Gliwice, 21.01.2018

Microprocessor Systems

Project

Alarm Clock with LED Dot Matrix MAX7219

Project supervisor:
dr hab. inż. Rober Czerwiński

Piotr Gryt
Macrocourse, sem. 5

1. Project description

Alarm Clock with LED Dot Matrix MAX7219

Hardware:
LED dot matrix display (MAX7219), Push buttons

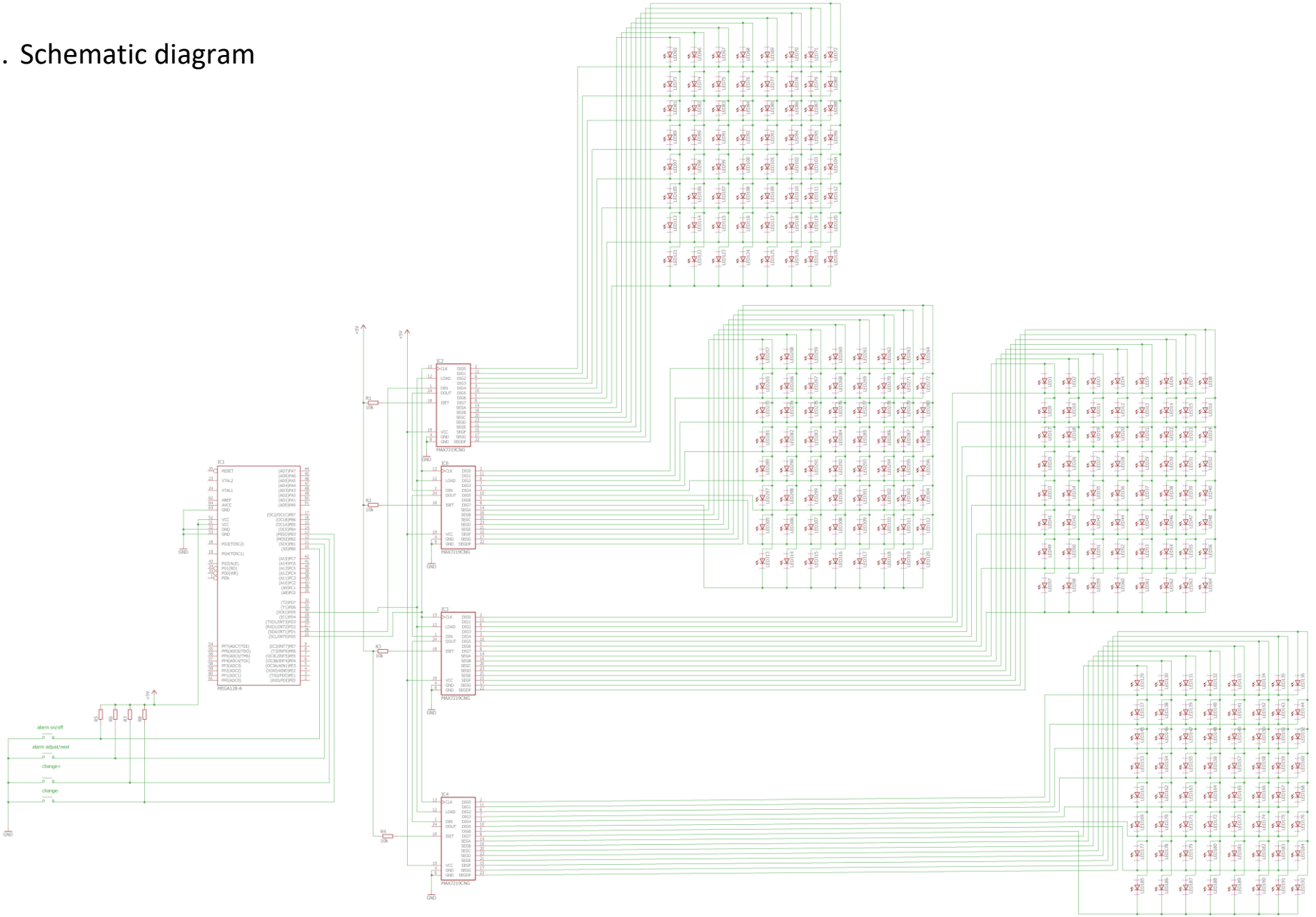
Description:
Display shows successively hours and minutes, then switches to month and day, and then a year. All of these are set at the start of the device (but with opposite order). Alarm may be set after pressing one of buttons and enabled on one other. When the alarm is enabled, the bottom line lights.

Buttons:

1) Increment current digit
2) Decrement current digit
3) Proceed to next digit (or leave if it was last digit)
4) Set the alarm clock
5) Enable alarm clock

Buttons 1, 2, 3 work only on initial setting of time and after pressing the button for setting the alarm clock.

2. Schematic diagram



3. Hardware difficulties

- For the MAX7219, serial data at DIN, sent in 16-bit packets, is shifted into the internal 16-bit shift register with each rising edge of CLK regardless of the state of LOAD.

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
x	x	x	x	ADDRESS				DATA							

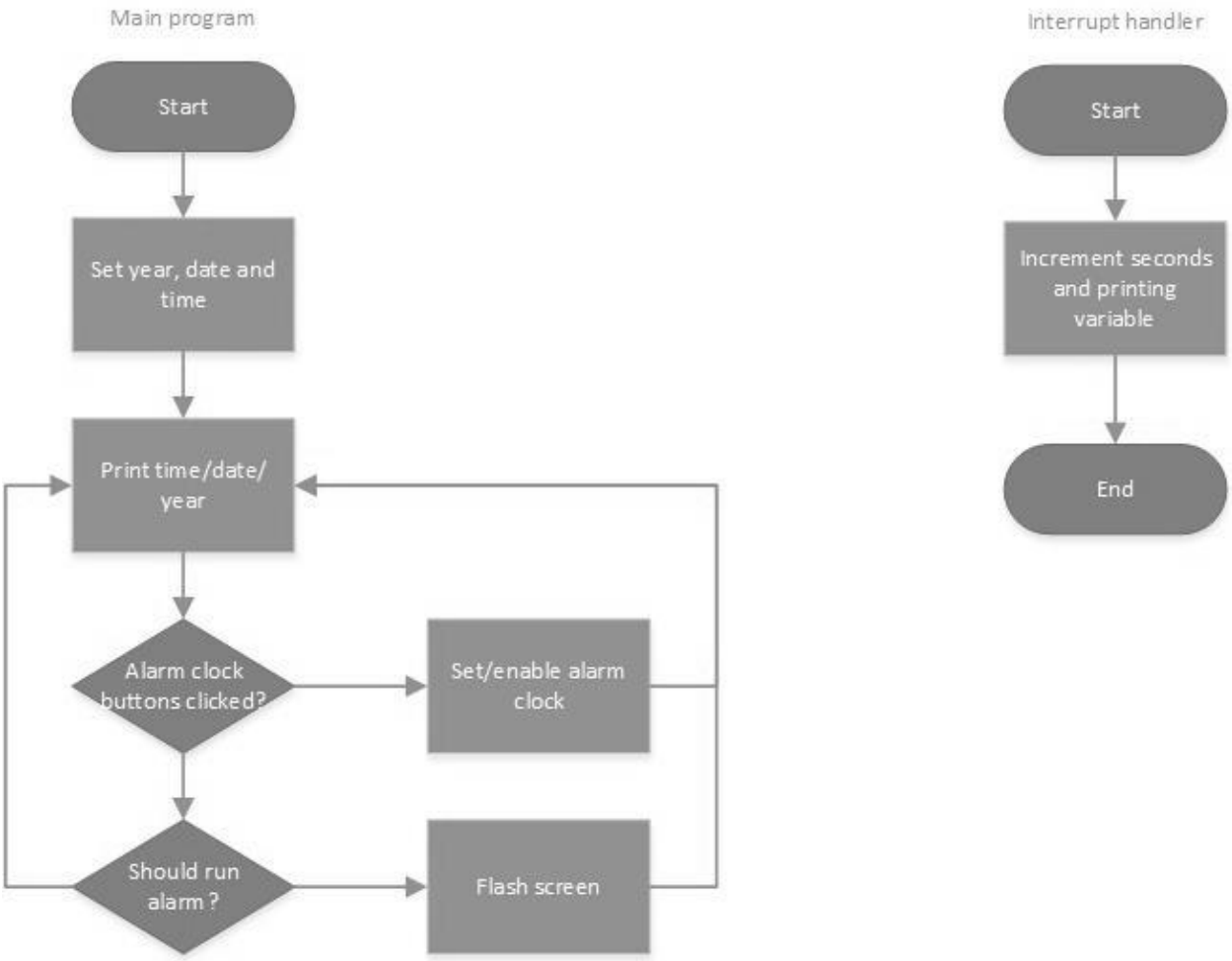
D8–D11 contain the register address. D0–D7 contain the data.

- The Alarm Clock increments seconds precisely due to an overflow interrupt and correctly set the TCNT1 register:

```
#pragma vector = TIMER1_OVF_vect
__interrupt void timer1()
{
	TCNT1 = 36735;
	time[5]++;	// increment seconds
}
```

- Signal from push buttons oscillates very much, so after clicking the button it changed it state many times which is not a desired effect. It was eliminated with some expanded login in the program.

4. Flowchart



5. Source Code

```
#include <stdio.h>
#include <ioavr.h>
#include <intrinsics.h>
#define MAX7219_DINDDR DDRD
#define MAX7219_DINPORT PORTD
#define MAX7219_DININPUT PD2
#define MAX7219_CLKDDR DDRD
#define MAX7219_CLKPORT PORTD
#define MAX7219_CLKINPUT PD3
#define MAX7219_LOADDDR DDRD
#define MAX7219_LOADPORT PORTD
```

```
#define MAX7219_LOADINPUT PD4
//setup number of chip attached to the board
#define MAX7219_ICNUMBER 4
//define registers
#define MAX7218_REGNOOP 0x00
#define MAX7218_REGDIGIT0 0x01
#define MAX7218_REGDIGIT1 0x02
#define MAX7218_REGDIGIT2 0x03
#define MAX7218_REGDIGIT3 0x04
#define MAX7218_REGDIGIT4 0x05
#define MAX7218_REGDIGIT5 0x06
#define MAX7218_REGDIGIT6 0x07
#define MAX7218_REGDIGIT7 0x08
#define MAX7218_REGDECODE 0x09
#define MAX7218_REGINTENSITY 0x0A
#define MAX7218_REGSCANLIMIT 0x0B
#define MAX7218_REGSHUTDOWN 0x0C
#define MAX7218_REGTEST 0x0F
/*
 * shift out a byte
 */
void max7219_shiftout(unsigned char bytedata)
{
	unsigned char j = 0;
	//the shift is made in reverse order for this ic
	for (j = 8; j > 0; j--)
	{
		unsigned char val = (bytedata & (1 << (j - 1))) >> (j -
1);
		MAX7219_CLKPORT &= ~(1 << MAX7219_CLKINPUT); //set the
serial-clock pin low
		if (val)
			MAX7219_DINPORT |= (1 << MAX7219_DININPUT);
		else
			MAX7219_DINPORT &= ~(1 << MAX7219_DININPUT);
		MAX7219_CLKPORT |= (1 << MAX7219_CLKINPUT); //set the
serial-clock pin high
```

```
	}
}
/*
 * shift out data to a selected number
 */
void max7219_send(unsigned char icnum, unsigned char reg, unsigned
char data)
{
	unsigned char i = 0;
	if (icnum < MAX7219_ICNUMBER)
	{
		MAX7219_LOADPORT &= ~(1 << MAX7219_LOADINPUT); //load
down
		//send no op to following ic
		for (i = icnum; i < (MAX7219_ICNUMBER - 1); i++)
		{
			max7219_shiftout(MAX7218_REGNOOP); //no op reg
			max7219_shiftout(MAX7218_REGNOOP); //no op data
		}
		//send info to current ic
		max7219_shiftout(reg); //send reg
		max7219_shiftout(data); //send data
		//send no op to previous ic
		for (i = 0; i < icnum; i++) {
			max7219_shiftout(MAX7218_REGNOOP); //no op reg
			max7219_shiftout(MAX7218_REGNOOP); //no op data
		}
		MAX7219_LOADPORT |= (1 << MAX7219_LOADINPUT); //load up
	}
}
/*
 * set shutdown for a selected ic
 */
void max7219_shutdown(unsigned char icnum, unsigned char value)
{
	if (value == 0 || value == 1)
		max7219_send(icnum, MAX7218_REGSHUTDOWN, value);
```

```
}
/*
 * set brightness for a selected ic
 */
void max7219_intensity(unsigned char icnum, unsigned char value)
{
	if (value < 16)
		max7219_send(icnum, MAX7218_REGINTENSITY, value);
}
/*
 * set test mode for a selected ic
 */
void max7219_test(unsigned char icnum, unsigned char value)
{
	if (value == 0 || value == 1)
		max7219_send(icnum, MAX7218_REGTEST, value);
}
/*
 * set active output for a selected ic
 */
void max7219_scanlimit(unsigned char icnum, unsigned char value)
{
	max7219_send(icnum, MAX7218_REGSCANLIMIT, value);
}
/*
 * set decode mode for a selected ic
 */
void max7219_decode(unsigned char icnum, unsigned char value)
{
	max7219_send(icnum, MAX7218_REGDECODE, value);
}
/*
 * init the shift register
 */
void max7219_init() {
	//output
	MAX7219_DINDDR |= (1 << MAX7219_DININPUT);
```

```
	MAX7219_CLKDDR |= (1 << MAX7219_CLKINPUT);
	MAX7219_LOADDDR |= (1 << MAX7219_LOADINPUT);
	//low
	MAX7219_DINPORT &= ~(1 << MAX7219_DININPUT);
	MAX7219_CLKPORT &= ~(1 << MAX7219_CLKINPUT);
	MAX7219_LOADPORT &= ~(1 << MAX7219_LOADINPUT);
}
void initDevices()
{
	DDRB = 0x00;
	PORTB = 0xFF;
	TCCR0 |= (1 << CS01);
	TCCR1B |= (1 << CS12);
	TIMSK |= (1 << TOIE0) | (1 << TOIE1) | (1 << TOIE3);
}
int numTable[10][7] = {
	{ 28, 34, 34, 34, 34, 34, 28 },		// 0
	{ 2, 2, 2, 2, 2, 2, 2 },		// 1
	{ 28, 2, 2, 28, 32, 32, 28 },		// 2
	{ 28, 2, 2, 28, 2, 2, 28 },			// 3
	{ 34, 34, 34, 28, 2, 2, 2 },		// 4
	{ 28, 32, 32, 28, 2, 2, 28 },		// 5
	{ 28, 32, 32, 28, 34, 34, 28 },		// 6
	{ 126, 2, 4, 8, 16, 16, 16 },		// 7
	{ 28, 34, 34, 28, 34, 34, 28 },		// 8
	{ 28, 34, 34, 28, 2, 2, 28 }		// 9
};
unsigned char timer_enabled = 1;
void Draw(unsigned char ic, int number)
{
	unsigned char cur_line = 0;
	for (cur_line = 0; cur_line < 7; cur_line++)
	{
		max7219_send(ic, cur_line + 1,
numTable[number][cur_line]);
	}
	if (timer_enabled)
```

```
            max7219_send(ic, 8, 255);
        else
            max7219_send(ic, 8, 0);
}
void DrawWithColon(unsigned char ic, int number)
{
        //int tempNum[7] = numTable[number];
        // OR
        int tempNum[7] = 0;
        unsigned char numCpy = 0;
        for (numCpy = 0; numCpy < 7; numCpy++)
        {
            tempNum[numCpy] = numTable[number][numCpy];
        }
        tempNum[2] += 128;
        tempNum[4] += 128;
        unsigned char cur_line = 0;
        for (cur_line = 0; cur_line < 7; cur_line++)
        {
            max7219_send(ic, cur_line + 1, tempNum[cur_line]);
        }
        if (timer_enabled)
            max7219_send(ic, 8, 255);
        else
            max7219_send(ic, 8, 0);
}
void DrawWithDot(unsigned char ic, int number)
{
        //int tempNum[7] = numTable[number];
        // OR
        int tempNum[7] = 0;
        unsigned char numCpy = 0;
        for (numCpy = 0; numCpy < 7; numCpy++)
        {
            tempNum[numCpy] = numTable[number][numCpy];
        }
        tempNum[6] += 128;
```

```
	unsigned char cur_line = 0;
	for (cur_line = 0; cur_line < 7; cur_line++)
	{
		max7219_send(ic, cur_line + 1, tempNum[cur_line]);
	}
	if (timer_enabled)
		max7219_send(ic, 8, 255);
	else
		max7219_send(ic, 8, 0);
}
void clearDisplay(unsigned char ic)
{
	for (int j = 0; j < 8; j++)
		max7219_send(ic, j + 1, 0);
}
void lightDisplay(unsigned char ic)
{
	for (int j = 0; j < 8; j++)
		max7219_send(ic, j + 1, 255);
}
void FormatDate();
int ButtonLogic(int temp_arr[], int cur_digit);
void SetTime();
void SetDate();
void SetYear();
void SetTimer();
void EnableTimer();
void SingleButton(void fnc(), int btn);
void CompareTimeWithTimer();
// GLOBAL VARIABLES
int year[4] = { 2, 0, 0, 0 };   // yyyy
int date[4] = { 0, 1, 0, 1 };   // mm.dd
int time[6] = 0;                // hh:mm:ss
int timer[4] = 0;               // hh:mm
int time_var = 0;
unsigned char blink_var = 0;
unsigned char click_available = 0;
```

```
unsigned char btn_clicked = 0;
unsigned char timer_hit = 0;
int timer_released = 0;
// INTERRUPT
#pragma vector = TIMER1_OVF_vect
__interrupt void timer1()
{
	TCNT1 = 36735;
	time[5]++;	// increment seconds
	time_var++;
	if (timer_hit)
		timer_released++;
}
int main(void) {
	max7219_init();
	initDevices();
	__enable_interrupt();
	unsigned char ic = 0;
	// init ic
	for (ic = 0; ic < MAX7219_ICNUMBER; ic++)
	{
		max7219_shutdown(ic, 1); //power on
		max7219_test(ic, 0); //test mode off
		max7219_decode(ic, 0); //use led matrix
		max7219_intensity(ic, 8); //intensity
		max7219_scanlimit(ic, 15);
	}
	clearDisplay(0);
	clearDisplay(1);
	clearDisplay(2);
	clearDisplay(3);
	// Setting time and date by user
	SetYear();
	SetDate();
	SetTime();
	// MAIN LOOP
	for (;;)
```

```
{
	FormatDate();
	SingleButton(SetTimer, 0x10);
	SingleButton(EnableTimer, 0x08);
	// PRINTING
	if (time_var < 5)
	{
		// print time
		Draw(3, time[0]);
		Draw(2, time[1]);
		DrawWithColon(1, time[2]);
		Draw(0, time[3]);
	}
	if (time_var >= 5 && time_var < 7)
	{
		// print date
		Draw(3, date[0]);
		Draw(2, date[1]);
		DrawWithDot(1, date[2]);
		Draw(0, date[3]);
	}
	if (time_var >= 7 && time_var < 9)
	{
		// print year
		Draw(3, year[0]);
		Draw(2, year[1]);
		Draw(1, year[2]);
		Draw(0, year[3]);
	}
	// reset
	if (time_var >= 9)
		time_var = 0;
	if (timer_hit)
	{
		if (timer_enabled)
			for (ic = 0; ic < MAX7219_ICNUMBER; ic++)
				lightDisplay(ic);
```

```
                        if (timer_released > 10)
                                timer_hit = 0;
                }
                else
                        CompareTimeWithTimer();
        }
}
void FormatDate()
{
        // FORMATTING DATE
        // correct time
        if (time[3] > 9)
        {
                time[3] = 0;
                time[2] += 1;
        }
        if (time[2] > 5)
        {
                time[2] = 0;
                time[1] += 1;
        }
        if (time[1] > 9)
        {
                time[1] = 0;
                time[0] += 1;
        }
        if (time[0] == 2 && time[1] > 3)
        {
                time[1] = 0;
                time[0] = 0;
                date[3] += 1; // increment date
        }
        // correct date
        if (date[3] > 9)
        {
                date[3] = 0;
                date[2] += 1;
```

```
        }
        if (date[2] == 3 && date[3] > 1)
        {
                date[2] = 0;
                date[3] = 1;
                date[1] += 1;
        }
        if (date[1] > 9)
        {
                date[1] = 0;
                date[0] += 1;
        }
        if (date[0] == 1 && date[1] > 2)
        {
                date[1] = 1;
                date[0] = 0;
                year[3] += 1;
        }
        // correct year
        int year_cond_var = 0;
        for (year_cond_var = 3; year_cond_var >= 0; year_cond_var--)
        {
                if (year[year_cond_var] > 9)
                {
                        year[year_cond_var] = 0;
                        year[year_cond_var - 1] += 1;
                }
        }
}
int ButtonLogic(int temp_arr[], int cur_digit)
{
        if (!btn_clicked)
        {
                int tmp = ~PINB;
                // Increment
                if (tmp & 0x01)
                {
```

```
                temp_arr[cur_digit]++;
                btn_clicked = 1;
            }
            // Decrement
            else if (tmp & 0x02)
            {
                temp_arr[cur_digit]--;
                btn_clicked = 1;
            }
            // Proceed
            else if (tmp & 0x04)
            {
                cur_digit++;
                btn_clicked = 1;
            }
        }
        else
        {
            if (!(~PINB & 0x01) && !(~PINB & 0x02) && !(~PINB &
0x04))
            {
                int i = 0;
                for (i = 0; i < 1000; i++) // to make some delay
                    ;
                btn_clicked = 0;
            }
        }
        return cur_digit;
}
void SetTime()
{
        int temp_time[4] = 0;
        int cur_digit = 0;
        // SET Hour/Minute
        for (;;)
        {
            // print time
```

```
                Draw(3, temp_time[0]);
                Draw(2, temp_time[1]);
                Draw(1, temp_time[2]);
                // DrawWithColon(1, temp_time[2]);
                Draw(0, temp_time[3]);
                cur_digit = ButtonLogic(temp_time, cur_digit);      //
Handle buttons
                // Formatting
                if (temp_time[0] > 2)
                        temp_time[0] = 0;
                if (temp_time[0] == 2 && temp_time[1] > 3)
                        temp_time[1] = 0;
                if (temp_time[1] > 9)
                        temp_time[1] = 0;
                if (temp_time[2] > 5)
                        temp_time[2] = 0;
                if (temp_time[3] > 9)
                        temp_time[3] = 0;
                unsigned char formatVar = 0;
                for (formatVar = 0; formatVar < 4; formatVar++)
                        if (time[formatVar] < 0)
                                temp_time[formatVar] = 0;
                if (cur_digit > 3)
                        break;
        }
        unsigned char time_cpy = 0;
        for (time_cpy = 0; time_cpy < 6; time_cpy++)
                time[time_cpy] = temp_time[time_cpy];
}
void SetDate()
{
        int temp_date[4] = { 0, 1, 0, 1 };
        int cur_digit = 0;
        // SET Day/Month
        for (;;)
        {
                // print time
```

```
            Draw(3, temp_date[0]);
            Draw(2, temp_date[1]);
            Draw(1, temp_date[2]);
            // DrawWithDot(1, temp_date[2]);
            Draw(0, temp_date[3]);
            cur_digit = ButtonLogic(temp_date, cur_digit);      //
Handle buttons
            // Formatting
            if (temp_date[0] > 1)
                  temp_date[0] = 0;
            if (temp_date[0] == 1 && temp_date[1] > 2)
                  temp_date[1] = 0;
            if (temp_date[1] > 9)
                  temp_date[1] = 0;
            if (temp_date[2] > 3)
                  temp_date[2] = 0;
            if (temp_date[2] == 3 && temp_date[3] > 1)
                  temp_date[3] = 0;
            if (temp_date[3] > 9)
                  temp_date[3] = 0;
            if (temp_date[0] == 0 && temp_date[1] < 1)
                  temp_date[1] = 1;
            if (temp_date[2] == 0 && temp_date[3] < 1)
                  temp_date[3] = 1;
            unsigned char formatVar = 0;
            for (formatVar = 0; formatVar < 4; formatVar++)
                  if (temp_date[formatVar] < 0)
                        temp_date[formatVar] = 0;
            if (cur_digit > 3)
                  break;
      }
      unsigned char time_cpy = 0;
      for (time_cpy = 0; time_cpy < 4; time_cpy++)
            date[time_cpy] = temp_date[time_cpy];
}
void SetYear()
{
```

```
	int temp_year[4] = { 2, 0, 0, 0 };
	int cur_digit = 2;
	// SET Year
	for (;;)
	{
		// print time
		Draw(3, temp_year[0]);
		Draw(2, temp_year[1]);
		Draw(1, temp_year[2]);
		Draw(0, temp_year[3]);
		cur_digit = ButtonLogic(temp_year, cur_digit);	//
Handle buttons
		// Formatting
		if (temp_year[2] > 9)
			temp_year[2] = 0;
		if (temp_year[3] > 9)
			temp_year[3] = 0;
		unsigned char formatVar = 0;
		for (formatVar = 0; formatVar < 4; formatVar++)
			if (temp_year[formatVar] < 0)
				temp_year[formatVar] = 0;
		if (cur_digit > 3)
			break;
	}
	unsigned char time_cpy = 0;
	for (time_cpy = 0; time_cpy < 4; time_cpy++)
		year[time_cpy] = temp_year[time_cpy];
}
void SetTimer()
{
	int temp_timer[4] = 0;
	unsigned char time_cpy = 0;
	for (time_cpy = 0; time_cpy < 4; time_cpy++)
		temp_timer[time_cpy] = timer[time_cpy];
	int cur_digit = 0;
	// SET Hour/Minute
	for (;;)
```

```
	{
		// print time
		Draw(3, temp_timer[0]);
		Draw(2, temp_timer[1]);
		Draw(1, temp_timer[2]);
		// DrawWithColon(1, temp_timer[2]);
		Draw(0, temp_timer[3]);
		cur_digit = ButtonLogic(temp_timer, cur_digit);	//
Handle buttons
		// Formatting
		if (temp_timer[0] > 2)
			temp_timer[0] = 0;
		if (temp_timer[0] == 2 && temp_timer[1] > 3)
			temp_timer[1] = 0;
		if (temp_timer[1] > 9)
			temp_timer[1] = 0;
		if (temp_timer[2] > 5)
			temp_timer[2] = 0;
		if (temp_timer[3] > 9)
			temp_timer[3] = 0;
		unsigned char formatVar = 0;
		for (formatVar = 0; formatVar < 6; formatVar++)
			if (temp_timer[formatVar] < 0)
				temp_timer[formatVar] = 0;
		if (cur_digit > 3)
			break;
	}
	for (time_cpy = 0; time_cpy < 4; time_cpy++)
		timer[time_cpy] = temp_timer[time_cpy];
}
void SingleButton(void fnc(), int btn)
{
	if (!btn_clicked)
	{
		int tmp = ~PINB;
		if (tmp & btn)
		{
```

```
                fnc();
                btn_clicked = 1;
            }
        }
        else
        {
            if (!(~PINB & btn))
            {
                int i = 0;
                for (i = 0; i < 10000; i++) // to make some delay
                    ;
                btn_clicked = 0;
            }
        }
}
void EnableTimer()
{
        if (timer_enabled)
            timer_enabled = 0;
        else
            timer_enabled = 1;
}
void CompareTimeWithTimer()
{
        int time_comp_var = 0;
        for (time_comp_var = 0; time_comp_var < 4; time_comp_var++)
        {
            if (time[time_comp_var] != timer[time_comp_var])
            {
                timer_hit = 0;
                return;
            }
        }
        timer_hit = 1;
        timer_released = 1;
}
```