

# Arduino - ez\_switch\_lib Crib Sheet

## Library Class Initiation

Class Name: **Switches**

Class Initiation Syntax: **Switches** my\_switches(num\_switches);

where 'my\_switches' is any name you wish to use for your project and 'num\_switches' is the number of switches you will be defining. For example:

```
1.  Switches my_switches(1); // define 1 switch
2.  #define  max_switches  8
    Switches console(max_switches);
3.  #define  num_buttons    4
    #define  num_toggles    3
    Switches ms(num_buttons + num_toggles);
4.  etc
```

declare the class instance early in your code, for example after any switch data but prior to the setup() function

## Available User Accessible Library Macros Definitions

#define	Value	Associated Functions	Comments
<a href="#">button switch</a>	1	-	differentiates switch type
<a href="#">toggle switch</a>	2	-	differentiates switch type
<a href="#">circuit C1</a>	INPUT	-	switch circuit requires an external pull down 10k ohm resistor
<a href="#">circuit C2</a>	INPUT_PULLUP	-	switch circuit requires no other components beyond the switch
<a href="#">switched</a>	true	<a href="#">read_switch</a> , <a href="#">read_button_switch</a> , <a href="#">read_toggle_switch</a>	signifies switch has been pressed/switch cycle complete; note that not <a href="#">switched</a> is <a href="#">!switched</a>
<a href="#">on</a>	true	-	used for toggle switch status; note that off is <a href="#">!on</a>
<a href="#">not_used</a>	true	-	helps self document code
<a href="#">add_failure</a>	-1	<a href="#">add_switch</a>	<a href="#">add_switch</a> could not insert a given switch, i.e. no space left
<a href="#">bad_params</a>	-2	<a href="#">add_switch</a>	invalid <a href="#">add_switch</a> parameters
<a href="#">link success</a>	0	<a href="#">link_switch_to_output</a>	output successfully linked to given switch
<a href="#">link failure</a>	-1	<a href="#">link_switch_to_output</a>	output pin could not be linked to given switch
<a href="#">none_switched</a>	255	<a href="#">read_button_switch</a> , <a href="#">read_toggle_switch</a>	<a href="#">last_switched_id</a> is initialised to this value and updated every time a switch is actuated

## Available User Accessible Library Variables

Switch Control Struct(ure)	Purpose
<code>struct switch_control {</code>	the core of the library – configs and current status of all declared switches
<code>byte switch_type;</code>	type of switch connected
<code>byte switch_pin;</code>	digital input pin assigned to the switch
<code>byte switch_circuit_type;</code>	the type of circuit wired to the switch
<code>bool switch_on_value;</code>	used for BUTTON SWITCHES only - defines what "on" means
<code>bool switch_pending;</code>	records if switch in transition or not
<code>long unsigned int switch_db_start;</code>	records debounce start time when associated switch starts transition
<code>bool switch_status;</code>	used for TOGGLE SWITCHES only - current state of toggle switch
<code>byte switch_out_pin;</code>	the digital output pin mapped to this switch, if any
<code>bool switch_out_pin_status;</code>	the status of the mapped output pin
<code>} *switches;</code>	memory will be created when class is initiated

Other Variables	Purpose
<code>byte last_switched_id;</code>	the switch_id of the <u>last</u> switch to be actuated. Use this in any interrupt service routine to know which switch has been actuated

## Available User Accessible Library Functions

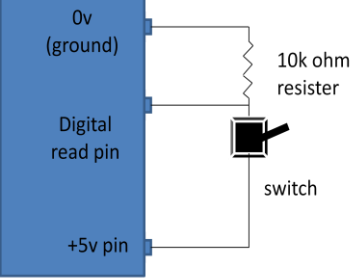
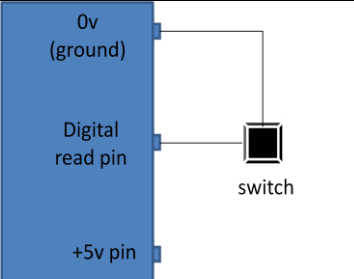
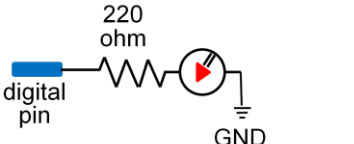
Function	Parameters	Value(s) Returned By Functions	Comments
<code>int add_switch</code>	<code>(byte sw_type, byte sw_pin, byte circ_type);</code>	<code>add_failure (-1),</code> <code>bad_params (-2)</code>	will add the specified switch to the switch control struct(ure), after which it will be available for reading
<code>int link_switch_to_output</code>	<code>(byte switch_id, byte output_pin, bool HorL);</code>	<code>link_success ( 0),</code> <code>link_failure (-1)</code>	will link the specified digital output pin to the specified switch_id, setting the output to the specified initial value (HorL)
<code>int num_free_switch_slots</code>	none	<code>&gt;= 0</code>	number of free switch slots remaining unused in the switch control structure
<code>bool read_switch</code>	<code>(byte sw_id);</code>	<code>switched (true),</code> <code>!switched (!true)</code>	will read the specified switch, irrespective of its type; will also switch(invert) ant linked output pin
<code>bool read_button_switch</code>	<code>(byte sw_id);</code>	<code>switched (true),</code> <code>!switched (!true)</code>	will read the specified button switch. <b>NO</b> linked output switching(inverting) will occur
<code>bool read_toggle_switch</code>	<code>(byte sw_id);</code>	<code>switched (true),</code> <code>!switched (!true)</code>	will read the specified toggle switch. <b>NO</b> linked output switching(inverting) will occur
<code>void print_switch</code>	<code>(byte sw_id);</code>	-	prints the switch control data for the specified switch_id
<code>void print_switches</code>	none	-	prints the switch control data for all declared switches
<code>void set_debounce</code>	<code>(int period);</code>	-	sets global debounce period to given millisecs

## Switch Mapping & Linking Table/Documentation

Project Name:							Date:	
Switch Configs				Linked Outputs			Notes	
Pin	Switch Type		Circuit Type		Pin	Initial Value		
	Button	Toggle	C1	C2		LOW		HIGH

(add more rows as needed)

## Standard & Simplest Switch & LED Wiring Schematics

<p><code>circuit_C1</code> – incorporating button <u>or</u> toggle switch <u>with</u> 10k ohm pull down resistor.</p> <p>Used with <code>pinMode</code> setting of <code>INPUT</code>, e.g.  <code>pinMode(pin_num, INPUT)</code>.</p>	 <p>Arduino Microcontroller</p> <p>Figure 1 – Circuit C1</p>
<p><code>circuit_C2</code> – incorporating button <u>or</u> toggle only.</p> <p>Used with <code>pinMode</code> setting of <code>INPUT_PULLUP</code>, e.g.  <code>pinMode(pin_num, INPUT_PULLUP)</code>.</p>	 <p>Arduino Microcontroller</p> <p>Figure 2 – Circuit C2</p>
<p>Standard wiring scheme for LED.</p>	

## Examples

```
1.  Switches my_switches(1); // define 1 switch
    ...
    byte switch_id = my_switches.add_switch(toggle_switch, 8, circuit_C1);
    if (switch_id < 0){
        // error creating a switch!
        ...
    } else {
        if (my_switches.link_switch_to_output(switch_id, LED_BUILTIN, LOW)) == link_failure {
            // error linking to output!
            ...
        }
    }
    // switch successfully created and linked


---


2.  #define max_switches 8
    byte switch_ids[max_switches];
    Switches console(max_switches);
    ...
    for (byte sw = 0; sw < max_switches; sw++){
        byte switch_id = console.add_switch(button_switch, 2 + sw, circuit_C2);
        if (switch_id >= 0){
            // switch added
            switch_ids[sw] = switch_id; // record switch's id for later use
        } else {
            // error creating a switch!
            ...
        }
    }


---


3.  do{
        if (my_switches.read_switch(switch_id) == switched){
            // switch has been actuated
            ...
        } while (true);


---


4.  do{
        if (my_switches.read_button_switch(switch_id) == switched){
            // switch has been actuated
            ...
        } while (true);


---


5.  if (console.switches[switch_id].switch_pending == true){
        // switch is in transition, waiting for completion of switching cycle
        ...
    }


---


6.  if (console.switches[switch_id].switch_type == toggle_switch &&
        console.switches[switch_id].switch_status == on){
        // this is a toggle switch which is currently on
        ...
    }
```