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 only 10
SREG_3 - Statusregister only 11
PWM_CNTR_ADDR 12
CS_Soft_PWM 13
CS_HW_PWM_16 14
Res_HW_PWM_HB + Res_HW_PWM_LB 15
Res_Soft_PWM 16
PWM_vs_CALC 17
Standart Einstellungen nach dem Flashen 18
CMD- Beispiele 1/3 19
CMD- Beispiele 2/3 20
CMD- Beispiele 3/3 21
Bekannte BUGS 22
Notizen 23
Change Log 24

Die Flowcharts sind teilweise
flöten gegangen.
Böses copy paste :- (

Features

- ***TWI- Interface, Auto Increment***
- ***Watchdog Überwachung***
- ***Insgesamt 18- Frei wählbare PWM-*** **Kanäle**

16x Kanal (8-Bit) Software PWM, Variable Auflösung 2-8 Bit
@ 16MHz Crystal Clock and 8- Bit Resolution = 245Hz PWM- Frequency
2x Kanal 16- Bit Hardware PWM(Fast PWM), Variable Auflösung 2-16

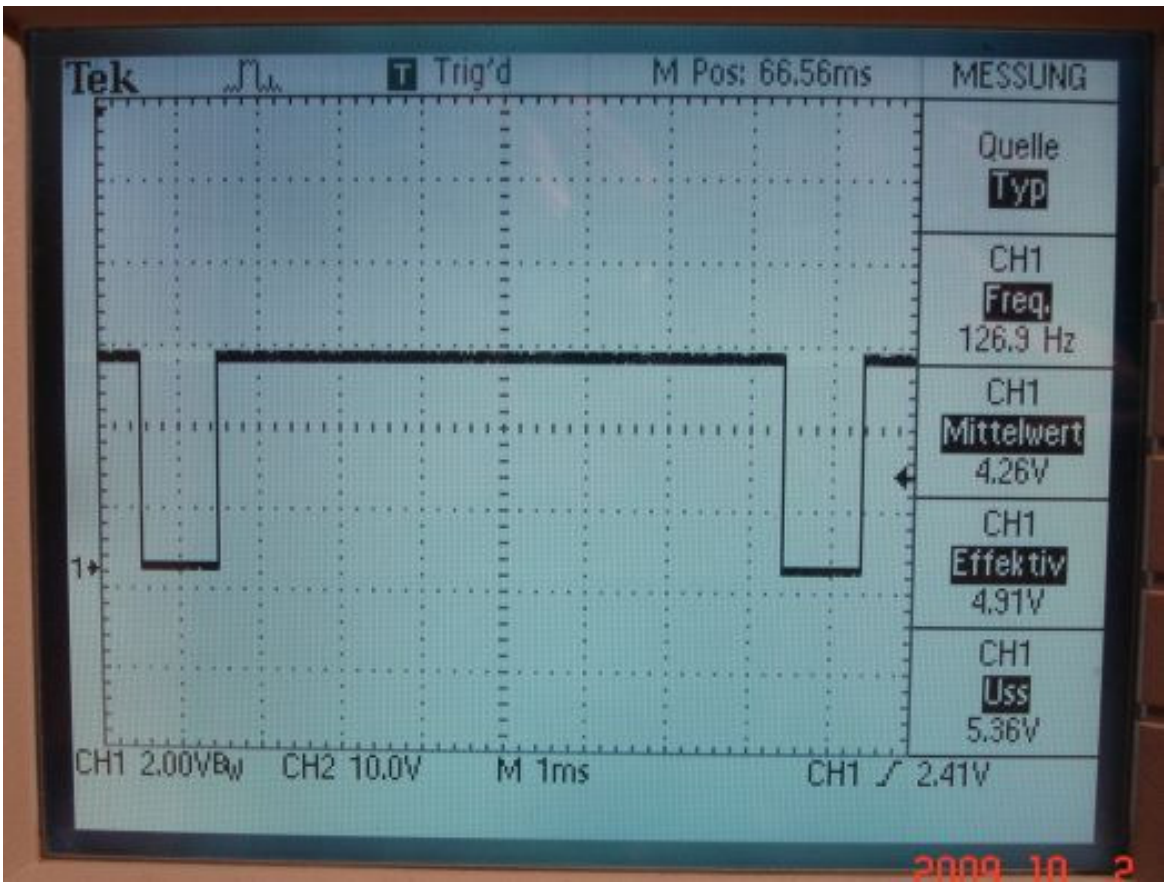
- **480 Byte TWI- SRAM**
- **384 Byte TWI- E²PROM**
- **CRC 8,16?? Serial or Table Realisierung noch nicht umgesetzt !!!!!!!!!!!!**
- ***Über TWI- Bus zugängliche Hardware Einstellungen***

– 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***

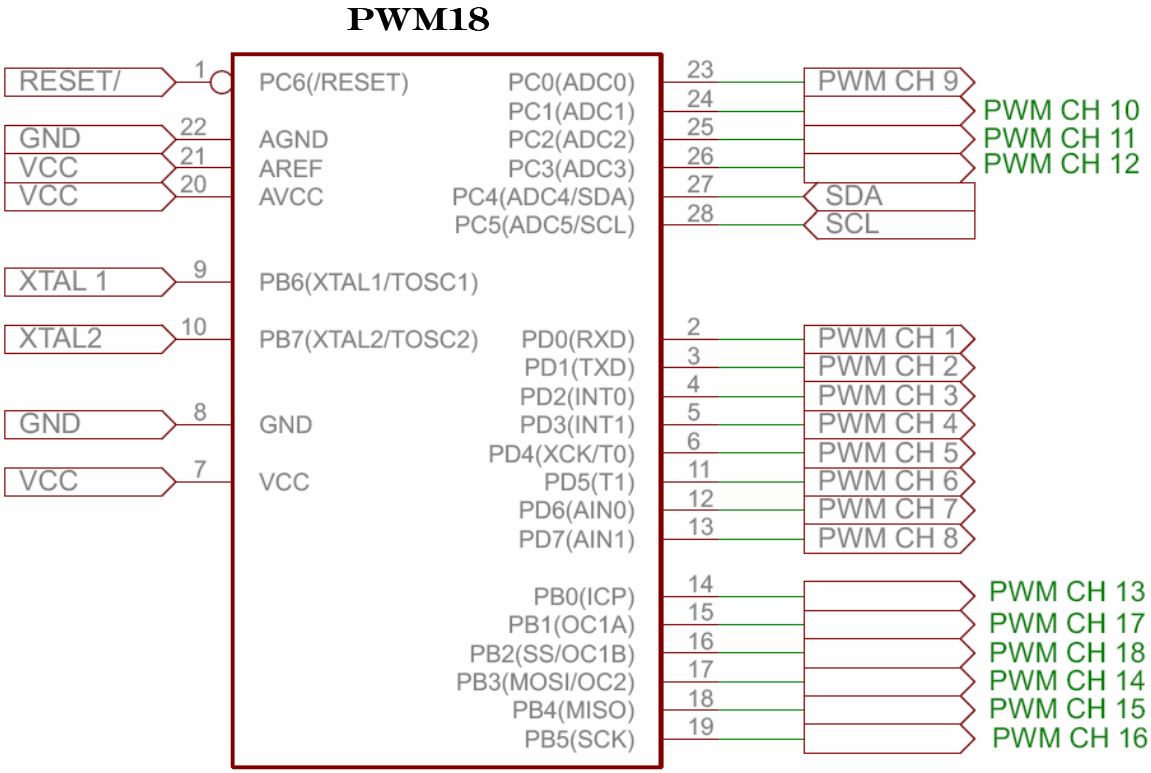
über einen einzigen Befehl abspeichern.
– „Power Up Load“ gespeicherte PWM – Werte nach Reset laden(EEPROM), oder by default alle Kanäle mit 0 initialisieren.
–

Software PWM sample @ 8MHz internal RC-Oscillator.



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.



Speicherbelegung 1/2

Byte	*Addresse EEPROM*	*EEPROM- Label*	*SRAM- Label*	*Addresse SRAM*	*SRAM-Unterteilung*	
1	$000	OCR_Ch_01_EEP	OCR_Ch_01_SRAM	$60	**Software PWM Channels (GPR)**	**PWM-Values Space**
2	$001	OCR_Ch_02_EEP	OCR_Ch_02_SRAM	$61		
3	$002	OCR_Ch_03_EEP	OCR_Ch_03_SRAM	$62		
4	$003	OCR_Ch_04_EEP	OCR_Ch_04_SRAM	$63		
5	$004	OCR_Ch_05_EEP	OCR_Ch_05_SRAM	$64		
6	$005	OCR_Ch_06_EEP	OCR_Ch_06_SRAM	$65		
7	$006	OCR_Ch_07_EEP	OCR_Ch_07_SRAM	$66		
8	$007	OCR_Ch_08_EEP	OCR_Ch_08_SRAM	$67		
9	$008	OCR_Ch_09_EEP	OCR_Ch_09_SRAM	$68		
10	$009	OCR_Ch_10_EEP	OCR_Ch_10_SRAM	$69		
11	$00A	OCR_Ch_11_EEP	OCR_Ch_11_SRAM	$6A		
12	$00B	OCR_Ch_12_EEP	OCR_Ch_12_SRAM	$6B		
13	$00C	OCR_Ch_13_EEP	OCR_Ch_13_SRAM	$6C		
14	$00D	OCR_Ch_14_EEP	OCR_Ch_14_SRAM	$6D		
15	$00E	OCR_Ch_15_EEP	OCR_Ch_15_SRAM	$6E		
16	$00F	OCR_Ch_16_EEP	OCR_Ch_16_SRAM	$6F		
17	$010	OCR_Ch_OC1AH_EEP	OCR_Ch_OC1AH_SRAM	$070	**Hardware PWM Channels**	
18	$011	OCR_Ch_OC1AL_EEP	OCR_Ch_OC1AL_SRAM	$071		
19	$012	OCR_Ch_OC1BH_EEP	OCR_Ch_OC1BH_SRAM	$072		
20	$013	OCR_Ch_OC1BL_EEP	OCR_Ch_OC1BL_SRAM	$073		
25	**$014**	**Control_Reg_EEP**	**Control_Reg_SRAM**	**$074**	**Hardware Settings space**	
26	$015	PWM_CNTR_ADDR_EEP	PWM_CNTR_ADDR_SRAM	$075		
27	$016	CS_Soft_PWM_EEP	CS_Soft_PWM_SRAM	$076		
28	$017	CS_HW_PWM_16_EEP	CS_HW_PWM_16_SRAM	$077		
29	$018	Res_HW_PWM_HB_EEP	Res_HW_PWM_HB_SRAM	$078		
30	$019	Res_HW_PWM_LB_EEP	Res_HW_PWM_LB_SRAM	$079		
31	$01A	Res_Soft_PWM_EEP	Res_Soft_PWM_SRAM	$07A		
32	$01B	PWM_vs_CALC_EEP	PWM_vs_CALC_SRAM	$07B		
33	$01C	CRC8_Poly_EEP	CRC8_Poly_SRAM	$07C		

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		
Stack	Bottom	**RAMEND- 128**	**128**
	TOP	**RAMEND**	

Die Befehle 1/2

Das besondere an diesen Befehlen(Registern) ist das Sie im SRAM- gespeichert werden. Nach einem Ausschaltvorgang oder Reset sind die Daten aus den Registern unbrauchbar !!
Deswegen gibt es für die Register 0x01-0x1D auch eine EEPROM- Speicherfunktion, siehe Seite Die Befehle 2/2 .

TWI - CMD [HEX]	*Funktion*	*R/W**	*Notes*
0	—	—	none
1	PWM Kanal 1	R/W	
2	PWM Kanal 2	R/W	
3	PWM Kanal 3	R/W	
4	PWM Kanal 4	R/W	
5	PWM Kanal 5	R/W	
6	PWM Kanal 6	R/W	
7	PWM Kanal 7	R/W	
8	PWM Kanal 8	R/W	
9	PWM Kanal 9	R/W	
0A	PWM Kanal 10	R/W	
0B	PWM Kanal 11	R/W	
0C	PWM Kanal 12	R/W	
0D	PWM Kanal 13	R/W	
0E	PWM Kanal 14	R/W	
0F	PWM Kanal 15	R/W	
10	PWM Kanal 16	R/W	
11	PWM Kanal 17 HB	R/W	
12	PWM Kanal 17 LB	R/W	
13	PWM Kanal 18 HB	R/W	
14	PWM Kanal 18 LB	R/W	
15	**Control_Reg**	R/W	**Steuerregister**
16	PWM_CNTR_ADDR	R/W	PW18 TWI- Busadressen Register
17	CS_Soft_PWM	R/W	Clock Select für Soft- PWM Zeitbasis
18	CS_HW_PWM_16	R/W	Frequenz für 16-Bit PWM(Kanal 17,18)
19	Res_HW_PWM_HB	R/W	Auflösung HB 16-Bit PWM (Kanal 17,18)
1A	Res_HW_PWM_LB	R/W	Auflösung LB 16-Bit PWM (Kanal 17,18)
1B	Res_Soft_PWM	R/W	Auflösung für Soft PWM
1C	PWM_vs_CALC	R/W	Interruptzeit für Tim2 Comp
1D	CRC8_Polynom	R/W	CRC8 Polynom für die Empfangenen Daten **NICHT UMGESETZT !**
BF	TWI- SRAM	R/W	Read/ Write – SRAM
	TWI- E²PROM	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. Ähnlich zu den R/W-Befehlen, nur das in Schreibrichtung keine Parameter übergeben werden können.
W- Nur Schreibbefehle, Autoinkrement Funktion ist nicht aktiv !

Die Befehle 2/2

TWI - CMD [HEX]	*Funktion*	*R/W**	*Notes*
C0	HW_ Settings Save	W	Betroffene Register siehe Seite Speicherbelegung(1/2) Hardware Settings space
C1	PWM_ Values Save	W	PWM Kanäle 1-18 auf einmal abspeichern
C2	PWM Kanal 1 Save	W	Kanal einzeln abspeichern
C3	PWM Kanal 2 Save	W	
C4	PWM Kanal 3 Save	W	
C5	PWM Kanal 4 Save	W	
C6	PWM Kanal 5 Save	W	
C7	PWM Kanal 6 Save	W	
C8	PWM Kanal 7 Save	W	
C9	PWM Kanal 8 Save	W	
CA	PWM Kanal 9 Save	W	
CB	PWM Kanal 10 Save	W	
CC	PWM Kanal 11 Save	W	
CD	PWM Kanal 12 Save	W	
CE	PWM Kanal 13 Save	W	
CF	PWM Kanal 14 Save	W	
D0	PWM Kanal 15 Save	W	
D1	PWM Kanal 16 Save	W	
D2	PWM Kanal 17 HB Save	W	
D3	PWM Kanal 17 LB Save	W	
D4	PWM Kanal 18 HB Save	W	
D5	PWM Kanal 18 LB Save	W	
D6	**Control_Reg Save**	W	
D7	PWM_CNTR_ADDR Save	W	PW18 TWI- Busadressen Register
D8	CS_Soft_PWM Save	W	
D9	CS_HW_PWM_16 Save	W	
DA	Res_HW_PWM_HB Save	W	
DB	Res_HW_PWM_LB Save	W	
DC	Res_Soft_PWM Save	W	
DD	PWM_vs_CALC Save	W	
DE	CRC8_Polynom Save	W	NICHT UMGESETZT !

R/W*- Read /Write, bei diesen Befehlen ist die Autoinkrement Funktion aktiv.
R- nur Lesebefehle, Autoinkrement Funktion ist Aktiv. Ähnlich zu den R/W- Befehlen, nur das in Schreibrichtung keine Parameter übergeben werden können.
W- Nur Schreibbefehle, Autoinkrement Funktion ist nicht aktiv !

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

Slave Receiver Modus

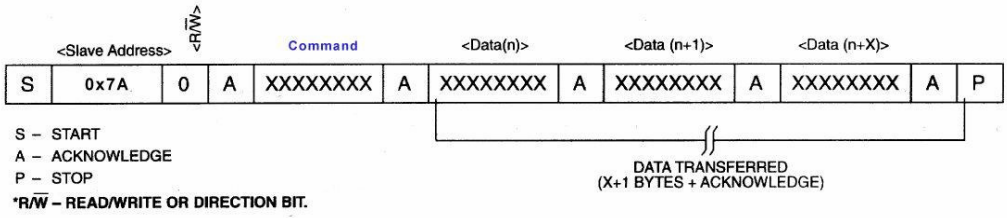
Es lassen sich je Zugriff maximal 128 Byte transportieren.
Sende = Empfangsbuffer = 128 Byte.

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

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 den nächsten Befehl hineinkopiert.

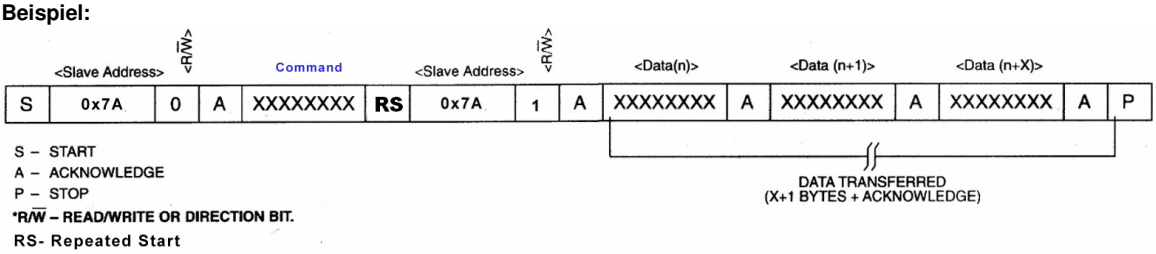
Beispiel:



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.
Um den Zustand im EEPROM zu sichern sollte CMD **0xD6** oder 0xC0 ausgeführt werden.
Am sonsten gehen die eventuell geänderten Einstellungen nach einem Neustart verloren.

Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	CRCRXEN	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 – **CRCRXEN: CRC- Slave Receiver Enabled**

not Supported

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

SREG_1 - Statusregister only

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_3 - Statusregister only

Beschreibung:
Internes Statusregister, ist für Benutzer irrelevant.
Die Veränderungen werden im Hauptregister SREG_3(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	HSWCL	PVWCL
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 – **HSWCL: Hardware Settings write successful**

Hardware Einstellungen wurden erfolgreich abgespeichert bei Logisch „1”
Nicht umgesetzt

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

PWM- Werte wurden erfolgreich abgespeichert bei Logisch „1”
Nicht umgesetzt

PWM_CNTR_ADDR

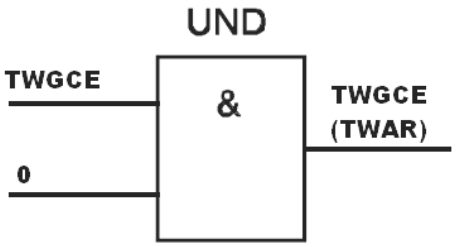
Beschreibung:
Dessen Wert ist die TWI- Busadresse.
Die Busadressen wird mit dem Befehl CMD 0x16 geschrieben. Diese Busadresse ist dann temporär bis zum Reset gültig, außer der Befehl CMD **0xD7** oder 0xC0 wurde ausgeführt. So ist dies dann die neue Busadresse auch nach Reset.

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 E²PROM’s wiederhergestellt werden, wobei alle andere Hardware Settings flöten gehen :-p

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	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	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
```

Beschreibung:
Von diesen ClockSel- Bits hängt die Geschwindigkeit der PWM- Outputs OC1A, 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	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/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
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

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.

Input Capture Register 1 – ICR1H and ICR1L

Bit	7	6	5	4	3	2	1	0
	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_SRAM

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“.

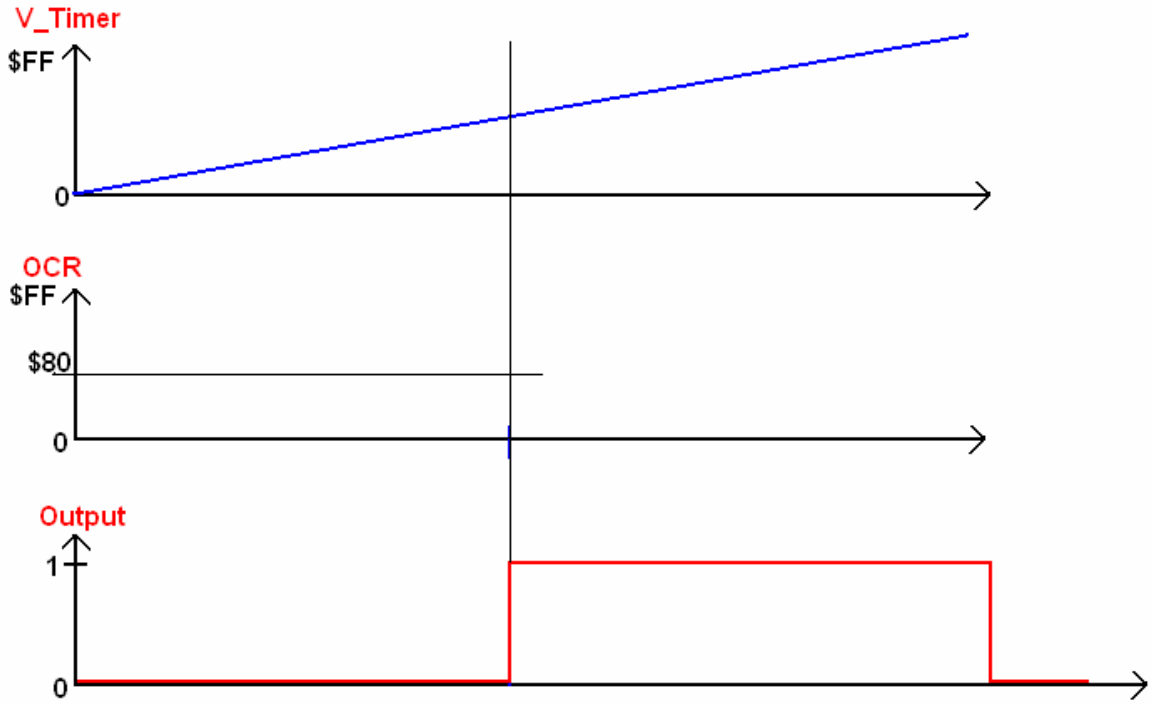
Res_Soft_PWM

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←Res_Soft_PWM_SRAM
Res_Soft_PWM_SRAM= 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_CALC

Beschreibung:
Eine Empfehlung des Hauses: dieses Register einfach in Ruhe lassen☺
Default Wert 0xFF.
Wird dessen Wert reduziert so beschleunigt sich die Soft- PWM um ein paar 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_CALC“ darf keinesfalls geringer als die längste Interruptzeit(siehe. „TIM2_COMP- Interruptzeiten“) sein.

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

Output Compare Register – OCR2

OCR2←PWM_vs_CALC_SRAM

Bit	7	6	5	4	3	2	1	0
	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- Interruptzeiten.
Im Interrupt werden maximal bis zu 155xClk Cycles verbrannt, ???
und minimal 43. ??????
Die ClockCycles müssen neu ermittelt werden

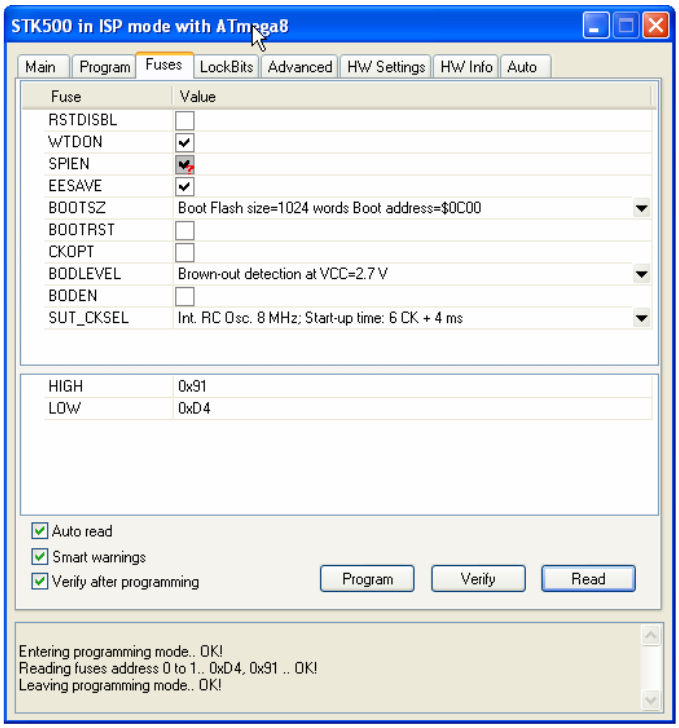
Standart Einstellungen nach dem Flashen

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: $FF
6) „Power Up Load PWM- Values“- Funktion abgeschaltet, somit werden die PWM-Kanäle by default mit Nullen „0“ geladen.
7)

Nicht vollständig

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 con Kanal 11).
10) **Stoppkondition**

u.s.w

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

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 E²PROM - Befehl „(0xBE)"

Mit diesen Befehl können Daten IN und AUS dem zugesicherten E²PROM- Bereich des µControllers geschrieben bzw. gelesen werden.

EEPROM- Adressen
TWI E²PROM Bottom: **0x0000**
TWI E²PROM Top: **0x017F**

Schreib und Lesezugriffe geschehen auf der gleichen Art und Weise
Wie beim Befehl TWI SRAM (0xBF).

Bekannte BUGS

Noch keine :-p

Notizen

BEZEICHNUNG
PWM18 V1.0
PWM– Steht für PWM- Firmware
18- Für Anzahl der PWM- Kanäle
V1.0- Firmware Version

Change Log

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.