MPS – Praktikum 2 – Sebastian Zill (769544)

Lernziele: PSoC Creator einrichten, Evaluierungs-Board anschließen und programmieren, Ein-Ausgabe über PC-Terminal.
Uart, I/O-Peripherie. C-Schlüsselwort volatile

1. **Board auf DEBUG Anschluss an PC USB-Port anschließen und Gerät installieren**
   PSoC-Creator mit Projekt *MPS21_Prakt_2.cyprj* **starten.**
   TeraTerm mit neuer seriellen Schnittstelle KitProg USB-UART(COM<xy>) **starten.**

   Einstellungen in geöffnetem Terminal vornehmen:
   Einstellungen > Serieller Port:
   - speed (baud): 115200
   - data: 8
   - parity: none
   - stop: 1

   Einstellungen > Terminal:
   Übertragen: CR, Absenden: CR+LF

   PSoC Creator: Build > Generate Application
   PSoC Creator: Debug > Program main.c auf board übertragen und starten.

2. **Erzeugen Sie eine einfache Bildschirmausgabe:**

```
5   #define _PROJECT_NAME   "MPS21_Prakt_2" // Project name
6   #define _VERSION_NR     "ver. 1.2"      // Version number. Please update
7   #define _PROCESSOR      "PSoC 5LP"      // Processor type
8   #define _DEVICE_NAME    "FreeSoC2"      // Board or device type
9   #define _AUTHOR         "Sebastian Zill"// Author
10  #define _MENU           "Instructions: \n\r Press x to quit! \n\r Press g to toggle green light \n\r Press
    y to toggle yellow light \n\r Press r to toggle red light \n\r Press the botton(s) on the very bottom to
    toggle white light \n\r"
11
12  // build string for display
13  #define _VERSTR       (_PROJECT_NAME" ("_VERSION_NR") - "_DEVICE_NAME", "_PROCESSOR", \n\r"\
14  _AUTHOR", "__DATE__" "__TIME__)
```

```
25  int main(void)
26  {
27      // initialize
28      UART_Start();                       // start UART
29      CyGlobalIntEnable;                  // enable global interrupts
30
31      // welcome and info text
32      sprintf( buffer, "\n\rWelcome! %s\n\r %s", _VERSTR, _MENU);   // format butter
33      UART_PutString( buffer );                                     // print buffer to UART
```

3. **Vorbereitung: Betrachten Sie *TopDesign*.**

a. **Überlegen Sie, wann die LEDs R, Y, G leuchten? Active high oder low?**
   Active-high, da nach Schaltplan die Diode direkt eine Versorgungsspannung von PWR_1 erhält und der Strom durch einen 1K / 820 Ohm zu GRND fließt.

b. **Überlegen Sie, wann LED_CW leuchtet? Transistor Q_2 wirkt als Inverter!**
   Da der Schaltkreis erst geschlossen ist, wenn der Knopf E_CW den Transistor als Inverter öffnet, immer dann wenn der Knopf gedrückt wird - active-low

c. Welchen Pegel nimmt Pin_CWEW an, wenn SW_7 bzw. SW_8 nicht gedrückt ist und wenn gedrückt ist.
   Beide Schalter als Tristate-Buffer öffnen ihre Leitung erst, wenn Sie gedrückt sind, unter der Annahme auf den Pins liegt ebenfalls ein GND zum Anschluss:
   Einer der Schalter gedrückt: Pegel LOW
   (Schaltkreis geschlossen - Positiv fließt nach GND)
   Keiner der Schalter gedrückt: Pegel HIGH
   (Schaltkreis nicht geschlossen - Positiv von isr_CWEW bleibt)

4. Lesen vom Zeichen vom Terminal und Schalten der LED(s)

```
    // declarations
    uint8_t chr = 0;                    // chars from UART


    /* +++ application loop - iterated permanently +++ */
    for(;;)
    {
        /* +++ get uart char +++ */
        chr = UART_GetChar();           //  = 0: while no input
                                        // != 0: got input
        // TASK 4
        // +++ Evaluate chr +++
        if ( chr != 0 )                 // gotten chr input
        {
            switch (chr)
            {
                case 'x':
                    UART_PutString( "... Bye! \n\r" );
                    return 0;

                case 'g':
                    UART_PutString( "Green light toggled! \n\r" );
                    if ( Pin_N_G_Read() == LED_OFF )
                        Pin_N_G_Write( LED_ON );
                    else
                        Pin_N_G_Write( LED_OFF );
                    break;

                case 'y':
                    UART_PutString( "Yellow light toggled! \n\r" );
                    if ( Pin_N_Y_Read() == LED_OFF )
                        Pin_N_Y_Write( LED_ON );
                    else
                        Pin_N_Y_Write( LED_OFF );
                    break;

                case 'r':
                    UART_PutString( "Red light toggled! \n\r" );
                    if ( Pin_N_R_Read() == LED_OFF )
                        Pin_N_R_Write( LED_ON );
                    else
                        Pin_N_R_Write( LED_OFF );
                    break;

                default:
                    sprintf( buffer, "erwarte ein <x>, <g>, <y> oder <r>, aber <%c> wurde eingegeben\n\r",
                    chr );
                    UART_PutString( buffer );
                    break;
            }
            chr = 0;                    // reset chr for ISR loop
        }
```

5. Button CWEW abfragen und toggeln der weißen LEDs

```
        // TASK 5
        // +++ evaluate button CWEW +++
        bool pressed = false;
        if ( Pin_CWEW_Read() == 0 )     // if button pressed
        {
            if(!pressed)                // and not pressed before (prellen)
            {
                // toggle led accordingly
                if ( Pin_E_CW_Read() == LED_OFF )
                    Pin_E_CW_Write( LED_ON );
                else
                    Pin_E_CW_Write( LED_OFF );
                pressed = true;         // set pressed true to prevent double activating
            }
        }
        else if( Pin_CWEW_Read() != 0 ) // if button released
        {                               // and pressed still true,
            if(pressed)
                pressed = false;        // reset to false
        }
```

6. Blinken der roten LED unter DEBUG und „optimierten“ RELEASE

```
110     // TASK 6
111     // +++ LED blinken lassen, warten mit Schleife +++
112     volatile uint32_t loop;          // works with DEBUG, not with RELEASE
113     // why???
114     // loop not used initially - compiler "optimizing" it away
115     // what does 'volatile' do?
116     // volatile tells the compiler not to optimize anything that has to do with the volatile variable.
117     Pin_N_R_Write( LED_ON );                    // turn on
118     for ( loop = 0; loop < 1000000; loop++ );   // wait with loop
119     Pin_N_R_Write( LED_OFF );                   // turn off
120     for ( loop = 0; loop < 1000000; loop++ );   // wait with loop
```

7. Testen Sie jetzt mit 6. nochmal die Funktion der Keyboard-Eingabe und des Buttons

a. Sind Sie mit der Schnelligkeit der Reaktion auf Terminal-Eingabe zufrieden?

Nein - eine Taste muss ggf. lange gedrückt gehalten werden. Das Programm muss die beiden for-Warteschleifen durchlaufen - die Eingabe ist währenddessen blockiert.

b. Funktioniert der Button überhaupt noch zuverlässig?

Nein. Es besteht das gleiche Problem wie oben.

c. Warum ist das so? *Könnte man so eine Lösung erfolgreich verkaufen?*

Die Eingabe ist während den vorherigen Schleifen blockiert, bis man an der entsprechenden Stelle im code „angekommen ist“.

Nein, eine Wartezeit entbindet einige Anwendungsfälle von Ihrem Zweck und das Programm wird hierdurch unbrauchbar.