Mega2650 Timers

Henning Hargaard – 2.november 2016

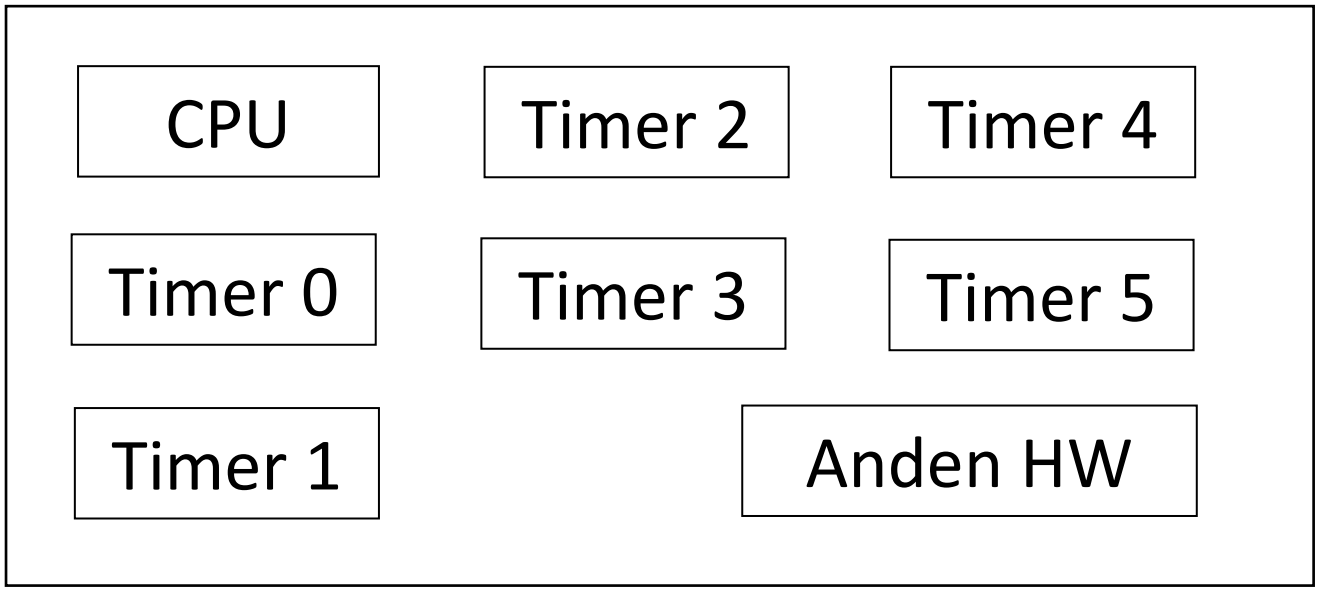
1. Formål

Næsten alle mikrocontrollers har en eller flere timere indbygget.
Timere har mange anvendelsesområder og kan for eksempel anvendes til at:

- Lave en præcis tidsforsinkelse.
- Sørge for, at vi får et interrupt (en programafbrydelse) med faste tidsintervaller.
- Tælle pulser (eksempelvis fra følere på et transportbånd).
- På et portben generere et firkantsignal, som har en given frekvens og 50% dutycycle (CTC mode).
- På et portben generere et firkantsignal, som har en given frekvens og variabel dutycycle (PWM mode).

Det er vigtigt at gøre sig klart, at en timer er en selvstændig hardware på chippen, og fungerer derfor uafhængigt af for eksempel CPU'en. Det betyder, at CPU'en vil kunne stå og lave beregninger samtidig med, at timerne arbejder med en anden opgave (som nævnt i listen med eksempler herover).

Tænk på microcontrolleren som en fabrikshal med maskiner, hvor CPU'en er en "maskine" og timerne hver især er andre "maskiner", der hver især samtidigt kan lave et stykke arbejde:



2. Generelt

Mega2560 har 6 timere, som kaldes **Timer 0**, **Timer 1**, **Timer 2**, **Timer 3**, **Timer 4** og **Timer 5**.

En timers helt centrale del er dens **tælleregister**, som for de 5 timere kaldes henholdsvis **TCNT0**, **TCNT1**, **TCNT2**, **TCNT3**, **TCNT4** og **TCNT5**. Man kan få adgang til tælleregistrene via I/O-registre, og man kan fra programmet både aflæse og skrive til tælleregistrene.

For Timer 0 og Timer 2 gælder, at deres tælleregister er **8 bit**, og kan altså maksimalt indeholde tallet 255. Vi kalder disse timere for 8 bit timere.

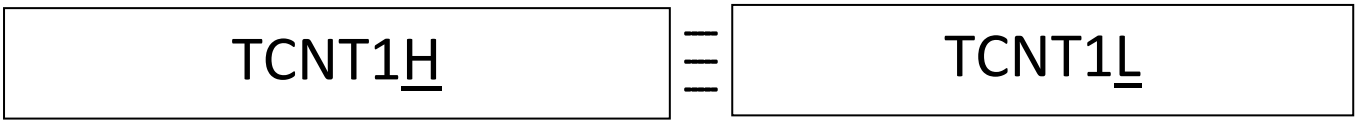
Mega2650 Timers

Henning Hargaard – 2.november 2016

For de øvrige timere (Timer 1, Timer 3, Timer 4 og Timer 5) gælder, at deres tælleregister er **16 bit**, og kan altså maksimalt indeholde tallet 65535.
Vi kalder disse timere for 16 bit timere.

Da Mega2560 er en 8 bit mikrocontroller, er I/O-registrene som bekendt 8 bit registre.
For ovennævnte 16 bit tælleregistre gælder derfor, at hvert 16 bit register fysisk er sammensat af to 8 bit registre.

For eksempel består registeret TCNT1 af de to "halvdele" TCNT1H og TCNT1L:



Hvis vi skriver assembly-kode og ønsker af aflæse eller skrive til et 16 bit register, må vi læse fra / skrive til hver "halvdel" for sig i en bestemt rækkefølge:

- **Rækkefølge ved skrivning: TCNT1H, derefter TCNT1L.**
- **Rækkefølge ved læsning: TCNT1L, derefter TCNT1H.**

Når vi skriver programmet i C, tager compileren (heldigvis) automatisk hånd om ovenstående problem med opsplitning i 16 bit registerets to halvdele, sådan, at vi blot kan skrive for eksempel:

```
TCNT1 = 12345;
unsigned int x = TCNT3;
```

Virkemåden af en timer er grundlæggende, at den "tæller" (typisk en opad) for hver periode af et tilført clocksignal.

Clocksignalet dannes ud fra en neddeling af CPU clockfrekvensen.
Denne neddeling sker ved hjælp af en hardwareblok, som kaldes en "prescaler".

Hver timer har sin egen prescaler, der kan neddele CPU clockfrekvensen med et heltal (kaldes N).

f_{cpu} → **Prescaler** (divide by **N**) → $f_{timerclock}$

Altså: $f_{timerclock} = f_{cpu} / N$

Mega2650 Timers

Henning Hargaard – 2.november 2016

Alle timere (undtagen Timer 2) har tilknyttet et ben, der alternativt kan anvendes som clocksignal for timeren.
Dette er smart, hvis man f.eks. ønsker at tælle pulser, der kunne komme fra en føler.
Disse tælleben kaldes for de enkelte timere **T0**, **T1**, **T3**, **T4** og **T5**, og de er fysisk placeret på følgende portben:

- **Timer 0 :**
  T0 = Port D, ben 7
- **Timer 1 :**
  T1 = Port D, ben 6
- **Timer 3 :**
  T3 = Port E, ben 6
- **Timer 4 :**
  T4 = Port H, ben 7
- **Timer 5 :**
  T5 = Port L, ben 2

←

På "vores hardware":

T0 er lav (0), når vi trykker på SW0.

T0 er høj, når vi ikke trykker på SW0.

Prescalerværdien N (eller alternativt om man vil anvende timerens eksterne tællerben som clock) bestemmer programmøren via specielle I/O register bits. Der er også mulighed for at vælge "no clock", der bevirker, at timeren vil "stå stille".

Valg af clock/prescaler for Timer 0:

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/(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 T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge

Mega2650 Timers

Henning Hargaard – 2.november 2016

Valg af clock/prescaler for Timer 1, Timer 3, Timer 4 og Timer 5:

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CSn2	CSn1	CSn0	TCCRnB
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

- **TCCRnB = TCCR1B, TCCR3B, TCCR4B eller TCCR5B.**

CSn2	CSn1	CSn0	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 Tn pin. Clock on falling edge
1	1	1	External clock source on Tn pin. Clock on rising edge

Valg af clock/prescaler for Timer 2:

7	6	5	4	3	2	1	0	
FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

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)

Mega2650 Timers

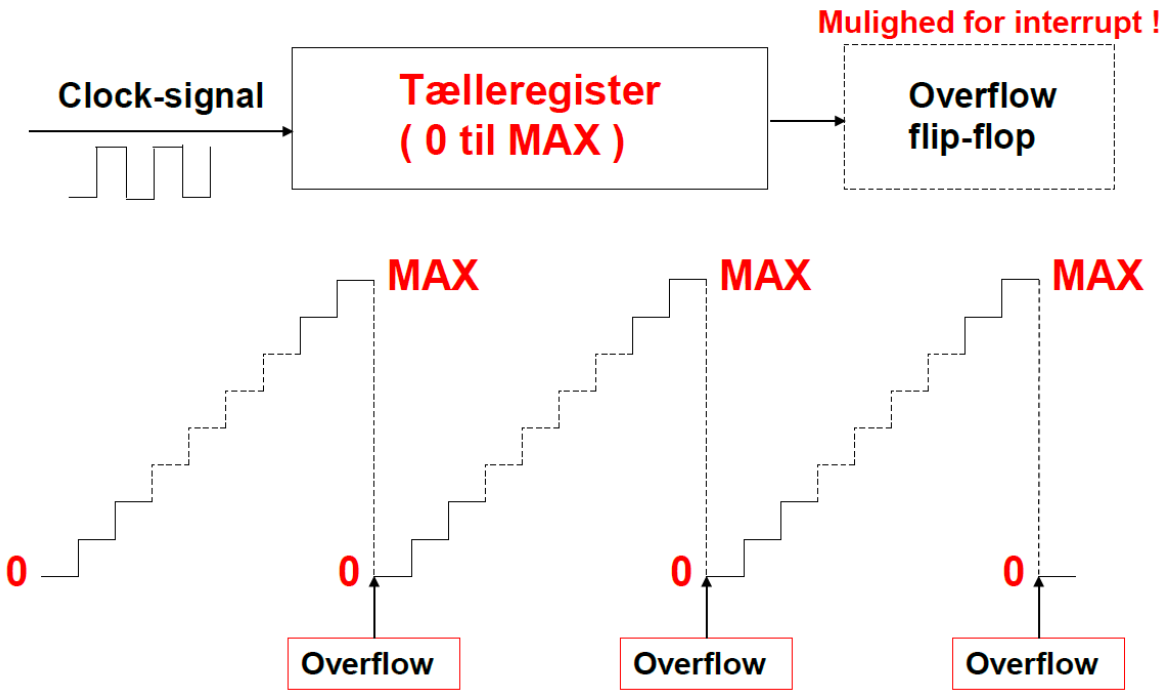
Henning Hargaard – 2.november 2016

3. Normal mode

”Normal mode” anvendes, hvis vi ønsker at:

- Lave en præcis tidsforsinkelse.
- Sørge for, at vi får et interrupt (en programafbrydelse) med faste tidsintervaller.

Når en timer er i ”normal mode” vil den ganske simpelt tælle en opad for hver periode af det tilførte clocksignal.
Når den maksimale værdi (255 for en 8 bit timer og 65535 for en 16 bit timer), vil tælleregisteret starte forfra fra 0 ved næste periode af clocksignalet:



Husk, at timerens clocksignal er: $f_{timerclock} = f_{cpu} / N.$

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 0 i ”normal mode”:

7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP
0	0	0	0	Normal	0xFF

Sådan sættes Timer 1, Timer 3, Timer 4 eller Timer 5 i ”normal mode”:

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCRnA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCRnB
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

- **TCCRnA = TCCR1A, TCCR3A, TCCR4A eller TCCR5A.**
- **TCCRnB = TCCR1B, TCCR3B, TCCR4B eller TCCR5B.**

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP
0	0	0	0	0	Normal	0xFFFF

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 2 i ”normal mode”:

7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP
0	0	0	0	Normal	0xFF

Hver gang timeren genstarter fra 0, har man et ”timer overflow”, hvorved en speciel bit i et I/O register samtidigt bliver sat (til et 1-tal). Man siger, at timerens overflow-flag bliver sat.
Hvis man i sin software vil vente på, at der sker et timer overflow, kan man i en sløjfe vente på, at dette sker (det kaldes polling af flaget).

Overflow-flaget kan kun nulstilles (resettes til 0) fra software (timeren kan kun sætte det).
Lidt specielt sker nulstilling af flaget ved at skrive et 1-tal til det !

Som det også er indikeret på figuren, er der mulighed for at få et interrupt, når overflowet sker (men kun, hvis programmøren har aktiveret interruptsystemet).
Ved at anvende interrupts undgår vi polling af flaget, der er en ineffektiv metode (CPU’en kan ikke lave andet end at vente på, at timeren laver overflow).

De fysiske placeringer af overflow-flaget er for de enkelte timere ses her:

Timer 0 overflow flag:

7	6	5	4	3	2	1	0	
–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
R	R	R	R	R	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mega2650 Timers

Henning Hargaard – 2.november 2016

Timer 1, Timer 3, Timer 4 og Timer 5 overflow flag:

7	6	5	4	3	2	1	0	
–	–	ICF1	–	OCF1C	OCF1B	OCF1A	TOV1	TIFRn
R	R	R/W	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Timer 2 overflow flag:

7	6	5	4	3	2	1	0	
–	–	–	–	–	OCF2B	OCF2A	TOV2	TIFR2
R	R	R	R	R	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Timerne kan anvendes til at lave en præcis tidsforsinkelse.
Hvis vi anvender polling, kan følgende metode bruges:

1. **Skriv en beregnet værdi x til tælleregisteret (TCNTn).**
2. **Giv timeren clocksignal (skriv til prescaler-bits).**
3. **Vent på, at overflow flaget bliver 1.**
4. **Fjern timerens clocksignal (vælg "no clock").**
5. **Nulstil overflow flaget (ved at skrive 1 til det).**

Tidsforsinkelsens størrelse bestemmes af punkt 1 og 2:

Delay = (("Max for timeren + 1" (256 eller 65536) - x) * Prescaler) / "CPU frekvens" [sekunder].

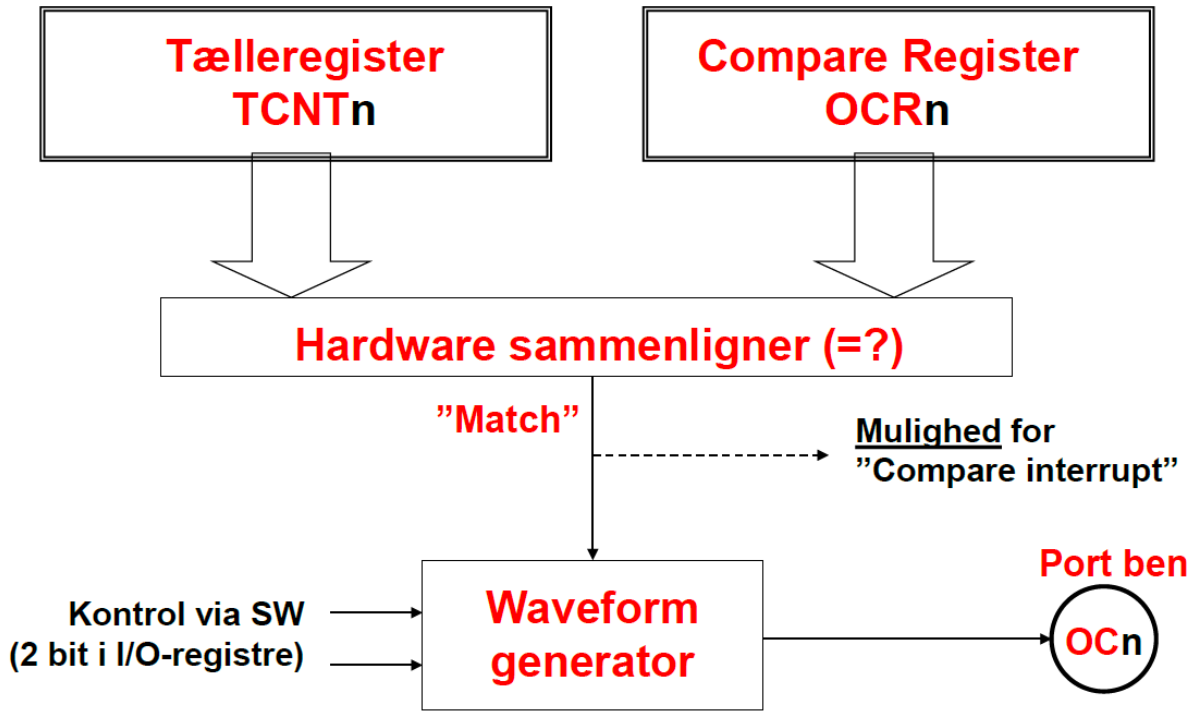
Herunder ses et eksempel på anvendelse af Timer 1 til at lave et delay på 3 sekunder:

```
void T1Delay()
{
  // 16000000 Hz /1024 = 15625 Hz
  // Vi har altså 15625 "trin" per sekund
  // - og ønsker 3 sekunder til overflow (= 1/3 Hz)
  TCNT1 = 65536-(3*15625);
  // Timer 1 i Normal Mode og PS = 1024
  TCCR1A = 0b00000000;
  TCCR1B = 0b00000101;
  // Afvent Timer 1 overflow flag
  while ((TIFR1 & (1<<0)) == 0)
  {}
  // Stop Timer 1 (ingen clock)
  TCCR1B = 0b00000000;
  // Nulstil Timer 1 overflow flag
  TIFR1 = 1<<0;
}
```

4. Output compare unit + waveform generator

Hver timer har tilknyttet noget ekstra hardware, som kaldes ”Output Compare Unit” og ”Waveform generator”.
Vi benytter os af denne hardware, når timeren anvendes i enten CTC eller i PWM mode (herom senere).

Denne figur viser opbygningen af kredsløbet:



Det fungerer således, at tælleregisterets indhold altid sammenlignes med indholdet af et register, som vi kalder ”Output Compare Registeret” (OCR). Compare registeret og tælleregisteret har samme størrelse (8 eller 16 bit), og normalt skriver vi fra programmet en fast værdi til OCR.

Bemærk, at sammenligningskredsløbet er lavet i hardware, og derfor fungerer uafhængigt af CPU’en.

Hvis tælleregister og OCR indeholder samme tal, har vi en situation, som vi kalder ”match”, og der sendes et signal
til et andet specielt stykke hardware, som vi kalder ”Waveform genaratoren”.

Waveform generatoren er i stand til at styre et bestemt digitalt portben, som vi på figuren kalder ”OCn”. Bemærk forskellen på ”OCRn” (som er et register) og ”OCn” (som er et portben).

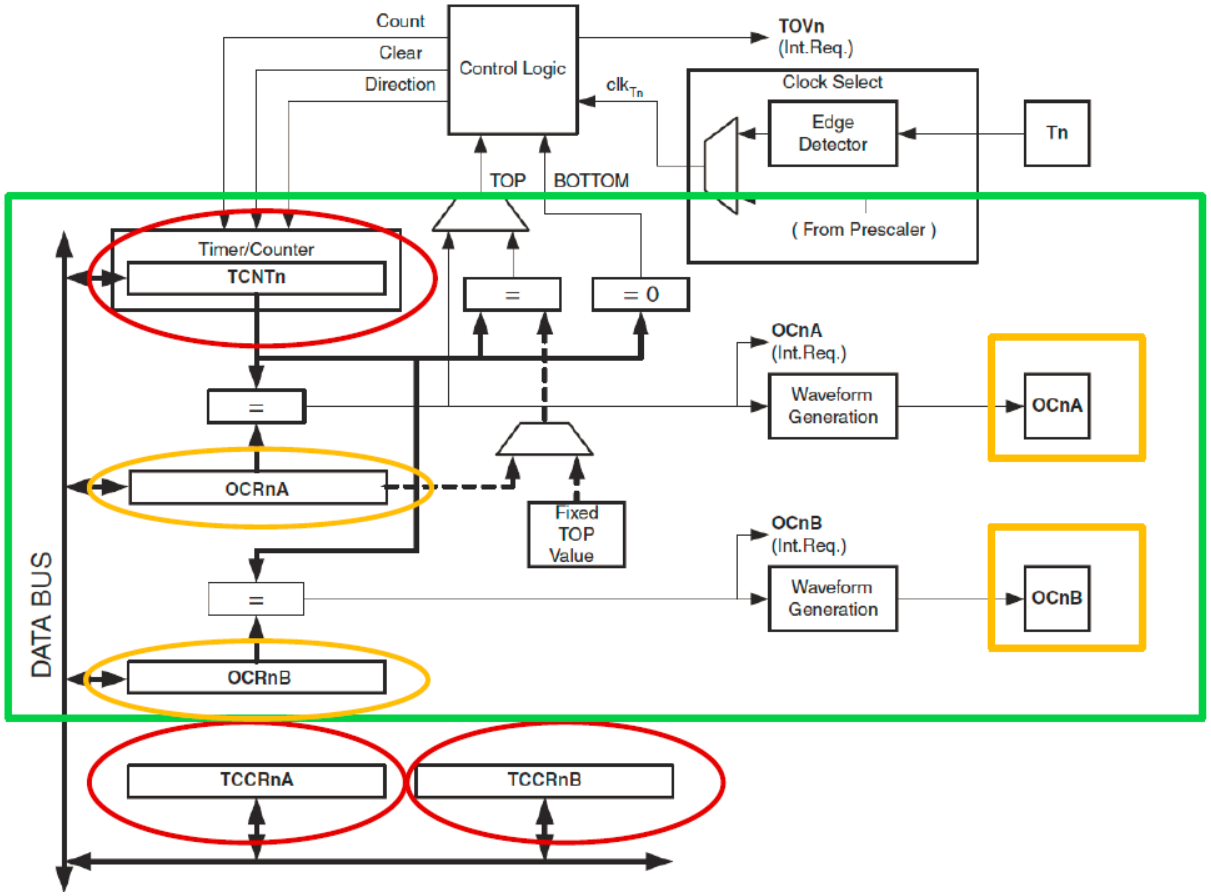
Waveform generatoren styres via 2 bits i et I/O-register. De 2 bit vælger 1 ud af 4 måder, hvorpå waveform generatoren skal manipulere portbenet, hvis der optræder et ”match”. En af de 4 måder vil altid være ”No action”, som i øvrigt er default efter RESET.

Mega2650 Timers

Henning Hargaard – 2.november 2016

Hvis vi kigger på de enkelte timeres blokdiagrammer, kan vi mere detaljeret se, hvordan mekanismen er realiseret:

Her er blokdiagrammet for Timer 0:



Det, som er indrammet med grønt, er interessant i denne sammenhæng.

Vi ser, at TCNT0 bliver sammenlignet med OCR0A, der ved match sender signal til en waveform generator, som igen styrer et portben, der hedder OC0A. Dette kalder vi "**system A**".

Vi ser også, at TCNT0 samtidigt bliver sammenlignet med OCR0B, der ved match sender signal til en (anden) waveform generator, som igen styrer et portben, der hedder OC0B. Dette kalder vi "**system B**".

Timer 0 har altså 2 systemer, der hver især vil kunne sammenligne med tælleregisteret og styre hvert deres portben (måske på 2 forskellige måder). Vi er faktisk på grund af dette i stand til f.eks. med Timer 0 samtidigt at kunne generere 2 PWM signaler med forskellige duty cycles.

OC0A = PB, ben 7

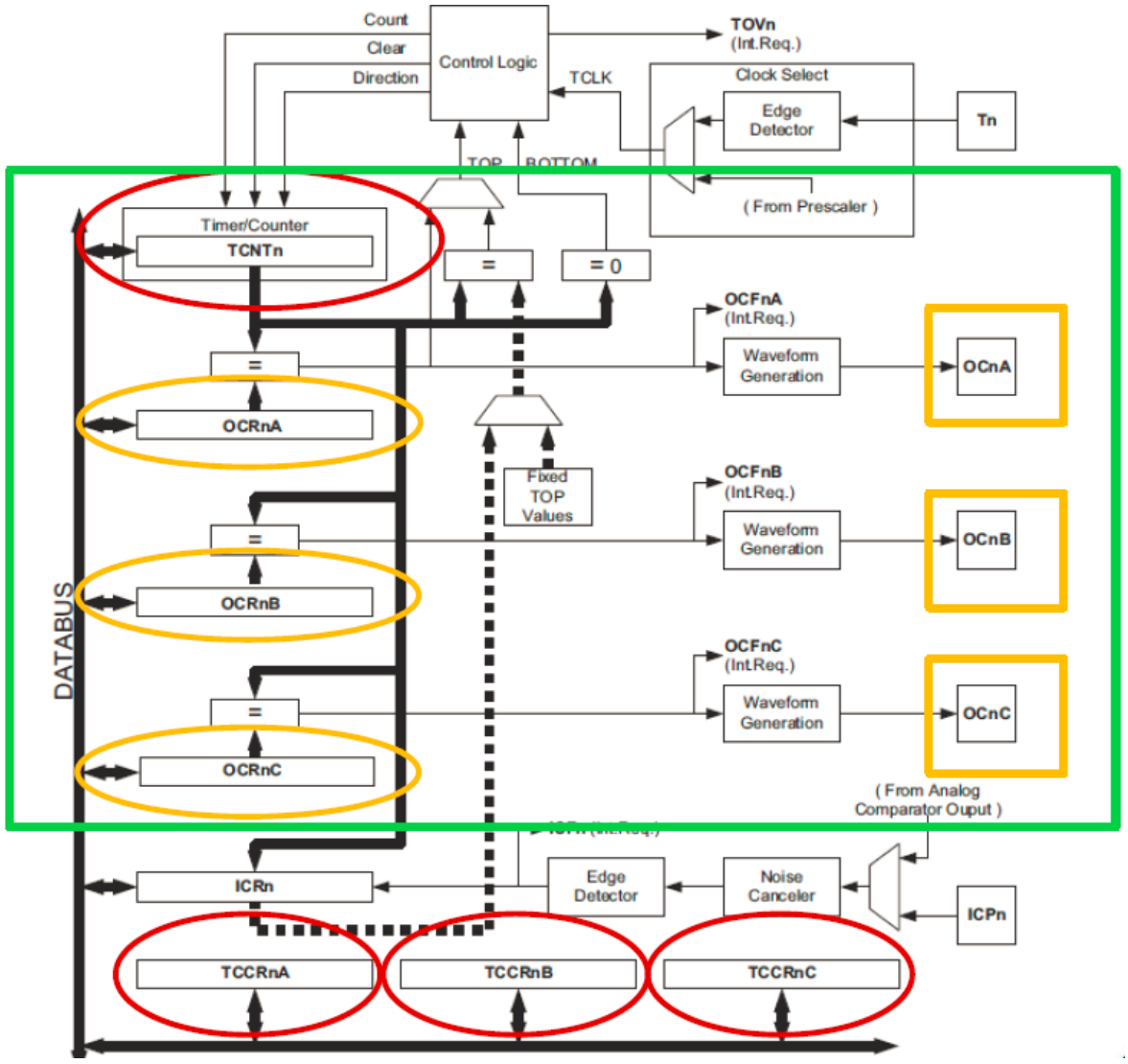
OC0B = PG, ben 5

PS: På vores hardware er PB ben7 (OC1A) forbundet til LED7.

Mega2650 Timers

Henning Hargaard – 2.november 2016

Her er blokdiagrammet for Timer 1, Timer 3, Timer 4 eller Timer 5:



Læg mærke til, at hver af disse 16 bit timere har hele 3 ”systemer”, med hver deres OCR, waveform generator og portben. Vi kalder systemerne henholdsvis ”system A”, ”system B” og ”system C”.

Timer 1	Timer 3
OC1A = PB, ben 5	OC3A = PE, ben 3
OC1B = PB, ben 6	OC3B = PE, ben 4
OC1C = PB, ben 7	OC3C = PE, ben 5

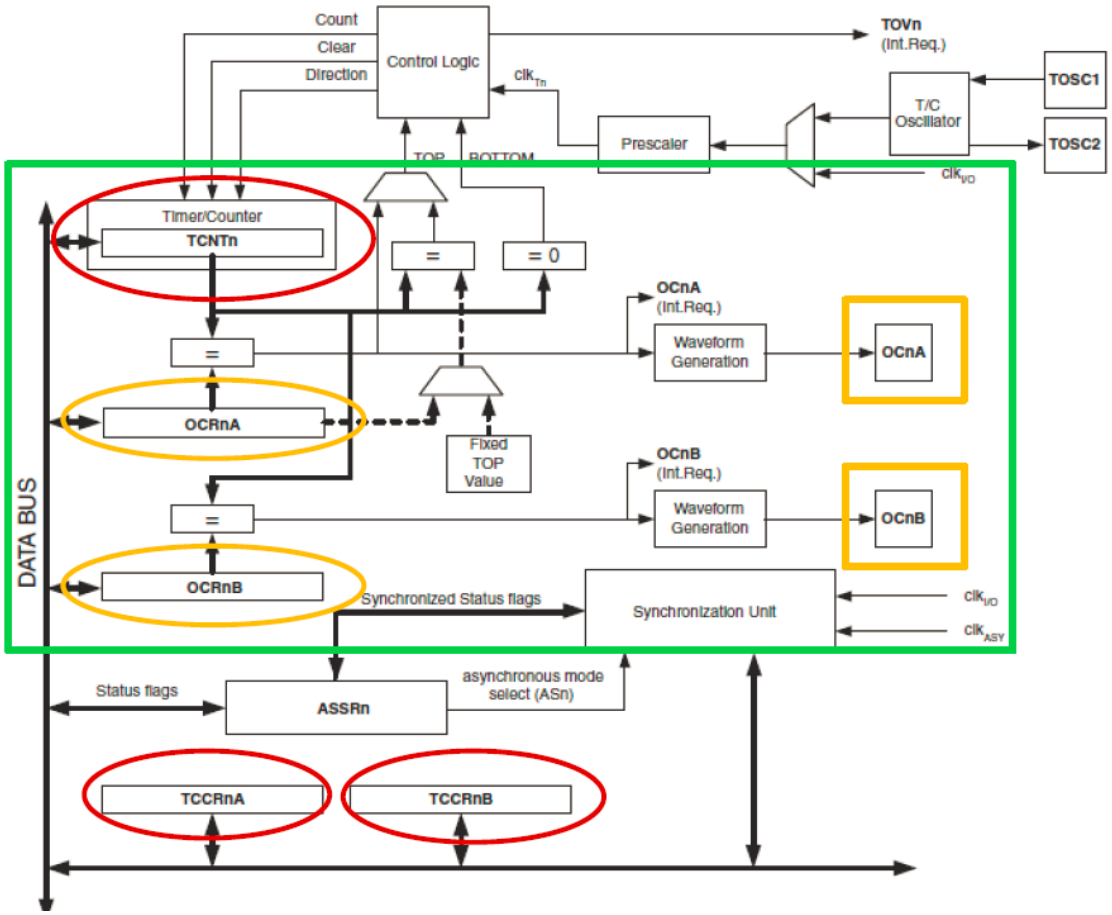
Timer 4	Timer 5
OC4A = PH, ben 3	OC5A = PL, ben 3
OC4B = PH, ben 4	OC5B = PL, ben 4
OC4C = PH, ben 5	OC5C = PL, ben 5

PS: PB på vores hardware er forbundet til LEDs.

Mega2650 Timers

Henning Hargaard – 2.november 2016

Til sidst vises blokdiagrammet for Timer 2:



Heraf ses, at Timer 2 (ligesom Timer 0) har 2 systemer: ”system A” og ”system B”.

OC2A = PB, ben 4

OC2B = PH, ben 6 PS: På vores hardware er PB ben 4 (OC2A) forbundet til LED4.

Med hensyn til 16 bit timernes OCR registre gælder, at hvert 16 bit register fysisk er sammensat af to 8 bit registre.

For eksempel består registeret OCR1A af de to ”halvdele” OCR1AH og OCR1AL:

OCR1AH	=	OCR1AL

Når vi skriver programmet i C, tager compileren (heldigvis) automatisk hånd om opsplitning i 16 bit registerets to halvdele, sådan, at vi blot kan skrive for eksempel:

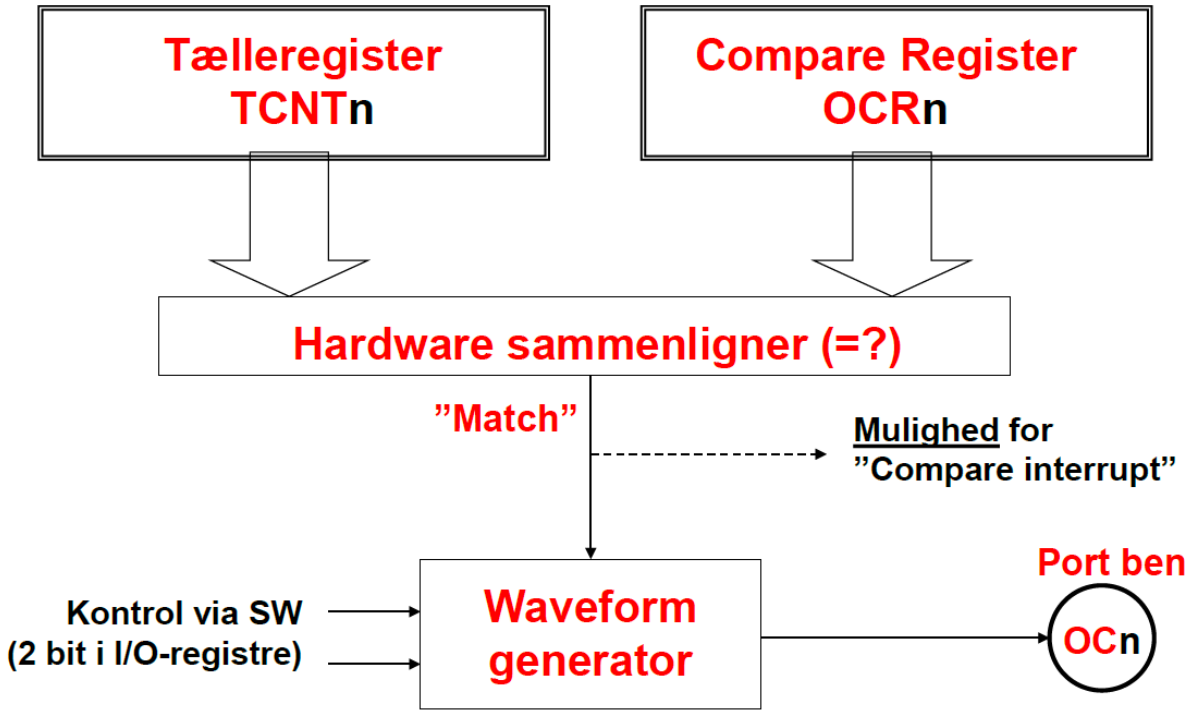
```
OCR1A = 12345;
```

5. CTC mode

Hvis vi har behov for at generere et (firkant-)signal med en bestemt frekvens på et portben, er CTC mode meget egnet. Signalet vil have en duty cycle på 50 %.

Fidusen er, at vi anvender timeren med dens "Output Compare Unit" og waveform generator til at lave signalet.
Når først vi har initieret timeren til at lave signalet, vil det automatisk blive genereret, alt imens vores CPU vil kunne lave noget andet. Tænk igen på timeren som en selvstændig "maskine".

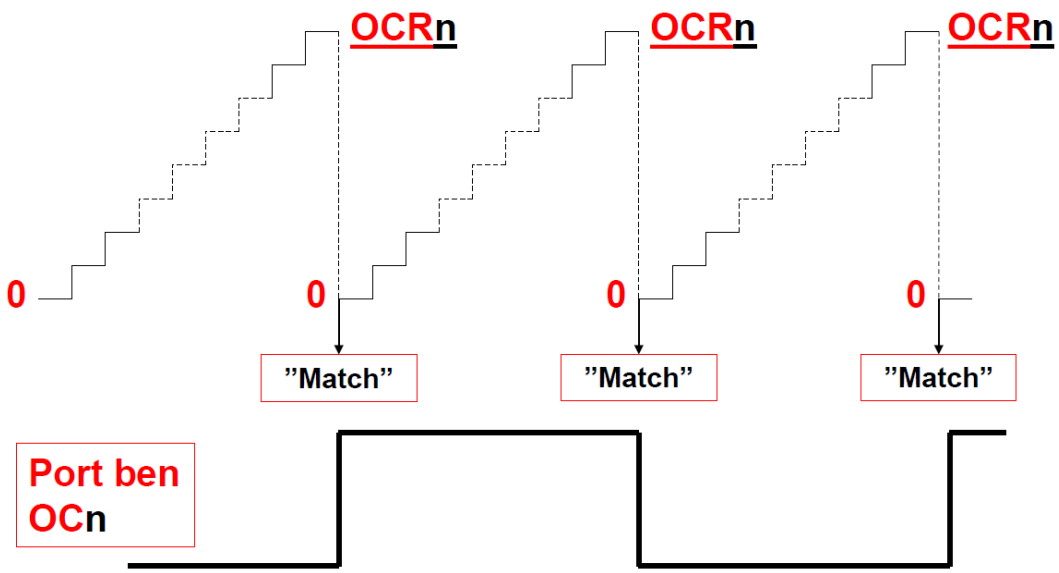
Vi anvender altså dette kredsløb og sætter waveform generatoren til at "toggle" portbenet OCn ved hver match:



Mega2650 Timers

Henning Hargaard – 2.november 2016

Når en timer sættes i ”CTC mode”, tæller den anderledes end i ”normal mode”:



Som vi kan se af figuren, tæller den opad fra 0, men når indholdet af tælleregisteret bliver lig med indholdet af OCRn registeret, genstarter tælleren automatisk fra 0.

Da vi jo også får et match (tælleregister = OCRn) i det øjeblik tælleren nulstilles, vil vi få genereret et firkantsignal på OCn benet, hvis vi programmerer waveform generatoren til at toggle benet ved hvert match.

Det er klart, at vi vil kunne ændre frekvensen af signalet ved at ændre tallet i OCRn registeret.
Man vil kunne resonere sig frem til følgende sammenhæng:

Ben frekvens = f_{cpu} / (2 * N * (1 + OCRn))
N er timerens prescaler-værdi

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 0 i ”CTC mode”:

7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX

Sådan sættes Timer 1, Timer 3, Timer 4 eller Timer 5 i ”CTC” mode:

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

OBS: I denne mode styres TOP (og dermed frekvensen) af 16 bit registeret ICR1 !

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 2 i ”CTC mode”:

7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

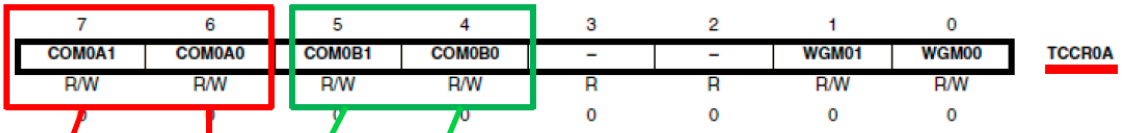
7	6	5	4	3	2	1	0	
FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX

Mega2650 Timers

Henning Hargaard – 2.november 2016

Opsætning af waveform generator for Timer 0 i CTC mode:



7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

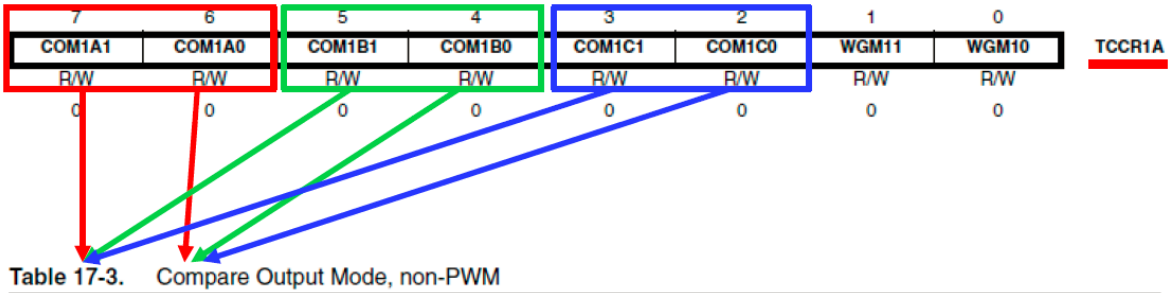
Table 16-2. Compare Output Mode, non-PWM Mode

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected
0	1	Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match
1	1	Set OC0A on Compare Match

Table 16-3. Compare Output Mode, non-PWM Mode

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Toggle OC0B on Compare Match
1	0	Clear OC0B on Compare Match
1	1	Set OC0B on Compare Match

Opsætning af waveform generator for Timer 1, Timer 3, Timer 4 eller Timer 5 i CTC mode:



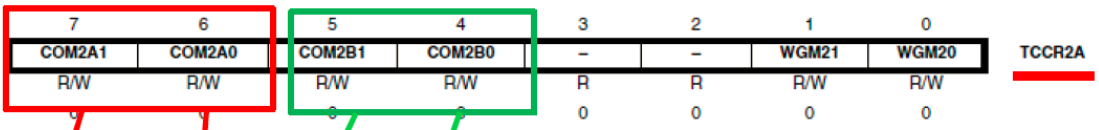
7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCR1A
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Table 17-3. Compare Output Mode, non-PWM

COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected
0	1	Toggle OCnA/OCnB/OCnC on compare match
1	0	Clear OCnA/OCnB/OCnC on compare match (set output to low level)
1	1	Set OCnA/OCnB/OCnC on compare match (set output to high level)

Rød = A-systemet.
Grøn = B-systemet.
Blå = C-systemet.

Opsætning af waveform generator for Timer 2 i CTC mode:



7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

Table 20-2. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected
0	1	Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match
1	1	Set OC2A on Compare Match

Table 20-3. Compare Output Mode, non-PWM Mode

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected
0	1	Toggle OC2B on Compare Match
1	0	Clear OC2B on Compare Match
1	1	Set OC2B on Compare Match

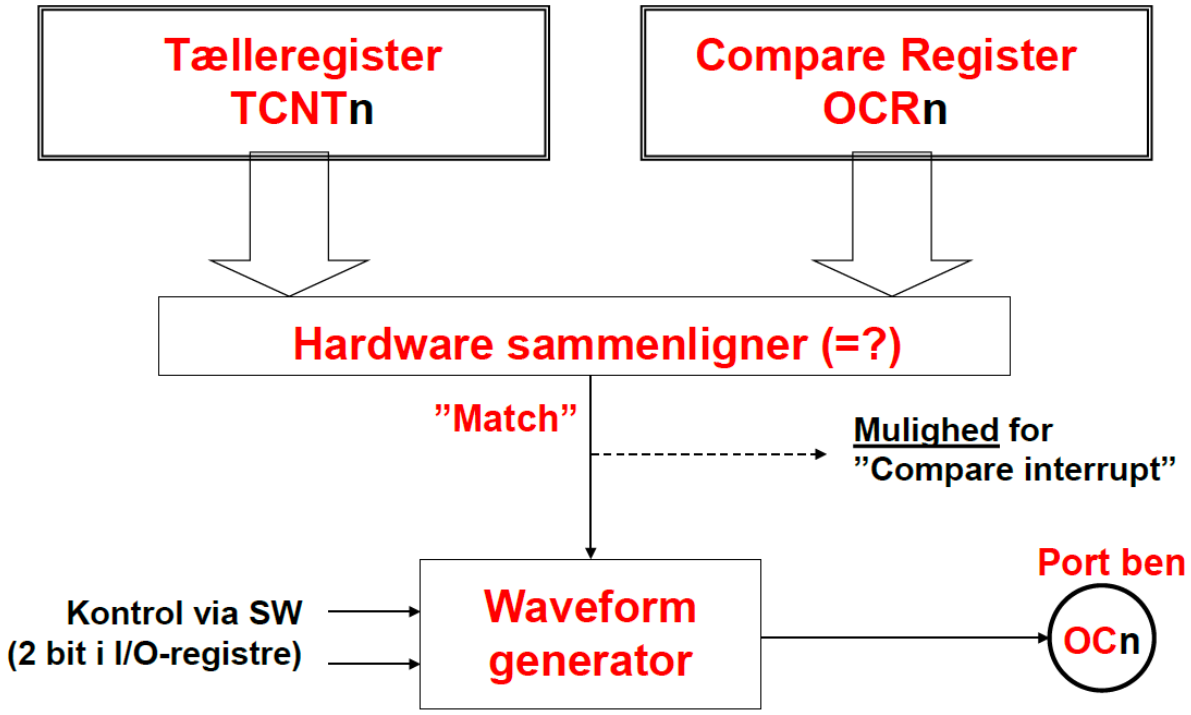
OBS: Der er vigtigt at huske, at sætte det anvendte OC ben op til at være en udgang.
Ellers får vi ikke signalet ud på benet!

6. PWM mode

Hvis vi har behov for at generere et (firkant-)signal med en bestemt duty cycle på et portben, er PWM mode meget egnet.

Som ved CTC mede er fidusen, at vi anvender timeren med dens ”Output Compare Unit” og waveform generator til at lave signalet.
Når først vi har initieret timeren til at lave signalet, vil det automatisk blive genereret, alt imens vores CPU vil kunne lave noget andet.

Vi anvender altså dette kredsløb og sætter waveform generatoren til at styre portbenet OCn ved hver match:



Timerne har flere forskellige ”PWM modes”, der fungerer på hver sin måde.

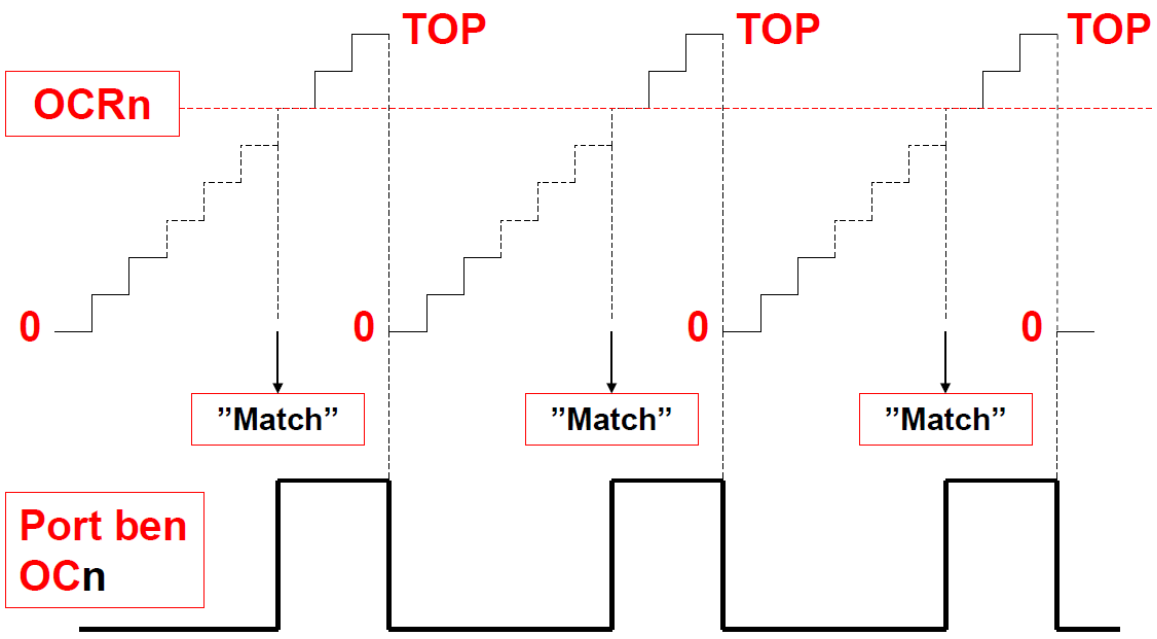
Disse modes har navne som ”Fast PWM”, ”Phase Correct PWM”, ”Phase and Frequency Corret PWM”.

Den vigtigste opdeling er dog opdeling efter **”Fast PWM”** og **”Ikke-fast PWM”**, da måden de fungerer på er væsentlig forskellige.

Mega2650 Timers

Henning Hargaard – 2.november 2016

Når en timer er i ”Fast PWM” mode, fungerer den sådan:



Vi kan se, at der tælles opad fra 0 til ”TOP”, hvorefter tælleren genstarter fra 0.
Det er vigtigt at vide, at ”TOP” er en værdi, der afhænger af, hvilken PWM mode, man har valgt (der er som nævnt mange modes). Hvis man for eksempel har valgt ”Fast PWM, 10 bit” er ”TOP” = 1023.

Tricket er nu at skrive en værdi til ”OCRn” registeret (som ligger mellem 0 og ”TOP”).
Når tælleregisteret bliver lig med OCRn, har vi et match, og der vil blive sendt et signal til waveform generatoren.
Waveform generatoren kan vi eksempelvis programmere til at sætte portben OCn høj, når vi er ved at tælle opad og til at sætte benet OCn lavt, når vi når ”TOP” (det omvendte er også en mulighed).

Som det ses af figuren, vil vi derved få skabt et firkantsignal med en given duty cycle på ben OCn.
Hvis vi ønsker at ændre på signalets duty cycle, skriver vi blot en anden værdi til OCRn registeret.

Man vil kunne resonere sig frem til følgende formel for signalets duty cycle:

$$\text{Duty cycle} = 1 - (OCRn / TOP)$$

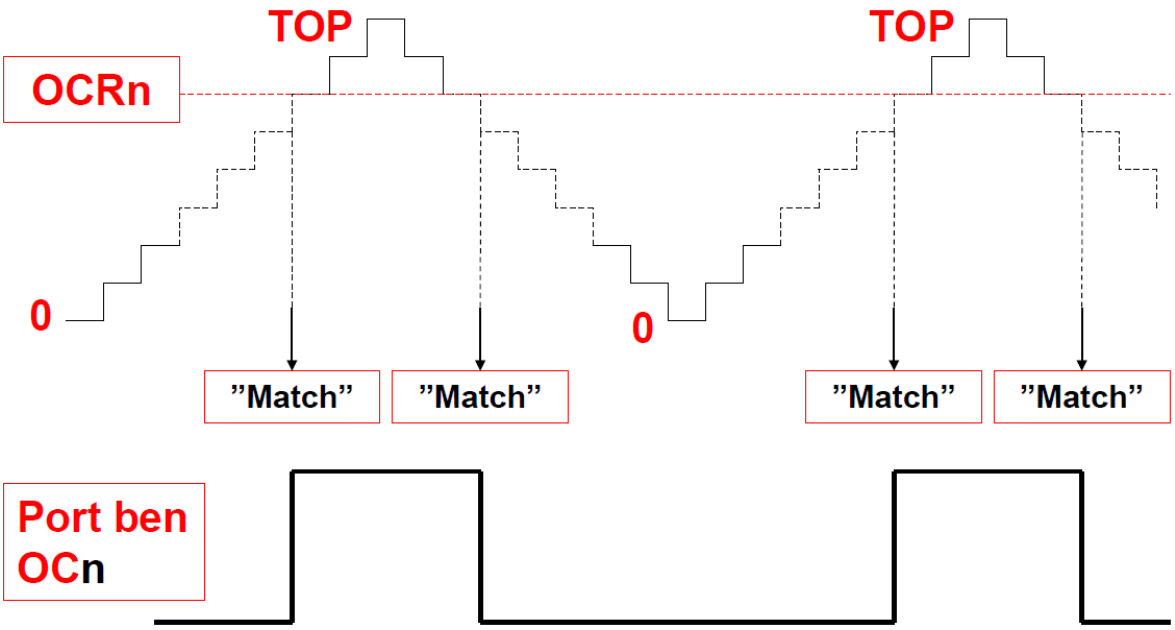
- og følgende formel for signalets frekvens:

$$\text{Frekvens} = f_{cpu} / (N * (1+TOP))$$

Mega2650 Timers

Henning Hargaard – 2.november 2016

Når en timer er i ”Ikke-fast PWM” mode, fungerer den sådan:



Vi kan se, at der tælles opad fra 0 til ”TOP”, hvorefter der vendes rundt og tælles nedad fra ”TOP” til 0.
Det er vigtigt at vide, at ”TOP” er en værdi, der afhænger af, hvilken PWM mode, man har valgt (der er som nævnt mange modes). Hvis man for eksempel har valgt ”Phase Correct PWM, 9 bit” er ”TOP” = 511.

Tricket er nu at skrive en værdi til ”OCRn” registeret (som ligger mellem 0 og ”TOP”).
Når tælleregisteret bliver lig med OCRn, har vi et match, og der vil blive sendt et signal til waveform generatoren.
Waveform generatoren kan vi eksempelvis programmere til at sætte portben OCn høj, når vi er ved at tælle opad og til at sætte benet OCn lavt, når vi er ved at tælle nedad (det omvendte er også en mulighed).

Som det ses af figuren, vil vi derved få skabt et firkantsignal med en given duty cycle på ben OCn.
Hvis vi ønsker at ændre på signalets duty cycle, skriver vi blot en anden værdi til OCRn registeret.

Man vil kunne resonere sig frem til følgende formel for signalets duty cycle:

$$\text{Duty cycle} = 1 - (OCRn / TOP)$$

- og følgende formel for signalets frekvens:

$$\text{Frekvens} = f_{cpu} / (N * 2 * TOP)$$

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 0 i ”PWM mode”:

7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	TOP	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

OBS : TOP er afhængig af mode !

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 1, Timer 3, Timer 4 eller Timer 5 i ”PWM mode”:

7	6	5	4	3	2	1	0	
COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10	TCCRnA
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

7	6	5	4	3	2	1	0	
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCRnB
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGMn3	WGMn2 (CTCn)	WGMn1 (PWMn1)	WGMn0 (PWMn0)	Timer/Counter Mode of Operation	TOP	Update of OCRnx at	TOVn Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCRnA	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICRn	BOTTOM	BOTTOM
9	1	0	0	1	PWM,Phase and Frequency Correct	OCRnA	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICRn	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCRnA	TOP	BOTTOM
12	1	1	0	0	CTC	ICRn	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICRn	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCRnA	BOTTOM	TOP

OBS : TOP er afhængig af mode !

Mega2650 Timers

Henning Hargaard – 2.november 2016

Sådan sættes Timer 2 i ”PWM mode”:

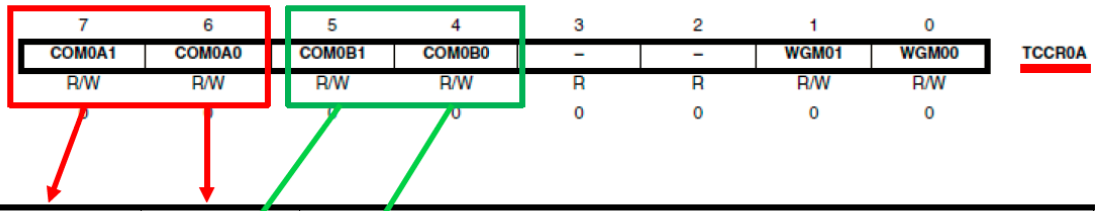
7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	
7	6	5	4	3	2	1	0	
FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[(1)(2)]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

OBS : TOP er afhængig af mode !

Mega2650 Timers

Opsætning af waveform generator for Timer 0 i FAST PWM mode:

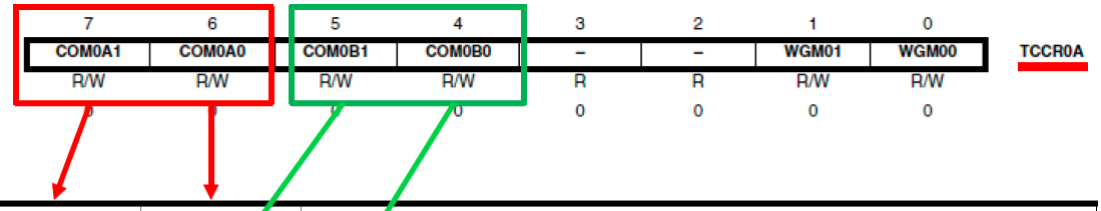


7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match, set OC0A at BOTTOM (non-inverting mode)
1	1	Set OC0A on Compare Match, clear OC0A at BOTTOM (inverting mode)

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on Compare Match, set OC0B at BOTTOM (non-inverting mode)
1	1	Set OC0B on Compare Match, clear OC0B at BOTTOM (inverting mode)

Opsætning af waveform generator for Timer 0 i IKKE-FAST PWM mode:



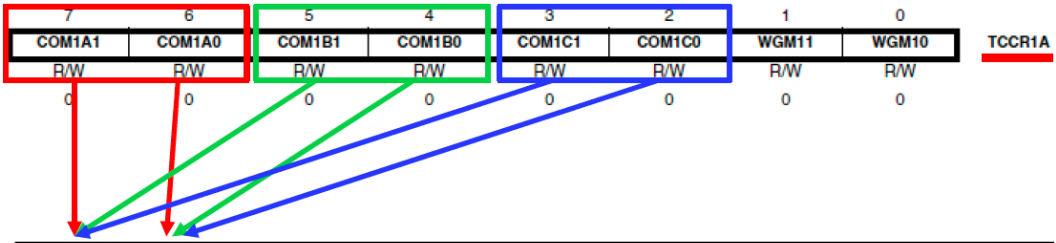
7	6	5	4	3	2	1	0	
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

COM0A1	COM0A0	Description
0	0	Normal port operation, OC0A disconnected
0	1	WGM02 = 0: Normal Port Operation, OC0A Disconnected WGM02 = 1: Toggle OC0A on Compare Match
1	0	Clear OC0A on Compare Match when up-counting. Set OC0A on Compare Match when down-counting
1	1	Set OC0A on Compare Match when up-counting. Clear OC0A on Compare Match when down-counting

COM0B1	COM0B0	Description
0	0	Normal port operation, OC0B disconnected
0	1	Reserved
1	0	Clear OC0B on Compare Match when up-counting. Set OC0B on Compare Match when down-counting
1	1	Set OC0B on Compare Match when up-counting. Clear OC0B on Compare Match when down-counting

Mega2650 Timers

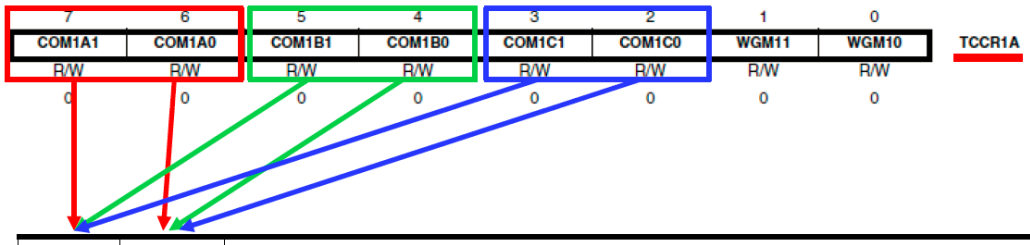
Opsætning af waveform generator for Timer 1, Timer 3, Timer 4 eller Timer 5 i FAST PWM mode:



COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected
1	0	Clear OCnA/OCnB/OCnC on compare match, set OCnA/OCnB/OCnC at BOTTOM (non-inverting mode)
1	1	Set OCnA/OCnB/OCnC on compare match, clear OCnA/OCnB/OCnC at BOTTOM (inverting mode)

Rød = A-systemet.
Grøn = B-systemet.
Blå = C-systemet.

Opsætning af waveform generator for Timer 1, Timer 3, Timer 4 eller Timer 5 i IKKE-FAST PWM mode:



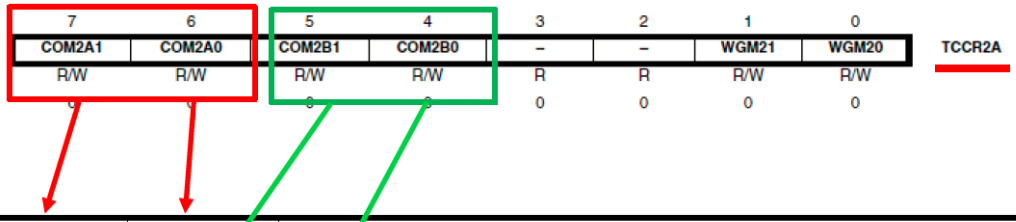
COMnA1 COMnB1 COMnC1	COMnA0 COMnB0 COMnC0	Description
0	0	Normal port operation, OCnA/OCnB/OCnC disconnected
0	1	WGM13:0 =9 or 11: Toggle OC1A on Compare Match, OC1B and OC1C disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B/OC1C disconnected
1	0	Clear OCnA/OCnB/OCnC on compare match when up-counting Set OCnA/OCnB/OCnC on compare match when downcounting
1	1	Set OCnA/OCnB/OCnC on compare match when up-counting Clear OCnA/OCnB/OCnC on compare match when downcounting

Rød = A-systemet.
Grøn = B-systemet.
Blå = C-systemet.

Mega2650 Timers

Henning Hargaard – 2.november 2016

Opsætning af waveform generator for Timer 2 i FAST PWM mode:

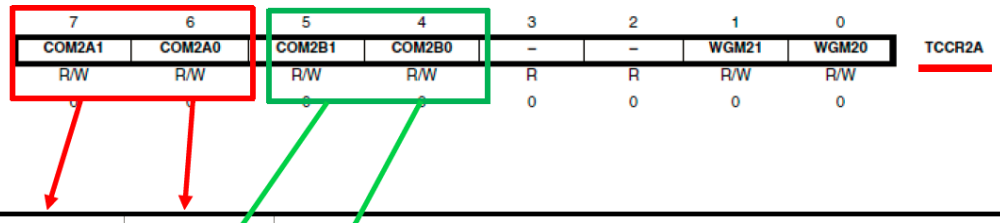


7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected WGM22 = 1: Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match, set OC2A at BOTTOM (non-inverting mode)
1	1	Set OC2A on Compare Match, clear OC2A at BOTTOM (inverting mode)

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected
0	1	Reserved
1	0	Clear OC2B on Compare Match, set OC2B at BOTTOM (non-inverting mode)
1	1	Set OC2B on Compare Match, clear OC2B at BOTTOM (inverting mode)

Opsætning af waveform generator for Timer 2 i IKKE-FAST PWM mode:



7	6	5	4	3	2	1	0	
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
R/W	R/W	R/W	R/W	R	R	R/W	R/W	
0	0	0	0	0	0	0	0	

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected
0	1	WGM22 = 0: Normal Port Operation, OC2A Disconnected WGM22 = 1: Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match when up-counting Set OC2A on Compare Match when down-counting
1	1	Set OC2A on Compare Match when up-counting Clear OC2A on Compare Match when down-counting

COM2B1	COM2B0	Description
0	0	Normal port operation, OC2B disconnected
0	1	Reserved
1	0	Clear OC2B on Compare Match when up-counting Set OC2B on Compare Match when down-counting
1	1	Set OC2B on Compare Match when up-counting Clear OC2B on Compare Match when down-counting

7. I/O registre

For detaljeret beskrivelse af de I/O registre, som anvendes af timerne: Se siderne 22-50 i det blå hæfte ”Mega2560 I/O registers”.