

# AVR-8-bit-Mikrocontroller

## Arbeiten mit CodeVisionAVR C-Compiler

### Teil 00 - Inhaltsangabe

#### Teil 01 - Einführung

- 1 Eine Einführung in **C**
  - 1.1 Warum **C** ?
  - 1.2 Wie entstand **C** ?
  - 1.3 Der AVR-Mikrocontroller in einem eingebetteten System

#### Teil 02 - Erste Schritte

- 2 Grundausstattung und Installation der Werkzeuge
  - 2.1 Testboard, Mikrocontroller, Programmer
    - 2.1.1 Testboard und Mikrocontroller
    - 2.2.2 ISP-Programmer
  - 2.2 AVR Studio installieren
  - 2.3 Versteht mich der Mikrocontroller ?
  - 2.4 CodeVisionAVR **C**-Compiler installieren
  - 2.5 Editieren von Quell-Dateien

#### Teil 03 - Aufbau eines C-Projektes

- 3 Was ist ein **C**-Projekt ?
  - 3.1 Erzeugen eines **C**-Projektes
    - 3.1.1 Ein neues Projekt beginnen
    - 3.1.2 Ein **C**-Projekt generieren
  - 3.2 Dateistruktur eines **C**-Projektes
  - 3.3 Einbindung von AVR Studio in den CVAVR
  - 3.4 AVR Studio Debugger
  - 3.5 **C**-Compiler-Optionen

#### Teil 04 - Der Preprozessor

- 4 Preprozessor-Anweisungen
  - 4.1 Struktur der **C**-Quell-Programme
  - 4.2 **#include**-Anweisung
  - 4.3 **#define**-Anweisung (Makro)
    - 4.3.1 Makros ohne Parameter
    - 4.3.2 Makros mit Parametern
  - 4.4 **#undef**-Anweisung
  - 4.5 **#if**-, **#ifdef**-, **#ifndef**-, **#else**- und **#endif**-Anweisungen
  - 4.6 Andere Preprozessor-Anweisungen

#### Teil 05 - Syntax der C-Programmierung

- 5 Die Syntax der **C**-Programmiersprache
  - 5.1 **C**-Quell-Programme
    - 5.1.1 Kommentare
    - 5.1.2 Deklarationen (Vereinbarungen)
    - 5.1.3 Die Funktion **main**
    - 5.1.4 Schlüsselwörter (Keywords) des CodeVisionAVR **C**-Compilers
  - 5.2 Konstanten und Variablen
    - 5.2.1 Zahlensysteme
    - 5.2.2 Datentypen
    - 5.2.3 Konstanten
    - 5.2.4 Variablen
  - 5.3 Operatoren
    - 5.3.1 Arithmetische Operatoren
    - 5.3.2 Relationale Operatoren
    - 5.3.3 Logische und bitweise wirkende Operatoren
    - 5.3.4 Andere Operatoren und Shortcuts
  - 5.4 Komplexe Objekte in **C**
    - 5.4.1 Funktionen
    - 5.4.2 Funktions-Prototypen
    - 5.4.3 Pointers und Arrays
      - 5.4.3.1 Pointers
        - 5.4.3.1.1 Pointers in Verbindung mit **flash** und **eeprom**
        - 5.4.3.1.2 Pointers in Verbindung mit **typedef**
      - 5.4.3.2 Arrays

# AVR-8-bit-Mikrocontroller

## Arbeiten mit CodeVisionAVR C-Compiler

### Teil 00 - Inhaltsangabe

- 5.4.3.2.1 Ein-dimensionale Arrays
- 5.4.3.2.2 Zwei-dimensionale Arrays
- 5.4.3.2.3 Drei-dimensionale Arrays
- 5.4.3.3 Benutzen der Array-Namen als Pointers
- 5.4.3.4 Arrays von Pointers
- 5.4.3.5 Pointers auf Funktionen (Funktionszeiger)
- 5.4.3.6 Funktionen in Verbindung mit `typedef`
- 5.4.4 Strukturen und Unionen
  - 5.4.4.1 Strukturen
  - 5.4.4.2 Unionen
- 5.4.5 Komplexe Typen (eine Zusammenfassung)
- 5.5 Steuerung des Programmablaufs
  - 5.5.1 Anweisungsblöcke { . . . }
  - 5.5.2 Die Anweisung `if`
  - 5.5.3 Die Anweisung `if` in Verbindung mit `else`
  - 5.5.4 Die Fallunterscheidung `switch`
  - 5.5.5 Die Schleife `for`
  - 5.5.6 Die Schleife `while`
  - 5.5.7 Die Schleife `do` in Verbindung mit `while`
- 5.6 Arbeiten mit den Ein-/Ausgabe-Ports

### Teil 06 - Modularer Aufbau der AVR-Projekte

#### 6 Modularer Aufbau

- 6.1 Das Konzept
- 6.2 Nomenklatur
- 6.3 Die globalen Header-Dateien
  - 6.3.1 Die globale Header-Datei `typedefs.h` (Typ- und Bit-Definitionen)
  - 6.3.2 Die globale Header-Datei `iomx.h` (Definitionen aller Register-Bits)
  - 6.3.3 Die globale Header-Datei `macros.h` (Definitionen von Makros)
  - 6.3.4 Die globale Header-Datei `switches.h` (Definitionen von Schalter-Makros)
- 6.4 Die Module
- 6.5 Anwendung der Module
  - 6.5.1 Das Modul `application` (Anwendungs-Steuerung)
  - 6.5.2 Das Modul `lcd_2wire` (Ausgabe auf LCD-20x4)
  - 6.5.3 Das Modul `num_conversion` (Typ-Konvertierung nach ASCII)
  - 6.5.4 Das Modul `adc_ref-1-1` (ADC mit interner Referenz 1,1 V)
  - 6.5.5 Das Modul `rc5_decoder` (RC5-IR-Fernsteuerung-Dekoder)
  - 6.5.6 Das Modul `rc5_encoder` (RC5-IR-Fernsteuerung-Enkoder)
  - 6.5.7 Das Modul `usart` (USART-Steuerung)
  - 6.5.8 Das Modul `twi_master` (I2C- bzw. TWI-Steuerung)
  - 6.5.9 Das Modul `timer0_pwm` (TIMER0-Steuerung)

### Teil 07 - Anhänge

#### 7 Anhang

- 7.1 Begriffe und Definitionen
- 7.2 Eigene Bibliothek
- 7.3 AVR-Projekte (Programmbeschreibungen)
  - [7.3.1 AVR-Projekt PB\\_LED](#)
  - [7.3.2 AVR-Projekt 2 Draht LCD](#)
  - [7.3.3 AVR-Projekt IRDMS](#)

# AVR-8-bit-Mikrocontroller

## Arbeiten mit CodeVisionAVR C-Compiler

### Teil 00 - Inhaltsangabe

#### 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 ([info@alenck.de](mailto:info@alenck.de)).

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 - Beispiele:
•	Bild 5.1.4-02:	Daten-Adress-Raum	Benummerung eines Bildes
•	Beispiel 5.1.4-03:	EEPROM-Speicherung	Benummerung eines Beispiels/-gruppe
•	Form 5.1.3-01:	Die <code>main</code> -Funktion	Benummerung einer Formdarstellung
•	Tabelle 5.1.4-01:	Schlüsselwörter vom CVAVR	Benummerung einer Tabelle

#### Gravierende Änderungen gegenüber der Vorversion

1. Neuer Abschnitt **5.6** in der Inhaltsangabe