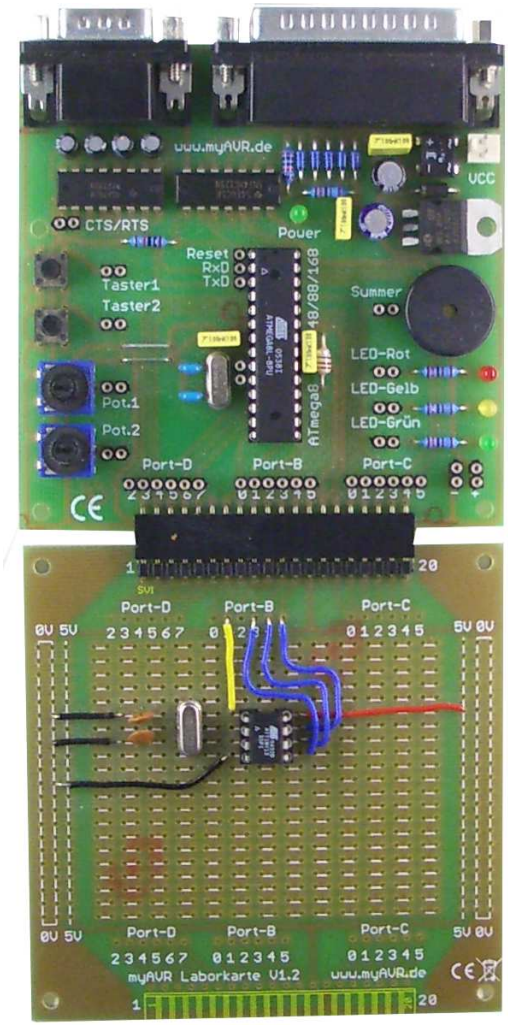


myTinyProg MK1 LPT

Erstellt von: Dipl. Ing. Toralf Riedel
Dipl. Ing. Päd. Alexander Huwaldt

Inhalt

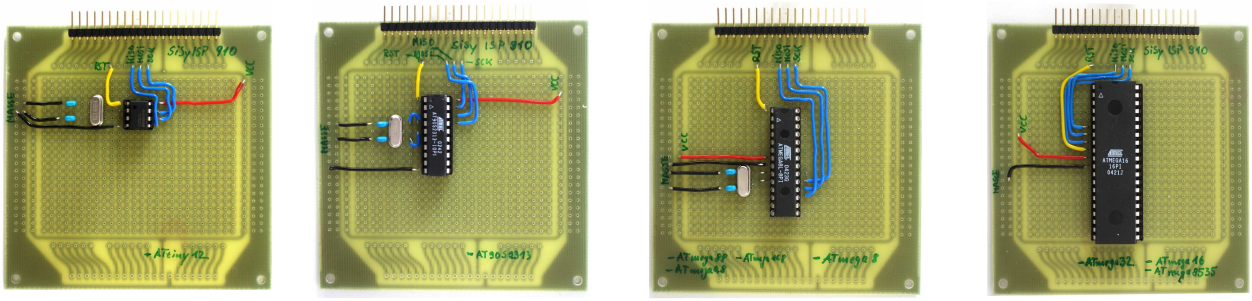
Promenadenring 8
02708 Löbau
Deutschland

www.myAVR.de
support@myavr.de

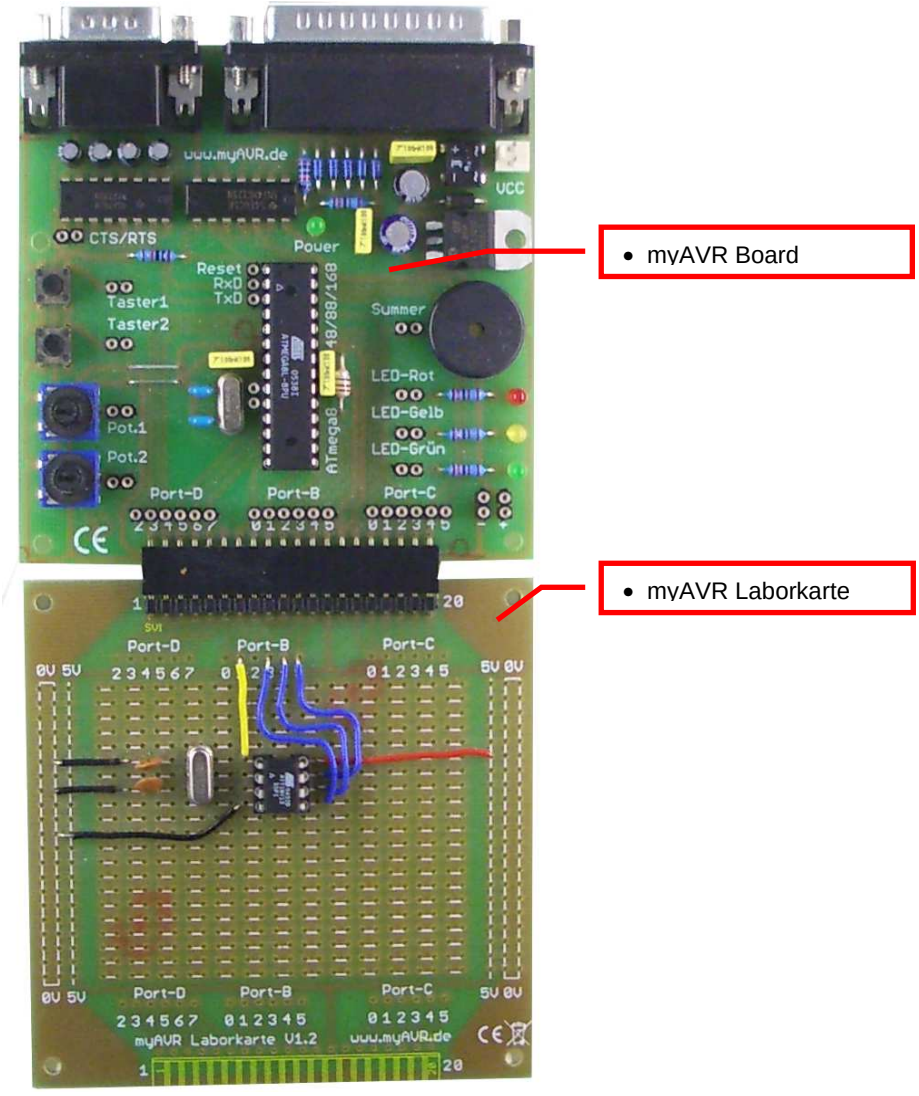
Tel: ++49 (0) 358 470 222
Fax: ++49 (0) 358 470 233

Einleitung

Das vorliegende Projekt myTinyProg MK1 LPT beschreibt eine Möglichkeit, das Lern- und Experimentiersystem myAVR als einen komfortablen seriellen AVR910-Programmer für AVR-Tiny Controller zu nutzen. Die Basis ist dafür ist eine Laborkarte A und wenige Bauteile, wie ein DIP8-Sockel, ein Quarz und zwei Keramikkondensatoren. Darüber hinaus soll das Projekt Anregungen geben, diesen Einsatzfall zum Beispiel auch für Controller der AVR-Mega-Serie weiter zu entwickeln.



Dieses Projekt ist geeignet für das myAVR Board MK1 LPT und das myAVR Board MK2 USB.



Wir wünschen Ihnen viel Erfolg und Spaß beim Lernen und Experimentieren.

Ihr myAVR-Team

Grundlagen

Für das Brennen (Programmieren, Flashen) des fertigen Mikrocontrollerprogramms (*.HEX oder *.BIN) gibt es grundsätzlich zwei Möglichkeiten:

Zum einen kann man ein Programmiergerät verwenden, in das man den Chip einsetzt und programmiert. Dazu muss das Programmiergerät über die parallele, serielle oder USB Schnittstelle an den PC angeschlossen werden. Der Mikrocontroller ist aus dem Zielsystem zu entfernen und auf den entsprechenden Sockel des Programmiergerätes zu stecken. Dann kann das Programm in den FLASH-Speicher des Controllers übertragen werden. War dieser Vorgang erfolgreich, kann der Controller aus dem Programmiergerät entnommen und wieder in das Zielsystem eingebaut werden. Ein solches Programmiergerät ist zum Beispiel der myMultiProg MK1 LPT von myAVR.

Eine weitere Lösung für die Programmierung des Mikrocontrollers ist das sogenannte „In System Programming“ (ISP). Dabei muss der Controller nicht aus dem Zielsystem ausgebaut werden, sondern kann direkt im System programmiert und getestet werden. Dafür muss das Zielsystem eine ISP-Schnittstelle bereitstellen. Mit einer zusätzlichen Hardware, dem sogenannten ISP-Programmer, der an dem LPT-Port, die COM-Schnittstelle oder dem USB-Port angeschlossen wird, kann aus der Entwicklungsumgebung heraus das Programm direkt in das Zielsystem übertragen werden. Als Experimentierhardware für 28-polige DIL Controller von ATMEL wie dem ATmega8 und Pinkompatible eignet sich das myAVR Board MK1 LPT oder myAVR Board MK2 USB. Für ATMEL AVR Controller mit anderen Pin-Zahlen wie z.B. der Tiny-Serie aber auch für andere Mega-Controller eignet sich die hier beschriebene Erweiterung zum myAVR Board auf Basis der Laborkarte A.

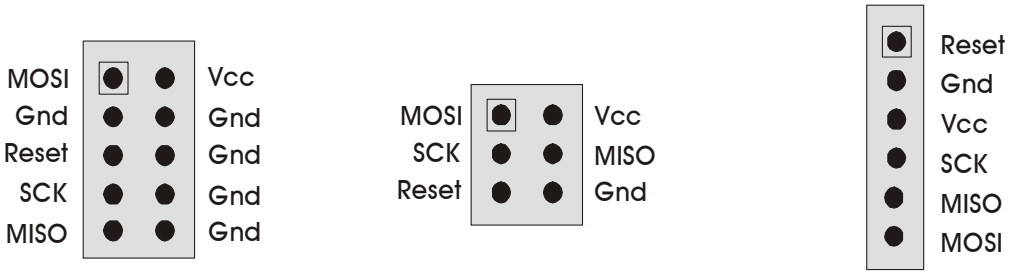


Abbildung: ISP- Anschlüsse, ISP- Standards, Atmel 10-polig, 6-polig, TwinAVR

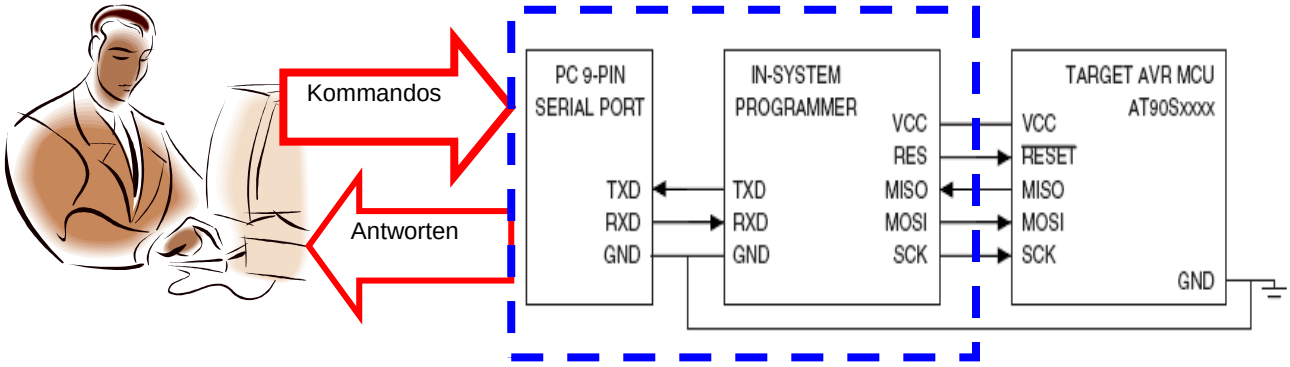
Die folgende Tabelle zeigt die Bedeutungen der einzelnen Signalleitungen der ISP-Schnittstelle. Diese sind die Leitungen des SPI (Serial Processor Interface).

Pin	Name	Comment
SCK	Serial Clock	Programming clock, generated by the in-System programmer (master)
MOSI	Master Out – Slave In	Communication line frome In-System programmer (master) to target AVR being programmed (slave)
MISO	Master In – Slave Out	Communication line from target AVR (slave) to In-System programmer (master)
GND	Common Ground	The two system must share the same commom ground
RESET	Target AVR MCU Reset	To eneble in-System programming, the target AVR Reset must be bept active. To simplify this, the In-System programmer should control the target AVR Reset.
Vcc	Target Power	of target operating an any volltage. The target can have power supplied througt the In-System programming connector for the duration of the programming cycle.

Für die Programmer-Hardware gibt es eine Reihe von konkreten Lösungen. Atmel selbst stellt im Application Note AVR910 eine Programmerlösung vor, die inzwischen von den meisten Programmierumgebungen unterstützt wird.

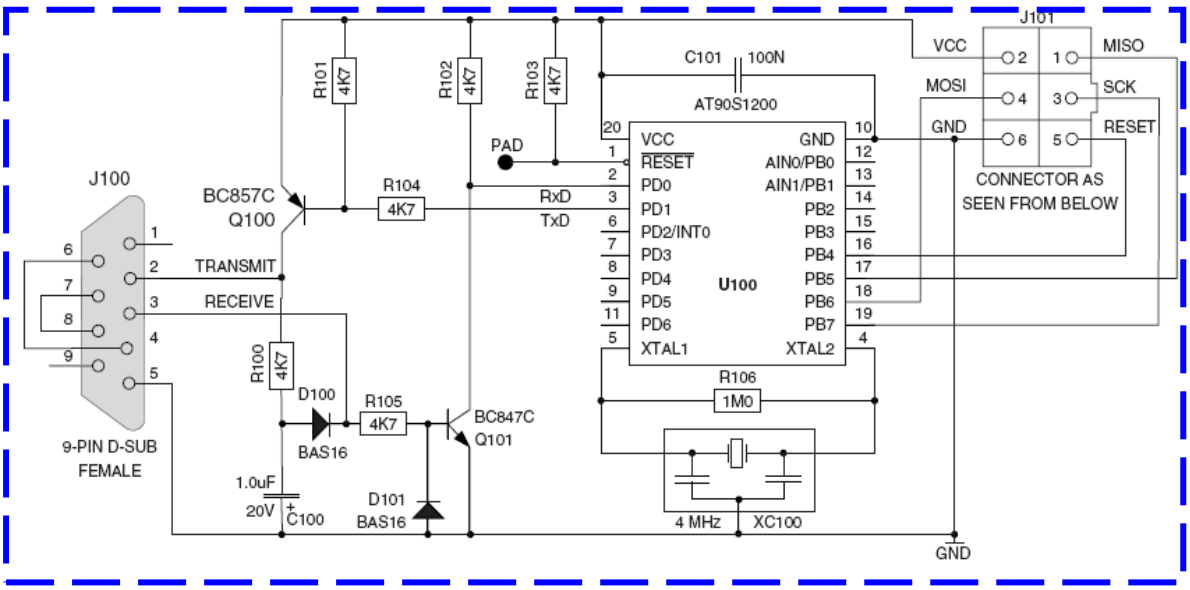
Atmel Application Note 910

Atmels Application Note AVR910 stellt zwei Aspekte einer einfachen Programmerlösung dar. Zum einen eine Hardwarelösung und zum anderen ein Softwareprotokoll. Die Hardware basiert auf einem RS232 Interface, einem AVR-Controller und dem eigentlichen ISP-Interface. Die Programmierumgebung sendet über den COM-Port an den Programmer Kommandos entsprechend dem AVR910 Protokoll. Diese werden durch die Firmware in ISP-Kommandos (ISP-Protokoll) umgesetzt. Da der AT90S1200 über kein SPI (Serial Processor Interface) verfügt, ist in der Firmware eine Softwarelösung dafür zu finden (Soft-SPI).



AVR910 Hardware

Die originale Hardwarelösung sieht im Detail wie folgt aus:
Sie besitzt eine einfache Pegelanpassung für die COM-Schnittstelle des PC (RS232) und einen AT90S1200. Die Firmware für den AT90S1200, welche das AVR910 Protokoll realisiert, stellt Atmel auch bereit.



Ausgehend von diesem Schema liegt der Schluss nahe, dass ein myAVR Board ja ebenfalls über diese Komponenten verfügt.

- Die Pegelanpassung für die serielle Schnittstelle übernimmt beim myAVR Board MK1 LPT ein RS232 Trei ber und für die Firmware haben wir einen ATmega8, der bereits über ein SPI verfügt.
  Also beste Vorraussetzungen, daraus einen AVR910 Programmer zu machen ;-)

AVR910 Firmware

Wichtiger als die eigentliche AVR910 Hardware ist die Softwarekomponente, die das Kommunikationsprotokoll mit der Entwicklungsumgebung realisiert. Die folgende Tabelle zeigt die AVR910 Kommandos, die per serielle Schnittstelle vom Brennprogramm an den Programmer gesendet werden.

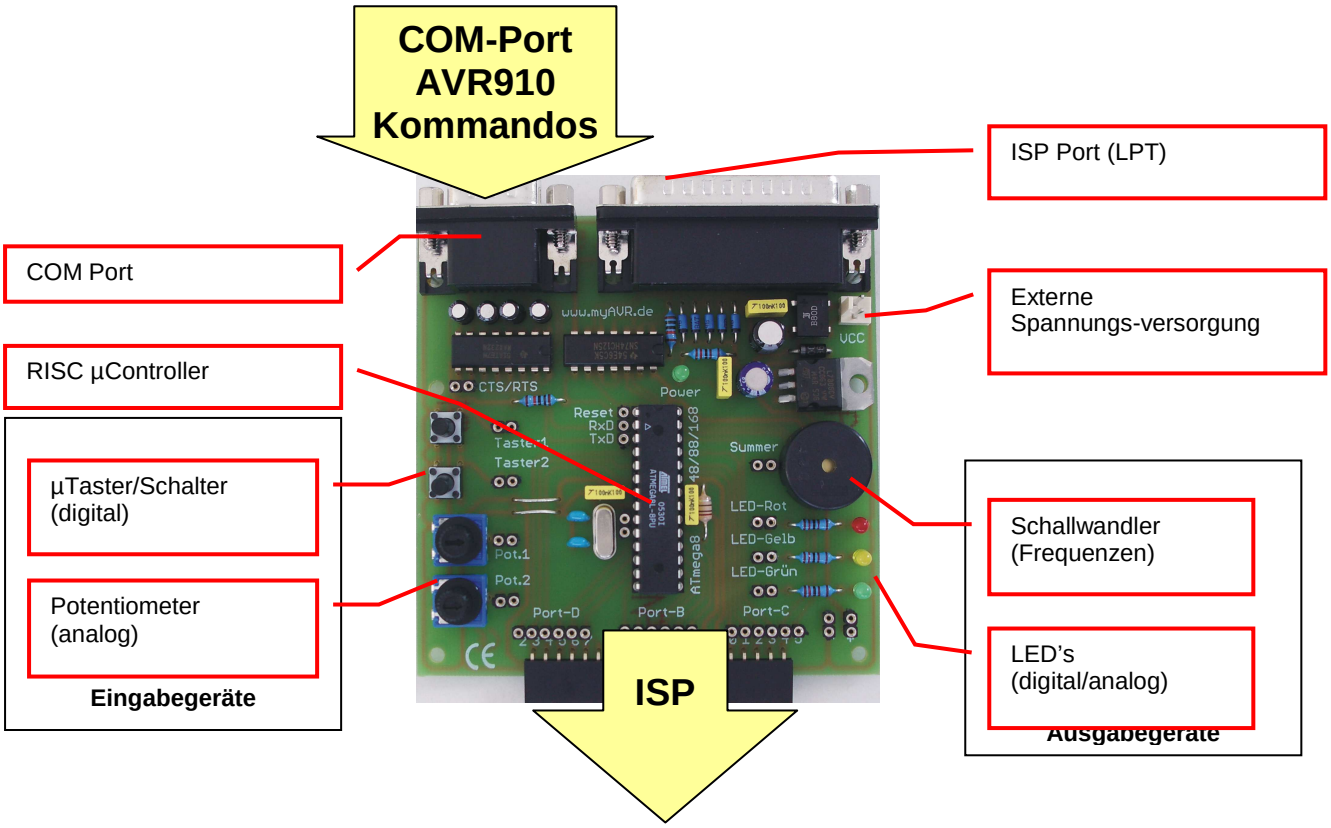
Kommando	Code	Beschreibung	wait
Programming Enable	**0xAC 0x53** ** **	Nach Power UP und RESET Low muss als erstes dieses Kommando gesendet werden, danach akzeptiert der Controller Programmier-Kommandos	
Chip Erase	**0xAC 0x8*** ** **	Bevor der Programmspeicher neu programmiert werden kann, ist dieser erst zu löschen. Dies bewirkt, dass jedes Byte mit 0xFF (alle Bits High) überschrieben wird. Der Programmiervorgang setzt dann nur noch die Low-Bits.	x
Write FLASH Low Write FLASH Higth	**0x60** adrh adrl byte **0x68** adrh adrl byte	Jeder Maschinenbefehl des AVR besteht aus 2 Byte (16 Bit Worte). Diese sind einzeln zu übertragen.	x
Read FLASH Low Write FLASH Higth	**0x20** adrh adrl ** **0x28** adrh adrl **	Jeder Maschinenbefehl des AVR besteht aus 2 Byte. Diese können nur einzeln gelesen werden.	
Write EEPROM	**0xC0** adrh adrl byte	Schreibt ein Byte in den EEPROM. Dieser ist byteorientiert.	x
Read EEPROM	**0xA0** adrh adrl **	Liest ein Byte aus dem EEPROM. Dieser ist byteorientiert.	
Read Vendor Code	**0x30** ** 0x00 **	Liest den Herstellercode aus. Atmel signiert mit 0x1e.	
Read Family Code	**0x30** ** 0x01 **	Liest den Gruppencode aus. Der Mega8 liefert zum Beispiel 0x07. Darin ist Controllerfamilie und FLASH-Size codiert.	
Read Part Number	**0x30** ** 0x02 **	Liest den Teilcode aus. Der Mega8 liefert zum Beispiel 0x93. Aus Herstellercode, Gruppencode und Teilcode ergibt sich die Controllersignatur Mega8 = 0x1e9307	

Einige Kommandos erfordern controllerspezifische Ausführungszeiten. Diese können den Datenblättern der Controller entnommen werden.

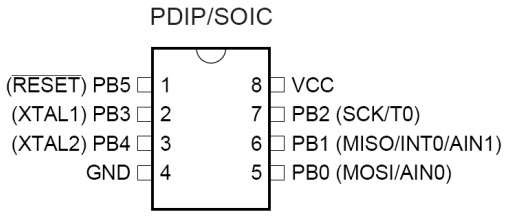
Hardware myTinyProg MK1 LPT

Hardwareüberblick

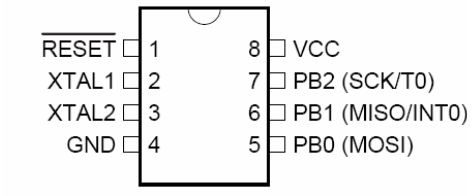
Das myAVR Board MK1 LPT besitzt folgende Struktur:
LPT - Programmer, RS232 Interface, ATmega8 Controller, Peripherie, Erweiterungsbuchse. Bei diesem Projekt soll das myAVR Board als AVR910 Programmer arbeiten.



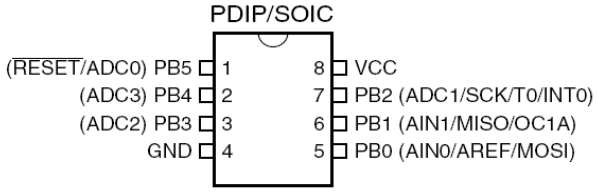
ATtiny12



AT90S2323



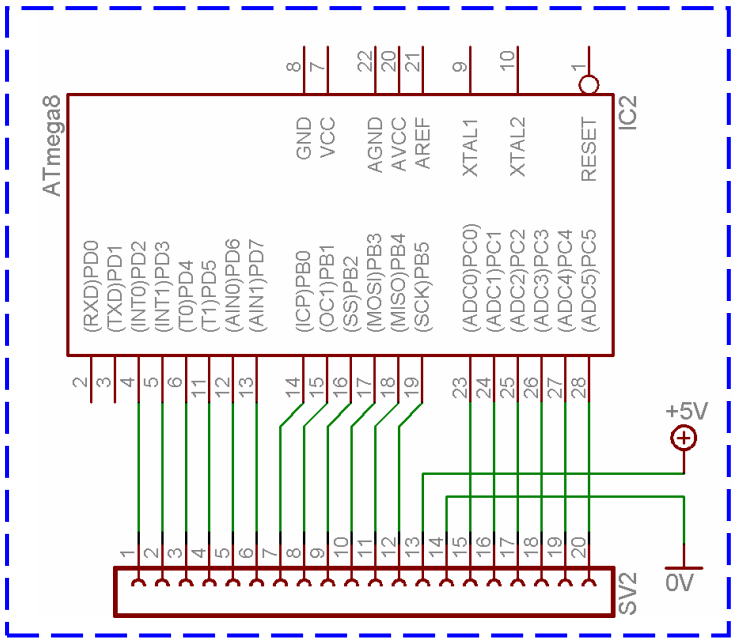
ATtiny15



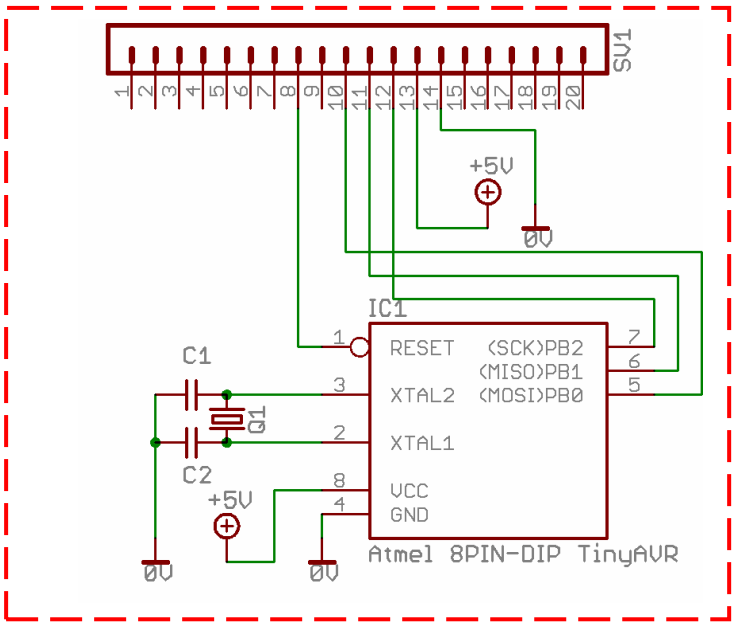
Schaltplan myTinyProg MK1 LPT

Die Schaltung soll die Programmierung der Atmel Controller AT90S2323, ATtiny12, 13 und 15 ermöglichen. Dabei sind die ISP Leitungen entsprechend der PIN-Belegung der Controller und der Erweiterungsbuchse des myAVR-Boards zu verbinden. Die SPI Signale sind am Mega8 auf Port B 3, 4 und 5 verfügbar.

myAVR Board MK1 LPT



Laborkarte



Die „Rückführung“ der Portleitungen PB0 bis PB5 zur Stiftleiste an Positionen Port C.0 bis Port C.5 sind hier nicht dargestellt.

Stückliste

Das myAVR Board besitzt folgende Struktur:
Peripherie zum Testen von Anwendungen, Erweiterungsbuchse.

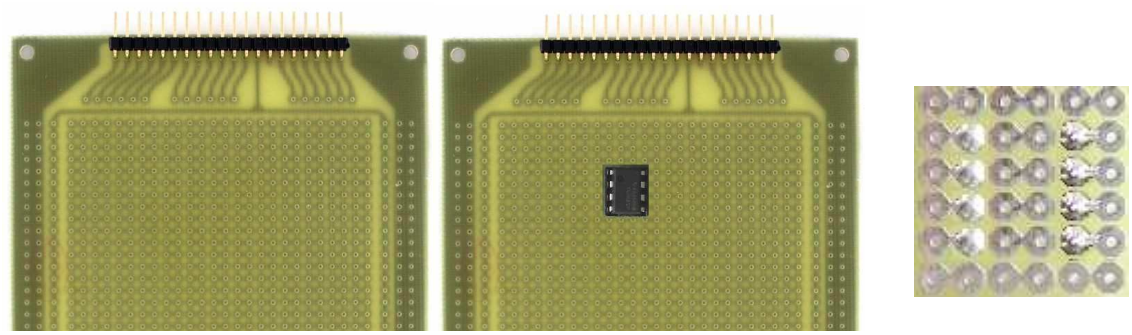
Bezeichnung im Schaltplan	Bauteil	Artikelnummer Suchbegriff	Anzahl
IC1	ATtiny12/13/15 DIP	ATTINY 13-20 DIP (www.myavr.de bzw. www.reichelt.de)	1
J1	IC Sockel 8 DIP	GS 8 oder GS 8P (www.reichelt.de)	1
X1	Laborkarte A (Platine)	myAVR Laborkarte (www.myAVR.de)	1
SV1	Stiftleiste 20polig	Stiftleiste 20polig (www.myAVR.de)	1
	Schaltdraht ca. 20 Zentimeter (Farbsortiment)	Patchkabel (www.myAVR.de)	

Nach Bedarf für externen Takt:

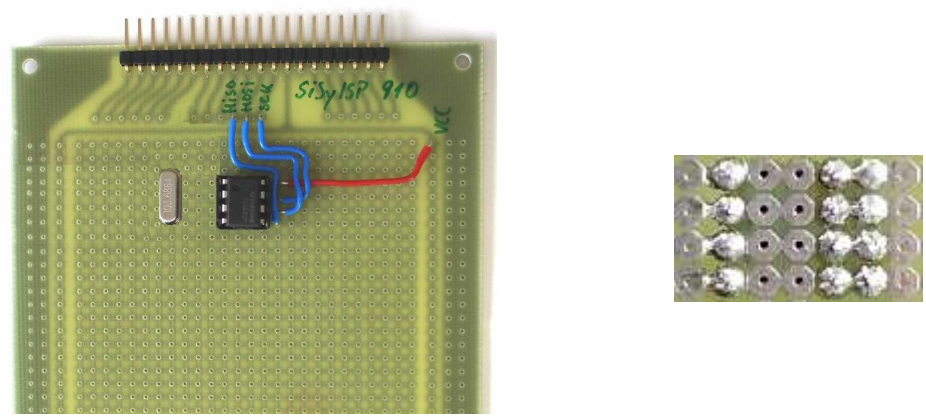
C1, C2	KERKO 33pF	KERKO 33P (www.reichelt.de)	2
Q1	Quarz 3,6864 MHz	Quarz 3,6 (www.myAVR.de)	1

Realisierung auf einer myAVR Laborkarte A

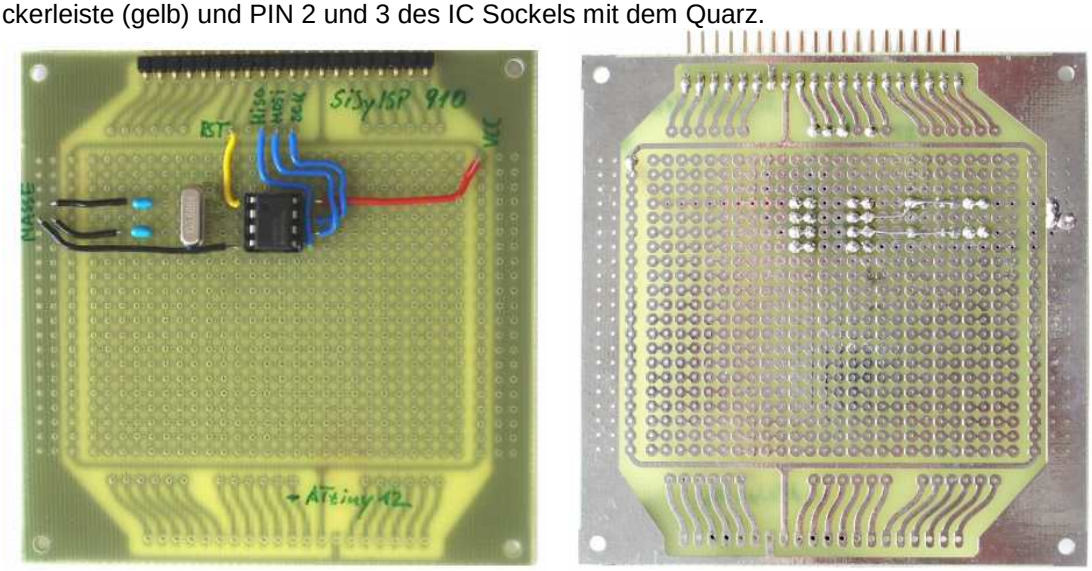
1. Löten Sie die Buchsenleiste und den IC-Sockel auf die Laborkarte. Achten Sie darauf, dass die paarigen Lötaugen der Laborkarte den einfachen Anschluss gewährleisten.



2. Verbinden Sie die PINS 5,6 und 7 des IC Sockels mit Port B 3,4 und 5 an der Steckerleiste (blau) und PIN 8 des IC Sockels mit der Versorgungsspannung (rot).



3. Ergänzen Sie Quarz und Kondensator und verbinden Sie PIN 1 des IC Sockels mit Port B.1 an der Steckerleiste (gelb) und PIN 2 und 3 des IC Sockels mit dem Quarz.



4. Stellen Sie alle Verbindungen entsprechend des Schaltplans her. Danach kann der Controller (z.B. ATtiny13) in den IC Sockel eingesetzt werden.

myTinyProg Firmware

Die AVR910 Firmware für dieses Projekt ist als HEX-Datei *myTinyProg.hex* im Downloadbereich von www.myAVR.de verfügbar. Diese unterstützt unter anderem die Controller AT90S2323, ATtiny12, ATtiny13, ATtiny15 aber auch eine Reihe Mega AVR und setzt die oben beschriebene Schaltung voraus.

Port B.1 = RST
Port C.0 = rote LED (per Patchkabel auf dem myAVR Board)
Port C.2 = grüne LED (per Patchkabel auf dem myAVR Board)
Port B.3 = MISO
Port B.4 = MOSI
Port B.5 = SCK

Beschreibung der Firmware

Es werden unter anderen folgende AVR910 Kommandos unterstützt:

- Enter programming mode	'P'
- Set address	'A'
- Write program memory, low byte	'c'
- Write program memory, high byte	'C'
- Read program memory	'R'
- Write data memory	'D'
- Read data memory	'd'
- Chip erase	'e'
- Leave programming mode	'L'
- Read signature bytes	's'
- Return supported device codes	't'
- Return software identifier	'S'
- Return sofware version	'V'
- Return hardware version	'v'
- Return programmer type	'p'
- Set LED	'x'
- Clear LED	'y'

Der Betriebszustand des *myTinyProg* wird über zwei LEDs angezeigt

- rote LED = betriebsbereit,
- grüne LED = Brennvorgang.

Die Einstellungen für den COM-Port des PC's müssen sein:

- 19200 Baud,
- 8 Datenbits,
- 1 Stoppbit,
- keine Parität sein.

Brennen der Firmware

Die Firmware muss zuerst wie bisher über den LPT-Port oder mySmartUSB MK2 auf den Mega8 des myAVR Boards gebrannt werden.

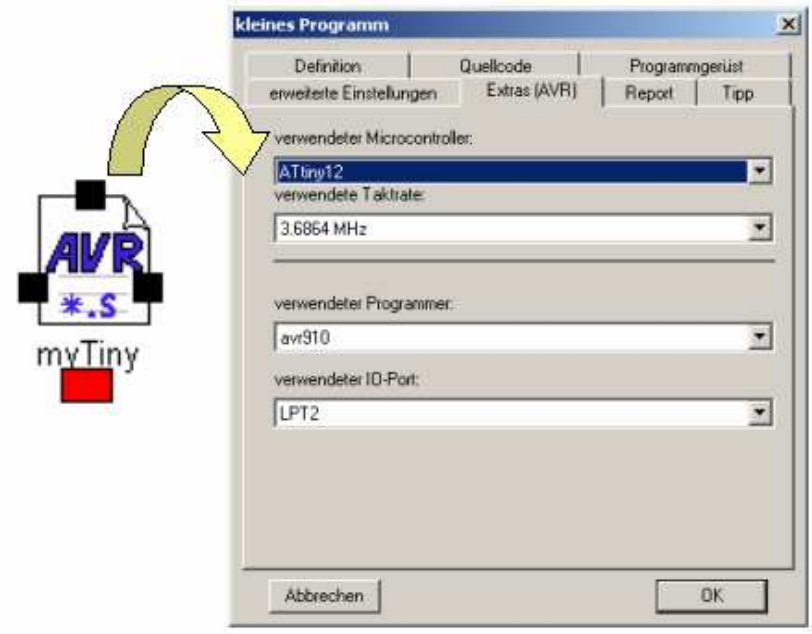
Brennen (Programmieren) des Tiny's

myTinyProg MK1 LPT wird mittels der Buchsenleiste mit dem myAVR Board verbunden.

myAVR Board MK1 LPT
Das LPT-Kabel ist abzuziehen und der serielle Anschluss wird zum Programmieren des Tiny's benutzt.

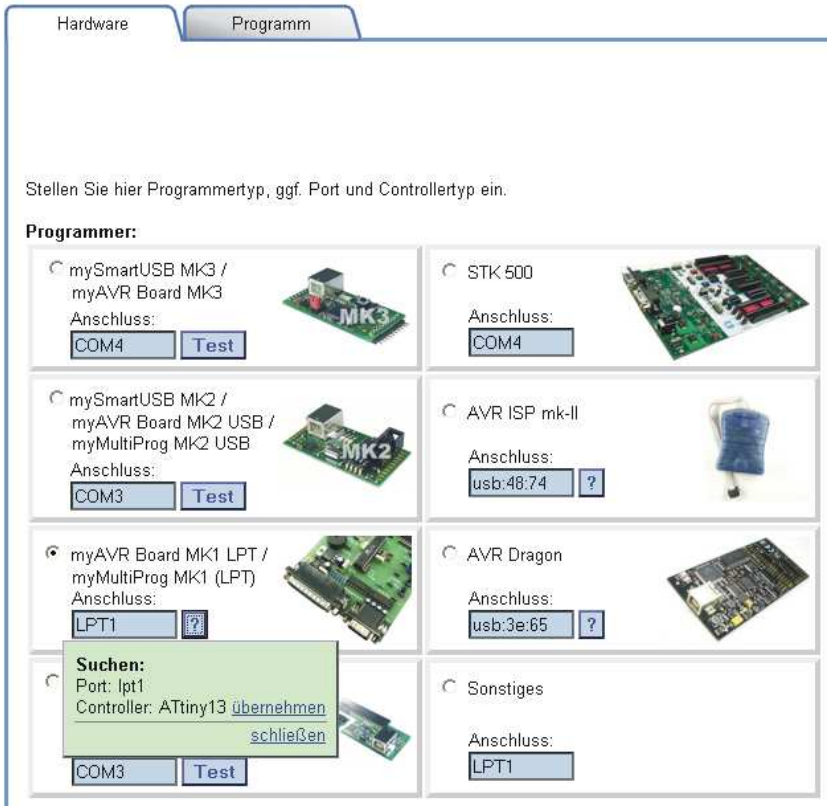
myTinyProg mit SiSy AVR

In SiSy AVR können Sie myTinyProg jetzt als AVR910 Programmer ansprechen. Dazu sind auf dem entsprechenden Programmmodul im Dialog „Definieren“ unter „Extras (AVR)“ die Eintragungen für den verwendeten Mikrocontroller, die Taktrate, den Programmer und den IO-Port einzutragen. Die Abbildung zeigt ein mögliches Beispiel.



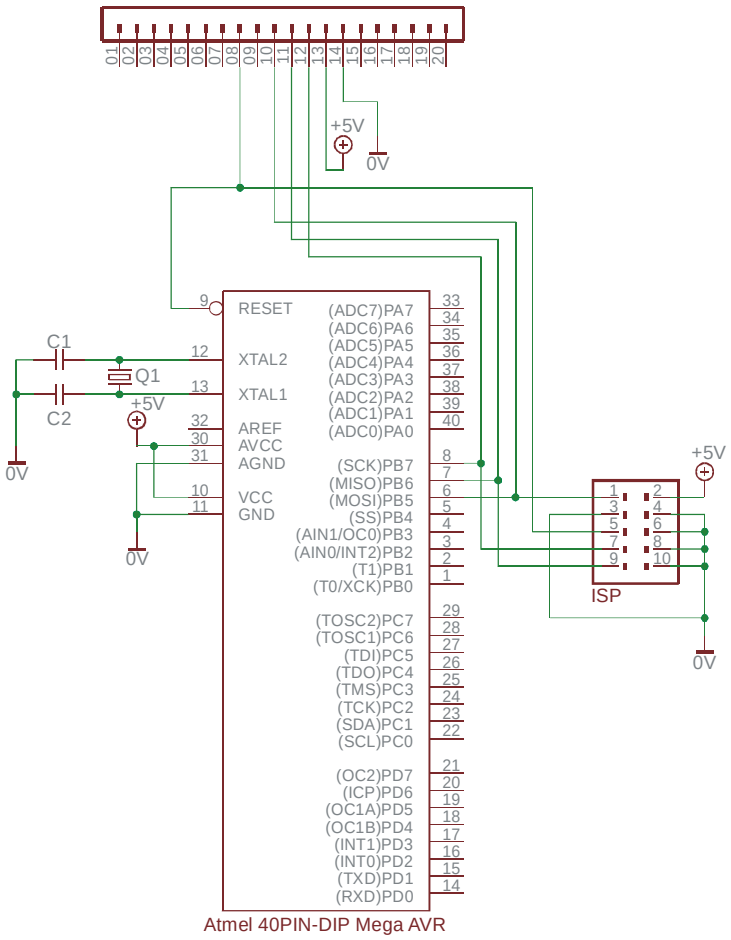
myTinyProg mit mAVR Workpad PLUS

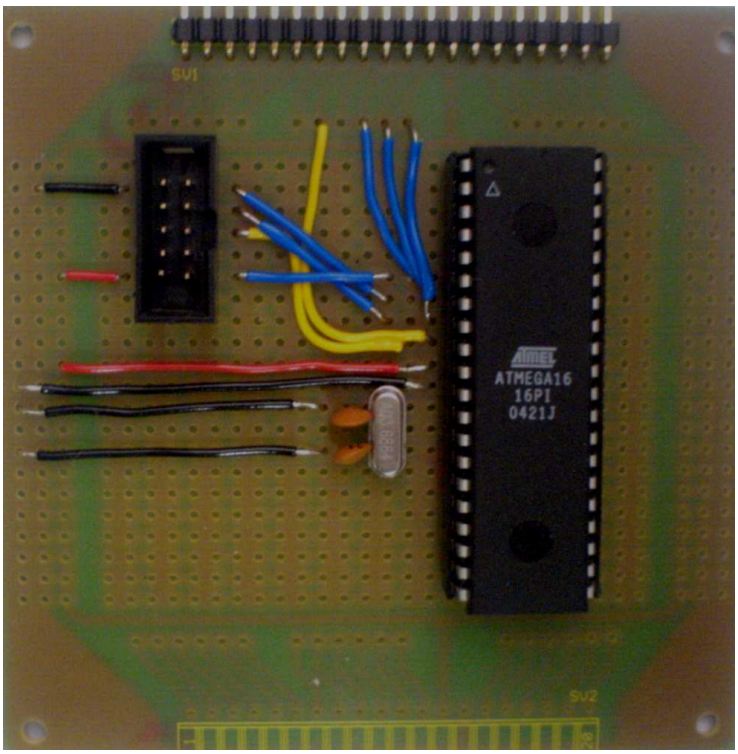
Mit dem Programm myAVR Workpad PLUS können Sie jetzt ebenfalls myTinyProg als AVR910 Programmer ansprechen. Die Eintragungen für den verwendeten Mikrocontroller, Programmer und Port können Sie unter der Menüfolge *Extras/Einstellungen* vornehmen oder Sie hinterlegen diese Angaben im Dateikopf. Entsprechende Hinweise dazu erhalten Sie ebenfalls unter der Menüfolge *Extras/Einstellungen*.



Abbildungen können vom Inhalt abweichen. Änderungen im Sinne des technischen Fortschrittes behält sich der Hersteller vor

Projektbeispiel für einen ATmega16





Quellenverzeichnis

www.atmel.com
http://www.atmel.com/dyn/resources/prod_documents/DOC0931.PDF
http://www.atmel.com/dyn/resources/prod_documents/doc2521.pdf
http://www.atmel.com/dyn/resources/prod_documents/doc2585.pdf
http://www.atmel.com/dyn/resources/prod_documents/DOC0943.PDF
http://www.atmel.com/dyn/resources/prod_documents/AVR910.zip
http://www.atmel.com/dyn/resources/prod_documents/doc1006.pdf
http://www.atmel.com/dyn/resources/prod_documents/doc2535.pdf
http://www.atmel.com/dyn/resources/prod_documents/doc1187.pdf

www.sisy.de
www.myAVR.de