Reprogramming the Keyboard Interrupt

Intro

First off, trying to quickly poll the keyboard in standard C++ is a joke. There is not way to get multiple keystrokes *easily* and quickly. When you finally support multiple keystrokes, you are limited to probably only 2, three tops. The point isn't that you need more than 3 at a time, but it says something about the speed the program is polling the keyboard. It just isn't fast enough. We want to focus our attention on OUTPUT, not waste cycles getting input! For each of the functions below, you must set it as the keyboard interrupt. This varies from compiler to compiler. I've tried to give you an idea how to do it, but the important part is the code that replaces the interrupt. Here's a couple of flavors :)

Here's a helpful listing of all the make and break codes for they keyboard: Make and Break Codes

DJGPP Inline Assembly

```
long keytable=(long)&key_table[0];

int Keyboard_ISR(_go32_dpmi_registers *r)
{ __asm__ __volatile__(“
        sti
        xorl %%ebx,%%ebx
        movw $0x60,%%dx
        inb %%dx,%%al
        xor %%ah,%%ah
        movw %%ax,%%bx
        movw $0x61,%%dx
        inb %%dx,%%al
        or $0x82,%%al
        outb %%al,%%dx
        andb $0x7f,%%al
        outb %%al,%%dx

        cmpb $84,%%bx
        jb  Make
        cmpb $211,%%bx
        ja  End
        movl _keytable,%%eax            // Write 0 in table
        subw $128,%%bx
```

```
        addl %%ebx,%%eax
        movw $0,(%%eax)
        jmp End

        Make:                   //Write 1 in table
        movl _keytable,%%eax
        addl %%ebx,%%eax
        movl $1,(%%eax)
        End:

        movw $0x20,%%ax
        movw $0x20,%%dx
        outw %%ax,%%dx
        cli
        "
        :
        :
        :"%eax","%dx","%ebx");
}
```

NOTE: See the DJGPP and Protected Mode Section for setting the interrupt.

C++

```
void interrupt Keyboard_ISR(...)
{unsigned char key;
 union REGS regs
 asm{sti}
 regs.h.al=inp(0x60);
 regs.h.ah=0;
 key = regs.x.ax;
 regs.h.al=inp(0x61)
 regs.h.al |=0x82
 outp(0x61,regs.h.al);
 regs.h.al &= 0x7f;
 outp(0x61,regs.h.al);
 outp(0x20,0x20);
 asm{cli};

 if(key < 84)
   {key_table[key]=1;
   }
 else if(key < 212)
   {key_table[key-128]=0;
   }
}
```

Other C++ Inline Assembly

```
void interrupt Keyboard_ISR(...)
{unsigned char key;
asm{sti
       in al,0x60
       xor ah,ah
       mov key,ax
       in al,0x61
       or al,0x82
       out 0x61,al
       and al,0x7f
       out 0x61,al
       mov al,0x20
       out 0x20,al
       cli
      }
 if(key < 84)
   {key_table[key]=1;
   }
 else if(key < 212)
   {key_table[key-128]=0;
   }
}
```

All you need to gain complete control over the keyboard interrupt is to have the address of the starting point in an array of unsigned characters of 83 elements (key_table). Each element in this array is a flag for wether a key is down or not. This way we can check as many combinations as possible, with the resolution of the keyboard timing interrupt being the limiting factor. And even that can be set faster to compensate. This normally isn't needed since the code is pretty darn fast! Each gets the key from the keyboard and tests wether it is a MAKE CODE (1-83) or a BREAK CODE(129-211). If a MAKE code is sent, then that element in the array is set to a one. If a BREAK code is sent, then the code-128 is set to 0, effectively turning it off. Just examine the array at any given time to see if a key is down!

NOTE: These functions do ALL the work, just hook them up with a valid array, and the interrupt, and away you go!

Hooking the Interrupt in Normal C++

```
#ifdef __cplusplus
   #define __CPPARGS ...
#else
   #define __CPPARGS
#endif
```

```
void interrupt Keyboard_ISR(__CPPARGS);    /* NEW interrupt prototype */
void interrupt (*OldInterrupt)(__CPPARGS);        /* Old interrupt function pointer */

void main(void)
{ OldInterrupt  = _dos_getvect(0x09);
  _dos_setvect(0x09,Keyboard_ISR);
  ...
  ...
  ...
  _dos_setvect(0x09,OldInterrupt);
}
```

All we have to do is include the two prototypes, Keyboard_ISR (name of new isr) and OldInterrupt (pointer to a function). We save the old ISR function so that we can return it before program exit, so that we don't mess anything up. It's just this simple. If you are using DJGPP please take a look at the DJGPP Protected Mode Programming tutorial. Before I get into the usual copyright stuff and closing statements, here's the huge Make and Break listing!

Make and Break Codes

```
#define MAKE_ESC            1
#define MAKE_1              2
#define MAKE_2              3
#define MAKE_3              4
#define MAKE_4              5
#define MAKE_5              6
#define MAKE_6              7
#define MAKE_7              8
#define MAKE_8              9
#define MAKE_9              10
#define MAKE_0              11
#define MAKE_MINUS          12
#define MAKE_EQUALS         13
#define MAKE_BKSP           14
#define MAKE_TAB            15
#define MAKE_Q              16
#define MAKE_W              17
#define MAKE_E              18
#define MAKE_R              19
#define MAKE_T              20
#define MAKE_Y              21
#define MAKE_U              22
#define MAKE_I              23
#define MAKE_O              24
#define MAKE_P              25
#define MAKE_LFT_BRACKET    26
```

```
#define MAKE_RGT_BRACKET    27
#define MAKE_ENTER          28
#define MAKE_CNTRL          29
#define MAKE_A              30
#define MAKE_S              31
#define MAKE_D              32
#define MAKE_F              33
#define MAKE_G              34
#define MAKE_H              35
#define MAKE_J              36
#define MAKE_K              37
#define MAKE_L              38
#define MAKE_SEMI           39
#define MAKE_APOS           40
#define MAKE_TILDE          41
#define MAKE_SHIFTL         42
#define MAKE_BACK_SLASH     43
#define MAKE_Z              44
#define MAKE_X              45
#define MAKE_C              46
#define MAKE_V              47
#define MAKE_B              48
#define MAKE_N              49
#define MAKE_M              50
#define MAKE_COMMA          51
#define MAKE_PERIOD         52
#define MAKE_FOWARD_SLASH   53
#define MAKE_SHIFTR         54
#define MAKE_PRT_SCRN       55
#define MAKE_ALT            56
#define MAKE_SPACE          57
#define MAKE_CAPS_LOCK      58
#define MAKE_F1             59
#define MAKE_F2             60
#define MAKE_F3             61
#define MAKE_F4             62
#define MAKE_F5             63
#define MAKE_F6             64
#define MAKE_F7             65
#define MAKE_F8             66
#define MAKE_F9             67
#define MAKE_F10            68
#define MAKE_NUM_LOCK       69
#define MAKE_SCROLL_LOCK    70
#define MAKE_HOME           71
#define MAKE_UP             72
#define MAKE_PGUP           73
```

```
#define MAKE_NUM_MINUS      74
#define MAKE_LT             75
#define MAKE_CENTER         76
#define MAKE_RT             77
#define MAKE_NUM_PLUS       78
#define MAKE_END            79
#define MAKE_DN             80
#define MAKE_PGDWN          81
#define MAKE_INS            82
#define MAKE_DEL            83

#define BREAK_ESC           129
#define BREAK_1             130
#define BREAK_2             131
#define BREAK_3             132
#define BREAK_4             133
#define BREAK_5             134
#define BREAK_6             135
#define BREAK_7             136
#define BREAK_8             137
#define BREAK_9             138
#define BREAK_0             139
#define BREAK_MINUS         140
#define BREAK_EQUALS        141
#define BREAK_BKSP          142
#define BREAK_TAB           143
#define BREAK_Q             144
#define BREAK_W             145
#define BREAK_E             146
#define BREAK_R             147
#define BREAK_T             148
#define BREAK_Y             149
#define BREAK_U             150
#define BREAK_I             151
#define BREAK_O             152
#define BREAK_P             153
#define BREAK_LFT_BRACKET   154
#define BREAK_RGT_BRACKET   155
#define BREAK_ENTER         156
#define BREAK_CNTRL         157
#define BREAK_A             158
#define BREAK_S             159
#define BREAK_D             160
#define BREAK_F             161
#define BREAK_G             162
#define BREAK_H             163
#define BREAK_J             164
```

```
#define BREAK_K                 165
#define BREAK_L                 166
#define BREAK_SEMI              167
#define BREAK_APOS              168
#define BREAK_TILDE             169
#define BREAK_SHIFTL            170
#define BREAK_BACK_SLASH        171
#define BREAK_Z                 172
#define BREAK_X                 173
#define BREAK_C                 174
#define BREAK_V                 175
#define BREAK_B                 176
#define BREAK_N                 177
#define BREAK_M                 178
#define BREAK_COMMA             179
#define BREAK_PERIOD            180
#define BREAK_FOWARD_SLASH      181
#define BREAK_SHIFTR            182
#define BREAK_PRT_SCRN          183
#define BREAK_ALT               184
#define BREAK_SPACE             185
#define BREAK_CAPS_LOCK         186
#define BREAK_F1                187
#define BREAK_F2                188
#define BREAK_F3                189
#define BREAK_F4                190
#define BREAK_F5                191
#define BREAK_F6                192
#define BREAK_F7                193
#define BREAK_F8                194
#define BREAK_F9                195
#define BREAK_F10               196
#define BREAK_NUM_LOCK          197
#define BREAK_SCROLL_LOCK       198
#define BREAK_HOME              199
#define BREAK_UP                200
#define BREAK_PGUP              201
#define BREAK_NUM_MINUS         202
#define BREAK_LT                203
#define BREAK_CENTER            204
#define BREAK_RT                205
#define BREAK_NUM_PLUS          206
#define BREAK_END               207
#define BREAK_DN                208
#define BREAK_PGDWN             209
#define BREAK_INS               210
#define BREAK_DEL               211
```

If you have any questions, comments, or some tips on how i can improve this page, please send me some Feedback!

Contact Information

I just wanted to mention that everything here is copyrighted, feel free to distribute this document to anyone you want, just don't modify it! You can get a hold of me through my website or direct email. Please feel free to email me about anything. I can't guarantee that I'll be of ANY help, but I'll sure give it a try :-)

Email : deltener@mindtremors.com
Webpage : http://www.invsereality.org

Created by
Justin Deltener