Single Character Input

Single-Character Input and Irvine32 Keyboard Procedures

In console mode on MS-Windows, handling single-character input involves dealing with the keyboard device driver, scan codes, virtual-key codes, and the message queue. Here's an explanation of the process and the relevant Irvine32 keyboard procedures:

Keyboard Input Process:

MS-Windows provides a device driver for the installed keyboard. When a key is pressed, it sends an 8-bit scan code to the computer's keyboard port.

Upon releasing the key, a second scan code is transmitted. MS-Windows translates these scan codes into 16-bit virtual-key codes, which are device-independent values that identify the key's purpose.

A message containing the scan code, virtual-key code, and related information is created by MS-Windows and placed in the message queue.

The message eventually reaches the currently executing program thread, identified by the console input handle.

Irvine32 Keyboard Procedures:

The Irvine32 library provides two related procedures for handling keyboard input: ReadChar and ReadKey.

ReadChar waits for an ASCII character to be typed at the keyboard and returns the character in the AL register.

ReadKey performs a no-wait keyboard check. If no key is waiting in the console input buffer, it sets the Zero flag.

If a key is found, the Zero flag is clear, and AL contains either zero or an ASCII code. The upper halves of EAX and EDX are overwritten.

Using ReadKey and Control Key State:

In ReadKey, if AL contains zero, the user may have pressed a special key (e.g., function key, cursor arrow).

AH register contains the keyboard scan code, which can be matched to a list of keyboard keys. DX contains the virtual-key code.

EBX contains state information about the states of the keyboard control keys.

Control Key State Values:

- **CAPSLOCK_ON:** The CAPS LOCK light is on.
- **ENHANCED_KEY:** The key is enhanced.
- **LEFT_ALT_PRESSED:** The left ALT key is pressed. LEFT_CTRL_PRESSED: The left CTRL key is pressed.
- **NUMLOCK_ON:** The NUM LOCK light is on.
- **RIGHT_ALT_PRESSED:** The right ALT key is pressed.
- **RIGHT_CTRL_PRESSED:** The right CTRL key is pressed.
- **SCROLLLOCK_ON:** The SCROLL LOCK light is on.
- **SHIFT_PRESSED:** The SHIFT key is pressed.

You can use these control key state values to determine the state of control keys while processing keyboard input.

The **ReadChar** and **ReadKey** procedures in the Irvine32 library simplify handling keyboard input in your assembly programs, making it easier to respond to user interactions.

Testing Keyboard Input with ReadKey and GetKeyState

This section covers testing keyboard input using ReadKey and GetKeyState, including a program that reports the state of the CapsLock key and another program that checks the state of the NumLock and Left Shift keys.

Testing Keyboard Input with ReadKey:

The program tests ReadKey by waiting for a keypress and then reporting the state of the CapsLock key.

A **delay factor** is included when calling ReadKey to allow MS-Windows

to process its message loop.

If ReadKey returns a non-zero value (a keypress has occurred), the program tests the value of EBX using the **CAPSLOCK_ON constant** to check the state of the CapsLock key.

It then displays a message indicating whether CapsLock is ON or OFF.

GetKeyState for Keyboard State Testing:

The GetKeyState API function allows you to test the state of individual keyboard keys. You pass it a virtual key value, like those identified in Table 11-4.

Table 11-4 Testing Keys with GetKeyState.

Key	Virtual Key Symbol	Bit to Test in EAX
NumLock	VK_NUMLOCK	0
Scroll Lock	VK_SCROLL	0
Left Shift	VK_LSHIFT	15
Right Shift	VK_tRSHIFT	15
Left Ctrl	VK_LCONTROL	15
Right Ctrl	VK_RCONTROL	15
Left Menu	VK_LMENU	15
Right Menu	VK_RMENU	15

It returns a value in EAX, and you need to test the value to determine the state of the key.

The program demonstrates using GetKeyState to check the state of the NumLock and Left Shift keys:

It calls GetKeyState with VK_NUMLOCK and checks if the lowest bit (bit 0) of AL is set.

(bit 0) of AL is set.

If it is set, it indicates that NumLock is ON.

It then calls GetKeyState with VK_LSHIFT and checks the high bit (bit 31) of EAX to determine if the Left Shift key is currently pressed.

Depending on the test results, it displays appropriate messages to report the state of the keys.

```
;Testing Keyboard Input with ReadKey (TestReadkey.asm)
INCLUDE Irvine32.inc
INCLUDE Macros.inc

.code
main PROC
L1:
    mov eax, 10      ; Delay for message processing
    call Delay
    call ReadKey     ; Wait for a keypress
    jz L1

    test ebx, CAPSLOCK_ON
    jz L2
    mWrite <"CapsLock is ON", 0dh, 0ah>
    jmp L3
L2:
    mWrite <"CapsLock is OFF", 0dh, 0ah>
L3:
    exit
main ENDP
END main
```

Program 2:

```
;GetKeyState
INCLUDE Irvine32.inc
INCLUDE Macros.inc

.code
main PROC
    INVOKE GetKeyState, VK_NUMLOCK
    test al, 1
    .IF !Zero?
        mWrite <"The NumLock key is ON", 0dh, 0ah>
    .ENDIF

    INVOKE GetKeyState, VK_LSHIFT
    test eax, 80000000h
    .IF !Zero?
        mWrite <"The Left Shift key is currently DOWN", 0dh, 0ah>
    .ENDIF

    exit
main ENDP
END main
```