List of Keys (Keyboard, Mouse and Controller)

Table of Contents

Mouse

General Buttons

Name	Description
LButton	Primary mouse button. Which physical button this corresponds to depends on system settings; by default it is the left button.
RButton	Secondary mouse button. Which physical button this corresponds to depends on system settings; by default it is the right button.
MButton	Middle or wheel mouse button

Advanced Buttons

Name	Description

XButton1	4th mouse button. Typically performs the same function as Browser_Back.
XButton2	5th mouse button. Typically performs the same function as Browser_Forward.

Wheel

Name	Description
WheelDown	Turn the wheel downward (toward you).
WheelUp	Turn the wheel upward (away from you).
WheelLeft WheelRight	Scroll to the left or right. These can be used as hotkeys with some (but not all) mice which have a second wheel or support tilting the wheel to either side. In some cases, software bundled with the mouse must instead be used to control this feature. Regardless of the particular mouse, Send and Click can be used to scroll horizontally in programs which support it.

Keyboard

> **Note:** The names of the letter and number keys are the same as that single letter or digit. For example: b is `B` and 5 is `5`.

Although any single character can be used as a key name, its meaning (scan code or virtual keycode) depends on the current keyboard layout. Additionally, some special characters may need to be escaped or enclosed in braces, depending on the context. The letters a-z or A-Z can be used to refer to the corresponding virtual keycodes (usually vk41-vk5A) even if they are not included in the current keyboard layout.

General Keys

Name	Description
CapsLock	`CapsLock` (caps lock key) **Note:** Windows IME may interfere with the detection and functionality of CapsLock; see CapsLock and IME for details.

Space	`Space` (space bar)
Tab	`Tab` (tabulator key)
Enter	`Enter`
Escape (or Esc)	`Esc`
Backspace (or BS)	`Backspace`

Cursor Control Keys

Name	Description
ScrollLock	`ScrollLock` (scroll lock key). While `Ctrl` is held down, `ScrollLock` produces the key code of `CtrlBreak`, but can be differentiated from `Pause` by scan code.
Delete (or Del)	`Del`
Insert (or Ins)	`Ins`
Home	`Home`
End	`End`
PgUp	`PgUp` (page up key)
PgDn	`PgDn` (page down key)
Up	`↑` (up arrow key)
Down	`↓` (down arrow key)
Left	`←` (left arrow key)
Right	`→` (right arrow key)

Numpad Keys

Due to system behavior, the following keys separated by a slash are identified differently depending on whether `NumLock` is ON or OFF. If `NumLock` is OFF but `Shift` is pressed, the system temporarily releases `Shift` and acts as though `NumLock` is ON.

Name	Description
Numpad0 / NumpadIns	`0` / `Ins`
Numpad1 / NumpadEnd	`1` / `End`

Numpad2 / NumpadDown	`2` / `↓`
Numpad3 / NumpadPgDn	`3` / `PgDn`
Numpad4 / NumpadLeft	`4` / `←`
Numpad5 / NumpadClear	`5` / typically does nothing
Numpad6 / NumpadRight	`6` / `→`
Numpad7 / NumpadHome	`7` / `Home`
Numpad8 / NumpadUp	`8` / `↑`
Numpad9 / NumpadPgUp	`9` / `PgUp`
NumpadDot / NumpadDel	`.` / `Del`
NumLock	`NumLock` (number lock key). While `Ctrl` is held down, `NumLock` produces the key code of `Pause`, so use `^Pause` in hotkeys instead of `^NumLock`.
NumpadDiv	`/` (division)
NumpadMult	`*` (multiplication)
NumpadAdd	`+` (addition)
NumpadSub	`-` (subtraction)
NumpadEnter	`Enter`

Function Keys

Name	Description
F1 - F24	The 12 or more function keys at the top of most keyboards.

Modifier Keys

Name	Description
LWin	Left `Win`. Corresponds to the `<#` hotkey prefix.
RWin	Right `Win`. Corresponds to the `>#` hotkey prefix. **Note:** Unlike `Ctrl`/`Alt`/`Shift`, there is no generic/neutral "Win" key because the OS does not support it. However, hotkeys with the `#` modifier can be triggered by either `Win`.

Control (or Ctrl)	`Ctrl`. As a hotkey (`Control::`) it fires upon release unless it has the tilde prefix. Corresponds to the `^` hotkey prefix.
Alt	`Alt`. As a hotkey (`Alt::`) it fires upon release unless it has the tilde prefix. Corresponds to the `!` hotkey prefix.
Shift	`Shift`. As a hotkey (`Shift::`) it fires upon release unless it has the tilde prefix. Corresponds to the `+` hotkey prefix.
LControl (or LCtrl)	Left `Ctrl`. Corresponds to the `<^` hotkey prefix.
RControl (or RCtrl)	Right `Ctrl`. Corresponds to the `>^` hotkey prefix.
LShift	Left `Shift`. Corresponds to the `<+` hotkey prefix.
RShift	Right `Shift`. Corresponds to the `>+` hotkey prefix.
LAlt	Left `Alt`. Corresponds to the `<!` hotkey prefix.
RAlt	Right `Alt`. Corresponds to the `>!` hotkey prefix. **Note:** If your keyboard layout has AltGr instead of RAlt, you can probably use it as a hotkey prefix via `<^>!` as described here. In addition, `LControl & RAlt::` would make AltGr itself into a hotkey.

Multimedia Keys

The function assigned to each of the keys listed below can be overridden by modifying the Windows registry. This table shows the default function of each key on most versions of Windows.

Name	Description
Browser_Back	Back
Browser_Forward	Forward
Browser_Refresh	Refresh
Browser_Stop	Stop
Browser_Search	Search
Browser_Favorites	Favorites
Browser_Home	Homepage
Volume_Mute	Mute the volume

Volume_Down	Lower the volume
Volume_Up	Increase the volume
Media_Next	Next Track
Media_Prev	Previous Track
Media_Stop	Stop
Media_Play_Pause	Play/Pause
Launch_Mail	Launch default e-mail program
Launch_Media	Launch default media player
Launch_App1	Launch This PC (formerly My Computer or Computer)
Launch_App2	Launch Calculator

Other Keys

Name	**Description**
AppsKey	`Menu`. This is the key that invokes the right-click context menu.
PrintScreen	`PrtSc` (print screen key)
CtrlBreak	`Ctrl`+`Pause` or `Ctrl`+`ScrollLock`
Pause	`Pause` or `Ctrl`+`NumLock`. While `Ctrl` is held down, `Pause` produces the key code of `CtrlBreak` and `NumLock` produces `Pause`, so use `^CtrlBreak` in hotkeys instead of `^Pause`.
Help	`Help`. This probably doesn't exist on most keyboards. It's usually not the same as `F1`.
Sleep	`Sleep`. Note that the sleep key on some keyboards might not work with this.
SC**nnn**	Specify for **nnn** the scan code of a key. Recognizes unusual keys not mentioned above. See Special Keys for details.
VK**nn**	Specify for **nn** the hexadecimal virtual key code of a key. This rarely-used method also prevents certain types of hotkeys from requiring the keyboard hook. For example, the following hotkey does not use the keyboard hook, but as a side-effect it is triggered by pressing *either* `Home` or NumpadHome:

```
^VK24::MsgBox "You pressed Home or NumpadHome
while holding down Control."
```

Known limitation: VK hotkeys that are forced to use the keyboard hook, such as `*VK24` or `~VK24`, will fire for only one of the keys, not both (e.g. NumpadHome but not `Home`). For more information about the VKnn method, see Special Keys.

Warning: Only Send, GetKeyName, GetKeyVK, GetKeySC and A_MenuMaskKey support combining VKnn and SCnnn. If combined in any other case (or if any other invalid suffix is present), the key is not recognized. For example, `vk1Bsc001::` raises an error.

Game Controller (Gamepad, Joystick, etc.)

Note: For historical reasons, the following button and control names begin with `Joy`, which stands for joystick. However, they usually also work for other game controllers such as gamepads or steering wheels.

Joy1 through Joy32: The buttons of the controller. To help determine the button numbers for your controller, use this test script. Note that hotkey prefix symbols such as ^ (control) and + (shift) are not supported (though GetKeyState can be used as a substitute). Also note that the pressing of controller buttons always "passes through" to the active window if that window is designed to detect the pressing of controller buttons.

Although the following control names cannot be used as hotkeys, they can be used with GetKeyState:

- **JoyX, JoyY, and JoyZ:** The X (horizontal), Y (vertical), and Z (altitude/depth) axes of the stick.
- **JoyR:** The rudder or 4th axis of the stick.
- **JoyU and JoyV:** The 5th and 6th axes of the stick.
- **JoyPOV:** The point-of-view (hat) control.
- **JoyName:** The name of the controller or its driver.
- **JoyButtons:** The number of buttons supported by the controller (not always accurate).
- **JoyAxes:** The number of axes supported by the controller.
- **JoyInfo:** Provides a string consisting of zero or more of the following letters to indicate the controller's capabilities: **Z** (has Z axis), **R** (has R axis), **U** (has U axis), **V** (has V axis), **P** (has POV control), **D** (the POV control has a limited number of discrete/distinct settings), **C** (the POV control is continuous/fine). Example string: ZRUVPD

For example, when using Xbox Wireless/360 controllers, JoyX/JoyY is the left stick, JoyR/JoyU the right stick, JoyZ the left and right triggers, and JoyPOV the directional pad (D-pad).

Multiple controllers: If the computer has more than one controller and you want to use one beyond the first, include the controller number (max 16) in front of the control name. For example, 2joy1 is the second controller's first button.

> **Note:** If you have trouble getting a script to recognize your controller, specify a controller number other than 1 even though only a single controller is present. It is unclear how this situation arises or whether it is normal, but experimenting with the controller number in the controller test script can help determine if this applies to your system.

See Also:

- Controller remapping: Methods of sending keystrokes and mouse clicks with a controller.
- Controller-To-Mouse script: Using a controller as a mouse.

Hand-held Remote Controls

Respond to signals from hand-held remote controls via the WinLIRC client script.

Special Keys

If your keyboard or mouse has a key not listed above, you might still be able to make it a hotkey by using the following steps:

1. Ensure that at least one script is running that is using the keyboard hook. You can tell if a script has the keyboard hook by opening its main window and selecting "View->Key history" from the menu bar.
2. Double-click that script's tray icon to open its main window.
3. Press one of the "mystery keys" on your keyboard.
4. Select the menu item "View->Key history"
5. Scroll down to the bottom of the page. Somewhere near the bottom are the key-down and key-up events for your key. NOTE: Some keys do not generate events and thus will not be visible here. If this is the case, you cannot directly make that particular key a hotkey because your keyboard driver or hardware handles it at a level too low for AutoHotkey to access. For possible solutions, see further below.
6. If your key is detectable, make a note of the 3-digit hexadecimal value in the second column of the list (e.g. **159**).
7. To define this key as a hotkey, follow this example:

```
SC159::MsgBox ThisHotkey " was pressed." ; Replace 159 with your key's
value.
```

Also see ThisHotkey.

Reverse direction: To remap some other key to *become* a "mystery key", follow this example:

```
; Replace 159 with the value discovered above. Replace FF (if needed) with
the
; key's virtual key, which can be discovered in the first column of the
Key History screen.
#c::Send "{vkFFsc159}" ; See Send {vkXXscYYY} for more details.
```

Alternate solutions: If your key or mouse button is not detectable by the Key History screen, one of the following might help:

1. Reconfigure the software that came with your mouse or keyboard (sometimes accessible in the Control Panel or Start Menu) to have the "mystery key" send some other keystroke. Such a keystroke can then be defined as a hotkey in a script. For example, if you configure a mystery key to send `Ctrl`+`F1`, you can then indirectly make that key as a hotkey by using `^F1::` in a script.

2. Try AHKHID from the archived forum. You can also try searching the forum for a keywords like `RawInput*`, `USB HID` or `AHKHID`.

3. The following is a last resort and generally should be attempted only in desperation. This is because the chance of success is low and it may cause unwanted side-effects that are difficult to undo:
   Disable or remove any extra software that came with your keyboard or mouse or change its driver to a more standard one such as the one built into the OS. This assumes there is such a driver for your particular keyboard or mouse and that you can live without the features provided by its custom driver and software.

CapsLock and IME

Some configurations of Windows IME (such as Japanese input with English keyboard) use CapsLock to toggle between modes. In such cases, CapsLock is suppressed by the IME and cannot be detected by AutoHotkey. However, the `Alt`+`CapsLock`, `Ctrl`+`CapsLock` and `Shift`+`CapsLock` shortcuts can be disabled with a workaround. Specifically, send a key-up to modify the state of the IME, but prevent any other effects by signalling the keyboard hook to suppress the event. The following function can be used for this purpose:

```
; The keyboard hook must be installed.
InstallKeybdHook
SendSuppressedKeyUp(key) {
    DllCall("keybd_event"
        , "char", GetKeyVK(key)
        , "char", GetKeySC(key)
        , "uint", KEYEVENTF_KEYUP := 0x2
        , "uptr", KEY_BLOCK_THIS := 0xFFC3D450)
}
```

After copying the function into a script or saving it as *SendSuppressedKeyUp.ahk* in a Lib folder and adding `#Include <SendSuppressedKeyUp>` to the script, it can be used as follows:

```
; Disable Alt+key shortcuts for the IME.
~LAlt::SendSuppressedKeyUp "LAlt"

; Test hotkey:
!CapsLock::MsgBox A_ThisHotkey

; Remap CapsLock to LCtrl in a way compatible with IME.
*CapsLock::
{
    Send "{Blind}{LCtrl DownR}"
    SendSuppressedKeyUp "LCtrl"
}
*CapsLock up::
{
    Send "{Blind}{LCtrl Up}"
}
```