AVR-8-bit-Mikrocontroller
Gruppe 600 - AVR-C-Projekte
Teil 601 - AVR_PB_LED

Teil 601 - AVR_PB_LED

Teil 602 - 2_Draht_LCD

Teil 603 - IRDMS

Teil 604 - Pegelsonde (noch in Arbeit)

AVR-8-bit-Mikrocontroller
Gruppe 600 - AVR-C-Projekte
Teil 601 - AVR_PB_LED

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**).

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**	Benummerung eines Bildes
• **Beispiel 5.1.4-03: EEPROM-Speicherung**	Benummerung eines Beispiels
• **Form 5.1.3-01: Die `main`-Funktion**	Benummerung einer Formdarstellung
• **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 **3xx_Programm_zzzzz**	**ASM-Programmierung** **ASM-Programm-Beispiel**
Gruppe 400: AVR-ASM-Projekte **4xx_Projekt_yyyyy** **4xx_Programm_zzzzz**	**ASM-Projekte** **ASM-Projekt-Bezeichnung** **ASM-Programm-Beispiel**
Gruppe 500: CodeVisionAVR C-Compiler **5xx_Programm_zzzzz**	**C-Programmierung** **C-Programm-Beispiel**
Gruppe 600: AVR-C-Projekte **6xx_Projekt_yyyyy** **6xx_Programm_zzzzz**	**C-Projekte** **C-Projekt-Bezeichnung** **C-Programm-Beispiel**

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

3 AVR-C-Projekt IRDMS (Infrared Distance Measurement Sensor)

1 Einfache Beschaltung von LEDs und Tastern

1.1 Zur Hardware der LEDs und Taster

Dieses Projekt stellt eine einfache Applikation zum Gebrauch der LEDs und der Taster auf dem Testboard des AVR-Projektes dar. Es ist für viele weitere Projekte geeignet, die eine oder mehrere LEDs zur Anzeige bzw. einen oder mehrere Taster benötigen.

Die Taster sind nicht entprellt - können aber universell eingesetzt werden. Sie ziehen beim Betätigen einen angeschlossenen Pin von **High**- auf **Low**-Potential. Pull-Up-Widerstände sorgen für die **High**-Potentiale jedes angeschlossenen Tasters an den Eingangs-Pins des Controllers im Ruhezustand.

Jede LED ist mit ihrer Kathode über einen Vorwiderstand von 330 Ω an +5 V geschaltet und Ihre Anode ist mit einem Leistungs-Ausgang eines Treiber-Bausteins **ULN2003** verbunden. Die Leistungs-Ausgänge der **ULN2003** sind Open-Collektor-Ausgänge mit einer Belastbarkeit bis zu 500 mA. Damit kann man Klein-Motore, Relais und eben auch kleine LEDs schalten:

Ein **High**-Potential an einem Eingang des **ULN2003** schaltet den korrespondierenden Ausgang auf **Low**-Potential und lässt damit eine angeschlossene LED aufleuchten.

1.2 Beschaltung

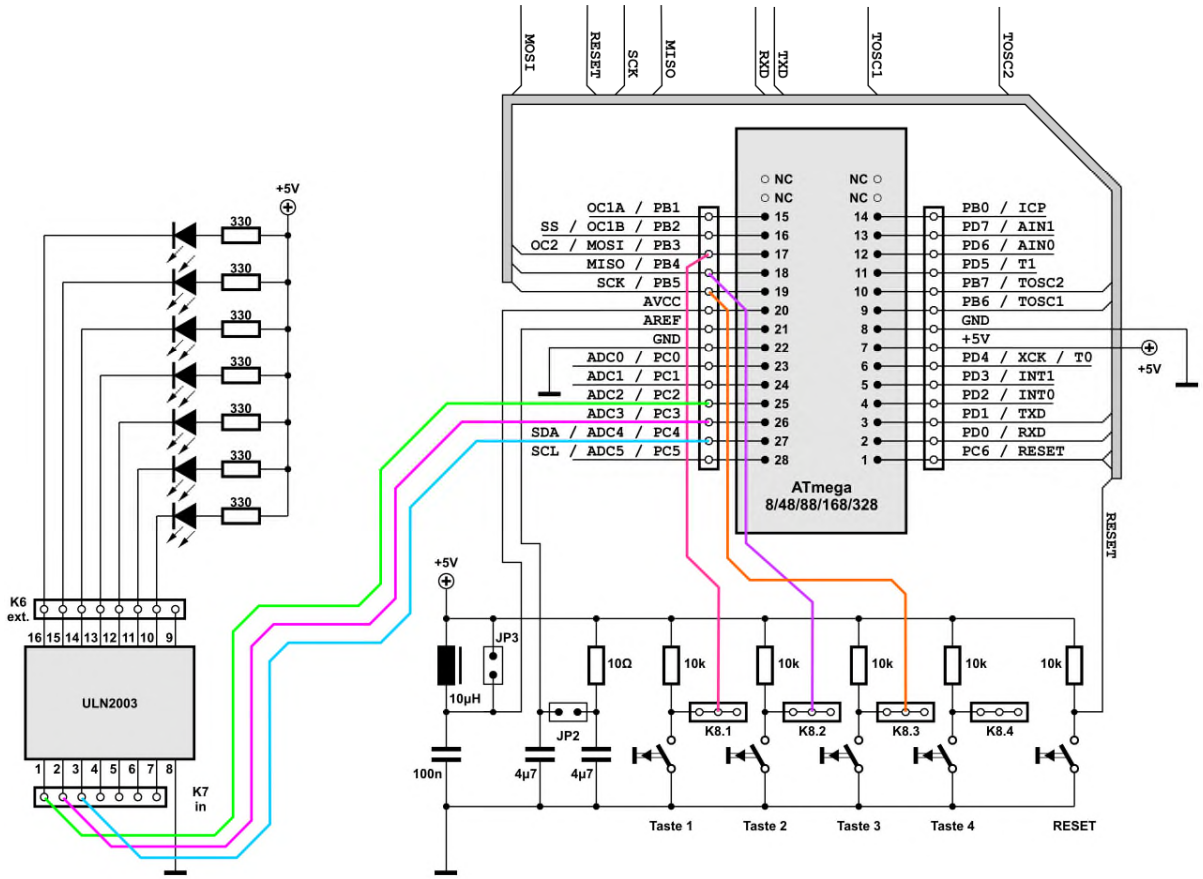

Bild 1.2-01: Beschaltung der LEDs und Taster (Bildvergrößerung)

Zunächst wird die Beschaltung des Testboards vorgenommen, d.h. die benötigten Pins werden mit dünnen flexiblen Drahtbrücken verbunden:

PC2	mit	LED1	PB3	mit	Taster1
PC3	mit	LED2	PB4	mit	Taster2
PC4	mit	LED3	PB5	mit	Taster3

1.3 Funktionsbeschreibung

- Taster S1 schaltet die LED1 ein, solange der Taster gedrückt ist.
- LED2 ist dauerhaft an und erlischt, solange Taster S2 gedrückt ist.
- Taster S3 schaltet bei jedem Tastendruck die LED3 um ('toggelt' sie).

Die Port-Funktionen

Es folgt eine kurze Wiederholung aus dem **Teil 503 - Preprozessor**. In diesem Teil wurden die vielen Ersetzungen am Beispiel dieses Projektes anhand von **#define**-Anweisungen beschrieben, so dass man hier vom Ergebnis dieser Ersetzungen ausgehen kann.

Die LEDs betreffend

LED1_DDR, LED2_DDR und LED3_DDR als auch LED1_BIT, LED2_BIT und LED3_BIT sind zuständig für das Datenrichtungsregister **DDRC** (Bit2, Bit3 und Bit4).

Merke: Ein Datenrichtungsregister (DDR) bestimmt, ob die Pins des betreffenden Ports als Eingänge **oder** als Ausgänge funktionieren sollen. Ein **High**-Potential an den betreffenden Pin des DDR schaltet den korrespondierenden Pin am Port als Ausgang und ein **Low**-Potential an diesem Pin schaltet den korrespondierenden Pin am Port als Eingang. Zum Beispiel:

```
DDRC = 00001100
```
bedeuet, dass Bit2 (PC2) und Bit3 (PC3) vom Port C als Ausgänge geschaltet werden.

Der letzte Schritt des Preprozessors in dieser Angelegenheit ergibt die realen Anweisungen:

```
// Initialisieren der LED´s
DDRC |= 0x04;                    // Setze LED1-Pin auf Ausgabe
DDRC |= 0x08;                    // Setze LED2-Pin auf Ausgabe
DDRC |= 0x10;                    // Setze LED3-Pin auf Ausgabe
```

Das sind also bitweise ODER-Verknüpfungen von Bit2, Bit3 und Bit4 im Datenrichtungsregister **DDRC** mit den vorher definierten Konstanten; das Ergebnis wird wieder im Datenrichtungsregister abgespeichert und sorgt so für die Initialisierung dieser Bits für die Ausgabe.

Ergebnis für das Datenrichtungsregister **DDRC** (Ursprungswert von **DDRC** ist nach dem Restart immer binär **00000000**) ergibt sich nach folgender Befehlskette:

```
DDRC = DDRC | 00000100 = 00000100
DDRC = DDRC | 00001000 = 00001100
DDRC = DDRC | 00010000 = 00011100
```
Bit2 wird als Ausgang konfiguriert
Bit2 und Bit3 werden als Ausgänge konfiguriert
Bit2 (PC2), Bit3 (PC3), Bit4 (PC4) sind Ausgänge !

Anmerkung: Damit kann über die Bit2 bis Bit4 am **Port C** eine Ausgabe (Schalten der LED's) stattfinden. Eine LED leuchtet erst dann, wenn das betreffende Bit des **Port C** logisch **1** (**TRUE**) aufweist (+5 Volt = **High**). Die Zählung der Bits erfolgt von rechts nach links beginnend mit Bit0, Bit1, Bit2 usw.

Die Tasten betreffend

Da die Tasten "Signal-Geber" für den Controller darstellen, werden durch die Ersetzungskette des Preprozessors die Pins des Datenrichtungsregisters **DDRB** auf Null (alle Pins werden als Eingänge konfiguriert) und die Eingabe-Bits Bit3 (PB3) bis Bit5 (PB5) von **PINB** (**Port B**) auf logisch **1** (**TRUE**) gesetzt.

AVR-8-bit-Mikrocontroller
Gruppe 600 - AVR-C-Projekte
Teil 601 - AVR_PB_LED

Im Hauptprogramm **main.c** treten die folgenden Initialisierungs-Anweisungen auf:

```
// Initialisieren der Tasten (Pushbuttons)
S1_INIT();
S2_INIT();
S3_INIT();
```

Sie korrespondieren mit den Definitionen in der Header-Datei **switches.h**, mit den Definitionen in der Header-Datei **application.h** und den Ersetzungen von BIT3, BIT4 und BIT5 in der Header-Datei **typedefs.h** so dass sich das Ergebnis in dieser Anwendung für das Datenrichtungsregister DDRB und die Eingabepins von PINB schließlich (Ursprungswerte von DDRB und PINB sind nach dem Restart immer binär **00000000**) nach folgenden Befehlsketten einstellt:

```
DDRB = DDRB & 11110111 = 00000000      alle Bits sind als Eingänge konfiguriert
DDRB = DDRB & 11101111 = 00000000      alle Bits sind als Eingänge konfiguriert
DDRB = DDRB & 11011111 = 00000000      alle Bits sind als Eingänge konfiguriert

PINB = PINB | 00001000 = 00001000      Bit3 ist auf Eingabe aktiviert
PINB = PINB | 00010000 = 00011000      Bit3 und Bit4 sind auf Eingabe aktiviert
PINB = PINB | 00100000 = 00111000      Bit3 (PB3), Bit4 (PB4), Bit5 (PB5) sind auf Eingabe aktiviert
```

Anmerkung: Das entspricht einer Spannung von jeweils +5 V (**High**-Potential) an den Pins PB3 bis PB5, die erst beim Betätigen einer Taste **S1** bis **S3** auf Masse (**Low**-Potential = 0 V) geschaltet werden.

Ausführliche Erläuterungen werden im Quell-Programm **main.c** gegeben.

Der Programm-Quell-Code (**C**- und Header-Dateien) ist zu finden unter
601_Projekt_AVR_PB_LED