Function	Arguments	Returns	Description
UsbInitialise	None	r0 is result code	This method is the all-in-one method that loads the USB driver, enumerates all devices and attempts to communicate with them. This method generally takes about a second to execute, though with a few USB hubs plugged in this can be significantly longer. After this method is completed methods in the keyboard driver become available, regardless of whether or not a keyboard is indeed inserted. Result code explained below.
UsbCheckForChange	None	None	Essentially provides the same effect as UsbInitialise, but does not provide the same one time initialisation. This method checks every port on every connected hub recursively, and adds new devices if they have been added. This should be very quick if there are no changes, but can take up to a few seconds if a hub with many devices is attached.
KeyboardCount	None	r0 is count	Returns the number of keyboards currently connected and detected. UsbCheckForChange may update this. Up to 4 keyboards are supported by default. Up to this many keyboards may be accessed through this driver.
KeyboardGetAddress	r0 is index	r0 is address	Retrieves the address of a given keyboard. All other functions take a keyboard address in order to know which keyboard to access. Thus, to communicate with a keyboard, first check the count, then retrieve the address, then use other methods. Note, the order of keyboards that this method returns may change after calls to UsbCheckForChange.
KeyboardPoll	r0 is address	r0 is result code	Reads in the current key state from the keyboard. This operates via polling the device directly, contrary to the best practice. This means that if this method is not called frequently enough, a key press could be missed. All reading methods simply return the value as of the last poll.
KeyboardGetModifiers	r0 is address	r0 is modifier state	Retrieves the status of the modifier keys as of the last poll. These are the shift, alt control and GUI keys on both sides. This is returned as a bit field, such that a 1 in the bit 0 means left control is held, bit 1 means left shift, bit 2 means left alt, bit 3 means left GUI and bits 4 to 7 mean the right versions of those previous. If there is a problem r0 contains 0.

Function	Arguments	Returns	Description
KeyboardGetKeyDownCount	r0 is address	r0 is count	Retrieves the number of keys currently held down on the keyboard. This excludes modifier keys. Normally, this cannot go above 6. If there is an error this method returns 0.
KeyboardGetKeyDown	r0 is address, r1 is key number	r0 is scan code	Retrieves the scan code (see Table 4.1) of a particular held down key. Normally, to work out which keys are down, call KeyboardGetKeyDownCount and then call KeyboardGetKeyDown up to that many times with increasing values of r1 to determine which keys are down. Returns 0 if there is a problem. It is safe (but not recommended) to call this method without calling KeyboardGetKeyDownCount and interpret 0s as keys not held. Note, the order or scan codes can change randomly (some keyboards sort numerically, some sort temporally, no guarantees are made).
KeyboardGetKeyIsDown	r0 is address, r1 is scan code	r0 is status	Alternative to KeyboardGetKeyDown, checks if a particular scan code is among the held down keys. Returns 0 if not, or a non-zero value if so. Faster when detecting particular scan codes (e.g. looking for ctrl+c). On error, returns 0.
KeyboardGetLedSupport	r0 is address	r0 is LEDs	Checks which LEDs a particular keyboard supports. Bit 0 being 1 represents Number Lock, bit 1 represents Caps Lock, bit 2 represents Scroll Lock, bit 3 represents Compose, bit 4 represents Kana, bit 5 represents Power, bit 6 represents Mute and bit 7 represents Compose. As per the USB standard, none of these LEDs update automatically (e.g. Caps Lock must be set manually when the Caps Lock scan code is detected).
KeyboardSetLeds	r0 is address, r1 is LEDs	r0 is result code	Attempts to turn on/off the specified LEDs on the keyboard. See below for result code values. See KeyboardGetLedSupport for LEDs' values.

CSUD Result Codes

Code	Description
0	Method completed successfully.
-2	Argument: A method was called with an invalid argument.
-4	Device: The device did not respond correctly to the request.
-5	Incompatible: The driver is not compatible with this request or device.
-6	Compiler: The driver was compiled incorrectly, and is broken.
-7	Memory: The driver ran out of memory.
-8	Timeout: The device did not respond in the expected time.
-9	Disconnect: The device requested has disconnected, and cannot be used.