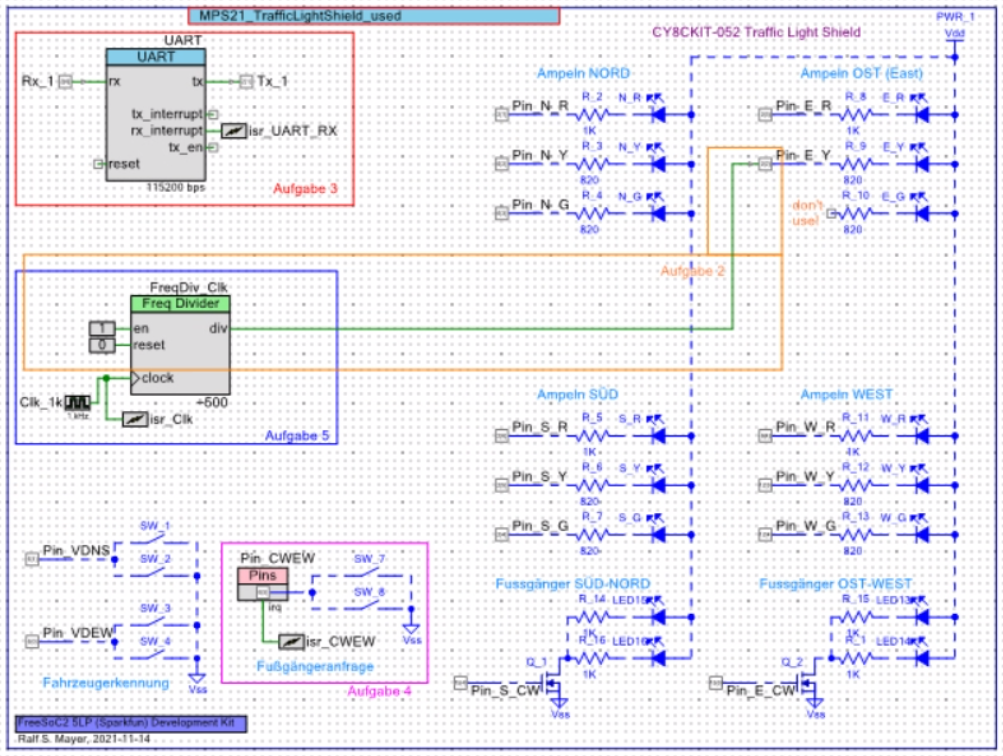
Praktikum 3

Lernziele: Verarbeitung von Ereignissen als Interrupts oder direkt in der Hardware (SoC)

**1. Starten Sie PSoC-Creator und laden Sie das Projekt Termin 3: MPS21\_Prakt\_3.**

**a. Betrachten Sie das neue Hardware-Design in TopDesign.cysch.  
(Nicht erschrecken: enthält jetzt alle Funktionen, die mit dem Traffic-Light-Shield und zusammen mit der UART möglich sind. Nur ein Teil wird Termin 3 benötigt.)**



**b. Starten Sie TeraTerm und verbinden Sie mit COM<xy> (Datei > Neue Verbindung >Seriell). Beachten Sie die UARTEinstellungen in TopDesign.cysch.**

Einstellungen > Serieller Port:

- speed (baud): 115200  
- data: 8  
- parity: none  
- stop: 1

Einstellungen > Terminal:

Übertragen: CR, Absenden: CR+LF

**c. Betrachten Sie die HW-Verbindung von FreqDiv\_Clk (div) mit Pin\_E\_Y.  
Welche Funktion hat FreqDiv\_Clk (div) und wie ist Pin\_E\_Y konfiguriert?**

Die FreqDiv\_Clk teilt die Frequenz einer clock (Clk\_1k) durch einen spezifizierten Wert (500) und gibt das Ergebnis als Takt an den Pin\_E\_Y weiter: 2Hz -> 2mal die Sekunde.

Der Pin wiederum verbindet über den Takt der FreqDiv einen Widerstand von 820 Ohm und eine gelbe Diode (E\_Y) mit dem Stromfluss zu GND. Der Pin kann dann den Stromfluss zu der gelben Diode schließen und diese leuchtet.

**d. Welche Funktion erfüllen die Komponenten isr\_UART\_RX, isr\_Clk, und isr\_CWEW?**

isr\_UART\_RX, isr\_Clk, isr\_CWEW sind alle 3 Komponenten, die für definierte Interruptschnittstellen der Hardware zuständig sind.

Diese Schnittstellen unterbrechen, falls in der angeschlossenen Hardware eine Aktion / ein Event vorliegt, den normalen Programmabfluss um auf diese Events im Programm entsprechend eingehen zu können, und anschließend mit dem normalen Programm an entsprechender Stelle fortsetzen zu können.

isr\_UART\_RX: Interrupt Service bei Tastatureingaben   
isr\_CWEW: Interrupt nach Knopfdruck  
isr\_Clk: Interrupt Service nach Taktsignal

**2. Betrachten Sie Pin\_E\_Y**

**a. Muss für die Funktion der LED E\_Y etwas im Code konfiguriert und/oder initialisiert werden?**

Ja. Es müssen 2 Makros für den Zustand an und aus (0u) und (1u) der LEDs definiert werden, um mit diesen dann entsprechend eine Stromzufuhr auf dem zugehörigen Pin des Stromkreises steuern zu können.

**b. Mit welcher Frequenz blinkt die LED?**

Clk\_1k: 1kHz = 1000 Hz = 1000x pro Sekunde  
1000x/sec \* 60 = 60000x/min  
60000x/min / 500 = 120x/min

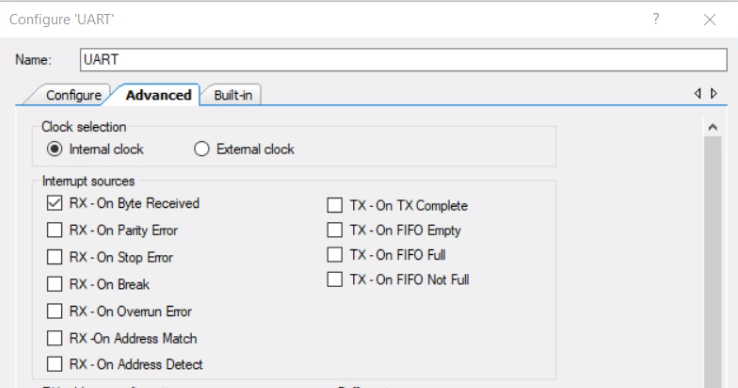
Die LED blinkt 120-mal die Minute

**c. Wird das Blinken durch das Programm main in irgendeiner Weise beeinflusst?  
(Testen Sie dies, z.B. durch eine Delay im Programm)**

Nein, die main feuert über eine Millionen Mal pro Sekunde – hier ist ein so geringer Einfluss vorhanden, dass dieser nicht spürbar ist.

**3. UART-Interrupt**

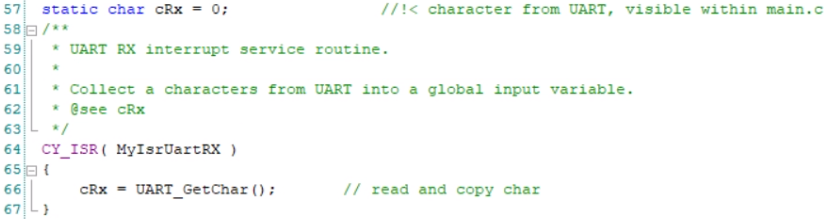
**a. Welches Ereignis löst den UART-Interrupt aus? (Hinweis: UART-Komponente öffnen, Reiter advanced)**

Der UART-ISR\_RX Interrupt wird beim Ereignis OnByteReceived durch die ISR ausgelöst:  
  
Hierfür wird eine Char Flag-Variable über MyIsrUartRX gesetzt und im loop behandelt.

**b. Wo im Code ist die ISR (bereits vollständig) implementiert, und wo wird sie registriert?**

Die ISR ist von ca. Zeile 64-67 bereits als Buttonpress auf MyIsrUartRX implementiert. Dabei wird eine Char Flag-Variable auf cRx gesetzt und im loop behandelt.

In Zeile ~57 wird über static char cRx dieser character registriert.



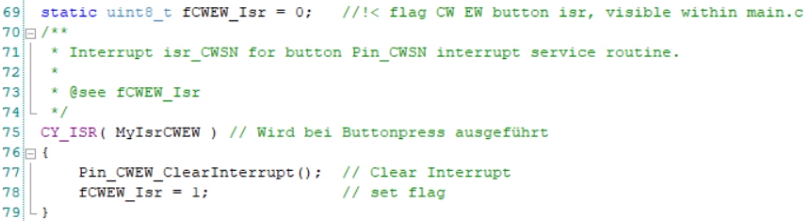
**c. Das Menu-Template reagiert jetzt nur auf den Interrupt, erklären Sie den Code**

Die Eingabe eines chars wird immer durch den ISR\_RX Interrupt aufgerufen und in cRx eingelesen. Nur diese chars werden in der Dauerschleife dann in der if Bedingung aufgegriffen und das Menü-Programm nur mit diesen betreten.

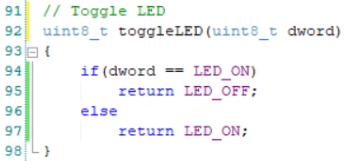
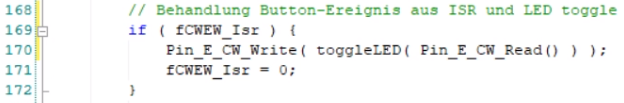
**4. Button-Ereignis aus Interrupt MyIsrCWEW**

**a. Wo im Code ist die ISR (bereits vollständig) implementiert, und wo wird sie registriert?**

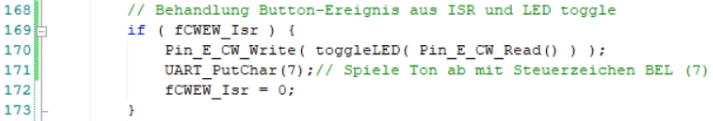
Dieser Interrupt ist von ~Zeile 75 - 79 definiert:



**b. Toggeln Sie die LED Pin\_E\_CW wenn der Button gedrückt wird. Erweitern Sie dazu den vorgegebenen Code.**

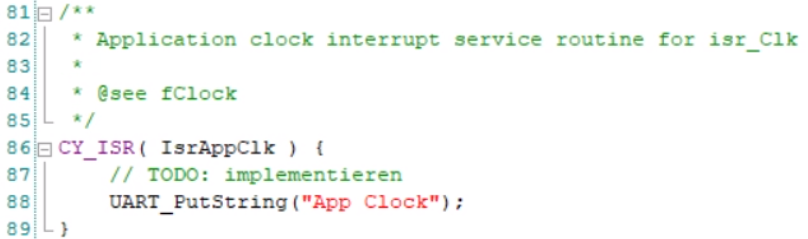
**  
  
**

**c. Freiwillig: geben Sie ein Tonsignal auf Ihrem Terminal aus, wenn die Taste gedrückt wird. Hinweis: ASCII-Tabelle, Steuerzeichen!**

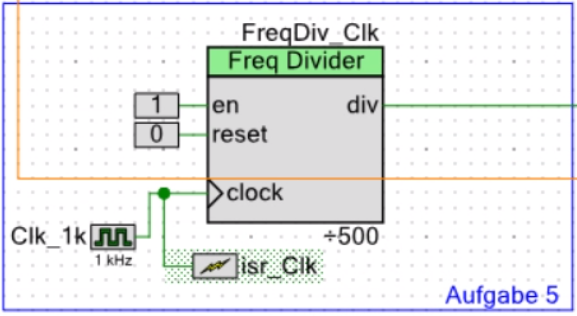
****

Steuerzeichen BEL („BELL“) **wird zum** Tonausgeben verwendet.

**5. Timer-Ereignis isr\_Clk**

****

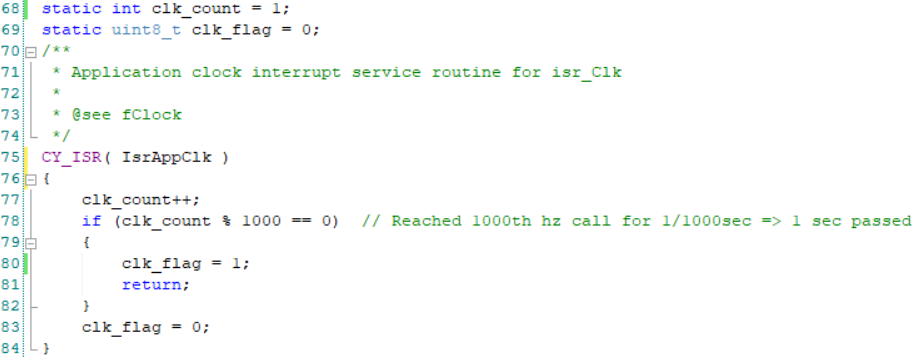
**a. Wie häufig wird dieser Interrupt ausgelöst?**



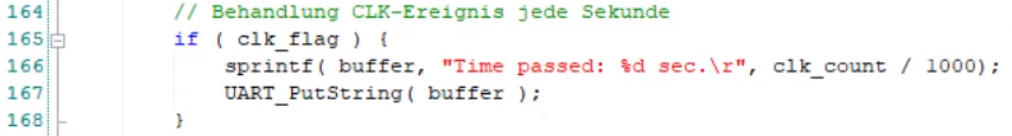
Da die isr\_Clock noch direkt an der Clk\_1k Leitung **vor** dem FrqDiv\_Clk liegt, fängt der Interrupt noch die 1kHz ab.  
Hz zu Zeit: (1 / Hz)s => Ausgelöst jede 1/1000 Sekunde.

**b. Wie könnte man die vergangene Zeit seit dem Start des Programms messen?**

Man könnte eine Zählervariable hochzählen, wenn das Ereignis feuert und diese nach einer vergangenen Unit - Timestamp - Minute ausgeben.  
 **Zeigen Sie diese über das Menu an.**



**c. Setzen Sie in der ISR alle Sekunden ein Flag, welches Sie in Ihrem main-Programm auswerten. Zeigen Sie die seit Programmstart vergangene Zeit kontinuierlich in ein und derselben Terminalzeile an. Hinweis: ASCII-Steuerzeichen!**



Zweiter Aufgabenteil siehe bereits Aufgabe 5b.

**e. Wie oft - schätzen Sie – wird die Endlosschleife in main pro Sekunde durchlaufen? Messen Sie! (Dazu ggf. c. und d. auskomment**ieren)

Schätzung: Ohne Interrupts wird die Endlosschleife jedes Mal in der Baudrate auf einen empfangenen char überprüfen, also 115200bps = 115200Hz = 115,0 kHz

Messung: ~800 kHz = ~800000Hz