SOFTWARE PWM CONTROLLER ATMEGA 8
TYP: PW18 V1.0

INHALTSVERZEICHNIS

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

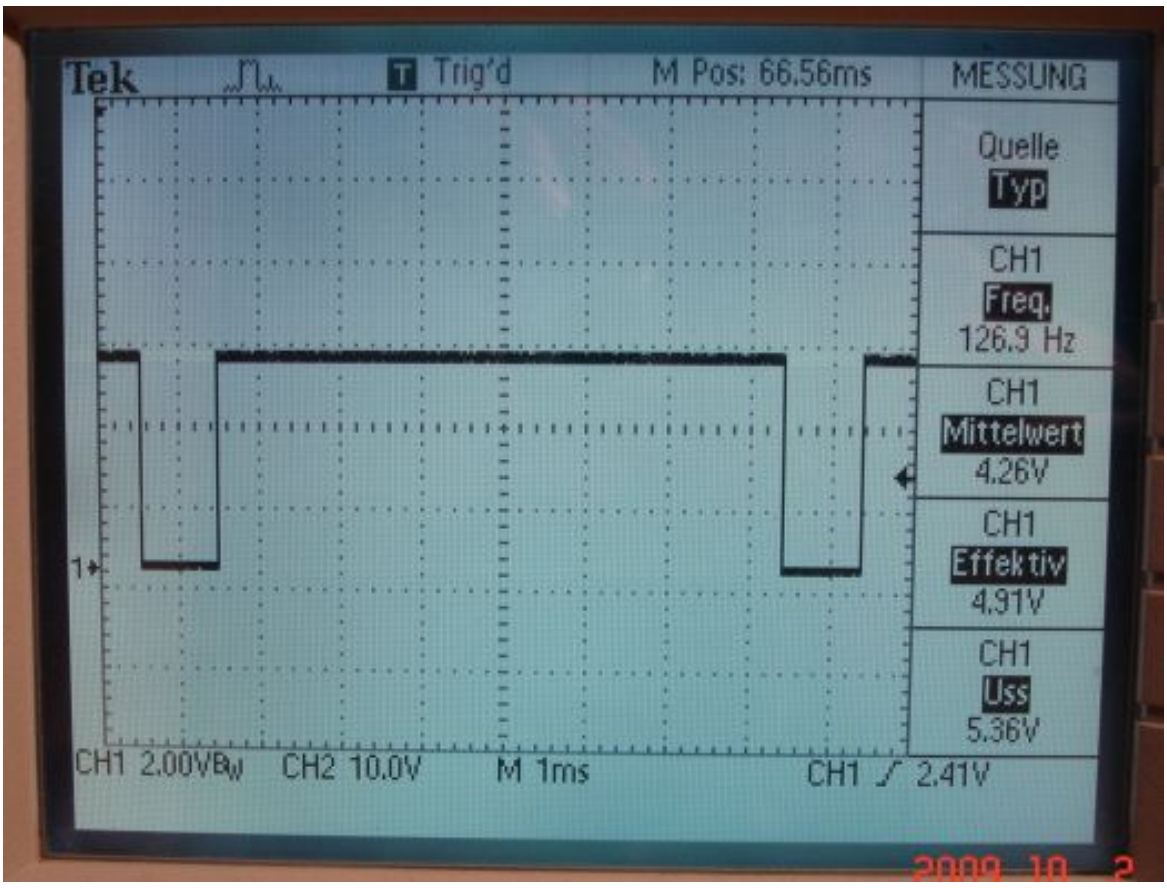
- **512 Byte TWI- SRAM noch nicht umgesetzt !!!!!!!!!!!!**
- **256 Byte TWI- E²PROM noch nicht umgesetzt !!!!!!!!!!!!**
- **CRC 8,16?? Serial or Table Realisierung noch nicht umgesetzt !!!!!!!!!!!!**
- ***Über TWI- Bus zugängliche Hardware Einstellungen***

– 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***

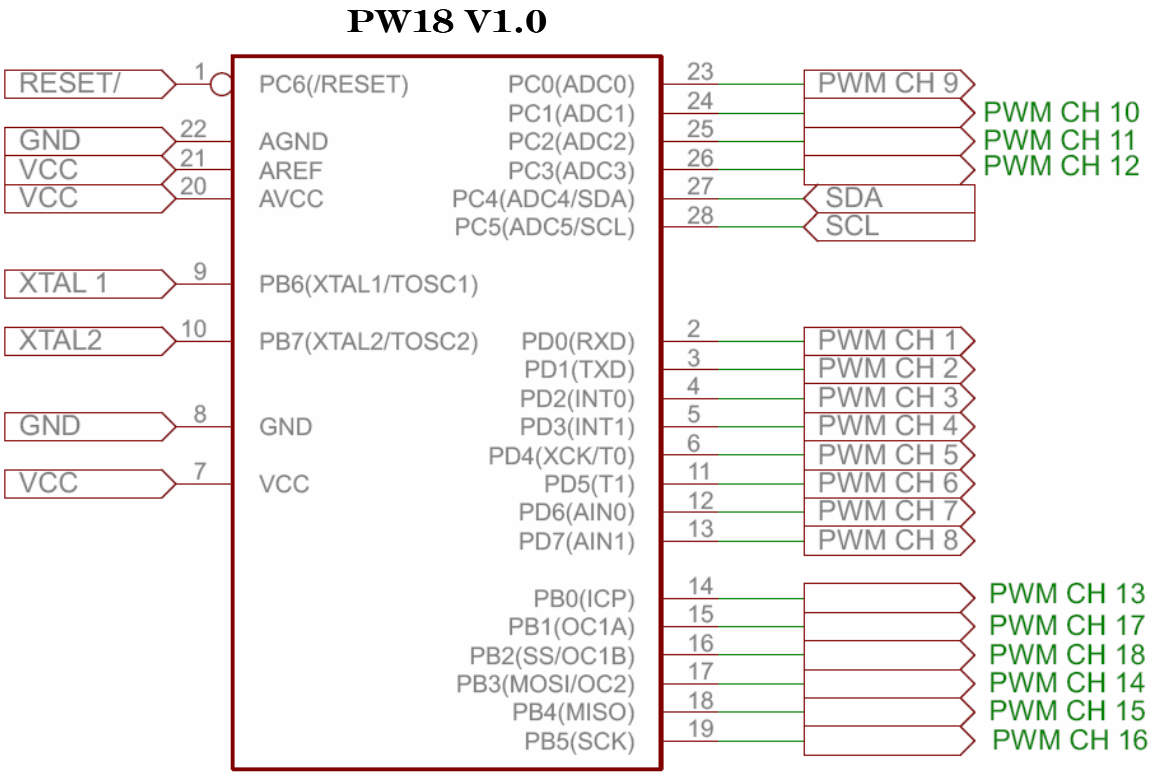
über einen Befehl abspeichern.
– „Power Up Load“ gespeicherte PWM – Werte nach Reset laden(EEPROM), oder by default alle Kanäle mit 0 initialisieren.

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.
Bei Crystal- Anwendung nicht vergessen die Fuses neu zu Füttern.



Speicherbelegung 1/2

Byte	*Addresse EEPROM*	*EEPROM- Speicherwerte*	*SRAM- Kopie*	*Addresse SRAM*	*TWI - CMD*	*SRAM- Unterteilung*	
—	—	—	—	—	**0**	—	
1	$000	OCR_Ch_01_EEP	OCR_Ch_01_SRAM	$60	**1**	**Software PWM Channels (GPR)**	**PWM-Values Space**
2	$001	OCR_Ch_02_EEP	OCR_Ch_02_SRAM	$61	**2**		
3	$002	OCR_Ch_03_EEP	OCR_Ch_03_SRAM	$62	**3**		
4	$003	OCR_Ch_04_EEP	OCR_Ch_04_SRAM	$63	**4**		
5	$004	OCR_Ch_05_EEP	OCR_Ch_05_SRAM	$64	**5**		
6	$005	OCR_Ch_06_EEP	OCR_Ch_06_SRAM	$65	**6**		
7	$006	OCR_Ch_07_EEP	OCR_Ch_07_SRAM	$66	**7**		
8	$007	OCR_Ch_08_EEP	OCR_Ch_08_SRAM	$67	**8**		
9	$008	OCR_Ch_09_EEP	OCR_Ch_09_SRAM	$68	**9**		
10	$009	OCR_Ch_10_EEP	OCR_Ch_10_SRAM	$69	**0A**		
11	$00A	OCR_Ch_11_EEP	OCR_Ch_11_SRAM	$6A	**0B**		
12	$00B	OCR_Ch_12_EEP	OCR_Ch_12_SRAM	$6B	**0C**		
13	$00C	OCR_Ch_13_EEP	OCR_Ch_13_SRAM	$6C	**0D**		
14	$00D	OCR_Ch_14_EEP	OCR_Ch_14_SRAM	$6D	**0E**		
15	$00E	OCR_Ch_15_EEP	OCR_Ch_15_SRAM	$6E	**0F**		
16	$00F	OCR_Ch_16_EEP	OCR_Ch_16_SRAM	$6F	**10**		
17	$010	OCR_Ch_OC1AH_EEP	OCR_Ch_OC1AH_SRAM	$070	**11**	**Hardware PWM Channels**	
18	$011	OCR_Ch_OC1AL_EEP	OCR_Ch_OC1AL_SRAM	$071	**12**		
19	$012	OCR_Ch_OC1BH_EEP	OCR_Ch_OC1BH_SRAM	$072	**13**		
20	$013	OCR_Ch_OC1BL_EEP	OCR_Ch_OC1BL_SRAM	$073	**14**		
25	**$014**	**Control_Reg_EEP**	**Control_Reg_SRAM**	**$074**	**15**	**Hardware Settings space**	
26	$015	PWM_CNTR_ADDR_EEP	PWM_CNTR_ADDR_SRAM	$075	**16**		
27	$016	CS_Soft_PWM_EEP	CS_Soft_PWM_SRAM	$076	**17**		
28	$017	CS_HW_PWM_16_EEP	CS_HW_PWM_16_SRAM	$077	**18**		
29	$018	Res_HW_PWM_HB_EEP	Res_HW_PWM_HB_SRAM	$078	**19**		
30	$019	Res_HW_PWM_LB_EEP	Res_HW_PWM_LB_SRAM	$079	**1A**		
31	$01A	Res_Soft_PWM_EEP	Res_Soft_PWM_SRAM	$7A	**1B**		
32	$01B	PWM_vs_CALC_EEP	PWM_vs_CALC_SRAM	$07B	**1C**		
	—	—	HW_ Settings Save	—	**C0**		
	—	—	PWM_ Values Save	—	**C1**		

Speicherbelegung 2/2

Byte	***Addresse* *EEPROM***	***EEPROM- Speicherwerte***	***SRAM-***	***Addresse SRAM***	***TWI - CMD***	***SRAM- Unterteilung***
—	—	—	—	—	—	—

AVR- Speicherreservierungen

SRAM		**Adresse [^16]**	**Größe[Byte]**
Reserviert für RX Befehle	Bottom	SRAM Start	128
	TOP	E0	
TWI Receive Buffer	Bottom	E1	128
	TOP	Bla Bla	
TWI Transmitt Buffer	Bottom	Bla Bla	128
	TOP	Bla Bla	
Daten Speicher	Bottom	Bla Bla	512
	TOP	Bla Bla	
Stack	Bottom	RAMEND- 128	**128**
	TOP	**RAMEND**	

Die Befehle 1/3

TWI - CMD	*Funktion*	*Notes*
0	—	none
1	PWM Kanal 1	
2	PWM Kanal 2	
3	PWM Kanal 3	
4	PWM Kanal 4	
5	PWM Kanal 5	
6	PWM Kanal 6	
7	PWM Kanal 7	
8	PWM Kanal 8	
9	PWM Kanal 9	
0A	PWM Kanal 10	
0B	PWM Kanal 11	
0C	PWM Kanal 12	
0D	PWM Kanal 13	
0E	PWM Kanal 14	
0F	PWM Kanal 15	
10	PWM Kanal 16	
11	PWM Kanal 17 HB	
12	PWM Kanal 17 LB	
13	PWM Kanal 18 HB	
14	PWM Kanal 18 LB	
15	**Control_Reg**	
16	PWM_CNTR_ADDR	PW18 TWI- Busadressen Register
17	CS_Soft_PWM	
18	CS_HW_PWM_16	
19	Res_HW_PWM_HB	
1A	Res_HW_PWM_LB	
1B	Res_Soft_PWM	
1C	PWM_vs_CALC	
C0	HW_ Settings Save	Speicherbefehl für Hardwaresettings
C1	PWM_ Values Save	Speicherbefehl für PWM- Werte

Die Befehle 2/3

Die zwei Befehlstypen:
Typ1: Konfigurationsbefehle(Konfigurationsbereich)
Typ2: Speicherbefehle (Speicherbereich)

Die Konfigurationsbefehle
Für alle Befehle 0x00 bis 0x80 ist Autoinkrement aktiv. Ein Befehl aus diesem Bereich ist in der Lage ein Byte an Informationen aufzunehmen.
Beispiel: CMD 0x01+1Byte, CMD 0x02+1Byte ….CMD $BE+1Byte

Werden im Schreibmodus mehrere Bytes geschrieben so wandert jedes nächste Datenbyte in den darauf folgenden Befehl.
Beispiel: Empfangen: SLA+W, CMD(1), Byte2, Byte3, Byte4…..
CMD(1) setzt den Datenpointer auf PWM CH 1.
Das zweite Byte2 wird ins PWM CH 1(CMD1) geschrieben, Byte3 - PWM CH 2(CMD2), Byte4 (CMD3) ins PWM CH 3 u.s.w.

Die Speicherbefehle
coming soon……

PWM- Kanäle Werte abspeichern CMD 0xC0

Welche Register zu den **PWM- Kanäle Werten** gehören, kann man anhand der Tabelle siehe Seite Speicherbelegung /SRAM-Unterteilung/ „PWM- Values Space“ erkennen.
Wie über TWI- Bus geschrieben wird, kann man auf Seite „Das TWI- Protocoll“ **„1) Schreiben“** sehen. Doch was geschrieben werden soll wird hier erklärt.

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 abgespeichert.
ASCII: S = HEX: 53 = DEC: 83
Wird als **Byte2** etwas abweichendes von „S“ geschrieben, so findet der Speichervorgang nicht statt.

Bekannter Fehler: Werte abgespeichert, doch nach Reset immer noch Nuller. Dies passiert wenn im Control Register das Flag „PWUPL“ nicht gesetzt wurde. Hat man das Flag(PWUPL) gesetzt “1“ so sollten die Hardwaresettings daraufhin gespeichert werden.
Da sonst nach neustart der ursprüngliche „PWUPL“=0 geladen wird.

Hardware Settings abspeichern CMD 0xC1

Welche Register zu den „Hardware Settings“ gehören, kann man anhand der Tabelle siehe Seite Speicherbelegung /SRAM-Unterteilung/ „Hardware Settings space“ erkennen.
Wie über TWI- Bus geschrieben wird, kann man auf Seite „Das TWI- Protocoll“ **„1) Schreiben“** sehen. Doch was geschrieben werden soll wird hier erklärt.

6) Start
7) SLA+W
8) **Byte 1: 0xC0** (CMD)
9) **Byte 2: ‚S’** (Großbuchstabe).
10) **Stoppkondition**

Wird als Byte 2 etwas ungleich „S“ geschrieben so findet kein Speichervorgang Statt.
ASCII: S = HEX: 53 = DEC: 83

Slave Receiver Modus

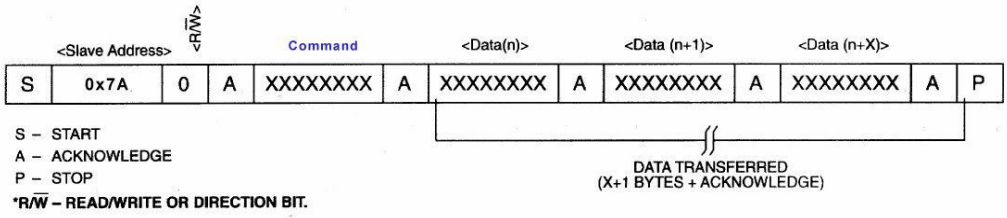
Es lassen sich je Zugriff maximal 128 Byte transportieren.
Sende = Empfangsbuffer = 128 Byte.

1) Schreiben in den Konfigurationsbereich (Befehle 0x00 bis 0x80)
2) Lesen aus dem Konfigurationsbereich (Befehle 0x00 bis 0x80)

1) Schreiben

Ein einfacher Schreibbefehl besteht aus **mindestens 2 Byte**.
Byte1: Ist das Befehlsbyte siehe Speicherbelegung „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.

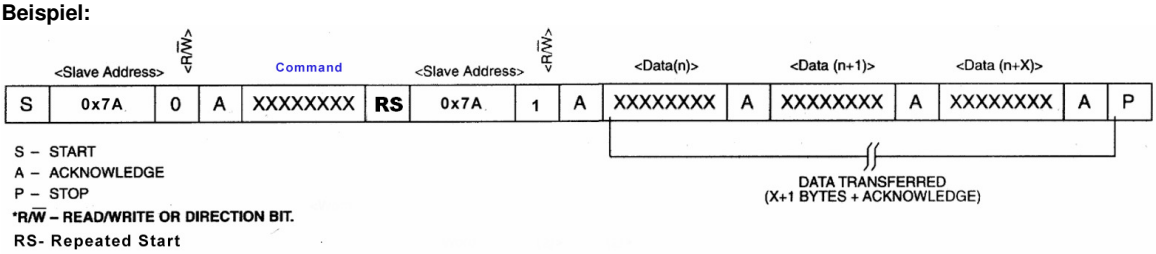
Beispiel:



2) Lesen

Ein Lesebefehl besteht aus **nur 1 Byte**.
Byte1(CMD): Ist das Befehlsbyte siehe Speicherbelegung „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 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, 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

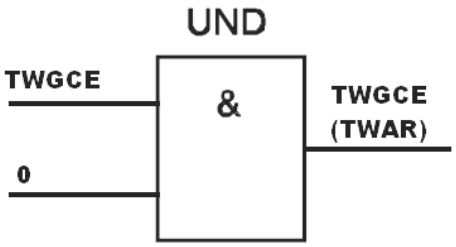
Beschreibung:
Dessen Wert ist die TWI- Busadresse.
Die Busadressen wird mit dem Befehl CMD $16 geschrieben. Diese Busadresse ist dann temporär bis zum Reset gültig, außer der Befehl CMD 0xC0 wurde ausgeführt. So ist dies dann die neue Busadresse auch nach Reset.

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

CS_Soft_PWM

Beschreibung:
Software PWM- Clock Select Bytes. Es wird empfohlen den Wert by default zu lassen.
Standardmäßig werden die CS- Bits mit 1 geladen, weil bei höheren Werten die SoftPWM 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	CS02	CS01	CS00
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 –

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.

```
; 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 CS- Bits hängt die Geschwindigkeit der PWM- Outputs OC1A, OC1B ab.

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

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

Comprende.
Somit wurde die neue Auflösung übernommen und temporär gespeichert.
Will man die gesetzte Auflösung auch nach Reset nicht missen, so sollte anschließend der Befehl 0xC0 ausgeführt werden. Mehr Details auf Seite „Die Befehle“
(Neue Hardware Settings abspeichern).

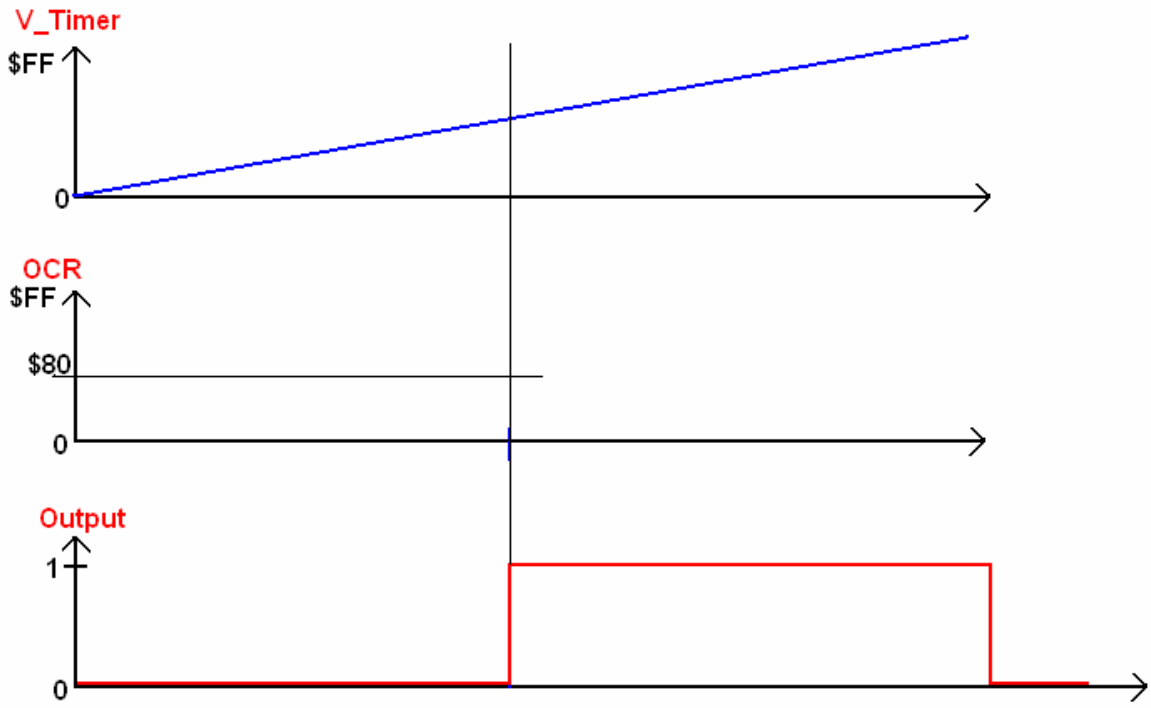
Res_Soft_PWM

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.

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).

OCR2←PWM_vs_CALC_SRAM

Output Compare Register – OCR2

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

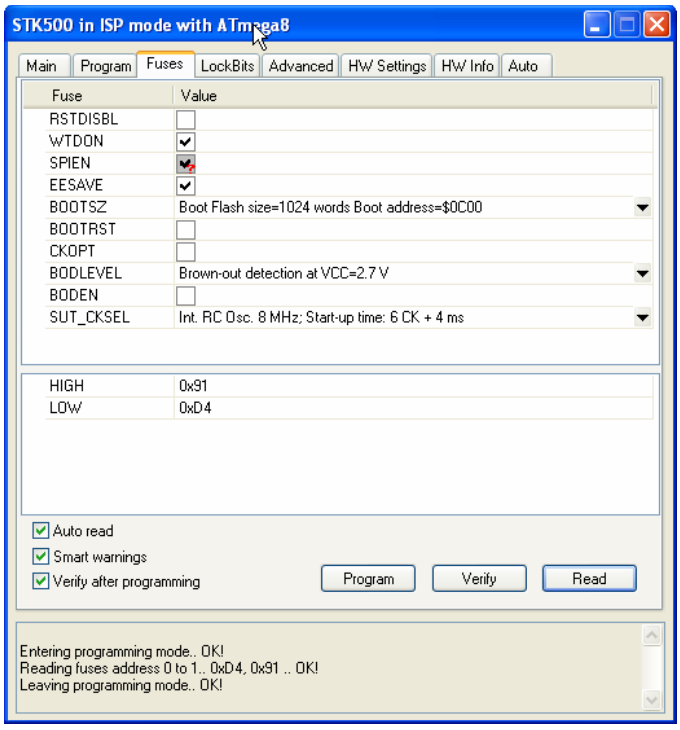
Standart Einstellungen nach dem Flashen

1) TWI- Busadresse 0x7A
2) Software PWM 8-Bit Auflösung
3) Hardware PWM 16- Bit Auflösung
4) ClockSelect von Soft-(Timer2) und Hard-(Timer1) PWM 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.



Bekannte BUGS

Noch keine :-p

Notizen

BEZEICHNUNG
PW18 V1.0
PW – Steht für PWM Firmware
18- Für Anzahl der PWM Kanäle
V1.0- Firmware Version

BLA BLA BLA(temporär Copy)

Stimmt es?
Für die Befehle 0x81 bis 0xFE gibt es ebenso eine Art Autoincrement Funktion, jedoch innerhalb eines Befehls.
0xBF ist für die Datenspeicherung im SRAM zuständig.
Beispiel: SLA+W, CMD($BF), High_Byte(Adr), Low_Byte(Adr),, Data1, Data2, Data3…..
Mit High, Low_Byte wird der Datenpointer gesetzt, und alles ab Data1 wird mit Autoinkrement in den SRAM geschrieben.

Change Log

PW18 v1.0
Die erste veröffentlichte Version.

SRAM		Adresse [^16]	Größe[Byte]
Reserviert für RX Befehle	Bottom	SRAM Start	128
	TOP	E0	
TWI Receive Buffer	Bottom	E1	128
	TOP	Bla Bla	
TWI Transmitt Buffer	Bottom	Bla Bla	128
	TOP	Bla Bla	
Daten Speicher	Bottom	Bla Bla	512
	TOP	Bla Bla	
Stack	Bottom	RAMEND- 128	**128**
	TOP	**RAMEND**	