

Teil 201 - Experimentierboards

1 Experimentierboards zum Testen und Programmieren von AVR-Mikrocontroller

- 1.1 Mit welchen Mitteln AVR-Mikrocontroller programmiert werden
- 1.2 Starterkit STK500
- 1.3 Entwicklungs-Tool AVR Dragon
- 1.4 ATM18-Controllermodul und ATM18-Testboard
- 1.5 AVR-ALE-Testboard

Teil 202 - ISP-Programmieradapter

2 ISP-Programmieradapter

- 2.1 ISP Bezogen auf die verschiedenen Schnittstellen
 - 2.1.1 Serielle Schnittstelle
 - 2.1.2 Parallele Schnittstelle
 - 2.1.3 USB-Schnittstelle
- 2.2 CC2-AVR-Programmer alias USBprog
 - 2.2.1 Aufbau
 - 2.2.2 Arbeitsweise
 - 2.2.3 Firmware-Änderung

Teil 203 - AVR-ALE-Testboard

- 3 Beschreibung des AVR-ALE-Testboard
 - 3.1 Schaltungsaufbau
 - 3.2 Stromversorgung
 - 3.3 Einsatz verschiedener AVR-Mikrocontroller
 - 3.4 Anzahl LEDs und Tasten
 - 3.5 LCD-Interface und 20x4-LCD
 - 3.5.1 Erzeugung des Enable-Signals für das LCD
 - 3.5.2 LCD-Backlight
 - 3.6 Ansteuerung von Relais
 - 3.7 RS-232-Schnittstelle
 - 3.8 USART-Testboard-Schnittstelle

Teil 204 - AVR Studio

4 Einsatz des AVR Studio

- 4.1 AVR Studio installieren
- 4.2 Testboard und Programmer zusammenschalten
 - 4.2.1 Treiber AVRISP mkII neu installieren
- 4.3 Starten von AVR Studio
 - 4.3.1 AVR Studio und CC2-AVR-Programmer
 - 4.3.2 Mikrocontroller-Einstellungen im AVR Studio

Teil 205 - Assembler und AVR Studio

5 Assembler und AVR Studio

- 5.1 Der Übersetzer (Assembler)
- 5.2 Ein neues Projekt erzeugen
 - 5.2.1 Der Projekt-Bereich
 - 5.2.2 Bearbeiten der Assemblerdatei
 - 5.2.3 Assemblieren des Quell-Codes
- 5.3 Simulation des Codes
 - 5.3.1 Programmausführung im Einzelschrittverfahren
 - 5.3.2 Debugger-Stopp-Punkte
- 5.4 Verändern des Programmtextes



5.4.1 Überwachen von Variablen

5.4.2 Anzeigen der Prozessordetails

5.4.3 Speichern des Projekts

5.5 Erzeugen eines weiteren ASM-Projektes im Schnelldurchgang

5.6 Flashen eines ASM-Programms in ein Mikrocontroller ATmega88

Teil 206 - C-Compiler und AVR Studio

<u>6 CodeVisionAVR C-Compiler und AVR Studio</u> <u>6.1 CodeVisionAVR C-Compiler installieren</u> <u>6.2 Erzeugen eines C-Projektes</u> <u>6.2.1 Ein neues Projekt beginnen</u> <u>6.2.2 Ein C-Projekt konfigurieren</u> <u>6.2.3 Arbeitsschritte zur Generierung eines C-Projektes</u> <u>6.3 Einbinden von AVR Studio in den CVAVR</u> <u>6.4 AVR Studio Debugger für CVAVR</u> <u>6.5 Flashen eines C-Programms in ein Mikrocontroller ATmega88</u>

Teil 207 - Editor - UltraEdit

7 Editor - UltraEdit

7.1 Kopf- und Fuß-Zeile

7.1.1 Einstellungen für Assembler-Programme

7.1.2 Einstellungen für C-Compiler-Programme

7.2. Syntaxhervorhebung (Syntax Highlighting)

7.2.1 AVR-Assembler

7.2.1.1 Syntaxbefehle für den AVR-Assembler

7.2.1.2 Farben und Schriftstile der Gruppen für den AVR-Assembler

7.2.2 CodeVisionAVR C-Compiler

7.2.2.1 Syntaxbefehle für den CVAVR C-Compiler

7.2.2.2 Farben und Schriftstile der Gruppen für den CVAVR C-Compiler

7.3 Wortsammlung für AVR-Assembler

7.4 Wortsammlung für CodeVisionAVR C-Compiler

Hinweis

Externe Anschaltungen und Hardware-Erweiterungen werden in der **Gruppe 400 - ASM-Projekte** und in der **Gruppe 600 - C-Projekte** detailliert beschrieben.

6 CodeVisionAVR C-Compiler und AVR Studio

Anmerkungen:

1. Dieser **Teil 206** setzt eigentlich Programmierkenntnisse in der Programmiersprache **C** zur Erzeugung lauffähiger Programme mit dem **CodeVisionAVR C-Compiler** voraus. Da hier allerdings nur die Handhabung von **AVR Studio** und das in der **Gruppe 500** detailliert behandelten CodeVisionAVR C-Compilers beschrieben werden soll, werden die Schritte anhand eines fertigen kleinen Programms erläutert.

2. Von der Seite http://lehrer.schule.at/Doblinger/files/AVR Studio 4 Tutorial.pdf (die leider nicht mehr auffindbar ist) sind in diesem Abschnitt einige Anregungen entnommen worden.

6.1 CodeVisionAVR C-Compiler installieren V2.60

Die Installation der Compiler-Version aus dem Download-Center vom <u>www.cczwei.de</u> des ATM18-Projektes ist nicht unter **Windows 7** bzw. **Windows 10** lauffähig!

Auch ein neuer Versuch mit **CodeVisionAVR Evaluation V2.05.7a** von der **HP Info Tech-Homepage** http://www.hpinfotech.ro/html/download.htm (**nicht mehr existent**) führte leider nicht zum vollen Erfolg, da die Beispielprogramme unter den Einschränkungen der Demoversion nicht lauffähig sind.

Ein "Bettelbrief" an den Hersteller HP Info Tech mit Hinweis auf das ATM18-Projekt

http://www.cczwei-forum.de/cc2/thread.php?threadid=1797

hat dem Verfasser dann eine verbilligte Vollversion V2.60 eingebracht.

ANMERKUNG (Stand 27.07.2017): In der Zwischenzeit sind einige spätere Versionen erschienen, die hier NICHT behandelt werden! Die Versionen ab V3.0 würden eine vollständige Überarbeitung der Gruppe 200 - Einsetzen von AVR-Tools notwendig machen (z.B. AVR-Studio).

TROST: Für den **ATMega88** funktioniert die Programmierung auch weiterhin vorzüglich. Inzwischen wurde die Version unter **Windows 10** installiert und läuft auch dort bisher einwandfrei.

Nach dem Download steht die Datei **setup.exe** zur Verfügung. Bei der Installation wird man sogar gefragt, ob man Deutsch als Sprache wünscht. Allerdings wird der Wunsch später nicht mehr erfüllt!



Bild 6.1-01: Begrüßungs-Fenster vom CodeVisionAVR C-Compiler

Tools_206

🕵 Setup - CodeVisionAVR C Compiler	
Lizenzvereinbarung Lesen Sie bitte folgende, wichtige Informationen bevor Sie fortfahren.	
Lesen Sie bitte die folgenden Lizenzvereinbarungen. Benutzen Sie bei Bedarf die Bildlaufleiste oder drücken Sie die "Bild Ab"-Taste.	
SOFTWARE LICENCE	
The use of CodeVisionAVR indicates your understanding and acceptance of the following terms and conditions. This license shall supersede any verbal or prior verbal or written, statement or agreement to the contrary. If you do not understand or accept these terms, or your local regulations prohibit "after sale" license agreements or limited disclaimers, you must cease and desist using this product immediately.	
This product is (C) Copyright 1998-2012 Pavel Haiduc and HP InfoTech s.r.l., all rights reserved.	~
Ich akzeptiere die Vereinbarung Ich lehne die Vereinbarung ab	
<zurück weiter=""></zurück>	Abbrechen

Bild 6.1-02: Lizenzvereinbarung akzeptieren

Die positive Antwort auf die obligatorische Anfrage nach dem Akzeptieren der Lizenzbestimmungen wird dann die Installation eingeleitet:

🔀 Setup - CodeVisionAVR C Compiler	_ 🗆 🗙
Passwort Diese Installation wird durch ein Passwort geschützt.	
Bitte geben Sie das Passwort ein, und klicken Sie danach auf "Weiter". Achten Sie auf ko Groß-/Kleinschreibung.	orrekte
Passwort	
<zurück weiter=""></zurück>	Abbrechen

Bild 6.1-03: Passwort eingeben

Mit jeder neuen Programmversion ist ein neues Passwort beim Installieren einzugeben. Das Passwort und die Download-Adresse erhält man per E-Mail vom Hersteller **HP Info Tech s.r.l.**

Aus verständlichen Gründen kann das Setup-Passwort und die Lizenz-Nummer, welche man beim Kauf erhält, hier nicht wiedergegeben werden, so dass das entsprechende Menü "ausgepunktet" dargestellt wird.

🔂 Setup - CodeVisionAVR C Compiler	_ 🗆 🗙
Ziel-Ordner wählen Wohin soll CodeVisionAVR C Compiler installiert werden?	
Das Setup wird CodeVisionAVR C Compiler in den folgenden Ordner installieren. Klicken Sie auf "Weiter", um fortzufahren. Klicken Sie auf "Durchsuchen", falls Sie einen anderen Ordner auswählen möchten.	
CACVAVI2 Durchsuchen	
Mindestens 76,8 MB freier Speicherplatz ist erforderlich.	
<zurück weiter=""> Abbr</zurück>	echen

Bild 6.1-04: Installations-Ordner (sollte nicht verändert werden)

Der Ordner c:\cvavr2 ist sehr wichtig und sollte nicht verschoben werden. Denn an diesem Ort werden auch die Bibliotheken abgespeichert. Sollte der Ordner noch nicht vorhanden sein, so fragt das Installationsprogramm, ob es den Ordner erstellen soll.

] Setup - CodeVisionAVR C Compiler	
Startmenü-Ordner auswählen Wo soll das Setup die Programm-Verknüpfungen erstellen?	
Das Setup wird die Programm-Verknüpfungen im folgenden Startmenü-Ordner erstellen.	
Klicken Sie auf "Weiter", um fortzufahren. Klicken Sie auf "Durchsuchen", falls Sie einen anderen Ordner auswählen möchten.	
CodeVisionAVR Durchsuc	hen
< Zurück Weiter >	Abbrechen

Bild 6.1-05: Startmenü-Ordner

Auch der Ordner für das Startmenü CodeVisionAVR sollte auf seinem vorgesehenen Platz belassen werden. Man findet die notwendigen Dateien schneller wieder (zum Beispiel):

C:\ProgramData\Microsoft\Windows\Start Menu\Programs\CodeVisionAVR

🔀 Setup - CodeVisionAVR C Compiler	_ 🗆 🗙
Installation durchführen Das Setup ist jetzt bereit, CodeVisionAVR C Compiler auf Ihrem Computer zu installieren.	
Klicken Sie auf "Installieren", um mit der Installation zu beginnen, oder auf "Zurück", um Ihre Einstellungen zu überprüfen oder zu ändern.	
Ziel-Ordner: c:\cvavr2	<u> </u>
Startmenü-Ordner: CodeVisionAVR	
	•
< Zurück 🔏 Installieren Ab	brechen

Bild 6.1-06: Installation durchführen

🔂 Setup - CodeVisionAVR C Compiler
Installiere Warten Sie bitte während CodeVisionAVR C Compiler auf Ihrem Computer installiert wird.
Dateien werden entpackt c:\\examples\USB\AVR270 USB Mouse\series6-hidmouse-2_0_6-doc\doc\html\a00058.html
Abbrechen

Bild 6.1-07: Installation läuft . . .

🔀 Setup - CodeVisionAVR C Compiler	
Information Lesen Sie bitte folgende, wichtige Informationen bevor Sie fortfahren.	
Klicken Sie auf "Weiter", sobald Sie bereit sind mit dem Setup fortzufahren.	
Welcome to the CodeVisionAVR C Compiler V2 and the CodeWizardAVR Automatic Program Generator.	
The compiler is designed to be used along with the AVR Studio V4.19, V5.1 or Atmel Studio 6.0 or later debuggers, which can be downloaded from the ATMEL web site: www.atmel.com	
For JTAGICE 3 programming support to be available, AVR Studio 5.1 or Atmel Studio 6 or later must be installed.	
Details about using CodeVisionAVR with these debuggers are available in the Help topics: - CodeVisionAVR IDE Tools The AVR Studio Debugger - CodeVisionAVR C Compiler Reference Using the AVR Studio 4.19 Debugger - CodeVisionAVR C Compiler Reference Using the AVR Studio 5.1 and Atmel Studio 6	•
Weiter >	

Bild 6.1-08: Setup - Wichtige Informationen

Text siehe weiter unten!



Bild 6.1-09: Vor der Fertigstellung wird noch nach der 8-stelligen License ID zum Aktivieren gefragt. Sollte das Programm bereits auf einem anderen PC aktiviert worden sein, so ist die Aktivierung auf diesem anderen PC zu deaktivieren und auf dem neuen PC neu zu aktivieren.

Danach ist die Installation abgeschlossen

Die neue Version wird gleich gestartet:



Bild 6.1-10: Man beachte, dass die "Version 2.60 Standard" keine Evaluation (Testversion) ist

👫 CodeVisionAVR
<u>File Edit Search View Project Tools Settings Help</u>
林 共 共 時 , 10 12 12 12 10 1、 10 目目目 20 (小 秋 、 本 参 目 10 , 20 (小 秋 の , ?) (0 , 20)
🟗 Code Navigator 🔍 🌣 🖸 😼 Code Information 🖤 🐺 🖸 💱 Function Call Tree 😤 🖾 🔤 🖉 Code Te 🖤 🛱 🖬 🔂 🚱 Clip 🖤
CodeVisionAVR No Project Other Files If () { if () { <
Hessages
<u> </u>

Bild 6.1-11: Erstes Bild nach der Installation (Bildvergrößerung)

Damit ist <u>CVAVR</u> installiert . . .

... und der erste Aufruf sollte ungefähr wie im **Bild 6.1-11** aussehen. Bei späteren Aufrufen, wenn schon mit dem CVAVR "gespielt" worden sein sollte, "erinnert" sich das Tool an die letzte Einstellung und gibt diese wieder. Wenn man nicht gerade an einem größeren Projekt über einen ausgedehnten Zeitraum arbeitet, bei dem diese Verfahrensweise sehr vorteilhaft ist, so wird man sicherlich bei einem neuen Projekt beginnen wollen. Das Eröffnungsfenster erhält man nach erstmaligem Aufruf oder allemal nach **File => Close All** - es sind jedoch ggf. einzelne Spalten ausgeblendet.

Zum CVAVR ist auch schon viel geschrieben worden, so dass es fast müßig ist, dem noch etwas hinzufügen zu wollen. Aber: Bei der Installation gibt es noch allerlei Kleinigkeiten, auf die es sich lohnt

hinzuweisen. Der lange Hinweis bei der Installation (**Bild 6.1-08: Setup**) ist schnell weggeklickt. Er ist aber in einigen Punkten sehr aufschlussreich und deshalb als Original-Zitat hier zum Nachlesen wiederholt:

Welcome to the CodeVisionAVR C Compiler V2 and the CodeWizardAVR Automatic Program Generator.

The compiler is designed to be used along with the AVR Studio V4.19, V5.1 or Atmel Studio 6.0 or later debuggers, which can be downloaded from the ATMEL web site: www.atmel.com (man landet neuerdings bei: https://www.microchip.com/)

For JTAGICE 3 programming support to be available, AVR Studio 5.1 or Atmel Studio 6 or later must be installed.

Details about using CodeVisionAVR with these debuggers are available in the Help topics:

- CodeVisionAVR IDE|Tools|The AVR Studio Debugger
- CodeVisionAVR C Compiler Reference|Using the AVR Studio 4.19 Debugger
- CodeVisionAVR C Compiler Reference Using the AVR Studio 5.1 and Atmel Studio 6 Debuggers and in the chapters: 2.4.1, 3.21 and 3.22 of the CodeVisionAVR User Manual.

1. Installing CodeVisionAVR

If you will use the Compiler under Windows 2000, XP, Vista or Windows 7 you must first install and run it with Administrator privileges. On subsequent runs under Windows 2000 or XP you may also have Power User privileges.

After installing under Windows 7, right click on the CodeVisionAVR icon and select **Properties|Shortcut|Advanced** and check the **Run as Administrator** check box.

For installing under Windows Vista, the following steps should be performed, having an Administrator account:

- the Windows Vista User Account Control must be disabled using the steps below:

- press the Windows button and click on the Control Panel link
- in the Control Panel window, under the User Accounts and Family Safety, click on the Add or remove user accounts link
- a new window will open, click on the Go to the main User Accounts page link located at the bottom of the window
- a new window will open, click on the Turn User Account Control on or off link located at the bottom of the window
- in the new window that will open, uncheck the Use User Account Control (UAC) to help protect your computer check box and press the OK button to confirm
- the computer will have to be restarted for this setting to become effective

- run the CodeVisionAVR setup.exe installer

- after the installation is complete, right click on the CodeVisionAVR icon on the desktop
- select Properties in the popup menu that will open
- a new window called CodeVisionAVR Properties will open
- select the Compatibility tab
- check the Privilege Level|Run this program as an administrator check box and press the OK button to confirm.

2. New features/bug fixes in CodeVisionAVR V2.60

2.1 Compiler

- Added support for the Ilitek ILI9325 graphic TFT LCD controller (only for Advanced license).
- Improved the library for the Solomon Systech SSD1289 graphic TFT LCD controller.
- Modified the alphanumeric LCD library (alcd.h) to be compatible with the Novatech NT3881 controller.

2.2 CodeWizardAVR

- Added the possibility to automatically set all the timer configuration registers for user specified operating mode, period and output duty cycle(s).
- Fixed: the Timer5 OC5A, OC5B, OC5C outputs were assigned to non-existent PORTM, instead of PORTL, for the ATmega640/1280/2560 chips.
- Fixed: the OC4A, OC4B, OC4C OC5A, OC5B, OC5C outputs are not present for the ATmega1281/2561 chips.
- Fixed: the ADTS0..3 bits in the SFIOR or ADCSRB registers were not set correctly, when using one of the
- ADC auto-trigger modes.
- Fixed: the ADC auto-trigger modes were missing for the ATmega32 chip.
- Fixed: USART code generated for the ATmega64 chip, used UMSEL00..UMSEL11 bit names instead of UMSEL0 and UMSEL1 in the UCSR0C and UCSR1C registers.
- Fixed: ADC code generated for the ATmega64 chip, used the ADATE bit name, which is not present in the ADCSRA register of this chip.

2.3 Chip Programmer

- Added check for conflicts between the 'EEPROM|Program' and 'Preserve EEPROM' options.
- Fixed error for restoring EEPROM contents after chip erase, when the 'Preserve EEPROM' option was enabled and

╆ Tools_206

Atmel Studio 6 was set as debugger.

3. New features/changes in CodeVisionAVR V2.60, not available in old V1.25.x compilers

3.1 Compiler

- ANSI C compatible C front-end
- the bool data type was added (stdbool.h)
- the @ operator can now be used with variables located in EEPROM too
- the & unary operator can now be applied to I/O registers declared using the sfrb and sfrw keywords.
- It will return a pointer to the RAM address where these registers are mapped.
- there is no need to specify the 'flash' or 'eeprom' memory attribute during structure or union type definition. Therefore the same structure or union data type can be easily located in any memory area during variable declaration.
- the compiler now makes distinction between the 'const' type qualifier and the 'flash' memory attribute. In order to maintain compatibility with V1.25.x projects, the Project|Configure|C Compiler|Code Generation|Store Global Constants in FLASH Memory must be checked. If this option is not checked, identifiers declared with the 'const' type qualifier will be placed in RAM.
- the preprocessor does not use the old 'funcused' directive, when it is found it is always evaluated to 1
- added the #message preprocessor directive
- the abs, cabs, labs and fabs functions were moved from the math.h header to the stdlib.h header
- improved error and warning checking
- improved linker: only the modified C source files are compiled when the Project|Build command is executed
- improved COFF object file generator
- improved code optimizer
- enhanced libraries, including MMC/SD/SD HC FLASH Memory Card and FAT support
- I/O registers bits definitions were added to the device header files.
- For projects created with prior versions, these definitions are not enabled by default.

In order to enable them the Project/Configure/C Compiler/Code Generation/Preprocessor/Include I/O Registers Bits Definitions option must be activated.

- For newly created projects, this option is enabled by default.
- in order to eliminate naming conflicts with I/O registers bits definitions, the names of the SPCR0, SPDR0, SPSR0 registers were changed to SPCR, SPDR, SPSR in the following header files: mega164.h, mega324.h, mega644.h, mega644p.h.
- in order to eliminate naming conflicts with I/O registers bits definitions, the names of the **PCINT0**, **PCINT1**, **PCINT2**, **PCINT3** interrupt vectors were changed to **PC_INT0**, **PC_INT1**, **PC_INT2**, **PC_INT3** in the following header files: 90usb1286.h, 90usb1287.h, 90usb646.h, 90usb647.h, 90usb162.h, 90usb162.h, mega1280.h, mega1281.h, mega2560.h, mega2561.h, mega640.h, mega1284p.h, mega162.h, mega165.h, mega165.h, mega168.h, mega168.h, mega325p.h, mega3250.h, mega3250.h, mega3250.h, mega3250.h, mega3250.h, mega644p.h, mega644p.h, mega6445.h, mega6450.h, mega6490.h, mega6490.h, mega888.h, mega648.h, tiny10.h, tiny24.h, tiny25.h, tiny44.h, tiny45.h, tiny48.h, tiny84.h, tiny 85.h, tiny88.h, tiny261.h, tiny461.h, tiny861.h, tiny2313.h.
- If these interrupt vectors are used in your programs, their names must be updated.

- in order to eliminate naming conflicts with I/O registers bits definitions, the names of the **INT0...INT7** interrupt vectors were changed to **EXT_INT0...EXT_INT7** in the following header files: mega1280.h, mega1281.h, mega2560.h, mega2561.h, mega640.h, mega1284p.h, mega164.h, mega324.h, mega644.h, mega644p.h.

If these interrupt vectors are used in your programs, their names must be updated.

- removed the limitation: when a literal char string enclosed between quotes, is passed as an argument to a function parameter of pointer to char, the pointer now can point to any memory type: RAM, EEPROM or FLASH, not only FLASH like in previous versions. The CodeVisionAVR C Compiler Reference|Constants Help topic and User Manual chapter were updated to reflect this change.

3.2 IDE

- completely redesigned text Editor with improved syntax highlighting for both C and AVR assembly
- added function parameters and global structure/union members auto complete
- improved code folding
- code folding state can be saved and restored
- bookmarks state can be saved and restored
- added automatic matching brace highlighting when the user places the cursor before the brace
- added automatic file saving at specified time interval
- the Code Navigator was redesigned and displays project information in a more logical way
- an additional Code Information tab is present after a project build. It displays detailed information about the included header files, preprocessor macro definitions, type definitions, global/static variable declarations, memory allocation and function definitions.
- added the Functions Call Tree tab in the Navigator
- the new Project|Configure|Files|Output Directories allows to specify in which directories the files generated by the compiler/linker will be placed.
- improved toolbar

It is important to note that in CodeVisionAVR V2 the .asm, .hex, .eep, .cof files created by the compiler have the name of the project .prj file.

3.3 CodeWizardAVR

- Generates code using symbolic names for peripheral configuration registers' bits.

- Improved user interface.

╆ Tools_206

4. New ATxmega chips support in CodeVisionAVR

- the Standard C I/O Functions use by default the USARTC0. This can be changed by defining the
- _ATXMEGA_USART_ macro as described in the corresponding Help topic.
- the SPI functions use by default the SPIC controller on PORTC. This can be changed by defining the _ATXMEGA_SPI_ and ATXMEGA_SPI_ preprocessor macros as described in the corresponding Help topic.
- _ATXMEGA_SPI_PORT_ preprocessor macros as described in the corresponding Help topic.
- the RTC functions do not yet support the ATxmega chips.

5. Example programs

The Compiler is supplied with the following example programs:

- ATxmega chips in \EXAMPLES ATxmega
- Graphic LCDs in \EXAMPLES\Graphic LCD
- SD FLASH Memory Cards access in \EXAMPLES\SDCARD
- USB in \EXAMPLES\USB
- AVR 109 Bootloader
- AES Encryption-Decryption
- "ATmega8535 ADC on the STK500" in \EXAMPLES\ADC8535
- "Real Time Clock with ATmega103, Atmel Application Note AVR134" in \EXAMPLES\AVR134
- "Calling functions written in assembler from C" in \EXAMPLES\C_ASM
- "Moving LED" in \EXAMPLES\LED
- "Accessing the EEPROM" in \EXAMPLES\EEPROM
- "LCD Demo" in \EXAMPLES\LCDDEMO
- "Definition of user characters in LCDs" in \EXAMPLES\LCDCHAR
- "LM75 Thermometer" in \EXAMPLES\THERM75
- "LCD Thermometer" in \EXAMPLES\THERMLCD
- "DS1820/DS18S20 Multipoint thermometer" in \EXAMPLES\DS1820
- "DS1990 Serial Number iButton" in \EXAMPLES\DS1990
- "AD7896 Digital voltmeter" in \EXAMPLES\SPI
- "MAX1241 Digital voltmeter" in \EXAMPLES\MAX1241
- "4x4 Keypad" in \EXAMPLES\KEYPAD
- "Simple multifile project" in \EXAMPLES\MULTFILE
- "Redirecting the output of standard C I/O functions to USART0, USART1
- and LCD for an ATmega128 chip" in \EXAMPLES\2USART_LCD
- Tiny5 Level Meter
- uIP_Crumb644-Net_CVAVR web server
- TWI Master-Slave.

After starting CodeVisionAVR, execute the File|Open menu command and load the Project (*.prj) file for the example you like. Then execute the Project|Build All menu command to compile and link the project.

Take some time and carefully read the Help.

Auch C-Programme werden ausschließlich in Form von Projekten bearbeitet, so dass jetzt an die Bildung eines C-Projektes herangegangen werden soll.

6.2 Erzeugen eines C-Projektes

📕 L:\AVR_PB_LED							<u>l ×</u>
GOV IL AVR_PB_LED					💌 🛃 🛛 AVR_PI	3_LED durchsuchen	2
Datei Bearbeiten Ansicht Extras ?							
Organisieren 🔻 Freigeben für 🔻 Brenne	en Neuer	r Ordner				i – 🗍 (0
Icherungs-LW 1000 GB (I:)		Name 🔶	Größ	e	Änderungsdatum	Тур	
 Wechsel-LW2 160 GB (J:) ExtVideo-LW 1000 GB (K:) KINGSTON (L:) ALEs_ASM_Progs AVR_PB_LED Wechseldatenträger (M:) Netzwerk Systemsteuerung Papierkorb 		 Nicht angegeben (7) — application.c application.h iomx8.h macros.h main.c switches.h typedefs.h 	<i>A</i>	11 KB 36 KB 46 KB 4 KB 6 KB 4 KB 9 KB	15.08.2009 14:17 15.08.2009 21:34 15.08.2009 15:08 15.08.2009 20:07 15.08.2009 21:18 15.08.2009 20:09 15.08.2009 20:13	C compiler source file C compiler header file C compiler header file C compiler header file C compiler source file C compiler header file C compiler header file	
	•	•					
7 Elemente							
7 Elemente					減 Compu	ter	

Bild 6.2-01: Einen neuen Projekt-Ordner mit (vorhandenen) Quell-Dateien anlegen

Im Beispiel wird zur Demonstration einer Projekt-Generierung ein neuer Projekt-Ordner **AVR_PB_LED** auf einem USB-Stick KINGSTON (L) angelegt, in dem alle fertigen ***.c**- und ***.h**-Dateien des fertigen Projektes **AVR_PB_LED** übertragen werden.

Siehe: Gruppe 600 - AVR C-Projekte, Teil 601 - AVR_PB_LED

Einfache Beschaltung von LEDs und Tastern, der Quell-Code steht unter 601_Projekt_AVR_PB_LED

Der Ordner-Name wurde vom Namen des bestehenden Projektes auf der Festplatte entlehnt. Dieser Ordner und die folgenden Schritte dienen gleichermaßen zur Demonstration, wie man als "Neuling" fertig erstellte Projekte zum Testen benutzen kann. Wenn man ein neues Projekt beginnt, fängt man ja bei "Adam und Eva" an, d.h. man legt einen neuen Projekt-Ordner an, übernimmt **oder** erstellt darin die Quell-Dateien und marschiert dann durch die hier beschriebenen Instanzen.

6.2.1 Ein neues Projekt beginnen

Oben wurde das Compiler-Tool **CodeVisionAVR** installiert; jetzt wird das erste Projekt (wenn auch nur entlehnt) durch Start dieses Tools erzeugt:



Bild 6.2.1-01: CVAVR für ein neues Projekt starten



Bild 6.2.1-02: Anfrage, ob der CodeWizardAVR benutzt werden soll

Es soll der **CodeWizardAVR** nicht verwendet werden: **=> No**; es meldet sich das Fenster **Create New Project**. Dieses verweist ggf. zunächst auf den Pfad der CVAVR-Installation:

C:\cvavr2\bin

weil der Compiler-Hersteller offenbar davon ausgeht, dass alle Projekte hier konzentriert werden sollen. Da aber das Projekt in einem eigens dafür vorgesehenen Ordner angelegt werden soll, müssen hier neue Einstellungen vorgenommen werden.

Im Feld Speichern in: wird der neu eingerichtete Ordner gesucht.

Hier: AVR_PB_LED => Speichern



Bild 6.2.1-03: Create New Project

Bild 6.2.1-04: Ordner AVR_PB_LED suchen

Klick auf 본 und KINGSTON (L) wählen

AVR_PB_LED markieren => Öffnen

Jetzt muss noch der Dateiname für das **Project file (*.prj)** benannt werden. Sinnvoll ist es, den gleichen Namen wie für den Projekt-Ordner zu wählen - nämlich **AVR_PB_LED**. Nur diese Datei braucht später aufgerufen zu werden, um Änderungen im Projekt vorzunehmen. Sie soll wie folgt abgelegt werden (hier beispielhaft auf dem USB-Stick L:):

SK Create New Project	🕵 Create New Project	9 ×
Speichem in: 👔 AVR_PB_LED 💽 🌀 🏂 🛤	Speichem in: 📙 AVR_PB_LED 💽 😨 🤔 💌 🕶	
Name Image: State of the st	Name Größe Anderungsda Zuletzt besucht Es wurden keine Suchergebnisse gefunden. Desktop Bibliotheken Computer Netzwerk 	tum <u>-</u>
×		F
Dateiname: Speichern	Dateiname: AVR_PB_LED	sichem
Dateityp: Project files (*.prj)	Dateityp: Project files (*.prj)	rechen

L:\AVR_PB_LED\AVR_PB_LED.prj

Bild 6.2.1-05: Es wird eine *.prj-Datei angefordert

Bild 6.2.1-06: Benennen der *.prj-Datei

Es öffnet sich zusätzlich das Fenster **Configure Project AVR_PB_LED.prj** mit der Aufforderung, dem Projekt nun die Quell-Dateien hinzuzufügen (besser: bekannt zu machen, da sie sich ja bereits im Projekt-Ordner befinden).

Configure Project AVR_PB_LED.prj	×
Files Compiler Before Build After Build	1
Input Files Output Directories	
J:\AVR_PB_LED\AVR_PB_LED.prj	
Z Add	
±1 Berrow	
TI Edit File N	ame III
▲ WOAS Di	
	Add Ella to Project
	Suchen in: AVR_PB_LED V G f P
	Name A Größe - Änderungsdatum -
	Zuletzt besucht
	Unker 18.03.2012 23:47 Desktop List List 18.03.2012 23:47
	Obj 18.03.2012 23:47
	Bibliotheken main.c 6 KB 15.08.2009 22:18
	Computer
	Dateiname: Offinen
	Uatetyp: IC Compiler files (*.c)

Bild 6.2.1-07: Configure Project Part 1 - Input Files

Bild 6.2.1-08: Neue Projekt-Struktur

Im Hauptfenster von CVAVR sieht man bereits, dass auf der linken Seite im **Code Navigator** der Projekt-Name **AVR_PB_LED** und in der Kopfzeile der Name der Projekt-Datei **AVR_PB_LED.prj** eingetragen sind.

Add =>

CodeVisionAVR - L:\AVR_PB_LED\AVR_PB_LEStrj
File Edit Search View Project Tools Settings Help
▶ ▶ ♥ ■ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●
林 共 共 辞 , 101111220160160111111111111111111111111
e Ta Ta 😰 😵 👻 🗃 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖬 🖉
🗱 Code Navigator 🔰 🌣 🖉 😼 Code Information 👻 77 🖾 💱 Function Call Tree 👎 🎑 🛛 (LyAVR_PB_LED(AVR_PB]) 🚟 Code Te 🖤 77 🖾 🚱 Clip 🖤 7
B 💥 CodeVisionAVR
Project AVR_PB_LED
Get () J
do {

Bild 6.2.1-09: Hauptfenster von CVAVR

Jetzt müssen alle C-Dateien zum Kompilieren und Binden dem Projekt bekannt gemacht werden. Das Menü lässt automatisch nur C-Dateien zu und weist damit bereits darauf hin, dass nur die C-Dateien den anfänglichen Start bestimmen. Die Header-Dateien werden erst während der Kompilierung (genauer: während des Preprozessor-Laufs) aus den C-Dateien heraus aufgerufen und in das Coding eingefügt.

🛠 CodeVisionAVR - J:\AVR_PB_LED\AVR_PB_LED.prj		×
Eile Edit Search View Project Tools Settings Help	👫 Add File to Project 🔠	×
] 🗅 🗁 😌 • 🖬 🗑 🗇 🖄 🖄 🖄 💩 🎒	Suchen in: 🜗 AVR_PB_LED 💽 🌀 🍺 📂 🖽 -	<i>•</i> ∎ (++ ,
純 鸟 色 岬 慕 體 🗸 🕒 🗅 😫 😒 📎 🚺	Name A Größe V Änderungsdatum	
🖶 😘 🕞 1 🗽 😽 😴 🕾 🖬 🗖 🛤 🔲 🗊 🗊	Nicht angegeben (6)	Ē
🐮: Code Navigator 🛛 💝 🏹 🌄 Code Informa 🔇	Zuletzt besucht Exe 18.03.2012 23:47	Cli 💙 🐺 🔀
CodeVisionAVR	Linker 18.03.2012 23:47	
Project: AVR_PB_LED	Desktop	
Cher Files	Obj 18.03.2012 23:47	
	application.c 11 KB 15.08.2009 15:17	
	Bibliotheken 6 KB 15.08.2009 22:18	
	🖉 🖉 🎽	
	Computer	
	A	
	Netzwerk	
	•	
E Messages	Dateiname: "main c" "application c"	_ ♥擧⊠
🚊 Errors 📑 Warnings		
	Dateityp: C Compiler files (*.c)	

Bild 6.2.1-10: C-Dateien dem Projekt bekannt machen

Alle C-Dateien auswählen => Öffnen und OK im Configure Project-Menü

Achtung: Der Vorgang dauert eine Weile - Bitte nicht die Geduld verlieren!

Noch mal der Reihe nach (wegen der Überschneidung der Menüs):

Bild 6.2.1-06: Benennen der *.prj-Datei: AVR_PB_LED.prj => Speichern

Bild 6.2.1-09: Hauptfenster von CVAVR - zur Übersicht

- Bild 6.2.1-07: Configure Project Part 1 (das Menü bleibt zunächst noch sichtbar) => Add
- Bild 6.2.1-08: Neue Projekt-Struktur

Bild 6.2.1-10: C-Dateien dem Projekt bekannt machen => *.C-Dateien markieren => Öffnen

Bild 6.2.1-07: Configure Project => OK

Durch OK im Configure Project werden alle *.c-Dateien in das Projekt übernommen.

疑 CodeVisionAVR - L:\AVR_PB_LED\AVR_PB_LED.prj
File Edit Search View Project Tools Settings Help
Ů ⊂ ♂・見渡御 ビ 悩 😋 🏙 🛯 塾 曇。 ∽ ♀ ※ 噛 噛 亀 Ҳ 🗋 魯 左 左 // ※ 👒 ● - ● - 奥 奥 昌 昌 兆 個 (分。
#4 # #7 #5 #2 . 10 12 12 12 10 10 10 10 10 10 10 10 10 10 10 10 10
● 福 福 1 隆 😪 📽 🗑 🖬 🖬 🗰 📰 🗰
Set Code Navigator Set Code I Code VisionAVR Poiett AVR_PB_LED Poiett AVR_PB_LED.asm Beflies AVR_PB_LED.ist Cother Files
E Messages V 7 🛙
Errors B Warnings
1:1 Insert

Bild 6.2.1-11: Hauptfenster von CVAVR



Bild 6.2.1-12: Die Spalten "Function Call Tree", "Code Templates" und "Clipboard History" werden zunächst entfernt, um Platz für die *.c-Dateien zu erhalten

In der "File Pane" (Editier-Feld, das ist die 4. Spalte von links) - wo bereits die leere *.txt-Datei unter "Notes" vorgemerkt ist - können alle *.c-Dateien auch nebeneinander oder untereinander angezeigt werden. In der AVR_PB_LED.txt-Datei kann das Projekt unter "Notes" detailliert beschrieben werden - das soll uns hier aber noch nicht interessieren.

Wenn die Felder nicht gleich so erscheinen, wie sie im **Bild 6.2.1-12** dargestellt sind, so ist zu bemerken, dass die Felder - wie in anderen Anwendungen auch - minimiert, maximiert oder "gezogen" werden können, was hier geschehen ist.

Um Platz für die Quelldateien zu schaffen, werden jetzt die Spalten Function Call Tree, Code Templates und Clipboard History zunächst entfernt (🖾 vergl. Bild 6.2.1-12) und die "File Pane" verbreitert:



Bild 6.2.1-13: Verbreiterte "File Pane"

Mit ein wenig Jonglieren werden die Spalten gestaltet:

- 1. Spalten nebeneinander
- 2. Öffnen von main.c im Code Navigator
- 3. Öffnen von application.c im Code Navigator

Wie man sieht, kann man im großen Fenster noch eifrig alle Quell-Dateien verändern/editieren. Die farbliche Kennzeichnung der Syntaxhervorhebung weicht auch hier vom Beispieltext etwas ab, da der integrierte Editor vom CVAVR für die Syntaxhervorhebung nicht so viele Einstellmöglichkeiten bietet wie der UltraEdit (siehe **Teil 207 Editor - UltraEdit**).

CodeVisionAVR - L:\AVR_PB	_LED\AVR_PB_LED.prj		
Ele Edit Search Vew Project	Tools Settings Help		
Sac 🖓 🗟 🖬 🗱 🖬 😋 🥸	10 Q D. S. Prov 3 3 3 8 6	※ ■ 母 世世 # ※ № ● · @ · 요.2 目音 ● ● ● Ⅰ	
· # 也 # # # # # • • • • • • •	LLOLORPHE & G	N. 🕹 🕸 🖘 🗵 🕐 . 🗏 🖓 🗞 📾 . 🣍 🕵 🚷 .	
<u> </u>	<u>.</u> • • • • • • • • • • • • • • • • • • •		
Code Veragelor V V La	Croalitymetran V + L	NARP, PB, ELAmonic	144/9_PS (ED/optication of
9 19 Project AVR_PB_LED	E R Fendions	heles inches S	Copications Co
- Costorioso	-/S oup_nityou;		
5 men c			3 HODE = Tell C
E P List Fires		5	approximation to visit
AVB_P8_LED aim		e main.c - VI.1	C Transfer
AVP.P8_LED.map		E	8 Werwine
- B Other Files		Projekt 601	Compliance
		11 Module.(*.c.ind *.b).: application	11 Decm
		12 Reconcipte Rendvare Tractourst, Faster 27, 52, 53 and 1851, 1852, 1853	12 Autor Alvin Lenix (ALX27) besinned auf dem Nodul epplication von Cho Junnar 22
		14 Compiler / CodeVisionAVR	1 8 ×
		16 Detam	
		17 Autor	12 // Inseler-Date: einforges/
		19	15 findlude "application.h" // Initialisiorung der Aswendung
		20 Dieres Projekt shellt eine Appliketion sur Demonstretion der Taster (Pushbottons) und LXDr auf dem Testboerd der. 21 Die Taster sind nicht entprolit/	2
		22 33 Die 1931 der im Behammennt 202 und hat gebreachter Gerte 21 232	22 //- 22 // Stobale Verieblen:
		24 Die 1802 ist is Rubezostend XIN und bei getrueckter Teste S2 AUS	24 //
		25 Die 1805 teggelt keim Betaatigen der Taste S5 von ADS nach BIB und ungehehrt 26	25 volatile 000 app_reset_source: 26 //
		27	27
		20 Die Definitionen sind hier vie felgt festgelegt:	29 //Baupt-Funktion
		30 31 Executed Afress (A-Fine)	10 void app_init(void)
		32 *** LEDR	32 I ann reach sturre - NUXR: // Stokers des Wiknowstroller/WIXI-Status-Besisters
		33 IACT FC2 (kann in application.h geammert worden) 44 IST2 PCF (kann in application.h geamiert worden)	34 // Bib7=5 -> Reserviertes Bit stets 0 (micht belegt)
		35 LED2 PC4 (kenn in application.h gemendert werden)	33 // Bit6=3 => Reservientes Bit stats 0 (micht belegt) 34 // Bit6=3 => Reservientes Bit stats 0 (micht belegt)
		37 *** Taxte	27 // Bit4-3 -> Reservices Sit states 0 (micht belegt)
		Si FB2 (kann in application.h genemiest verden) Di FD2 (kann in application.h genemiest verden)	30 // DIDJ #DRF -> Recolding Reset Flag 34 // Ditz NORF -> Brown-out Reset Flag
		40 K3 PDS (keen in application.) generate verden)	40 // Bill SXRP -> Sciencel Report Flog
			42 NCCER = 0x00; // Losschen aller Reset Flags fuer nachten RESET-Syklus
		() // Header-Deteien einfoepen:	6 //
		45 finclude "application.b"	45 // Doroh Voreinstellung ("sterbup.ess") ist der Webchdog Timer ausgescheltet
		46	49 // Telef Lisk MATCHDOG
		45 // Globels Verishien:	48 WDTCOR - WDCE WDC) // Enable Watchdog Reset (Watchdog Timer Control Register; -1fms) 48 Amount
		49 //	50
		51	51 //- 51 for the versingestelline System Taktes may youn oin Quare als Taktesher gerachit varde
		58 // Bauptprogram main	51 #inder une constal clock // Abfrage, ob ein guare als Taktgeber gewachlt wurde
		54 //	ss #1f (MCD_CLOCE_FREQUENCY 1600000)
			54 CLEFR - CLEFCE: // Enable Clock Presseler mit CLEFR - 0x00 (CLEFCT ist Dit7 - 0x00) 55 CLEFR - 0: // Setem Clock Presseler, Division durch 1 (-1000s dmi 1400s-CleFr)
		<pre>si // Settem des System Tettes (EV Clock), Ausschalten mittet Benetiter Mitgele Mobile si app_init(): // Initialisierung der Anvendung</pre>	52 felif (HCU CLOCE FREQUENCY - 8050300) //
		50 // Trinisliniaren der Testen (Drebbebbons)	by CLEAR - CLEAR - OR -
	•		
- Messagas			9.4 v
Enors 🔂 Warnings			

Bild 6.2.1-14: Die *.c-Dateien sind bereit zum Editieren (Bildvergrößerung)



Bild 6.2.1-15: Vergrößerung von Bild 6.2.1-14 - Projekt-Stand vor der Kompilierung

6.2.2 Konfiguration des C-Projektes

Bevor das Programm kompiliert wird, sollen vorher noch einige andere Listen zur Konfiguration des Projektes genauer betrachtet werden. Dazu wird erst einmal das Konfigurations-Menü erneut aufgerufen:

			Configure Project AVR_PB_LED.prj	×
			Files C Compiler Before Build After Build	
			Input Files Output Directories	1
				1
			Executable Files:	
			Exe	
			Object Files:	P
			List Files:	8-1
CodeVisionAVR - L:\AVF	R_PB_LED\AVR_PB_LEI	D.prj	Linker Files:	8
File Edit Search View P	roject Tools Settings H	elp		Par.
		50		
	<u>C</u> omplie	F9 1		
] 🕈 🔁 🗗 👔 😵 😼 🎴	<u>B</u> uild	Shift+F9		
🐮: Code Navigator 🕺	Build All	Ctrl+F9		
🕞 👯 CodeVisionAVR 🛛 🖉	<u>S</u> top Compilation	-		
📄 🛱 Project: AVR_PB_ 📐	Clean Up			
Notes	Tafawaahian			
J application.c	1 Information			
Headers	Go to Next Error	F8		
🖃 👘 List Files 📕	Go to Previous Error	Ctrl+F8		
AVR_PB_LE	Go to Next Warning	F4		
	Go to Previous Warning	Ctrl+E4		
Uther Files	Notes			
ġ.	<u>Export</u>			
5	Configure 🔊			
	~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~			Пеір

Bild 6.2.2-01: Aufruf des Konfigurations-Menüs

Bild 6.2.2-02: Configure Project Part 2 - Output Directories

```
Project => Configure => Configure Project AVR_PB_LED.prj => Output Directories
```

In dieser Abbildung werden die Ausgabe-Ordner angezeigt, in denen bei der Kompilierung die Ergebnis-Dateien abgelegt werden.

### **Executable Files:**

*.hex-Dateien vom Linker

#### **Object Files:**

*.obj-Dateien nach dem Lauf des Assemblers

#### List Files:

*.asm-Dateien nach der Kompilierung

#### Linker Files:

*.a- und *.o-Dateien zum Linken/Binden

⇒ KINGSTON (L:)
 ⊕ ↓ ALES_ASM_Progs
 ⊟ ↓ AVR_PB_LED
 ↓ Exe
 ↓ Linker
 ↓ List
 ↓ Obj

Configure Project AVR_PB_LED.prj	Bild 6.2.2-03: Configure Project
Files C Compiler   Before Build   After Build	Part 3 - Code Generation
Code Generation Libraries Messages Globally #define Paths	
Clock:     16.000000     MHz     RAM       Data Stack Size:     256     bytes       Heap Size:     0     bytes       Internal RAM Size:     1024     bytes	Hier können Einstellungen für den Chip und den Compiler zur Generierung der Ausgabe- Dateien vorgenommen werden. Die Einstel- lungen werden wie angegeben vorgenom- men. Besonders zu beachten sind:
	Chip: ATMega88
Code Generation Ontimization Level: Bit Variables Size: 16 Vise GPIOR >31	Clock: 16 MHz
Maximal   Promote char to int   char is unsigned	Brogram Type: Application
Program Type: 6 bit enums Enhanced Par. Passing	Data Stack Size: 256 bytes
Application	Data Stack Size: 256 bytes
(s)printf Features:	Heap Size: 0 Bytes
int width 🗾 🗌 Use an External Startup Initialization <u>F</u> ile	Internal RAM Size: 1024 bytes
(s)scanf Features:  ✓ Clear Global ⊻ariables at Program Startup Stack End Markors	Bit Variable Size: 16
int, width File Output Format(s): COF ROM HEX EEP	Use <u>G</u> PIOR>31 nicht gesetzt
Preprocessor ✓ Create <u>P</u> reprocessor Output Files ✓ Include I/O Registers Bits <u>D</u> efinitions	<b>File Output Format(s):</b> Formate der Ausgabe-Dateien (hier ist vor- rangig das Format <b>HEX</b> für ein ausführbares Programm von Interesse).
✓ QK       X Cancel       ? Help         Configure Project AVR_PB_LED.prj         Files       C Compiler       Before Build       After Build         Code Generation       Libraries       Messages       Globally #define       Paths	Bild 6.2.2-04: Configure Project Part 4 - Messages
✓ Enable Warnings	Das ist die Liste <b>Messages</b> , in der man ange-
Warnings:	ben kann, welche Hinweise und Warnungen man erhalten möchte.
possible loss of precision     array index is out of range     Adv lose tig and drage is out of range	
	beim Testen bestimmter Warnungen kann beim Testen sehr hilfreich sein, um unerreich- baren Code anzuzeigen oder davor zu war- nen, dass Variable deklariert sind, die gar nicht benutzt werden.
<ul> <li>✓ function parameter name doesn't match its previous declaration</li> <li>✓ local variable was declared, but not referenced</li> <li>✓ local variable set, but not used</li> <li>✓ local variable is used before its value is set</li> <li>✓ global variable/constant was declared, but not referenced</li> <li>✓ label declared, but not referenced</li> <li>✓ label declared, but not referenced</li> <li>✓ local variable</li> </ul>	
<ul> <li>✓ expression with possibly no effect</li> <li>✓ unknown escape sequence</li> <li>✓ shift result will be 0</li> <li>✓ overflow is possible in 8 bit addition</li> </ul>	
<ul> <li>overflow is possible in 8 bit multiplication</li> <li>overflow is possible in 8 bit shift left</li> <li>overflow is possible in 16 bit multiplication</li> <li>overflow is possible in 16 bit multiplication</li> <li>overflow is possible in 16 bit shift left</li> <li>possibly incorrect assignment</li> <li>pointless integer comparison</li> <li>controlling expression is constant</li> </ul>	

7

Files       C Compiler       Before Build       After Build         Code Generation       Libraries       Messages       Globally #define       Paths	Bild 6.2.2-05: Configure Project Part 5 - Paths
#include paths (one per line):	<ul> <li>clude- und library-Dateien angelegt werden:</li> <li>#include paths (one per line): für globale Header-Dateien</li> <li>Library paths (one per line): für Bibliotheksdateien</li> <li>Für die hier generierten AVR-Projekte wird</li> </ul>
Library paths (one per line):	von dieser Moglichkeit <b>kein Gebrauch</b> ge- macht.

# 6.2.3 Arbeitsschritte zur Generierung eines C-Projektes

L:\AVR_PB_LED						
	🚱 🕞 🗸 🔽 🖌 🖌 🖌 🗸 🗸 🗸 🗸 🗸					
Datei <u>B</u> earbeiten <u>A</u> nsicht	: E <u>x</u> tras					
Organisieren 🔻 Freigeb	en für 🔻 Brennen Net	uer Ordner		•	0	
🖕 Favoriten	Name 🔺	Größe	Änderungsdatum	Тур		
	<ul> <li>Nicht angegeben (14) –</li> </ul>					
Desktop	👢 Exe		15.09.2012 21:13	Dateiordner		
	👢 Linker		15.09.2012 21:13	Dateiordner		
Computer	👢 List		15.09.2012 21:13	Dateiordner		
🔃 Netzwerk	👢 Obj		15.09.2012 21:13	Dateiordner		
📴 Systemsteuerung	application.c	11 KB	15.08.2009 14:17	C compiler source file		
Papierkorb	application.h	36 KB	15.08.2009 21:34	C compiler header file		
	📽 AVR_PB_LED.prj	6 KB	16.09.2012 13:15	CodeVisionAVR Proj		
	AVR_PB_LED.prj~	6 KB	16.09.2012 11:32	PRJ~-Datei		
	Z AVR_PB_LED.txt	0 KB	16.09.2012 13:15	Text Document		
	iomx8.h	46 KB	15.08.2009 15:08	C compiler header file		
	macros.h	4 KB	15.08.2009 20:07	C compiler header file		
	📄 main.c	6 KB	15.09.2012 21:58	C compiler source file		
	switches.h	4 KB	15.08.2009 20:09	C compiler header file		
	typedefs.h	9 KB	15.08.2009 20:13	C compiler header file		
14 Elemente						
14 Elemente					1.	

Bild 6.2.3-01: Ausgangslage zur Kompilierung - Stand der Dateien



Bild 6.2.3-02: Ausgangslage zur Kompilierung - CVAVR (Bildvergrößerung)

Die Ausgangslage für die Kompilierung sind in einem C-Projekt die sog. Quell-Dateien. In diesem Projekt sind das:

- Das Hauptprogramm main.c
- Das Modul application.c incl. application.h
- Die Header-Datei typedefs.h
- Die Header-Datei iomx8.h
- Die Header-Datei macros.h
- Die Header-Datei switches.h

Aus ihnen entsteht in grundsätzlich 4 Arbeitsschritten das fertige "Kompilat", das heißt die ausführbare ***.hex**-Datei:

**1. Arbeitsschritt "Lauf des Preprozessors**": Auswerten aller Anweisungen, die mit # beginnen. Das sind Anweisungen, die ausschließlich für den Preprozessor bestimmt sind (u. a. Einfügen von Header-Dateien, die mit #include "file" in den Quell-Dateien aufgeführt sind.

**2.** Arbeitsschritt "Lauf des C-Compilers" zur Übersetzung der in der Programmiersprache C geschriebenen Befehle in Assembler-Anweisungen => *.asm-Dateien (dieses sind ebenfalls editierbare Text-Dateien).

**3.** Arbeitsschritt "Lauf des Assemblers" zur Übersetzung der Assembler-Anweisungen in Maschinen-Befehle (Module im Objekt-Code) => *.a-Dateien und *.o-Dateien

**4. Arbeitsschritt "Lauf des Linkers (Binders)"** zur Verbindung aller Objekt-Module zu einem lauffähigen ausführbaren Computer-Programm => *.hex-Datei

In CVAVR kann das C-Projekt sequentiell, d.h. in Einzel- bzw. Stufenschritten, generiert werden:

Project	=>	Compile (F9)	<b>Dieser Schritt</b> produziert die Objekt-Dateien für den Linker. Die Kompilierung wird nur für die nach dem letzten Durchlauf veränderten Programm-Module durchgeführt.
Project	=>	Build (Shift+F9)	<b>Dieser Schritt</b> produziert ein neues Assembler-Quell-Pro- gramm mit der Erweiterung <b>*.asm</b> wobei nur die veränderten Quell-Dateien neu kompiliert werden. Wenn keine Fehler er- kannt wurden, wird automatisch der <b>Atmel-AVR-Assembler</b> aufgerufen, der die eben erzeugte <b>.asm</b> -Datei assembliert. Das Ausgabe-File ist eine ausführbare Programm-Datei mit dem For- mat <b>*.hex</b> .

Project => Build All (Ctrl+F9) Vollständige Kompilierung/Assemblierung/Verlinkung aller Quellen; Zusammenfassung aller Arbeitsschritte zu einem Arbeitsgang.

Da das Projekt bereits an anderer Stelle getestet wurde, sind keine Fehler (mehr) zu erwarten, so dass sofort mit dem Schritt **Build All (Ctr+F9)** fortgesetzt wird:

KodeVisionAVR - L:\AVR_PB_LED\AVR_PB_LED,	prj	
File Edit Search View Project Tools Settings Help	p	
🗋 🗁 😷 🖬 😹 🗿 💞 🔯 Check Syntax		■ 🛍 X 🗎 💩 症 症 ∥ ※   ∝β   ● • ∞ • 의 의 🖺 🖶 🕊 🖶 (↔) -
静 约 約 船 鼎 鼎 🎝 🗍 🏪 Compile	F9	🛐 🐜 🛼 . 💩 🛎 🛸 🗏 💟 . 🔣 🍇 🚸 🍕 . ! ? 💥 🌍 .
申 🖥 둼 🧃 🏗 😽 😼 🤽 🤷 Build St	hift+F9	
🐮: Code Navigator 🐕 Build All 💦 📿 📿	Ctrl+F9	_:\AVR_PB_LED\main.c
E- Code Vision AVR		Notes main.c 🔀
E-G Project: AVR_PB_ Clean Up		1 🗗 /*

Bild 6.2.3-03: Project => Build All (Ctrl+F9)

Wenn die vollständige Kompilierung fehlerfrei (aber wahrscheinlicht nicht kommentarlos !) gelaufen ist, werden über dem Haupt-Fenster von CVAVR zwei Informations-Listen am Ende der Projekt-Generierung offeriert. Sie zeigen die Ergebnisse für die Kompilierung und die Assemblierung.



Bild 6.2.3-04: Nach der Kompilierung mit Build All (Ctrl+F9) (Bildvergrößerung)



Bild 6.2.3-05: Compiler-Meldungen

Bild 6.2.3-06: Assembler-Meldungen

Wenn man tiefer in die Materie eingestiegen ist, so ergeben diese Listen sehr aufschlussreiche Hinweise zum Kompilat und den benutzten Ressourcen (Speicher-Belegungen). An dieser Stelle mögen die Hinweise

No errors und Assembly complete, 0 errors.

genügen. Sie zeigen an, dass das Projekt fehlerfrei kompiliert und assembliert wurde und ein ablauffähiges Objekt-Programm erzeugt wurde.

### 6.3 Einbinden von AVR Studio in den CVAVR

Der Compiler CVAVR ist dafür vorgesehen, mit dem Debugger vom AVR Studio ab der Version 4.13 zu kommunizieren. Dazu ist es notwendig, die Lokalisation von AVR Studio, d.h. den Ordner-Pfad von AVR Studio auf der System-Platte, der Anwendung CVAVR bekannt zu machen. Diese Lokalisation ist (unter der Annahme, dass AVR Studio auf seinem "angestammten" Platz installiert wurde):

C:\Program Files (x86)\Atmel\AVR Tools\AvrStudio4\AVRStudio.exe

Der Eintrag in CVAVR wird im Menü Debugger Settings vorgenommen:





💐 Debugger Settings	×
Debugger: AVR Studio 4.19	
Directory and Filename: 🛷	AND N
C:\Program Files (x86)\Atmel\AVR Tools\AvrStudio4\AVRStudio	8::-
<u>∭ ✓ O</u> K <u>X</u> Cancel <b>?</b> <u>H</u> elp	

Bild 6.3-02: Einstellen der Datei AVRStudio.exe

Settings => Debugger => Debugger Settings => Directory and Filename: C:\Program Files (x86)\Atmel\AVR Tools\AvrStudio4\AVRStudio.exe => OK

Sollte **AVRStudio.exe** rest an der vorgesehenen Lokalisation installiert worden sein, so kann sie mit der "File-Such-Funktion" gefunden werden:

👫 Debugger Dire	ectory and FileName			×
<u>S</u> uchen in:	📜 AvrStudio4		▼ 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
<b>\$</b>	Name  Nicht angegeben (8)	▼ Größe ▼	Änderungsdatum	▼ Тур
Zuletzt besucht	<ul> <li>bin</li> <li>dli</li> <li>edit</li> <li>resources</li> <li>skins</li> <li>AVRStudio.exe</li> <li>PluginManager.exe</li> <li>xmlconvert.exe</li> </ul>	3.223 KB 55 KB 284 KB	06.09.2012 21:49 06.09.2012 21:50 06.09.2012 21:49 06.09.2012 21:49 06.09.2012 21:49 25.08.2011 21:05 25.08.2011 20:58 25.08.2011 20:59	Dateiordner Dateiordner Dateiordner Dateiordner Dateiordner Anwendung Anwendung Anwendung
Netzwerk				
	Datei <u>n</u> ame: AVF Datei <u>t</u> yp: Exe	RStudio.exe	•	Ö <u>f</u> fnen Abbrechen

Bild 6.3-03: Suchen der Datei AVRStudio.exe

**Hinweis:** Hier findet ein gleitender Übergang zum nächsten Abschnitt statt. Man befindet sich scheinbar noch in der Anwendung **CVAVR**. Denn aus dieser heraus wird der **Debugger** aufgerufen, der aber schon Bestandteil des **AVR Studio** ist.

### 6.4 AVR Studio Debugger

Mit dem Debugger des AVR Studios kann man - bevor man den Objekt-Code des fertig kompilierten Projektes in den Flash-Speicher des Mikrocontrollers überträgt - Schritt für Schritt den Programmablauf simulieren.

CodeVisionAVR - L:\AVR_PB	_LED\Ay_PB_LED.p	rj	
<u>File Edit Search View Project</u>	Tools Settings Help		
) 🖸 🗁 🤰 - 📕 💓 🗿 🗂 🖄 🖒	& Code <u>W</u> izardAVR	Shift+F2	🛍 Ҳ 🗋 曇 症 症 ∥ ※ 💁 🌒 • ∞ • ∞ • 의 흳 🖶 🖶 🕊 🖶 🙌 •
) 🛤 🟥 🟥 📭 🏦 🏥 🔉 🖻 🔧 🤋	🛃 🎽 <u>D</u> ebugger 🔉	Shift+F3	🞭 🖏 - ) 💩 🕷 🛸 🗏 💟 - ) 🔣 🍇 🚸 🍕 🥥 - 🛛
🕒 🗄 🖪 👖 🗽 😽 🗟 🛍	Schip Programier	Shift+F4	
🏦:: Code Navigator 🛛 🛛 🌣 🏹 🖬	🚊 <u>T</u> erminal	Shift+F5	<b>v</b>
E CodeVisionAVR	LCD Vision	Shift+F6	
🖃 💾 Project AVR_PB_LED			
💦 Notes	≪§ <u>C</u> onfigure		
🖃 🕒 application.c	42 //		
🗄 📑 Warnings	43 // Header-Da	teien einfu	legen:

### Bild 6.4-01: Aufruf des Debuggers zum Projekt AVR_PB_LED

Tools => Debugger (Shift+F3) => öffnet AVR Studio => <u>File</u> => Open File...(Strg+O)

### Es startet AVR Studio (aus CVAVR heraus !):

Dabei wird ggf. das Startfenster mit einem Willkommensdialog geöffnet, wenn die Option Show this dialog on open aktiviert war. Da nicht mit Assembler oder AVR GCC gearbeitet wird, sondern ein AVR-Projekt mit CVAVR erstellt werden soll, wird das Häkchen bei Show dialog at startup entfernt - und das Fensterchen erscheint zukünftig nicht mehr.

Es ist zu beachten, dass bereits einige Optionen voreingestellt worden sein können, so dass das Abbild von AVR Studio nach der ursprünglichen Installation anders ausgesehen haben könnte.

ź	- A\	/R S	tudio												_	
:	File	Pro	oject Build	View -	Tools D	ebug	Help									
F			New File		Strg+N	1 9	6.6	- 4	: 🐴		2 2 2	1 I I I I I I I I I I I I I I I I I I I	100	¥ 🥫	ait	: <b>A</b>
		Þ	Open File	- An-	Strg+O	<u>k</u> 7	- i 📾		**	2 5	e² AUTO					
Γ			Close	V						I,	/O View					<b>•</b> ×
		Ы	Save		Strg+S					ь	읡 - 💦				→	
			Save As								Name				Value	
	4	Ø	Save All													
	oipr	6	Print		Strg+P											
Ν	Sti		Print Setup						<b>~</b> ×	<	•				1	
П	X		· · · ·								Name	Address		Value	Bits	
	۲		Exit													
			_						<u> </u>							
	🖺 Bu	ild	🕕 Message	Fin Fin	d in Files	🛛 🖉 🖓 Br	eakpoir	nts and	Trace	ŀ	•					
															CA	P NI



🐲 Öffnen				×
L:\AVR_PB_LED	AN		VR_PB_LED	durchsu 🔎
Organisieren 🔻 Neuer Ordn	er		8	• 🗌 📀
Name 🔶	Größe	Änderungsdatum	Тур	
<ul> <li>Nicht angegeben (25)</li> </ul>				
👢 Exe		23.09.2012 19:39	Dateiordner	
🐌 Linker		23.09.2012 19:39	Dateiordner	
👢 List		23.09.2012 19:39	Dateiordner	
👢 Obj		23.09.2012 19:39	Dateiordner	
application.c	11 KB	15.08.2009 14:17	C compiler source file	
application.cci	1 KB	23.09.2012 19:49	CCI-Datei	
application.h	36 KB	15.08.2009 21:34	C compiler header file	
application.hci	1 KB	23.09.2012 19:49	HCI-Datei	
application.i	9 KB	23.09.2012 19:49	C compiler library file	
AVR_PB_LED.cbi	1 KB	23.09.2012 19:49	CBI-Datei	
AVR_PB_LED.cof	4 KB	23.09.2012 19:49	COFF symbolic debug file	
AVR_PB_LED.fct	1 KB	23.09.2012 19:49	FCT-Datei	
🚨 AVR_PB_LED.prj	6 KB	23.09.2012 19:50	CodeVisionAVR Project File	
AVR_PB_LED.prj~	5 KB	23.09.2012 19:40	PRJ~-Datei	
📄 iomx8.h	46 KB	15.08.2009 15:08	C compiler header file	
iomx8.hci	11 KB	23.09.2012 19:49	HCI-Datei	
i macros.h	4 KB	15.08.2009 20:07	C compiler header file	
macros.hci	1 KB	23.09.2012 19:49	HCI-Datei	
📄 main.c	6 KB	15.09.2012 21:58	C compiler source file	
main.cci	1 KB	23.09.2012 19:49	CCI-Datei	
📄 main.i	9 KB	23.09.2012 19:49	C compiler library file	
switches.h	4 KB	15.08.2009 20:09	C compiler header file	
switches.hci	1 KB	23.09.2012 19:49	HCI-Datei	
📄 typedefs.h	9 KB	15.08.2009 20:13	C compiler header file	
typedefs.hci	4 KB	23.09.2012 19:49	HCI-Datei	
Datei <u>n</u> am	e: AVR_PB_LI	ED.cof	▼ All Files (*.*)	•
			Öffnen	Abbrechen

Bild 6.4-03: Suchen der COFF-Datei

Die benötigte **COFF**-Datei befindet sich im Verzeichnis L:\AVR_PB_LED des Projekt-Ordners vom Beispiel-Projekt AVR_PB_LED und hierher soll auch die Datei AVR_PB_LED.cof.aps abgespeichert werden:

😻 Save AVR Studio Project	File					×
	A Contraction		-	AVR_PB_LED durchs	uchen	2
Organisieren 🔻 Neuer Org	iner					0
Name 🔺	Größe	Änderungsdatum	Тур			
<ul> <li>Nicht angegeben (4)</li> </ul>						
👢 Exe		23.09.2012 19:39	Datei	ordner		
👢 Linker		23.09.2012 19:39	Datei	ordner		
👢 List		23.09.2012 19:39	Datei	ordner		
👢 Obj		23.09.2012 19:39	Datei	ordner		
Dateiname: AVR_	PB_LED_cor.aps	TE.				<b></b>
Dateityp: AVR S	tudio Project File	s (*.aps)				•
Ordner ausblenden				<u>Speichern</u>	Abbrech	en //

Bild 6.4-04: AVR_PB_LED.cof.aps speichern

Es dauert einen Moment, bis man die Debug-Plattform **AVR Simulator** für den Mikrocontroller **ATmega88** einstellen kann:

Select device and debug p	latform	
Studio 4	Select debug platform and device Debug platform: AVR Dragon AVR ONE! AVR Simulator AVR Simulator 2 ICE200 ICE40 ICE50 JTAG ICE JTAGICE mkll	Device: ATmega64C1 ATmega64M1 ATmega8 ATmega8515 ATmega8535 ATmega88A ATmega88P ATmega88PA ATmega88PA ATmega88PA ATmega88PA ATmega84VA ATmega8HVA ATmega8U2
	Copen platform	options next time debug mode is entered
	<u> «Back</u> <u>N</u> ext»	Finish <u>C</u> ancel Help

Bild 6.4-05: Debug-Plattform für den ATmega88 einstellen

Nach einer weiteren kurzen Verweildauer öffnet sich AVR Studio mit der für das C-Projekt **AVR_PB_LED** eingestellten Debug-Plattform mit zahlreichen **Toolbars**, von denen besonders hervorzuheben sind:

- Prozessor-Daten (Counter, Pointer, Register usw.)
- Die aktuelle Quell-Datei D:\AVR_PB_LED\main.c mit einem gelben Pfeil auf die erste ausführbare Anweisung
- Ein-/Ausgabe-Werte (I/O View; Hardware-Ressourcen)
- Nachrichten über die laufenden Aktivitäten (Message Output)

Jetzt "schiebt" man sich die Seiten der Felder so hin, dass eine vernünftige Übersicht erscheint:



Bild 6.4-06: Debug-Plattform mit Quell-Datei main.c (Bildvergrößerung)

Mit der Funktionstaste **F10** bzw. **F11** vom **PC** kann man nun Schritt für Schritt den Programmablauf simulieren. Wenn man dabei den Cursor auf eine Variable positioniert, dann bekommt man den jeweiligen Wert angezeigt. Im Menüpunkt <u>Debug</u> kann man sehen was sonst noch so alles möglich ist. Hier nur ein paar Hinweise der zahlreichen Möglichkeiten des Debuggers:

Wenn man in CVAVR eine Änderung im Quell-Programm vornimmt, neu kompiliert und wieder nach AVR Studio wechselt, dann merkt AVR Studio, dass sich die *.cof-Datei geändert hat und fragt nach, ob diese Datei neu geladen werden soll. Wenn diese Anfrage mit **JA** beantwortet wird, dann startet der Debugger nach dem Neuladen wieder von vorne.

Wenn AVR Studio neu aus CVAVR heraus gestartet wird, dann kann ein vorhandenes Projekt unter dem Menüpunkt **Project => Recent Project** *Projektname* wieder unkompliziert geladen werden.



🐄 AV	R SI	tudio	
E File	Pro	ject Build View Tool	ls Debug Help
: L1		Project Wizard	造 🗃 🤊 🕅 🔁 🙀 🗄 🦽 🏄 🧏 🏂 🦉 📰 🧮
: Trace		New Project	北上本 : 🗰 🗰 🗰 🦌 🖓 🕅 🕕
		Open Project	
	ø	Save Project	
		Close Project	
		Recent Projects	1 AVR_PB_LED_cof.aps
			2 L:\ALEs_ASM_Progs\ALE_0002\ALE_0002.aps
			3 L:\ALEs_ASM_Progs\ALE_0001\ALE_0001.aps

Bild 6.4-07: Fortsetzen des Debugging nach einem Neustart

Wenn man in diesem Projekt beispielsweise auf **PORTC** klickt, dann kann man während der schrittweisen Programmausführung (mit **F10**) sehen, wie sich **PC2**, **PC3**, **PC4** (entsprechend dem kurzen Aufleuchten der LED's) nach dem Reset verändern.

# 6.5 Flashen eines C-Programms in ein Mikrocontroller ATmega88

Das Programm AVR_PB_LED ist jetzt kompiliert und simuliert worden. Aber ein Test im "Wirkbetrieb" ist natürlich noch wichtiger. Dazu muss der Objektcode (das kompilierte Programm) in den Mikrocontroller (hier in den Mikrocontroller auf dem AVR-ALE-Testboard) geflasht werden und es müssen die notwendigen Verbindungen zwischen den Pins hergestellt werden. Die Beschaltung ist identisch mit dem Abschnitt **1.2 Beschaltung** im **Teil 601 - AVR_PB_LED**:



Bild 6.5-01: Beschaltung der LEDs und Taster (Bildvergrößerung)

Dann wird das Board mit seinem Netzteil mit Spannung versorgt.

Der ISP-Programmieradapter wird mit seinem USB-Stecker in einen freien USB-Steckplatz des PCs gesteckt, in dem das neue Programm abgespeichert wurde.

Überflüssige Schaltkreise und die Stromversorgung wurden in dem Beschaltungsbild weggelassen. AVR Studio wird nun erneut gestartet.

Dann wird die logische Verbindung zum CC2-AVR Programmer (ISP-Programmieradapter) hergestellt. Die Einstellungen wurden ja schon im Abschnitt 4.3.2 Mikrocontroller-Einstellungen im AVR Studio von Teil 204 AVR Studio vorgenommen, so dass man sich damit nicht mehr aufhalten muss:



Bild 6.5-02: Mit dem CC2-AVR-Programmer eine Verbindung herstellen

AVRISP mkII in ISP mode with ATmega88	
Main Program Fuses LockBits Advanced HW Settings HW Info Auto	
Device	
Erase Device	
Eraco dovice before fleeb programming	
Flash	
C Use Current Simulator/Emulator FLASH Memory	
(• Input HEX File	
Program Verify Read	
EEPROM	
C Use Current Simulator/Emulator EEPROM Memory	
Program Verify Read	
ELE Production File Format	
Save From: V FLASH V EEPROM V FUSES LOCKBITS Fuses and lockbits settings	
Program Save saving to ELF	
Detecting on 'USB'	
Getting isp parameter SD=0x05 OK	-

Bild 6.5-03: Objektdatei (*.hex-File) suchen

😻 Öffnen				×
G O	2		👻 🛃 🛛 Exe durch	nsuchen 🛛 🔎
Organisieren ▼ Neuer Ordner				= - 🗌 📀
🗄 🧼 Sicherungs-LW 100 📥 🛛 Na	ame 🔺	Größe	Änderungsdatum	Тур
⊕	icht angegeben (1) —			
E KINGSTON (L:)	AVR_PB_LED.hex	2 КВ	23.09.2012 19:49	HEX-Datei
🔏 🐌 Exe				
💞 📜 Linker				
👢 List				
👢 Obj				
🕀 💽 Netzwerk				
🕀 📴 Systemsteuerung				
Papierkorb				
▼ 4				
Datei <u>n</u> ame: AV	/R_PB_LED.hex		Flash Intel He	x Files (*.hex;* ▼
	•		Ö <u>f</u> fnen	Abbrechen



evice Erase Device Frase device before flash programming Verify device after programming Iash Use Current Simulator/Emulator FLASH Memory Input HEX File I:AVR_PB_LED\Exe\AVR_PB_LED.hex Frogram Verify Read EPROM Use Current Simulator/Emulator EEPROM Memory Input HEX File Frogram Verify Read LF Production File Format Program Fuses Fuses and lockbits settings must be specified before saving to ELF Fige Fige NUSB' Program Fuses Fuse Fuse	in f	Program Fuses LockBits Advanced HW Settings HW Info Auto
Erase Device         ✓ Erase device before flash programming         Iash         ○ Use Current Simulator/Emulator FLASH Memory         ● Input HEX File       L:\AVR_PB_LED\Exe\AVR_PB_LED.hex         Program       Verify         Read         EPROM         ● Use Current Simulator/Emulator EEPROM Memory         ● Input HEX File         ● Program         Verify         Read	Devid	
Erase device before flash programming Verify device after programming Iash Use Current Simulator/Emulator FLASH Memory Input HEX File L:AVP_PB_LED\Exe\AVP_PB_LED.hex Program Verify Read EPROM Ourse Current Simulator/Emulator EEPROM Memory Input HEX File Program Verify Read EPROM Frogram Verify Read EPROM EPROM Frogram Verify Read EPROM EPROM Frogram Verify Read EPROM Frogram Verify Read EVENT Simulator/Emulator EEPROM Memory Input HEX File Program Verify Read EVENT Simulator Simu		Erase Device
lash       O Use Current Simulator/Emulator FLASH Memory <ul> <li>Input HEX File</li> <li>L'AVR_PB_LED\Exe\AVR_PB_LED.hex</li> <li>Imput Program</li> </ul> Imput HEX File                Program       Verify       Read                EPROM <ul> <li>Use Current Simulator/Emulator EEPROM Memory</li> <li>Input HEX File</li> <li>Imput HEX File</li> <li>Imput HEX File</li> <li>Program</li> <li>Verify</li> <li>Read</li> </ul> Is program       Verify       Read       Imput HEX File	ŀ	<ul> <li>Erase device before flash programming</li> <li>Verify device after programming</li> </ul>
Output HEX File       L'AVR_PB_LED\Exe\AVR_PB_LED.hex         Program       Verify         Read    EPROM          Output HEX File	lash	h
Input HEX File   L'AVR_PB_LED\Exe\AVR_PB_LED.hex     Program     Verify     Read     EPROM     © Use Current Simulator/Emulator EEPROM Memory     © Input HEX File     Program     Verify     Read     EPROM     © Use Current Simulator/Emulator EEPROM Memory     © Input HEX File     Program     Verify     Read     LF Production File Format     nput ELF File:     Save From:     FLASH   EPROM   FUSES   LOCKBITS   Fuses and lockbits settings   must be specified before   saving to ELF        ting on 'USB'	- 0	🔿 Use Current Simulator/Emulator FLASH Memory
Program       Verify       Read         EPROM       Imput EPROM       Imput HEX File       Imput HEX File         Imput HEX File       Imput HEX File       Imput HEX File       Imput HEX File         Imput ELF File:       Imput ELF File:       Imput FUSES       LOCKBITS         Save From:       Imput FLASH       EEPROM       FUSES       LOCKBITS         Frogram       Save       Save       Saving to ELF         ting on 'USB'       Ye mkll with serial number 0000A0012825 found       Save	6	Input HEX File     L:\AVR_PB_LED\Exe\AVR_PB_LED.hex
EPROM © Use Current Simulator/Emulator EEPROM Memory © Input HEX File Program Verify Read LF Production File Format nput ELF File: Save From: ♥ FLASH ♥ EEPROM ♥ FUSES ♥ LOCKBITS Fuses and lockbits settings must be specified before saving to ELF ting on 'USB' P mkll with serial number 0000A0012825 found		Program Verify Read
EPROM		
Ouse Current Simulator/Emulator EEPROM Memory     ● Input HEX File	EPF	ROM
Input HEX File     Program     Verify     Read     Read     IF Production File Format     nput ELF File:     Save From:    FLASH    EEPROM    FUSES    LOCKBITS     Program     Save     Save     saving to ELF     ting on 'USB'     Program		Use Current Simulator/Emulator EEPROM Memory
Program       Verify       Read         LF Production File Format          nput ELF File:          Save From:       ✓ FLASH       ✓ EEPROM         Program       Save       Save source for the specified before saving to ELF         ting on 'USB'       SP mkll with serial number 0000A0012825 found	(	Input HEX File
LF Production File Format nput ELF File:		Program Verify Read
nput ELF File:	ELF F	Production File Format
Save From: FLASH EEPROM FUSES LOCKBITS Fuses and lockbits settings Program Save saving to ELF ting on 'USB'	Input	it ELF File:
Program Save Save Save Save Save Save Save Save	Save	
Program Save saving to ELF	Jave	must be specified before
ting on 'USB' P mkll with serial number 0000A0012825 found		Program Save saving to ELF
ting on 'USB' P mkll with serial number 0000A0012825 found		
ting on 'USB' YP mkll with serial number 0000A0012825 found		
SP mkll with serial number 0000A0012825 found	etina	a on 'I ISB'
	ISP n	mkll with serial number 0000A0012825 found.

Bild 6.5-05: Programm flashen ==> und das Programm testen ==> alles O.K.