

# Tipps fürs Programmieren

## Student Group

First Name	Surname	Matrikel Nr.

## Table of Contents

- Tipps fürs Programmieren** ..... 2
- Software System Design* ..... 2
- Vorgaben für Bewertung* ..... 2
- Häufige Fehler und Debugging* ..... 2
- Allgemeines* ..... 3
- serielle Schnittstellen* ..... 3
- Programmierung des ST7565 im Display ERC 128 64 - 1* ..... 3
- Verwenden von Ports* ..... 4

# Tipps fürs Programmieren

- Sofern es sich um eine Aufgabe mit Hardware-Software-Co-Design handelt, kann mit der Erstellung der Software bzw. der Softwaresystem-Entwicklung bereits parallel zur Schaltplan-Erstellung bzw. direkt danach begonnen werden.
- Machen Sie sich dazu zunächst Gedanken,
  - **was** die Software tun muss (übergeordnete Tätigkeiten),
  - und in **welcher Abfolge** sie das tun soll
- Dann können Sie sich überlegen,
  - wie diese einzelnen Tätigkeiten **C-Funktionen** zugeordnet werden können,
  - wie die C-Funktionen von einander abhängen,
  - welche Schnittstellen die C-Funktionen untereinander benötigen, (Datentyp, Wertebereich, Name)
- **Erst dann sollten Sie überlegen wie der Code aussieht.** Hier hilft ein Blick in die Datenblätter und Application Notes der µController und Chips. Dort sind nicht selten Algorithmen oder Codeschnipsel vorgegeben.

## Software System Design

Falls Sie noch nicht wissen, wie die Software oder die zu verwendende Hardware genau genutzt wird, sollten folgende Tipps helfen:

- Suchen Sie die Komponente nicht (nur) auf deutsch. Die Ergebnisse vermehren sich um ein Vielfaches, wenn in Englisch gesucht wird. Zur Übersetzung empfiehlt sich [Linguee](#).
- Nutzen Sie auch die Bildersuche, wenn Sie den genauen Begriff nicht kennen.
- Bei unklaren Begriffen, bieten sich auch folgende Suchworte an: *Arduino* + *<englische Übersetzung von dem "Ding" was gesucht wird>* + ggf. *Project*. Alternativ bietet sich statt Arduino auch AVR oder Atmel an. Für C-Code kann ein "c code" zusätzlich gesucht werden. Code liegt häufig auf Github, entsprechend kann sich in Google ein Hinzufügen von `site:github.com` empfehlen.

## Vorgaben für Bewertung

Die [Vorgaben für die Programmierung](#) enthalten Hinweise, wie der abgegebene Code aussehen soll.

## Häufige Fehler und Debugging

- Tipps zum Debugging und zu häufigen Fehlern finden sich bei den [Tipps für die Fehlersuche \(Elektronik-Labor\)](#).
- Versuchen Sie möglichst nach jeder kleinen Änderung Ihr Programm zu testen. Wenn Sie drei Punkte ändern und dann erst testen, dann wissen Sie nicht, an welcher Änderung es liegt!
  - Die 25 häufigsten Fehler beim Programmieren

[oder als](#)

Paper

[und weitere](#)

41 häufige Fehler

## Allgemeines

- Eine schöne Einführung in die Embedded Softwareentwicklung ist im Buch [Sensornetzwerke in Theorie und Praxis - Embedded Systems-Projekte erfolgreich realisieren](#) von Kollegen Meroth und Sora zu finden. Dort wird der Einstieg in das Feld die (in Hardware) eingebettete Softwareentwicklung erklärt. Aus dem Hochschulnetz bzw. mit VPN können Sie dieses direkt bei Springer Link betrachten. Eine andere schöne Einführung findet sich auf [Mikrocontroller.net](#).
- Zum Programmieren muss die Hardware noch nicht vollständig sein. Wenn Sie einen Mikrocontroller der ATmega Familie nutzen wollen, so können Sie z.B. mit dem MiniMEXLE, MEXLE2020 oder Simulide bereits Software entwickeln und testen.
- Tipps für die  

Programmierung von ATMEL Chips
- Falls Sie lange Tabellen benötigen, sollten Sie die Daten im Programmspeicher (EEPROM) und nicht im Datenspeicher (SRAM) ablegen. In der Regel ist der Programmspeicher um den Faktor 5..10 größer
- Es gibt keine [If-Schleife!](#)
- Falls Sie im Netz nach Lösungen suchen, so beachten Sie, dass bei Arduinos in der Regel C++ (z.B. file.cpp) genutzt wird. Dies ist in den seltensten Fällen direkt kompatibel. Andererseits lassen sich aber die Konzepte übernehmen.
- `for(x = 0 ; x < 400 ; x++)` : Wenn x als 8 bit integer definiert ist, wird diese Schleife endlos lange laufen...
- Die Variablentypen sind bei Rechnungen zu beachten, sonst wird aus `c=a/b` mit `int a=5` und `int b=2` eine 2. Hier hilft ein expliziter Typecast: `c=(float)a/b`

## serielle Schnittstellen

- Die Programmierung eines AVR-Chips über USB (sofern dies der Chip ermöglicht), geschieht über das Tool [FLIP](#)
- Falls Sie einen externen Baustein über einen Mikrocontroller ansteuern wollen, ist folgendes zu beachten: Überprüfen Sie, ob der externe Baustein auf die positive Flanke triggert oder auf die negative. In der Regel lässt sich dies beim externen Baustein nicht ändern. Dies kann auf der Seite des Mikrocontrollers per Software geändert werden.
- Falls eine weitere I2C Schnittstelle benötigt wird, so finden sich [Vorlagen dazu im Netz](#).

## Programmierung des ST7565 im Display ERC 128 64 - 1

- Das Display  

ERC 128 64 - 1

mit 128 Pixel in x-Richtung und 64 in y-Richtung ist in 8×8 Teile unterteilt. Die 8×8 Pixel werden auch Page genannt. In Software sind 132×65 Pixel ansprechbar - die Ausgabe ist aber nur auf 128×64 Pixel.
- Je 8bit vertikal sind im  

ST7565

in einem Byte gespeichert.
- Die Kommandos, welche über SPI genutzt werden können, sind im Datasheet beschrieben.
- Über SPI kann nur auf das Display geschrieben werden. Ein Lesen ist nicht möglich.
- Lesen Sie den Beispiel-Code des Herstellers und suchen Sie ob es Tutorials für den ST7565 gibt.
- Beachten Sie die Einbaurichtung bei der Ansteuerung des Displays.

## Verwenden von Ports

- Folgendes ist zu beachten, falls Sie JTAG-Ports - z.B. PF4..7 bei ATMEGA16U4 - anderweitig verwenden wollen/müssen (JTAG Ports = Ports an denen die ProgrammierHW angeschlossen wird): Die JTAG-Ports können nicht ohne weiteres direkt genutzt werden. Die JTAG Schnittstelle muss zunächst über folgenden Code deaktiviert werden.

```
MCUCR |= (1<<JTD);  
MCUCR |= (1<<JTD);
```

Wichtig: Es das Control Register muss zweimal geschrieben werden.

- Für eine zeitkritische Ausgabe von aufeinanderfolgenden Bits (z.B. für die Ansteuerung von [intelligenten LEDs](#)) müssen unbedingt Interrupts genutzt werden. Es lohnt sich zusätzlich auf [USART](#) zurückzugreifen. Bei USART werden die zu versendenden Daten zunächst ins UDRn Register gegeben und dann in das Shiftregister übertragen.
- Falls Sie einen externen Oszillator oder Quarz benutzen, werden zwei Ports dafür verwendet (Ports XTAL = "Crystal"). Wenn Sie diese Ports per DDR versehentlich zu einem Ausgang definieren, hat der Chip keinen Takt mehr. Das heißt diese Portzuweisung ist das letzte was der Chip macht... Es ist danach nur noch per Debuggerschnittstelle möglich diesen wieder zu beleben.

From:

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

Permanent link:

[https://wiki.mexle.org/microcontrollertechnik/tipps\\_fuers\\_programmieren?rev=1620551684](https://wiki.mexle.org/microcontrollertechnik/tipps_fuers_programmieren?rev=1620551684)

Last update: **2021/05/09 11:14**

