AutoHotKey

Ein X für ein U Vormachen

Franz Fiala

Wenn auf einer Tastatur eine wichtige Taste fehlt, muss eine andere, seltener gebrauchte Taste dafür geopfert werden. Bei der Suche nach einem geeigneten Programm für den Austausch der Tastenfunktion fand ich ***AutoHotKey***, dessen Funktionalität weit über die hier beschriebenen Beispiel hinausgeht. Und gerade diese Universalität war es, die das Programm zu einem ständigen Begleiter machen.

Das Programm kann

- ***Remapping*** Tasten umbelegen
- ***Hotkeys*** definieren
- ***Hotstrings*** definieren

Ich habe für meine PCNEWS-Anwendung die Tastaturbelegung um viele Zeichen erweitert, die sonst nur über die Anwendung Zeichensatztabelle oder die Option „Einfügen Symbole“ in Word verfügbar wären. Auch eine Signatur wird als Tastenkürzel definiert.

Der wichtigste Vorteil: diese Tastenkürzel funktionieren in jedem Programm. Man kann eine definierte Signatur sowohl im Mailer als auch im Textverarbeitungsprogramm oder im Publisher einsetzen.

Eine persönliche Tastaturbelegung kann auf jedem anderen PC verwendet werden. Da jedes Skript in eine portable EXE-Datei verwandelt werden kann, muss AutoHotKey auf den anderen Rechnern nicht installiert sein.

Aber zurück zum Anlassfall, der IBM-Tastatur:

Man muss eine bestehende Taste der Tastatur umdefinieren, damit sie die Funktion der Windows-Taste übernehmen kann; eine Taste, die entbehrlich ist. Da auf der IBM-Tastatur zwei Strg-Tasten sind (eine linke und eine rechte), habe ich mich entschlossen, diese Taste zu „opfern“.

Wie funktioniert nun diese Umdefinition?

- Installation der Anwendung AutoHotKey
- Textdatei am Desktop mit dem Namen win.ahk. Das Dateisymbol ändert sich von Text auf AutoHotKey.
- In diese Textdatei schreibt man die folgende Zeile.
  RControl::Lwin
- Mit der rechten Maustaste öffnet man das Kontextmenü und startet das Skript.
- In der Status-Zeile erscheint für jedes AutoHotKey-Skript ein Symbol, mit dem man das Skript anhalten bedienen und auch wieder entladen kann.

Das wär's auch schon gewesen. Damit wird die entbehrliche Taste Strg-rechts zur Windows-Taste und die Tastatur ist auch in Windows 8 verwendbar.

Die sehr umfangreiche Dokumentation zu AutoHotkey machte mich neugierig und die vielen weiteren Möglichkeiten haben mich angeregt, diese Möglichkeiten auszunutzen. Das Programm hat sich als so praktisch erwiesen, dass ich es allen „Tastaturumbelegern“ empfehlen kann.

Der Grund...

...für diesen Artikel

Franz Fiala

Der Anlass für diesen Artikel war ein sehr originelles und gleichzeitig auch nützliches Geschenk. Der Redakteur der PCNEWS bekam von ***Günter Hartl*** eine Original IBM-Tastatur, Baujahr 1989, ca. zwei kg, großer Tastenhub, praktisch ungebraucht. Verglichen mit den modernen Kurzhubtastaturen ist das Vertipp-Risiko bei dieser Tastatur viel geringer.

Zwei kleine Nachteile hatte die Tastatur:

(1) Pegelprobleme

Einfach an einen USB-Port oder an die Tastaturbuchse anschließen funktionierte nur bedingt. Wahrscheinlich auf Grund von Pegelunterschieden zwischen den alten TTL-Logikbausteinen in der Tastatur und den modernen Interfaces kam es oft zu einer wiederholten Auslösung einer Taste, die nur durch Ab- und Anstecken wieder verschwunden ist.

Für die Lösung dieses Problems musste ich ein Kabel mit einem speziellen Interface verwenden, das man bei einem Entwickler in den USA bestellen kann.

(2) Tastaturlayout

Auf der IBM-Tastatur fehle die Windows-Taste und die Taste für das Kontext-Menü.

Bis Windows 7 war die Windows-Taste nicht zwingend aber mit Windows 8 braucht man diese Taste, um vom Desktop wieder zum Kachel-Menü zu kommen. (Es geht auch mit der Maus, aber dass eine so wichtige Funktion nicht tastaturbedienbar ist, wäre doch störend. Auch das wäre allein noch nicht das Problem gewesen, denn für die Windows-Taste gibt die Ersatz-Tastenkombination Strg-Esc. Aber diese Kombination substituiert zwar die Windows-Taste aber wenn es gilt, eine Kombination aus Win-Taste und einer weiteren Taste zu ersetzen, dann ist Endstation.

Dieses Problem, einen Ersatz für die Windows-Taste zu finden, kann von mehreren kleinen Tools gelöst werden. Aber keines dieser ausprobierten Tools ist so universell einsetzbar wie das hier vorgestellte ***AutoHotKey***. Und daher kam es zu diesem Artikel.

Der Grund...

...für den Siegeszug des IBM-PC

Franz Fiala

Für den Erfolg von Microsoft werden viele Gründe genannt. Und man darf keinen weglassen, sonst wäre das alles nicht passiert über das wir uns heute manchmal auch ärgern aber das wir im Großen und Ganzen längst akzeptiert haben.

Der PC hatte gegenüber dem Mitbewerb den großen Vorteil, dass sich Software, besonders Spiele in großer Zahl auf genau dieser Hardware etabliert hat. Was hätte Bill Gates schon der Vertrag mit IBM genutzt, wenn die Spieleprogrammierer andere Geräte genutzt hätten.

Erst alle diese Zufälligkeiten zusammen ergaben den Siegeszug des PC und von MSDOS.

Ein Grund dafür war die Umgehung des BIOS und die direkte Ansprache des Bildschirmspeichers. Dadurch konnte man Spiele einigermaßen schnell programmieren.

Der zweite Grund war die Art der Tastatur. Der Mitbewerb (zum Beispiel der DEC-Rainbow aber auch andere) verwendete eine Terminal-Tastatur als Eingabegerät, also eine selbständige Hardware, die ASCII-Zeichen produziert. Die Tastatur entscheidet, wann ein Zeichen gesendet wird. Das Drücken einer Sondertaste allein Shift, Strg bewirkte nichts, weil erst nach dem Drücken des alphanummerischen Zeichens feststand, ob ein Steuerzeichen, ein Kleinbuchstabe oder ein Großbuchstabe gesendet wird. Die Taste Alt gab es nicht.

Die IBM-Tastatur war aber ein Stück Hardware, das ohne den PC keinen Sinn ergab, denn die Tastatur sendete pro Tastendruck zwei Zeichen, eines beim Drücken und eines beim Loslassen der Taste. (Make- und Break-Kode). Und zwar jede Taste, auch Shift, Strg und Alt. Und das ist auch heute noch so.

Dieser Umstand war ein Eldorado für Spieleprogrammierer, denn sie konnten jede einzelne Taste zur Spielbeeinflussung verwenden und konnten auch die Schnelligkeit der Reaktion als Information heranziehen.

Also der schnelle Zugriff auf das Bild war wichtig. Aber fast noch wichtiger war diese sehr informative Tastatur, die viel über die Bedienung durch einen Spieler sagen konnte.

IBM Modell M, 1989, 101 Tasten, oben und ihre Vorläufer die IBM-AT-Tastatur, 84 Tasten links unten und die IBM-Terminaltastatur, 122 Tasten, rechts unten

Maustasten	
LButton	Linke Maustaste
RButton	Rechte Maustaste
MButton	Mittlere Maustaste oder Mausradtaste

Maustasten Erweitert	
XButton1	Vierte Maustaste (= Browser_Back)
XButton2	Fünfte Maustaste (=Browser_Forward)

Mausrad	
WheelDown	Mausrad nach unten drehen (zum Benutzer hin)
WheelUp	Mausrad nach oben drehen (vom Benutzer weg)
WheelLeft WheelRight	Mausrad nach links oder rechts drehen

Multimedia	
Browser_Back	Zurück
Browser_Forward	Vorwärts
Browser_Refresh	Aktualisieren
Browser_Stop	Stopp
Browser_Search	Suchen
Browser_Favorites	Favoriten
Browser_Home	Startseite
Volume_Mute	Lautstärke stummschalten
Volume_Down	Lautstärke verringern
Volume_Up	Lautstärke erhöhen
Media_Next	Nächster Track
Media_Prev	Vorheriger Track
Media_Stop	Stopp
Media_Play_Pause	Wiedergabe/Anhalten
Launch_Mail	E-Mail-Programm öffnen
Launch_Media	Media Player öffnen
Launch_App1	Arbeitsplatz öffnen
Launch_App2	Taschenrechner öffnen

Sondertasten	
AppsKey	MENÜ-Taste.
PrintScreen	DRUCK-Taste
CtrlBreak	
Pause	PAUSE-Taste
Break	BREAK-Taste
Help	HELP-Taste = F1
Sleep	SLEEP-Taste
SCnnn	Ersetze nnn mit dem Scancode einer Taste.
VKnn	Ersetze nn mit dem hexadezimalen virtuellen Code einer Taste.

Allgemein	
CapsLock	FESTSTELL-Taste
Space	LEERTASTE
Tab	TABULATOR-Taste
Enter	ENTER-Taste
Escape (Esc)	ESC-Taste
Backspace (BS)	RÜCKTASTE

Cursor-Steuerung	
ScrollLock	ROLLEN-Taste
Delete Del	ENTF-Taste
Insert Ins	EINFG-Taste
Home	POS1-Taste
End	ENDE-Taste
PgUp	BILD-NACH-OBEN-Taste
PgDn	BILD-NACH-UNTEN-Taste
Up	NACH-OBEN-Pfeiltaste
Down	NACH-UNTEN-Pfeiltaste
Left	NACH-LINKS-Pfeiltaste
Right	NACH-RECHTS-Pfeiltaste

Ziffernblock	
NumLock AN	NumLock AUS
Numpad0	NumpadIns
Numpad1	NumpadEnd
Numpad2	NumpadDown
Numpad3	NumpadPgDn
Numpad4	NumpadLeft
Numpad5	NumpadClear
Numpad6	NumpadRight
Numpad7	NumpadHome
Numpad8	NumpadUp
Numpad9	NumpadPgUp
NumpadDot	NumpadDel
NumpadDiv	NumpadDiv
NumpadMult	NumpadMult
NumpadAdd	NumpadAdd
NumpadSub	NumpadSub
NumpadEnter	NumpadEnter

Funktion	
F1 - F24	Funktionstasten

Modifikator	
LWin	Linke WIN-Taste
RWin	Rechte WIN-Taste
Control Ctrl	STRG-Taste
Alt	ALT-Taste
Shift	UMSCHALT-Taste
LControl LCtrl	Linke STRG-Taste
RControl RCtrl	Rechte STRG-Taste
LShift	Linke UMSCHALT-Taste
RShift	Rechte UMSCHALT-Taste
LAlt	Linke ALT-Taste.
RAlt	Rechte ALT-Taste.

Symbol	
#	WIN-Taste (Windows-Logo-Taste).
!	ALT-Taste
^	STRG-Taste
+	UMSCHALT-Taste
&	Ein Und-Zeichen kann zwischen zwei beliebigen Tasten oder Maustasten verwendet werden, um diese zu einem benutzerdefinierten Hotkey zu verbinden. Siehe unten für weitere Details.
<	Verwendet die linke Version der angegebenen Modifikatortaste. Zum Beispiel ist **<!a** das gleiche wie **!a**, außer dass die linke ALT-Taste zum Auslösen des Hotkeys benötigt wird.
>	Verwendet die rechte Version der angegebenen Modifikatortaste.
<^>!	ALTGR-Taste (alternativer Schriftsatz). Hat deine Tastaturbelegung die ALTGR-Taste anstelle einer rechten ALT-Taste, kann diese Symbolreihe als ALTGR verwendet werden. Zum Beispiel: **<^>!m::MsgBox Sie haben ALTGR+M gedrückt**.
*	Platzhalter: Führt den Hotkey auch dann aus, wenn zusätzliche Modifikatortasten gedrückt gehalten werden. Das wird oft in Verbindung mit der Neubelegung von Tasten verwendet. Zum Beispiel: ***#c::Run Calc.exe** ; WIN+C, UMSCHALT+WIN+C und STRG+WIN+C usw. werden den folgenden Hotkey auslösen. ***ScrollLock::Run Notepad** ; Die ROLLEN-Taste löst den folgenden Hotkey aus, selbst wenn Modifikatortasten gedrückt gehalten werden.
~	Beim Drücken des Hotkeys wird seine ursprüngliche Funktion nicht blockiert (bzw. nicht im System versteckt). In den beiden folgenden Beispielen wird der Mausklick des Benutzers weiterhin an das aktive Fenster gesendet: **~RButton::MsgBox Sie haben die rechte Maustaste gedrückt**. **~RButton & C::MsgBox Sie haben C gedrückt, während die rechte Maustaste gedrückt wird**.
$	Dieses Präfix ist normalerweise nur notwendig, wenn das Script den Send-Befehl verwendet, um Tasten zu senden, die den Hotkey selbst enthalten, der sich ansonsten selbst auslösen würde. Das **$**-Präfix erzwingt den Tastatur-Hook zum Implementieren des Hotkeys, wodurch als Nebeneffekt der Send-Befehl den Hotkey nicht mehr auslösen kann. Dieser Präfix ist das gleiche wie, als hätte man **#UseHook** irgendwo vor der Hotkey-Definition angegeben.
UP	Das Wort **UP** kann nach dem Namen eines Hotkeys erfolgen, um den Hotkey dazu zubringen, beim Loslassen ausgeführt zu werden, anstatt beim Drücken. Im folgenden Beispiel wird die linke WIN-Taste mit der linken STRG-Taste neu belegt: ***LWin::Send {LControl Down}** ***LWin Up::Send {LControl Up}**

Was kann AutoHokey?

AutoHotKey hat drei Grundfunktionen mit sehr ähnlicher Syntax:

Hotkeys

Einer Taste oder einer Tastenkombination eine bestimmte Funktion zuweisen (Programm aufrufen). Man kann mit einem Hotkey auch mehrere Programme ausführen.

Syntax
```
<Tasten>::Run <Programm>
```

Remapping

Tasten umkodieren. So, wie beim ersten Beispiel gezeigt, wird einer Taste oder einer Tastenkombination ein anderer Wert zugewiesen. Das funktioniert auch mit Maustasten oder Joystick-Tasten

Syntax
```
<Taste(n)>::Send <Tasten>
<Taste(n)>::<Taste>
```

Hotstrings

Gemeint ist die Substitution von Abkürzungen durch einen Volltext, zum Beispiel *„mfg."* durch *„mit freundlichen Grüßen"*

Syntax
```
::<Kürzel>::<Text>
```

Da aber ***AutoHotKey*** neben diesem Kürzelsystem auch über eine eigene Programmiersprache ***AHK*** mit Schnittstellen zum Dateisystem, zur Registry usw. verfügt, kann man es zum Programmieren verschiedenster Problemstellungen verwenden.

Die entstehenden Skripts können entweder als unabhängige Skripts oder auch als ein gemeinsames Skript in einer Datei ausgeführt werden.

Skripts können in eine Exe-Datei kompiliert werden und sind danach portabel und nicht mehr von der Installation von ***AutoHotKey*** auf einem Rechner abhängig.

Die Möglichkeiten sind derart vielfältig, dass man nur raten kann, in der deutschspracheigen Hilfe-Datei zu blättern und sich von den zahlreichen Beispielen, auch im Forum anregen zu lassen.

Wie arbeitet AutoHotKey?

Wie das Programm eine Tastatur beeinflusst, bestimmen kleine Konfigurationsdateien (***AHK***-Skripts), die als Parameter an das Programm übergeben werden.

Jeder Hotkey, jedes Mapping und jeder Hotstring kann in einer eigenen Datei mit der Endung **.ahk** gespeichert werden oder man kann auch alle oder einige in einer einzigen Datei zusammenfassen; einfach innerhalb einer AHK-Datei aneinanderreihen.

Es können beliebig viele solcher Skripte geladen und wieder entfernt werden ohne den PC neu starten zu müssen

Die so definierten Tastenkürzel oder auch Textbausteine funktionieren in allen Programmen.

Man kann beliebig viele solcher Skripte laden. Jedes Skript äußert sich in einem Symbol im Infobereich rechts in der Taskleiste. Man kann jedes Skript temporär ausschalten oder entladen. Das Testen der Skripts in daher überaus einfach.

Arbeitstechnik

Um ein Skript (Textdatei mit Endung **.ahk**) zu laden, klickt man mit der rechten Maustaste darauf. Im Kontextmenü finden sich gleich am Anfang drei Einträge, die AHK-Dateien betreffen:

- *Run Script*
- *Compile Script*
- *Edit Script*

Run Script lädt das Script und fügt ein Symbol in die Taskleiste ein. *Compile Script* erzeugt eine Exe-Datei und Edit Script öffnet das Skript im Editor.

In der Taskleiste werden nun so viele AHK-Symbole angezeigt als Skripts geladen worden sind. Man kann sie durch Überfahren mit der Maus unterscheiden, weil sie dann in einer Sprechblase den Namen bekanntgeben. Über einen Klick mit der rechten Maustaste sieht man die Menüpunkte *Open, Help, Window Spy, Reload, Edit, Suspend, Pause und Exit*. *Help* öffnet eine ***AHK***-Hilfedatei, *Window Spy* gibt einige Daten zum aktuellen Fenster und zur Mausposition bekannt, mit *Reload* kann das Skript neu geladen werden (etwa, wenn man es im Editor korrigiert hat), mit *Edit* öffnet man das Skript im Editor, mit *Suspend* wird das Skript temporär ausgeschaltet, mit *Pause* wird das aktuelle Skript angehalten und mit *Exit* wird das Skript entladen.

Das Testen eines Skript verläuft also so, dass man den Text des Skripts im Editor verändert und dann in der Statusleiste beim Symbol für dieses Skript den Menüpunkt *Reload* aktiviert. Keine Registry-Einträge, kein Reboot des Rechners erforderlich.

Wenn man intensiv testet, ist sogar der Griff zur Tastleiste lästig. Man kann dann an den Anfang des Skripts einige Kodezeilen einfügen und das Skript neu starten.

```
SetTimer,UPDATEDSCRIPT,1000
UPDATEDSCRIPT:
FileGetAttrib,attribs,%A_ScriptFullPath%
IfInString,attribs,A
{
FileSetAttrib,-A,%A_ScriptFullPath%
SplashTextOn,,,Script aktualisiert,
Sleep,500
Reload
}
Return
```

Diese Kodezeilen sorgen dafür, dass das Skript bei jedem Speichern des Textes automatisch neu geladen wird.

Die Sprache *AHK*

Zunächst muss man wissen, dass in ***AHK***, der Sprache von ***AutoHotKey*** jede Taste am PC und auch jede Aktion der Maus durch ein reserviertes Wort repräsentiert wird. **Space** ist die Leertaste, **PgUp** ist die BildNachOben-Taste, usw. Die vollständige Liste: findet sich auf der vorigen Seite.

Alphazeichen, Zahlen und Zeichen brauchen meist keinen eigenen Namen. Zum Beispiel: **b** ist die B-Taste und **5** die 5-Taste.

Jetzt muss man noch wissen, dass die Umschaltetasten in ***AutoHotKey*** der Einfachheit halber abgekürzt werden mit

Taste	***AHK***-Kürzel
⇧	+
Strg	^
Alt	!
AltGr	<^>!
⊞	#

Weitere Sprachelemente werden in den Beispielen vorgestellt. Eine Gesamtübersicht über die Sprache findet sich in der Hilfedatei zu ***AutoHotKey***.

Bevor wir daran gehen, Hotkeys und Remapping zu implementieren, müssen wir uns eine Übersicht verschaffen, welche Möglichkeiten dazu überhaupt bestehen. Es muss ja sichergestellt werden, dass ein Hotkey oder auch ein Remapping möglichst nicht mit einem bereits in einem Programm definierten Hotkey konkurriert.

Wie viele verschiedene Kodes kann man einer Taste zuordnen?

Verwenden wir als Beispiel die Taste Q. Diese Taste kann mit jeweils einer von fünf Umschaltetasten sechs Zeichen generieren.

Taste	Zeichen
Q	q
⇧ Q	Q
Strg Q	Strg-q
Alt Q	<unbelegt>
AltGr Q	@
⊞ Q	Suchmenü

Zwar sendet eine Tastatur in allen diesen Fällen für die Taste Q selbst nur einen Kode aber durch das gemeinsame Drücken mit den jeweiligen Umschaltetasten, die auch einen Kode senden, ergibt sich immer ein anderes resultierendes Zeichen.

Aber wie viele Zeichen könnte die Taste Q insgesamt erzeugen?

Da es fünf Umschaltetasten gibt: ⇧ Strg Alt AltGr und ⊞ und dazu auch die grundlegende Möglichkeit keine Umschaltetaste zu benutzen, kann man mit einer Taste 2^6=64 Zeichen generieren.

Es gibt etwa 64 Zeichen-Tasten. Daher kann eine Tastatur etwa 4096 Zeichen generieren. Theoretisch, denn wer wird schon gerne das Zeichen über die Tastatur erzeugen wollen, das sich aus der Tastenkombination ⇧ Strg Alt AltGr ⊞ Q ergibt.

Wie würde man also einer Taste über Hotkey diese verschiedenen Möglichkeiten verpassen? Welche Tastenkombination soll man wählen?

Die meisten Shortcuts nutzen aus praktischen Gründen auch nur ein Umschaltzeichen, manchmal auch zwei, selten drei aber nicht mehr Kombinationen aus.

Man ist ja nicht allein am PC, da gibt es auch das Betriebssystem, das uns viele Tastenkürzel vorgibt und jedes aktive Programm beansprucht auch noch einen Satz von Tastenkürzel.

Wenn also das eigene Tastenkürzel systemweit Gültigkeit haben soll, darf es nicht mit den bereits definierten Kürzel konkurrieren.

Das Betriebssystem konzentriert sich auf Tastenkombinationen mit der Windows-Taste. **(siehe umseitige Tabelle)**

Mehrere Programme mit einem Hotkey ausführen

```
RAlt & c::
{
   Run, c:\Programme\Mozilla Firefox\firefox.exe
   Run, c:\Programme\Mozilla Sunbird\sunbird.exe
   Run, c:\Programme\Mozilla Thunderbird\thunderbird.exe
}
return
```

In den Anwendungen kommen dann weitere Tastenkürzel dazu aber meist immer nur in einfachen Kombinationen einer Umschaltetaste und einem Zeichen.

Hotkeys

Ein Hotkey ist die Zuordnung eines Befehls zu einem bestimmten Tastendruck.

In einer AHK-Datei wird das durch das Zeichen :: ausgedrückt. Links von :: steht der Hotkey und rechts davon der Ausführungsbefehl.

Vor dem Abschluss der Installation wird vorgeschlagen, ein Demo-Skript zu installieren. Dieses Demo-Skript generiert zwei Hotkeys:

```
#z::Run www.autohotkey.com
^!n::
IfWinExist Untitled - Notepad
 WinActivate
else
 Run Notepad
return
```

Erste Zeile

Win Z : Öffnet die (alte) Homepage von ***AutoHotKey*** (**#** ist das Kürzel für die Windows-Taste)

Zweite Zeile

Strg Alt n : öffnet einen Editor (Strg=^, Alt=**!** Und **n** ist einfach die Taste n . Danach folgt ein Kodeblock, der mit **return** abgeschlossen wird.

Dieser Kodeblock hat den Zweck, dass bei nochmaligem Drücken der Taste sich nicht noch einmal dasselbe Fenster öffnet. Wenn es also nichts ausmacht oder sogar gewünscht ist, dass mehrere Instanzen eines Programms geöffnet werden können, genügt eine Zeile, sonst verwendet man das vorgestellte Konstrukt.

Folgt man den diversen Tipps in Foren, kann man dieses Verhalten auch durch eine einmalige Einstellung am Beginn des Skripts einstellen:

```
#SingleInstance, Force
```

Mit dieser Zeile am Beginn des Skript ist das Programmkonstrukt nicht notwendig und jedes Programm kann nur in einer Instanz aufgerufen werden.

Die Zeile

```
IfWinExist Untitled - Notepad
```

muss bei einer deutschen Windows-Installation ersetzt werden durch

```
IfWinExist Unbenannt - Editor
```

Hinweis: Hat man Evernote installiert, funktioniert die zweite Tastenkombination nicht, denn die ist in Evernote bereits für eine „Neue Notiz" belegt. Mit Hotkeys kann man also häufig benötigte Programme aufrufen.

Mehr noch, man kann auch mit einem Hotkey beliebig viele Programme aufrufen. Ein Beispiel dazu, siehe Kasten vorige Seite unten. Die benötigten Aufrufe werden in getrennte Zeilen geschrieben und durch geschwungene Klammern zusammengefasst, das Ganze mit **return** abgeschlossen. Aufgerufen wird der Multi-Aufruf mit dem Hotkey AltGr c .

Shortcuts AHK

Win Shift a	Access
Win Shift b	Bildschirmtastatur
Win Shift c	Chrome
Win Shift d	OneDrive
Win Shift e	Excel
Win Shift f	FileZilla
Win Shift g	Gimp
Win Shift i	InkScape
Win Shift j	IrfanView
Win Shift n	Notepad
Win Shift Alt o	Notepad++
Win Shift p	Powerpoint
Win Shift r	Rechner
Win Shift s	Snipping Tool
Win Shift t	TotalCommander
Win Shift u	Publisher
–	
Win Shift w	Word
Win Shift x	OneNote
Win Shift y	OneDrive (lokal)
Win Shift z	Zeichentabelle

Shortcuts Windows 8.1 1(3)

Win	Start
Win F1	Windows Hilfe
Win Maus-Rad	Zoomen im Start-Menü
Win → / Win ←	Fenster rechts/links einrasten
Win ↑ / Win Shift ↑	Fenster voll/vertikal einrasten
Win ↓ / Win Shift ↓	Fenster voll/vertikal wiederherstellen
Win Pos1	Alle Fenster minimieren
Win SP	Sprachumschaltung
Win Strg SP	Vorige Eingabe
Win Tab / Win Shift Tab	Nächste/Vorige App
Win 0...9	Anwendung starten
Win Shift 0...9	Anwendung neu
Win Strg 0...9	Anwendung n starten
Win Alt 0...9	Anwendung n Links
Win Shift Strg 0...9	Anwendung n neu
Win A	<nicht belegt>
Win B / Win Strg B	Fokus auf Taskbar /App mit Meldung
Win C	Charms anzeigen
Win D	Desktop zeigen/verstecken
Win E	Windows-Explorer
Win F / Win Strg F	Suche Dateien/Computer
Win G	<nicht belegt>
Win H	Teilen-Menü
Win I	Einstellungsmenü
Win J	<nicht belegt>
Win K	Geräte-Menü
Win L	Sperrt das System
Win M / Win Shift M	Fenster (Taskleiste) minimieren/maximieren
Win N	<nicht belegt>
Win O	<nicht belegt>
Win P	Projektor Optionen
Win Q	Suche Überall/in App
Win R	Ausführen-Dialog
Win S	Suche Windows/Internet
Win T	Tasks
Win U	Bedienungshilfen
Win V	<nicht belegt>
Win W	Sucheinstellungen
Win X	Admin Tools
Win Y	<nicht belegt>
Win Z	App Leiste anzeigen
Win . / Win Shift .	Apps umschalten
Win ,	Desktop anzeigen (kurz)
Win +	Bildschirmlupe ein
Win Esc	Bildschirmlupe aus
Win Enter	Startet die Sprachausgabe
Win Pause	Systeminformationen
PrintScreen	Bildschirmfoto in Zwischenablage
Win PrintScreen	Bildschirmfoto in Bibliothek
Alt PrintScreen	Bildschirmfoto des aktuellen Fensters

Für eigene Hotkeys bietet sich daher an, zwei Umschaltetasten und ein Zeichen zu verwenden.

Bei mir ist zum Beispiel das Programm Microsoft Access häufig in Verwendung. Die Tastenkombination Win Shift a könnte daher Access aufrufen. Das zugehörigt ***AHK***-Kommando lautet.

```
#+a::Run MsAccess
```

Die Raute ist die Windows-Taste, das Pluszeichen ist die Shift-Taste. **Run** führt das nachfolgend angegebene Programm auf.

Man kann nun weitere Tastenkürzel zum Schnellstart von Programmen definieren, wie zum Beispiel im Kasten oben gezeigt wird. Damit man sich später auf die Hotkeys erinnert, kann man sich eine Tabelle dafür anfertigen.

Remapping

Das Remapping ist den Hotkeys syntaktisch sehr ähnlich. Der Unterschied ist, dass auf der rechten Seite statt einem Befehl eine weitere Tastenkombination steht.

Um einen Hotkey zu definieren, verwendet man das Symbol **::**

Links von **::** steht die Taste, die betätigt wird und rechts von **::** steht die Aktion, die ausgeführt werden soll. Diese Aktion kann nun wieder eine Taste sein, dann erfolgt ein ***Mapping***, es kann aber auch ein Programmaufruf sein, dann ist es ein ***Hotkey***.

In meiner ursprünglichen Aufgabenstellung ging es darum, aus der rechten Steuerungs-Taste die linke Windows-Taste zu machen. Daher lautet das ***AHK***-Kommando

```
RControl::Lwin
```

Shortcuts Windows 8.1 2(3)

F1	Hilfe (kontextsensitiv)
⊞ F1	Windows-Hilfe
F2	Zelle editieren (Excel)
F3	Suchen (kontextsensitiv)
F4	Adressleiste anzeigen (Explorer)
Alt F4	Fenster schließen, Windows beenden
Strg F4	Tab schließen
F5 / Strg r	Aktualisieren (Browser)
F6	Zwischen Objekten umschalten
F7	
F8	Abgesicherter Modus (Windows Start)
F9	
F10	Menüleiste aktivieren (aktive App)
⇧ F10	Kontextmenü (rechte Maustaste)
F11	Vollbildmodus ein/aus
F12	

Damit ist diese einfache Aufgabe auch schon erledigt.

Doch kann man dieses Feature des Remapping- auch ein bisschen ausbauen und die Tastatur intensiver mit selteneren Zeichen belegen.

Mapping-Versuche

Tasten vertauschen

Die einfachste Umbelegung ist das Vertauschen zweier Tasten:

```
b::i
i::b
```

Shortcuts Windows 8.1 3(3)

Esc	Abbrechen
Alt PrintScreen	Bildschirmfoto des aktuellen Fensters
Alt F4	Fenster schließen
Alt Enter	Eigenschaften Desktop/Explorer-Objekt
⇧ Alt	Sprache wechseln
Alt Tab / Alt ⇧ Tab	Nächstes/Vorige Task
Alt Esc / Alt ⇧ Esc	Nächstes/Voriges Fenster
Alt Buchstabe	Befehl ausführen
Alt SP	Kontextmenü
→ / ←	Nächstes/voriges Menü (Untermenü)
Alt → / ←	Weiterleiten/Zurück
Alt ⇧ / ⇩	Bildschirm oben/unten
Strg SP	Chinesisches Eingabemethoden-Editor
Strg → / ←	Nächstes/voriges Wort
Strg ↓ / ↑	Nächster/voriger Absatz
Strg ⇧	Tastaturlayout umschalten
Strg Esc	Wie Windows-Taste (nicht kombinierbar)
Strg ⇧ Esc	Taskmanager
Strg → / ← / ↓ / ↑ + SP	Auswählen mehrerer Elemente
⇧ → / ← / ↓ / ↑ + SP	Auswählen mehrerer Elemente
Entf / ⇧ Entf	Löschen/ endgültig löschen
Strg ⇧ → / ← / ↓ / ↑	Textblock markieren
Strg A	Alle Objekte wählen
Strg C / Strg Einf	Objekte in Zwischenablage kopieren
Strg R	Aktualisieren
Strg V / ⇧ Einf	Zwischenablage einfügen
Strg X	Objekte in Zwischenablage löschen
Strg Y	Wiederholen
Strg Z	Rückgängig
Strg Alt Tab	Geöffnete Apps umschalten
Strg Alt Entf	Boot-Menü
Strg Alt ↑ ↓ → ←	Bild drehen

Hotkeys.ahk

```
#+a::Run MsAccess
#+b::Run osk.exe
#+c::Run Chrome.exe
#+!c::Run Chrome.exe -incognito
#+d::Run Chrome.exe http://onedrive.live.com
#+e::Run Excel
#+f::Run "C:\Program Files (x86)\FileZilla FTP Client\filezilla.exe"
#+g::Run "C:\Program Files\GIMP 2\bin\gimp-2.8.exe"
#+i::Run "C:\Program Files (x86)\Inkscape\inkscape.exe"
#+j::Run "C:\Program Files (x86)\IrfanView\i_view32.exe"
#+n::
IfWinExist Unbenannt - Editor
 WinActivate
else
 Run Notepad
return
#+o::Run "C:\Program Files (x86)\Notepad++\notepad++.exe"
#+p::Run PowerPnt
#+r::Run Calc ;Taschenrechner
#+s::Run %windir%\system32\SnippingTool.exe
#+t::if not WinExist( "ahk_class TTOTAL_CMD" )
   Run "C:\Program Files\totalcmd\TOTALCMD64.EXE"
  WinActivate
  Return
#+u::Run MsPub ;Microsoft Publisher
#+w::Run WinWord
#+x::Run OneNote
#+y::Run explorer.exe S:\OneDrive
;#+z::Run CharMap ;Zeichentabelle
#+z::
IfWinExist Zeichentabelle
 WinActivate
else
 Run CharMap
return
```

Hier tauschen die Tasten b und i ihren Platz. Großbuchstaben werden ebenfalls getauscht.

Hinweis: bei diesen einfachen Umbelegungen tauschen die beiden Tasten den Platz, d.h. dieser Tausch betrifft auch alle Kombinationen der Tasten mit den Umschaltetasten.

Wer sich schon einmal darüber geärgert hat, dass sich die Feststelltaste für Großbuchstaben CapsLock unbeabsichtigt eingeschaltet hat, kann folgende Änderung vornehmen:

```
+Capslock::Capslock
Capslock::Ctrl
```

Die erste Zeile bewirkt, dass man CapsLock immer noch betätigen kann aber nur durch gleichzeitiges Drücken von Shift. Die zweite Zeile macht aus CapsLock eine weitere Strg -Taste.

Die Namen der Steuerzeichen-Tasten erfährt man in der Tabelle, die auch für die Hotkeys verwendet wurde.

Um eine Taste zu benennen, deren Zeichen nicht verwendet werden kann, weil es mit der Syntax von AutoHotKey in Konflikt geraten würde, benutzt man die Schreibweise **SCxx**, wobei **xx** der Scancode ist.

Die Scankodes findet man in der Tabelle weiter hinten oder genauer zum Beispiel in PCNEWS-21 oder hier
http://www.marjorie.de/ps2/start.htm

Verwenden wir zum Probieren eine selten benutze Taste, die mit den einfachen Hochkommas, rechts neben dem ß: ´ Diese Taste hat den Scancode **0x0d**. Der Tastenname ist daher **SD0D**.

Wenn also die Taste **SD0D** gedrückt wurde, soll ein bestimmtes Zeichen generiert werden zum Beispiel **A**. Die komplette Zeile lautet:

```
SC0D::A
```

Dieses Mapping bewirkt dass die Taste **SC0D** zur Taste a wird.

Jetzt sollen gemeinsam mit den Umschaltetataste dieser Taste weitere Zeichen zugeordnet werden.

Beginnen wir mit einer einfachen Übung:

```
SC0D::A
+SC0D::B
```

Beabsichtigt ist, dass der einfache Druck auf die Apostroph-Taste **SC0D** ein **A** generiert und die Kombination mit ⇧ ein **B**. Es wird aber in beiden Fällen ein **A** generiert., weil eben in der ersten Zeile **SC0D** zu **A** wird und das mit allen Kombinationen mit Umschaltetasten.

Wenn man will, dass ausschließlich eine dieser vielen Kombinationen umbelegt wird, muss man das Kommando **Send** verwenden.

Jetzt schaut die Umbelegungsanweisung in der Sprache ***AHK*** so aus :

```
SCOD::Send A
+SCOD::Send B
^SCOD::Send C
+^SCOD::Send D
!SCOD::Send E
!+SCOD::Send F
!^SCOD::Send G
!+^SCOD::Send H
#SCOD::Send I
#+SCOD::Send J
#^SCOD::Send K
#+^SCOD::Send L
#!SCOD::Send M
#!+SCOD::Send N
#!^SCOD::Send O
#!+^SCOD::Send P
<^>!SCOD::Send Q
<^>!+SCOD::Send R
…
```

Man sieht, wie viele verschiedene Buchstaben man ein und derselben Taste mit verschiedenen Kombinationen der Umschaltetasten zuweisen könnte. Das ist natürlich nur eine Übung, niemand wird das ausnutzen wollen.

Man könnte, etwa als Mathematiker oder als Austro-Grieche auf die Idee kommen, die Tastatur auch mit dem griechischen Alphabet zu belegen. Dazu kann man prinzipiell die Taste AltGr verwenden, würde die Taste AltGr nicht einige Zeichen erzeugen, nämlich @€ und µ. Meine Lösung schaut so aus:

symbols.ahk

Das griechische Alphabet braucht kein Q, daher kann das Zeichen @ bleiben, wo es ist. Das µ steht ohnehin an der Stelle, an der es auch im griechischen Alphabet stehen würde, es kann daher ebenfalls bleiben.

Man verschiebt also das € in die Zahlenreihe, (**symbols.ahk**) damit sind alle Alpha-Zeichen frei von einer Doppelbelegung.

Die AltGr -Taste bietet sich für die unbelegten Zeichen in der Zahlenreihe an.

```
RAlt & 1::Send {U+00b9} ;Eins hochgestellt
RAlt & 5::Send {U+20ac} ;Euro
```

…

Es ist mir aufgefallen, dass das jüngste Zeichen, „das große scharfe s“, manchmal nicht mit AlrGr ⇧ ß gesendet werden kann. Die letzte Zeile dieser Symboldefinitionen **symbols.ahk** generiert dieses Zeichen.

Die Umlaut-Tasten bekommen die Zusatzfunktion, dass sie mit der AltGr-Taste die Html-Ersatzkodes generieren.

Hinweis **RAlt** entspricht **<^>!**

greek.ahk

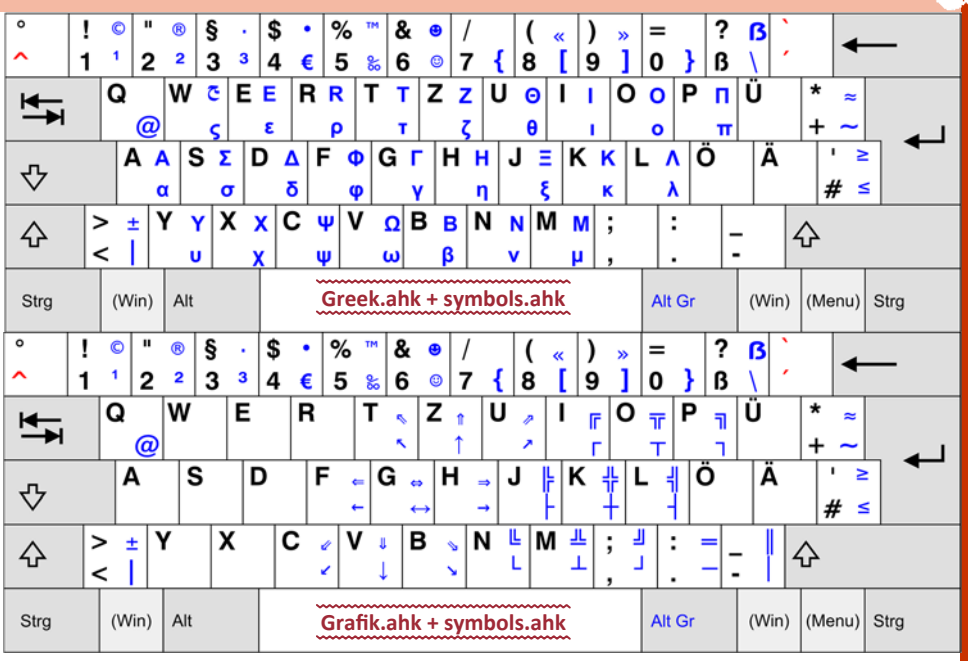
Man ordnet jedem Alpha-Zeichen das entsprechende griechische Zeichen zu und aktiviert den griechischen Kleinbuchstaben mit AltGr und den griechischen Großbuchstaben mit ⇧ AltGr .

Um jetzt zum Beispiel die Taste A mit dem Alphazeichen zu belegen, benötigt man zwei AHK-Zeilen:

```
RAlt & a::Send(0x03b1); Alpha klein α
!RAlt & a::Send(0x0391); Alpha groß A
```

Und so weiter für alle anderen griechischen Zeichen.

Wenn man daher beabsichtigt, die Tastatur so umzubelegen, dass jede Taste gemeinsam mit AltGr und ⇧ AltGr eine weitere Bedeutung bekommt, dann lohnt es sich, dafür eine Funktion zu schreiben und damit eine Taste in einer Zeile abzuarbeiten. Die Funktion nennen wir **ChkShift**.



symbols.ahk

```
;Zeichen
<^>!1::Send {U+00b9} ;¹
+<^>!1::Send {U+00a9} ;©
+<^>!2::Send {U+00ae} ;®
+<^>!3::Send {U+00b7} ;·
<^>!4::Send {U+20ac} ;€
+<^>!4::Send {U+2022} ;·
<^>!5::Send {U+2030} ;‰
+<^>!5::Send {U+2122} ;™
<^>!6::Send {U+263a} ;☺
+<^>!6::Send {U+263b} ;●
+<^>!8::Send {U+00ab} ;«
+<^>!9::Send {U+00bb} ;»
+<^>!SC56::Send {U+00b1} ;±
<^>!SC2b::Send {U+2264} ;≤
+<^>!SC2b::Send {U+2265} ;≥
+<^>!SC1b::Send {U+2248} ;≈
+<^>!SC0c::Send {U+1E9E} ;ß
<^>!SC28::Send &auml;
+<^>!SC28::Send &Auml;
<^>!SC27::Send &ouml;
+<^>!SC27::Send &Ouml;
<^>!SC1a::Send &uuml;
+<^>!SC1a::Send &Uuml;
+<^>!SC1b::Send &szlig;
```

grafik.ahk

```
;Rahmenzeichen
RAlt & i::ChkShift(0x250c, 0x2554) ;┌ ╔
RAlt & o::ChkShift(0x252c, 0x2566) ;┬ ╦
RAlt & p::ChkShift(0x2510, 0x2557) ;┐ ╗
RAlt & j::ChkShift(0x251c, 0x2560) ;├ ╠
RAlt & k::ChkShift(0x253c, 0x256c) ;┼ ╬
RAlt & l::ChkShift(0x2524, 0x2563) ;┤ ╣
RAlt & n::ChkShift(0x2514, 0x255A) ;└ ╚
RAlt & m::ChkShift(0x2534, 0x2569) ;┴ ╩
RAlt & ,::ChkShift(0x2518, 0x255D) ;┘ ╝
RAlt & .::ChkShift(0x2500, 0x2550) ;─ ═
RAlt & -::ChkShift(0x2502, 0x2551) ;│ ║
;Pfeile
RAlt & t::ChkShift(0x2196, 0x21d6) ;↖⇖
RAlt & z::ChkShift(0x2191, 0x21d1) ;↑⇑
RAlt & u::ChkShift(0x2197, 0x21d7) ;↗⇗
RAlt & f::ChkShift(0x2190, 0x21d0) ;←⇐
RAlt & g::ChkShift(0x2194, 0x21d4) ;↔⇔
RAlt & h::ChkShift(0x2192, 0x21d2) ;→⇒
RAlt & c::ChkShift(0x2199, 0x21d9) ;↙⇙
RAlt & v::ChkShift(0x2193, 0x21d3) ;↓⇓
RAlt & b::ChkShift(0x2198, 0x21d8) ;↘⇘

ChkShift(ShiftUpCode,ShiftDownCode)
{
  ShiftUpCode := Chr(ShiftUpCode)
  ShiftDownCode := Chr(ShiftDownCode)
  GetKeyState, state, Shift
  if state = D
    Send %ShiftDownCode%
  else
    Send %ShiftUpCode%
}
```

greek.ahk

```
;griechisches Alphabet
RAlt & a::ChkShift(0x3b1, 0x0391) ;Alpha
RAlt & b::ChkShift(0x3b2, 0x0392) ;Beta
RAlt & c::ChkShift(0x3c8, 0x03a8) ;Psi
RAlt & d::ChkShift(0x3b4, 0x0394) ;Delta
RAlt & e::ChkShift(0x3b5, 0x0395) ;Espilon
RAlt & f::ChkShift(0x3c6, 0x03a6) ;Phi
RAlt & g::ChkShift(0x3b3, 0x0393) ;Gamma
RAlt & h::ChkShift(0x3b7, 0x0397) ;Eta
RAlt & i::ChkShift(0x3b9, 0x0399) ;Iota
RAlt & j::ChkShift(0x3be, 0x039e) ;Xi
RAlt & k::ChkShift(0x3ba, 0x039a) ;Kappa
RAlt & l::ChkShift(0x3bb, 0x039b) ;Lambda
RAlt & m::ChkShift(0x3bc, 0x039c) ;Mu
RAlt & n::ChkShift(0x3bd, 0x039d) ;Nu
RAlt & o::ChkShift(0x3bf, 0x039f) ;Omicron
RAlt & p::ChkShift(0x3c0, 0x03a0) ;Pi
RAlt & r::ChkShift(0x3c1, 0x03a1) ;Rho
RAlt & s::ChkShift(0x3c3, 0x03a3) ;Sigma
RAlt & t::ChkShift(0x3c4, 0x03a4) ;Tau
RAlt & u::ChkShift(0x3b8, 0x0398) ;Theta
RAlt & v::ChkShift(0x3c9, 0x03a9) ;Omega
RAlt & w::ChkShift(0x3c2, 0x03a2) ;Sigma1
RAlt & x::ChkShift(0x3c7, 0x03a7) ;Chi
RAlt & y::ChkShift(0x3c5, 0x03a5) ;Upsilon
RAlt & z::ChkShift(0x3b6, 0x0396) ;Zeta
ChkShift(ShiftUpCode,ShiftDownCode)
{ ShiftUpCode := Chr(ShiftUpCode)
  ShiftDownCode := Chr(ShiftDownCode)
  GetKeyState, state, Shift
  if state = D
    Send %ShiftDownCode%
  else
    Send %ShiftUpCode%
}
```

```
ChkShift(ShiftUpCode,ShiftDownCode)
{
  ShiftUpCode := Chr(ShiftUpCode)
  ShiftDownCode := Chr(ShiftDownCode)
  GetKeyState, state, Shift
  if state = D
    Send %ShiftDownCode%
  else
    Send %ShiftUpCode%
}
```

Die Funktion verwandelt die Parameter in Zeichen und prüft den Zustand der ⇧ -Taste. Wenn sie gedrückt ist, sendet das Programm den ersten Kode, sonst den zweiten. Die Funktion **Chr** verwandelt die Hex-Zahlen in ein Zeichen, die Prozentzeichen sorgen dafür dass nicht der Text **ShiftUpCode** sondern der Variablenwert gesendet wird.

Wer wenig mit griechischen Buchstaben anfangen kann, könnte die alternative Belegung auch mit den Rahmen– und Pfeilzeichen ergänzen: **grafik.ahk**

Das sich daraus ergebende Tastaturlayout (Bild vorige Seite oben) kann sich sehen lassen. Entweder aktiviert man die griechischen oder grafischen Ergänzungen. Es gibt auch noch unbelegte Tasten, die man je nach Arbeitsgebiet mit Zeichen versehen kann.

Die Rahmenzeichen bieten sich auch zur Belegung der nummerischen Tastatur an. Daher wurde auch ein solches Mapping angefertigt (**grafik_num.ahk**, bei der Webversion dieses Artikels). Da aber auf modernen Laptops solche Tastaturen kaum mehr vorkommen, kann eine solche Belegung nur am Stand-PC mit einer Volltastatur eingesetzt werden.

Hotstring, Textsubstitutionen

Unser Autor ***Günter Hartl*** verwendet gerne das Zeichen … als eine Art *„geistiges ad libitum“*, n dessen Stelle der Leser seine Gedanken beliebig weiterentwickeln kann. Tippt man dieses Zeichen in Word, ersetzt Word die drei Punkte automatisch durch das Zeichen *„Horizontale Ellipse“*. Aber einfachere Programme, wie zum Beispiel das Notepad tun das nicht. Hier hilft folgende Sequenz:

```
::...::… ;Ellipse
```

Und weil das so praktisch ist, kann man gleich weitere folgen lassen:

```
::mfg::Mit freundlichen Grüßen
::hv::Hochachtungsvoll
::lg::Liebe Grüße
::gwg::Grün-Weiße Grüße
```

Man sieht, der Schreiber ist ein Rapidler! Es ist aber auch für „Andersgläubige“ leicht möglich, ihre ureigene Grußformel unterzubringen .

Signaturen

Die bisher vorgestellten Abkürzungen werden in derselben Zeile erledigt. Wenn aber der zu ersetzende Text länger ist (zum Beispiel eine Signatur), dann benötigt man ein Klammer-Konstrukt, gefolgt vom Schlüsselwort **return**.

```
::ff::
(
Franz Fiala
Siccardsburggasse 4/1/22
1100 Wien
0664-1015070
franz@fiala.cc
)
return
```

Wenn man an irgendeiner Stelle im Text die Buchstabenfolge **ff** (vorher und nachher ein Blank oder Return), dann substituiert AHK dafür den Signaturtext.

Dass man nach dem Kürzeltext **ff** ein Blank oder Return folgen lassen muss, ist ein Schönheitsfehler, den man aber durch eine syntaktische Besonderheit umgehen kann, indem man in der ersten Zeile schreibt

```
:*:ff::
```

Der Stern bewirkt, dass jedes Auftreten von ff sofort die Signatur einblendet.

Das hat aber wieder den Nachteil, dass auch bei Texten mit einem Doppel-f der Signaturtext subsituiert wird. Das kann man wieder umgehen, indem man der Abkürzung ein Zeichen voranstellt, das in Texten üblicherweise nicht vorkommt, etwa die Raute. Das verlängert zwar das Kürzel um einen Buchstaben, macht aber das Konstrukt gegen unabsichtliche Auslösung ziemlich sicher. Die Zeile lautet daher letztlich

program.ahk

```
;Programmieren
:*:#if#::if ()`n{U+007b}`n{U+007d}`nelse`n{U+007b}`n{U+007d}`n
:*:#sw#::switch ()`r{U+007b}`rdefault:`rbreak;`rcase :`rbreak;`r{U+007d}`n
:*:#st#::<style rel="stylesheet" type="text/css">`n</style>`n
:*:#sc#::<script type="text/javascript">`n<{U+0021}--`n//-->`n</script>`n
:*:#ss#::<link href=".css" rel="stylesheet" type="text/css" />`n
:*:#ul#::<ul>`n<li>`n</li>`n<li>`n</li>`n</ul>`n

:*:#ö:: Send &ouml;
:*:#ä:: Send &auml;
:*:#ü:: Send &uuml;
:*:#Ö:: Send &Ouml;
:*:#Ä:: Send &Auml;
:*:#Ü:: Send &Uuml;
:*:#ß:: Send &szlig;
:*:#ß:: Send &SZlig;
```

#if#

```
if ()
{
}
else
{
}
```

#sw#

```
switch ()
{
default:
break;
case :
break;
}
```

#st#

```
<style rel="stylesheet" type="text/
css">
</style>
```

#sc#

```
<script type="text/javascript">
<!--
//-->
</script>
```

#ss#

```
<link href=".css" rel="stylesheet"
type="text/css" />
```

#ul#

```
<ul>
<li>
</li>
<li>
</li>
</ul>
```

MAKE	BREAK	Tastenbedeutung für deutsche Tastaturbelegung
0x01	0x81	Escape
0x02	0x82	Zahlentaste 1 oder Rufzeichen
0x03	0x83	Zahlentaste 2 oder doppeltes Anführungszeichen
0x04	0x84	Zahlentaste 3 oder Paragrafzeichen
0x05	0x85	Zahlentaste 4 oder Dollarzeichen
0x06	0x86	Zahlentaste 5 oder Prozentzeichen
0x07	0x87	Zahlentaste 6 oder kaufmännisches Und
0x08	0x88	Zahlentaste 7 oder Schrägstrich
0x09	0x89	Zahlentaste 8 oder runde Klammer auf
0x0A	0x8A	Zahlentaste 9 oder runde Klammer zu
0x0B	0x8B	Zahlentaste 0 oder Gleichheitszeichen
0x0C	0x8C	Buchstabentaste Scharfes s oder Fragezeichen
0x0D	0x8D	einfaches oder verkehrtes Anführungszeichen
0x0E	0x8E	Backspace
0x0F	0x8F	Tabulator
0x10	0x90	Buchstabentaste q
0x11	0x91	Buchstabentaste w
0x12	0x92	Buchstabentaste e
0x13	0x93	Buchstabentaste r
0x14	0x94	Buchstabentaste t
0x15	0x95	Buchstabentaste y
0x16	0x96	Buchstabentaste u
0x17	0x97	Buchstabentaste i
0x18	0x98	Buchstabentaste o
0x19	0x99	Buchstabentaste p
0x1A	0x9A	Buchstabentaste ü
0x1B	0x9B	Pluszeichen oder Malzeichen
0x1C	0x9C	Enter, Eingabetaste
0x1D	0x9D	Linke Control-Taste
0x1E	0x9E	Buchstabentaste a
0x1F	0x9F	Buchstabentaste s
0x20	0xA0	Buchstabentaste d
0x21	0xA1	Buchstabentaste f
0x22	0xA2	Buchstabentaste g
0x23	0xA3	Buchstabentaste h
0x24	0xA4	Buchstabentaste j
0x25	0xA5	Buchstabentaste k
0x26	0xA6	Buchstabentaste l
0x27	0xA7	Buchstabentaste ö
0x28	0xA8	Buchstabentaste ä
0x29	0xA9	Circonflex oder Gradzeichen
0x2A	0xAA	Linke Hochstelltaste
0x2B	0xAB	Raute und Hochkomma
0x2C	0xAC	Buchstabentaste z
0x2D	0xAD	Buchstabentaste x
0x2E	0xAE	Buchstabentaste c
0x2F	0xAF	Buchstabentaste v
0x30	0xB0	Buchstabentaste b
0x31	0xB1	Buchstabentaste n
0x32	0xB2	Buchstabentaste m
0x33	0xB3	Beistrich und Strichpunkt
0x34	0xB4	Punkt und Doppelpunkt
0x35	0xB5	Bindestrich und Unterstreichung
0x36	0xB6	Rechte Hochstelltaste
0x37	0xB7	Stern, Malzeichen
0x38	0xB8	Linke ALT-Taste
0x39	0xB9	Zwischenraum
0x3A	0xBA	Feststelltaste
0x3B	0xBB	Funktionstaste F1
0x3C	0xBC	Funktionstaste F2
0x3D	0xBD	Funktionstaste F3
0x3E	0xBE	Funktionstaste F4
0x3F	0xBF	Funktionstaste F5
0x40	0xC0	Funktionstaste F6
0x41	0xC1	Funktionstaste F7
0x42	0xC2	Funktionstaste F8
0x43	0xC3	Funktionstaste F9
0x44	0xC4	Funktionstaste F10
0x45	0xC5	Nummernfeststellung für rechten Kursorblock
0x46	0xC6	SCROLL-LOCK oder, mit Control gemeinsam BREAK
0x47	0xC7	Zahlentaste 7 oder Kursor auf Position 1
0x48	0xC8	Zahlentaste 8 oder up
0x49	0xC9	Zahlentaste 9 oder PgUp
0x4A	0xCA	grey-minus
0x4B	0xCB	Zahlentaste 4 oder left
0x4C	0xCC	Zahlentaste 5 oder center
0x4D	0xCD	Zahlentaste 6 oder right
0x4E	0xCE	grey-plus
0x4F	0xCF	Zahlentaste 1 oder end
0x50	0xD0	Zahlentaste 2 oder down
0x51	0xD1	Zahlentaste 3 oder PgDn
0x52	0xD2	Zahlentaste 0 oder Ins
0x53	0xD3	Beistrich oder Zeichen löschen
0x54	0xD4	System request
0x55	0xD5	
0x56	0xD6	Kleiner-Zeichen oder Größer-Zeichen
0x57	0xD7	Funktionstaste F 11
0x58	0xD8	Funktionstaste F 12

```
:*:#ff::
```

Ich verwende mehrere geringfügig verschiedene Signaturen:

`#ff`	normal, Inland
`#dff`	mit Titel, Inland
`#dffi`	mit Titel, international

Es kommt vor, dass man ein Datum einfügen möchte. Auch dazu eignet sich ein solcher Hotstring:

```
:*:#dt::
  ;FormatTime,Datum,,dd.MM.yy - HH:mm:ss
  FormatTime,Datum,,yyyy-MM-dd HH:mm
  Send, %Datum%
Return
```

Programmkode

Wer viel kodiert, muss oft komplizierte, wiederkehrende Konstrukte eintippen. Das kann man mit AHK abkürzen. Hier einige Beispiele:

Beispiel: Die Eingabe **#if#** bewirkt, dass dieser Text durch ein if-else-Konstrukt ersetzt wird.

Links

AutoHotKey
http://ahkscript.org/

AutoHotKey (alt)
http://www.autohotkey.com/

AutoHotKey für Linux

Für Linux gibt es AutoHotKey nicht aber es gibt zwei Programme, die sich als Ersatz eignen:

Xdotool
http://manpages.ubuntu.com/manpages/precise/man1/xdotool.1.html

Zenity
http://manpages.ubuntu.com/manpages/precise/man1/zenity.1.html

Wikipedia
http://de.wikipedia.org/wiki/AutoHotkey

Deutsche Dokumentation
http://ragnar-f.github.io/docs/Hotkeys.htm

15 wichtigste AKH-Skripts
http://michaelsonntag.net/meine-15-wichtigsten-autohotkey-snippets/

AHK WIKI
http://www.ghisler.ch/wiki/index.php/AutoHotkey

Viele AHK Skripts
http://www.donationcoder.com/Software/Skrommel/wett

Virtuelle Tastaturen bei Windows 8/8.1

Betrifft Beitrag auf de folgenden Seite.

Die Bilder rechts zeigen das neue Tastaturlayout für die Apps **(oben)** und die klassische Tastatur aus dem Bereich der „erleichterten Bedienung“ **(unten)**.

Man kann zwar das Layout der neuen Windows-8-Tastatur ändern und auch auf Handschrift-Eingabe schalten, nicht aber auf das klassische Tastaturlayout.

Diese beiden Tastaturen schließen einander aus. Ist also die eine aktiviert, kann man die andere nicht öffnen.

Die Tastatur für erleichterte Bedienung hat außerdem eine Textvorhersage **(Bild rechts, unten)** und man fragt sich, warum dieses Element nicht als eine grundsätzliche Eingabehilfe für alle Texteingaben eingesetzt wird.


App-Bildschirmtastatur


App-Bildschirmtastatur, altenatives Layout


App-Bildschirmtastatur, Zahlen und Zeichen

App-Bildschirmtastatur, Handschrift und Hilfe

Bildschirmtastatur klassisch


Bildschirmtastatur klassisch, Navigation


Bildschirmtastatur klassisch,Textvorhersage