

# Microcontroller Teil 2 Programmieren der Pins

DI (FH) Andreas Pötscher

HTL Litec

Special-Function-Register (SFRs) sind im Microcontroller das Bindeglied zwischen CPU und Peripherie. Mit ihnen lassen sich nicht nur GPIO-Pins programmieren, sondern auch z.B: der Analog-Digitalwandler des ATmega2560, die seriellen Schnittstellen, usw. Damit SFRs im C-Programm verwendet werden können, muss folgender Header inkludiert werden:

```
#include <avr/io.h>
```

Für jeden GPIO-Port existieren drei Special-Function-Register. Für Port L beispielsweise heißen diese drei Register.

DDRL

PORTL

PINL

- ▶ Jedes dieser Register beinhaltet acht voneinander unabhängige Bits.

- ▶ Jedes dieser Register beinhaltet acht voneinander unabhängige Bits.
- ▶ Jedes Bit ist für einen Pin dieses Ports zuständig.

- ▶ Jedes dieser Register beinhaltet acht voneinander unabhängige Bits.
- ▶ Jedes Bit ist für einen Pin dieses Ports zuständig.
- ▶ Das dritte Bit von DDRL ist also für die Einstellung von PL3 zuständig.

- ▶ Jedes dieser Register beinhaltet acht voneinander unabhängige Bits.
- ▶ Jedes Bit ist für einen Pin dieses Ports zuständig.
- ▶ Das dritte Bit von DDRL ist also für die Einstellung von PL3 zuständig.
- ▶ Im weiteren Verlauf werden die Register DDRn, PORTn und PINn bezeichnet. Wobei n für den Port steht.

- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.



- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.
- ▶ Diese haben die Basis 16. Also einen Zahlenraum von 0 bis F(15).

- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.
- ▶ Diese haben die Basis 16. Also einen Zahlenraum von 0 bis F(15).
- ▶ Das hat den Vorteil, dass mit einer Stelle genau 4 Bit beschrieben werden können.

- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.
- ▶ Diese haben die Basis 16. Also einen Zahlenraum von 0 bis F(15).
- ▶ Das hat den Vorteil, dass mit einer Stelle genau 4 Bit beschrieben werden können.
- ▶ Ein 8Bit Register kann also mit einer zweistelligen hex Zahl beschrieben werden.

- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.
- ▶ Diese haben die Basis 16. Also einen Zahlenraum von 0 bis F(15).
- ▶ Das hat den Vorteil, dass mit einer Stelle genau 4 Bit beschrieben werden können.
- ▶ Ein 8Bit Register kann also mit einer zweistelligen hex Zahl beschrieben werden.
- ▶ Wobei die erste Stelle die höherwertigen 4 Bits darstellt und die zweite Stelle die 4 niederwertigsten Bits.

- ▶ Zu programmieren der SFRs wird meist eine Hexadezimalzahl verwendet.
- ▶ Diese haben die Basis 16. Also einen Zahlenraum von 0 bis F(15).
- ▶ Das hat den Vorteil, dass mit einer Stelle genau 4 Bit beschrieben werden können.
- ▶ Ein 8Bit Register kann also mit einer zweistelligen hex Zahl beschrieben werden.
- ▶ Wobei die erste Stelle die höherwertigen 4 Bits darstellt und die zweite Stelle die 4 niederwertigsten Bits.
- ▶ Im Code wird um Hexadezimal Zahlen darzustellen die Vorsilbe 0x verwendet.

Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

► //Stelle            7654 3210

Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle            7654 3210
- ▶ //Binär            0010 1001

Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle            7654 3210
- ▶ //Binär            0010 1001
- ▶ //Wertigkeit      8421 8421



Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

|                |      |      |
|----------------|------|------|
| ▶ //Stelle     | 7654 | 3210 |
| ▶ //Binär      | 0010 | 1001 |
| ▶ //Wertigkeit | 8421 | 8421 |
| ▶ //Dezimal    | 2    | 9    |

Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

|                |      |      |
|----------------|------|------|
| ► //Stelle     | 7654 | 3210 |
| ► //Binär      | 0010 | 1001 |
| ► //Wertigkeit | 8421 | 8421 |
| ► //Dezimal    | 2    | 9    |
| ► //Hex        | 0x2  | 9    |

Es sollen die Bits 0, 3 und 5 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle           7654 3210
- ▶ //Binär           0010 1001
- ▶ //Wertigkeit      8421 8421
- ▶ //Dezimal         2     9
- ▶ //Hex             0x2     9
- ▶ DDRA = 0x29;

Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

► //Stelle            7654 3210

Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle            7654 3210
- ▶ //Binär            1010 1101

Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle            7654 3210
- ▶ //Binär            1010 1101
- ▶ //Wertigkeit       8421 8421

Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

|                |      |      |
|----------------|------|------|
| ▶ //Stelle     | 7654 | 3210 |
| ▶ //Binär      | 1010 | 1101 |
| ▶ //Wertigkeit | 8421 | 8421 |
| ▶ //Dezimal    | 10   | 13   |

Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

|                |      |      |
|----------------|------|------|
| ► //Stelle     | 7654 | 3210 |
| ► //Binär      | 1010 | 1101 |
| ► //Wertigkeit | 8421 | 8421 |
| ► //Dezimal    | 10   | 13   |
| ► //Hex        | 0xA  | D    |



Es sollen die Bits 0, 2, 3, 5 und 7 (auf den Wert 1) gesetzt werden.

- ▶ //Stelle           7654 3210
- ▶ //Binär           1010 1101
- ▶ //Wertigkeit      8421 8421
- ▶ //Dezimal         10    13
- ▶ //Hex             0xA    D
- ▶ DDRA = 0xAD;

- ▶ Die acht Bits des DDRn-Registers bestimmen, ob die 8 GPIO-Pins des Ports „n“ Ein- oder Ausgänge sind.

- ▶ Die acht Bits des DDRn-Registers bestimmen, ob die 8 GPIO-Pins des Ports „n“ Ein- oder Ausgänge sind.
- ▶ Ist ein Bit auf 1, so ist der zugehörige GPIO-Pin ein Ausgang.

- ▶ Die acht Bits des DDRn-Registers bestimmen, ob die 8 GPIO-Pins des Ports „n“ Ein- oder Ausgänge sind.
- ▶ Ist ein Bit auf 1, so ist der zugehörige GPIO-Pin ein Ausgang.
- ▶ Ist ein Bit auf 0, so ist der zugehörige GPIO-Pin ein Eingang.

- ▶ Die acht Bits des DDRn-Registers bestimmen, ob die 8 GPIO-Pins des Ports „n“ Ein- oder Ausgänge sind.
- ▶ Ist ein Bit auf 1, so ist der zugehörige GPIO-Pin ein Ausgang.
- ▶ Ist ein Bit auf 0, so ist der zugehörige GPIO-Pin ein Eingang.
- ▶ Das DDRn Register hat den Initialwert 0x00. Es sind also alle GPIOs Eingänge.

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

► `DDRB = 0xA3;`

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

- ▶ `DDRB = 0xA3;`
- ▶ `//Stelle`                7654 3210

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

- ▶ `DDRB = 0xA3;`
- ▶ `//Stelle`            7654 3210
- ▶ `//Wertigkeit`       8421 8421



Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

- ▶ `DDRB = 0xA3;`
- ▶ `//Stelle`            7654 3210
- ▶ `//Wertigkeit`      8421 8421
- ▶ `//Hex`              0xA     3

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

- ▶ `DDRB = 0xA3;`
- ▶ `//Stelle`            7654 3210
- ▶ `//Wertigkeit`      8421 8421
- ▶ `//Hex`            0xA    3
- ▶ `//Dezimal`        10    3

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

```
▶ DDRB = 0xA3;  
▶ //Stelle      7654 3210  
▶ //Wertigkeit  8421 8421  
▶ //Hex        0xA   3  
▶ //Dezimal    10   3  
▶ //Binär      1010 0011
```

Welche GPIO-Pins von Port B sind nach folgendem Befehl Eingänge, und welche Ausgänge?

- ▶ `DDRB = 0xA3;`
- ▶ `//Stelle`            7654 3210
- ▶ `//Wertigkeit`      8421 8421
- ▶ `//Hex`            0xA    3
- ▶ `//Dezimal`        10    3
- ▶ `//Binär`           1010 0011
- ▶ Da die Bits Nr. 7, 5, 1 und 0 im DDRB-Register auf eins gesetzt wurden, sind die GPIO-Pins PB7, PB5, PB1 und PB0 Ausgänge.

- ▶ Auch bei diesem Register ist jedes Bit für den korrespondierenden GPIO-Pin „zuständig“.

- ▶ Auch bei diesem Register ist jedes Bit für den korrespondierenden GPIO-Pin „zuständig“.
- ▶ Allerdings haben die Bits in einem PORTn-Register eine Doppelfunktion, je nachdem, ob die zugehörigen GPIO-Pins Ein- oder Ausgänge sind

- ▶ Auch bei diesem Register ist jedes Bit für den korrespondierenden GPIO-Pin „zuständig“.
- ▶ Allerdings haben die Bits in einem PORTn-Register eine Doppelfunktion, je nachdem, ob die zugehörigen GPIO-Pins Ein- oder Ausgänge sind
- ▶ Ist ein GPIO-Pin ein Ausgang, so gibt eine 0 im zugehörigen Bit des PORTn-Registers einen Low-Pegel (0 Volt) an diesem Ausgangspin aus. Ein 1-Bit im PORTn-Register gibt einen High-Spannungspegel (5 Volt) am Ausgangspin aus.

- ▶ Auch bei diesem Register ist jedes Bit für den korrespondierenden GPIO-Pin „zuständig“.
- ▶ Allerdings haben die Bits in einem PORTn-Register eine Doppelfunktion, je nachdem, ob die zugehörigen GPIO-Pins Ein- oder Ausgänge sind
- ▶ Ist ein GPIO-Pin ein Ausgang, so gibt eine 0 im zugehörigen Bit des PORTn-Registers einen Low-Pegel (0 Volt) an diesem Ausgangspin aus. Ein 1-Bit im PORTn-Register gibt einen High-Spannungspegel (5 Volt) am Ausgangspin aus.
- ▶ Wenn ein GPIO-Pin ein Eingang ist, so aktiviert bzw. deaktiviert das zugehörige Bit im PORTn-Register den internen pullup-Widerstand des Eingangspins. Mit einem auf 1 gesetzten Bit im PORTn-Register wird der zugehörige interne pullup-Widerstand aktiviert, mit einem auf 0 gesetzten Bit hingegen deaktiviert.



Was bewirken die beiden folgenden Befehle?

► `DDRD = 0xF0;`

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`              `//1111 0000`

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`            `//1111 0000`
- ▶ `PORTD = 0xCC;`        `//1100 1100`

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`            `//1111 0000`
- ▶ `PORTD = 0xCC;`        `//1100 1100`
- ▶ PD7 und PD6 -> Ausgänge mit High Pegel (5V)

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`            `//1111 0000`
- ▶ `PORTD = 0xCC;`        `//1100 1100`
- ▶ PD7 und PD6 -> Ausgänge mit High Pegel (5V)
- ▶ PD5 und PD4 -> Ausgänge mit Low Pegel (0V)

Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`            `//1111 0000`
- ▶ `PORTD = 0xCC;`        `//1100 1100`
- ▶ PD7 und PD6 -> Ausgänge mit High Pegel (5V)
- ▶ PD5 und PD4 -> Ausgänge mit Low Pegel (0V)
- ▶ PD3 und PD2 -> Eingänge mit internem Pull-Up



Was bewirken die beiden folgenden Befehle?

- ▶ `DDRD = 0xF0;`
- ▶ `PORTD = 0xCC;`
- ▶ `//Stelle`                      7654 3210
- ▶ `DDRD = 0xF0;`            `//1111 0000`
- ▶ `PORTD = 0xCC;`        `//1100 1100`
- ▶ PD7 und PD6 -> Ausgänge mit High Pegel (5V)
- ▶ PD5 und PD4 -> Ausgänge mit Low Pegel (0V)
- ▶ PD3 und PD2 -> Eingänge mit internem Pull-Up
- ▶ PD1 und PD0 -> Eingänge ohne internem Pull-Up

**Beispiel** Bringen Sie die 3 LEDs zum leuchten.

- ▶ Bits dieser Register werden von der GPIO-Peripherie auf null oder eins gesetzt, und der Wert dieser Register wird im C-Programm eingelesen.

- ▶ Bits dieser Register werden von der GPIO-Peripherie auf null oder eins gesetzt, und der Wert dieser Register wird im C-Programm eingelesen.
- ▶ Die Peripherie liest den Spannungspegel am Eingang ein und schreibt dementsprechend eine 0 oder 1 in das dementsprechende Bit im PINn Register.

- ▶ Bits dieser Register werden von der GPIO-Peripherie auf null oder eins gesetzt, und der Wert dieser Register wird im C-Programm eingelesen.
- ▶ Die Peripherie liest den Spannungspegel am Eingang ein und schreibt dementsprechend eine 0 oder 1 in das dementsprechende Bit im PINn Register.
- ▶ Damit können z.B. Taster eingelesen werden.

# Das PINn-Register (Beispiel)

Der Pin PC6 wird mit einem Taster und einem externen pullup-Widerstand beschaltet. Im Programm soll der Anweisungsblock einer if-Verzweigung nur dann ausgeführt werden, wenn der Taster gedrückt ist. Wie lautet der entsprechende if-Befehl?



Figure 1: Taster an AVR Mikrocontroller mit pull-up Widerstand

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
- ▶ PINC                    ?1??   ????



Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
- ▶ PINC                    ?1?? ????
- ▶ 0x40                    0100 0000

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
- ▶ PINC                    ?1?? ????
- ▶ 0x40                   0100 0000
- ▶ PINC & 0x40    0100 0000

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
- ▶ PINC                    ?1?? ????
- ▶ 0x40                    0100 0000
- ▶ PINC & 0x40    0100 0000
- ▶ Falls der Taster gedrückt ist, ist Bit Nr. 6 im PINC-Register null:

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
- ▶ PINC                    ?1?? ????
- ▶ 0x40                    0100 0000
- ▶ PINC & 0x40        0100 0000
- ▶ Falls der Taster gedrückt ist, ist Bit Nr. 6 im PINC-Register null:
- ▶ PINC                    ?0?? ????

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
  - ▶ PINC                    ?1?? ????
  - ▶ 0x40                    0100 0000
  - ▶ PINC & 0x40        0100 0000
- ▶ Falls der Taster gedrückt ist, ist Bit Nr. 6 im PINC-Register null:
  - ▶ PINC                    ?0?? ????
  - ▶ 0x40                    0100 0000

Als erstes werden alle Bits des PINC-Registers ausmaskiert, mit Ausnahme des Bits Nr. 6. Dieses Ausmaskieren erfolgt mit einer bitweisen UND-Verknüpfung mit dem Bitmuster 0x40. Dieses Bitmuster enthält lauter Nullen, nur in Bit Nr. 6 ist eine Eins.

- ▶ Falls der Taster nicht gedrückt ist, ist Bit Nr. 6 im PINC-Register eins:
  - ▶ PINC                    ?1?? ????
  - ▶ 0x40                    0100 0000
  - ▶ PINC & 0x40    0100 0000
- ▶ Falls der Taster gedrückt ist, ist Bit Nr. 6 im PINC-Register null:
  - ▶ PINC                    ?0?? ????
  - ▶ 0x40                    0100 0000
  - ▶ PINC & 0x40    0000 0000

**Zusammengefasst:** Bei gedrücktem Taster liefert PINC & 0x40 als Ergebnis 0x00, bei nicht gedrücktem Taster ist das Ergebnis hingegen 0x40. Der Anweisungsblock der folgenden if-Verzweigung wird also nur bei gedrücktem Taster ausgeführt:

```
if((PINC & 0x40) == 0x00)
{
    //Button pressed ...
}
else
{
    //Button not pressed ...
}
```

## Beispiel

- ▶ Wenn der Button gedrückt ist soll die gelbe LED leuchten.
- ▶ Wenn der Button nicht gedrückt ist die grüne LED.