

# SOFTWARE PWM CONTROLLER ATMEGA 8

## TYP: PWM18

### INHALTSVERZEICHNIS

Features .....	2
PWM- Controller Pinout .....	3
Speicherbelegung 1/2 .....	4
Speicherbelegung 2/2 .....	5
Die Befehle 1/2.....	6
Die Befehle 2/2.....	7
Das TWI- Protokoll .....	8
Control_Reg .....	9
SREG_1 - Statusregister TWI- Protokoll.....	10
SREG_2 - Statusregister Atomic Updates .....	11
SREG_3- Statusregister.....	12
PWM_CNTR_ADDR .....	13
CS_Soft_PWM.....	14
CS_HW_PWM_16.....	15
Res_HW_PWM_HB + Res_HW_PWM_LB .....	16
Res_Soft_PWM.....	17
PWM_vs_BUS .....	18
Standart Einstellungen nach dem Flashen.....	19
CMD- Beispiele 1/3 .....	20
CMD- Beispiele 2/3 .....	21
CMD- Beispiele 3/3 .....	22
Notizen .....	23
Change Log .....	24

## Features

- *TWI- Interface, Auto Increment*
- *Watchdog Überwachung*
- *Insgesamt 18- Frei wählbare PWM- Kanäle*
  - 16x Kanal (8-Bit) Software PWM, Variable Auflösung 2-8 Bit
  - @ 8MHz Internal RC and 8- Bit Resolution = 128- 250Hz PWM- Frequency
  - @ 16 MHz Crystal Clock and 8- Bit Resolution = 250- 500Hz PWM- Frequency
  - 2x Kanal 16- Bit Hardware PWM(Fast PWM), Variable Auflösung 2-16
- *480 Byte TWI- SRAM*
- *384 Byte TWI- E<sup>2</sup>PROM*
- *Über TWI- Bus zugängliche Hardware Einstellungen*
  - TWI- Slave Adresse, Standart SLA= **0x7A**
  - Clock Select Bytes Timer 1, 2 (PWM- Frequency)
  - Auflösung 16- Bit PWM(ICR1)
  - Auflösung 8-Bit Software PWM
  - Interruptzeit für Software PWM(!)
- *Alle Hardware Einstellungen lassen sich im EEPROM*
  - über einen einzigen Befehl abspeichern.
  - „Power Up Load“ gespeicherte PWM – Werte nach Reset laden(EEPROM), oder by default alle Kanäle mit 0 initialisieren.

# PWM- Controller Pinout

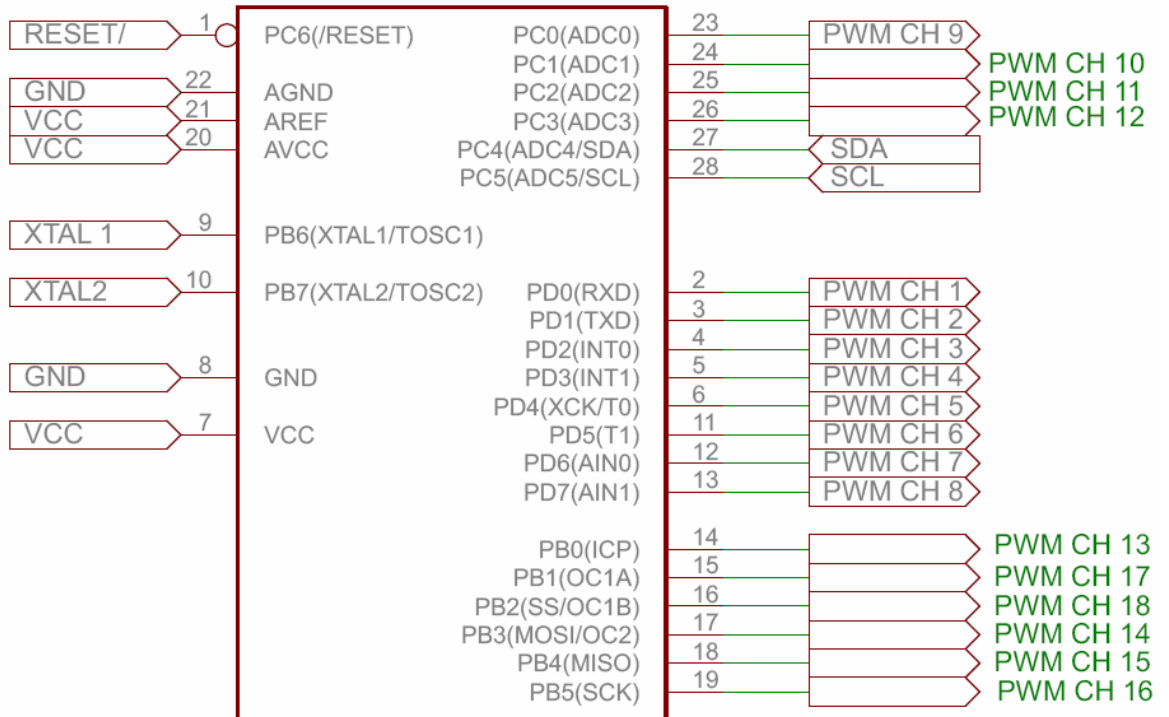
Reset, ISP, Clock und PowerSupply sollten entsprechend AVR Hardware Design Considerations AVR042 beschaltet werden.

Wobei ISP- nicht zwingend notwendig ist.

Ich würde ISP- empfehlen, da an der Firmware weiterhin fleißig gefeilt wird ☺

Crystal als ClockSource ist ratsam, wenn man die maximale Software- PWM Frequenz ausreizen möchte.

## PWM18





## AVR- Speicherreservierungen

SRAM		Adresse [*16]	Größe[Byte]
Reserviert für RX Befehle	Bottom	SRAM Start	128
	TOP		
TWI Receive Buffer	Bottom	E0	128
	TOP		
TWI Transmitt Buffer	Bottom	160	128
	TOP		
Temporary	Bottom		32
	TOP		
Daten Speicher	Bottom		480
	TOP		
<b>Stack</b>	Bottom	<b>RAMEND- 128</b>	<b>128</b>
	TOP	<b>RAMEND</b>	

EEPROM		Adresse [*16]	Größe[Byte]
Reserviert für RX Befehle	Bottom	EEPROM Start	128
	TOP		
EEPROM- Datenspeicher Bereich	Bottom		384
	TOP		

## Die Befehle 1/2

Das besondere an diesen Registern ist, dass sie im SRAM- gespeichert werden. Nach einem Ausschaltvorgang oder Reset, sind die Daten aus den Registern unbrauchbar !!

Deswegen gibt es für die Register 0x01-0x1C auch eine EEPROM- Speicherfunktion, siehe Seite Die Befehle 2/2 .

<b>TWI - CMD [HEX]</b>	<b>Funktion</b>	<b>R/W*</b>	<b>Notes</b>
<b>0</b>	—	—	none
<b>1</b>	PWM Kanal 1	R/W	
<b>2</b>	PWM Kanal 2	R/W	
<b>3</b>	PWM Kanal 3	R/W	
<b>4</b>	PWM Kanal 4	R/W	
<b>5</b>	PWM Kanal 5	R/W	
<b>6</b>	PWM Kanal 6	R/W	
<b>7</b>	PWM Kanal 7	R/W	
<b>8</b>	PWM Kanal 8	R/W	
<b>9</b>	PWM Kanal 9	R/W	
<b>0A</b>	PWM Kanal 10	R/W	
<b>0B</b>	PWM Kanal 11	R/W	
<b>0C</b>	PWM Kanal 12	R/W	
<b>0D</b>	PWM Kanal 13	R/W	
<b>0E</b>	PWM Kanal 14	R/W	
<b>0F</b>	PWM Kanal 15	R/W	
<b>10</b>	PWM Kanal 16	R/W	
<b>11</b>	PWM Kanal 17 HB	R/W	
<b>12</b>	PWM Kanal 17 LB	R/W	
<b>13</b>	PWM Kanal 18 HB	R/W	
<b>14</b>	PWM Kanal 18 LB	R/W	
<b>15</b>	<b>Control_Reg</b>	R/W	<b>Steuerregister</b>
<b>16</b>	PWM_CNTR_ADDR	R/W	PW18 TWI- Busadressen Register
<b>17</b>	CS_Soft_PWM	R/W	Clock Select für Soft- PWM Zeitbasis
<b>18</b>	CS_HW_PWM_16	R/W	Frequenz für 16-Bit PWM( Kanal 17,18)
<b>19</b>	Res_HW_PWM_HB	R/W	Auflösung HB 16-Bit PWM ( Kanal 17,18)
<b>1A</b>	Res_HW_PWM_LB	R/W	Auflösung LB 16-Bit PWM ( Kanal 17,18)
<b>1B</b>	Res_Soft_PWM	R/W	Auflösung für Soft PWM
<b>1C</b>	PWM_vs_BUS	R/W	Interruptzeit für Tim2 Comp
<b>1D</b>	SREG_3	R	Statusinformationen
<b>BF</b>	TWI- SRAM	R/W	Read/ Write – SRAM
<b>BE</b>	TWI- E <sup>2</sup> PROM	R/W	Read/ Write - EEPROM

**R/W\***- Read /Write, bei diesen Befehlen ist die Autoinkrement Funktion aktiv.

**R**- sind nur Lesebefehle, Autoinkrement Funktion ist Aktiv.

Nur Lesebefehle sind identisch zu den R/W- Befehlen, nur das in Schreibrichtung keine Parameter ausgewertet werden können.

**W**- Nur Schreibbefehle, Autoinkrement Funktion ist nicht aktiv !

## Die Befehle 2/2

<b>TWI - CMD [HEX]</b>	<b>Funktion</b>	<b>R/W*</b>	<b>Notes</b>
<b>C0</b>	HW_ Settings Save	W	Betroffene Register siehe Seite Speicherbelegung(1/2) Hardware Settings space
<b>C1</b>	PWM_ Values Save	W	PWM Kanäle 1-18 auf einmal abspeichern
<b>C2</b>	PWM Kanal 1 Save	W	Kanal einzeln abspeichern
<b>C3</b>	PWM Kanal 2 Save	W	
<b>C4</b>	PWM Kanal 3 Save	W	
<b>C5</b>	PWM Kanal 4 Save	W	
<b>C6</b>	PWM Kanal 5 Save	W	
<b>C7</b>	PWM Kanal 6 Save	W	
<b>C8</b>	PWM Kanal 7 Save	W	
<b>C9</b>	PWM Kanal 8 Save	W	
<b>CA</b>	PWM Kanal 9 Save	W	
<b>CB</b>	PWM Kanal 10 Save	W	
<b>CC</b>	PWM Kanal 11 Save	W	
<b>CD</b>	PWM Kanal 12 Save	W	
<b>CE</b>	PWM Kanal 13 Save	W	
<b>CF</b>	PWM Kanal 14 Save	W	
<b>D0</b>	PWM Kanal 15 Save	W	
<b>D1</b>	PWM Kanal 16 Save	W	
<b>D2</b>	PWM Kanal 17 HB Save	W	
<b>D3</b>	PWM Kanal 17 LB Save	W	
<b>D4</b>	PWM Kanal 18 HB Save	W	
<b>D5</b>	PWM Kanal 18 LB Save	W	
<b>D6</b>	<b>Control_Reg Save</b>	W	
<b>D7</b>	PWM_CNTR_ADDR Save	W	PW18 TWI- Busadressen Register
<b>D8</b>	CS_Soft_PWM Save	W	
<b>D9</b>	CS_HW_PWM_16 Save	W	
<b>DA</b>	Res_HW_PWM_HB Save	W	
<b>DB</b>	Res_HW_PWM_LB Save	W	
<b>DC</b>	Res_Soft_PWM Save	W	
<b>DD</b>	PWM_vs_BUS Save	W	

**R/W\***- Read /Write, bei diesen Befehlen ist die Autoinkrement Funktion aktiv.

**R**- sind nur Lesebefehle, Autoinkrement Funktion ist Aktiv.

Nur Lesebefehle sind identisch zu den R/W- Befehlen, nur das in Schreibrichtung keine Parameter ausgewertet werden können.

**W**- Nur Schreibbefehle, Autoinkrement Funktion ist nicht aktiv !

**Alle Befehle von 0xC0- 0xDD müssen mit groß ‚S‘ bestätigt werden, sonst kommt ein Speichervorgang nicht zustande, siehe Beispiele auf den Seiten „CMD- Beispiele“ .**

# Das TWI- Protokoll

## Slave Receiver Modus

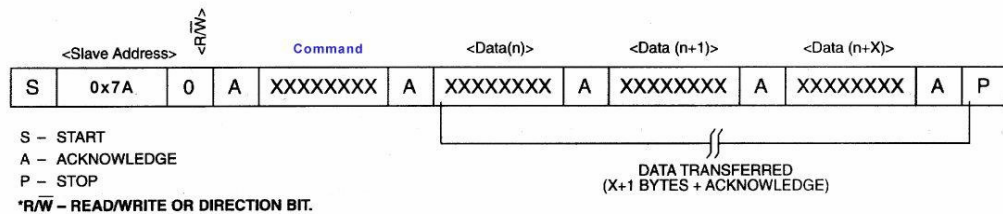
Es lassen sich je Zugriff maximal 128 Byte transportieren.  
Sende = Empfangsbuffer = 128 Byte.

- 1) Schreiben (Register 0x00 bis 0x80) und (0xC0 bis 0xDD).
- 2) Lesen (Befehle 0x00 bis 0x80).

### 1) Schreiben

Ein einfacher Schreibbefehl besteht aus **mindestens 2 Byte**.  
**Byte1:** Ist das Befehlsbyte siehe Seite Befehle „TWI-CMD“  
**Byte2:** Datenbyte, je nach Befehl, notfalls ein Dummybyte, aber immer mindestens zwei Byte.  
**Byte3 +(n)** wird durch Autoinkrement in das nächsten Register hineinkopiert.

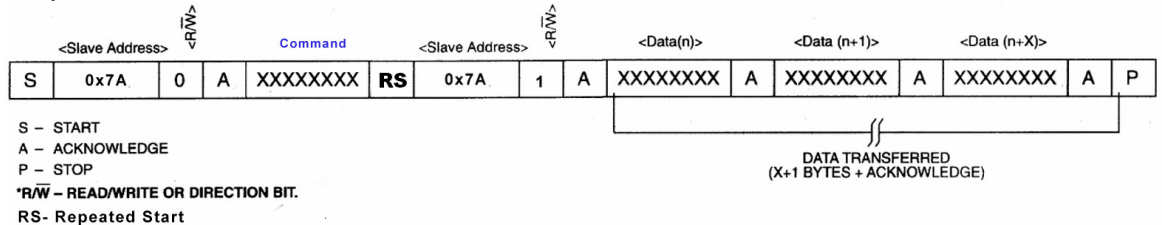
Beispiel:



### 2) Lesen

Ein Lesebefehl besteht aus **nur 1 Byte**.  
**Byte1(CMD):** Ist das Befehlsbyte siehe Seite Befehle „TWI-CMD“  
Autoinkrement ist auch hier aktiv.

Beispiel:





## Control\_Reg

### Beschreibung:

Dieses Register ist für das Verhalten nach einem Reset(Einschaltphase) zuständig.

Bit

	7	6	5	4	3	2	1	0
	—	—	—	—	—	—		PWUPL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 –
- Bit 6 –
- Bit 5 –
- Bit 4 –
- Bit 3 –
- Bit 2 –
  
- Bit 1 –
- Bit 0 – **PWUPL: Power Up load saved PWM- Values**

Bei Logisch „1“ werden die PWM- Werte für Kanal 1-18 aus dem EEPROM geladen. Dies ist sinnvoll wenn die PWM- Kanäle nach dem einschalten einen Wert abweichend von 0x00- besitzen sollen.

Bei Logisch =0 werden alle oben aufgelistete PWM- Kanäle by Default mit 0 geladen.

## SREG\_1 - Statusregister TWI- Protokoll

### Beschreibung:

Internes Statusregister, für Benutzer irrelevant.

Die Veränderungen werden im Hauptregister SREG\_1(R22) vorgenommen.

Bit

	7	6	5	4	3	2	1	0
	UPDA	UPDA2	UPDA3	UPDA4	TWWRG	UPDA5	SL_RX	SL_TX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 – **UPDA: Update Flag**

Mehr Details siehe TWI\_Programm PAP.

- Bit 6 – **UPDA2: Update Flag 2**

Mehr Details siehe TWI\_Programm PAP.

- Bit 5 – **UPDA3: Update Flag 3**

Mehr Details siehe Programm PAP.

- Bit 4 – **UPDA4: Update Flag 4**

Mehr Details siehe Programm PAP.

- Bit 3 – **TWWRG: Two Wire Wrong Operation**

Allgemeine Zugriffsverweigerung innerhalb des TWI- Protokolls.

Mehr Details siehe TWI\_Programm PAP.

- Bit 2 – **UPDA5: Update Flag 5**

Mehr Details siehe TWI\_RXD\_Decoder PAP.

- Bit 1 – **SL\_RX: Slave Receiver Mode**

Wird dann gesetzt, wenn “// Own SLA+R has been received; ACK has been returned” = \$A8(TWSR) im Statusregister steht. Dieses Flag wird nicht ausgewertet und ist nicht von Bedeutung! Mehr Details siehe TWI\_Programm PAP.

- Bit 0 – **SL\_TX: Slave Transmitter Mode**

Wird dann gesetzt, wenn “Own SLA+W has been received; ACK has been returned” = \$60(TWSR) im Statusregister steht. Dieses Flag wird nicht ausgewertet und ist nicht von Bedeutung! Mehr Details siehe TWI\_Programm PAP.

## SREG\_2 - Statusregister Atomic Updates

### Beschreibung:

Internes Statusregister, ist für Benutzer irrelevant.

Die Veränderungen werden im Hauptregister SREG\_2(R23) vorgenommen.

Die Flags(UPDA32-30) dienen zu Updatefreigaben innerhalb der TIM2 COMP ISR.

Bit

	7	6	5	4	3	2	1	0
	—	—	—	UPDA32	UPDA31	UPDA30	—	—
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 –

- Bit 6 –

- Bit 5 –

- Bit 4 – **UPDA32:**

UPDA32= ICR1A- Write Enable. Mehr Details siehe TIM2\_COMP PAP.

- Bit 3 – **UPDA31:**

UPDA31= OCR1B- Write Enable. Mehr Details siehe TIM2\_COMP PAP.

- Bit 2 – **UPDA30:**

UPDA30= OCR1A- Write Enable. Mehr Details siehe TIM2\_COMP PAP.

- Bit 1 –

- Bit 0 –

## SREG\_3- Statusregister

### Beschreibung:

Dieses Register sollte unmittelbar nach einem Schreibvorgang ausgelesen werden, da sonst die Statusinformationen mit dem nächsten Schreibbefehl verfallen.

Ein Lesebefehl wiederum hat keinen Einfluss auf dieses Register.

Bit

	7	6	5	4	3	2	1	0
	—	—	—	SLAR	ELAR	SBWSL	HSWSL	PVWSL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 –
- Bit 6 –
- Bit 5 –

- Bit 4 – **SLAR: SRAM last address reached.**

Wird gesetzt „1“, wenn im Schreibmodus die höchste SRAM- Adresse erreicht wurde, und umgekehrt.

- Bit 3 – **ELAR: EEPROM last address reached.**

Wird gesetzt „1“, wenn im Schreibmodus die höchste EEPROM- Adresse erreicht wurde, und umgekehrt.

- Bit 2 – **SBWSL: Single Byte write successful**

Die Speicherbefehle 0xC2- 0xDD und TWI- E<sup>2</sup>PROM (0xBE) greifen auf eine und dieselbe EEPROM Speicherroutine, und beeinflussen somit dieses Flag. Sollten während einem Schreibversuch die geschriebenen Daten mit den zu schreibenden Daten nicht übereinstimmen, so wird dieses Flag gelöscht „0“, und umgekehrt.

Weitere Details lassen sich auch dem PAP „EEP\_Byte\_Save“ entziehen.

- Bit 1 – **HSWSL: Hardware Settings write successful**

Ist high wenn der Befehl „HW\_ Settings Save“ (0xC0) erfolgreich ausgeführt wurde.

- Bit 0 – **PVWSL: PWM- Values write successful**

Ist high wenn der Befehl „PWM\_ Values Save“(0xC1) erfolgreich ausgeführt wurde.

# PWM\_CNTR\_ADDR

## Beschreibung:

PWM\_CNTR\_ADDR ist die TWI- Busadresse.

Die Busadressen wird mit dem Befehl CMD 0x16 geschrieben. Diese Busadresse ist dann temporär bis zum Reset gültig, außer der Befehl CMD **0xD7** oder 0xC0 wurde ausgeführt. So ist dies dann die neue Busadresse auch nach Reset.

TWI (Slave) Address Register – TWAR

Bit

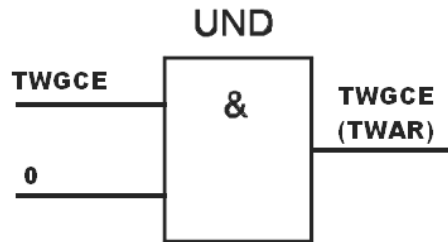
	7	6	5	4	3	2	1	0
	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7..1 – TWA: TWI (Slave) Address Register

Diese 7 Bits stellen die Slaveadresse des PWM- Controllers da.

- Bit 0 – TWGCE: TWI General Call Recognition Enable Bit

Dieses Bit kann nicht gesetzt werden, weil es beim auslesen TWGCE & „0“ Verknüpft wird, somit ist das TWGCE immer Zero.



## NOTES:

### Bus Adresse wiederherstellen:

Die default Adresse \$7A kann durch erneutes Flashen des E<sup>2</sup>PROM's wiederhergestellt werden, wobei alle andere Hardware Settings flöten gehen :-p

## CS\_Soft\_PWM

### Beschreibung:

Software PWM- Clock Select Bits. Es wird empfohlen den Wert by default zu lassen.

Standardmäßig werden die CS- Bits mit 1 geladen, weil bei höheren CS- Werten die Soft- PWM sonst viel zu langsam wird.

Dieses Register ist für die Clock Settings von TimerCounter 2 beim AVR zuständig. Die fixe Werte für Bit 7-3 werden beim initialisieren geladen, und sind nicht veränderbar. Wobei die Bits 2-0 die einzigen variierbaren Werte darstellen. Somit kann man die kompletten Einstellungen nicht zerschießen. Der Grundgedanke ist eigentlich eine Schutzmassnahme gegen nicht entsprechende Daten Eingabe. Die initialisierungswerte für Bit 7-3 kann man aus dem Codeschnipsel entziehen.

Bit

	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS22	CS21	CS20
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- Bit 7 –
- Bit 6 –
- Bit 5 –
- Bit 4 –
- Bit 3 –
- Bit 2:0 – CS22:0: Clock Select

Table 46. Clock Select Bit Description

CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$\text{clk}_{T2S}/(\text{No prescaling})$
0	1	0	$\text{clk}_{T2S}/8$ (From prescaler)
0	1	1	$\text{clk}_{T2S}/32$ (From prescaler)
1	0	0	$\text{clk}_{T2S}/64$ (From prescaler)
1	0	1	$\text{clk}_{T2S}/128$ (From prescaler)
1	1	0	$\text{clk}_{T2S}/256$ (From prescaler)
1	1	1	$\text{clk}_{T2S}/1024$ (From prescaler)

### TCCR2 – Initialisierungs- Ausschnitt

```

; Timer Counter2 Init. // CTC- Mode
ldi    temp, (0<<WGM20)|(0<<COM21)|(1<<WGM21)|(0<<COM20)|(0<<CS22)|(0<<CS21)|(0<<CS20)
out    TCCR2, temp
ldi    temp, 1<<OCIE2
out    TIMSK, temp
    
```

Als erstes wird TCCR2 mit den oberen Werten initialisiert, daraufhin werden die Daten aus dem EEPROM siehe CS\_Soft\_PWM\_EEP= \$015 geladen und über eine Bitmaske nur die CS- Bits verändert. Siehe Codeausschnitt unten.

```

; Clock Select Software PWM. TCCR2 CS
adiw   Low_Byte, 1 // EEPROM- Pointer
rcall  EEPROM_read // Clock Select Software PWM
andi   temp, (0<<WGM20)|(0<<COM21)|(0<<WGM21)|(0<<COM20)|(1<<CS22)|(1<<CS21)|(1<<CS20)
ST     X+, temp
in     temp2, TCCR2
andi   temp2, (1<<WGM20)|(1<<COM21)|(1<<WGM21)|(1<<COM20)|(0<<CS22)|(0<<CS21)|(0<<CS20)
or     temp, temp2
out    TCCR2, temp
    
```

# CS\_HW\_PWM\_16

## Beschreibung:

Von diesen Clock Select- Bits hängt die Geschwindigkeit des PWM- Outputs OC1A und OC1B ab. Standardmäßig werden die CS- Bits mit 1 geladen.

Dieses Register ist für die Clock Settings von TimerCounter 1 zuständig. Die fixe Werte für Bit 7-3 werden beim initialisieren ins TCCR1B geladen und sind nicht veränderbar.

Wobei die Bits 2-0 die einzigen variierbaren Werte darstellen. Somit kann man die kompletten Einstellungen nicht zerschießen. Der Grundgedanke ist eigentlich eine Schutzmaßnahme gegen nicht entsprechende Dateneingabe.

Die Initialisierungswerte für Bit 7-3 kann man aus dem Codeschnipsel entziehen.

TCCR1A-  
Timer/Counter1  
Control Register A –  
TCCR1A

Bit

	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

TCCR1B-  
Timer/Counter1  
Control Register B –  
TCCR1B

Bit

	7	6	5	4	3	2	1	0
	ICNC1	ICES1	—	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

Table 40. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{IO}/1$ (No prescaling)
0	1	0	$clk_{IO}/8$ (From prescaler)
0	1	1	$clk_{IO}/64$ (From prescaler)
1	0	0	$clk_{IO}/256$ (From prescaler)
1	0	1	$clk_{IO}/1024$ (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

## TCCR1A,1B – Initialisierungs- Ausschnitt

```

; Timer Counter 1 Init , MODE 14, Fast PWM , TOP = ICR1
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ldi    temp,
(1<<COM1A1)|(1<<COM1A0)|(1<<COM1B1)|(1<<COM1B0)|(0<<FOC1A)|(0<<FOC1B)|(1<<WGM11)|(0<<WGM10)
out    TCCR1A, temp

ldi    temp, (0<<ICNC1)|(0<<ICES1)|(0<<5)|(1<<WGM13)|(1<<WGM12)|(0<<CS12)|(0<<CS11)|(0<<CS10) //CS->
.max 5
out    TCCR1B, temp
    
```

## Res\_HW\_PWM\_HB + Res\_HW\_PWM\_LB

### Beschreibung:

Legt die Auflösung für die PWM- Outputs CH 17,18(OC1A, OC1B) fest.

Die Werte für die Auflösung werden erst dann übernommen, wenn zuerst ins cmd\_19 und cmd\_1A die neue Auflösung geschrieben wurde. Wobei ein ICR1 update(Atomic Write) durch den Befehl cmd\_1A freigegeben wird.

Die Daten aus diesen beiden Registern (Res\_HW\_PWM\_HB + Res\_HW\_PWM\_LB) werden in ICR1H und ICR1L in der TIM2 COM ISR geladen. TimerCounter1 wird in CTC- Modus betrieben, somit bestimmt die ICR Einheit die Auflösung.

Bit	7	6	5	4	3	2	1	0
Input Capture Register 1 – ICR1H and ICR1L	ICR1 [15:8]							
	ICR1 [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

ICR1H←Res\_HW\_PWM\_HB\_SRAM

ICR1L←Res\_HW\_PWM\_LB\_SRA

### Beispiel: Auflösung ändern

- 1) Start
- 2) SLA+W
- 3) 0x19 (CMD)
- 4) 0xFF ( Auflösung High\_Byte)
- 5) 0xFF (Auflösung Low\_Byte)
- 6) Stoppkondition

Somit wurde die neue Auflösung übernommen und temporär gespeichert. Will man die gesetzte Auflösung auch nach einem Reset nicht missen, so sollte anschließend der Befehl **0xC0** oder **0xDA** und **0xDB** ausgeführt werden. Mehr Details auf Seite „Die Befehle“.



## Res\_Soft\_PWM

### Beschreibung:

Auflösung für Software PWM- Kanäle 1-16.

Wird die Auflösung von 8 auf 7-Bit umgestellt, so steigt die Software PWM- Frequenz auf das doppelte.

**Achtung!** mit sinkender Auflösung in 1- Bit Schritten verdoppelt sich die PWM- Frequenz ums doppelte.

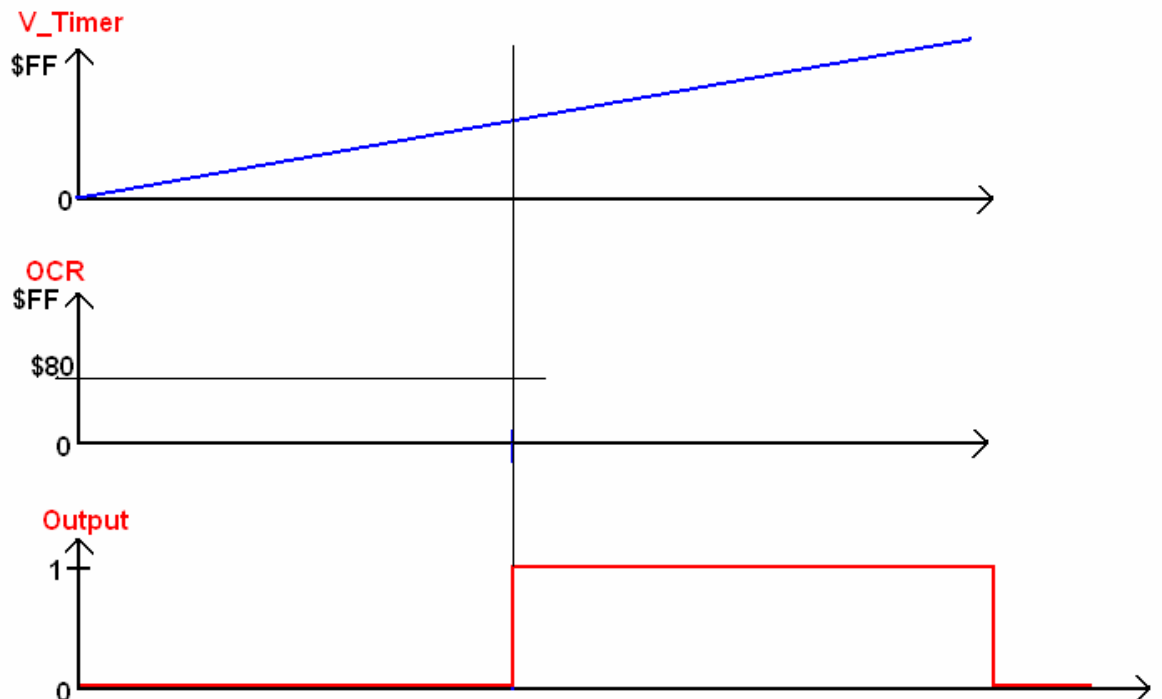
Bit

	7	6	5	4	3	2	1	0
	Res_Soft_PWM_SRAM							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

$R20 \leftarrow \text{Res\_Soft\_PWM\_SRAM}$

$\text{Res\_Soft\_PWM\_SRAM} = \text{PWM\_Cnt\_Top}(R20)$ .

Das Register Res\_Soft\_PWM bestimmt den Top- Wert des PWM\_Counters(V\_Timer). Dieser ist in seiner Funktion Identisch mit einem Hardware Timer in CTC- Modus.



# PWM\_vs\_BUS

## Beschreibung:

Default Wert 0xFF.

Wird dessen Wert reduziert so beschleunigt sich die Soft- PWM um einige Hz.

Wurde ein zu geringer Wert ausgewählt, so hat der AVR viel zu wenig Zeit für das TWI-Protokoll, weshalb unter Umständen der AVR unerreichbar bleibt!!

Der kleinste Wert von „PWM\_vs\_BUS“ darf keinesfalls geringer als die längste Interruptzeit( siehe. „TIM2\_COMP- Interruptzeiten“) sein.

Dieses Register bestimmt den TimerCounter2 Top- Wert(CTC).

OCR2←PWM\_vs\_BUS\_SRAM

Bit	7	6	5	4	3	2	1	0
Output Compare Register – OCR2	OCR2							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

## TIM2\_COMP- Zeiten.

TIM2\_COMP Max. clock cycles = **96**

TIM2\_COMP Min. clock cycles = **47**

### **Für wem die default Frequenz nicht ausreicht !**

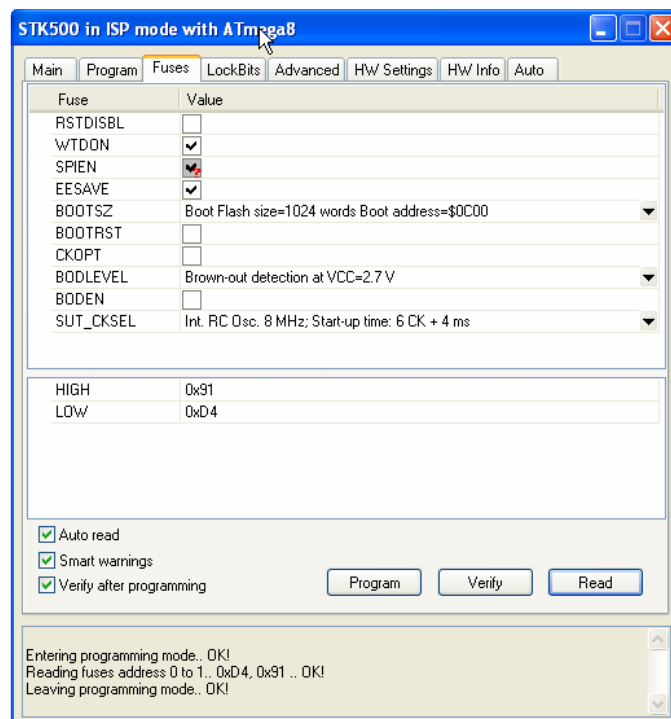
Getestet mit einem Wert von **128**. Die Firmware läuft weiterhin Fehlerfrei. Die PWM- Frequenz ist dabei um das Doppelte gestiegen wobei es sich immer noch um 8- Bit Soft- PWM handelt.

## Standart Einstellungen nach dem Flashen

- 1) TWI- Busadresse: 0x7A
- 2) Software PWM Auflösung: 8-Bit
- 3) Hardware PWM Auflösung: 16- Bit
- 4) ClockSelect von Soft-(Timer2) und Hard- PWM(Timer1) \_ CLK divided by 1
- 5) Interruptzeit für Soft PWM(PWM\_vs\_BUS): \$FF
- 6) Control Register wird mit 0x00 geladen

### FUSE Settings:

- 1) RSTDISBL sollte Logisch „0“ bleiben.
- 2) Wirklich wichtig ist nur WTDON, damit der Wachhund auch mal was schafft ☺
- 3) Will man sich einen Quarz sparen so sollte SUT\_CKSEL wie im Bild gesetzt werden.



## CMD- Beispiele 1/3

### Aufgabe:

PWM Kanal 10 sollte einen neuen Wert erhalten. (Register **schreiben**)

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte1: 0x0A** (CMD für Kanal 10))
- 4) **Byte2: 0x64** (neuer Wert).
- 5) Stoppkondition

Jetzt wurde der neue PWM- Wert für Kanal 10 temporär übernommen.

### Aufgabe:

PWM Kanal 6 sollte im EEPROM **gespeichert** werden.

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte1: 0xC7** (CMD)
- 4) **Byte2: ,S'** (Großbuchstabe).
- 5) Stoppkondition

Jetzt wurde der neue PWM- Wert von Kanal 6 im EEPROM abgespeichert.

### Aufgabe:

Mehrere Kanäle mit neuen Werten **überschreiben**.

PWM- Kanäle 8-11 sollte mit neuen Werten beschrieben werden.

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte 1: 0x08** (CMD)
- 4) **Byte 2: 0x32** (neuer Wert für Kanal 8).
- 5) **Byte 3: 0x16** (neuer Wert für Kanal 9).
- 6) **Byte 4: 0x80** (neuer Wert für Kanal 10).
- 7) **Byte 5: 0x30** (neuer Wert für Kanal 11).
- 8) **Stoppkondition**  
**u.s.w**

Jetzt wurden die neuen PWM- Wert für die Kanäle 8-11 temporär übernommen.

### Aufgabe:

Mehrere Kanäle **auslesen**.

PWM- Kanäle 8-11 sollte ausgelesen werden.

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte 1: 0x08** (CMD)
- 4) **Repeated Start**
- 5) SLA+R
- 6) **RX\_Byte 1: 0xXX** (Wert von Kanal 8).
- 7) **RX\_Byte 2: 0xXX** (Wert von Kanal 9).
- 8) **RX\_Byte 3: 0xXX** (Wert von Kanal 10).
- 9) **RX\_Byte 4: 0xXX** (Wert von Kanal 11).
- 10) **Stoppkondition**  
**u.s.w**

## CMD- Beispiele 2/3

### Aufgabe:

Alle PWM- Kanäle 1-18 auf einmal ab**speichern**. (CMD= PWM\_ Values Save)

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte1: 0xC1** (CMD)
- 4) **Byte2: ,S'** (Großbuchstabe).
- 5) Stoppkondition

Danach sind die neuen PWM- Werte für Kanal 1-18 im EEPROM gespeichert.

### Aufgabe:

Alle Hardware Settings Register auf einmal **speichern**.(CMD= HW\_ Settings Save)

### Lösung:

- 1) Start
- 2) SLA+W
- 3) **Byte 1: 0xC0** (CMD)
- 4) **Byte 2: ,S'** (Großbuchstabe).
- 5) Stoppkondition

Danach sind die neuen Hardware Settings im EEPROM gespeichert.

### Allgemeine Infos:

ASCII: S = HEX: 53 = DEC: 83

Welche Register zu den „**Hardware Settings**“ gehören, kann man anhand der Tabelle siehe Seite Speicherbelegung 1/2 /SRAM-Unterteilung/ „Hardware Settings space“ erkennen.

Welche Register zu den „**PWM\_ Values**“ gehören, kann man anhand der Tabelle siehe Seite Speicherbelegung 1/2 /SRAM-Unterteilung/ „PWM- Values Space“ erkennen.

### **TWI SRAM- Befehl (0xBF)**

Mit diesen Befehl können Daten IN und AUS dem zugesicherten SRAM- Bereich des µControllers geschrieben bzw. gelesen werden.

#### **SRAM- Adressen:**

TWI- SRAM Bottom: **0x0000**

TWI- SRAM Top: **0x01E0**

### Aufgabe:

Daten in den SRAM- Bereich schreiben.

### Lösung:

- 1) SLA+W
- 2) 0xBF (CMD)
- 3) 0x00 (High\_Byte Adresse)
- 4) 0x00 (Low\_Byte\_Adresse)
- 5) 0x13 ( Daten)
- 6) 0x64 (Daten) ....
- 7) Stoppkondition

## CMD- Beispiele 3/3

### Aufgabe:

Daten aus'm SRAM- Bereich lesen.

### Lösung:

- 1) SLA+W
- 2) 0xBF (CMD)
- 3) 0x00 (High\_Byte Adresse)
- 4) 0x00 (Low\_Byte Adresse)
- 5) Repeated Start
- 6) SLA+R
- 7) **RX\_Byte 1:** 0xXX (SRAM Daten)
- 8) **RX\_Byte 2:** 0xXX (SRAM Daten) ....
- 9) Stoppkondition

**Achtung ! Das zuletzt ausgelesene DatenByte muss durch den Master mit „NACK“ quittiert werden.**

### TWI E<sup>2</sup>PROM - Befehl „(0xBE)“

Mit diesen Befehl können Daten IN und AUS dem zugesicherten E<sup>2</sup>PROM- Bereich des  $\mu$ Controllers geschrieben bzw. gelesen werden.

#### **EEPROM- Adressen**

TWI E<sup>2</sup>PROM Bottom: **0x0000**

TWI E<sup>2</sup>PROM Top: **0x017F**

Schreib und Lesezugriffe geschehen auf derselben Art und Weise wie beim Befehl TWI SRAM (0xBF).

## ***Notizen***

### **BEZEICHNUNG**

#### **PWM18 V1.0**

PWM– Steht für PWM- Firmware

18- Für Anzahl der PWM- Kanäle

V1.0- Firmware Version

## **Change Log**

### **PWM18 v1.3 (05.10.2009)**

- 1) Im Source Code und in der Beschreibung wurde SREG\_3 zu SREG\_2 umgetauft.
- 2) Im Source Code und in der Beschreibung wurde PWM\_vs\_CALC zu PWM\_vs\_BUS.
- 3) CRC8 Umsetzung ist zwar realisierbar jedoch wurde diese Funktion gestrichen, weil die wenigsten kommerziellen IIC- IC über so eine Funktion verfügen.
- 4) Das letzte Register 0x1C(PWM\_vs\_BUS) konnte nicht den richtigen Wert bei Abfrage zurückgeben. Fehler wurde beseitigt.
- 5) SRAM- letzte Adresse kann nicht beschrieben/gelesen werden. Referenzwert korrigiert. Fehler wurde behoben.
- 6) Statusregister SREG\_3 ist hinzugekommen. Siehe Seite SREG\_3.

### **PWM18 v1.2 (04.10.2009)**

- 1) Bei lesen der Register 0x01- 0x1C gab's einen Lesefehler, dieser wurde beseitigt.
- 2) Es ist ein neuer Befehl zur EEPROM- Speicherung dazugekommen. Die Beschreibung wurde auf's neue überarbeitet. Insbesondere die Punkte Befehle und Beispiele.

### **PWM18 v1.1 (02.10.2009)**

Es wurden neue Befehle 0xC2-0xDD hinzugefügt. Mit diesen Befehlen können sämtlich Register einzeln abgespeichert werden. Siehe Seite „Die Befehle 2/3“

### **PWM18 v1.0(01.10.2009)**

Die erste veröffentlichte Version.