

Erstes Projekt mit dem myAVR Board 2 USB

- ✓ WinAVR ist installiert
- ✓ AVR Studio ist installiert
- ✓ USB-Treiber für das myAVR Board 2 USB und mySmartUSB ist installiert
- ✓ Erster Verbindungstest mit myAVR QuickProg ist erfolgreich

Einstellungen

Programmer:

Anschluss:

gefundenener Prozessor

Einstellungen

Programmer:

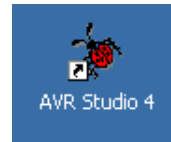
Anschluss:

gefundenener Prozessor

Prozessor: ATmega8
Flash: 8 KByte
EEProm: 512 Byte
SRAM: 1 KByte

myAVR QuickProg baut die Verbindung auf und findet den µController

- myAVR QuickProg schließen
- AVR Studio starten => New Project => AVR GCC



Create new project

Project type:

- Atmel AVR Assembler
- AVR GCC

Project name:

Create initial file Create folder

Initial file: .c

ICE50
JTAG ICE
AVR Simulator
ICE200

ATmega649
ATmega6490
ATmega8
ATmega8515
ATmega8535
ATmega88
ATtiny11
ATtiny12

Open platform options

- Quelltext eingeben, hier einfach aus Word-Datei kopieren

```

#include <avr/io.h>           // Einbin
                             // diese

void main( void )           // Die Fu
{                             // es wir
                             // es wir

```

```

#include <avr/io.h>           // Einbindung der allgemeinen "io.h" Header-Datei
                             // diese Datei enthält Definitionen der Namen der Register

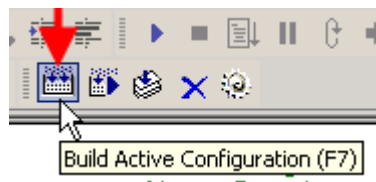
void main( void )           // Die Funktion "main" wird aufgerufen,
{                             // es wird kein Wert mitgegeben (void)
                             // es wird kein Wert zurückgegeben ("void" vor main)

    DDRB = 0b00000001;      // Port B, Pin 0 als Ausgang (1) definieren
    PORTB = 0b00000001;     // Port B, Pin 0 aktiv setzen (andere Schreibweise 0x00
                             //                               -->Hexadezimal)

    while ( 1 )             // Endlosschleife,
    {                         // damit kein weiterer Quellcode ausgewertet wird
        ;                   // --> Bedingung ist erfüllt ( 1 = true )
    }                       // imaginärer Befehl ";" wird ständig ausgeführt
}

```

- Compilieren und Objektdatei erzeugen



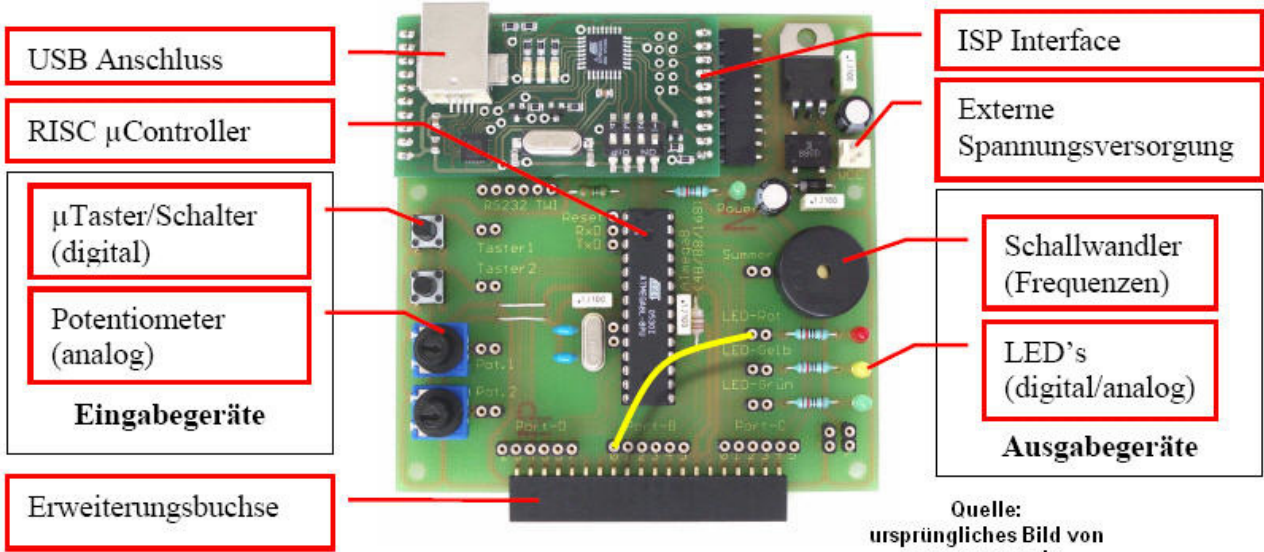
- Compilieren und Simulation ausführen



- Board mit PC via USB verbinden (eventuell oben schon erledigt)
- PORTB0 mit Anschluss der roten LED verbinden
- Dip-Schalter kontrollieren (1 ON, 2+3+4 OFF, Werkseinstellung).

=> => siehe Abbildung auf der folgenden Seite

Blockbild / Übersicht



- Atmega8 programmieren (brennen)

The screenshots illustrate the AVR programming process:

- AVR Studio - C:\Dokumente und Einstellungen\lehrer\Eigene Dateien\at06_at**: The main interface with the **Tools** menu open, highlighting **AVR Prog...**.
- AVRprog**: The programming window showing:
 - Hex file: led.hex
 - Buttons: Browse..., Exit...
 - Flash section: Program, Verify, Read
 - EEPROM section: Program, Verify, Read
 - Device: ATmega8
- Browse**: A file explorer window showing the selection of **led.hex** (Typ: HEX-Datei, Größe: 345 Byte).

Additional text in the AVRprog window:

```

funktion "main" wird aufgerufen,
kein Wert mitgegeben (void)
kein Wert zurückgegeben ("void" vor main)

Pin 0 als Ausgang (1) definieren
Pin 0 aktiv setzen (andere Schreibweise 0x00
-->Hexadezimal)
    
```

Red arrows indicate the flow from the menu to the file selection and then to the programming options.

Rote LED leuchtet?! Yes!!! Geschafft!!!