Debugging und Bugfixing - Tipps für die Fehlersuche

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

Debugging und Bugfixing - Tipps für die Fehlersuche

Allgemein

Um die Fehlerursachen zu finden, empfiehlt sich folgendes, generisches Vorgehen:

1. Verstehen
    1.1 Definieren des “Gut-Falls”: Wie wäre zu erkennen, dass das Programm korrekt läuft?

    1.2 Versuchen Sie genau zu verstehen, unter welchen Umständen der fehlerhafte Zustand auftritt.
    Wie ist dieser zu reproduzieren? Ohne Reproduzierbarkeit ist ein Fehler nicht auffindbar!

    - Welche Eingaben, Spannungen waren vorhanden?
    - Gleicher Fehler bei geänderten Eingaben?
    - Prüfen Sie auch vermeintlich klare Dinge

    Beim Verständnisaufbau hilft manchmal weniger Theorie und mehr ausprobieren.

    Wichtig: Schreiben Sie auf, was funktioniert und was nicht (am besten mit Ablegen des verwendeten Codes). Dann lässt sich auch nach einigen Tests nachvollziehen, was womit getan wurde bzw. nicht funktionierte.

    Und: Ein Fehler der auf mysteriöse Weise wieder verschwindet, kommt genauso wieder (i.d.R. dann, wenn keine Zeit mehr da ist). Ein Fehler der nicht korrigiert wurde, wurde nicht korrigiert..

    1.3 War die Teilfunktion schon jemals lauffähig?
        Falls die Funktion an anderer Stelle bereits lauffähig war, dann Ja.

        1.3.1 Ja, Teilfunktion war schon lauffähig
            Gehe zu 2.

        1.3.2 Nein, Teilfunktion war noch nie lauffähig

            1.3.2.1 Suchen
                Gab es weltweit noch keinen, der diese Teilfunktion implementiert hat? Um das zu beantworten hilft ein Blick in Google.

                Lernen Sie besser_suchen_mit_google

                Falls es jemanden gab, so war die Funktion an anderer Stelle bereits lauffähig --> Gehe zu 2. .

            1.3.2.2 Reduzieren
                Ähnlich Punkt 2. und 3. ist es zielführend Das System soweit zu vereinfachen, dass es lauffähig ist, auch wenn die Funktion darunter leidet.

D.h. im Extremfall das Flashen mit einer leeren main Funktion (Geht das Flashen?) oder mit einer, welche nur einen PIN / LED aktiviert.

1.3.2.3 Kreativ sein

Falls die Teilfunktion noch nirgends implementiert wurde (wirklich?), dann muss etwas mehr Kreativität genutzt werden.

2. Vermuten

Erstelle eine Hypothese an was der Fehler liegen kann. Dann ist das (fehlerhafte) System leichter über “Teile und Herrsche” zu bändigen.

3. Aufteilen

Breche das System (Hardware und Softwarekomponenten) und die Interaktionen (Schnittstellen, Funktionsaufrufe) auf ein Minimum herunter.

Alternativ: Breche das System in Teile und prüfe ob die Einzelteile fehlerfrei sind.

Falls es schon einen Teil des Programms gab, so sollte dieser wieder hergestellt werden. Die Änderungen sollten dann im folgenden Schritt Zeile für Zeile (bzw. Funktionsblock für Funktionsblock) eingefügt und auf der Hardware auf getestet werden.

4. Ersetzen

Ersetze Systeme (Hardware und Softwarekomponenten) und die Interaktionen (Schnittstellen, Funktionsaufrufe) durch einfachere.

1. Vereinfache die Software
    1. Kompiliert der Beispiel-Code aus der Vorlesung?
       Kompiliert das gewünschte C-File, wenn alles (includes, variablen, functions), außer main() ausgeblendet wird?
    2. Kompiliert eine **einfache Software**, welche eine (wechselnde) Ausgabe beinhaltet (z.B. LED, Port)?
       Besser einfach beginnen, also nicht mit einem über SPI angeschlossendem Display.
2. Vereinfache die Hardware
    1. Nutze Oszilloskop und Logic Analyzer statt ausgebende Komponenten.
    2. Nutze Funktionsgenerator statt eingebende Komponenten.
    3. Nutze bereits getestete Komponenten.
3. Vereinfache die Interaktion

5. Beobachten und zurück zu Start

Tritt der Fehler exakt gleich auf?
Welche neuen Hypothesen lassen sich aufstellen?

Häufig ist es von Vorteil eine Ausgabemöglichkeit zu schaffen.Zur Ausgabe bietet sich z.B. ein

unbenutzter PIN oder - falls schon vorhanden und in Software ansprechbar - ein Display an.

1. Die Ausgabemöglichkeit kann als genutzt werden, um den Programmablauf zu überprüfen. z.B. kann die Ausgabe eines Buchstabens auf dem Display als Zeile eingefügt und so das Erreichen dieser Zeile überprüft werden. Beispiel

```
...
void main()
{
  initLCD();
  initADC();

  while(1)
  {
     doThis();
     if (something)
     {
        LCDprint('s',1,1);
        doThat();
     }
  };
}
```

2. Ähnliches geht auch beim Sprung in ein Unterprogramm. Wird dieses aber mehrmals in der Hauptschleife aufgerufen, kann es sich anbieten eine Hilfsvariable einzufügen. Damit ist es möglich nur den Sprung beim vermuteten Fehlverhalten zu betrachten: Beispiel

```
...
void main()
{
  int dummy=0;
  ...
  while(1)
  {
     doThis();
     if (something)
     {
        dummy=1;
        doThis();
        dummy=0;
     }
  };
}

void doThis()
{
   ...
   SomeCode;
   if(dummy==1) LCDprint('s',1,1);
   ...
```

```
}
```

3. Falls sich das Unterprogramm in einer weiteren Datei (z.B. eingebundene Library) befindet, so muss die Hilfsvariable übergeben werden. Temporär ist dafür die Definition/Deklaration einer externen Variable sinnvoll. Beispiel: Es wurde die Datei ADC.h inkludiert. Aus main() wird setADCgain() aufgerufen. In dieser Funktion wird ein Fehler erwartet. Sinnvoll ist es nun die Variable dummy in ADC.h zu deklarieren (extern int dummy;) und in main.c zu definieren (int dummy=0;). Dann kann die Variable in ADC.C auch ohne weitere Definition/Deklaration verwendet werden. Näheres dazu auch auf Wikipedia. Beispiel
main.c

```
...
#include ADC.h

int dummy=0;
void main()
{
  ...
  while(1)
  {
     setADCgain();
     ...
     if (something)
     {
        dummy=1;
        setADCgain();
        dummy=0;
     }
  };
}
```

ADC.h

```
...
extern int dummy;
```

ADC.c

```
...
void setADCgain()
{
   ...
   SomeCode;
   if(dummy==1) LCDprint('s',1,1);
   ...
}
```

6. Bugfixen
   Sobald der Bug eingekreist wurde, geht es ans korrigieren. Hier hilft meist die Theorie weiter.

   Wichtig: Prüfen Sie, ob der Fehler noch auftritt. Ist der Fehler auch bei Variation der Eingaben weg?

Software

- Compiliert der Code?
  - Ist die richtige Zielhardware (z.B. ATmega328PB) gewählt?
  - Sind bei den eingebundenen Bibliotheken die Randbedingungen der Zielhardware berücksichtigt? (z.B. hat der ATmega328PB andere Benamung der I2C und SPI Register, da diese mehrfach vorhanden sind)
- Wichtig ist, dass Sie zunächst eine Ausgabemöglichkeit schaffen. Die kann ein digitaler (oder analoger) Output Pin oder eine LCD-Anzeige sein. Ohne diese ist eine strukturierte Fehlersuche schwer möglich. Versuchen Sie die Ausgabemöglichkeit als erstes zu programmieren und zu testen.
- Wenn Sie zur Fehlersuche Ihr Programm ändern, machen Sie eine Sicherungskopie des Programms. Weiterhin sollten Sie die zur Fehlersuche geänderten Zeilen markieren (z.B. mit dem Kommentar DEBUGGING!)
- Falls das Programm nicht die von Ihnen gewünschte Ergebnisse liefert, dann versuchen Sie zunächst möglichst weiträumig Code auszukommentieren. Falls dann das Programm noch einen sinnvollen Ablauf zeigt, kann Schritt für Schritt die Auskommentierung aufgehoben werden.
- Nutzen Sie die Möglichkeit nach Zwischenschritten eine Wertänderung auszugeben (z.B. Ausgabe am LCD). Damit kann ermittelt werden ab welcher Zeile der tatsächliche Ablauf vom gewünschten abweicht.
- Prüfen Sie mit Hilfe der AVR Checkliste nach bekannten Fehlerquellen.
- Versuchen Sie den Code so umzuschreiben, dass dieser durch einem Online Compiler (z.B. OnlineGDB) testbar wird.
- Falls Sie dann immer noch nichts finden: Schreiben Sie eine Frage an den Betreuer. Achten Sie dabei auf das richtige Stellen von Fragen

Hardware

- Versuchen Sie bei der Fehlersuche Schritt für Schritt vorzugehen.
- Ist die Stromversorgung an und richtig eingestellt? (z.B. Strombegrenzung nicht auf 0A, kein electric Fuse aktiv)
- Falls die Spannung zusammenbricht oder das “F” für Fuse aktiv ist, spricht dies für einen Kurzschluss der Versorgung gegen Masse. Wenn Sie gesockelte Chips haben, können diese entfernt und geprüft werden, ob der Fehler noch anliegt.
- Sind Messgeräte und Stromversorgung an und richtig eingestellt (z.B. Abschlusswiderstand)?
- Passt die Verkabelung? Ist Ground korrekt angeschlossen?
- Sind die passenden Widerstände an der richtigen Position? Und sind die Dioden in richtiger Richtung bestückt?
- Können Spannungen als Zwischenwerte gemessen werden?
- Sind die Pins richtig angeschlossen? Z.B. Spannungsversorgung am OPV
- Funktioniert das Flashen über das Proggi? Falls nicht:
  - Ist der korrekte USB-Anschluss gewählt?
  - Ist der Fehler auch mit einem anderen Proggi + anderen Kabeln vorhanden?

    - Ist in der Schaltung die SPI-Schnittstelle in Verwendung? Sind weitere Bauteile zwischen Programmierbuchse und Controllerpin, bzw. zwischen Programmierbuchse und anderen Spannungen (wie GND)?
- Falls Sie einen Fehler in der Kommunikation vermuten: verbinden Sie RX mit TX und überprüfen Sie, ob am gleichen Bauteil das korrekte Signal ankommt.

Häufige Fehler

- **F_CPU not defined for** (z.B. <util/delay.h>) Das beste ist die Frequenz F_CPU im AVR Studio direkt anzugeben:
    - Gehe zu Menu: `Projekt » (ProjektName) Eigenschaften » Toolchain » AVR/GNU C Compiler » Symbols`
    - Füge F_CPU=8000000 (bzw. Passende Frequenz) ein
- **Das Programm kompiliert nicht TWSR not found** : Falls Sie einen modernen AVR Chip nutzen (z.B. 328PB) so kann dieser mehrere SPI und I2C Schnittstellen haben. Damit haben sich bei diesem Target auch die Register- und Interruptvektornamen geändert. Statt TWSR ist dann TWSR0 oder TSWR1 zu verwenden - je nach gewünschtem Pin. Dies ist am einfachsten über defines der fehlerhaften Namen, also `#define TWSR TWSR0` usw.
- **Beim Flashen der realen Hardware über `Tools » Device Programming` finde ich im Tool nur "Simulation", aber kein STK500.** Versuchen Sie zunächst über Tools » Add tagret... STK500 und den entsprechenden Serial Port zu wählen. Falls Ihr Rechner mehrere USB Ausgänge hat, müssen Sie diese (COM1...COMx) beim Programmieren ausprobieren.
- **Beim Flashen der realen Hardware erhalte ich "Erasing device failed", "Error status received: Got 0xc9, expected 0x00 (An unknown command was sent)"**.
    - Steht bei Device Programming das Interface auf ISP? Falls nicht kann dies die Ursache sein. Das Programming geschieht immer mittels ISP.
    - Hat das USB-Kabel/Progi/Adapterplatine/Kabel ein Problem? Probieren Sie eine andere Variante der Komponenten durch
- **Mein Chip hat keinen Speicherplatz mehr** bzw **Ich erhalte ein 'Memory Overflow' Fehler** Falls Sie Daten statt im SRAM im EEPROM speichern wollen, so können Sie das Befehlswort "PROGMEM" nutzen. Details dazu finden Sie z.B. auf der Seite von Microchip
- **Mein Programm scheint irgendwo nicht weiter zu kommen**. Dies kann verschiedene Gründe haben:
    - Endlosschleife
    - Speicherüberlauf im RAM: sobald die Speicherauslastung des RAM über ca 75% steigt, sind Probleme wie spontane Resets bei Bearbeiten von Pointern, Arrays, Strings oder Structs wahrscheinlich. Die kann über Debugging herausgefunden werden (entweder mit Steppen mit Debugger oder Ausgabe von Werten nach jeder Zeile).

I2C

- **Auf den I2C Leitungen ändert sich nichts, obwohl der IC etwas ausgeben sollte:**
    1. Überprüfen Sie die Pullup-Widerstände: Sind welche verbaut? Welche Größe haben diese? (typisch: 10kOhm). Wenn keine Verbaut sind, so wechselt das Signal nur zwischen 0V niederohmig und 0V hochohmig. Dies ist am Oszilloskop nicht zu unterscheiden.
    2. Ist ein hochohmiger Widerstand R_L entlang der Leitungen verbaut? Falls ja erzeugt dieser einen Spannungsteiler mit dem Pullup-Widerstand. Wenn R_L groß ist, so liegt zwischen R_L und Pull-up fast die Versorgungsspannung an.
- **Der Master soll Daten vom Slave empfangen, aber hängt sich manchmal auf** Im "Master Receiver Mode" muss der Master das Ende der Kommunikation dem Slave mitteilen. Dazu muss beim Lesen der Daten TWEA = 0 gesetzt werden. Ansonsten kann es sein, dass der

Slave meint er müsse noch Daten senden. Das kann unter Umständen dazu führen, dass der Slave die Datenleitung SDA am Ende der Kommunikation auf Low legt und damit den I2C stört.

Tipps für die Fehlerkorrektur (Bugfixing)

- Bei größeren (Serien)Projekten steht einer einfachen Elektronik-Fehlerkorrektur häufig die komplexe Validierung und Tests der Hardware im Weg. Entsprechend kann es sich anbieten die Fehler über ein Mod-Board - also einem kleinen Zusatzboard - zu korrigieren. Dafür bietet sich beispielsweise ein kompakter Chip, wie der 6 Pin SOT PIC10F206 oder der Attiny10 an.