```
#include <avr/io.h>
#define F_CPU         16000000L         // CPU clock speed 16 MHz
#define F_SCL         100000L           // I2C clock speed 100 kHz
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>

///////////////%%%%%%%%%%%%%%%%/////////////////&&&&&&&&&&&&&&&&/////////////////////////////
////((((((((((((((((((/////////////////"////////////////////


#define SECONDS_REGISTER   0x00
#define MINUTES_REGISTER   0x01
#define HOURS_REGISTER     0x02
#define DAYOFWK_REGISTER   0x03

#define YEARS_REGISTER     0x06

#define DAYS_REGISTER      0x04
#define MONTHS_REGISTER    0x05

#define TW_ACK        0xC4              // return ACK to slave

#define TW_START      0xA4              // send start condition (TWINT,TWSTA,TWEN)
#define TW_READY     (TWCR & 0x80)      // ready when TWINT returns to logic 1.
#define TW_STATUS    (TWSR & 0xF8)      // returns value of status register

#define DS1307        0xD0              // I2C bus address of DS1307 RTC
#define TW_SEND       0x84              // send data (TWINT,TWEN)

#define TW_STOP       0x94              // send stop condition (TWINT,TWSTO,TWEN)
#define I2C_Stop()    TWCR = TW_STOP    // inline macro for stop condition

#define TW_NACK       0x84              // read data with NACK (last byte)
#define READ          1


uint8_t I2C_Start(void);
uint8_t I2C_SendAddr(uint8_t addr);
uint8_t I2C_Write (uint8_t data);
uint8_t I2C_ReadNACK (void);
```

```
uint8_t I2C_ReadRegister(uint8_t deviceRegister);
uint8_t I2C_ReadRegister(uint8_t deviceRegister);

void DS1307_GetTime(uint8_t *hours, uint8_t *minutes, uint8_t *seconds);
void DS1307_GetDate(uint8_t *months, uint8_t *days);

void SetTime_Date(void);
void WriteTime(void);
void WriteDate(void);
void TwoDigits(uint8_t data);

void RTC_GetDateTime(void);
void RTC_SetDateTime(void);
///////////////%%%%%%%%%%%%%%%%%%/////////////////&&&&&&&&&&&&&&&&&&////////////////////////////////
////((((((((((((((((((/////////////////"////////////////////

void homeDisplay();
void updateHomeDisplay();

uint8_t rtc_sec1 = 0;
uint8_t rtc_sec10 = 0;
uint8_t rtc_min1 = 0;
uint8_t rtc_min10 = 0;
uint8_t rtc_hour1 = 0;
uint8_t rtc_hour10 = 0;
uint8_t rtc_date1 = 0;
uint8_t rtc_date10 = 0;
uint8_t rtc_month1 = 0;
uint8_t rtc_month10 = 0;

uint8_t alm_min1 = 0;
uint8_t alm_min10 = 0;
uint8_t alm_hour1 = 3;
uint8_t alm_hour10 = 2;

uint8_t crnt_s1 = 0;
uint8_t crnt_s10 = 0;
uint8_t crnt_m1 = 0;
uint8_t crnt_m10 = 0;
uint8_t crnt_h1 = 0;
uint8_t crnt_h10 = 0;
uint8_t crnt_d1 = 0;
uint8_t crnt_d10 = 0;
```

```
uint8_t crnt_month1 = 0;
uint8_t crnt_month10 = 0;

uint8_t temp1 = 0;
uint8_t tempor = 0x00;


int getKey(void);

void alarmOff(void);
void alarmOn(void);

void updateHomeDisplay(void);


void sendCommand(unsigned char command);
void sendCharacter(unsigned char character);
void sendString(char *string_of_characters);
void sendNbr(int nbr);
void clearDisplay(void);


void snooze(void);
void on(void);
void off(void);
void setAlarm(void);
void setDate(void);
void setTime(void);
void Clock(void);

uint8_t count = 0;
volatile uint8_t Mode = 0;
volatile uint8_t buttons1;
volatile uint8_t buttons2;
volatile uint8_t buttons3;
volatile uint8_t buttons4;

////////////////%%%%%%%%%%%%%%%%/////////////////&&&&&&&&&&&&&&&&////////////////////////////////
////((((((((((((((((((/////////////////"////////////////////

void I2C_Init()
{
        TWSR = 0;                       // set prescalar to zero
```

```
        TWBR = ((F_CPU/F_SCL)-16)/2;        // set SCL frequency in TWI bit register
}

uint8_t I2C_Start() //  generate a TW start condition
{   TWCR = TW_START;                    // send start condition
        while (!TW_READY);                  // wait
        return (TW_STATUS==0x08);           // return 1 if found; 0 otherwise
}

uint8_t I2C_SendAddr(uint8_t addr) //  send bus address of slave
{   TWDR = addr;                        // load device's bus address
        TWCR = TW_SEND;                     // and send it
        while (!TW_READY);                  // wait
        return (TW_STATUS==0x18);           // return 1 if found; 0 otherwise
}

uint8_t I2C_Write (uint8_t data)            // sends a data byte to slave
{   TWDR = data;                        // load data to be sent
        TWCR = TW_SEND;                     // and send it
        while (!TW_READY);                  // wait
        return (TW_STATUS!=0x28);           // return 1 if found; 0 otherwise
}

uint8_t I2C_ReadACK ()                  // reads a data byte from slave
{     TWCR = TW_ACK;                    // ack = will read more data
        while (!TW_READY);                  // wait
        return TWDR;
        //return (TW_STATUS!=0x28);
}

uint8_t I2C_ReadNACK ()                 // reads a data byte from slave
{
        TWCR = TW_NACK;                     // nack = not reading more data
        while (!TW_READY);                  // wait
        return TWDR;
}

void I2C_WriteRegister(uint8_t deviceRegister, uint8_t data)
{
        I2C_Start();
        I2C_SendAddr(DS1307);               // send bus address
        I2C_Write(deviceRegister);          // first byte = device register address
        I2C_Write(data);                    // second byte = data for device register
```

```
	I2C_Stop();
}


uint8_t I2C_ReadRegister(uint8_t deviceRegister)
{
	uint8_t data = 0;
	I2C_Start();
	I2C_SendAddr(DS1307);		// send device bus address
	I2C_Write(deviceRegister);	// set register pointer
	I2C_Start();
	I2C_SendAddr(DS1307+READ);	// restart as a read operation
	data = I2C_ReadNACK();		// read the register data
	I2C_Stop();			// stop
	return data;
}

void DS1307_GetTime(uint8_t *hours, uint8_t *minutes, uint8_t *seconds) // returns hours,
minutes, and seconds in BCD format
{
	*hours = I2C_ReadRegister(0x02);
	*minutes = I2C_ReadRegister(0x01);
	*seconds = I2C_ReadRegister(0x00);
}

void DS1307_GetDate(uint8_t *months, uint8_t *days) // returns months, days, and years in
BCD format
{
	*months = I2C_ReadRegister(0x05);
	*days = I2C_ReadRegister(0x04);
}

// HÄMTA TID OCH DATUM PÅ RTC
void RTC_GetDateTime()
{
	uint8_t temp;

	uint8_t hours, minutes, seconds;
	DS1307_GetTime(&hours,&minutes,&seconds);
	temp = hours;			// get upper 4 bits
	temp &= 0xF0;		// get upper 4 bits
	temp = temp>>4;			// right shift 4 high bits
	hours &= 0x0F;			// get lower 4 bits
```

```
rtc_hour10 = temp;
rtc_hour1 = hours;
_delay_ms(1);

temp = minutes;                 // get upper 4 bits
temp &= 0xF0;           // get upper 4 bits
temp = temp>>4;                 // right shift 4 high bits
minutes &= 0x0F;        // get lower 4 bits

rtc_min10 = temp;
rtc_min1 = minutes;
_delay_ms(1);

temp = seconds;                         // get upper 4 bits
temp &= 0xF0;                   // get upper 4 bits
temp = temp>>4;                         // right shift 4 high bits
seconds &= 0x0F;          // get lower 4 bits

rtc_sec10 = temp;
rtc_sec1 = seconds;
_delay_ms(1);

uint8_t months, days;
DS1307_GetDate(&months,&days);

temp = months;                          // get upper 4 bits
temp &= 0xF0;                   // get upper 4 bits
temp = temp>>4;                         // right shift 4 high bits
months &= 0x0F;                     // get lower 4 bits

rtc_month10 = temp;
rtc_month1 = months;
_delay_ms(1);

temp = days;                    // get upper 4 bits
temp &= 0xF0;                   // get upper 4 bits
temp = temp>>4;                         // right shift 4 high bits
days &= 0x0F;             // get lower 4 bits

rtc_date10 = temp;
rtc_date1 = days;
```

```
}

// STÄLL IN TID OCH DATUM PÅ RTC (set_clock)
void RTC_SetDateTime(){

        tempor = 0x00;
        tempor = (rtc_month10 << 4) + (rtc_month1);
        I2C_WriteRegister(0x05, tempor);

        tempor = 0x00;
        tempor = (rtc_date10 << 4) + (rtc_date1);
        I2C_WriteRegister(0x04, tempor);

        tempor = 0x00;
        tempor = (rtc_hour10 << 4) + (rtc_hour1);
        I2C_WriteRegister(0x02, tempor);

        tempor = 0x00;
        tempor = (rtc_min10 << 4) + (rtc_min1);
        I2C_WriteRegister(0x01, tempor);

}

////////////////%%%%%%%%%%%%%%%%/////////////////&&&&&&&&&&&&&&&&&////////////////////////////////
/////((((((((((((((((((/////////////////"////////////////////

int getKey()
{

        buttons1 = PIND & 0b00001000;
        buttons2 = PIND & 0b00000100;
        buttons3 = PIND & 0b00000010;
        buttons4 = PIND & 0b00000001;

        if(buttons1 == 8) return 1;
        if(buttons2 == 4) return 2;
        if(buttons3 == 2) return 3;
        if(buttons4 == 1) return 4;

        return 0;

}
```

```
void sendNbr(int nbr){
	switch(nbr){
		case 0:
		sendString("0");
		break;
		case 1:
		sendString("1");
		break;
		case 2:
		sendString("2");
		break;
		case 3:
		sendString("3");
		break;
		case 4:
		sendString("4");
		break;
		case 5:
		sendString("5");
		break;
		case 6:
		sendString("6");
		break;
		case 7:
		sendString("7");
		break;
		case 8:
		sendString("8");
		break;
		case 9:
		sendString("9");
		break;
	}
}

void setAlarm()
{
	sendCommand(0x01);//clear
	_delay_ms(5);
	sendCommand(0xC0);//dis on
	_delay_ms(5);
	sendCommand(0x80); // cursor first line
	_delay_ms(5);
```

```
sendString("Set ALhour: ");

_delay_ms(5);
sendNbr(alm_hour10);
_delay_ms(5);
sendNbr(alm_hour1);
_delay_ms(5);
while(1){

        if(getKey() == 1){
                alm_hour10 = (alm_hour10 + 1)%3;
                sendCommand(0x8C); // cursor first line
                sendNbr(alm_hour10);
                _delay_ms(10);
        }

        if(getKey() == 2){
                alm_hour1 = (alm_hour1 + 1)%10;
                sendCommand(0x8D); // cursor first line
                sendNbr(alm_hour1);
                _delay_ms(10);
        }

        if((alm_hour10 >=2) && (alm_hour1 >= 4) )
        {
                alm_hour10 = 0;
                alm_hour1 = 0;
                sendCommand(0x8C); // cursor first line
                _delay_ms(5);
                sendNbr(alm_hour10);
                sendNbr(alm_hour1);
                _delay_ms(10);
        }

        if(getKey() == 4){
                break;
        }
}

sendCommand(0x01);//clear
_delay_ms(5);
sendCommand(0xC0);//dis on
```

```
_delay_ms(5);
sendCommand(0x80); // cursor first line

sendString("Set ALMmin: ");
sendNbr(alm_min10);
_delay_ms(5);
sendNbr(alm_min1);
_delay_ms(5);
while(1){
	if(getKey() == 1){
		alm_min10 = (alm_min10 + 1)%7;
		sendCommand(0x8C); // cursor first line
		sendNbr(alm_min10);
		_delay_ms(10);
	}
	if(getKey() == 2){
		alm_min1 = (alm_min1 + 1)%10;
		sendCommand(0x8D); // cursor first line
		sendNbr(alm_min1);
		_delay_ms(10);
	}
	if((alm_min10 >=6) && (alm_min1 >= 0) )
	{
		alm_min10 = 0;
		alm_min1 = 0;
		_delay_ms(5);
		sendCommand(0x8B); // cursor first line
		_delay_ms(5);
		sendNbr(alm_min10);
		sendNbr(alm_min1);
		_delay_ms(10);
	}
	if(getKey() == 4){
		break;
	}
}
sendCommand(0x01);//clear
_delay_ms(5);
sendCommand(0xC0);//dis on
_delay_ms(5);
sendCommand(0x80); // cursor first line
sendString("Alarm set:");
sendNbr(alm_hour10);
```

```
	sendNbr(alm_hour1);
	sendString(":");
	sendNbr(alm_min10);
	sendNbr(alm_min1);
	_delay_ms(100);
}

void  Clock()
{
	uint8_t count = 0;
	homeDisplay();

	while(1){
		if((alm_min1 == rtc_min1) && (alm_min10 == rtc_min10) && (alm_hour1 ==
rtc_hour1) && (alm_hour10 == rtc_hour10))
		{
			alarmOn();
		}
		updateHomeDisplay();

		if(getKey() == 1)
		{
			snooze();
		}

		if(getKey() == 2)
		{

			if(count==1)
			{
				off();
				_delay_ms(5);
			}
			if(count!=1)
			{
				on();
				_delay_ms(5);
			}
			count = (count + 1)%2;
		}

		if(getKey() == 3)
		{
```

```
                alarmOff();
            }

            if(getKey() == 4)
            {
                break;
            }

        }
}


void off()
{
        sendCommand(0x01);//clear
        _delay_ms(5);
        DDRA &= 0b00000000;
        _delay_ms(5);
}

void on()
{
        DDRA |= 0b11111111;
        _delay_ms(5);
        sendCommand(0xC0);//dis on
        homeDisplay();
        _delay_ms(5);
}



void setTime_Date()
{
        sendCommand(0x01);//clear
        _delay_ms(5);
        sendCommand(0xC0);//dis on
        _delay_ms(5);
        sendCommand(0x80); // cursor first line
        sendString("Set hour: ");
        sendNbr(rtc_hour10);
        _delay_ms(5);
        sendNbr(rtc_hour1);
        _delay_ms(5);
```

```
while(1)
{
	if(getKey() == 1)
	{
		rtc_hour10 = (rtc_hour10 + 1)%3;
		sendCommand(0x8A); // cursor
		sendNbr(rtc_hour10);
		_delay_ms(10);
	}
	if(getKey() == 2)
	{
		rtc_hour1 = (rtc_hour1 + 1)%10;
		sendCommand(0x8B); // cursor
		sendNbr(rtc_hour1);
		_delay_ms(10);
	}
	if((rtc_hour10 >=2) && (rtc_hour1 >= 4) )
	{
		rtc_hour10 = 0;
		rtc_hour1 = 0;
		sendCommand(0x8A); // cursor
		sendNbr(rtc_hour10);
		sendNbr(rtc_hour1);
		_delay_ms(10);
	}
	if(getKey() == 4)
	{
		break;
	}
}


sendCommand(0x01);//clear
_delay_ms(5);
sendCommand(0xC0);//dis on
_delay_ms(5);
sendCommand(0x80); // cursor first line

sendString("Set minute: ");
sendNbr(rtc_min10);
_delay_ms(5);
sendNbr(rtc_min1);
while(1)
```

```
{
        if(getKey() == 1)
        {
                rtc_min10 = (rtc_min10 + 1)%7;
                sendCommand(0x8C); // cursor
                sendNbr(rtc_min10);
                _delay_ms(10);
        }
        if(getKey() == 2)
        {
                rtc_min1 = (rtc_min1 + 1)% 10;
                sendCommand(0x8D); // cursor
                sendNbr(rtc_min1);
                _delay_ms(10);
        }

        if((rtc_min10 >=6) && (rtc_min1 >= 0) )
        {
                rtc_min10 = 0;
                rtc_min1 = 0;
                sendCommand(0x8C); // cursor
                sendNbr(rtc_min10);
                sendNbr(rtc_min1);
                _delay_ms(10);
        }

        if(getKey() == 4)
        {
                break;
        }
}
sendCommand(0x01);//clear
_delay_ms(5);
sendCommand(0xC0);//dis on
_delay_ms(5);
sendCommand(0x80); // cursor first line

sendString("Time set:");
sendNbr(rtc_hour10);
_delay_ms(5);
sendNbr(rtc_hour1);
_delay_ms(5);
sendString(":");
```

```
	_delay_ms(5);
	sendNbr(rtc_min10);
	_delay_ms(5);
	sendNbr(rtc_min1);
	_delay_ms(100);

////////////////////////////////////DAAAAAAAAAAAAAAATEEEEEEEEEEEEEEEEEEE////////////////////////

////////////////////////////////////DAAAAAAAAAAAAAAATEEEEEEEEEEEEEEEEEEE////////////////////////

////////////////////////////////////DAAAAAAAAAAAAAAATEEEEEEEEEEEEEEEEEEE////////////////////////

////////////////////////////////////DAAAAAAAAAAAAAAATEEEEEEEEEEEEEEEEEEE////////////////////////
	sendCommand(0x01);//clear
	_delay_ms(5);
	sendCommand(0xC0);//dis on
	_delay_ms(5);
	sendCommand(0x80); // cursor first line
	sendString("Set month: ");
	_delay_ms(5);
	sendNbr(rtc_month10);
	_delay_ms(5);
	sendNbr(rtc_month1);
	_delay_ms(5);
	while(1)
	{
		if(getKey() == 1)
		{
			rtc_month10 = (rtc_month10 + 1)%2;
			sendCommand(0x8B); // cursor
			sendNbr(rtc_month10);
			_delay_ms(10);
		}
		if(getKey() == 2)
		{
			rtc_month1 = (rtc_month1 + 1)%10;
			sendCommand(0x8C); // cursor
			_delay_ms(5);
			sendNbr(rtc_month1);
			_delay_ms(10);
		}
		if(getKey() == 4)
		{
```

```
                break;
        }

        if((rtc_month10 >=1) && (rtc_month1 >= 3) )
        {
                rtc_month10 = 0;
                rtc_month1 = 0;
                sendCommand(0x8B); // cursor
                sendNbr(rtc_month10);
                sendNbr(rtc_month1);
        }
}


sendCommand(0x01);//clear
_delay_ms(5);
sendCommand(0xC0);//dis on
_delay_ms(5);
sendCommand(0x80); // cursor first line
sendString("Set date: ");
_delay_ms(5);
sendNbr(rtc_date10);
_delay_ms(5);
sendNbr(rtc_date1);
_delay_ms(5);



while(1)
{
        if(getKey() == 1)
        {
                rtc_date10 = (rtc_date10 + 1)%4;
                sendCommand(0x8A); // cursor
                sendNbr(rtc_date10);
                _delay_ms(10);
        }
        if(getKey() == 2)
        {
                rtc_date1 = (rtc_date1 +1)%10;
                sendCommand(0x8B); // cursor first line
                sendNbr(rtc_date1);
                _delay_ms(10);
```

```
		}
		if(getKey() == 4)
		{
			break;
		}
	}

	if((rtc_date10 >=3) && (rtc_date1 >= 2) )
	{
		rtc_date10 = 0;
		rtc_date1 = 0;
		sendCommand(0x8A); // cursor first line
		sendNbr(rtc_date10);
		sendNbr(rtc_date1);
		_delay_ms(10);
	}



	sendCommand(0x01);//clear
	_delay_ms(5);
	sendCommand(0xC0);//dis on
	_delay_ms(5);
	sendCommand(0x80); // cursor first line

	sendString("Date set: ");

	sendNbr(rtc_month10);
	sendNbr(rtc_month1);
	sendString("/");
	sendNbr(rtc_date10);
	sendNbr(rtc_date1);
	_delay_ms(100);
	RTC_SetDateTime();
	_delay_ms(10);
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE         ////////////////////////////////////////
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE////////////////////////////////////////
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE////////////////////////////////////////
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE////////////////////////////////////////
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE////////////////////////////////////////
	////////////////KLAAAAAAAAAAAAAAR MED TIME/DATE////////////////////////////////////////
}
```

```
void homeDisplay()
{
	sendCommand(0x01);//clear
	sendCommand(0x80);//first line
	sendNbr(crnt_month10);
	sendNbr(crnt_month1);
	sendString("/");
	sendNbr(crnt_d10);
	sendNbr(crnt_d1);
	sendString(" ");
	sendNbr(crnt_h10);
	sendNbr(crnt_h1);
	sendString(":");
	sendNbr(crnt_m10);
	sendNbr(crnt_m1);
	sendString(":");
	sendNbr(crnt_s10);
	sendNbr(crnt_s1);
}

void updateHomeDisplay()
{
	RTC_GetDateTime();
	if(crnt_s1 != rtc_sec1)
	{
		sendCommand(0x8D);
		sendNbr(rtc_sec1);
		crnt_s1 = rtc_sec1;
	}

	if(crnt_s10 != rtc_sec10)
	{
		sendCommand(0x8C);
		sendNbr(rtc_sec10);
		crnt_s10 = rtc_sec10;
	}

	if(crnt_m1 != rtc_min1)
	{
		sendCommand(0x8A);
		sendNbr(rtc_min1);
		crnt_m1 = rtc_min1;
	}
```

```
if(crnt_m10 != rtc_min10)
{
        sendCommand(0x89);
        sendNbr(rtc_min10);
        crnt_m10 = rtc_min10;
}

if(crnt_h1 != rtc_hour1)
{
        sendCommand(0x87);
        sendNbr(rtc_hour1);
        crnt_h1 = rtc_hour1;
}

if(crnt_h10 != rtc_hour10)
{
        sendCommand(0x86);
        sendNbr(rtc_hour10);
        crnt_h10 = rtc_hour10;
}

if(crnt_d1 != rtc_date1)
{
        sendCommand(0x84);
        sendNbr(rtc_date1);
        crnt_d1 = rtc_date1;
}

if(crnt_d10 != rtc_date10)
{
        sendCommand(0x83);
        sendNbr(rtc_date10);
        crnt_d10 = rtc_date10;
}
if(crnt_month1 != rtc_month1)
{
        sendCommand(0x81);
        sendNbr(rtc_month1);
        crnt_month1 = rtc_month1;
}

if(crnt_month10 != rtc_month10)
```

```
        {
                sendCommand(0x80);
                sendNbr(rtc_month10);
                crnt_month10 = rtc_month10;
        }

}


void alarmOff()
{
        PORTB &=~(1 << 7);
        homeDisplay();
        updateHomeDisplay();
}
void alarmOn()
{

        sendCommand(0x01);//clear
        _delay_ms(5);
        sendCommand(0xC0);//dis on
        _delay_ms(5);
        sendCommand(0x80); // cursor first line

        sendString("WAKE UP!!! ");
        while(1)
        {
                _delay_us(50);
                PORTB |= (1 << 7);
                _delay_us(50);
                PORTB &=~(1 << 7);
                if(getKey() == 3)
                {
                        alm_min1--;
                        alarmOff();
                                if(alm_min1>=10)
                                {
                                        alm_min10 = alm_min10 +1;
                                }

                                if(alm_min10 >=6)
                                {
                                        alm_hour1 = alm_hour1+1;
```

```
                        }

                        if(alm_hour1>=10)
                        {
                                alm_hour10 = alm_hour10+1;
                        }

                        if(alm_hour10>=6)
                        {
                                alm_hour10 = 0;
                        }
                        break;
                }
                if(getKey() == 1)
                {
                        snooze();
                        break;
                }

                _delay_us(50);
                PORTB |= (1 << 7);
                _delay_us(50);
                PORTB &=~(1 << 7);
        }
}
void snooze()
{
        alarmOff();
        alm_min1 = alm_min1 +5;
        if(alm_min1>=10)
        {
        alm_min10 = alm_min10 +1;
        }

        if(alm_min10 >=6)
        {
        alm_hour1 = alm_hour1+1;
        }

        if(alm_hour1>=10)
        {
        alm_hour10 = alm_hour10+1;
        }
```

```
        if(alm_hour10>=6)
        {
        alm_hour10 = 0;
        }
        _delay_us(100);
        alarmOff();
        updateHomeDisplay();
        _delay_ms(5);
        sendCommand(0xC0);
        _delay_ms(5);
        sendString("Snooze: ");
        _delay_ms(5);
        sendNbr(alm_hour10);
        _delay_ms(5);
        sendNbr(alm_hour1);
        _delay_ms(5);
        sendString(":");
        _delay_ms(5);
        sendNbr(alm_min10);
        _delay_ms(5);
        sendNbr(alm_min1);
}
void clearDisplay()
{
        sendCommand(0x01);
}

void sendCommand(unsigned char command)
{
        PORTA = command;
        PORTB &=~ (1<<4); // 1 på rs
        PORTB &=~ (1<<3); //0 på r/w
        PORTB |= (1<<2); //enable
        _delay_ms(2);
        PORTB &=~ (1<<2); //enable klar
}
void sendCharacter(unsigned char character)
{
        PORTA = character;
        PORTB |= (1<<4); // 1 på rs
        PORTB &=~ (1<<3); //0 på r/w
        PORTB |= (1<<2); //enable
```

```
	_delay_ms(2);
	PORTB &=~ (1<<2); //enable klar
}
void sendString(char *string_of_characters)
{
	while(*string_of_characters>0){
		_delay_us(5);
		sendCharacter(*string_of_characters++);
		_delay_us(5);
	}
}


int main(void)
{

	DDRB |= 0b10011100; //rs rw e
	DDRA |= 0b11111111; // led output
	DDRC = 0b00110000;//sda,5 scl,6

	DDRD &=~ 0b00001111;//knapp

	sendCommand(0x38); //5x7 matrix
	_delay_ms(5);
	sendCommand(0x01); //clear
	_delay_ms(5);
	//sendCommand(0x10); //blinking
	_delay_ms(5);
	sendCommand(0x0c); //display on
	_delay_ms(5);
	sendCommand(0x80); // cursor first line


	_delay_ms(5);
	sei();
	RTC_GetDateTime();
	while(1){
				if((alm_min1 == rtc_min1) && (alm_min10 == rtc_min10) &&
(alm_hour1 == rtc_hour1) && (alm_hour10 == rtc_hour10))
				{
					alarmOn();
				}
		if(getKey() == 4)
```

```
                {
                        Mode = Mode + 1;
                }

                Mode = Mode%3;

                switch(Mode)
                {
                        case 0: //Vanlig klocka
                        Clock();
                        break;

                        case 1: //alarmsinställning
                        setAlarm();

                        Mode++;
                        break;

                        case 2: //tidsinställning Datuminställning
                        setTime_Date();
                        Mode++;
                        break;
                        Mode = Mode%3;
                }

        }
}
```