

# **AWAC02**

*Documentation*

### **1 Overview**

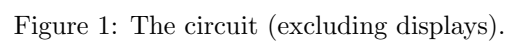
The documentation is meant as a guide for the developer. Additional aid is found in the source header files where most functions are documented.

## 2 Source Files

In this section an overview of the project files are given. The files are listed in order of relevance.

File(s)	Description
<code>main.c</code>	The main file. Contains the clock-mode state-machine.
<code>constants.h</code>	Global constants for defining number of displays, enabling logging etc.
<code>menu.h, menu.c</code>	The state machine for the menu.
<code>utilities.h, utilities.c</code>	High level utilities. Mostly for applying and showing settings.
<code>log.h, log.c</code>	Logging functions. Everything here will be unavailable if the LOG constant is undefined.
<code>rtc.h, rtc.c</code>	High level functions to set and show the external RTC clock.
<code>display.h, display.c</code>	Low level functions to interface with the Qwiic Alphanumeric Display
<code>helpers.h, helpers.c</code>	Simple helper-functions.
<code>external_interrupts.h, external_interrupts.S</code>	Register and extract the external interrupts (button presses and RTC alarms).
<code>time.h, time.S</code>	Start and use the hardware times.
<code>sleep.h, sleep.S</code>	Set the processor sleep state.
<code>twi.h, twi.S</code>	Interface to the TWI (I2C) hardware. Read and write to external devices.
<code>eeeprom.h, eeeprom.S</code>	Read and write to the EEPROM (non-volatile storage).
<code>flash.h, flash.S</code>	Read constant data from flash. For example an ASCII conversion table.
<code>io.h, io.S</code>	Direct control of pins.
<code>usart.h, usart.S</code>	Send data through USART. This is used to implement printf in 'log.c'
<code>math.S</code>	Low level math for assembly routines.

In figure 1 the circuit is shown. The value of R1-R7, C1 and C5-C7 is only approximate. You can try different values. In addition to these components two Sparkfun Qwiic Alphanumeric Displays are used. One of the displays must be modified to disable the pull-up resistors and change the slave address to 0x71.



## 4 State Machine

The state machine for clock mode is in `main.c` and the state machine for the menu mode is in `menu.c`. They are implemented using the `goto` statement which can lead to hard to understand code. To give the programmer a better overview of the states and make development easier some states are documented here and shown in flow-charts. The program starts in the `enter_clock_mode` state. The clock mode state machine is illustrated in Figure 2.

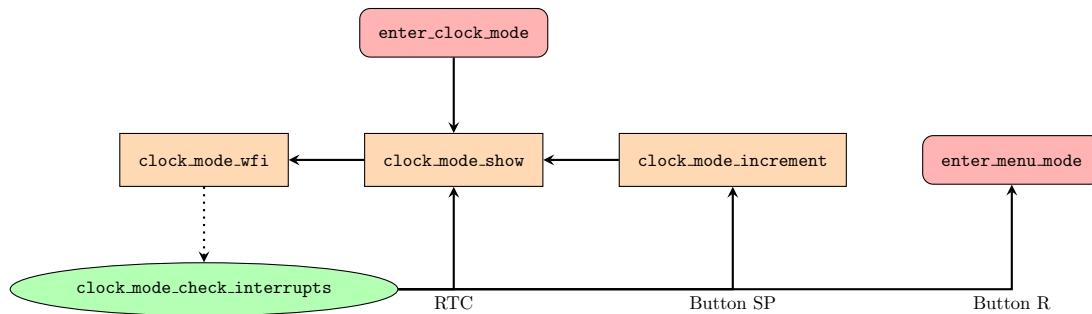


Figure 2: The clock mode state machine

The menu mode state machine is a bit more complicated. A simple overview is shown in Figure 4. There are three basic state types in the menu. *Menu* states, *choose* states and *set* states. The menu states are very simple. Each show some text on the display, then wait for a button interrupt. If the L or R button is pressed the state is changed to the previous or next menu state. If the SP button is pressed it is changed to a choose or set state.

A choose state is used when there is multiple options for the setting that is edited. For example when changing time. A choose state first shows the current value of that setting, sets an RTC alarm for when it is changed and waits for an interrupt. If the interrupt is from the RTC the new value is shown. If it is from the L or R buttons the state changes to the previous or next choose state. If it is from the SP button the state changes to the corresponding set state.

A set state works a lot like a choose state. A RTC interrupt changes the value shown and SP interrupt changes state (go back to the choose state). But a L or R button interrupt de- or increments the value instead.

Figure 3 shows the program flow between a choose and a set state in detail. The actual implementation varies slightly between types.

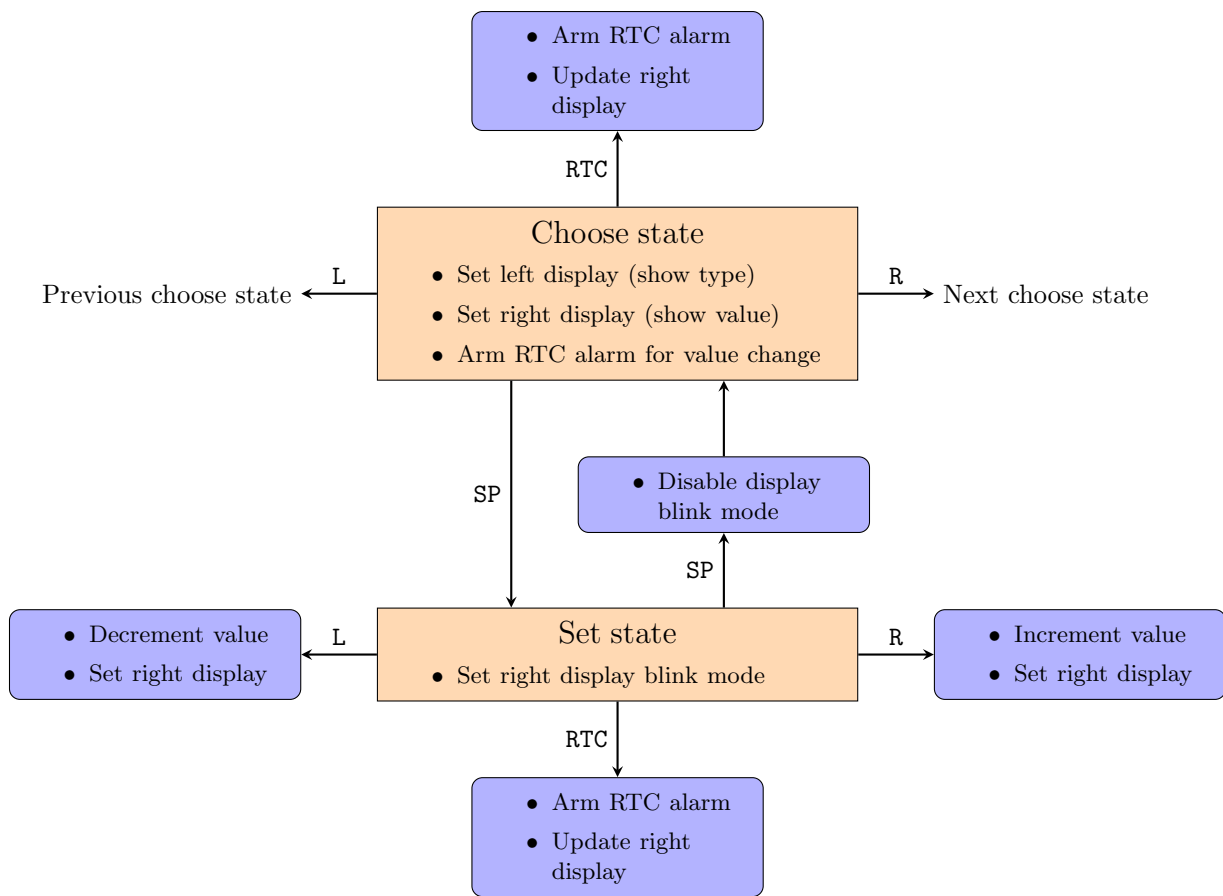


Figure 3: A choose and set state combo

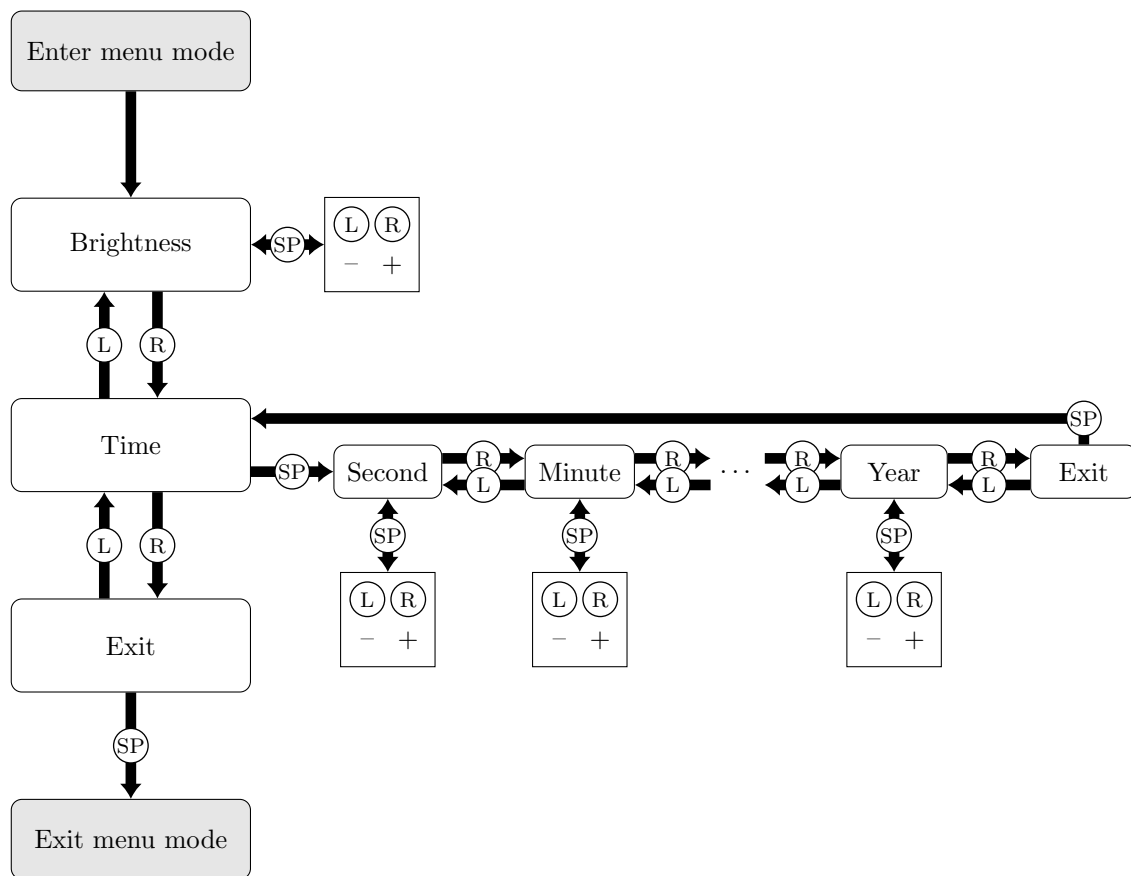


Figure 4: An overview of the menu