LABOR 3 | *Informationssysteme*
Interrupts

Ziel des Labors (4 Perioden)

1. In diesem Labor sollen Sie Ihre theoretischen Kenntnisse über Interrupts praktisch anwenden. Sie werden zunächst ein einfaches Interrupt realisieren und anschliessend mit den Auswirkungen von Interrupts auf den normalen Ablauf eines Programms experimentieren.
2. Dieses Labor wird in Zweiergruppen durchgeführt.
3. Die Dauer des Labors beträgt 4 Perioden.
4. Alle Dateien in Zusammenhang mit diesem Labor finden Sie auf der Website der Vorlesung (http://sin.begincoding.net).

Teil 1 – Grundlegende Interrupts

Wir haben in der Vorlesung gesehen, wie Interrupts funktionieren. Wir werden diese Kenntnisse nun anhand eines einfachen Beispiels anwenden. Wir werden ein Programm erstellen, das möglichst wenig Energie verwendet und das beim Drücken des Buttons OK den Zustand aller LED ändert. Wir werden dazu Interrupts verwenden, deren Funktionsschema nachstehend abgebildet ist. Das Schema zeigt, wie einige externe Quellen und die Peripheriegeräte den Prozessor unterbrechen können.

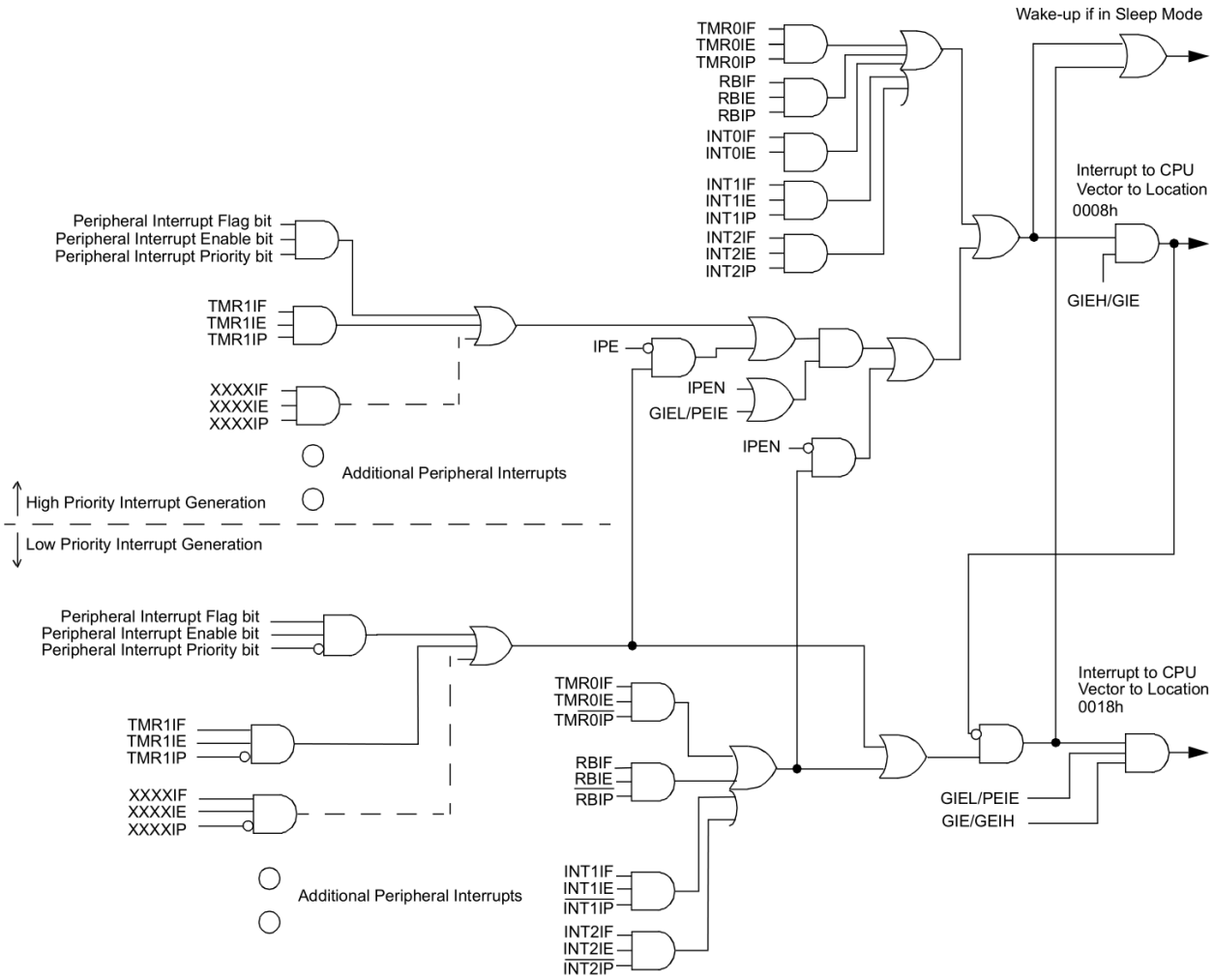

Abb. 1 – Funktionsschema der Interrupts des PIC18F87K22 (Quelle: Microchip Datasheet)

Aufgabe 1

1. Konfigurieren Sie die Interrupts, damit Sie auf den zentralen Tastern des PICEBS2 Boards reagieren (active high).
2. Schreiben Sie den Code der Interrupt-Routine, mit der der Zustand aller an PORTH angeschlossenen LED geändert werden kann.
3. Testen Sie, ob Ihr Programm wie gewünscht funktioniert.
4. Messen Sie den mittleren Stromverbrauch des Boards. Diese Messung soll zwischen den weissen und gelben Testpunkten der PICEBS2-Karte durgeführt werden. Dabei ist zu berücksichtigen, dass ein 1 Ohm Widerstand zwischen den 2 Testpunkten plaziert ist.
5. Sorgen Sie dafür, dass der Prozessor in den Sleep-Modus übergeht, wenn er nicht die Buttons bearbeitet.
6. Messen Sie den Stromverbrauch erneut. Was stellen Sie fest? Haben Sie dies erwartet?

   .....

   .....

7. Berühren Sie mit Ihrem Finger die Pins des Mikrokontrollers und messen Sie den Strom. Was stellen Sie fest und warum?

   .....

   .....

Teil 2 – Interrupt-Latenzzeit

Theoretische Erklärung

Wenn der Prozessor einen Interrupt akzeptiert, muss dieser den laufenden Befehl beenden und den ersten Befehl der Interrupt-Routine laden. Die dafür notwendige Zeit wird Latenzzeit genannt (*interrupt latency time*). Sie wird in Anzahl CPU-Zyklen gemessen. Wenn diese Zeit abgelaufen ist, führt der Prozessor den Befehl bei der Adresse 0x08 (*interrupt vector*) aus, falls der Prioritätsmechanismus nicht benutzt wird oder es sich um einen Interrupt mit hoher Priorität handelt.

Im zweiten Teil des Labors werden die Auswirkungen und Konsequenzen der Interrupt-Latenzzeiten veranschaulicht.

Aufgabe 2

Auf dem Pin RH0 des Prozessors soll ein Rechtecksignal erzeugt werden. Dieses Signal soll die höchstmögliche Frequenz aufweisen. Gleichzeitig soll die Anzahl Impulse auf dem Button BT0/RB0 (links) des Boards gezählt werden. Es darf kein Impuls verpasst werden. Dieser Wert (8 Bit) wird in den Speicher verschoben und kann mithilfe des Debuggers angezeigt werden.

1. Schreiben Sie den Code, um das schnellstmöglichste Rechtecksignal auf dem Pin R zu erzeugen.
2. Visualisieren Sie Ihr Signal mittels des Oszilloskops und vergewissern Sie sich, dass es sich wirklich um ein Rechtecksignal handelt. Welche Frequenz erhalten Sie?

   .....

3. Schreiben Sie den für die Verwaltung der Interrupts auf dem Button notwendigen Code. Sie können die Ergebnisse der Aufgabe 1 wiederverwenden. Vergessen Sie nicht, die Register zu speichern.

4. Was stellen Sie beim Signal auf dem Oszilloskop fest, wenn Sie auf den Button drücken? Wie können Sie dies erklären?

.....

.....

.....

.....

Aufgabe 4 – First steps in C programming

For this last task we will program interrupts using the C programming language (even though we might not have seen the theory about it). The task to be done is to measure the speed of an interrupt on RB0, which corresponds to the speed of a fan that produces one pulse per revolution. The connection schematic will be provided in class.

For this task, you have to create a new C project as follows :

1. Create a new projet in MPLAB as usual but instead of assembler (**`mpasm`**), choose the **`xc8`** compiler.
2. Download the configuration source files (**`picebs.h`** and **`picebs.c`**) for the PICEBS2 board and copy them to the projet folder (there are located here http://sin.begincoding.net/documentation/C/)
3. Right-click on the projet and choose *Add exisiting item* and select the **`picebs2.h`** file
4. Add a new main source file by selecting *File -> New file*
5. Add the following base code (which does nothing).

```
#include "picebs2.h"

void interrupt myInterruptRoutine(void)
{

}

void main()
{

    while (1)
    {

    }
}
```

As in Java, the main function is the entry point of your program. Here, the function **`myInterruptRoutine`** will be located at address 0x8 in the code memory (the interrupt vector). After having checked that everything compiles, complete this code to count the number of interrupts and display the result on the bar graph. The speed should be displayed in arbitrary time units.