

INHA University in Tashkent



School of Computers and Information Engineering

Spring semester 2020

Embedded Software & Design

SOC3050

Report

Digital Alarm Clock using Atmega128 Microcontroller

Team: Project 42

U1610016 Akmal Karimov

U1610041 Azizbek Kobilov

U1610142 Mirkamol Khamidov

Contents

1. Algorithm explanation	Ошибка! Закладка не определена.
2. User manual	6
3. Working video URL	7
4. Team member contribution	7

1. Algorithms explanation

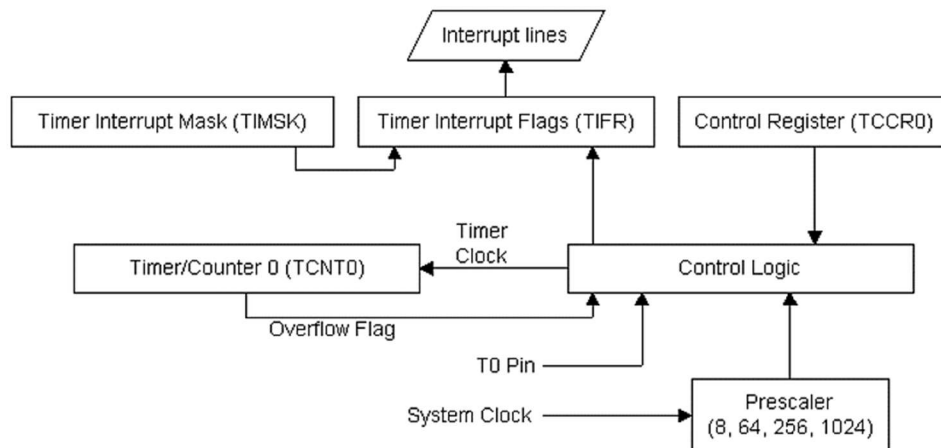
Let us look through the algorithms used in our code in their order of used functions:

1. `Init_Timer0()` function

TCCR0 (Timer Counter control register) is set to CTC (Clear Timer on Compare) mode with 1024 prescale and Output Compare Interrupt and Global Interrupts are enabled

We set OCR0 register to 1 so that the Timer/Counter register counts until the compare match occurs giving up to 2 machine cycles. *Why 2? Why Compare register is not set 99 to give 100 cycles?*

(further in the Interrupt Vector part it is explained why)



2. `Init_Ports()`

SWITCH inputs - all turned on, LED outputs – all turned off, LCD outputs

3. `switchController()`

This function is used twice in the code in Clock and Alarm parts for Setting the Date and Time values. 5 switches are used: 1 2 3 4 and 8 in respective order, 1 and 2 switches are used to update the values, more specifically to decrease and increase, whereas 3 and 4 are used to move to previous or next target, last switch is used to go exit to a Main menu. Switching is made using *Polling* approach, since we allow our presses to be continued in every 200ms (while just holding the switch)

4. `changeValue(int* target, int mod)`

Used several times while setting the Clock values for date and time, and includes 3 different approaches to retrieve the updated value:

- 1) if our targets are minutes and seconds it will have a range of [0 ~ 59] values
- 2) if our target is 12-hour periods it will have 0 or 1 values (to update the Flag)
- 3) all other targets will have range of [1~N] where N is their max value

5. `fullHourChange(int n)`

The same concept of retrieving value as in previous function but only for 24-hour with the update of the period (AM/PM) when is called

6. `getWeekday(int y, int m, int d)`

If we know the year, month and day, we can mathematically get the Weekday, and the algorithm is known as “Doomsday Algorithm” hackerearth.com/blog/developers/how-to-find-the-day-of-a-week/

7. `getMaxDayValue()`

Getting the maximum day of a given month using the array of predefined days and the leap year calculation to find it specifically for February

8. `getDate()`

Used to find whether the year is leap or not and to get the Weekday with max day
Also, print the Date if user in the Clock screen

9. `printTime()` | `printDate()` | `printAlarm()`

Used for printing the time and date values if specific format and update the Cursor’s position

10. `goClock()` | `goStopwatch()` | `goAlarm()`

Used to move to the specific mode screen with updating the flags: 1 for entered mode, others – 0

11. `setClock()`

Used to set the clock in start. Also, allows to reset the clock values only if we are in the Clock mode by pressing the 7th switch. To update the values and move in either directions we use the `switchController()` and `changeValue(int*, int)` functions mentioned before

12. `printStopwatch()`

While printing the stopwatch mode screen in specific format it allows us to complete 4 actions of Starting, Pausing, Continuing and Stopping the stopwatch with just 2 switches, where the first 3 actions are made on 1 switch

13. `setAlarm()`

Same principal as in Setting the Clock as well as retrieving the time from Clock with the interactive UX/UI design to facilitate and assist the user to set up the Alarm

14. `Led_Alarm()`

When the time on the Clock will match with the Alarm time the LED will blink for 5 times in every 2 seconds

15. `mainMenu()`

Main menu with an intuitive design is called when the Exit switch is pressed. Mode Flags are reset

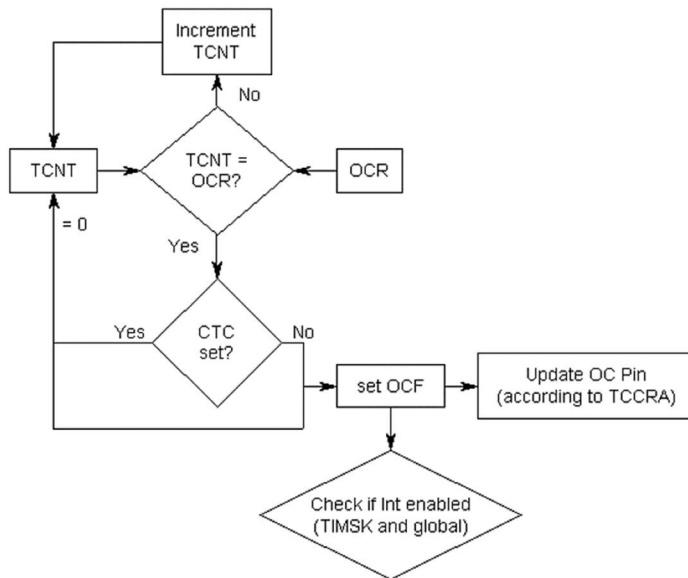
16. `Welcome()`

Is called only once in the beginning of the program to guide the user.
Initially any switch press is allowed to move forward.

17. `main()`

First, we welcome and guide the user, then ask him to set the clock and finally, let him to enjoy it. 😊

18. ISR(TIMER0_COMP_vect) - Timer/Counter0 Compare Match Interrupt Vector



First, let's discuss how to get 1 sec delay with a CTC mode with `OCR0 = 99` (100 counts):

- 1) The external clock is 14.7456Mhz (147456 machine cycles):
Period: $T = 1/14.7456\text{MHz} = 0.067817 \text{ uS}$
- 2) After 1024 prescale: $0.067817 * 1024 = 69.4444 \text{ uS}$
- 3) Set `OCR0 = 99` then `OCF0` interval is 6.94444 ms
- 4) 1s Delay in Timer0 CTC mode = $6.94444 \text{ ms} * X = 1 \text{ Second}$
- 5) $X = 144$ (loop count)

Why then Compare register is set to 1 (2 cycles)?

In order to get the Millisecond values with the Resolution of 1/100 sec used in the Stopwatch mode. And besides that, we have polling methods and dozens of LCD operations which consume some Time Delay in each of their function calls.

Our solution:


- 1) Set `OCR0 = 1` (2counts) then `OCF0` interval is 0.13889 ms
- 2) 1s Delay in Timer0 CTC mode = $0.13889 \text{ ms} * X = 1 \text{ Second}$
- 3) $X = 100$ (stopwatch ms) * 60 (loop count) $\approx 8.5 \text{ ms}$

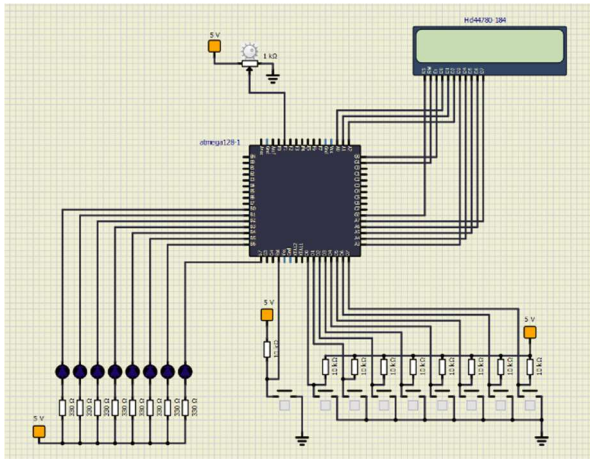
(!) Although it is not exactly 1 second we have checked and tested for several times and it works the same as the delay for 1 second in the Computer's time.


So, every 2 cycles the Timer's Interrupt Vector is called and when counter hits the loop count, we reset it and increment our milliseconds. The same concept for Seconds, Minutes, Hours and other Date Values:

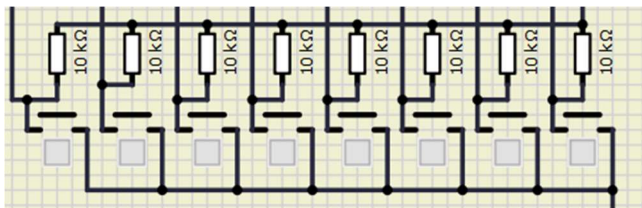
"If hit the limit, reset urself and let the parent increment"

1. User Manual

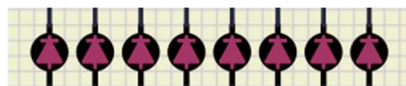
1. Download Desktop App – “SimulIDE – 0.3.12.18 - SR8 Win32” 
2. Get a Circuit - “ATMEGA128_20200426_1330” and open that in the App



3. Right click on Black Square to select “Load firmware” and choose our .hex file
4. Press on Start (Red button on the top) 
5. Manipulate with Switch buttons in respective order (1 to 8)



6. Although what switches will be used are shown in our app itself, there are 1 2 3 4 and 7 8 switches are in use (5 and 6 are for future implementations)
7. To go back (exit) use the last switch
8. To set Date and Time use 1 and 2 switches to change the value (1 – downward, 2 – upward) and to move around use 3 and 4 switches (3 – go to Previous target, 4 – go to Next target)
9. To reset the Clock, use the 7 switch and repeat the process
10. The same switches are used for Alarm and the same process to Set up
11. When it will Alarm the LEDs will blink every 2 seconds for 5 times



2020 30/11 Mon
PM 12:00:02

Alarm isn't Set:
AM 00:00

1-Clock 2-Alarm
3-Stopwatch

1-Pause 2-Stop
00:00:02:93

12. In the Stopwatches you will interact with 2 switches to Start, Pause, Restart and Stop processes and even when you start a stopwatch and go to other screens as Main Menu or Clock itself, the stopwatch will continue working, meaning no any other screen will interrupt each other from their process (like threading)

13. Enjoy Intuitive and User-friendly UI/UX from our team Project 42

WELCOME
Press for Next

2. Working video URL

The scenario of running our app is provided on youtube. You can go to the link and watch the video where we test a scenario and explain each step.

Link: <https://www.youtube.com/watch?v=tKRRiR4Tb5o&feature=youtu.be>

3. Team member contribution

Akmal Karimov – u1610016:

- Brainstorming
- Making algorithm for clock and date
- Writing code (Timing part, clock part and date part)
- Code Optimization
- Documentation of Software Specification

Aziz Kobilov – u1610041:

- Brainstorming
- Making algorithm for alarm and stopwatch
- Writing code (Alarm and Stopwatch)
- Code Optimization
- Documentation of Requirement Analysis

Mirkamol Khamidov – u1610142:

- Brainstorming
- Making algorithm for timing and precisions for time
- Writing code (Timing part, date part)
- Code Optimization
- Documentation of Report