Tipps fürs Programmieren

Student Group

First Name	Surname	Matrikel Nr.

Table of Contents

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 bereits Software entwickeln und testen.
- 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
- 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 Sotware 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.

Umgang mit Fehlern

Programmier-Fehler

- “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
- **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.

Debugging

- Zum Auffinden von Fehlern gibt es verschiedene Vorgehen. Folgendes sehe ich als sinnvoll und zielführend an:
    1. Definieren des “Gut-Falls”: Wie wäre zu erkennen, dass das Programm korrekt läuft?
    2. Definieren des Ausgangszustands: Gab es schon ein Teil des Programms, welcher korrekt lief?
    3. Falls es schon einen Teil des Programms gab, so sollte dieser wieder hergestellt werden. Die Änderungen sollten dann Zeile für Zeile (bzw. Funktionsblock für Funktionsblock) eingefügt und auf der Hardware auf getestet werden.
    4. Zur Ausgabe bietet es sich an z.B. ein unbenutzten PIN oder - falls schon vorhanden und in Software ansprechbar - ein Display zu nutzen.
    5. 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();
     }
  };
}
```

    6. 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);
   ...
}
```

7. 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 weiter 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);
   ...
}
```