Gruppe: Leander Stark, Jonas Dollinger Datum: 16.11.2020

Aufgabe 3

- Einfache Anwendung des kooperativen Multitasking.
- Timer im free-running-mode
- Abfrage des Timers mittels Polling
- Programmierung von Verzögerungszeiten mittels einem nicht blockierendem Konzept innerhalb der Main-Super-Loop

Parallele Ausführung verschiedener LED-Blinkzyklen

```
1.  while(1)
2.      {
3.       _LATF0 = 1;
4.          if (_T1IF) {
5.              _T1IF = 0; //Flag clearen
6.              Count++;
7.              if (Count >= HEARTBEAT_MS){
8.                  _LATF0 = 0;
9.                  Count = 0;
10.
11.                 Blink0();
12.                 Blink1();
13.                 Blink2();
14.                 Blink3();
15.
16.                 Taster0();
17.                 Taster1();
18.                 Taster2();
19.                 Taster3();
20.
21.             }
22.         }
23.
24.     }
```

Im obigen Bild stellt die äußere while()-Schleife die sog. Superloop dar. Dieser Loop wird zyklisch, abhängig von der Systemfrequenz durchlaufen. Nach einer Millisekunde setzt der Hardware-timer das Interrupt-Flag (T1IF), die erste if()-Schleife wird begangen, die Flag wieder resettet und im Anschluss der Counter (Count) um +1 inkrementiert. Sobald dieser Vorgang 10 mal geschehen ist, läuft das Programm in die zweite if()-Schleife und führt die eigentlichen Aktionen aus. Diese Funktionen sind klein und schnell zu durchlaufen, damit die Bedingung für einen nicht-blockierenden Ablauf erfüllt ist.

```
1.  uint16_t T_blink0 = 600;
2.  uint16_t T_blink1 = 1000;
3.  uint16_t T_blink2 = 1400;
4.  uint16_t T_blink3 = 2200;
5.
6.  int c0,c1,c2,c3 = 0;
7.  void Blink0(){ //T=600ms
8.      c0++;
9.      if (c0>=(T_blink0/(2*HEARTBEAT_MS))){
10.         TLED0();
11.         c0=0;
12.     }
13. }
```

Die Blink0-Funktion wird aus der Superloop aufgerufen, nachdem jeweils 10ms vergangen sind. Die vier auf dem Board verbauten LEDs werden mit unterschiedlichen Blinkfrequenzen angesteuert. Im obigen Beispiel wurde eine Periodendauer von 600ms gewählt, was bedeutet dass die LED zyklisch nach 300ms ihren Zustand wechselt. Der Zähler c0 zählt in 10ms-Schritten nach oben und wird deshalb in Zeile 9 mit der Halben Periodendauer verglichen und dann durch die aktuelle Heartbeatzeit von 10ms geteilt. Die Funktion TLED0() ist in der User.h definiert, und entspricht dem Assembler Befehl für das Led-toggle (weniger Rechenleistung).

Ungefähre Auslastung des Prozessors bestimmen [Nur BlinkX()]

In der Superloop wird der Pin auf dem Expansion Port 2 (Pin 25) auf 1 gesetzt. Solange die Funktionen BlinkX() nicht aufgerufen werden bleibt der Pin auf 1. Sobald der Ablauf aber diese Funktionen erreicht hat wird der Pin auf 0 gesetzt und erst wieder nach einem vollständigen Durchlauf zurückgesetzt. Somit kann eine Leerlaufzeit, eine Rechenzeit und eine Gesamtzeit bestimmt werden. Die ungefähre Prozessorauslastung in Prozent ergibt sich dann zu:

$$Prozessorauslastung\ in\ \% = \frac{Rechenzeit}{Gesamtzeit} * 100$$

Für eine Systemfrequenz von 100MHz ergab sich eine ungefähre Auslastung von 0,0225%.

Veränderung der Blinkgeschwindigkeit mit Hilfe der Taster

```
1.  void Taster0(){
2.      static uint16_t pressedtime = 0;
3.      static bool last_state = 0;
4.      if (T0) {
5.          last_state = 1;
6.          pressedtime = pressedtime + 10;
7.          }
8.      if(T0 != last_state){
9.              last_state = 0;
10.             if (pressedtime > 30) {
11.                 T_blink0 = 2*pressedtime;
12.                 }
13.             pressedtime = 0;
14.     }
15. }
```

Die Funktion Taster0() wird alle 10ms zyklisch aufgerufen. Solange der Taster0 nicht gedrückt wird, werden beide If-Abfragen nicht ausgeführt. Solange der Taster gedrückt ist, wird die Variable pressedtime alle 10ms um +10 erhöht. Dabei wird in der Variable last_state der letzte Status des Tasters abgespeichert. Sobald der Taster nicht mehr gedrückt ist, wird die 2. If-Abfrage durchlaufen und die Periodendauer der LED auf den gewünschten Wert gesetzt. In Zeile 10 wurde noch eine Überprüfung auf Prellen des Tasters eingefügt, welche Druckzeiten von unter 30ms ignoriert.

Ungefähre Auslastung des Prozessors bestimmen

Die Prozessorauslastung wurde wie oben beschrieben bestimmt. Jetzt haben die Funktionen TasterX() jedoch auch noch ein Einfluss auf die Auslastung. Für unterschiedliche Systemfrequenzen wurden folgende Werte ermittelt:

Systemfrequenz in MHz	Auslastung in %
50	0,106
100	0,0529
140	0,0375
7,37	0,724