The Carpe Diem Wake-Up Light

Andrea Larsson & Sofia Pettersson

Handledare: Bertil Lindvall
Digitala Projekt, EITF11

Innehållsförteckning

Inledning
I kursen Digitala Projekt (EITF11) har vi fått i uppgift att utveckla en enklare digital prototyp. Detta innefattar bland annat planering, hårdvarukonstruering och implementering av mjukvara genom C-programmering. Gruppen har valt att göra en alarmklocka som istället för en alarmsignal har en ljusfunktion, inspirerad av Philips Wake-Up Light (Philips, 2014).

Kravspecifikation
I det här projektet är målet att tillverka en digital alarmklocka med en ljusfunktion. Fokus på alarmklockan är ljusfunktionen, förutom denna hålls klockan så enkel som möjligt.

Funktionella krav
- Användaren skall kunna se tiden på displayen
- Användaren skall vid knapptryck kunna se vad alarmets inställda tid är
- Användaren skall kunna ställa in tid med hjälp av knappar på alarmklockan
- Användaren skall kunna ställa in alarmtid med hjälp av knappar på alarmklockan
- Användaren skall kunna aktivera/avaktivera alarmet med hjälp av en knapp
- Användaren skall kunna se på displayen om alarmet är aktiverat
- När tiden har nått den inställda alarmtiden skall lampor succesivt tändas

I mån av tid:

- När tiden har nått den inställda alarmtiden ska en rolig text visas på displayen fram tills alarmet stängs av
- Användaren skall kunna välja mellan flera olika alarmsignaler med hjälp av en knapp
- När tiden har nått den inställda alarmtiden skall en högtalare ska låta

Icke-funktionella krav
- Systemet ska kunna hålla koll på en tid i taget
- Systemet ska kunna hålla koll på en alarmtid i taget
- Systemet är begränsat till det närmaste dygnet, dvs. användaren kan inte ställa in en alarmtid mer än 24 timmar framåt i tiden
- Lampornas ljusstyrka och tändningsprocess är inte justerbar

Hårdvara
I genomförandet av detta projekt används diverse hårdvarukomponenter som listas nedan.

Processor
En processor av modellen Atmega16 används till den här prototypen. Den har totalt 40 pinnar, varav 32 st är I/O-pinnar och övriga pinnar är kopplade till ett JTAG interface. Pinnarna är uppdelade i A, B, C respektive D-portar. Nio stycken lysdioder är kopplade till A-porten och en display är kopplad till både B- och C-porten. Till C-porten är även en JTAG kopplad som fungerar som ett interface mellan prototypen och dess mjukvara. Till sist är fyra stycken knappar och en

grind kopplade till D-porten. Prototypen använder också en intern timer/counter som har klockfrekvensen 8 MHz.

Display

Eftersom en önskad funktion hos alarmklockan är att visa både aktuell tid och inställd alarmtid på displayen har en tvåradig (16x2) Sharp Dot Matrix LCD använts. Displayen får information om vad som ska skrivas ut genom att processorn skickar data.

Potentiometer

En potentiometer används för att justera displayens kontrast.

Lysdioder

För att demonstrera funktionen av ett "wake-up light" används 9 stycken lysdioder. Dessa tänds successivt i rader om tre i samband med att alarmet har nått din inställda tid. Lamporna är parallellkopplade och för att undvika kortslutning används resistorer (se Resistorer).

Knappar

För att göra det möjligt för användaren att ställa in tiden, alarmtiden och aktivera/avaktivera alarmet används fyra knappar. Knapparna fungerar på samma sätt som strömbrytare, vilket innebär att så fort en knapp är nedtryckt skickas en spänning i form av en etta till en av processorns portar. Vad som händer när respektive knapp är nedtryckt är programmerat i mjukvaran. Den översta knappen till vänster aktiverar eller avaktiverar alarmet och den översta till höger tillåter användaren att ändra alarmtiden. De två nedersta knapparna används för att antingen ställa in tiden eller alarmtiden beroende på om den översta knappen till vänster är nedtryckt eller inte.

Kondensatorer

Kondensatorer är kopplade till knapparna för att undvika brus eller störningar vid knapptryckningar.

Grind

För att processorn ska reagera vid ett knapptryck, används en grind. Syftet med grinden är att sammankoppla knapparnas signaler till en signal som går till processorn. Ena sidan av grinden är kopplad till knapparna och andra sidan är kopplad till "INT0". Detta innebär att så fort en knapp är nedtryckt genereras ett externt avbrott. Det är först när avbrott har genererats som processorn undersöker vilken knapp signalen kom ifrån.

Resistorer

Flera olika resistorer har använts i prototypen. Till varje knapp är ett motstånd på 100 kΩanslutet. Anledningen till att jordade motstånd är anslutna till knapparna är för att processorn alltid ska motta antingen en 1:a eller 0:a som insignal från knapparna. Till varje lysdiod är ett motstånd på 500 Ωanslutet och syftet med dessa motstånd är att undvika att lamporna kortslut.

Mjukvara

Mjukvaran till alarmklockan är programmerad i C i AVR Studio 6. Eftersom att huvudprogrammet är uppbyggt av en while-loop, kan mjukvaran ständigt kontrollera om någon av hårdvarans enheter har genererat ett avbrott. Ett avbrott innebär antingen att en av knapparna ger utslag eller att mikroprocessorn gör ett tidsavbrott. När ett knappavbrott görs, går programmet in i en metod som finner vilken specifik knapp som gjort utslag. Beroende på vilken knapp som gett utslag, skickas information vidare till en annan metod som skriver ut resultatet på displayen. Till skillnad från ett knappavbrott, genereras tidsavbrott kontinuerligt. Ett tidsavbrott används för att tiden ska räknas upp från att klockan aktiveras.

Arbetsprocessen

Arbetet inleddes enligt instruktion genom att en kravspecifikation för prototypen skrevs. Nästa steg i arbetsprocessen var att gruppen skapade ett kopplingsschema i programmet PowerLogic, samt en grovt skissad pseudokod för den framtida mjukvaran. Gruppen blev tilldelad lämpliga elektroniska komponenter och utefter kopplingsschema samt med hjälp av datablad för de olika komponenterna konstruerades sedan prototypen. Efter testning av prototypens hårdvara började mjukvaran att utformas. Testning av mjukvara har skett kontinuerligt under arbetets gång.

Under arbetets gång har det dykt upp diverse problem. Många problem gick att lösa genom att lägga till ett antal komponenter som inte fanns med på det utsprungliga kopplingsschemat. Komponenter som tillkom i efterhand är:

- Potentiometern, för att kunna reglera displayens kontrast.
- Kondensatorer, för att reglera knapparnas insignaler.
- Grind, för att hantera knapparnas insignaler.

Andra problem uppstod i slutet av arbetsprocessen. När prototypen användes har till exempel oförutsedda förändringar av tid och alarmtid skett och displayen har även stängt av sig vid ett antal tillfällen. Efter att gruppen sett över hur avbrotten gjordes i mjukvaran, kunde felen upptäckas och åtgärdas.

Resultat

Projektet har resulterat i en prototyp som fungerar mycket väl. Alla planerade funktioner går att använda som avsett och gruppen har även haft tid att lägga till en extrafunktion som i kravspecifikationen står under rubriken “I mån av tid”. Denna funktion innebär att när alarmet utlöses visas en rolig text på displayen.

Slutsats & Diskussion

Projektet har inte bara resulterat i en välfungerande prototyp utan även en helt ny kunskapsbas. Genom att utföra C-programmering, hårdvarukonstruktion och databladstolkningar har vi fått en inblick och förståelse i digital teknik som vi annars inte skulle fått.

Om kursen skulle löpt under en längre tid skulle det vara möjligt att utveckla alarmklockan ytterligare. Först och främst skulle fler krav från kravspecifikationen under rubriken “I mån av tid” utvecklas. Detta skulle framförallt innebära att addera en högtalare för att framkalla olika alarmsignaler. Ett annat förslag skulle vara att användaren hade möjlighet att spara olika alarmtider på klockan. Dessa funktioner skulle innebära stora förändringar i både mjuk - och hårdvara och därmed lämnar vi det till framtida kursdeltagare att utveckla.

Källförteckning

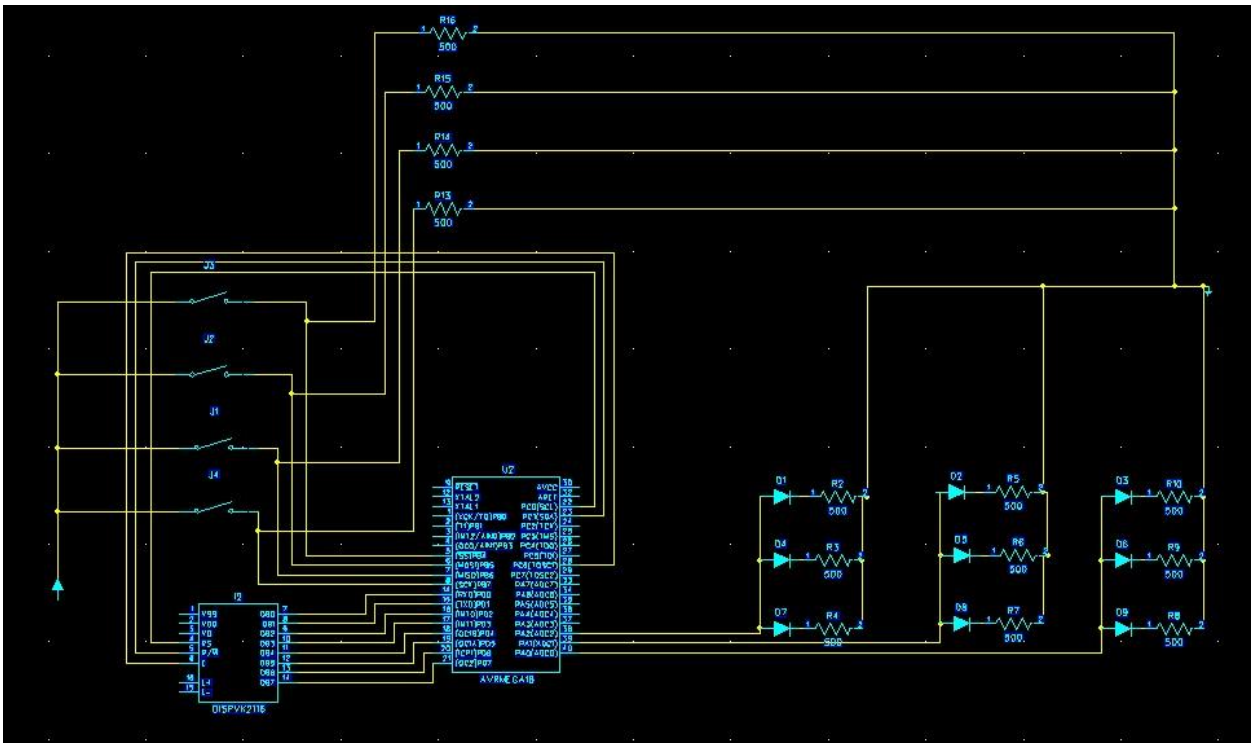
Philips Wake-up Light <http://www.usa.philips.com/c-p/HF3470_60/wake-up-light>
Hämtad: 2014-04-20

Databladet AVR-ATmega16 High-performance AVR 8-bit Microcontroller
<http://www.eit. lth.se/fileadmin/eit/courses/edi021/datablad/Processors/ATmega16.pdf>
Hämtad:2014-03-25.

Databladet Sharp Dot-Matrix LCD Units Alfanumerisk teckendisplay
<http://www.eit.lth.se/fileadmin/ eit/courses/edi021/datablad/Display/LCD.pdf>
Hämtad: 2014-03-24.

Bilagor

Kopplingsschema



Källkod

```
//variables
int alarm_activated = 0; // = 1 if activated
int set_button = 0; // = 1 if set-button is pressed
int alarm_triggered = 0;
int hour = 0;
int minute = 0;
int second = 0;
int alarmHour = 0;
int alarmMinute = 0;
int count = 0;  // to count number of interrupts
int display_time = 1;
int buttons = 1;


void setup() {
	DDRA = 0x07;
	DDRB = 0xff;
	DDRC = 0xc2;
	DDRD = 0x00;
	disp_init();
	show_time(hour, minute, second);

	//Timer-interrupt
	TCCR0|=(1<<CS02)|(1<<CS00); //Prescaler = FCPU/1024
	TIMSK|=(1<<TOIE0); // Enable Overflow Interrupt Enable (samma som 0x01)
	TCNT0 = 0x00; // Initialize Counter
```

```
	//External interrupt
	GICR = 1<<INT0;
	MCUCR = 1<<ISC01 | 1<<ISC00;

	sei(); // enable ALL interrupts
}

//Display:
void disp_clear() {

	write_cmd(0x01); //clear display
	_delay_ms(5);
	write_cmd(0x38); //function set
}

void disp_init() {
	disp_clear();
	_delay_ms(5);
	write_cmd(0x0c); //display ON
	_delay_ms(5);
	write_cmd(0x06); //entry mode set
	_delay_ms(5);
}

void disp_nextRow() {
	write_cmd(0xC0); //Change row
	_delay_ms(5);
}

void write_cmd(char value) { //write cmd for display setup

	PORTB = value;
	_delay_ms(5);
	set_pin('C', PC1, 0); // RS
	set_pin('C', PC6, 0); // RW
	_delay_ms(5);
	set_pin('C', PC7, 1); // E
	_delay_ms(5);
	set_pin('C', PC7,  0); // E (sends to disp)
	_delay_ms(5);
}

void disp_writeCh(char val){

	PORTB=val;
	set_pin('C', PC6,0); //RW
	_delay_ms(5);
	set_pin('C', PC1, 1);	//RS
	_delay_ms(5);
	set_pin('C',PD7, 1); //Enable
	_delay_ms(5);
	set_pin('C',PD7, 0); //Enable
	_delay_ms(5);
}

void disp_writeNum(int number){
	int num1 = number/10;
```

```
	disp_nbr(num1);
	int num2 = number%10;
	disp_nbr(num2);
}

void disp_nbr(int number) {
	if (number == 0) {
		disp_writeCh('0');
	}
	if (number == 1) {
		disp_writeCh('1');
	}
	if (number == 2) {
		disp_writeCh('2');
	}
	if (number == 3) {
		disp_writeCh('3');
	}
	if (number == 4) {
		disp_writeCh('4');
	}
	if (number == 5) {
		disp_writeCh('5');
	}
	if (number == 6) {
		disp_writeCh('6');
	}
	if (number == 7) {
		disp_writeCh('7');
	}
	if (number == 8) {
		disp_writeCh('8');
	}
	if (number == 9) {
		disp_writeCh('9');
	}
}

void set_time(int hour, int minute, int second) {

	disp_writeCh('S');
	disp_writeCh('e');
	disp_writeCh('t');
	disp_writeCh(' ');
	disp_writeCh('t');
	disp_writeCh('i');
	disp_writeCh('m');
	disp_writeCh('e');
	disp_writeCh(':');
	disp_writeCh(' ');
}

void show_time()
{
	disp_clear();
	disp_writeCh('T');
	disp_writeCh('i');
	disp_writeCh('m');
```

```
		disp_writeCh('e');
		disp_writeCh(':');
		disp_writeCh(' ');
		disp_writeNum(hour);
		disp_writeCh(':');
		disp_writeNum(minute);
		disp_writeCh(':');
		disp_writeNum(second);
		disp_nextRow();
		if(set_button == 1) {
			set_alarm();
		} else if(alarm_activated == 1 && set_button == 0) {
			alarm_on();
		}

}

void set_alarm() {

		disp_writeCh('A');
		disp_writeCh('l');
		disp_writeCh('a');
		disp_writeCh('r');
		disp_writeCh('m');
		disp_writeCh(':');
		disp_writeCh(' ');
		disp_writeNum(alarmHour);
		disp_writeCh(':');
		disp_writeNum(alarmMinute);
}

void alarm_on() {
		disp_writeCh('A');
		disp_writeCh('l');
		disp_writeCh('a');
		disp_writeCh('r');
		disp_writeCh('m');
		disp_writeCh(' ');
		disp_writeCh('o');
		disp_writeCh('n');
}

void check_alarm() {
		if (alarm_activated == 1 && hour == alarmHour && minute == alarmMinute) {
			led_on();
			disp_clear();

			disp_writeCh('*');
			disp_writeCh('*');
			disp_writeCh(' ');
			disp_writeCh('C');
			disp_writeCh('a');
			disp_writeCh('r');
			disp_writeCh('p');
			disp_writeCh('e');
			disp_writeCh(' ');
			disp_writeCh('D');
			disp_writeCh('i');
```

```
                disp_writeCh('e');
                disp_writeCh('m');
                disp_writeCh(' ');
                disp_writeCh('*');
                disp_writeCh('*');

                alarm_triggered = 1;
        }
}

//LED

void led_on() // LED on
{
        set_pin('A', PA0, 1);
        if (second == 3){
                set_pin('A', PA1, 1);
        }
        if (second == 6) {
                set_pin('A', PA2, 1);
        }
}

void led_off() //LED off
{
        set_pin('A', PA0, 0);
        set_pin('A', PA1, 0);
        set_pin('A', PA2, 0);
}

void set_pin(char port, char pin, char state){ //sets value on specific pin

        char set = _BV(pin);
        if('A' == port) {
                set &= PORTA;
                if(set && !state) { //ändrar från 1 till 0
                        PORTA ^= set;
                } else if (set == 0x00 && state) { //ändrar från 0 till 1
                        set = _BV(pin);
                        PORTA ^= set;
                }
        } else if ('B' == port) {
                set &= PORTB;
                if(set && !state) { //ändrar från 1 till 0
                        PORTB ^= set;
                } else if (set == 0x00 && state) { //ändrar från 0 till 1
                        set = _BV(pin);
                        PORTB ^= set;
                }
        } else if ('C' == port) {
                set &= PORTC;
                if(set && !state) { //ändrar från 1 till 0
                        PORTC ^= set;
                } else if (set == 0x00 && state) { //ändrar från 0 till 1
                        set = _BV(pin);
                        PORTC ^= set;
                }
        } else if ('D' == port) {
```

```
                set &= PORTD;
                if(set && !state) { //ändrar från 1 till 0
                        PORTD ^= set;
                } else if (set == 0x00 && state) { //ändrar från 0 till 1
                        set = _BV(pin);
                        PORTD ^= set;
                }
        }
}

//Interrupts

ISR(TIMER0_OVF_vect) { //körs vid varje interrupt
        count++;
        TIFR = 0x01;
        if( count == 32 ) {
                second++;
                if (second > 59) {
                        second = 0;
                        minute++;
                        if (minute > 23) {
                                minute = 0;
                                hour++;
                                if (hour > 23) {
                                        hour = 0;
                                }
                        }
                }
                display_time = 1;
                count = 0;
        }
}

ISR(INT0_vect) {
        cli();
        GIFR = 1<<INTF0;
        _delay_ms(500);
        check_buttons();
        _delay_ms(150);
        sei();
}

//Buttons
void check_buttons() {
        _delay_ms(150);
        char btn1 = _BV(PD0);
        char btn2 = _BV(PD1);
        char btn3 = _BV(PD4);
        char btn4 = _BV(PD5);
        btn1 &= PIND;
        btn2 &= PIND;
        btn3 &= PIND;
        btn4 &= PIND;

        if(btn1 == 0x01) { // alarm activated
                if(alarm_activated == 0) { // alarm activated
                        alarm_activated = 1;
                } else if (alarm_activated == 1) { // alarm deactivated
```

```
                alarm_activated = 0;
                if(alarm_triggered == 1) {
                        led_off();
                        alarm_triggered = 0;
                }
        }
    }
    if(btn2 == 0x02) { //set alarm
        if(set_button == 0) { // alarm activated
                set_button = 1;
        } else if (set_button == 1) {
                set_button = 0;
        }
    }
    if(btn3 == 0x10 && set_button == 0) { // set time
        hour++;
                if (hour > 23) {
                        hour = 0;
                }
        }
    if(btn4 == 0x20 && set_button == 0) { // set time
        minute++;
        if (minute > 59) {
                minute = 0;
                hour++;
                if (hour > 23) {
                        hour = 0;
                }
        }
    }
    if(btn3 == 0x10 && set_button == 1) { //set alarm
        alarmHour++;
        if (alarmHour > 23) {
                alarmHour = 0;
        }
    }
    if(btn4 == 0x20 && set_button == 1) { // set alarm
        alarmMinute++;
        if (alarmMinute > 59) {
                alarmMinute = 0;
                alarmHour++;
                if (alarmHour > 23) {
                        alarmHour = 0;
                }
        }
    }
    _delay_ms(150);
}
```

```
//Huvudprogram

void main(void)
{
	setup();
	while(1){
		if (display_time == 1){
			show_time();
			check_alarm();
			display_time = 0;
		}
	}
	return;
}
```