

Teil 501 - Einführung

1 Eine Einführung in C

- 1.1 Warum C?
- 1.2 Wie entstand C?
- 1.3 Der AVR-Mikrocontroller in einem eingebetteten System
- 1.4 Werkzeuge (siehe Gruppe 200)

Teil 502 - Aufbau eines C-Projektes

2 Was ist ein C-Projekt?

2.1 Erzeugen eines C-Projektes 2.1.1 Ein neues Projekt beginnen 2.1.2 Ein C-Projekt generieren 2.2 Dateistruktur eines C-Projektes 2.3 Einbindung von AVR Studio in den CVAVR 2.4 AVR Studio Debugger 2.5 C-Compiler-Optionen

Teil 503 - Preprozessor

3 Preprozessor-Anweisungen

- 3.1 Struktur der C-Quell-Programme
- 3.2 #include-Anweisung
- 3.3 #define-Anweisung (Makro)
 - 3.3.1 Makros ohne Parameter
 - 3.3.2 Makros mit Parametern
- 3.4 **#undef**-Anweisung

3.5 #if-, #ifdef-, #ifndef-, #else- und #endif-Anweisungen

3.6 Andere Preprozessor-Anweisungen

Teil 504 - Syntax der C-Programmiersprache

- 4 Die Syntax der C-Programmiersprache
 - 4.1 C-Quell-Programme
 - 4.1.1 Kommentare
 - 4.1.2 Deklarationen (Vereinbarungen)
 - 4.1.3 Die Funktion main
 - 4.1.4 Schlüsselwörter (Keywords) des CodeVisionAVR C-Compilers
 - 4.2 Konstanten und Variablen
 - 4.2.1 Zahlensysteme
 - 4.2.2 Datentypen
 - 4.2.3 Konstanten
 - 4.2.4 Variablen
 - 4.3 Operatoren
 - 4.3.1 Arithmetische Operatoren
 - 4.3.2 Relationale Operatoren
 - 4.3.3 Logische und bitweise wirkende Operatoren
 - 4.3.4 Andere Operatoren und Shortcuts
 - 4.4 Komplexe Objekte in C
 - 4.4.1 Funktionen
 - 4.4.2 Funktions-Prototypen
 - 4.4.3 Pointers und Arrays
 - 4.4.3.1 Pointers



- 4.4.3.1.1 Pointers in Verbindung mit flash und eeprom
- 4.4.3.1.2 Pointers in Verbindung mit typedef
- 4.4.3.2 Arrays
 - 4.4.3.2.1 Ein-dimensionale Arrays
 - 4.4.3.2.2 Zwei-dimensionale Arrays
 - 4.4.3.2.3 Drei-dimensionale Arrays
- 4.4.3.3 Benutzen der Array-Namen als Pointers
- 4.4.3.4 Arrays von Pointers
- 4.4.3.5 Pointers auf Funktionen (Funktionszeiger)
- 4.4.3.6 Funktionen in Verbindung mit typedef
- 4.4.4 Strukturen und Unionen
 - 4.4.4.1 Strukturen
 - 4.4.4.2 Unionen

4.4.5 Komplexe Typen (eine Zusammenfassung)

4.5 Steuerung des Programmablaufs

- 4.5.1 Anweisungsblöcke { . . . }
- 4.5.2 Die Anweisung if
- 4.5.3 Die Anweisung if in Verbindung mit else
- 4.5.4 Die Fallunterscheidung switch
- 4.5.5 Die Schleife for
- 4.5.6 Die Schleife while
- 4.5.7 Die Schleife do in Verbindung mit while
- 4.6 Arbeiten mit den Ein-/Ausgabe-Ports

Teil 505 - Modularer Aufbau der AVR-C-Projekte

5 Modularer Aufbau

- 5.1 Das Konzept
- 5.2 Nomenklatur
- 5.3 Die speziellen Header-Dateien
 - 5.3.1 Die spezielle Header-Datei typedefs.h (Typ- und Bit-Definitionen)
 - 5.3.2 Die spezielle Header-Datei iomx.h (Definitionen aller Register-Bits)
 - 5.3.3 Die spezielle Header-Datei macros.h (Definitionen von Makros)
 - 5.3.4 Die spezielle Header-Datei switches.h (Definitionen von Schaltern)
- 5.4 Die AVR-C-Module
- 5.5 Anwendung der angepassten AVR-C-Module
 - 5.5.1 Das AVR-C-Modul application (Anwendungs-Steuerung)
 - 5.5.2 Das AVR-C-Modul lcd 2wire (Ausgabe auf LCD-20x4)
 - 5.5.3 Das AVR-C-Modul num conversion (Typ-Konvertierung nach ASCII)
 - 5.5.4 Das AVR-C-Modul adc ref 1 1 (ADC mit interner Referenz 1,1 V)
 - 5.5.5 Das AVR-C-Modul **rc5_decoder** (RC5-IR-Fernsteuerung-Dekoder)
 - 5.5.6 Das AVR-C-Modul rc5 encoder (RC5-IR-Fernsteuerung-Enkoder)
 - 5.5.7 Das AVR-C-Modul usart (USART-Steuerung)
 - 5.5.8 Das AVR-C-Modul twi_master (IR2- bzw. TWI-Steuerung)
 - 5.5.9 Das AVR-C-Modul timer0_pwm (TIMER0-Steuerung)

Teil 506 - Anhang

- 6 Anhang
 - 6.1 Begriffe und Definitionen
 - 6.2 Bibliothek

Vorbemerkung

Nichts ist vollkommen - und nichts ist endgültig! So auch nicht dieses Tutorial! Deshalb bitte immer erst nach dem neuesten Datum schauen. Vielleicht gibt es wieder etwas Neues oder eine Fehlerbereinigung oder eine etwas bessere Erklärung. Wer Fehler findet oder Verbesserungen vorzuschlagen hat, bitte melden (<u>info@alenck.de</u>).

Immer nach dem Motto: Das Bessere ist Feind des Guten und nichts ist so gut, dass es nicht noch verbessert werden könnte.

Bild-, Beispiel-, Form- und Tabellen-Nummern sind nach folgendem Schema aufgebaut, damit bei Einfügungen/Löschungen nicht alle Nummern wieder geändert werden müssen (hier bunt dargestellt):

Darstellungsart Abschnitt-LfdNummer: Beschreibung allgemeines Schema

- Bild 5.1.4-02: Daten-Adress-Raum
 Beispiel 5.1.4-03: EEPROM-Speicherung
 Form 5.1.3-01: Die main-Funktion
 Tabelle 5.1.4-01: Schlüsselwörter vom CVAVR
 Benummerung einer Tabelle
- Gravierende Änderungen gegenüber der Vorversion

1.

Völlig neue Strukturierung in **Gruppen** und **Teile**, um das Tutorial umfassend ordnen zu können. Die **Abschnitte** in den **Teilen** sind weitgehend erhalten geblieben.

| Gruppenbezeichnung | Kurzbezeichnung |
|---|-------------------------|
| Gruppe 100: Technologie der AVR-8-Bit-Mikrocontroller | Technologie |
| Gruppe 200: Einsetzen von AVR-Tools | Tools |
| Gruppe 300: Arbeiten mit AVR-Assembler | ASM-Programmierung |
| 3xx_Programm_yyyyy | ASM-Programm-Beispiel |
| Gruppe 400: AVR-ASM-Projekte | ASM-Projekte |
| 4xx_Projekt_yyyyy | ASM-Projekt-Bezeichnung |
| Gruppe 500: CodeVisionAVR C-Compiler | C-Programmierung |
| 5xx_Programm_yyyyy | C-Programm-Beispiel |
| Gruppe 600: AVR-C-Projekte | C-Projekte |
| 6xx_Projekt_yyyyy | C-Projekt-Bezeichnung |

xx steht für die laufende Nummer innerhalb des **Teils**, in dem das Programm bzw. das Projekt erscheint und **yyyyy** steht für die Programm- bzw. Projekt-Kurz-Bezeichnung.

2.

Notwendige Änderungen auf Grund Neuinstallation von Windows 7.

3.

Windows 7 machte eine Installation von CodeVisionAVR V2.60 als Vollversion notwendig. Daraus leiten sich auch viele Änderungen im Detail für die C-Programmierung (Gruppe 500) ab.

4.

Neu-Installation von AVR Studio Vers. 4.19 unter Windows 7

5.

Zur Demonstration des Tools AVR Studio ist in Gruppe 200 eine Trennung in Teil 205 - Assembler und AVR Studio und Teil 206 - C-Compiler und AVR Studio vorgenommen worden.

6.

ASM- und **C-Projekte** werden jeweils in eigenen Gruppen gesammelt (**Gruppe 400** für Assembler- und **Gruppe 600** für C-Projekte).

Vorbemerkung zu diesem Teil 502

In diesem **Teil 502** wird überwiegend der **Teil 206 - C-Compiler und AVR Studio** eingebettet (d.h. wiederholt), um die **C**-Programmierung mit Hilfe des CodeVisionAVR C-Compilers hier möglichst zusammenfassend darzustellen zu können.

2 Was ist ein C-Projekt ?

Ein **C-Projekt** ist die Summe aller Dateien, die zur Lösung eines Aufgaben-Komplexes (einer Anwendung) mittels **C**-Programmierung notwendig sind.

Es steht hier zunächst nicht die Lösung des Problems im Vordergrund und auch nicht die Aufgabenstellung, sondern es wird - um den grundsätzlichen Aufbau eines C-Projektes darzustellen - von fertigen Quell-Dateien ausgegangen, die bereits fertig erstellt, getestet und erprobt sind. Zur Anschauung wurde das fertige 601_AVR_C_Projekt_PB_LED (aus der Gruppe 600) gewählt. Zur Abkürzung des Namens wird es hier nur mit AVR_PB_LED bezeichnet.



Bild 2-01: Das Projekt AVR_PB_LED (hier: fertig kompiliert)

Im linken Teil des Bildes **Bild 2-01: Das Projekt AVR_PB_LED** ist im **Code Navigator** eine Aufstellung der Quell-Dateien zu sehen. Die Quell-Dateien mit dem Programm-Quell-Code (*.c-Dateien), d.h. mit den Programm-Befehlen (Anweisungen, Funktions-Aufrufe, Algorithmen usw.), sind:

- main.c ist das erste aufgerufene Haupt(main)programm mit dem C-Programm-Quell-Code, d.h. in der Regel mit vielen Funktionsaufrufen. Im rechten Teil ist der Kode dieses Hauptprogramms aufgerufen und kann dort editiert werden.
- application.c das ist die C-Quell-Datei mit der Initialisierung der Anwendung, d.h. mit den Anweisungen, die für die Einstellung des Taktes, der Interrupt-Vektoren usw. benötigt werden.

Es folgen die Header-Dateien (*.h-Dateien). Da sind als erstes die selbst generierten Header-Dateien mit den Beschreibungen der Konstanten, Makros, Datentypen, globalen Variablen und den sog. Funktions-Prototypen zu nennen:

- application.h Header-Datei zur Definition der im Projekt veränderbaren Parameter
- iomx8.h
 Header-Datei zur Definition der Bits des benutzten Mikrocontrollers ATmega88
- macros.h Header-Datei zur Definition anwendungsspezifischer Makros
- switches.h Header-Datei zur Definition der Tast-Schalter
- typedefs.h Header-Datei zur Definition "eigener Datentypen" (Definition äquivalenter Namen zu bestehenden Datentypen)

Und zum guten Schluss noch einige globale Header-Dateien, die ebenfalls in der linken Spalte im **Bild 2-01** auftauchen. Das sind Header-Dateien aus dem "Datei-Fundus", der mit CVAVR ausgeliefert wurde (hierzu mehr im **Teil 503 - Der Preprozessor** und **Teil 504 - Syntax der C-Programme**):

- ctype.h Header-Datei zur Definition von Funktionen zur Bestimmung von Zeichen-Typen (z.B. alphanumerische, alphabetische, dezimale Zeichen usw.).
- delay.h Header-Datei zur Definition von Funktionen zur Verzögerung schnell ablaufender Vorgänge.
- limits.h Header-Datei zur Definition von Minima und Maxima von Datentypen.
- mega88.h Header-Datei zur Definition von controllerspezifischen Registern, Ports, Timer usw.
- stdarg.h Header-Datei zur Definition von Makros zur Behandlung variabler Argumenten-Listen.
- stdio.h Header-Datei zur Definition von Funktionen zur Standard-Eingabe und Standard-Ausgabe.
- stdlib.h Header-Datei zur Definition von Standard-Bibliotheksfunktionen.
- string.h Header-Datei zur Definition von Funktionen zur Manipulation von Strings (Zeichenketten).

Wo findet man sie - oder viel wichtiger: wo findet sie der Compiler CVAVR?? Antwort: CVAVR hat sie beim Setup selbst in den von ihm eingerichteten Ordner

c:\cvavr2\inc

abgelegt und da der Aufruf des Namens mit spitzen Klammern geschieht, z.B. <delay.h>, schaut der Preprozessor als erstes hier nach:

| C:\cvavr2\inc | | | | | | |
|---|--------------------------|-------|--------------------|-----------|---|--|
| | | - | 🗕 🛃 🛛 inc durchsuc | hen | 2 | |
| <u>Datei B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | | | |
| Organisieren 🔻 🔄 Öffnen 🔻 Brennen Neuer Ordn | er | | | - | 0 | |
| System-LW 400 GB (C:) | Name 🔶 | Größe | Änderungsdatum | Тур | | |
| SRecycle.Bin | mega64c1.h | 6 KB | 21.07.2009 14:57 | H-Datei | | |
| Benutzer | mega64c1_bits.h | 53 KB | 21.07.2009 15:12 | H-Datei | | |
| 📜 cvavr2 | mega64m1.h | 7 KB | 21.07.2009 14:50 | H-Datei | | |
| 📙 bin | mega64m1_bits.h | 57 KB | 25.05.2010 10:17 | H-Datei | | |
| 📙 Code Templates | 📄 mega88.h | 4 KB | 14.04.2009 15:44 | H-Datei | | |
| k examples | 📄 mega88.hci 💖 | 3 KB | 23.09.2012 19:49 | HCI-Datei | | |
| examples atxmega | mega88a.h | 5 KB | 14.10.2010 08:54 | H-Datei | | |
| lib 🔽 | mega88p.h | 5 KB | 14.04.2009 15:45 | H-Datei | - | |
| mega88.h Änderungsdatum: 14.04.2009 15:44 H-Datei Größe: 3,95 KB | Erstelldatum: 07.09.2012 | 20:15 | | | | |
| 1 Element ausgewählt | | | 🔊 Computer | | | |

Bild 2-02: Auszug aus dem Datei-Ordner des lokalen Datenträgers (C:)

2.1 Erzeugen eines C-Projektes

Zur Demonstration einer Projekt-Generierung wird ein neuer Projekt-Ordner **AVR_PB_LED** angelegt, in dem alle fertigen ***.c-** und ***.h-**Dateien des fertigen AVR-Projektes **AVR PB_LED** übertragen werden.

Siehe: Teil 601 PB_LED - Einfache Beschaltung von LEDs und Tastern

Der Ordner-Name wurde vom Namen des bestehenden Projektes 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.

| L:\AVR_PB_LED | | | | | | | |
|--|--|-------|------------------|------------------------|------|--|--|
| 🚱 🕞 🗸 🗸 Computer 🔻 KINGSTON (L:) 🖛 AVR_PB_LED 🛛 🔹 😰 🗛 🖓 AVR_PB_LED durchsuchen 🛛 🖉 | | | | | | | |
| <u>Datei B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | Datei Bearbeiten Ansicht Extras ? | | | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen | Neuer Ordner | | | | 0 | | |
| Icherungs-LW 1000 GB (I:) | ▲ Name ▲ | Größe | Änderungsdatum | Тур | | | |
| Wechsel-LW2 160 GB (J:) ExtVideo-LW 1000 GB (K:) | Nicht angegeben (7) — application c | 11 KB | 15.08.2009.14.17 | C compiler source file | - | | |
| KINGSTON (L:) ALES_ASM_Progs | application.h | 36 KB | 15.08.2009 21:34 | H-Datei | | | |
| L AVR_PB_LED | iomx8.h | 46 KB | 15.08.2009 15:08 | H-Datei | | | |
| Wechseldatenträger (M:) | macros.h | 4 KB | 15.08.2009 20:07 | H-Datei | | | |
| Netzwerk | 📄 main.c | 6 KB | 15.09.2012 21:58 | C compiler source file | | | |
| Systemsteuerung | switches.h | 4 KB | 15.08.2009 20:09 | H-Datei | | | |
| | ▼ typedefs.h | 9 KB | 15.08.2009 20:13 | H-Datei | | | |
| 7 Elemente | | | | | | | |
| 7 Elemente | | | 🔊 Comp | uter | - // | | |

Bild 2.1-01: Einen neuen Projekt-Ordner mit Quell-Dateien anlegen

2.1.1 Ein neues Projekt beginnen

In der **Gruppe 200 - Einsetzen von AVR-Tools** wurde schon das Tool CodeVisionAVR installiert und auch schon ein Hinweis gegeben, wie man mit diesem Tool Quell-Dateien editieren kann. Jetzt wird das erste Projekt (wenn auch nur entlehnt) durch Start dieses Tools erzeugt:



Bild 2.1.1-01: CVAVR für ein neues Projekt starten



| 🕺 CodeVisionAVR | <u>- 🗆 ×</u> |
|---|---|
| Eile Edit Search View Project Tools Settings Help | |
| ▶▶♀♀~────────────────────────────────── |) · · ◎ · 의 의 阊 릠 ≞ · ⁄ · · |
| # # # # # . 6 1: 1: 2: 2: 0 | 💐 🍕 🗸 🛛 ? 🔛 🎯 🗸 |
|] 🖶 🏗 🚦 1 🗽 😻 👻 🚍 📭 🖃 💷 💷 💷 | |
| E:: Code Navigator Image: Code Vision AVR Image: CodeVision AVR Image: CodeVision AVR | Code Tem ♡ ♀ X if () { }; if () { }; do { while (); while () { }; for (;;) { |
| E Messages | ♡ 平 🛛 |
| 🖹 Errors 🖹 Warnings | |
| | |
| | |

Bild 2.1.1-02: 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 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.





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 <u>AVR_PB_LED</u>. 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:):



| 👫 Create New Project | X Create New Project |
|--|---|
| Speichem in: 🕌 AVR_PB_LED 💿 😰 📰 🔹 | Speichem in: 🏭 AVR_PB_LED 💽 🚱 🏂 🔛 |
| Name Image: Computer Subscription Zuletzt besucht Image: Computer Subscription Bibliotheken Image: Computer Subscription Netzwerk Image: Computer Subscription | Image: Second |
| Dateiname: Speich Dateityp: Project files (* prj) Z | Dateiname: AVR_PB_LED & Speichem Dateityp: Project files (*,prj) |

Bild 2.1.1-05: Es wird eine *.prj-Datei angefordert

Bild 2.1.1-06: 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).

| K Configure Project AVR_PB_LED.prj | × | | |
|---|-------------------|---|----------------------------|
| Files C Compiler Before Build After Build | | | |
| Input Files Output Directories | | | |
| J:\AVR_PB_LED\AVR_PB_LED.prj | New | | |
| | Add | | |
| | +: Bemove | | |
| | | | |
| | | | |
| | ↔ Move <u>U</u> p | | |
| | | | |
| | | | |
| | | Add File to Project | |
| | | | ✓ Größe ✓ Änderungsdatum ✓ |
| | | Nicht angegeben (6) Zuletzt besucht | 18 03 2012 23:47 |
| | | | 18.03.2012 23:47 |
| | | Desktop | 18.03.2012 23:47 |
| | | Bibliotheken main.c | 6 KB 15.08.2009 22:18 |
| | | | |
| | | Computer | |
| | | Netzwerk | |
| | | Dateiname: | Offnen |
| ✓ <u>D</u> K X Canc | el <u>? H</u> elp | Dateityp: C Compiler files (*.c) | Abbrechen |
| | | | |

Bild 2.1.1-07: Configure Project Part 1 - Input Files

Bild 2.1.1-08: 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_LE_prj |
|--|
| File Edit Search View Project Tools Settings Help |
| └♀♡~ 美食じは べ」 2 風母・ ♀♀ おもももべき 5 元 元 // ※ ◎ ~ ◎ ~ ♀ 2 2 書 * 相 け・ |
| 桷 华 华 华 恭 微 , [6] 12 12 12 10 16 10 19 10 19 12 10 10 10 10 10 10 10 10 10 10 10 10 10 |
| 中福福 1 監 💘 🗑 🖬 🖬 🐶 🗉 🕮 📮 |
| 🖹 Code Navigator 🛛 🖓 🖸 😪 Code Information 🖓 🛱 🖸 💱 Function Call Tree 🐺 🛛 🔛 LAVR_PB_LEDVAVR_PB |
| EndevisionAVR |
| Project AVR_PB_LED |
| General Control Contro |
| |
| |
| do (|

Bild 2.1.1-09: 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 | a 🔿 🖉 | |
|] 🗅 👄 😍 • 🔲 🗑 🗇 🖄 🖄 🖄 💩 🎒 | Suchen in: 🛛 AVR_PB_LED | • 🕝 🌶 📂 🔜 • | ·= (+) - |
|] 秭 鸟 色 岬 盘 體 🚛 🖻 📑 😒 😒 🔞 👠 📵 | Name - | Größe 🗸 Änderungsdatum 🗣 | |
| 🖶 🐂 🐂 1 🗽 😽 🦞 🚟 🖍 🗖 🗊 | Nicht angegeben (6) | | Ē . |
| 🐮: Code Navigator 🛛 💝 🏹 🔀 Code Informa 🔇 | Zuletzt besucht | 18.03.2012 23:47 | Cli 💙 74 🔀 |
| E-CodeVisionAVR | Linker | 18.03.2012 23:47 | |
| Project: AVR_PB_LED | Desktop | 18.03.2012 23:47 | |
| Uther Files | Obj | 18.03.2012 23:47 | |
| | application.c | 11 KB 15.08.2009 15:17 | |
| | Bibliotheken | 6 KB 15.08.2009 22:18 | |
| | N 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 - 1997 | | |
| | Computer | | |
| | | | |
| | Netzwerk | | |
| | • | <u> </u> | |
| E Messages | Dateiname: "main c" "application c" | V Öffnen | > ₩ ₩ 🔀 |
| 🚔 Errors 📑 Warnings | | | |
| | Dateityp: C Compiler files (*.c) | Abbrechen | |
| | | | |

Bild 2.1.1-10: 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 2.1.1-06: Benennen der *.prj-Datei: AVR_PB_LED.prj => Speichern

Bild 2.1.1-09: Hauptfenster von CVAVR - zur Übersicht

- Bild 2.1.1-07: Configure Project Part 1 (das Menü bleibt zunächst noch sichtbar) => Add
- Bild 2.1.1-08: Neue Projekt-Struktur

Bild 2.1.1-10: C-Dateien dem Projekt bekannt machen => *. c-Dateien markieren => Öffnen

Bild 2.1.1-07: Configure Project => OK

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

| 📈 CodeVisionAVR - L:\AVR_PB_LED\AVR_PB_LED.prj |
|---|
| Ele Edit Search View Project Tools Settings Help |
| ┗ ┣ ♬ ▾ 腸 蔥 ff ff ថ ថ ff 🛍 🕲 ฿ 鼻 , ㅂ ལ 빓 ங ங 🏝 Ҳ 🖺 唇 左 左 〃 ※ 👒 ◎ ፣ ◎ ፣ 의 의 🖁 昏 告 兆 個 (分 , |
|] AA # # # # # # # . > 12 12 12 00 > 0 = = = = B B B B A . ▲ # ♥ = U . B A Ø . ? 💥 (Ø . |
| |
| E: Code Navigator V # X Code I V # X If Code I V # X Code VisionAVR If () { If () { Project AVR_PB_LED If () { If () { B application.c If () { If () { B Headers If () { If () { B ListFiles If () { If () { M AVR_PB_LED.asm If () { If () { B AVR_PB_LED.asm If () { If () { D AVR_PB_LED.asm If () { If () { D Other Files If () { If () { |
| Messages |
| Errors B Warnings |
| |
| 1:1 Insert |

Bild 2.1.1-11: Hauptfenster von CVAVR



Bild 2.1.1-12: 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 2.1.1-12** 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 (siehe Bild 2.1.1-12) und die "File Pane" verbreitert:

| 👫 CodeVisionAVR - L:\AVR_PB_LED\AVR_PB_LED.prj | × |
|--|-----|
| Eile Edit Search View Project Tools Settings Help | |
| ┗ ┣ ♂ • 目前回回回 6 1 圖圖圖• ∽ ~ 2 點點圖/目 昼 左左〃※ № ● • ● • 里 目标• 目标• | |
|] A 4 5 # A # A # . D 12 12 12 0 💊 🕕 🖹 🖹 🖹 🖺 🕼 🖏 . 🛦 🛎 🆘 🗏 💟 . 🔣 🍇 🚸 🥥 . ? 💥 🚳 . | |
| | |
| 🖹 Code Navigator 🛛 🗢 후 🖬 🔩 📿 🕮 다시VR_PB_LEDVAVR_PB_LED M | |
| Image: State | A A |
| 🗖 Messages 🗸 🗸 🖓 | × |
| 🖹 Errors 🖹 Warnings | |
| | |
| 1:1 Insert | |

Bild 2.1.1-13: 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. Als erstes wird die Gelegenheit genutzt, dem Projekt eine Kurzbeschreibung in den **Project Notes** voranzustellen. Es wird einfach ein Text in das dafür vorgesehene Datei-Feld eingetragen:



Bild 2.1.1-14: Kurzbeschreibung des Projektes

| CodeVisionAVR - L:\AVR_PB_ | LED\AVR_PB_LED.prj | | | | | |
|--|-------------------------------|--|-------|-------------|--|--|
| Be Edit Search Yew Project | Tools Settings Holp | | | | | |
| Do 2 · E # # d 6 6 | 10 Q A | > ● 문론 / 2019 ● 101:53 名 응용权 科 H. | | | | |
| 44 (1) (1) (0) (1) (1) (1) (1) (1) (1) (1) | POLESBERS | | | | | |
| | | | | | | |
| | | | | - | | |
| Code Novigner V V L | | 22P PK PLAmma | | I VANH HR | CEDANDRICATION C | |
| D Paizet AVR PB LED | E B Functions | Notes Inches G | | coplication | n: 🚨 | |
| - St Notes | - f0 opp_int(void) | 18/* | - | 1 0 /* | | |
| - cpalcetons | ⊢/C opp_set_vatchdog_prescole | | 1 | 2 | | warm - mil a |
| Men c | | 4 601 AWE C Projekt PB LED | | 4 | | amplication.c * VI.1 ************************************ |
| th 2 ListFiles | | 5 | - | 5 | | |
| - AvB_P8_UED.osm | | c main.c · VI.1 | • | ¢ | | |
| AVR.P8_LEDIM | | | 1 | 2 20 | Ander | |
| Char Fire | | 9 Projekt 601 Demonstration der Taster und LEDs auf dem Testboard | | 9 Ce | capilerr CodeVisionAVR | |
| | | 10 Sussetzliche Reader/ keine | | 10 CB | hip ATnegadd | |
| | | 11 Module.(*.c und *.b)./ application | | 11 De | 4008 J Juni 2009 | a second state and second |
| | | 17 Menoscipts Marborn Finatorna, Taster XI, XZ, 33 188 1801, 1802, 1803 | | 15 40 | ILOF AIVIN LUNIX (ALAIS) Des | tarent aur den Modul application von bib Juerar |
| | | 14 Compiler/ CodeVisionAVR | | 14 -1 | | |
| | | 15 Chip / Almegeld | | 10 | | |
| | | 16 Detue | | 16 1/- | Reader Report stationers | |
| | | 17 ALCOT | | 12 11. | neaver-rever ernroehen. | |
| | | 15 | | 10 #10 | nclude "application.h" | // Initialisierung der Anwendung |
| | | 21 Dieses Projekt stellt eine Applikation zur Demonstration der Taster (Pushbottons) und IXDr auf dem Testboard der. | | 20 //- | | |
| | | 21 Die Taster sind nicht entpreilt? | 20 | 21 | | |
| | | 22 23 Die 1851 det is Bebautend 200 und hei gedrachter Terte 51 515 | 011 | 2 11 | Globale Variablen: | |
| | | 24 Die 1832 ist is Nubepostens XIN und bei gefriedtier Taste X2 ADA | 011 | 24 1/- | | |
| | | 25 Die 1809 toggeit beim Seisetigen der Taste 59 von ADS nach SIB und ungekehrt | 011 | 25 907 | lattle 000 app_reset_source; | 0 |
| | | 26 | 0.0 | 26 1/- | | |
| | | 21 21 Ja der Needer-Dahei emlinsbins & kommen die Definitionen für die Testen und LADe meandert verdes. | 1010 | 28 11- | | |
| | | 29 Die Definitionen sind hier vie folgt festgelegt: | 0.0 | 23 1/- | Haupt-Funktion | |
| | | 30 | 0.0 | 30 //- | id any init/maid) | |
| | | 3) Komponenter Albegald-Fins: | - CH | 32 0 1 | id app_inic(void) | |
| | | 33 INC P27 (kans in accilention: h generation) | - CHP | 33 * | app_reach_source = NCUSE; | // Sickern des Mikrocontroller(MCU)-Status-Registers |
| | | 34 LED2 FC3 (kann in application.h generativerden) | 2011 | 26 | | // Dit7=3 => Reserviertes Dit stets 0 (micht belegt) |
| | | 35 ISD2 PC4 (kenn in application.h generator verden) | 2011 | 33 | | // Bitted -> Nexerviertex hit state 0 (since beiegt) // Bitted -> Deserviertex bit state 0 (since belent) |
| | | 36 37 141 Tanta | 2011 | 37 | | // Bit4=0 => Reserviertes Bit stets 0 (micht belegt) |
| | | 35 51 FB2 (kens in explication, h generative) | ×11 | 20 | | // DitJ #DRF -> Watchdog Reset Flag |
| | | 39 SJ FD4 (kann in application.h gesendert verden) | ×01 | 39 | | // Bib2 BORF -> Brown-out Reset Fing |
| | | 40 K3 PES (kenn in application.h gesendert warden) | × III | 40 | | // Biti Arter -> Actinui Music Flag |
| | | | XIII | 42 H | MCUSR = 0x00; | // Loeschen aller Reset Flags fuer neechsten RESET-Eyklus |
| | | 43 // Beader-Dateien einfbegen/ | No. | 42 | | |
| | | 4 // | -201 | 44 //- | Press Description (Repress and) in de | - Makabalan Rissa anananakalan |
| | | 45 finclude "application.h" | 20 | 4 11. | tores constrained (starting (starting take) for the | Producty Flate Assystemeters |
| | | | -SH | 47 #10 | rdef USE MATCHDOG | |
| | | 4t // Globele Veriablen: | 20 | 48 1 | ADICOR - NDCE NDEJ | // Enable Watching Reset (Watching Timor Control Register) -16mm) |
| | | 49 // | -011 | 42 2 200 | Set P | 0 |
| | | S0 D08 mg | 0.0 | 51 1/- | | |
| | | 52 // | -011 | 52 // | Setzen des voreingestellten System-Taktes nu | er vonn ein Quere als Taktpaber gevachlt vurde |
| | | 55 //- Euptprogram main | -0II | 50 41.0 | 10+1 USE CRISTAL CLOCK | // Abfrage, ob ein Guarz als Taktgeber gewachlt wurde |
| | | 54 // | 1011 | 22 812 | f (MCU CLOCK PREQUENCY - 16000000) | |
| | | Se Di Usata (seta (seta) | 0.0 | 56 C | CLEFR - CLEFCE, | // Enable Clock Presceler mit CLMPR = 0x00 (CLMPCE ist Bit7 = 0x00) |
| | | 57 // Setzen des System-Taktes (CPO-Clock), Ausschalten nicht benutzter Mardware-Nodale | - CH | 57 C | CLEPR = 0: | // Setam Clouk Prescaler, Division durch 1 (=1605s hei 1605s-Quers) |
| | | 30 spp_init(); // Initialisierung der Anvendung | - CHP | 54 Fel | 117 (MCD GLOCK PREQUENCY - SOCODO) | // Fachia Clock Brancelar put CIVIE a Aria (CIVET un Butt a Aria) |
| | | 50 // Tablelington (Dather (Dather) | - | 60 C | CLEFR - 1: | // Setze clock Prescaler, Division durch 2 (= ONTs bei 16002-Guerz) |
| | | • • • • • • • • • • • • • • • • • • • | C III | • | | |
| | | | | | | |
| Free Parts | | | | | | 0.44 |
| Trees Carwanada | | | | _ | | |
| | | | | | | |
| | | | _ | _ | | |

Bild 2.1.1-15: Die *. c-Dateien sind bereit zum Editieren (Bildvergrößerung)



Bild 2.1.1-16: Vergrößerung von Bild 2.1.1-15 - Projekt-Stand vor der Kompilierung

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 |
| | |
| | Executable Files: |
| | Object Files: |
| | Obj 8:: |
| | List Files: |
| | List |
| CodeVisionAVR - L:\AVR DB LED\AVR DB LED | Linker Files: |
| | |
| <u>File Ealt Search View Project Tools Settings Help</u> | r |
| | |
| 」 赫维 丝 44 ⁰ 株 林静 ↓ ↓ ↓ [*] * <u>C</u> ompile | F9 |
| 🕆 🚡 🖥 👔 🎼 😼 😼 🦉 Build Shift- | -F9 |
| 🐮: Code Navigator 🎽 Build All 🛛 Ctrl- | -F9 |
| CodeVisionAVR Stop Compilation | |
| 🖻 💾 Project: AVR_PB_ 🍆 Clean Up | |
| application c | |
| | |
| Headers Go to Next Error | F8 |
| 🗎 📅 List Files 📮 Go to Previous Error Ctrl- | -F8 |
| AVR_PB_LLE Go to Next Warning | F4 |
| AVR_PB_LE Go to Previous Warning Ctri- | -F4 |
| Cher Files | |
| | |
| 명 _월 <u>E</u> xport | |
| State Configure | V OK X Cancel ? Help |
| | |
| Dild 0.4.4.47. Aufauf des Konfigurations I | Nanija Bild 2.1.1.19. Canfigura Drajaat |

Bild 2.1.1-17: Autrut des Konfigurations-Menus



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





Bild 2.1.1.-19: Configure Project Part 3 - Code Generation

Hier können Einstellungen für den Chip und den Compiler zur Generierung der Ausgabe-Dateien vorgenommen werden. Die Einstellungen werden wie angegeben vorgenommen. Besonders zu beachten sind:

| Chip: ATMega88 | 3 |
|----------------------|---------------|
| Clock: 16 MHz | |
| Program Type: | Application |
| Data Stack Size: | 256 bytes |
| Heap Size: | 0 Bytes |
| Internal RAM Size | e: 1024 bytes |
| Bit Variable Size: | 16 |
| Use <u>G</u> PIOR>31 | nicht gesetzt |

File Output Format(s):

Formate der Ausgabe-Dateien (hier ist vorrangig das Format **HEX** für ein ausführbares Programm von Interesse).

| Configure Project AVR_PB_LED.prj | 1 |
|--|--|
| Files C Compiler Before Build After Build | Bild 2.1.120: Configure Project |
| Code Generation Libraries Messages Globally #define Paths | Part 4 - Messages |
| 🔽 Enable <u>W</u> arnings 🔗 | - |
| Warnings: | Das ist die Liste Messages, in der man ange- |
| □ constant out of range □ possible loss of precision | ben kann, welche Hinweise und Warnungen |
| Array index is out of range RAM location address is out of range | man erhalten möchte. |
| ✓ interrupt vector exceeds the allowable range ✓ constant not initialized | Das Setzen bestimmter Warnungen kann |
| ✓ suspicious pointer conversion | beim Testen sehr hilfreich sein. um unerreich- |
| ✓ macro was redefined ✓ function must have a return value | baren Code anzuzeigen oder davor zu war- |
| unused function was removed by the linker | nen, dass Variable deklariert sind, die gar |
| Intraction parameter name doesn't match its previous declaration Jocel variable was doclared but not referenced | nicht benutzt werden. |
| Incarvanable was declared, but not referenced Incarvanable set, but not used Incarvanable is used before its value is set. | |
| I local variable/constant was declared, but not referenced | |
| empty line | |
| v expression with possibly no effect | |
| ✓ shift result will be 0 ✓ overflow is possible in 8 bit addition | |
| ✓ overflow is possible in 8 bit multiplication ✓ overflow is possible in 8 bit shift left | |
| ✓ overflow is possible in 16 bit addition ✓ overflow is possible in 16 bit multiplication | |
| ✓ overflow is possible in 16 bit shift left ✓ possibly incorrect assignment | |
| ✓ pointless integer comparison ✓ controlling expression is constant | |
| | |
| | |
| | |
| ✓ QK X Cancel ? Help | |
| Configure Project AVR PB LED.prj | 1 |
| Files C Compiler Before Build After Build | |
| Code Generation Libraries Messages Globally #define Paths | Bild 2.1.121: Configure Project |
| #include paths (one per line): | Fait 5 - Faills |
| C\cvavr2\inc | Hier können neue Pfade für zusätzliche #in- |
| | clude- und library-Dateien angelegt werden: |
| | , |
| | #include paths (one per line): |
| | für globale Header-Dateien |
| | Library paths (one per line): |
| | für Bibliotheksdateien |
| | |
| Library naths (one par line): | |
| Choraeve2llib | Für die hier generierten AVR-Projekte wird |
| C. CACATE (III) | won dieser woglichkeit kein Gebrauch ge- |
| | macht. |
| | |
| | Mit OK wird das Menü verlassen. |
| | |
| | |
| | |
| | |
| | |
| | |
| <u>A</u> ✓ <u>O</u> K X <u>C</u> ancel ? <u>H</u> elp | |

2.1.2 Ein C-Projekt generieren

| L:\AVR_PB_LED | | | | |
|---|--|-------|------------------|------------------------|
| G → I:\AVR_PB_LED | A | | 👻 🔯 🗛 AVR_PB | LED durchsuchen |
| <u>D</u> atei <u>B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen | Neuer Ordner | | | = - 1 🕐 |
| ☆ Favoriten | Name 🔺 | Größe | Änderungsdatum | Тур |
| | Nicht angegeben (14) — | | | |
| E Desktop | 👢 Exe | | 11.10.2012 09:58 | Dateiordner |
| ALE | 👢 Linker | | 11.10.2012 09:58 | Dateiordner |
| Computer | 📜 List | | 11.10.2012 09:58 | Dateiordner |
| 💽 Netzwerk | 👢 Obj | | 11.10.2012 09:58 | Dateiordner |
| 😝 Systemsteuerung | application.c | 11 KB | 15.08.2009 14:17 | C compiler source file |
| 夏 Papierkorb | application.h | 36 KB | 15.08.2009 21:34 | H-Datei |
| | 🚨 AVR_PB_LED.prj | 5 KB | 11.10.2012 10:30 | CodeVisionAVR Proj |
| | AVR_PB_LED.prj~ | 4 KB | 11.10.2012 09:58 | PRJ~-Datei |
| | AVR_PB_LED.txt | 1 KB | 11.10.2012 10:30 | Text Document |
| | iomx8.h | 46 KB | 15.08.2009 15:08 | H-Datei |
| | macros.h | 4 KB | 15.08.2009 20:07 | H-Datei |
| | 📄 main.c | 6 KB | 15.09.2012 21:58 | C compiler source file |
| | switches.h | 4 KB | 15.08.2009 20:09 | H-Datei |
| | typedefs.h | 9 KB | 15.08.2009 20:13 | H-Datei |
| | | | | |
| 14 Elemente | | | | |
| 14 Elemente | | | 🔊 Comput | ter . // |
| | | | | |

Bild 2.1.2-01: Ausgangslage zur Kompilierung - Stand der Dateien

| CodeVisionAVR - L:\AVR_PB | _LED\AVR_PB_LED.prj | |
|--|---|---|
| File Edit Search View Project | : <u>T</u> ools <u>S</u> ettings <u>H</u> elp 1 11 왕 왕 종 종 [요 여 옷 빠 빠 | ● × ● 長 〒 〒 // 次 ▲ ●・●・● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● |
| M 単位の数数。 19 19 19 | | |
| 🕈 🖬 🕤 👔 😻 🕾 🗞 | 🖬 💀 📰 🖽 . | |
| E: Code Navigator ♥ ₩ E: Code VisionAVB | Code Information V 7 2 | PL_PB_LED/main.c |
| Project AVR_PB_LED | E-F0 Functions | des mainc 22 ♡ |
| application.c | 2 | |
| Headers | 4 | 601_AVR_C_Projekt_PB_LED ************************************ |
| B-BIListFiles DIAVR_PB_LED.asm | 6 | ************************************** |
| AVR_PB_LED.lst | 8 | |
| - 🖰 Other Files | 9 10 | Projekt 601 Demonstration der Taster und LEDs auf dem Testboard Zusaetzliche Header: keine |
| | 11 | Module.(*.c und *.h).: application Benoetigte Hardware: Testboard, Taster S1, S2, S3 und LED1, LED2, LED3 |
| | 13 | Version |
| | 15 | Chip: ATmega88 |
| | 16 | Autor |
| | 18 | |
| | 20 | Dieses Projekt stellt eine Applikation zur Demonstration der Taster (Pushbottons) und LEDs auf dem Testboard dar. Die Taster sind nicht entprellt! |
| | 22 23 | Die LED1 ist im Ruhezustand AUS und bei gedrueckter Taste S1 EIN |
| | 24 | Die LED2 ist im Ruhezustand EIN und bei gedrueckter Taste S2 AUS Die LED3 tograalt beim Betaetigen der Taste S3 von AUS nach EIN und umgekehrt |
| | 25 | Die 2000 coggett beim betaetigen der 1990e 30 von MOS Hach bie die die Amerika |
| | 27 | In der Header-Datei application.h koennen die Definitionen fuer die Tasten und LEDs gewendert verden. |
| | 29 30 | Die Definitionen sind hier wie folgt festgelegt: |
| | 31 | Komponente: ATmega18-Pins: *** LEDs |
| | 33 | LED1 PC2 (kann in application.h gesendert verden) |
| | 35 | LED3 PC4 (kann in application.h geaendert verden) |
| | 36 | *** Taste |
| | 38 | S1 PB3 (kann in application.h geaendert verden) S2 PB4 (kann in application.h geaendert verden) |
| | 40 | <pre>S3 PB5 (kann in application.h geaendert verden) */</pre> |
| | 42 | ///// // Rader_Dateien einflieren: |
| | 44 | |
| | 45 46 | <pre>#include "application.h"</pre> |
| | 47 48 | //// Globale Variablen: |
| | 49 | // |
| | 51 | // |
| | 53 | //Hauptprogramm main |
| | 55 | □ void main (void) |
| | 56 | { // Setzen des System-Taktes (CPU-Clock), Ausschalten nicht benutzter Hardware-Module |
| | 58 | app_init(); // Initialisierung der Anvendung |
| | 60 | // Initialisieren der Tasten (Pushbuttons) S1_INIT(); |
| | 62 | S2_INIT(); S3_INIT(); |
| | 64 | - // Initialisieren der LEDs |
| | 66 | LED1_DDR = LED1_BIT; // Setze LED1-Pin auf Ausgabe |
| | 68 | LED3_DDR = LED3_BIT; // Setze LED3-Pin auf Ausgabe |
| | 69 70 | // Alle LEDs 3-mal aufleuchten lassen ("Lampen-Test") |
| | 71 72 | for (n = 3; n; n) |
| | 73 | LED1_PRT = LED1_BIT; // LED1 auf EIN LED2_PRT = LED2_BIT; // LED2 auf EIN |
| | 75 | LED3_PRT = LED3_BIT; // LED3 auf EIN |
| | 77 | delay_ms(250); // 250 ms Verzoegerung (1/4 Sekunde) |
| | 79 | LED1_PRT 4= ~LED1_BIT; // LED1 auf AUS |
| | 81 | LED3_PRT 6= ~LED3_BIT; // LED3 auf AUS |
| | 82 | delay_ms(200); // 200 ms Verzoegerung (1/5 Sekunde) |
| | 84 | - <u>D</u> |
| | 86 | // |
| | 88 | while (true) // Endlosschleife |
| | 90 | (// Eenn Teste Si gedrueckt wird, dann leuchtet IED1 |
| | 91 | if (S1_FRESSED()) |
| | 93 | LED1_PRT = LED1_BIT; // LED1 auf EIN |
| | 95 | i cise |
| | 97 | E (LED1_FRT &= ~LED1_BIT; // LED1 auf AUS |
| | 99 | |
| | 101 | <pre>// Wenn Taste S2 gedrueckt wird, dann schaltet LED2 auf AUS if (S2 FRESSED())</pre> |
| | 102 | E (LED2 PRT 4= ~LED2 BIT; // T.RD2 and AIG |
| | 104 | |
| | 106 | |
| | 108 | DEDE_FRI (- LEDZ_DII) // LEDZ GUT EIN |
| | 110 | // Wenn Taste S3 gedrueckt wird, dann toggle LED3 (EIN-AUS-EIN-AUS) |
| | 112 113 | □ 11 (S3_PRESSED()) □ (|
| | 114 | LED3_FRT ^= LED3_BIT; // Toggle LED3 |
| | 116 | S3_WAIT_NOT_PRESSED(); // Warte bis Taste S3 losgelassen wird) |
| | 118 | |
| | 120 | [™] 7/ |
| | | |
| Messages | | ◆ 平 図 |
| Errors Warnings | | |
| | | |
| | | |
| 110-00 | ada Tafarmatian musika. | the Barras ast Complet ust |
| Insert Co | oue information may be incomplete, as | are ne was not complete yet |

Bild 2.1.2-02: Ausgangslage zur Kompilierung - Hauptprogramm (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.cincl.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) | Dieser Schritt 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) | Dieser Schritt produziert ein neues Assembler-Quell-Pro- gramm mit der Erweiterung *.asm wobei nur die veränderten Quell-Dateien neu kompiliert werden. Wenn keine Fehler er- kannt wurden, wird automatisch der Atmel-AVR-Assembler aufgerufen, der die eben erzeugte .asm -Datei assembliert. Das Ausgabe-File ist eine ausführbare Programm-Datei mit dem For- mat *.hex . |

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:

| KCodeVisionAVR - L:\AVR_PB_LE | D\AVR_PB_LED.prj | |
|--|------------------------------------|---|
| <u>File Edit Search View</u> Project T | ools <u>S</u> ettings <u>H</u> elp | |
| 📘 🗁 🥭 📲 🗟 🗿 💞 🖻 Check S | Syntax | ■ 🛍 X 🗎 💩 症 症 ∥ ೫ ∞ ⊨ ∞ ⊤ ೨, Ջ 🖶 🖶 ≪ 📳 ₭) . |
| 🏘 🖄 🏥 🚜 🏥 🏥 📜 🏪 <u>C</u> ompile | e F9 | 🛐 🐏 🖏 . 💩 🛎 🐐 🗏 💟 . 🛛 🕵 🍇 🤻 🍕 . 🛛 ? 💥 🎯 . |
| 中 🗄 🗗 🏾 🏦 😽 😼 🤷 Build | Shift+F9 | |
| 🎏: Code Navigator 🧏 Build Al | Ctrl+F9 | _:\AVR_PB_LED\main.c |
| CodeVisionAVR | ompilation | Notes main.c 🛛 |
| Project: AVR_PB_ Scient U | p | 1 0 /* |

Bild 2.1.2-03: 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 2.1.2-04: Nach der Kompilierung mit Build All (Ctrl+F9) (Bildvergrößerung)



Bild 2.1.2-05: Compiler-Meldungen

Bild 2.1.2-06: Assembler-Meldungen

Wenn man tiefer in die Materie eingestiegen ist, so geben 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.

2.2 Dateistruktur eines C-Projektes

Das folgende Bild stellt die ineinander greifende Struktur des oben beispielhaft gewählten C-Projektes dar. Die Verschachtelung ergibt sich aus den Quell-Dateien. Weitere Informationen hierzu folgen in Teil 503 - Der Preprozessor, Teil 504 - Syntax der C-Programmiersprache und Teil 505 - Modularer Aufbau der AVR-Projekte.

C-Dateien: main.c, application.c

Header-Dateien:

application.h, typedefs.h, iomx8.h, macros.h, switches.h

Globale Header-Dateien vom CVAVR:

mega88.h, delay.h, stdio.h, starg.h, stdlib.h, string.h, ctype.h, limits.h



Bild 2.2-01: Datei-Struktur des Projektes AVR_PB_LED (Bildvergrößerung)

14.03.2020

Am Ende der Kompilierung sieht der Projekt-Ordner schon viel verwirrender und viel gefüllter aus:

| 📕 L:\AVR_PB_LED | | | | | |
|---|--|-------|------------------|----------------------------|--|
| | | | 💌 🛃 🗛 AVR_ | _PB_LED durchsuchen | |
| <u>D</u> atei <u>B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | | |
| Organisieren ▼ Freigeben für ▼ Brennen Neuer C | ordner | | | iii 🝷 🗍 😧 | |
| Favoriten | Name 🔺 | Größe | Änderungsdatum | Тур | |
| | Nicht angegeben (24) – | | | | |
| Desktop | 👢 Exe | | 11.10.2012 09:58 | Dateiordner | |
| Bibliotheken | 👢 Linker | | 11.10.2012 09:58 | Dateiordner | |
| Computer | 👢 List | | 11.10.2012 09:58 | Dateiordner | |
| Netzwerk | 👢 Obj | | 11.10.2012 09:58 | Dateiordner | |
| 📴 Systemsteuerung | application.c | 11 KB | 15.08.2009 14:17 | C compiler source file | |
| Papierkorb | application.cci | 1 KB | 11.10.2012 10:46 | CCI-Datei | |
| | application.h | 36 KB | 15.08.2009 21:34 | H-Datei | |
| | application.hci | 1 KB | 11.10.2012 10:46 | HCI-Datei | |
| | 📓 AVR_PB_LED.prj | 6 KB | 11.10.2012 10:47 | CodeVisionAVR Project File | |
| | AVR_PB_LED.prj~ | 6 KB | 11.10.2012 10:41 | PRJ~-Datei | |
| | 🔀 AVR_PB_LED.txt | 1 KB | 11.10.2012 10:30 | Text Document | |
| | iomx8.h | 46 KB | 15.08.2009 15:08 | H-Datei | |
| | iomx8.hci | 11 KB | 11.10.2012 10:46 | HCI-Datei | |
| | macros.h | 4 KB | 15.08.2009 20:07 | H-Datei | |
| | macros.hci | 1 KB | 11.10.2012 10:46 | HCI-Datei | |
| | 📄 main.c | 6 KB | 15.09.2012 21:58 | C compiler source file | |
| | switches.h | 4 KB | 15.08.2009 20:09 | H-Datei | |
| | switches.hci | 1 KB | 11.10.2012 10:46 | HCI-Datei | |
| | typedefs.h | 9 KB | 15.08.2009 20:13 | H-Datei | |
| | typedefs.hci | 4 KB | 11.10.2012 10:46 | HCI-Datei | |
| | main.cci | 1 KB | 11.10.2012 10:46 | CCI-Datei | |
| | AVR_PB_LED.fct | 1 KB | 11.10.2012 10:46 | FCT-Datei | |
| | AVR_PB_LED.cbi | 1 KB | 11.10.2012 10:46 | CBI-Datei | |
| | AVR_PB_LED.cof | 4 KB | 11.10.2012 10:46 | COFF symbolic debug file | |
| 24 Elemente | | | | | |
| 24 Elemente | | | 🔊 Com | puter // | |

Bild 2.2-02: Die Quell-Dateien und während der Kompilierung angelegte Zwischen-Dateien

| L:\AVR_PB_LED\Linker | | | | |
|---|--|-------|------------------|-----------------|
| G V L:\AVR_PB_LED\Linker | | | 👻 🛃 🛛 Linker d | lurchsuchen 🛛 🦻 |
| <u>Datei B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen Neuer O | rdner | | | i 🕶 🗍 😧 |
| | Name 🔺 | Größe | Änderungsdatum | Тур |
| | Nicht angegeben (14) — | | | |
| E Desktop | application.a | 14 KB | 11.10.2012 10:46 | A-Datei |
| Bibliotheken | application.o | 33 KB | 11.10.2012 10:46 | O-Datei |
| Computer | ctype.al | 9 KB | 11.10.2012 10:46 | AL-Datei |
| (Netzwerk | ctype.ol | 23 KB | 11.10.2012 10:46 | OL-Datei |
| 👺 Systemsteuerung | 📄 main.a | 10 KB | 11.10.2012 10:46 | A-Datei |
| 🧕 Papierkorb | 📄 main.o | 33 KB | 11.10.2012 10:46 | O-Datei |
| | math.al | 34 KB | 11.10.2012 10:46 | AL-Datei |
| | math.ol | 45 KB | 11.10.2012 10:46 | OL-Datei |
| | stdio.al | 37 KB | 11.10.2012 10:46 | AL-Datei |
| | stdio.ol | 46 KB | 11.10.2012 10:46 | OL-Datei |
| | stdlib.al | 33 KB | 11.10.2012 10:46 | AL-Datei |
| | stdlib.ol | 51 KB | 11.10.2012 10:46 | OL-Datei |
| | string.al | 40 KB | 11.10.2012 10:46 | AL-Datei |
| | string.ol | 37 KB | 11.10.2012 10:46 | OL-Datei |
| | | | | |
| 14 Elemente | | | | |
| 14 Elemente | | | 🔊 Compu | ter // |

Bild 2.2-03: Die Linker-Dateien

| L:\AVR_PB_LED\List | | | | | | |
|--|---|------------------------|--|---|--|--|
| COV 1. L:\AVR_PB_LED\List List durchsuchen | | | | | | |
| <u>D</u> atei <u>B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen Neue | er Ordner | | | = 🛛 🖸 | | |
| ☆ Favoriten | Name 🔺 | Größe | Änderungsdatum | Тур | | |
| Desktop Biblotheken ALE Computer Netzwerk Systemsteuerung Papierkorb | Nicht angegeben (3) AVR_PB_LED.asm AVR_PB_LED.lst AVR_PB_LED.map | 40 KB 70 KB 2 KB | 11.10.2012 10:46 11.10.2012 10:46 11.10.2012 10:46 | Assembler source file List file C compiler map file | | |
| 3 Elemente 3 Elemente | | | 🕵 Comput | er // | | |

Bild 2.2-04: Die List-Dateien

| L:\AVR_PB_LED\Obj | | | | | × | |
|---|--|-------|------------------|--------------------------|----|--|
| 🔄 🕞 📙 L:\AVR_PB_LED\Obj Obj durchsuchen 🗾 😥 | | | | | | |
| Datei <u>B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen Neuer | Ordner | | | III 🕶 🗍 😧 | | |
| ★ Favoriten | Name 🔶 | Größe | Änderungsdatum | Тур | | |
| | Nicht angegeben (10) | | | | _ | |
| Desktop | application.str1 | 1 KB | 11.10.2012 10:46 | STR1-Datei | | |
| al F | application.sym1 | 2 KB | 11.10.2012 10:46 | SYM1-Datei | | |
| Computer | application.sym2 | 1 KB | 11.10.2012 10:46 | SYM2-Datei | | |
| Ketzwerk | application.sym3 | 1 KB | 11.10.2012 10:46 | SYM3-Datei | | |
| 👺 Systemsteuerung | AVR_PB_LED.cof | 4 KB | 11.10.2012 10:46 | COFF symbolic debug file | | |
| 💱 Papierkorb | AVR_PB_LED.obj | 2 KB | 11.10.2012 10:46 | Atmel object file | | |
| | main.str1 | 1 KB | 11.10.2012 10:46 | STR1-Datei | | |
| | main.sym1 | 1 KB | 11.10.2012 10:46 | SYM1-Datei | | |
| | main.sym2 | 1 KB | 11.10.2012 10:46 | SYM2-Datei | | |
| | main.sym3 | 1 KB | 11.10.2012 10:46 | SYM3-Datei | | |
| | | | | | _ | |
| | | | | | | |
| 10 Elemente | | | | | | |
| 10 Elemente | | | 🔊 Cor | nputer | // | |

Bild 2.2-05: Die Obj-Dateien

| L:\AVR_PB_LED\Exe | | | | |
|---|---|--------------|--------------------------------------|--|
| G V L:\AVR_PB_LED\Exe | | | 💌 🛃 🛛 Exe du | rchsuchen 🔎 |
| <u>] D</u> atei <u>B</u> earbeiten <u>A</u> nsicht E <u>x</u> tras <u>?</u> | | | | |
| Organisieren 🔻 Freigeben für 🔻 Brennen Neuer C | Ordner | | | i 🕶 🚺 😧 |
| ★ Favoriten | Name 🔺 | Größe | Änderungsdatum | Тур |
| Desktop Bibliotheken ALE Computer Netzwerk Systemsteuerung Systemsteuerung Papierkorb | Nicht angegeben (2) — AVR_PB_LED.hex AVR_PB_LED.rom | 2 KB 3 KB | 11.10.2012 10:46 11.10.2012 10:46 | HEX-Datei Atmel FLASH contents file |
| 2 Elemente | | | | <u> </u> |
| 2 Elemente | | | 🔊 Compu | iter // |

Bild 2.2-06: Die Exe-Dateien

Die Datei **AVR_PB_LED.hex** ist schließlich die Datei, die in den Controller geflasht und dann zum Laufen gebracht wird.

2.3 Einbindung von AVR Studio in den CVAVR

Hinweis: Wer gleich ins "Volle" gehen will, der kann bei 2.5 C-Compiler-Optionen (Arbeiten mit dem CVAVR) gleich weiterarbeiten!

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:



Bild 2.3-01: Debugger-Eintrag in CVAVR vornehmen



Bild 2.3-02: 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 nicht an der vorgesehenen Lokalisation installiert worden sein, so kann sie mit der "File-Such-Funktion" gefunden werden:

| <mark>₭</mark> Debugger Dire | ctory and FileName | | | | | x |
|------------------------------|---------------------|-----------------|----------|------------------|-------------|---|
| Suchen in: | 👢 AvrStudio4 | | | ▼ 🕝 👂 📂 🎞 ▼ | | |
| C. | Name 🔺 | ▼ Größe | | Änderungsdatum | ▼ Тур | _ |
| | Nicht angegeben (8) |) | | | | |
| Zuletzt besucht | 👢 bin | | | 06.09.2012 21:49 | Dateiordner | |
| | 📜 di | | | 06.09.2012 21:50 | Dateiordner | |
| Dackton | 📕 edit | | | 06.09.2012 21:49 | Dateiordner | |
| Desktop | l resources | | | 06.09.2012 21:49 | Dateiordner | |
| | 📜 skins 🔬 | | | 06.09.2012 21:49 | Dateiordner | |
| Bibliotheken | * AVRStudio.exe | | 3.223 KB | 25.08.2011 21:05 | Anwendung | |
| | PluginManager.exe | | 55 KB | 25.08.2011 20:58 | Anwendung | |
| | xmlconvert.exe | | 284 KB | 25.08.2011 20:59 | Anwendung | |
| Computer | | | | | | |
| <u>(</u> | | | | | | |
| Netzwerk | | | | | | |
| | • | | | | | ▶ |
| | Datei <u>n</u> ame: | RStudio.exe | AS . | • | Öffnen | |
| | Dateityp: Exe | ecutables (*.ex | æ) | • | Abbrecher | ı |

Bild 2.3-03: 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.

2.4 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.prj | | | | | | | |
|--|---|-------------|---|--|--|--|--|
| <u>File Edit Search View Project</u> | Tools Settings Help | | | | | | |
| 📔 🗁 🙄 📲 🗑 🗿 🗂 🔞 🖒 | & Code <u>W</u> izardAVR | Shift+F2 | 🛍 X 📔 💩 ऌ ऌ ∥ Ж 💁 🎯 • ∞ • 의 의 🚇 🖶 🕊 🖶 🙌 . | | | | |
| 桷 鸟 色 峙 鼎 鼎 - 🖻 👎 🔧 | 🐞 <u>D</u> ebugger 🔊 | Shift+F3 | 😘 🖏 . 💩 🗯 🖘 🗏 💟 . 🔣 🍇 🐗 🍕 . ? 💥 🌍 . | | | | |
| ● 18 1 18 18 18 18 18 18 | Schip <u>P</u> rogrami | Shift+F4 | | | | | |
| 🐮: Code Navigator 🛛 💙 🐺 🛛 | 🗏 Terminal | Shift+F5 | •••• | | | | |
| CodeVisionAVR | LCD Vision | Shift+F6 | | | | | |
| 日日 ● Project AVR_PB_LED ● 予 ● application.c ● 予 Warnings | <u>Configure</u> <u>42</u> <u>//</u> <u>43</u> // Header-Da | teien einfu | legen: | | | | |

Bild 2.4-01: 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.

| 1 | AVR | Studio | | | | | | | _ [| |
|--------|-------|-----------------|---------------|-------------------|--------------|-----------|---------|-------|----------|----------------|
| Fil | le Pr | oject Build Vie | w Tools De | bug Help | | | | | | |
| | . L | New File | Strg+N | i 🤊 🕫 🔁 🙀 | : <u>M</u> 2 | 6 🏂 🏂 🏂 🕯 | e e i 🛲 | ¥ 🔋 | ait | 周 |
| 1 | | Open File | Strg+O | 上 木 : 65 68 | * | ≦_2 AUTO | | | | |
| | | Close | V | | | I/O View | | | | - × |
| | H | Save | Strg+S | | | 3] - 😣 | | - | → | |
| | | Save As | | | | Name | | | Value | |
| 4 | Ø | Save All | | | | | | | | |
| i i jo | 4 | Print | Strg+P | | | | | | | |
| Ν δ | | Print Setup | - | | - × | ▲ | | , | | |
| ŝ | | | | | | Name | Address | Value | Bits | |
| | | Exit | | | | | | | | |
| | | | | | | | | | | |
| | | | | | • | | | | | |
| E | Build | 🕕 Message 🖣 | Find in Files | 7 Breakpoints and | Trace | • | | | | ► |
| | | | | | | | | | CAP | NI / |



| 🐞 Öffnen | | | | × |
|---|---------------|------------------|----------------------------|-------------|
| E:\AVR_PB_LED | AN | | VR_PB_LE | D durchsu 🔎 |
| Organisieren 🔻 Neuer Ord | Iner | | 8 | - 🗌 📀 |
| Name 🔺 | Größe | Änderungsdatum | Тур | |
| Nicht angegeben (25) —— | | | | |
| 👢 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 | |
| 📄 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 | |
| Dateina | me: AVR PB II | ED.cof | ▼ All Files (*.*) | |
| | 1 | - | Öffnen | Abbrechen |

Bild 2.4-03: 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 | | | | × |
|---|---------------------|------------------|--------------------|--------|----------|
| | AN A | | VR_PB_LED durchsue | chen | 2 |
| Organisieren 🔻 Neuer Ord | Iner | | | | 0 |
| Name 🔺 | Größe | Änderungsdatum | Тур | | |
| Nicht angegeben (4) | | | | | |
| 👢 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 | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| Dateiname: AVR_I | PB_LED_cot.aps | TE. | | | |
| Datei <u>t</u> yp: AVR S | tudio Project Files | s (*.aps) | | | • |
| | | | | | |
| Ordner ausblenden | | | <u>Speichern</u> | Abbrec | hen |

Bild 2.4-04: 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 Simulator AVR Simulator 2 ICE200 ICE40 ICE50 JTAG ICE JTAGICE mkll | Device: ATmega64C1 ATmega64M1 ATmega8515 ATmega8535 ATmega88A ATmega88A ATmega88A ATmega8AA ATmega8AA ATmega8AA ATmega8HVA ATmega8U2 |
| | C Open platform | n aptions next time debug mode is entered |

Bild 2.4-05: 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:

| 😻 AVR Studio - | L:\AVR_PB_LED\main | I.C | | | | | |
|------------------------|--------------------------------------|--------------|--|----------------------------|-------------------------|---|----------------------------|
| Eile <u>P</u> roject B | uild <u>E</u> dit ⊻iew <u>T</u> ools | <u>D</u> ebu | ig <u>W</u> indow <u>H</u> elp | | | | |
| | U 🖁 🖻 🖁 · | 36 | " 🔁 🙀 i 🙆 🦽 🏂 🧏 🦉 | • £ E 66 X 1 | I ar 🕹 🕨 💷 🗄 🗉 🖸 | * 1 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 | .) 🗄 🧶 😓 🔣 🔤 🕷 |
| Trace Disabled | - 🖄 💥 🕂 👱 | 工 | 1 🗰 🧰 👫 🖓 💷 | | | | |
| Processor | ~ × | B | L:\AVR_PB_LED\main.c | | _ 🗆 🗙 | I/O View | ▼ × |
| Name | Value | | //Hauptprogramm main- | | | 📑 🕄 - 🧞 | ANALOG_COMPARA1 - |
| Program Cou | 0x000076 | | // | | = | Name | Value |
| Stack Pointer | 0x04FF | | { | | | ∃ DAD_CONVE | RTER |
| Y pointer | 0x0202 | | // Setzen des System- | Taktes (CPU-Clock |), Ausschalten nicht b | ■ DANALOG_C | OMPARATOR |
| Z pointer | 0x0004 | | app_init(); | | // Initialisi | I CPU | |
| Cycle Counter | 6287 | | // Initialisieren der | Tasten (Pushbutt | ons) | | |
| Frequency | 4.0000 MHz | | S1_INIT(); S2_INIT(); | | | | |
| Stop Watch | 1571.75 us | | S3_INIT(); | | | Port B Dat | ta Register 0x00 |
| SREG | | | // Toitiliciono dos | LED- | | Port B Dat | ta Direction Register 0x00 |
| Registers | 0.01 | | LED1 DDR = LED1 BIT: | LEDS | // Setze LED1 | Port B Inpu | ut Pins 0x00 |
| R00 | 0x01 | | LED2_DDR = LED2_BIT; | | // Setze LED2 | 🖃 🔁 PORTC | |
| R01 | 0x00 | | LED3_DDR = LED3_BIT; | | // Setze LED3 | Port C Dat | ta Register 0x00 |
| R03 | 0x00 | | // Alle LEDs 3-mal au | fleuchten lassen | ("Lampen-Test") | Port C Dat | a Direction Register 0x00 |
| R04 | 0x00 | | for $(n = 3; n; n)$ | | | Port C Inpu | ut Pins 0x00 |
| R05 | 0x00 | | LED1 PRT I= LED1 BT | Т: | // LED1 auf F | | |
| R06 | 0x00 | | LED2_PRT = LED2_BI | Γ; | // LED2 auf E | | |
| R07 | 0x00 | | LED3_PRT = LED3_BI | Γ; | // LED3 auf E | | INTER 1 |
| R08 | 0x00 | | delav ms(250); | | // 250 ms Ver: | I IMER COL | UNTER 2 |
| R09 | 0x00 | | ,, | | | 🗉 🚉 TWI | _ |
| R10 | 0x00 | | LED1_PRT &= ~LED1_B LED2 PPT &= ~LED2 B | IT; IT. | // LED1 auf Al | 🗉 🗄 USART0 | |
| R11 P12 | 0x00 | | LED3_PRT &= ~LED3_B | IT; | // LED3 auf A | 🗉 🛱 WATCHDO | G |
| R12 R13 | 0x00 | | 1.1. (200) | | | | |
| R14 | 0x00 | | delay_ms(200); } | | // 200 ms Ver: | | |
| R15 | 0x00 | | 1 | | | | |
| R16 | 0x00 | | // | | | | |
| R17 | 0x00 | | // Hauptschieffe: | | | | |
| R18 | 0x00 | | while (true) | | // Endlosschl | | |
| R19 | 0x00 | | { // Wenn Taste S1 re | drueckt wird dan | n leuchtet IED1 | | |
| R20 | 0x00 | | if (S1_PRESSED()) | | | Name | Address Value Bits |
| R21 | 0x00 | | { | DIT. | COLEDIANS F | P DDRB | 0x04 (0x24) 0x00 |
| R22 R23 | 0x00 | | LEDI_PRI = LEDI_ | 511; | // LEDI auf E | | |
| R24 | 0x00 | | else | | | | |
| R25 | 0x00 | | { IFD1 PPT &= ~IFD1 | BIT· | // IED1 auf & | | |
| R26 | 0x02 | | } | | ,, LEDT dur II | PORTC | 0x08 (0x28) 0x00 |
| R27 | 0x02 | | (Wang Tracks CO. as | dence where we have do not | | | . , |
| R28 | 0x00 | | if (S2 PRESSED()) | arueckt wird, dan | n schaltet LED2 auf AU. | | |
| R29 | 0x02 | | { | | | | |
| R30 | UXU4 | | LED2_PRT &= ~LED2 | _втг; | // LED2 auf Al | | |
| Rol | UXUU | | else | | | | |
| | | | { | DIT. | // LED2 and E | | |
| | | | LEDZ_PRI = LEDZ_ | 511; | // LED2 auf E | | |
| | | | // Wenn Taste S3 ge | drueckt wird, dan | n toggle LED3 (EIN-AUS | | |
| | | | if (S3_PRESSED()) | | | | |
| | | | | DIT. | (/ Toggio IED | | |
| | | | LEDJ_FRI - LEDJ_ | JII., | // Toggie LED - | | |
| | | 凹 | | | | | |
| | | 🗈 r | L:\AVR_PB_LED\main.c | | 4 Þ | | |
| Message | | | | | ▼ X | | |
| Loaded plugin | STK500 | | | | | 1 | |
| AVR Simulator | Please wait while configur | rina sir | mulator | | | | |
| AVR Simulator: | ATmega88 Configured O | K | ······ | | | | |
| Loaded object | file: L:\AVR_PB_LED\AVR | _PB_L | ED.cof | | | | |
| | | _ | | | | | |
| | | | | | <u> </u> | U | |
| 🖾 Build 🛄 Mes | sage 🖓 Find in Files | Break | cpoints and Tracepoints | | | | |
| | | | | | ATmega88 AVR Simulat | or Auto Stopped 🤅 | Ln 58, Col 1 CAP NUM OVR |

Bild 2.4-06: 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 **Debug** 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 Projektname** wieder unkompliziert geladen werden.

| 🤹 AV | 😸 AVR Studio | | | | | | | | | | |
|--|--------------|-----------------|---|--|--|--|--|--|--|--|--|
| File Project Build View Tools Debug Help | | | | | | | | | | | |
| : L1 | | Project Wizard | 🚨 🖆 🥂 🖫 🔁 🙀 🗄 🚣 🏄 🥻 🏂 🖉 📰 | | | | | | | | |
| : Trace | | New Project | 🕂 上 工 ! 🗰 📾 🗰 🦌 🖓 💷 🛛 | | | | | | | | |
| | | Open Project | | | | | | | | | |
| | 9 | 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 2.4-07: 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.

Es gibt jede Menge Möglichkeiten, Neues zu entdecken und auszuprobieren!

Nur Mut- es kann nichts Schlimmes passieren!

Der Mikrocontroller ist geduldig !!!

2.5 C-Compiler-Optionen (Arbeiten mit dem CVAVR)

Wenn das Projekt trivial ist (so wie hier) und beim Kompilieren keine Errors aufgetreten sind, kann man davon ausgehen, dass auch der Test-Lauf auf dem Testboard fehlerfrei verläuft.

Es sei denn: Es hat sich ein **LOGISCHER FEHLER** eingeschlichen, den der Compiler natürlich nicht bemerken kann! Dieser wird aber beim Test-Lauf bemerkt, weil die Funktion nicht so funktioniert wie sie soll.

Um das schnell herauszufinden, dafür ist das Testboard ja da!

Es muss also die Datei AVR_PB_LED.hex ist den Mikrocontroller übertragen - geflasht - werden. Der Flash-Vorgang sei hier als eine Kopie des Abschnitts 6.5 Flashen eines C-Programms in ein Mikrocontroller ATmega88 aus dem Teil 206 C-Compiler und AVR Studio wiederholt:

Zunächst wird das Testboard beschaltet so wie es auch im Abschnitt **1.2 Beschaltung** im **Teil 601 - PB_LED** beschrieben wird:



Bild 2.5-01: 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 2.5-02: 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 | 1 |
| Erase Device | |
| Erase device before flash programming Verify device after programming | |
| Flash C Use Current Simulator/Emulator FLASH Memory | |
| Input HEX File | |
| Program Verify Read | |
| EEPROM © Use Current Simulator/Emulator EEPROM Memory © Input HEX File | |
| Program Verify Read | |
| ELF Production File Format | |
| Input ELF File: | |
| Save From: FLASH EEPROM FUSES LOCKBITS Fuses and lockbits settings | |
| Program Save saving to ELF | |
| | |
| Detecting on 'USB' AVRISP mkll with serial number 0000A0012825 found. Getting isp parameter SD=0x05 OK | - |

Bild 2.5-03: Objektdatei (*.hex-File) suchen

| 😻 Öffnen | | | × |
|---|-------|------------------|----------------------|
| | | 💌 🔯 🛛 Exe durcl | hsuchen 🛛 🔎 |
| Organisieren 🔻 Neuer Ordner | | | = |
| E 🗇 Sicherungs-LW 100 📥 Name * | Größe | Änderungsdatum | Тур |
| | | | |
| 🖂 🥪 KINGSTON (L:) 🛛 🖉 AVR_PB_LED.hex | 2 KB | 23.09.2012 19:49 | HEX-Datei |
| | | | |
| Exe Linker List Obj Netzwerk Systemsteuerung Papierkorb | | | |
| | | | |
| Dateiname: AVR_PB_LED.hex | | Flash Intel He | x Files (*.hex;*.i 💌 |

Bild 2.5-04: Objektdatei öffnen

| AVRISP mkII in ISP mode with ATmega88 Main Program Fuses LockBits Advanced HW Settings HW Info Auto | _ 🗆 × |
|--|-------|
| Device Erase Device | |
| Erase device before flash programming Verify device after programming | |
| Flash C Use Current Simulator/Emulator FLASH Memory | |
| Input HEX File L:\AVR_PB_LED\Exe\AVR_PB_LED.hex | |
| Program Verify Read | |
| EEPROM Ourrent Simulator/Emulator EEPROM Memory Input HEX File | |
| Program Verify Read | |
| ELF Production File Format | |
| Input ELF File: | |
| Save From: V FLASH V EEPROM V FUSES LOCKBITS Fuses and lockbits setti | ngs |
| Program Save saving to ELF | e |
| Detecting on 'USB' | |
| Getting isp parameter SD=0x05 OK | - |

Bild 2.5-05: Programm flashen ==> und das Programm testen ==> alles O.K.

Und das Objekt-Programm ist immer wieder schnell geändert und schnell in den Controller geflasht, so dass dieser Test schnell und oft gestartet werden kann - bis die **LOGIK** stimmt!

CVAVR wird erneut aufgerufen. Der Compiler hat sich die "alte" Einstellung gemerkt, so dass gleich mit dem Projekt **AVR_PB_LED** weiter gearbeitet werden kann. Sollte inzwischen an einem anderen Projekt gearbeitet worden sein, so ist das Projekt über seine Projekt-Datei (hier: **AVR_PB_LED.prj**) schnell wieder mit seinen ganzen Dateien geladen.

Es wird angenommen, dass das Programm geändert wurde - WIE, das sei an dieser Stelle nicht von Bedeutung.



Bild 2.5-06: Aufruf des veränderten Projektes AVR_PB_LED



Bild 2.5-07: Neu-Kompilierung

Es wird ja angenommen, dass gerade eine Veränderung in der main.c vorgenommen wurde, so dass eine Neu-Kompilierung (mit allen Schritten) notwendig wird. Wenn keine Verschlimmbesserungen eingebaut wurden, dann werden wieder die gleichen Listen und Warnungen angezeigt, wie sie die Bilder **Bild 2.1.2-04, -05** und **-06** zeigen.

Bei der Neu-Kompilierung wurde auch eine neue AVR_PB_LED.hex erzeugt, die vom AVR Studio in den Mikrocontroller geflasht werden muss. Unter der Voraussetzung, dass das Testboard einschließlich der benötigten Ressourcen (dazu gehören auch die Pin-Verbindungen!) über den USBprog mit dem PC verbunden ist, wird AVR Studio gestartet und die logische Verknüpfung zwischen den Projekt-Programm AVR_PB_LED.hex auf dem PC und dem Mikrocontroller auf dem Testboard hergestellt:

| AVR Studio | | | | | | | | | | | | | |
|----------------|--|--|---------------|--------------|--|-----------|--|---------|------------|---|---|------------|---|
| Eile Project | <u>B</u> uild <u>V</u> iew | <u>T</u> ools | <u>D</u> ebug | <u>H</u> elp | | | | | | | | | |
| i 🗋 📂 🗔 🕯 | 1 U X I | 124 | n) (m. | Z 🖗 | ÷ #4 | 16 % | % X | る事 | F : | # | 🍵 🛛 | nr E ⊳ | 3 |
| Frace Disabled | Ŧ | \x x . | <u>6 v</u> | T I. | AWB | 1 | Se AUTI | 0 | | | | | |
| | | , | | A REAL | | | | | | | | | |
| | | | | | | | | | | | | | _ |
| | Select AVR | Program | ner | | | | | | | | e | 3 <u>×</u> | 1 |
| | Platform: STK500 AVRISP JTAG ICE JTAGICE n AVR Drage STAGICE n AVR Drage STAGICE n Tip: To auto button on th Note that a a debugging | kli hkli h-connect to re toolbar. tool cannot g session. Ir cted Mode. | the prog | rammer u | ort: JSB used last amming . 'Stop Do | time, pro | , ess the as it is co g' first. | 'Progra | mmmer' | Baud rai 115200 Baud rai active in | onnect Cancel te: te chan; nmediati | ges are | |

Bild 2.5-08: Logische Verbindung des USBprog mit dem AVR Studio herstellen

Con => AVRISPmkII => USB => Connect...

| AVRISP | mkll in ISP mod | e with ATmega88 | 8 <u>- </u> × |
|-----------|--|-------------------------------------|--|
| Main | Program Fuses | LockBits Advanced HW Settings | HW Info Auto |
| Dev | ice 🥳 | | |
| | Erase Device | | |
| F | Erase device be | fore flash programming 🛛 🔽 Veril | fy device after programming |
| - Flas | h | | |
| | O Use Current Sim | ulator/Emulator FLASH Memory | |
| | Input HEX File | | <u></u> |
| | Program | Verify | Read |
| EEF | ROM | | |
| | O Use Current Sim | ulator/Emulator EEPROM Memory | |
| | Input HEX File | | |
| | Program | Verify | Read |
| ELF | Production File Fo | mat | |
| 1 | nput ELF File | | |
| | Program | Save | Fuses and lockbits settings must be specified before saving to ELF |
| | | | |
| Detectin | g on 'USB' | | <u>ع</u> |
| Getting i | mkII with serial nur sp parameter SD= | iber 000040012825 found. 0x05 OK | |
| 1 | | | 7 |

Bild 2.5-09: Suchen der exe-Datei AVR_PB_LED.hex







Bild 2.5-12: AVR_PB_LED.hex anklicken

|)ffnen | | | | | 🗐 ? 🗙 |
|--------------------------|----------------------------|----------------------|-----------------|-------|-----------|
| Suchen in: | AVR_PB_LED | 1 | • | ⇔ 🗈 💣 | • |
| Zuletzt verwendete D. | Exe Link List Obj | | | | |
| Daten-LW | | | | | |
| Arbeitsplatz | | | | | |
| Netzwerkumgeb | Dateiname: | | | • | Öffnen |
| ung | Dateityp: | Flash Intel Hex File | s (*.hex;*.a90) | • | Abbrechen |

Bild 2.5-11: Unterverzeichnis Exe öffnen

| lffnen | | | | | 🔿 ? 🗙 |
|------------------------------------|-------------------------|--|-------------|----------|---------------------|
| Suchen in: | 🚞 Exe | | • | (= 🖻 💣 🗉 | |
| Zuletzt verwendete D Desktop | AVR_PB_LED. | hex El | | | |
| Daten-LW | | | | | |
| Arbeitsplatz | | | | | |
| Netzwerkumgeb ung | Dateiname: Dateityp: | AVR_PB_LED.hex Flash Intel Hex Files (* | .hex;*.a90) | • | Üffnen Abbrechen |

Bild 2.5-13: AVR_PB_LED.hex öffnen

| AVRISP mkl I in ISP mode with AT mega88 Main Program Fuses LockBits Advanced HW Settings HW Info Auto Device Erase Device | AVRISP mkII in ISP mode with AT mega88 Main Program Fuses LockBits Advanced HW Settings HW Info Auto Device Erase Device |
|---|--|
| Erase device before flash programming Verify device after programming | Erase device before flash programming Verify device after programming |
| Flash © Use Current Simulator/Emulator FLASH Memory © Input HEX File Program Verity Read EXADA © Use Current Simulator/Emulator EEPROM Memory © Use Current Simulator/Emulator EEPROM Memory © Input HEX File | Flash Ise Current Simulator/Emulator FLASH Memory Imput HEX File D:AVR_PB_LEDVExeVAVR_PB_LED.hex Program Verity Read EEPROM Imput HEX File Input HEX File Input HEX File |
| ELF Production File Format Input ELF File Program Save Save saving to ELF | ELF Production File Format Input ELF File Program Save saving to ELF |
| Detecting on 'USB' AVRISP mkII with serial number 0000A0012825 found. Getting isp parameter SD=0x05 0K | Erasing device DKI Programming FLASH DKI Reading FLASH DKI FLASH contents is equal to file DK Leaving programming mode DKI |

Bild 2.5-14: AVR_PB_LED.hex flashen

Bild 2.5-15: AVR PB LED.hex ist programmiert

HEUREKA! Die Tasten lassen die LED's wie gewollt AN- und AUSgehen!

Was ist zu unternehmen, wenn es nicht gleich so läuft - wenn man sich zum Beispiel vertippt hat??? Kehren wir zum alten immer wieder zitierten Beispiel-Projekt zurück und öffnen das vorhandene C-Projekt AVR_PB_LED. (Sollte das Projekt nicht mehr vorhanden sein, so ist es ratsam, die Phasen des Abschnittes 2.1 Erzeugen eines C-Projektes noch einmal abzulaufen).

Im folgenden Beispiel wurde "versehentlich" statt eines Semikolons ein Doppelpunkt gesetzt. Es werden im **Code Navigator** (linke Bildhälfte) angezeigte **Errors** durch Anklicken im Quell-Programm auf der rechten Bildhälfte sofort gefunden und markiert. Sollte das entsprechende Quell-Programm in der rechten Bildhälfte noch nicht geöffnet sein, so wird es geöffnet und die fehlerhafte Anweisung markiert. Der Compiler hat das gemerkt und führt **keine Assemblierung** durch:



Bild 2.5-16: Fehlererkennung im AVR_PB_LED



Bild 2.5-17: In Zeile 75 steht ein Doppelpunkt " : " statt eines Semikolons " ; "

Der Fehler wird direkt im rechten Fenster behoben und das Projekt **AVR_PB_LED** neu kompiliert. Dabei wird die Datei main.c automatisch im Projekt-Ordner berichtigt.

Will man eine gerade nicht angezeigte oder verdeckte *.c- oder *.h-Datei zu Editier-Zwecken auf der rechten Bildhälfte öffnen, so klickt man auf die gewünschte Datei in der links angezeigten Ordnerstruktur und schon ist sie da, zum Beispiel die Datei application.h:



Bild 2.5-18: Öffnen einer Datei z.B. zu Editierzwecken; hier: application.c

Die geänderten Anweisungen werden automatisch unmittelbar bei der Neu-Kompilierung auf der Festplatte in der Datei festgehalten, ohne dass eine explizite Anweisung zur Speicherung angefordert wird.

Eine große Hilfe kann es auch bedeuten, sich die in einer Header-Datei aufgeführten Makros (z.B. der Header-Datei macros.h) auflisten zu lassen. Dazu geht man auf das Register **Code Information**:



Bild 2.5-19: Auflisten der Makros in der Header-Datei macros.h

So gibt es in dem Compiler CVAVR noch eine ganze Menge zu entdecken. Einfach mal viele Knöpfe ausprobieren!

Im Abschnitt **2.1 Erzeugen eines C-Projektes** wurde anhand des praktischen Beispiels **AVR_PB_LED** nur die generelle Abfolge zur Erzeugung eines **C**-Projektes beschrieben, über eine Anordnung von zahlreichen Projekten auf der Festplatte "schwieg des Sängers Höflichkeit". Das Thema wird jedoch dadurch interessant, dass CVAVR zunächst für das **Create New Project** von seinem bestehenden Ordner zur Ablage ausgeht:

C:\cvavr2\bin

Es ist wohl wenig sinnvoll, zahlreiche AVR-Projekte gerade im Bereich der System-Platte abzulegen. Abgesehen davon, dass es viele "Ordnungen" gibt und jeder zu Recht auf seine als das "Alleinseligmachende" beharrt, sollte man sich an dieser Stelle darüber Gedanken machen, **WO** und **WIE** man seine und die kopierten Projekte ablegt, schließlich will man sie ja auch schnell wiederfinden. In dem Zusammenhang ist auch die "schwierigste Frage" zu stellen, die jeder Programmierer immer wieder von neuem beantworten muss: "Wie soll das Kind (besser: wie sollen **die** Kinder) heißen?!" Eine Möglichkeit ist es, unter einem "General"-Ordner alle AVR-Projekte nach dem Entstehungsdatum (zum Beispiel das Datum von der Datei **main.c**) abzulegen.

Der Autor hat sich für das Tutorial entschlossen, keinen "General-Ordner" **AVR_Projekt_Dateien** zu verwenden, sondern die AVR-Projekte in den **Teilen** zu belassen, wo sie angesprochen und erklärt werden. Dabei wird für kleine Beispiel-Programme die Form

xxx_Programm_yyyyy

und für ganze Projekte die Form

xxx_Projekt_yyyyy

gewählt. **xxx** steht für die Nummer des **Teils**, in dem das Programm bzw. das Projekt erscheint und **yyyyy** steht für die Programm- bzw. Projekt-Kurz-Bezeichnung.

Entsprechend werden die selbst erstellten globalen Header-Dateien und die Modul-Dateien in den folgenden Unterverzeichnissen abgelegt:

505_Globale_Header_Dateien

505_AVR_Modul_Dateien

Natürlich lassen sich noch etliche Kriterien für die Ordner-Festlegung heranziehen, z.B. nach technischer Realisierung, nach Themen-/Anwendungsbereichen usw. usw. Eins ist aber bei allen Projekten identisch: In allen Projekt- bzw. Programm-Ordnern stehen vor der Kompilierung

- alle *.c-Dateien, mindestens jedoch das Hauptprogramm main.c,
- die selbst erzeugten globalen Header-Dateien *.h,
- und die benötigten Modul-Dateien, bestehend aus *.c- und *.h-Dateien.

Später werden bei der Kompilierung noch weitere Unterverzeichnisse und zahlreiche weitere Zwischen-Dateien angelegt (vergl. Abschnitt **2.2 Dateistruktur eines C-Projektes**). Letztere braucht man nicht unbedingt aufzubewahren und können gelöscht werden. Sie werden ja bei jedem Kompilierungslauf erneut erzeugt und werden deshalb auch nicht in den Datei-Ordnern mitgeliefert.