楽になるのではないかと思われる。

「,」、「.」、「/」、「;」は、かな部分と変更することによって問題は解消される。

6 まとめ

最終的な結果として、MISAWA ホームの CUT Key Pocket を参考にして、テンキーに割り当てたキーマップ、それを基にして少々改良したキーマップは、それなりに使い勝手は良いと感じる。自分としては後者のキーマップは気に入ったが、やはり問題なのは記号が全く打てないとうことである。英語、日本語入力での文章を書く場合には、ピリオドや句読点等が用いられ、絶対的に必要不可欠である。もう 1 つ問題として、これらキーマップは NumLock と BackSpace を入れ換えているために、テンキーでの数字入力ができなくなってしまっている。

shift, control, BackSpace キーは、テンキーに割り当てたが、Space キーは、テンキーの左隣下にある Right キーに割り当てた。このように、テンキー以外のキーに割り当てないと使えないキーが他にも出てきてしまう。

他に、3 つのキーマップに関しては、Fig.4.3 変更図 3, Fig.4.4 変更図 4 は、文字、数字、 Return, BackSpace を変更しただけで、まだまだ設定不足なところがある。Fig.4.5 変更図 5 は、文字と数字を変更しただけだが、他の 2 つよりはまだ使いやすい。

今後の課題として、テンキーでの記号入力や shift, cotrol 以外の MODIFIER キーの設定を考察する必要がある。他の 3 つのキーマップも同様である。

xmodmap を使用することによってキーボード上のマッピングは、ユーザ毎に自由に変更できるため、決まった型にこだわらず、多種多様な片手で使えるキーボードが出来ていくのではないだろうか。

最後に、この研究で作成された Fig.4.5 変更例 5 の変更式を載せる。

```
!!
!! keymap change
!!
!!
!! 42  (6): 6 ampersand kana_O kana_o
!! 37  (1): 1 exclam kana_NU
!!
keycode 42 = 1 6 kana_O kana_o
keycode 37 = ampersand exclam kana_NU
!!
!!
!! 43  (7): 7 apostrophe kana_YA kana_ya
!! 38  (2): 2 quotedbl kana_FU
!!
keycode 43 = 2 7 kana_YA kana_ya
keycode 38 = apostrophe quotedbl kana_FU
!!
!!
```

```
!! 44  (8): 8 parenleft kana_YU kana_yu
!! 39  (3): 3 numbersign kana_A kana_a
!!
keycode 44 = 3 8 kana_YU kana_yu
keycode 39 = parenleft numbersign kana_A kana_a
!!
!!
!! 45  (9): 9 parenright kana_YO kana_yo
!! 40  (4): 4 dollar kana_U kana_u
!!
keycode 45 = 4 9 kana_YO kana_yo
keycode 40 = parenright dollar kana_U kana_u
!!
!!
!! 46  (0): 0 NoSymbol kana_WA kana_WO
!! 41  (5): 5 percent kana_E kana_e
!!
keycode 46 = 5 0 kana_WA kana_WO
keycode 41 = NoSymbol percent kana_E kana_e
!!
!!
!! 66  (y): y Y kana_N
!! 61  (q): q Q kana_TA
!!
keycode 68 = q y kana_N
keycode 61 = Q Y kana_TA
!!
!!
!! 67  (u): u U kana_NA
!! 62  (w): w W kana_TE
!!
keycode 67 = w u kana_NA
keycode 62 = W U kana_TA
!!
!!
!! 68  (i): i I kana_NI
!! 63  (e): e E kana_I kana_i
!!
keycode 68 = e i kana_NI
```

```
keycode 63 = E I kana_I kana_i
!!
!!
!! 69  (o): o O kana_RA
!! 64  (r): r R kana_SU
!!
keycode 69 = r o kana_RA
keycode 64 = R O kana_SU
!!
!!
!! 70  (p): p P kana_SE
!! 65  (t): t T kana_KA
!!
keycode 70 = t p kana_SE
keycode 65 = T P kana_KA
!!
!!
!! 89  (h): h H kana_KU
!! 84  (a): a A kana_CHI
!!
keycode 89 = a h kana_KU
keycode 84 = A H kana_CHI
!!
!!
!! 90  (j): j J kana_MA
!! 85  (s): s S kana_TO
!!
keycode 90 = s j kana_MA
keycode 85 = S J kana_TO
!!
!!
!! 91  (k): k K kana_NO
!! 86  (d): d D kana_SHI
!!
keycode 91 = d k kana_NO
keycode 86 = D K kana_SHI
!!
!!
!! 92  (l): l L kana_RI
```

```
!! 87  (f): f F kana_HA
!!
keycode 92 = f l kana_RI
keycode 87 = F L kana_HA
!!
!!
!! 93  (semicolon): semicolon plus kana_RE
!! 88      (g)    : g G kana_KI
!!
keycode 93 = g G kana_RE
keycode 88 = semicolon plus kana_KI
!!
!!
!! 112 (n): n N kana_MI
!! 107 (z): z Z kana_TSU kana_tsu
!!
keycode 112 = z n kana_MI
keycode 107 = Z N kana_TSU kana_tsu
!!
!!
!! 113 (m): m M kana_MO
!! 108 (x): x X kana_SA
!!
keycode 113 = x m kana_MO
keycode 108 = X M kana_SA
!!
!!
!! 114 (comma): comma less kana_NE kana_comma
!! 109   (c)  : c C kana_SO
!!
keycode 114 = c C kana_NE kana_comma
keycode 109 = comma less kana_SO
!!
!!
!! 115 (period): period greater kana_RU kana_fullstop
!! 110   (v)   : v V kana_HI
!!
keycode 115 = v V kana_RU kana_fullstop
keycode 110 = period greater kana_HI
```

```
!!
!!
!! 116 (slash): slash question kana_ME kana_conjunctive
!! 111   (b)  : b B kana_KO
!!
keycode 116 = b B kana_ME kana_conjunctive
keycode 111 = slash question kana_KO
!!
!!
!!
```

参考文献

[1] 研究室内部向け WWW page : student 宛てに出したメール 2002/05/20 : 卒研ネタ (片手でキーボード)

[2] 松田晃一 暦本純一 : 入門 X Window (アスキー出版局,1993),pp142–151

[3] 木下凌一 小嶋和子 日高明美 : 入門 X-Window OSF/Motif Window Manager (日刊工業新聞社,1990),pp154–159

[4] MISAWA Web page : `http://www.misawa.co.jp/CUTKEY/VC201_PAGES/index/con_index1.html`

母音ゾーン
あ
い
う
え
お
や行
わ行
半母音
子音ゾーン
か行
さ行
た行
な行
は行
ら行
ま行

母音ゾーン
A- F
I P
U L
Eっ
O
YWX
子音ゾーン
KGF
SZJ
TDV
NCQ
HBP
RML

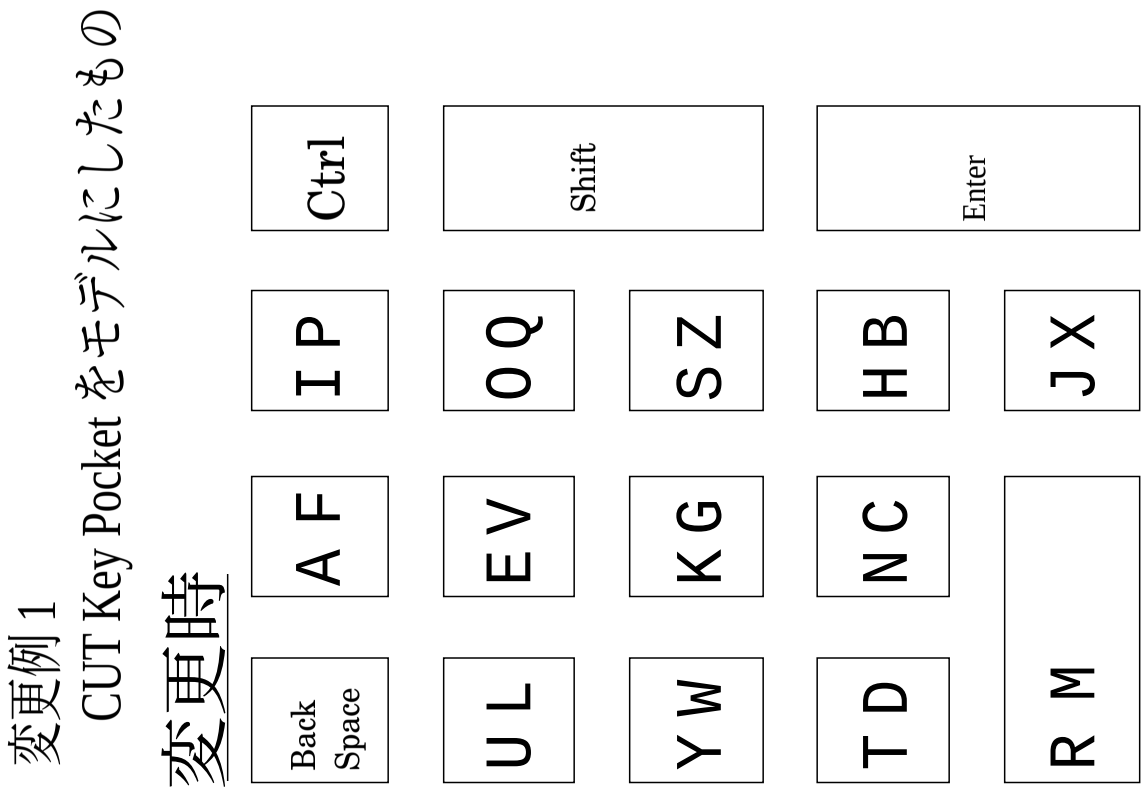

変更例 1
CUT Key Pocket をモデルにしたもの
変更時
Back Space
A F
I P
Ctrl
U L
E V
O Q
Shift
Y W
K G
S Z
T D
N C
H B
Enter
R M
J X
Space

Fig. 4.1 変更図 1

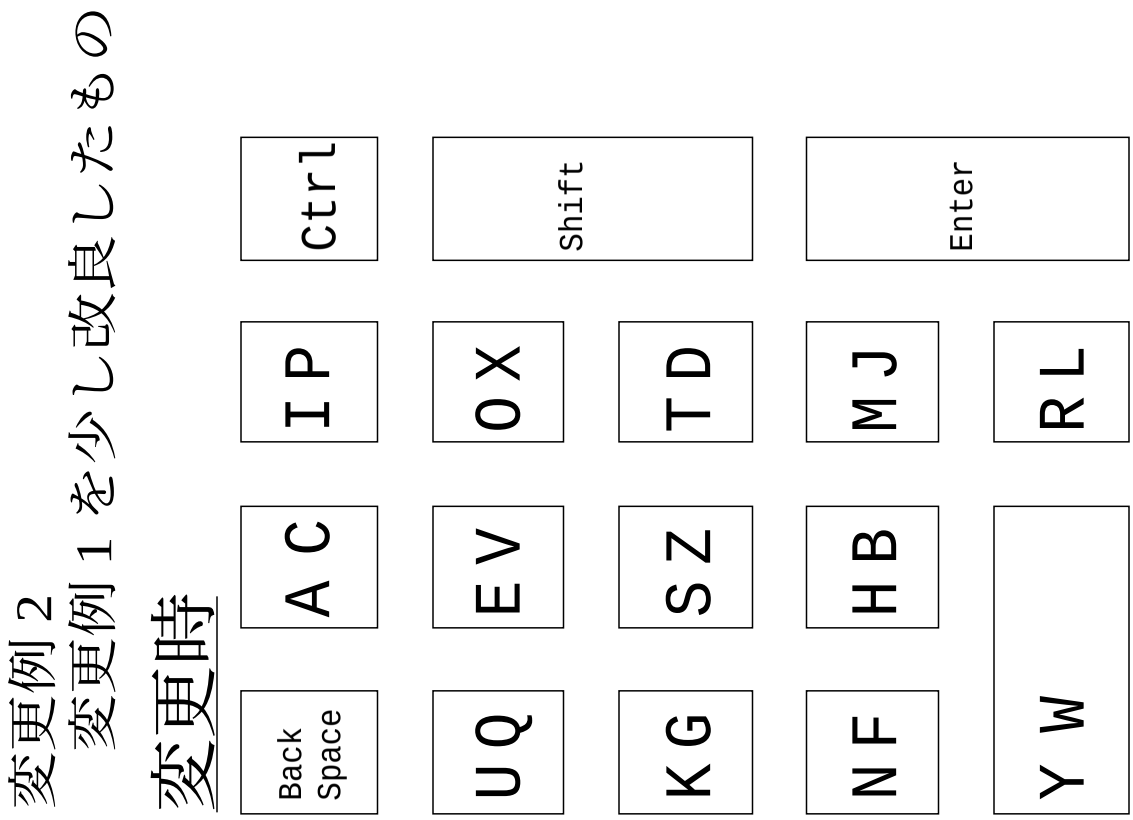

変更例 2
変更例 1 を少し改良したもの
変更時
Back Space
A C
I P
Ctrl
U Q
E V
O X
Shift
K G
S Z
T D
N F
H B
M J
Enter
Y W
R L
Space

Fig. 4.2 変更図 2

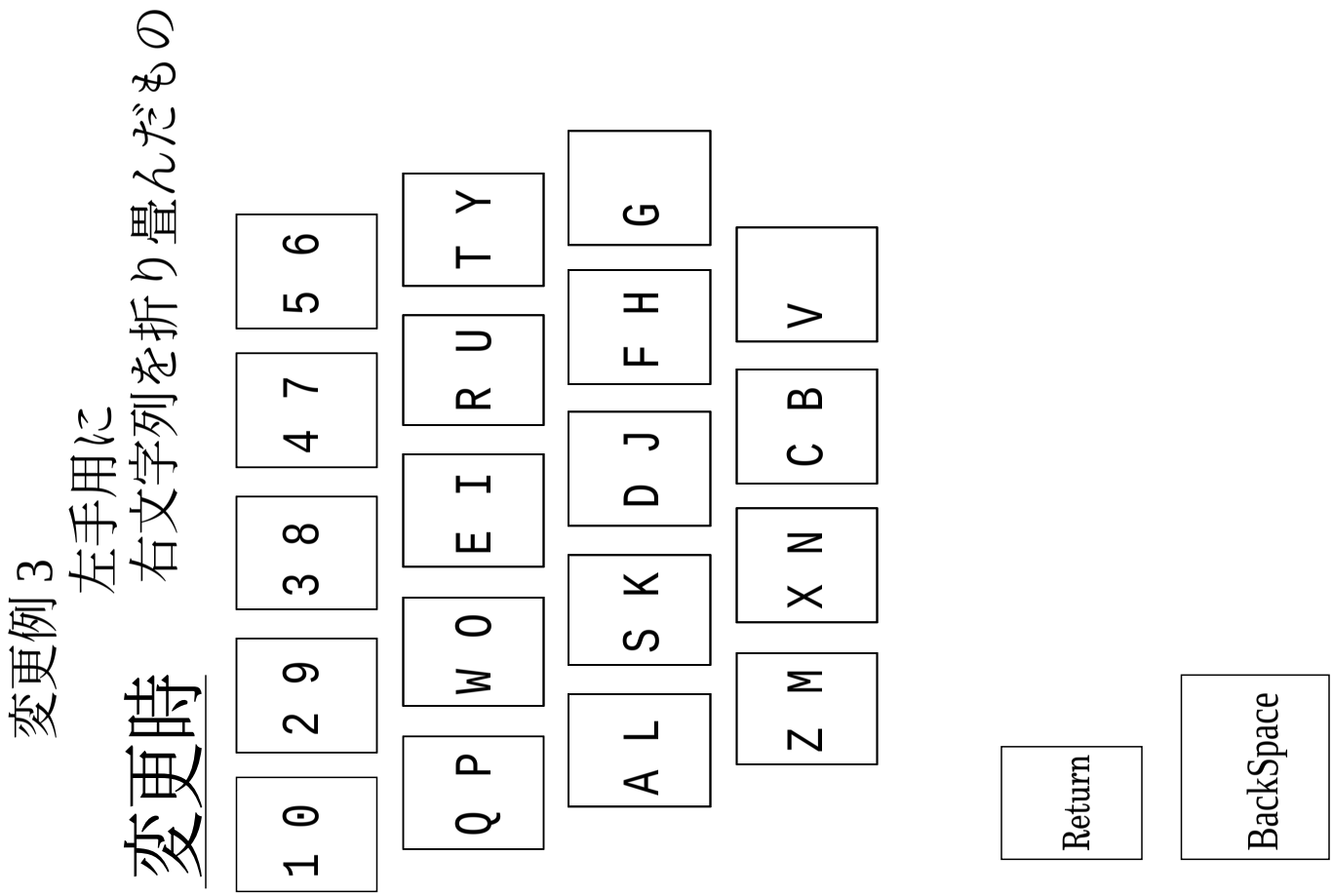

変更例 3
左手用に
右文字列を折り畳んだもの
変更時
1 0
2 9
3 8
4 7
5 6
Q P
W O
E I
R U
T Y
A L
S K
D J
F H
G
Z M
X N
C B
V
Return
BackSpace

Fig. 4.3 変更図 3

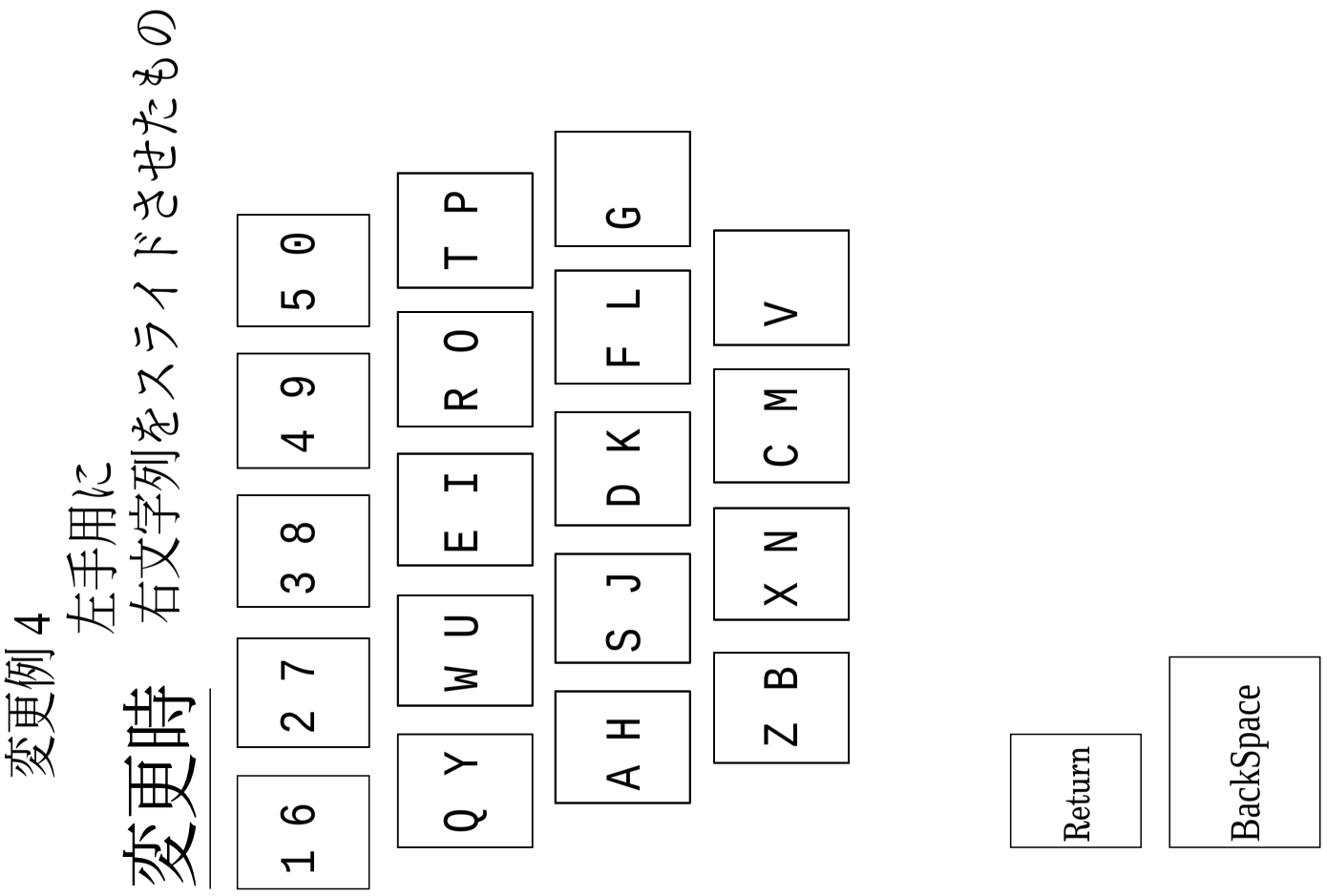

変更例 4
左手用に
右文字列をスライドさせたもの
変更時
1 6
2 7
3 8
4 9
5 0
Q Y
W U
E I
R O
T P
A H
S J
D K
F L
G
Z B
X N
C M
V
Return
BackSpace

Fig. 4.4 変更図 4

変更例 5

右手用に
左文字列からスライドさせたもの

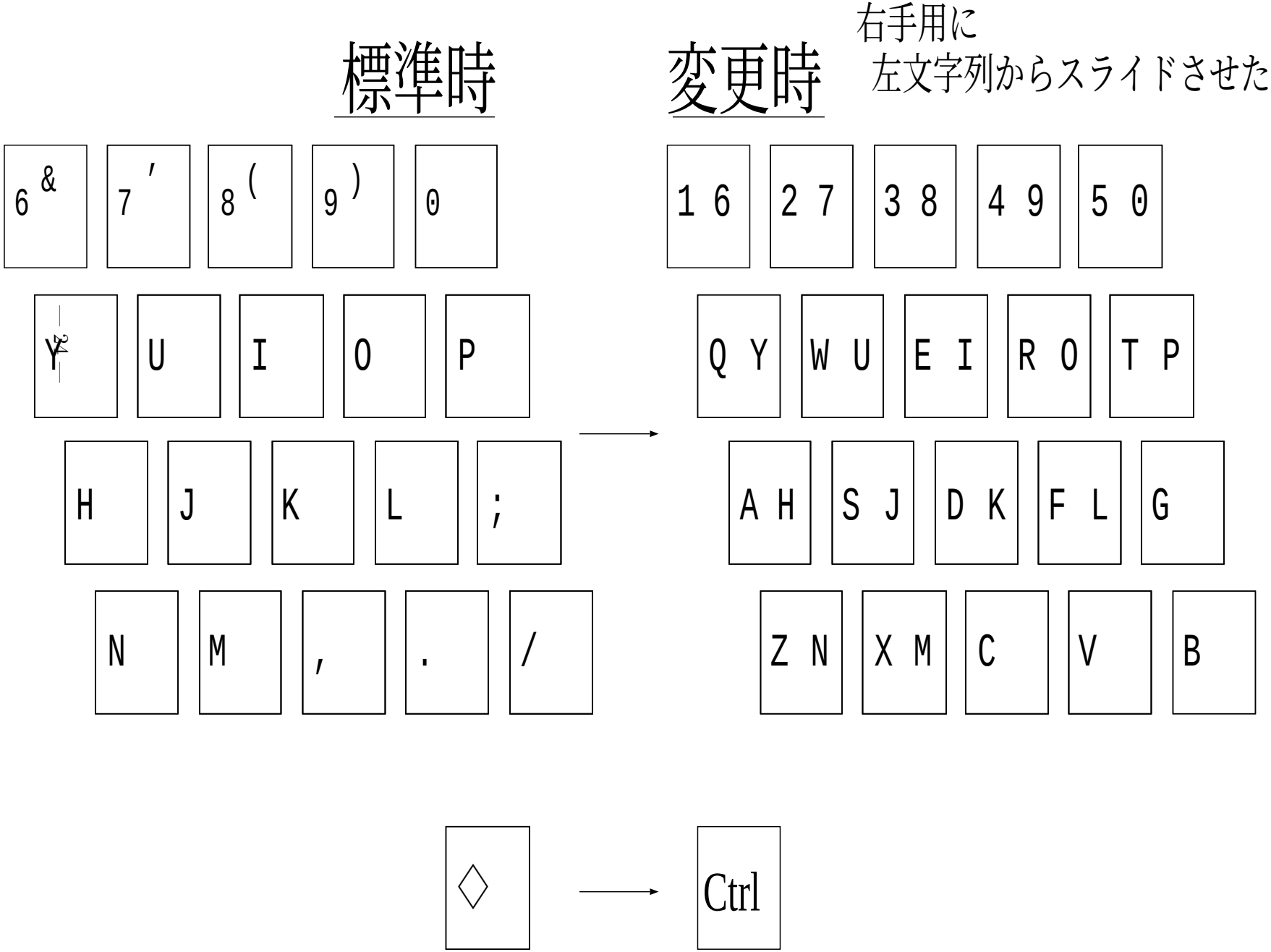
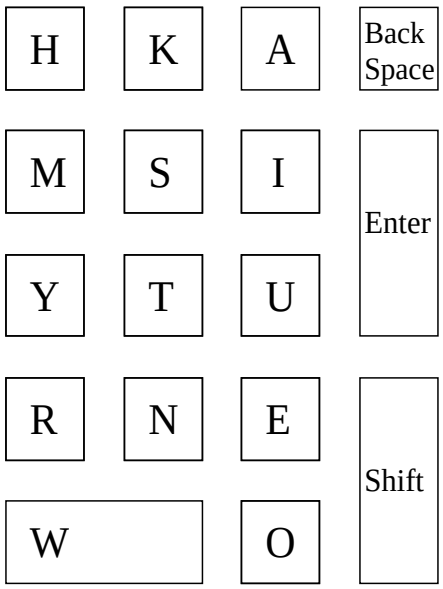


Fig. 4.5 変更図 5


H
K
A
Back Space
M
S
I
Enter
Y
T
U
R
N
E
Shift
W
O