HUMAN–COMPUTER INTERACTION, 1996, Volume 11, pp. 1–27


# One-Handed Touch Typing on a QWERTY Keyboard

**Edgar Matias**
*The Matias Corporation*

**I. Scott MacKenzie**
*University of Guelph*

**William Buxton**
*University of Toronto*

## ABSTRACT

"Half-QWERTY" is a new, one-handed typing technique designed to facilitate the transfer of two-handed touch-typing skill to the one-handed condition. It is performed on a standard keyboard with modified software or on a special half-keyboard with full-size keys. In an experiment using touch typists, hunt-and-peck typing speeds were surpassed after 3 to 4 hr of practice. Subjects reached 50% of their two-handed typing speed after about 8 hr. After 10 hr, all subjects typed between 41% and 73% of their two-handed speed, ranging from 23.8 to 42.8 words

**Edgar Matias** is a student at the University of Toronto, a member of the Input Research Group in the Department of Computer Science at the University of Toronto, and President of The Matias Corporation. **I. Scott MacKenzie** is a computer scientist whose interests include performance measurement, prediction, and modeling for human–computer interaction; he is an Assistant Professor in the Department of Computing and Information Science at the University of Guelph. **William Buxton** is a computer scientist with an interest in the human aspects of technology, input to computer systems, and collaborative work at a distance; he is an Associate Professor in the Department of Computer Science at the University of Toronto and Director of Interaction Research for Alias Research, Toronto.

CONTENTS

1. INTRODUCTION
2. HALF-QWERTY CONCEPT
    2.1. Flip Operation
    2.2. Modifier Keys
    2.3. Which Hand to Use?
    2.4. Design Space
    2.5. Hand Symmetry, Critical Invariance, and Skill Transfer
3. SKILL TRANSFER EXPERIMENT
    3.1. Method
    3.2. Results
        Temporal Analysis
        Error Analysis
        Speed Analysis
    3.3. Discussion
        Extended Sessions
        Modeling Expert Performance
        Skill Transfer Between Hands and Flip Inversion
4. DESIGN IMPLICATIONS
5. CONCLUSIONS

per minute (wpm). In extended testing, subjects achieved average one-handed speeds as high as 60 wpm and 83% of their two-handed rate. These results are important for providing access to disabled users and for designing compact computers.

## 1. INTRODUCTION

The QWERTY keyboard has been much maligned over the years. It has been called, by various authors "less than efficient" (Noyes, 1983, p. 269), "drastically suboptimal" (Gould, 1987, p. 16), "one of the worst possible arrangement[s] for touch typing" (Noyes, 1983, p. 267), "the wrong standard" (Gould, 1987, p. 23), and a "technological dinosaur" (Gopher & Raij, 1988, p. 601). Despite this, it has for various reasons (Litterick, 1981; Noyes, 1983; Potosnak, 1988) stood the test of time–a fact often overlooked by designers of alternative keyboards. Until recently, the massive skill base of QWERTY typists has been largely ignored, with new designs favoring "better" layouts. In this article, we are more conservative, preferring instead to argue that QWERTY is not an evolutionary dead end.

Our modern method of typing by touch was originally popularized by L. V. Longley and F. E. McGurrin in the latter part of the 19th century (Cooper, 1983). Curiously, despite more than 100 years of industrialization, QWERTY and the Longley and McGurrin technique remain largely unchanged. One of Longley's students would be comfortable on a modern computer keyboard, despite the alien machinery surrounding it. Similarly, we believe that this student would have little trouble acquiring the new, complementary, one-handed typing technique that we are about to propose. This article describes the new technique, with which a two-handed touch typist with very little retraining can type with one hand on a software-modified QWERTY keyboard. In effect, it is the one-handed equivalent of Longley and McGurrin's original eight-finger, two-handed typing technique. We call the technique *Half-QWERTY* because it uses only half of a QWERTY keyboard.

The present study examines the degree to which skill transfers from QWERTY to Half-QWERTY keyboards for typists already skilled in the use of a QWERTY keyboard. This was tested in an experiment using a standard keyboard for both the one-handed and two-handed conditions.

## 2. HALF-QWERTY CONCEPT[1]

Most one-handed keyboards are *chord*[2] keyboards. Half-QWERTY is not. The design builds on two principles:

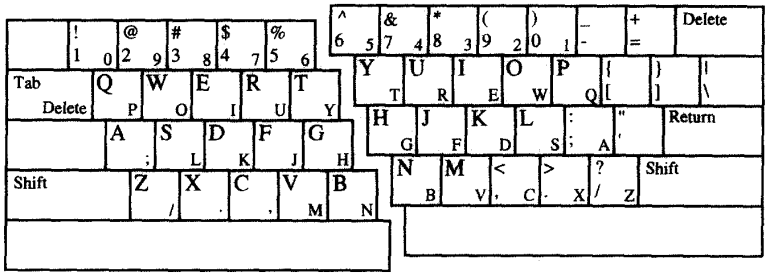
1. A user's ability to touch-type on a standard QWERTY keyboard.
2. The fact that human hands are symmetrical–one hand is a mirror image of the other–and the brain controls them as such.

A Half-QWERTY keyboard consists of all the keys used by one hand to type on a standard QWERTY keyboard, with the keys of the other hand unused or absent. When the spacebar is depressed, the missing characters are mapped onto the remaining keys in a mirror image (Figure 1), such that the typing hand makes movements homologous to those previously performed by the other hand. For example, in two-handed typing, the letter *J* is typed using the index finger of the right hand in the home row (see Figure 1, right side). Using the Half-QWERTY technique, *J* is entered with the left hand by holding down the spacebar and pressing the *F* key

1. U.S. Patent No. 5,288,158. European Patent No. 0,489,792. Australian Patent No. 647,750. Other patents pending. Half-QWERTY is a trademark of The Matias Corporation.

2. On chord keyboards, operators type by pressing one or more keys simultaneously. For example, pressing the *A* key types *A;* pressing the *B* key types *B;* pressing both keys simultaneously types some other arbitrary letter. Thus, a five-key chord keyboard can generate 31 different characters ($31 = 2^5 - 1$).

***Figure 1.*** **Left- and right-hand Half-QWERTY layouts on a standard QWERTY keyboard. When a key is depressed, the character in the upper left of the key is entered. When preceded by holding down the spacebar, the character in the lower right is entered.** ***Note:*** **Copyright © 1992 by The Matias Corporation. Used with permission.**




(index finger of the left hand in the home row; see Figure 1, left side). Notice that in both cases the index finger is in the home row to type *J*. Thus, using the spacebar as a modifier, a typist can generate the characters of either side of a full-size keyboard using only one hand. We call this mirror-image remapping of the keyboard the *flip* operation.

## 2.1. Flip Operation

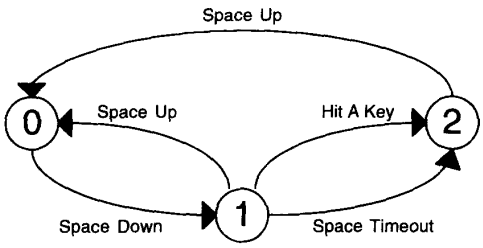
The flip operation consists of the following:

1. A spacebar capable of acting as a modifier key, in addition to its traditional role.
2. The mirror-image remapping of one half or both halves of a standard QWERTY keyboard, when the spacebar is depressed and held.

A state diagram governing the flip operation is shown in Figure 2. In State 0, the spacebar is up; in States 1 and 2, the spacebar is depressed. On a normal keyboard, depressing the spacebar generates a space character. If the spacebar is held down beyond a timeout value, space characters are generated repeatedly until the bar is released. Therefore, to generate one space, a typist depresses and releases the spacebar within a timeout value. Typing a space using Half-QWERTY works the same way. Depressing and releasing the spacebar within a timeout generates a space character.[3] In the state diagram, this corresponds to changing from State 0 to State 1 to State 0. In other words, if the spacebar is released while in State 1, a space is generated. This differs slightly from standard QWERTY. In QWERTY,

3. For this experiment, the timeout was 16/60 sec (or 267 msec).

***Figure 2.*** **Spacebar state-transition diagram. `If (state > 0) {key presses are flipped}; if (state == 1) {Space Up generates space character}`.**
***Note:*** **Copyright © 1992 by The Matias Corporation. Used with permission.**




the space is generated on the depression of the spacebar; in Half-QWERTY, it is generated on the release.

If a character key is struck while the spacebar is depressed (in State 1 or 2), that key is "flipped" (i.e., the mirror-image character is entered, and the state changes to 2–the "flip state"). While in State 2, the spacebar acts exactly like a modifier key: If a character key is struck, it is flipped; if the spacebar is released, the state returns to 0, and no space character is generated. State 2 is also the timeout state. If the user depresses the spacebar (State 1) and holds it down past the timeout value, the state changes to 2. The timeout serves to reduce the number of erroneous spaces generated as a side effect of using the spacebar as a modifier key. Occasionally, a typist depresses the spacebar with the intention of mirroring the state of another key but then changes his or her mind and releases the spacebar. Without the timeout, such actions would result in an unwanted space character. With it, the problem is alleviated.

We summarize the state diagram as follows. While in State 0 (the null state), the keyboard behaves as a QWERTY keyboard would. State 1 is ambiguous: It is not immediately clear whether a space character or the flipping of a subsequent key is desired. In State 2, the spacebar acts as a modifier key, flipping any character keys struck.

## 2.2. Modifier Keys

Modifier keys do not generate codes themselves but modify the code for a subsequent key struck while the modifier is active. Figure 3 shows the state diagram for the Shift key, as used in our experiment. If other modifier keys were implemented, they would behave in a similar manner. Odd-numbered states (1, 3, 5) indicate that the modifier key is depressed; even-numbered states (0, 2, 4) correspond to the release of the key. If the state is greater than 0, then the modifier key is active.

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.