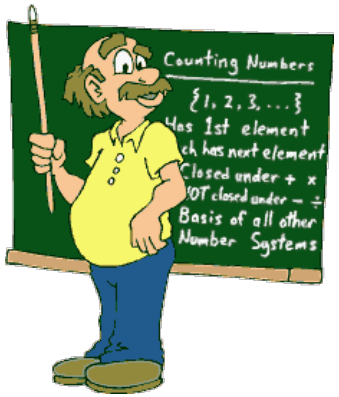
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MSYS

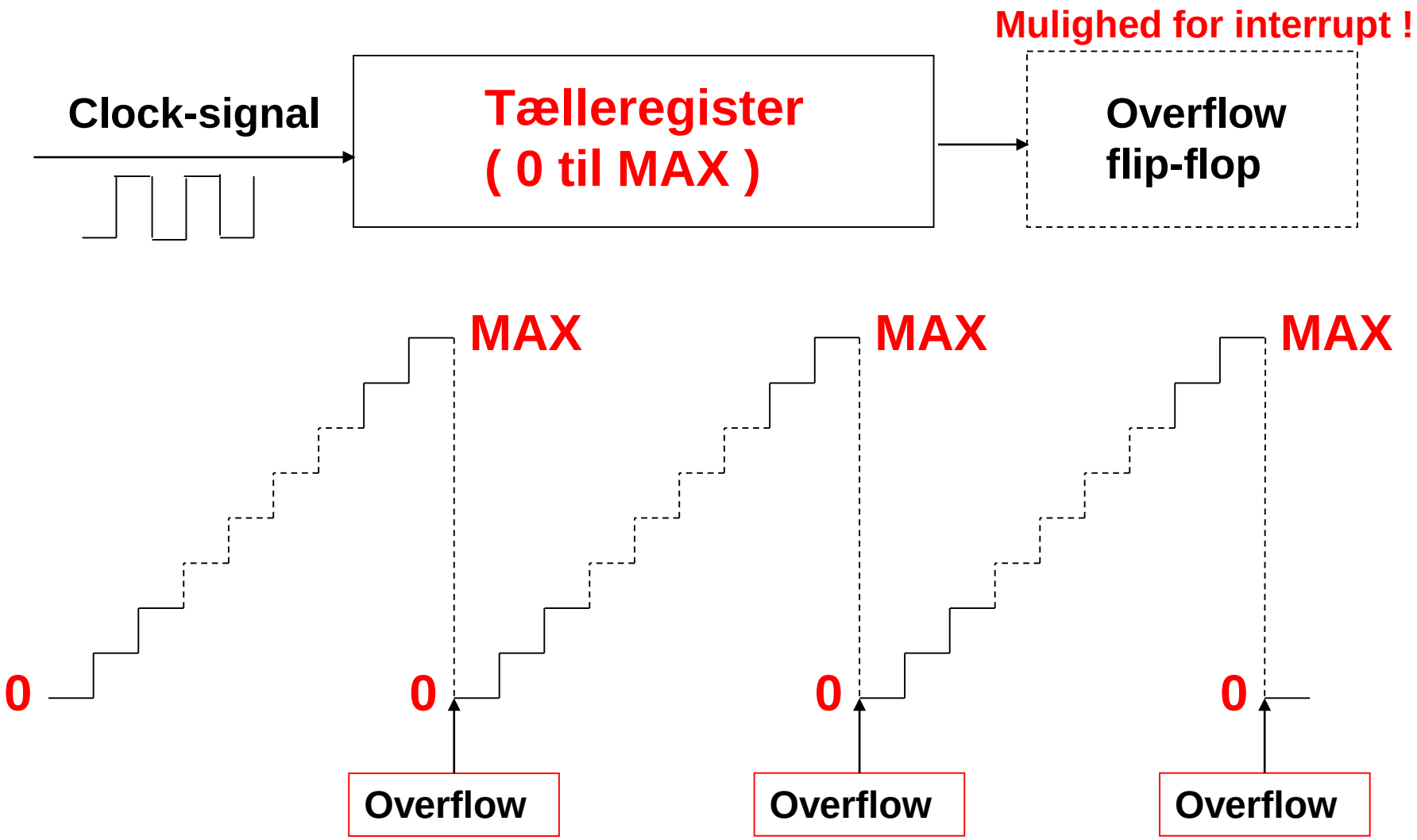
Microcontroller Systems

Lektion 13: Timers i Normal Mode



Version: 30-10-2017, Henning Hargaard

Timer i ”normal mode”



Mega32: 3 timere

- **Timer 0 :**
  8 bit (MAX = 255).
  Normal, CTC og PWM modes.

- **Timer 1 :**
  16 bit (MAX = 65535).
  Normal, CTC, mange PWM modes.
  (Mulighed for ”Input Capture”)

- **Timer 2 :**
  8 bit (MAX = 255).
  Normal, CTC og PWM modes.
  Asynkron mode (Real Time Clock).

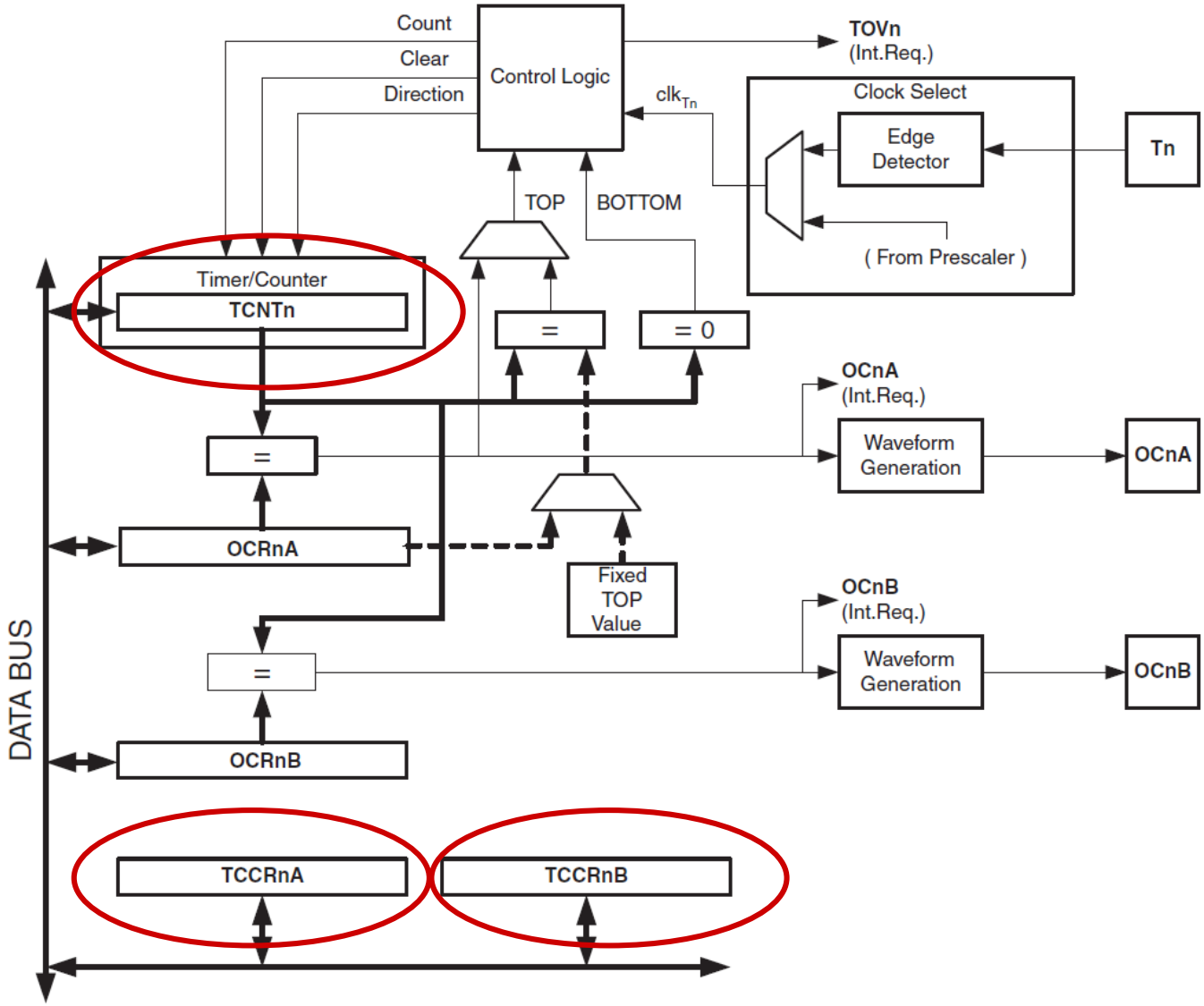
Mega2560: 6 timere

- **Timer 0 :**
  8 bit (MAX = 255).
  Normal, CTC og PWM modes.

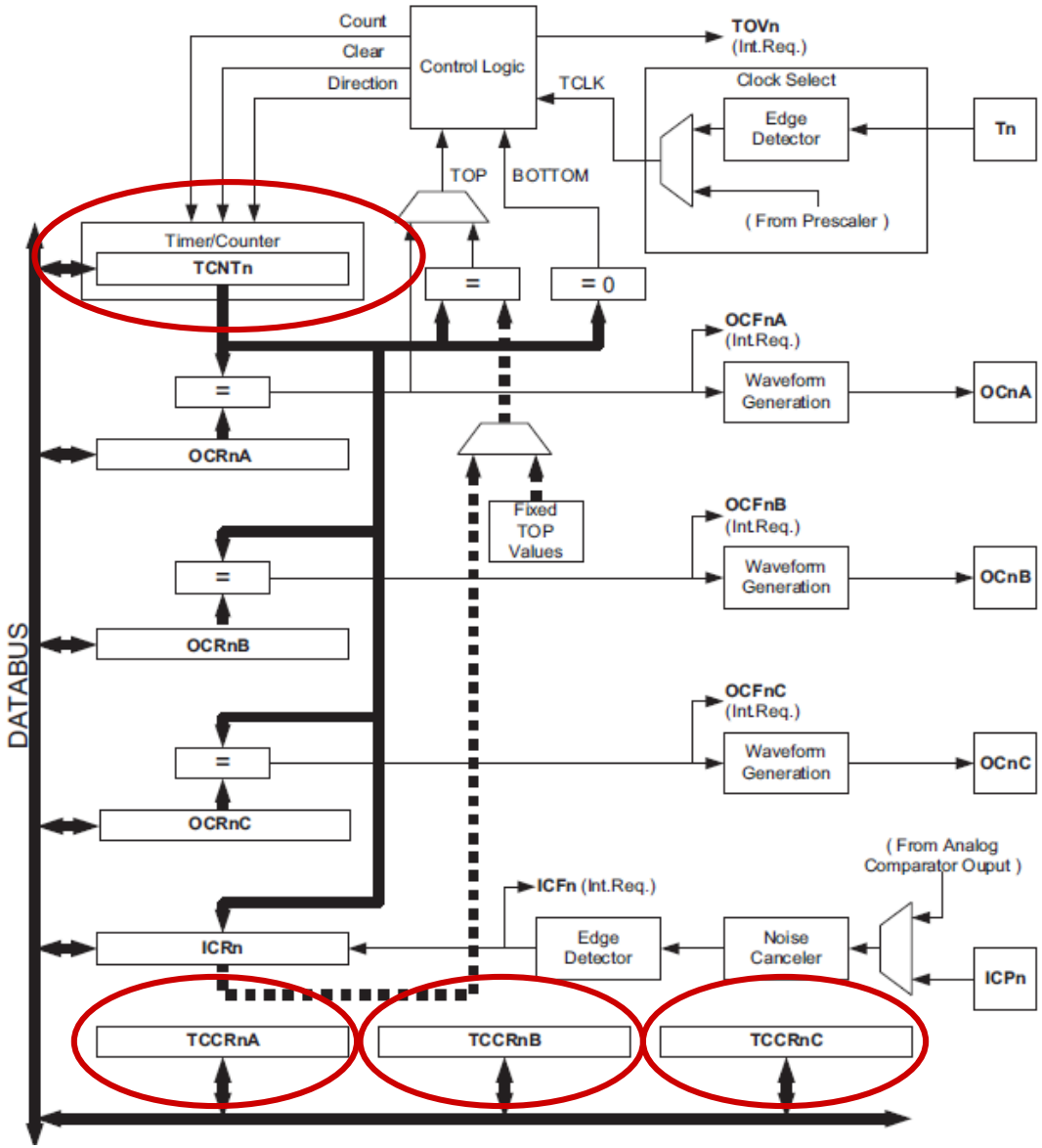
- **Timer 1, Timer 3, Timer 4 og Timer 5 :**
  16 bit (MAX = 65535).
  Normal, CTC, mange PWM modes.
  (Mulighed for ”Input Capture”)

- **Timer 2 :**
  8 bit (MAX = 255).
  Normal, CTC og PWM modes.
  Asynkron mode (Real Time Clock).

Mega2560: Timer 0 (8 bit)

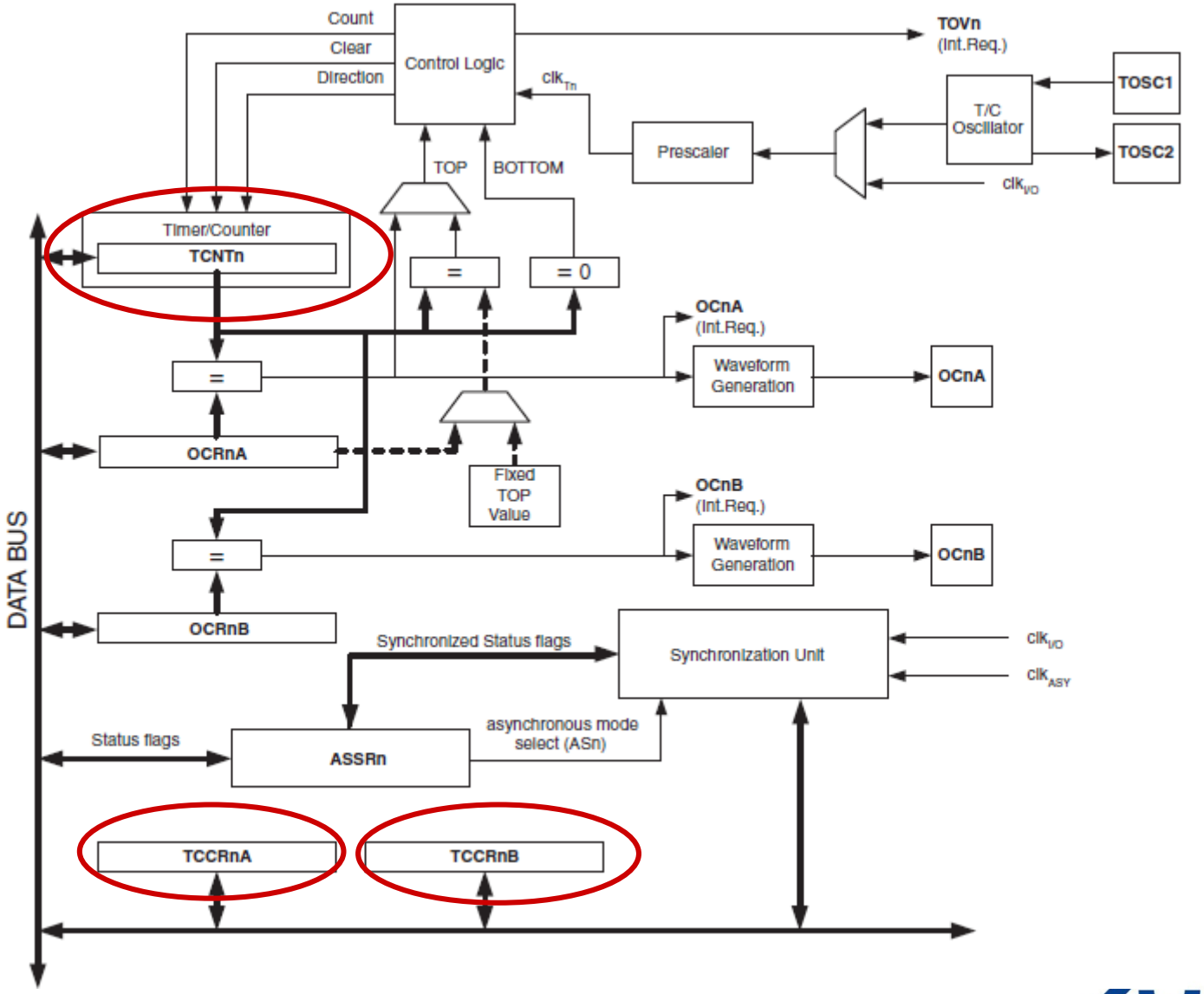


Mega2560: Timer 1,3,4,5 (16 bit)



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: Timer 2 (8 bit)



Timer register (8 bit)

Bit	7	6	5	4	3	2	1	0	
	TCNT0[7:0]								TCNT0
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	

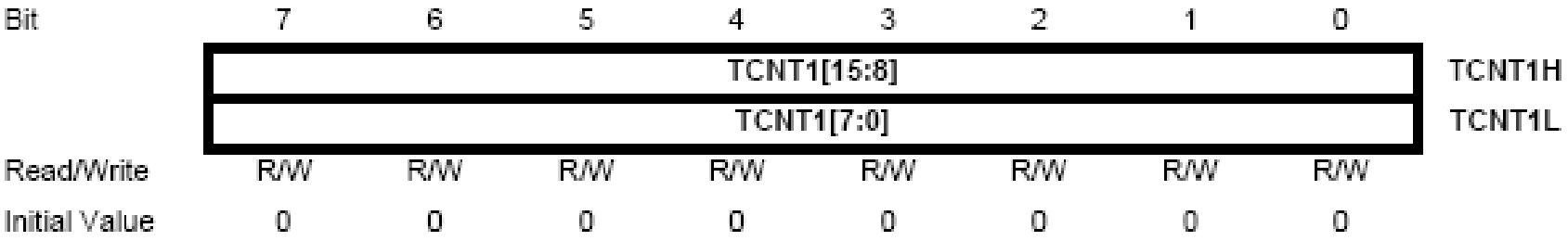
- TCNTn er selve tælle-registeret for timeren.
- Optælles automatisk af timer n clock-signalet.
- Bemærk, at registeret både kan læses fra og skrives til (fra programmet).

Timer register (16 bit)

Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
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	

- TCNTn er selve tælle-registeret for timer n.
- Optælles automatisk af timer n clock-signalet.
- Bemærk, at registeret både kan læses fra og skrives til (fra programmet).

Timer register (16 bit)



AVR GCC C:

```
#include <avr/io.h>

// Herefter er 16-bit adgang muligt:
TCNT1 = 12345;
unsigned int x = TCNT3;
```

- **Rækkefølge ved skrivning: TCNT1H, derefter TCNT1L.**
- **Rækkefølge ved læsning: TCNT1L, derefter TCNT1H.**
- **Kun aktuelt, når assembly anvendes.**

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Valg af Normal Mode

- Normal mode vælges normalt under opstart (initiering).
- Hvilke registre, der skal skrives til, afhænger af, om vi bruger Mega32 eller Mega2560.
  Desuden afhænger det af, hvilken timer, der drejer sig om.
- Normal mode er ”default” efter RESET.

Mega2560: Timer 0, 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

Mega2560: Timer 1,3,4,5. Normal mode

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

TCCRnA

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

TCCRnB

- **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

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: Timer 2, 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	

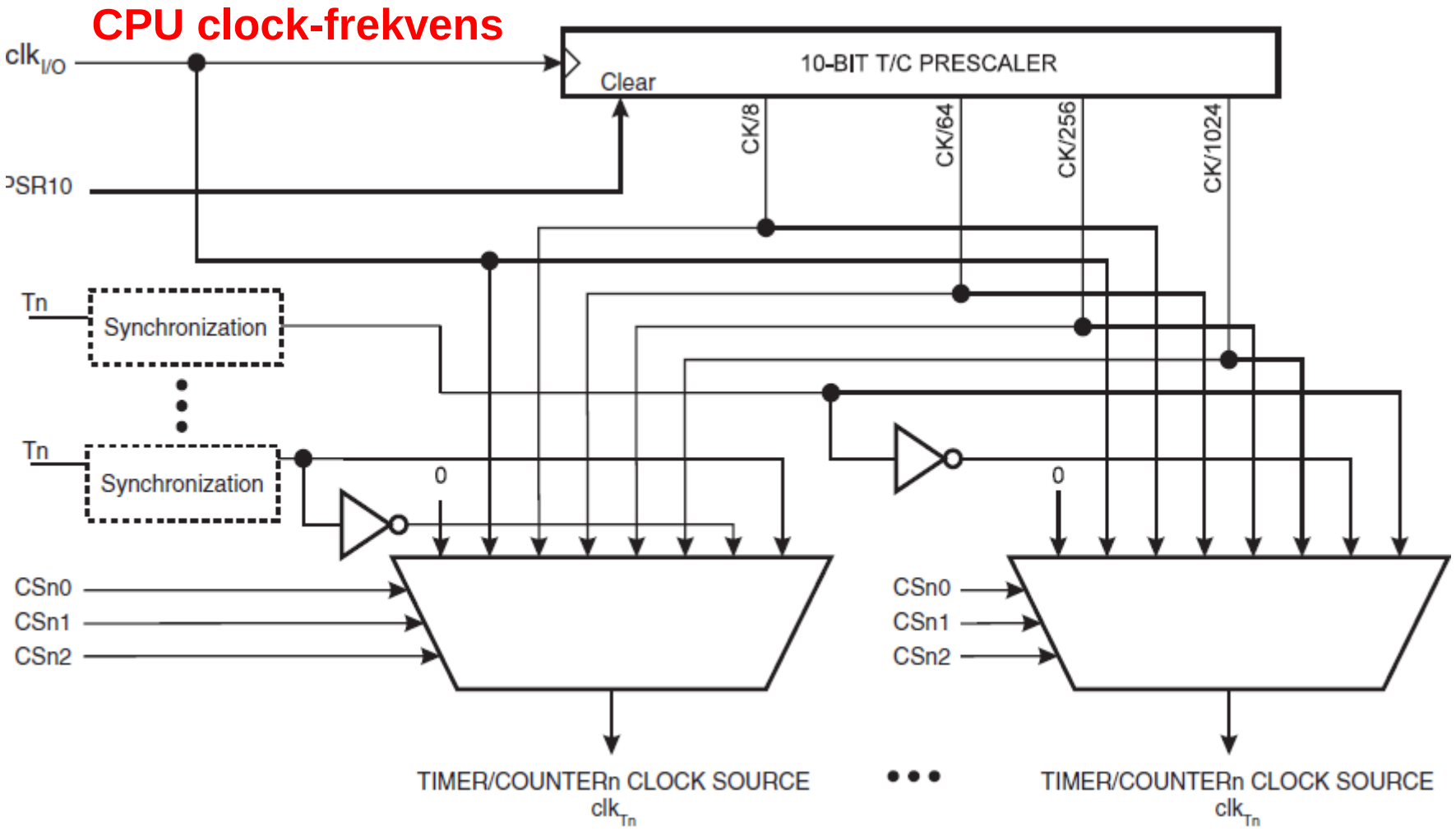
7	6	5	4	3	2	1	0	
FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
W	W	R	R	R/W	R/W	R/W	R/W	
0	0	0	0	0	0	0	0	

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

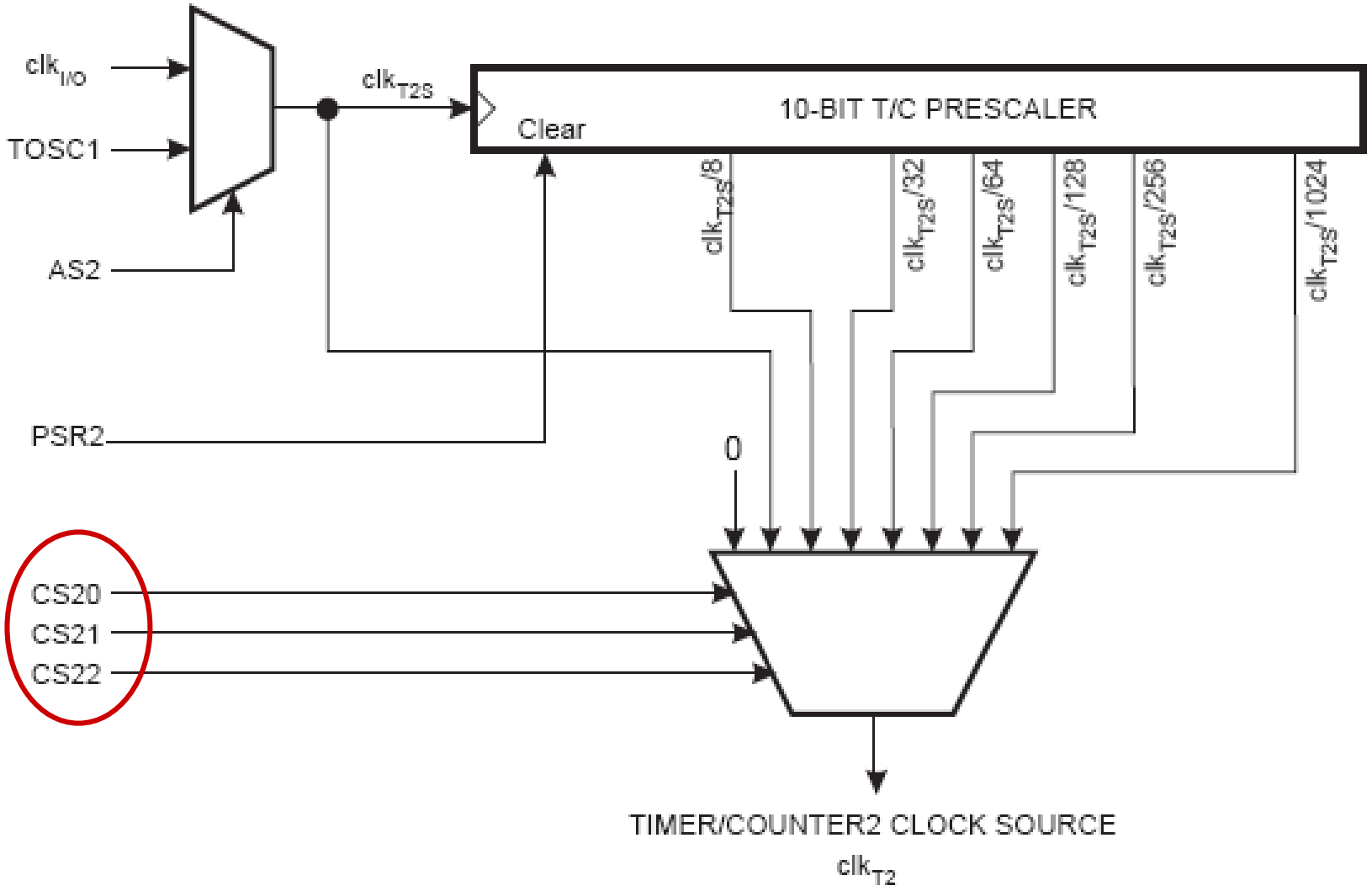
Valg af clocksignal (prescaling)

- En timers clocksignal vælges ved at indstille en ”prescaler”. Hver timer har en clock prescaler.
- Hvilke registre, der skal skrives til, afhænger af, om vi bruger Mega32 eller Mega2560. Desuden afhænger det af, hvilken timer, der drejer sig om.
- ”No clock” er default efter RESET.

Timer Prescaler (ikke Timer 2)



Prescaler for timer 2



Mega2560: Timer 0. Valg af clock

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

Mega2560: Timer 1,3,4,5. Valg af clock

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

Mega2560: Timer 2, Valg af clock

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)

Test (”socrative.com”: Room = MSYS)

- Antag, at vi anvender en CPU clockfrekvens på 3,6864 MHz, og at Timer 0 er initieret til Normal Mode.
  Timer 0's clock prescaler er sat til 64.
  Hvor ofte laver Timer 0 overflow ?

- A: 225 gange per sekund.
- B: 256 gange per sekund.
- C: 57600 gange per sekund.
- D: 64 gange per sekund.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test (”socrative.com”: Room = MSYS)

- Antag Timer 1 er i Normal Mode og CPU clockfrekvensen er 16 MHz.
  Hvad er den længste tid, der kan opnås mellem hvert Timer 1 overflow ?

- A: Ca. 244 sekunder.
- B: 15625 sekunder.
- C: Ca. 240 ms.
- D: Ca. 4,2 sekunder.

Eksterne Clock-ben

- Hvis der for en timer er valgt ”External Clock”, tæller timeren pulser på et microcontrollerben.
- Hver timer (undtaget Timer 2) har er ben, der kan bruges som eksternt clocksignal.

  Se næste slide.

Mega2560: Eksterne Clock-ben

- **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.

Timer overflow flag

- Hver gang en timer laver overflow, vil den sætte et ”overflow flag” (til 1).
  Flaget er et bit i et bestemt I/O register.

- Bliver kun nulstillet igen, hvis vi *skriver et 1-tal* til det (medmindre vi bruger interrupts).
  Det svarer til at ”sænke flaget”.

- Hvilke registre, der skal skrives til, afhænger af, om vi bruger Mega32 eller Mega2560.
  Desuden afhænger det af, hvilken timer, der drejer sig om.

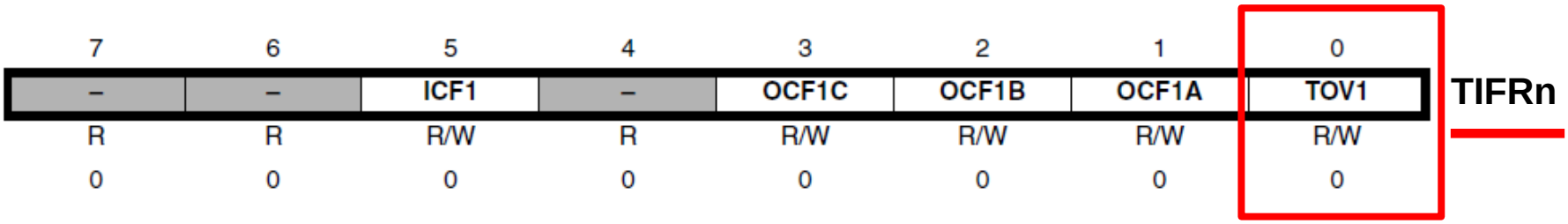
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: 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	

- **Bit 0 i registeret TIFR0 bliver sat til 1, hver gang Timer 0 laver overflow.**
- **Kan kun nulstilles igen, hvis vi *skriver et 1-tal* til det (medmindre vi bruger interrupts).**
- **C-kode:**
  TIFR0 = 0b00000001;

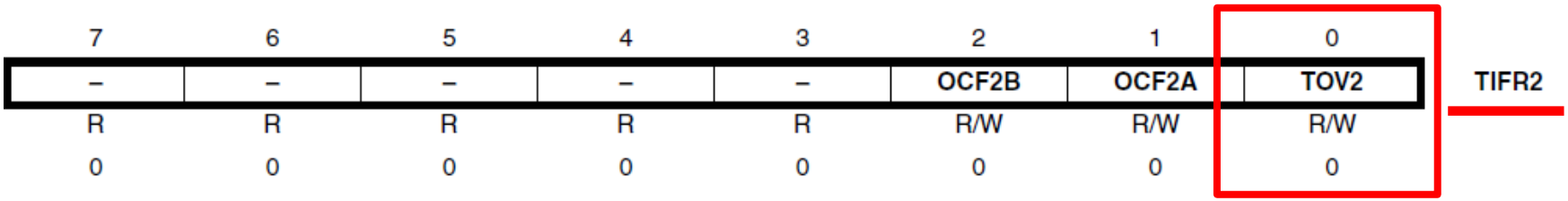
Mega2560: Timer 1,3,4,5. Overflow flags



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	

- **TIFRn = TIFR1, TIFR3, TIFR4 eller TIFR5.**
- **Bit 0 i registeret TIFR”n” bliver sat til 1, hver gang Timer ”n” laver overflow.**
- **Kan kun nulstilles igen, hvis vi *skriver et 1-tal* til det (medmindre vi bruger interrupts).**
- **C-kode:**
  TIFR4 = 0b00000001;

Mega2560: 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	

- **Bit 0 i registeret TIFR2 bliver sat til 1, hver gang Timer 2 laver overflow.**
- **Kan kun nulstilles igen, hvis vi *skriver et 1-tal* til det (medmindre vi bruger interrupts).**
- **C-kode:**
  TIFR0 = 0b00000001;

Delay med Timer i Normal mode

Metode:

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].

Eksempel fra bogen: Timer 0

Example 9-39

Write a C program to toggle all the bits of PORTB continuously with some delay. Use Timer0, Normal mode, and no prescaler options to generate the delay.

Solution:

```
#include "avr/io.h"
void T0Delay ( );
int main ( )
{
      DDRB = 0xFF;

      while (1)
      {
            PORTB =
            T0Delay
            PORTB =
            T0Delay
      }
}

void T0Delay ( )
{
      TCNT0 = 0x20;              //load TCNT0
      TCCR0 = 0x01;              //Timer0, Normal mode, no prescaler
      while ((TIFR&0x1)==0);     //wait for TF0 to roll over
      TCCR0 = 0;
      TIFR = 0x1;                //clear TF0
}
```

Mega2560: Timer 0 overflow 10000 gange i sekundet

```
void T0Delay()
{
  // 16000000 Hz /8 = 2000000 Hz
  // 2000000 Hz / 200 = 10000 Hz
  TCNT0 = 256-200;
  // Timer 0 i Normal mode og PS = 8
  TCCR0A = 0b00000000;
  TCCR0B = 0b00000010;
  // Vent på Timer 0 overflow flag
  while ((TIFR0 & (1<<0)) == 0)
  {}
  // Stop Timer 0 (ingen clock)
  TCCR0B = 0b00000000;
  // Nulstil Timer 0 overflow flag
  TIFR0 = 0b00000001;
}
```

Eksempel fra bogen: Timer 1

Example 9-42 (C version of Example 9-32)

Write a C program to toggle only the PORTB.4 bit continuously every second. Use Timer1, Normal mode, and 1:256 prescaler to create the delay. Assume XTAL = 8 MHz.

Solution:

XTAL = 8 MHz ➔ $T_{machine\ cycle}$ = 1/8 MHz = 0.125 µs = T_{clock}

Prescaler = 1:256 ➔ T_{clock} = 256 × 0.125 µs = 32 µs

1 s/32 µs = 31,250 clocks = 0x7A12 clocks ➔ 1 + 0xFFFF – 0x7A12 = **0x85EE**

```
#include "avr/io.h

void T1Delay ( );

int main ( )
{
      DDRB = 0xFF;

      while (1)
      {
            PORTB
            T1Dela
      }
}

void T1Delay ( )
{
      TCNT1H = 0x85;     //TEMP = 0x85
      TCNT1L = 0xEE;

      TCCR1A = 0x00;     //Normal mode
      TCCR1B = 0x04;     //Normal mode, 1:256 prescaler

      while ((TIFR&(0x1<<TOV1))==0);      //wait for Tf0 to roll over

      TCCR1B = 0;
      TIFR = 0x1<<TOV1;       //clear TOV1
}
```

Mega2560: Timer 1 overflow hver 3. sekund

```
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;
}
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test (”socrative.com”: Room = MSYS)

```
void T0Delay()
{
  TCNT0 = 117;
  TCCR0A = 0b00000000;
  TCCR0B = 0b00000011;
  // Vent på Timer 0 overflow flag
  while ((TIFR0 & (1<<0)) == 0)
  {}
  // Stop Timer 0 (ingen clock)
  TCCR0B = 0b00000000;
  // Nulstil Timer 0 overflow flag
  TIFR0 = 0b00000001;
}
```

A: 46,8 us

B: 8,7 us

C: 7,48 ms

D: 556 us

E: 7,31 us

Mega2560 anvendes.
CPU frekvens = 16 MHz.
Hvad er forsinkelsen ?

Test (”socrative.com”: Room = MSYS)

- En Mega2560's clockfrekvens er 3,6864 MHz. Hvordan initieres Timer 2 til normal mode, således at vi får et Timer 2 overflow interrupt præcis 14400 gange hvert sekund ?

- A: TCCR2A = 0b00000000;
  TCCR2B = 0b00000001;
- B: TCCR2A = 0b00000000;
  TCCR2B = 0b000000110;
- C: TCCR2A = 0b00000000;
  TCCR2B = 0b00000000;

Slut på lektion 13