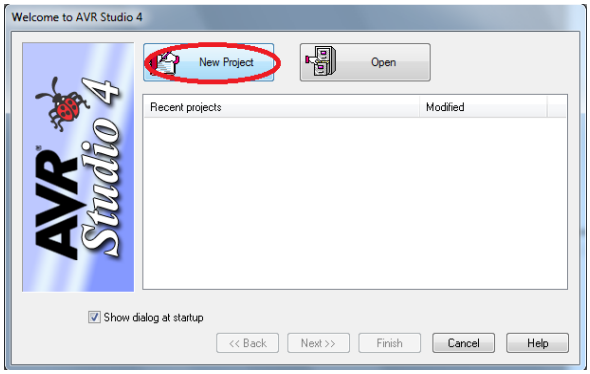
AVR Studio

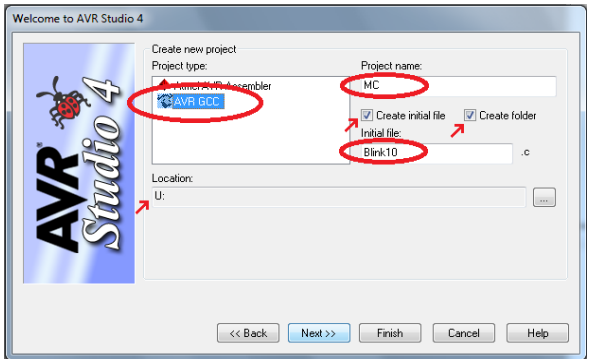
In diesem Praktikum wird mit dem AVR Baustein ATtiny2313 von Atmel gearbeitet. Die AVR Bausteine gliedern sich in mehrere Familien. Die ATtinys sind dabei die kleinsten Mikrocontroller, als nächstes kommen die weit verbreiteten ATmega- Controller und als letztes die XMEGA- Bausteine.

Das AVR Studio ist die Oberfläche um Mikrocontroller des Herstellers Atmel zu programmieren. In dieser Oberfläche werden Editor, Compiler (Assembler oder GCC) , Simulator und Programmer zusammengefasst.

Erster Start des AVR Studios



Da das AVR Studio seine einzelnen Komponenten in Form eines Projektes verwaltet, wählen Sie beim ersten Start 'New Project' aus.



Bei 'Project type' klicken Sie AVR GCC an, um den C- Compiler auszuwählen.

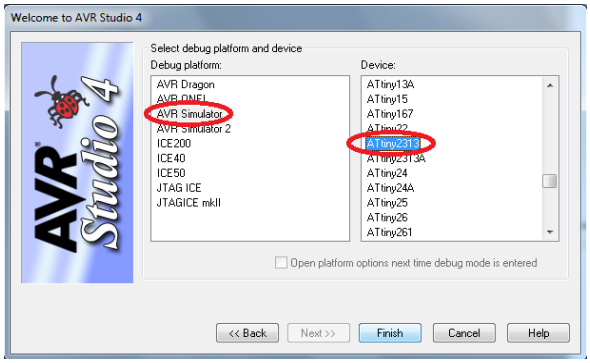
Unter 'Location' wählen Sie 'IHR_LOGIN' (\\caelabsrv\homes\students)(U:). In diesem Pfad werden alle vom AVR- Studio erzeugten Dateien gespeichert.

Geben Sie für 'Project name:' und 'Initial file:' entsprechend sinnvolle Namen ein.

Als Vorschlag:
Projektname = MC, für Mikrocontroller
Initialdatei = Blink10

Achten Sie darauf das die beiden Check boxen 'Create initial file' und 'Create folder' markiert sind. Der Projektname wird dann das Arbeitsverzeichnis des AVR- Studios und der Name der Projektdatei. Der bei 'Initial file' angegebene Name ist die Quelldatei (Textdatei) in der das eigentliche Programm steht.

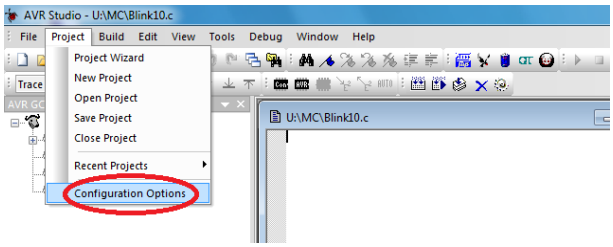
Wenn Sie die Einstellungen gemacht haben, klicken Sie auf 'next>>' um fortzufahren.



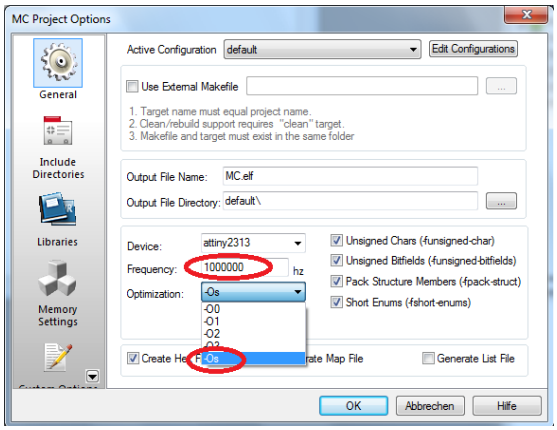
Hier wählen Sie unter 'Debug platform' den AVR Simulator und als 'Device' den Controller ATtiny2313, der in diesem Praktikum genutzt wird.

Damit sind die ersten Einstellungen abgeschlossen und Sie Klicken 'Finish' um das Projekt zu öffnen.

Mikrocontroller 2



Unter 'Project' > 'Configuration Options' müssen noch zwei Einstellungen vorgenommen werden.



Die eine Eingabe ist die Geschwindigkeit mit welcher der Controller arbeitet. Sie ist vom Hersteller auf 1MHz eingestellt, daher muss bei 'Frequency:' 1000000 eingetragen werden. Man kann bei den Controllern der AVR- Reihe über die Fusebits andere Taktquellen und Arbeitsgeschwindigkeiten einstellen, daher ist dieser Eintrag beim ersten Start offen.

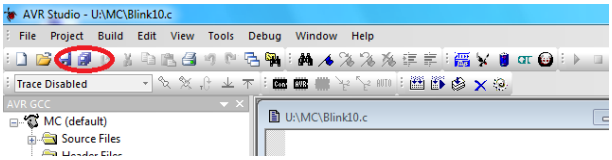
Zweitens wird dem Compiler mitgeteilt wie der Maschinencode komprimiert werden soll. Die Standardeinstellung ist hier '-O0', also keine Komprimierung. Bitte wählen Sie hier '-Os' (small) die Komprimierung für einen möglichst kleinen Code, da unser Baustein nur wenig Programmspeicher zur Verfügung stellt.

Beenden Sie diese Eingabe mit 'OK'.

Nun können Sie Ihr Programm schreiben.

Speichern eines Projekts

Speichern Sie das Projekt, und damit auch die Quelldatei, mit 'Project > Save Project' oder mit dem 'Save All' Button.



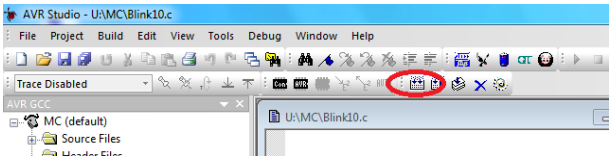
! Achtung !

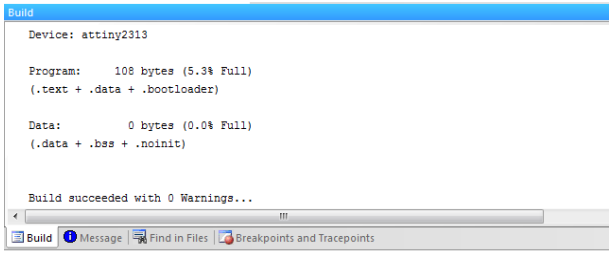
Benutzen Sie nicht den 'Save Button' links daneben oder 'File > Save'. Denn dann wird nur die Quelldatei, aber nicht das Projekt mit den eben gemachten Einstellungen, gespeichert.

Mikrocontroller 2

Compilieren

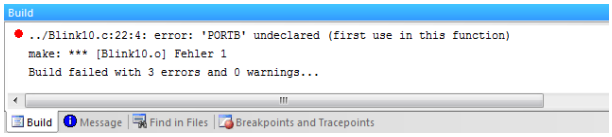
Wenn Sie ihr Programm fertig geschrieben haben können Sie es mit dem 'Build Acvite Configuration' Button übersetzen. Dieser Vorgang speichert die Quelldatei übrigens auch schon mal.



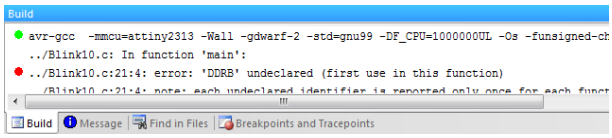


Hier sehen Sie ob das Programm compiliert worden ist, oder welche Fehler und Warnungen aufgetreten sind.
Die Meldung 'Build succeeded with 0 Warnings...' zeigt an, dass das Programm ohne Fehler übersetzt wurde.

Wenn Fehler angezeigt werden, lesen Sie die Meldungen und berichtigen Sie das Programm entsprechend. Fangen Sie dabei immer mit dem ersten, am weitesten oben stehendem, Fehler an, da häufig Folgefehler auftreten, die sowieso verschwinden wenn der erste Fehler behoben ist.



Es sind 3 Fehler aufgetreten. Es wird aber nur der letzte Fehler angezeigt. Diesen Fehler bearbeiten ist nicht sinnvoll.

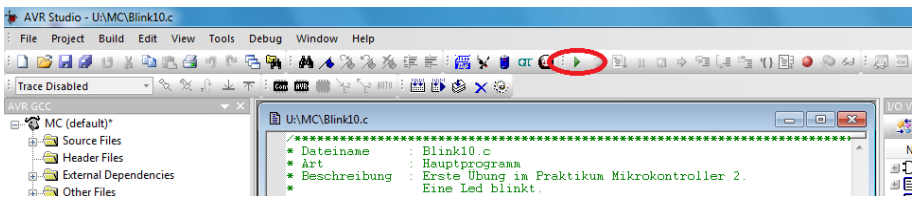


Nach dem herauf scrollen in diesem Fenster sieht man mehr. Die erste Zeile (grüner Punkt) ist der Start des Compilers. Danach folgen die Rückmeldungen vom Übersetzen. Der erste rote Punkt markiert also den ersten Fehler. Hier muss mit der Fehlerkorrektur begonnen werden.

Ist das Programm ohne Fehler und Warnungen übersetzt worden können Sie es an den Mikrocontroller übertragen oder den Programmablauf im Simulator testen.

Debuggen

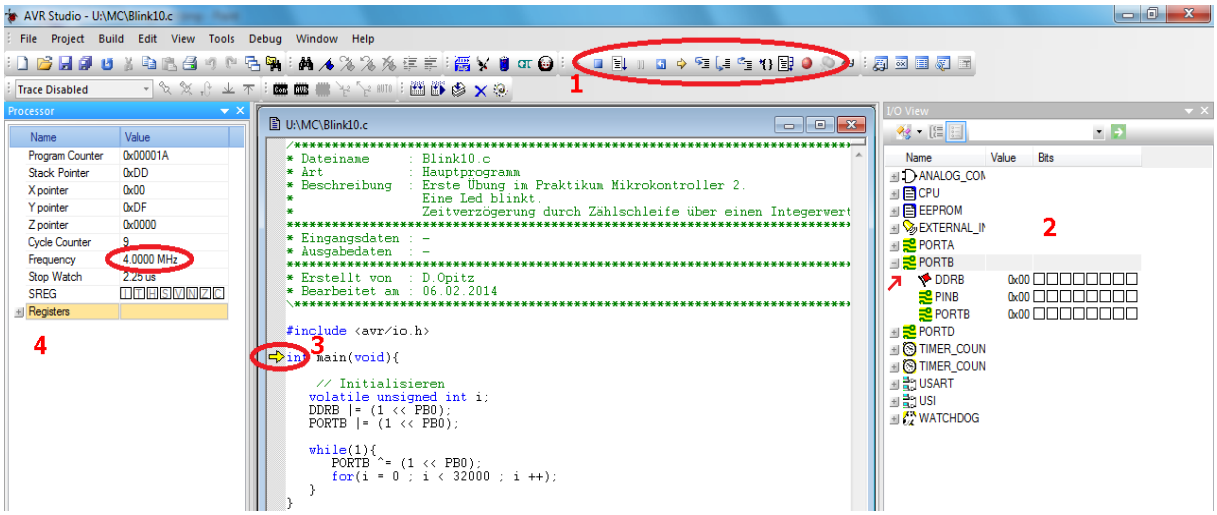
Debuggen heißt in unserem Fall, das übersetzte Programm wird im Simulator ausgeführt. Dabei kann das Programm schrittweise (Zeile für Zeile), bis zu einem Breakpoint oder komplett ablaufen. Hier können natürlich die Speicherinhalte, Register und Portpinzustände angezeigt werden. Damit kann der korrekte Ablauf des Programms kontrolliert werden.



Mit dem 'Start Debugging' Button wird der Simulator gestartet.

Jetzt, bei laufendem Simulator, verändern sich die Anzeigen ein wenig.

Mikrocontroller 2

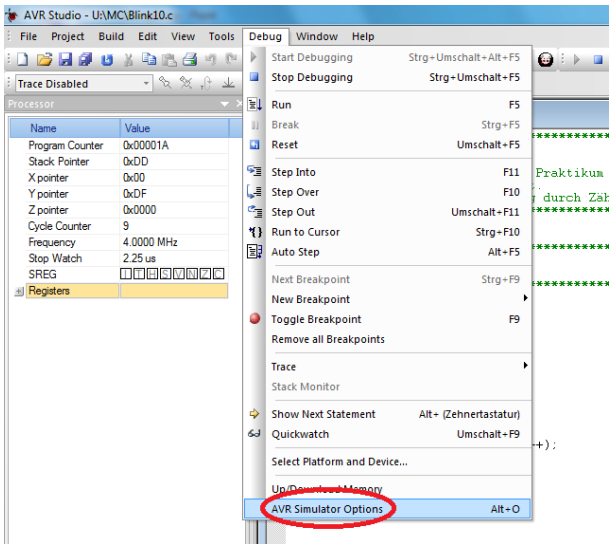


(1) Die Buttons rechts des 'Start Debugging' Pfeils werden freigeschaltet. Hier finden sich Funktionen um den Simulator zu steuern. Wichtig sind der erste Button (Quadrat - 'Stopp Debugging'), zum verlassen des Debuggers. Der nächste rechts daneben ('Run'), um das Programm bis zum nächsten Breakpoint laufen zu lassen. Und der Button etwa in der Mitte ('Step Into'), um zeilenweise voran zu kommen.

(2) Im I/O View rechts im Bild, werden die Spezial Function Register und ihre Inhalte angezeigt. Diese Register steuern die Funktionen des Tiny3213, wie Portpins, Timer, Schnittstellen und so weiter.
In dem ersten Programm dieses Praktikums soll ein Portpin regelmäßig von high auf low auf high wechseln. Der Portpin ist Bit 0 an Port B, darum habe ich in diesem Bild den Port B, mit einem Klick auf das Plus davor, aufgeklappt um die Werte zu sehen.

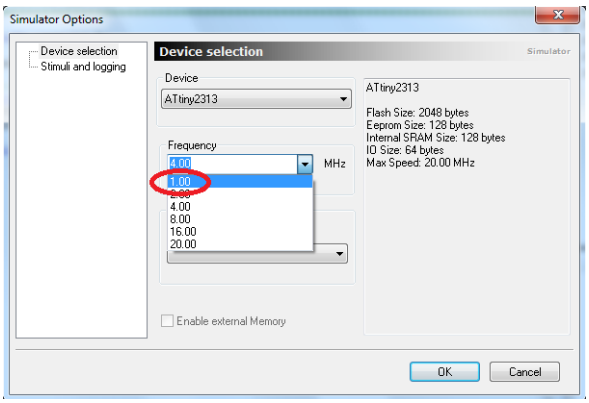
(3) Im Editorfenster, in der Mitte, zeigt der gelbe Pfeil auf die Zeile, die gerade vom Simulator bearbeitet wird. Mit Step Into schaltet man ihn eine ausführbare Zeile weiter und lässt den Simulator damit die nächste Anweisung bearbeiten. Dabei werden dann auch die Werte der veränderten Register aktualisiert.

(4) Im linken Fenster werden die internen Register des Prozessors und einige Systemeinstellungen angezeigt. Hier wird auch eine Frequency angegeben, die nicht mit der in der Konfiguration eingestellten synchronisiert wird. Damit die Stoppuhr darunter die richtigen Zeiten anzeigt, muss hier natürlich auch 1 MHz stehen.



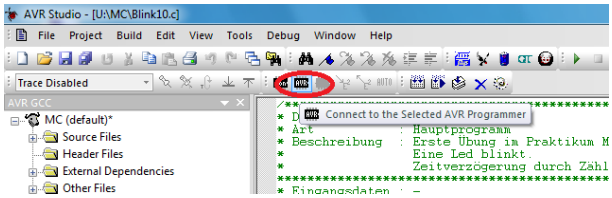
Der Wert der Frequenz lässt sich unter 'Debug > AVR Simulator Options' einstellen.

In der Auswahl klicken Sie 1.00 und bestätigen dann mit 'OK'.

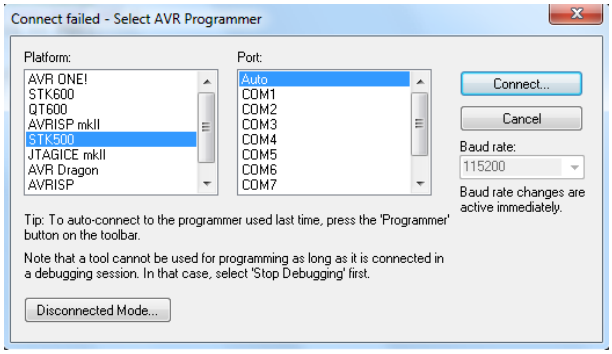


Programm übertragen

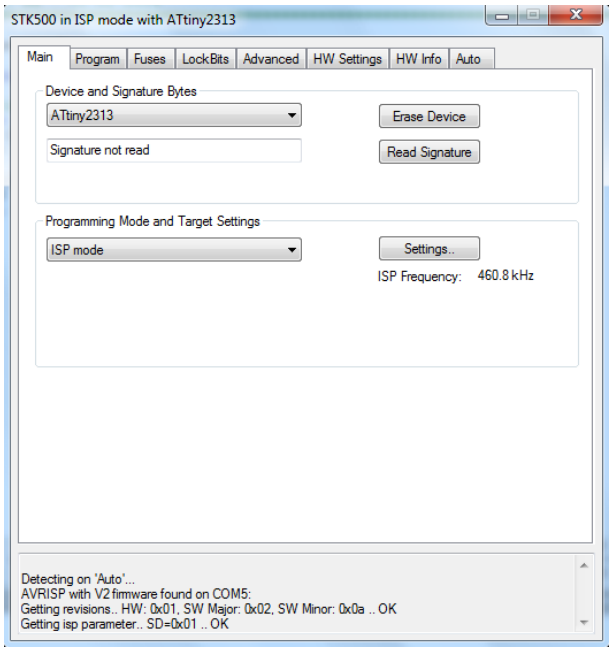
Um das Programm in den Mikrocontroller übertragen zu können muss die Hardware natürlich fertig angeschlossen sein. Denken Sie dabei an den Programmer und die Stromversorgung des Testaufbaus. Außerdem muss das Programm fehlerfrei compiliert worden sein und sollte auch schon einen Testlauf im Simulator absolviert haben.



Wenn Sie das AVR- Studio frisch gestartet haben, müssen Sie erst den Programmer mit dem Rechner verbinden. Klicken Sie dazu auf den Button 'Connect to the Selected AVR Programmer'.



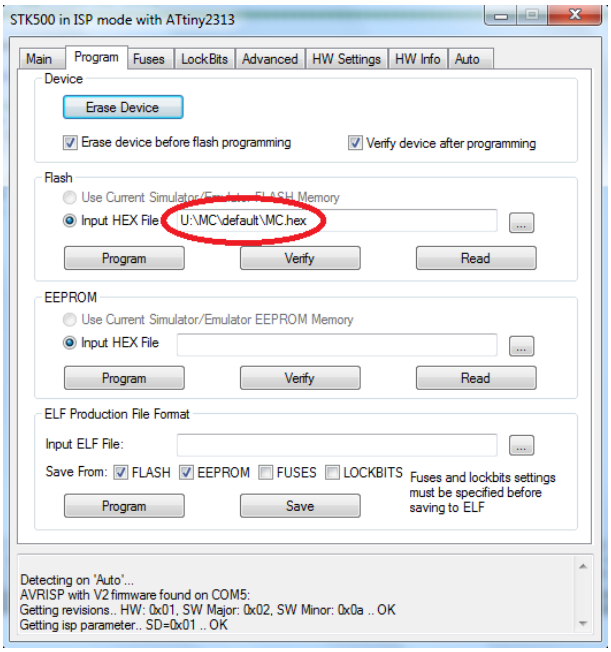
Sehen Sie jetzt dieses Fenster ist der Verbindungsaufbau fehlgeschlagen. Kontrollieren Sie die Leitungen (USB, Programmer und Stromversorgung), schließen Sie das Fenster und versuchen Sie es nochmal.



Normalerweise wird dieses Fenster geöffnet, wobei der zuletzt benutzte Reiter aktiv ist. Hier ist gleich der erste Reiter (Main) gewählt.

In diesem Menü sehen Sie den Typ des Controllers, der programmiert wird und ob die Kennung (Signature) gelesen wurde. Prüfen Sie die Kennung mit 'Read Signature'. Die Kennung des Bausteins wird nun ausgelesen und dann mit der eingestellten verglichen. Jetzt sollte in der Zeile unter Typ die Meldung 'Signature matches selected Device' stehen. Damit ist sichergestellt das der Programmieradapter arbeitet, die Anschlüsse richtig sind und der ausgewählte Baustein in der Schaltung steckt.

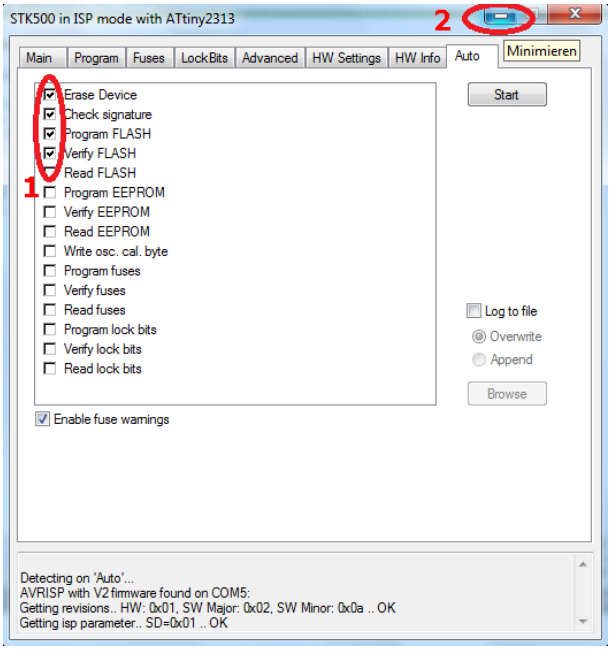
Mit dem Button 'Read Signature' lösen Sie außerdem noch einen Reset des Controllers aus. Dadurch wird auch das aktuell im Baustein stehende Programm neu gestartet.



Im gleichen Fenster unter dem Reiter Program müssen Sie einstellen welche Datei das zu brennende Programm enthält. Der Pfad dieser Datei setzt sich aus dem Pfad des Projekts plus default zusammen und der Dateiname ist der Projektname mit der Endung Hex.

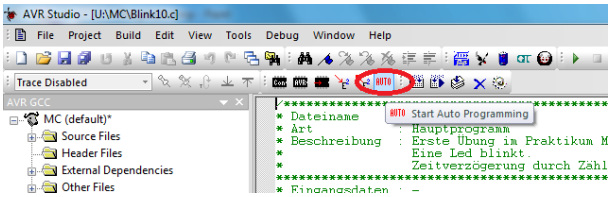
Mit dem Button 'Program' kann das Programm gleich zum Mikrocontroller übertragen werden, aber es gibt noch eine elegantere Methode.

Der Button 'Verify' überprüft ob Bausteininhalt und Programmdatei gleich sind und 'Read' ließt den Mikrocontroller in eine Datei aus.



Unter dem Reiter Auto wird eingestellt welche Funktionen ablaufen wenn 'Auto Programming' gestartet wird. (1) Sie markieren die ersten vier Check Boxen in diesem Menü. Damit wird der Controller beim starten von 'Auto Programming'
- gelöscht
- seine Kennung geprüft
- das Programm übertragen und
- getestet ob das Programm angekommen ist.

Diese Abfolge kann mit dem Button 'Start' oder direkt aus dem AVR Studio aufgerufen werden. Dazu müssen Sie dieses Fenster minimieren (2), aber nicht schließen.

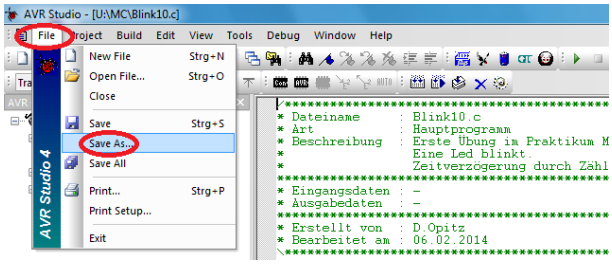


Mit dem Button 'AUTO' kann ab jetzt das Programm übertragen werden, denn dieser Button startet das 'Auto Programming'.

! Achtung !
Die in dem Fenster zum Verbinden des Programmers gemachten Einstellungen werden nicht im Projekt gespeichert. Wenn Sie ein anderes Projekt, mit einem anderen Mikrocontroller, öffnen, hat das keinen Einfluss auf den hier eingestellten Typ. Auch ein Neustart des AVR Studios ändert daran nichts. Sie müssen den Programmieradapter neu einrichten.

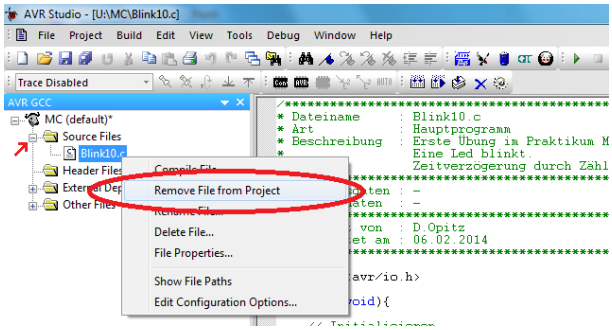
Neues Programm

Um ein neues Programm zu schreiben, ist es nicht notwendig, und häufig auch nicht sinnvoll, ein neues Projekt zu erzeugen. Sind die meisten Projekteinstellungen für das neue Programm gleich, reicht es einfach die Quelldatei auszutauschen. Erst wenn ein anderer Controllertyp oder eine neue Arbeitsfrequenz zu Einsatz kommen sollte ein neues Projekt gestartet werden.



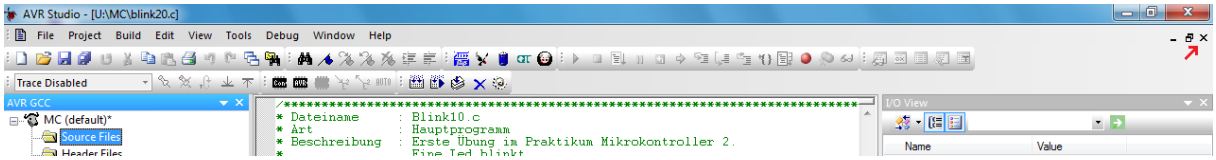
Oft sind in dem neuen Programm auch große Teile des alten Programms enthalten. Sie speichern daher das alte Programm unter dem Namen des Neuen, und erzeugen so eine Kopie des alten Programms mit dem neuen Namen.

Klicken Sie auf 'File > Save As' und geben Sie für die zweite Aufgabe den Namen Blink20.c an.

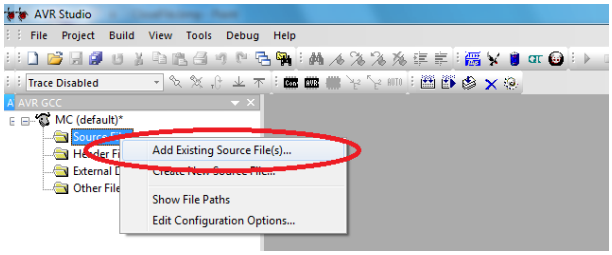


Dann öffnen Sie den Ordner 'Source Files' und klicken mit der rechten Maustaste auf die aktuelle Quelldatei (Blink10.c). In dem Menü das sich jetzt geöffnet hat, klicken Sie 'Remove File from Project' um diese Datei aus dem Projekt zu entfernen. Die Datei wird dabei weder gelöscht noch gespeichert.

Falls die Datei im Editorfenster nicht verschwindet, müssen Sie diese von Hand aus dem Editor entfernen.



Klicken Sie zum leeren des Editors auf das Kreuz ganz rechts im Fenster des AVR- Studios.



Klicken Sie nun mit der rechten Maustaste auf 'Source Files' und dann wählen Sie 'Add Existing Source File(s)...' in dem neuen Menü. Wählen Sie die neue Datei (Blink20.c) aus. Als letztes müssen Sie die neue Datei noch, mit einem Doppelklick auf den Namen, im Editor öffnen.

Damit ist der Tausch der Quelldatei fertig und Sie können mit der neue Datei weiterarbeiten.