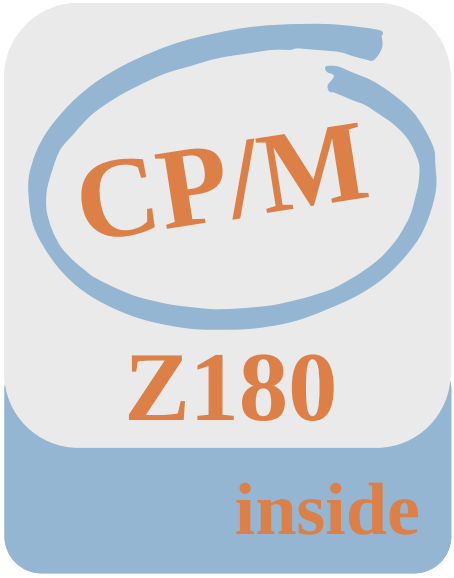
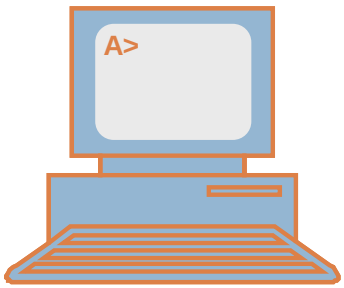
RETROCOMPUTING

Z180 - STAMP
AVR - STAMP



Joe G.

Dokumentation



ÄNDERUNGSVERZEICHNIS

Änderung			geänderte Kapitel	Beschreibung	Autor	neuer Zustand
Nr.	Datum	Version				
1	12.09.2014	V01.00		Startversion	Joe G.	OK
2	26.09.2014	V01.01	Z180	Versorgungsspannungen am Z180-Stamp Modul, (Pin2, Pin3)	Joe G.	OK
3	28.09.2014	V01.02	Z180	Versorgungsspannungen am Z180-Stamp Modul, (Pin2, Pin3)	Joe G.	OK
4	22.12.2014	V01.03	Inbetriebnahme	Monitorbeschreibung	Joe G.	OK

Retrocomputing

STROMVERSORGUNG

Sowohl das AVR-Stamp als auch das Z180-Stamp Modul besitzen bis auf wenige Ausnahmen ein identisch belegtes Anschlusssystem. Über dieses Anschlusssystem werden auch alle benötigten Versorgungsspannungen geführt. Wird eine ausschließlich externe Spannungsversorgung gewählt, reicht die Beschaltung der entsprechenden Versorgungspins. Zu Debug- oder Entwicklungszwecken können jedoch auch beide Module über alternative Beschaltungen versorgt werden. Diese Varianten werden im Folgenden kurz skizziert.

AVR-Stamp

Das AVR-Stamp Modul benötigt ausschließlich 3.3V. Die auf dem Modul zusätzlich vorhandene 5V Leitung wird nur über das Stecksystem durchgeschleift. Weiterhin kann diese 5V Leitung über die auf dem AVR-Modul vorhandene USB-Buchse mit 5V versorgt werden. Wird JP1 geschlossen, liegt die 5V USB-Spannung gleichzeitig auf den Anschluss B2 (*Achtung! Das Stamp-Modul wird damit noch nicht mit 3.3V versorgt*). Weiterhin sollte die maximale Belastbarkeit der USB-Schnittstelle beachtet werden.

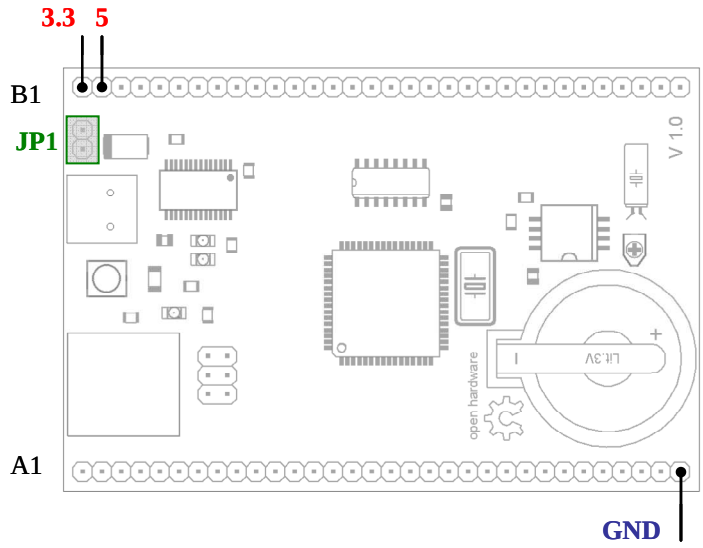


Abb.1: Versorgungsspannungen am AVR-Stamp Modul

Z180-Stamp

Das Z180-Stamp Modul besitzt neben den schon erwähnten Versorgungspins der Anschlussleisten A und B noch zusätzliche Versorgungsvarianten.

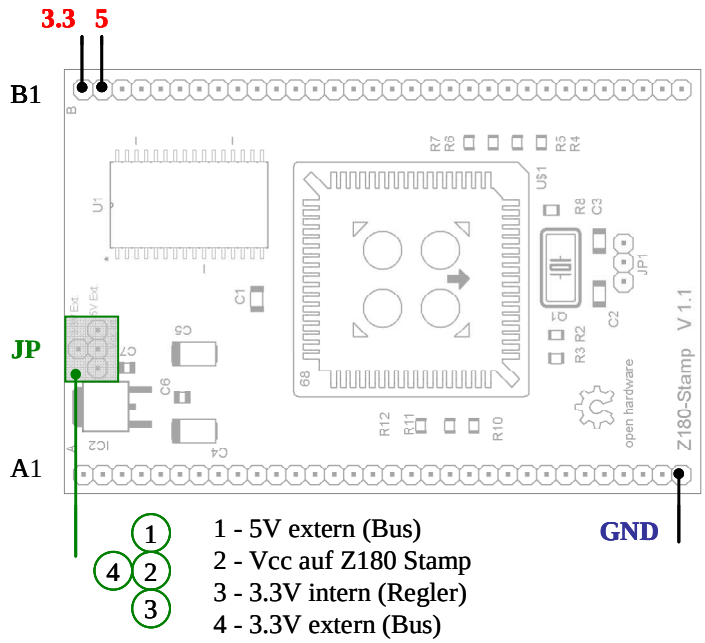


Abb.2: Versorgungsspannungen am Z180-Stamp Modul

Damit sind die folgenden Versorgungsvarianten möglich:

A – 3.3V Spanungsversorgung von Außen

Die Versorgungsspannung von 3.3V wird über das äußere Stecksystem an Pin B1 angelegt. Damit der Z180 versorgt wird, sind die Kontakte 2 und 4 zu schließen. Der interne Regler wird nicht benötigt.

B – 5V Spanungsversorgung von Außen, 5V intern

Die Versorgungsspannung von 5V wird über das äußere Stecksystem an Pin B2 angelegt. Damit der Z180 versorgt wird, sind die Kontakte 1 und 2 zu schließen. Der interne Regler wird nicht benötigt. Das Modul läuft nun mit 5V (!) Versorgungsspannung. Dieser Betrieb ist NICHT in Kombination mit dem AVR-Stamp zulässig, da dieses Modul ausschließlich mit 3.3V betrieben wird.

C – 5V Spanungsversorgung von Außen (z.B. USB) , 3.3V intern

Die Versorgungsspannung von 5V wird über das äußere Stecksystem an Pin B2 angelegt. Zur internen 3.3V Versorgung der Z180 wird der interne Spannungsregler genutzt. Dazu sind die Kontakte 2 und 3 zu schließen. Soll auch das AVR-Modul versorgt werden, sind zusätzlich die Kontakte 2 und 4 zu schließen. Damit werden den intern erzeugten 3.3V auf das äußere Stecksystem (Pin B1) gelegt. Läuft das AVR-Modul mit einer 5V USB-Spannung (JP1), erfolgt in dieser Beschaltungsvariante auch die 3.3V Versorgung über den internen Regler.

BOOTLOADER

Der Bootloader ist ein spezielles Programm im AVR, dessen Aufgabe nur darin besteht, das eigentliche Controllerprogramm in den Flashspeicher des Controllers zu laden. Der Ladevorgang des Anwenderprogrammes erfolgt über die serielle Schnittstelle des Controllers. Normalerweise startet der Controller die Abarbeitung seiner Programmierung an der Stelle 0x0000. Da der Bootloader-Bereich des AVR jedoch am Ende des Flash-Speichers liegt, ist der AVR so zu konfigurieren, dass er nach einem RESET von der Bootloaderadresse gestartet wird. Diese Konfiguration ist wie alle wichtigen und grundlegenden Konfigurationen über die Fuses des AVRs geregelt.

Boot Size Configuration

Der ATmega1280/1281 besitzt einen eine maximalen Bootspeicher von 4096 Words (Bootadresse 0xF000). Für den verwendeten Bootloader *FastBoot* von Peter Dannegger [1] reichen jedoch 512 Words aus. Damit liegt die Bootadresse bei 0xFE00. Diese Konfiguration ist über die beiden Fuses **BOOTSZ1** (Bit 2) und **BOOTSZ0** (Bit 1) festzulegen. Beide erhalten den Wert **1**. Weiterhin muss dem AVR mitgeteilt werden, dass er nach einem RESET direkt in die Bootloader Sektion springt. Das wird mit der Fuse **BOOTRST (Bit 0)** festgelegt. Auch diese muss den Wert **1** erhalten. Bei Unklarheiten hilft ein Blick in das Datenblatt des ATmega1280/1281.

Fuse Configuration

Für den generellen Betrieb im CP/M System sind weitere Fuse Einstellungen notwendig. Dazu ist das Fuse High Byte, das Fuse Low Byte und das Extended Fuse Byte richtig zu konfigurieren.

Fuse High Byte

Fuse High Byte	Bit No	Description	Default Value
OCDEN	7	Enable OCD	1 (unprogrammed) OCD disabled
JTAGEN	6	Enable JTAG	0 (programmed) JTAG enabled)
SPIEN	5	Enable Serial Program and Data Downloading	0 (programmed) SPI prog. Enabled
WDTON	4	Watchdog Timer always on	1 (unprogrammed)
EESAVE	3	EEPROM memory is preserved through the Chip Erase	1 (unprogrammed) EEPROM not preserved
BOOTSZ1	2	Select Boot Size	0 (programmed)
BOOTSZ0	1	Select Boot Size	0 (programmed)
BOOTRST	0	Select Reset Vector	1 (unprogrammed)

Fuse High Byte erhält den Wert: **0xD6**

Fuse Low Byte

Fuse High Byte	Bit No	Description	Default Value
CKDIV8	7	Divide clock by 8	0 (programmed)
CKOUT	6	Clock output	1 (unprogrammed)
SUT1	5	Select start-up time	1 (unprogrammed)
SUT0	4	Select start-up time	0 (programmed)
CKSEL3	3	Select Clock source	0 (programmed)
CKSEL2	2	Select Clock source	0 (programmed)
CKSEL1	1	Select Clock source	1 (unprogrammed)
CKSEL0	0	Select Clock source	0 (programmed)

Fuse Low Byte erhält den Wert: **0xAF**

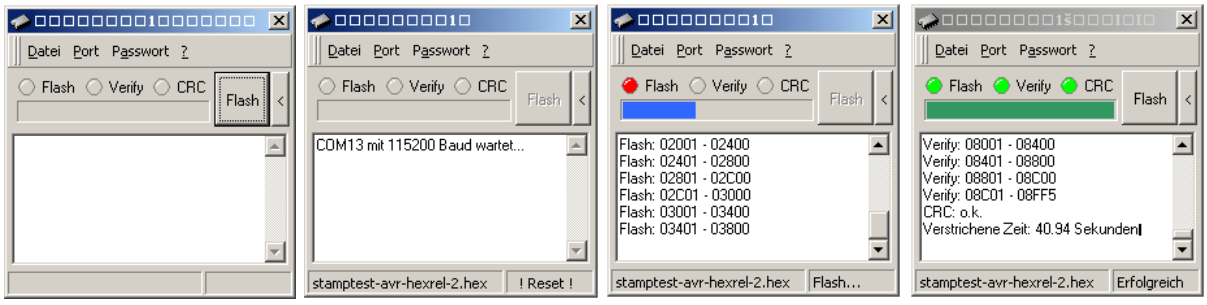
Extended Fuse

Fuse High Byte	Bit No	Description	Default Value
BODLEVEL2	2	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL1	1	Brown-out Detector trigger level	1 (unprogrammed)
BODLEVEL0	0	Brown-out Detector trigger level	1 (unprogrammed)

Extended Fuse Byte erhält den Wert: **0x5F**

Host Software

Um auf der PC-Seite das AVR Anwenderprogramm mittels serieller Schnittstelle zu übertragen, ist eine geeignete Software notwendig. Unter dem Betriebssystem WINDOWS bietet sich die Software *AVRFlash* [2] oder *Updateloader* [3] an. Die Bedienung ist sehr einfach. Unter dem Menüpunkt Datei wird die HEX-Datei (Intel-Hex-Format) gewählt, die auf den Controller übertragen werden soll. Unter dem Menüpunkt Port werden der COM-Port und die Baudrate eingestellt. Als Baudrate ist 115200 zu wählen. Nun kann durch das Betätigen des Flash Buttons der Download ausgelöst werden. Das Programm wartet nun auf einen RESET am AVR System. Nach dem RESET startet der Download. Ein erfolgreicher Download wird über die drei grünen LEDs signalisiert. Gleichzeitig kann der eigentliche Downloadvorgang über die LED2 und LED3 auf dem AVR-Stamp beobachtet werden.



INBETRIEBNAHME

Die Inbetriebnahme der einzelnen Funktionen wie RTC, RAM oder SD-Card werden durch den *Stamp-Monitor* unterstützt. Der komplette Funktionsumfang, sowie die Bediensyntax kann über den Befehl *help* bzw. *?* abgefragt werden. Zur Kommunikation mit dem *Stamp-Monitor* kann ein beliebiger Terminal-Emulator wie z.B. *Tera Term* verwendet werden. Dazu sind die folgenden Kommunikationsparameter zu wählen.

Baud rate	115200
Data	8 bit
Parity	none
Stop	1 bit
Flow control	none

Nach dem betätigen des Reset-Tasters auf dem AVR-Stamp-Modul muss sich der *Stamp-Monitor* mit einer Startinfo melden. Läuft die Startsequenz vollständig durch, so können ab dieser stelle die einzelnen Monitorfunktionen genutzt werden.

RTC

Auf dem AVR-Stamp-Modul befindet sich eine Echtzeituhr (RTC) welche über eine Pufferbatterie gestützt wird. Die Kommunikation der RTC (IC4) mit dem AVR erfolgt über einen I2C-Bus. Dazu muss zunächst hardwareseitig die korrekte I2C Busadresse der RTC eingestellt werden. Standartmäßig ist die Adresse A0h für Write und A1h für Read vorgesehen. Dazu ist das Pin 3 (A0-RTC) von IC4 auf GND zu legen. Jetzt hat der *Stamp-Monitor* Zugriff auf die RTC.

Abfragen der Uhr

Der Befehl ***date*** ohne Argumente liefert das aktuelle Datum und die Uhrzeit der RTC.

```
=>date
```

```
Mon  Dec  22  10:30:00  2014
```

Stellen der Uhr

Das Stellen der Uhr erfolgt über das Kommando **date MMDDhhmmCCYY.ss**

Bsp.: 22.12.2014 um 10:30 Uhr

```
=> date 122210302014.00
```

Der Monitor quittiert eine korrekte Eingabe mit der Meldung des Datums und der Zeit.

```
Mon  Dec  22  10:30:00  2014
```

SD-Card

Das AVR-Stamp-Modul verfügt über zwei Varianten eine SD-Card anzuschließen. Die Basisvariante besteht aus einem Micro-SD-Sockel direkt auf dem AVR Board. Für eine Erweiterung kann eine zweite SD-Card an der externen Stiftleiste des AVR-Stamp-Moduls bestückt werden.

Funktion	Port-Pin	Steckverbinder
CS	PG4	A12
WP	PG5	A13
SCK	PB1	A14
MOSI	PB2	A15
MISO	PB3	A16
CD	PG3	A21

Um das CS-Pin (PG4/A12) zur Kartenerkennung zu nutzen, ist an dieser Stelle ein Pulldown Widerstand von ca. 330k vorzusehen. In Kombination mit dem internen Pullup Widerstand der SD-Card ergibt sich eine einfache Möglichkeit eine gesteckte SD-Card zu detektieren.

Test einer SD-Card

Die Low-Level Funktionen der SD-Card werden über den **sd** Befehl bereitgestellt.

Mit dem Befehl **sd status x**

x=0 On-Board-Card

x=1 externe Card

kann der aktuelle Status des jeweiligen SD-Sockels abgefragt werden. Der Monitor antwortet mit einer entsprechenden Statusmeldung.

Code	Meldung
01	Drive not initialized
02	No medium in the drive
04	Write protected
08	Fast SPI clock

Nach einem Monitor-Reset oder einem Kartenwechsel sollte der Befehl

```
=> sd status 0
```

bei eingelegter Karte die Antwort

```
Socket status: 01
```

ergeben (Drive not initialized).

Im nächsten Schritt kann die Karte mit dem Befehl

```
=> sd init 0
```

initialisiert werden. Als Antwort erhält man bei erfolgreicher Initialisierung den Disk-Status (siehe Statustabelle). Die Antwort

```
rc=08
```

zeigt also eine erfolgreiche Initialisierung (Fast SPI clock) an.

Mit dem Befehl

```
=> sd info 0
```

kann nun der Zustand der SD-Card ermittelt werden. Die Monitorausgabe beinhaltet eine Vielzahl von Informationen. Nähere Auskunft über die Registerbelegungen sind in [4] zu finden. Verlaufen alle diese Schritte erfolgreich, kann die SD-Card bzw. darauf aufbauende Kommandos verwendet werden.

Filesystem

Neben den Low-Level Funktionen der SD-Card beherrscht der Monitor auch das FAT16 und FAT32 Dateisystem. Das ermöglicht dem AVR oder dem Z180 einen eleganten Dateizugriff. Zunächst kann der Status des auf der Karte implementierten Filesystems ermittelt werden.

```
=> fatstat 0:

FAT type:               2
Bytes/Cluster:          16384
Number of FATs:         2
Root DIR entries:       512
Sectors/FAT:            242
Number of clusters:     61935
FAT start (lba):        1
DIR start (lba,cluster): 485
Data start (lba):       517
Volume name:
Volume S/N:             1234-5678

66 files, 924663 bytes.
6 folders.
990960 KB total disk space.
989120 KB available.
```

Über **fatls 0:** werden alle Dateien und Verzeichnisse im Wurzelverzeichnis angezeigt.

```
=> fatls 0:
----A 2014/12/18 15:37    199238  stamp-monitor-avr-hexrel-4.hex
----A 2014/09/13 22:17    103673  stamptest-avr-hexrel-2.hex
----A 2014/12/21 15:28    199258  stamp-monitor-hexrel-4.1.hex
D---- 2014/12/21 18:51         0  stamptest-hexrel-1
----A 2014/10/14 21:45    113565  stamp-monitor.hex
   4 File(s),    615734 bytes total
   1 Dir(s),     989120K bytes free
```

Die erweiterte Syntax für Laufwerke, Namen und Verzeichnisse ist unter [5] nachzulesen.

Quellenangaben

[1] http://www.mikrocontroller.net/articles/AVR_Bootloader_FastBoot_von_Peter_Dannegger
[2] http://www.mikrocontroller.net/articles/AVR_Bootloader_FastBoot_von_Peter_Dannegger/Tutorial_ATtiny13
[3] http://www.leo-andres.de/2012/09/updateloader-benutzeroberflache-fur-avr-bootloader/
[4] http://dlnmh9ip6v2uc.cloudfront.net/datasheets/Components/General/SDSpec.pdf
[5] http://elm-chan.org/fsw/ff/en/filename.html

ANHANG A

Pinzuordung der Stamp Module

AVR	Z180	all		A	B		all	Z180	AVR
		DESEL		1	1		+3.3 volts		
/ARESET	nc			2	2		+5.0 volts		
		A19		3	3			A18	PE4
		D0		4	4			A17	PE3
		D1		5	5			A16	PE2
		D2		6	6		A15		
		D3		7	7		A14		
		D4		8	8		A13		
		D5		9	9		A12		
		D6		10	10		A11		
		D7		11	11		A10		
PG4	/RTS0			12	12		A9		
PG5	/CTS0			13	13		A8		
SCK	/DCD0			14	14		A7		
MOSI	TXA0			15	15		A6		
MISO	RXA0			16	16		A5		
PB4	CKA0			17	17		A4		
PB5	TXA1			18	18		A3		
PB6	RXA1			19	19		A2		
PB7	CKA1			20	20		A1		
PG3	TXS			21	21		A0		
PG2	RXS			22	22			/NMI	nc
PG1	CKS			23	23		/ZRESET		
PG0	/DREQ1			24	24		/BUSREQ		
SDA	/TEND1			25	25		/BUSACK		
SCL	/HALT			26	26			/WAIT	nc
nc	/RFSH			27	27			PHI	nc
		CLKO		28	28		/RD		
nc	/INT0			29	29		/WR		
nc	/INT1			30	30		/M1		
nc	/INT2			31	31		/MREQ		
		GND		32	32		/IORQ		