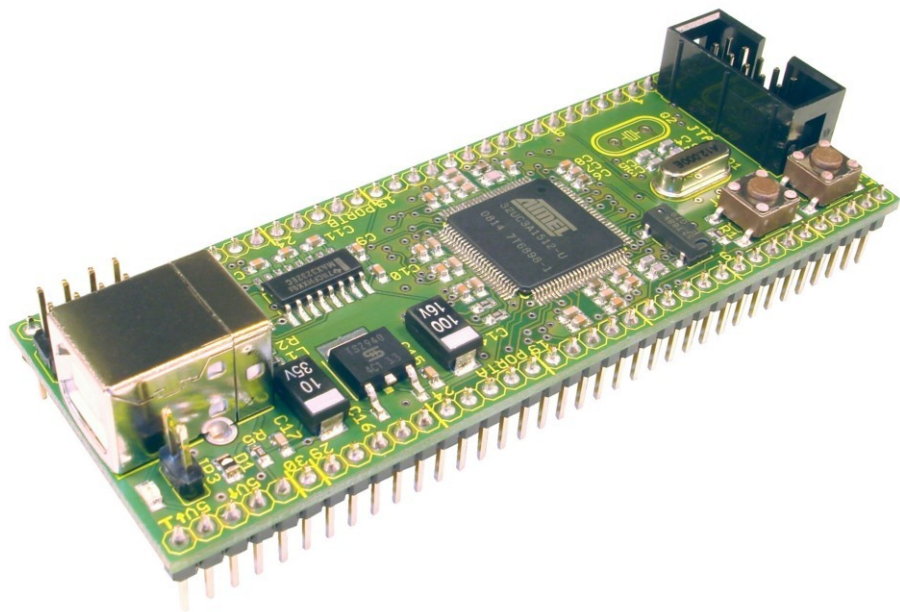


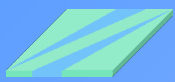
ALVIDI

Erster Schritt

*Vom Herunterladen bis
zum Programmieren*

Revision vom 23.03.2011





Verzeichnis

1.	Einleitung	3
2.	Herunterladen	4
3.	Installation	5
4.	AVR32 Studio	6
4.1.	Neues Projekt	8
4.2.	Erstellen einer neuen Quellcodedatei	8
4.3.	Hinzufügen der Bibliotheken	10
4.4.	Compilieren des Projekts	12
5.	Hardware	13
5.1.	Starten des Bootloaders	13
5.2.	Installieren des USB-Treibers	14
6.	Programmieren	15
6.1.	Erweiterung des Programms	15
6.2.	Installation des Programmers	18
6.3.	Programmieren mit AVR32 Studio	19
7.	Quellen	21

1. Einleitung

Die ähnliche Information, die Sie hier sehen, finden Sie auch auf der Webseite von Atmel. Wir stellen Ihnen mit Hilfe unseres Produktes, AVR32-Modul, unsere Erfahrungen mit AVR32 in diesem Dokument vor.

Dieses Dokument ermöglicht Ihnen einen schnellen Einstieg in die AVR32-Controller-Familie. Es führt Sie vom Herunterladen kostenloser Software und Tools von Atmel bis zum Programmieren des Controllers.

Es ist bekannt, dass der erste Schritt sehr schwer ist. Wir haben diesen Schritt gemacht und präsentieren ihm in diesem Dokument.

Um sich die Bekanntschaft mit AVR32 zu machen, benötigen Sie folgendes:

- Software
 - AVR32 Studio
 - AVR32 GNU Toolchain
 - AVR32 UC3 Software Framework
 - FLIP 3.3.1 oder höher

- Hardware
 - AVR32-Modul AL-UC3AEB mit AT32UC3A1512 Controller **oder**
 - AVR32-Modul AL-UC3BMB mit AT32UC3B0256 Controller **oder**
 - AVR32-Board AL-UC3AVRBIT mit AT32UC3A0512 Controller

- Programmer
 - AVR JTAGICE MKII **oder**
 - USB-Bootloader (auf jedem Controller UC3-Serie vorinstalliert)

Zum Programmieren stehen Ihnen zwei Möglichkeiten zur Verfügung. Als Erste kommt AVR JTAGICE MKII. Ein großer Nachteil dieser Programmiermöglichkeit ist ein hoher Kostenaufwand und daher ist sie für wenige interessant. Als Zweite steht der kostenlose USB-Bootloader zur Wahl. Wie der Name schon verrät, benötigt er nur eine USB-Verbindung mit dem Computer.

Jeder Controller der UC3-Serie ist mit dem USB-Bootloader vorprogrammiert, deswegen können alle drei oben genannten Hardware mit Hilfe eines USB-Kabels programmiert werden.

2. Herunterladen

Das komplette Software finden Sie auf der Webseite von Atmel.

- **AVR32 Studio 2.6 for Windows (installer)** (272 MB, revision 2.6, updated 9/10)
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4116
dieses File enthält die Entwicklungsumgebung für AVR32 - Controller. Vorerst müssen Sie sich registrieren. Nachdem Sie alle (*) Felder ausgefüllt haben, erscheint das Paket zum Runterladen.
- **AVR32 GNU Toolchain** (53 MB, revision 2.1.6, updated 3/09)
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4118
dieses File enthält die C-Bibliotheken, flashprogrammier Werkzeuge, Assembler, Linker und Compiler unter Windows und Linux
- **AVR32 UC3 (A oder B) Software Framework**
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4192
dieses File enthält zahlreiche Beispiele, Treiber, Quellcodes, fertige Projekte, HTML-Dokumentation, Softwareservices ...
- **FLIP 3.4.2 for Windows (Java Runtime Environment included)**
http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886
FLIP (**FL**exible **I**n-system **P**rogrammer) unterstützt In-System Programmierung von Flash Geräten durch RS232, USB oder CAN. Hier befindet sich unter anderen der Treiber für USB-Bootloader.

3. Installation

1. Installieren Sie als Erstes **AVR32Studio-2.x.x-Setup.exe**

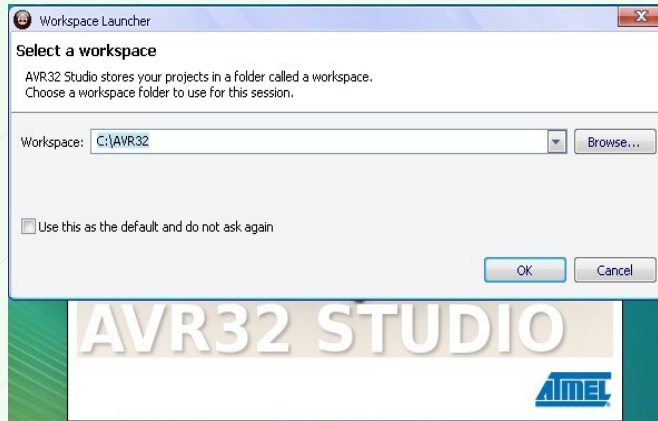


Folgen Sie den Anweisungen im Fenster und installieren Sie AVR32 Studio. Die Installation wird einige Minuten in Anspruch nehmen. Bitte achten Sie darauf, dass Ihre Firewall die komplette Installation zulässt.

2. Installieren Sie als Nächstes **avr32-gnu-toolchain-2.x.x.exe**
3. Entpacken Sie **AVR-SoftwareFramework-2.3.1.zip**, z.B. auf Partition C:\
4. Installieren Sie **JRE - Flip Installer – 3.4.2.exe**

4. AVR32 Studio

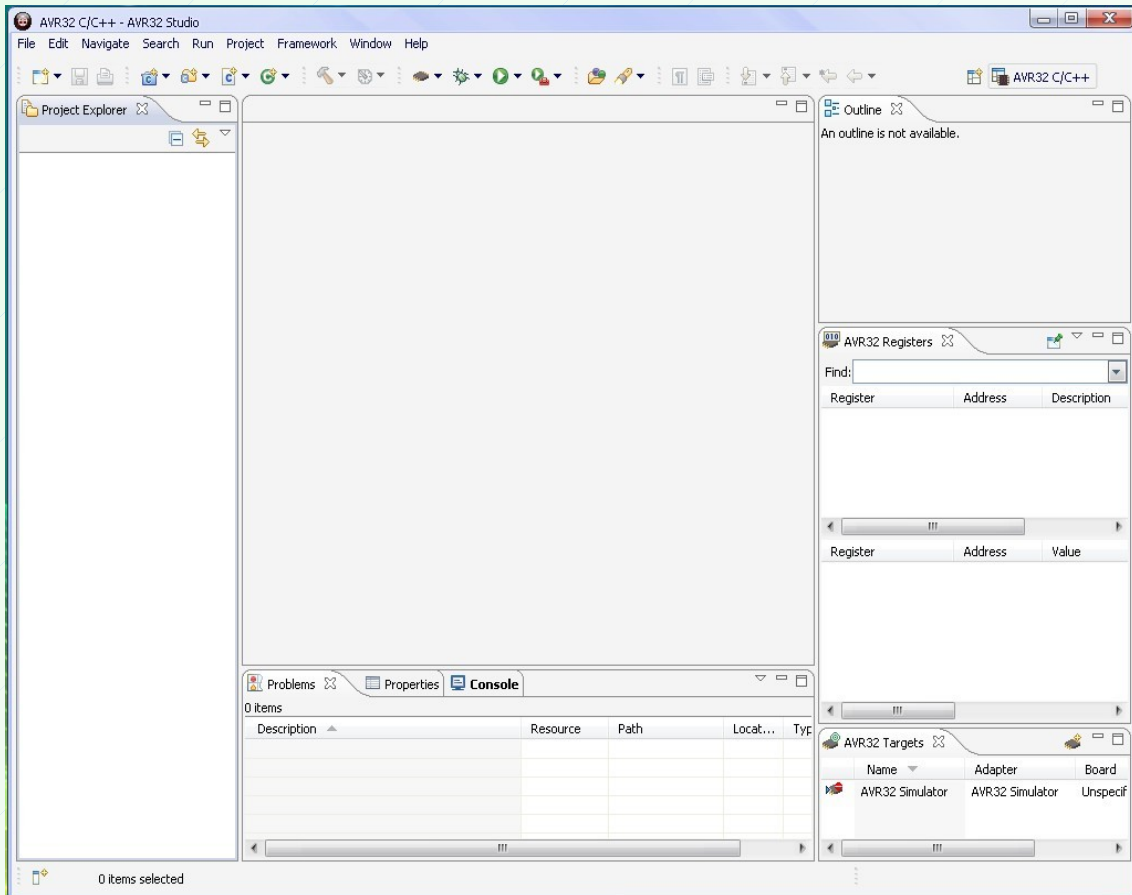
Starten Sie AVR32 Studio *Start* → *Programmieren* → *Atmel AVR Tools* → *AVR32Studio*.



Legen Sie in diesem Fenster fest, in welchem Ordner soll Ihr Projekt gespeichert werden und klicken Sie auf die Schaltfläche „OK“.

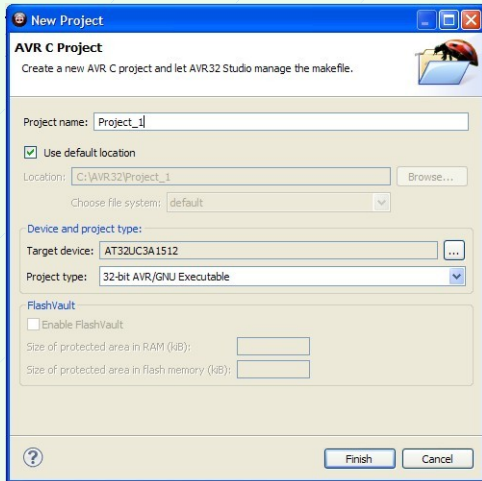


Sie begrüßt jetzt das Fenster „Welcome“. Hier können Sie mehr Informationen über die Software, Hardware und Programmer bekommen. Schließen Sie das Fenster indem Sie hinter dem Wort „Welcome“ weißes Kreuz anklicken.



Danach erscheint die obere Abbildung einer Entwicklungsumgebung. Wie man sieht, hat diese Umgebung kaum Ähnlichkeit mit AVR Studio 4. In den nächsten Kapiteln werden wir die AVR32 Studio mehr unter die Lupe nehmen.

4.1. Neues Projekt

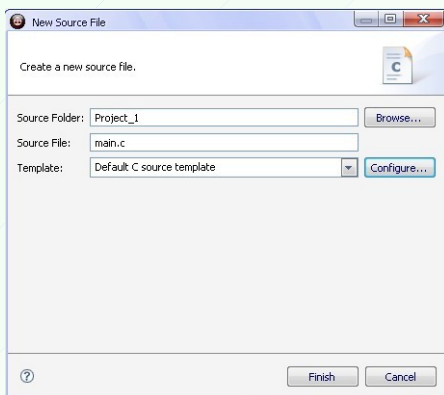
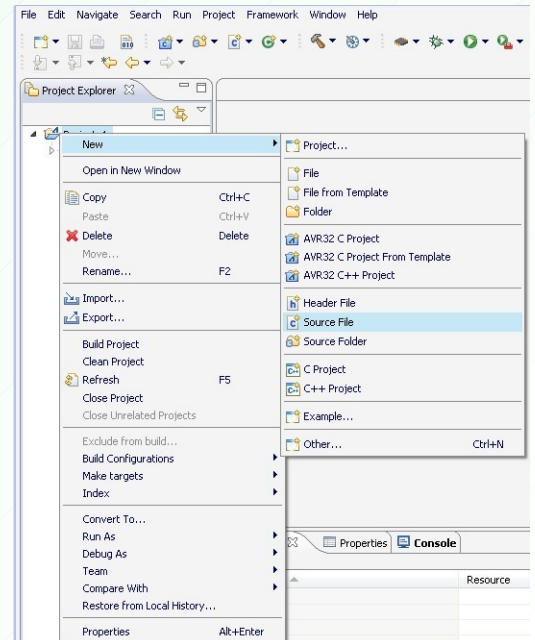


Starten Sie ein neues Projekt *File* → *New* → *AVR32 C Project*.

Geben Sie einen Namen dem Projekt im Feld *Project name*, z.B. *Project_1*. Wählen Sie den Controller Ihrer Hardware im Feld *Target device*, z.B. für AVR32-Modul UC3A1512, und im Feld *Project type* **32-bit AVR/GNU Executable**. Klicken Sie anschließend auf die Schaltfläche „Finish“.

4.2. Erstellen einer neuen Quellcodedatei

Erstellen Sie einen neuen Quellcode *File* → *New* → *Source File* oder klicken Sie mit rechter Maustaste auf Ihren Projekt, z.B. *Project_1*, und wählen Sie *New* → *Source File*



Geben Sie einen Namen im Feld *Source File* ein, z.B. *main.c* und klicken Sie auf die Schaltfläche „Finish“.

Schreiben wir ein kleines Programm. In diesem Programm verändern wir zeitweise den Ausgangszustand des Pins 3 vom Port A.

```
#include "gpio.h" //driver of atmel include in AVR32 UC3A Framework C:\AT32UC3x-1.x.x\DRIVERS\GPIO
#include "compiler.h" //driver of atmel include in AVR32 UC3A Framework C:\AT32UC3x-1.x.x\UTILS

int main(void)
{
    U32 i; //you will find this definition of >U32< in driver "compiler.h"

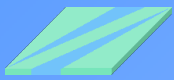
    while(1)
    {
        gpio_set_gpio_pin(AVR32_PIN_PA03); //set the pin 3 on port A as high-output

        for(i=0; i<1000; i++); //wait loop

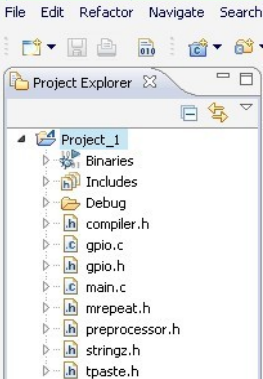
        gpio_clr_gpio_pin(AVR32_PIN_PA03); //set the pin 3 on port A as low-output

        for(i=0; i<1000; i++); //wait loop
    }
}
```

Übernehmen Sie den oben abgebildeten Quellcode in *main.c*.

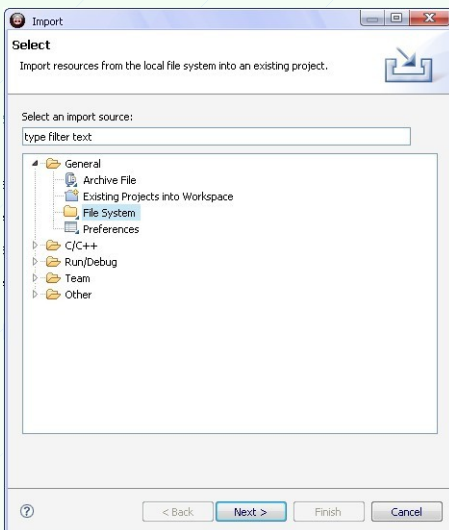
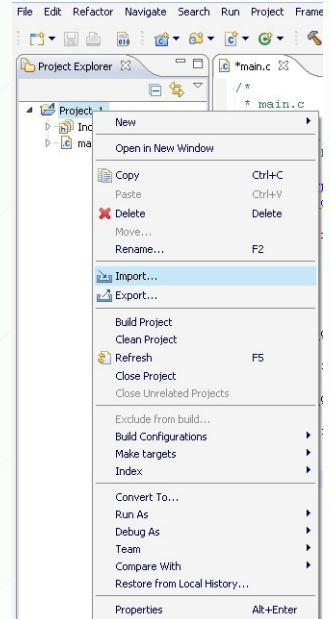


4.3. Hinzufügen der Bibliotheken

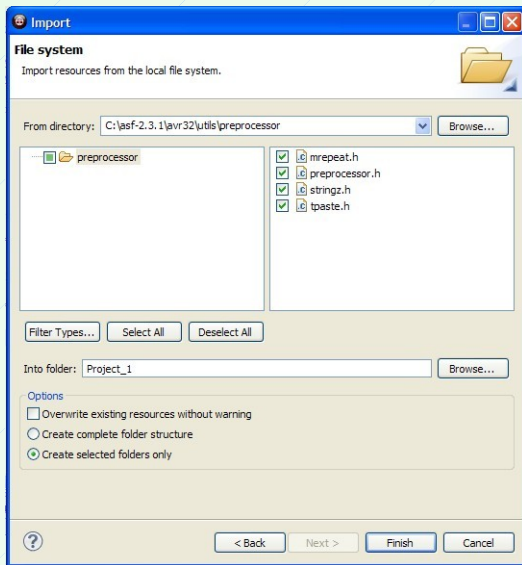


Nachdem wir das Programm geschrieben haben, müssen wir noch ein Paar Bibliotheken zu unserem Projekt hinzufügen. Diese Bibliotheken findet man in AVR32 UC3 Framework, z.B. im Ordner <C:\asf-2.3.1\avr32>

Klicken Sie mit rechter Maustaste auf den Projektnamen und wählen Sie *Import...* oder auf *File* → *Import...*



Wählen Sie im Ordner *General* → *File System* und klicken Sie auf die Schaltfläche „Next >“.



In diesem Fenster „Import“ können wir die Bibliotheken zu unserem Projekt hinzufügen. Geben Sie im Feld *From directory:* den Zielordner ein, z.B.

<C:\asf-2.3.1\avr32\utils\preprocessor> Der Inhalt dieses Ordners erscheint im unteren rechten Feld. Wählen Sie die benötigten Bibliotheken und klicken Sie auf die Schaltfläche „Finish“.

Es ist leider nicht möglich alle Bibliotheken auf einen Schlag zu importieren, wenn sie in unterschiedlichen Ordner liegen. Aus diesem Grund müssen wir in unserem Fall drei Mal das Fenster „Import“ aufrufen.

Die für uns benötigten Bibliotheken finden Sie in folgenden Ordnern:

- compiler.h und parts.h → <C:\asf-2.3.1\avr32\utils>
- gpio.h und gpio.c → <C:\asf-2.3.1\avr32\drivers\gpio>
- mrepeat.h, preprocessor.h, stringz.h und tpaste.h → <C:\asf-2.3.1\avr32\utils\preprocessor>

Wichtig!!!

Nachdem Sie alle Files importiert haben, öffnen Sie Header-File *compiler.h* und ändern Sie die Zeile

```
#include "interrupt.h"
```

durch

```
//#include "interrupt.h"
```

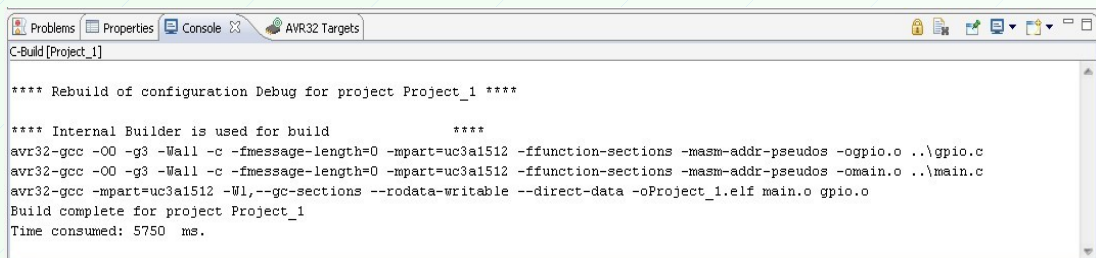
4.4. Compilieren des Projekts

Als erstes muss das gesamte Projekt gespeichert werden, *File* → *Save ALL*.

Compilieren Sie das Projekt:

- *Project* → *Build ALL* **oder**
- Tastenkombination [*Strg*]+[*B*] **oder**
- rechte Maustaste auf Ihren Projekt → *Build Project*

Das Ergebnis des Compilierens werden Sie im Fenster „*Console*“ sehen, wie es in unterem Abbild dargestellt ist.



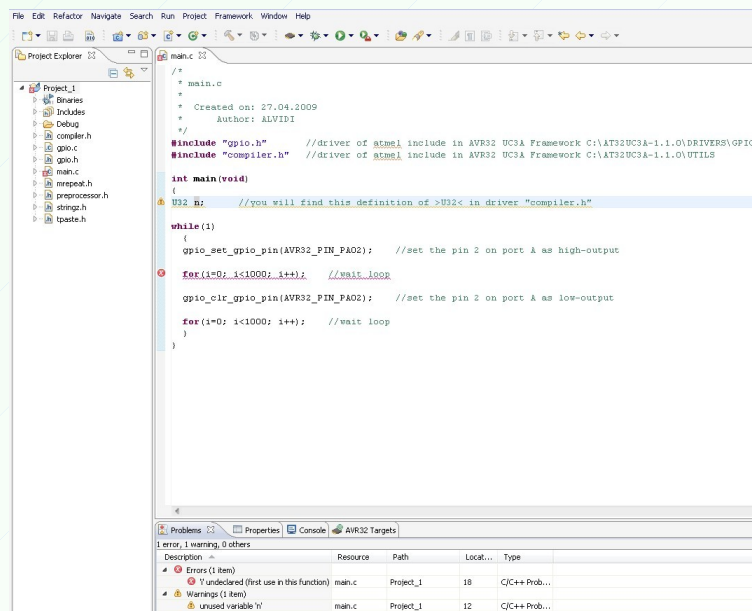
```

C-Build [Project_1]

**** Rebuild of configuration Debug for project Project_1 ****

**** Internal Builder is used for build ****
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=uc3a1512 -ffunction-sections -masm-addr-pseudos -ogpio.o ..\gpio.c
avr32-gcc -O0 -g3 -Wall -c -fmessage-length=0 -mpart=uc3a1512 -ffunction-sections -masm-addr-pseudos -omain.o ..\main.c
avr32-gcc -mpart=uc3a1512 -U1,--gc-sections --rodata-writable --direct-data -oProject_1.elf main.o gpio.o
Build complete for project Project_1
Time consumed: 5750 ms.
  
```

Falls während des Compilierens Fehler im Programm gefunden werden, dann erscheinen sie im Fenster „*Problems*“. Die untere Abbildung stellt diesen Fall vor.



```

main.c
/*
 * main.c
 * Created on: 27.04.2009
 * Author: ALVIDI
 */
#include "gpio.h" //driver of atmel include in AVR32 UC3A Framework C:\AT32UC3A-1.1.0\DRIVERS\GPIO
#include "compiler.h" //driver of atmel include in AVR32 UC3A Framework C:\AT32UC3A-1.1.0\UTILS

int main(void)
{
    U32 B; //you will find this definition of >U32< in driver "compiler.h"

    while(1)
    {
        gpio_set_gpio_pin(AVR32_PIN_PA02); //set the pin 2 on port A as high-output
        for(i=0; i<1000; i++); //wait loop

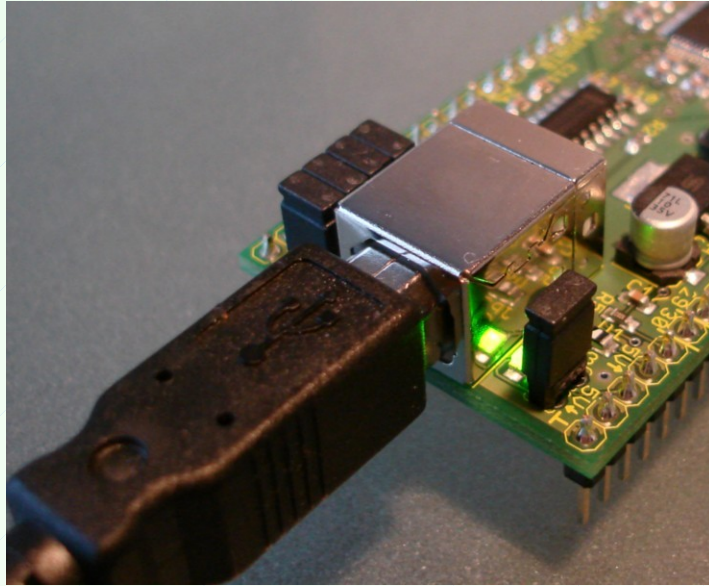
        gpio_clr_gpio_pin(AVR32_PIN_PA02); //set the pin 2 on port A as low-output
        for(i=0; i<1000; i++); //wait loop
    }
}
  
```

Description	Resource	Path	Locat...	Type
1 error, 1 warning, 0 others				
Errors (1 item)				
Undeclared (first use in this function)	main.c	Project_1	10	C/C++ Prob...
Warnings (1 item)				
unused variable 'i'	main.c	Project_1	12	C/C++ Prob...

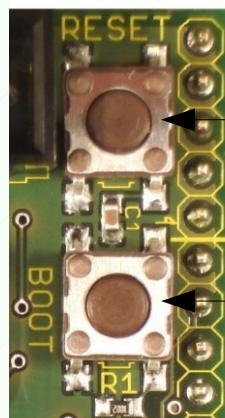
5. Hardware

5.1. Starten des Bootloaders

Stellen Sie eine USB-Verbindung zwischen Ihrem Computer und dem Hardware Gerät, z.B. AVR32-Modul. Wenn Jumper JP3 beim AVR32-Modul gesetzt ist, wird das Modul über USB mit 5V versorgt und die *Power LED* wird grün leuchten.



Halten Sie die „BOOT“-Taste gedrückt und drücken Sie kurzzeitig die „RESET“-Taste. Somit starten Sie den Bootloader



2. kurzzeitig drücken

1. gedrückt halten

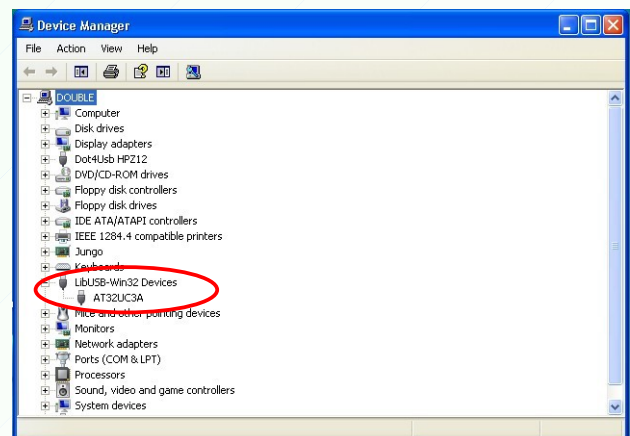
5.2. Installieren des USB-Treibers

Bei den bestehender USB-Verbindung und gestartetem Bootloader erscheint die untere Abbildung.

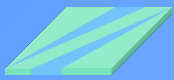


Wählen Sie *Install from a list or location (Advanced)* und klicken Sie auf die Schaltfläche „Next >“.

Geben Sie in weißem Feld den Zielordner des Treibers <C:\Program Files\Atmel\FIip 3.4.2\usb> ein und klicken Sie auf die Schaltfläche „Next >“.



Nach einer erfolgreichen Installation erscheint die linke obere Abbildung. Im rechten Abbild *Device Manager* wird das angeschlossene Gerät unter **LibUSB-Win32 Devices** → **AT32UC3A** sichtbar.



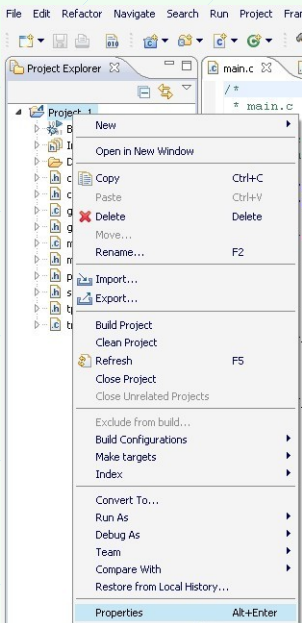
6. Programmieren

6.1. Erweiterung des Programms

Zum Programmieren mit USB-Bootloader müssen wir unseres Programm mit zwei Bibliotheken erweitern:

- trampoline_uc3.h und trampoline_uc3.S → <C:\asf-2.3.1\avr32\utils\startup>

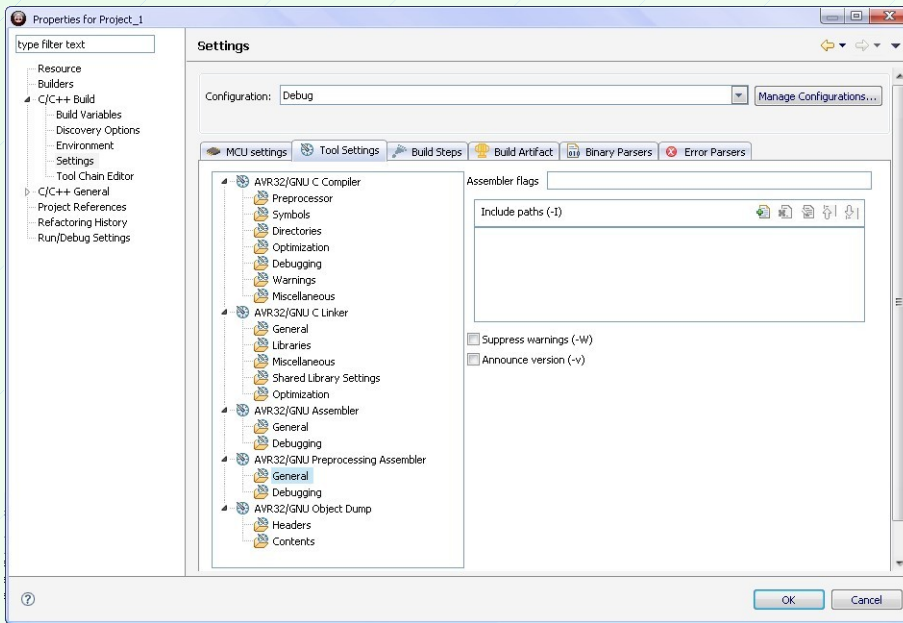
Wie man die Bibliotheken hinzufügt, siehe Kapitel 4.3 *Hinzufügen der Bibliotheken*.



Im nächsten Schritt müssen wir zwei Pfade zu unserem Projekt hinzufügen.

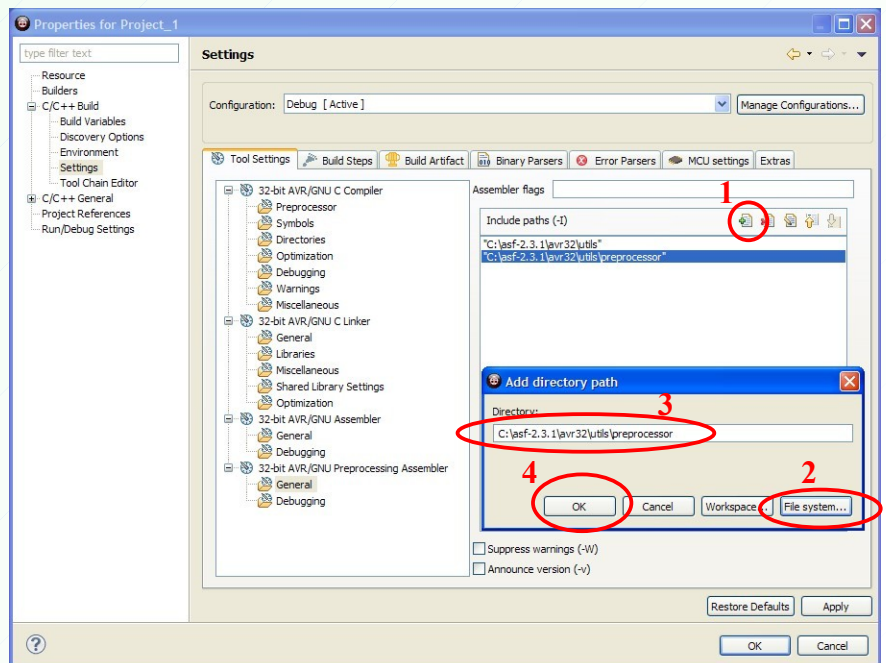
Klicken Sie mit rechter Maustaste auf den Projektnamen und wählen Sie *Properties* oder auf *File* → *Properties*

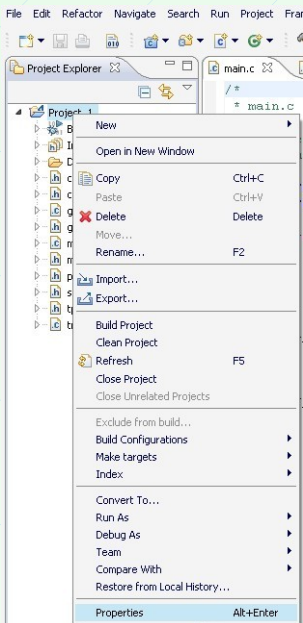
Wählen Sie im linken Teil dieses Fensters *C/C++Build* → *Settings*. Klicken Sie auf Sub-Fenster *Tool settings* → *AVR32/GNU Preprocessing Assembler* → *General*.



Im rechten Teil dieses Fensters sehen wir das Feld *Include paths (-I)*. In diesem Feld müssen wir zwei Ordner hinzufügen, indem wir das weiße Blatt mit grünem Kreuz anklicken und nacheinander folgende Ordner dazufügen:

[C:\asf-2.3.1\avr32\utils](#) und [C:\asf-2.3.1\avr32\utils\preprocessor](#)

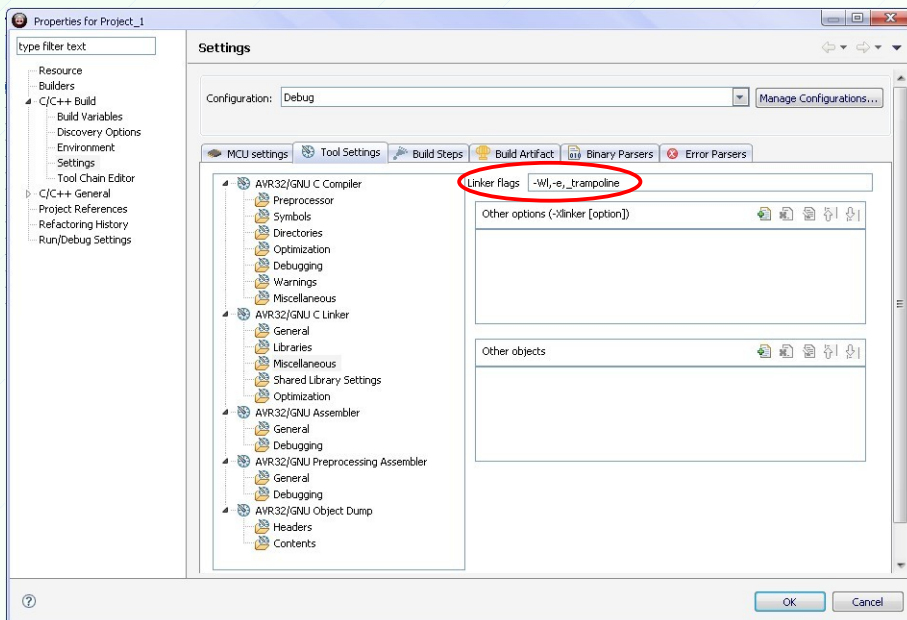


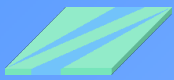


In diesem Schritt müssen wir dem Linker mitteilen, dass wir mit dem Bootloader arbeiten.

Klicken Sie mit rechter Maustaste auf den Projektnamen und wählen Sie *Properties* oder auf *File* → *Properties*

Wählen Sie im linken Fenster *C/C++ Build* → *Settings*. Klicken Sie auf Sub-Fenster *Tool settings* → *AVR32/GNU C Linker* → *Miscellaneous*. Geben Sie im Feld *Linker flags* *-Wl,-e,_trampoline* ein und bestätigen Sie die Änderungen mit der Schaltfläche „OK“.



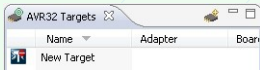


6.2. Installation des Programmers

Wie man mit dem Bootloader programmiert wird im pdf-File *AVR32 UC3 USB DFU Bootloader* von Atmel erklärt.

Siehe: http://www.atmel.com/dyn/resources/prod_documents/doc7745.pdf

Öffnen Sie das Fenster AVR32 Targets. Sie finden dieses Fenster auch unter *Window* → *Show View* → *AVR32 Targets*.



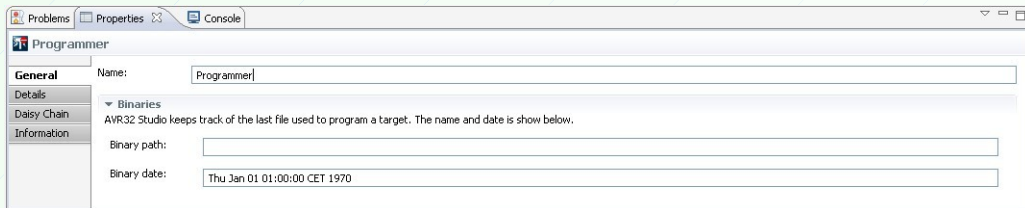
Klicken Sie auf das Symbol „Creates a new target“ im grauen Feld. Anschließend erscheint im weißen Feld *New Target*.



Mit der rechte Maustaste auf *New Target*, wählen Sie *Properties*.

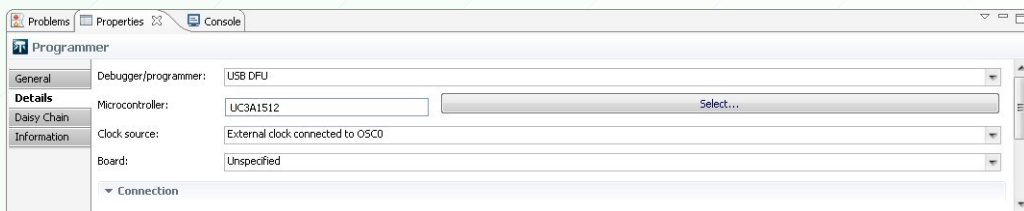
Im Fenster *Properties* müssen wir unseren Programmer konfigurieren.

Geben Sie im Feld *General* → *Name*: einen Namen dem Programmer ein, z.B. Programmer.



Im Fenster *Properties* → *Details* werden folgende Hardwareparameter gesetzt:

- * - **Debugger/programmer**: USB DFU
- * - **Microcontroller**: UC3A1512
- * - **Clock source**: Internal RC oscillator **oder** External clock connected to OSC0

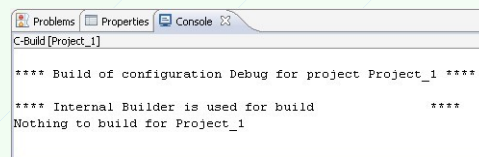


6.3. Programmieren mit AVR32 Studio

Alle bisherige Einstellungen sind einmalig. Daher das Schwierigste haben wir hinter uns gebracht. Jetzt bleibt das angenehme Teil.

1. Starten Sie den Bootloader (siehe Kapitel 5.1. *Starten des Bootloaders*).
2. Speichern Sie das Projekt *File* → *Save All*
3. Compilieren Sie das Projekt (siehe Kapitel 4.4. *Compilieren des Projekts*)

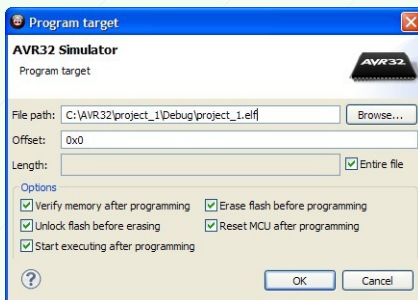
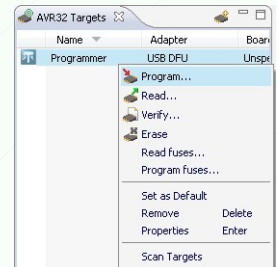
Falls Sie die unten stehende Meldung bekommen, heißt es, dass es im Programm seit dem letzten Compilervorgang nichts geändert hat.



```

C-Build [Project_1]
**** Build of configuration Debug for project Project_1 ****
**** Internal Builder is used for build ****
Nothing to build for Project_1
    
```

4. Klicken Sie mit der **rechten Maustaste** auf den von Ihnen vor-konfigurierten **Programmer** im Fenster *AVR32 Targets* und wählen Sie dort *Program ...*

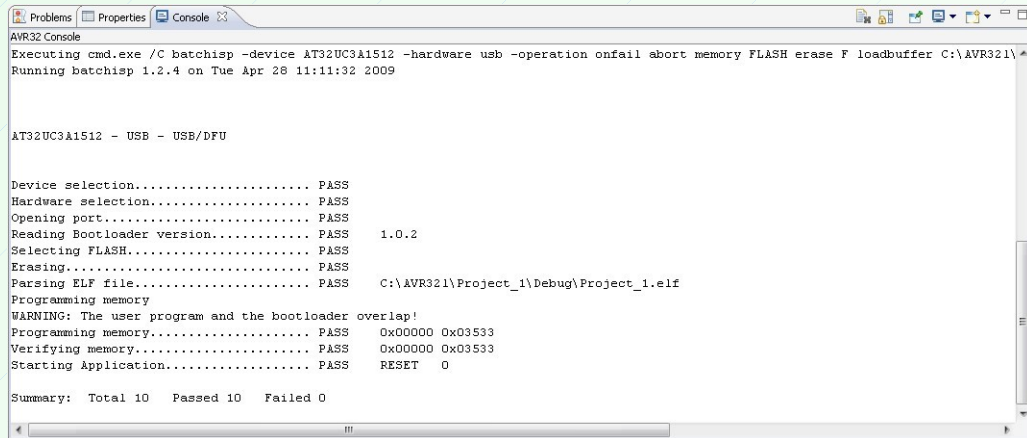


5. Geben Sie im Feld *File path*: das compilierte Elf-File ein. In unserem Fall ist es die folgende Adresse:
C:\AVR32\Project_1\Debug\Project_1.elf

Wichtig!

Wählen Sie **alle** Optionsfelder, wie es im linken Abbild dargestellt ist und klicken Sie anschließend auf die Schaltfläche „OK“.

Das Ergebnis des Programmieren wird im Fenster *Console* detailliert dargestellt (siehe untere Abbildung) .



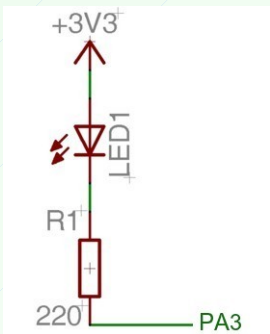
```
AVR32 Console
Executing cmd.exe /C batchisp -device AT32UC3A1512 -hardware usb -operation onfail abort memory FLASH erase F loadbuffer C:\AVR321\
Running batchisp 1.2.4 on Tue Apr 28 11:11:32 2009

AT32UC3A1512 - USB - USB/DFU

Device selection..... PASS
Hardware selection..... PASS
Opening port..... PASS
Reading Bootloader version..... PASS    1.0.2
Selecting FLASH..... PASS
Erasing..... PASS
Parsing ELF file..... PASS    C:\AVR321\Project_1\Debug\Project_1.elf
Programming memory
WARNING: The user program and the bootloader overlap!
Programming memory..... PASS    0x00000 0x03533
Verifying memory..... PASS    0x00000 0x03533
Starting Application..... PASS    RESET    0

Summary: Total 10 Passed 10 Failed 0
```

Um sich zu vergewissern, dass das Programm richtig funktioniert, schließen Sie an Port A Pin 3 in einer Reihe Widerstand 220 Ω und ein LED wie im unteren Schaltplan. Wenn die LED blinkt, dann haben Sie alles richtig gemacht.



Die weiteren Beispiele für AVR32 Controller Serie UC3 finden Sie im AVR32UC3x Software Framework. Im Ordner <C:\asf-2.3.1\avr32\drivers> befinden sich die Treiber und Quellcodebeispiele für ADC, PWM, RTC, USART, ...

7. Inhaltsverzeichnis

- **AVR32015: AVR32 Studio getting started**
http://www.atmel.com/dyn/resources/prod_documents/doc32086.pdf
- **AVR32 UC3 USB DFU Bootloader**
http://www.atmel.com/dyn/resources/prod_documents/doc7745.pdf