

Übung 11: SW Download, Debugging

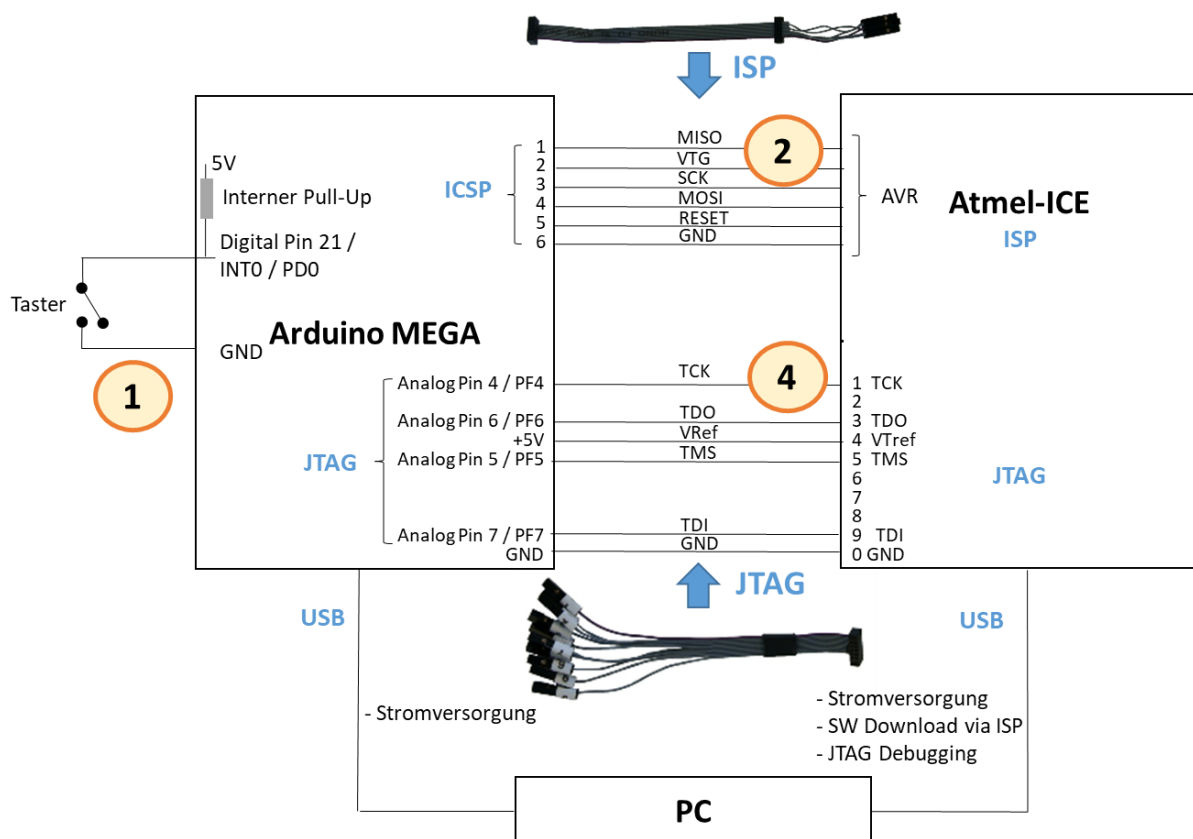
Nur die Aufgabe 2 können Sie ohne zusätzliche Hardware (Atmel-ICE) durchführen. Das Übungsblatt wird deshalb in der Zentralen Fragestunde als Live Demo vorgeführt.

Hardware:

- Basis: Arduino, Steckbrett, Kabel
- Atmel-ICE und Kabel für ISP-Programmierung

Informationen:

- Datenblatt ATmega2560: Kapitel 27, 28, 30. *JTAG und Memory Programming*
- Atmel-ICE, Kabel für ISP-Programmierung



Aufgabe 1: Vorbereitung

- Verbinden Sie einen Taster mit dem Arduino Mega (Teil ① der Abbildung). Durch den internen Pull-Up ist kein extra Widerstand notwendig.
- Extrahieren Sie das mitgelieferte zip-Archiv und öffnen Sie das Atmel-Studio¹ Projekt `ISP-JTAG.atsln`. Dieses Übungsblatt funktioniert **nicht** mit der Arduino IDE.
- Das Programm in der Datei `main.c` soll jeden Tastendruck an *Digital Pin 21* (*INT0*) erkennen². Jeder Tastendruck erhöht einen interne Zähler (Variable counter), der aktuelle Zählerstand wird periodisch über die serielle Konsole ausgegeben. Versuchen Sie das Programm grob zu verstehen.

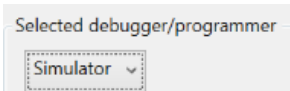

¹ Eigene Installation möglich, optional! <https://www.microchip.com/mplab/avr-support/atmel-studio-7>

² Entprellung ist für dieses Übungsblatt nicht verlangt.


Aufgabe 2: In-System Programming (ISP) über SPI

- a) Stellen Sie die ISP-Verbindung zwischen Arduino Mega und dem Atmel-ICE wie in Teil ② der Abbildung dargestellt her.
- b) Verbinden Sie den Atmel-ICE per USB **mit dem PC auf dem Atmel Studio läuft!** Für den Arduino Mega genügt eine beliebige USB Stromversorgung, notfalls ein 2. PC!
- c) Atmel Studio muss für den Betrieb mit einem ISP-Programmer konfiguriert werden.
- Wählen Sie den Menüpunkt „*Project-ISP-JTAG-Eigenschaften*“
 - Überprüfen, dass unter „*Device*“ der ATmega2560 ausgewählt ist.
 - Unter „*Tool-Selected debugger/programmer*“ den Atmel-ICE auswählen und als „*Interface*“ ISP auswählen.
 - Ggfs. beim Dozenten melden, falls Labor PC das Admin-Passwort benötigt.
- d) Laden Sie das Programm per ISP – nicht per Bootloader – auf den Mikrocontroller. Drücken Sie auf den grünen Pfeil, siehe rechts. Das startet die ISP Programmierung, das Programm wird nun per ISP direkt in das Flash geschrieben. 
- e) Testen Sie Ihr Programm, indem Sie die serielle Konsole (siehe rechtes Symbol) beobachten und dabei den Taster drücken. **Achtung:** Hierzu müssen Sie **kurzfristig (!!!)** eine direkte USB-Verbindung zwischen Ihrem Laptop und dem Arduino Mega herstellen. 
- f) Jeder Atmel Mikrocontroller hat einen eindeutigen Signaturcode. Lesen Sie den Signaturcode über die ISP-Schnittstelle aus und vergleichen Sie mit Seite 328 des Handbuchs!
- „*Tools*“, „*Device Programming*“
- g) Umgekehrter Weg: Laden Sie per *ISP* unter „*Tools-Device Programming*“ das aktuell installierte Programm aus dem Flash des Mikrocontrollers als HEX-Datei auf Ihren PC³. Wie viele Datenbytes sichert eine einzelne Checksumme jeweils ab?
- h) Standardmäßig ist auf dem Arduino JTAG aktiviert (Fuse JTAGEN). Jedoch ist das On-Chip Debug System (Fuse OCDEN) häufig deaktiviert. Um die entsprechende Fuse zu programmieren (Dialog „*Fuses*“), wird nun ebenfalls ISP eingesetzt. Es muss wie rechts abgebildet aussehen.
- | | | |
|---|-------------|-------------------------------------|
|  | HIGH.OCDEN | <input checked="" type="checkbox"/> |
|  | HIGH.JTAGEN | <input checked="" type="checkbox"/> |


Aufgabe 3: Debugging über Simulation

- a) Wählen Sie als „*Debugger/Programmer*“ im Menü „*Project*“-„*ISP-JTAG-Eigenschaften*“ anstelle des Atmel-ICE eine Simulation. Das führt dazu, dass das komplette Programm **nicht** auf den ATmega geladen wird, sondern nur simuliert wird. 
- b) Setzen Sie 2 Breakpoints durch Doppelklick auf den linken Rand neben den folgenden Kommandos:
- Kommando „sei“
 - Interrupt Service Routine: „counter++“.
- Starten Sie anschließend die Simulation durch Drücken des blauen Pfeiles. 

³ Das kann etwas dauern.

- c) Das Programm wird automatisch zu Beginn angehalten. Gehen Sie mit „Continue“ zum ersten Breakpoint bei `sei()`. Prüfen Sie, dass das Bit `ISC01` im `EICRA` Register durch die Instruktion davor auch tatsächlich korrekt gesetzt wurde.
Tipp: Schauen Sie sich unter „Debugging“-„Window“ den Arbeitsspeicher („Memory“) an. Die Adresse des `EICRA`-Registers entnehmen Sie dem Handbuch auf S.399.
- d) Öffnen Sie dann die „I/O View“ unter „Debugging“-„Window“ und erzeugen Sie künstlich eine fallende Flanke, die dann einen Interrupt auslöst:
- Pin `PD0` auf 1 setzen.
 - Einzelschritt.
 - Pin `PD0` auf 0 setzen (=fallende Flanke).
 - Einzelschritt.
 - Was passiert?
- e) Stoppen Sie die Simulation und versuchen Sie die Simulation erneut ohne Debugging zu starten („grüner Pfeil“). Geht das?
- 

Aufgabe 4: Hardware-Debugging mit JTAG

- a) Führen Sie die Verkabelung gemäß der Abbildung (Teil ④) durch. Das ISP-Kabel am besten nur auf der Atmel-ICE Seite abstecken.
- b) Konfigurieren Sie im Menü „Project“-„ISP-JTAG-Eigenschaften“ den Atmel-ICE sowie als Interface JTAG. Zum Test versuchen Sie nun wie in Aufgabe 2f) den Signaturcode auszulesen.
- c) Setzen Sie die gleichen Breakpoints wie in Aufgabe 3b. Starten Sie anschließend das Debugging!
- Welcher Zustand wird für Pin `PD0` im I/O Fenster angezeigt?
 - Halten Sie dann den Taster gedrückt, während Sie einen weiteren Einzelschritt ausführen. Was Welcher Zustand wird nun für Pin `PD0` angezeigt? Was passiert?
- d) Durch Debugging und ISP Programmieren wurde der Arduino-Bootloader gelöscht. Im „Device Programming“-Dialog bitte unter „Memories“ den mitgelieferten Arduino Bootloader `stk500boot_v2_mega2560.hex` wieder ins Flash laden.
- 
- 