Hex Datei mit Atmel Studio 6 erstellen

Es werden generell keine Atmel Studio Dateien ins Repository geladen, da jeder seine Dateien an anderen Orten liegen hat und weil nicht jeder das Atmel Studio 6 benutzt. Hab dafür Verständnis, wenn jeder seine Projektdateien hochlädt, haben wir bald keinen Überblick mehr.

Wir unterstützen das bauen der Hex-Dateien mit Hilfe von GNU Makefiles. Allerdings werde ich hier beschreiben, wie man ein Projekt mit dem Atmel Studio anlegt, damit man sich selber die Hex Dateien erstellen kann, da nicht jeder weiß wie das geht.

Es ist notwendig, das WinAVR installiert ist, siehe hierzu start-avr.pdf.

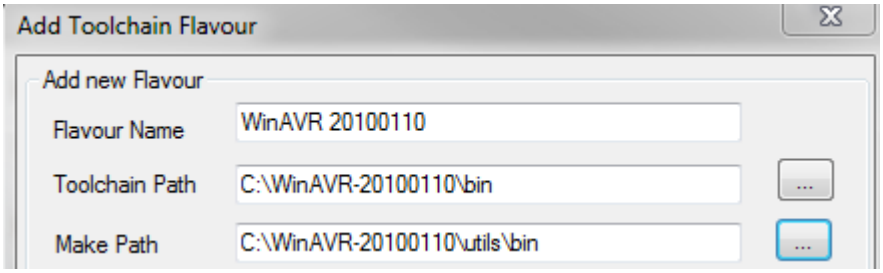
Alles ist mit Hilfe von doxygen dokumentiert, siehe hierfür start-avr.pdf.

Mein freebus_avr Verzeichnis wird mit git nach d:\freebus geklont, alle Pfadangaben beziehen sich hierauf. Falls ihr ein anderes Verzeichnis verwendet, muss es anpasst werden.

AVR Toolchain einrichten

Es muss WinAVR installiert sein, siehe start-avr.pdf. Jetzt richten wir den WinAVR als Standard Toolchain ein. Dafür starten wir AtmelStudio und gehen auf Tools → Options.

Jetzt auf Toolchain → Flavour Configuration. Bei Toolchain Atmel AVR 8-bit auswählen und auf Add Flavour klicken. Jetzt folgendermassen ausfüllen:



Dann WinAVR 20100110 auswählen und auf Set As Default klicken.

Falls es eine neue WinAVR Version geben sollte, kann diese parallel installiert werden. Auf diese Art kann man den Build mit unterschiedlichen Versionen testen.

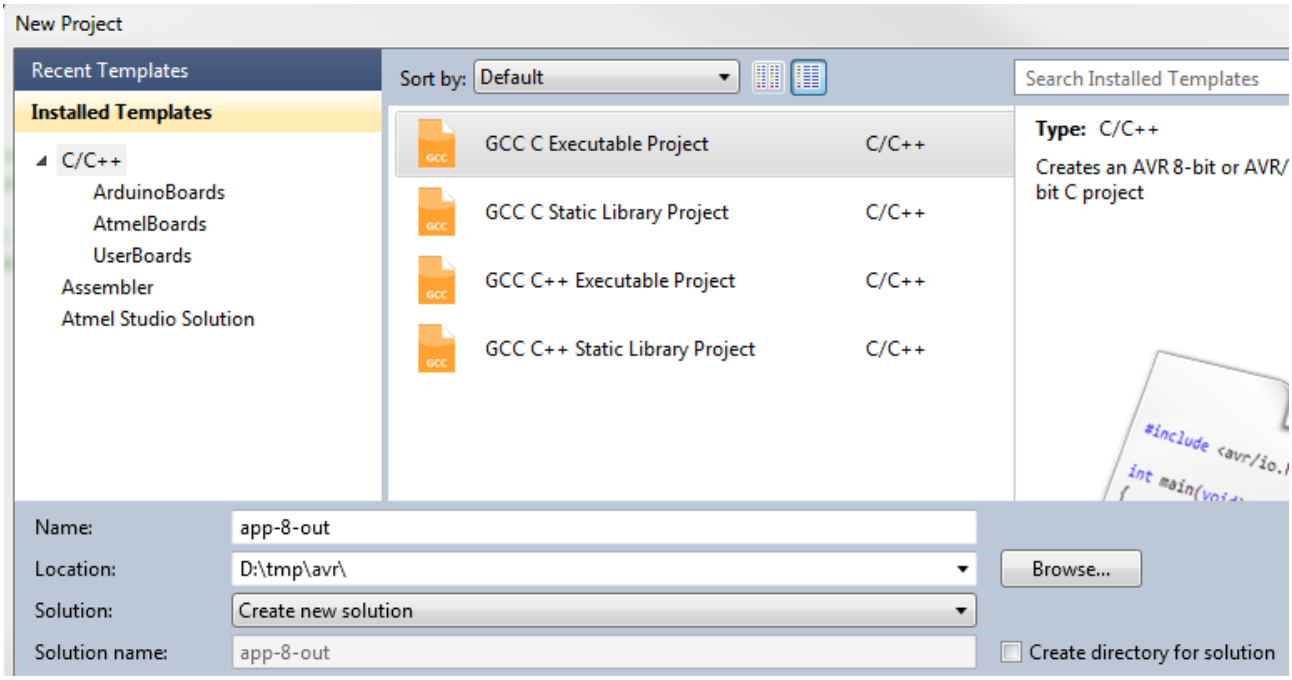
Neues Projekt anlegen

Zuerst muss ein neues Projekt angelegt werden, es muss in einem vorläufigem Verzeichnis (tmp Verzeichnis) erstellt werden, da wir dann manuell eine Datei editieren müssen. Diese verschieben wir dann in das entsprechende Verzeichnis. Ich erkläre es anhand der app-8-out Applikation. Das Verzeichnis für die Applikation ist app-8-out und die Datei heißt fb_relais_app.c und fb_relais_app.h.

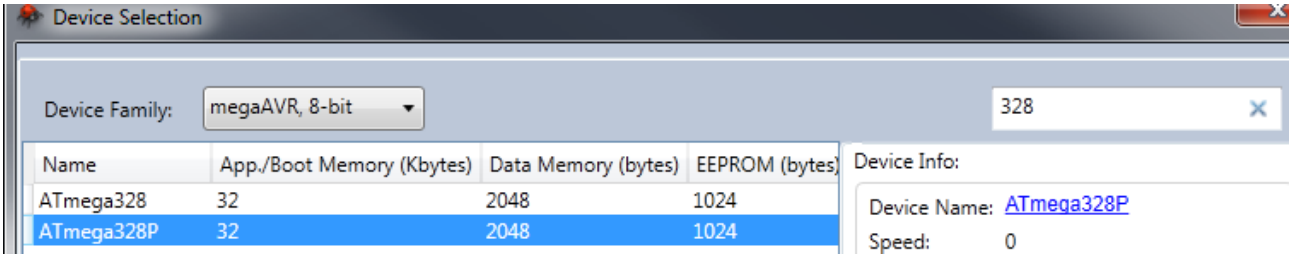
AtmelStudio 6 starten und dann auf:

File → New → Project

Als Name tragen wir das Verzeichnis der Applikation ein, bei uns app-8-out.



Im nächsten Fenster wählen wir unseren Mikrokontroller aus:



Jetzt das Projekt speichern und AtmelStudio beenden.

Nun müssen wir die Datei app-8-out.cproj mit einem Texteditor editieren. Ziemlich weit unten gibt es die Zeile:

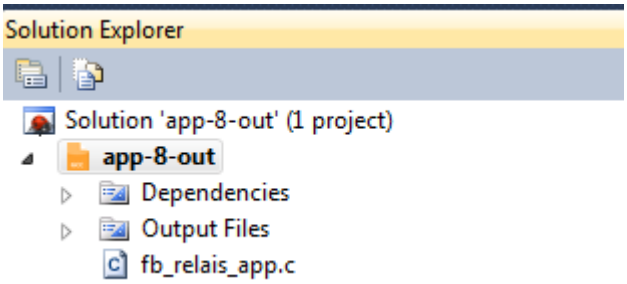
<Compile Include="app-8-out.c">

Diese ändern wir in:

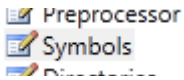
<Compile Include="fb_relais_app.c">

Jetzt verschieben wir die Dateien app-8-out.atsln, app-8-out.atsuo und app-8-out.cproj in der Verzeichnis d:\freebus\freebus_avr\app-8-out. Das tmp Verzeichnis kann nun gelöscht werden.

Jetzt starten wir das AtmelStudio wieder und öffnen das Projekt im Verzeichnis d:\freebus\freebus_avr\app-8-out\app-8-out.cproj. Im Solution Explorer auf der rechten Seite sehen wir nun app-8-out auf welches wir mit der rechten Maustaste klicken und auf Properties gehen.



Jetzt ist es notwendig einige Projekteinstellungen zu machen. Klickt auf Toolchain dann auf den Punkt Symbols:



Jetzt müssen die entsprechenden Defines eingetragen werden. Um hier eine Dokumentation zu erhalten, muss doxygen installiert sein (siehe start-avr.pdf). Ein Eingabeaufforderung starten (Tastenkürzel hierfür Windows Taste + R drücken und cmd eingeben.

Wir gehen noch in das passende Verzeichnis:

d:

cd freebus

cd freebus_avr

doxygen

(Falls er doxygen nicht finden sollte, muss in Windows der Pfad entsprechend erweitert werden oder beim Aufruf von doxygen den vollen Pfad angeben).

Jetzt erscheinen jede Menge Zeilen und die Dokumentation wird unter d:\freebus\freebus_avr\doc\html erstellt. Im Browser die Datei index.html öffnen.

Hier ist alles erklärt, welche Defines es gibt, wie die Fuses zu setzen sind und welche Lib benutzt werden muss. Weiter zu den Einstellungen.

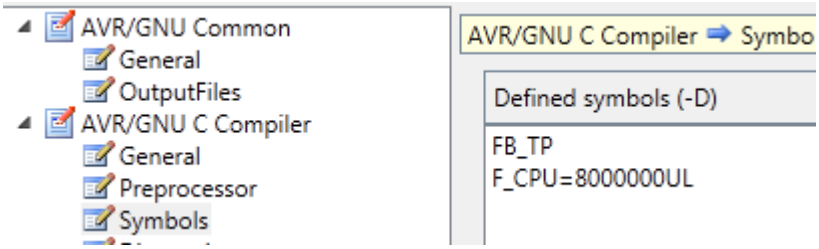
In meinem Beispiel verwende ich das AVR RF Kontroller Board, welches ohne das Funkmodul mit Quarz bestückt ist. Wir sehen in der Doxygen Docu das wir die folgenden Defines definieren müssen:

FB_TP für das AVR Funk Kontrollerboard ohne Funkteil

F_CPU=8000000UL das unsere Kontroller mit 8MHz läuft

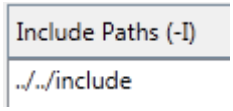
DEBUG_UART, falls wir debugging über RS232 haben wollen, wir lassen das weg, da wir ein Hexfile bauen, das nicht im Labor laufen soll, sondern in einer echten Installation.

Wenn wir damit fertig sind, sieht es so aus:



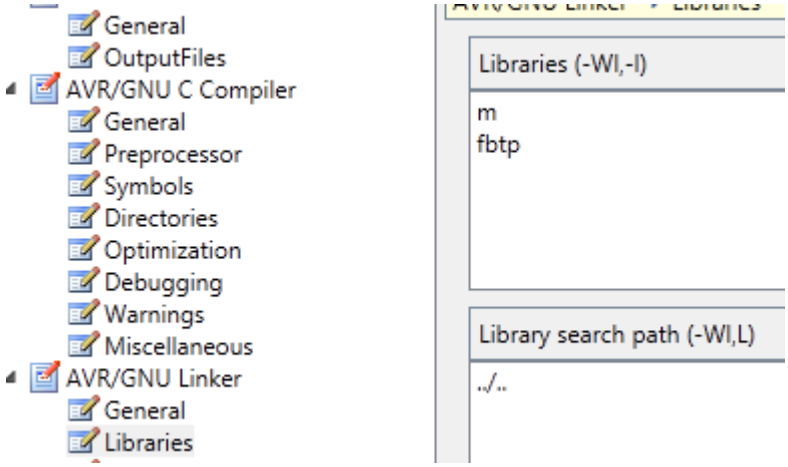
Jetzt tragen wir das Verzeichnis ein, in dem die Include Dateien liegen.

Directories klicken auf das Symbol klicken um ein Verzeichnis hinzuzufügen, nun wählen wir D:\freebus\freebus_avr\include aus klicken auf Relative Path und OK. Jetzt sollte es so aussehen:



Jetzt legen wir noch fest, welche Lib wir brauchen, ein Blick in die Doxygen Dokumentation sagt uns wir brauchen die Datei libfbtp.a. Also Unter Link auf Libraries klicken. Oben bei Libraries auf das Icon zum hinzufügen klicken und fbtb eingeben (ein Lib fängt immer mit lib an, deswegen muss es nicht angegeben werden).

Unter Library Path wählen wir D:\freebus\freebus_avr und klicken wieder Relative Path an. Es sollte so aussehen:



Unter Advanced stellen wir noch sicher das WinAVR 20100110 ausgewählt ist. Jetzt speichern wir das Projekt und können es bauen. Anschliessend sollten unter dem Verzeichnis D:\freebus\freebus_avr\app-8-out\Debug die Datei app-8-out.hex welche die Hex Datei ist, die wir auf den Kontroller flashen können.