2 Sound und Timer

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

2. Sound und Timer

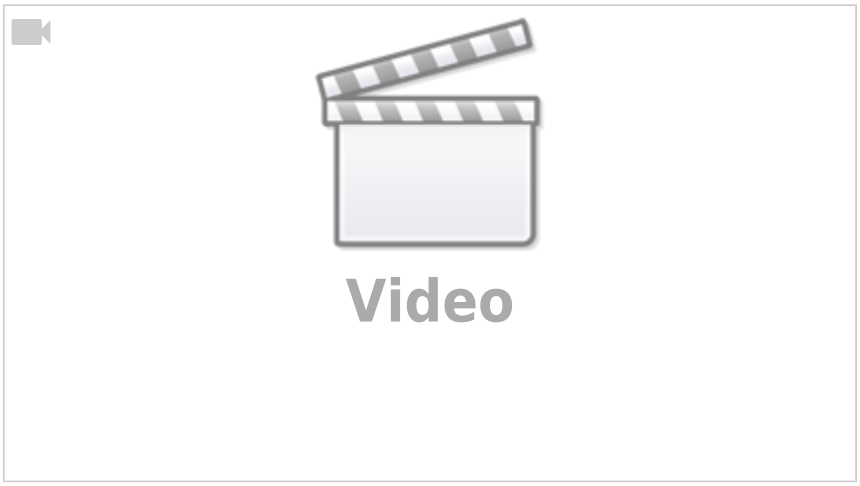
schnelles Takten für Anfänger

Ziele

Nach dieser Lektion sollten Sie:

1. wissen, wie man im Atmel Studio eine Puls-Signal ausgibt

Video



Übung

I. Vorarbeiten

1. Laden Sie folgende Datei herunter:
    1. 2_sound.simu
    2. 2_sound.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 `2_sound.simu`
    2. Laden Sie `2_sound.hex` als firmware auf den 328 Chip
2. Betrachtung der neuen Komponenten
    1. In der Simulation sind nun neben der LED weitere Komponenten zu sehen, die im Folgenden erklärt werden sollen
        1. ein Display Hd44780
        2. zwei Frequenzmesser, mit der Anzeige 0Hz
        3. ein Lautsprecher mit Schalter
        4. ein Oszilloskop
    2. Starten Sie die Simulation
        1. Wenn Sie die Lautsprecher des PCs aktiv haben, werden Sie ein Knacken (und ggf. Brummen) vernehmen. Dies rührt von der unvollständigen Simulation her. Wenn Sie den den Schalter vor dem Lautsprecher schließen, so sollten Sie - zusätzlich zum Knacken - einen aufsteigenden und abfallenden Ton hören. Es ist möglich, dass dieser nach wenigen Sekunden aufhört, auch das ist eine Eigenschaft der unvollständigen Simulation.
        2. Am Frequenzmesser ist dennoch zu sehen, dass ein Signal mit Frequenzen

zwischen $375...1600Hz$ ausgegeben wird.
3. Im Gegensatz dazu scheint das Oszilloskopbild still zu stehen und keine variierende Frequenz anzuzeigen. Um dies zu beheben, sollte am Oszilloskop der Haken bei Aut bzw. Auto entfernt werden. Nun ist eine Einstellung der x- und y-Werte über die Drehregler möglich.
4. Die LED zeigt nun an, ob es sich um einen Aufsteigenden oder abfallenden Ton handelt
5. Das Display zeigt zusätzlich an, welches Experiment geladen ist

3. Das Programm zu diesem Hexfile soll nun erstellt werden

II. Eingabe in Atmel Studio

1. öffnen Sie Atmel Studio
2. legen Sie ein neues “GCC C Executable Project” mit dem Namen `2_sound` für einen ATmega328 an
3. Bevor das eigentliche Coding beginnt sollte immer eine Beschreibung dem Code vorangestellt werden. Hierzu kann folgende Vorlage verwendet werden:

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

Experiment 2:   Sound-Generator
=============   ===============

Dateiname:      2_Sound.c

Autoren     :    Peter Blinzinger
                Prof. G. Gruhler     (Hochschule Heilbronn,
Fakultät T1)
                D. Chilachava        (Georgische Technische
Universität)

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:       Auf dem kleinen Lautsprecher des MiniMEXLE-Boards
(Buzzer)
                wird ein sirenenartiger Sound ausgegeben. Zwischen
den auf-
                und absteigenden Tönen bleibt die Frequenz kurz
stabil.
                Die Frequenz wird mit dem Timer 0 (im CTC-Mode)
erzeugt und
                direkt über den Output-Compare-Pin im Toggle-Mode
ausgegeben.
```

```
Displayanzeige: +----------------+
                |- Experiment 2 -|
                | Creating Sound |
                +----------------+

Tastenfunktion: keine

Jumperstellung: Schalter muss fuer den Buzzer betätigt sein

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

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

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

4. Ändern Sie die folgenden Informationen, je nach Programm:
    1. `Autor`: Der folgende Code basiert auf eine Version verschiedener Autoren, diese sind hier angegeben. Wenn Sie einen eigenen Code generieren sollte hier Ihr Name stehen. Dies ermöglicht eine Nachvollziehbarkeit bei Unklarheiten
    2. `Version`: Um die Aktualität der Software zu erkennen sollte mindestens das Datum angegeben und bei Änderung immer aktualisiert werden
    3. `Hardware`: Die getestete bzw. verwendete Hardware sollte angegeben werden, um nachvollziehen zu können, wie der Code getestet werden kann. Bei Simulationen sollte hier mindestens der verwendete Chipsatz (z.B. ATmega328) angegeben werden.
    4. `Software`: Zum weiterverwenden des Codes ist die Angabe der Entwicklungsumgebung (engl. integated Development environment IDE) wichtig. Nicht selten gibt es bei größeren Projekten Schwierigkeiten, wenn IDE und Compiler geändert werden.
    5. `Funktion`: Die Funktion des Programms sollte kurz erklärt werden. Damit wird dem Leser bereits vor dem Code schon Hinweise gegeben
    6. `Display-Anzeige`: Ähnlich der Funktion ist auch eine Darstellung der (erwartbaren) Anzeige sinnvoll.
    7. `Tastenfunktion`: Bei zukünftigen Anwendungen kann eine Eingabe von Tastenstellungen sinnvoll sein. Dies sollte hier angegeben werden
    8. `Jumperstellungen`: Jumper bieten die Möglichkeit unterschiedliche Schaltungsteile der Hardware zu verbinden oder zu trennen. Damit können Hardwarefunktionalitäten aktiviert oder deaktiviert werden. Wenn verschiedene Jumperstellungen für ein Programm wichtig sind, so sollten diese angegeben werden.
    9. `Fuses im uC`: Beim Microcontroller (auch µC oder uC abgekürzt) bieten die Möglichkeit interne Konfigurationen anzupassen. Diese werden über Fuses eingestellt werden. Sind hier Funktionen für das Programm notwendig, so sollten diese angegeben werden
    10. `Header-Files`: Werden weitere Softwareteile genutzt, so sollten diese über Header-Dateien eingebunden. Diese sollten bereits schon vor dem eigentlichen Codeteil kurz erklärt werden
5. Nach der Beschreibung steht im Code der Deklarationsbereich:

```
// 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 <util/delay.h>           // Definition von Delays
(Wartezeiten)
#include "lcd_lib_de.h"           // Funktionsbibliothek zum LCD-
Display

// Konstanten
#define MIN_PER     59            // minimale Periodendauer in
"Timerticks"
#define MAX_PER     255            // maximale Periodendauer in
"Timerticks"
#define WAIT_TIME   2000        // Wartezeit zwischen Flanken in
ms

// 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

// Funktionsprototypen
void initDisplay(void);         // Initialisierung Display und
Startanzeige
void initPorts(void);           // Initialisierung der I/O-Ports
void initTimer(void);           // Timer 0 initialisieren
(Soundgenerierung)
void init(void);                // generelle
Initialisierungsfunktion
```

1. Eingabe und Kompilieren des Code
    1. Ersetzen Sie den vorhandenen Code, durch den rechts stehenden Code
    2. Kompilieren Sie den Code durch `Build` » `Build solution`
    3. Im unteren Teil des Fensters sollte nun die Ausgabe des Kompilers sichtbar werden. Diese sollte `========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========` lauten
2. Auswählen der hex-Datei
    1. im Atmel Studio finden Sie rechts im Fenster den “Solution Explorer”
    2. gehen Sie dort im Solution Explorer zu `Solution` » `Einfuehrung_v01` » `Output`

`Files`
3. klicken Sie mit rechter Maustaste auf `Einfuehrung_v01.hex` und wählen Sie Pfad und Name aus

III. Ausführung in Simulide

1. Öffnen Sie SimulIDE (unter ...\bin\simulide.exe)
    1. links in SimulIDE sollten Sie den Komponenten Browser finden. Wählen Sie dort `Micro»AVR»atmega»atmega328`
    2. Ziehen Sie den Eintrag `atmega328` per Drag and Drop in den Arbeitsbereich (rechter, beiger Teil des Fensters)
    3. Es sollte nun ein Chip names `atmega328-1` dargestellt sein
2. Erstellen der Ausgangsschaltung
    1. Im Programm wurde im auf PortD das 6bit angesprochen. Entsprechend soll auch hier am Port D der Ausgang 6 genutzt werden. Am Chip ist dieser mit D6 gekennzeichnet
    2. Fügen Sie eine LED (im Komponenten Browser über Output LED) und ein Massepotential ein (Sources Ground)
    3. Die Komponenten können mit dem Kontextmenu (Rechtsklick) gedreht und gespiegelt werden. Außerdem ist mit der Auswahl von `Properties` im Kontextmenu die Änderung von
    4. Verbinden Sie die LED mit Masse und mit Port D6. Achten Sie auf die richtige Richtung der LED. Die Verbindungen lassen sich dadurch erstellen, dass auf ein Komponenten-Pin geklickt wird und die Linie zu einem nächsten Komponenten-Pin gezogen wird.
3. Flashen der Software
    1. Klicken Sie rechts auf den Microcontroller und wählen Sie `Load firmware`
    2. Fügen Sie hier den Pfad und Name des oben erstellten `Einfuehrung_v01.hex` ein und öffnen Sie dieses
4. Starten der Simulation
    1. klicken Sie im Menu den Power-on Button
    2. Die Simulation startet
5. Bugfixing
    1. vermutlich ist bei Ihnen zu sehen, dass die Diode nicht gleichmäßig an und aus dargestellt wird. Dies ist kein Fehler des Simulationsprogramms. Es wurde noch eine wichtige Komponente vergessen, welche immer bei der Verwendung von diskreten LEDs verwendet werden muss. Fügen Sie diese ein und Testen Sie die Schaltung nochmal

Sie sollten sich nach der Übung die ersten Kenntnisse mit dem Umgang der Umgebung angeeignet haben. Zum Festigen des der Fähigkeiten bieten sich folgende Aufgaben an:

Aufgaben

1. Welche Vorgaben für die Softwareentwicklung wurden verletzt, trotzdem das Programm lauffähig ist? (Interrupts werden erst in späteren Übungen erklärt)
2. Wie könnte ein Ampel-Licht-Abfolge oder Lauflicht aus 4 Dioden erstellt und programmiert werden? Welche Optimierungen könnten im Code vorgenommen werden? Welche Komponente in SimulIDE kann genutzt werden? Wie kann die Farbe der LEDs geändert

werden?

3. Lesen Sie auf Mikrocontroller.net im Kapitel Warteschleifen die “erste Seite”, also bis:

> Abhängig von der Version der Bibliothek verhalten sich die Bibliotheksfunktionen etwas unterschiedlich.