Matrix-clock-esp32

1.0

Generated by Doxygen 1.12.0

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Topic Documentation

4.1 Konfiguracja

Macros

- #define **MINI**

  Tryb miniaturowy dla urządzenia.
- #define KILL_INSTEAD_OF_HALT

  Tryb WROOM (do skonfigurowania w ESP-IDF).
- #define **SEPARATOR**

  Separator danych.
- #define **DEBOUNCE_TIME_MS** 10

  Czas odbijania w ms.
- #define **CLOCK_PRIORITY** 2

  Priorytet zegara.
- #define **MOSI** 6

  Pin MOSI dla MINI.
- #define **CS** 7

  Pin CS dla MINI.
- #define **CLK** 4

  Pin CLK dla MINI.
- #define **BL** 0

  Poziom podświetlenia lewej części dla MINI.
- #define **BC** 10

  Poziom podświetlenia środkowej części dla MINI.
- #define **BR** 2

  Poziom podświetlenia prawej części dla MINI.
- #define **CORE** 0

  Rdzeń CPU dla MINI.
- #define **MAX_COUNT** 4

  Maksymalna liczba urządzeń w systemie.
- #define **MAX_DATA_SIZE_BYTES** MAX_COUNT∗16

  Maksymalny rozmiar danych w bajtach.
- #define **ROWS** 8

  Liczba wierszy w wyświetlaczu.
- #define **DISPLAY_STACK** 4096

Rozmiar stosu wyświetlacza w bajtach.

- #define **CONDITION_CHECK_INTERVAL_MS** 2

  Interwał sprawdzania warunku w ms.

- #define BLOCK_UNTIL(action, bool_condition)

  Makro blokujące do momentu spełnienia warunku.

4.1.1 Detailed Description

4.1.2 Macro Definition Documentation

4.1.2.1 BLOCK_UNTIL

```
#define BLOCK_UNTIL(
              action,
              bool_condition)
```

Value:

```
    \                                 while(!(bool_condition)){vTaskDelay(pdMS_TO_TICKS(CONDITION_CHECK_INTERVAL_MS));},
                                      bool_condition = !bool_condition;   \
                                      action;                            \
                                      bool_condition = !bool_condition
```

Makro blokujące do momentu spełnienia warunku.

Parameters

action	Akcja wykonywana po spełnieniu warunku.
bool_condition	Warunek logiczny.

4.1.2.2 KILL_INSTEAD_OF_HALT

```
#define KILL_INSTEAD_OF_HALT
```

Tryb WROOM (do skonfigurowania w ESP-IDF).

Włączenie trybu debugowania.

Zatrzymanie zadania zamiast jego zabijania.

Chapter 5

Data Structure Documentation

5.1 Clock Struct Reference

Struktura przechowująca aktualny czas.

`#include <timeControl.h>`

Data Fields

- uint8_t time [4]
- uint8_t timeChars [4][8]
- bool ready
- uint8_t intensity

5.1.1 Detailed Description

Struktura przechowująca aktualny czas.

Zawiera czas jako liczby i graficzną reprezentację w formie tablicy znaków. Przechowuje również jasność wyświetlacza oraz gotowość do aktualizacji.

Note

Flaga `ready` pozwala minimalizować błędy podczas wysyłania danych do wyświetlacza.

5.1.2 Field Documentation

5.1.2.1 intensity

`uint8_t intensity`

Jasność wyświetlacza (0-15).

5.1.2.2 ready

```
bool ready
```

Flaga gotowości do aktualizacji.

5.1.2.3 time

```
uint8_t time[4]
```

Aktualny czas w formacie liczbowym.

5.1.2.4 timeChars

```
uint8_t timeChars[4][8]
```

Graficzna reprezentacja czasu.

The documentation for this struct was generated from the following file:

- main/include/timeControl.h

Chapter 6

File Documentation

6.1 buttons.h

```
00001 #ifndef BUTTONS_H
00002 #define BUTTONS_H
00003
00004 #include <driver/gpio.h>
00005 #include <esp_log.h>
00006 #include <freertos/FreeRTOS.h>
00007 #include <freertos/task.h>
00008
00009 #include "defines.h"
00010 #include "timeControl.h"
00011
00012 #define BUTTON_CLICKED(x) gpio_get_level(x) == 0
00013 #define BUTTON_NOTCLICKED(x) gpio_get_level(x) == 1
00014
00023 static enum State {EDIT_MODE_NONE, EDIT_MODE_HOURS, EDIT_MODE_MINUTES};
00024
00037 void buttons_update(Clock *a, TaskHandle_t *t);
00038
00047 esp_err_t buttons_init();
00048
00049 #endif
```

6.2 defines.h

```
00001
00014
00016 #define MINI
00017
00019 //#define WROOM
00020
00022 //#define DEBUG
00023
00025 #define KILL_INSTEAD_OF_HALT
00026
00028 #define SEPARATOR
00029
00031 #define DEBOUNCE_TIME_MS 10
00032
00034 #define CLOCK_PRIORITY 2
00035
00036 #ifdef WROOM
00038     #ifndef MOSI
00039         #define MOSI 23
00040     #endif
00042     #ifndef CS
00043         #define CS 5
00044     #endif
00046     #ifndef CLK
00047         #define CLK 18
00048     #endif
00049     #define BL 99
00050     #define BC 99
00051     #define BR 99
00052     #define CORE 1
```

```
00053 #endif
00054
00055 #ifdef MINI
00057     #ifndef MOSI
00058         #define MOSI 6
00059     #endif
00061     #ifndef CS
00062         #define CS 7
00063     #endif
00065     #ifndef CLK
00066         #define CLK 4
00067     #endif
00068     #define BL 0
00069     #define BC 10
00070     #define BR 2
00071     #define CORE 0
00072 #endif
00073
00075 #ifndef MAX_COUNT
00076     #define MAX_COUNT 4
00078     #define MAX_DATA_SIZE_BYTES MAX_COUNT*16
00079 #endif
00080
00082 #ifndef ROWS
00083     #define ROWS 8
00084 #endif
00085
00087 #define DISPLAY_STACK 4096
00088
00090 #define CONDITION_CHECK_INTERVAL_MS 2
00091
00098 #define BLOCK_UNTIL(action, bool_condition)
      while(!(bool_condition)){vTaskDelay(pdMS_TO_TICKS(CONDITION_CHECK_INTERVAL_MS));};    \
00099                                             bool_condition = !bool_condition;   \
00100                                             action;                            \
00101                                             bool_condition = !bool_condition
```

6.3 max7219.h

```
00001 #ifndef MAX7219H
00002 #define MAX7219H
00003 #include "esp_log.h"
00004 #include <driver/spi_master.h>
00005 #include <string.h>
00006 #include "defines.h"
00007
00008 extern spi_device_handle_t SPi;
00009
00017 esp_err_t SPI_init();
00018
00031 esp_err_t max7219_sendm(const uint8_t reg, const uint8_t data);
00032
00045 esp_err_t max7219_sendrow(const uint8_t reg, const uint8_t *data);
00046
00059 esp_err_t max7219_displayTime(const uint8_t *time);
00060
00073 esp_err_t max7219_changeIntensity(const uint8_t b);
00074
00089 esp_err_t max7219_underline(uint8_t bits);
00090
00101 esp_err_t max7219_init();
00102
00112 esp_err_t max7219_clear();
00113
00114 #endif
```

6.4 timeControl.h

```
00001 #ifndef TIMECONTROL_H
00002 #define TIMECONTROL_H
00003
00004 #include <stdint.h>
00005 #include <stdbool.h>
00006 #include <string.h>
00007 #include "max7219.h"
00008
00017 static const uint8_t digits[10][8] __attribute__((section(".rodata"))) = {
00018     // Cyfra 0
00019     {
```

```
00020         0b00111100,
00021         0b01000010,
00022         0b01000110,
00023         0b01001010,
00024         0b01010010,
00025         0b01100010,
00026         0b00111100,
00027         0x00,
00028     },
00029     // Cyfra 1
00030     {
00031         0b00001000,
00032         0b00011000,
00033         0b00101000,
00034         0b00001000,
00035         0b00001000,
00036         0b00001000,
00037         0b00111110,
00038         0x00,
00039     },
00040     // Cyfra 2
00041     {
00042         0b00111100,
00043         0b01000010,
00044         0b00000010,
00045         0b00001100,
00046         0b00110000,
00047         0b01000000,
00048         0b01111110,
00049         0x00,
00050     },
00051     // Cyfra 3
00052     {
00053         0b00111100,
00054         0b01000010,
00055         0b00000010,
00056         0b00011100,
00057         0b00000010,
00058         0b01000010,
00059         0b00111100,
00060         0x00,
00061     },
00062     // Cyfra 4
00063     {
00064         0b00000100,
00065         0b00001100,
00066         0b00010100,
00067         0b00100100,
00068         0b01111110,
00069         0b00000100,
00070         0b00000100,
00071         0x00,
00072     },
00073     // Cyfra 5
00074     {
00075         0b01111110,
00076         0b01000000,
00077         0b01111100,
00078         0b00000010,
00079         0b00000010,
00080         0b01000010,
00081         0b00111100,
00082         0x00,
00083     },
00084     // Cyfra 6
00085     {
00086         0b00111100,
00087         0b01000000,
00088         0b01111100,
00089         0b01000010,
00090         0b01000010,
00091         0b01000010,
00092         0b00111100,
00093         0x00,
00094     },
00095     // Cyfra 7
00096     {
00097         0b01111110,
00098         0b00000010,
00099         0b00000100,
00100         0b00001000,
00101         0b00010000,
00102         0b00100000,
00103         0b00100000,
00104         0x00,
00105     },
00106     // Cyfra 8
```

```
00107     {
00108         0b00111100,
00109         0b01000010,
00110         0b01000010,
00111         0b00111100,
00112         0b01000010,
00113         0b01000010,
00114         0b00111100,
00115         0x00,
00116     },
00117     // Cyfra 9
00118     {
00119         0b00111100,
00120         0b01000010,
00121         0b01000010,
00122         0b00111110,
00123         0b00000010,
00124         0b00000010,
00125         0b00111100,
00126         0x00,
00127     },
00128 };
00129
00138 typedef struct {
00139     uint8_t time[4];
00140     uint8_t timeChars[4][8];
00141     bool ready;
00142     uint8_t intensity;
00143 } Clock;
00144
00153 esp_err_t clock_init(Clock *a);
00154
00161 inline bool clock_inc_intensity(Clock *a);
00162
00169 inline bool clock_dec_intensity(Clock *a);
00170
00177 inline void clock_set_intensity(Clock *a, const uint8_t i) __attribute__((weak));
00178
00185 void clock_update_timeChar(const uint8_t place, Clock *a);
00186
00193 bool clock_add_minute(Clock *a);
00194
00201 bool clock_add_hour(Clock *a);
00202
00208 void clock_sub_minute(Clock *a);
00209
00215 void clock_sub_hour(Clock *a);
00216
00224 void clock_update(Clock *a);
00225
00235 void Clock_Loop(void *); // deprecated
00236
00243 void dummy(Clock *);
00244
00245
00246 #endif
```

6.5 main/main.c File Reference

Główny plik programu dla aplikacji zarządzającej zegarem i przyciskami z użyciem ESP-IDF.

```
#include <stdio.h>
#include "esp_log.h"
#include "esp_heap_caps.h"
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <esp_attr.h>
#include "include/timeControl.h"
#include "include/max7219.h"
#include "include/buttons.h"
```

Functions

- void app_main (void)

  Główna funkcja aplikacji.

Variables

- Clock **t**

  Struktura reprezentująca zegar.
- TaskHandle_t **clockHandle**

  Handler zadania zegara.

6.5.1 Detailed Description

Główny plik programu dla aplikacji zarządzającej zegarem i przyciskami z użyciem ESP-IDF.

Author

Jan Kowalski

Anna Nowak

Date

2024-12-04

6.5.2 Function Documentation

6.5.2.1 app_main()

```
void app_main (
            void )
```

Główna funkcja aplikacji.

Funkcja inicjalizuje moduły zegara oraz przycisków, a następnie tworzy zadanie zegara na określonym rdzeniu. Na końcu wywoływana jest funkcja obsługująca aktualizację przycisków. Inicjalizacja zegara.

Funkcja `clock_init` inicjalizuje strukturę zegara.

Parameters

t	Wskaźnik do struktury zegara.

Inicjalizacja przycisków.

Funkcja `buttons_init` przygotowuje moduł obsługujący przyciski.

Tworzenie zadania zegara.

Zadanie zegara jest przypisane do określonego rdzenia.

Parameters

dummy	Funkcja reprezentująca zadanie zegara.
Clock	Nazwa zadania.
DISPLAY_STACK	Rozmiar stosu zadania.
&t	Wskaźnik do struktury zegara przekazywany do zadania.
CLOCK_PRIORITY	Priorytet zadania.
&clockHandle	Handler do utworzonego zadania.
CORE	Rdzeń, na którym zadanie ma działać.

Aktualizacja przycisków.

Funkcja `buttons_update` obsługuje aktualizację stanu przycisków.

Parameters

t	Wskaźnik do struktury zegara.
clockHandle	Handler do zadania zegara.

Index