```c
#include <avr/io.h>
#include <avr/interrupt.h>
#include <string.h> //importerar string och funktionerna som strlen
#include <util/delay.h>

int h1 = 0; // h1h2:m1m2:s1s2 den aktiva tiden för klockan
int h2 = 0;
int m1 = 0;
int m2 = 0;
int s1 = 0;
int s2 = 0;
int s2Old = 0;

int setH1 = 0; //tid för att sätta klockan
int setH2 = 0;
int setM1 = 0;
int setM2 = 0;

int ah1 = 0; // ah1ah2:am1am2:as1as2 tiden för alarmet
int ah2 = 0;
int am1 = 0;
int am2 = 0;

int alarmOnOff = 0; // har koll på om alarmet är på eller av. = 1 på, = 0 av.
int alarmRinging = 0; // 0 - alarmet ringer ingte, 1 - alarmet ringer
int settingTime = 0; // 0 - inget, 1 - sätter tid
int settingAlarm = 0;  // 0 - inget, 1 - sätter alarm
int settingButton = 0; //0 - loppar inte över checkButtons, 1- loppar över checkButtons (för att för
   settings vyerna att stanna.


/*Får ut de första 5 siffrorna är för PORTB (som inte påverkar displayen).
*Viktigt då vi inte vill att dessa ska ändras då vi ändrar på PB0-PB2 (de tre sista).
*Det kallas bit masking.
*/
char bitMaskPortB(){
    //sparar vad den nuvarande port B är.
    char currentPortB = PORTB;

    //Använder en AND operation för att få ut vilka av de första fem pin som är high och low.
    char bFirstFive = currentPortB & 0b11111000;

    //Returnerar abcde000
    return bFirstFive;
}

/*Väntar på att displayens busy flag är av. Alltså om BF = 0 (tredje från vänster)*/
void busy(){
    //Sätter alla pins på port A till input (alltså att de kommer att ta emot info till displayen)
    DDRA = 0b00000000;

    //Skapa en bit mask av port b
    char bFirstFive = bitMaskPortB();

    //Sätter port b att man vill läsa busy flag, sätter E till low
    PORTB = 0b00000010| bFirstFive;

    //Sätter på funktionen för att läsa data från RAM (DDRAM) och E till high
    PORTB = 0b00000110| bFirstFive;

    //Tar emot infromation från displayen
    char info = PINA;

    //Jämför busy flag om BF = 1 så kommer checkFlag = 0b10000000 annars 0b00000000.
```

```
    char checkFlag = info & 0b10000000;

    //Upprepar de tidigare stegen i en lopp tills BF = 0
    while(checkFlag == 0b10000000){
        PORTB = 0b00000010| bFirstFive;
        PORTB = 0b00000110| bFirstFive;
        info = PINA;
        checkFlag = info & 0b10000000;
    }
    //Sätter alla pins i port A till output.
    DDRA = 0b11111111;
}

/*Metod för att skriva ett kommando till displayen*/
void commandToDisplay(char commandB, char commandA){
    //Väntar tills BF = 0
    busy();

    //Sätter E i high (PB2 = 1) + RS & R/W
    char bFirstFive = bitMaskPortB();

    PORTB = commandB | bFirstFive;

    //Sätter DB7-DB0
    PORTA = commandA;

    //Sätter E i low
    PORTB = 0b00000000 | bFirstFive;

    //Sätter port A till 0 (ej viktigt)
    PORTA = 0b00000000;

}

/*Sätt function set*/
void functionSet(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00111000: Sätter function set till 8-bit, 2 rader, 5*8, de sista två siffrorna spelar ingen roll om 1 eller 0.
    commandToDisplay(0b00000100, 0b00111000);
}

/*Sätt på displayen*/
void displayOn(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00001100: Sätt på displayen
    commandToDisplay(0b00000100, 0b00001100);
}

/*Metod för att clear display.*/
void clearDisplay(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00000001: Clear display
    commandToDisplay(0b00000100, 0b00000001);
}

/*Metoden för att skriva en char.*/
void writeChar(char character){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 1,)
    commandToDisplay(0b00000101, character);
}

/*Skriver ut siffror då writeChar metoden inte kan göra det med direkta siffor.*/
void writeNumber(int nbr){
```

```
    switch (nbr)
    {
        case 1:
        writeChar(0x31);
        break;

        case 2:
        writeChar(0x32);
        break;

        case 3:
        writeChar(0x33);
        break;

        case 4:
        writeChar(0x34);
        break;

        case 5:
        writeChar(0x35);
        break;

        case 6:
        writeChar(0x36);
        break;

        case 7:
        writeChar(0x37);
        break;

        case 8:
        writeChar(0x38);
        break;

        case 9:
        writeChar(0x39);
        break;

        case 0:
        writeChar(0x30);
        break;
    }


}

/*Metoden för att skriva en string.*/
void writeString(char string[]){
    //loop över string (i c är det en vektor)
    int length = strlen(string);
    for(int c = 0; c < length; c++){
        //Skriver ut på displayen varje tecken från string
        writeChar(string[c]);
    }
}

/*Sätt på markören*/
void cursorOn(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00001100: Sätt på markören
    commandToDisplay(0b00000100, 0b00001111);
}

/*Stäng av markören */
```

```
void cursorOff(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00001100: Stänger av markören
    commandToDisplay(0b00000100, 0b00001100);
}

/*Flytta markören till bestämd plats.*/
void moveCursor(char location){
    if(location<0b01111111){
        //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
        //avnänder set DDRAM adress för att flytta markören till önskad plats
        commandToDisplay(0b00000100, 0b10000000|location);
    }
}

/*Sätter flyttar markören till första rutan på diplayen*/
void cursorHome(){
    //0b00000100: Sätter E i high (PB2 = 1, PB1(R/W) = 0,  PB0 (RS) = 0,)
    //0b00001100: Sätt på displayen
    commandToDisplay(0b00000100, 0b00000010);
}

//Vyn för klockan
void clockView(){
    clearDisplay();
    moveCursor(0x40);
    writeString("Time: ");
    writeNumber(h1);
    writeNumber(h2);
    writeString(":");
    writeNumber(m1);
    writeNumber(m2);
    writeString(":");
    writeNumber(s1);
    writeNumber(s2);

    moveCursor(0x14);
    writeString("Alarm:");

    if(alarmOnOff == 0){
        moveCursor(0x1B);
        writeString("OFF");


        } else{
        writeNumber(ah1);
        writeNumber(ah2);
        writeString(":");
        writeNumber(am1);
        writeNumber(am2);



    }
    cursorOff();
}

//Vyn för när man ställer in klockan
void timeSettingView(){

    clearDisplay();

    moveCursor(0x04);
    writeString("Set the Time");
```

```
    moveCursor(0x40);
    writeString("Time:");
    writeNumber(setH1);
    writeNumber(setH2);
    writeString(":");
    writeNumber(setM1);
    writeNumber(setM2);

    cursorOff();
}

//Vyn för när man sätter alarmet
void alarmSettingView(){
    clearDisplay();
    moveCursor(0x03);
    writeString("Set the Alarm");

    moveCursor(0x40);
    writeString("Alarm:");
    writeNumber(ah1);
    writeNumber(ah2);
    writeString(":");
    writeNumber(am1);
    writeNumber(am2);

    cursorOff();
}

//Vyn när almarmen går av. Vi borde även displaya tiden här.
void alarmGoesOffView(){
    clearDisplay();
    moveCursor(0x47);
    writeString("Ring!");
    cursorOff();

}

//Metod för att kolla om alarmet ska gå av
void alarmGoesOff(){
    if(h1== ah1 && h2 == ah2 && m1 == am1 && m2 == am2 && alarmOnOff == 1 && settingAlarm == 0){
        alarmGoesOffView();
        summerOn();
        alarmRinging = 1;
        lightSensorOn();
    }

}

//Sätter på summer så den börjar låta
void summerOn(){
    //Behåller display inställningen samma (3 sista siffrorna)
    PORTB |=  0b00001000;

}

//Stänger av summern så den slutar låta
void summerOff(){
    //Behåller display inställningen samma (3 sista siffrorna)

    PORTB &= (0<<PB3);
}

//Stänger av ljussensors så den inte reagerar på ljus
void lightSensorOff(){
```

```
    DDRD = 0b00000100;
}

//Sätter på ljussensorn så den reagerar på ljus
void lightSensorOn(){
    DDRD = 0b00000000;
}

//Läser av vilken knappt som har blivit tryckt
void checkButtons() {

    char btn1 = (1 << PD4); //set time, vänster.
    char btn2 = (1 << PD5); //set alarm, höger
    char btn3 = (1 << PD6); //ner, sätt minuter
    char btn4 = (1 << PD7); // upp, sätt timmar samma som 0b10000000
    btn1 &= PIND; // kollar om set time knappen är 1 eller 0
    btn2 &= PIND;
    btn3 &= PIND;
    btn4 &= PIND;

    if(btn1 == 0b00010000) { //om set time är 1
        if(settingTime == 0){ //vill sätta tiden
            settingTime = 1;
            settingButton = 1;

            setH1 = h1;
            setH2 = h2;
            setM1 = m1;
            setM2 = m2;

            } else if (settingTime == 1){ //går från att sätta tiden till vanlig
            h1 = setH1;
            h2 = setH2;
            m1 = setM1;
            m2 = setM2;

            settingTime = 0;
            settingButton = 0;

        }
    }

    if(btn2 == 0b00100000){ //om set alarm är 1
        if(alarmOnOff == 0) { //om man vill sätta alarmet

            alarmOnOff = 1;
            settingAlarm = 1;
            settingButton = 1;
            summerOff();

            } else { //alarmet är på, alarmOnOff = 1
            if(alarmRinging == 1){ // alarmet ringer och man vill stänga av det
                alarmOnOff = 0;
                summerOff();
                alarmRinging = 0;
                lightSensorOff();

                } else if(settingAlarm == 0){ //man vill stäng av alarmet (som inte ringer),
                  settingAlarm = 0
                alarmOnOff = 0;
                ah1 = 0;
                ah2 = 0;
                am1 = 0;
                am2 = 0;
```

```
            }

            settingButton = 0; //man vill alltid gå ur settingAlarmView, hindar att knapparna
              fungerar efter att mna går ur alarm setting vyn
            settingAlarm = 0;  //man vill alltid gå ur settingAlarmView
        }
    }


    if(btn3 == 0b01000000 && settingTime== 1){ //sätt minuter tid
        setM2++;
        if (setM2 > 9){
            setM1++;
            setM2 = 0;
        }
    }

    if(btn4 == 0b10000000 && settingTime== 1){ //sätter timmar tid
        setH2++;

        if (setH2 > 9){
            setH1++;
            setH2 = 0;
        }
    }

    if(btn3 == 0b01000000 && settingAlarm == 1) { //alarm minuter
        am2++;

        if (am2 > 9){
            am1++;
            am2 = 0;
        }
    }
    if(btn4 == 0b10000000 && settingAlarm == 1){ //alarm timmar
        ah2++;

        if (ah2 > 9){
            ah1++;
            ah2 = 0;
        }
    }

}

void timerSetup()
{
    TCCR1B |= (1 << WGM12 | 1 << CS10 | 1 << CS12);  //Timer/Counter Control Register 1B. Sets up
      timer with prescaler = 1024 and CTC enabled. 8 bits.
    TCNT1 = 0x00; // Initialize counter
    OCR1A = 15625; // Output Compare Register 1A. Set the CTC compare value.  16 bits
    TIMSK |= ((1 << OCIE1A));  // Timer Interrupt Mask Register, Output Compare a match Interrupt
      1A. Enables the CTC interrupt. Vi använder CTC, så processorn ska ta emot signal när räknaren
      matchar.
    sei();  // enable Global interrupts.
}

ISR(TIMER1_COMPA_vect) { //Funktion som körs varje gång ett avbrott genereras   //External events
  trigger an interrupt – the normal control flow is interrupted and an Interrupt Service Routine
  (ISR) is called.
    //TIFR = 0x01; //Interrupt request har genererats. (Behövs detta?)
    if (s2<9) {
        s2Old = s2;
        s2++;
```

```
    }
    else {
        s2=0;
        if (s1<5) {
            s1++;
        }
        else {
            s1=0;
            if (m2<9) {
                m2++;
            }
            else {
                m2=0;
                if (m1<5) {
                    m1++;
                }
                else {
                    m1=0;
                    if (h1<2) {
                        if(h2<9) {
                            h2++;
                        }
                        else {
                            h2=0;
                            h1++;
                        }
                    }
                    else {
                        if (h2<3) {
                            h2++;
                            }else {
                            h1=0;
                            h2=0;
                            m1=0;
                            m2=0;
                            s1=0;
                            s2=0;
                        }
                    }
                }
            }
        }
    }
}

ISR(INT1_vect) {
    GIFR = 1<<INTF1;
    _delay_ms(100);
    checkButtons(); //kallar på knappmetoden
    _delay_ms(100);
}

ISR(INT0_vect) {
    GIFR = 1<<INTF0; //aktiveras när man lyser på ljussensorn
    _delay_ms(100);
    alarmOnOff = 0;
    summerOff();
    alarmRinging = 0;
    lightSensorOff();
    settingButton = 0;
    settingAlarm = 0;
    _delay_ms(100);
}
```

```c
/*Fixar med alla funktioner.*/
void start(){

    //Sätt vilka håll pins ska vara (DDR*)
    DDRA = 0xFF; // display, output
    DDRB = 0xFF; // E, R/w, RS (display) + summer, output
    DDRC = 0x00; //används inte av oss, input
    DDRD = 0x00; // knappar + ljussensor, input

    PORTD = 0x00;

    //Avbrottsrutiner
    GICR |= 1<<INT1 | 1<<INT0;
    MCUCR |= 1<<ISC11 | 1<<ISC10 | 1<<ISC01 | 1<<ISC00;


    //Fixar med timern
    timerSetup();

    //Förbereder displayen
    functionSet();
    clearDisplay();
    displayOn();
    cursorOn();
    cursorHome();

}

int main(void)
{
    start();

    while (1)
    {
        _delay_ms(10);


        if(settingButton == 1 && settingTime == 1){
            timeSettingView();


            } else if (settingButton == 1 && settingAlarm == 1){
            alarmSettingView();

        }

        if(s2 != s2Old && alarmRinging == 0 && settingButton == 0){
            clockView();
            alarmGoesOff();
        }

    }
}
```