

# Watchdog <-> Wachhund



# Gliederung

- **Einleitung**
- **Watchdog**
  - Was ist Watchdog & wofür wird es eingesetzt?
  - Wie funktioniert Watchdog
  - Welche Arten von Watchdog gibt es?
    - Hardwareseitig
    - Softwareseitig
  - Einstellungen am Atmega
- **Brown-out-detection**
  - Definition & Nutzen
  - Umsetzung am ATmega

# Fallbeispiel

```
uint8_t x;
```

```
x = 10;
```

```
while (x >= 0)
```

```
{
```

```
    // tu was
```

```
    x--;
```

```
}
```

# Watchdogarten

- Wie verbaut?
  - Intern im  $\mu\text{C}$
  - Auf Platine verbauter Mikroelektronik-Baustein
  - Externe Mikroelektronikkomponente
- Time-out Watchdog
- Fenster-Watchdog
- Intelligenter Watchdog

# Einstellungen am ATmega

Bit	7	6	5	4	3	2	1	0	
	-	-	-	WDCE	WDE	WDP2	WDP1	WDP0	WDTCR
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 bis 5 – **Res**: Reservierte Bits

Bit 4 – **WDCE**: Watchdog Change Enable

Bit 3 – **WDE**: Watchdog Enable

Bit 2 bis 0 – **WDP2, WDP1, WDP0**: Watchdog Timer Prescaler

# Einstellungen am ATmega

## Bit 4 – WDCE: Watchdog Change Enable

- muss gesetzt sein, um Watchdog zu sperren
- Bleibt nur 4 Taktzyklen gesetzt

## Bit 3 – WDE: Watchdog Enable

- Bei 1 -> Watchdog freigegeben
- Bei 0 -> Watchdogfunktion abgeschaltet

# Einstellungen am ATmega

Sperren des Watchdogs nach folgendem Ablauf:

1. **GLEICHZEITIG** eine **1** in Bits **WDCE** und **WDE** schreiben
2. Innerhalb der nächsten **4 Taktzyklen 0** in **WDE-Bit** schreiben

# Einstellungen am ATmega

WDP2	WDP1	WDP0	Number of WDT Oscillator Cycles	Typical Time-out at $V_{CC} = 3.0V$	Typical Time-out at $V_{CC} = 5.0V$
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

**Time-out-Zeiten des Watchdogs bei  $V_{CC} = 3.0V$  und  $V_{CC} = 5.0V$**

# Einstellungen am ATmega

## C Code Example

```
void WDT_off(void)
{
    /* reset WDT */
    _WDR();
    /* Write logical one to WDCE and WDE */
    WDTCR |= (1<<WDCE) | (1<<WDE);
    /* Turn off WDT */
    WDTCR = 0x00;
}
```

# Brown-out-detection

**Brown-out-detection** = löst Reset/Neustart aus



**Zu deutsch:** verminderte Lichtstärke

**Im Kontext:** Absinken der Versorgungsspannung

# Eintreten/Ursache

Mikrokontroller = PKW

- Wann fällt die Betriebsspannung?
  - ➔ – Schwächer werdende Batterien
  - Normaler Ausschaltvorgang, wenn Kondensator vorhanden ist

# Einstellungen am ATmega

- Atmega 8 hat Brown-out-detection (BOD) auf dem Chip implementiert
- Versorgungsspannung vergleichen mit einem festen Triggerlevel
- Triggerlevel kann entweder auf 2.7 V oder 4.0 V eingestellt werden
- BOD-Schaltung kann mit BODEN-Fuses ein- bzw. ausgeschaltet werden

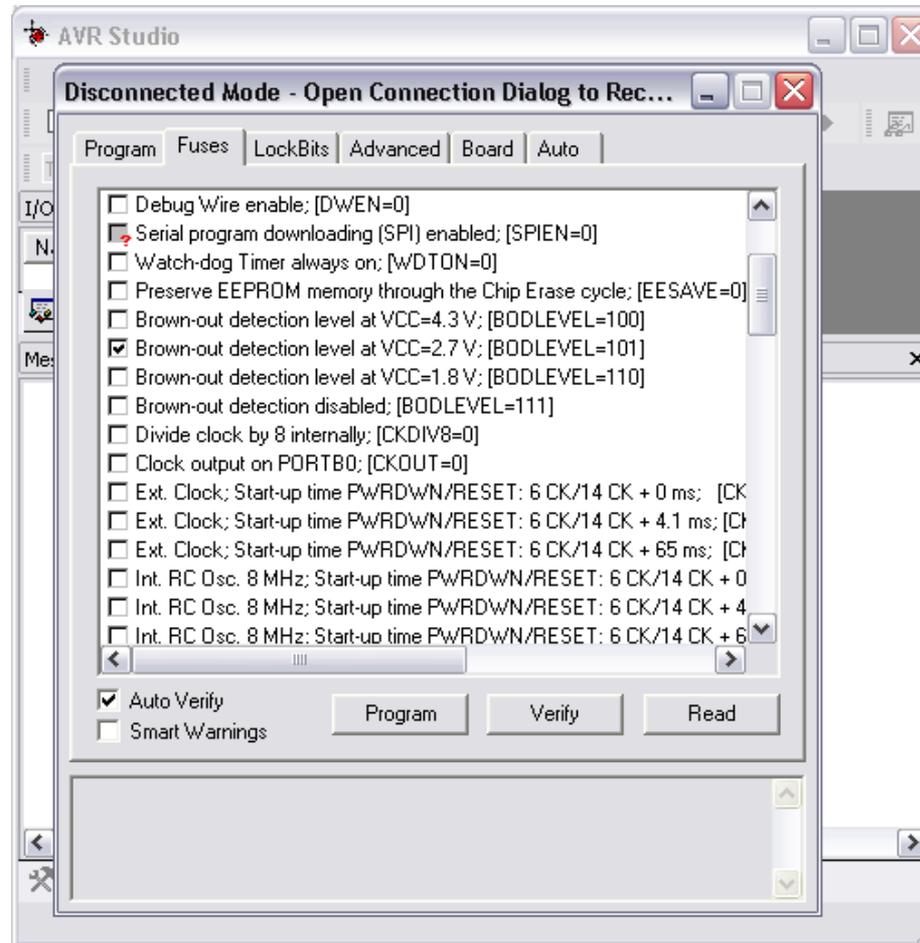
# Einstellungen am ATmega

**Fuse-Bits** (= Speicherzellen, die programmiert werden und bestimmte Funktionen aktivieren/deaktivieren)

**BODEN** aktiviert/deaktiviert BOD

**BODLEVEL** Einstellung des Spannungswertes, bei dem der Unterspannungsschutz aktiv werden soll

# Einstellungen am ATmega



Fragen, Anregungen???