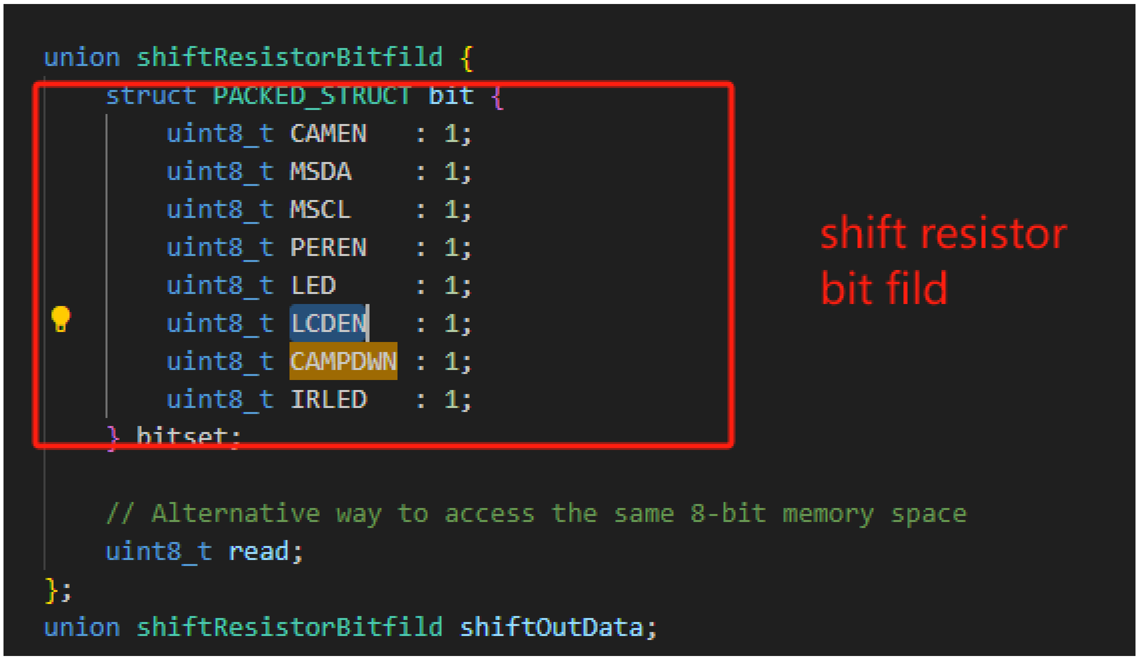
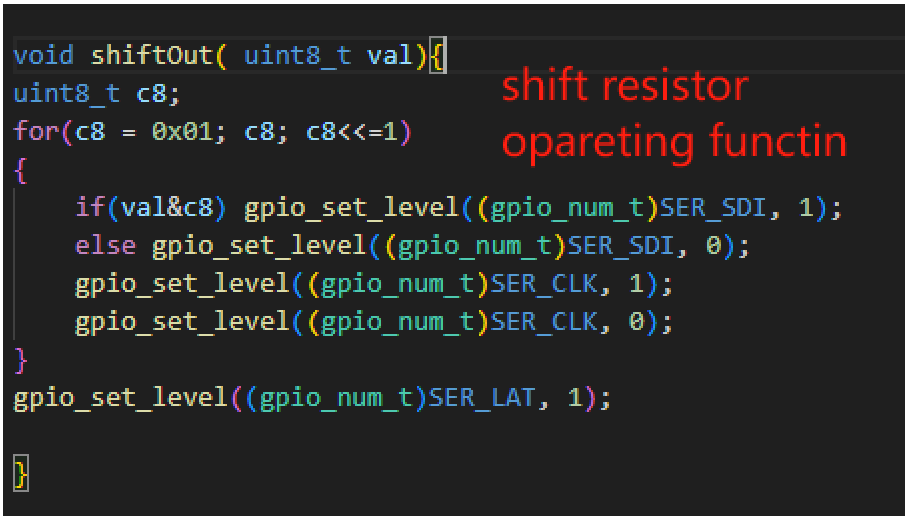
Sozib Al Mamun

Embedded Software Engineer



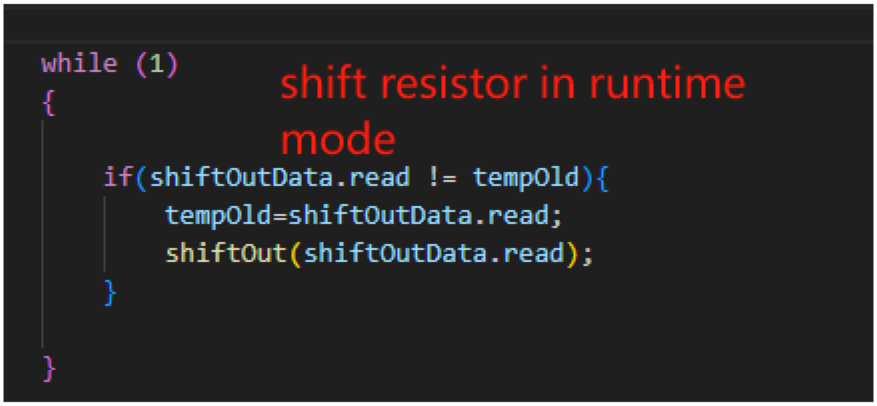
```
union shiftResistorBitfild {
    struct PACKED_STRUCT bit {
        uint8_t CAMEN   : 1;
        uint8_t MSDA    : 1;
        uint8_t MSCL    : 1;
        uint8_t PEREN   : 1;
        uint8_t LED     : 1;
        uint8_t LCDEN   : 1;
        uint8_t CAMPDWN : 1;
        uint8_t IRLED   : 1;
    } bitset;

    // Alternative way to access the same 8-bit memory space
    uint8_t read;
};
union shiftResistorBitfild shiftOutData;
```

shift resistor bit fild



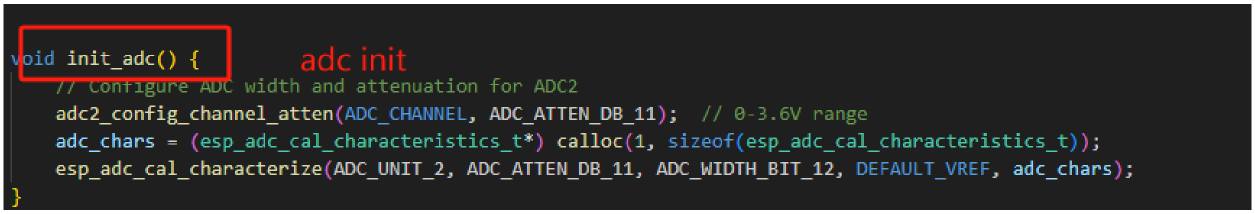
```
void shiftOut( uint8_t val){
uint8_t c8;
for(c8 = 0x01; c8; c8<<=1)
{
    if(val&c8) gpio_set_level((gpio_num_t)SER_SDI, 1);
    else gpio_set_level((gpio_num_t)SER_SDI, 0);
    gpio_set_level((gpio_num_t)SER_CLK, 1);
    gpio_set_level((gpio_num_t)SER_CLK, 0);
}
gpio_set_level((gpio_num_t)SER_LAT, 1);

}
```

shift resistor opareting functin



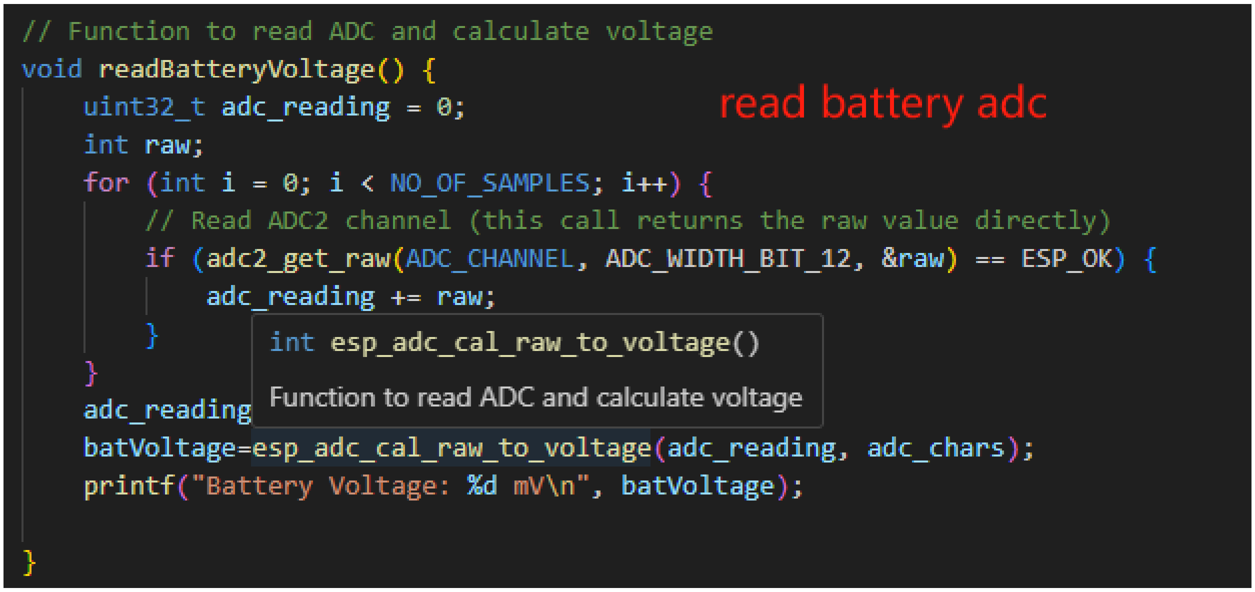
shift resistor in runtime mode

```
while (1)
{

    if(shiftOutData.read != tempOld){
        tempOld=shiftOutData.read;
        shiftOut(shiftOutData.read);
    }

}
```



adc init

```
void init_adc() {
    // Configure ADC width and attenuation for ADC2
    adc2_config_channel_atten(ADC_CHANNEL, ADC_ATTEN_DB_11);  // 0-3.6V range
    adc_chars = (esp_adc_cal_characteristics_t*) calloc(1, sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_characterize(ADC_UNIT_2, ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12, DEFAULT_VREF, adc_chars);
}
```

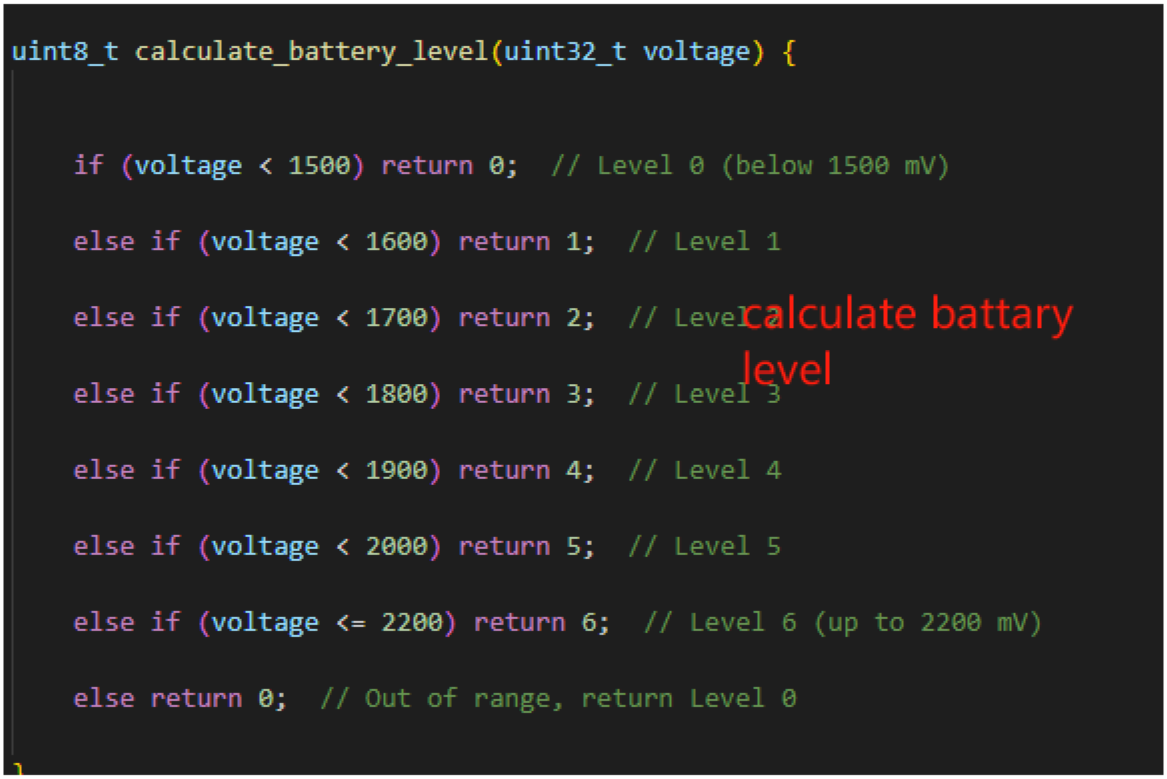


read battery adc

```
// Function to read ADC and calculate voltage
void readBatteryVoltage() {
    uint32_t adc_reading = 0;
    int raw;
    for (int i = 0; i < NO_OF_SAMPLES; i++) {
        // Read ADC2 channel (this call returns the raw value directly)
        if (adc2_get_raw(ADC_CHANNEL, ADC_WIDTH_BIT_12, &raw) == ESP_OK) {
            adc_reading += raw;
        }
    }
    adc_reading
    batVoltage=esp_adc_cal_raw_to_voltage(adc_reading, adc_chars);
    printf("Battery Voltage: %d mV\n", batVoltage);

}
```

```
int esp_adc_cal_raw_to_voltage()
```

Function to read ADC and calculate voltage



```
uint8_t calculate_battery_level(uint32_t voltage) {


    if (voltage < 1500) return 0;  // Level 0 (below 1500 mV)

    else if (voltage < 1600) return 1;  // Level 1

    else if (voltage < 1700) return 2;  // Level 2

    else if (voltage < 1800) return 3;  // Level 3

    else if (voltage < 1900) return 4;  // Level 4

    else if (voltage < 2000) return 5;  // Level 5

    else if (voltage <= 2200) return 6;  // Level 6 (up to 2200 mV)

    else return 0;  // Out of range, return Level 0

}
```

calculate battary level