TOSHKENT SHAHRIDAGI INHA UNIVERSITETI
INHA UNIVERSITY IN TASHKENT

# Embedded System

## Design Specification Document

**Team name:**

CD Projekt Green


**Team Members:**

| | |
|---|---|
| Oybek Khakimjanov | U1610179 |
| Dilmurod Nasibullaev | U1610058 |
| Anvar Abdulsatarov | U1610025 |

## Description

This document is designed to be a reference for any person wishing to find out about project or any person interested in the architecture and functionalities of CD Projekt Green project made on AVR microcontroller.

This project is AtMega128 microcontroller based digital alarm clock.

This document describes all the design specification and criteria that should be accomplished in our final assignment. A brief overview of functionalities and how we realized it, will be discussed in detail.

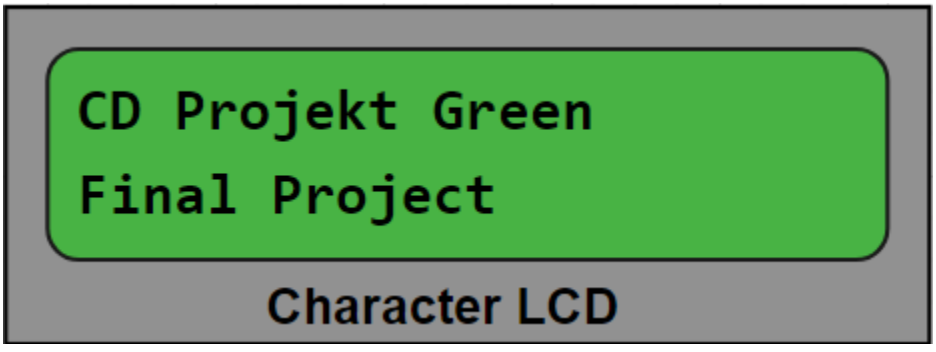
**Components Required:**

- Atmel Studio
- SimulIDE
- Circuit scheme
- Virtual AVR Microcontroller

The digital clock should be able to operate in the following three modes:

1. Normal time display
2. Alarm
3. Stopwatch

Our main priority is to achieve an independent cooperation of these three modes. For example, when a user decides to set the alarm or stopwatch, the time should work in a background mode without being delayed or reset.




A detailed overview and explanation of these modes is provided in Requirements Specification and Analysis document.

# Basic Concepts

Before proceeding further, let's understand some basic concepts and fundamental tradeoffs of modern microcontrollers.

## Interrupts

**Interrupts** can stop the main program from executing to perform a separate interrupt service routine (ISR). When the ISR is completed, program control is returned to the main program at the instruction that was interrupted.

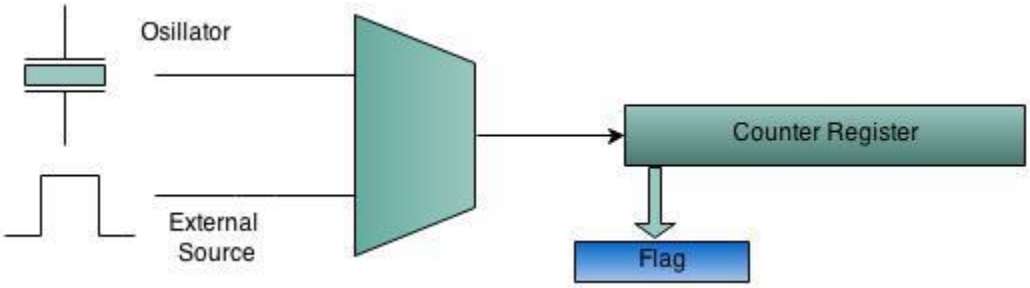
These interrupts each have a separate program vector in the program memory space. All interrupts are assigned individual enable bits which must be written in one logic together with the Global Interrupt Enable bit in the Status Register in order to enable the interrupt.

The lowest addresses in the program memory space are by default defined as the Reset and Interrupt Vectors. They have determined priority levels; the lower the address the higher the priority level. RESET has the highest priority, and next is the External Interrupt Request 0 (INT0).

The table below shows interrupt vector table for Atmega128:

| Vector No. | Program Address(2) | Source | Interrupt Definition |
|---|---|---|---|
| 1 | $0000(1) | RESET | External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset |
| 2 | $0002 | INT0 | External Interrupt Request 0 |
| 3 | $0004 | INT1 | External Interrupt Request 1 |
| 4 | $0006 | INT2 | External Interrupt Request 2 |
| 5 | $0008 | INT3 | External Interrupt Request 3 |
| 6 | $000A | INT4 | External Interrupt Request 4 |
| 7 | $000C | INT5 | External Interrupt Request 5 |
| 8 | $000E | INT6 | External Interrupt Request 6 |
| 9 | $0010 | INT7 | External Interrupt Request 7 |
| 10 | $0012 | TIMER2 COMP | Timer/Counter2 Compare Match |
| 11 | $0014 | TIMER2 OVF | Timer/Counter2 Overflow |
| 12 | $0016 | TIMER1 CAPT | Timer/Counter1 Capture Event |
| 13 | $0018 | TIMER1 COMPA | Timer/Counter1 Compare Match A |
| 14 | $001A | TIMER1 COMPB | Timer/Counter1 Compare Match B |
| 15 | $001C | TIMER1 OVF | Timer/Counter1 Overflow |
| 16 | $001E | TIMER0 COMP | Timer/Counter0 Compare Match |
| 17 | $0020 | TIMER0 OVF | Timer/Counter0 Overflow |

## Timers




**Timers/Counters** are essential part of any modern MCU. Remember it is the same hardware unit inside the MCU that is used either as Timers or Counter. Timers/counters are an independent unit inside a micro-controller. They basically run independently of what task CPU is performing. Hence, they come in very handy, and are primarily used for the following:

- **Internal Timer**: As an internal timer the unit, ticks on the oscillator frequency. The oscillator frequency can be directly feed to the timer or it can be pre-scaled. In this mode, it used to generate precise delays. Or as precise time counting machine.
- **External Counter**: In this mode, the unit is used to count events on a specific external pin on a MCU.
- **Pulse width Modulation(PWM) Generator:** PWM is used in speed control of motors and various other applications.

# System Overview

To begin with, to simplify our job and to avoid code repetition, we included several headers to create an overall concise format.

An included **`lcd_n.h`** header file contains methods and macros for performing printing operations. This file will be provided inside project's archive.

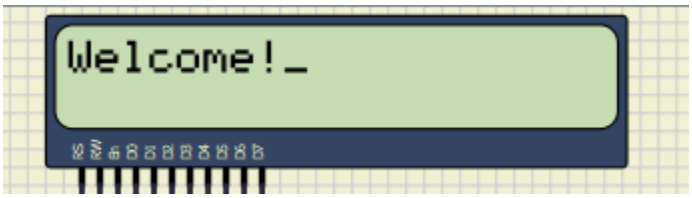
The function `initDevices()` performs the following initialization operations:

- Resets all interrupts and enables devices
- Sets data direction of **DDRA** and **DDRG** ports to output by setting '0xFF'
- Initializes Character LCD and clear all values from the screen
- Initializes interrupts as 'EICRA = 0xAA', 'EIMSK = 0x0F' and PORTD as input.

# Implementation Overview

## Welcome Screen

At the beginning of the execution, port, interrupt are initialized and calculation of leap year happens. Then, `welcomeScreen()` function is called to display "Welcome!" message on the screen. The content of the screen is shown with the help of the function `showScreen()`, which is called inside main function's *infinite loop* and it decides what content to show based on the value of the global variable called "mode".




## Clock

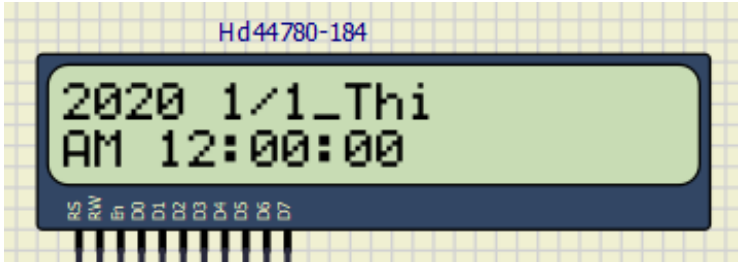
Initial mode is the clock, which corresponds to flag 0 (mode=0).

The "Welcome!" message is printed for two seconds and after that the time is being shown on the screen. You can set the necessary time at any stage of the execution.

In a clock mode (mode = 0), there are 8 variables displayed on the LCD:

- Year, Day, Month
- Day of the week
- The period of a day (AM/PM)
- Hours, Minutes, Seconds

The picture below demonstrates a clock mode of the program:



To change the values of any of these variables, we need to use Interrupt_1 by invoking function `SIGNAL(INT1_vect)`. One can change a focus from one variable to the next one by pressing a switch connected at PIND0. If the cursor is at variable corresponding to “year”, then when Interrupt_1 happens, the cursor will shift to “Day” variable.

While working in a normal mode, digital clock should print the time, date (year, month, day) and day of the week.

Having shifted the focus we can change the values by pressing the switch at PIND2 or PIND3 which corresponds to Interrupt_2 and Interrupt_3 respectively. These two interrupts are bound to functions SIGNAL(INT2_vect) and SIGNAL(INT3_vect).

They serve as “Arrow up” and “Arrow down” on usual desktop digital clocks.

## Alarm

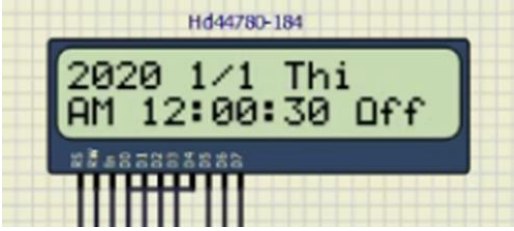
**Alarm** - is a clock that is designed to alert an individual at a specified time.

To switch from clock to alarm mode (from 0 to 1), user should press a button connected to PIND0.

At the mode 1, alarm values do not change over time unless you change them by yourself, because it is NOT the clock.


In the alarm mode, there is one variable more than clock which is the variable Alarm On/Off.

Printed elements are the same as in Clock mode, but in the end, you can also set Alarm On/Off to make it ring in specific time.

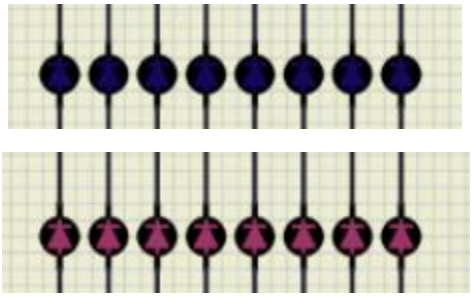



Program checks every second to find whether alarming time and date are same with current clock time and date. If they match, it sets isAlarm to 1.

The Alarm “ring” is specified in main function. If isLaram flag is 1 it invokes function `invokeAlarm()` and displays message ““Wake up!!!” on LCD.


In addition to LCD message, extra blinking LEDs effects was added into implementation.

The pictures below show how a blinking should happen:



# Stopwatch

**Stopwatch** is a handheld timepiece designed to measure the amount of time that elapses between its activation and deactivation. The clock is started and stopped by a person pressing a button. Pressing the top button starts the timer running, and pressing the button a second time stops it, leaving the elapsed time displayed.


In this project, the resolution of the stopwatch will be 1/100 of the sec.

**Important:** Switching to stopwatch or timer mode shouldn't break up a normal operation of the clock. All the computations of the clock must be performed transparently and independently.

In Stopwatch mode (mode = 2), there are 4 variables displayed on the LCD: Hours, Minutes, Seconds, Milliseconds (1/100 of second in this case, but we called the value as millisecond to make it simpler to understand).

**Interrupt_1** (invoking SIGNAL(INT1_vect) function) or **PIND1** is to start and stop stopwatch:


**Interrupt_2** (invoking SIGNAL(INT2_vect) function) is to reset all the values of stopwatch structure to 0.

Finally, **Interrupt_3** is not set at all, because there is no need for it to be usable, as stopwatch functionality is already satisfied by Interrupt 0, 1 and 2;