Programmiershield für ATtiny 2x/4x/8x

basierend auf Arduino Uno als Programmer

Christoph Schwärzler, OE1CGS

November 2015

Hardware

Der Programmer dient der Programmierung von Mikrocontrollern der Typen ATtiny 24, 25, 44, 45, 84 und 85 der Firma Atmel. Diese können sowohl im PDIP als auch im SOIC Gehäuse programmiert werden.

Durch einen Umschalter können die Mikrocontroller noch im Programmer ersten Funktionstests unterzogen werden.

Der Programmer wurde als Arduino Shield für den Arduino Uno in Fritzing entworfen und die Platine über Fritzing angefertigt (Bild 1). Sowohl das Fritzing-File[1] als auch die entsprechenden Gerber-Files[2] können von meiner Website geladen werden.

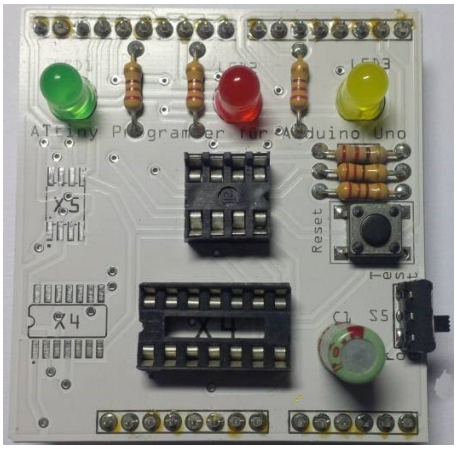

Bild 1: Die Hardware des ATtiny Programmierers

Das Shield wird für die Programmierung der Mikrocontroller auf einen Arduino Uno gesteckt und von diesem versorgt sowie gesteuert. Leuchtdioden informieren über den Programmierfortschritt und –erfolg. Diese haben die folgende Bedeutung:

Grüne LED:	Heartbeat (Betriebsbereitschaft)
Gelbe LED:	Programming (Kommunikation)
Rote LED:	Error (Fehlerhafte Programmier.)

Das Programmiershield verfügt auch über einen Reset-Taster für den zu programmierenden µC.

Die Pins 2 (PB3/PB0) und 3 (PB4/PB1) sind über Schutzwiderstände von 220 Ω an die Arduino Pins 1 (TX) und 0 (RX) herausgeführt. Damit kann im Testmodus z.B. die serielle Schnittstelle des Arduino sofort zum Testen des µC noch direkt auf dem Programmiershield benutzt werden. Dies wird auch für die Kalibrierung (siehe unten) benutzt.

Die zu programmierenden Mikrocontroller werden für die Programmierung entweder in den Sockel gesteckt (PDIP), oder vorübergehend auf die Platine des Shields gepresst (SOIC). Dies kann z.B. mit einer Wäscheklammer erfolgen.

Programmierung

ArduinoISP installieren

Vor dem Aufstecken der Programmiershields muss auf den Arduino Uno der Sketch „ArduinoISP“ aufgespielt werden. Dieser befindet sich bereits in der Arduino IDE unter „Datei/Beispiele“.

Wurde im Mikrocontroller (z.B. aus Versehen) die Fuse gesetzt die ihn in eine Taktrate kleiner als 1 MHz gesetzt hat (z.B. durch den watchdog timer bei 128 kHz), so kann der µC dem SCK-Signal des Originalsketches nicht mehr folgen. Dann muss alternativ der modifizierte Sketch „ArduinoISP_slow_SCK“ benutzt werden und dort ein Teiler gewählt werden, der die Clockrate genügend verlangsamt.

Nachdem der Arduino Uno mit dem Programmiersketch versehen ist, wird die Stromversorgung (USB-Kabel) getrennt und das Programmiershield aufgesteckt. Dabei ist die Beschriftung identisch ausgerichtet wie diejenige des Arduino Uno Boards.

Spätestens jetzt muss der Schalter S5 auf „Prog“ gestellt werden.

Nach dem Einschalten der Stromversorgung leuchten alle drei Leuchtdioden in der Reihenfolge Gelb-Rot-Grün kurz auf bevor die grüne LED (Heartbeat) dauerhaft langsam blinkt. Ist dies der Fall, so ist der Arduino bereit.

Einsetzen des Mikrocontrollers

Nun wird die Stromversorgung (USB-Kabel) wieder getrennt, bevor der µC auf das Programmiershield gesetzt wird.

Insgesamt gibt es vier Fassungen/Kontaktfelder, wo ein µC eingesetzt werden kann. Dies sind zwei Fassungen für µC im DIP-Gehäuse (8 bzw. 14 Anschlüsse) und 2 Kontaktfelder für SOIC-Gehäuse (ebenfalls 8 bzw. 14 Anschlüsse). Wenn SMD-Mikrocontroller programmiert werden, müssen diese genau auf die Kontaktfelder ausgerichtet werden und dann mit mittlerem Druck auf diesen fixiert werden (z.B. mit einer Wäscheklammer). In jedem Fall ist auf die korrekte Richtung zu achten, dazu sind auf der Platine kleine Punkte bzw. Einkerbungen in der Fassung bei Pin 1 vorhanden. Vorsicht bei 8-poligen DIP Gehäusen, diese sind genau verkehrt zu den anderen drei Fassungen/Kontaktfeldern einzustecken!

Nachdem der µC angebracht ist, kann das USB-Kabel wieder verbunden werden. Die LED’s müssen sodann wie oben beschrieben aufleuchten.

Auswahl des µC in der Arduino IDE

Die folgenden Anweisungen beziehen sich auf die Version 1.6.2 der Arduino IDE.

Nun muss unter Werkzeuge/Platine der zu programmierende µC ausgewählt werden. Gleichzeitig wird hier die Taktquelle und Frequenz ausgewählt (Bild 2). Standard ist dabei der interne Oszillator bei 8 Mhz.

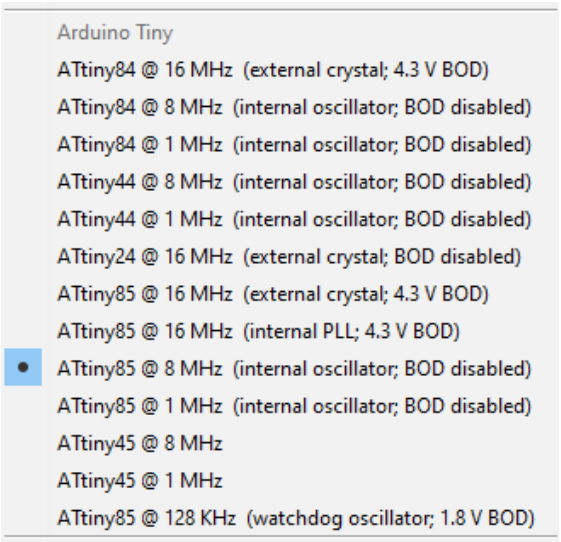

Bild 2: Auswahl des Mikrocontrollers unter Werkzeuge/Platine

Als nächstes muss unter Werkzeuge/Programmer der „Arduino als ISP“ an Stelle des sonst üblichen Programmers (typisch „AVRISP mkII“) ausgewählt werden (Bild 3).

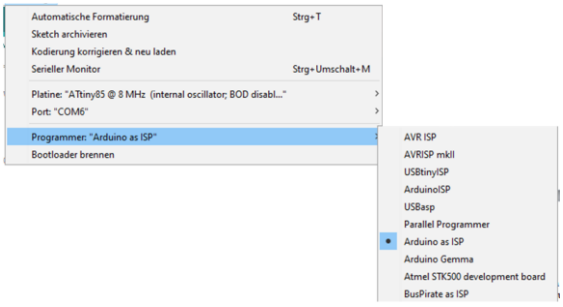

Bild 3: Auswahl des Programmers unter Werkzeuge/Platine

Damit sind die Einstellungen fertig.

Bootloader brennen (wenn erforderlich)

Vor dem erstmaligen Programmieren eines neuen µC oder wenn die Einstellungen der Platine (z.B. Taktrate) geändert wurden ist es erforderlich, dass der Befehl „Bootloader brennen“ ausgeführt wird. Damit wird zwar kein echter Boootloader auf die ATtiny’s gebrannt, aber es werden die entsprechenden Fuses gesetzt um im vorgesehenen Modus zu arbeiten.

Soll nur ein neues Programm auf einen bereits beschriebenen µC im identen Modus geladen werden, so kann dieser Schritt ausgelassen werden.

Oszillator kalibrieren (wenn erforderlich)

Der interne RC-Oszillator der ATtiny’s ist fabrikseitig voreingestellt, kann aber für jedes Exemplar ganau kalibriert werden. Dies ist insbesondere dann erforderlich, wenn der µC später mit anderen Bauteilen über die getaktete Schnittstellen (wie z.B. der serielle Port) kommunizieren soll, oder die internen Zeitzähler mit hoher Genauigkeit benutzt werden sollen.

Dazu gibt es die Variable „OSCCAL“, in welche bei jedem Programmbeginn in void setup der für diesen Baustein ermittelte Wert ZZ geschrieben wird. Dieser wird üblicherweise als Hexadezimalzahl angegeben:

OSCCAL = 0xZZ;

Doch zuerst muss dieser Wert einmalig für jeden µC ermittelt werden. Dies geschieht am einfachsten mit dem Sketch „Interaktiv_to Serial_with_Details“ des TinyTuners (Arduino IDE / Datei / Sketchbook / libraries /TinyTuner / Interaktiv to Serial with Details). Für µC mit kleinem Speicher (2x, 4x) ist dieser Sketch zu groß; hier muss Interaktiv to Serial benutzt werden.

Nachdem der Sketch hochgeladen wurde (siehe folgendes Kapital) muss der Schalter S5 auf dem Programmiershield in die Stellung „Test“ gebracht werden und auf der Arduino IDE der serielle Monitor mit der Einstellung 9600 Baud geöffnet werden. Nachdem das Programm mit einem Druck auf die Reset-Taste auf dem Programmiershield neu gestartet wurde müssen auf dem Monitor Zeichen erscheinen. Ist der voreingestellte Wert für den Oszillator weit vom korrekten Wert entfernt, so kann es sein, dass die Zeichen „sinnlos“ sind. Jetzt muss jedenfalls auf der Tastatur die Taste „x“ eingegeben und über den seriellen Monitor gesendet werden. Spätestens nach der zweiten oder dritten Eingabe von „x“ sollten die Zeichen lesbar werden. Die Iteration wird beendet, wenn der geeignete Wert für OSCCAL gefunden ist. Dieser wird im Monitor ausgegeben und wird notiert sowie in spätere Sketches für diesen µC eingebaut.

Programm hochladen

Nun wird der gewünschte Sketch geladen, ggf. verifiziert und wie üblich hochgeladen.

Während die grüne LED (etwas verlangsamt) weiterblinkt, leuchtet kurz die rote LED auf um dann von der gelben LED abgelöst zu werden. Diese

flackert während der Programmierung (kurz bei kleinen Programmen und länger bei umfangreichen) und erlischt bei Abschluss. Blinkt anschließend nur wieder die grüne LED im gewohnten Rhythmus, so war die Programmierung erfolgreich. Wenn jedoch zusätzlich die rote LED dauernd leuchtet, so war ein Fehler beim Brennen des Mikrocontrollers.

Testing

Erste Funktionstests können onboard erfolgen. Dazu wird der Schalter S5 auf dem Shield in die Stellung „Test“ gebracht. Dadurch wird der Arduino Uno durch einen low-pegel auf einen permanenten Resetstatus (I/O hochohmig) gebracht und es können die Anschlüsse des ATtiny daher ohne Störung durch den Arduino benutzt werden. Sämtliche LED’s auf dem Programmiershield sind jetzt ausgeschalten.

Die Pins 2 und 3 der µC sind über Schutzwiderstände an die Digital-Pins 0 und 1 des Arduino geführt. Diese sind gleichzeitig die Pins für RX und TX der seriellen Schnittstelle. Daher kann der serielle Monitor direkt für Testzwecke eingesetzt werden.

Eine weitere Möglichkeit des Testens besteht darin, Drahtbrücken aus den Pins der jeweils freien PDIP Fassung zu führen, die mit den zu prüfenden Pins des eingesetzten µC verbunden sind (Bild 4).

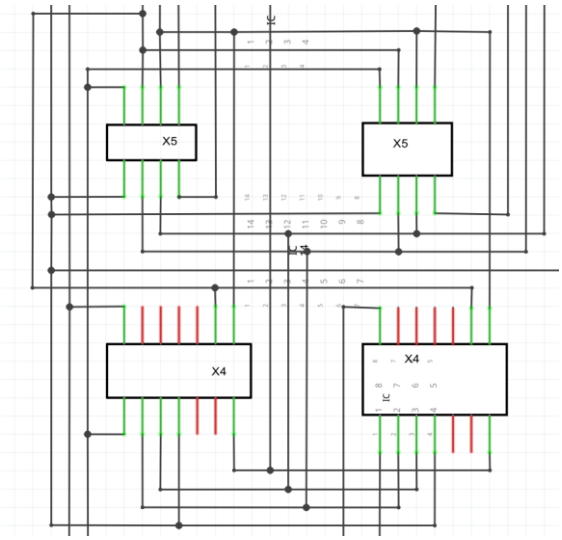

Bild 4: Verbindungen zwischen den Fassungen/Kontaktfeldern. Details siehe Fritzing-File.

Diese können dann mit Tastern oder LED’s zur Ein/Ausgabeprüfung während des Programmablaufs benutzt werden.

[1] *www.oe1cgs.at/download/ATtiny programmer.fzz*
[2] *www.oe1cgs.at/download/ATtiny programmer Gerber.zip*