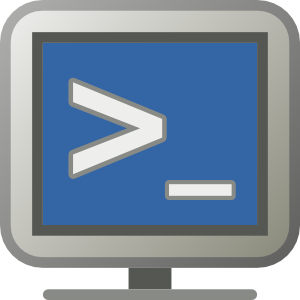
****

**AVR-CMD >Dokumentation**

*by Me & You*

**Inhalt**

[Software](#_9sdr5t514utf)

[Geplant](#_wt1l0giy74st)

[Anforderungen](#_s6kelkeucon7)

[Einstellungen](#_8gkjhmdxn0xh)

[Variablen und Parameter](#_e9aw8jxyygc)

[Werte](#_gazjye6sqvpn)

[Variablen](#_gfqqdn5gyd69)

[Port Register](#_yg9qkvh6ifa3)

[Datei Argumente](#_4xx3g5bayom6)

[Strings](#_330eoedgzguq)

[Dateinamen](#_n45cj0jx9lq6)

[Befehle](#_qoxpovz2z50x)

[Allgemein](#_xg700so9kq7i)

[delay x y](#_jg7ffqh0dqjx)

[print x (y)](#_dyonr3d44t6c)

[Mathematik](#_n3vpxcjtvtf4)

[Dateien](#_ebprcpao6wie)

[show x](#_kuao1wmm2k7w)

[open x](#_fdruj0kujgwe)

[exec x (y0 ... yn)](#_ehx6oll5g27e)

[del x](#_n1bvnia7tdqm)

[new x](#_4gfpqlr34u33)

[Schleifen und Bedingungen](#_lcuf68p4342q)

[loop x … endloop](#_qfdc4uyad088)

[while x op y … endwhile](#_6owisswjut8w)

[if x op y … else … endif](#_f82bv0shrzum)

[Hardware Planning](#_hkep7pxwkp4g)

[Aufbau](#_pvnyq5s480rf)

[Feature List](#_ecey6e87zukl)

[Anhang](#_9zmwqs7bfksl)

[SD Tests](#_khs1vgjbcfc3)

[http://elm-chan.org/fsw/ff/00index\_e.html](#_ut34opu2rtfo)

[http://www.holger-klabunde.de/avr/avrboard.htm#FullFAT](#_ow2qv3u8jo81)

[http://www.ulrichradig.de/home/index.php/avr/eth\_m32\_ex](#_svcohnx9a1jw)

# Software

## Geplant

* i/o abstrahieren (uart / lcd / telnet etc.)
* SD/FAT Lib wählen (siehe [3.1.](#_khs1vgjbcfc3))
* help
* script editor
* adc use
* multitasking
* i2c / SPI / UART für User nutzbar machen

## Anforderungen

Development Board ist ein Arduino Mega 2560. Jedoch wird nur der Bootloader genutzt. Der Code ist in C geschrieben.

Als Entwicklungsumgebung dient Eclipse mit AVR-Plugin und WINAVR.

Terminal-Client (z.B. Putty, [http://gidf.de/putty](http://www.gidf.de/putty))

Anschluss für SD-Karte (an SPI des Controllers)

SD(HC)-Karte mit FAT16/32

## Einstellungen

In folgenden Dateien müssen Einstellungen vorgenommen werden:

*config.h*

*sd/mmc\_config.h*

Alle Optionen sind in den Kommentaren in den Dateien erklärt.

## Variablen und Parameter

### Werte

Grundsätzlich wird mit *uint16* gerechnet. Somit ist 65535 der maximale Wert für Eingaben, Variablen, etc. Es wird folglich auch nur mit positiven Zahlen gerechnet. Deswegen ist auf den Überlauf zu achten.

Eingabe von Werten in folgendem Format möglich:

* + - * Ziffern *0-9* = dezimal
      * *0xhhhh* = hexadezimal. Bsp.: *0xB00B*
      * *0bxxxx xxxx xxxx xxxx* = binär (ohne Leerzeichen). Bsp.: *0b1010010110100101*
      * *0oxxx xxx* = oktal. Bsp.: *0o177777*
      * *on* = 1
      * *out* = 1
      * *off* = 0
      * *in* = 0

### Variablen

* + - * werden mit $x angesprochen
      * x ist eine Zahl von 0 bis VA\_BUF (siehe [1.3.](#_8gkjhmdxn0xh)). Bsp: *$5, $23*

### Port Register

* + - werden mit xy(z) angesprochen. Bsp.: *DA1, PB*
    - x = D (DDR), P (PIN), O (PORT)
    - y = A-D (Port Bezeichnung)
    - z = 0-7 (Pin des Ports, optional)

### Datei Argumente

* nur bei Ausführung von Dateien verfügbar
* werden mit @x angesprochen
* x ist eine Zahl zwischen 0 und (VA\_BUF-2) (siehe [1.3.](#_8gkjhmdxn0xh)). Bsp.: *@5, @23*

### Strings

* müssen in “” stehen wenn sie Leerzeichen enthalten
* können ohne “” angegeben werden, wenn sie keine Leerzeichen enthalten
* Escape für “ ist \”. Bsp.: *“Michael \”AIR\” Jordan”*

### Dateinamen

* dürfen keine Leerzeichen enthalten
* kein LFN-Support (bis jetzt)

# 

# 

# Befehle

## Allgemein

### delay x y

Verzögerung für eine bestimmte Zeit.

x = m:milliseconds, u:microseconds

y = value

Bsp.: *delay m 10*

*delay u $1*

*delay m PB*

*delay u @1*

### print x (y)

Option 1: x = Variable

y = {x,b,o,i}:Basis

Option 2: x = Variable oder String, Inhalt der Variablen wird als String ausgegeben

Bsp.: *print $1 x*

*print @1 i*

*print “Hallo Welt!”*

*print @2*

## Mathematik

***cmd x y***

x = Eine Variable $x, dort wird auch das Ergebnis der Operation abgelegt

y = Ein Wert, eine Variable, ein Port Register oder ein Datei Argument

cmd = eines der folgenden Kommandos:

* *add*, addiert y zu x
* *sub*, subtrahiert y von x
* *mul*, multipliziert x mit y
* *div*, dividiert x durch y
* *mod*, Rest von x durch y
* *or*, bitweises ODER von x und y
* *xor*, bitweises EXCLUSIV ODER von x und y
* *and*, bitweises UND von x und y
* *shr*, shift x um y nach rechts
* *shl*, shift x um y nach links

Bsp.: *add $1 10*

*sub $2 $1*

*mod $2 @3*

## Dateien

### show x

Gibt den Inhalt einer Datei aus

x = Dateiname

Bsp.: *show test.txt*

### open x

Öffnet eine Datei zum beschreiben. Zunächst wird der momentane Inhalt ausgegeben. Die Eingabe erfolgt dann Zeilenweise und wird mit [RETURN] bestätigt. Die Zeilen werden an das Ende der Datei geschrieben. Bearbeiten von bereits eingegebenen Daten nicht möglich.

x= Dateiname

Beenden durch Eingabe von ‘*%exit*’ (ohne Anführungszeichen).

Bsp.: *open test.txt*

*set pb7 on*

*delay m 500*

*set pb7 off*

*>*

*…*

*> %exit*

### exec x (y0 ... yn)

Führt eine Datei aus. Die Datei wird zeilenweise Ausgeführt.

x = Dateiname

yn = Argumente auf die innerhalb der Datei mit @x zugegriffen werden kann

Bsp.: *exec test.txt*

*exec test.txt 10 0xA3 “Hallo Welt” $1 PB*

### del x

Löscht eine Datei.

x = Dateiname

Bsp.: *del test.txt*

### new x

Erstellt eine Datei.

x = Dateiname

Bsp.: *new test.txt*

## Schleifen und Bedingungen

### loop x … endloop

Führt eine Schleife x-mal aus. Die Schleife beginnt mit der ersten Zeile nach *loop* und endet bei *endloop*. Ist x eine Variable oder ein Datei Argument können diese innerhalb der Schleife geändert werden. DON’T USE LOOPS IN LOOPS!

x = Value

Bsp.: *Loop 5*

*set pb7 on*

*delay m 500*

*set pb7 off*

*delay m 500*

*endloop*

*loop @1 ...*

*loop 0xA3 ...*

### while x op y … endwhile

Führt eine Schleife aus, solange die Bedingung x op y TRUE ist. Die Schleife beginnt mit der while-Zeile und endet bei *endWhile*. Sind weder x noch y Variablen, oder wird nicht mindestens eine der Variablen in der Schleife verändert entsteht eine Endlosschleife. DON’T USE WHILE IN WHILE!

x,y = Value

op = Operand: < , > , = , !

Bsp.: *while $0 > 0*

*print $0 i*

*sub $0 1*

*endwhile*

### if x op y … else … endif

Prüft ob die Bedingung x op y TRUE ist. Ist dies nicht der Fall, werden die folgenden Zeilen bis zu einem *endif* (oder *else*, wenn vorhanden) ignoriert. Ist die Bedingung wahr, werden alle Zeilen zwischen *else* und *endif* ignoriert, falls *else* vorhanden. *If* in *if* ist erlaubt.

x,y = Value

op = Operand: < , > , = , !

Bsp.: *if $0 > 0*

*print “True”*

*else*

*print “False”*

*endif*

# Hardware Planning

## Aufbau

Es wird das Arduino Shield System benutzt. Diese können einfach gestapelt werden. Bisher sind folgende Ebenen vorgesehen.

(1) Base Board Arduino Mega 2560

(2) Shield mit der Peripherie. Alle(?) Kontakte sind für LCD und weitere Shields weiter nach oben durchgeführt.

(3) LCD Shield bildet den Abschluss.



## Feature List

grün = gesetzt, rot = gesetzt für Shield v1

* + **Mega2560 (als Entwicklungsplattform gesetzt)**
  + SD-Card (in LCD integriert)
  + ETH mit ENC28J60 (oder die Version vom original arduino shield?) alternativ Wifi (C3000 von TI oder elecric imp) zweiteres hat den Vorteil, dass als SD einfacher zu handlen und interne SW entlastet den mega2850. Allgemein find ich Wifi besser
  + GLCD

<http://www.ebay.de/itm/SainSmart-3-2-TFT-LCD-Modul-Touch-Panel-PCB-adapter-SD-Reader-fur-Arduino-2560-/320934310788?pt=Bauteile&hash=item4ab92cf384>

* + Touch (in LCD integriert)
  + KEYBOARD: PS2/USB (USB würde auch mit Adapter gehen, dann aber keine USB Geräte außer Keyboard), Alternativ USB-Host-Controller oder SoftUSB auf Chip, Bluetooth?
  + Akku (Ladeschaltung, Anschlüsse für LiPo vorsehen)
  + zusätzlicher RAM (wird wahrscheinlich später nötig, denke das kann bis zur Rev1 warten)
  + Audio? (Mikro/Ohrhörer. MP3 Hardware-Codec)
  + GPS?
  + Funkmodul für verbindung untereinander? (entfällt bei wifi)
  + AD-Wandler für OSZI-Funktion
  + Logicanalizer 8Bit

# Anhang

## SD Tests

### <http://elm-chan.org/fsw/ff/00index_e.html>

Mit Multi File Unterstützg. Noch nicht getestet.

### <http://www.holger-klabunde.de/avr/avrboard.htm#FullFAT>

Mit Multi File Unterstützung. Liest Karten zuverlässig.

Erste Tests gemacht. SD Karte ist eine Toshiba SD M512. Formatiert mit FAt16. Bis jetzt nur Geschwindigkeitstest für single SD:

*1MiB schreiben/lesen, erste Zahl gibt Buffer-Größe an*

*Start writing files*

*512 7.41s 138.1kB/s*

*Start reading files*

*512 2.23s 459.1kB/s*

*Start writing files*

*128 7.55s 135.6kB/s*

*Start reading files*

*128 2.37s 432.0kB/s*

*Start writing files*

*64 7.76s 131.9kB/s*

*Start reading files*

*64 2.55s 401.5kB/s*

*Start writing files*

*32 8.14s 125.7kB/s*

*Start reading files*

*32 2.92s 350.6kB/s*

Eine SanDisk 32GB SD FAT32 funktioniert auch. Billig-Ding aus China (2€). Wahrscheinlich Ausschuss-Ware, die Qualitätstests nicht bestanden hat. Sind mehr als ca. 2GB geschrieben, geht anscheinend der FAT kaputt. Zeiten ähnlich (lesen war langsamer).

### <http://www.ulrichradig.de/home/index.php/avr/eth_m32_ex>

Wird im Moment verwendet. Läuft unzuverlässig. Karte wird häufig nicht erkannt. Kein Multi File Support. Soll durch [3.1.1.](#_ut34opu2rtfo) oder [3.1.2.](#_e2igj8ogql8p) ersetzt werden.