7 Uhr und Zeitraster

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

7. Uhr und Zeitraster

Ziele

Nach dieser Lektion sollten Sie:

1. wissen, wie man ....

Übung

I. Vorarbeiten

1. Laden Sie folgende Datei herunter:
    1. 7_mexleclock.simu
    2. 7_mexleclock.hex
    3. lcd_lib_de.h

II. Analyse des fertigen Programms

1. Initialisieren des Programms
    1. Öffnen Sie SimulIDE und öffnen Sie dort mittels  die Datei `7_mexleclock.simu`
    2. Laden Sie `7_mexleclock.hex` als firmware auf den 328 Chip
    3. Zunächst wird eine Startanzeige mit dem Namen des Programms dargestellt.
    4. Als nächstes ist im Display eine Uhr mit dem Format HH:MM:SS Menu zu sehen
    5. Die Tasten 2 und 3 ermöglichen das Einstellen der Stunde und Minute. Werden die Minuten hochgezählt, so werden die Sekunden auf 0 gesetzt.

III. Eingabe in Atmel Studio

```
/*
============================
============================
====================

Experiment 7:   7_mexleclock
mit Stunden-, Minuten- und
Sekunden-Anzeige
=============
============================
============================
=

Dateiname:
7_MexleClock.c

Autoren:        Peter
Blinzinger
```

```
/*====================================
========================================
=
```

Ändern Sie auch hier wieder die Beschreibung am Anfang des C-Files, je nachdem was Sie entwickeln

```
                Prof. G.
Gruhler (Hochschule
Heilbronn)
                D.
Chilachava     (Georgische
Technische Universitaet)

Version:        1.2 vom
01.05.2020

Hardware:       MEXLE2020
Ver. 1.0 oder höher
                AVR-USB-
PROGI Ver. 2.0

Software:
Entwicklungsumgebung:
AtmelStudio 7.0
                C-Compiler:
AVR/GNU C Compiler 5.4.0

Funktion:       Digitaluhr
mit Anzeige von Stunden,
Minuten und Sekunden. Eine
                einfache
Stellfunktion ist mit den
Tasten S2 und S3 realisiert.

Displayanzeige: Start (fuer
2s):        Betrieb:
                +-----------
-----+       +-------------
---+
                |-
Experiment 7 -|        |===
00:00:00 ===|
                | Digital
Clock  |        |   Std Min
|
                +-----------
-----+       +-------------
---+

Tastenfunktion: S2:    Std
(zaehlt Stunden bei Flanke
aufwaerts. Überlauf bei 24)
                S3: Min
(zaehlt Minuten bei Flanke
aufwaerts. Überlauf bei 60)
(setzt Sekunden beim
Druecken zurueck auf 00)
```

```
Deklarationen
=================================
```

1. Hier wird wieder geprüft ob die Frequenz des Quarz bereits eingestellt wurde und - falls nicht - dessen Frequenz eingestellt.

2. Die Header-Dateien entsprechen denen der letzten Programme.

3. Auch die Makros entsprechen denen der letzten Programme.

4. Die Konstanten entsprechen denen der letzten Programme.
   Zusätzlich wird der Ausdruck ASC_NULL durch den Hexadezimalwert einer '0' in ASCII, also 0x30 und ASC_COLON durch den ASCI-Wert eines Doppelpunkts, also 0x3A, ersetzt

5. Bei den Variablen entsprechen einige denen der letzten Programme.
6. Für die Uhr werden Stunden, Minuten, Sekunden und Zehntelsekunden mit Anfangswerten deklariert.

7. Bei den Funktionsprototypen sind einige bekannte Unterprogramme vorhanden. Details werden weiter unten erklärt.

```
Jumperstellung: keine
Auswirkung

Fuses im uC:    CKDIV8: Aus
(keine generelle Vorteilung
des Takts)

Header-Files:   lcd_lib_de.h
(Library zur Ansteuerung
LCD-Display Ver. 1.3)

Module:         1)
Taktgenerator
                2) Zaehler
fuer Uhr    (Takt: 1 s)
                3)
Anzeigetreiber     (Takt:
100 ms)
                4)
Stellfunktion      (Takt: 10
ms)

    Modul 1:    Das Modul
"Taktgenerator" erzeugt den
Takt von 1s fuer die Uhr.
                Zusaetzliche
Takte: 10 ms fuer
Stellfunktion
100 ms fuer Anzeige.

                Verwendung
von Hardware-Timer 0 und T0
Overflow-Interrupt.
Frequenzen: Quarzfrequenz
12,288 MHz.
Timer-Vorteiler        /  8
=>   1,536 MHz
Hardware-Timer         /256
=>   6 kHz / 166 µs
Software-Vorteiler     / 60
=> 100 Hz  /  10 ms
Hundertstel-Zaehler    / 10
=>  10 Hz  / 100 ms
Zehntel-Zaehler        / 10
=>   1 Hz  /   1 s

    Modul 2:     Das Modul
"Zaehler fuer Uhr" wird
durch den Takt 1s
aufgerufen.
                Sekunden,
```

Hauptprogramm ==========

1. Das Hauptprogramm ähnelt sehr stark dem Up/Down Counter. Entsprechend werden die Zeilen 143-157 hier nicht weiter erklärt.

2. In der Endlosschleife sind auf der ersten Ebene wieder nur If-Abfragen zu den Flags `takt10ms` und `takt100ms` zu finden.

    1. Alle $10ms$ (bzw. wenn das entsprechende Flag gesetzt wird) wird das Flag zurückgesetzt und das Unterprogramm `uhrStellen()` aufgerufen

    2. Alle $100ms$ (bzw. wenn das entsprechende Flag gesetzt wird) wird das Flag zurückgesetzt und das Unterprogramm `uhrAnzeigen()` aufgerufen

    3. Alle $1s$ (bzw. wenn das entsprechende Flag gesetzt wird) wird das Flag zurückgesetzt und das Unterprogramm `uhrZaehlen()` aufgerufen

Interrupt Routine ==========

1. Mit dem Befehl `ISR()` wird eine Interrupt Service Routine für den OVerFlow Interrupt für TIMER0 angelegt.
2. Der Überlauf-Interrupt durch den Timer0 wird erst bei Überlauf des 8-Bit Wert ausgeführt. Auch hier ergibt sich durch den Prescaler und Modus (TCCR0A und TCCR0B) eine Periode von $T_{ISR}= 0,16\bar{6}ms$.

```
Minuten und Stunden werden
als Binaerzahlen gezaehlt
                Sekunden und
Minuten zaehlen 00..59, die
Stunden 00..23.
                Ein "Tick"
auf dem Lautsprecher wird
jede Sekunde ausgegeben.

    Modul 3:    Das Modul
"Anzeigetreiber" startet
alle 100 ms. Es gibt die
Hintergrundinformationen und
die aktuelle Uhrzeit aus.
Darstellung auf der Anzeige
(mittig in Zeile 1):
[23:59:59]
    Modul 4:     Das Modul
"Stellfunktion" ist an den
10 ms-Takt gekoppelt.
                Es dient
1. zum Einlesen und
Entprellen der Stelltasten
- Auswertung der fallenden
Flanke 1=> 0
2. zum Ausfuehren der
Stellfunktion:
- S2 zaehlt die Stunden
aufwaerts
- S3 zaehlt die Minuten
aufwaerts
- solange Taste S3
gedrueckt: Sekunden = 00
(einfache Synchronisierung
der Uhr!)
                  Beim
Stellen kein Uebertrag von
den Minuten auf die Stunden.

    Die Kopplung der Module
wird ueber global definierte
Variable realisiert:

    1-Bit-Variable:
takt10ms:    Taktgenerator
=> Stellfunktion
takt100ms:    Taktgenerator
=> Anzeigetreiber
takt1s:        Taktgenerator
=> Zaehler fuer Uhr
```

3. Die Ermittlung von `Timertick`, `vorteiler`, `takt10ms`, `hundertstel` und `takt100ms` ist hier wieder gleich dem im Up/Down Counter.
4. Eine große Änderung ist, dass bereits im Interrupt alle 10ms die Unterfunktion `readButton()` aufgerufen wird.

```
Taster initialisieren ==============
```

1. Das Einstellen des Data Direction Registers und der Pullups wurde bereits in vorherigen Programmen erklärt.

```
Funktion Tasten einlesen ==============
```

1. In dieser Funktion werden zunächst die Stellungen aller Taster eingelesen (vgl. `counterCounting(void)` bei Up/down Counter).

2. Neu hier ist, dass die Bedienung der Schalter nur das Flag `castBit` setzen. Falls dieses gesetzt ist, wird die Variable `castVar` hochgezählt. Falls diese 6 überschreitet wird sie auf 1 zurückgesetzt.

```
Anzeigefunktion Wuerfel
========================
```

```
    8-Bit-Variable:
sekunden    Stellfunktion =>
Zaehler => Anzeige
minuten
stunden

============================
============================
====================*/

// Deklarationen
============================
============================
======

// Festlegung der
Quarzfrequenz
#ifndef F_CPU
// optional definieren
#define F_CPU 12288000UL
// ATmega 328 mit 12,288 MHz
Quarz
#endif

// Include von Header-
Dateien
#include <avr/io.h>
// I/O-Konfiguration (intern
weitere Dateien)
#include <avr/interrupt.h>
// Definition von Interrupts
#include <util/delay.h>
// Definition von Delays
(Wartezeiten)
#include "lcd_lib_de.h"
// Header-Datei fuer LCD-
Anzeige

// Makros
#define SET_BIT(PORT, BIT)
((PORT) |=  (1 << (BIT))) //
Port-Bit Setzen
#define CLR_BIT(PORT, BIT)
((PORT) &= ~(1 << (BIT))) //
Port-Bit Loeschen
#define TGL_BIT(PORT, BIT)
((PORT) ^=  (1 << (BIT))) //
Port-Bit Toggeln

// Konstanten
#define VORTEILER_WERT
```

1. Hierüber wird die Augenzahl in der zweiten Zeile an Position 7 ausgegeben.

```
Initialisierung Display-Anzeige
========================
```

1. Die Funktion `initDisplay()` wird zu Beginn des Programms aufgerufen und führt zunächst die Initialisierung des Displays aus.
2. Danach wird der erste Text auf den Bildschirm geschrieben und damit der Programmname dargestellt.
3. Nach zwei Sekunden wird der Auswahlbildschirm angezeigt.

```
60        // Faktor
Vorteiler = 90
#define HUNDERTSTEL_WERT
10         // Faktor
Hundertstel = 10
#define ZEHNTEL_WERT
10         // Faktor Zehntel
= 10

#define SPEAK_PORT
PORTD    // Port-Adresse
fuer Lautsprecher
#define SPEAK_BIT
5        // Port-Bit fuer
Lautsprecher

#define ASC_NULL
0x30        // Das Zeichen
'0' in ASCII
#define ASC_COLON
0x3A        // Das Zeichen
':' in ASCII

// Variable
unsigned char vorteiler
= VORTEILER_WERT;    //
Zaehlvariable Vorteiler
unsigned char hundertstel
= HUNDERTSTEL_WERT; //
Zaehlvariable Hundertstel
unsigned char zehntel
= ZEHNTEL_WERT;        //
Zaehlvariable Zehntel
unsigned char sekunden
= 56;                //
Variable Sekunden
unsigned char minuten
= 34;                //
Variable Minuten
unsigned char stunden
= 12;                //
Variable Stunden

bool timertick;
// Bit-Botschaft alle
0,166ms (bei Timer-
Interrupt)
bool takt10ms;
// Bit-Botschaft alle 10ms
bool takt100ms;
// Bit-Botschaft alle 100ms
```

```
bool takt1s;
// Bit-Botschaft alle 1s

bool sw2_neu = 1;
// Bitspeicher fuer Taste 2
bool sw3_neu = 1;
// Bitspeicher fuer Taste 3
bool sw2_alt = 1;
// alter Wert von Taste 2
bool sw3_alt = 1;
// alter Wert von Taste 3

// Funktionsprototypen
void init_Taster(void);
//Taster initialisieren
void uhrStellen(void);
// Stellfunktion
void uhrAnzeigen(void);
// Anzeigefunktion
void uhrZaehlen(void);
// Uhrfunktion
void initDisplay(void);
// Init Anzeige

// Hauptprogramm
============================
============================
======
int main()
{
    // Initialisierung
    init_Taster();
//Taster initialisieren
    initDisplay();
// Initialisierung LCD-
Anzeige

    TCCR0A = 0;
// Timer 0 auf "Normal Mode"
schalten
    SET_BIT(TCCR0B, CS01);
// mit Prescaler /8
betreiben
    SET_BIT(TIMSK0, TOIE0);
// Overflow-Interrupt
aktivieren

    SET_BIT(DDRD,
SPEAK_BIT);    // Speaker-
Bit auf Ausgabe
```

```
    sei();
// generell Interrupts
einschalten

    // Hauptprogrammschleife

    while(1)
// unendliche Warteschleife
mit Aufruf der
// Funktionen abhaengig von
Taktbotschaften
    {
        if (takt10ms)
// alle 10ms:
        {
            takt10ms = 0;
//        Botschaft "10ms"
loeschen
            uhrStellen();
//        Tasten abfragen,
Uhr stellen
        }
        if (takt100ms)
// alle 100ms:
        {
            takt100ms = 0;
//        Botschaft "100ms"
loeschen
            uhrAnzeigen();
//        Uhrzeit auf
Anzeige ausgeben
        }
        if (takt1s)
// alle Sekunden:
        {
            takt1s = 0;
//        Botschaft "1s"
loeschen
            uhrZaehlen();
//        Uhr weiterzaehlen
        }
    }
    return 0;
}

// Interrupt-Routine
============================
============================
==
ISR (TIMER0_OVF_vect)
/*  In der Interrupt-Routine
```

```
sind die Softwareteiler
realisiert, die die Takt-
    botschaften (10ms,
100ms, 1s) fuer die gesamte
Uhr erzeugen. Die Interrupts
    werden von Timer 0
ausgeloest (Interrupt Nr. 1)

    Veraenderte Variable:
vorteiler
hunderstel
zehntel

    Ausgangsvariable:
takt10ms
takt100ms
takt1s
*/
{
    timertick = 1;
// Botschaft 0,166ms senden
    --vorteiler;
// Vorteiler dekrementieren
    if (vorteiler==0)
// wenn 0 erreicht: 10ms
abgelaufen
    {
        vorteiler =
VORTEILER_WERT;    //
Vorteiler auf Startwert
        takt10ms = 1;
//    Botschaft 10ms senden
        --hundertstel;
//    Hunderstelzaehler
dekrementieren

        if (hundertstel==0)
// wenn 0 erreicht: 100ms
abgelaufen
        {
            hundertstel =
HUNDERTSTEL_WERT; // Teiler
auf Startwert
            takt100ms = 1;
//    Botschaft 100ms senden
            --zehntel;
//    Zehntelzaehler
dekrementieren

            if (zehntel==0)
// wenn 0 erreicht: 1s
```

```
abgelaufen
            {
                zehntel =
ZEHNTEL_WERT;    //
Teiler auf Startwert
                takt1s = 1;
//    Botschaft 1s senden
            }
        }
    }
}

// Taster initialisieren
============================
==========================
void init_Taster(void)
{
    DDRB = DDRB & 0xE1;
// Port B auf Eingabe
schalten
    PORTB |= 0x1E;
// Pullup-Rs eingeschaltet
    _delay_us(10);
// Wartezeit Umstellung
Hardware-Signal
}

// Stellfunktion
============================
============================
======
void uhrStellen(void)
/*  Die Stellfunktion der
Uhr wird alle 10ms
aufgerufen. Dadurch wir eine
    Entprellung der
Tastensignale realisiert.
Das Stellen wir bei einer
    fallenden Flanke des
jeweiligen Tastensignals
durchgefuehrt. Darum
    muss fuer einen weiteren
Stellschritt die Taste
erneut betaetigt werden.

    Eine Flanke wird durch
(alter Wert == 1) UND
(aktueller Wert == 0)
erkannt.

    Mit der Taste S2 werden
```

```
die Stunden aufwaerts
gestellt.
    Mit der Taste S3 werden
die Minuten aufwaerts
gestellt (kein Uebertrag)
    Solange Taste S3
gedrueckt ist werden die
Sekunden auf 00 gehalten

    Veraenderte Variable:
stunden
minuten
sekunden

    Speicher fuer Bits:
sw2Alt
sw3Alt
*/
{
    sw2_neu = (PINB & (1 <<
PB2));    // Tasten von Port
einlesen
    sw3_neu = (PINB & (1 <<
PB3));

    if
((sw2_neu==0)&(sw2_alt==1))
// wenn Taste 2 eben
gedrueckt wurde:
    {
        stunden++;
//    Stunden hochzaehlen,
Ueberlauf bei 23
        if (stunden==24)
            stunden = 00;
    }
    if
((sw3_neu==0)&(sw3_alt==1))
// wenn Taste 3 eben
gedrueckt wurde:
    {
        minuten++;
//    Minuten hochzaehlen,
Ueberlauf bei 59
        if (minuten==60)
            minuten = 00;
    }
    if (sw3_neu==0)
// solange Taste 3
gedrueckt:
        sekunden = 00;
```

```
//       Sekunden auf 00
setzen

    sw2_alt = sw2_neu;
// aktuelle Tastenwerte
speichern
    sw3_alt = sw3_neu;
//    in Variable fuer alte
Werte
}

// Anzeigefunktion Uhr
============================
============================
void uhrAnzeigen(void)
/*  Die Umrechnung der
binaeren Zaehlwerte auf BCD
ist folgendermaßen geloest:
    Zehner: einfache
Integer-Teilung (/10)
    Einer:  Modulo-
Ermittlung (%10), d.h. Rest
bei der Teilung durch 10
*/
{
    lcd_gotoxy(0,4);
// Cursor auf Start der
Zeitausgabe setzen
    lcd_putc(ASC_NULL +
stunden/10);    // Stunden
Zehner als ASCII ausgeben
    lcd_putc(ASC_NULL +
stunden%10);    // Stunden
Einer als ASCII ausgeben
    lcd_putc(ASC_COLON);
// Doppelpunkt ausgeben
    lcd_putc(ASC_NULL +
minuten/10);    // Minuten
als ASCII ausgeben
    lcd_putc(ASC_NULL +
minuten%10);    //
    lcd_putc(ASC_COLON);
// Doppelpunkt ausgeben
    lcd_putc(ASC_NULL +
sekunden/10);   // Sekunden
als ASCII ausgeben
    lcd_putc(ASC_NULL +
sekunden%10);   //
}

// Initialisierung Display-
```

```
Anzeige
============================
================
void initDisplay()
// Start der Funktion
{
    lcd_init();
// Initialisierungsroutine
aus der lcd_lib
    lcd_gotoxy(0,0);
// Cursor auf 1. Zeile, 1.
Zeichen
    lcd_putstr("- Experiment
7 -"); // Ausgabe Festtext:
16 Zeichen

    lcd_gotoxy(1,0);
// Cursor auf 2. Zeile, 1.
Zeichen
    lcd_putstr(" Digital
Clock  ");    // Ausgabe
Festtext: 16 Zeichen

    _delay_ms(2000);
// Wartezeit nach
Initialisierung

    lcd_gotoxy(0,0);
// Cursor auf 1. Zeile, 1.
Zeichen
    lcd_putstr("=== 00:00:00
==="); // Ausgabe Festtext:
16 Zeichen

    lcd_gotoxy(1,0);
// Cursor auf 2. Zeile, 1.
Zeichen
    lcd_putstr("   Std Min
");    // Ausgabe Festtext:
16 Zeichen
}
// Ende der Funktion

// Zaehlfunktion Uhr
============================
============================
==
void uhrZaehlen (void)
// wird jede Sekunde
gestartet
/*  Die Uhr wird im
```

```
Sekundentakt gezaehlt. Bei
jedem Aufruf wird auch ein
    "Tick" auf dem
Lautsprecher ausgegeben.
Ueberlaeufe der Sekunden
zaehlen
    die Minuten, die
Ueberlaeufe der Minuten die
Stunden hoch.

    Veraenderte Variable:
sekunden
minuten
stunden
*/
{
    TGL_BIT (SPEAK_PORT,
SPEAK_BIT); // "Tick" auf
Lautsprecher ausgeben
// durch Invertierung des
Portbits

    sekunden++;
// Sekunden hochzaehlen
    if (sekunden==60)
// bei Überlauf:
    {
        sekunden = 0;
//  Sekunden auf 00 setzen
        minuten++;
//  Minuten hochzaehlen
        if (minuten==60)
//    bei Ueberlauf:
        {
            minuten = 0;
//    Minuten auf 00 setzen
            stunden++;
//    Stunden hochzaehlen
            if (stunden==24)
//    bei Ueberlauf:
                stunden = 0;
//    Stunden auf 00 setzen
        }
    }
}
```

IV. Ausführung in Simulide

1. Geben Sie die oben dargestellten Codezeilen ein und kompilieren Sie den Code.
2. Öffnen Sie Ihre hex-Datei in SimulIDE und testen Sie, ob diese die gleiche Ausgabe

erzeugt

Bitte arbeiten Sie folgende Aufgaben durch:

Aufgabe

1. Darstellung des Augenwerts ändern
    1. Laden Sie folgende Dateien herunter: 6a_mexlecast_extern.simu, 6a_mexlecast_extern.hex
    2. Simulieren Sie die Schaltung mit der beigefügten hex-Datei
    3. Ändern Sie das bisherige Programm so, dass Sie das gleiche Ergebnis erhalten, bzw. zumindest die Ansteuerung der LEDs.