Gruppe: Leander Stark, Jonas Dollinger Datum: 15.12.2020

Aufgabe 6

- Versetzen Sie den Mikrocontroller nach 10 Sekunden in der Hauptprogrammschleife in die Stromsparmodi Sleep bzw. Idle Modus um Strom zu sparen.
- Das Aufwecken soll über einen beliebigen Taster (Hinweis es sind 5 Taster auf dem Board) erfolgen. (Interrupt!)
    - Danach soll die Hauptschleife wieder 10 Sekunden bis zum nächsten Ruhezustand abgearbeitet werden.
- Achtung, auch der Betriebszähler Interrupt weckt den Controller auf.
    - Nach der Aktualisierung der Betriebszeit sollte der Controller dann wieder in den Ruhemodus versetzt werden
- Messen Sie die Stromaufnahme Im Betrieb, bzw. in den verschiedenen Stromsparmodi bei unterschiedlichen Systemfrequenzen und vergleichen Sie diese auf geeignete Art und Weise.
    - **Achtung:** Diese Teilaufgabe diesmal nur, falls Sie entsprechende Messgeräte / Leitungen Zuhause haben!
    - Beachten Sie die Stromaufnahme der anderen Komponenten auf dem Board
    - Nehmen Sie eine zusätzliche Systemfrequenz von 70 MHz zu den bis jetzt verwendeten (7.37, 8, 100 und 140 MHz) für Ihre Analysen.
    - Wie variiert die Stromaufnahme im Betrieb mit der Systemfrequenz?
    - Wie variiert die Stromaufnahme bei den Stromsparmodi in Bezug zu dem aktivierten Stromsparmodi (µC Spannungsregler im Sleep nicht vergessen)
    - Die Mikrocontroller internen Spannungsregler können im Sleep Modus aktiviert / deaktiviert werden.
- Messen Sie wie lange es dauert bis der Controller nach dem Aufweckereignis über einen Tastendruck weiter arbeitet. (Wake Up Time)
    - z.B. durch ein Signal an einem Pin, beim Eintritt in den Stromspar Modi auf 0 beim Aufwecken wieder auf 1
    - Idle / Sleep Mode + Systemfrequenz 7.37, 8, 70, 100 und 140 MHz
- Welche Erkenntnisse können sie aus den obigen Messungen in Bezug Stromaufnahme und Wake Up Time zu den verschiedenen Modi und Systemfrequenzen ziehen?
- Validieren Sie anhand der LED Anzeige nach dem "Aufwecken" ob die Zeit ordnungsgemäß weitergelaufen ist.
    - Die LED's verändern während dem Stromsparmodi Ihren Zustand nicht!
    - Was könnte unternommen werden um:
        - Während der Stromsparmodi noch mehr Strom zu sparen?
        - Die Zeit in der der Controller für den Betriebszähler im Stromsparmodus aktiv ist, so kurz wie nur möglich zu halten?
        - Sie können gerne Ihre Ideen im Programm umsetzen und präsentieren!

Stromsparmodus nach 10s

```
1.  //Interrupt Service Routine
2.  void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void)
3.      {
4.          _T1IF = 0; //Clear Timer1 interrupt flag
5.          u32_uptime_seconds++;
6.          u32_inactive_seconds++;
7.          if (u32_inactive_seconds >= 10){
8.              LED0 = 0;
9.              //Sleep();
10.             Idle();
11.         }
12.     }
```

Im Programmablauf der Timer1-Interrupt Routine, befindet sich in Zeile 7 die if-Abfrage, welche nach 10s Betriebszeit den µC in den Sleep- oder Idle-Modus versetzt.

Auch während des Sleep/Idle-Modus wird die Betriebszeit weiter aktualisiert, indem der Mikrocontroller durch den Timer Interrupt kurz aufwacht, die Variable inkrementiert und dann wieder zurück in den Sleep/Idle Modus fällt.

Taster-Interrupts

```
1.  _CNIE = 1; //Change Notification Interrupt Enable
2.  _CNIP = 1; // Interupt priority heruntersetzten
3.  _CNIEG14 = 1; //CN für Taster aktivieren
4.  _CNIEG15 = 1;
5.  _CNIEG12 = 1;
6.  _CNIEG13 = 1;
7.  _CNIEG9 = 1;
```

Der Notification Change Interrupt muss zum einen durch den Befehl in Zeile 1 aktiviert werden, zum anderen wurde die Priorität auf das gleiche Level des Timer Interrupts heruntergesetzt.

Die Befehle in den Zeilen 3 bis 7 sorgen dafür, das ein Betätigen der Taster ein NC-Interrupt auslöst.

```
1.  //Change Notification Interrupt Routine
2.   void __attribute__((interrupt,auto_psv)) _CNInterrupt(void)
3.   {
4.       //LED0 = 1;
5.       asm("BSET 0xE15, #0");
6.       _CNIF = 0;
7.
8.       u32_inactive_seconds = 0;
9.   }
```

In der Interrupt Service Routine wird für eine spätere Messung eine LED aktiviert, und das Interrupt-Flag sowie der Counter zurückgesetzt. Somit kann der Mikroprozessor wieder für 10s den normalen Betrieb aufnehmen.

Sleep-Wake Zeiten

Frequenz [MHz]	Zeit [ms]
7,37	0,1780
8	0,1598
70	0,1750
100	0,1827
140	0,1636

Mittels eines Logic-Analyzers wurde der zeitliche Unterschied zwischen Tastendruck und dem Aufleuchten der LED0 gemessen, welche zu Leuchten beginnt sobald der Mikrocontroller das Change-Notification Interrupt ausführt.

Abbildung 1: Beispielmessung mit f=140Mhz aus dem Sleep-Modus

In obiger Tabelle sieht man die verschiedenen Zeiten, die der Mikrocontroller braucht, um bei verschiedenen Frequenzen aus dem Sleep-Zustand in den hochgefahrenen Zustand zu wechseln. Es lässt sich in den Messungen kein signifikanter Einfluss der Systemfrequenz auf die Zeiten feststellen. Dennoch brauchte der µC bei einer Systemfrequenz von 140 MHz am wenigsten Zeit (0,1636ms) um den Zustand zu wechseln.

Idle-Wake Zeiten

Frequenz [MHz]	**Zeit [µs]**
7,37	9,83
8	9,25
70	1,00
100	0,79
140	0,54

Der Messaufbau ist bei dieser Messreihe gleichgeblieben, doch nun wurde anstatt den Mikrocontroller in den Sleep-Modus zu versetzen, der Idle-Modus gewählt.

Hier sieht man nun auch eine signifikante Abnahme der Zeit, proportional mit der gewählten Systemfrequenz.

Unsere Annahme schließt darauf, dass beim Sleep-Wake Zyklus in jedem Fall zuerst der Oszillator hochgefahren und in den stabilen Zustand kommen muss, was immer eine (verglichen zum Idle-Wake Zyklus) längere Zeit benötigt. Dafür hat man im Sleep-Modus den Vorteil, dass der Energieverbrauch gegenüber des Idle-Modus geringer ist, da die Peripherie nicht mit Strom versorgt werden muss.

Validierung der LED-Anzeige

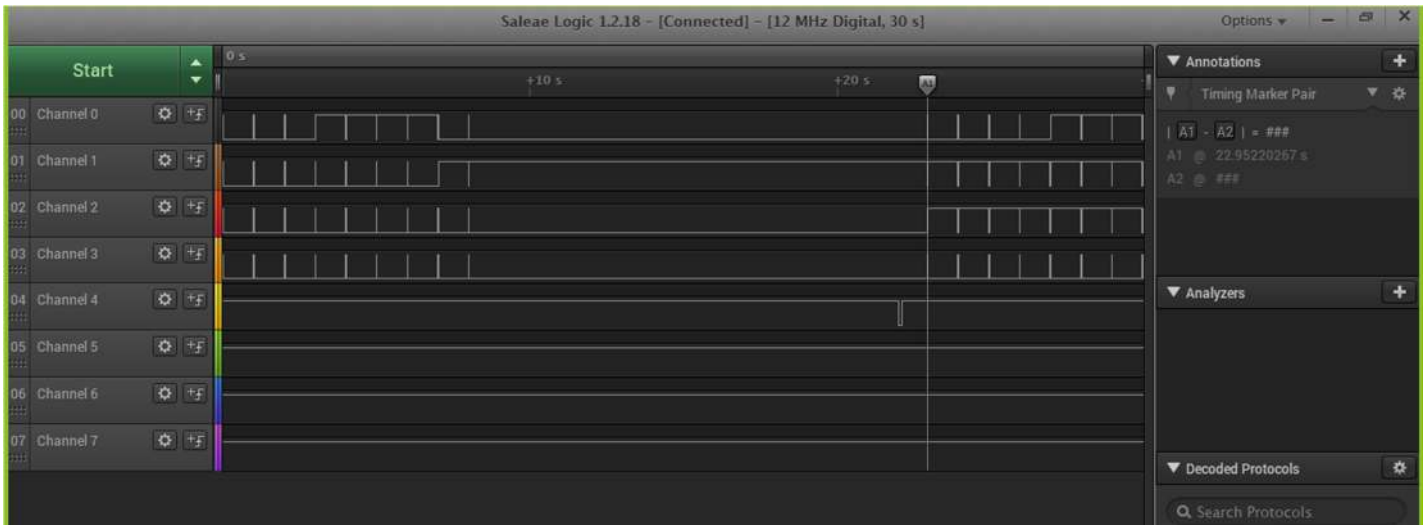

Abbildung 2: Messung der LED-Zustände nach dem Aufwecken

Kurz nach dem Programmieren des Microcontrollers wurde die Messung gestartet. Nach ca. 10s verändern die LEDs (Channel 0 bis 3) ihren Zustand nicht mehr. Nach ungefähr 10 weiteren Sekunden wurde der Taster (Channel 4) gedrückt, um den Prozessor aufzuwecken. Die LEDs zeigten dann eine Bit-Kombination von 0110 an.

Mit dem Wissen, das dies den Bits 2 bis 5 der Betriebszeit entspricht, ergibt sich ein minimaler Wert von 0b0011000 = 24s. Dies entspricht auch ungefähr der Erwartung. Die Betriebszeit wurde also ordnungsgemäß auch während dem Sleep / Idle Modus aktualisiert.

Weitere Energiesparmöglichkeiten

Man könnte im Sleep-Modus die Zeit, die der Betriebszeitzähler zum Inkrementieren benötigt möglichst kurzhalten, indem Assembler-Befehle verwendet werden.

Im Idle-Modus läuft der Oszillator die komplette Zeit durch, was bedeutet dass man an dieser Stelle optimieren kann indem die Systemfrequenz auf den minimalen Wert (Hier: 7,37 MHz) reduziert wird.