Hochschule Esslingen
University of Applied Sciences

Computer Architecture Lab 2

Clock–HCS12 Interrupts & I/O

im Studiengang Technische Informatik
der Fakultät Informationstechnik

Wintersemester 2022

Nikita Teztlaff, Fabian Zaiser

December 13, 2022

Contents

1 Preparation

1.1 Task 2.1: timer.asm

```
; Import symbols

; Include derivative specific macros
        INCLUDE 'mc9s12dp256.inc'

; Defines
ONESEC      equ 100
TENMS       equ 1875            ; 10 ms
TIMER_ON    equ $80             ; tscr1 value to turn ECT on was $70(0111 0000)
                                ; changed to $80(1000 0000)
TIMER_CH4   equ $10             ; Bit position for channel 4
TCTL1_CH4   equ $03             ; Mask corresponds to TCTL1 OM4, OL4

; RAM: Variable data section
.data:  SECTION
ticks:    ds.b 1                ; Ticker counter

; ROM: Constant data
.const: SECTION

.intVect: SECTION
        ORG $FFE6
int12:  DC.W isrECT4


; ROM: Code section
.init:  SECTION

;*****************************************************************
; Public interface function: initLCD ... Initialize Ticker (called once)
; Parameter: -
; Return:    -
initTicker:
        ldab #TIMER_ON          ; Timer master ON switch
        stab TSCR1
        bset TIOS,#TIMER_CH4    ; Set channel 4 in "output compare" mode

        bset TIE,#TIMER_CH4     ; Enable channel 4 interrupt; bit 4 corresponds to channel 4

        movb #0, ticks          ; Set tick counter to 0

        ; Set timer prescaler (bus clock : prescale factor)
        ; In our case: divide by 2^7 = 128. This gives a timer
        ; driver frequency of 187500 Hz or 5.3333 us time interval
        ldab TSCR2
        andb #$f8
        orab #7                 ;changed from 8 to 7 or divider is 128 so 2^7 not 2^8
        stab TSCR2
        bclr TCTL1,#TCTL1_CH4   ; Switch timer on
        rts
```

Figure 1.1: fixed code

See in the comments what was changed and why it was changed.

2 Functional requirements

2.1 Clock and temperature measuring requirements

- The current time shall be displayed on the LCD in format HH:MM:SS and updated once per second
- Hours range from 0 to 23, minutes and seconds range from 0 to 59
- LED on Port B.0 shall toggle once per second
- The clock shall be initialized with the time 11:59:30. By presseing SW2 the clock shall switch from normal mode to set mode
- In Set Mode pressing the buttons SW3,SW4 and SW5 shall increase the hours, minutes and seconds. The LED on Port B.7 indicate when the set mode is active
- The LCD panel shall also display the temprature measured by a temperature sensitive resistor (simulated by a potentiometer in the lab) on analog port AD.7. The voltage range is 0 ... 5V representing a temperature of -30 ... +70°C. The temperature shall be updated every second
- The first line of the LCD display shall periodically toggle between the name of all group members and the text "© IT W2021/22". In this case "Nikita Tetzlaff & Fabian Zaiser" and "© IT W2021/22"

2.2 Software requirements

- For the LCD and the LED we use the already existing modules LCD.asm, decToASCII.asm and LED.asm
- 1s tick generated by module ticker.asm from prep task 2.1 shall be used for the clock
- The modules shall be split e.g. for AD converter with driver functions
- Temperature calculations shall be combined in one module and the clock functions in another module
- The main program main() shall be kept as short as possible. The main program shall call initialization functions for all hardware components and then run in a loop, calling the clock and the temperature functions once per second. The interrupt routines shall only deal with the interrupt hardware, but not include more complex operations. Communication between the ISRs and the other parts of the program shall be done via global variables. Especially, interrupt routines must not directly access the LCD display
- The state of the buttons shall rather be tested by polling instead of interrupts

3 User Interface of the program

3.1 LCD Display

- 1st line: Names of all group members or "© IT WS2021/2022" alternating every 10s
- 2nd line:23:55:31 25°C Current time in format HH:MM:SS decimal, left aligned Temperature in degrees Celsius decimal, right aligned.
  Numbers < 10 shall be display without leading zeros, no "+" sign for positive values!

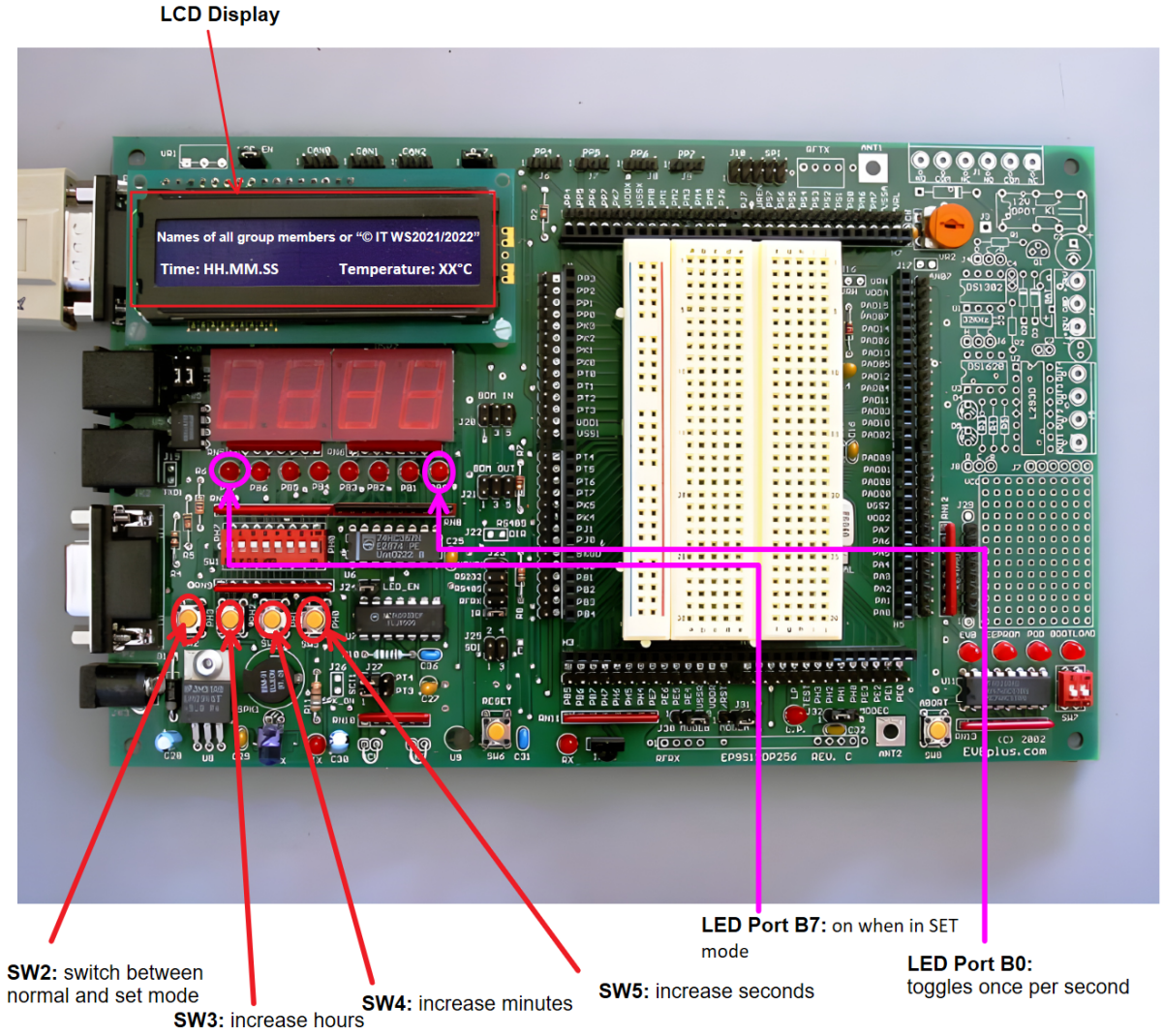


Figure 3.1: DRAGON-12 EVALUATION BOARD

3.2 LED display

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

3.3 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)

4 Module overview

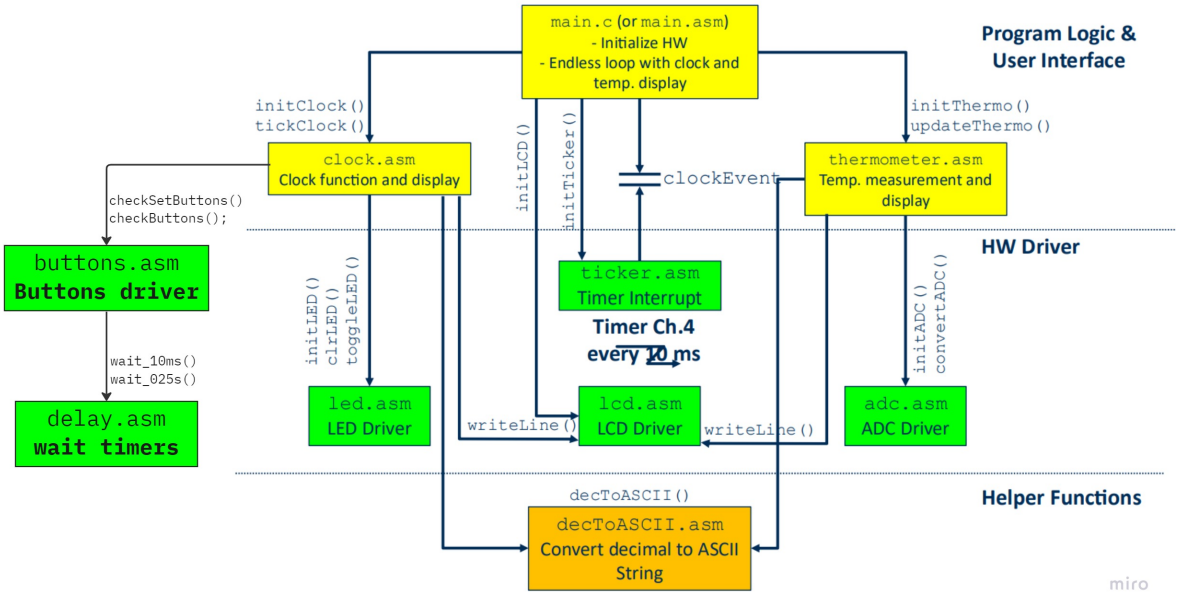


Figure 4.1: overview over all modules

5 Data dictionary

5.1 List of all global variables

Module, where declared (in [] other modules, where used)	**Variable name**	**C like datatype**	**Purpose**
main [ticker]	clockEvent	Boolean (unsigned char)	Periodically indicates one second has elapsed
[clock]	SETMODE	Boolean (unsigned char)	Indicates if SetMode is active
[printCopyright]	secondsCopyrightUnchanged	int (unsigned char)	counts seconds the copyright text wasn't changed (reseted every change
[printCopyright]	firstLineSelection	Boolean (unsigned char)	differentiate between the to different texts for the first line
[clock,thermo]	displayOutput	char [17] (char array)	clock and thermo manipulate the "string" output of this string is in main loop
clock	hours	int	Hours (as numbers)
	minutes	int	Minutes (as numbers)
	seconds	int	Seconds (as numbers)
[decToASCII]	converted	char[6]	decToASCII saves result here
[buttons]	pressedButton	unsigned char	which button was pressed determined by buttons.asm
[buttons]	stayInSetMode	unsigned char	enabled when entering SetMode. Disabled when SW2 was pressed to disable SetMode
	hoursForOutput	int	is used when in 12hour mode so that hours as number is seperated from output (only in 12 hour mode)
thermo	temp	int	raw temperature value from adcIsr
[ad]	tempOutput	int	converted teperature value by ADtoTemp
[decToASCII]	convertedTemp	char[6]	result of tempoutput converted to ASCII by decToASCII
[ad]	maxTemp	int	Set max Temperature for thermometer
[ad]	minTemp	int	Set min Temperature for thermometer
[ad]	adcResolution	int	set resolution of the sensor /adcIsr

Figure 5.1: list of global variables

5.2 Hardware resources

Module	HCS12 or Dragon12 HW resource	Purpose
Clock Thermometer	CPU Port K LCD Display	Display: Line 1; Text Line 2: Time and
LED [clock]	Port B, PortJ.1 LEDs	Various status signals
AD [Thermo]	ATD0 Channel 7	Temperature Sensor
Clock [buttons]	Port H.3...0 Buttons SW2...SW5	Set time / mode
Ticker [Main]	Enhanced Capture Timer Channel 4	10 ms ticker

Figure 5.2: list of hardware resources

6 Flowcharts of all modules

6.1 main.c

6.1.1 main

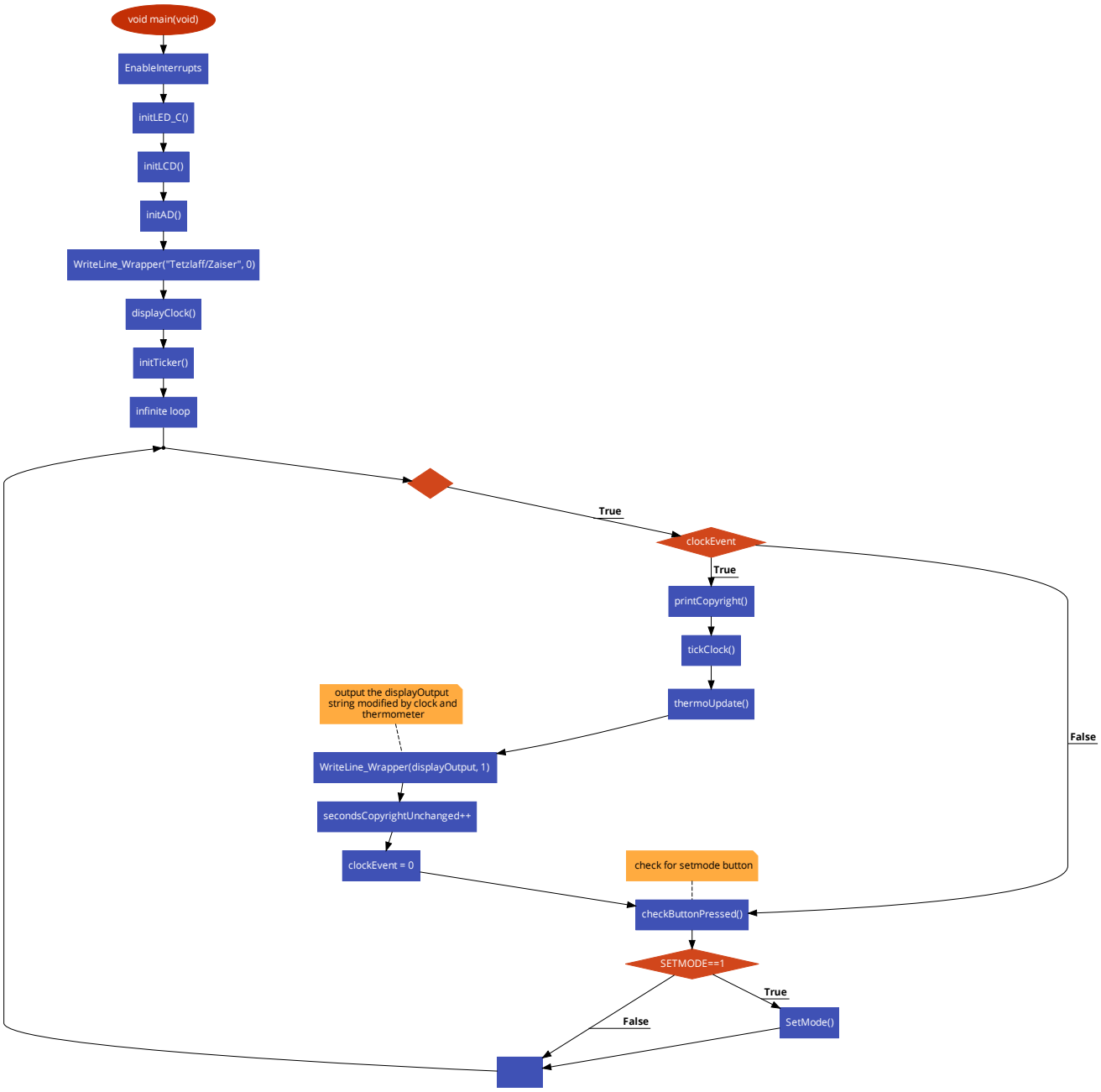


Figure 6.1: main() flowchart

6.1.2 printCopyright

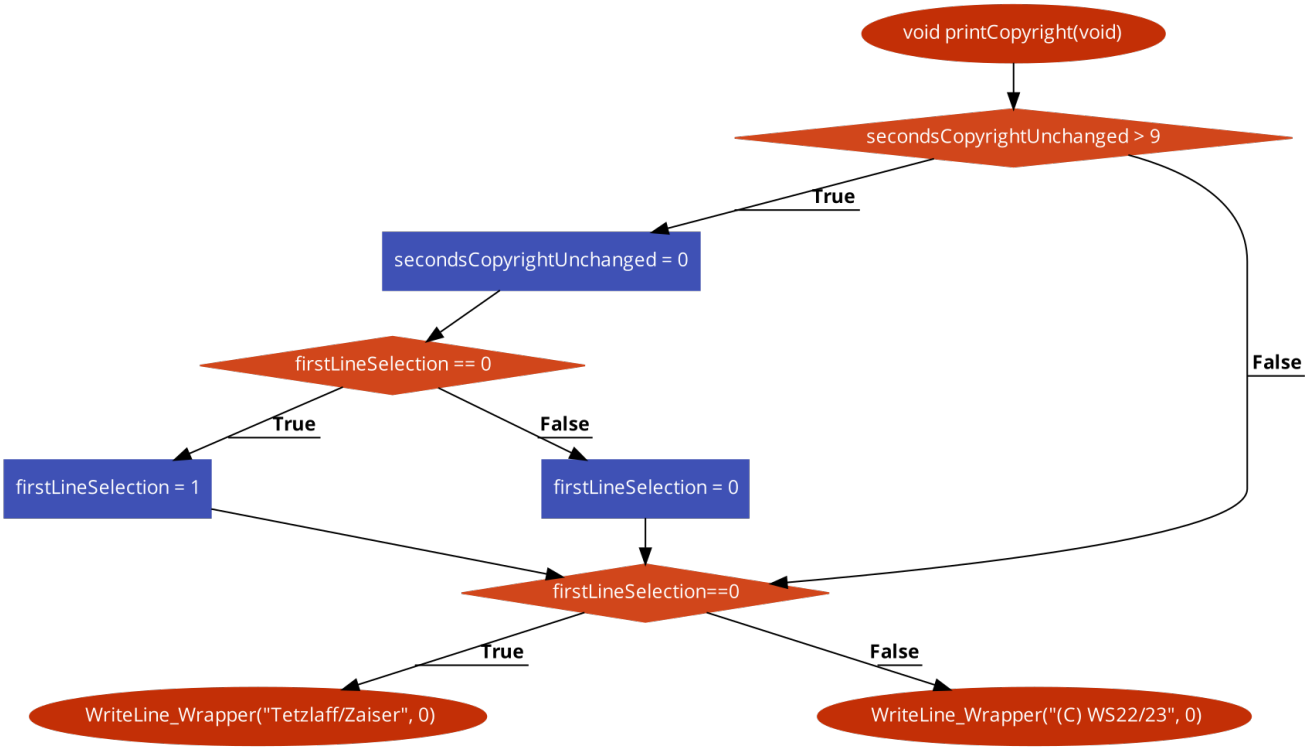


Figure 6.2: printCopyright() flowchart

6.2 thermo.c

6.2.1 thermoUpdate

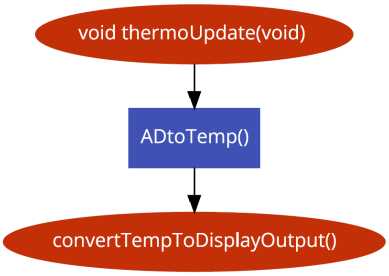


Figure 6.3: thermoUpdate() flowchart

6.2.2 convertTempToDisplayOutput

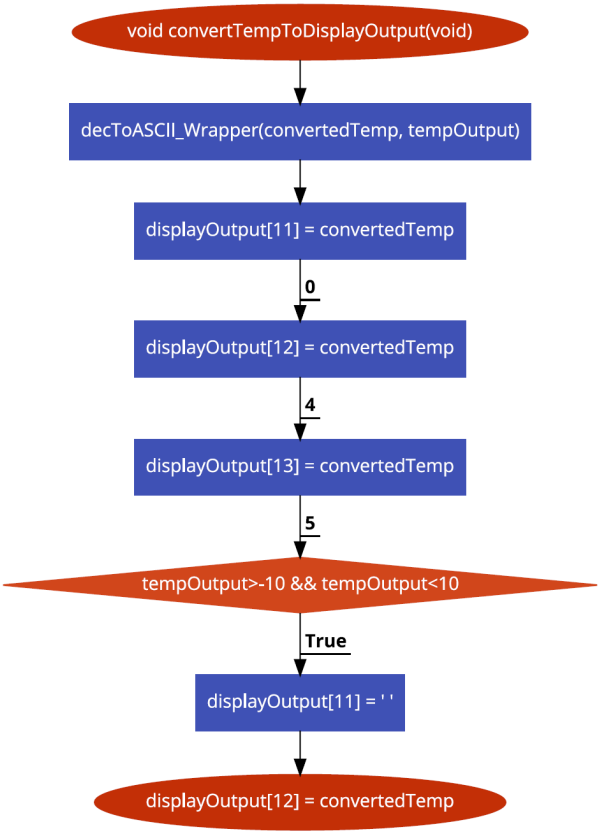
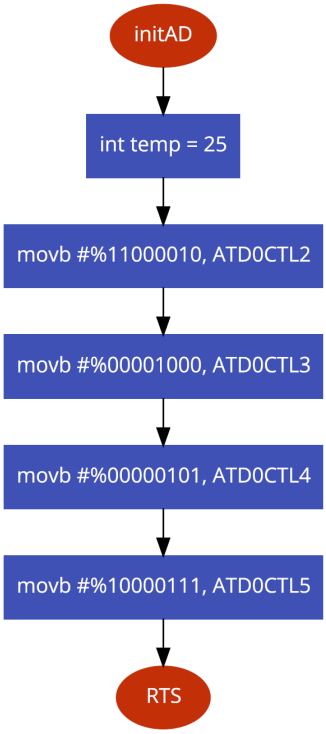


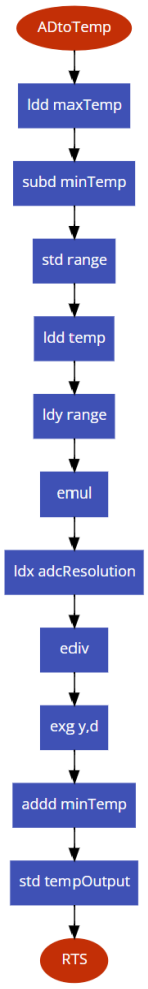
Figure 6.4: convertTempToDisplayOutput() flowchart

6.3 ad.asm

6.3.1 initAD and ADtoTemp



(a) initAD flowchart



(b) ADtoTemp flowchart

6.3.2 adcIsr

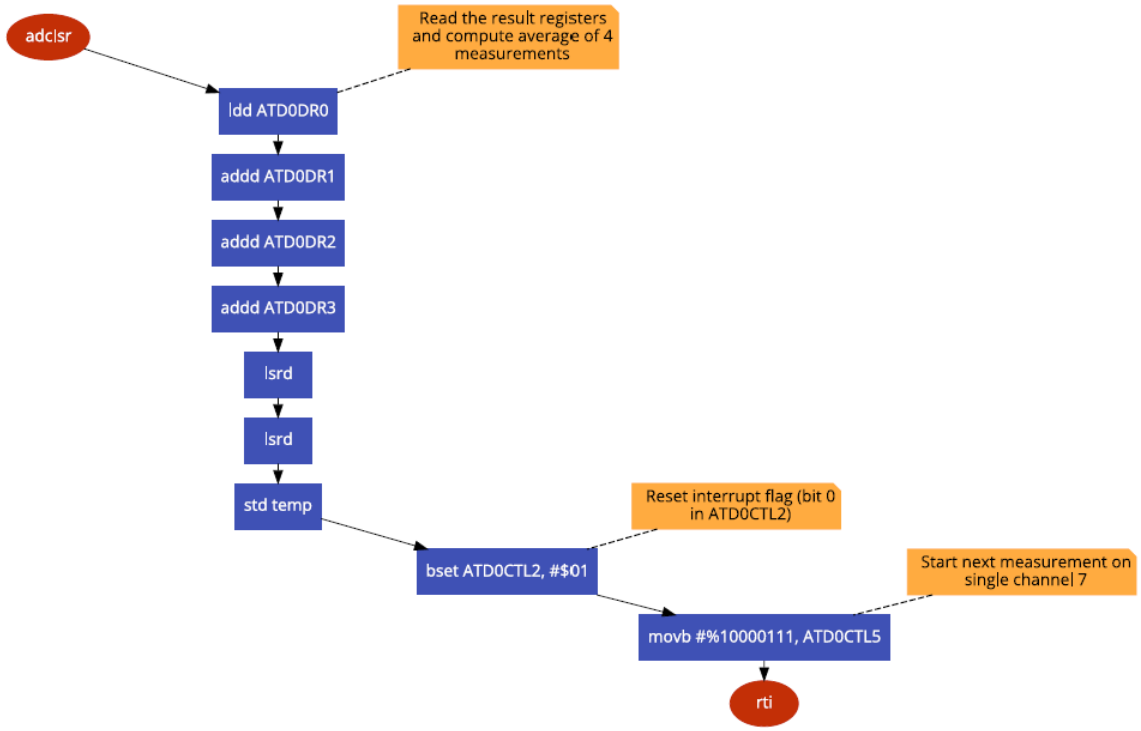
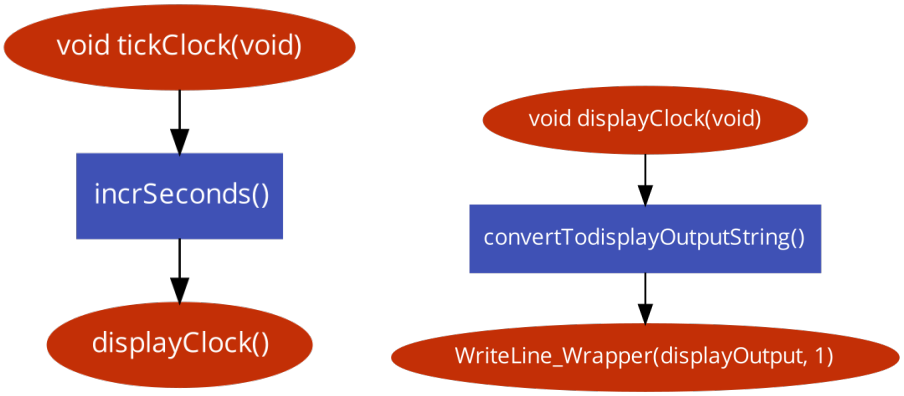


Figure 6.5: adcIsr flowchart

6.4 clock.c

6.4.1 tickClock and displayClock



(a) tickClock flowchart (b) displayClock flowchart

6.4.2 inrement of hours/minutes/seconds

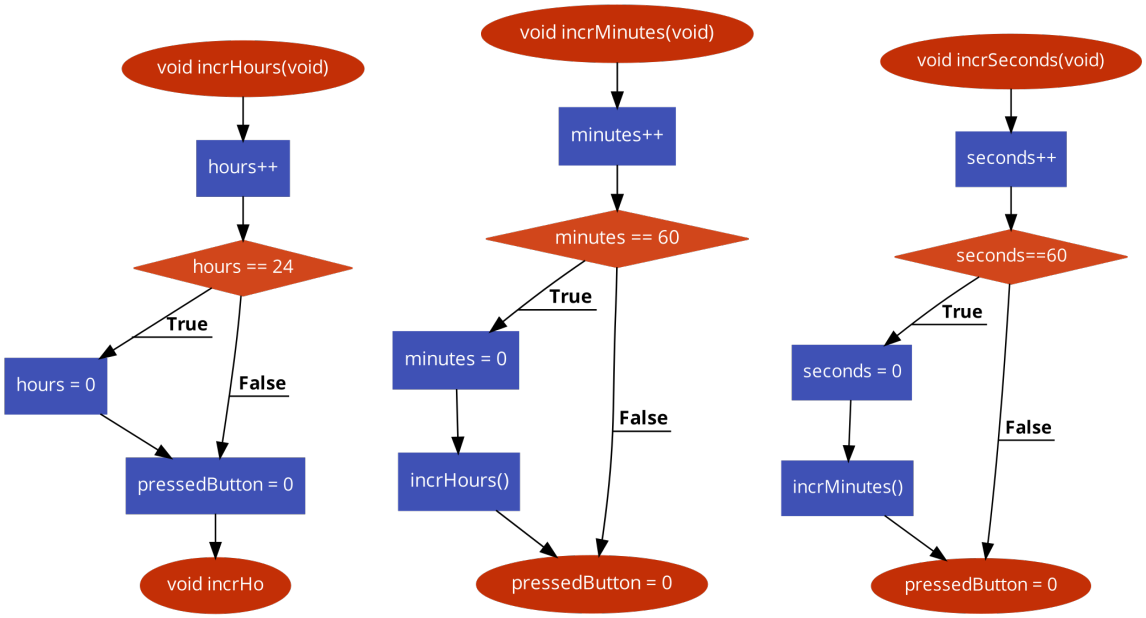


Figure 6.6: increment time values flowcharts

6.4.3 convertToDisplayString

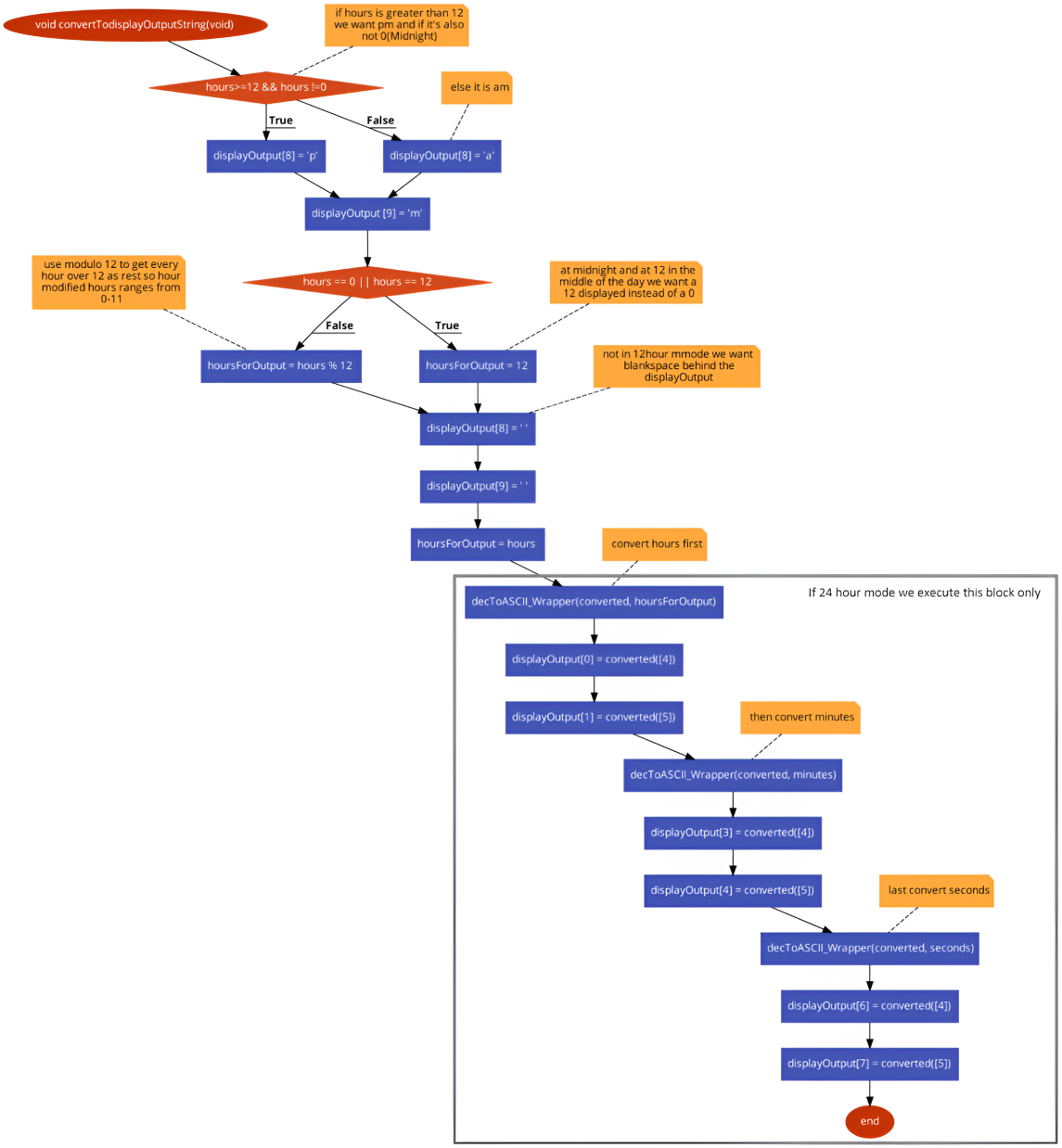


Figure 6.7: convertToDisplayString flowchart

6.4.4 SetMode

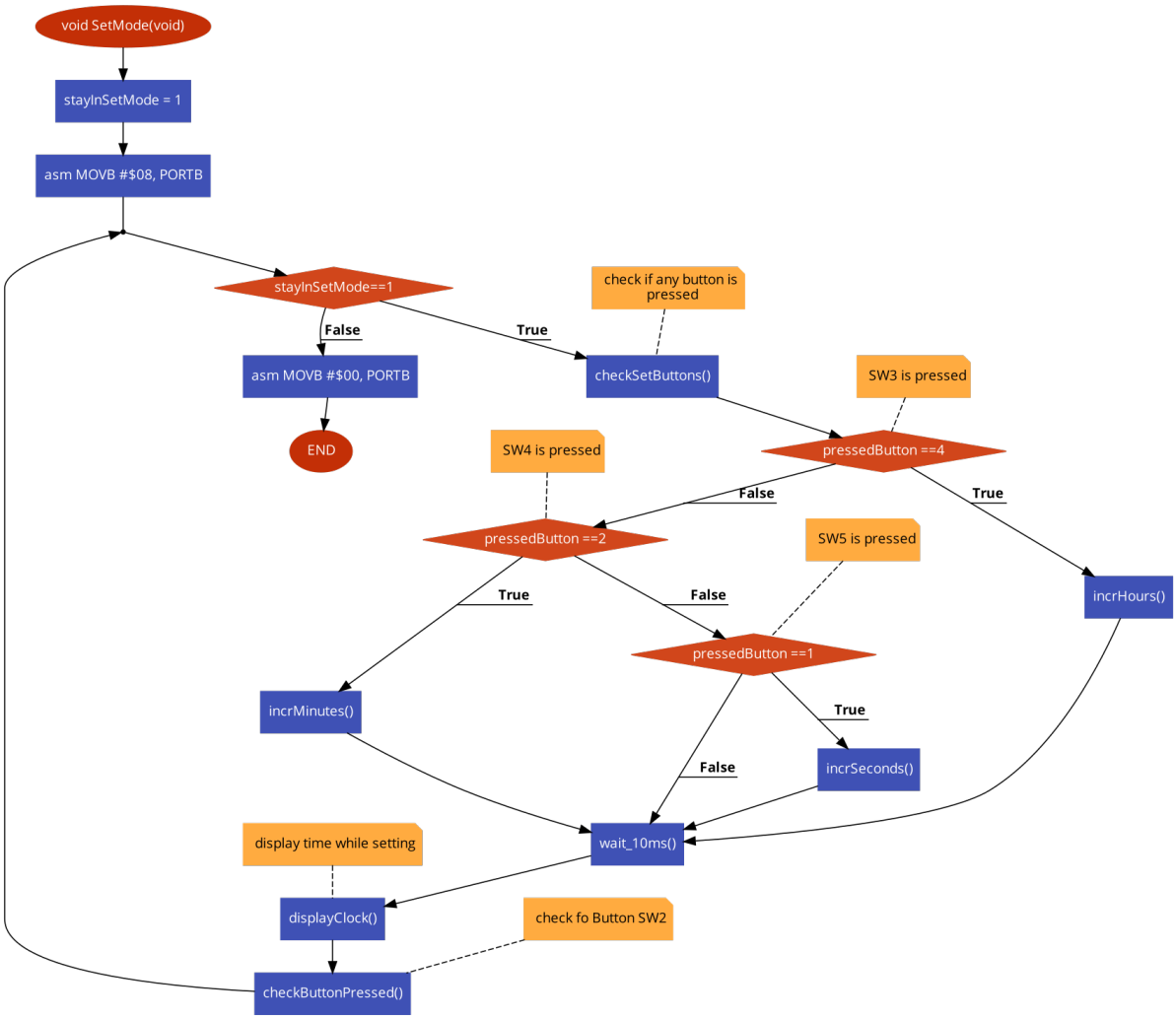


Figure 6.8: SetMode flowchart

6.5 Ticker.asm

6.5.1 initTicker

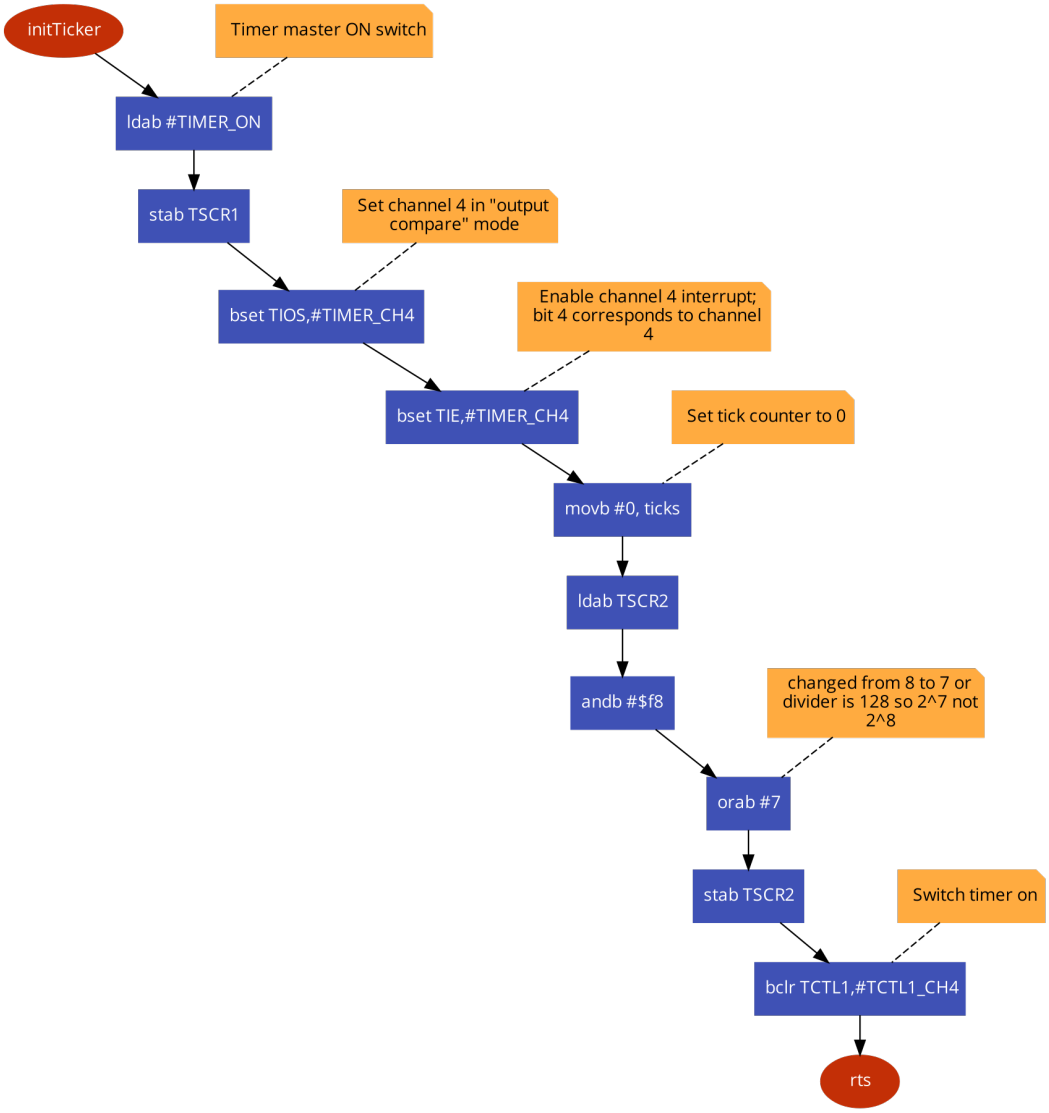


Figure 6.9: initTicker flowchart

6.5.2 isrECT4

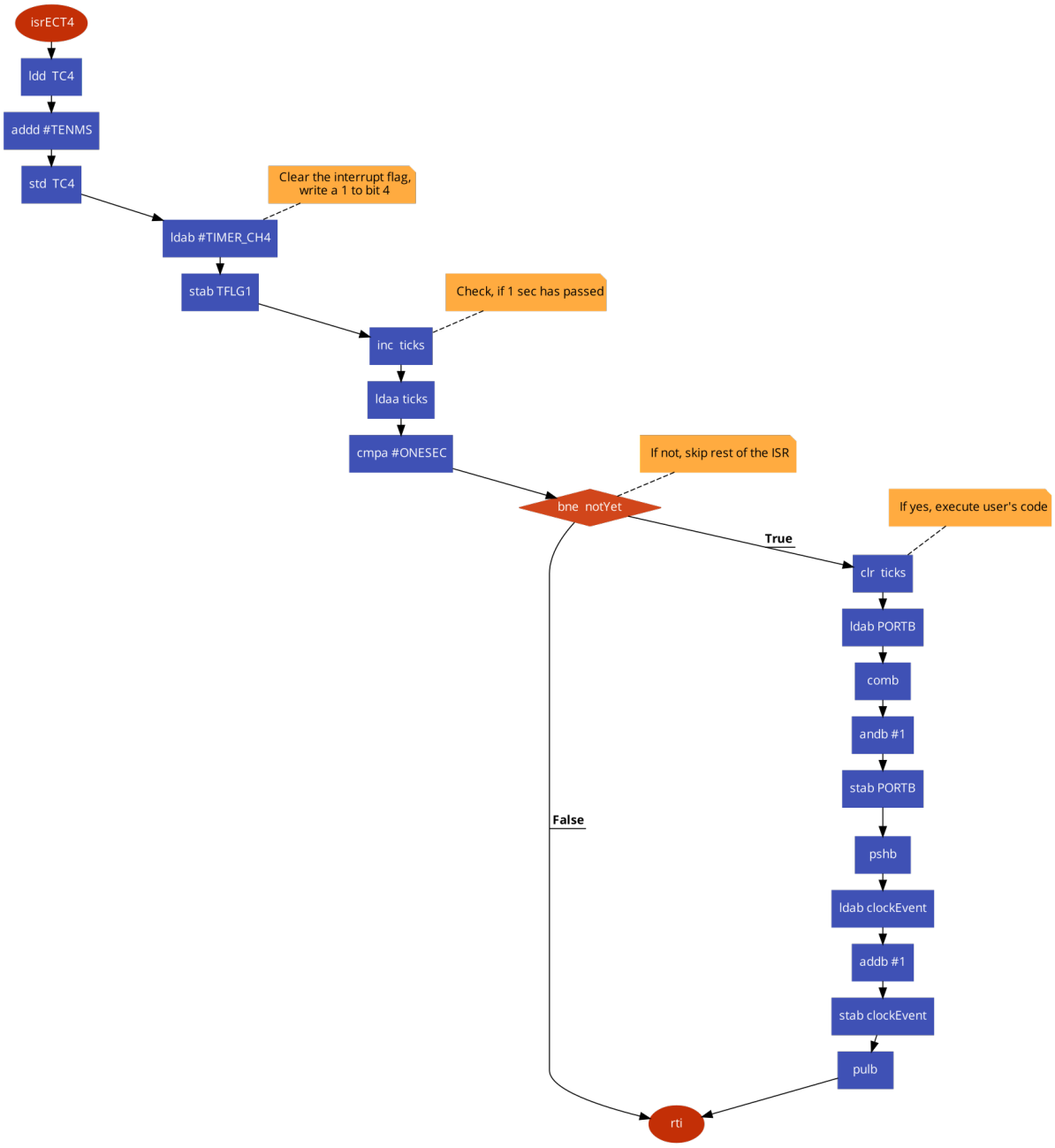


Figure 6.10: isrECT4 flowchart

6.6 buttons.asm

6.6.1 checkButton

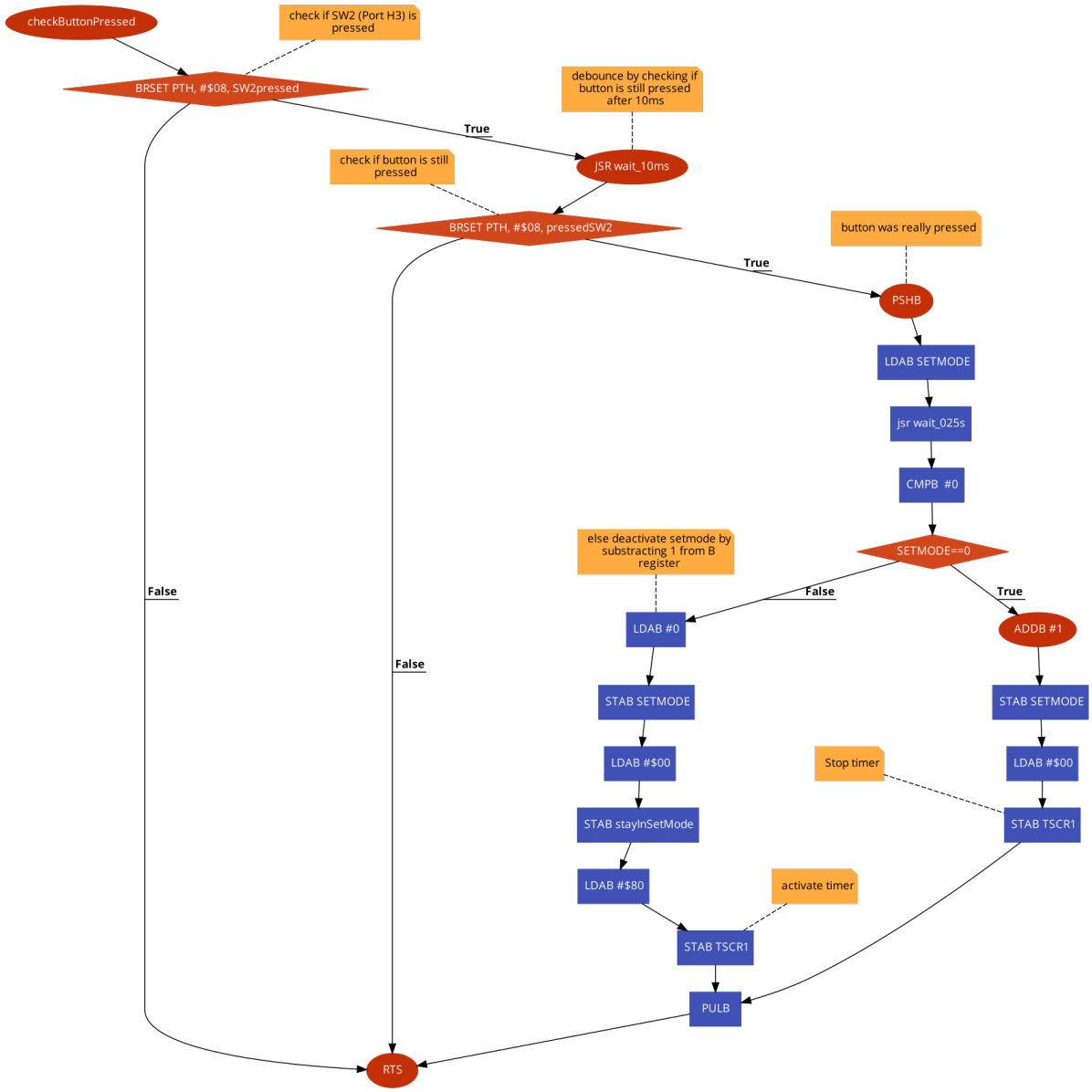


Figure 6.11: checkButton flowchart

6.6.2 checkSetButtons

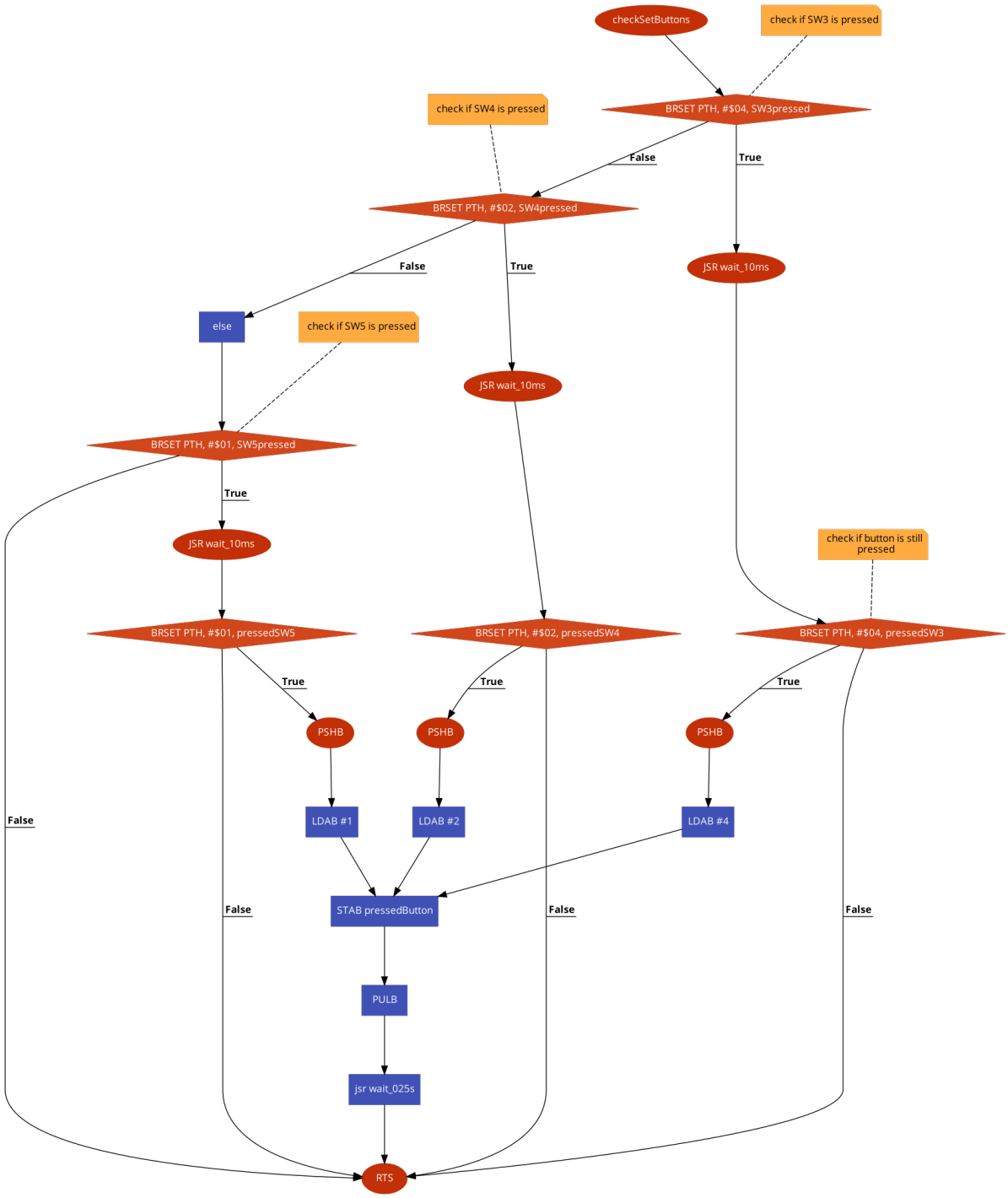


Figure 6.12: checkSetButtons flowchart

6.7 decToASCII.asm

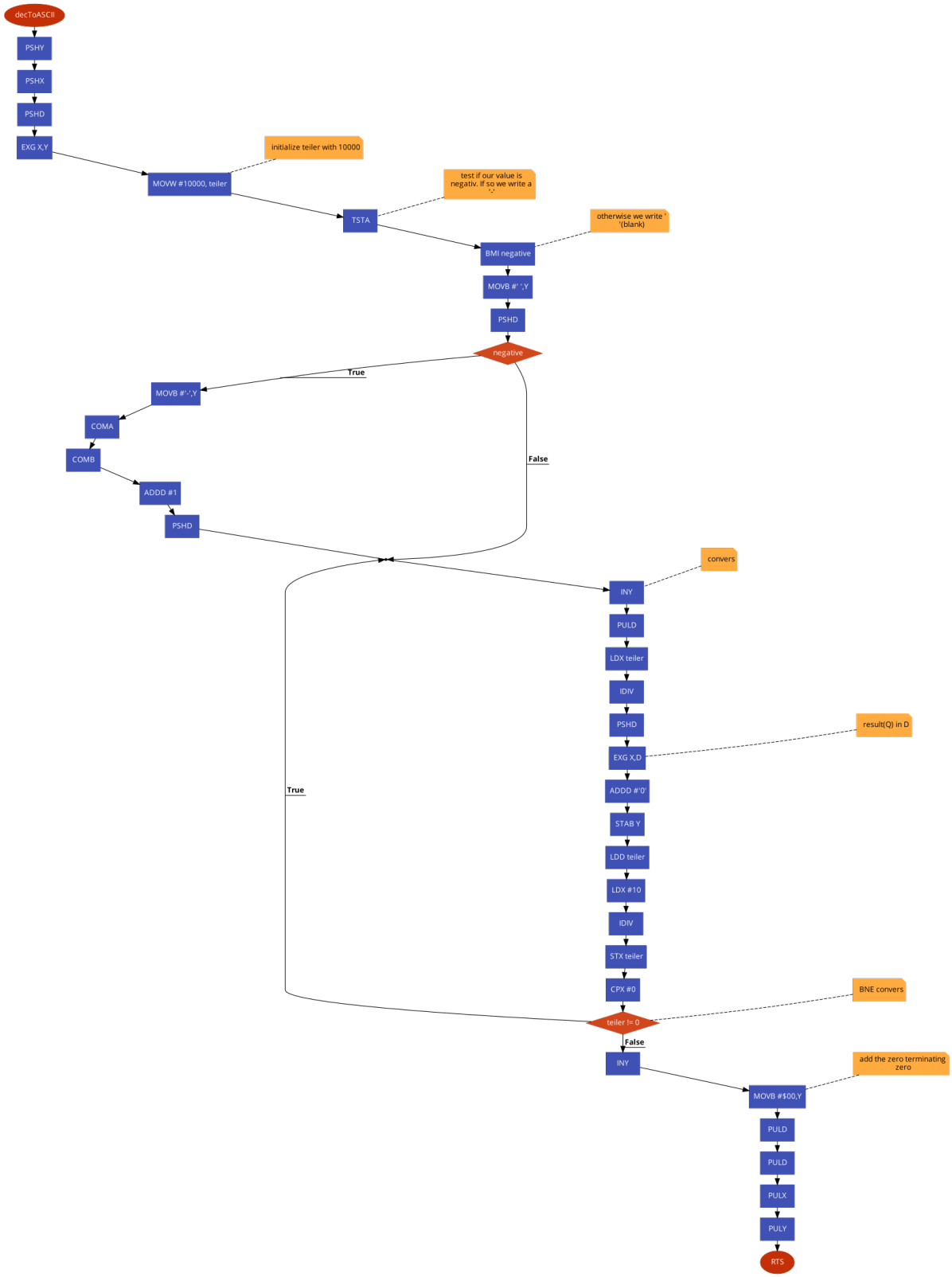


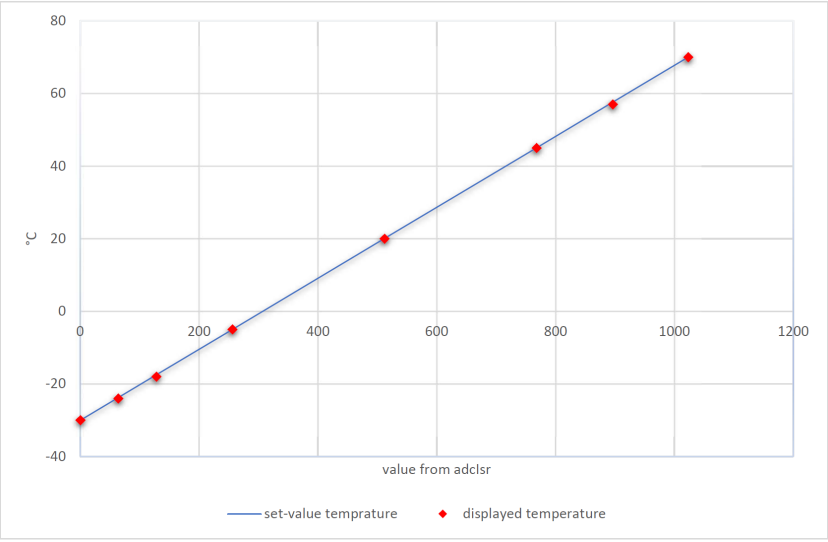
Figure 6.13: decToASCII flowchart

7 Testing

In this chapter we want to test ADtoTemp. The ADtoTemp function gets it's values from the adclsr which gets values from a potentiometer. Since we simulate the board we don't have a potentiometer. While simulating the adclsr won't be initialized and inactive. The adclsr stores the raw values for the temperature in the variable temp in thermo.c. The ADtoTemp uses the temp variable to convert it to a temperature. To convert a raw temperature value to a temperature to display we use a simple linear formula to map the values equally distributed in the range between -30 and 70. Since that is the range of values the potentiometer outputs. The formula we use is tempOutput $= (temp \cdot 100)/1023 - 30$. To thest if the ADtoTemp funtion works correctly we manipulate the temp variable manually. Since we have a linear mapping of the raw values to tempertaures we know exactly which temperature we expect for each raw value.

value from adc	°C expected	°C displayed
0	-30	-30
64	-24	-24
128	-17	-18
256	-5	-5
512	20	20
768	45	45
896	58	57
1023	70	70

(a) values tested



(b) test results as chart

In the list and the chart we see that some displayed temperatures deviate from our expected temperatures. This is because of rounding. For example: for the value 128 a more exact value for the temperature expected would be 17,49°C and our ADtoTemp typecasts the result to an int. It rounds this value up to 18 in this case. In exception of a few values the output of our ADtoTemp funtion is pretty spot on and for our use of a clock with a temperature output this is precise enough.