Prüfungsvorleistung

Entwurf und Programmierung eines Kurzzeitweckers / einer Eieruhr

für den ATmega328

Vorgelegt am: ??.01.2023

Von: R. Jehring M. Seegert F. Mehlhorn M. Schröder

Studiengang: Vernetzte Intelligente Systeme | Elektrotechnik - Automation

Seminargruppe: EAw22m

Matrikelnummer: 59026 ? ? 59255

Aufgabenstellungen zum Testat (Prüfungsvorleistung)

Seminargruppe EAw22m

**Aufgabe:**

Es ist ein Kurzzeitwecker / Eieruhr zu programmieren.

**Randbedingungen:**

Der Kurzzeitwecker soll nach einer einstellbaren Zeit einen akustischen Alarm ausgeben.

Er soll eine Zeitspanne von 1...59 Minuten erlauben.

Die restlich verbliebene Zeit ist in Minuten und Sekunden anzuzeigen.

Nach Start des Programms muss die Zeitspanne (in ganzen Minuten) eingestellt werden.

Ta 1 — Zeit inkrementieren

Ta 2 — Zeit dekrementiere

Ta 3 — Start des Timers

Sobald die Zeit abgelaufen ist, bleibt die Restzeit bei 00:00 stehen und ein Alarm ertönt.

Der Alarm kann mit Ta 3 abgeschaltet werden.

Der Wecker kann mit Ta 3 vor Ablauf der Zeit gestoppt werden.

Nach Stopp oder nach Ablauf der Zeit ist eine Neueingabe der Zeitdauer möglich.

**Realisierung:**

Mit einem Arduino Uno (ATmega328) und Arduino Studio.

# Inhaltsverzeichnis Seite

[Inhaltsverzeichnis Seite III](#_Toc123734889)

[Abbildungsverzeichnis Seite IV](#_Toc123734890)

[Tabellenverzeichnis Seite V](#_Toc123734891)

[Abkürzungsverzeichnis VI](#_Toc123734892)

[Vorwort 1](#_Toc123734893)

[1. Überschrift 1 2](#_Toc123734894)

[1.1 Überschrift 2 2](#_Toc123734895)

# Abbildungsverzeichnis Seite

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

# Tabellenverzeichnis Seite

Es konnten keine Einträge für ein Abbildungsverzeichnis gefunden werden.

# Abkürzungsverzeichnis

BCD – Binary Coded Decimal

# Vorwort

# Timer

## Registerübersicht

Timer 0/Timer 2

|  |  |  |
| --- | --- | --- |
| Register | Inhalt | Beschreibung |
| TCCR0A/TCCR2A | 0x02 | CTC-Modus |
| TCCR0B | 0x03 | Taktteiler /64 |
| TCCR2B | 0x04 | Taktteiler/256 |
| OCR0A/OCR2A | 250 | Auslösung Compare-Match bei Zählweite 250 |
| TIMSK0/TIMSK2 | 0x02 | Compare Match A Interrupt aktivieren |

## Berechnung der Frequenzen

Timer 1

|  |  |
| --- | --- |
| Vorteiler = 64  Zählweite = 250  Umläufe Timer = 1000 | f = fCPU /(Vorteiler\*Zählweite\*Umläufe)  f = 16MHz/(64\*250\*1000)  f = 1Hz |

Timer 2

|  |  |
| --- | --- |
| Vorteiler = 256  Zählweite = 250  Umläufe Timer = 1  Anweisungsdurchläufe bis Bildschirmaktualisierung = 4 | f = fCPU /(Vorteiler\*Zählweite\*Umläufe\*Anweisungsdurchläufe)  f = 16MHz/(256\*250\*1\*4)  f = 62,5Hz |

## Interrupt-Service-Routines

ISR-Timer 0

Die ISR des Timers 0 wird aufgrund der vorher berechneten Taktzeit und der Zählweite 1000x/s ausgelöst. Dabei wird jedes Mal die Variable milliseconds inkrementiert (Speicherung Umläufe). Hat diese den Wert 1000 erreicht, wird Sie auf 0 gesetzt und die Variable seconds dekrementiert. Diese gibt die Restzeit in s für die Eieruhr an. Sobald seconds den Wert 0 erreicht, wird die Variable isAlarm = 1 gesetzt. Diese löst in main() den Alarm aus. Die Variable milliseconds wird dabei nur inkrementiert, wenn die Variable isTimer den Wert 1 besitzt. Dadurch kann die Eieruhr gestartet und gestoppt werden.

ISR-Timer 2

Die ISR des Timers 2 wird aufgrund der vorher berechneten Taktzeit und der Zählweite 500x/s ausgelöst. Bei Aufruf werden jeweils eins der vier 7-Segment-Anzeigen aktualisiert. Die Variable seg gibt die zu aktualisierende Anzeige an und wird bei jedem Aufruf inkrementiert. Hat Sie den Wert 4 erreicht, wird Sie wieder auf 1 gesetzt. Dadurch werden alle Anzeigen mit demselben Zeitabstand aktualisiert. Die Aktualisierungsrate des Bildschirms beträgt damit 62,5Hz.