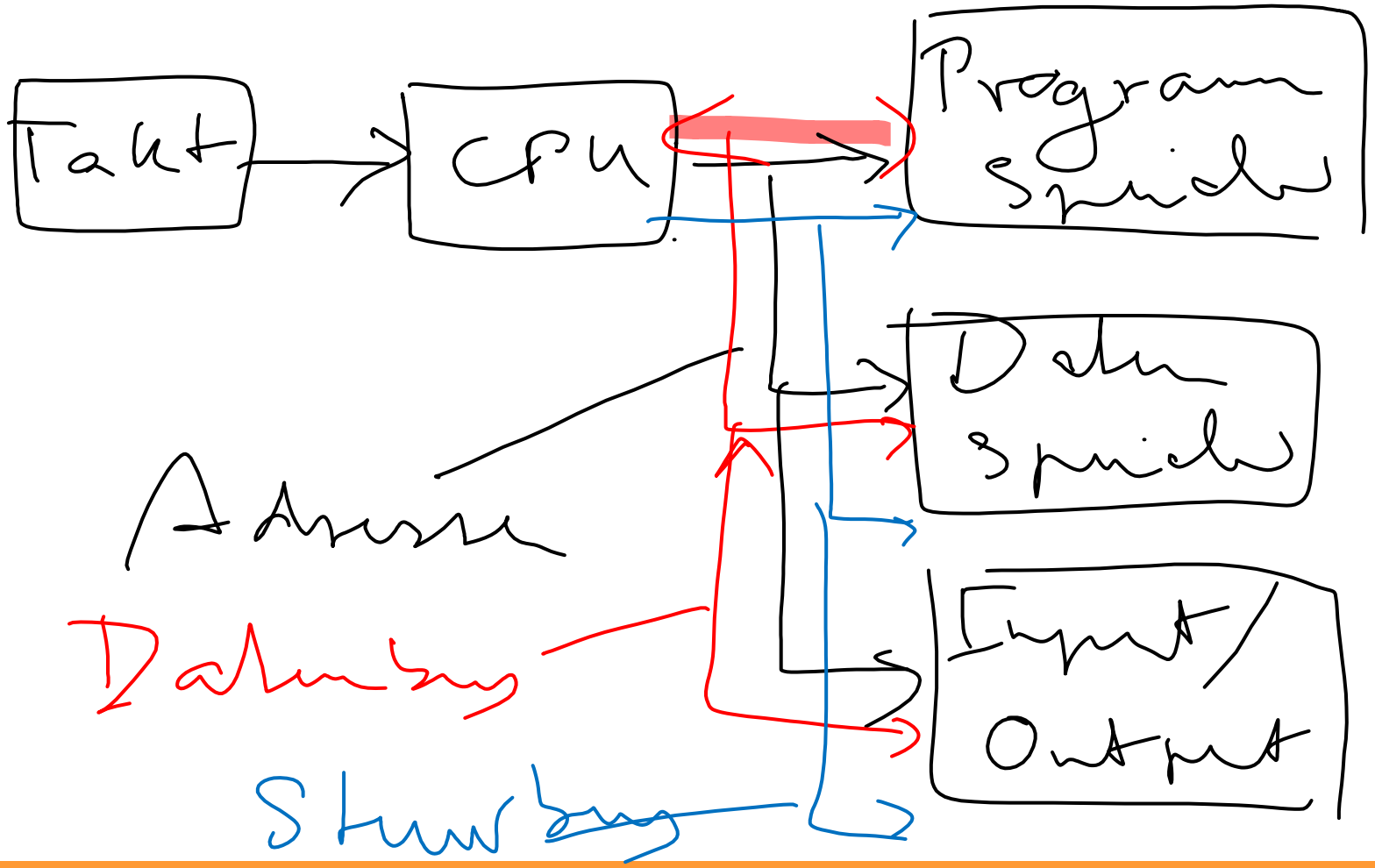
Mikrocomputertechnik

2. Kapitel: Aufbau und Entwicklung mit Mikrocontrollern

Martin Versen / Franz Perschl

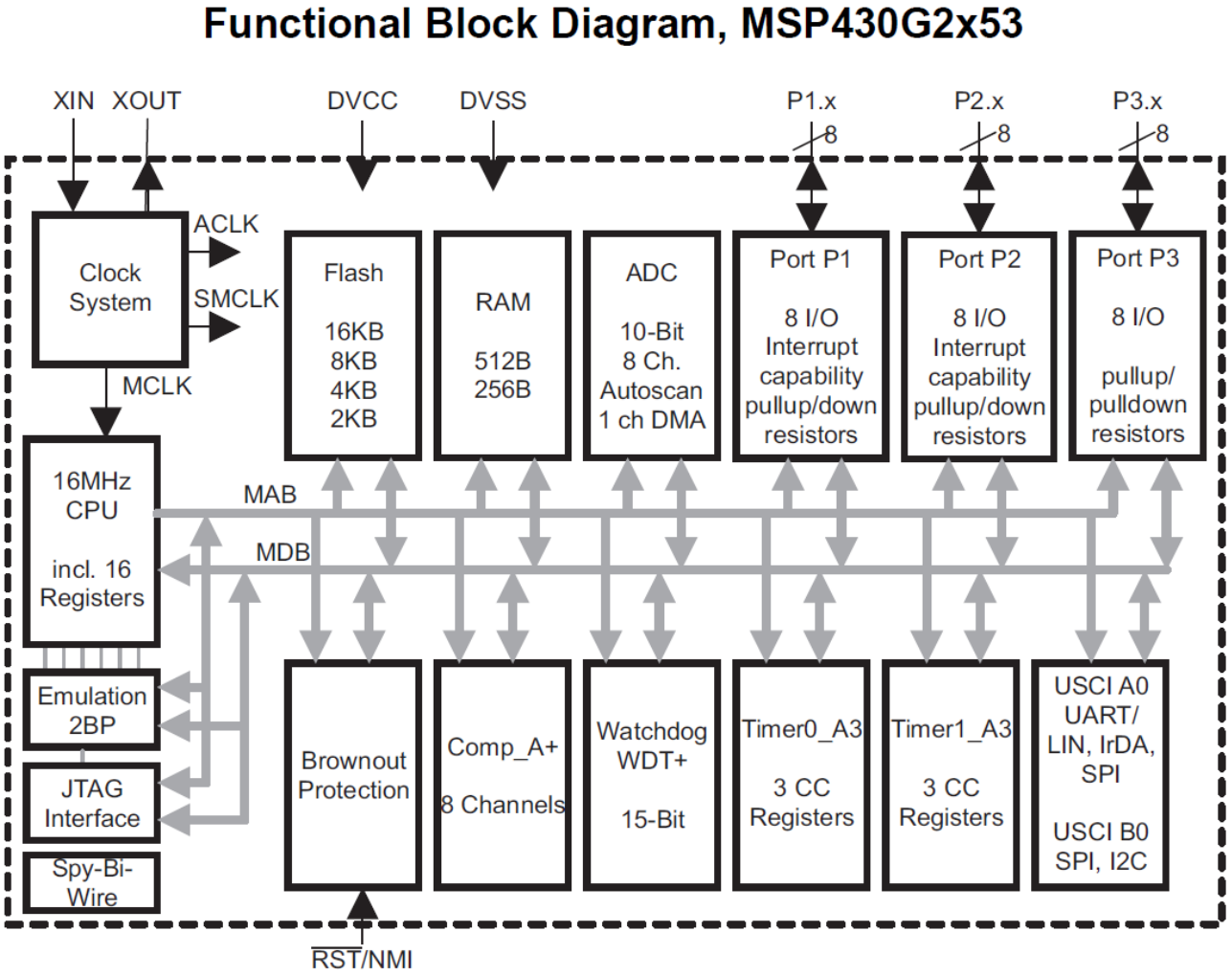
Grundlegender Aufbau eines Mikrocontroller



Mikrocontroller - Peripherie

- GPIO General Purpose I/O
- Timer
- Pulsweitenmodulation
  - PWM → DAC Funktion
- Schnittstellen
  - → SPI, I^2C, UART, USB, RS232, CAN, LIN
- ADC

Grundlegender Aufbau eines Mikrocontroller



Quelle: Texas Instruments, SLAU735, Mai 2013.

Entwicklung mit Mikrocontrollern (1)

- Heute ist es relativ einfach, einen Mikrocontroller zu programmieren (im Vergleich zu ~1990)
  - Flashspeicher erlauben ein vielfaches Schreiben von Programmen ohne lange Wartezeit und ohne den Chip unter UV-Licht zu löschen.
  - Reduzierte Kosten für Silizium bzw. Transistor pro Chip – Hardware für Debugging ist bereits im Mikrocontroller implementiert.
  - Hersteller bieten die meist freie Entwicklungssoftware an.

Entwicklung mit Mikrocontrollern (2)

- Gemeinsamkeiten mit der reinen Softwareprogrammierung
  - Schreiben des Programms
  - Kompilieren
- Unterschiede zur reinen Softwareprogrammierung:
  - Programm wird auf ein Ziel, den Mikrocontroller, geladen statt es „laufen zu lassen“
  - Im Falle von Fehlern: im Extremfall Debugging mit Oszilloskop und Logikanalysator (logic analyzer)

Gliederung zur Mikrocomputertechnik

Entwicklungsumgebung IDE (1) (Integrated Development Environment)

- **Editor** – häufig mit Farbe für „Syntax Highlighting“
- **Assembler** oder **Compiler** produziert den ausführbaren Code
- **Linker** kombiniert die Dateien und Subroutinen aus den Libraries für einen spezifischen Mikrocontroller
- **Simulatoren** erlauben die Modellierung von CPU und Speicher (Einbindung von Hardware und Interrupts schwierig)

Entwicklungsumgebung IDE (2) (Integrated Development Environment)

- **Embedded Emulator läuft im Mikrocontroller selbst und wird durch einen Debugger im PC kontrolliert. PC und Mikrocontroller kommunizieren über ein spezielles Interface, z.B. die JTAG-Schnittstelle**

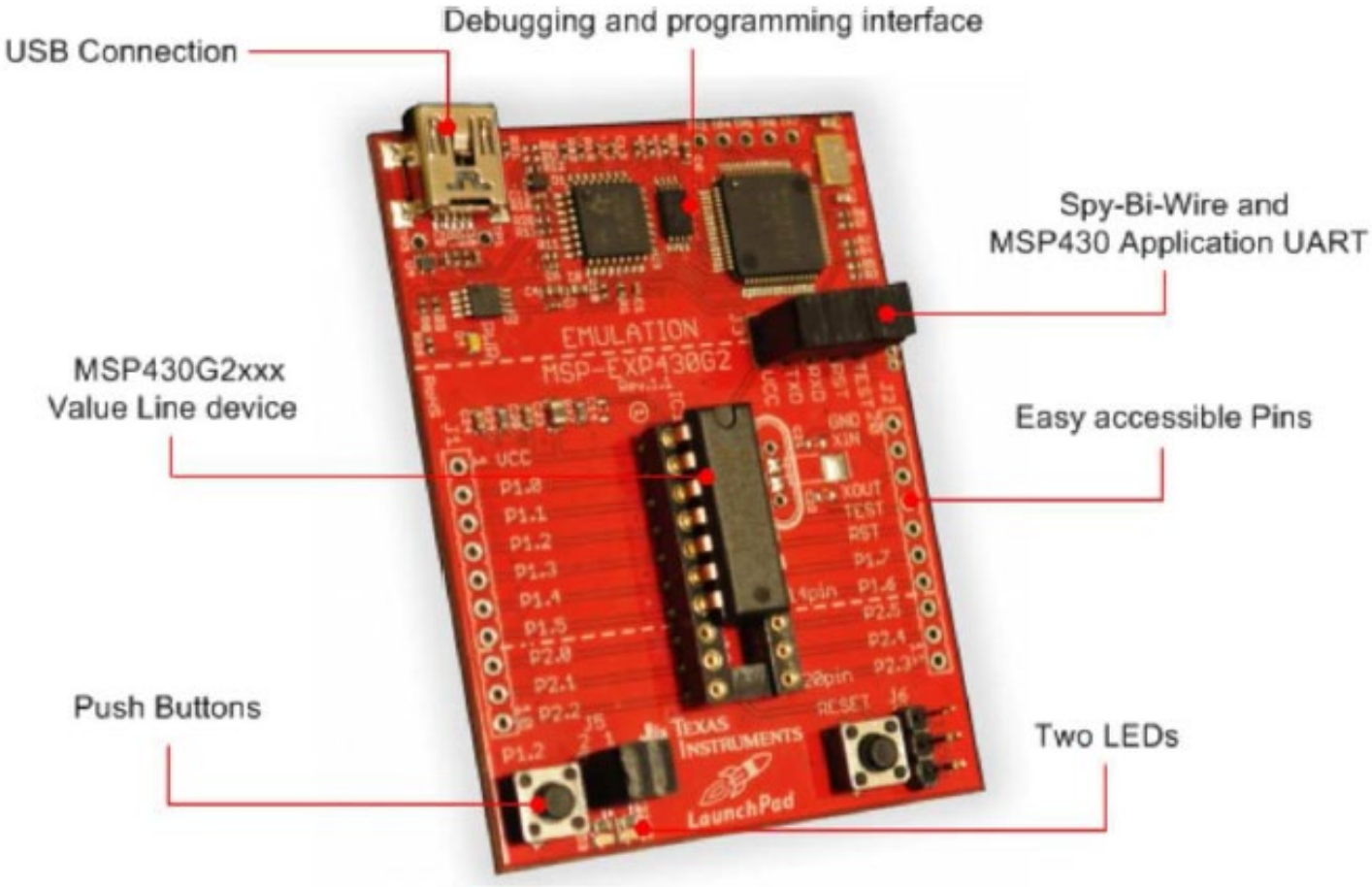
- **In-Circuit-Emulator (ICE) ist eine teure Hardware, die vom PC kontrolliert wird und die den Mikrocontroller emuliert, d.h. alle Funktionen nachbildet. Für kleine Mikrocontroller heutzutage überflüssig.**

- **Flash-Programmierer lädt („brennt“) das Programm in den Mikrocontroller.**

Gliederung zur Mikrocomputertechnik

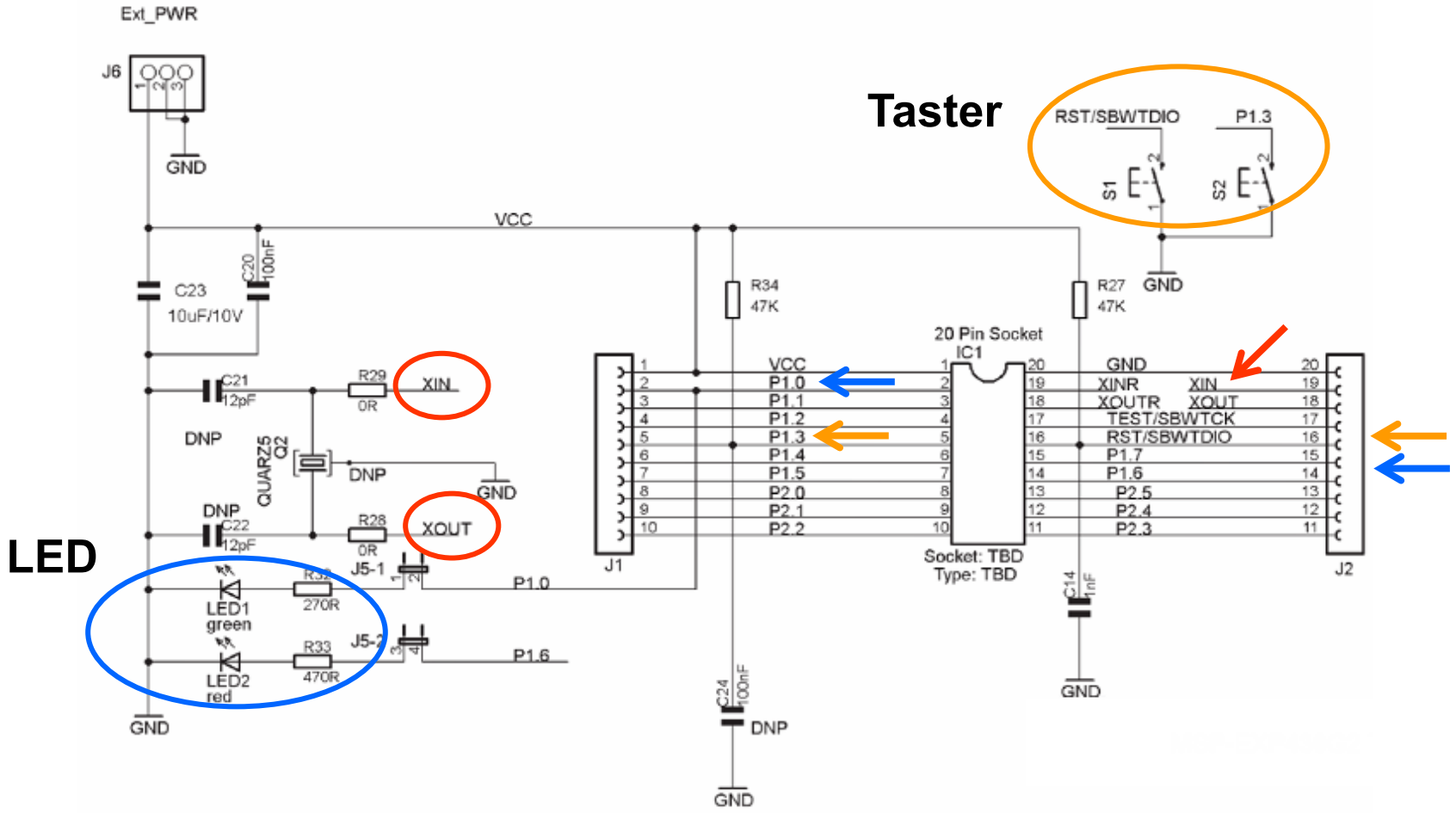
1. Einleitung und Motivation
2. Entwicklung mit Mikrocontrollern
   - 2.1 Die Entwicklungsumgebung
   - 2.2 LaunchPad MSP-EXP430G2
   - 2.3 Code Composer Studio und erstes Programm

Übersicht MSP-EXP430G2 LaunchPad



Quelle: Texas Instruments, SLAU318, Juli 2010.

Schaltplan mit Peripherie



- **Anschlüsse 3, 4, 6, 7 und 15 stehen bei dieser Belegung (und Benutzung der Bauteile) noch zur Verfügung.**

Quelle: Texas Instruments, SLAU318, Juli 2010.

Funktionen des LaunchPad

- USB debugging and programming interface featuring a driverless installation and application UART
- serial communication with up to 9600 Baud
- Supports all MSP430G2xx and MSP430F20xx devices in PDIP14 or PDIP20 packages
- Two general-purpose digital I/O pins connected to green and red LEDs for visual feedback
- Two push buttons for user feedback and device reset
- Easily accessible device pins for debugging purposes or as socket for adding customized extension boards
- High-quality 20-pin DIP socket for an easy plug-in or removal of the target device

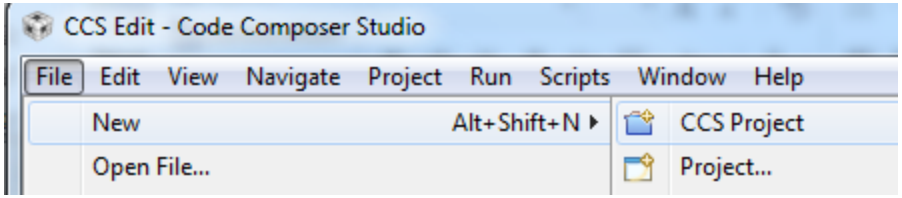
Auszug aus: Texas Instruments, SLAU318, Juli 2010.

Gliederung zur Mikrocomputertechnik

Code Composer Studio

Erstellen eines neuen Projekts (1)

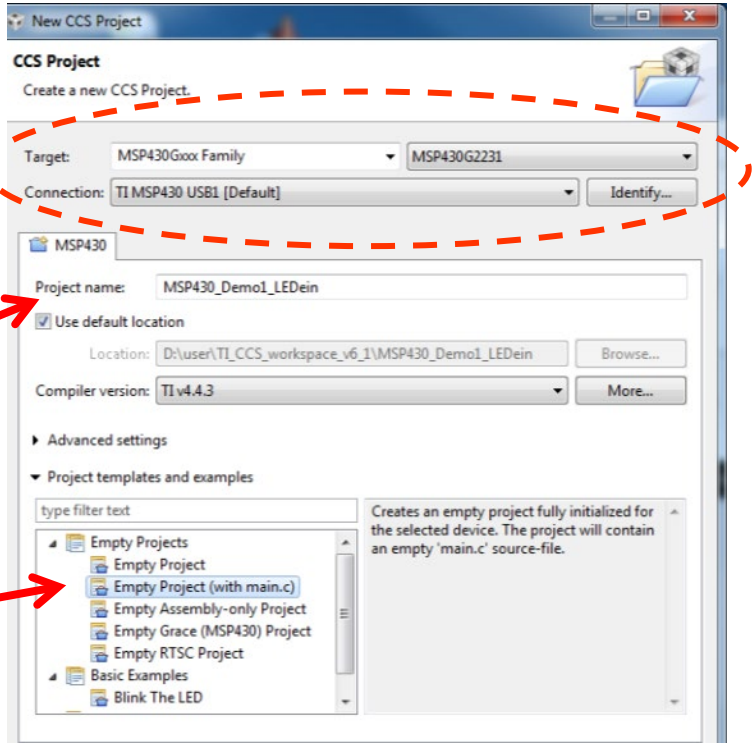
- **Menüauswahl**
  File → New → CCS Project



- **Device Settings: hier erfolgt die Auswahl des Bausteins MSP430G2553**

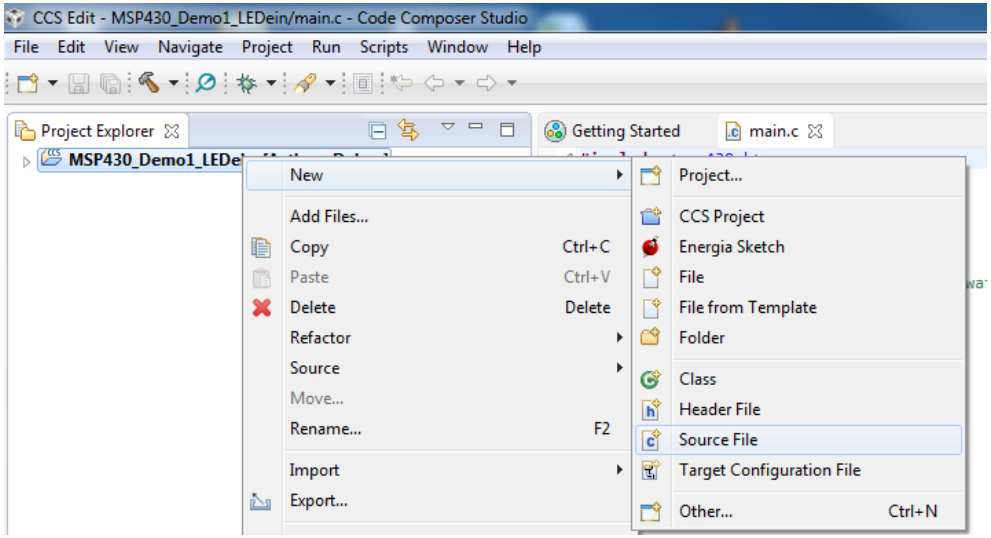
- **Dialog „Dateiname“**
  hier:
  „MSP430_Demo1_LEDein“

- **Bei der richtigen Auswahl wird mit „Finish“ eine „main.c“ automatisch erzeugt.**

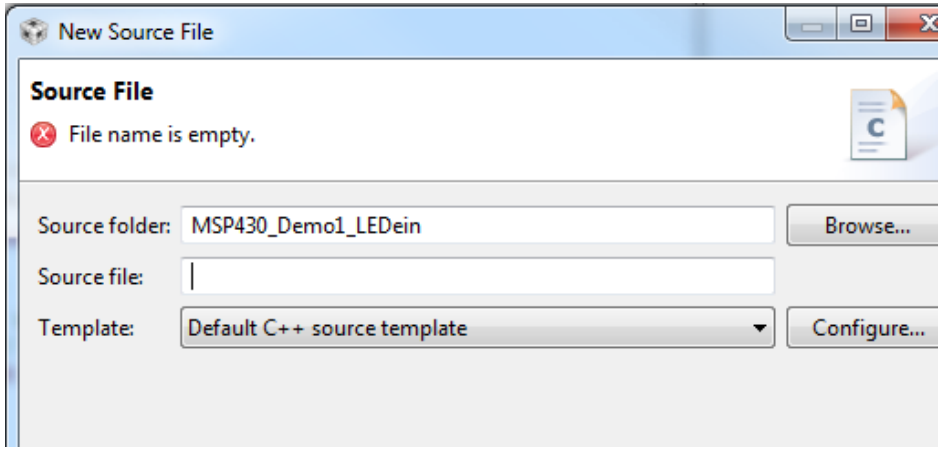


Alternative: Erstellen eines neuen Source Files

- **Rechte Maustaste auf dem neu erstellten Projekt:**
  → New → Source File



- **Dialog „Dateiname“ für eine C++ Vorlage**

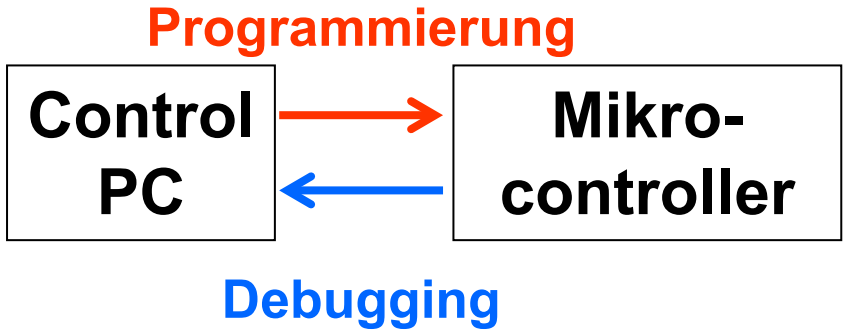


Gliederung zur Mikrocomputertechnik

Programmierung und Debugging: Kommunikation in zwei Richtungen

- **Programmierung:**
  PC → Mikrocontroller

- **Debugging**
  Mikrocontroller → Control-PC



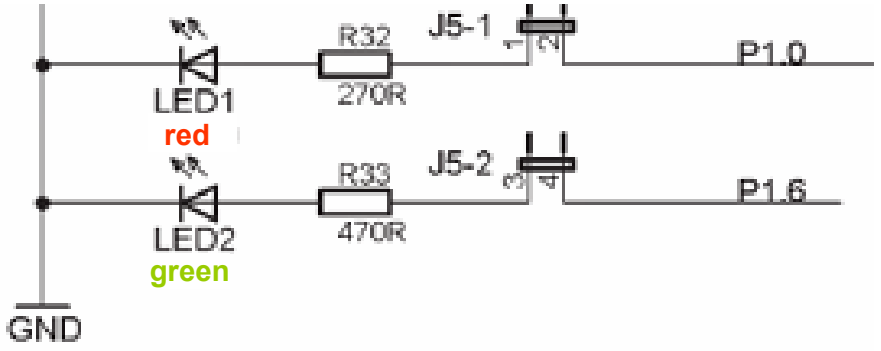
- **Drei Kommunikationsarten:**
  - **Bootstrap Loader (Serielles Interface)**
    BSL → maskierter ROM Speicher, geschützt gegen unberechtigten Zugriff
  - **Konventionelles JTAG (4 Leitungen)**
  - **Spy-by-Wire Programmier-Interface (2 Leitungen)**

JTAG – Joint Action Test Group – IEEE Standard 1149.1

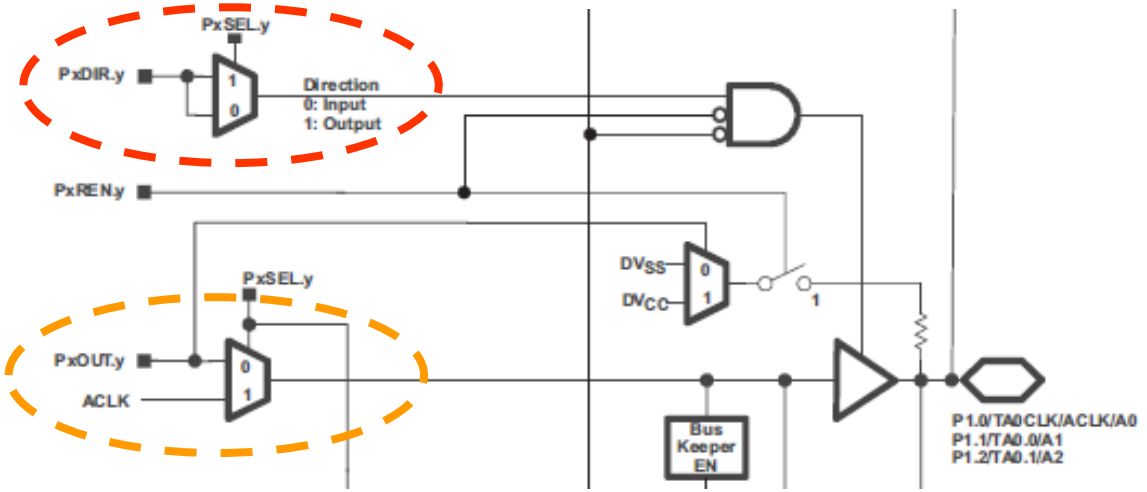
- Das konventionelle JTAG Interface besteht aus vier Leitungen:
  - TDI, TDO, TMS, TCK (Input, ouput, control und Clock)
  - Nachteil: 4 Ressourcen eines kleinen Controllers sind festgelegt
  - Lösung: Multiplexing der Pin-Funktionen durch TEST Anschluss
  - TI: 14-Pin-Anschluss mit 4 Signalen + VDD + VSS+ RST/NMI durch ein „flash emulation tool (FET)“ (aka. „pod“ oder „wiggler“)
  - Anschluss am parallelen Port oder USB-Anschluss des Control-PC mit Kosten >50€
- Spy-By-Wire (TI) verwendet nur zwei Leitungen
- In beiden Fällen vollen Zugriff auf alle Register (CPU, RAM, ROM)

Logische Ausgänge und LED (→ siehe Datenblatt S. 37)

- LED an den Pins P1.0 und P1.6 sind „active high“



- Register aus dem Datenblatt zur Ansteuerung der digitalen Pins



- LED soll leuchten:
  (Ausgang) PxDIR.y = 1
  (active high) PxOUT.y = 1

Quellen: Texas Instruments, SLAU318, Juli 2010.
Texas Instruments, SLAS694E, rev. Jan 2011.

Demo mit CCS

Logische Ausgänge und LED

- C-Code für die grüne LED ein- und rote LED ausgeschaltet

```
#include  <msp430.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;          // Watchdog-Timer anhalten
  P1DIR = 0x__;                      // P1.0 und P1.6 in Ausgangsrichtung
  P1OUT = 0x__;                      // nur die LED2 am P1.6 eingeschaltet

  for (;;) {                         // infinite loop
    }
}
```

Logische Ausgänge und LED

- C-Code für die grüne LED ein- und rote LED ausgeschaltet

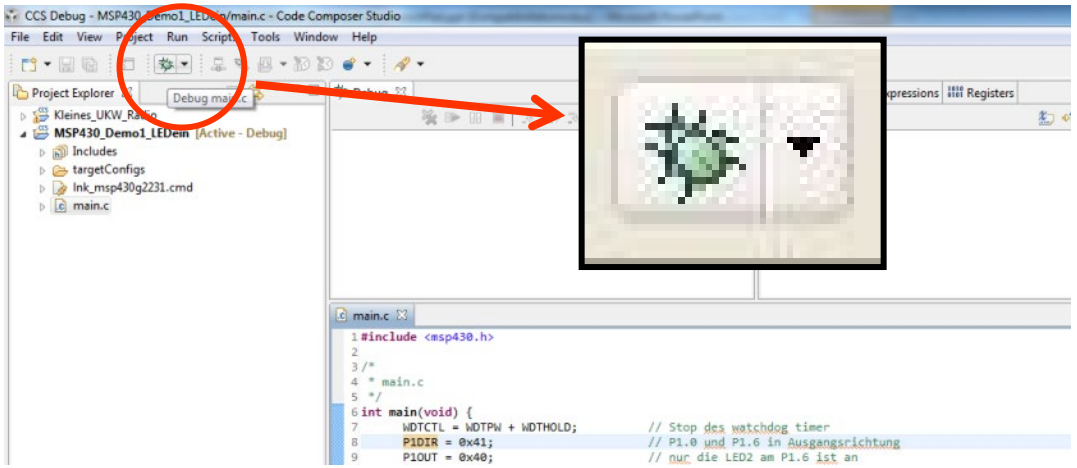
```
#include  <msp430g2231.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;          // Watchdog-Timer anhalten
  P1DIR = 0x41;                      // P1.0 und P1.6 in Ausgangsrichtung
  P1OUT = 0x40;                      // nur die LED2 am P1.6 eingeschaltet

  for (;;) {                         // infinite loop
    }
}
```

Start des Programms

- **Build/Debug des neuen Programms:**



- **Starten des neuen Programms auf dem Mikrocontroller im „Debug“ Fenster**