Microcontroller Teil 4 Programmiertechnik

DI (FH) Andreas Pötscher

HTL Litec

Verknüpfungssteuerungen

Verknüpfungssteuerungen sind die einfachsten Steuerungen. Dabei wird nur Aufgrund von Eingangszuständen ein Ausgangszustand eingestellt. Die Reihenfolge oder die zeitliche Abfolge von Eingängen hat keinen Einfluss auf die Ausgänge.

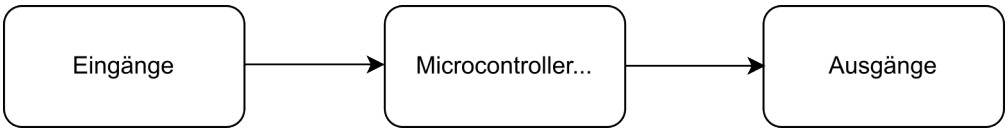

Figure 1: Verknüpfungssteuerungen

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 3 Ausgänge setzen. Aufgrund der Eingänge werden alle Ausgänge gesetzt. Dabei müssen in jeder Verzweigung alle Ausgänge gesetzt werden.

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 3 Ausgänge setzen. Aufgrund der Eingänge werden alle Ausgänge gesetzt. Dabei müssen in jeder Verzweigung alle Ausgänge gesetzt werden.
- GOTO 2

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 3 Ausgänge setzen. Aufgrund der Eingänge werden alle Ausgänge gesetzt. Dabei müssen in jeder Verzweigung alle Ausgänge gesetzt werden.
- GOTO 2
- **Beispiel**

Präprozessor Makros

Zur Vereinfachung und Vermeidung von Redundanzen kann das Ansteuern der Ausgänge mit `#define` Präprozessormakros programmiert werden. Der Präprozessor kopiert den Code vor dem eigentlichen Kompilieren an die richtigen Stellen.

```
#define PINLED1 1
PORTA |= 0x01 << PINLED1;
```

Beim übersetzten des Programmes wird dabei als erstes der Präprozessor ausgeführt. Der Code sieht dann so aus.

```
PORTA |= 0x01 << 1;
```

Beispiel

Wahrheitstabellen

Die Logik für eine Verknüpfungssteuerung lässt sich sehr gut mit einer Wahrheitstabelle darstellen. Dabei wird für jeden Ein- und Ausgang eine Spalte und für jede Eingangskombination eine Zeile angelegt. Bei drei Sensoren und zwei Leuchten sieht das dann folgendermaßen aus:

S1	S2	S3	L1	L2
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Wahrheitstabellen

LINZER TECHNIKUM
HTL Paul-Hahn-Straße

Beispiel:

- Leuchte1 leuchten soll wenn 1 oder 2 von den 3 Sensoren logisch 1 melden. Ansonsten nicht:
- Leuchte2 leuchten soll wenn alle 3 Sensoren logisch 1 melden.

S1	S2	S3	L1	L2
0	0	0	**0**	**0**
0	0	1	**1**	**0**
0	1	0	**1**	**0**
0	1	1	**1**	**0**
1	0	0	**1**	**0**
1	0	1	**1**	**0**
1	1	0	**1**	**0**
1	1	1	**0**	**1**

Ablaufsteuerungen

Bei Auflaufsteuerungen werden nicht nur Eingangszustände sondern auch interne Zustände berücksichtigt. Damit sind Abfolgesteuerungen möglich.

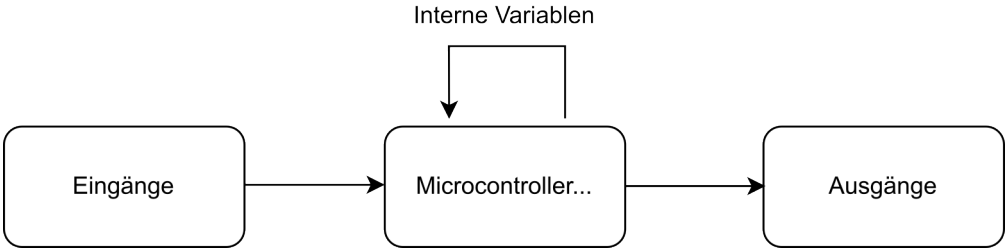

Figure 2: Ablaufsteuerungen

State Machine

Ein solches Verhalten wird in der Informatik als Statemachine oder Zustandsautomat dargestellt. Im einfachsten Fall gibt es dabei 2 States oder Zustände. Es können aber natürlich auch mehr sein.

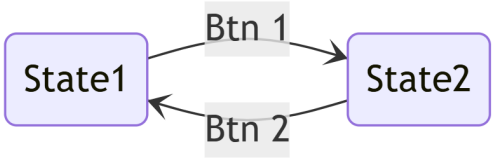

Figure 3: Statemachine

Der State wird im Programm in einer eigenen Variablen gespeichert.

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 **Initialsieren der Variable in der der Zustand (State) gespeichert wird.**

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 **Initialsieren der Variable in der der Zustand (State) gespeichert wird.**
- 3 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 **Initialsieren der Variable in der der Zustand (State) gespeichert wird.**
- 3 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 4 **Setzen der Zustandsvariablen.**

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 **Initialsieren der Variable in der der Zustand (State) gespeichert wird.**
- 3 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 4 **Setzen der Zustandsvariablen.**
- 5 Ausgänge setzen. Aufgrund der Eingänge **und der Zustandsvariablen** werden alle Ausgänge gesetzt. Dabei müssen in jeder Verzweigung alle Ausgänge gesetzt werden.

Das Programm kann dabei immer nach folgendem Schema einfach aufgebaut werden.

- 1 Initialisierung. Hier werden die Ein- und Ausgänge definiert.
- 2 **Initialsieren der Variable in der der Zustand (State) gespeichert wird.**
- 3 Eingänge lesen. Am Anfang des Main Loop werden alle Eingänge gelesen und in Variablen gespeichert
- 4 **Setzen der Zustandsvariablen.**
- 5 Ausgänge setzen. Aufgrund der Eingänge **und der Zustandsvariablen** werden alle Ausgänge gesetzt. Dabei müssen in jeder Verzweigung alle Ausgänge gesetzt werden.
- GOTO 3

Damit kann zum **Beispiel** eine Steuerung mit einem Ein- und einem Austaster realisiert werden. Ein Led lässt sich dann nur einschalten wenn sich die Steuerung im Ein-Modus befindet.

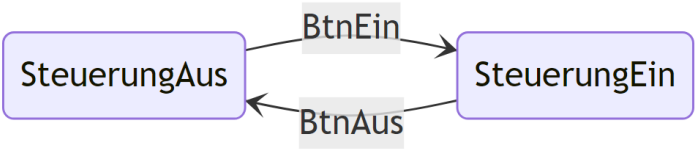

Figure 4: Statemachine des Beispiels

Dabei kommt das Ergebnis nicht nur auf die Zustände der Eingänge an, sondern auch auf die Reihenfolge in der diese Betätigt werden. Werden z. B. folgende Eingänge in der gegeben Reihefolge gesetzt:

```
1. PINB2 (tasterLed) = true
```

Ist die LED an PA1 **nicht** eingeschaltet.

Ablaufsteuerung

Werden aber die Eingänge in folgender Reihenfolge betätigt.

1. `PINB0 (tasterEin) = true`
2. `PINB0 (tasterEin) = false`
3. `PINB2 (tasterLed) = true`

ist die LED an PA1 eingeschaltet. Dadurch, dass der Eingang PINB0 auf true gesetzt wird, wird interne Variable `controlOn` auf true gesetzt. Das Ergebnis wenn PINB2 auf true gesetzt wird verändert sich dadurch.

Ablaufsteuerung

Der zeitliche Ablauf kann in einem Signal-Zeit Diagramm dargestellt werden.

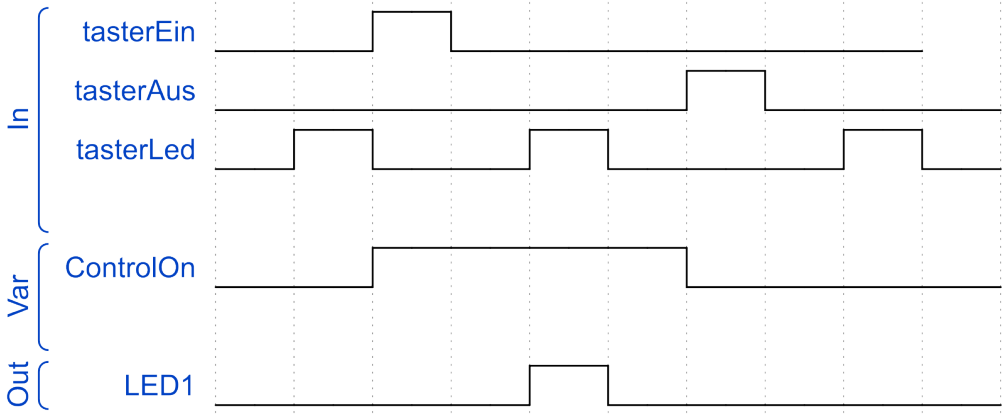

Figure 5: Signal Zeitverlauf