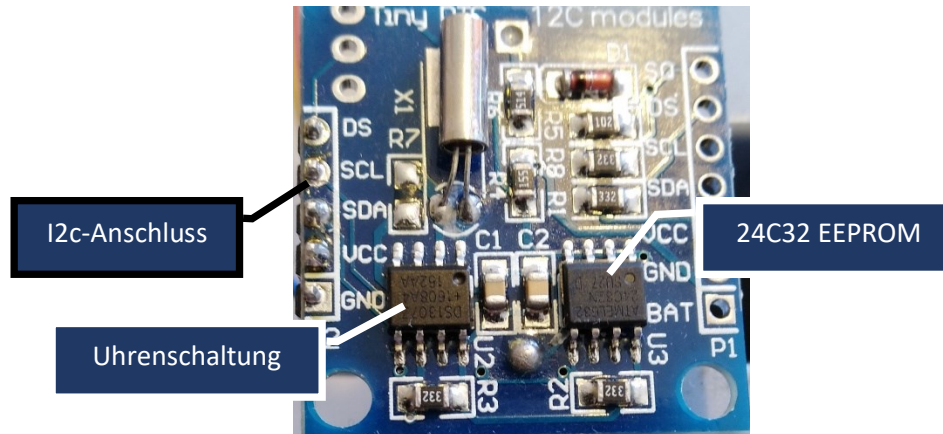


Real Time Clock (RTC)

Das RTC-(Real Time Clock)-Modul verfügt über 2 integrierte Schaltkreise:

- Uhren-Schaltung
- EEPROM-Speicher 24C32



Die Uhrenschaltung wird von einer Batterie auf der Rückseite der Platine versorgt. Die Uhr läuft daher immer, unabhängig vom Mikrocontroller.

Das EEPROM dient der Speicherung von Daten. Ein EEPROM ist ein nichtflüchtiger Speicher. D. h. beim Ausfall der Stromversorgung behält das EEPROM seinen Inhalt.

E	Electrical
E	Erasable
P	Programable
R	Read
O	Only
M	Memory

Das EEPROM 24C32 verfügt über eine Speicherkapazität von 4096 Bytes.

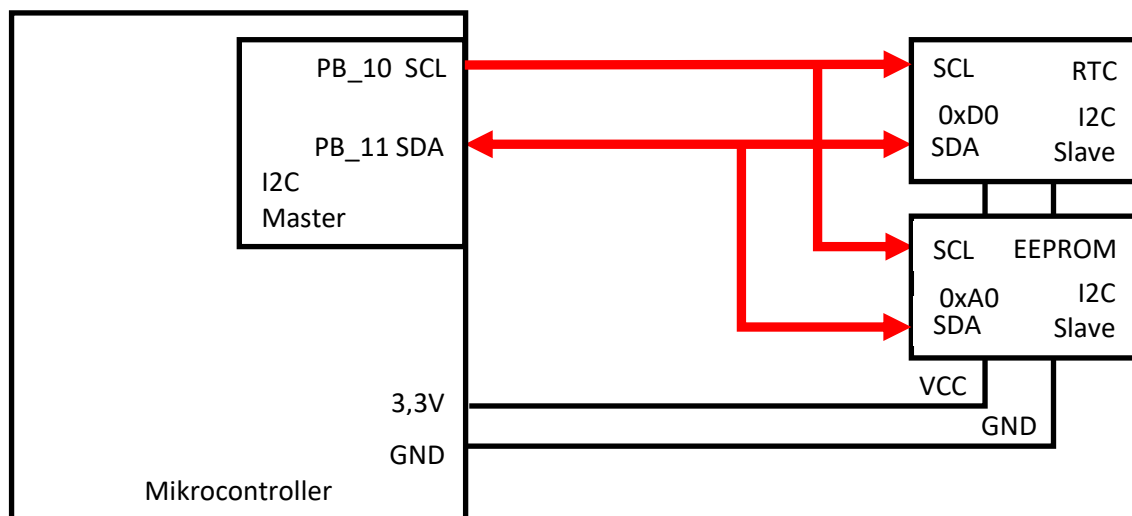


Abbildung 1: Anschlussschema

Programmierung:

- Deklaration der i2c-Schnittstelle
I2C i2c(PB_11,PB_10);
- Daten aus dem EEPROM abrufen:
i2c.write(0xA0,speicheradresse,2,1);
i2c.read(0xA0,daten,anzahl,0);
mit:
char speicheradresse[2]={*Highbyte, Lowbyte*};
 - Highbyte: Highbyte der Speicheradresse im EEPROM
 - Lowbyte: Lowbyte der Speicheradresse im EEPROM
char daten[*anzahl*]={ *Datenbytes* }; //anzahl = Anzahl der Datenbytes im array
- Daten in das EEPROM schreiben:
i2c.write(0xA0,speicheradresse,2,1);
i2c.write(0xA0,daten,anzahl,0);
mit:
char speicheradresse[2]={*Highbyte, Lowbyte*};
 - Highbyte: Highbyte der Speicheradresse im EEPROM
 - Lowbyte: Lowbyte der Speicheradresse im EEPROM
Die Speicheradresse geht von 0x0000 bis maximal 0xFFFF

char daten[*anzahl*]={ *Datenbytes* }; //anzahl = Anzahl der Datenbytes im array

Hinweis: Spätestens nach 64 Bytes muss dem EEPROM ca. 10 ms Zeit gegeben werden, um die Daten in seinen Festwertspeicher zu übertragen.

- Daten aus der RTC abrufen:
i2c.write(0xD0,speicheradresse,1,1);
i2c.read(0xD0,daten,7,0);
mit:
char speicheradresse[1]={0}; //die Speicheradresse ist nur ein Byte groß
char daten[7]={ *Datenbytes* };

• Daten in der RTC eintragen (Uhr stellen):
i2c.write(0xD0,speicheradresse,1,1);
i2c.write(0xD0,daten,7,0);
mit:
char speicheradresse[1]={0}; //die Speicheradresse ist nur ein Byte groß
char daten[7]={ *Datenbytes* };

Datenformat:

Speicheradresse	Bedeutung	Beispiel
0	Sekunde	0x29 = 29 Sekunden
1	Minute	0x53 = 53 Minuten
2	Stunden	0x13 = 13 Stunden
3	Wochentag	0x00 = Montag
4	Tag im Monat	0x18 = 18. des Monats
5	Monat	0x08 = August
6	Jahr	0x20 =2020

Aufgaben:

1. Uhr stellen und anzeigen
 - a. Bei Druck auf Taste PA_10 soll die Uhr mit dem festen Wert:
19:56:00 Uhr, Montag, 18. August 2020 gestellt werden
 - b. Die aktuelle Uhrzeit und das Datum soll im Sekunden-Rhythmus aus der Uhr
ausgelesen werden und auf dem LCD ausgegeben werden.
2. Zeitschaltuhr: Bis zu 16 Einschalt- und 16 Ausschaltzeiten sollen im EEPROM
gespeichert werden. Wenn die Uhrzeit eine Einschaltzeit erreicht wird der Ausgang
PC_7 auf 1 gesetzt. Wenn die Uhrzeit eine Ausschaltzeit erreicht wird der Ausgang
PC_7 auf 0 rückgesetzt.

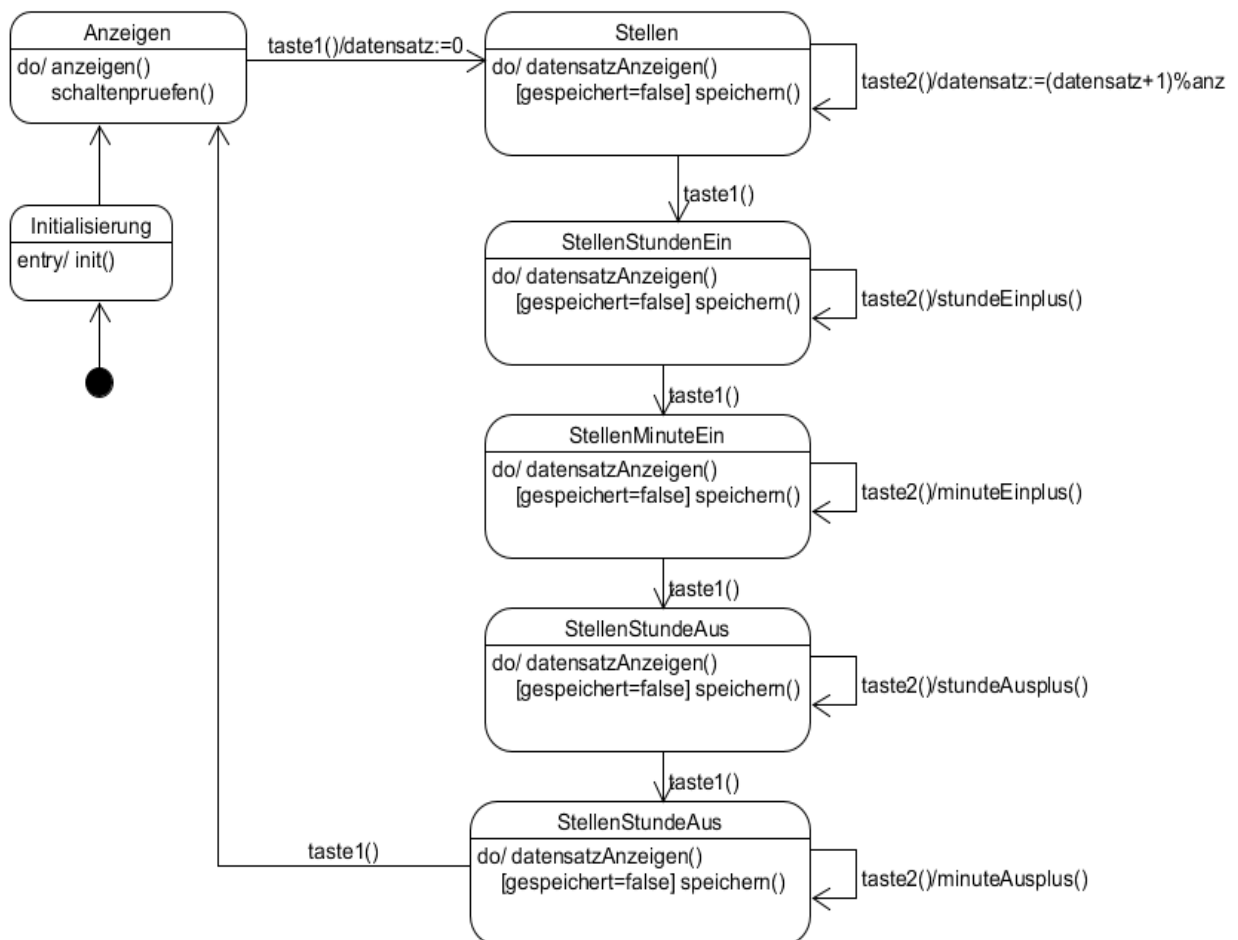


Abbildung 2: Zustandsdiagramm der Zeitschaltuhr

Programminweise:

- Uhr stellen wie in Aufgabe 1 hier in Operation: anzeigen()
- Zustandsvariable: PortOut zustand(Portc,0x3F);
- In der Operation init() werden die Ports initialisiert, das LCD gelöscht und das EEPROM in arrays ausgelesen.
- anzeigen() zeigt Datum und Uhrzeit an und überprüft die Taste zum Reset von Datum und Uhrzeit auf den Anfangswert
- schalterpruefen() vergleicht die Uhrzeit (Stunden und Minuten) mit den programmierten Ein- und Ausschaltzeiten und schaltet den Ausgang PC_7 entsprechend.

- `datensatzanzeigen()` zeigt die Datensatznummer (0 .. 15), die Einschaltuhrzeit (Stunde und Minute) sowie die Ausschaltuhrzeit des Datensatzes an
- Die Operationen: `stundeEinplus()`, `minuteEinplus()`, `stundeAusplus()`, `minuteAusplus()` erhöhen die jeweilige Zeitangabe und setzen die globale Variable *gespeichert* auf false.
- `speichern()` überträgt die Datensätze (arrays) ins EEPROM

Musterlösung:

```
#include "mbed.h"
```

```
#include "platform/mbed_thread.h"
```

```
#include "LCD.h"
```

```
#define RTC 0xD0
```

```
#define EEPROM 0xA0
```

```
#define Anzeigen      0b000001
```

```
#define Stellen       0b000010
```

```
#define StellenStundeEin 0b000100
```

```
#define StellenMinuteEin 0b001000
```

```
#define StellenStundeAus 0b010000
```

```
#define StellenMinuteAus 0b100000
```

```
#define anz 2
```

```
//sda: PB11, SCL: PB10
```

```
I2C i2c(PB_11,PB_10);
```

```
Serial pc(USBTX, USBRX);
```

```
DigitalIn setzen(PB_0);
```

```
DigitalOut fehler(PC_6);
```

```
DigitalOut schaltausgang(PC_7);
```

```
Lcd mylcd;
```

```
PortOut anzeige(PortC,0xFF);
```

```
InterruptIn T1(PA_10);
```

```
InterruptIn T2(PA_6);
```

```
PortOut zustand(PortC,0x3F);
```

```
char daten[8];
```

```
char datenSet[8]={0,0,0x56,0x19,0x01,0x18,0x08,0x20};
```

```
char einH[16];
```

```
char einM[16];
```

```
char ausH[16];
```

```
char ausM[16];
```

```
char adresse[2];
```

```
bool gespeichert=true;
```

```
int datensatz=0;
```

```
void anzeigen()
```

```
{
```

```
    char* wota;
```

```
    daten[0]=0;
```

```
    i2c.write(RTC,daten,1,1);
```

```
    i2c.read(RTC,daten,7,0);
```

```
    switch (daten[3])
```

```
    {
```

```
        case 0: wota="Mo.";break;
```

```
        case 1: wota="Di.";break;
```

```
        case 2: wota="Mi.";break;
```

```

        case 3: wota="Do.";break;
        case 4: wota="Fr.";break;
        case 5: wota="Sa.";break;
        case 6: wota="So.";break;
    }
    mylcd.cursorpos(0);
    mylcd.printf("%2x:%2x:%2x      ",daten[2], daten[1], daten[0]);
    mylcd.cursorpos(0x40);
    mylcd.printf("%s den %x.%x.%x      ",wota, daten[4], daten[5], daten[6]);
    if (setzen==1)
    {
        i2c.write(0xD0,datenSet,8,0);
        while (setzen==1);
    }
}

int toBCD(int w)
{
    int r;
    r=((w/10)<<4)+w%10;
    return r;
}

void schaltenpruefen()
{
    for (int i=0;i<anz;i++)
    {
        if (daten[1]==toBCD(einM[i]) && daten[2]==toBCD(einH[i])) schaltausgang=1;
    }
    for (int i=0;i<anz;i++)
    {
        if (daten[1]==toBCD(ausM[i]) && daten[2]==toBCD(ausH[i])) schaltausgang=0;
    }
}

void laden()
{
    adresse[0]=0;
    adresse[1]=0;
    fehler=i2c.write(EEPROM,adresse,2,1);
    fehler=i2c.read(EEPROM,einH,16,1);
    fehler=i2c.read(EEPROM,einM,16,1);
    fehler=i2c.read(EEPROM,ausH,16,1);
    fehler=i2c.read(EEPROM,ausM,16,1);
}

void speichern()
{
    adresse[0]=0;
    adresse[1]=0;
    fehler=i2c.write(EEPROM,adresse,2,1);
    fehler=i2c.write(EEPROM,einH,16,1);
    fehler=i2c.write(EEPROM,einM,16,0);
}

```

```

        wait_us(10000);
        adresse[0]=0;
        adresse[1]=32;
        fehler=i2c.write(EEPROM,adresse,2,1);
        fehler=i2c.write(EEPROM,ausH,16,1);
        fehler=i2c.write(EEPROM,ausM,16,0);
        gespeichert=true;
    }

void datensatzAnzeigen()
{
    mylcd.cursorpos(0);
    mylcd.printf("%2d E: h:%2d m:%2d    ",datensatz,einH[datensatz],einM[datensatz]);
    mylcd.cursorpos(0x40);
    mylcd.printf("A:h:%2d m:%2d    ",ausH[datensatz],ausM[datensatz]);
}

void stundeEinPlus()
{
    einH[datensatz]=(einH[datensatz]+1)%23;
    gespeichert=false;
}

void minuteEinPlus()
{
    einM[datensatz]=(einM[datensatz]+1)%60;
    gespeichert=false;
}

void stundeAusPlus()
{
    ausH[datensatz]=(ausH[datensatz]+1)%23;
    gespeichert=false;
}

void minuteAusPlus()
{
    ausM[datensatz]=(ausM[datensatz]+1)%60;
    gespeichert=false;
}

void taste1()
{
    switch(zustand)
    {
        case Anzeigen: datensatz=0;
            zustand=Stellen;
            break;
        case Stellen:
            zustand=StellenStundeEin;
            break;
        case StellenStundeEin:
            zustand=StellenMinuteEin;

```

```

        break;
    case StellenMinuteEin:
        zustand=StellenStundeAus;
        break;
    case StellenStundeAus:
        zustand=StellenMinuteAus;
        break;
    case StellenMinuteAus:
        zustand=Anzeigen;
        break;
    }
}

```

```

void taste2()
{
    switch(zustand)
    {
        case Stellen: datensatz=(datensatz+1)%anz;
            break;
        case StellenStundeEin: stundeEinPlus();
            break;
        case StellenMinuteEin: minuteEinPlus();
            break;
        case StellenStundeAus: stundeAusPlus();
            break;
        case StellenMinuteAus: minuteAusPlus();
            break;
    }
}

```

```

void init()
{
    setzen.mode(PullDown);
    T1.mode(PullDown);
    T1.rise(&taste1);
    T2.mode(PullDown);
    T2.rise(&taste2);
    mylcd.clear();
    laden();
}

```

```

int main()
{
    init();
    zustand=Anzeigen;
    while (true) {
        switch (zustand)
        {
            case Anzeigen:
                anzeigen();
                schaltenpruefen();
                break;
            default: datensatzAnzeigen();

```



```
        if (!gespeichert)
        {
            speichern();
        }
        break;
    }
}
```