INHA University in Tashkent

School of Computers and Information Engineering

Spring semester 2020

Embedded Software & Design

SOC3050

# Requirement Analysis

Digital Alarm Clock using Atmega128 Microcontroller

Team: Project 42

|  |  |
|---|---|
| U1610016 | Akmal Karimov |
| U1610041 | Azizbek Kobilov |
| U1610142 | Mirkamol Khamidov |

## Contents

# 1. Introduction

This section gives a scope description and overview of everything included in this document. Also, the purpose of this document is described, and a list of abbreviations and definitions is provided.

An embedded system is a computer system which is a combination of a CPU, memory, and I/O peripheral devices. It has a special function within a larger mechanical or electrical system. Modern embedded systems are often based on microcontrollers. This project uses SimulIDE which replaces the AVR ATmega128.

## 1.1 Purpose of the system

The purpose of this document is to give a detailed description of the requirements for the project. It illustrates the purpose and complete declaration for the development of the system. It also explains system constraints, interface, and interactions with other external applications. This document is primarily intended to be proposed to a Professor for its approval and a reference for developing the first version of the system for the development team.

The project purpose is to make a digital clock on ATmega128. ATmega128 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. It is a highly complex microcontroller where the number of I/O locations supersedes the 64 I/O locations reserved in the AVR instruction set. However, we will test this project on SimulIDE. It is a Real Time Electronic Circuit Simulator which can simulate AVR model that we need.

## 1.2 Scope of the system

Program to show clock, date, and has functions such as alarm and stopwatch, can be implemented as a usual digital clock on needed hardware. It is written in C language. To use the clock we need hex file.

Furthermore, the program can be enhanced. Several other functions can be added to modes. For example, reminder and calendar functions can be new modes for our project.

## 1.3 Objectives and success criteria of the project

The main criterion to success is when clock, alarm and stopwatch work with 100% precision and without any errors. And the factor which leads to the first and main criterion is the right calculation of a 1/100 second. The next main factor is, certainly, the code.

## 1.4 Definitions, acronyms, and abbreviations

AVR – family of microcontrollers which are modified Harvard architecture 8-bit RISC single-chip microcontrollers.

ATmega128 - the high-performance, low-power microchip in AVR family.

Timer0 – 8-bit timer that generates a time delay in AVR.

ISR - an interrupt handler function (interrupt service routine) that runs with global interrupts initially disabled by default with no attributes specified.

TCCR0 (timer/counter control register) – register for setting modes of operation in Timer0.

OCR0 (Output Compare Register) – its content is compared with the content of TCNT0.

TCNT0 – counter register.

### 1.5 References

https://exploreembedded.com/wiki/AVR_Timer_programming

https://chipenable.ru/index.php/programming-avr/171-avr-timer-t0-ch1.html

https://web.ics.purdue.edu/~jricha14/Timer_Stuff/TIMSK.htm

https://www.electronicwings.com/avr-atmega/atmega1632-clear-timer-on-compare-match-ctc-mode

https://en.wikipedia.org/wiki/Requirements_analysis#Use_cases

https://microchipdeveloper.com/8bit:timer0

https://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html#gad28590624d422cdf30d626e0a506255f

Book: “The avr microcontroller and embedded system using assembly and c” by Muhammad Ali Mazidi, Sarmad Naimi, and Sepehr Naimi (chapters: 7, 9, 10, 12)

Also, slides and online video materials provided in lectures.

### 1.6 Overview

The parts below describe the functionality and the problems of the current system, requirements elicitation and the analysis, a functional overview of the system, functional and non-functional requirements, and, finally, the scenario.

## 2. Current system

The current system state is a working code without errors. Hex file is needed for running the code on simulator. This file is built in Atmel Studio 7, with the help of the main code and its header files.

## 3. Proposed system

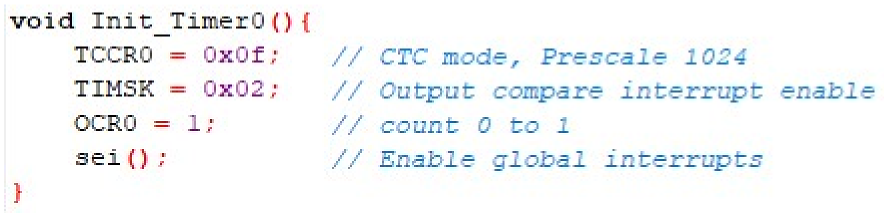
Our proposed system was to make a digital clock with features like setting date, time, alarm and stopwatch. We wrote the code in C programming language using Atmel Studio 7. We tested our system on SimulIDE.

## 3.1 Overview

The user needs SimulIDE_0.3.12-SR8 to run the app in simulated ATmega128 device. The system requires to set date, period of day and time in the beginning. To make an input switches of the device are used. After this menu is displayed where user chooses one of three modes: clock, alarm, stopwatch.

## 3.2 Functional requirements

- First, we initialize ports A for LCD, b for output on led and D for input through switches
- UI functions like Welcome are called
- Set date and time function is called which uses switches for input;
- Menu UI is called to take inputs for choosing mode
- Initializer for Timer0 is called



```
void Init_Timer0(){
    TCCR0 = 0x0f;    // CTC mode, Prescale 1024
    TIMSK = 0x02;    // Output compare interrupt enable
    OCR0 = 1;        // count 0 to 1
    sei();           // Enable global interrupts
}
```

The function above sets Timer0 mode, prescale, interrupt mask and OCR0 flag to compare with TCCR0.

For ISR of Timer0 we needed the following calculations:

1. TOV0 = (1/(14.7456Mhz) * 1024(prescale)) * 100 (CTC) = 6.945s
2. 6.945s * 144 = 1 sec
3. in our case -> CTC = 60 , multiplier = 100

So, in ISR we count until 60 to increment a centisecond (1/100). Then if the count is centiseconds equals 100, we increment second value and so on. So, this is the logic of timing.

In alarm there should be inputs for setting time of alert (blinking of leds). So, if the the time variables equal the alarm variables, the system starts blinking the leds.

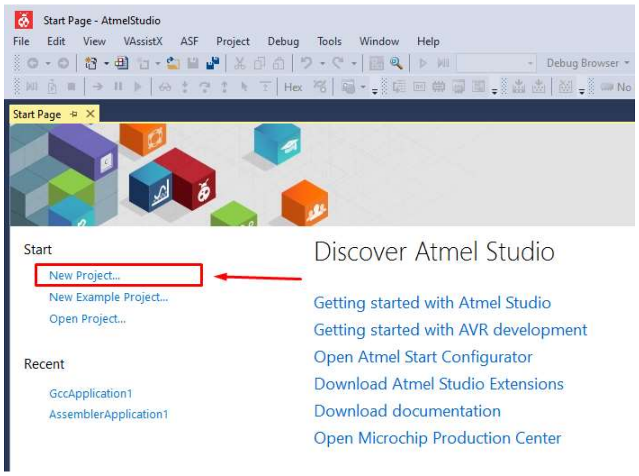
Stopwatch works as clocks. It differs from clock by having its own variables to count in ISR and display.

## 3.3 Non-functional requirements

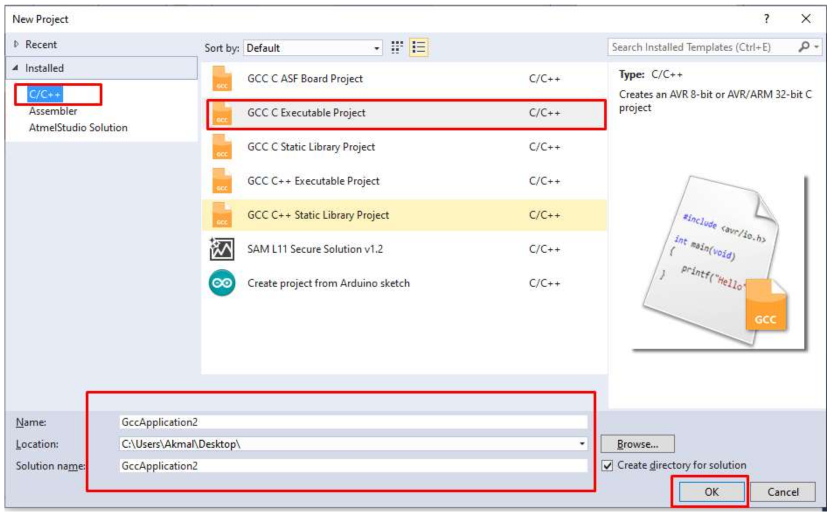
We provide both source code and hex file for running in the system or simulator. So, user only needs to install SimulIDE_0.3.12-SR8 to run the app in simulated ATmega128 device, or, in case of device, user needs a bootloader to connect and download the hex file in AVR ATmega128.

In case of there is no hex file, user needs to install Atmel Studio 7 and build the provided source code to generate a hex file.

Atmel Studio (7.0) should be downloaded and installed. After the platform is opened "New Project" should be created.


Start Page - AtmelStudio
File Edit View VAssistX ASF Project Debug Tools Window Help
Debug Browser
Hex
Start Page
Start
New Project...
New Example Project...
Open Project...
Recent
GccApplication1
AssemblerApplication1
Discover Atmel Studio
Getting started with Atmel Studio
Getting started with AVR development
Open Atmel Start Configurator
Download Atmel Studio Extensions
Download documentation
Open Microchip Production Center

Then the type of template and the project's directory path should be chosen.



| | |
|---|---|
| GCC C ASF Board Project | C/C++ |
| GCC C Executable Project | C/C++ |
| GCC C Static Library Project | C/C++ |
| GCC C++ Executable Project | C/C++ |
| GCC C++ Static Library Project | C/C++ |
| SAM L11 Secure Solution v1.2 | C/C++ |
| Create project from Arduino sketch | C/C++ |

Type: C/C++

Creates an AVR 8-bit or AVR/ARM 32-bit C project

```
#include <avr/io.h>
int main(void)
{
    printf("Hello"
}
```

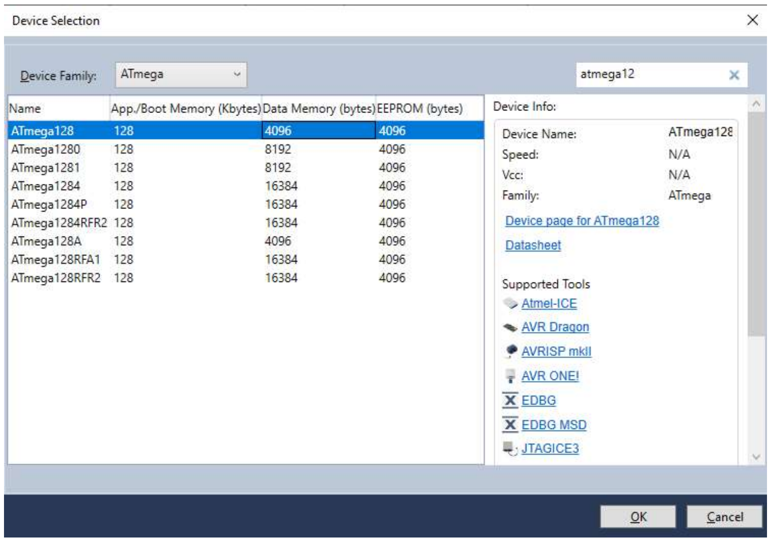
Name: GccApplication2

Location: C:\Users\Akmal\Desktop\

Solution name: GccApplication2

☑ Create directory for solution

In the Device Selection dialog select ATmega as the Device family. Then choose ATmega128 (or any other Chips you want to use) and press OK.



| Name | App./Boot Memory (Kbytes) | Data Memory (bytes) | EEPROM (bytes) |
|---|---|---|---|
| ATmega128 | 128 | 4096 | 4096 |
| ATmega1280 | 128 | 8192 | 4096 |
| ATmega1281 | 128 | 8192 | 4096 |
| ATmega1284 | 128 | 16384 | 4096 |
| ATmega1284P | 128 | 16384 | 4096 |
| ATmega1284RFR2 | 128 | 16384 | 4096 |
| ATmega128A | 128 | 4096 | 4096 |
| ATmega128RFA1 | 128 | 16384 | 4096 |
| ATmega128RFR2 | 128 | 16384 | 4096 |

Device Info:

| | |
|---|---|
| Device Name: | ATmega128 |
| Speed: | N/A |
| Vcc: | N/A |
| Family: | ATmega |

Device page for ATmega128

Datasheet

Supported Tools

Atmel-ICE

AVR Dragon

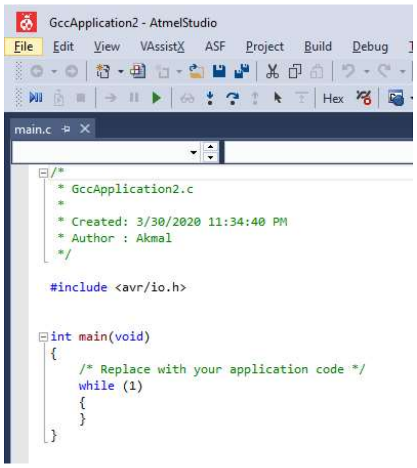
AVRISP mkII

AVR ONE!

EDBG

EDBG MSD

JTAGICE3

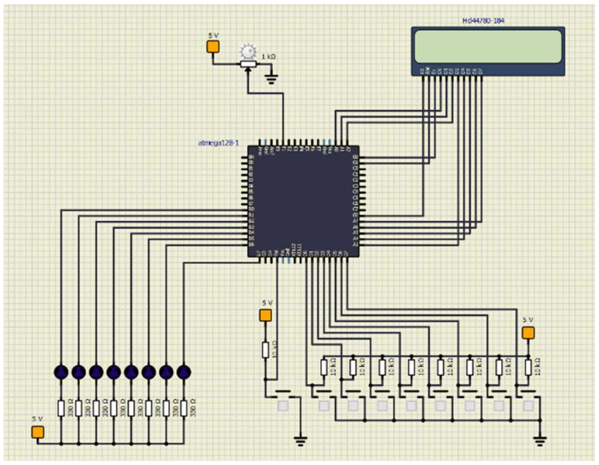


```
/*
 * GccApplication2.c
 *
 * Created: 3/30/2020 11:34:40 PM
 * Author : Akmal
 */

#include <avr/io.h>


int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Finally, in the created project C file we write the code for the digital clock. The code is provided, so, the user needs to copy and paste our code to main.c and include header files to the project. After this Build → Build Solution should be pressed. And that all to generate hex file. However, this file is provided. Instructions above is to open code and change some parts.

User needs to download SimulIDE_0.3.12-SR8 to run the app in simulated ATmega128 device.

After downloading and opening the simulator, simulation circuit should be opened. This circuit is also provided - ATMEGA128_20200426_1330.simu



Other steps of using simulator is written in Manual Part of the Report.

## 4. Scenario

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