Gruppe: Leander Stark, Jonas Dollinger Datum: 23.11.2020

Aufgabe 4

- Verwendung eines Timers im Gated Timer Mode
- Parametrierung und Betrieb des Timers im 32 Bit Modus
    - 32 Bit Lese- und Schreibzugriff auf des Timer-Zähl-Register
- Remappable Pin dem Gate Eingang des Timers zuweisen

Initialisierung des 32-Bit Timers

```
1.  void init_t2_t3(){
2.      T3CONbits.TON = 0; // Stop any 16-bit Timer3 operation
3.      T2CONbits.TON = 0; // Stop any 16/32-bit Timer3 operation
4.      T2CONbits.T32 = 1; // Enable 32-bit Timer mode
5.      T2CONbits.TCS = 0; // Select internal instruction cycle clock
6.      T2CONbits.TGATE = 1; // Enable Gated Timer mode
7.      T2CONbits.TCKPS = 0b00; // Select 1:1 Prescaler
8.      TMR3 = 0x00; // Clear 32-bit Timer (msw)
9.      TMR2 = 0x00; // Clear 32-bit Timer (lsw)
10.     PR3 = 0xFFFF; // Load 32-bit period value (msw)
11.     PR2 = 0xFFFF; // Load 32-bit period value (lsw)
12.     IPC2bits.T3IP = 0x01; // Set Timer3 Interrupt Priority Level
13.     IFS0bits.T3IF = 0; // Clear Timer3 Interrupt Flag
14.     IEC0bits.T3IE = 0; // Enable Timer3 interrupt
15.     T2CONbits.TON = 1; // Start 32-bit Timer
16. }
```

In dieser Funktion wird der 32-Bit Timer initialisiert. Dafür werden zwei 16-Bit Timer (Timer2 und Timer3) zu einem 32-Bit Timer zusammengesetzt, welcher dann über das Timer2 Register angesprochen wird. Da wir die Periodenwerte für diese Aufgabenstellung nicht benötigen, werden diese auf die maximalen Werte gesetzt.

Super Loop

```
1.  _T2CKR = 96;
2.  while(1)
3.  {
4.   _LATF0 = 0;
5.
6.      if (_T1IF) {
7.          _T1IF = 0; //Flag clearen
8.          Count++;
9.          if (Count >= HEARTBEAT_MS){
10.             _LATF0 = 1;
11.             Count = 0;
12.
13.             Blink0();
14.             Blink1();
15.             Blink2();
16.             Blink3();
17.
18.             Taster0();
19.             Taster1();
20.             Taster2();
21.             Taster3();
22.             measureProcesstime();
23.
24.         }
25.     }
26. }
```

Um die Zeiten für die Auslastung zu messen wurde in der vorherigen Aufgabe der Pin RP96 (LATF0) verwendet. In Zeile 1 wird der Eingang für den Gated Timer nun auf diesen Pin gemapped. Solange der Prozessor die Aufgaben wie BlinkX() und TasterX() bearbeitet zählt der Timer hoch. Somit kann die Rechenzeit bestimmt werden. Die Funktion measureProcesstime() wird alle 10ms aufgerufen und bestimmt nach 10s die mittlere Auslastung des Prozessors.

measureProcesstime()

```
1.  void measureProcesstime(){
2.      static uint16_t time = 0;
3.      time++;
4.      if (time >= MESSZEIT_MS){
5.          float Auslastung;
6.          uint16_t lsw = TMR2;
7.          uint16_t msw = TMR3HLD;
8.
9.          float Auslastung1 = (float)lsw * 10;
10.         float Auslastung2 = (float)msw *65535*10;
11.         Auslastung = (Auslastung1 + Auslastung2) / (FCY);
12.         time = 0;
13.         TMR3HLD=0;
14.         TMR3 = 0; // Clear 32-bit Timer (msw)
15.         TMR2 = 0;
16.     }
17. }
18.
19. #define HEARTBEAT_MS 10
20. #define MESSZEIT_MS 1000 // entspricht einer Messzeit von 10s, gilt nur für HEARTBEA
    T_MS=10
```

Diese Funktion wird von der Main()-Funktion nach 10ms aufgerufen und zählt in den ersten 1000 Aufrufen ($n_{Aufrufe} = \frac{10000ms}{10ms} = 1000$) eine Variable „time“ hoch. Sobald nach 10s die innere if()-Schleife betreten wird, werden zuerst in Zeile 6 und 7 die Timer-Werte der beiden Timer in die Variablen lsw und msw geschrieben. Da der Timer2, nachdem er bei 65535 angekommen ist überläuft, zählt dann Timer3 um +1 hoch. Damit entspricht eine 1 im Timer3 65535 cycles. Die Auslastung wird für beide Timer separat berechnet und in Prozent angegeben:

$$Auslastung1\ [\%] = \frac{lsw * 100}{10s}$$

$$Auslastung2\ [\%] = \frac{msw * 65535 * 100}{10s}$$

Um dann die Gesamtlast zu erhalten werden die beiden Prozentwerte summiert, und durch die aktuelle halbe Systemfrequenz dividiert. In den letzten Zeilen wird abschließend dafür gesorgt, dass die Timer wieder bei „0“ anfangen, und sich beim nächsten Durchlauf nicht summieren.

Gruppe: Leander Stark, Jonas Dollinger Datum: 23.11.2020

Messergebnisse

Für verschiedene Systemfrequenzen wurden folgende Werte im Debug-Modus bestimmt:

Systemfrequenz in MHz	Auslastung in %
50	0,114
100	0,0572
140	0,041
7,37	0,776