

6 Würfel und Zufall

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

6. Würfel und Zufall

.....

2

Ziele

.....

2

Übung

.....

2

6. Würfel und Zufall

Ziele

Nach dieser Lektion sollten Sie:

1. wissen, wie man Zufallswerte in einem deterministischen System einfach erstellen kann.

Übung

I. Vorarbeiten

1. Laden Sie folgende Datei herunter:

1. [6_mexle_cast.simu](#)
2. [6_mexle_cast.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 6_mexle_cast.simu
2. Laden Sie 6_mexle_cast.hex als firmware auf den 328 Chip
3. Zunächst wird eine Startanzeige mit dem Namen des Programms dargestellt.
4. Als nächstes ist im Display ein Menu zu sehen. Durch Tastendruck ein Würfel gestartet (Druch auf Taste 1) oder gestoppt (4) werden kann. Die Augenzahl des Würfels ist in der Mitte der unteren Zeile dargestellt.

2. Das Programm zu diesem Hexfile soll nun erstellt werden

III. Eingabe in Atmel Studio

```

/*=====
=====
=
/*=====
=====
=====
Experiment 6:  MEXLEcast
Elektronischer Wuerfel auf
MiniMEXLE
=====
=====
=====

Dateiname:
MEXLEcast_de.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
30.04.2020

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: Es wird ein
elektronischer Wuerfel mit
Anzeige auf dem Display
realisiert.
Mit zwei Tasten S1 = Start
und S4 = Stop wird der
Wuerfel
gesteuert. Der Wuerfel wird
mit 10ms-Takt gezaehlt. Die
Anzeige
erfolgt als Ziffer im 100ms-
Takt.

Displayanzeige: Start (fuer
2s): Betrieb:
+-----
-----+ +-----
---+
| -
Experiment 6 - |
| Electronic Cast |
| Electronic
Cast | | Start 1
Stop |
+-----
-----+ +-----
---+

Tastenfunktion: S1: Start
(Set-Funktion Flip-Flop)

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.
5. Alle Variablen außer castBit entsprechen denen der letzten Programme. castBit ist ein "Merker", bzw. Flipflop welches das Würfeln aktiviert oder deaktiviert
6. Bei den Funktionsprototypen sind einige bekannte Unterprogramme vorhanden. Details werden weiter unten erklärt.

Hauptprogramm =====

1. Das Hauptprogramm ähnelt sehr stark dem [Up/Down Counter](#).
2. Zunächst werden zwei Initialisierungsroutinen aufgerufen (siehe weiter unten)
3. Dann werden wieder die "Timer/Counter Control Register" des Timers 0 TCCR0A und TCCR0B gesetzt. Der 8-Bit Timer und auch hier

S4: Stop
(Reset-Funktion Flip-Flop)

Jumperstellung: keine
Auswirkung

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

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

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

```
// Deklarationen
=====
=====
=====
```

```
// Festlegung der
Quarzfrequenz
#ifndef F_CPU
// optional definieren
#define F_CPU 12288000UL
// MiniMEXLE 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)
```

im "Normal Mode" zum hochzählen genutzt.
Auch hier gibt das Register TCCR0B den
Prescaler an.

4. Auch hier wird über die "Timer Interrupt
MaSK" TIMSK0 durch das Bit T0IE0 ("Timer
Overflow Interrupt Enable") der Interrupt bei
Überlauf aktiviert.
5. Mit dem Befehl sei() wird die Bearbeitung
von Interrupts aktiv
6. In der Endlosschleife sind auf der ersten Ebene
wieder nur If-Abfragen zu den Flags \$10ms\$
und \$100ms\$ zu finden.

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

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

Interrupt Routine

```
=====
```

1. Mit dem Befehl ISR() wird eine Interrupt
Service Routine für den OVerFlow Interrupt für
TIMER2 angelegt.
2. Der Überlauf-Interrupt durch den Timer2 wird
erst bei Überlauf des 8-Bit Wert ausgeführt.
Auch hier ergibt sich durch den Prescaler und
Modus (TCCR2A und TCCR2B) eine Periode von
\$T_{\{ISR\}} = 0,16\bar{6}\$ms\$.
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.

```
((PORT) &= ~(1 << (BIT))) // Taster initialisieren =====
Port-Bit Loeschen
#define TGL_BIT(PORT, BIT)
((PORT) ^= (1 << (BIT))) //
Port-Bit Toggeln
```

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

```
// Konstanten
#define VORTEILER_WERT
60 // Faktor
Vorteiler (Timerticks 0,111
ms)
#define HUNDERTSTEL_WERT
10 // Faktor
Hunderstel (1/100 s)
```

Funktion Tasten einlesen =====

```
// Variable
unsigned char vorteiler
= VORTEILER_WERT; //
Zaehlvariable Vorteiler
unsigned char hundertstel
= HUNDERTSTEL_WERT; //
Zaehlvariable Hunderstel
```

1. In dieser Funktion werden zunächst die Stellungen aller Taster eingelesen (vgl. counterCounting(void) bei [Up/down Counter](#)).
2. Neu hier ist, dass über if ((sw1_neu==0) & (sw1_alt==1)) die positive Flanke (=aufsteigende Flanke) erkannt wird und dies im Flag sw1_slope gespeichert wird.

```
unsigned char castVar = 1;
// Variable für Wuerfel-
Zaehler
```

```
bool timertick;
// Bit-Botschaft alle
0,111ms (Timer-Interrupt)
bool takt10ms;
// Bit-Botschaft alle 10ms
bool takt100ms;
// Bit-Botschaft alle 100ms
```

Anzeigefunktion Wuerfel
=====

```
bool sw1;
// Bitspeicher fuer Taste 1
bool sw4;
// Bitspeicher fuer Taste 4
bool castBit = 0;
// Flip-Flop-Bit fuer
Start/Stop
```

Initialisierung Display-Anzeige
=====

```
// Funktionsprototypen
void init_Taster(void);
// Taster initialisieren
void timerInt0(void);
// Init Zeitbasis mit Timer
0
void castCounting(void);
// Zaehlfunktion Wuerfel
```

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.

```
void castDisplay(void);
// Anzeige Wuerfel
void initDisplay(void);
// Initialisierung Display

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

    TCCR0A = 0;
// Timer 0 auf "Normal Mode"
schalten
    TCCR0B |= (1<<CS01);
// mit Prescaler /8
betreiben
    TIMSK0 |= (1<<TOIE0);
// Overflow-Interrupt
aktivieren

    sei();
// generell Interrupts
einschalten

    // Hauptprogrammschleife

    while(1)
// unendliche Warteschleife
    {
        if (takt10ms)
// alle 10ms:
        {
            takt10ms = 0;
//      Botschaft "10ms"
loeschen
            castCounting();
//      Tasten abfragen,
Wuerfel zaehlen

        }
        if (takt100ms)
// alle 100ms:
        {
```

```

        takt100ms = 0;
// Botschaft "100ms"
loeschen
        castDisplay();
// Wuerfelwert ausgeben
    }
}
return 0;
}

// Interrupt-Routine
=====
=====
==
ISR (TIMER0_OVF_vect)
/* In der Interrupt-Routine
sind die Softwareteiler für
die Taktbotschaften
    (10ms, 100ms) realisiert.
Die Interrupts stammen von
Timer 2 (Interrupt 1)

    Verwendete Variable:
vorteiler
hunderstel

    Ausgangsvariable:
takt10ms
takt100ms
*/
{
    timertick = 1;
// Botschaft 0,111ms senden
    --vorteiler;
// Vorteiler dekrementieren
    if (vorteiler==0)
// wenn 0 erreicht: 10ms
abgelaufen
    {
        vorteiler =
VORTEILER_WERT;    //
Vorteiler auf Startwert
        takt10ms = 1;
//    Botschaft 10ms senden
        --hundertstel;
//    Hunderstelzähler
dekrementieren

        if (hundertstel==0)
// wenn 0 erreicht: 100ms
abgelaufen

```

```
        {
            hundertstel =
HUNDERTSTEL_WERT; // Teiler
auf Startwert
            takt100ms = 1;
//      Botschaft 100ms 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
}

// Wuerfelfunktion
=====
=====
=====
void castCounting(void)
{
    // Einlesen der
Tastensignale
    sw1 = (PINB & (1 <<
PB1));
    sw4 = (PINB & (1 <<
PB4));
    // Auswertung der Tasten

    if (sw1==0)
// solange Taste 1
gedrueckt:
        castBit = 1;
//      Flip-Flop
"Wuerfeln" Setzen

    if (sw4==0)
// solange Taste 4
gedrueckt:
        castBit = 0;
//      Flip-Flop
```



```
"Wuerfeln" Ruecksetzen

    if (castBit)
// Solange Flip-Flop
"Wuerfeln" gesetzt
    {
        castVar++;
//      Würfelwert
hochzaehlen im Takt 10ms
        if (castVar>6)
//      groesser als 6?
            castVar=1;
//      => auf 1 setzen
    }
}

// Anzeigefunktion Wuerfel
=====
=====
void castDisplay(void)
{
    lcd_gotoxy(1,7);
// Cursor auf
Ausgabeposition
    lcd_putc(castVar+0x30);
// ASCII-Wert des
Wuerfelzaehlers ausgeben
}

// 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
6 -"); // Ausgabe Festtext:
16 Zeichen

    lcd_gotoxy(1,0);
// Cursor auf 2. Zeile, 1.
Zeichen
    lcd_putstr("Electronic
Cast "); // Ausgabe
```

```
Festtext: 16 Zeichen

    _delay_ms(2000);
// Wartezeit nach
Initialisierung

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

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

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.

From:
<https://wiki.mexle.org/> - **MEXLE Wiki**

Permanent link:
https://wiki.mexle.org/microcontrollertechnik/6_wuerfel_und_zufall?rev=1602555749

Last update: **2021/05/09 10:07**

