INHA University in Tashkent

School of Computers and Information Engineering

Spring semester 2020

Embedded Software & Design

SOC3050

SW Design specification

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

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

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 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

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.

1.4 Overview

The remainder of this document includes four chapters.

The second one provides an overview of the system functionality, the system constraints, and assumptions about the product.

The third chapter provides the requirements specification in precise terms and a description of the different system interfaces. Different specification techniques are used to specify the requirements more precisely for different audiences.

The fourth chapter deals with future plans and suggestions for development.

2. General Description

This section gives an overview of the entire system. The system is explained in the context of how the system interacts with the real world, and present the basic functionality. Finally, system limitations are presented.

2.1 Product Functions

At the beginning of the application (after you start simulation on SimulIDE) the date (in order: year, month, day), time and period of day (AM or PM). After this the mode should be chosen (clock, alarm or stopwatch). We can set alarm by giving exact time. When the time of alarm comes, all leds start blinking during a minute or until when you want to stop. The next function is stopwatch with resolution of 1/100 sec.

All functions:

- setting date and time;

- choosing the mode on the menu;

- displaying clock in format:

    Line 1: year date / month day;

    Line 2: hour(AM/PM), minute, second;

- setting alarm;

- stopwatch

2.2 Operating Principle

This section describes the timing logic for clock. For timing Timer0 on CTC mode is used.

- 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.

2.3 Constraints

There was a problem in calculation of 1/100 second. In the lectures we calculate the second as 144 value of CTC counter. This means, the OCR0 = 99. To calculate the 1/100 we changed the value of OCR0 to 1. After this our calculations showed result which does not match exactly with 1 second. So, we decided to match the most accurate value. We decided to choose 99 counts as 1/100 second. And this gives us very accurate timing.

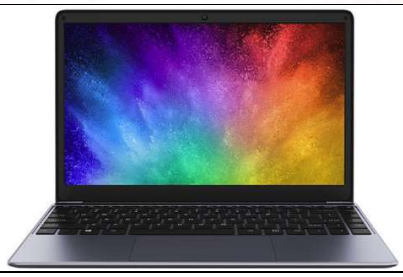
Another constraint is that we do not use all switches, to be precise, switches 5 and 6. So, this switches have no function.

3. Specific Requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

3.1 Hardware Requirements

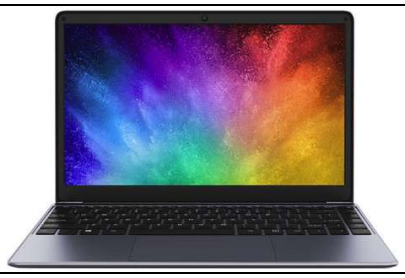
If the kit for Atnega128 is available:

	ATmega128 AVR microcontroller
	I/O Board
	LCD Control
	USB Cable
	Laptop

Board Specifications:

CPU	ATMEGA128A
Architecture	AVR
Operating Voltage	5V USB
Flash Memory	128KB of 4KB used By Bootloader
SRAM	4K
Clock Speed	14.7456Mhz
Analog I/O Pin	8
EEprom	4K
PWM OUTPUT	6
Uart	2
I2C	1
Digital I/O Pin	40
PCB Size	100 X 79
USB Cable Provided	Mini B Type Cable 1.5m
LCD Provided	16X2 Character LCD

In our case we need only a laptop which supports Atmel Studio 7, and SimulIDE:

	Laptop

Software part:

- Windows OS (recommended)
- Atmel Studio (7.0)
- SimulIDE_0.3.12-SR8
- Bootloader (in case of kit)
- The code is written in C language

3.2 Program Components:

Program consists of 4 main files:

- main.c – the whole code logic in C language
- _main.h – header file which includes some settings (such as variable types and so on)
- _lcd.h – header file which includes base functions (such as Initialization of Ports, LCD display, Commands and so on)
- Timer0_clock.hex – hex file generated by Atmel Studio after building project. It is used for download into SimulIDE to run the app. (in case of kit – downloading by Bootloader to Atmega128).

3.3 Software Requirements

- Windows OS (recommended)
- Atmel Studio (7.0)
- SimulIDE_0.3.12-SR8
- Bootloader (in case of kit)
- The code is written in C language
- Libs:
    - <avr/io.h>
    - <util/delay.h>
    - <avr/interrupt.h>
    - <stdio.h>
    - <string.h>
    - <stdlib.h>

3.3 Code Design

In this part, you can see code chunks with explanations.

```
#include "_main.h"
#include "_lcd.h"

#define CTCnum 59 // CTC number ~ 60

// CONSTs
const int MONTHS[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
const char WEEKDAYS[7][4] = {"Mon", "Tue", "Wed", "Thu", "Fri" , "Sat", "Sun"};

// DATE + TIME VALUES
int year = 2020, month = 11, day = 30, maxDayValue, leapYear; // for Date
int fullHour, hour, minute, second, msecond; // for Clock
int stopwatchHour, stopwatchMinute, stopwatchSecond, stopwatchMsec; // for Stopwatch
int alarmHour, alarmMinute;  // for Alarm
int inputNum; // for switch inputing values

// CHARs
unsigned char tCNT, blinks, blinksDelay;
char date[16], time[16], stopwatch[16], alarm[16], weekday[4];
char midday[3] = "AM", aMidday[3] = "AM";
char xPos, switchKey, targetIndex;

// FLAGs
int isPM, isAlarmPM;
char stopwatch_ON, alarm_ON, isRinging, clockIsSet;
char inMain, inStopwatch, inAlarm, inClock = 1;
char OFF = 0xff;
```

```
// ************************************************************************
// ************************  FUNCTIONS  ***********************************
// ************************************************************************
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
}

void Init_Ports(){
    // Switch inputs - all work
    PORTD = 0xff; DDRD = 0x00;
    // LED outputs - turn off
    PORTB = 0xff; DDRB = 0xff;
    // LCD outputs
    PORTA = 0x00; DDRA = 0xff;
}

void Init_Devices(){
    Init_Ports();
    LCD_Init();
}

// used for switch inputs when setting Clock or Alarm
void switchController(){
    switch(switchKey)
    {
        case 0xfe: // 1st - decrease number
        inputNum = -1;
        break;

        case 0xfd: // 2nd - increase number
        inputNum = +1;
        break;

        case 0xfb: // 3rd - go prev
        inputNum = 0;
        if(targetIndex!=0) targetIndex--; //min 0 value
        break;

        case 0xf7: // 4th - go next
        inputNum = 0;
        if(targetIndex!=11) targetIndex++; //max 10 value
        break;

        case 0x7f: // last - exit to main
        if(clockIsSet) mainMenu();
        break;
    }
}
```

```
// ************************************************************************
// ***************************** CLOCK ************************************
// ************************************************************************

// change values of Date & Time on input (Decrease or Increase)
void changeValue(int* target, int mod)
{
    // mod = # of possible values
    // mod = 60 -> result range[0~59]
    // mod = 2 -> result range[0~1]
    // mod = N -> result range[1~N]
    int zero = (mod == 60) ? -1 : 0;
    int result = (mod == 2) ? 0 : mod;
    *target += inputNum;
    *target %= mod;
    *target = (*target == zero) ? ((mod == 60) ? 59 : result) : abs(*target);
}

void fullHourChange(int n){
    fullHour += n;
    fullHour %= 24;
    fullHour = (fullHour == -1) ? 23 : fullHour;

    isPM = (fullHour > 11) ? 1 : 0;
    strcpy(midday, (isPM) ? "PM" : "AM");
}

// knowing year, month, day -> get Week day
int getWeekday(int y, int m, int d)
{
    static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
    y -= m < 3;
    return ((y + y/4 - y/100 + y/400 + t[m-1] + d - 1) % 7);
}
```

```
// get max day of the given month
void getMaxDayValue()
{
    maxDayValue = MONTHS[month - 1];
    if(month == 2)
    maxDayValue = (leapYear == 1) ? 29 : 28;
}

// time in 12-hour method
void printTime(){
    sprintf(time, "%s %02d:%02d:%02d", midday, hour, minute, second);
    LCD_PosPrint(0, 1, time);
    LCD_Pos(xPos, 1);
}

void printDate(){
    sprintf(date, "%04d %02d/%02d %s", year, day, month, weekday);
    LCD_PosPrint(0, 0, date);
    LCD_Pos(xPos, 0);
}
// update and get the Date (maxDayValue and Weekday)
void getDate()
{
    // whether given year is leap or not
    leapYear = (((year % 4 == 0) && (year % 100!= 0)) || (year % 400 == 0)) ? 1 : 0;
    getMaxDayValue();
    int index = getWeekday(year, month, day);
    strcpy(weekday, WEEKDAYS[index]);
    if (inClock) printDate();
}
// CLOCK Screen
void goClock(){
    inStopwatch = inAlarm = inMain = 0;
    inClock = 1;
    Cursor_NoBlink();
    getDate();
    printTime();
}
```

```
158 void setClock()
159 {
160     Cursor_Blink();
161     getDate();
162     clockIsSet = switchKey = targetIndex = 0;
163     LCD_Pos(3,0);
164 
165     // indexes ->
166     //0 = year | 1 = month | 2 = day | 3 = hour | 4 = min | 5 = sec
167     while(targetIndex < 6)
168     {
169         switchKey = 0xff & PIND; // all inputs work
170         switchController();
171 
172         // if Input Pressed
173         if(switchKey != 0xff)
174         {
175             // allow only 1 press in 200ms
176             _delay_ms(200);
177 
178             // change DATE
179             if (targetIndex == 0){
180                 xPos = 3;
181                 year += inputNum;
182                 getDate();
183             }
184             else if(targetIndex == 1)
185             {
186                 xPos = 9;
187                 changeValue(&month, 12);
188                 getDate();
189             }
190             else if(targetIndex == 2){
191                 xPos = 6;
192                 changeValue(&day, maxDayValue);
193                 getDate();
194             }
```

```
195
196             // change TIME
197             else if(targetIndex == 3){
198                 xPos = 4;
199                 changeValue(&hour, 12);
200                 fullHourChange(inputNum);
201
202             }
203             else if(targetIndex == 4){
204                 xPos = 7;
205                 changeValue(&minute, 60);
206             }
207             else if(targetIndex == 5){
208                 xPos = 10;
209                 changeValue(&second, 60);
210             }
211
212             // print the DATE & TIME on LCD
213
214             if(targetIndex > 2) printTime();
215             else printDate();
216         }
217     }
218
219     clockIsSet = 1;
220     goClock();
221 }
```

```
// ************************************************************************
// ************************** STOPWATCH ***************************
// ************************************************************************

void printStopwatch()
{
    sprintf(stopwatch, "%02d:%02d:%02d:%02d", stopwatchHour, stopwatchMinute, stopwatchSecond, stopwatchMsec);
    LCD_PosPrint(0, 1, stopwatch);

    switchKey = 0xff & PIND;
    switch(switchKey)
    {
        case 0xfe: // 1st switch - START/PAUSE
            stopwatch_ON = !stopwatch_ON; // clicker
            LCD_PosPrint(0, 0, stopwatch_ON ? "1-Pause" : "1-Start");
            break;
        case 0xfd: // 2nd switch - STOP
            LCD_PosPrint(0, 0, "1-Start");
            stopwatch_ON = 0;
            stopwatchHour = stopwatchMinute = stopwatchSecond = stopwatchMsec = 0;
            break;
        case 0x7f: // last switch - EXIT
            mainMenu();
            break;
    }
}

// STOPWACH Screen
void goStopwatch()
{
    inClock = inAlarm = inMain = 0;
    inStopwatch = 1;
    Cursor_NoBlink();
    LCD_PosPrint(0, 0, "1-Start   2-Stop");
    printStopwatch();
}
```

```
// ************************************************************************
// ************************** ALARM ***************************************
// ************************************************************************

void printAlarm(){
    sprintf(alarm, "%s %02d:%02d", aMidday, alarmHour, alarmMinute);
    LCD_PosPrint(0,1,alarm);
    LCD_Pos(xPos, 1);
}

void setAlarm(){

    switchKey = 0xff & PIND;
    switchController();

    if(switchKey != 0xff)
    {
        // allow only 1 press in 200ms
        _delay_ms(200);
        if(targetIndex > 3) targetIndex = 3;
        // change TIME
        if(targetIndex == 0){
            xPos = 0;
            changeValue(&isAlarmPM, 2);
            strcpy(aMidday, isAlarmPM ? "PM" : "AM");
        }
        else if(targetIndex == 1){
            xPos = 4;
            changeValue(&alarmHour, 12);
        }
        else if(targetIndex == 2){
            xPos = 7;
            changeValue(&alarmMinute, 60);
            LCD_Clear();
            LCD_PosPrint(0,0,"Wanna Set?");
        }
        // alarm is set
        else if(targetIndex == 3){
            LCD_PosPrint(0, 0, "Alarm is Set:");
            alarm_ON = 1;
        }

        // dont show alarm if last switch pressed
        if(switchKey!= 0x7f) printAlarm();
    }
}
```

```
// ALARM screen
void goAlarm()
{
    inStopwatch = inClock = inMain = 0;
    inAlarm = 1;
    Cursor_Blink();
    LCD_PosPrint(xPos, 0, alarm_ON ? "Alarm is Set:" : "Alarm isn't Set:");
    printAlarm();

    if(!alarm_ON){
        strcpy(aMidday, midday);
        isAlarmPM = isPM;
        alarmHour = hour;
        alarmMinute = minute;
    }
}

void Led_Alarm()
{
    // alarming flag
    isRinging = (blinks < 11 && strcmp(aMidday,midday) == 0
                && alarmHour == hour && alarmMinute == minute) ? 1 : 0;

    // turn of led
    if (!isRinging) PORTB = 0xff;
    // blink every 2 sec
    else if(blinksDelay > 1){
        blinks++;
        OFF = ~OFF;
        PORTB = OFF;
        blinksDelay = 0;
    }
}
```

```
// ************************************************************************
// ********************** ISR for timer0 CTC mode **********************
// ************************************************************************

// TOV0 = (1/(14.7456Mhz) * 1024(prescale)) * 100 (CTC) = 6.945s
// 6.945s * 144 = 1 sec
// in our case -> CTC = 60 , multiplier = 100
ISR(TIMER0_COMP_vect)
{
    tCNT++;
    if(tCNT == CTCnum){
        tCNT = 0;
        msecond++;

        // reset milliseconds and update seconds
        if(msecond == 99){
            second++;
            msecond = 0;
            if(isRinging) blinksDelay++;
        }
        // reset seconds and update minutes
        if(second == 60){
            minute++;
            second=0;
        }
        // reset minutes and update both 12 and 24 hours
        if(minute == 60){
            hour++;
            fullHour++;
            minute=0;
            //reset 12-hours
            if(hour == 13) hour = 1;
        }
        // update midday values
        if(fullHour == 12)
            fullHourChange(0);
        // reset 24-hours w/ #-blinks and update days and middays
```

```
        if(fullHour == 24){
            fullHourChange(0);
            LCD_Clear();
            day++;
            getDate();
            blinks = 0;
        }
        // reset days and update months
        if(day > maxDayValue){
            month++;
            day = 1;
        }
        // reset months and update year
        if (month>12)
        {
            year++;
            month = 1;
        }

        // ====================================
        // WHERE I AM - Places
        // ====================================
        if(inMain){
            switchKey = 0xff & PIND;
            if(switchKey == 0xfe) goClock(); //1st switch
            if(switchKey == 0xfd) goAlarm(); //2nd switch
            if(switchKey == 0xfb) goStopwatch(); //3st switch
        }
        else if(inStopwatch) printStopwatch();
        else if(inClock) printTime();
        else if(inAlarm) setAlarm();

        // if pressed -> allow only 1 press in 200ms
        if(switchKey != 0xff) _delay_ms(200);

        // =====   If Flags ON =======
        if(alarm_ON) Led_Alarm();
```

```
415         if(stopwatch_ON)
416         {
417             stopwatchMsec = msecond + 1;
418             // update seconds
419             if(stopwatchMsec == 99)
420                 stopwatchSecond++;
421             // reset seconds and update minutes
422             if (stopwatchSecond == 60){
423                 stopwatchMinute++;
424                 stopwatchSecond = 0;
425             }
426             // reset minutes and update hours
427             if (stopwatchMinute == 60){
428                 stopwatchHour++;
429                 stopwatchMinute = 0;
430             }
431         }
432 
433         // ============ for cases when SETTING Clock =============
434         switchKey = 0xff & PIND;
435         // to GO to MainMenu if last Switch Pressed and Clock is Set
436         if(switchKey == 0x7f && clockIsSet) mainMenu();
437         // press 7th switch to RESET CLOCK while ONLY in Clock Screen and it is Set
438         else if(inClock && switchKey == 0xbf && clockIsSet) setClock();
439     }
440 }
```

```
// ************************************************************
// ******************** WELCOME + MENU ************************
// ************************************************************

void mainMenu()
{
    targetIndex = xPos = 0;
    inStopwatch = inClock = inAlarm = 0;
    inMain = 1;
    Cursor_NoBlink();
    LCD_PosPrint(xPos, 0, "1-Clock  2-Alarm");
    LCD_PosPrint(xPos, 1, "  3-Stopwatch   ");
}

void Welcome()
{
    char TEXT[16][16] = {
        "    WELCOME    ",
        "Press for Next ",
        "We use switches",
        "1 2 3 4 and 7 8",
        "Inside Clock:  ",
        "1 2 3 4     7 8",
        "  Go Back - 8  ",
        "Reset Clock - 7",
        "Update num: 1 2",
        "3-prev   4-next",
        "Inside Alarm:  ",
        "1 2 3 4       8",
        "In Stopwatch:  ",
        "1-start 2-stop ",
        "First SET DATE ",
        "-------------->"
    };

    for(int i = 0; i < 16; i++)
    {
```

```
    for(int i = 0; i < 16; i++)
    {
        switchKey = 0xff;
        LCD_PosPrint(0, 0, TEXT[i]);
        LCD_PosPrint(0, 1, TEXT[++i]);
        while(switchKey == 0xff) switchKey = 0xff & PIND;
        _delay_ms(200);
    }
}

// ****************************************************************
// ************************** MAIN ********************************
// ****************************************************************

int main(void)
{
    // set all them to ZERO
    fullHour = hour = minute = second = msecond = stopwatchHour = stopwatchMinute
    = stopwatchSecond = stopwatchMsec = alarmHour
    = alarmMinute = inputNum = tCNT = blinks = blinksDelay
    = xPos = targetIndex = isPM = isAlarmPM
    = stopwatch_ON = alarm_ON = isRinging = inMain
    = inStopwatch = inAlarm = clockIsSet = 0;

    Init_Devices();
    Welcome();
    Cursor_Blink();
    setClock();
    mainMenu();
    Init_Timer0();

    while (1);
}
```

Discussions and Future Plans

This project was excellent practice for AVR programming. We learned a plenty of interesting things from this project. For example, AVR programming on C, Timer programming, Interrupt programming, LCD interfacing and so on. Not only practice, but also theory was quite interesting.

In future, we can upgrade our digital clock and add some more functions. We can add calendar mode, reminder mode, calculator mode or even game mode and other features. There is a possibility to implement some sensors and joystick for future plans.