

## Einleitung

Ursprünglich ist der high voltage (HV)-Programmer von Chan auf [http://elm-chan.org/works/avr\\_x/report\\_e.html](http://elm-chan.org/works/avr_x/report_e.html) entstanden. Nachdem ich diesen aufgebaut und ihn für sehr nützlich und gut befunden hatte, entschloss ich mich, diese Anleitung und das Layout dazu zur Verfügung zu stellen. Der ISP-Programmer ist eine Version mit dem HCT125 (ähnlich Atmel AVR ISP Dongle), welcher im Layout mit eingeschlossen ist. Beide Programmer arbeiten unabhängig voneinander, der HV-Programmer ist dafür gedacht, bei falsch gesetzten Fuse-Bits diese zurückzusetzen, falls man an den Mikrocontroller sonst nicht mehr herankommt. Durch entsprechende Jumper kann man zwischen dem ISP- und HV-Programmiermode umschalten.

## Problem

Die meisten der heutigen Mikrocontroller werden normalerweise mit einer Software und einem Hardwareadapter (ISP=Incircuit Serial Programmer) programmiert. Da die Atmel-Controller sogenannte Fuse-Bits besitzen, gibt es hier ein Problem, wenn diese Fuse-Bits „irgendwie“ gesetzt werden. Die Fuse-Bits geben z.B. an, ob der Controller mit einem externen Quarz getaktet oder die Reset-Leitung als normaler Pin fungieren soll. Bei unüberlegtem oder versehentlichem Setzen dieser Fuse-Bits kann der Controller nicht mehr über den ISP-Adapter programmiert werden, der Controller bleibt sozusagen für den ISP verschlossen.

## Lösung

Zur Lösung dieses Problems kann man sich entweder ein STK500 Programmiergerät von Atmel kaufen (ca. 100,- €) oder diesen preiswerten Adapter selber zusammenbauen. Dieser Programmer bietet die Möglichkeit der „high voltage“ (HV)-Programmierung, nur mit dieser ist es möglich, die Fuse-Bits wieder in einen default-Zustand zu setzen und danach wiederum mit dem ISP-Programmieradapter den Controller wie gewohnt zu programmieren. „high voltage“ bedeutet, dass der Controller mit 5V und 12V gebrannt wird.

## Vor- und Nachteile

Die Vorteile dieses HV-Programmers liegen darin, dass man keinen zusätzlichen, bereits programmierten Controller benötigt, um ihn aufzubauen. Der Programmer kann diverse Atmel-Controller programmieren, siehe dazu auch die Liste am Ende dieses Dokuments.

Als Nachteil wäre zu erwähnen, das durch die Anzahl der Bauteile ein gewisser Zeitaufwand benötigt wird, um ihn aufzubauen. Für diejenigen, die nur Windows-Oberflächen und Mausklicks gewohnt sind, könnte das Kommandozeilentool auch ein Nachteil sein. Dazu jedoch später, die Bedienung ist kinderleicht!

## Bauteileliste

Wer diesen hier beschriebenen Programmer nachbauen will, benötigt zunächst folgende Bauteile (Im Bausatz enthalten):



Anz.	<b>Widerstände</b>	Anz.	<b>Stecker, Buchsen</b>
20	1k	1	SUBD M25-polig
2	2k2	3	IC-Fassung 14-polig
1	4k7	1	IC-Fassung 20-polig
1	8 x 100k (Netzwerk)		<b>Halbleiter</b>
1	10k	2	1N4004
3	100R	2	2SA1345
1	100k	2	2SC3402
1	680R	1	74HC125N
	<b>Kondensatoren</b>	1	74HCT299N
1	10n RM2,5	1	78L05Z
2	100µF oder 47µF/16V RM2,5	1	78L12Z
5	100nF RM2,5	1	B80C500
1	220µF/35V oder 470µF /35V RM3,5	1	BC327
	<b>Stecker, Buchsen</b>	1	BC337
1	ML6 (AMP Stiftleiste mit Wanne)	4	LED3MM
1	ML20 (AMP Stiftleiste mit Wanne)		<b>Taster, Netzteil, Draht</b>
1	Stiftleiste, 8 Pole (JP1, JP2, JP3, JP4)	1	Resettaster
3	Buchsenleiste, gedrehte Kontakte, 32 Pins	1	Steckernetzteil 2x 8,5V
1	Stromversorgungsbuchse	1	Draht d=0,5mm, isoliert, 2,5m
2	Jumper	1	Leiterplatte

### Stromlaufplan

Der Stromlaufplan auf der nächsten Seite wird zugrunde gelegt. Wie man erkennen kann, gibt es drei Jumper, die vor jeder Programmierung gewissenhaft und richtig gesetzt werden müssen. Folgende Tabelle zeigt die notwendigen Jumperstellungen, wenn man den ISP oder HV Modus benutzen möchte:

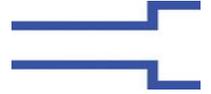
<b>Jumper</b>	<b>ISP</b>	<b>HV</b>
JP1	2-3	1-2
JP2	1-2	2-3
JP3	Nicht relevant	Für 8-poligen µC 1,2 : Signal R/B 3 : Signal BS1
JP4	STK200 oder 300 kompatibel STK200: 1-2 STK300: offen	offen

### Erläuterung zu den Jumpern:

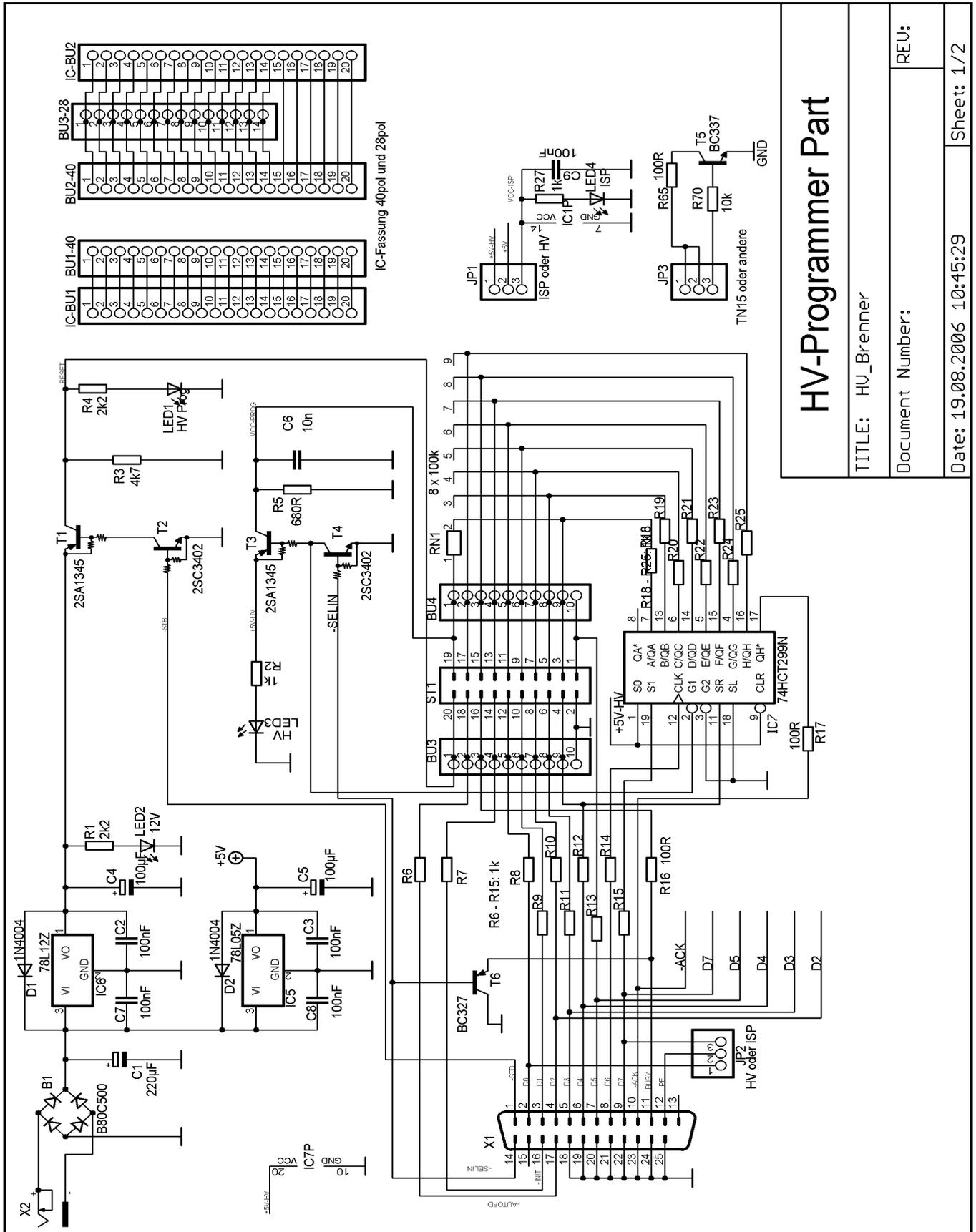
Bei JP1 wird die 5V Versorgungsspannung entweder auf das IC1 (ISP programmieren) oder IC7 gelegt.

Bei JP2 wird vom DSUB-Stecker Pin12 im ISP Modus mit Pin 2 verbunden, im HV Modus wird Pin 12 mit Pin 9 verbunden.

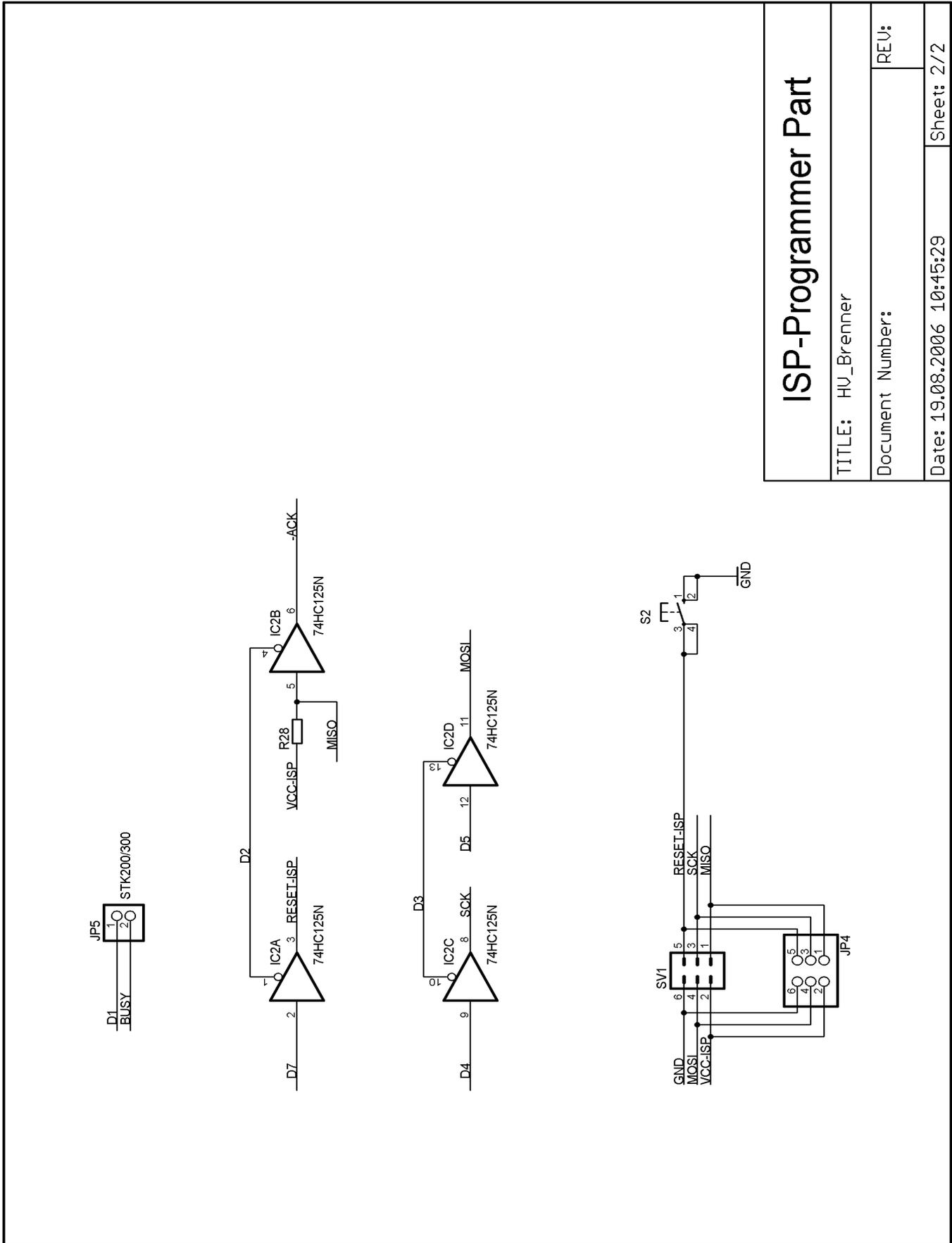
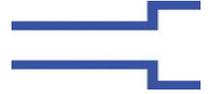
JP3 ist nur für den HV-Modus relevant. Siehe Verdrahtung für 8-polige Atmels.



JP4 gibt die Möglichkeit, zwischen STK200 und STK300 Kompatibilität umzuschalten (default: offen).



<b>HV-Programmer Part</b>	
TITLE: HV_Brenner	
Document Number:	
Date: 19.08.2006 10:45:29	Sheet: 1/2



<b>ISP-Programmer Part</b>	
TITLE: HV_Brenner	
Document Number:	REU:
Date: 19.08.2006 10:45:29	Sheet: 2/2



Desweiteren sind im Stromlaufplan diverse Buchsen- und Stiftleisten zu erkennen. Hier die Erklärungen für alle Steckverbinder:

X1	Parallelport, Stecker (männlich), wird mit 1zu1 Verlängerungskabel an den PC angeschlossen.
X2	Spannungsversorgung, sollte zwischen 14 und 16V AC oder DC betragen.
ST1	Adapter Stiftleiste für Erweiterungsboard, z.B. Boards für Controller im smd-Gehäuse. Es sind alle Signale vorhanden, um den Controller im parallelen Modus zu brennen.
ST2	ISP-Adapter, 6-polig. Nach Standard aufgebaut. Ebenfalls für Erweiterungsboards.
JP1	5V Spannungsversorgung entweder zum ISP- oder HV Brennen (siehe Tabelle oben).
JP2	Pin 12 von X1 entweder zum ISP- oder HV Brennen (siehe Tabelle oben).
JP3	R/B und BS1 Signal für 8-polige Controller (siehe Tabelle oben).
JP4	Entweder STK200 oder STK300 kompatibel
BU1A, BU1B	Die Signale an diesen Buchsen müssen, je nach MC-Typ, mit IC-BU1 und IC-BU2 verbunden werden. Eine genaue Anschlussbelegung ist unten zu finden.
BU2	ISP Signale zum Verbinden nach IC-BU1 oder IC-BU2.
BU1-40, BU2-40, BU3-28	Buchsenleisten für die Controller (8 – 28-polig und 40-polig)

#### LEDs

LED1 – Programmier-LED für HV

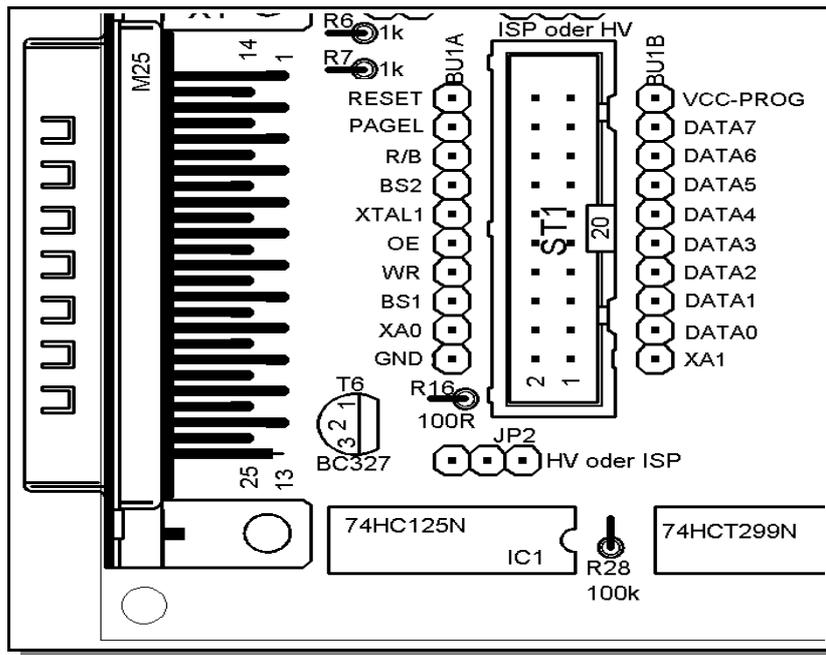
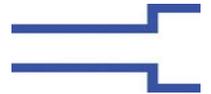
LED2 – 12V Programmierspannung vorhanden

LED3 – Zeigt an, dass JP1 auf den HV Programmiermodus eingestellt ist

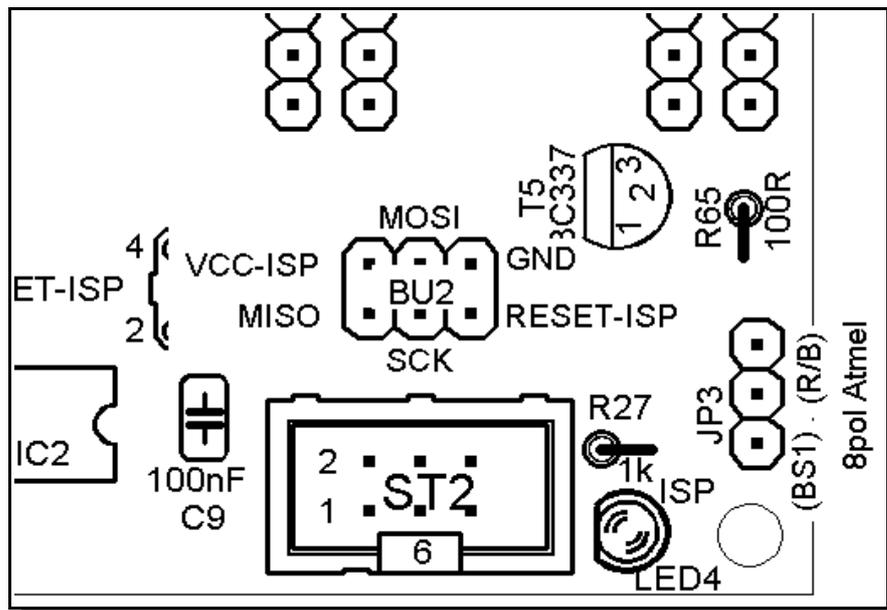
LED4 – Zeigt an, dass JP1 auf den ISP Programmiermodus eingestellt ist

#### Belegung der Buchsenleiste BU1A/BU2A und ST1 (HV Programmierung)

Um einen Controller im HV-Modus wieder in einen default-Zustand zurückzusetzen, sind die Signale an den Buchsenleisten BU1A/BU1B oder ST1 notwendig. Wenn ein Controller z.B. die Datenleitungen D0 bis D7 nicht benötigt (z.B. Tiny 15 o.ä.), bleiben diese unbenutzt. Die Belegung geht aus folgendem Bild hervor:

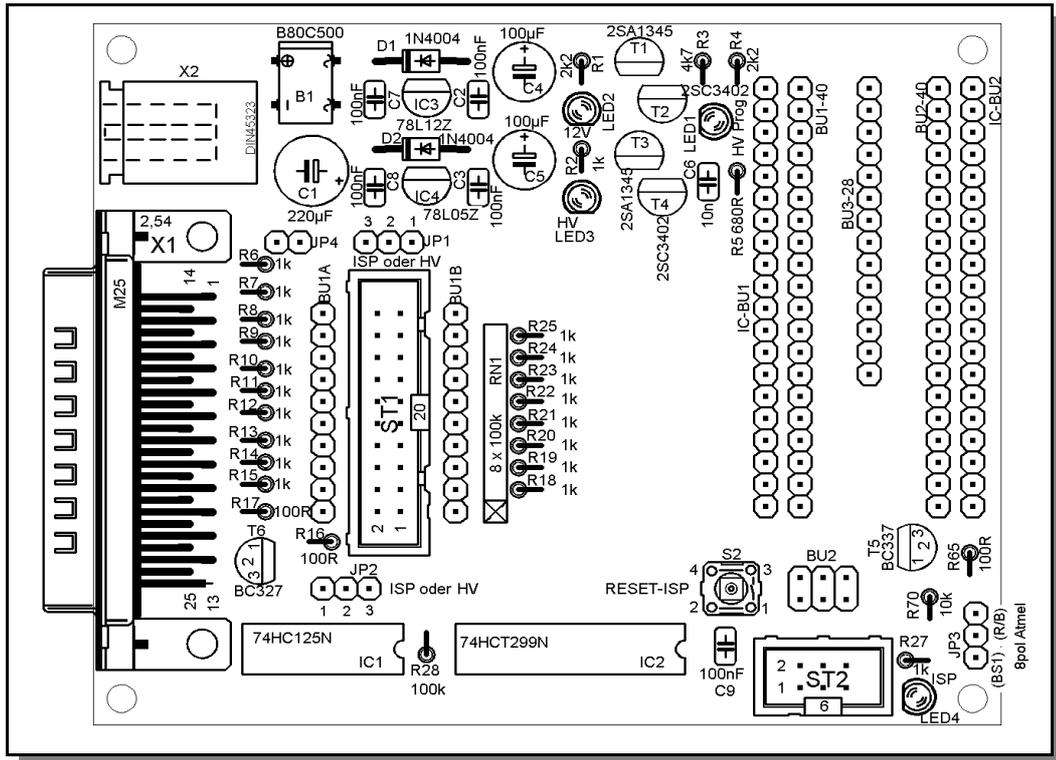
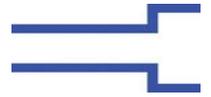


Pinbelegung BU1A, BU1B und ST1



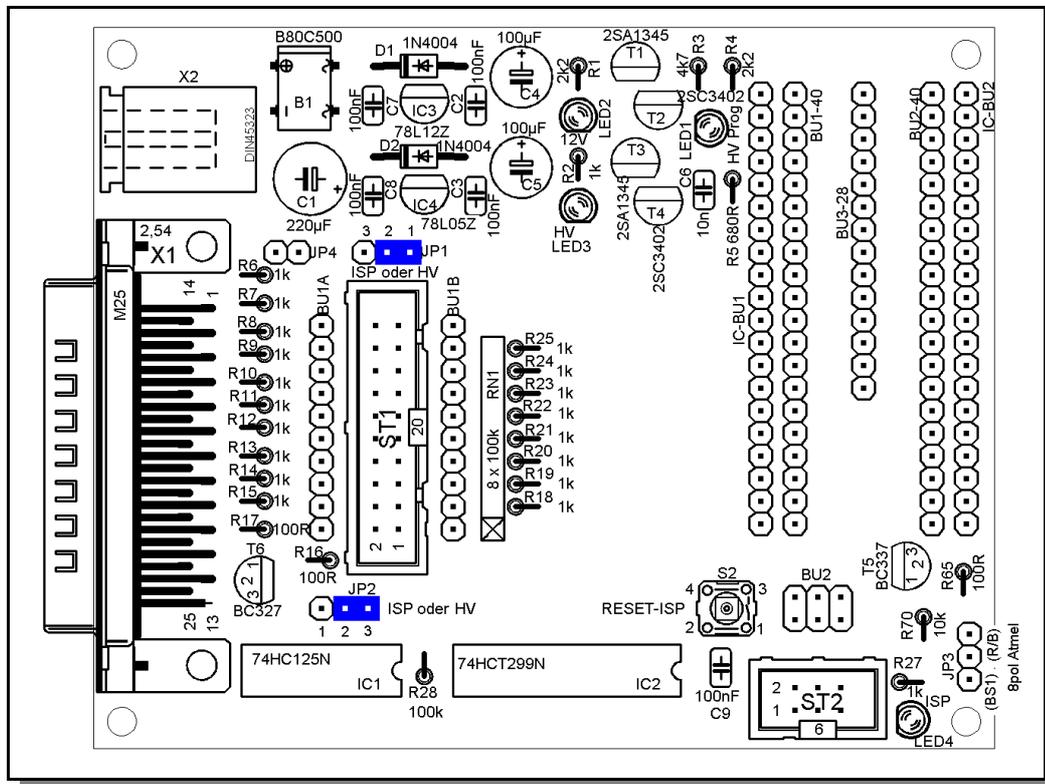
Pinbelegung BU2, ST2, ISP Anschluss

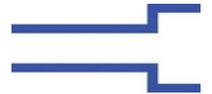
Der ISP Anschluss entspricht der „Norm“ des 6-poligen Atmel-Verbinders. Auch hier besteht mit ST2 die Möglichkeit, diesen Anschluss für smd-Controller zu nutzen.  
Bestückungsplan



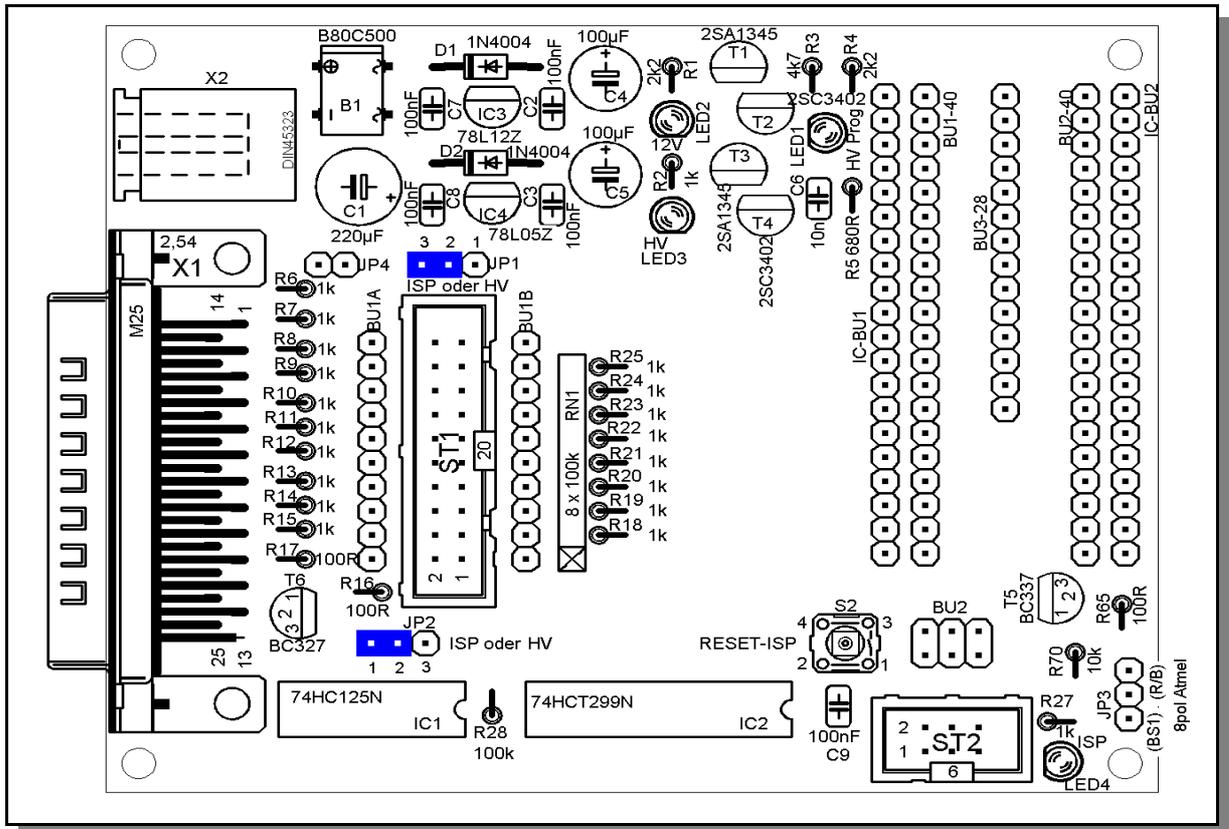
Bestückungsplan für den Programmierer

Jumper für HV Modus

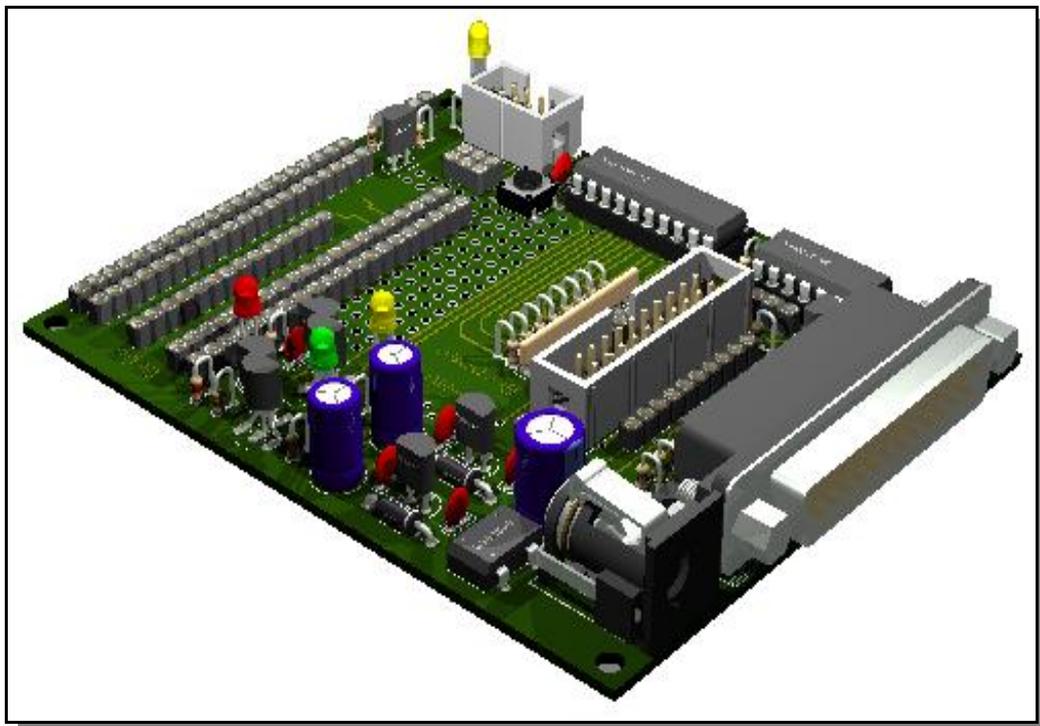




Jumper ISP-Modus



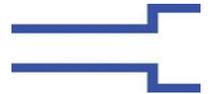
3D-Ansicht



(Erstellt mit eagle3D)

Inbetriebnahme

Zur Inbetriebnahme benötigt man nur einen Trafo mit ca. 12-16V AC



Ausgangsspannung (z.B. Steckernetzteil, im Bausatz enthalten). Vor Inbetriebnahme sollten alle Jumper und auch die Drahtbrücken (siehe unten) gesetzt sein. Dann können das Parallelkabel (X1) und die Stromversorgung (X2) angeschlossen und eingeschaltet werden.

#### Software ISP-Programmierung

Für den ISP-Programmiermodus kann PonyProg ([www.lancos.com](http://www.lancos.com)) verwendet werden. In PonyProg wird als Adapter der parallele ISP I/O-Adapter ausgewählt (unter „setup – Interface setup“).

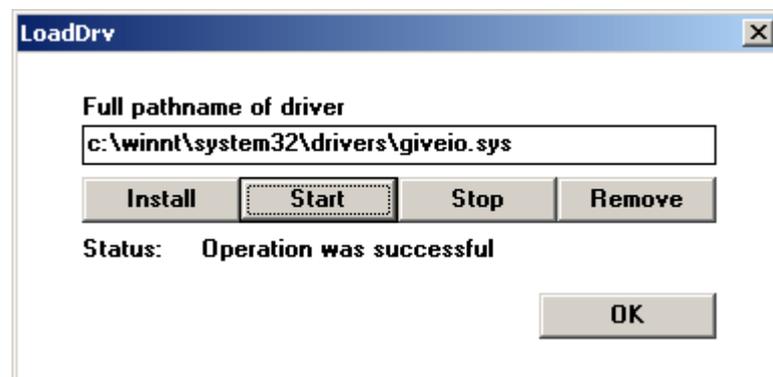
#### Software HV-Programmierung

Zum Betrieb des HV-Brenners ist folgende Software notwendig:

- avrpp, kann von der Internetseite [http://elm-chan.org/works/avr\\_x/report\\_e.html](http://elm-chan.org/works/avr_x/report_e.html) heruntergeladen werden
- giveio.sys, kann von der Internetseite <http://www.irdeo.de/ntdriver.zip> heruntergeladen werden

#### Parallelporttreiber

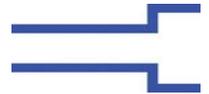
Zunächst wird der Parallelporttreiber giveio.sys in das Verzeichnis c:\winnt\system32\drivers (bei XP c:\windows\...) kopiert. Anschließend wird das Tool „LOADDRV“ gestartet. Es muss der komplette Pfadname inklusive der zu startenden giveio.sys angegeben werden. Anschließend wird der Treiber gestartet:



Starten des giveio.sys Treibers

#### Brennerprogramm

Das Brennerprogramm ist im Zip-File avrstool32.zip enthalten. Das File wird in einem Ordner zunächst entpackt. Anschließend wird das Kommando „cmd“ ausgeführt (Start->Ausführen->cmd). Im Commandfenster wechselt man nun in das Verzeichnis, wo das zip-File entpackt wurde. Der Aufruf des Tools „avrpp“ zeigt die Optionen an, die man zusätzlich mit diesem Tool angeben kann:



```
C:\WINNT\system32\cmd.exe
D:\Elektronikthemen\Web HU Programmierer\src avrpp>avrpp
AVRPP - AVR Parallel Programming tool R0.38 (C)ChaN,2006 http://elm-chan.org/

Write code and/or data : <hex file> [<hex file>] ...
Verify code and/or data : -v <hex file> [<hex file>] ...
Read code, data or fuse : -r<pin>{f}
Write fuse byte : -f<!h!x>[<bin>]
Lock device : -l[<bin>]
Erase device : -e
Copy calibration bytes : -c
Control port [-p1] : -p<n>
For more options, refer avrx32.txt.

Supported Device:
AT90S 1200,2313,2323,2333,2343,4414,4433,4434,8515,8535
ATtiny 10,11,12,13,15,22,24,25,26,28,44,45,84,85,2313
ATmega 8,16,32,48,64,88,103,128,161,162,163,165,168,169,323,325/329,406,603,640,
645/649,1280,1281,2560,2561,3250/3290,6450/6490,8515,8535
AT90CAN32,64,128, AT90PWM 2,3

D:\Elektronikthemen\Web HU Programmierer\src avrpp>
```

avrpp Aufruf

Es werden außerdem die unterstützten Mikrocontroller angezeigt. Nach Einstecken des Controllers in eine der IC-Fassungen (Achtung, Hinweise im Stromlaufplan) kann der Brenner an die parallele Schnittstelle angeschlossen und die Spannungsversorgung von 12-14V ~ eingeschaltet werden.

Mit dem Kommando „avrpp -rf“ (read fusebits) kann grundsätzlich festgestellt werden, ob der Brenner funktioniert. Das Auslesen der Bits sollte daher immer zuerst geschehen:

```
C:\WINNT\system32\cmd.exe
645/649,1280,1281,2560,2561,3250/3290,6450/6490,8515,8535
AT90CAN32,64,128, AT90PWM 2,3

D:\Elektronikthemen\Web HU Programmierer\src avrpp>avrpp -rf
Put a device on the socket and type Enter...
PAR->Detected device is ATmega8.

Low: 11100001
  | | | | |
  | | | | | CKSEL[3:0] Clock source selection
  | | | | | SUT[1:0] Startup time
  | | | | | BODEN <1:Disable BOD, 0:Enable BOD>
  | | | | | BODLEVEL <1:2.7V, 0:4.0V>

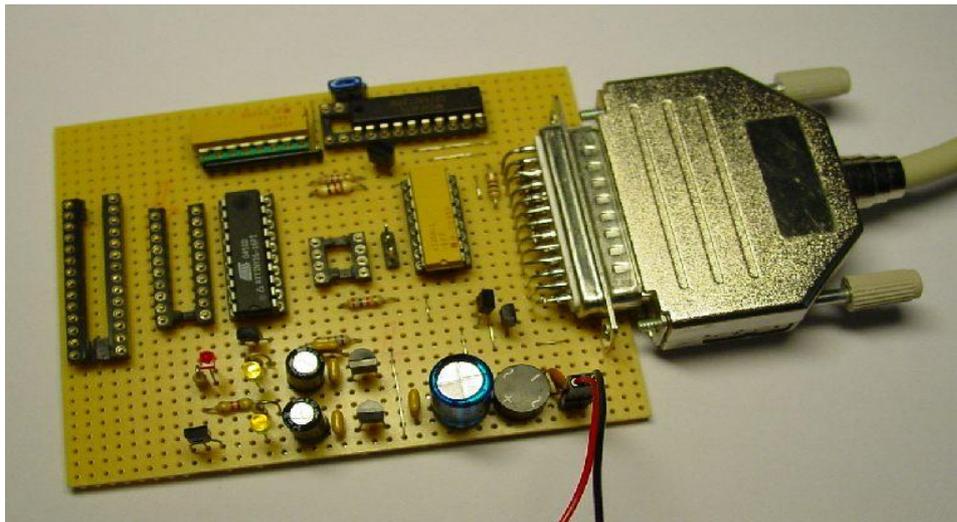
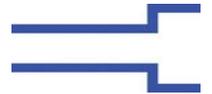
High:11011001
  | | | | |
  | | | | | BOOTRST *Refer to data sheet
  | | | | | BOOTSZ[1:0] *Refer to data sheet
  | | | | | EESAVE <Retain EEPROM at chip erase 1:No, 0:Yes>
  | | | | | CKOPT <Oscillation Mode 1:Low amplitude, 0:Full swing>
  | | | | | SPIEN <1:Disable ISP, 0:Enable ISP> *Available only in Parallel mode
  | | | | | WDTON <1:WDT normal, 0:WDT always on>
  | | | | | RSTDISBL <RESET pin 1:Enable, 0:Disable(PC6)>

Cal: 175 177 166 168

D:\Elektronikthemen\Web HU Programmierer\src avrpp>
```

Aufruf des Programms mit der Option -rf (read fusebits)

Der Programmierer hat den Atmega 8 erkannt und zeigt nun die Fusebits an. Mit dem Kommando avrpp -fl , fh und fx werden die Fuse-High- Low- und Extended Bytes mit Defaultwerten beschrieben.

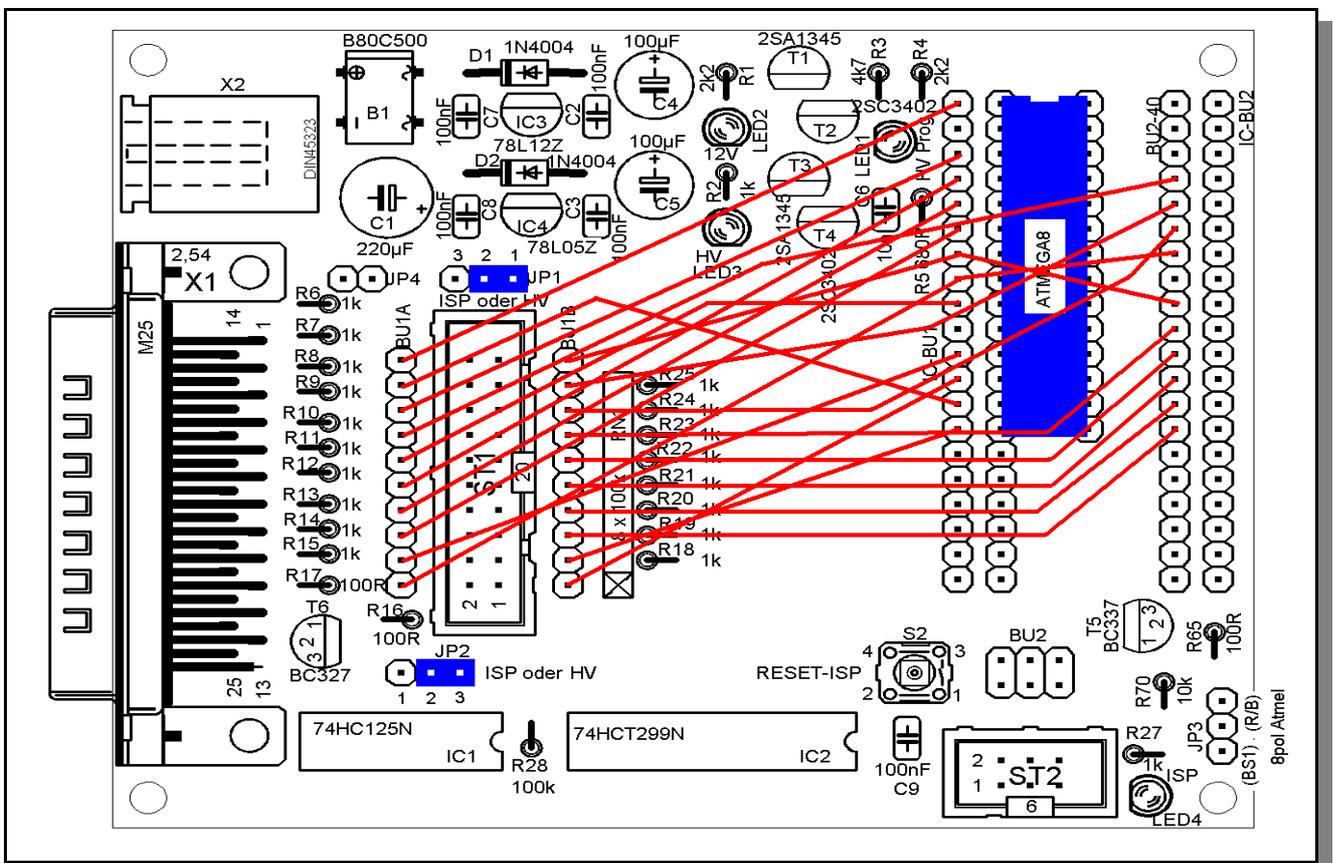


Erster HV-Prototyp (ohne ISP) – Attiny26 wiederbelebt!

Beispiel: ATMEGA8

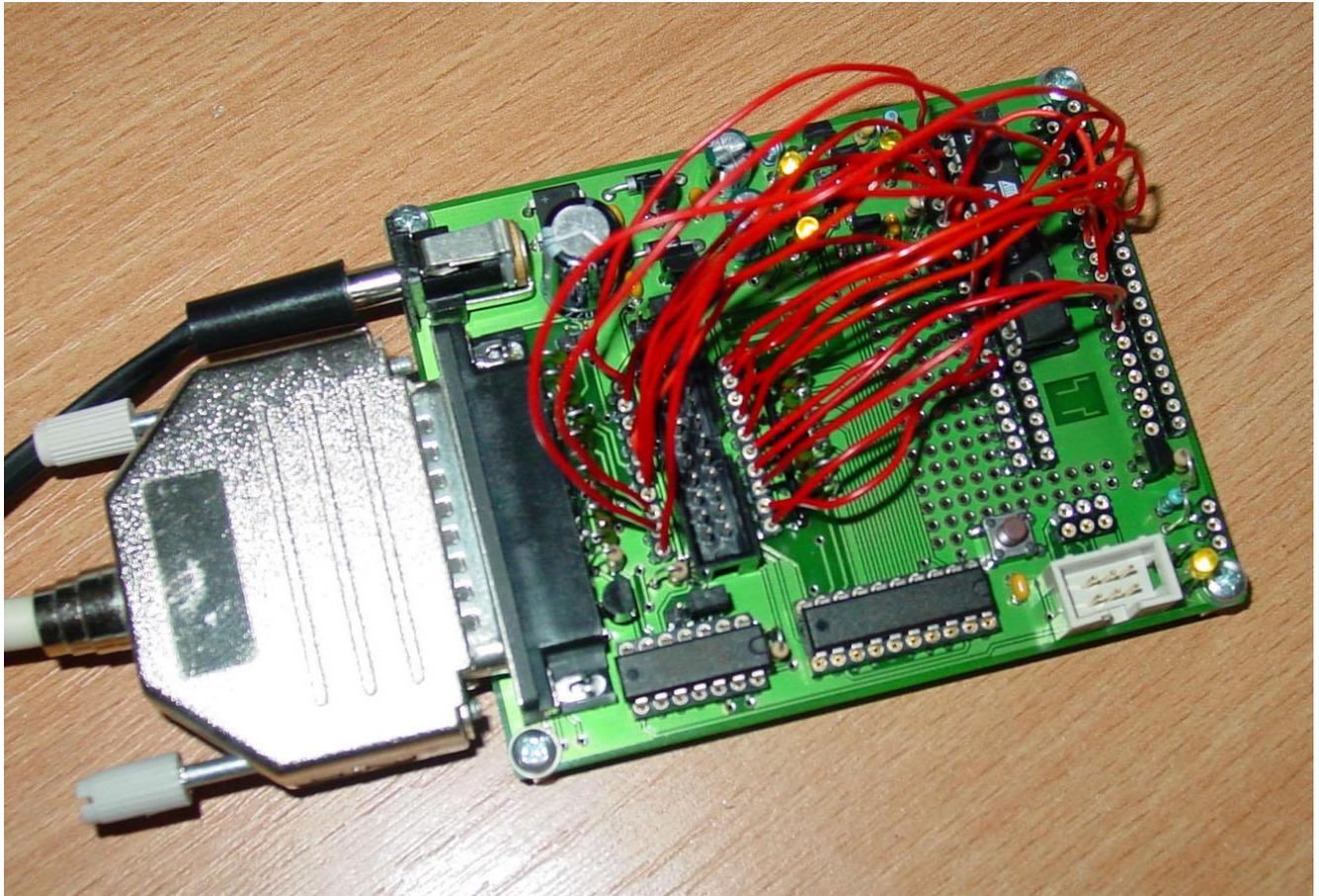
Einem Atmega8 wurden versehentlich die Fusebits so gesetzt, daß man ihn über den ISP-Adapter nicht mehr erreicht. Lösung:

1. Einstellen der Jumper auf den HV-Modus (siehe oben).
2. Einsetzen des Atmegas in die Fassung.
3. Setzen der Drahtbrücken, wie unten zu sehen.
4. Brennen mit avrpp.exe -fl ; avrpp.exe -fh

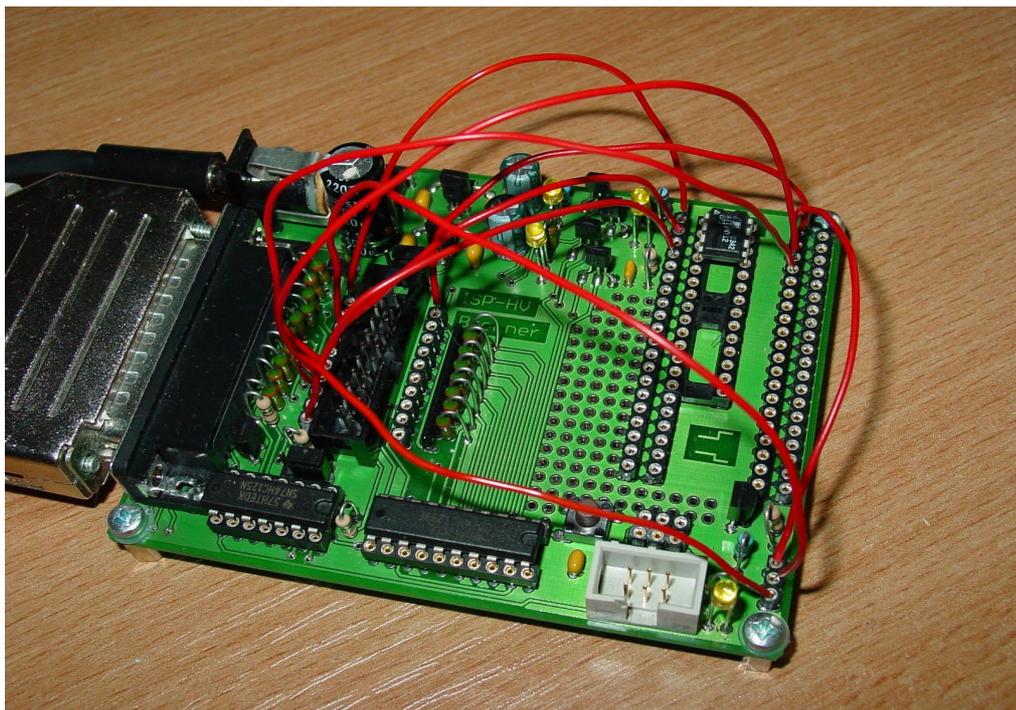


Verdrahtung ATmega 8 für HV Programmierung

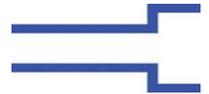
Nach dem Zurücksetzen sollte der Controller über den ISP-Adapter wieder normal zu erreichen sein.



ATmega 8 bei der Wiederbelebung



Tiny 12 – Fusebits einlesen:



```
C:\WINNT\system32\cmd.exe
Supported Device:
AT90S 1200,2313,2323,2333,2343,4414,4433,4434,8515,8535
ATtiny 10,11,12,13,15,22,24,25,26,28,44,45,84,85,2313
ATmega 8,16,32,48,64,88,103,128,161,162,163,165,168,169,323,325/329,406,603,640,
645/649,1280,1281,2560,2561,3250/3290,6450/6490,8515,8535
AT90CAN32,64,128, AT90PWM 2,3

Drücken Sie eine beliebige Taste . . .

C:\Dokumente und Einstellungen\scully>"D:\Arbeit-Geschäft\E-Elektronikentwicklung\
_WEB HU Programmierer\CD HU_Prog\Software avrpp\avrpp.exe" -rf
Put a device on the socket and type Enter...
PAR->Unknown device (FF-FF-FF).
HUS->Detected device is ATtiny12.

Low: 01010010
    | | | | |
    | | | | | CKSEL[3:0] Clock source selection
    | | | | | RSTDISBL <1:Use RESET pin, 0:Disable RESET(used as PB5)>
    | | | | | SPIEN <1:Disable ISP, 0:Enable ISP>
    | | | | | BODEN <1:Disable BOD, 0:Enable BOD>
    | | | | | BODLEVEL <1:1.8V, 0:2.7V>

Cal: 55

C:\Dokumente und Einstellungen\scully>
```

Tiny 12 wird erkannt!

**Hinweis für den Tiny 13:** Dieser muss mit der Option -8 ausgelesen werden, also avrpp.exe -rf -8:

```
C:\WINNT\system32\cmd.exe
Auswählen C:\WINNT\system32\cmd.exe

D:\>avrpp.exe -r -8
Put a device on the socket and type Enter...
HUS->Detected device is ATtiny13.

Device Signature = 1E-90-07
Flash Memory Size = 1024 bytes
Flash Memory Page = 32 bytes x 32 pages
EEPROM Size = 64 bytes
EEPROM Page = 4 bytes x 16 pages

D:\>avrpp.exe -rf -8
Put a device on the socket and type Enter...
HUS->Detected device is ATtiny13.

Low: 01101010
    | | | | |
    | | | | | CKSEL[1:0] Clock selection
    | | | | | SUT[1:0] Startup time selection
    | | | | | CKDIV8 Clock division ratio <1:1/1, 0:1/8>
    | | | | | WDTON <1:WDT normal, 0:WDT always on>
    | | | | | EESAVE <Retain EEPROM at chip erase 1:No, 0:Yes>
    | | | | | SPIEN <1:Disable ISP, 0:Enable ISP> *Available only HUS mode

High:---11111
    | | | | |
    | | | | | RSTDISBL <RESET pin 1:Enable, 0:Disable(PB5)>
    | | | | | BODLEVEL[1:0] <BOD 11:None, 10:1.8V, 01:2.7V, 00:4.3V>
    | | | | | DWEN <On-Chip Debugging via RESET pin 1:Disable, 0:Enable>
    | | | | | SPMEN <SPM instruction 1:Disable, 0:Enable>

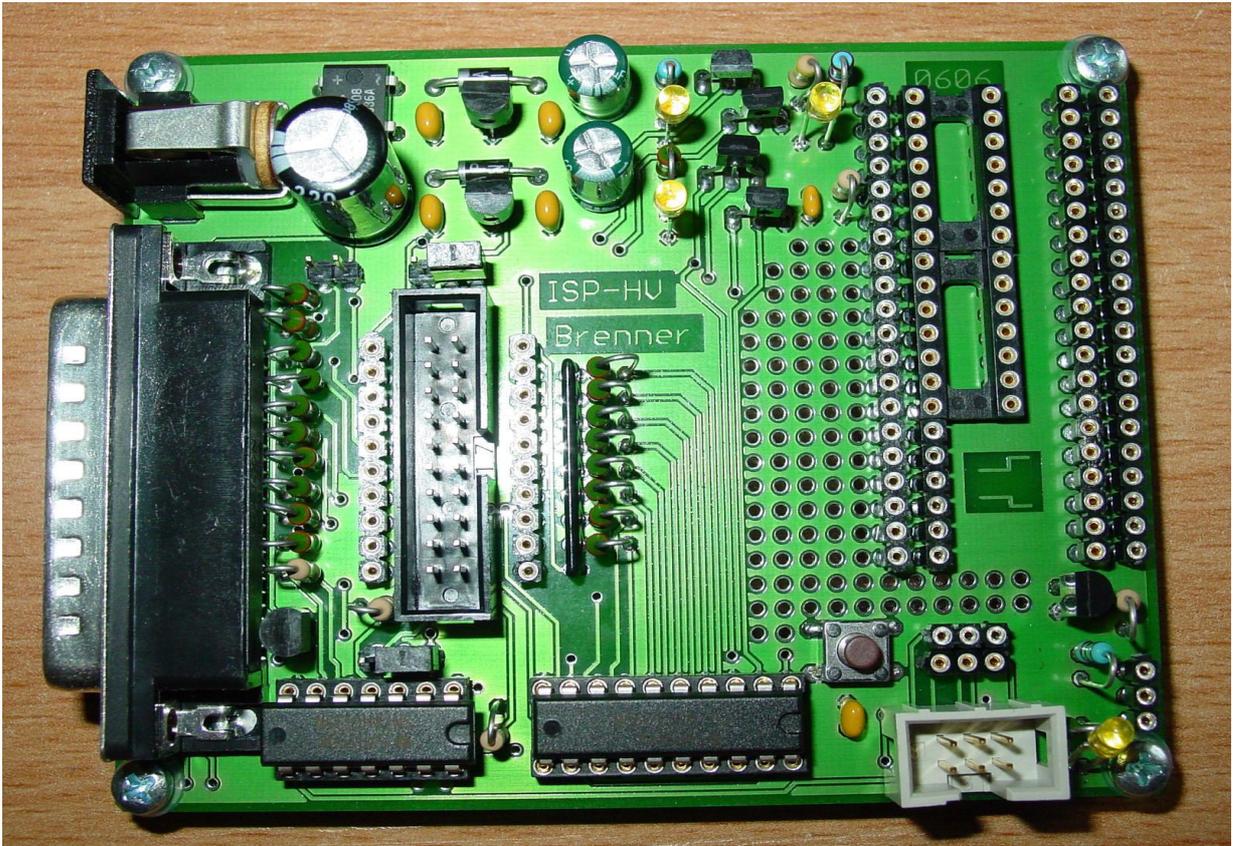
Cal: 93

D:\>
```

Tiny 13 auslesen

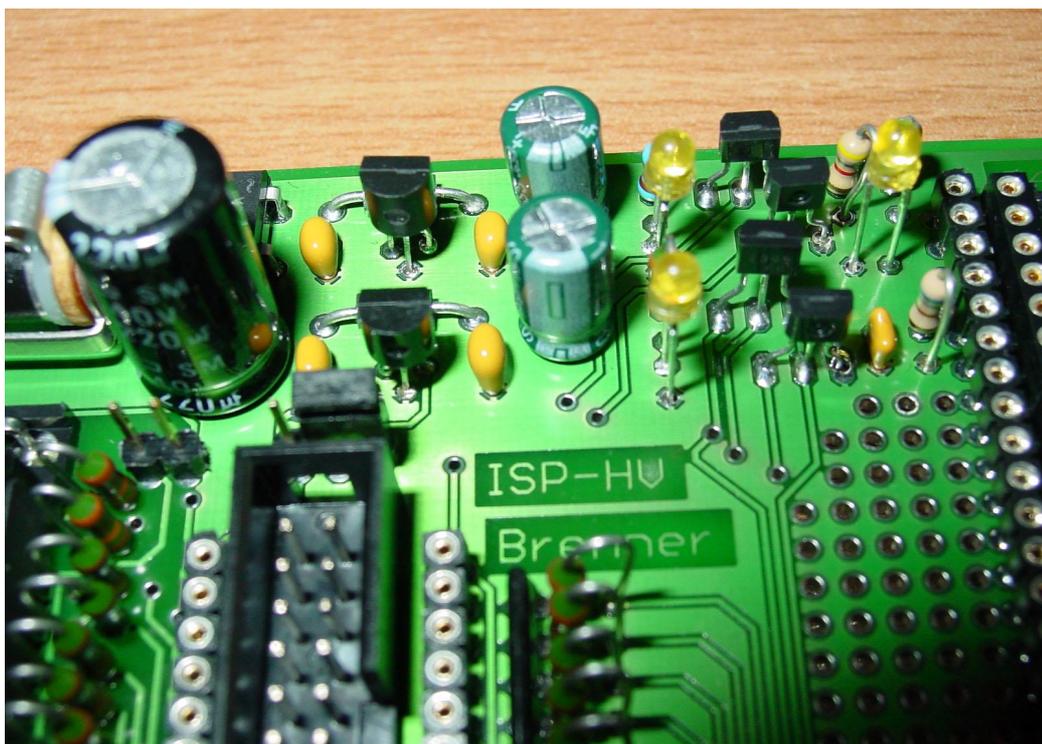
Speziell beim Tiny13 kann es zu Problemen beim HV-Brennen kommen. Eventuell stellen Sie die Fusebits ungewollt um, in diesem Fall muss der Parallelport de- und wieder reaktiviert werden. Im Datenblatt vom Tiny13 gibt es ebenfalls Hinweise zu

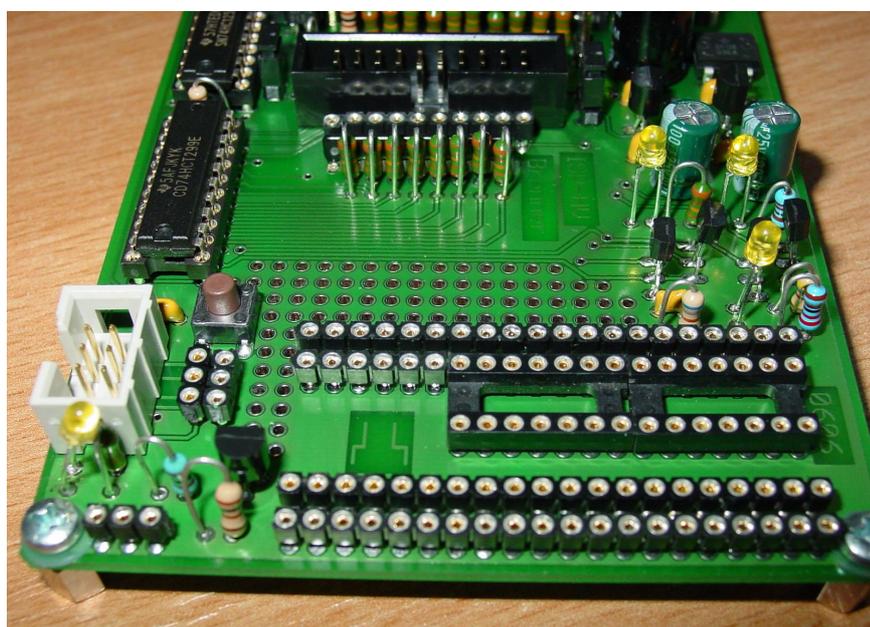
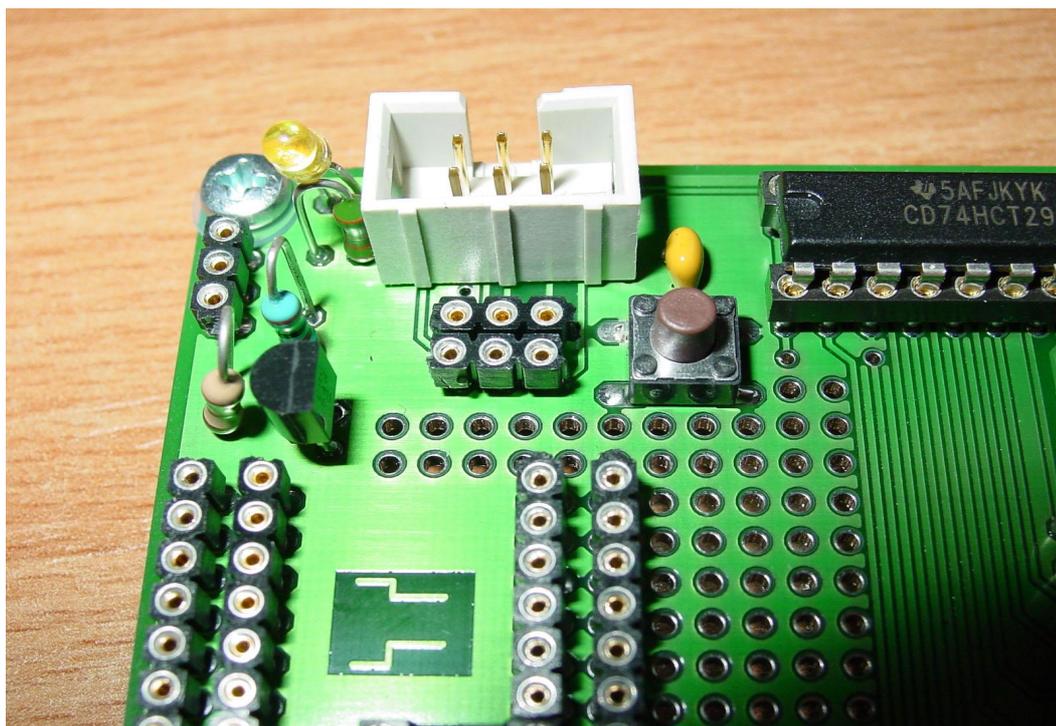
diesem Problem.



Bestückte Leiterplatte

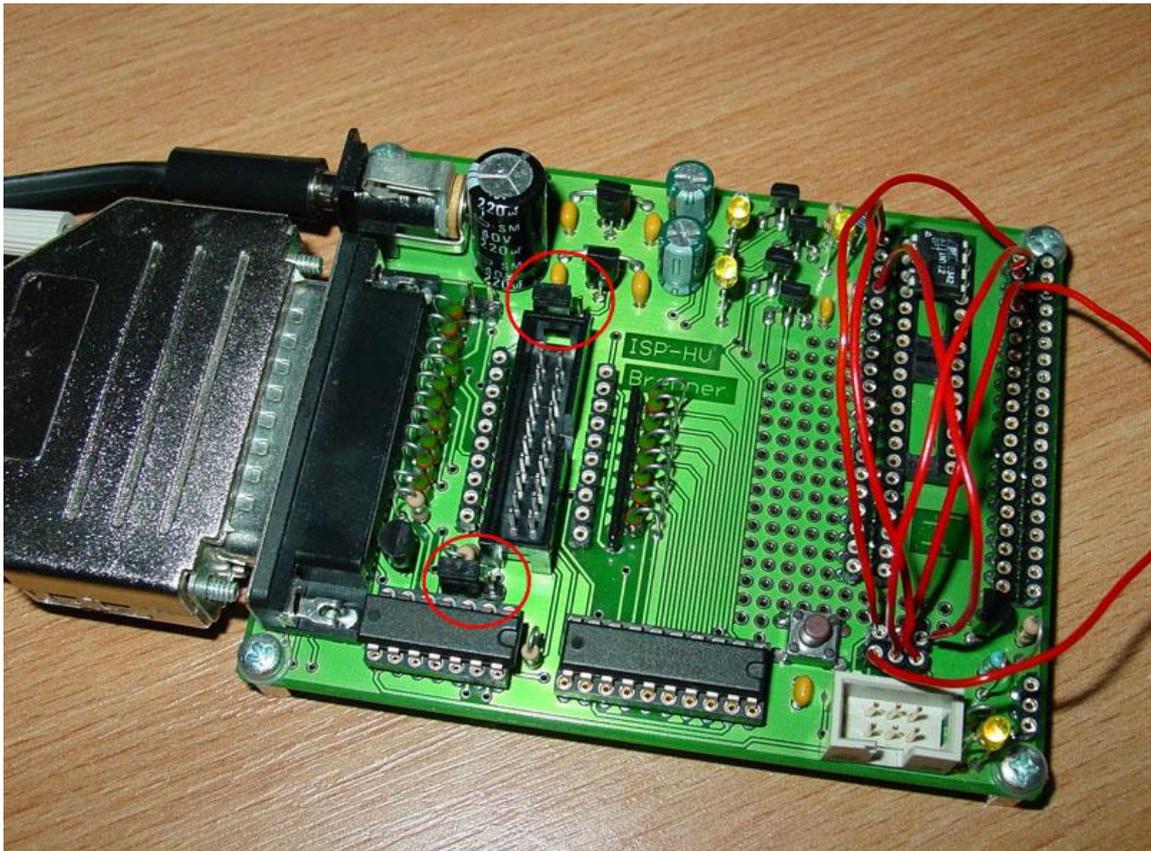
Bestückungsdetails:





Beispiel: Attiny 12 im „ISP-Mode“

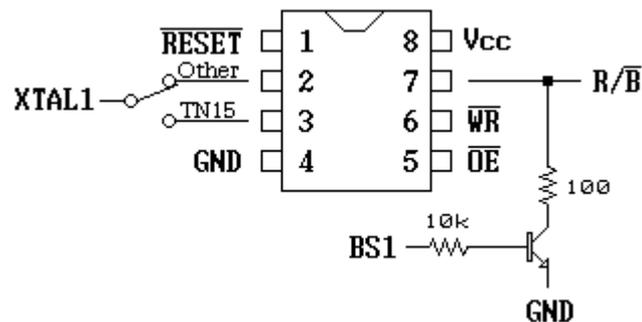
Das folgende Bild zeigt einen Attiny12, der im ISP-Mode angeschlossen ist. Im Brennerprogramm, z.B. Ponyprog muss als Interface „AVR ISP I/O“ eingestellt werden, damit der Controller erkannt und programmiert werden kann. Wichtig sind auch die beiden Jumper, siehe Bild. Diese müssen auf ISP eingestellt werden.

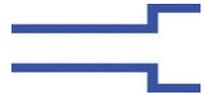


Tiny12 am ISP

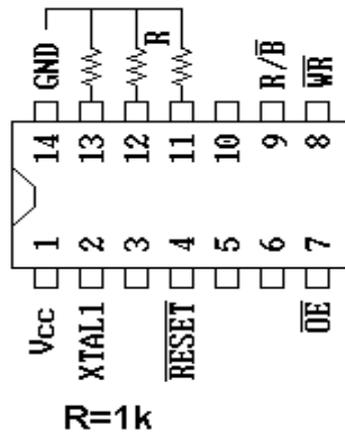
Pinbelegung zur HV-Programmierung (Quelle: Chan)

**DIP 8 Atmels:**

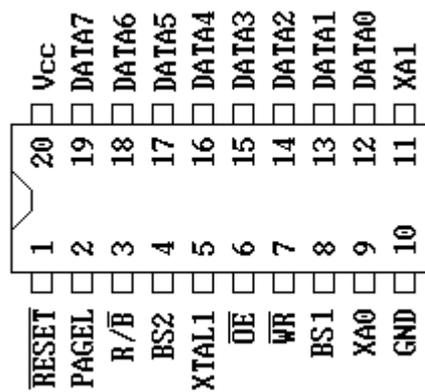




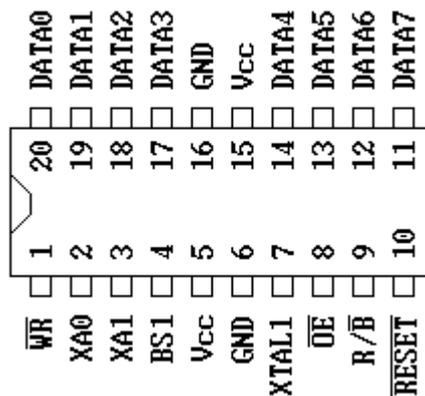
**DIP 14 Atmels (z.B. Tiny24):**

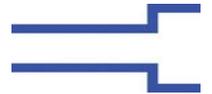


**DIP 20 Atmels (außer Tiny26!):**

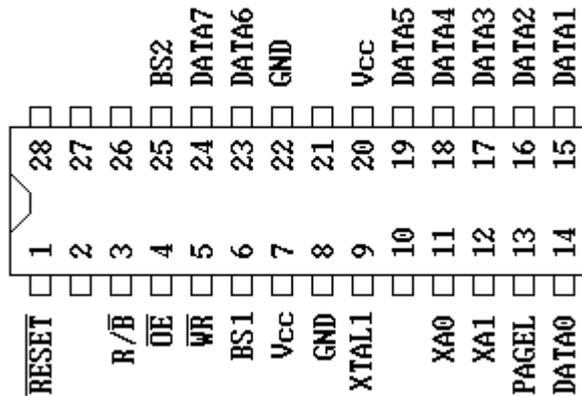


**DIP 20 Atmel Tiny26:**

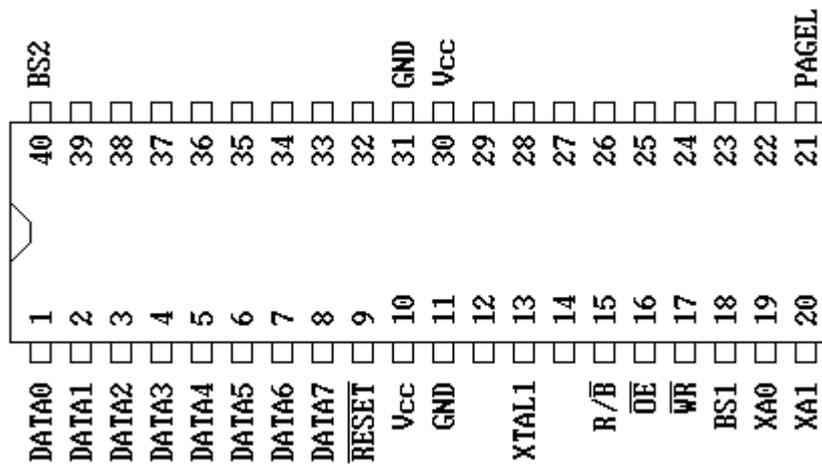




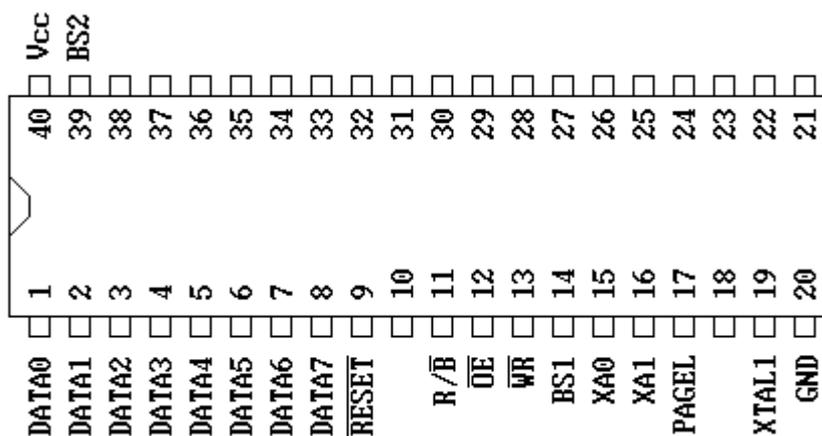
**DIP 28 Atmels:**

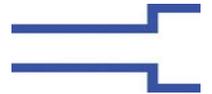


**DIP 40 Atmels (8535 basierend):**



**DIP 40 Atmels (8515 basierend):**





Unterstützte Controller (Quelle Chan)

AT90S1200	ATtiny10	Atmega161, Atmega162, ATmega165
AT90S2313	ATtiny11	Atmega169, Atmega8515, ATmega8535
AT90S4414	ATtiny12	Atmega163, Atmega323, ATmega8
AT90S8515	ATtiny15	Atmega16, Atmega32, ATmega48
AT90S4434	ATtiny22	Atmega88, Atmega168, ATmega325/9
AT90S8535	ATtiny13	Atmega3250/90, ATmega645/9
AT90S2333	Attiny25	Atmega6450/90, ATmega64
AT90S4433	ATtiny45	Atmega128, ATmega2561
AT90S2323	ATtiny85	Atmega603, ATmega103
AT90S2343	ATtiny24	AT90CAN32, AT90CAN64, AT90CAN128
	ATtiny44	Atmega406, AT90PWM2/3
	ATtiny84	Atmega644, Atmega640, ATmega1280
	Attiny2313	Atmega1281, ATmega2560
	ATtiny26	
	ATtiny28	

Benutzte Quellen:

HV-Programmer: chan, [http://elm-chan.org/works/avr\\_x/report\\_e.html](http://elm-chan.org/works/avr_x/report_e.html)

ISP-Programmer: Roland Walter, <http://www.rowalt.de>

GIVEIO.SYS: <http://www.irdeo.de/ntdriver.zip>

Eagle3D ist zu finden unter <http://www.matwei.de>

Den Komplettbausatz incl. Leiterplatte, Netzteil, Parallelkabel und CD kann man im Onlineshop bei [www.b-redemann.de](http://www.b-redemann.de) bestellen.

Für alle hier gemachten Angaben kann keine Garantie gewährleistet werden. Es wird keine Haftung für materielle oder sonstige Schäden übernommen.