# Arduino/ESP 32 – **ez_switch_lib** Crib Sheet (v3.00)

## Library Class Instantiation

Class Name: **Switches**

Class Instantiation 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 |
|---|---|---|---|
| button_switch | 1 | - | differentiates switch type |
| toggle_switch | 2 | - | differentiates switch type |
| circuit_C1 | INPUT | - | switch circuit requires an external pull down 10k ohm resistor |
| circuit_C2 | INPUT_PULLUP | - | switch circuit requires no other components beyond the switch |
| circuit_C3 | INPUT_PULLDOWN | - | switch circuit requires no other components beyond the switch |
| switched | true | read_switch, read_button_switch, read_toggle_switch | signifies switch has been pressed/switch cycle complete; note that not switched is !switched |
| on | true | - | used for toggle switch status; note that off is !on |
| not_used | true | - | helps self document code |
| add_failure | -1 | add_switch | add_switch could not insert a given switch, i.e. no space left |
| bad_params | -2 | add_switch | invalid add_switch parameters |
| link_success | 0 | link_switch_to_output | output successfully linked to given switch |
| link_failure | -1 | link_switch_to_output | output pin could not be linked to given switch |
| none_switched | 255 | read_button_switch, read_toggle_switch | last_switched_id is initialised to this value and updated every time a switch is actuated |

## Available User Accessible Library Variables

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

| Other Variables | Purpose |
|---|---|
| `byte last_switched_id;` | the switch_id of the last 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 |
|---|---|---|---|
| `int  add_switch` | `(byte sw_type, byte sw_pin, byte circ_type);` | `add_failure (-1), bad_params (-2)` | will add the specified switch to the switch control struct(ure), after which it will be available for reading |
| `int  link_switch_to_output` | `(byte switch_id, byte output_pin, bool HorL);` | `link_success ( 0), link_failure (-1)` | will link the specified digital output pin to the specified switch_id, setting the output to the specified initial value (`HorL`) |
| `int  num_free_switch_slots` | none | `>= 0` | number of free switch slots remaining unused in the switch control structure |
| `bool read_switch` | `(byte sw_id);` | `switched (true), !switched(!true)` | will read the specified switch, irrespective of its type; will also switch(invert) ant linked output pin |
| `bool read_button_switch` | `(byte sw_id);` | `switched (true), !switched(!true)` | will read the specified button switch. **NO** linked output switching(inverting) will occur |
| `bool read_toggle_switch` | `(byte sw_id);` | `switched (true), !switched(!true)` | will read the specified toggle switch. **NO** linked output switching(inverting) will occur |
| `void reset_switch` | `(uint8_t sw_id)` | - | will reset the specified switch (sw_id) so that it is recorded as not in transition (ie pending) |
| `void reset_switches` | none | - | will reset ALL declared switches so that they are recorded as not in transition (ie pending) |
| `bool button_is_pressed` | `(uint8_t sw_id)` | `true` if specified switch is being pressed, `false` otherwise | examines the specified button switch and will return `true` if it is in a state of transition, ie actually being pressed, otherwise returns `false.` Note: • The function is ONLY relevant for switches that are declared as type |

| Function | Parameters | Value(s) Returned By Functions | Comments |
|---|---|---|---|
| | | | BUTTON<br>• If used with a none button switch the function will return `false`<br>• If the button switch has a linked output pin defined then this function WILL NOT automatically switch (toggle) the linked output. If this is required use the overloaded variant of this function with a second parameter of `true`<br>• This function is equivalent to its overload variant with the following parameters:<br>`button_is_pressed(sw_id,false);` |
| `bool button_is_pressed` | `(uint8_t sw_id, bool process_link)` | `true` if specified switch is being pressed, `false` otherwise | examines the specified button switch and will return `true` if it is in a state of transition, ie actually being pressed, otherwise returns `false`.<br>Note:<br>• The function is ONLY relevant for switches that are declared as type BUTTON<br>• If used with a none button switch the function will return `false`<br>• This is an overloaded variant function of the base version (`button_is_pressed(sw_id)`), and it possesses a second parameter which is should be set to either `true` or `false`, respectively indicating that any linked output should be automatically switched (toggled) or not<br>• If the button switch has no linked output then the second parameters is not relevant and the function behaves as its base variant - `button_is_pressed(sw_id);`<br>• If the button switch has a linked output and this second parameter is set to `true` then the linked output will be automatically switched (toggled) if the switch is detected to be released. Conversely, if the button switch has a linked output and this second parameter is set to `false` then the linked output WILL NOT be automatically switched (toggled) on switch release. |
| `void print_switch` | `(byte sw_id);` | - | prints the switch control data for the specified switch_id |
| `void print_switches` | `none` | - | prints the switch control data for all declared switches |
| `void set_debounce` | `(int period);` | - | 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 | C3 | | LOW | HIGH | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

(add more rows as needed)

## Standard & Simplest Switch & LED Wiring Schematics

Arduino AND ESP 32 Microcontrollers:

`circuit_C1`, equivalent to `INPUT` – incorporating button or toggle switch with 10k ohm pull down resistor.

Used with `pinMode` setting of `INPUT`, e.g.
`pinMode(pin_num, INPUT);`

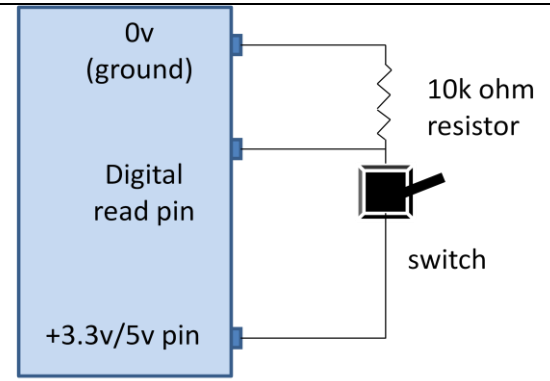
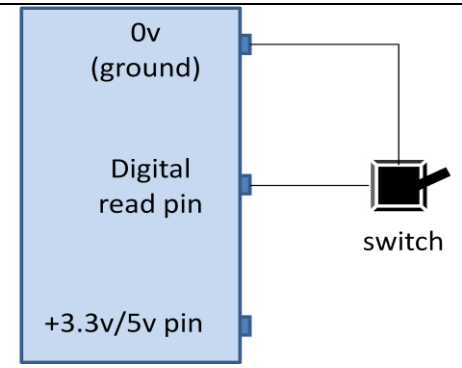
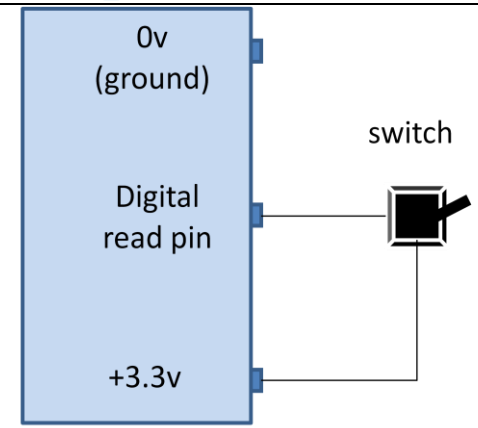
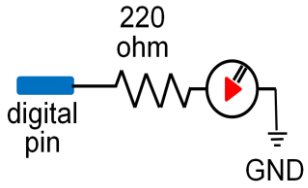



**Figure 1 – `circuit_C1` (`INPUT`)**

Note:
1. Arduino operates at 5v
2. ESP32 operates at 3.3v

| | |
|---|---|
| Arduino AND ESP 32 Microcontrollers:<br><br>`circuit_C2`, equivalent to `INPUT_PULLUP` – incorporating button or toggle, NO external resistor.<br><br>Used with `pinMode` setting of `INPUT_PULLUP`, e.g. `pinMode(pin_num, INPUT_PULLUP)`. |  <br>**Figure 2 – `circuit_C2`, (`INPUT_PULLUP`)**<br><br>Note:<br>1. Arduino operates at 5v<br>2. ESP32 operates at 3.3v |
| ESP 32 Microcontrollers ONLY:<br><br>`circuit_C3`, equivalent to `INPUT_PULLDOWN` – incorporating button or toggle, NO external resistor.<br><br>Used with `pinMode` setting of `INPUT_PULLDOWN`, e.g. `pinMode(pin_num, INPUT_PULLDOWN)` |  <br>**Figure 3 – `circuit_C3`, (`INPUT_PULLDOWN`)**<br><br>Note:<br>ESP32 operates at 3.3v |
| Standard wiring scheme for LED. |  |

## Examples

1.
```
Switches my_switches(1); // define 1 switch
...
int 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
int switch_ids[max_switches];
Switches console(max_switches);
...
for (byte sw = 0;sw < max_switches;sw++){
  // ESP 32 pins start at GPIO 25 and run to GPIO 32
  int switch_id = console.add_switch(button_switch, 25 + sw, circuit_C3);
  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.button_is_pressed(switch_id){
  // 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
  ...
}
```