ALarm Clock Project using ESP32 Microcontroller

Generated by Doxygen 1.9.1

Chapter 1

Alarm clock

This project is an alarm clock as described below:

- The clock time starts at 00:00 (developer can set an initial time on code). Alarm start set to 00:00 and the initial alarm enable state will be defined by the slide switch value (on or off).
- Clock immediately starts ticking. The 4 digit seven-segment display has a colon (:) in the middle for second counting, the clock should blink this colon once every half second (so one blink period is 1 second).
- Clock shows in the 4 seven-segment displays the hour and minute in 24h format with zero padding. So 1:24am is 01:24 and 1:24 pm is 13:24.
- Clock have 4 buttons to control and configure time and alarm: MENU, PLUS (+), MINUS (-) and OK.
- Clicking on the MENU button each time will cycle to a different menu option:

    SET -> To set up the clock time
    AL -> To set up the clock alarm time
    (go back to show time)

- Setting up the time:
  - When the clock is showing the time, click on the MENU button once. (Displays show SET)
  - Click the OK button, the display show the time again, but this time with the HOUR blinking once every second (and the colon steady) to indicate the user can change the hour.
  - Use the +/- buttons to increment/decrement the hour. Click the OK button to set the hour.
  - Now the minutes blink to indicate the user can change the minutes.
  - Use the +/- buttons to increment/decrement the minutes. Click the OK button when done.
  - Now the clock show the new configured time (and colon continue blinking again)
- Setting up the alarm:
  - When the clock is showing the time, click on the MENU button once and then once again (display show AL).
  - Click the OK button, two scenarios can happen here:
    - If the alarm switch is in the enable position:
      - The display show the alarm time with the HOUR blinking once every second (and the colon steady) to indicate the user can change the alarm hour.
      - Use the +/- buttons to increment/decrement the hour. Click the OK button to set the hour. Now the minutes blink to indicate the user can change the minutes.
      - Use the +/- buttons to increment/decrement the minutes. Click the OK button when done.

        - Now the clock go back to show the time and alarm be enabled.
    - If the alarm switch is in the disabled position:
        - The display show Off and after a few seconds it will go back to show the time because the alarm is disabled.

- Triggering alarm:

  At any moment, if the alarm is enabled and the clock time matches the configured alarm time, the buzzer make an alarm sound and the display starts blinking the time once every half second until the user clicks on the OK button which will stop the alarm sound/blink.

1.1 Getting Started

You can either compile and run the code using the online Wokwi simulator, or offline using VSCode with Wokwi and PlatformIO IDE extensions.

1.1.1 Using the Wokwi Online Simulator

Create a new project in Wokwi (www.wokwi.com), select the ESP32 and Arduino framework:

- Replace the following files content on just created project with the ones in this repository (sketch.ino, diagram.json, libraries.txt)
- Create all the other files on the project (clock.cpp, clock.h, alarm_tone.cpp, alarm_tone.h, tm1637.cpp, tm1637.h) and copy the contents of them.

Now you are ready to start working on the project

1.1.2 Using the VSCode with PlatformIO and Wokwi extensions.

- Install VSCode.
- Install the `Wokwi Simulator` extension
- Install `PlatformIO IDE` extension.
- Open the root directory of the project in VSCode from `File > Open Folder` menu.
- Go to platformio from the left-side bar and click `Build` to compile the code.
- Open the `diagram.json` file to run the simulation.

1.2 License

`License`

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 AlarmTone Class Reference

Public Member Functions

- void **init** (uint8_t pin)
- void **play** ()
- void **stop** ()

Private Attributes

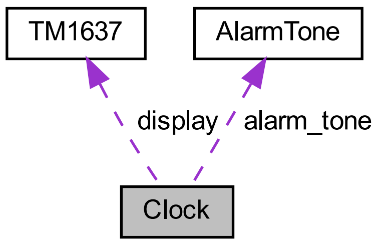
- uint8_t **_pin**
- bool **_playing**
- uint8_t **_tone_index**
- unsigned long **_last_tone_time**

The documentation for this class was generated from the following files:

- src/alarm_tone.h
- src/alarm_tone.cpp

4.2 Clock Class Reference

Collaboration diagram for Clock:



Public Member Functions

- Clock ()

  An empty Clock constructor.
- void init (TM1637 ∗display, uint8_t buzzer_pin)

  Initialize internal variables, set display to use and buzzer pin.
- void set_time (uint8_t hours, uint8_t minutes, uint8_t seconds)

  Set the time hour, minutes and seconds to internal binary representation.
- void set_alarm (uint8_t hours, uint8_t minutes)

  Set the alarm hour, minutes and seconds.
- void check_alarm ()

  Check if alarm needs to be triggered.
  Called by the ISR. If the current time equals the alarm time. It changes the state to `STATE_ALARM` and sets the alarm down counter to 30 seconds. also modifies the blinking state to blink both the left and rigth digits and the midddle colon.
- void show ()

  Show the time, alarm, or menu on display.
- void run ()

  Start running the clock This function MUST not block, everything should be handled by interrupts.
- void setup_timer ()

  Attaches the class member timer to the interrupt service routine to run the interrupt every 0.5 seconds.
- void update_time ()

  Attaches the class member timer to the interrupt service routine to run the interrupt every 0.5 seconds.
- void set_temp_time (int8_t offset)

  A temporary variable to hold time adjusted by plus and minus buttons.The time isn't stored unless the OK button is pressed. When in the set menus (for the alarm and the clock), this function modifies the time on the display by an offset.
- void commit_temp_time ()

  Commit (store) the value of the temporary time variable in the final storage.
- void handleButtonMenuPress ()

  Handles Menu button press.
- void handleButtonOkPress ()

  Handles OK button press.
- void handleButtonPlusPress ()

  Handles + button press.
- void handleButtonMinusPress ()

  Handles − button press.
- void handleSwitchAlarmChange (bool alarm_pin)

  Enables or disables alarm. Handles the alarm switch change.

Private Attributes

- TM1637 ∗ display = NULL

  7-segment Display
- hw_timer_t ∗ timer = NULL

  Timer variable to count time.
- AlarmTone ∗ alarm_tone

  The buzzer variable. Pointing to the buzzer object.
- uint32_t **time** = 0
- uint32_t **alarm** = 0
- uint32_t ∗ time_to_set = nullptr

A pointer of the current time to set. Points to either clock or alarm.

- uint32_t temp_time = 0
- uint32_t timestamp = 0

  timestamp in milliseconds. Used for incrementing the time counter and advancing the clock.
- uint8_t state = STATE_CLOCK

  Current state of the clock.
- uint8_t set_digit = DIGITS_LEFT

  The current digit in focus in the SET or Alarm Menu.
- bool **alarm_enabled** = 0
- uint8_t blink_state = POINT

  Blinking state: middle point (colon), left two digits, right two digits.
- uint8_t display_state = DIGITS_LEFT | POINT | DIGITS_RIGHT

  Display state: middle point (colon), left two digits, right two digits.
- uint8_t alarm_off_counter

  Counter for Alarm off display message.
- uint8_t alarm_counter

  Counter for Alarm sound and display.

4.2.1 Member Function Documentation

4.2.1.1 commit_temp_time()

```
void Clock::commit_temp_time ( )
```

Commit (store) the value of the temporary time variable in the final storage.

The final storage can be either the `time` or `alarm` varialbe. Depends on the `time_to_set` class member (pointer variable).

4.2.1.2 init()

```
void Clock::init (
            TM1637 * display,
            uint8_t buzzer_pin )
```

Initialize internal variables, set display to use and buzzer pin.

Parameters

display	The 7-segment display object, an instance of the TM1637 class.
buzzer_pin	The buzzer output pin number.

4.2.1.3 set_alarm()

```
void Clock::set_alarm (
             uint8_t hours,
             uint8_t minutes )
```

Set the alarm hour, minutes and seconds.

See `set_time()` method.

Parameters

hours	Hours.
minutes	Minutes.

4.2.1.4 set_temp_time()

```
void Clock::set_temp_time (
             int8_t offset )
```

A temporary variable to hold time adjusted by plus and minus buttons.The time isn't stored unless the OK button is pressed. When in the set menus (for the alarm and the clock), this function modifies the time on the display by an offset.

Parameters

offset	An offset to increment the Clock::time variable with. Negative offset decrements the time.

4.2.1.5 set_time()

```
void Clock::set_time (
             uint8_t hours,
             uint8_t minutes,
             uint8_t seconds )
```

Set the time hour, minutes and seconds to internal binary representation.

The class member time variable is a `uint32_t` number which represents the hour|min|secs in a binary format (17 bits):

16 15 14 13 12	11 10 9 8 7 6	5 4 3 2 1 0
H H H H H	m m m m m m	s s s s s s

For example, the number: 76717 in binary:

```
1 0 0 1 0 | 1 0 1 1 1 0 | 1 0 1 1 0 1
```

Means: 1 0 0 1 0 -> 18 (hour)
1 0 1 1 1 0 -> 46 (min)
1 0 1 1 0 1 -> 45 (sec)

So this is 18:46:45

Parameters

hours	The hours (24 hour format).
minutes	Minutes
seconds	Seconds

4.2.1.6 setup_timer()

```
void Clock::setup_timer ( )
```

Attaches the class member timer to the interrupt service routine to run the interrupt every 0.5 seconds.

Source: `https://www.electronicwings.com/esp32/esp32-timer-interrupts`

4.2.1.7 show()

```
void Clock::show ( )
```

Show the time, alarm, or menu on display.

This function checks the current state stored in the class member variable `state` and changes the 7-segment display accordingly.
The blinking is controlled by the `blink_state` variable.
For example:
If `blink_state = 0b100` (blinking the middle colon), and `display_state` = 0b111 (display all the objects hours, minutes, and middle colon), a call to the function every 0.5 seconds will cycle the display state betwen `0b111` to `0b011`, creating the blinking effect.

The xor operation is used to toggle the display:

$$\text{display\_state} = \text{display\_state} \oplus \text{blink\_state}$$

4.2.1.8 update_time()

```
void Clock::update_time ( )
```

Attaches the class member timer to the interrupt service routine to run the interrupt every 0.5 seconds.

Source: `https://docs.espressif.com/projects/arduino-esp32/en/latest/api/timer.↩ html`

Increments the timestamp by 0.5 seconds on every call.

Adds 0.5 seconds (500 milliseconds). Resets the counter every day.
$\text{timestamp} = (\text{timestamp} + 500) \bmod (24 \times 60 \times 60 \times 1000)$

4.2.2 Member Data Documentation

4.2.2.1 temp_time

```
uint32_t Clock::temp_time = 0  [private]
```

The variable on display that is being modified in the set menu.
This variable isn't stored unless the OK button is pressed. Pressing the menu button cancels the variable storage.

The documentation for this class was generated from the following files:

- src/clock.h
- src/clock.cpp

4.3 TM1637 Class Reference

Public Member Functions

- **TM1637** (uint8_t, uint8_t)
- void **init** (void)
- int **writeByte** (int8_t wr_data)
- void **start** (void)
- void **stop** (void)
- void **display** (int8_t DispData[ ])
- void **display** (uint8_t BitAddr, int8_t DispData)
- void **displayNum** (float num, int decimal=0, bool show_minus=true)
- void **displayStr** (char str[ ], uint16_t loop_delay=500)
- void **clearDisplay** (void)
- void **set** (uint8_t=BRIGHT_TYPICAL, uint8_t=0x40, uint8_t=0xc0)
- void **point** (boolean PointFlag)
- void **coding** (int8_t DispData[ ])
- int8_t **coding** (int8_t DispData)
- void **bitDelay** (void)

Public Attributes

- uint8_t **cmd_set_data**
- uint8_t **cmd_set_addr**
- uint8_t **cmd_disp_ctrl**
- boolean **_PointFlag**

Private Attributes

- const int **DIGITS** = 4
- uint8_t **clkpin**
- uint8_t **datapin**

The documentation for this class was generated from the following files:

- src/tm1637.h
- src/tm1637.cpp

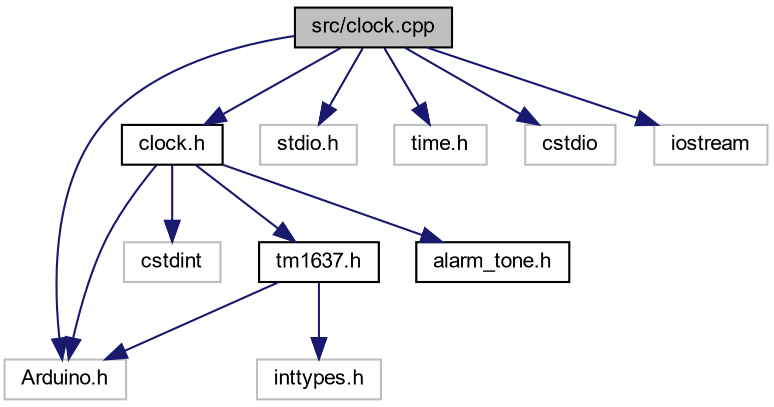
Chapter 5

File Documentation

5.1 src/clock.cpp File Reference

```
#include <Arduino.h>
#include "clock.h"
#include "stdio.h"
#include "time.h"
#include <cstdio>
#include <iostream>
```

Include dependency graph for clock.cpp:



Functions

- void ARDUINO_ISR_ATTR onTimer ()

  The interrupt service routine for the clock timer. This iterrupt is called every 0.5 seconds.

5.1.1 Detailed Description

Implementation of the Clock class.

This file contains the implementation to the clock class.

5.1.2 Function Documentation

5.1.2.1 onTimer()

```
void ARDUINO_ISR_ATTR onTimer ( )
```

The interrupt service routine for the clock timer. This iterrupt is called every 0.5 seconds.

An explanation of how to use timer interrupts can be found in
`Arduino-ESP32 Timer API`

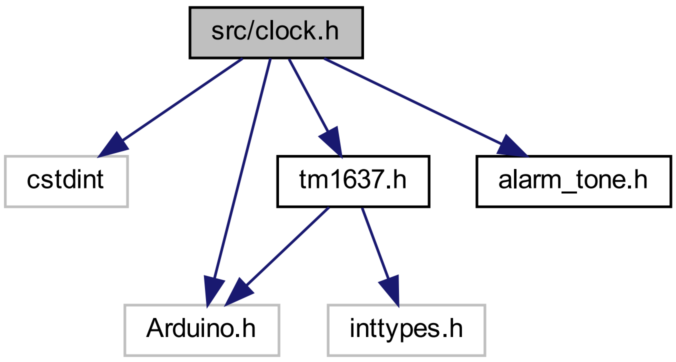
Returns

void

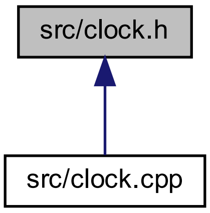
5.2 src/clock.h File Reference

```
#include <cstdint>
#include <Arduino.h>
#include "tm1637.h"
#include "alarm_tone.h"
```

Include dependency graph for clock.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Clock

Enumerations

- enum ClockState {
  STATE_CLOCK = 0 , STATE_MENU_SET = 1 , STATE_MENU_ALARM = 2 , STATE_SET_CLOCK = 3 ,
  STATE_SET_ALARM = 4 , STATE_ALARM_OFF = 5 , STATE_ALARM = 6 }

  An enum to define the states of the clock.
- enum DigitState { DIGITS_LEFT = 0b10 , DIGITS_RIGHT = 0b01 , POINT = 0b100 }
- enum ButtonType { **BUTTON_MENU** , **BUTTON_PLUS** , **BUTTON_MINUS** , **BUTTON_OK** }

  Button type enum.

Variables

- Clock **clk**

5.2.1 Detailed Description

Interfaces the Clock class.

This header defines the interface to the clock class.

5.2.2 Enumeration Type Documentation

5.2.2.1 ClockState

```
enum ClockState
```

An enum to define the states of the clock.

Enumerator

STATE_CLOCK	Normal state. Display clock.
STATE_MENU_SET	Menu (Displaying "SET")
STATE_MENU_ALARM	Menu (Displaying "AL")
STATE_SET_CLOCK	Set clock state. Blinking the selected digit being set.
STATE_SET_ALARM	Set alarm state. Blinking the selected digit being set.
STATE_ALARM_OFF	Menu after selecting alarm if the alarm is off (Displaying "OFF")
STATE_ALARM	The alarm state. The buzzer sounds and the display is blinking with the alarm time.

5.2.2.2 DigitState

```
enum DigitState
```

Enumerator

DIGITS_LEFT	left digits of the display (hours)
DIGITS_RIGHT	right digits of the display (minutes)
POINT	Point (middle colon)

Index