1. Title: Button-Controlled ESP32 Sleep Mode

2. Problem Statement:

How to controlled esp32 Deep sleep and active mode to reduce the power consumption of esp32 while doing nothings, and also to make its state toggle whenever needed.

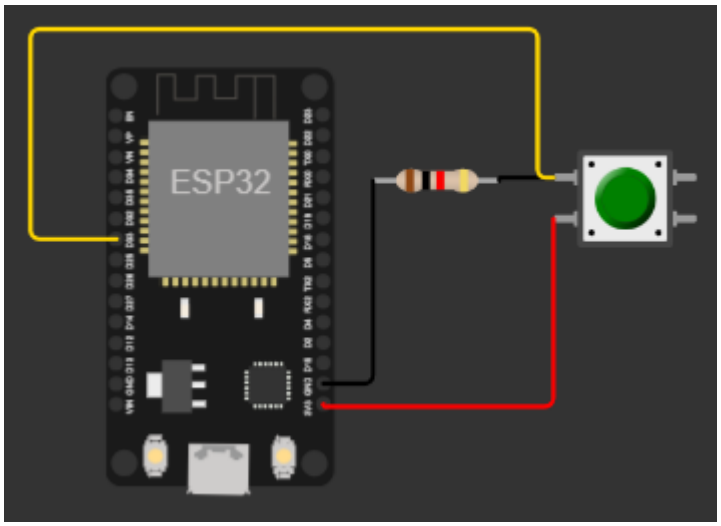
3. Requirement:

ESP32 has to go to deep sleep mode using a button to be held for 2 sec and then go back to active mode using the same button to be held for 2 seconds. If the button get release before 2000 ms then ESP32 should remain in its state either in deep sleep or in an active mood.

a. Hardware: ESP32 Development Board, a Button connected to its GPIO

b. Software: VS Code (running ESP-IDF), go to 1st link for installation idf in vsCode
Example is deep_sleep (…\esp\esp-idf\examples\system\deep_sleep)

4. Connection Diagram:



5. Example Description:

In this example, the code work like it will go to deep sleep automatically which program flash, the after every 20 sec the program will wake up (to active mode) and then go back to deep sleep again, and this process in an infinite loop. For more details you can read the “README.md” file in the project file.

Also the ULP processor wakeup call is by default so to make this code according to the project requirement make the following…

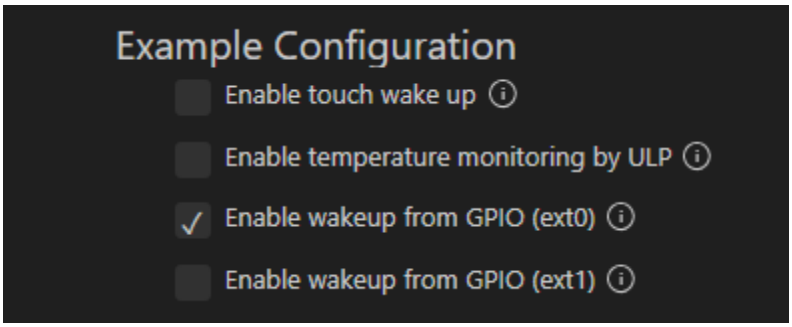
6. Changing the code up to the Requirement:

First open the example in the VS Code, and then go to the main file (deep_sleep_example_main.c)…

6.1. Configuration Changes:

Open the example and go the “menuconfig” button in bottom row for ESP-IDF SDK Configuration. After opening the file scroll down to Example configuration and tick only the “Enable wakeup from GPIO (ext0)” and Untick remainings.

Like:



6.2. Main(.c) file Changes:

Open “deep_sleep_exmaple_main.c” file, first define GPIO2 and GPIO33 for blinking led and push button respectively.

```
44  #endif
45  #endif
46
47  #define BUTTON_GPIO     GPIO_NUM_33
48  #define LED_GPIO        GPIO_NUM_2
49
50  static RTC_DATA_ATTR struct timeval sleep_enter_tim
51
52  #ifdef CONFIG_EXAMPLE_ULP_TEMPERATURE_WAKEUP
```

In this example there will also be a timer wake up, which will bring the esp32 to active mood from deep sleep after every 20 sec, so just comment out these lines also. So that you can only interface External wake up using push button. As shown below:

```
219
220     vTaskDelay(1000 / portTICK_PERIOD_MS);
221
222     /*const int wakeup_time_sec = 20;
223     printf("Enabling timer wakeup, %ds\n", wakeup_time_sec);
224     esp_sleep_enable_timer_wakeup(wakeup_time_sec * 1000000);*/
225
226 #if CONFIG_EXAMPLE_EXT0_WAKEUP
227     const int ext_wakeup_pin_0 = 39;
228
```

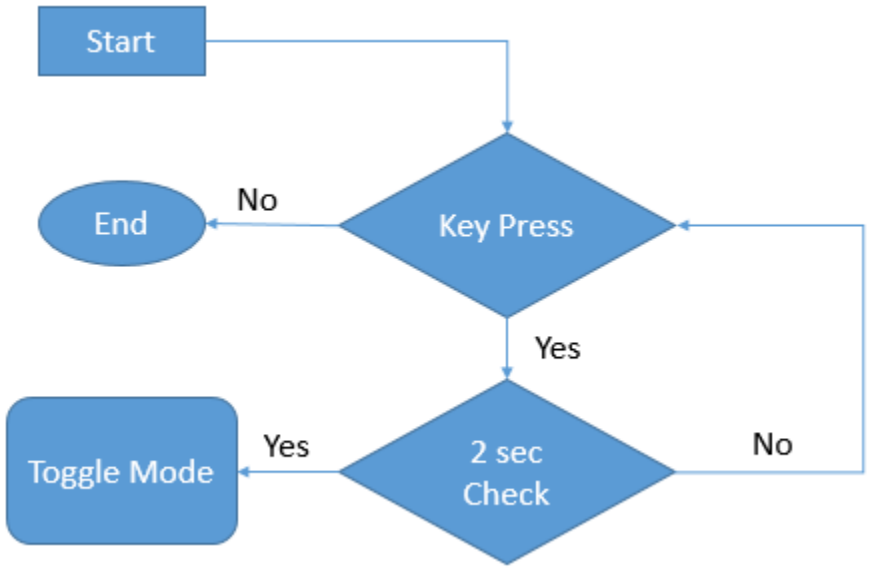
Add the following the code to the main.c file for configuring the BUTTON_GPIO and LED_GPIO pins, just above the “switch” statement as:

```
111
112     gpio_config_t io_conf;
113
114     // Configure BUTTON_GPIO as an input pin
115     io_conf.intr_type = GPIO_INTR_DISABLE; // No interrupt needed
116     io_conf.pin_bit_mask = 1ULL << BUTTON_GPIO;
117     io_conf.mode = GPIO_MODE_INPUT;
118     io_conf.pull_up_en = GPIO_PULLUP_ENABLE; // Assuming you're using a pull-up configuration
119     gpio_config(&io_conf);
120
121     // Configure LED_GPIO as an output pin
122     gpio_set_direction(LED_GPIO, GPIO_MODE_OUTPUT);
123
124     switch (esp_sleep_get_wakeup_cause()) {
```

Also comment out the following two lines:

```
346     rtc_gpio_isolate(GPIO_NUM_12);
347 #endif
348
349     //printf("Entering deep sleep\n");           //--------------------------
350     //gettimeofday(&sleep_enter_time, NULL);
351
352 #ifdef CONFIG_EXAMPLE_ULP_TEMPERATURE_WAKEUP
353 #if CONFIG_IDF_TARGET_ESP32
354     start_ulp_temperature_monitoring();
```

6.2.1. Scenario: How to detect 2 Sec key hold



At first, when the main function start it will output a message saying: "**Not a deep sleep reset**" and then it will **enable the EXT0 wake up call** by printing its message,

Now comment out the "**esp_deep_sleep_start()**" function at the end of main() function and add an 1st **infinite while loop** as:

Pseudo code:

```
While true:
    If button is pressed:
        Record current time as call1
        While button is still pressed:
            Get current time as now
            Calculate time difference as count
            If count >= 2000 and button is still pressed:
                Display "Entering deep sleep due to holding button for {count}ms"
                Delay for stability
                Record time before deep sleep as sleep_enter_time
                Enter deep sleep mode
            Else if button is released:
                Display "Button released before {count}ms holding"
                Exit button-holding loop
    Else:
        Blink the LED
        Delay for LED blinking

    Delay for loop iteration
```

```c
    //esp_deep_sleep_start();
    while (1)
    {
        if(gpio_get_level(BUTTON_GPIO) == 0)
        {
            struct timeval call1;
            gettimeofday(&call1, NULL);
            //printf("Wake up from ext0.\n");
            //printf("Wake up from button. Time spent in deep sleep: %dms\n", sleep_time_ms);
            while(1){
                struct timeval now;
                gettimeofday(&now, NULL);
                int count = (now.tv_sec - call1.tv_sec) * 1000 + (now.tv_usec - call1.tv_usec) / 1000;
                if( count >= 2000 && gpio_get_level(BUTTON_GPIO) == 0){
                    printf("Entering into deep sleep due to holding button for %dms.\n", count);
                    vTaskDelay(1000 / portTICK_PERIOD_MS);
                    gettimeofday(&sleep_enter_time, NULL);
                    esp_deep_sleep_start();
                }
                else if(gpio_get_level(BUTTON_GPIO) == 1) {
                    printf("Button got release before %dms holdings.\n", count);
                    break;
                }
            }
        }
        else {
            // Blink the LED
            printf("Blinking the led.\n");
            gpio_set_level(LED_GPIO, 1); // LED on
            vTaskDelay(250 / portTICK_PERIOD_MS);
            gpio_set_level(LED_GPIO, 0); // LED off
            vTaskDelay(250 / portTICK_PERIOD_MS);
        }

        vTaskDelay(10 / portTICK_PERIOD_MS);
    }

}
```

Explanation of the above code: in the loop will check whether the key is pressed or not, if not (**if(False)**) it will blink the led with some delay, and if the key is pressed (**if(True)**) it will get that time for the system (**using gettimeofday()**) and store it in **call1** variable, then it will enter into the **while loop** and store another time in the **now** variable, the **count** variable take the difference between the time when the button is pressed and time now while the button has been holding, and check whether the button is still pressed or not and the time duration is greater than 2 seconds or not if both conditions satisfied it will enter into the deep sleep and if not it will break the **2nd while loop** and will blink the led again and again.

6.3. **Question:** if the esp32 is in deep sleep (which means its mean processor is powered off), then how it will understand the button pressed and how will it calculate the duration for the button to be held for 2 or more seconds?

Answer: In deep sleep mode only the **RTC and its Peripherals and some IRAM** are in active mood and some **GPIO** also detects the external wake-up call. So when the button is pressed it

(RTC) will make a wakeup, so when the controller wakes up it will wait for the button hold duration if more than two seconds it will go to the main work, and if not it will go back to sleep again.

Similarly also add the following code part in the **switch case** condition of **ESP_SLEEP_WAKEUP_EXT0:** as

```
If wakeup reason is ESP_SLEEP_WAKEUP_EXT0:
    Record current time as call1
    Display "Wake up from ext0"

    While true:
        Get current time as now
        Calculate time difference as count
        If count < 2000 and button is not pressed:
            Display "Entering into deep sleep again due to not holding button for {count}ms"

            // Disable EXT0 wakeup source
            Disable EXT0 wakeup
            // Enable EXT0 wakeup source on GPIO 39
            Enable EXT0 wakeup on GPIO 39

            Enter deep sleep mode
        Else if count >= 2000 and button is pressed:
            Display "Button pressed for {count}ms"
            Display "Wake up from button. Time spent in deep sleep: {sleep_time_ms}ms"
            Delay for stability
            Exit button-holding loop

    End loop
End case
```

```
125  #if CONFIG_EXAMPLE_EXT0_WAKEUP             //-------------------------------------------------------------------------
126          case ESP_SLEEP_WAKEUP_EXT0: {
127              struct timeval call1;
128              gettimeofday(&call1, NULL);
129              printf("Wake up from ext0.\n");
130              while(1){
131                  struct timeval now;
132                  gettimeofday(&now, NULL);
133                  int count = (now.tv_sec - call1.tv_sec) * 1000 + (now.tv_usec - call1.tv_usec) / 1000;
134                  if( count < 2000 && gpio_get_level(39) == 1){
135                      printf("Entering into deep sleep agian due to not holding button for %dms.\n", count);
136
137                      // Disable EXT0 wakeup source
138                      esp_sleep_disable_wakeup_source(ESP_SLEEP_WAKEUP_EXT0);
139                      // Enable EXT0 wakeup source on GPIO 39
140                      esp_sleep_enable_ext0_wakeup(GPIO_NUM_39, 0);
141
142                      esp_deep_sleep_start();
143                  }
144                  else if(count >= 2000 && gpio_get_level(39) == 0) {
145                  //else {
146                      printf("Button got pressed for %dms.\n", count);
147                      printf("Wake up from button. Time spent in deep sleep: %dms\n", sleep_time_ms);
148                      vTaskDelay(1000 / portTICK_PERIOD_MS);
149                      break;
150                  }
151              }
152              break;
153          }
154  #endif // CONFIG_EXAMPLE_EXT0_WAKEUP
```

Code Explanation: in this piece of code, the controller will be wakeup it will check the button hold duration and then decide further as mentioned above.

7. **Issue**: in the above code, I have added this (line 138 and 140) for disabling the wake-up source and then enabling it because without this once wake-up is called, then it detached the ext0 button from the wakeup source so make it wake up again and again so I have to call it again before going to deep sleep in an Infinite while loop.

8. **Note**: in most of the **if statement** I have to make sure both the condition of time duration and button is held within that duration satisfied so that it can toggle the state, the reason is if someone pressed a button and release it and then pressed it back within that duration so it should not toggle the state.

9. **Links:**

   Getting esp idf in vs Code:

   https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md

Prepared By: Ali Huzaifa

Internee at EmbedINN Pvt. Ltd.

Date: 17 Aug 2023