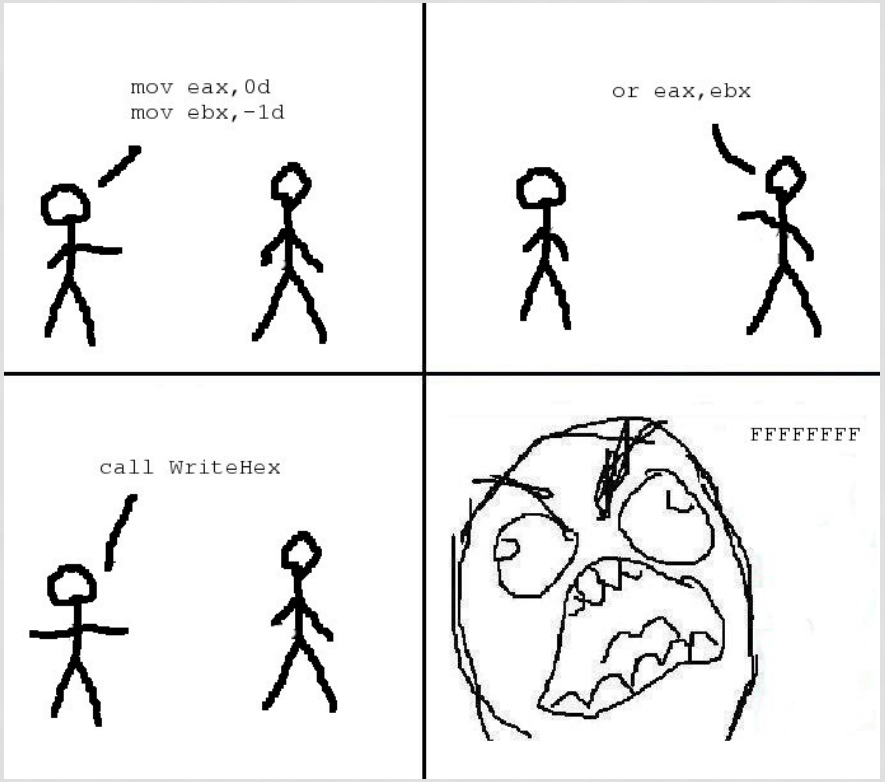


ASSEMBLY & C

WEEK 4-2

AGENDA

week	onderwerp	week	week
1	de structuur van AVR-assembly AVR instructies AVR registers en I/O ATmega memory map Atmel Studio AVR expressies en directives AVR addressing modes	3	de structuur van C-programma's ATMEL studio en AVR libc typen, constanten en operatoren AVR register access in C control statements functies & stackframe visibility scope arrays & strings struct & enum
2	flow of control spring instructies, control structuren Arduino UNO AVR studio stack & subroutines interrupts timer/counters switch bounce	4	interrupts in C TM1638 led&key UART PWM & ADC using a TTC-scheduler state diagram

AGENDA

- **PWM**
- ADC
- een co-operatieve scheduler
- state diagram

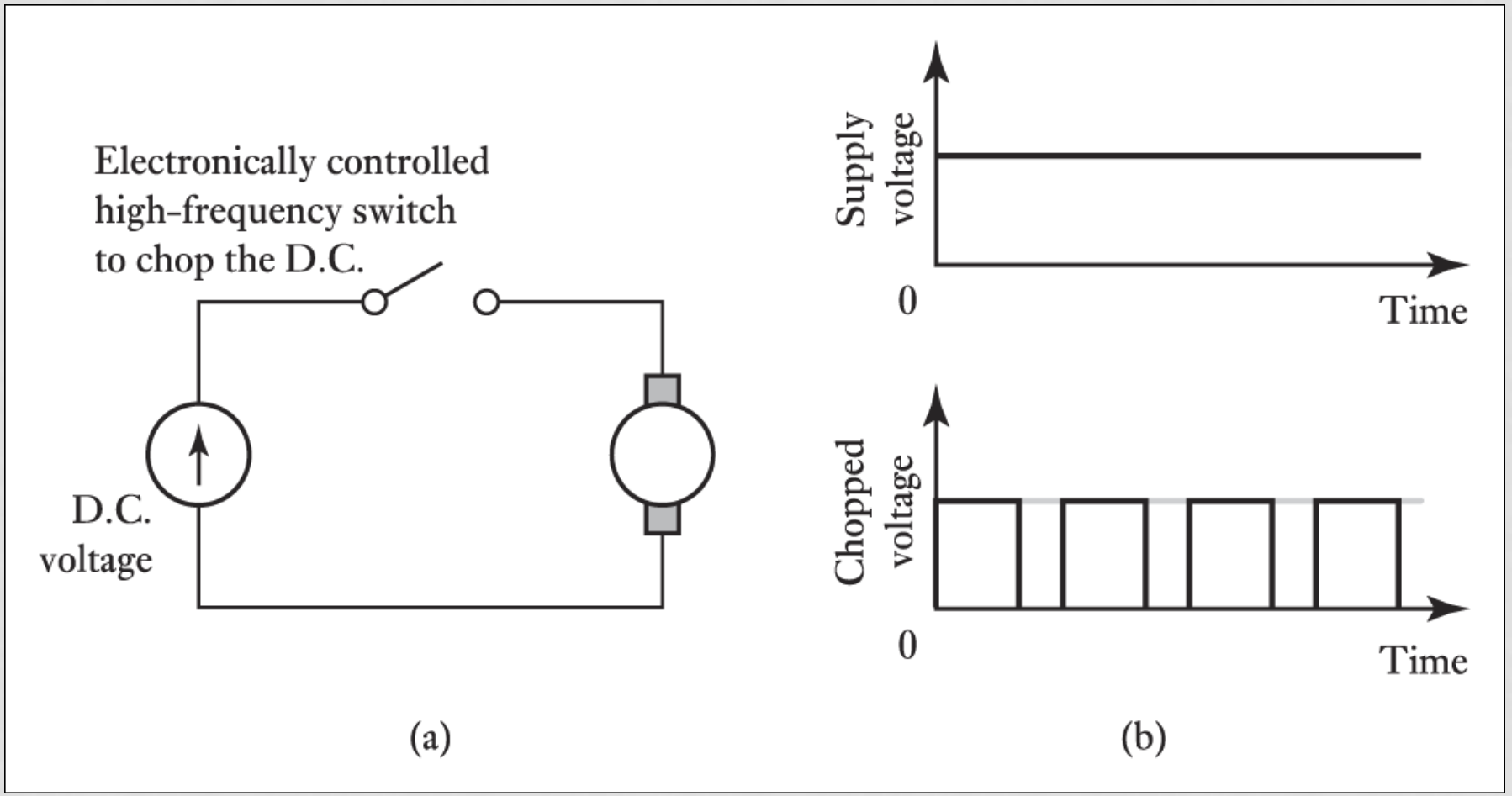
ANALOGE AANSTURING

- stel een motor die op gelijkspanning draait bij 5 Volt 3000 rpm
- we willen deze motor op 1000 rpm laten draaien
- analoge aansturing
  - spanning verlagen
    - motor zal bij, afhankelijk van het type, bij 2 - 2.5V stoppen met draaien
  - stroom verlagen
    - dit kan gemaakt worden met speciale (maar dure) elektronica

ANALOGE AANSTURING

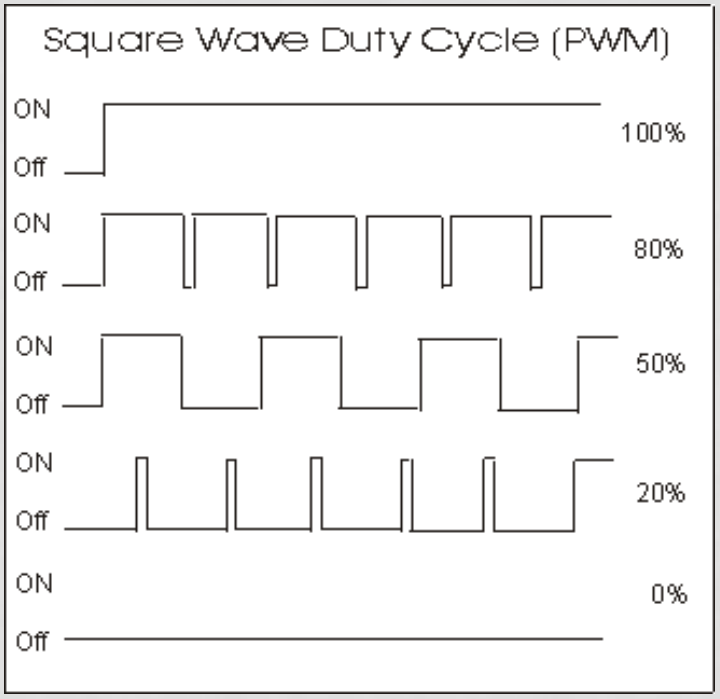
- vermogen overdragen aan analoge componenten zoals : motor, lamp, oven, rem
- analoge aansturing
  - inefficiënte vermogensoverdracht
    - veel vermogen word omgezet in warmte
  - instabiel, gevoelig voor temperatuur (ruis) en EM velden
  - nauwkeurigheid is kostbaar

PWM

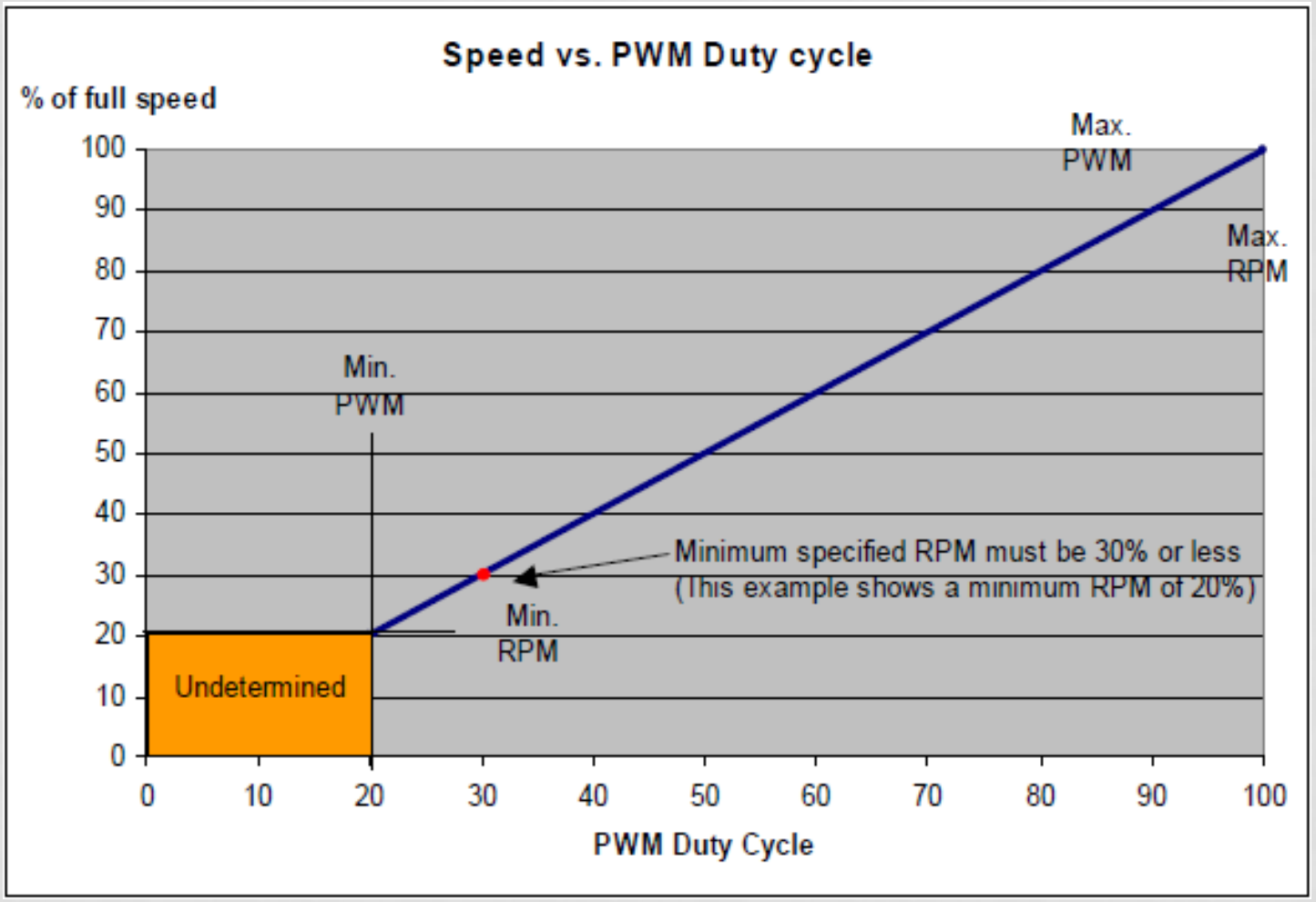

Electronically controlled
high-frequency switch
to chop the D.C.
D.C.
voltage
(a)
Supply
voltage
0
Time
Chopped
voltage
0
Time
(b)

DUTY CYCLE

- oppervlakte komt overeen met het vermogen
- verhouding tussen de tijd dat het signaal ‘hoog’ is en de totale tijd noemt men de duty cycle

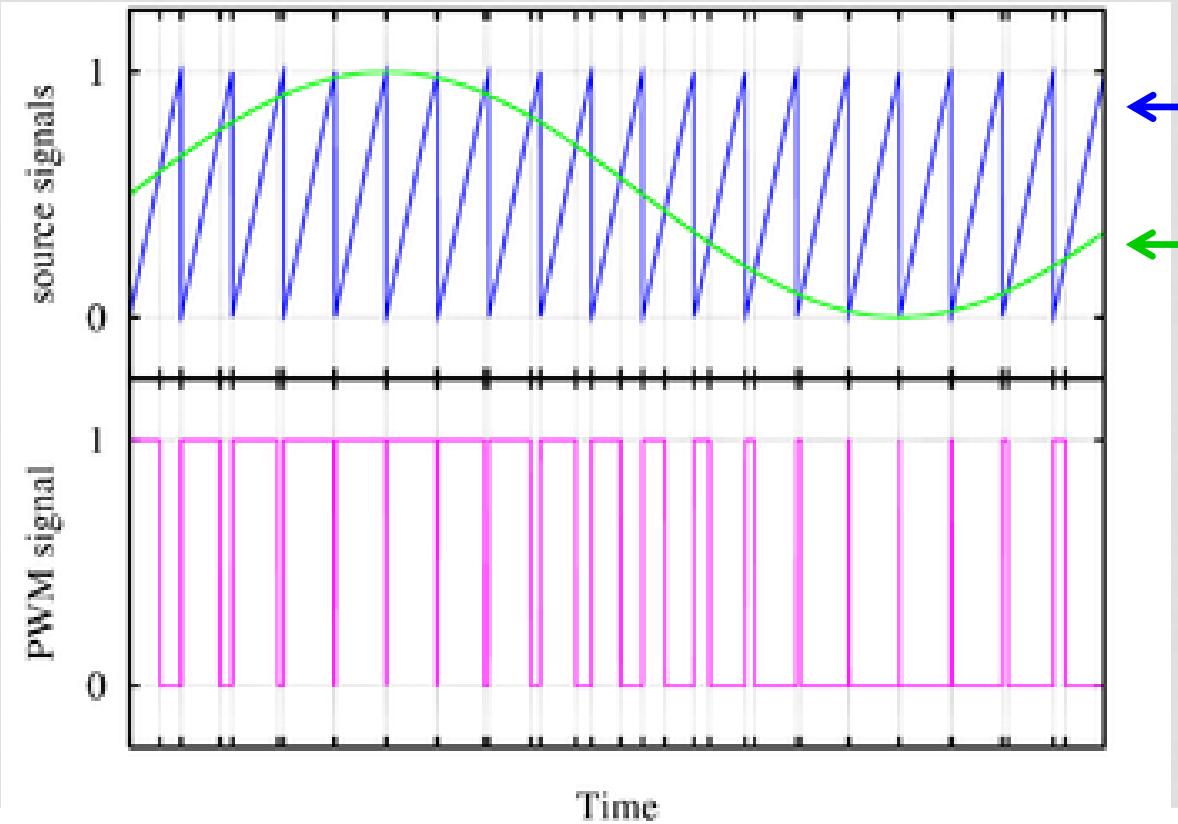


NIET-LINEAIR GEDRAG



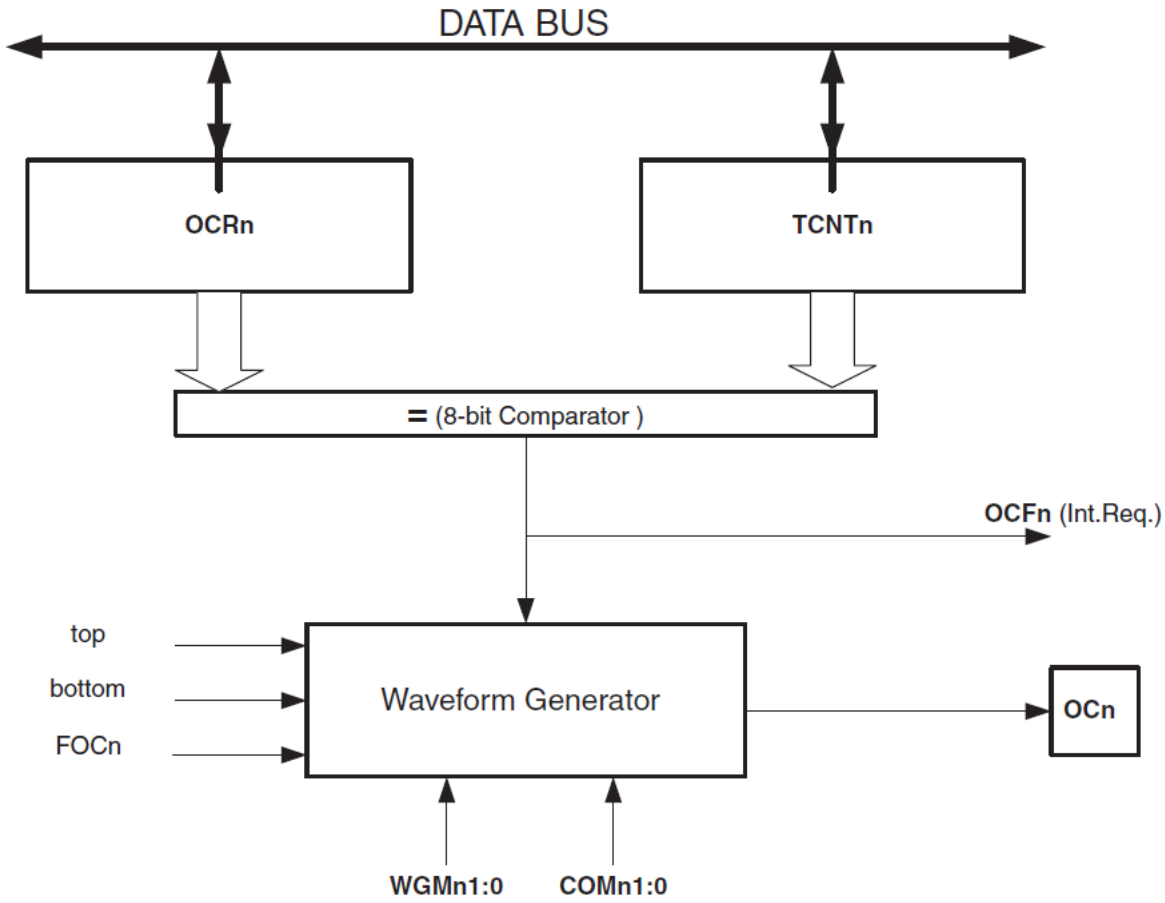
PWM

- modulatie van de duty cycle
- wat is de optimale frequentie ?
  - d.w.z. hoe snel telt de teller ?



BLOK DIAGRAM

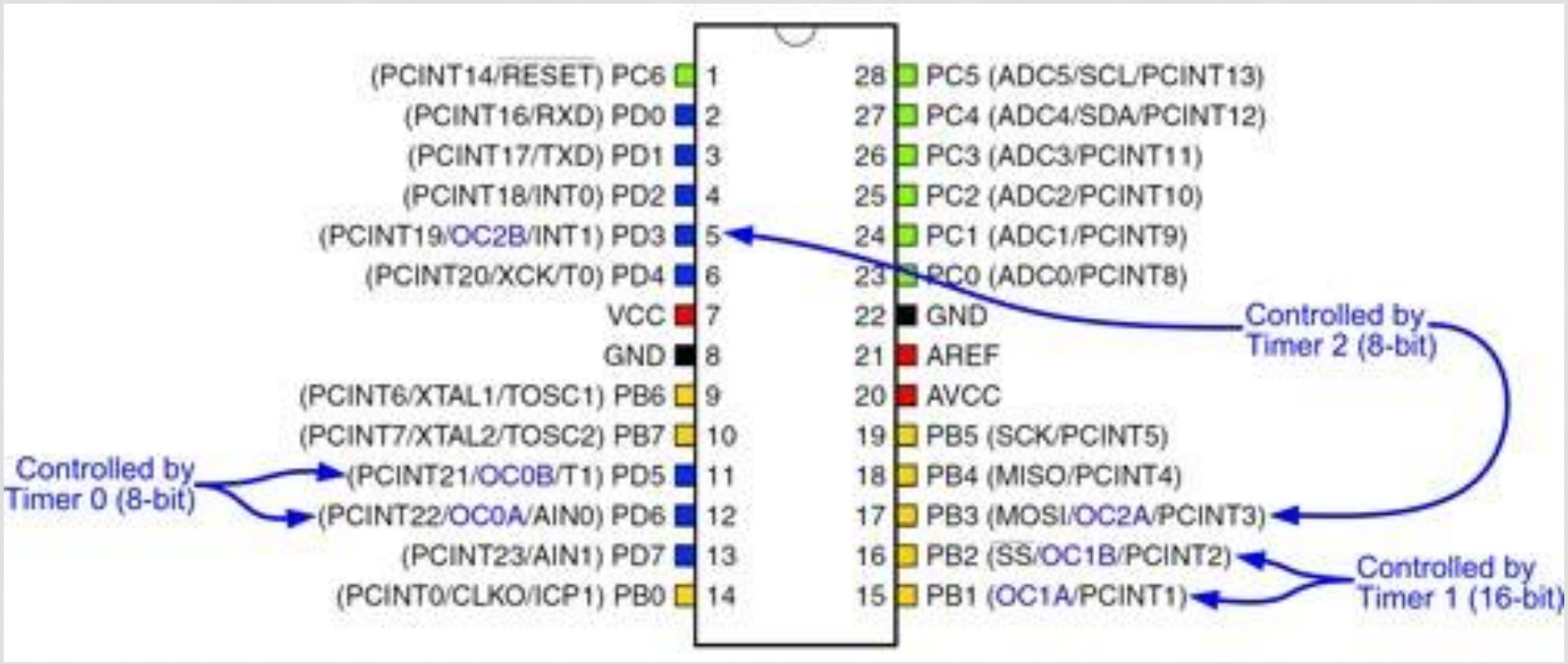
Figure 29. Output Compare Unit, Block Diagram



TIMERS/COUNTERS

- er zijn 3 timer/counters on-chip (328P)
  - timer 0 & timer 2 : 8 bit
  - timer 1 : 16-bit
- bij elke clock-cycle wordt teller "automatisch" verhoogd of verlaagd
- de systeem klok kan worden gedeeld : prescaling met 8, 64, 256 of 1024
- timer 0 en timer 1 hebben 2 onafhankelijke Output Compare Units (A & B)

OUTPUT TIMER 0



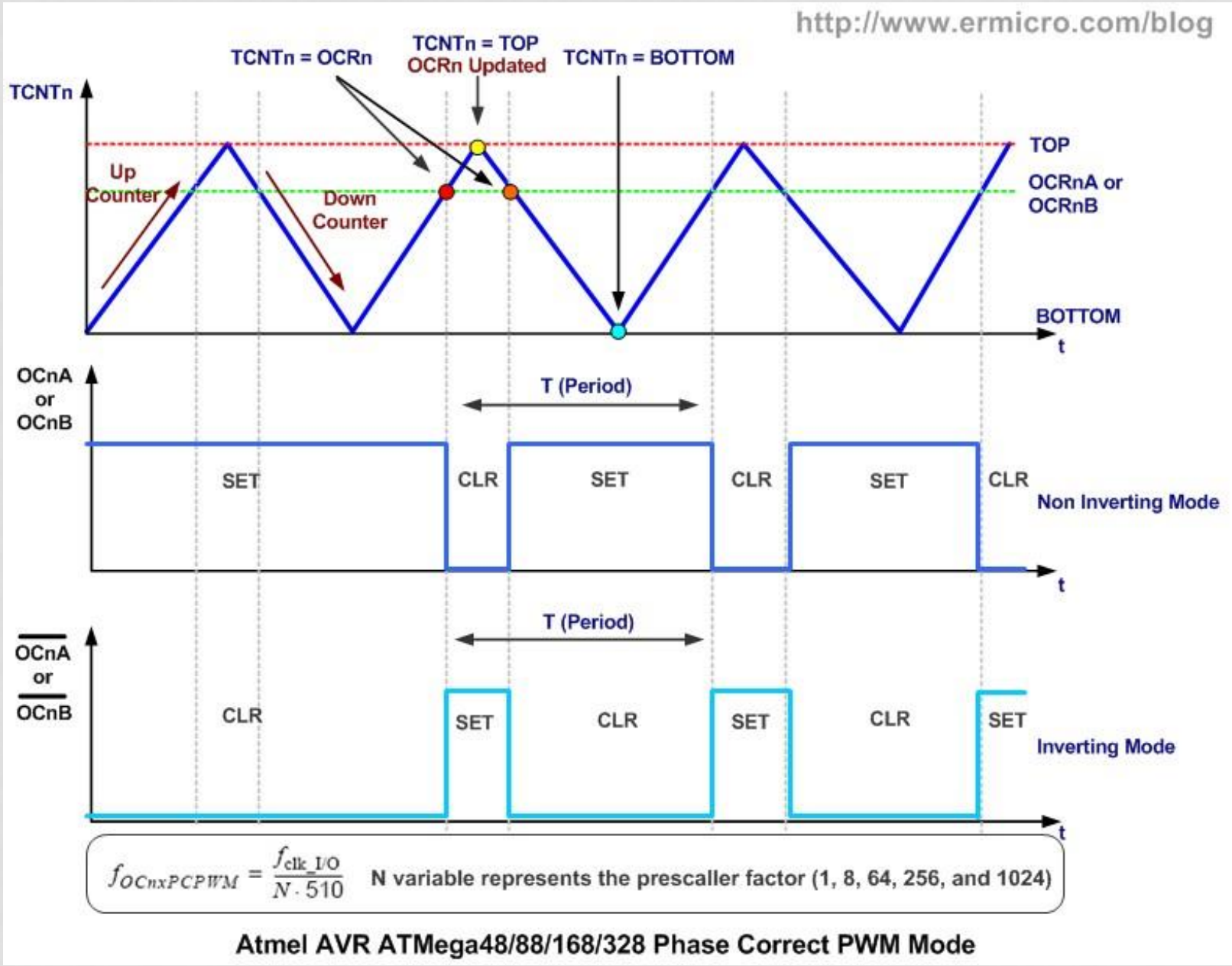
CONFIGURATIE PWM

- TCCR0A

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	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

Notes: 1. MAX = 0xFF
2. BOTTOM = 0x00



Atmel AVR ATMega48/88/168/328 Phase Correct PWM Mode

PWM VOORBEELD

```
#include <avr/io.h>
#define F_CPU 16E6  // used in _delay_ms, 16 MHz
#include <util/delay.h>

void init_timer()
{
   // Phase Correct PWM 8 Bit, Clear OCA0 on Compare Match
   // Set on TOP
   TCCR0A = (1 << WGM00) | (1 << COM0A1);
   // prescale = 64, fPWM = fCPU /(N*510)= 16E6/(64*510) = 490 Hz
   TCCR0B = (1 << CS01) | (1 << CS00);
   // init PWM value
   OCR0A = 0;
}

void init_port()
{
   // OC0A = PD6 is PWM output
   DDRD=0xff;     // set port D as output
}
```

PWM VOORBEELD

```
int main(void)
{
   uint8_t pwm = 0;
   init_timer();
   init_port();
   while(1) {
      OCR0A = pwm;
      pwm++;
      _delay_ms(15); // period = 256 * 15ms = 4s
   }
}
```

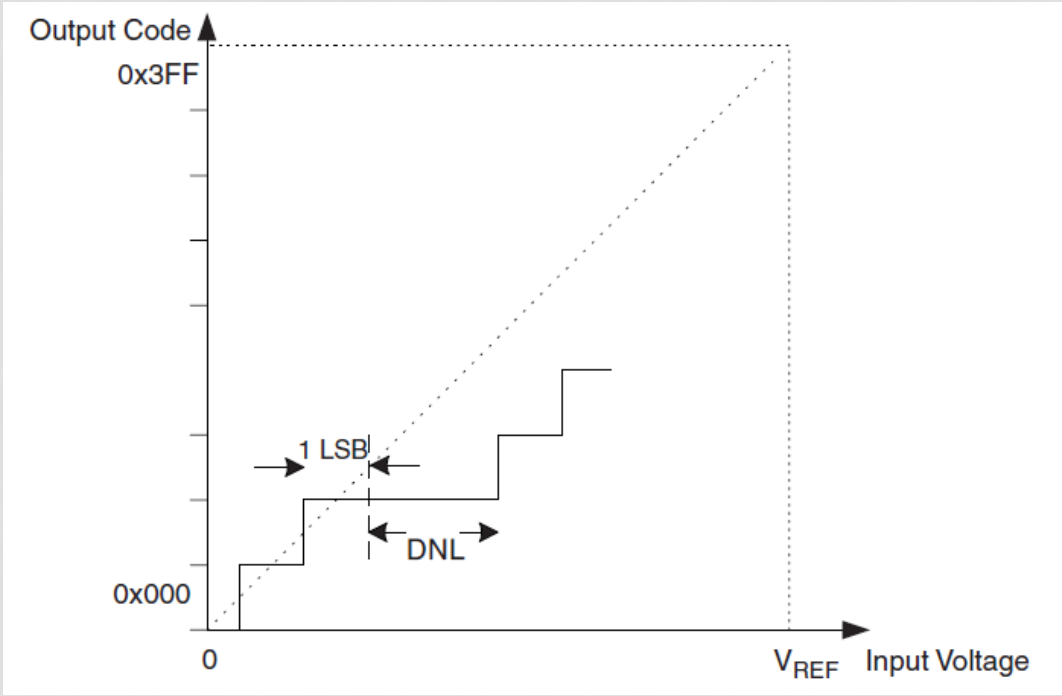
AGENDA

- PWM
- **ADC**
- een co-operatieve scheduler
- state diagram

ADC

- converteert een analoge (continue) ingangsspanning naar een getal van 10 bits
- dus analoge ingang van 0V - Vcc (5V) wordt verdeeld over 1024 stapjes

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$



ADC

- resolutie : 10 bit
- ingangsspanning : 0V - AREF (AREF = Vcc tenzij je Aref input gebruikt)
- output in ADCH+ADCL
  - 10 bit output verspreid over twee 8 bit registers
  - default onderste 8 bit in ADCL
  - links uitvullen (left align) plaatst de bovenste 8 bit in ADCH (0..255), en 2 bit in ADCL (negeren)
- prescaling zodanig dat ADC clock 50..200 kHz
  - systeem klok 16 MHz, prescaling = 128 geeft ADC klok 125 kHz

ADC

- start conversie : set ADSC-bit in ADCSRA
  - conversie duurt 13 ADC cycles
- 6 kanalen (6 ingangen poort C)
  - vb. ADC0 = PC0

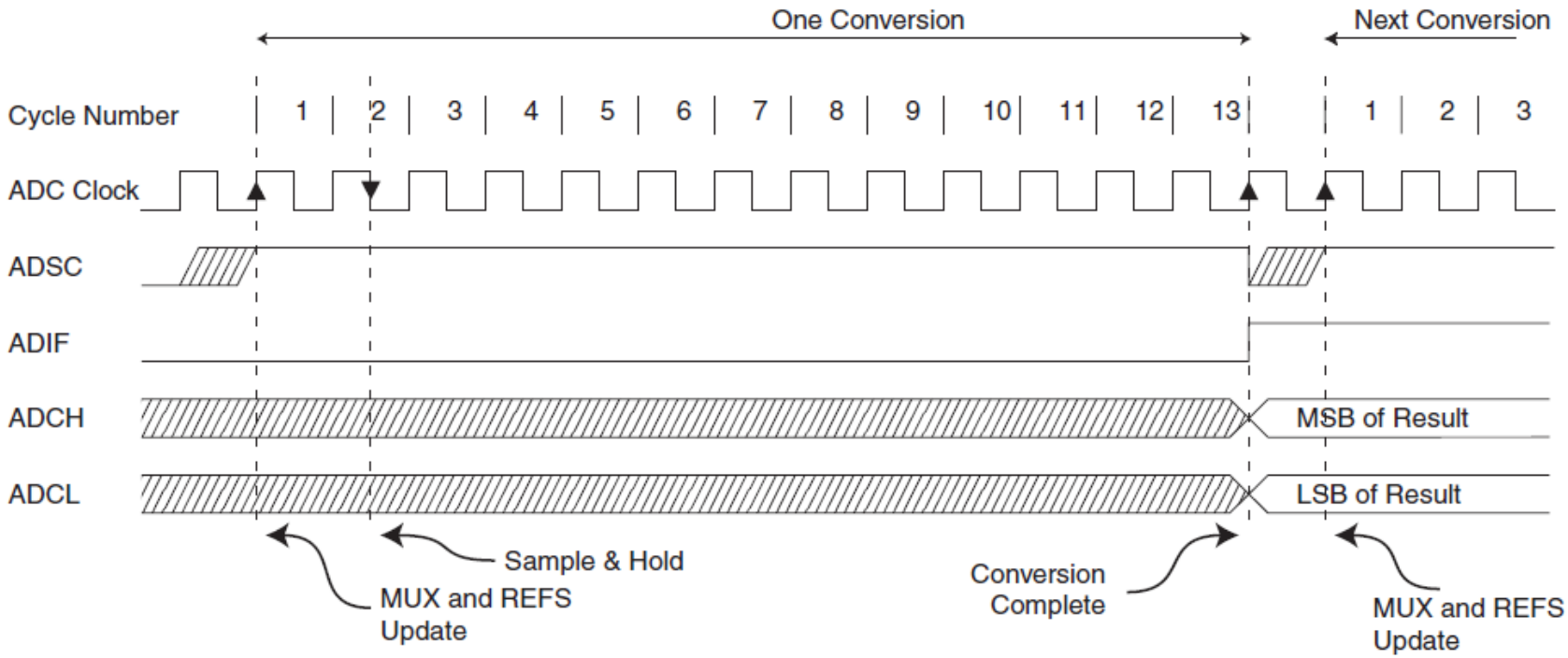
ADC VOORBEELD

```
void init_adc()
{
  // ref=Vcc, left adjust the result (8 bit resolution),
  // select channel 0 (PC0 = input)
  ADMUX = (1<<REFS0)|(1<<ADLAR);
  // enable the ADC & prescale = 128
  ADCSRA = (1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
}

uint8_t get_adc_value()
{
  ADCSRA |= (1<<ADSC); // start conversion
  loop_until_bit_is_clear(ADCSRA, ADSC);
  return ADCH; // 8-bit resolution, left adjusted
}
```

ADC CONVERSIE

Figure 102. ADC Timing Diagram, Single Conversion



AGENDA

- PWM
- ADC
- **een co-operatieve scheduler**
- state diagram

SCHEDULING

- wat als je programma 5 taken "tegelijk" moet uitvoeren ?
  - temperatuur lezen (ADC)
  - knopjes lezen
  - UART lezen
  - UART zenden
- het toewijzen van taken aan de CPU = scheduling

SCHEDULING

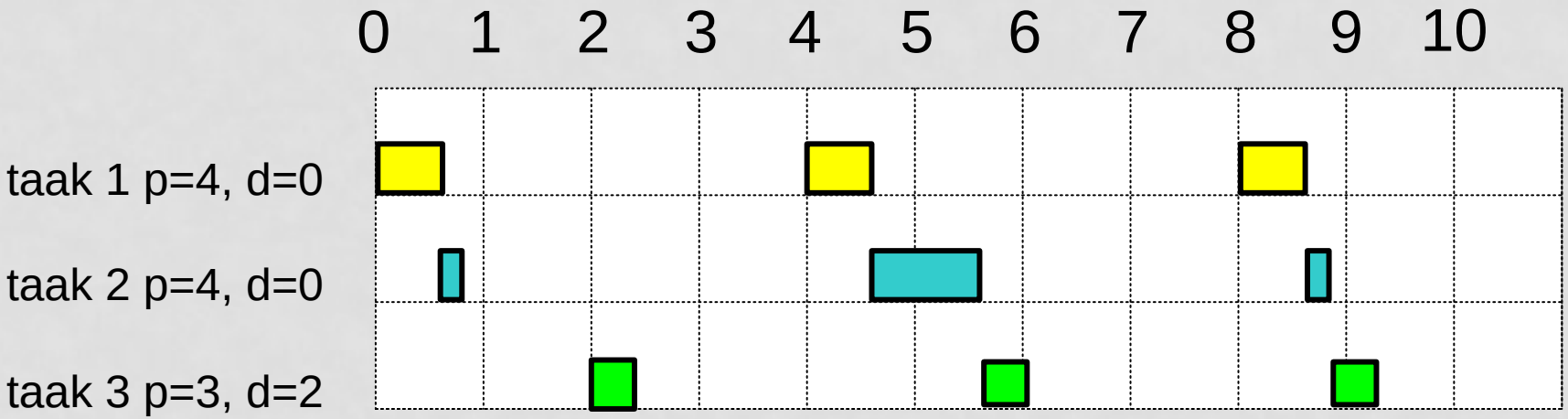
- timer-triggered : er is maar één interrupt, namelijk de timer interrupt
- periodiek : elke periode een event
- co-operatief : taken worden niet onderbroken
  - een taak wordt nooit onderbroken maar altijd geheel uitgevoerd
  - m.a.w. single-tasking (één taak tegelijk)
- voordelen :
  - geen shared data problemen (één ISR)
  - geen context switching
  - eenvoudig, betrouwbaar en overzichtelijk

CO-OPERATIEF EN PRE-EMPTIEF

- co-operatief : geen onderbreking taken

- pre-emptief : taak kan onderbroken worden

TTC : TIMER TRIGGERED & CO-OPERATIEF



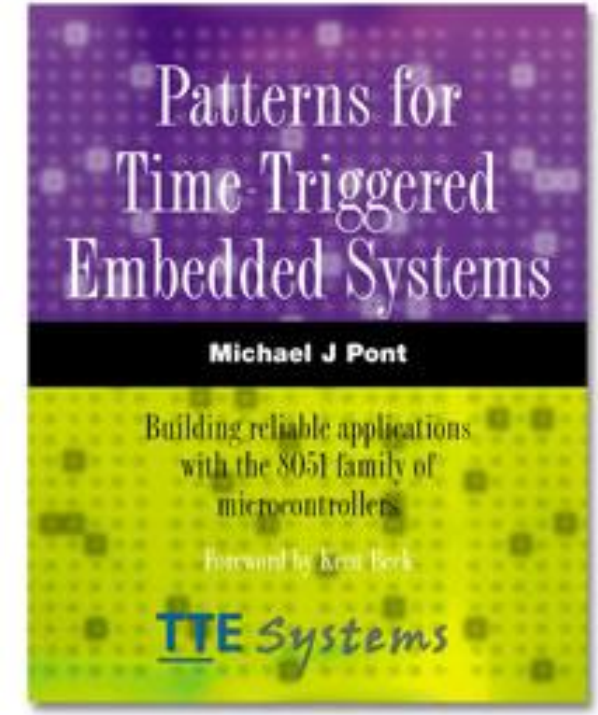
- p=periode, d=delay
- stel p1 = 15ms en p2 = 20ms, wat is de optimale timer periode ?

FUNCTION QUEUE SCHEDULING

- idee : zet taken in een lijst en loop elke timer tick door de lijst
- elke taak kan een eigen prioriteit krijgen

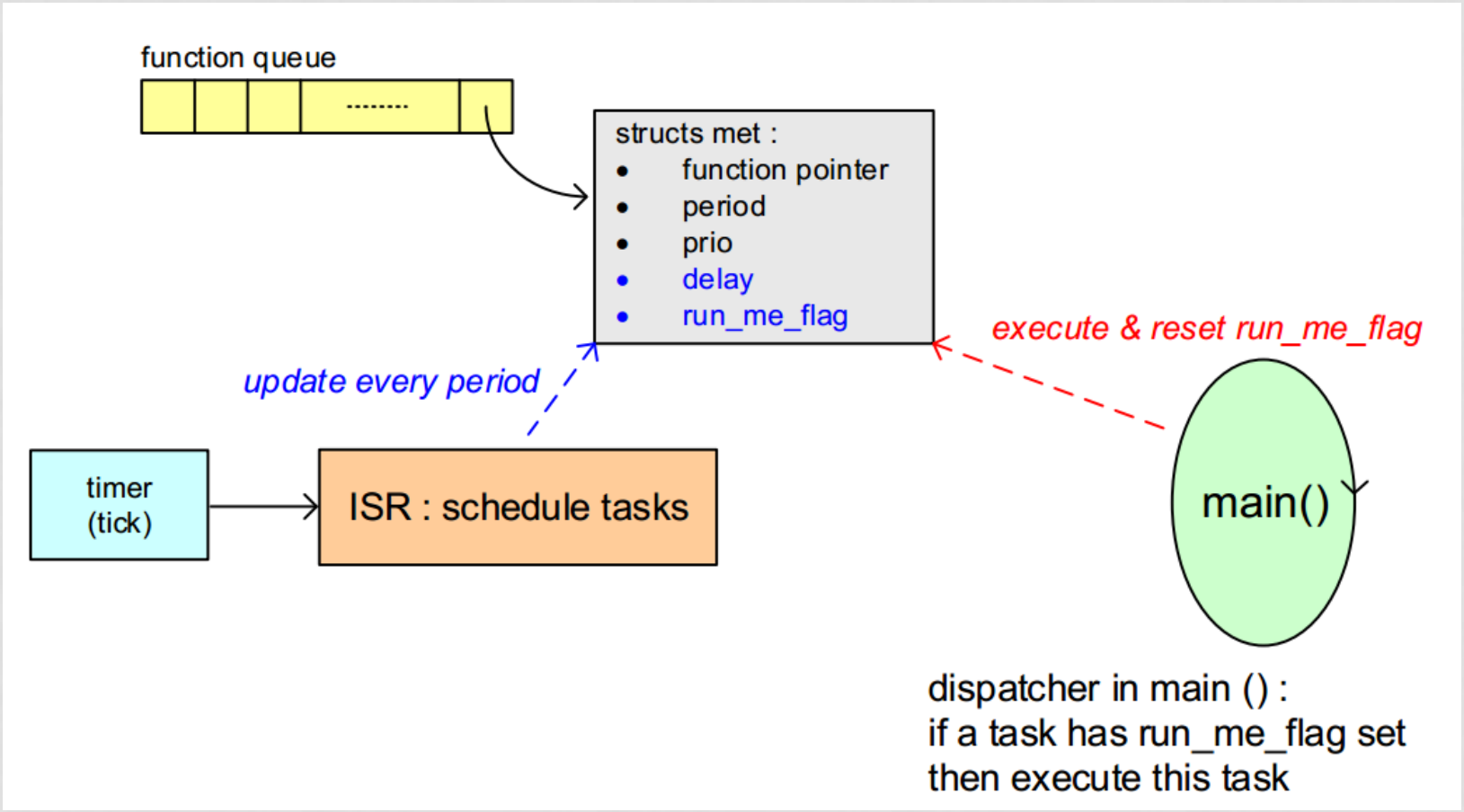
- is 'starvation' mogelijk ?
- hoe lang moet taak met hoogste prioriteit maximaal wachten ?

FUNCTION QUEUE SCHEDULING



boek (1000 blz):
http://vili.pmmf.hu/~zamek/c/8751/pttes_2010_07a.pdf

FUNCTION QUEUE SCHEDULING



```
191    This is the scheduler ISR.  It is called at a rate
192    determined by the timer settings in SCH_Init_T1().
193
194  -*------------------------------------------------------------------*/
195
196  ISR(TIMER1_COMPA_vect)
197  {
198     unsigned char Index;
199     for(Index = 0; Index < SCH_MAX_TASKS; Index++)
200     {
201        // Check if there is a task at this location
202        if(SCH_tasks_G[Index].pTask)
203        {
204           if(SCH_tasks_G[Index].Delay == 0)
205           {
206              // The task is due to run, Inc. the 'RunMe' flag
207              SCH_tasks_G[Index].RunMe += 1;
208
209              if(SCH_tasks_G[Index].Period)
210              {
211                 // Schedule periodic tasks to run again
212                 SCH_tasks_G[Index].Delay = SCH_tasks_G[Index].Period;
213                 SCH_tasks_G[Index].Delay -= 1;
214              }
215           }
216           else
217           {
218              // Not yet ready to run: just decrement the delay
219              SCH_tasks_G[Index].Delay -= 1;
220           }
221        }
222     }
223  }
```

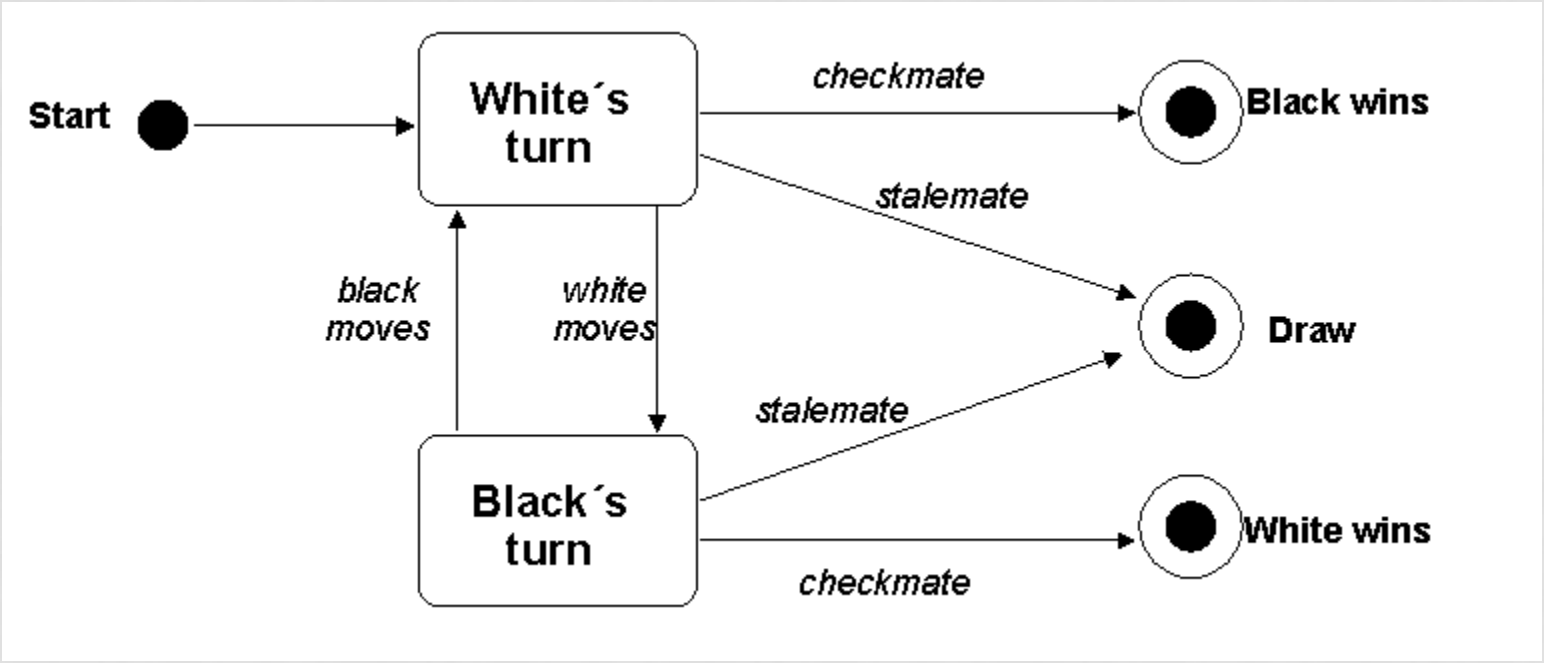
```
11    SCH_Dispatch_Tasks()
12
13    This is the 'dispatcher' function.  When a task (function)
14    is due to run, SCH_Dispatch_Tasks() will run it.
15    This function must be called (repeatedly) from the main loop.
16
17  -*---------------------------------------------------------------*/
18
19  void SCH_Dispatch_Tasks(void)
20  {
21     unsigned char Index;
22
23     // Dispatches (runs) the next task (if one is ready)
24     for(Index = 0; Index < SCH_MAX_TASKS; Index++)
25     {
26        if((SCH_tasks_G[Index].RunMe > 0) && (SCH_tasks_G[Index].pTask != 0))
27        {
28           (*SCH_tasks_G[Index].pTask)();  // Run the task
29           SCH_tasks_G[Index].RunMe -= 1;   // Reset / reduce RunMe flag
30
31           // Periodic tasks will automatically run again
32           // - if this is a 'one shot' task, remove it from the array
33           if(SCH_tasks_G[Index].Period == 0)
34           {
35              SCH_Delete_Task(Index);
36           }
37        }
38     }
39  }
```

***co-operatief*, geen onderbreking**

AGENDA

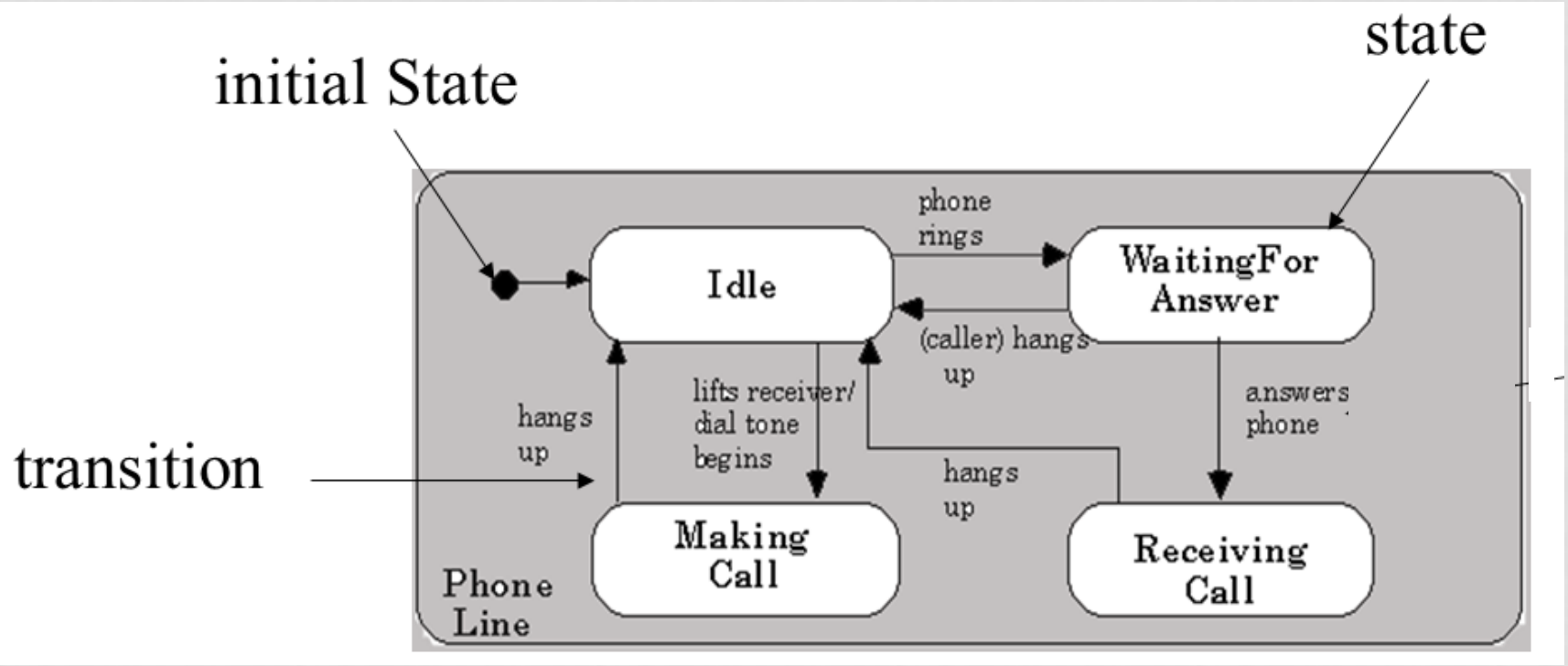
- PWM
- ADC
- een co-operatieve scheduler
- **state diagram**

PLAYING CHESS

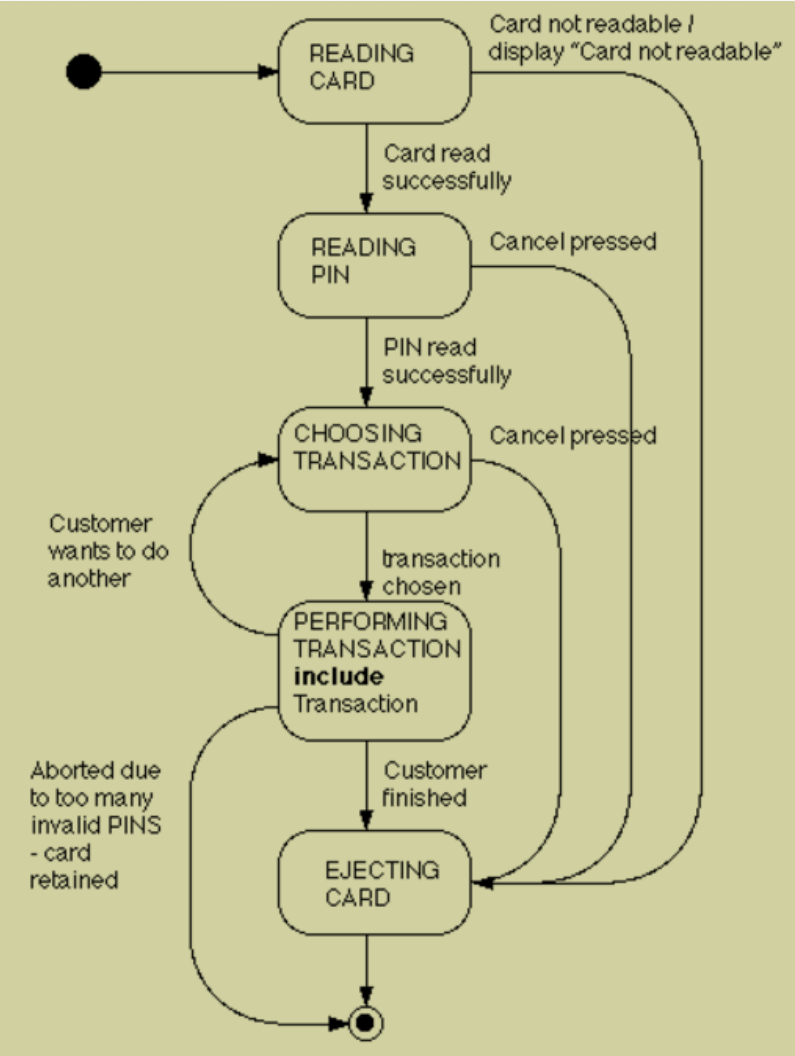


- state = iets abstracts
- symbolen begin & end state
- een transitie is een conditie, geen actie (het is *geen* AD)

PHONE CALL


initial State
state
transition
Idle
WaitingFor Answer
Making Call
Receiving Call
Phone Line
phone rings
(caller) hangs up
lifts receiver/ dial tone begins
answers phone
hangs up
hangs up

ATM MACHINE


READING CARD
Card not readable / display "Card not readable"
Card read successfully
READING PIN
Cancel pressed
PIN read successfully
CHOOSING TRANSACTION
Cancel pressed
Customer wants to do another
transaction chosen
PERFORMING TRANSACTION include Transaction
Aborted due to too many invalid PINS - card retained
Customer finished
EJECTING CARD