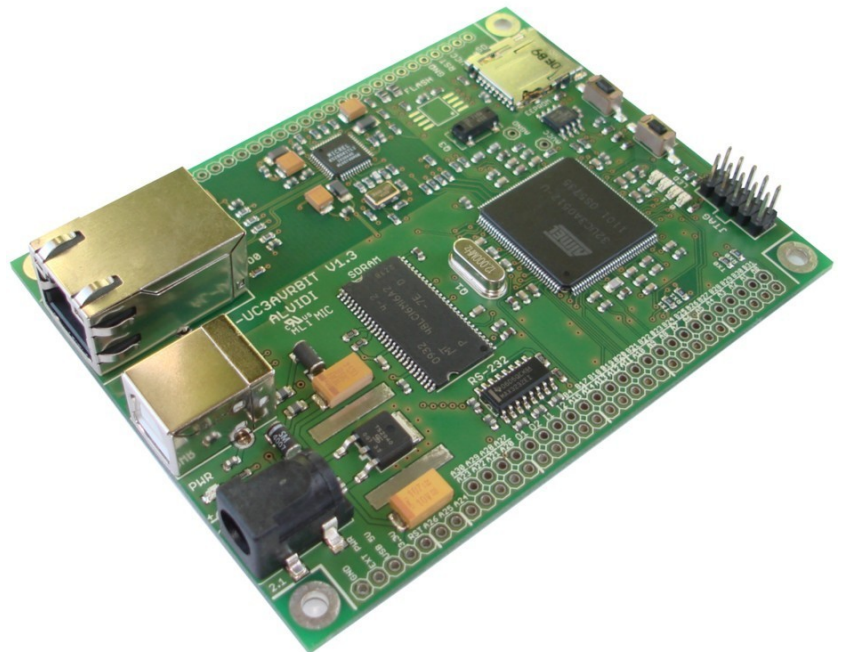


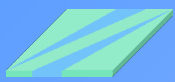
**ALVIDI**

# **Test Software**

*kompatibel mit dem Board  
AL-UC3AVRBIT*

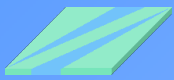
**Revision vom 15.04.2011**





## Verzeichnis

<b>1.</b>	<b>Einleitung</b>	<b>3</b>
<b>2.</b>	<b>Beispiele</b>	<b>3</b>
2.1.	Ethernet	3
2.2.	SDRAM	9
2.3.	microSD Kartenslot	14
2.4.	DataFlash	19
2.5.	EEPROM	23
<b>3.</b>	<b>Alle Links im Überblick</b>	<b>26</b>
<b>4.</b>	<b>Disclaimer</b>	<b>28</b>



## 1. Einleitung

Dieses Dokument beschreibt die Kommunikation zwischen AT32UC3A0512 Controller und Peripherie auf dem AVR32 Board AL-UC3AVRBIT anhand Atmel Beispiele aus AVR Software Framework. In dem AVR Software Framework finden Sie auch alle notwendigen Treiber und Bibliotheken mit genauer Beschreibung in HTML-Datei. Bitte laden Sie dieses File runter und entpacken Sie z.B. auf C:\

- **AVR Software Framework 2.3.1 (30 MB, revision 2.3.1, updated 3/11)**

[http://www.atmel.com/dyn/products/tools\\_card.asp?tool\\_id=4192&category\\_id=163&family\\_id=607&subfamily\\_id=2138](http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4192&category_id=163&family_id=607&subfamily_id=2138)

Für Einsteiger empfehlen wir mit dem Dokument „**Erster Schritt**“ anzufangen.

- **Erster Schritt (2 MB, Revision vom 23.03.2011)**

[http://alvidi.de/data\\_sheets/erster\\_schritt.pdf](http://alvidi.de/data_sheets/erster_schritt.pdf)

## 2. Beispiele

### 2.1. Ethernet

Mit seinen Produkten bietet Atmel Hardware- und Softwareumgebung mit Beispielprogrammen, Treibern und Bibliotheken an. Um Ethernet auf dem AVR32 Board zu testen, nehmen wir das Beispielprogramm „*macb\_example.c*“ aus dem Ordner `C:\asf-2.3.1\avr32\drivers\macb\example` Zum fehlerfreien Kompilieren werden wir zuerst alle Bibliotheken und Treiber für das Atmel Produkt ATEVK1100 in das Testsoftware integrieren. Anschließend ersetzen bzw. ergänzen wir die vorhandene Bibliotheken und Treiber mit dem Quellcode für das Board AL-UC3AVRBIT. Unser Ablauf sieht folgendermaßen aus:

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen (siehe Seite 10 „Erster Schritt“)
- das Programm erweitern (siehe Seite 15 „Erster Schritt“)
- Ethernet Projekt an das AVR32 Board UC3AVRBIT anpassen
- das Ethernet Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen
- ein fertig kompiliertes File für Ethernet Projekt finden Sie hier:
- <http://www.alvidi.de/prog/ethernet.elf> (11,1 MB) oder
- <http://www.alvidi.de/prog/ethernet.zip> (1,79 MB)

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen

C:\asf-2.3.1\avr32\drivers\macb\example\macb\_example.c

C:\asf-2.3.1\avr32\drivers\macb\macb.c und macb.h

C:\asf-2.3.1\avr32\drivers\macb\example\at32uc3a0512\_evk1100\conf\_eth.h

C:\asf-2.3.1\avr32\utils\debug\print\_funcs.c, print\_funcs.h, debug.c und debug.h

C:\asf-2.3.1\avr32\utils\compiler.h und parts.h

C:\asf-2.3.1\avr32\utils\preprocessor\preprocessor.h, mrePEAT.h, stringz.h und tpaste.h

C:\asf-2.3.1\common\utils\interrupt.h

C:\asf-2.3.1\common\utils\interrupt\interrupt\_avr32.h

ersetzen Sie in *"interrupt.h"*

```
#include <parts.h>
```

mit

```
#include "parts.h"
```

und

```
#include "interrupt/interrupt_avr32.h"
```

mit

```
#include "interrupt_avr32.h"
```

ersetzen Sie in *"interrupt\_avr32.h"*

```
#include <compiler.h>
```

```
#include <preprocessor/tpaste.h>
```

mit

```
#include "compiler.h"
```

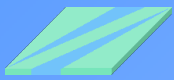
```
#include "tpaste.h"
```

und

```
#include <intc.h>
```

mit

```
#include "intc.h"
```



C:\asf-2.3.1\avr32\drivers\intc\exception.S, intc.c und intc.h  
C:\asf-2.3.1\avr32\drivers\usart\usart.c und usart.h  
C:\asf-2.3.1\avr32\boards\evk1100\evk1100.h, led.h und led.c  
C:\asf-2.3.1\common\boards\board.h

ergänzen Sie *"board.h"* unter

```
#define UC3_L0_QT600          26
#define USER_BOARD          99
#define DUMMY_BOARD         100
```

mit dieser Zeile

```
#define BOARD                 EVK1100
```

ersetzen Sie in diesem File die Zeile

```
#include "evk1100/evk1100.h"
```

mit

```
#include "evk1100.h"
```

C:\asf-2.3.1\avr32\drivers\gpio\gpio.c und gpio.h  
C:\asf-2.3.1\avr32\drivers\flashc\flashc.h und flashc.c  
C:\asf-2.3.1\avr32\drivers\pm\power\_clocks\_lib.c, power\_clocks\_lib.h, pm.c, pm.h und pm\_conf\_clocks.c  
C:\asf-2.3.1\avr32\components\ethernet\_phy\ethernet\_phy.h und zwei Ordner dp83848 und dummy\_phy

ersetzen Sie in dem integrierten Ordner dp83848 *"dp83848.h"*

```
#include "macb.h"
```

mit

```
#include "../macb.h"
```

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für ATEVK1100 in Ihrem Projektordner → Debug liegen.

## ● das Programm erweitern

Falls die Programmierung mit dem USB Bootloader erfolgt, muss das Programm ergänzt werden. In der Dokumentation "Erster Schritt" Kapitel 6.1. Erweiterung des Programms erfahren Sie mehr über diesen Schritt.

## ● Ethernet Projekt an das AVR32 Board UC3AVRBIT anpassen

in „*ethernet\_phy.h*“ ergänzen Sie unterhalb

```
#if defined( PHY_DP83848 )
#include "dp83848/dp83848.h"
#elif defined( PHY_RTL8201 )
#include "rtl8201/rtl8201.h"
```

mit folgenden Zeilen

```
#elif defined ( PHY_KSZ8041TL )
#include "ksz8041tl.h"
```

laden Sie diese Bibliotheken runter und integrieren sie in das Projekt

<http://alvidi.de/lib/ksz8041tl.h>

<http://alvidi.de/lib/uc3avrbit.h>

[http://alvidi.de/lib/led\\_func.c](http://alvidi.de/lib/led_func.c)

[http://alvidi.de/lib/led\\_func.h](http://alvidi.de/lib/led_func.h)

In Bibliothek "*board.h*" ändern Sie die Zeilen

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support

#define BOARD                 EVK1100
```

wie folgt:

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support
#define UC3AVRBIT           27  //!< UC3AVRBIT board with AT32UC3A0512

#define BOARD                UC3AVRBIT
```

und

```
#if BOARD == EVK1100
#include "evk1100.h"
```

ersetzen Sie mit

```
#if BOARD == UC3AVRBIT
#include "uc3avrbit.h"
#elif BOARD == EVK1100
#include "evk1100.h"
```

C:\asf-2.5.1\avr32\services\delay\delay.c und delay.h

C:\asf-2.5.1\avr32\drivers\cpu\cycle\_counter\cycle\_counter.h

ersetzen Sie im File *"macb\_example.c"* die Zeile

```
#elif ( (BOARD == EVK1100) || (BOARD==EVK1105) )
```

mit

```
#elif ( (BOARD == EVK1100) || (BOARD==EVK1105) || (BOARD==UC3AVRBIT) )
```

ersetzen Sie in der Bibliothek *"print\_funcs.h"* die Zeile

```
#if BOARD == EVK1100
```

mit:

```
#if BOARD == UC3AVRBIT
# define DBG_USART                (&AVR32_USART0)
# define DBG_USART_RX_PIN          AVR32_USART0_RXD_0_0_PIN
# define DBG_USART_RX_FUNCTION    AVR32_USART0_RXD_0_0_FUNCTION
# define DBG_USART_TX_PIN          AVR32_USART0_TXD_0_0_PIN
# define DBG_USART_TX_FUNCTION    AVR32_USART0_TXD_0_0_FUNCTION
# define DBG_USART_BAUDRATE        57600
#elif BOARD == EVK1100
```

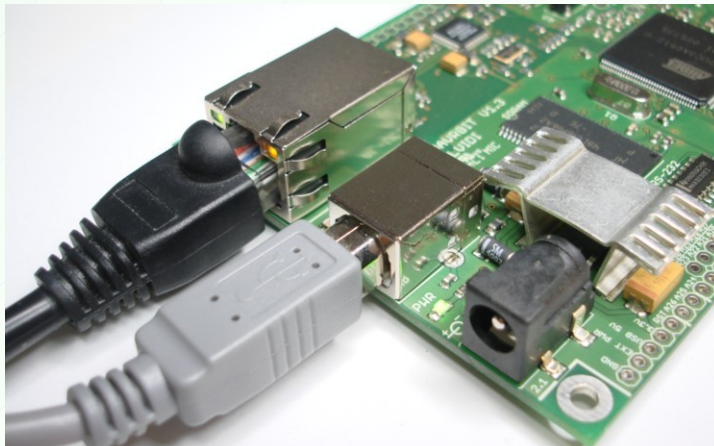
**Wichtig!**

**Entfernen** Sie aus diesem Projekt ATEVK1100 Files: [evk1100.h](#), [led.c](#) und [led.h](#)

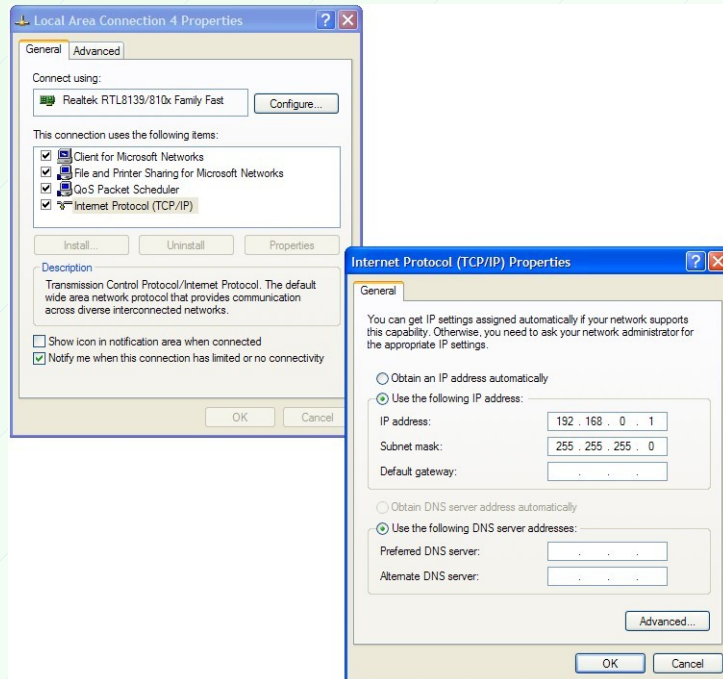
Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für UC3AVRBIT in Ihrem Projektordner → Debug liegen.

- das Ethernet Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen

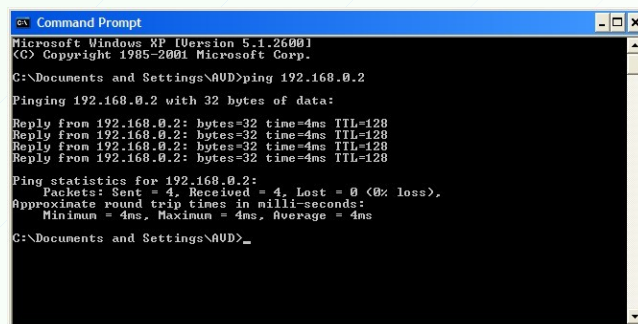
Verbinden Sie AVR32 Board und Ihr PC mit Hilfe der Ethernet und USB Kabel



Falls Sie keine freie Ethernet RJ45 Buchse in Ihrem PC haben, benötigen Sie entweder eine zusätzliche *Ethernet Karte* mit Ethernet Crossover Kabel oder ein *Ethernet Switch*. Für den Test müssen wir die angeschlossene Schnittstelle konfigurieren. Dafür gehen Sie auf Start → Einstellungen → Netzwerkverbindungen → Lokal Area Connection X (es handelt sich hier um eine neu erkannte Ethernet Verbindung) mit rechter Maustaste → Eigenschaften → Internet Protokoll (TCP/IP) → Eigenschaften. Tragen Sie im Fenster IP Adresse 192.168.0.1 und im Fenster Subnetzmaske 255.255.255.0 ein. Nachdem Sie alles eingetragen haben, sieht es bei Ihnen folgendermaßen aus:



Als nächstes muss „Eingabeaufforderung“ gestartet werden. Dafür gehen Sie auf Start → Programme → Zubehör → Eingabeaufforderung. Schreiben Sie dort „ping 192.168.0.2“ und bestätigen Sie die Eingabe mit Enter-Taste. Wenn alles, wie oben beschrieben, gemacht wurde, erhalten Sie folgendes Abbild.



Ethernet Test ist erfolgreich abgeschlossen!

## 2.2. SDRAM

Um SDRAM auf dem AVR32 Board zu testen, nehmen wir das Beispielprogramm „*sdramc\_example.c*“ aus dem Ordner

*C:\asf-2.3.1\avr32\components\memory\sdram\mt48lc16m16a2tg7e\example*

Zum fehlerfreien Compilieren werden wir zuerst alle Bibliotheken und Treiber für das Atmel Produkt ATEVK1100 in die Testsoftware integrieren. Anschließend ersetzen bzw. ergänzen wir die vorhandenen Bibliotheken und Treiber mit dem Quellcode für das Board AL-UC3AVRBIT. Unser Ablauf sieht folgendermaßen aus:

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen (siehe Seite 10 „Erster Schritt“)
- das Programm erweitern (siehe Seite 15 „Erster Schritt“)
- SDRAM Projekt an das AVR32 Board UC3AVRBIT anpassen
- das SDRAM Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen
- ein fertig kompiliertes File für SDRAM Projekt finden Sie hier:
- <http://www.alvidi.de/prog/sdram.elf> (9,30 MB) oder
- <http://www.alvidi.de/prog/sdram.zip> (1,48 MB)

● neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)

● Bibliotheken und Treiber hinzufügen

*C:\asf-2.3.1\avr32\components\memory\sdram\mt48lc16m16a2tg7e\example\sdramc\_example.c*

*C:\asf-2.3.1\avr32\components\memory\sdram\mt48lc16m16a2tg7e\mt48lc16m16a2tg7e.h*

*C:\asf-2.3.1\avr32\drivers\ebi\sdramc\sdramc.c* und *sdramc.h*

*C:\asf-2.3.1\avr32\utils\debug\print\_funcs.c* und *print\_funcs.h* *debug.c* und *debug.h*

*C:\asf-2.3.1\avr32\utils\compiler.h* und *parts.h*

*C:\asf-2.3.1\avr32\utils\preprocessor\preprocessor.h*, *mrepeat.h*, *stringz.h* und *tpaste.h*

*C:\asf-2.3.1\avr32\boards\evk1100\evk1100.h*, *led.h* und *led.c*

*C:\asf-2.3.1\common\boards\board.h*

ergänzen Sie "*board.h*" unter

```
#define UC3_L0_QT600          26
#define USER_BOARD          99
#define DUMMY_BOARD         100
```

mit dieser Zeile

```
#define BOARD                 EVK1100
```

ersetzen Sie in diesem File die Zeile

```
#include "evk1100/evk1100.h"
```

mit

```
#include "evk1100.h"
```

```
C:\asf-2.3.1\common\utils\interrupt.h
```

```
C:\asf-2.3.1\common\utils\interrupt\interrupt_avr32.h
```

ersetzen Sie in *"interrupt.h"*

```
#include <parts.h>
```

mit

```
#include "parts.h"
```

und

```
#include "interrupt/interrupt_avr32.h"
```

mit

```
#include "interrupt_avr32.h"
```

ersetzen Sie in *"interrupt\_avr32.h"*

```
#include <compiler.h>
```

```
#include <preprocessor/tpaste.h>
```

mit

```
#include "compiler.h"
```

```
#include "tpaste.h"
```

und

```
#include <intc.h>
```

mit

```
#include "intc.h"
```

```
C:\asf-2.3.1\avr32\drivers\intc\exception.S, intc.c und intc.h
```

```
C:\asf-2.3.1\avr32\drivers\usart\usart.c und usart.h
```

C:\asf-2.3.1\avr32\drivers\gpio\gpio.c und gpio.hc

ersetzen Sie in "evk1100.h"

```
#define SDRAM_PART_HDR "mt48lc16m16a2tg7e/mt48lc16m16a2tg7e.h"
```

mit

```
#define SDRAM_PART_HDR "mt48lc16m16a2tg7e.h"
```

C:\asf-2.3.1\avr32\drivers\cpu\cycle\_counter\cycle\_counter.h

C:\asf-2.3.1\avr32\drivers\pm\power\_clocks\_lib.c, power\_clocks\_lib.h, pm.c und pm.h

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für ATEVK1100 in Ihrem Projektordner → Debug liegen.

## ● das Programm erweitern

Falls die Programmierung mit dem USB Bootloader erfolgt, muss das Programm ergänzt werden. In der Dokumentation "Erster Schritt" Kapitel 6.1. Erweiterung des Programms erfahren Sie mehr über diesen Schritt.

## ● SDRAM Projekt an das AVR32 Board UC3AVRBIT anpassen

In Bibliothek "board.h" ändern Sie die Zeilen

```
#define USER_BOARD 99 //!< User-reserved board (if any).  
#define DUMMY_BOARD 100 //!< Dummy board to support
```

```
#define BOARD EVK1100
```

wie folgt:

```
#define USER_BOARD 99 //!< User-reserved board (if any).  
#define DUMMY_BOARD 100 //!< Dummy board to support  
#define UC3AVRBIT 27 //!< UC3AVRBIT board with AT32UC3A0512
```

```
#define BOARD UC3AVRBIT
```

und

```
#if BOARD == EVK1100
```

ersetzen Sie mit

```
#if BOARD == UC3AVRBIT  
#include "uc3avrbit.h"  
#elif BOARD == EVK1100
```

laden Sie diese Bibliotheken runter und importieren sie in das Projekt

<http://alvidi.de/lib/uc3avrbit.h>

[http://alvidi.de/lib/led\\_func.c](http://alvidi.de/lib/led_func.c)

[http://alvidi.de/lib/led\\_func.h](http://alvidi.de/lib/led_func.h)

ergänzen Sie in der Bibliothek "*print\_funcs.h*" die Zeile

```
#if BOARD == EVK1100
```

wie folgt:

```
#if BOARD == UC3AVRBIT
# define DBG_USART                (&AVR32_USART0)
# define DBG_USART_RX_PIN          AVR32_USART0_RXD_0_0_PIN
# define DBG_USART_RX_FUNCTION    AVR32_USART0_RXD_0_0_FUNCTION
# define DBG_USART_TX_PIN          AVR32_USART0_TXD_0_0_PIN
# define DBG_USART_TX_FUNCTION    AVR32_USART0_TXD_0_0_FUNCTION
# define DBG_USART_BAUDRATE        57600
#elif BOARD == EVK1100
```

und in „*sdramc\_example.c*“

```
#if BOARD == EVK1100
# define LED_SDRAM_WRITE          LED_BIO_RED
```

durch

```
#if BOARD == UC3AVRBIT
# define LED_SDRAM_WRITE          LED0_GPIO
# define LED_SDRAM_READ           LED1_GPIO
# define LED_SDRAM_ERRORS         LED2_GPIO
# define LED_SDRAM_OK             LED3_GPIO
#elif BOARD == EVK1100
```

ersetzen Sie in File "*sdramc.c*" die Zeile

```
#if BOARD == EVK1100 || BOARD == EVK1104 || BOARD == EVK1105 || BOARD ==
UC3_A3_XPLAINED
```

durch

```
#if BOARD == EVK1100 || BOARD == EVK1104 || BOARD == EVK1105 || BOARD ==
UC3_A3_XPLAINED || BOARD == UC3AVRBIT
```

**Wichtig!**

**Entfernen** Sie aus diesem Projekt ATEVK1100 Files: [evk1100.h](#), [led.c](#) und [led.h](#)

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für UC3AVRBIT in Ihrem Projektordner → Debug liegen.

- das SDRAM Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen

Das Programm kann man auf zwei Weisen testen: mit LED oder mit einem Terminal Programm z.B. HyperTerminal

○ LED-TEST

**1. Schritt**

LED0 blinkt / dabei wird SDRAM mit dem bestimmten Inhalt beschrieben

**2.Schritt**

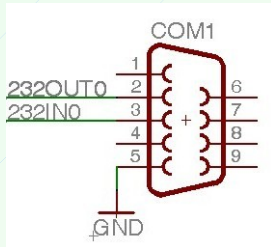
LED1 blinkt / dabei wird SDRAM mit dem beschriebenen Inhalt verglichen

**3.Schritt**

LED4 blinkt / wenn der Test erfolgreich war oder

LED3 blinkt / wenn ein Fehler aufgetreten ist

○ TERMINAL-TEST



**1.Schritt**

Schließen Sie an das AVR32 Board die serielle Schnittstelle an.  
232OUT0 = OUT1, 232IN0 = IN1, GND =GND

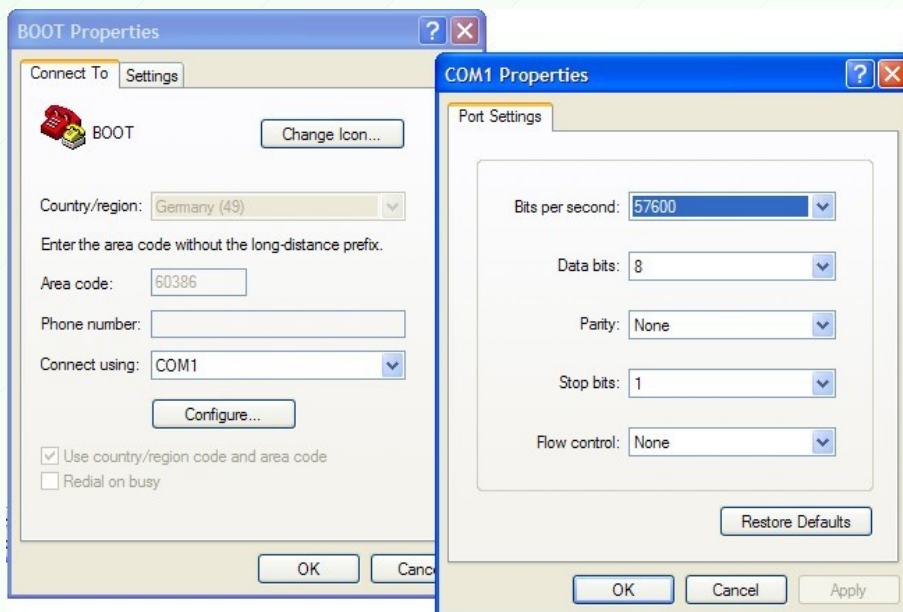
**2.Schritt**

Starten Sie z.B. HyperTerminal

Starten → Programmieren → Zubehör → Kommunikation → Hyperterminal

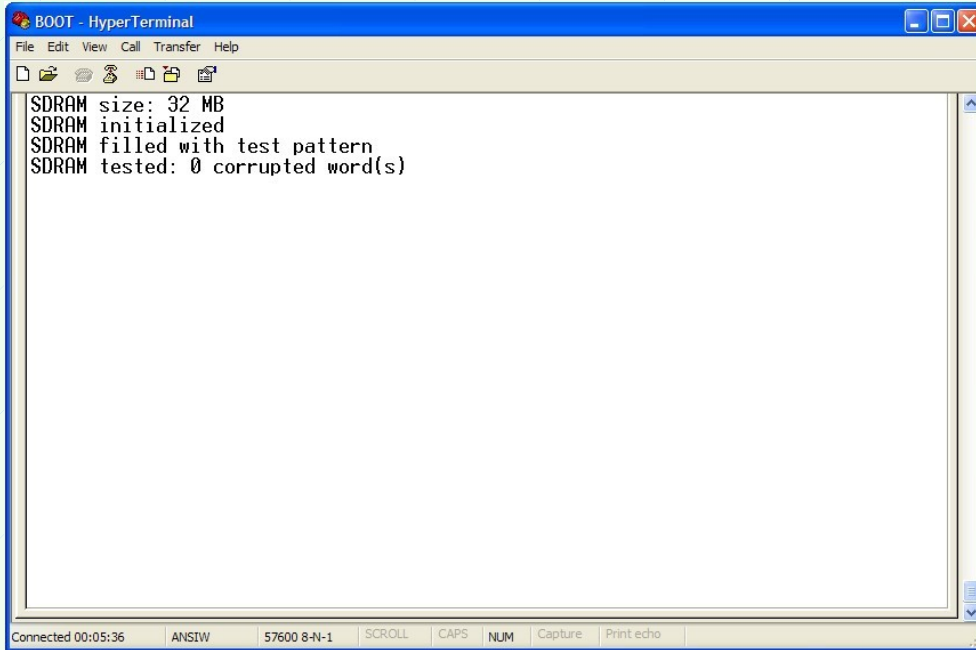
**3.Schritt**

Gehen Sie auf Datei → Eigenschaften → Konfigurieren und übernehmen Sie die Angaben, wie im unteren Abbild



#### 4.Schritt

Klicken Sie in oberer Leiste des HyperTerminal Anrufen → Verbinden. Sie erhalten die untere Abbildung.



```
BOOT - HyperTerminal
File Edit View Call Transfer Help
SDRAM size: 32 MB
SDRAM initialized
SDRAM filled with test pattern
SDRAM tested: 0 corrupted word(s)
Connected 00:05:36  ANSI  57600 8-N-1  SCROLL  CAPS  NUM  Capture  Print echo
```

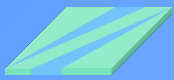
SDRAM Test ist erfolgreich abgeschlossen!

## 2.3. microSD Kartenslot

Um microSD Kartenslot auf dem AVR32 Board zu testen, nehmen wir das Beispielprogramm „*sd\_mmc\_spi\_example.c*“ aus dem Ordner

*C:\asf-2.3.1\avr32\components\memory\sd\_mmc\sd\_mmc\_spi\example* Zum fehlerfreien Compilieren werden wir zuerst alle Bibliotheken und Treiber für das Atmel Produkt ATEVK1100 in die Testsoftware integrieren. Anschließend ersetzen bzw. ergänzen wir die vorhandenen Bibliotheken und Treiber mit dem Quellcode für das Board AL-UC3AVRBIT. Unser Ablauf sieht folgendermaßen aus:

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen (siehe Seite 10 „Erster Schritt“)
- das Programm erweitern (siehe Seite 15 „Erster Schritt“)
- microSD Kartenslot Projekt an das AVR32 Board UC3AVRBIT anpassen
- das microSD Kartenslot Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen
- ein fertig kompiliertes File für microSD Kartenslot Projekt finden Sie hier:
- <http://www.alvidi.de/prog/sdcard.elf> (11,1 MB) oder
- <http://www.alvidi.de/prog/sdcard.zip> (1,78 MB)



- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen

C:\asf-2.3.1\avr32\components\memory\sd\_mmc\sd\_mmc\_spi\example\sd\_mmc\_spi\_example.c und dummy.h

C:\asf-2.3.1\avr32\utils\compiler.h und parts.h

C:\asf-2.3.1\avr32\utils\preprocessor\preprocessor.h, mrepeat.h, stringz.h und tpaste.h

C:\asf-2.3.1\common\utils\interrupt.h

C:\asf-2.3.1\common\utils\interrupt\interrupt\_avr32.h

ersetzen Sie in *"interrupt.h"*

```
#include <parts.h>
```

durch

```
#include "parts.h"
```

und

```
#include "interrupt/interrupt_avr32.h"
```

durch

```
#include "interrupt_avr32.h"
```

erstzen Sie in *"interrupt\_avr32.h"*

```
#include <compiler.h>
```

```
#include <preprocessor/tpaste.h>
```

durch

```
#include "compiler.h"
```

```
#include "tpaste.h"
```

und

```
#include <intc.h>
```

durch

```
#include "intc.h"
```

C:\asf-2.3.1\avr32\drivers\intc\exception.S, intc.c und intc.h  
C:\asf-2.3.1\avr32\drivers\usart\usart.c und usart.h  
C:\asf-2.3.1\avr32\boards\evk1100\evk1100.h, led.h und led.c  
C:\asf-2.3.1\common\boards\board.h

ergänzen Sie *"board.h"* unter

```
#define UC3_L0_QT600          26
#define USER_BOARD          99
#define DUMMY_BOARD         100
```

mit dieser Zeile

```
#define BOARD                 EVK1100
```

ersetzen Sie in diesem File die Zeile

```
#include "evk1100/evk1100.h"
```

durch

```
#include "evk1100.h"
```

C:\asf-2.3.1\avr32\drivers\pm\power\_clocks\_lib.c, power\_clocks\_lib.h, pm.c, pm.h und pm\_conf\_clocks.c  
C:\asf-2.3.1\avr32\drivers\gpio\gpio.c und gpio.h  
C:\asf-2.3.1\avr32\drivers\spi\spi.c und spi.h  
C:\asf-2.3.1\avr32\drivers\pdca\pdca.h und pdca.c  
C:\asf-2.3.1\avr32\utils\debug\print\_funcs.c und print\_funcs.h debug.c und debug.h  
C:\asf-2.3.1\avr32\components\memory\sd\_mmc\sd\_mmc\_spi\sd\_mmc\_spi.c und sd\_mmc\_spi.h  
C:\asf-2.3.1\avr32\components\memory\sd\_mmc\sd\_mmc\_spi\example\conf\conf\_access.h und conf\_sd\_mmc\_spi.h

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für ATEVK1100 in Ihrem Projektordner → Debug liegen.

## ● das Programm erweitern

Falls die Programmierung mit dem USB Bootloader erfolgt, muss das Programm ergänzt werden. In der Dokumentation "Erster Schritt" Kapitel 6.1. Erweiterung des Programms erfahren Sie mehr über diesen Schritt.

- **microSD Kartenslot Projekt an das AVR32 Board UC3AVRBIT anpassen**

laden Sie diese Bibliotheken runter und integrieren sie in das Projekt

<http://alvidi.de/lib/uc3avrbit.h>

[http://alvidi.de/lib/led\\_func.c](http://alvidi.de/lib/led_func.c)

[http://alvidi.de/lib/led\\_func.h](http://alvidi.de/lib/led_func.h)

In der Bibliothek "*board.h*" ändern Sie die Zeilen

```
#define USER_BOARD          99  ///< User-reserved board (if any).
#define DUMMY_BOARD         100  ///< Dummy board to support
```

```
#define BOARD                EVK1100
```

wie folgt

```
#define USER_BOARD          99  ///< User-reserved board (if any).
#define DUMMY_BOARD         100  ///< Dummy board to support
#define UC3AVRBIT           27  ///< UC3AVRBIT board with AT32UC3A0512
```

```
#define BOARD                UC3AVRBIT
```

und ersetzen Sie

```
#if BOARD == EVK1100
    #include "evk1100.h"
```

durch

```
#if BOARD == UC3AVRBIT
    #include "uc3avrbit.h"
#elif BOARD == EVK1100
    #include "evk1100.h"
```

ergänzen Sie in der Bibliothek "*print\_funcs.h*" die Zeile

```
#if BOARD == EVK1100
```

wie folgt

```
#if BOARD == UC3AVRBIT
#   define DBG_USART                (&AVR32_USART0)
#   define DBG_USART_RX_PIN         AVR32_USART0_RXD_0_0_PIN
#   define DBG_USART_RX_FUNCTION   AVR32_USART0_RXD_0_0_FUNCTION
#   define DBG_USART_TX_PIN         AVR32_USART0_TXD_0_0_PIN
#   define DBG_USART_TX_FUNCTION   AVR32_USART0_TXD_0_0_FUNCTION
#   define DBG_USART_BAUDRATE       57600
#elif BOARD == EVK1100
```



## 2.4. DataFlash

Um Dataflash auf dem AVR32 Board zu testen, nehmen wir das Beispielprogramm „*at45dbx\_example.c*“ aus dem Ordner

**C:\asf-2.3.1\common\components\memory\data\_flash\at45dbx\\_asf\_v1\example** Zum fehlerfreien Compilieren werden wir zuerst alle Bibliotheken und Treiber für das Atmel Produkt ATEVK1100 in die Testsoftware integrieren. Anschließend ersetzen bzw. ergänzen wir die vorhandenen Bibliotheken und Treiber mit dem Quellcode für das Board AL-UC3AVRBIT. Unser Ablauf sieht folgendermaßen aus:

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen (siehe Seite 10 „Erster Schritt“)
- das Programm erweitern (siehe Seite 15 „Erster Schritt“)
- Dataflash Projekt an das AVR32 Board UC3AVRBIT anpassen
- das Dataflash Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen
- ein fertig kompiliertes File für Dataflash Projekt finden Sie hier:
  - <http://www.alvidi.de/prog/dataflash.elf> (10,2 MB) oder
  - <http://www.alvidi.de/prog/dataflash.zip> (1,63 MB)

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen

**C:\asf-2.3.1\common\components\memory\data\_flash\at45dbx\\_asf\_v1\example\at45dbx\_example.c, conf\_access.h** und **conf\_at45dbx.h**

**C:\asf-2.3.1\avr32\utils\compiler.h** und **parts.h**

**C:\asf-2.3.1\avr32\utils\preprocessor\preprocessor.h, mrepeat.h, stringz.h** und **tpaste.h**

**C:\asf-2.3.1\common\utils\interrupt.h**

**C:\asf-2.3.1\common\utils\interrupt\interrupt\_avr32.h**

ersetzen Sie in *"interrupt.h"*

```
#include <parts.h>
```

durch

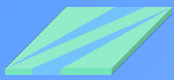
```
#include "parts.h"
```

und

```
#include "interrupt/interrupt_avr32.h"
```

durch

```
#include "interrupt_avr32.h"
```



erstzten Sie in *"interrupt\_avr32.h"*

```
#include <compiler.h>
#include <preprocessor/tpaste.h>
```

durch

```
#include "compiler.h"
#include "tpaste.h"
```

und

```
#include <intc.h>
```

durch

```
#include "intc.h"
```

C:\asf-2.3.1\avr32\drivers\intc\exception.S, intc.c und intc.h

C:\asf-2.3.1\avr32\drivers\usart\usart.c und usart.h

C:\asf-2.3.1\avr32\boards\evk1100\evk1100.h, led.h und led.c

C:\asf-2.3.1\common\boards\board.h

ergänzen Sie *"board.h"* unter

```
#define UC3_L0_QT600          26
#define USER_BOARD          99
#define DUMMY_BOARD         100
```

mit dieser Zeile

```
#define BOARD                EVK1100
```

ersetzen Sie in diesem File die Zeile

```
#include "evk1100/evk1100.h"
```

durch

```
#include "evk1100.h"
```

C:\asf-2.3.1\avr32\drivers\pm\power\_clocks\_lib.c, power\_clocks\_lib.h, pm.c, pm.h und pm\_conf\_clocks.c

C:\asf-2.3.1\avr32\drivers\gpio\gpio.c und gpio.h

C:\asf-2.3.1\avr32\drivers\spi\spi.c und spi.h

C:\asf-2.3.1\avr32\utils\debug\print\_funcs.c und print\_funcs.h debug.c und debug.h

C:\asf-2.3.1\common\components\memory\data\_flash\at45dbx\\_asf\_v1\at45dbx.c und at45dbx.h

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für ATEVK1100 in Ihrem Projektordner → Debug liegen.

- **das Programm erweitern**

Falls die Programmierung mit dem USB Bootloader erfolgt, muss das Programm ergänzt werden. In der Dokumentation "Erster Schritt" Kapitel 6.1. Erweiterung des Programms erfahren Sie mehr über diesen Schritt.

- **Dataflash Projekt an das AVR32 Board UC3AVRBIT anpassen**

laden Sie diese Bibliotheken runter und integrieren sie in das Projekt

<http://alvidi.de/lib/uc3avrbit.h>

[http://alvidi.de/lib/led\\_func.c](http://alvidi.de/lib/led_func.c)

[http://alvidi.de/lib/led\\_func.h](http://alvidi.de/lib/led_func.h)

In der Bibliothek "*board.h*" ändern Sie die Zeilen

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support

#define BOARD                EVK1100
```

wie folgt

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support
#define UC3AVRBIT           27  //!< UC3AVRBIT board with AT32UC3A0512

#define BOARD                UC3AVRBIT
```

und ersetzen Sie

```
#if BOARD == EVK1100
    #include "evk1100.h"
```

durch

```
#if BOARD == UC3AVRBIT
    #include "uc3avrbit.h"
#elif BOARD == EVK1100
    #include "evk1100.h"
```

ergänzen Sie in der Bibliothek *"print\_funcs.h"* die Zeile

```
#if BOARD == EVK1100
```

wie folgt

```
#if BOARD == UC3AVRBIT
# define DBG_USART (&AVR32_USART0)
# define DBG_USART_RX_PIN AVR32_USART0_RXD_0_0_PIN
# define DBG_USART_RX_FUNCTION AVR32_USART0_RXD_0_0_FUNCTION
# define DBG_USART_TX_PIN AVR32_USART0_TXD_0_0_PIN
# define DBG_USART_TX_FUNCTION AVR32_USART0_TXD_0_0_FUNCTION
# define DBG_USART_BAUDRATE 57600
#elif BOARD == EVK1100
```

**Wichtig!**

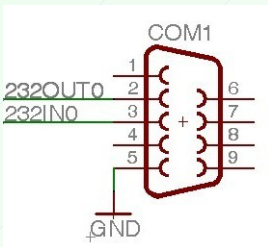
**Entfernen** Sie aus diesem Projekt ATEVK1100 Files: **evk1100.h**, **led.c** und **led.h**

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für UC3AVRBIT in Ihrem Projektordner → Debug liegen.

- das Dataflash Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen

#### 1.Schritt

Schließen Sie an das AVR32 Board die serielle Schnittstelle an.  
232OUT0 = OUT1, 232IN0 = IN1, GND =GND



#### 2.Schritt

Starten Sie z.B. HyperTerminal

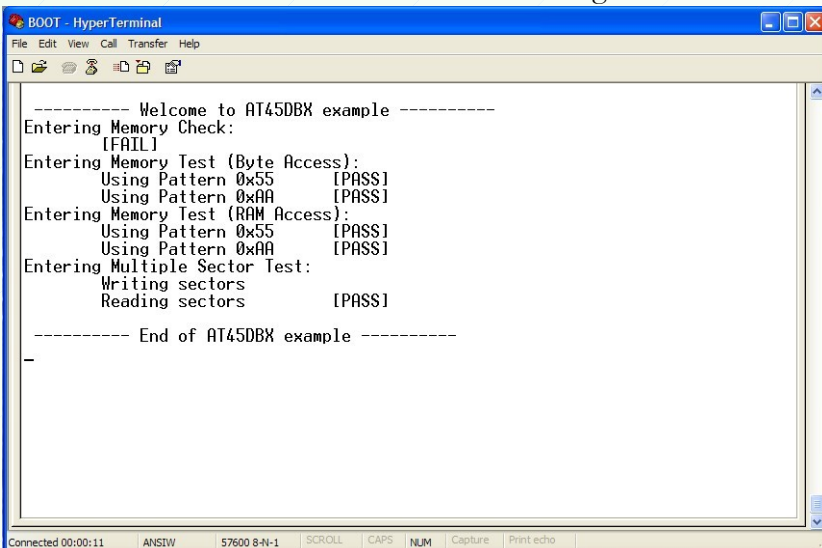
Starten → Programmen → Zubehör → Kommunikation → Hyperterminal

#### 3.Schritt

Gehen Sie auf Datei → Eigenschaften → Konfigurieren **57600, 8, keine, 1, keine**

#### 4.Schritt

Klicken Sie in oberer Leiste des HyperTerminal Anrufen → Verbinden. Sie erhalten die untere Abbildung.



Entering Memory Check: [FAIL]

– bedeutet, dass der Speicher falsch konfiguriert wurde. Diese Einstellung können Sie in

„conf\_at45dbx.h“ in der Zeile

```
#define AT45DBX_MEM_SIZE AT45DBX_8MB
```

ändern. Somit gilt:

```
AT45DB161D-SU = AT45DBX_2MB
```

```
AT45DB321D-SU = AT45DBX_4MB
```

```
AT45DB642D-CNU = AT45DBX_8MB
```

## 2.5. EEPROM

Um EEPROM auf dem AVR32 Board zu testen, laden Sie das Beispielprogramm „*eeprom\_example.c*“ von unserer Homepage [http://alvidi.de/lib/eeprom\\_example.c](http://alvidi.de/lib/eeprom_example.c) runter. Zum fehlerfreien Compilieren werden wir zuerst alle Bibliotheken und Treiber für das Atmel Produkt ATEVK1100 in die Testsoftware integrieren. Anschließend ersetzen bzw. ergänzen wir die vorhandenen Bibliotheken und Treiber mit dem Quellcode für das Board AL-UC3AVRBIT. Unser Ablauf sieht folgendermaßen aus:

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen (siehe Seite 10 „Erster Schritt“)
- das Programm erweitern (siehe Seite 15 „Erster Schritt“)
- EEPROM Projekt an das AVR32 Board UC3AVRBIT anpassen
- das EEPROM Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen
- ein fertig kompiliertes File für EEPROM Projekt finden Sie hier:
- <http://www.alvidi.de/prog/eeprom.elf> (10,2 MB) oder
- <http://www.alvidi.de/prog/eeprom.zip> (1,62 MB)

- neues Projekt anlegen (siehe Seite 8 „Erster Schritt“)
- Bibliotheken und Treiber hinzufügen

[http://alvidi.de/lib/eeprom\\_example.c](http://alvidi.de/lib/eeprom_example.c)

<http://alvidi.de/lib/eeprom.h>

<http://alvidi.de/lib/eeprom.c>

C:\asf-2.3.1\avr32\utils\[compiler.h](#) und [parts.h](#)

C:\asf-2.3.1\avr32\utils\preprocessor\[preprocessor.h](#), [mrepeat.h](#), [stringz.h](#) und [tpaste.h](#)

C:\asf-2.3.1\common\utils\[interrupt.h](#)

C:\asf-2.3.1\common\utils\interrupt\[interrupt\\_avr32.h](#)

erstzen Sie in "*interrupt.h*"

```
#include <parts.h>
```

durch

```
#include "parts.h"
```

und

```
#include "interrupt/interrupt_avr32.h"
```

durch

```
#include "interrupt_avr32.h"
```

ersetzen Sie in *"interrupt\_avr32.h"*

```
#include <compiler.h>
#include <preprocessor/tpaste.h>
```

durch

```
#include "compiler.h"
#include "tpaste.h"
```

und

```
#include <intc.h>
```

durch

```
#include "intc.h"
```

C:\asf-2.3.1\avr32\drivers\intc\exception.S, intc.c und intc.h

C:\asf-2.3.1\avr32\drivers\usart\usart.c und usart.h

C:\asf-2.3.1\avr32\boards\evk1100\evk1100.h, led.h und led.c

C:\asf-2.3.1\common\boards\board.h

ergänzen Sie *"board.h"* unter

```
#define UC3_LO_QT600          26
#define USER_BOARD          99
#define DUMMY_BOARD         100
```

mit dieser Zeile

```
#define BOARD                 EVK1100
```

ersetzen Sie in diesem File die Zeile

```
#include "evk1100/evk1100.h"
```

durch

```
#include "evk1100.h"
```

C:\asf-2.3.1\avr32\drivers\pm\power\_clocks\_lib.c, power\_clocks\_lib.h, pm.c und pm.h pm

C:\asf-2.3.1\avr32\drivers\gpio\gpio.c und gpio.h

C:\asf-2.3.1\avr32\utils\debug\print\_funcs.c und print\_funcs.h debug.c und debug.h

C:\asf-2.3.1\avr32\drivers\twi\twi.c und twi.h

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für ATEVK1100 in Ihrem Projektordner → Debug liegen.

- das Programm erweitern

Falls die Programmierung mit dem USB Bootloader erfolgt, muss das Programm ergänzt werden. In der Dokumentation "Erster Schritt" Kapitel 6.1. Erweiterung des Programms erfahren Sie mehr über diesen Schritt.

- EEPROM Projekt an das AVR32 Board UC3AVRBIT anpassen

laden Sie diese Bibliotheken runter und integrieren sie in das Projekt

<http://alvidi.de/lib/uc3avrbit.h>

[http://alvidi.de/lib/led\\_func.c](http://alvidi.de/lib/led_func.c)

[http://alvidi.de/lib/led\\_func.h](http://alvidi.de/lib/led_func.h)

In der Bibliothek "*board.h*" ändern Sie die Zeilen

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support

#define BOARD                 EVK1100
```

wie folgt

```
#define USER_BOARD          99  //!< User-reserved board (if any).
#define DUMMY_BOARD         100  //!< Dummy board to support
#define UC3AVRBIT           27  //!< UC3AVRBIT board with AT32UC3A0512

#define BOARD                UC3AVRBIT
```

und ersetzen Sie

```
#if BOARD == EVK1100
    #include "evk1100.h"
```

durch

```
#if BOARD == UC3AVRBIT
    #include "uc3avrbit.h"
#elif BOARD == EVK1100
    #include "evk1100.h"
```

ergänzen Sie in der Bibliothek "*print\_funcs.h*" die Zeile

```
#if BOARD == EVK1100
```

wie folgt

```
#if BOARD == UC3AVRBIT
#   define DBG_USART                (&AVR32_USART0)
#   define DBG_USART_RX_PIN         AVR32_USART0_RXD_0_0_PIN
#   define DBG_USART_RX_FUNCTION   AVR32_USART0_RXD_0_0_FUNCTION
#   define DBG_USART_TX_PIN         AVR32_USART0_TXD_0_0_PIN
#   define DBG_USART_TX_FUNCTION   AVR32_USART0_TXD_0_0_FUNCTION
#   define DBG_USART_BAUDRATE       57600
#elif BOARD == EVK1100
```

### Wichtig!

Entfernen Sie aus diesem Projekt ATEVK1100 Files: [evk1100.h](#), [led.c](#) und [led.h](#)

Wenn alles richtig gemacht wurde, sollte nach dem Kompilieren ein fertiges elf-File für UC3AVRBIT in Ihrem Projektordner → Debug liegen.

- das EEPROM Programm einspielen (siehe Seite 18 & 19 „Erster Schritt“)
- eingespieltes Programm testen

#### 1.Schritt

Schließen Sie an das AVR32 Board die serielle Schnittstelle an.  
232OUT0 = OUT1, 232IN0 = IN1, GND =GND

#### 2.Schritt

Starten Sie z.B. HyperTerminal

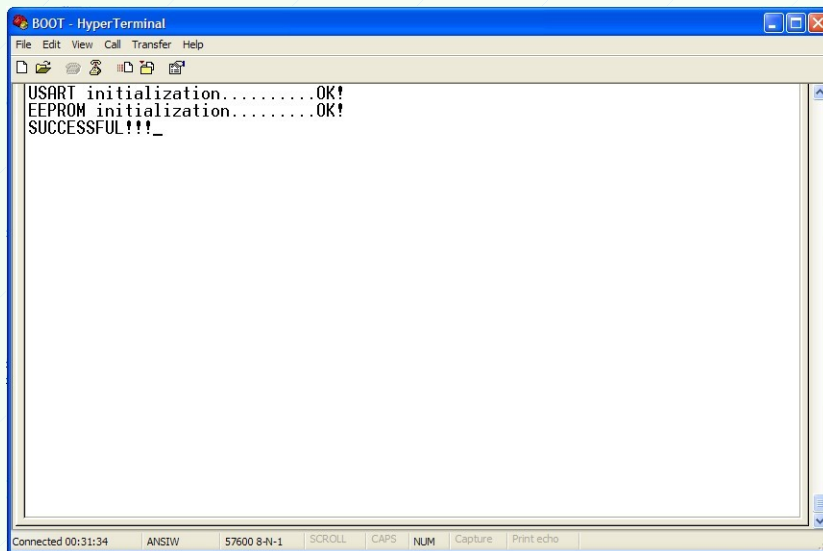
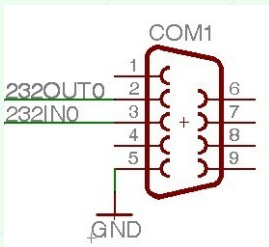
Starten → Programmen → Zubehör → Kommunikation → Hyperterminal

#### 3.Schritt

Gehen Sie auf Datei → Eigenschaften → Konfigurieren 57600, 8, keine, 1, keine

#### 4.Schritt

Klicken Sie in oberer Leiste des HyperTerminal Anrufen → Verbinden. Sie erhalten die untere Abbildung.



## 3. Alle Links im Überblick

Auf der nächste Seite sind alle Bibliotheken, die in diesem Dokument zum Einsatz gekommen sind, in alphabetischer Reihenfolge mit den entsprechenden Links aufgelistet.

File	Link
at45dbx_example.c	C:\asf-2.3.1\common\components\memory\data_flash\at45dbx\_asf_v1\example\
at45dbx.c	C:\asf-2.3.1\common\components\memory\data_flash\at45dbx\_asf_v1\
at45dbx.h	C:\asf-2.3.1\common\components\memory\data_flash\at45dbx\_asf_v1\
board.h	C:\asf-2.3.1\common\boards\
compiler.h	C:\asf-2.3.1\avr32\utils\
conf_access.h	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\example\conf\
conf_access.h	C:\asf-2.3.1\common\components\memory\data_flash\at45dbx\_asf_v1\example\
conf_at45dbx.h	C:\asf-2.3.1\common\components\memory\data_flash\at45dbx\_asf_v1\example\
conf_eth.h	C:\asf-2.3.1\avr32\drivers\macb\example\at32uc3a0512_evk1100\
conf_sd_mmc_spi.h	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\example\conf\
cycle_counter.h	C:\asf-2.3.1\avr32\drivers\cpu\cycle_counter\
debug.c	C:\asf-2.3.1\avr32\utils\debug\
debug.h	C:\asf-2.3.1\avr32\utils\debug\
delay.c	C:\asf-2.5.1\avr32\services\delay\
delay.h	C:\asf-2.5.1\avr32\services\delay\
dp83848 (folder)	C:\asf-2.3.1\avr32\components\ethernet_phy\
dummy_phy (folder)	C:\asf-2.3.1\avr32\components\ethernet_phy\
dummy.h (SPI example)	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\example\
eeprom_example.c	<a href="http://alvidi.de/lib/eeprom_example.c">http://alvidi.de/lib/eeprom_example.c</a>
eeprom.c	<a href="http://alvidi.de/lib/eeprom.c">http://alvidi.de/lib/eeprom.c</a>
eeprom.h	<a href="http://alvidi.de/lib/eeprom.h">http://alvidi.de/lib/eeprom.h</a>
ethernet_phy.h	C:\asf-2.3.1\avr32\components\ethernet_phy\
evk1100.h	C:\asf-2.3.1\avr32\boards\evk1100\
exception.S	C:\asf-2.3.1\avr32\drivers\intc\
flashc.c	C:\asf-2.3.1\avr32\drivers\flashc\
flashc.h	C:\asf-2.3.1\avr32\drivers\flashc\
gpio.c	C:\asf-2.3.1\avr32\drivers\gpio\
gpio.h	C:\asf-2.3.1\avr32\drivers\gpio\
intc.c	C:\asf-2.3.1\avr32\drivers\intc\
intc.h	C:\asf-2.3.1\avr32\drivers\intc\
interrupt_avr32.h	C:\asf-2.3.1\common\utils\interrupt\
interrupt.h	C:\asf-2.3.1\common\utils\
ksz8041tl.h	<a href="http://alvidi.de/lib/ksz8041tl.h">http://alvidi.de/lib/ksz8041tl.h</a>
led_func.c	<a href="http://alvidi.de/lib/led_func.c">http://alvidi.de/lib/led_func.c</a>
led_func.h	<a href="http://alvidi.de/lib/led_func.h">http://alvidi.de/lib/led_func.h</a>
led.c	C:\asf-2.3.1\avr32\boards\evk1100\
led.h	C:\asf-2.3.1\avr32\boards\evk1100\
macb_example.c	C:\asf-2.3.1\avr32\drivers\macb\example\
macb.c	C:\asf-2.3.1\avr32\drivers\macb\
macb.h	C:\asf-2.3.1\avr32\drivers\macb\
mrepeat.h	C:\asf-2.3.1\avr32\utils\preprocessor\
mt48lc16m16a2tg7e.h	C:\asf-2.3.1\avr32\components\memory\sdram\mt48lc16m16a2tg7e\
parts.h	C:\asf-2.3.1\avr32\utils\
pdca.c	C:\asf-2.3.1\avr32\drivers\pdca\
pdca.h	C:\asf-2.3.1\avr32\drivers\pdca\
pm_conf_clocks.c	C:\asf-2.3.1\avr32\drivers\pm\
pm.c	C:\asf-2.3.1\avr32\drivers\pm\
pm.h	C:\asf-2.3.1\avr32\drivers\pm\
power_clocks_lib.c	C:\asf-2.3.1\avr32\drivers\pm\
power_clocks_lib.h	C:\asf-2.3.1\avr32\drivers\pm\
preprocessor.h	C:\asf-2.3.1\avr32\utils\preprocessor\
print_funcs.c	C:\asf-2.3.1\avr32\utils\debug\
print_funcs.h	C:\asf-2.3.1\avr32\utils\debug\
sd_mmc_spi_example.c	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\example\
sd_mmc_spi.c	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\
sd_mmc_spi.h	C:\asf-2.3.1\avr32\components\memory\sd_mmc\sd_mmc_spi\
sdramc_example.c	C:\asf-2.3.1\avr32\components\memory\sdram\mt48lc16m16a2tg7e\example\
sdramc.c	C:\asf-2.3.1\avr32\drivers\ebi\sdramc\
sdramc.h	C:\asf-2.3.1\avr32\drivers\ebi\sdramc\
spi.c	C:\asf-2.3.1\avr32\drivers\spi\
spi.h	C:\asf-2.3.1\avr32\drivers\spi\
stringz.h	C:\asf-2.3.1\avr32\utils\preprocessor\
tpaste.h	C:\asf-2.3.1\avr32\utils\preprocessor\
twi.c	C:\asf-2.3.1\avr32\drivers\twi\
twi.h	C:\asf-2.3.1\avr32\drivers\twi\
uc3avrbit.h	<a href="http://alvidi.de/lib/uc3avrbit.h">http://alvidi.de/lib/uc3avrbit.h</a>
usart.c	C:\asf-2.3.1\avr32\drivers\usart\
usart.h	C:\asf-2.3.1\avr32\drivers\usart\

## **3. Disclaimer**

Wir haften nicht für die Eignung der Software für einen bestimmten, vom Kunden beabsichtigten Verwendungszweck.

Wir übernehmen keine Haftung für die Fehler, die im Laufe der Nutzung auftreten können.

Wir übernehmen keine Verantwortung für mögliche Schäden, die bei der Nutzung des Programms entstehen können.