```
#include <Arduino.h>

#include "main.h"

#define DALI_TX_PIN  GPIO_NUM_32
#define DALI_RX_PIN  GPIO_NUM_33
#define LED_PIN      GPIO_NUM_2
#define PB1          GPIO_NUM_21
#define PB2          GPIO_NUM_19
#define PB3          GPIO_NUM_18
#define PB4          GPIO_NUM_17
#define PB5          GPIO_NUM_34
#define PB6          GPIO_NUM_35

#define ON     1
#define OFF    0
#define PB_ON  1                // button pressed
#define PB_OFF 0                // button not pressed

#define BROADCAST_DP 0b11111110
#define BROADCAST_C  0b11111111
#define OFF_DP       0b00000000
#define ON_C         0b00000101
#define OFF_C        0b00000000
#define QUERY_STATUS 0b10010000
#define RESET        0b00100000

#define INITIALISE          0xA5
#define RANDOMISE           0xA7
#define SEARCHADDRH         0xB1
#define SEARCHADDRM         0xB3
#define SEARCHADDRL         0xB5
#define PRG_SHORT_ADDR      0xB7
#define COMPARE             0xA9
#define WITHDRAW            0xAB
#define TERMINATE           0xA1

#define START_SHORT_ADDR    0

#define DOWN                0b00000010
#define UP                  0b00000001

#define DALI_HALF_BIT_TIME        416  // [us] half bit time
#define DALI_TWO_PACKET_DELAY      10  // [ms] delay between two packets
#define DALI_RESPONSE_DELAY_COUNT  15  // [-] maximal number of half bits

#define DALI_ANALOG_LEVEL         650

unsigned long actual_time=0;           //  actual time

int active_group = 0;

//-----------------------
bool button1_old = PB_OFF;
bool button3_old = PB_OFF;
bool button2_old = PB_OFF;
bool button4_old = PB_OFF;
bool button5_old = PB_OFF;
bool button6_old = PB_OFF;

unsigned long delta_time;
unsigned long time_b2 =0;
unsigned long time_b3 =0;

String comMsg = "";                    // variable for storing input

// new specification of delay functions
// using delayMilisec() and delayMicrosec() may interfere with web interface functions
void delayMilisec(unsigned long t) {
  uint32_t s = millis();
  while(millis() - s < t) {
    yield();
  }
  return;
}

void delayMicrosec(unsigned int t) {
  uint32_t s = micros();
  while(micros() - s < t) {
    yield();
```

```
  }
  return;
}

// convert input to device encoding
// convention:
//   short address:     SA           from     0 to 63,
//   group address:     G +   100,   from   100 to 115
//   as is byte:        B + 0x100,   from 0x100 to 0x1FF
//   default broadcast: -1           in all other cases
uint8_t parse_device(int value){            // convert input to device encoding
    uint8_t device;
    if(value>=0 && value<=63)               // short address 0 - 63
    {
        device = (value << 1) | 0x01;
    }
    else if (value>=100 && value<=115)      // group address G(0 - 15) + 100
    {
        device = ((value-100) << 1) | 0x81;
    }
    else if (value>=0x100 && value<=0x1FF)  // as is byte    B(0x00-0xff) + 0x100
    {
        device = value & 0xFF;
    }
    else
    {
        device = BROADCAST_C;               // in all other cases, broadcast
    }
    return device;
}

// help for serial terminal
void PrintHelp() {
    Serial.println();
    Serial.println("Help");
    Serial.println("-------------------");
    Serial.println("-1 = broadcast");
    Serial.println("(Switch Off) off: device");
    Serial.println("(Set Max Level) smax: device light level");
    Serial.println("(Set Min Level) smin: device light level");
    Serial.println("(Set System Failure Level) sfail: device light level");
    Serial.println("(Set Power On Level) spower: device light level");
    Serial.println("(Set Fade Time) sft: device time");
    Serial.println("(SetFadeRate) sfr: device rate");
    Serial.println("(Set Short Address) ssa: device number");
    Serial.println("(Set Scene) ss: device number_of_scene light level");
    Serial.println("(Go To Scene) gts: device number_of_scene");
    Serial.println("(Identify Device) id: device");
    Serial.println("(Add To Group) atg: device number_of_group");
    Serial.println("(Remove From Group) rfg: device number_of_group");
    Serial.println("(Set Operating Mode) som: device number_of_mode(hex) (0 default)");
    Serial.println();
}

void setup() {
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);

    pinMode(DALI_TX_PIN, OUTPUT);
    digitalWrite(DALI_TX_PIN, HIGH);

    pinMode(DALI_RX_PIN, INPUT);

    // buttons
    pinMode(PB1, INPUT_PULLUP);
    pinMode(PB2, INPUT_PULLUP);
    pinMode(PB3, INPUT_PULLUP);
    pinMode(PB4, INPUT_PULLUP);
    pinMode(PB5, INPUT);           // external pull-down
    pinMode(PB6, INPUT);           // external pull-down

    Serial.begin(115200);
    Serial.println(" ");
    Serial.println("Welcome to DALI Control Software");
    Serial.println("Running ...");
}

void loop() {
    int num1, num2, num3;
```

```
//button settings-----------------------------------------------------------
    bool button1_new = !digitalRead(PB1);    // inverted
    bool button2_new = !digitalRead(PB2);    // inverted
    bool button3_new = !digitalRead(PB3);    // inverted
    bool button4_new = !digitalRead(PB4);    // inverted
    bool button5_new =  digitalRead(PB5);
    bool button6_new =  digitalRead(PB6);

// button - OFF --------------------------------------------------------------
    if(button1_old == PB_OFF && button1_new == PB_ON){
        active_group = 0;
        Off(-1);
    }
    button1_old = button1_new;

// button - Go To Scene 2 ----------------------------------------------------
    if(button2_old == PB_OFF && button2_new == PB_ON){
        SetFadeTime(-1,4);
        SetFadeRate(-1,1);
        GoToScene(-1,2);
        active_group = 2;
    }
    button2_old = button2_new;

// button - Go To Scene 3 ----------------------------------------------------
    if(button3_old == PB_OFF && button3_new == PB_ON){
        SetFadeTime(-1,4);
        SetFadeRate(-1,1);
        GoToScene(-1,3);
        active_group = 3;
    }
    button3_old = button3_new;

// button - Go To Scene 4 ----------------------------------------------------
    if(button4_old == PB_OFF && button4_new == PB_ON){
        SetFadeTime(-1,4);
        SetFadeRate(-1,1);
        GoToScene(-1,4);
        active_group = 4;
    }
    button4_old = button4_new;

// brightness increase using the button --------------------------------------
if(active_group > 0 && button6_new == PB_ON){
  actual_time=millis();
  delta_time=actual_time-time_b2;
  if(delta_time>30){
    StepUp(100+active_group);
    time_b2=actual_time;
  }
}

// brightness decrease using the button --------------------------------------
if(active_group > 0 && button5_new == PB_ON){
  actual_time=millis();
  delta_time=actual_time-time_b3;
  if(delta_time>30){
    StepDown(100+active_group);

    time_b3=actual_time;
  }
}

// --------------------------------------------------------------------------------
    if (Serial.available()) {
        char c = Serial.read();
        comMsg += c;
    }

// input handling ------------------------------------------------------------
    if(comMsg.length() > 0 && comMsg.endsWith("\n")){
        comMsg.trim();
        printf("%s\n",comMsg.c_str());
        printf("msg length = %d\n", comMsg.length());

// functions-----------------------------------------------------------------

        // handle string from serial
        if(comMsg == "help") {
            PrintHelp();
```

```
    }
    else if(sscanf(comMsg.c_str(), "on %i",&num1)==1)   // on
    {
        On(num1);
        printf("lights on\n");
    }
    else if(sscanf(comMsg.c_str(), "off %i",&num1)==1)  // off
    {
        Off(num1);
        printf("Lights off\n");
    }
    else if(sscanf(comMsg.c_str(), "smax %i %i",&num1,&num2)==2) // set max
    {
        SetMaxLevel(num1,num2);
        RecallMaxLevel(num1);
        printf("set max level\n");
    }
    else if(sscanf(comMsg.c_str(), "smin %i %i",&num1,&num2)==2) // set min
    {
        SetMinLevel(num1,num2);
        RecallMinLevel(num1);
        printf("set min level\n");
    }
    else if(sscanf(comMsg.c_str(), "sfail %i %i",&num1,&num2)==2) // set failure level
    {
        SetSystemFailureLevel(num1,num2);
        printf("set failure level\n");
    }
    else if(sscanf(comMsg.c_str(), "spower %i %i",&num1,&num2)==2) // set power on
    {
        SetPowerOnLevel(num1,num2);
        printf("set power on level\n");
    }
    else if(sscanf(comMsg.c_str(), "sft %i %i",&num1,&num2)==2) // set fade time
    {
        SetFadeTime(num1,num2);
        printf("set fade time \n");
    }
    else if(sscanf(comMsg.c_str(), "sfr %i %i",&num1,&num2)==2) // set fade rate
    {
        SetFadeRate(num1,num2);
        printf("set fade rate\n");
    }
    else if(sscanf(comMsg.c_str(), "ssa %i %i",&num1,&num2)==2) // set short address
    {
        SetShortAddress(num1,num2);
        printf("set short address\n");
    }
    else if(sscanf(comMsg.c_str(), "ss %i %i %i",&num1,&num2,&num3)==3) // set scene
    {
        SetScene(num1,num2,num3);
        printf("set scene level\n");
    }
    else if(sscanf(comMsg.c_str(), "gts %i %i",&num1,&num2)==2) // go to scene
    {
        GoToScene(num1,num2);
        printf("go to scene\n");
    }
    else if(sscanf(comMsg.c_str(), "id %i",&num1)==1) // identify device
    {
        IdentifyDevice(num1);
        printf("identify device\n");
    }
    else if(sscanf(comMsg.c_str(), "atg %i %i",&num1,&num2)==2) // add to group
    {
        AddToGroup(num1,num2);
        printf("add to group\n");
    }
    else if(sscanf(comMsg.c_str(), "rfg %i %i",&num1,&num2)==2) // remove from group
    {
        RemoveFromGroup(num1,num2);
        printf("remove from group\n");
    }
    else if(sscanf(comMsg.c_str(), "som %i %i",&num1,&num2)==2) // set operating mode
    {
        SetOperatingMode(num1,num2);
        printf("set operating mode\n");
    }
    comMsg="";
}
```

```
    delay(1);
}

// control
commands-------------------------------------------------------------------------------------------
-----

void On(int value){
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, ON_C);
    delayMilisec(DALI_TWO_PACKET_DELAY);
    Serial.println("On");
} // on

void Off(int value){
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, OFF_C);
    delayMilisec(DALI_TWO_PACKET_DELAY);
    Serial.println("Off");
} // off

void Up (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x01);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // up

void Down (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x02);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // down

void StepUp (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x03);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // step up

void StepDown (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x04);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // step down

void SetMaxLevel(int value, int maxLevel) {
    uint8_t device = parse_device(value);

    DaliTransmitCMD(0xA3, maxLevel); // DataTransferRegister DTR0
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2A);   // StoreDTRasMaxLevel
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2A);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set max level

void SetMinLevel(int value, int minLevel) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, minLevel); // DataTransferRegister DTR0
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2B);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2B);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set min level

void SetSystemFailureLevel (int value, int failureLevel) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, failureLevel); // DataTransferRegister DTR0
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2C);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2C);
    delayMilisec(DALI_TWO_PACKET_DELAY);
```

```
} // set system failure level

void SetPowerOnLevel (int value, int failureLevel) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, failureLevel); // DataTransferRegister DTR0
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2D);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2D);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set power on level

void RecallMaxLevel (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x05);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // recall max level

void RecallMinLevel (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x06);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // recall min level

void GoToScene (int value, int goToScene) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x10+(goToScene&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // go to scene

void IdentifyDevice (int value) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x25);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x25);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // identify device

void SetOperatingMode(int value, int mode) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, mode);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x23);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x23);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set operating mode

void SetFadeTime(int value, int fadeTime) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, fadeTime);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2E);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2E);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set fade time

void SetFadeRate(int value, int fadeRate) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, fadeRate);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2F);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x2F);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set fade rate

void SetScene(int value, int numberScene, int levelScene) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, levelScene);
```

```
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x40+(numberScene&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x40+(numberScene&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set scene

void SetShortAddress(int value, int shortAddress) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(0xA3, (shortAddress << 1) | 0x01);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x80);
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x80);
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // set short address

void AddToGroup (int value, int group) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x60+(group&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x60+(group&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // add to group

void RemoveFromGroup (int value, int group) {
    uint8_t device = parse_device(value);
    DaliTransmitCMD(device, 0x70+(group&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);

    DaliTransmitCMD(device, 0x70+(group&0x0F));
    delayMilisec(DALI_TWO_PACKET_DELAY);
} // remove from group

// end of
functions----------------------------------------------------------------------------------------------------
-----------

// physical layer commands
bool SearchAndCompare(long SearchAddr)
{
    bool Response = 0;

    uint8_t HighByte = SearchAddr >> 16;
    uint8_t MiddleByte = SearchAddr >> 8;
    uint8_t LowByte = SearchAddr;

    for (uint8_t i = 0; i < 3; i++)
    {
        DaliTransmitCMD(SEARCHADDRH, HighByte);
        delayMilisec(DALI_TWO_PACKET_DELAY);
        DaliTransmitCMD(SEARCHADDRM, MiddleByte);
        delayMilisec(DALI_TWO_PACKET_DELAY);
        DaliTransmitCMD(SEARCHADDRL, LowByte);
        delayMilisec(DALI_TWO_PACKET_DELAY);
    }
    DaliTransmitCMD(COMPARE, 0x00);
    delayMicrosec(7 * DALI_HALF_BIT_TIME);

    for (uint8_t i = 0; i < DALI_RESPONSE_DELAY_COUNT; i++)
    {
        if (analogRead(DALI_RX_PIN) < DALI_ANALOG_LEVEL)
        {
            Response = 1;
            digitalWrite(LED_PIN, HIGH);
            break;
        }

        delayMicrosec(DALI_HALF_BIT_TIME);
    }

    return Response;
}

void DaliTransmitCMD(uint8_t Part1, uint8_t Part2)
```

```
{
    uint8_t DALI_CMD[] = { Part1, Part2 };

    digitalWrite(DALI_TX_PIN, LOW);
    delayMicrosec(DALI_HALF_BIT_TIME);
    digitalWrite(DALI_TX_PIN, HIGH);
    delayMicrosec(DALI_HALF_BIT_TIME);

    for (uint8_t CmdPart = 0; CmdPart < 2; CmdPart++)
    {
        for (int i = 7; i >= 0; i--)
        {
            bool BitToSend = false;

            if ((DALI_CMD[CmdPart] >> i) & 1)
                BitToSend = true;

            if (BitToSend)
                digitalWrite(DALI_TX_PIN, LOW);
            else
                digitalWrite(DALI_TX_PIN, HIGH);

            delayMicrosec(DALI_HALF_BIT_TIME);

            if (BitToSend)
                digitalWrite(DALI_TX_PIN, HIGH);
            else
                digitalWrite(DALI_TX_PIN, LOW);

            delayMicrosec(DALI_HALF_BIT_TIME);
        }
    }

    digitalWrite(DALI_TX_PIN, HIGH);
}
```