Praktikum Mikrocomputertechnik (EIT - B3, MEC - B5)

Versuch 01 – Ohne Musterlösung

Version: 2024-10-13b

Inhaltsverzeichnis

1 Verwendete Hardware, wichtige Dokumente

1.1 Launchpad EXP-MSP430G2

Im Praktikum verwenden wir eine Platine mit dem Namen **EXP-MSP430G2**.
Mehr Informationen hierzu finden Sie im Dokument **slau318**, zu finden im Learning Campus unter der Adresse https://learning-campus.th-rosenheim.de/mod/resource/view.php?id=303216 oder auch im WWW.

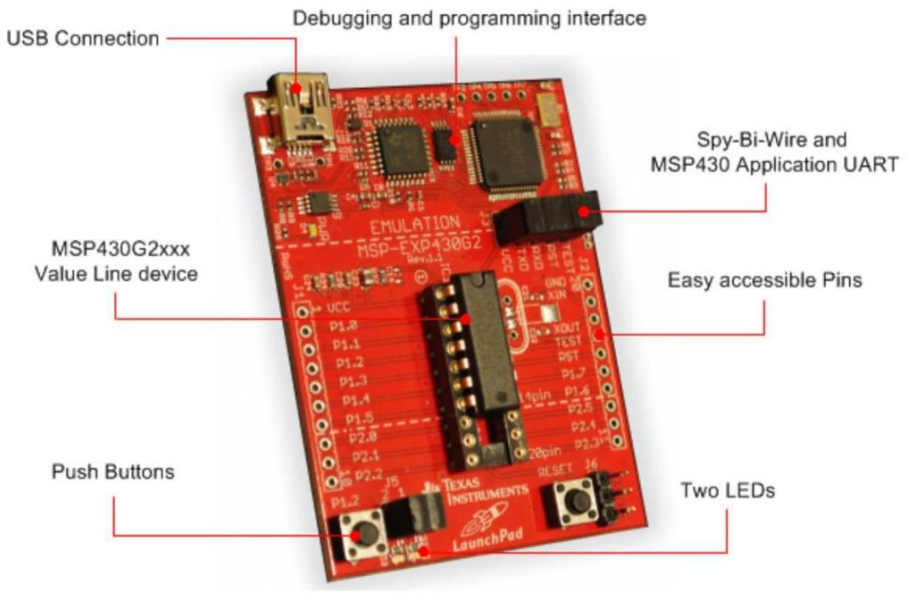
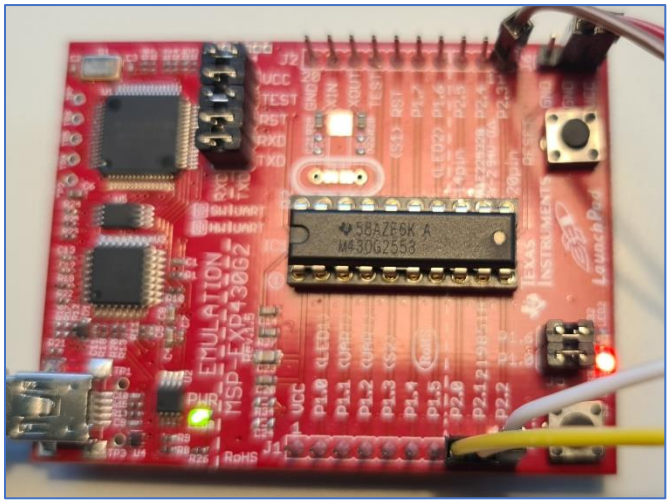


Figure 1. MSP-EXP430G2 LaunchPad Development Kit Overview

https://learning-campus.th-rosenheim.de/mod/resource/view.php?id=303216

1.2 Mikrocontroller MSP430G2553

Auf Ihrem LaunchPad ist der Mikrocontroller **MSP430G2553** verbaut.
Mehr Informationen hierzu finden Sie im Dokument **slau735**, zu finden im Learning Campus unter der Adresse https://learning-campus.th-rosenheim.de/mod/resource/view.php?id=303215 oder auch im WWW.



Allgemeine Informationen zu den Controllern der MSP430x2xx Familie finden Sie im Dokument **slau144**, zu finden im Learning Campus unter der Adresse https://learning-campus.th-rosenheim.de/mod/resource/view.php?id=303213 oder auch im WWW.

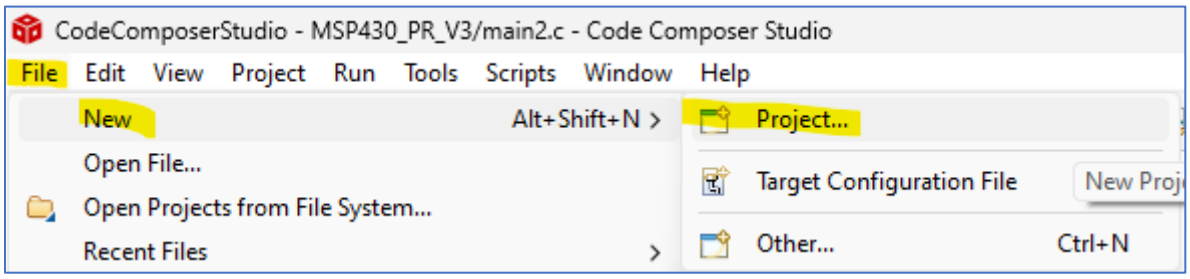
1.3 Weitere Hardware

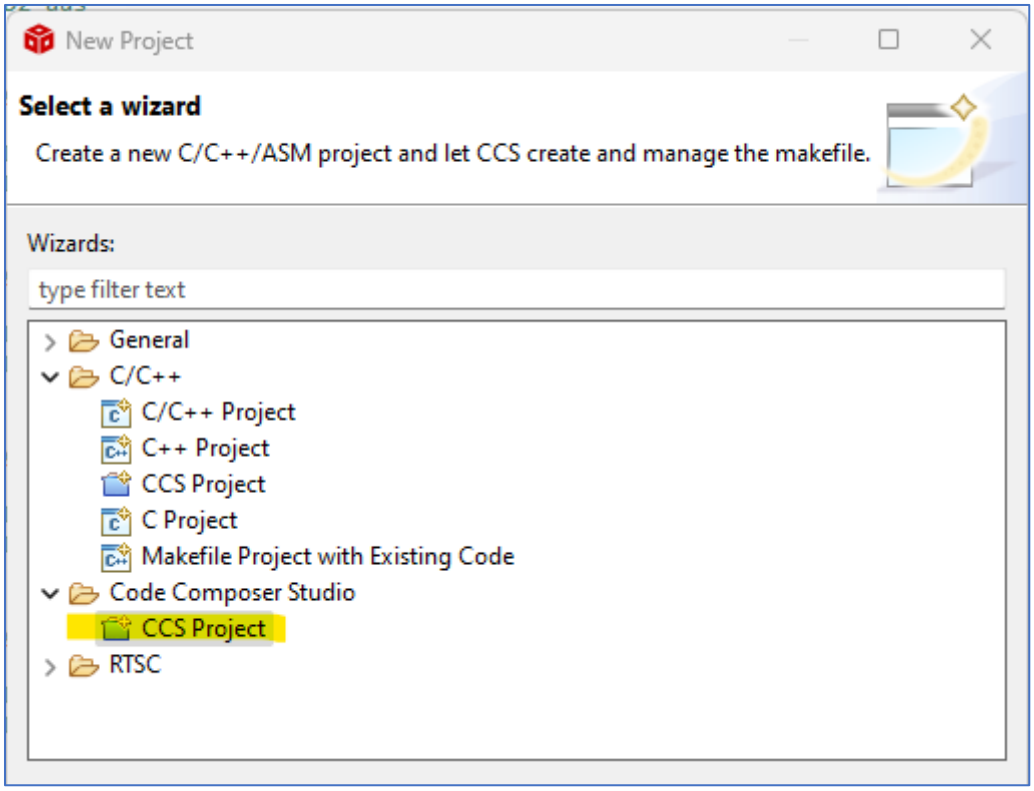
Im Laufe des Praktikums werden sie verschiedene Hardwarekomponenten erhalten und für Ihre Programmierübungen verwenden.
Zum größten Teil wurden diese Komponenten an der TH-Rosenheim entwickelt und sogar gefertigt. Hier geht ein besonderer Dank an Stefan Kipfelsberger für seine Engagement und seine ansteckende Begeisterung.

2 Versuch 01 – Digital IO

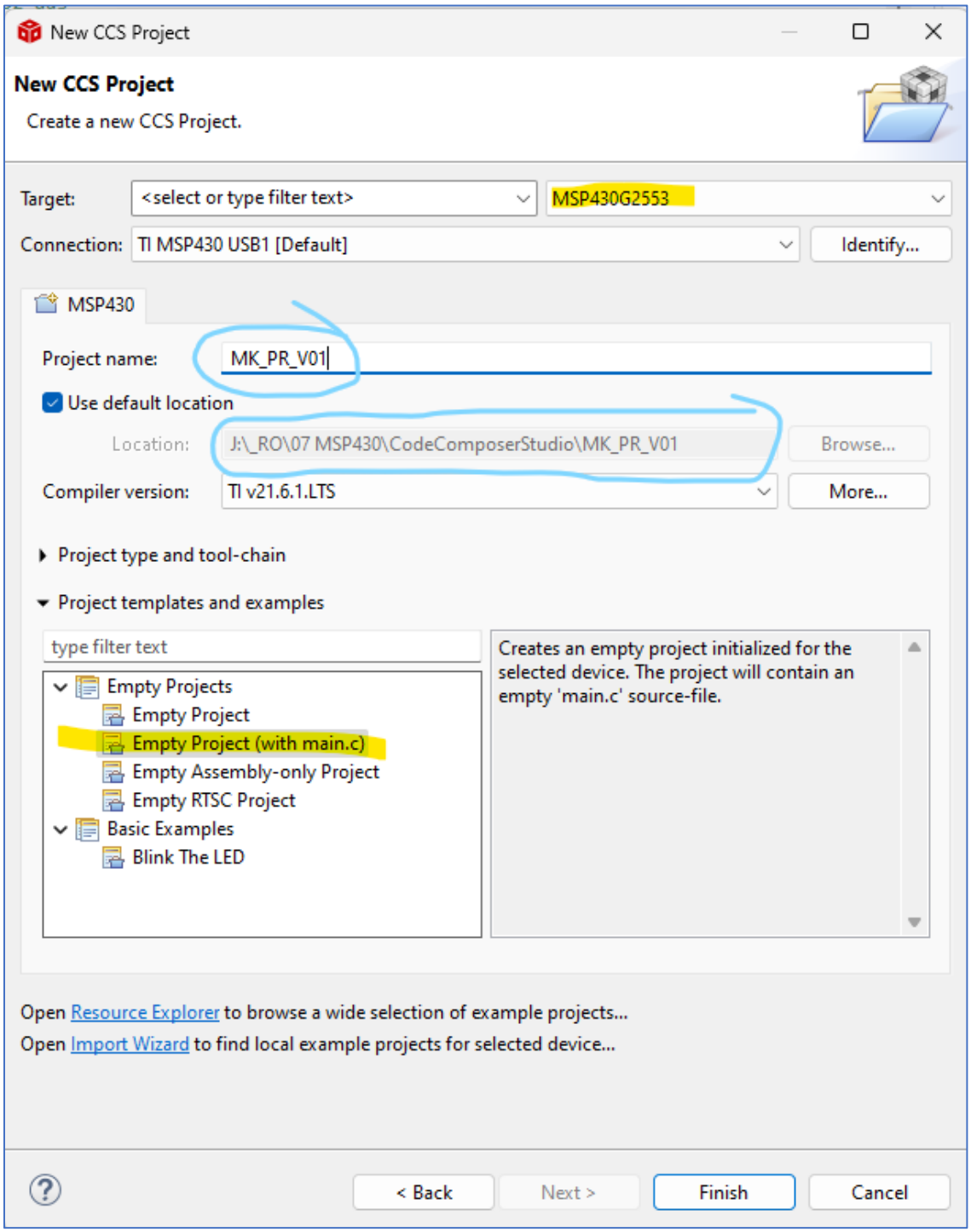
2.1 CodeComposerStudio

Erstellen Sie ein neues CCS-Projekt:



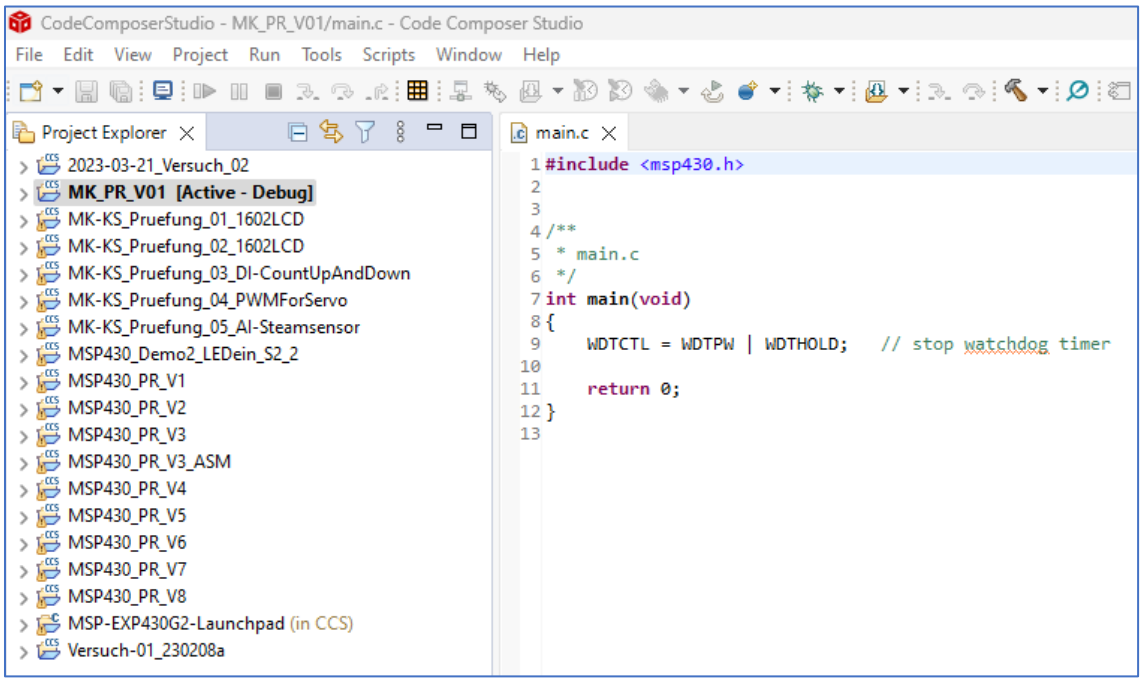


Konfigurieren Sie das Projekt.
Vergeben Sie einen sinnvollen Projektnamen:



Wichtig: Bei **Target** müssen Sie den **Controller** auswählen, der auf Ihrem Launch Pad verbaut ist. Siehe hierzu auch 1.2.

Nach dem Klicken von **Finish** öffnet sich das neu erstellte Projekt im Project Explorer. Die Datei **main.c** wird im Editor geöffnet und kann jetzt editiert werden.



2.2 Erste Schritte – Schalten der LEDs

2.2.1 Aufgabenstellung

Um die Entwicklungsumgebung kennen zu lernen und uns mit den Registern unseres MSP vertraut zu machen, wollen wir die LEDs auf dem LaunchPad zum Leuchten bringen.

2.2.2 Informationssammlung: Schaltplan

Zunächst möchten wir die grüne LED zum Leuchten bringen.
Dazu müssen wird im Schaltplan der Platine nachsehen, mit welchem Pin des MSP die grüne LED verbunden ist:

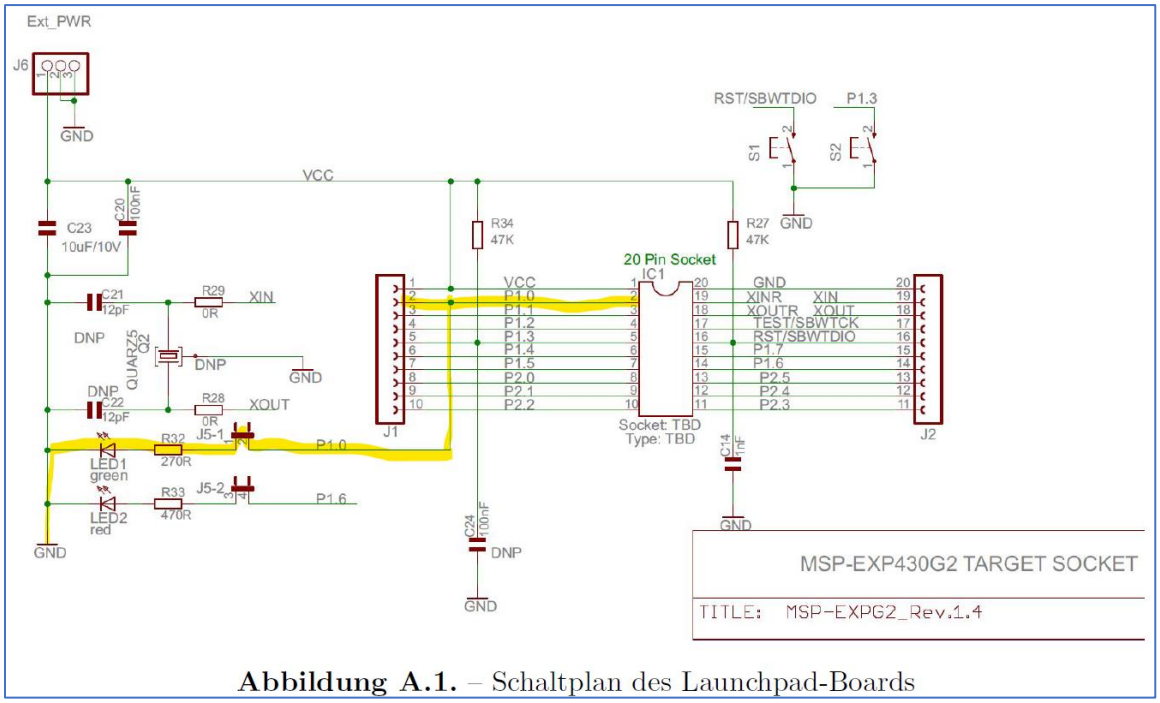
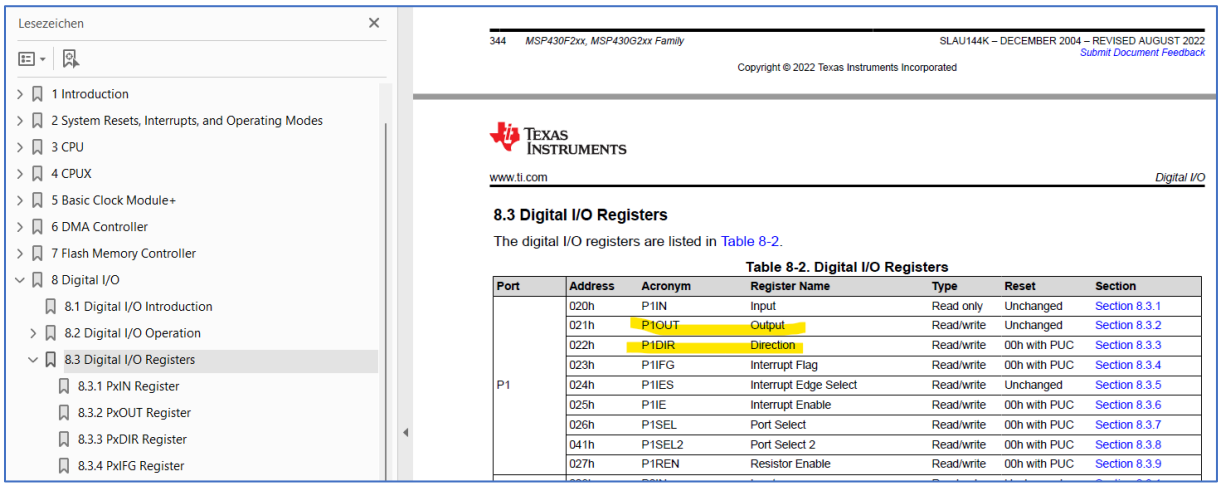


Abbildung A.1. – Schaltplan des Launchpad-Boards

Die grüne LED ist verbunden mit dem Pin _____
und somit mit Port _____
Wichtig ist, daß auf dem Board der Jumper _____ gesteckt ist, damit die LED mit GND verbunden ist.

2.2.3 Informationssammlung Datenblatt

Zum Schalten eines digitalen Ausgangs müssen bei unserem Kontroller zwei Register beschrieben werden. Informationen zu dein digitalen I/O Registern finden wir im **Datenblatt** zum Controller **SLAU144K** im Kapitel 8:



Wir müssen die Register **P1DIR** und **P1OUT** so konfigurieren, daß die LED an Pin P1.0 leuchtet.

Deshalb muß dieser Pin zuerst als Output konfiguriert werden und anschließend HI gesetzt werden.

8.3.3 PxDIR Register

Port x Direction Register

Figure 8-5. PxDIR Register

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 8-5. P1DIR Register Description

Bit	Field	Type	Reset	Description
7-0	PxDIR	R/W	0h	Port x direction. Each bit corresponds to one channel on Port x. 0b = Port configured as input 1b = Port configured as output

8.3.2 PxOUT Register

Port x Output Register

Figure 8-4. PxOUT Register

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 8-4. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	R/W	Unchanged	Port x output. Each bit corresponds to one channel on Port x. The reset value is undefined. When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

2.2.4 Kleine C-Auffrischung

Tragen Sie in die folgende Tabelle die Bedeutung der Operatoren ein

|	
||	
&	
&&	
~	

Tragen Sie in die folgende Tabelle die Resultate der Ausdrücke ein

0x01 | 0x02	
0x01 || 0x02	
0x01 & 0x02	
0x01 && 0x02	
~0x01	

Um Bits in einer Integer-Variablen zu setzen, kann man den Ausdruck **|=** verwenden.
Um Bits in einer Integer-Variablen zu löschen, kann man den Ausdruck **&=~** verwenden.
Somit kann man durch folgende Programmzeile Bit 3 eines Bytes setzen:

```
u8MyByte |= 0x08;
```

Erklären sie kurz, warum der Ausdruck **u8MyByte |= 0x08** das Bit 3 in **u8MyByte** setzt

_____

Mit dieser Programmzeile lassen sich die Bits 1 und 2 eines Bytes rücksetzen:

```
u8MyByte &=~ 0x06;
```

Erklären sie kurz, warum der Ausdruck **u8MyByte &=~ 0x06** die Bits 1 und 2 in **u8MyByte** zurücksetzt

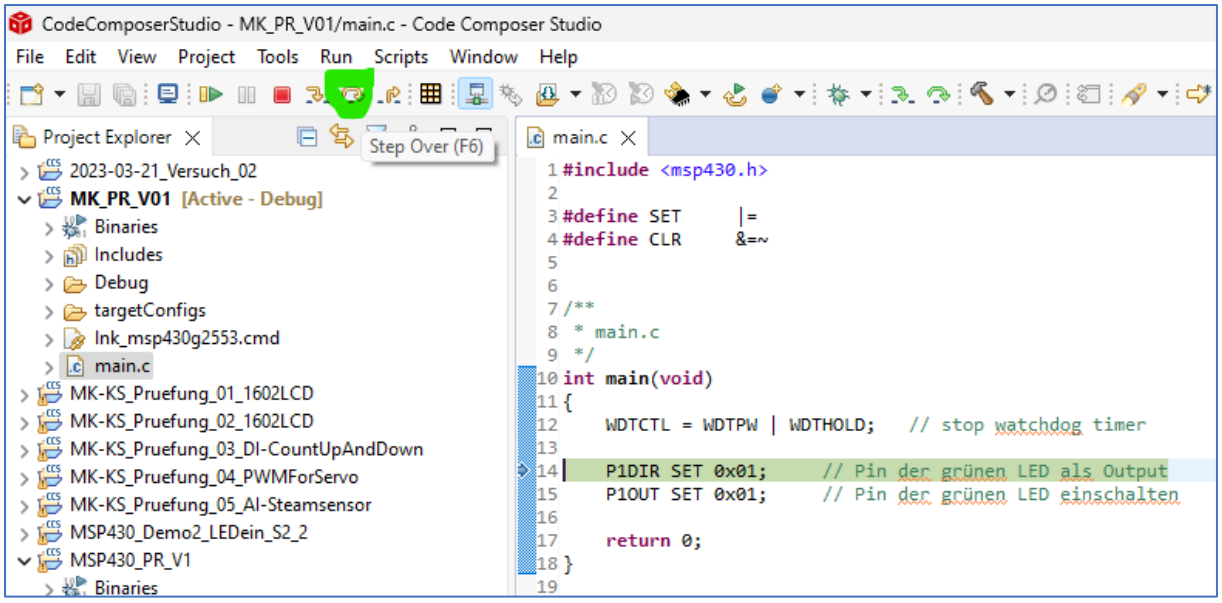
_____

Da die oben gezeigten C-Symbole für den ungeübten Programmierer etwas kryptisch erscheinen mögen, empfiehlt es sich, diese Ausdrücke in einem Makro zu verstauen:

```
#define SET     |=
#define CLR     &=~
```

2.2.5 Programmierung von main_01_00.c – Erste Schritte

Erweitern sie **main.c** so, daß die grüne LED eingeschaltet wird.
Verwenden sie beim Testen Ihres Programms die **Step Over (F6)** Funktionalität



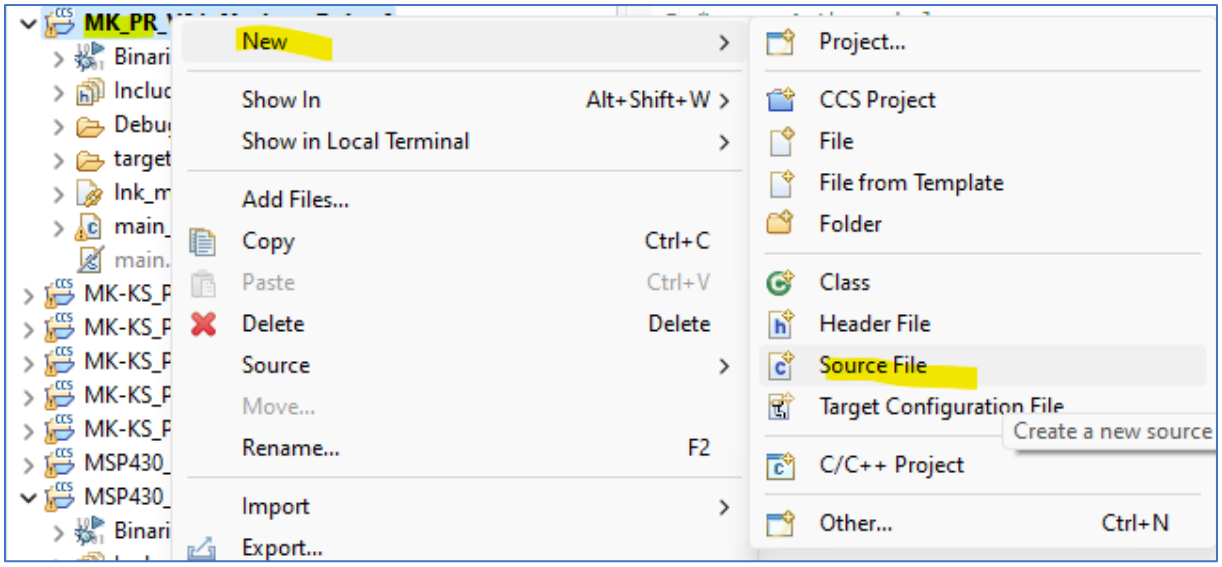
Es ist unschön, daß der Zugriff auf die LED über den Wert 0x01 erfolgt.
Besser lesbar wird das Programm durch folgende Verbesserung:

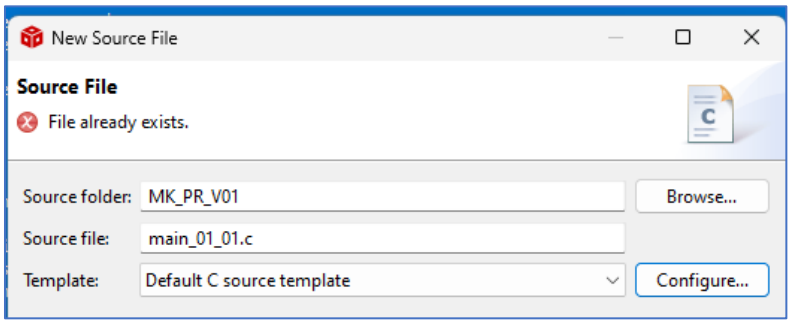
```
5
6 #define LED_green   0x01
7
8
9 /**
10  * main.c
11  */
12 int main(void)
13 {
14     WDTCTL = WDTPW | WDTHOLD;    // stop watchdog timer
15
16     P1DIR SET LED_green;      // Pin der grünen LED als Output
17     P1OUT SET LED_green;      // Pin der grünen LED einschalten
18
19     return 0;
20 }
```

Anmerkung:
Sollte bei Ihnen an Stelle der grünen die rote LED leuchten, dann geht es Ihnen wie mir:
Offensichtlich ist der Schaltplan in 2.2.2 fehlerhaft.

2.2.6 Programmierung von main_01_01.c – Periodisches Schalten eine LED

Erzeugen Sie - innerhalb des bestehenden Projekts - eine neue Datei mit Namen **main_01_01.c**.



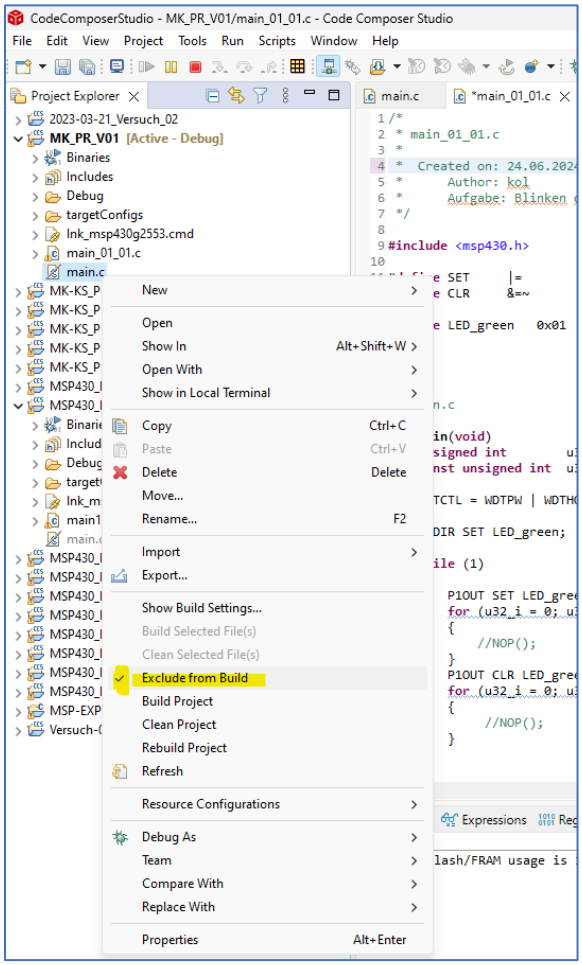


Erweitern Sie darin Ihr Programm so, daß die grüne LED periodisch ein und ausgeschaltet wird.

Verwenden Sie zur Verzögerung eine `for`-Schleife.

Verwenden sie die Variable `const unsigned long int c_u32CountMax`, um den maximal zulässigen Zählerstand der Schleife zu speichern.

Schließen sie die alte main Datei (**main.c**) vom Builden aus, indem sie im CCS die Option **Exclude from Build** für diese Datei einschalten:



*main_01_01.c

```c
/*
 * main_01_01.c
 *
 *  Created on: 24.06.2024
 *      Author: kol
 *      Aufgabe: Blinken der grünen LED an P1.0
 */

#include <msp430.h>

#define SET     |=
#define CLR     &=~

#define LED_green   0x01



/**
 * main.c
 */
int main(void)
{   unsigned long int        u32_i;                 // Variable für Zählschleife
    const unsigned long int  u32CountMax = 20000;   // Maximaler Zählerstand der Verögerungsschleife

    WDTCTL = WDTPW | WDTHOLD;   // stop watchdog timer

    P1DIR SET LED_green;     // Pin der grünen LED als Output

    while (1)
    {
        P1OUT SET LED_green;     // Grüne LED einschalten
        for (u32_i = 0; u32_i < u32CountMax; u32_i++)
        {
            //NOP();
        }
        P1OUT CLR LED_green;     // Grüne LED auschalten
        for (u32_i = 0; u32_i < u32CountMax; u32_i++)
        {
             //NOP();
        }
    }
```

2.2.7 Programmierung von main_01_02.c – Schalten von zwei LEDs

Erzeugen Sie eine neue Datei mit Namen **main_01_02.c**.

Erweitern Sie darin Ihr Programm main_01_01.c. so, daß die grüne und die rote LED periodisch ein- und ausgeschaltet werden.

2.3 Auswerten des Tasters S2 – main_01_03.c

2.3.1 Aufgabenstellung

In diesem Programm wollen wir den Wert des digitalen Eingangs am Taster S2 auswerten. Wenn der Taster gedrückt ist, sollen die LEDs abwechselnd blinken (Rot an -> grün aus, Rot aus -> grün an).
Man beachte, daß es verschiedene Versionen des TI LaunchPads gibt. Nicht auf allen Boards ist S2 mit einem **Pullup-Widerstand** beschaltet. Deshalb muß diese Funktionalität über die entsprechenden Register eingestellt werden.

2.3.2 Informationssammlung

Nötige Hintergrundinformationen zum Setup der entsprechenden Register finden Sie in den Kapiteln 2.2.2 und 2.2.3.

Außerdem von Interesse:

8.3.3 PxDIR Register

Port x Direction Register

Figure 8-5. PxDIR Register

7	6	5	4	3	2	1	0
PxDIR							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 8-5. P1DIR Register Description

Bit	Field	Type	Reset	Description
7-0	PxDIR	R/W	0h	Port x direction. Each bit corresponds to one channel on Port x. 0b = Port configured as input 1b = Port configured as output

8.3.9 PxREN Register

Port x Pullup or Pulldown Resistor Enable Register

Figure 8-11. PxREN Register

7	6	5	4	3	2	1	0
PxREN							
rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0	rw-0

Table 8-11. PxREN Register Description

Bit	Field	Type	Reset	Description
7-0	PxREN	R/W	0h	Port x pullup or pulldown resistor enable. Each bit corresponds to one channel on Port x. When the port is configured as an input, setting this bit enables or disables the pullup or pulldown 0b = Pullup or pulldown disabled 1b = Pullup or pulldown enabled

8.3.2 PxOUT Register

Port x Output Register

Figure 8-4. PxOUT Register

7	6	5	4	3	2	1	0
PxOUT							
rw	rw	rw	rw	rw	rw	rw	rw

Table 8-4. PxOUT Register Description

Bit	Field	Type	Reset	Description
7-0	PxOUT	R/W	Unchanged	Port x output. Each bit corresponds to one channel on Port x. The reset value is undefined. When I/O configured to output mode: 0b = Output is low. 1b = Output is high. When I/O configured to input mode and pullups/pulldowns enabled: 0b = Pulldown selected 1b = Pullup selected

8.3.1 PxIN Register

Port x Input Register

Figure 8-3. PxIN Register

7	6	5	4	3	2	1	0
PxIN							
r	r	r	r	r	r	r	r

Table 8-3. PxIN Register Description

Bit	Field	Type	Reset	Description
7-0	PxIN	R	Unchanged	Port x input. Each bit corresponds to one channel on Port x. The reset value is undefined. 0b = Input is low 1b = Input is high

2.3.3 Fragen zu den verwendeten Registern

Der Pin des Tasters S2 soll ein digitaler Input werden. Beantworten Sie die folgenden Fragen:

Mit welchem Pin des MSP430 ist S2 verbunden?

Wie sind die Register P1DIR, P1REN und P1OUT zu beschreiben?
P1DIR =
P1REN =
P1OUT =

In welchem Register und wie kann man den Zustand des Pins des Schalters lesen?

Welcher Wert steht in P1IN im Bit des Schalters, wen der Schalter gedrückt ist?

2.3.4 Programmierung

Erzeugen Sie eine neue Datei mit Namen **main_01_03.c**.

Erweitern Sie darin Ihr bestehendes Programm **main_01_02.c** zunächst so, daß die grüne und die rote LED periodisch ein- und ausgeschaltet werden.

Wenn das funktioniert, erstellen Sie eine weitere Datei **main_01_04.c**.
Erweitern darin Ihr Programm so, daß die LEDs abwechselnd blinken.