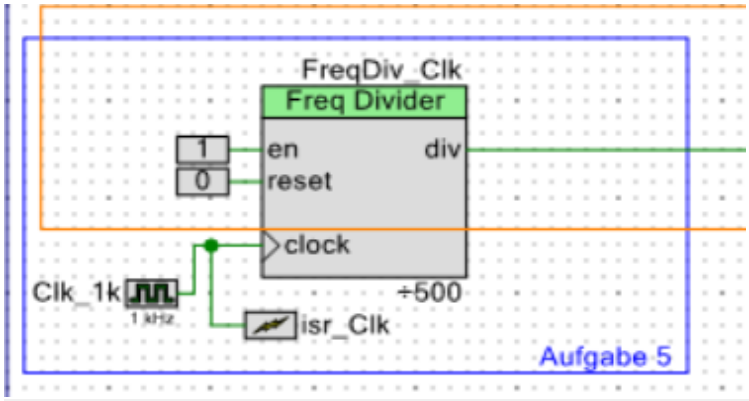
Mikroprozessorsysteme, Praktikum 2, SS22
Student 1: Damir Maksuti
Matrikelnr: 765984
Student 2: Jamil Boujada
Matrikelnr: 769479

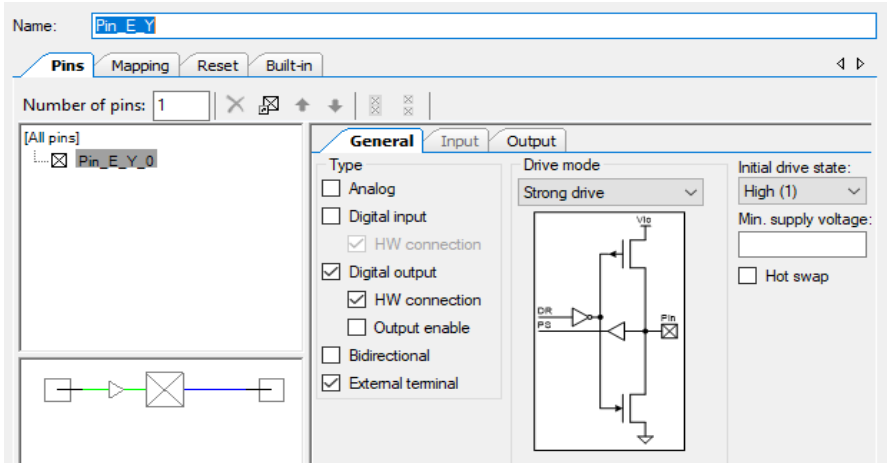
1. Starten Sie PSoC-Creator und laden Sie das Projekt Termin 2: MPS22_Prakt_2.

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 (Datei > Neue Verbindung >Seriell). Beachten Sie die UART Einstellungen in TopDesign.cysch.

C). Betrachten Sie die HW-Verbindung von FreqDiv_Clk (div) mit Pin_E_Y. Welche Funktion hat FreqDiv_Clk (div)?





Der Frequenzteiler vermindert eine Eingangsfrequenz in ein bestimmtes Teilungsverhältnis. Häufig werden hierzu T-Flipflops verwendet mit denen ein (Rückwärts-)Zähler realisiert werden kann.

Hierzu haben wir versucht die Informationen aus dem Datasheet in eine uns verständlichen Sprache zu übersetzen (Bei Fehler bitte kommentieren).

I/0 Verbindungen:

En – Input: Bei einem Low und steigender Flanke in der Clock wird der interne counter nicht inkrementiert und die Ausgangsfrequenz (div) nicht geändert. Der En Input gibt also an, wann gezählt wird und wann nicht.

Reset – Input: reseted den internen counter bei der nächsten steigenden Flanke. Infolgedessen wird ein low ausgegeben bis zur nächsten fallenden Flanke. Ab diesem Zeitpunkt startet der Zähler wieder neu.

clock – Input: bestimmt die Teilfrequenz.

Div: ist der Output, an dem die verminderte Frequenz ausgegeben wird. Das Teilverhältnis beträgt 500. Die Eingangsfrequenz 1 Khz. Daraus ergibt sich die folgende Rechnung.

Div: 1000 Hz / 500 = 2 Hz

wie ist Pin_E_Y konfiguriert?

Pin_E_Y Konfigurationen: Initial drive state = high, HW connection = on, Output enable = off

D). Welche Funktion erfüllen die Komponenten isr_UART_RX, isr_Clk, und isr_CWEW? Beantworten Sie als Vorbereitung alle Fragen und dokumentieren Sie später im Protokoll.

Interrupt Service Routinen im Szenario:

isr_UART_RX: Interrupt beim Lesen eines Characters.

isr_Clk: Der Interrupt erfolgt in regelmäßigen Abständen und wird von der clock ausgelöst. MIthilfe einer IF Bedingung in der Isr Funktion können wir einen Sekundentakt erzeugen.

isr_CWEW: Nachdem einer der Tasten (SW7/SW8) gedrückt wird und sich der Pegel auf low ändert, wird dieser interrupt ausgelöst.

Div ist der Output, an dem das Clock-Signal durch den Wert 500 geteilt ausgegeben wird.

Div => 1000 Hz / 500 = 2 Hz

Aufgabe 2 - Pin_E_Y:

Muss für die Funktion der LED `E_Y` etwas im Code konfiguriert und/oder nitialisiert werden?

Nein, da alle nötigen Konfigurationen an der hardware vorgenommen werden können.

Mit welcher Frequenz blinkt die LED?

2Hz

Wird das Blinken durch das Programm main in irgendeiner Weise beeinflusst?

Nein, da die clock unabhängig vom Programmablauf den Takt vorgibt

Aufgabe 3 - Interrupt UART

Welches Ereignis löst den UART - Interrupt aus?

Der Empfang der Eingabe eines Characters.

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

Implementierung

```
static char cRx = 0;                    //!<


/**
 * UART RX interrupt service routine.
 *
 * Collect a characters from UART into a
 * @see cRx
 */
CY_ISR( MyIsrUartRX ) {
    cRx = UART_GetChar();        // read a
}
```

Registrierung

```
 */
int main(void)
{
    /* Initialize and start UART, register ISR */
    UART_Start();                       // start UART
    isr_UART_RX_StartEx( MyIsrUartRX ); // register ISR, enable itr uart
```

Das Menu-Template reagiert jetzt nur auf dem Interrupt, erklären Sie den Code

cRx wird mit 0 initialisiert. Damit für die Überwachung der UART nicht mehr das Hauptprogramm zuständig ist, registrieren und implementieren wir die ISR MyIsrUartRx. Die If Anweisung in der Forschleife wird dann ausgeführt, wenn vorher ein interrupt eingetreten ist. Sollte die If Anweisung ein true ergeben, so wird der eingelesen Char temporär zwischengespeichert und die Variable cRx auf 0 zurückgesetzt.

Aufgabe 4 - Interrupt Button-Ereignis
Wo im Code ist die ISR (bereits vollständig) implementiert, und wo wird sie registriert?

Implementierung

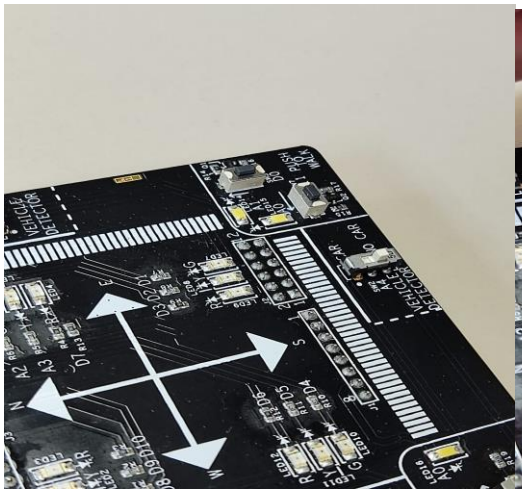
```
static uint8_t fCWEW_Isr = 0;   //!< flag CW EW button isr, visible within main.c
/**
 * Interrupt isr_CWSN for button Pin_CWSN interrupt service routine.
 *
 * @see fCWEW_Isr
 */

// Fußgängeranfrage
CY_ISR( MyIsrCWEW ) {
    /* Clear Interrupt first ! */
    Pin_CWEW_ClearInterrupt();
    fCWEW_Isr = 1;              // set flag
}
```

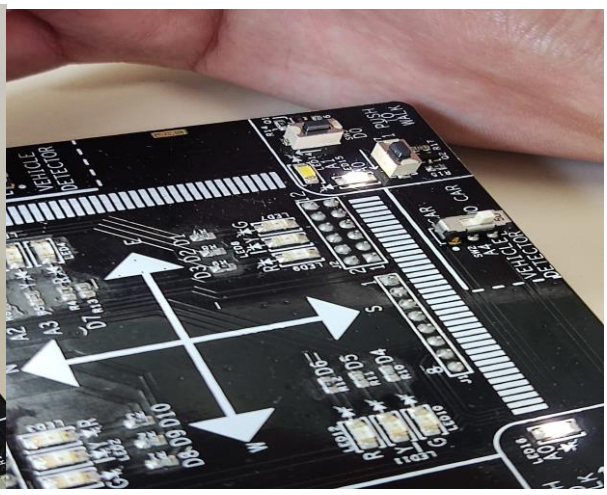
Registrierung

Toggeln Sie die LED Pin_E_CW wenn der Button gedrückt wird.

LEDs sind aus



Nach dem Drücken der Taste



Im Praktikum haben wir gemerkt, dass diese Variante führte nur unregelmäßig zum gewünschten Ergebnis führt.

```
// Behandlung Button-Ereignis aus ISR
if ( fCWEW_Isr ) {
    // TODO: implementieren und LED Pin_E_CW toggeln!
    fCWEW_Isr=0;
    Pin_E_CW_Write(!Pin_E_CW_Read());

}
```

Hier steuern wir die LEDs mithilfe einer Variablen, ohne dass wir den Pin_E_CW status abfragen müssen. Mit der Variablen CWEW_LED merken wir uns den nächsten gewünschten Zustand.

```
    // Behandlung Button-Ereignis aus ISR
    if ( fCWEW_Isr ) {
        // TODO: implementieren und LED Pin_E_CW toggeln!
        if(CWEW_LED){

        Pin_E_CW_Write(1);
        CWEW_LED = 0;

        }else{
         Pin_E_CW_Write(0);
        CWEW_LED = 1;
        }

         fCWEW_Isr=0;



    }
```

Aufgabe 5 - Interrupt Timer-Ereignis
Wie häufig wird das Interrupt ausgelöst?

Bei einer Frequenz von 1000 Hz wird der Interrupt 1000mal die Sekunde ausgelöst

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

```
 */
static int clkInteruptCounter = 0;
static int secondsSinceStart = 0;
static int newSecondHasStarted = 0;

// Sekundentakt
CY_ISR( IsrAppClk ) {
    // TODO: implementieren
    clkInteruptCounter++;
    if(clkInteruptCounter == 1000){

        clkInteruptCounter = 0;
        secondsSinceStart++; // setzen des Flags
        newSecondHasStarted = 1; // start loop iteration
    }

}
```

Nach dem 1000sten Interrupt wird die IF-Bedingung in der wahr und somit die boolische Variable NewSecondHasStarted auf true gesetzt. Ist die Bedingung wahr so wird auch die Variable secondsSinceStart inkrementiert. Zu diesem Zeitpunkt ist secondsSinceStart = 1.

```
8
9           if( newSecondHasStarted && secondsSinceStart < 2 ){
0               loopCount++;
1           }
2       } // end for
3   }
4
5   /* [] END OF FILE */
6
```

Die obenstehende If Bedingung in der For-loop wird true, sobald NewSecondHasStarted true ist und secondsSinceStart < 2 ist. Damit haben wir den Beginn sowie das Ende der zweiten Sekunde markiert. Nur innerhalb dieser einen Sekunde wird die lokale Variable loopCount mit jeder Iteration inkrementiert. Die Lösung haben wir uns im Praktikum erarbeitet. Im Nachgang hätten wir wahrscheinlich auch nur mit secondsSinceStart arbeiten können. In etwa so:

if (secondsSinceStart && secondsSinceStart < 2) loopCount++ ;

```
126             // +++ Menuesteuerung +++
127             switch ( c ) {
128                 case 'q':
129                 case 'Q':   // exit
130                     CySoftwareReset();
131                     break;
132                 case 'z':   // zeitabfrage
133                     sprintf(buffer, "Seconds: <%d> \r", secondsSinceStart);
134                     UART_PutString(buffer);
135                     break;
136                 case 'l':
137                      sprintf(buffer, "Loop Count: <%d> \n\r", loopCount);
138                      UART_PutString(buffer);
139                     break;
140                 default:
141                     sprintf(buffer, "invalid Input: <%c>", c);
142                     UART_PutString( buffer );    // Buchstabe auf Bildschirm ausgeben
143                     break;
144             }  // end switch
145         } // end if cRx
```

Im Folgenden sind zwei switch cases implementiert, um die Sekunden seit Programmstart und die Anzahl der Iterationen pro Sekunde ausgegeben. Die Anzahl der Iterationen pro Sekunden wird nur einmalig festgehalten.

Zeigen Sie die seit Programmstart vergangene Zeit kontinuierlich in ein und derselben Terminal Zeile an.

ASCII Steuerzeichen:

```
// Applikationsschleife
for(;;)
{
    /*
     * Character aus Uart ISR abfragen
     * Beipsiel für einfache Menuesteuerung
     * Achtung: Für UART in diesem Programm ist ein zusätzlicher FTDI USB
     */
    sprintf(buffer, "Seconds: <%d> \r", secondsSinceStart);
    UART_PutString(buffer);
    if ( cRx ) {
```

Wie oft - schätzen Sie – wird die Endlosschleife in main pro Sekunde durchlaufen?

Die ISR am Timer unterbricht den Programmfluss 1000mal pro Sekunde und führt ihn temporär an einer anderen Stelle fort. Wir vermuten, dass dadurch wenige Schleifendurchgänge möglich sind als ohne Messung. Wir schätzen 600 Iterationen.

Beobachtungen

Mit output pro Iteration nur 824 Iterationen pro Sekunde. Ohne den UART Output Steigert sich die Anzahl der Iterationen auf 775000

```
Welcome! MPS21_Prakt_3, FreeSoC2, PSoC 5LP, ver. 1.0
Jamil Boujada, Damir Maksuti (Template MPS 3), May 23 2022 11:45:35
Seconds: <12>
Seconds: <19>
Loop Count: <824>
Seconds: <32>
```