

```

/*****
/* Program      : SWITCH.C                               */
/* Function     : Switch Sense and Control Procedures    */
/* Author      : John F. Fitter B.E.                   */
/*                                                     */
/* Rev No.    Rev date   Test date   Test platform   Description           */
/* -----    - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */
/*      00     6jun98                PIC16C77-ME      Original                */
/*                                                     */
/*          Copyright © 1998 Eagle Air Australia Pty. Ltd. All rights reserved */
/*****

```

```
#define _SWITCH_C
```

```

#include <commdefs.h>
#include "switch.h"
#include "lcd44780.h"
#include "main.h"
#include "delays.h"

```

```

// Procedure to read a 6 key keyboard arranged as normally open switch contacts connected to
// port a and one common connected to port b bit 0;
// The switch byte is a number from 0 to 7. Keys are numbered from 1 to 6, the auxilliary i/o
// is 7 (highest priority) and 0 means no key is pressed. The first scanned key which is
// pressed is the one returned.

```

```

// The procedure returns true if any switch has changed state since the last scan.
// The procedure returns false if any of the following conditions are true;
//                               the debounce delay has not expired
//                               the backlight is off (asleep)
//                               the auto-repeat timeout delay has not expired

```

```

unsigned char read_switches() {
    unsigned char sw_new;

    if(dbc_elaps) return false; // read switches if debounce expired
    else {
        dbc_elaps = debounce; // set debounce dly
        sw_new = 6; // preload the switch byte
        PORTA = 0; // clear all port A latches
        TRISA = 0x1f; // make all switches inputs
        sw_sense_dir = B_IN; // make switch sense pin an input
        aux_sense_dir = B_IN; // make aux sense pin an input
        port_b_pullups(true); // turn on port B pullups
        delay_2us(); // allow port to settle

        while(sw_new && sw_sense) { // sense switch (negative logic)
            --sw_new; // decrement the switch byte
            TRISA >>= 1; // make next switch an output
            TRISA |= 0x20; // ensure other switches are inputs
            delay_2us(); // allow port to settle
        }
        PORTA = 0; // clear port A latches
        TRISA = 0xff; // make port A all inputs

        if(!aux_sense) sw_new = KB_ENTER; // sense aux (same as the enter key)
        else if(c_status.ext_key) // external simulated keypress is
            sw_new = ser_data - 0xe0; // KB_xxxx + 0xe0

        port_b_pullups(false); // turn off port B pullups

        if(sw_new) { // is key pressed (or still pressed) ?
            if(auto_rep) return false; // in autorepeat delay, then ignore key
            if(c_status.awake) { // are we awake ?
                if(sw_new != sw_char) { // switches changed state and awake ?
                    sw_char = sw_new; // yes, save the changed switches
                    auto_rep = autorepdly; // set the autorepeat timer on keypress
                    c_status.ext_key = false; // could be from the PC so reset flag
                }
            } else { // not awake or in autorepeat delay
                goto_sleep(false); // then wake up from sleep
                return false; // and ignore the keypress
            }
            goto_sleep(false); // reset sleep timer and beep
        } else {
            sw_char = 0; // no key pressed so reset switch byte
        }
    }
}

```

```
        auto_rep = 0;                                // and reset autorepeat delay timer
    }
}
return true;
}

void init_switches() {
    PORTA = 0;                                        // clear port A latches
    TRISA = 0xff;                                    // make port A all inputs
    port_b_pullups(false);                          // turn off port B pullups
    sw_sense_dir = B_IN;                            // make switch sense pin an input
    aux_sense_dir = B_IN;                          // make aux sense pin an input
    sw_char = 0;                                    // clear key pressed
    read_switches();                                // unconditional read
}

// ***** EOF SWITCH.C *****
```