```
#include <avr/io.h>
#include "lcd.h"
#include <avr/interrupt.h>
#include <util/delay.h>

#define F_CPU 1000000UL

char buttonPressed;
int hour1, hour2, min1, min2, s1, s2, cursorPlace, motorPlace, aPressed;
#define DD_MOSI    PINB3
#define DD_SCK     PINB5
#define DDR_SPI    PORTB
#define CSN_LOW() PORTB &= ~ (1<<PB4);
#define CSN_HIGH() PORTB |= (1<<PB4);

ISR(INT0_vect) { //Interupt när knapp på tangentbordet trycks ner
    DDRA = 0x00;
    _delay_ms(10);
    PORTD &= ~_BV(PD3);         //Skickar Output Enable när INT0 triggas
    _delay_ms(10);
    buttonPressed = PINA & 0x0F;
    if(aPressed == 1){
        if(buttonPressed >= 0 && buttonPressed <= 9){
            set_time(buttonPressed);
        }
        if(buttonPressed == 10){
            confirm_time();
        }
        if(buttonPressed == 11){
            motor_turn(1);
        }
        if(buttonPressed == 12){
            cursor_left();
        }
        if(buttonPressed == 13){
            cursor_right();
        }
        if(buttonPressed == 14){
            reset_program();
        }
        if(buttonPressed == 15){
            motor_turn(2);
        }
    } else if(aPressed == 0){
        if(buttonPressed >= 0 && buttonPressed <= 9){
            return;
        }
        if(buttonPressed == 10){
            _delay_ms(10);
            set_alarm();
        }
        if(buttonPressed == 11){
            motor_turn(1);
        }
        if(buttonPressed == 12){
            return;
        }
        if(buttonPressed == 13){
            return;
        }
        if(buttonPressed == 14){
            reset_program();
        }
        if(buttonPressed == 15){
            motor_turn(2);
        }
    } else {
        if(buttonPressed >= 0 && buttonPressed <= 9){
```

```
            return;
        }
        if(buttonPressed == 10){
            return;
        }
        if(buttonPressed == 11){
            motor_turn(1);
        }
        if(buttonPressed == 12){
            return;
        }
        if(buttonPressed == 13){
            return;
        }
        if(buttonPressed == 14){
            reset_program();
        }
        if(buttonPressed == 15){
            motor_turn(2);
        }
    }
}

void set_alarm() //Visar "sätt alarm"-menyn
{
    disp_alarm();
    hour1 = 0;
    hour2 = 0;
    min1 = 0;
    min2 = 0;
    s1 = 0;
    s2 = 0;
    cursorPlace = 0;
    disp_time(hour1, hour2, min1, min2, s1, s2);
    disp_cursorFirst(); //Sätter markören längs ned till vänster
    aPressed++;
    return;
}

void motor_turn(val) //Vrider motorn, 1 motsols, 2 medsols.
{
    disp_motorTurning(val);
    if(val == 1 ){
        SPI_MasterTransmit(0x01);
    } else if(val == 2){
        SPI_MasterTransmit(0x02);
    }
}

void cursor_left() //Flyttar markören 1 steg åt vänster
{
    if(cursorPlace == 0){
        return;
    }
    cursorPlace--;
    _delay_ms(10);
    disp_cursorShift(1); //move cursor left
    if(cursorPlace == 2 || cursorPlace == 5){
        cursor_left();
    }
    return;
}

void cursor_right() //Flyttar markören 1 steg åt höger
{
    if(cursorPlace == 7){
        return;
    }
```

```
    cursorPlace++;
    _delay_ms(10);
    disp_cursorShift(2); //move cursor right
    if(cursorPlace == 2 || cursorPlace == 5){
        cursor_right();
    }
    return;
}

void reset_program() //Startar om programmet, nollställer timer osv
{
    hour1 = 0;
    hour2 = 0;
    min1 = 0;
    min2 = 0;
    s1 = 0;
    s2 = 0;
    cursorPlace = 0;
    aPressed = 0;
    TCNT1 = 0;
    disp_Welcome();
}

void show() //Visar inskrivna siffror
{
    disp_alarm();
    disp_time(hour1, hour2, min1, min2, s1, s2);
    return;
}

void confirm_time() //Startar nedräkning
{
    disp_alarmSet();
    disp_time(hour1, hour2, min1, min2, s1, s2);
    aPressed++;
    startCountdown();
    return;
}

void set_time(int val) //Sätter vald tid på vald plats
{
    if(cursorPlace == 0){
        hour1 = val;
        show();
    } else if(cursorPlace == 1){
        hour2 = val;
        show();
    } else if(cursorPlace == 3){
        if(val >= 6){
            min1 = 6;
            min2 = 0;
        } else {
            min1 = val;
        }
        show();
    } else if(cursorPlace == 4){
        min2 = val;
        if(min1 == 6){
            min2 = 0;
        }
        show();
    } else if(cursorPlace == 6){
        if(val >= 6){
            s1 = 6;
            s2 = 0;
        } else {
            s1 = val;
        }
```

```
        show();
    } else if(cursorPlace == 7){
        s2 = val;
        if(s1 == 6){
            s2 = 0;
        }
        show();
    }
    cursorPlace = 0;
    return;
}

void startCountdown()
{
    sei();
    TCCR1B |= (1 << CS10); // Set up timer
    int i = 0;
    for (;;){
        if (TCNT1 >= 49999){
            TCNT1 = 0; // Reset timer value
            i++;
            if(i > 160){
                s2--;
                if(s2 == -1){
                    s2 = 9;
                    s1--;
                    if(s1 == -1){
                        s1 = 5;
                        min2--;
                        if(min2 == -1){
                            min2 = 9;
                            min1--;
                            if(min1 == -1){
                                min1 = 5;
                                hour2--;
                                if(hour2 == -1){
                                    hour2 = 9;
                                    hour1--;
                                    if(hour1 == -1){
                                        break;
                                    }
                                }
                            }
                        }
                    }
                }
                disp_alarmSet();
                disp_time(hour1, hour2, min1, min2, s1, s2);
                i = 0;
            }
        }
    }
    disp_timeUp();
    SPI_MasterTransmit(0x03);
}


void SPI_MasterTransmit(char cDATA)
{
    CSN_LOW(); //Slaveselect low
    SPDR = cDATA; //Skriver till buffert för att starta SPI
    while(!(SPSR & (1<<SPIF))); //Fortsätter till lyckad sändning
    CSN_HIGH();     //Slaveselect high
}

void main(void)

{
```

```
    _delay_ms(50);
    hour1 = 0;
    hour2 = 0;
    min1 = 0;
    min2 = 0;
    s1 = 0;
    s2 = 0;
    cursorPlace = 0;
    aPressed = 0;

    DDRA = 0xFF;
    DDRD = 0xFB;
    GICR = 1<<INT0;                 // Enable INT0
    MCUCR = 1<<ISC01 | 1<<ISC00;    // Trigger INT0 on all activity
    GIFR = 1<<INTF0;                // External Interupt Flag 0
    SREG = 0x80;                    // Global Interupt Set

    DDR_SPI = (1<<DD_MOSI)|(1<<DD_SCK);
    DDRB = (1<<DDB5)|(0<<DDB6)|(1<<DDB7)|(1<<DDB4);      //Config SPI-ports
    DDRB &=~ (1<<PB6);
    SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0)|(1<<SPR1);  //Enable SPI and set to Master, clo
ckrate to fck/16

    _delay_ms(50);
    disp_init();
    disp_home();
    disp_Welcome();

    CSN_HIGH();

    sei();                          //start interupts

    while(1)
    {
    }

}
```

```
#include <avr/io.h>
#include "lcd.h"
#include <avr/interrupt.h>
#define F_CPU 1000000UL
#include <util/delay.h>

char t1, t2, t3, t4, t5, t6;
int t;
char d;

void write_cmd(char val) {
    _delay_ms(5);
    PORTA=val;
    _delay_ms(5);
    PORTD=0xCB;
    PORTD=0x8B;
}

void disp_writeCh(char val) {
    _delay_ms(5);
    PORTA=val;
    PORTD=0xEB;
    PORTD=0x8B;
}

void disp_clear() {
    _delay_ms(10);
    write_cmd(0x01); //clear display
    _delay_ms(10);
    write_cmd(0x38);  //function set
    _delay_ms(10);
}


void disp_init() {

    disp_clear();
    _delay_ms(5);
    write_cmd(0x0F); //display on
    _delay_ms(5);
    write_cmd(0x06); //entry mode set;
}


void disp_home(){
    write_cmd(0x03);
}

void disp_Welcome()
{
    DDRA = 0xFF;
    disp_clear();
    _delay_ms(10);
    disp_writeCh('W');
    disp_writeCh('e');
    disp_writeCh('l');
    disp_writeCh('c');
    disp_writeCh('o');
    disp_writeCh('m');
    disp_writeCh('e');
    disp_writeCh('!');
    _delay_ms(10);
    write_cmd(0xC0); //Byter rad
    _delay_ms(10);
    disp_writeCh('A');
    disp_writeCh(':');
    disp_writeCh('S');
    disp_writeCh('A');
```

```
    disp_writeCh(' ');
    disp_writeCh('B');
    disp_writeCh(':');
    disp_writeCh('C');
    disp_writeCh('C');
    disp_writeCh('W');
    disp_writeCh(' ');
    disp_writeCh('F');
    disp_writeCh(':');
    disp_writeCh('C');
    disp_writeCh('W');
}

void disp_alarm()
{
    DDRA = 0xFF;
    disp_clear();
    _delay_ms(10);
    disp_writeCh('S');
    disp_writeCh('e');
    disp_writeCh('t');
    disp_writeCh(' ');
    disp_writeCh('a');
    disp_writeCh('l');
    disp_writeCh('a');
    disp_writeCh('r');
    disp_writeCh('m');
    disp_writeCh(':');
    write_cmd(0xC0); //Byter rad
    _delay_ms(10);
}

void disp_time(int hour1, int hour2, int min1, int min2, int s1, int s2)
{
    DDRA = 0xFF;
    t1 = 0x30 + hour1;
    t2 = 0x30 + hour2;
    t3 = 0x30 + min1;
    t4 = 0x30 + min2;
    t5 = 0x30 + s1;
    t6 = 0x30 + s2;
    _delay_ms(10);
    disp_writeCh(t1);
    disp_writeCh(t2);
    disp_writeCh(':');
    disp_writeCh(t3);
    disp_writeCh(t4);
    disp_writeCh(':');
    disp_writeCh(t5);
    disp_writeCh(t6);
    disp_cursorFirst();
}

void disp_alarmSet()
{
    DDRA = 0xFF;
    disp_clear();
    _delay_ms(10);
    disp_writeCh('T');
    disp_writeCh('i');
    disp_writeCh('m');
    disp_writeCh('e');
    disp_writeCh('r');
    disp_writeCh(' ');
    disp_writeCh('s');
    disp_writeCh('e');
    disp_writeCh('t');
    disp_writeCh('!');
```

```
    write_cmd(0xC0); //Byter rad
    _delay_ms(10);
}

void disp_cursorFirst() //Sätter markören på första tecknet på undre raden
{
    DDRA = 0xFF;
    _delay_ms(10);
    write_cmd(0x02); //Cursor home
    _delay_ms(10);
    write_cmd(0xC0); //Byter rad
}

void disp_cursorShift(int i) //Flyttar markören, i = 1 vänster, i = 2 höger
{
    DDRA = 0xFF;
    if(i == 1){
        _delay_ms(10);
        write_cmd(0x10); //Markör vänster
    } else if(i == 2){
        _delay_ms(10);
        write_cmd(0x14); //Markör höger
    }
}

void disp_motorTurning(int i) //Skriver ut att motorn vrids
{
    DDRA = 0xFF;
    disp_clear();
    disp_writeCh('M');
    disp_writeCh('o');
    disp_writeCh('t');
    disp_writeCh('o');
    disp_writeCh('r');
    disp_writeCh('n');
    disp_writeCh(' ');
    disp_writeCh('v');
    disp_writeCh('r');
    disp_writeCh('i');
    disp_writeCh('d');
    disp_writeCh('s');
    disp_writeCh(':');
    disp_cursorFirst();
    if(i == 1){
        disp_writeCh('M');
        disp_writeCh('o');
        disp_writeCh('t');
        disp_writeCh('s');
        disp_writeCh('o');
        disp_writeCh('l');
        disp_writeCh('s');
    } else if(i == 2){
        disp_writeCh('M');
        disp_writeCh('e');
        disp_writeCh('d');
        disp_writeCh('s');
        disp_writeCh('o');
        disp_writeCh('l');
        disp_writeCh('s');
    }
}

void disp_timeUp()
{
    DDRA = 0xFF;
    disp_clear();
    _delay_ms(10);
    disp_writeCh('T');
```

```
    disp_writeCh('i');
    disp_writeCh('m');
    disp_writeCh('e');
    disp_writeCh(' ');
    disp_writeCh('i');
    disp_writeCh('s');
    disp_writeCh(' ');
    disp_writeCh('u');
    disp_writeCh('p');
    disp_writeCh('!');
    write_cmd(0xC0); //Byter rad
    _delay_ms(10);
    disp_writeCh('D');
    disp_writeCh('a');
    disp_writeCh('g');
    disp_writeCh('s');
    disp_writeCh(' ');
    disp_writeCh('a');
    disp_writeCh('t');
    disp_writeCh('t');
    disp_writeCh(' ');
    disp_writeCh('v');
    disp_writeCh('a');
    disp_writeCh('k');
    disp_writeCh('n');
    disp_writeCh('a');
    disp_writeCh('!');
    _delay_ms(10);
}
```