# Lab 2: Clock – HCS12 Interrupts & I/O

# Dokumentation

**Tim Jauch**

**Ergün Bickici**

# Contents

# List of Figures

# List of Tables

# 1 Requirements

The requirements for the clock and the thermometer are as follows: The current time should be displayed on the LCD in the format HH:MM:SS and updated every second (hour range: 0–23, minute and second range: 0–59). The LED on Port B.0 should toggle every second. For more details, refer to chapter A.2 in the appendix. At program start, the time should be initialized to 11:59:30. By briefly pressing button SW2, the clock should switch from Normal Mode to Set Mode. In Set Mode, the user can adjust the hours, minutes, and seconds by pressing buttons SW3, SW4, and SW5. While Set Mode is active, the LED on Port B.7 should be turned on. In addition to the time, the LCD should also display the temperature (e.g., 25°C). The temperature is measured using a temperature-sensitive resistor (simulated in the lab by a potentiometer) connected to the analog port AD.7. The voltage range is 0–5 V, representing a temperature of –30 to +70°C. Your program must read the analog voltage and convert it into a temperature value in "°C." The temperature display should also update every second. The first line of the LCD display should periodically toggle between the names of all group members and the text "© IT SS2025.".

# 2 User Interface description

## 2.1 LCD display

- 1st line: Names of all group members or "© IT SS2025" alternating every 10s
- 2nd line: 23:55:31 25oC Current time in format HH:MM:SS decimal, left aligned. Temperature in degrees Celsius decimal, right aligned. Numbers ¡ 10 shall be displayed without leading zeros, no "+" sign for positive values!

## 2.2 LED display

- LED0: toggles once per second (in Normal Mode and in Set Mode)
- LED7: off in Normal Mode, on in Set Mode.

## 2.3 Operating Buttons

- SW2: Short press to switch to Set Mode, press again to switch back to Normal Mode
- SW3: Press to increment the hours (in clock Set Mode only)
- SW4: Press to increment the minutes (in clock Set Mode only)
- SW5: Press to increment the seconds (in clock Set Mode only)

# 3 Debugging

Needed to change the value to turn on **ECT**:

```
TIMER_ON equ $80 ; tscr1 value to turn ECT on (value turn on)
```

Needed to change value for **AND** and **XOR** operation to set the timer prescaler to 128, which equals a timer driver frequency of 187500 Hz:

```
ldab TSCR2
andb #$80
orab #7
stab TSCR2
```

# 4 Data Dictionary

| **Module** | **Variable** | **C-like datatype** | **Purpose** |
|---|---|---|---|
| clock.c, main.c | seconds | unsigned char | counting up seconds |
| clock.c, main.c | minutes | unsigned char | counting up minutes |
| clock.c, main.c | hours | unsigned char | counting up hours |
| main.c, constant.h | SELECT12HOURS | unsigned char | switch between 12/24h model |

Table 1: Module variables and their purpose

# 5 Module Overview

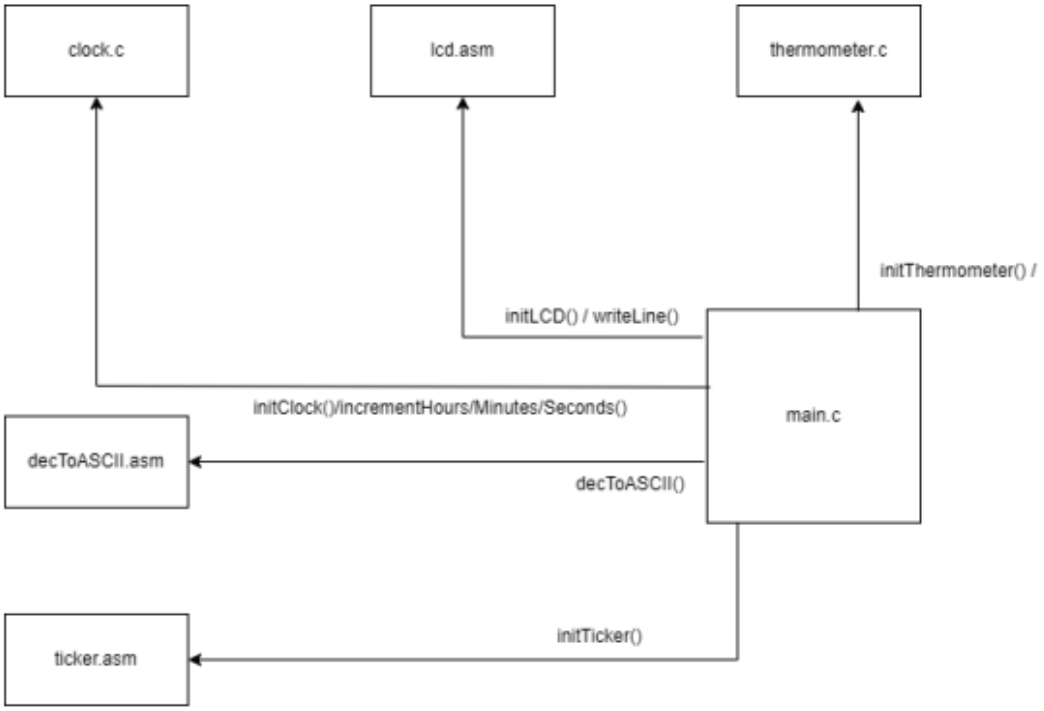



Figure 1: Module Overview

# 6 Interface Descriptions of All Subroutines

## 6.1 main.c

**decToASCII_Wrapper** Wrapper function for `decToASCII`.

- **Parameters:** `char*` pointer to the destination string; `int` number to convert.
- **Return:** -

**WriteLine_Wrapper** Wrapper function for `writeLine`.

- **Parameters:** `char*` pointer to LCD print location; `int` LCD line definition.
- **Return:** -

**initLED_C** Initializes the output port for LEDs and LCD (called once).

- **Parameters:** -
- **Return:** -

**toggleLED_C** Switches LED state.

- **Parameters:** `char` bitmask to toggle LEDs.
- **Return:** -

**incrementCounter** Advances to the next entry in the array of strings for LCD line 0.

- **Parameters:** `unsigned char*` pointer; `unsigned char` max value.
- **Return:** -

**isButtonPressed** Checks if a button is pressed.

- **Parameters:** `char` button to check.
- **Return:** `char` button state (on/off).

**WriteLine2_C** Prepares time/temperature data for LCD display.

- **Parameters:** -
- **Return:** -

## 6.2 ticker.asm

**initLCD** Initializes the ticker (called once).

- **Parameters:** -
- **Return:** -
- **Registers:** Unchanged upon return.

**isrECT4** Interrupt service routine triggered by the timer ticker every 10 ms.

- **Parameters:** -
- **Return:** -
- **Registers:** Unchanged upon return.

## 6.3 decToASCII.asm

**decToASCII** Converts a given decimal value (time) to ASCII (called before displaying time).

- **Parameters:** -
- **Return:** -
- **Registers:** Unchanged upon return.

## 6.4 clock.c

**initClock** Initializes global time variables (called once).

- **Parameters:** -
- **Return:** -

**incrementHours** Increments the hours.

- **Parameters:** -
- **Return:** -

**incrementMinutes** Increments the minutes.

- **Parameters:** -
- **Return:** -

**tickClock** Calls `incrementSeconds` with a clock event.

- **Parameters:** -
- **Return:** -

## 6.5 thermometer.c

**interrupt service** Measures the current temperature (triggered by interrupt vector 22).

- **Parameters:** -
- **Return:** -

**initThermometer** Initializes the analog input (called once).

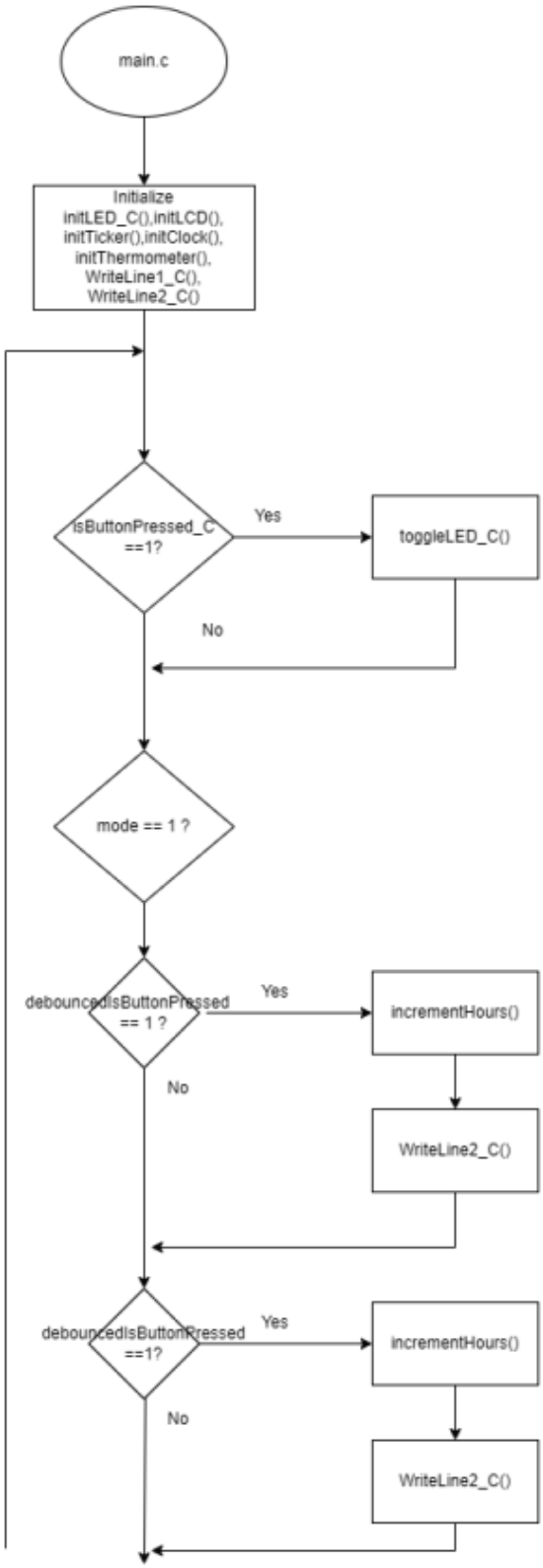
- **Parameters:** -
- **Return:** -

**convertTemperature** Converts the analog value to degrees Celsius.

- **Parameters:** -
- **Return:** -

# 7 Flow charts

## 7.1 Main



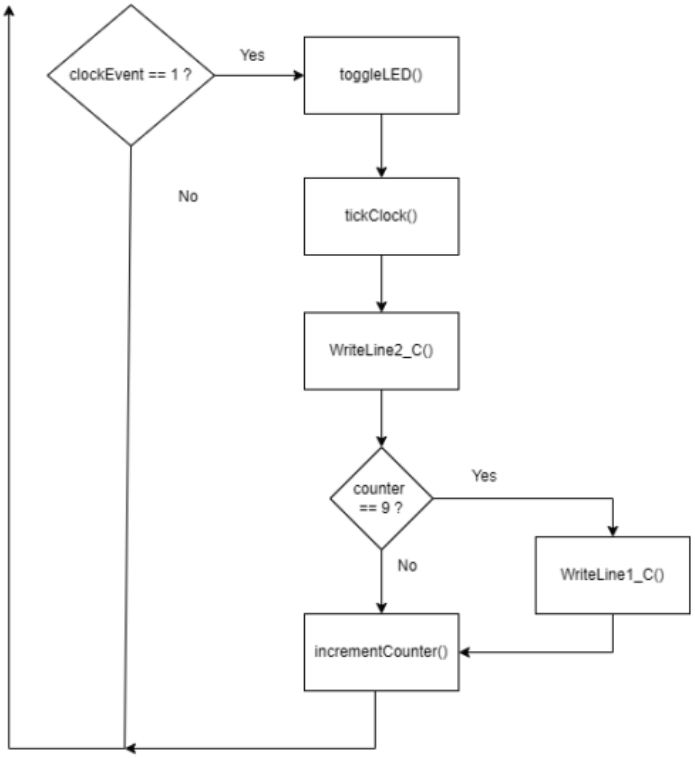



Figure 2: Main

## 7.2 Thermometer


Figure 3: Thermometer

## 7.3 Ticker

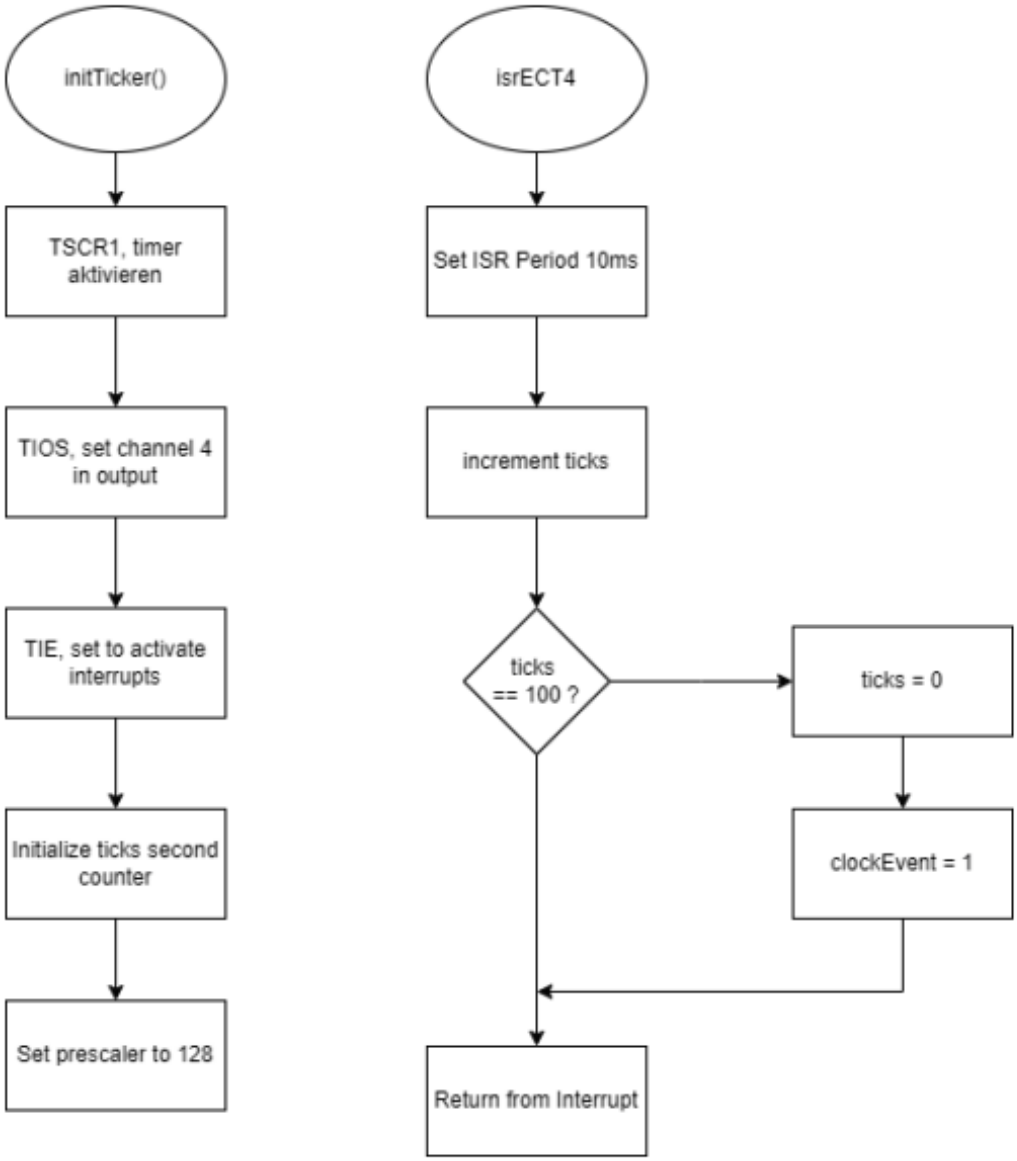



Figure 4: Ticker

## 7.4 Clock

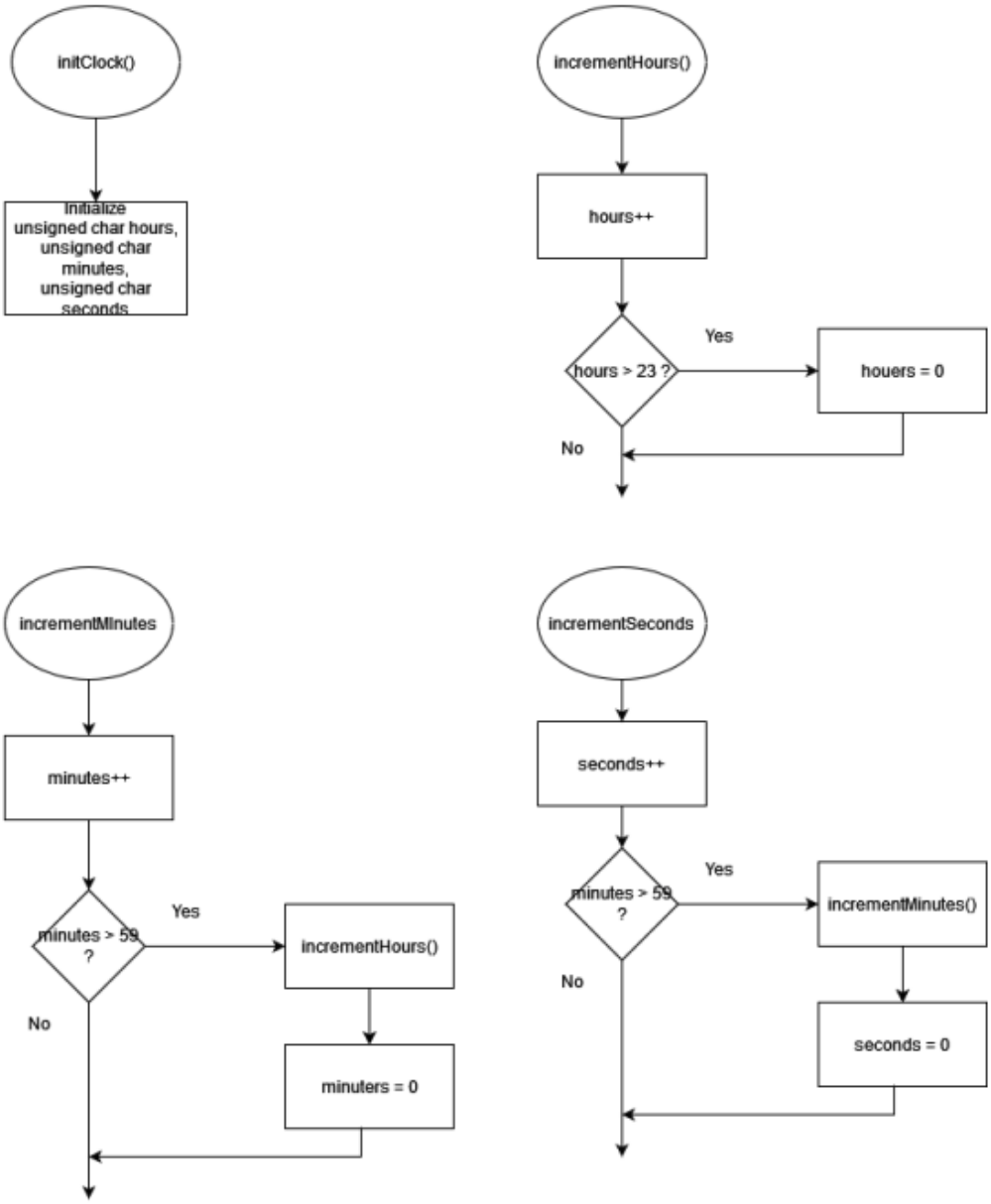



Figure 5: Clock

## 7.5 DecToASCII

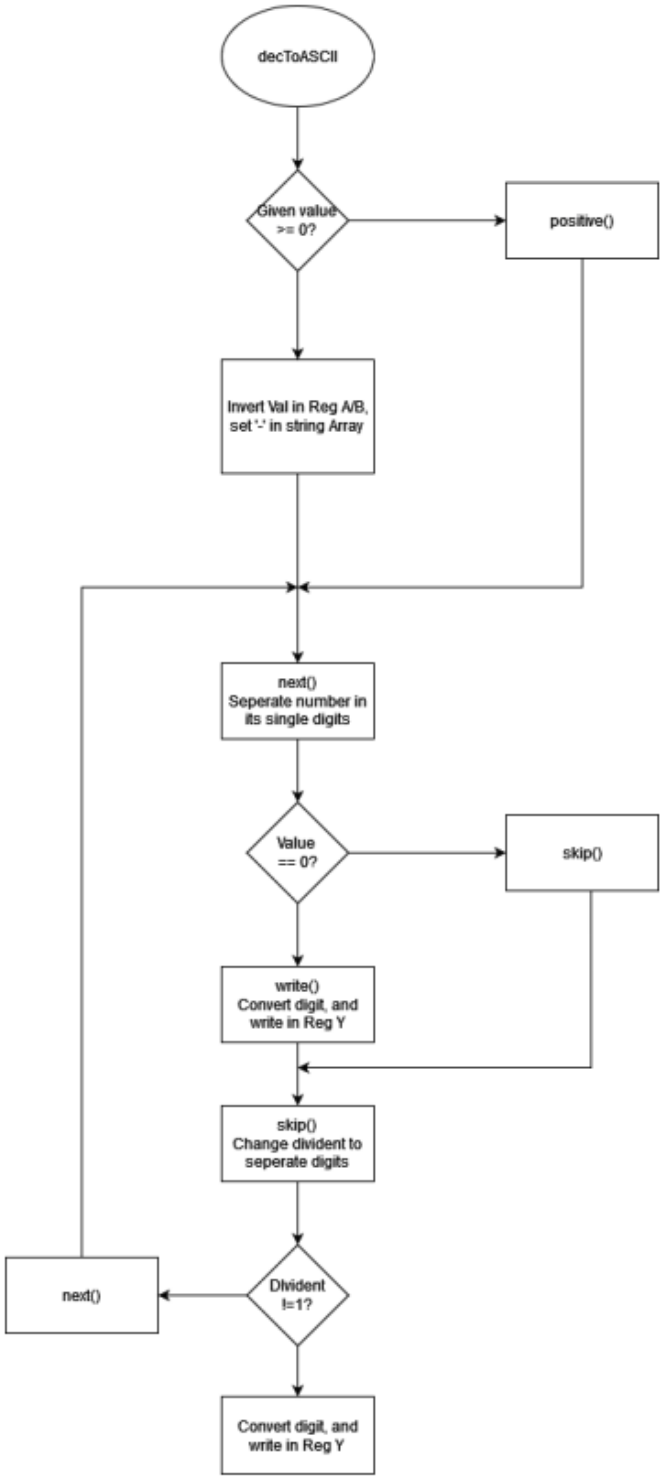



Figure 6: DecToASCII