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**

  MINI oznacza urządzenie pod które skonfigurowano wyprowadzenia i działanie. Tutaj oznacza esp32c3 super mini zamiast tego mozna uzyc WROOM.
- #define KILL_INSTEAD_OF_HALT

  WROOM oznacza urządzenie pod pod które skonfigurowano wyprowadzenia i działanie. Tutaj oznacza esp32 wroom zamiast tego mozna uzyc MINI.
- #define **SEPARATOR**

  użycie tego makra sprawia że na wyświetlaczu godziny są oddzielone od minut znakiem ':'
- #define **DEBOUNCE_TIME_MS** 10

  Czas który mija pomiędzy sprawdzaniem stanu styków, w celu zapobiegnieciu ich drgań
- #define **CLOCK_PRIORITY** 2

  Priorytet taska zegara, większy od funkcji main ponieważ chcemy mieć pewność że sprawdznie przycisków nie przeszkodzi w poprawnym odliczaniu czasu.
- #define **MOSI** 6

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

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

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

  Makro określające pin przycisku lewego dla MINI.
- #define **BC** 10

  Makro określające pin przycisku środkowego dla MINI.
- #define **BR** 2

  Makro określające pin przycisku prawego dla MINI.
- #define **CORE** 0

  Rdzeń na którym ma być wykonywane zadanie zegara w przypadku użycia urządzenia MINI.
- #define **MAX_COUNT** 4

  Maksymalna liczba wyświetlaczy podłączonych do jednego interfejsu spi, odpowiada bezpośrednio za rozmiar danych wysyłanych w pojedynczej transakcji.
- #define **MAX_DATA_SIZE_BYTES** MAX_COUNT∗16

Maksymalny rozmiar danych w bajtach wysyłanych w jednej transakcji.

- #define **ROWS** 8

  Liczba wierszy w wyświetlaczu.

- #define **CLOCK_STACK** 4096

  Rozmiar stosu dla zadania zegara w bajtach.

- #define **CONDITION_CHECK_INTERVAL_MS** 2

  makro używane podczas blokowania wątku w celu zapewnienia warunku, odpowiada za czas ponownego sprawdzenia.

- #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.

pętla while(!warunek){};

Parameters

action	Akcja wykonywana po spełnieniu warunku.
bool_condition	Warunek logiczny.

4.1.2.2 KILL_INSTEAD_OF_HALT

```
#define KILL_INSTEAD_OF_HALT
```

WROOM oznacza urządzenie pod pod które skonfigurowano wyprowadzenia i działanie. Tutaj oznacza esp32 wroom zamiast tego mozna uzyc MINI.

Makro odpowiedzialne za tryb debugowania, włącza komentarze w funkcjach

Opcja stworzona do debugowania, pozwala na zatrzymanie taska zegara zamiast jego restartowanie

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.

Struktura zawiera zmienne potrzebne do poprawnego wyświetlania i aktualizacji czasu

Note

Flaga `ready` pozwala minimalizować błędy podczas wysyłania danych do wyświetlacza, ale nie jest wymagana gdy nie mamy zamiaru bawić się przyciskami a po prostu ustawić czas

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 wyświetlacza.

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
00025 static enum State {EDIT_MODE_NONE, EDIT_MODE_HOURS, EDIT_MODE_MINUTES};
00026
00042 void buttons_update(Clock *a, TaskHandle_t *t);
00043
00053 esp_err_t buttons_init();
00054
00055 #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 CLOCK_STACK 4096
00088
00090 #define CONDITION_CHECK_INTERVAL_MS 2
00091
00100 #define BLOCK_UNTIL(action, bool_condition)
      while(!(bool_condition)){vTaskDelay(pdMS_TO_TICKS(CONDITION_CHECK_INTERVAL_MS));};    \
00101                                             bool_condition = !bool_condition;   \
00102                                             action;                            \
00103                                             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
00048 esp_err_t max7219_sendrow(const uint8_t reg, const uint8_t *data);
00049
00064 esp_err_t max7219_displayTime(const uint8_t *time);
00065
00078 esp_err_t max7219_changeIntensity(const uint8_t b);
00079
00094 esp_err_t max7219_underline(uint8_t bits);
00095
00106 esp_err_t max7219_init();
00107
00120 esp_err_t max7219_clear();
00121
00122 #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     { 0x3C, 0x42, 0x46, 0x4A, 0x52, 0x62, 0x3C, 0x00 }, // 0
00019     { 0x08, 0x18, 0x28, 0x08, 0x08, 0x08, 0x3E, 0x00 }, // 1
```

```
00020     { 0x3C, 0x42, 0x02, 0x0C, 0x30, 0x40, 0x7E, 0x00 }, // 2
00021     { 0x3C, 0x42, 0x02, 0x1C, 0x02, 0x42, 0x3C, 0x00 }, // 3
00022     { 0x04, 0x0C, 0x14, 0x24, 0x7E, 0x04, 0x04, 0x00 }, // 4
00023     { 0x7E, 0x40, 0x7C, 0x02, 0x02, 0x42, 0x3C, 0x00 }, // 5
00024     { 0x3C, 0x40, 0x7C, 0x42, 0x42, 0x42, 0x3C, 0x00 }, // 6
00025     { 0x7E, 0x02, 0x04, 0x08, 0x10, 0x20, 0x20, 0x00 }, // 7
00026     { 0x3C, 0x42, 0x42, 0x3C, 0x42, 0x42, 0x3C, 0x00 }, // 8
00027     { 0x3C, 0x42, 0x42, 0x3E, 0x02, 0x02, 0x3C, 0x00 }, // 9
00028 };
00029
00038 typedef struct {
00039     uint8_t time[4];
00040     uint8_t timeChars[4][8];
00041     bool ready;
00042     uint8_t intensity;
00043 } Clock;
00044
00053 esp_err_t clock_init(Clock *a);
00054
00061 inline bool clock_inc_intensity(Clock *a);
00062
00069 inline bool clock_dec_intensity(Clock *a);
00070
00077 inline void clock_set_intensity(Clock *a, const uint8_t i) __attribute__((weak));
00078
00085 void clock_update_timeChar(const uint8_t place, Clock *a);
00086
00093 bool clock_add_minute(Clock *a);
00094
00101 bool clock_add_hour(Clock *a);
00102
00108 void clock_sub_minute(Clock *a);
00109
00115 void clock_sub_hour(Clock *a);
00116
00124 void clock_update(Clock *a);
00125
00135 void Clock_Loop(void *); // deprecated
00136
00145 void dummy(Clock *);
00146
00147
00148 #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

Sławomir Matonóg
Maciej Szymonek

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(w c3 musi to być rdzeń 0 ale w wroomie niekoniecznie). 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