Dokumentation für ein Beispielprojekt mit Standard C Programmierung des Grasshopperboards (ATMEL AVR32 AP7000 CPU)

Stand: 06.04.2008

Einleitung

Dieses Dokument soll ein Beispiel C-Projekt für das Grasshopperboard mit der AVR32 Toolchain und AVR32 Studionutzung unter Windows XP beschreiben.
Es soll beschrieben werden, wie mit Standard C eine Anwendung programmiert und dann auf das Board im Flash gespeichert wird. Dazu wird NICHT der Bootloader U-Boot genutzt. Das komplette Flash wird gelöscht, d.h. der Bootloader U-Boot und auch das vorinstallierte Linux gehen verloren. Man kann sie später wieder mit dem JTAG ICE MKII auf das Board programmieren.

Benötigte Hardware

- Grasshopper Board (logo)
- JTAG ICE MKII von ATMEL zum programmieren
- Zum Nutzen der Ausgabe von printf() auf dem UART wird ein Pegelwandler von 3.3V zu RS232 benötigt. Eine UART-Schnittstelle ist bei dem Grasshopperboard auf einen 4-poligen Pfostenverbinder gelegt (zu finden in der Nähe der Quarze neben dem JTAG Pfostenverbinder). Für den Pegelwandler kann man z.B. einen MAX3232 verwenden. ACHTUNG NICHT MAX232!!

Notwendige Software Pakete von ATMEL

Im Moment noch zu zu finden unter: http://www.atmel.no/beta_ware
Ich habe die „stable“ Versionen nicht probiert. Diese Versionen hier liefen bei mir aber ohne Problem.

- AVR32 Studio 2.0 second release candidate
- GNU Toolchain 2.0 second release candidate

RTFM -> Empfohlene Docs:

Leider habe ich keine brauchbare Dokumentation über die Implementierung der Newlib (libc/libm) gefunden. Ich habe teilweise den Toolchain-Source genutzt, zu finden unter http://www.atmel.com/dyn/resources/prod_documents/avr32-gnu-toolchain-1.3.2-0.exe

Es ist scheinbar einiges implementiert aber leider nicht dokumentiert. Hilfreich ist auch das Wiki bei http://www.avrfreaks.com

Die BSP (Board Suppport Packages) zum STK1000 und NGW von ATMEL helfen auch weiter und natürlich die Application Notes (als Dokument oder auch als Beispielprogramm), ebenfalls von ATMEL. Folgende Dokumente sind hilfreich:

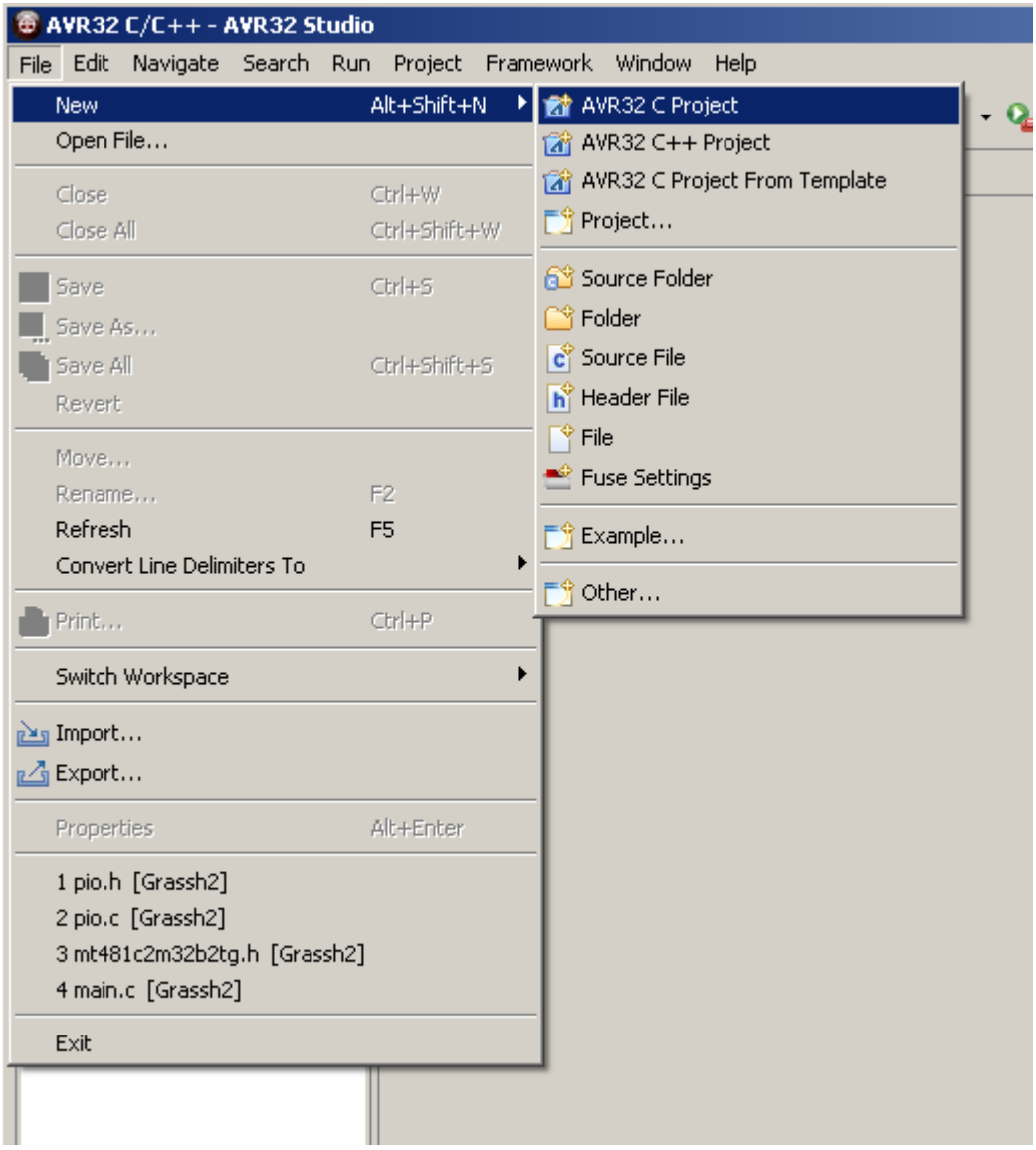
- ATMEL: AVR32015: AVR32 Studio getting started
- ATMEL: AVR32006 : Getting started with GCC for AVR32
- ATMEL: AVR32000: Introduction to AVR32 Headerfiles
- ATMEL: AT32AP7000 AP7000 Manual
- ATMEL: AVR32 Architecture Document
- ATMEL: AVR32 AP Technical Reference Manual

Toolchain Installation

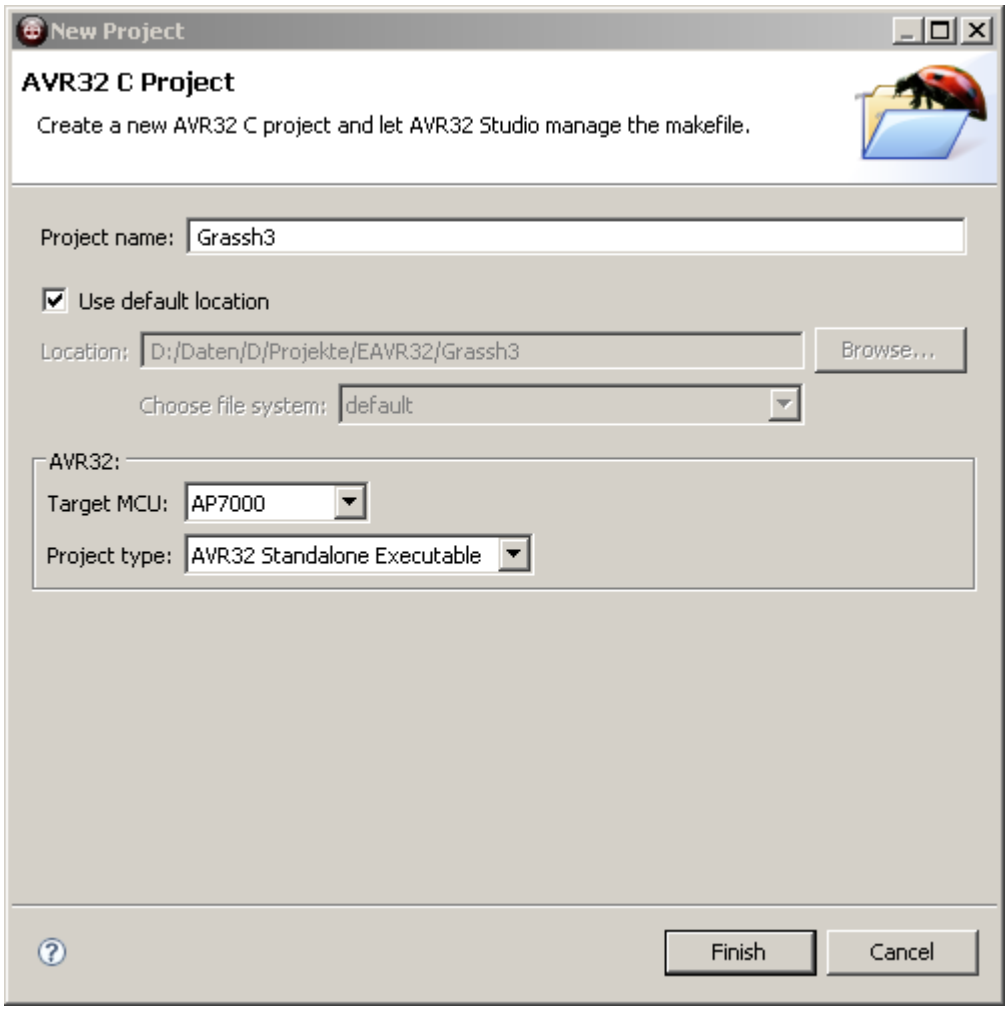
AVR32-Studio und die Toolchain sind beides ausführbare EXE-Dateien, die wie gewohnt starten und installieren. Wie man dann mit dem AVR32-Studio ein Projekt erstellt, kompiliert und mit dem JTAG ICE testet, ist im oben genannten Dokument „AVR32015: AVR32 Studio getting started“ beschrieben. Im Folgenden auch mehr oder weniger genau:

Beispielprojekt erstellen

Im AVR32-Studio mit File-New-AVR32 C Project ein neues Projekt erstellen.



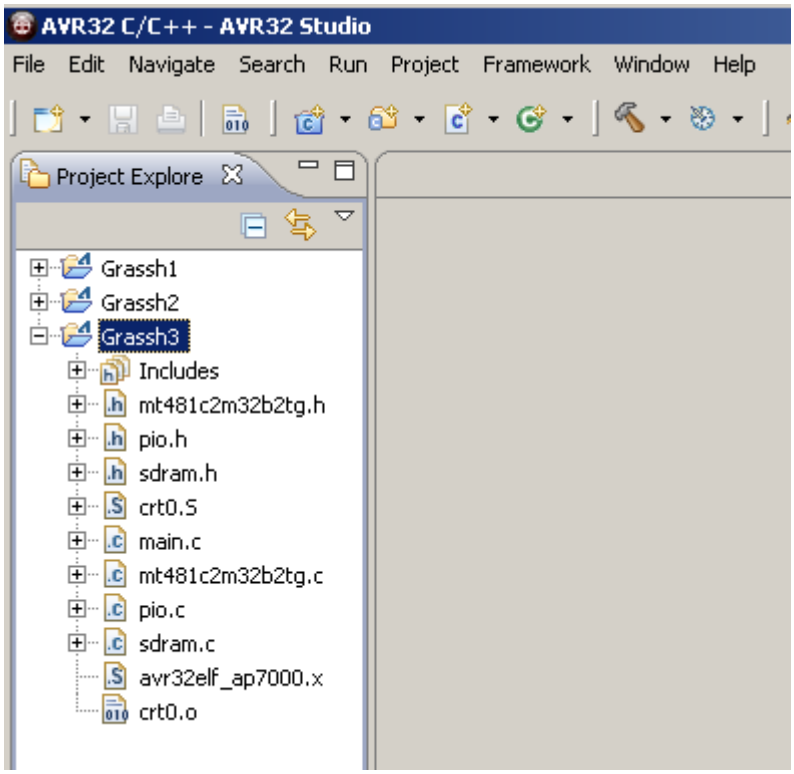
Es erscheint folgendes Fenster:



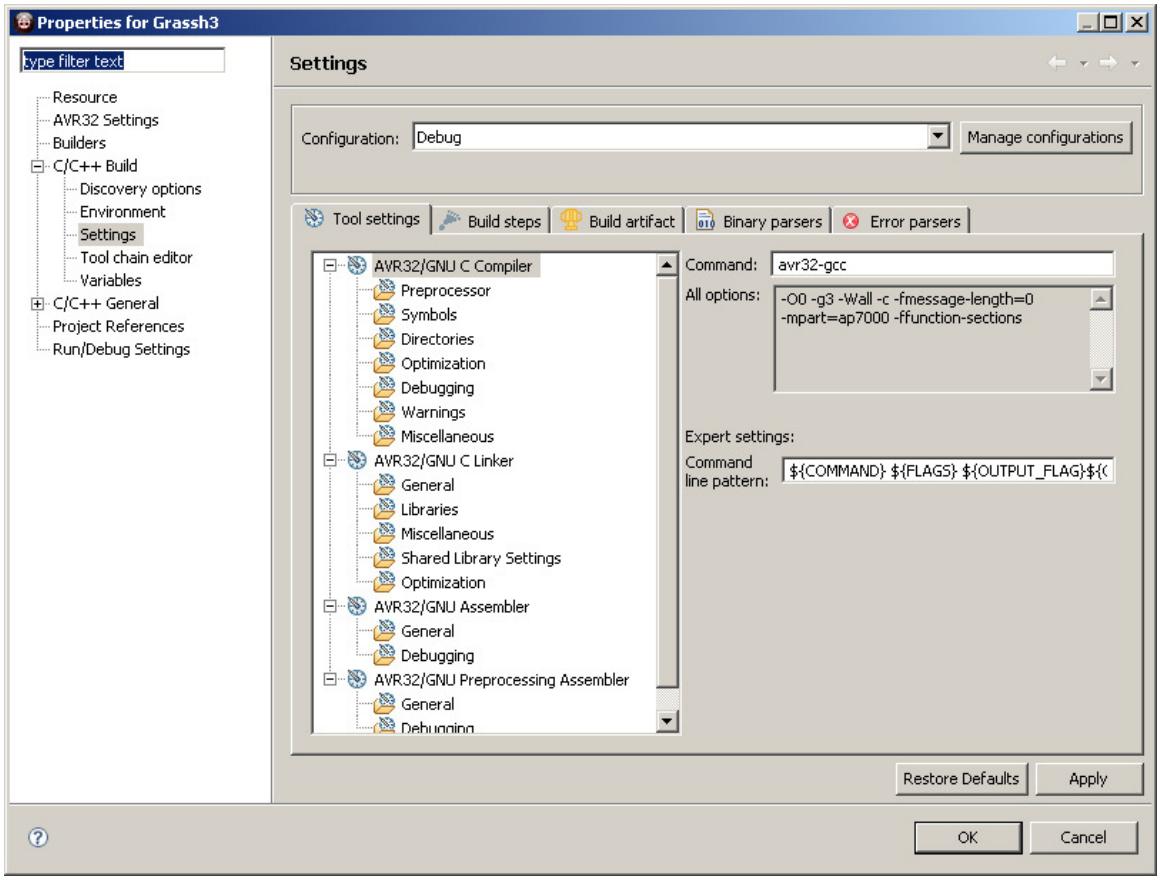
Hier den Projektnamen eintragen, die CPU AP7000 und den Projecttype „AVR32 Standalone Executable“ einstellen.

Jetzt müssen die mitgelieferten Beispiel C-Sourcen importiert werden. Dazu rechter Mausklick auf das Projekt und „Import“ wählen. Dann in dem Fenster die Option „General-FileSystem“ wählen, die Directory auswählen, in der die Dateien liegen und die *.c, *.h *.S *.o Files importieren.

Der Projektbaum im AVR32-Studio sollte dann so ähnlich aussehen (Projekt Grassh3 hier):



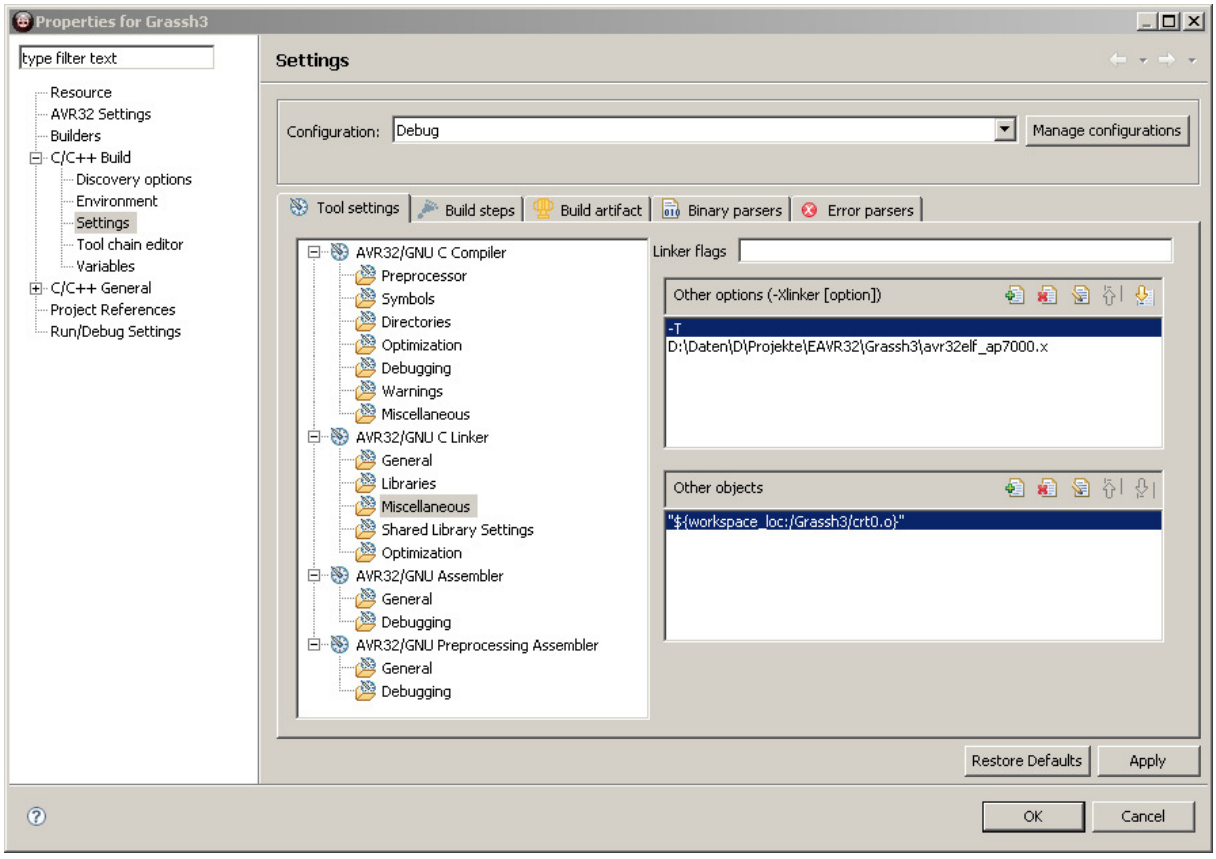
Jetzt müssen noch die Project-Properties eingestellt werden. Das die Einstellungen für den Compiler und den Linker. Dazu das Projekt anklicken (markieren) und im Menu unter Project-Properties die Property-Seite öffnen. Das sieht dann so aus:



Jetzt unter C/C++ Build das Menu „Settings“ öffnen. Das sieht so aus wie die Grafik oben. Hier kann man jetzt die Compiler und Linkereinstellungen vornehmen.

Es müssen in unserem Fall nur Linkereinstellungen vorgenommen werden, der Rest stimmt mit den Default-Einstellungen. Dazu unter: AVR32/GNU C Linker

- General: „Do not use standard startfiles“ markieren, alles andere darf nicht markiert sein.
- Miscellaneous: Other Options (-Xlinker) zwei Einträge erzeugen: 1) nur ein „-T“ und 2) die Datei „avr32elf_ap7000.x“ inkl. Pfad eintragen (siehe Grafik unten).
- Miscellaneous: Other Objects: die Datei crt0.o inkl. Pfad eintragen (siehe Grafik unten).



Jetzt im Projektbaum die Datei „avr32elf_ap7000.x“ mit der rechten Maustste anklicken und Properties öffnen. Hier können für jeden Sourcefile spezielle Einstellungen vorgenommen werden. Für diese Datei muß „Exclude resource from build“ unter C(C++ Build-Settings markiert werden, da das Studio diese Datei sonst kompilieren will (warum das so ist weiß ich nicht).

Beispielprojekt kompilieren

Jetzt kann man das Projekt kompilieren. Dazu unter Projekt „Build Project“ anklicken. Unter den Studio Views (Window-Show Views) „Problems“ bzw. in dem View „Console“ kann man sehen, ob alles erfolgreich war. Bei mir sieht die Consolenausgabe im Studio so aus:

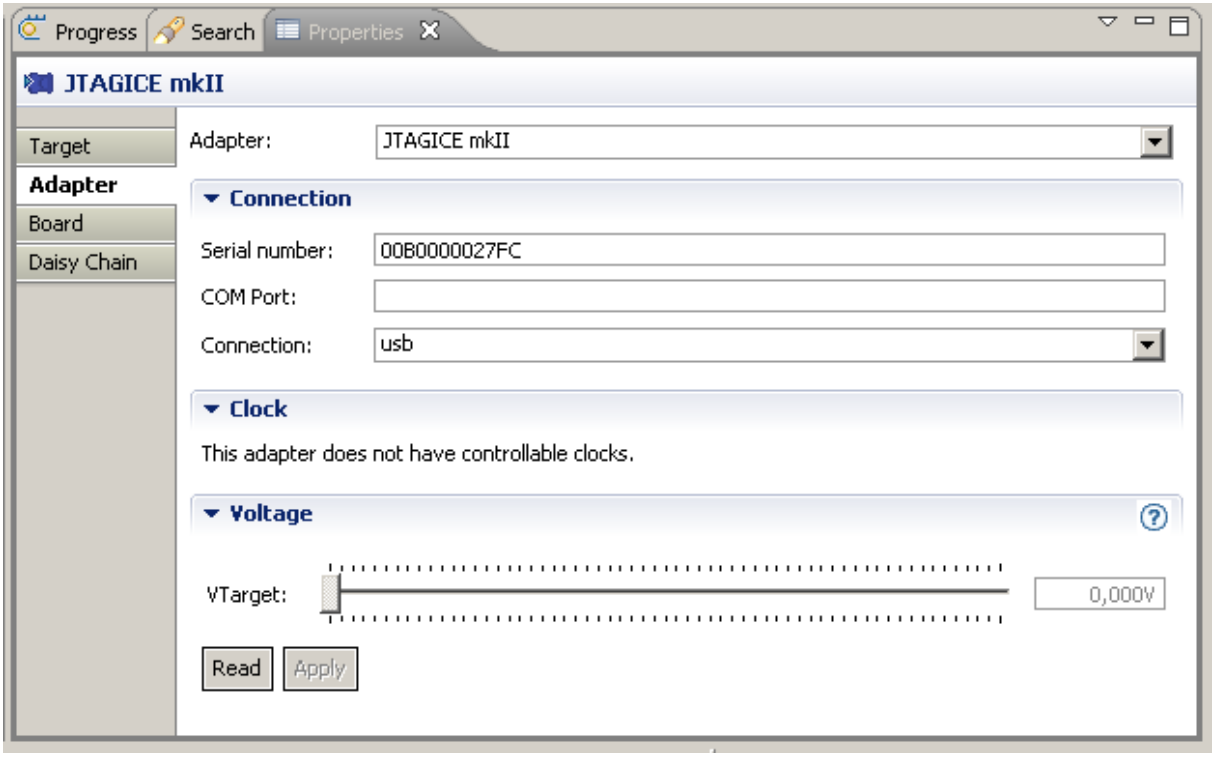
```
**** Rebuild of configuration Debug for project Grassh3 ****

**** Internal Builder is used for build               ****
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=ap7000 -ffunction-
sections -osdram.o ..\sdram.c
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=ap7000 -ffunction-
sections -omain.o ..\main.c
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=ap7000 -ffunction-
sections -omt481c2m32b2tg.o ..\mt481c2m32b2tg.c
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=ap7000 -ffunction-
sections -opio.o ..\pio.c
```

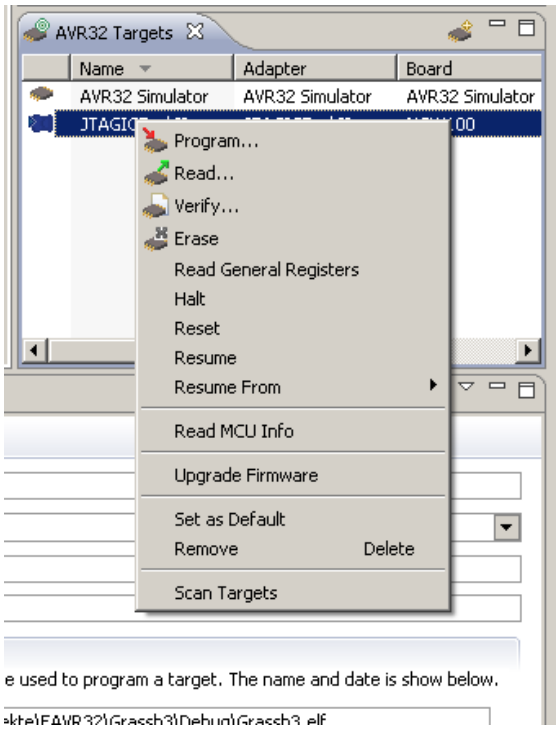
```
avr32-gcc -nostartfiles -Xlinker -T -Xlinker
D:\Daten\D\Projekte\EAVR32\Grassh3\avr32elf_ap7000.x -mpart=ap7000 -Wl,--
gc-sections -oGrassh3.elf sdram.o pio.o mt481c2m32b2tg.o main.o
D:\Daten\D\Projekte\EAVR32\Grassh3\crt0.o
Build complete for project Grassh3
Time consumed: 3219  ms.
```

Beispielprojekt mittels JTAG ICE MKII auf das Grasshopperboard bringen:

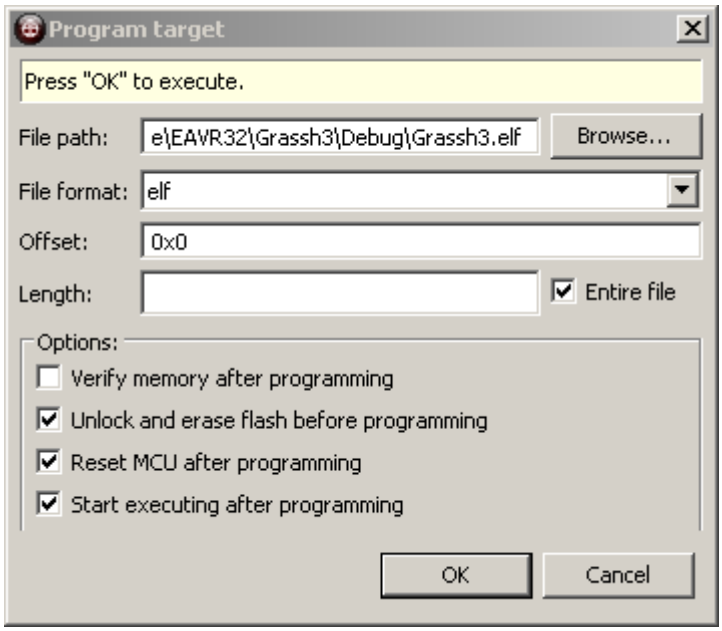
Ich gehe von vorhandenen Hardwareverbindungen aus, also PC, JTAG ICE und Grasshopper sind verbunden und eingeschaltet. Auch die serielle Schnittstelle des Grasshoppers ist mit dem PC über den Pegelwandler verbunden und ein Terminalprogramm (115200 Baud, 8 Datenbit, no parity, 1 Stop bit) ist aktiv. Von richtig installiertem AVR32-Studio inkl. des USB-Treibers für dem ICE gehe ich auch aus.
Im Studio jetzt in der View „AVR32 Targets“ rechter Mausklick und „Scan Targets“ aufrufen wenn der JTAG ICE nicht schon als Target vorhanden ist. Er sollte den ICE dann finden. Jetzt den ICE anklicken. Im View „Properties“ unter Adapter connection =”USB” eintragen. Unter Board das “NGW100” eintragen. MCU = „AP7000“. Da das Studio den Grasshopper nicht kennt (ich weiß auch nicht, wie man es ihm beibringt) wird das NGW100 eingetragen, da es vom Flash und er Adressierung des Flashs mit dem Grasshopper kompatibel ist. Alle anderen Einstellungen im View „Properties“ kann man so lassen.



Jetzt kann man mit rechtem Mausklick im Fenster „AVR32 Targets“ auf den ICE ein Kontext-Menu öffnen.



Mit Klick auf Programm öffnet sich der Dialog zum Programmieren des Grasshoppers.



Dort die Daten wie oben angegeben eintragen. Als Filepath aus dem Workspace das Projekt wählen und dort das Directory „Debug“. Dort sollte der erzeugte ELF-File stehen.
Dann OK klicken und der Grasshopper sollte programmiert werden. Wenn das beendet ist, dann sollten die rote LED und von den grünen die jeweils äußeren gleichzeitig blinken. Auf dem Terminal sollten Ausgaben erscheinen.

Wie man das Programm mit dem Debugger testet, ist hier nicht beschrieben. Dazu die Hilfe des Studios studieren oder in Foren z.B. www.avrfreaks.net lesen.

Viel Spaß.