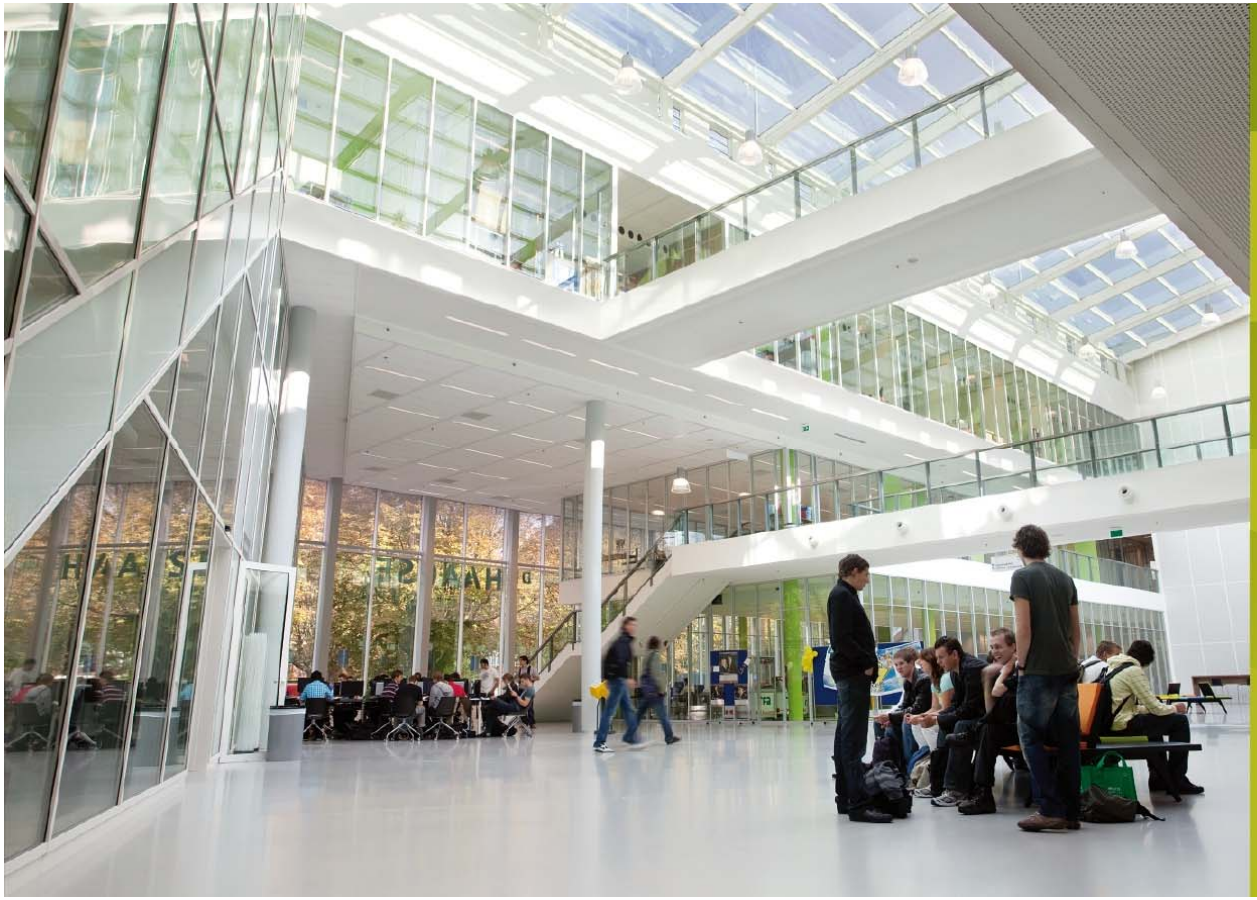


Academie voor Technology, Innovation & Society Delft
Academie voor ICT & Media

Microcontrollers

Week 5 – Introductie microcontroller
Jesse op den Brouw (met dank aan Ben Kuiper)
INLMIC/2018-2019

HAAGSE HOGESCHOOL

Week 5

- Interrupts
- Timers

Interrupts

- Stel dat een programma snel moet reageren op een *event* (gebeurtenis) van buitenaf.

- Dit kan worden opgelost door de ingang te *pollen* en indien het event optreedt, kan direct met de afhandeling worden begonnen.

- Dit pollen kost tijd en de processor kan geen andere taken uitvoeren, zie code.

```
loop:  in     r16,PINA
       andi   r16,0x01
       brne   loop
```

Interrupts

- Slimmer is om de hardware een signaal aan de processor te geven dat het event heeft plaatsgevonden, zodat de processor direct actie kan ondernemen.

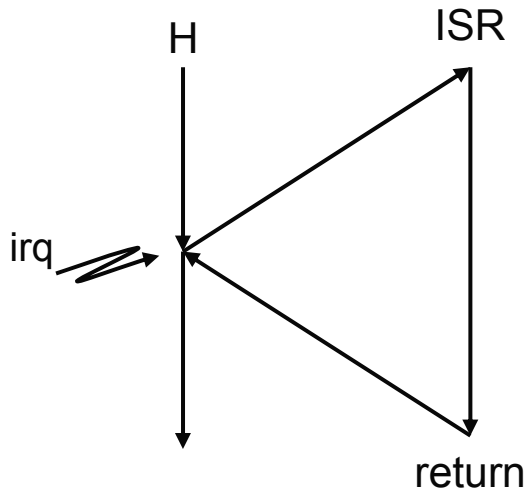
- Het lopende programma wordt dan *geïnterrumpeerd* (onderbroken).

- Dit wordt een *interrupt* genoemd. Een aanvraag voor een onderbreking (het signaal) wordt een *interrupt request* (IRQ) genoemd.

Interrupts

- Nadat een interrupt request door de processor is herkend, wordt het lopende programma onderbroken (de huidige instructie wordt afgemaakt), en wordt een interrupt service routine (ISR) uitgevoerd.

- De ISR handelt dan de interrupt af. Daarna gaat de processor verder waar het gebleven was.



Interrupts

- De AVR kent vele interrupts:

  Seriële interface – karakter verzonden/ontvangen
  ADC – conversie klaar
  Timer – timer overflow, timer compare match
  EEPROM – klaar met opslaan data
  Externe interrupts – signalering
  …

- Elke interrupt heeft een eigen ISR.

Interrupts

- ISR's lijken veel op gewone subroutines, maar er zijn verschillen:
- Een ISR wordt gestart door een interrupt, niet een instructie*.
- Direct nadat de interrupt is herkend, wordt de Global Interrupt Enable vlag (I-flag) in het Status Register uitgezet (waarom?).
- Voor terugkeer moet de instructie `reti` (Return From Interrupt) gebruikt worden. Dit zet de I-flag weer aan.

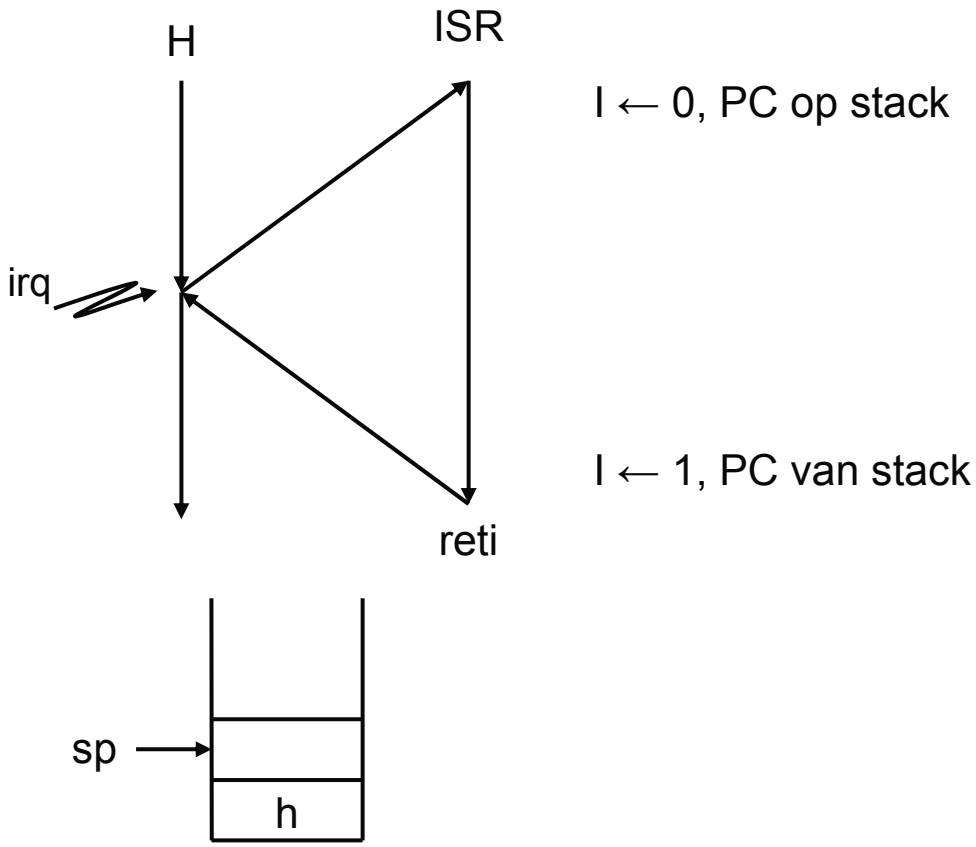
The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

* er zijn processoren die instructies hebben die software interrupts genereren, zoals de Intel x86-familie.

Interrupts

- Het terugkeeradres wordt, net als bij subroutines, op de stack gezet.



Interrupts

- Twee instructies beinvloeden direct de I-flag.

  sei - set interrupt enable flag

  cli - clear interrupt enable flag

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
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	

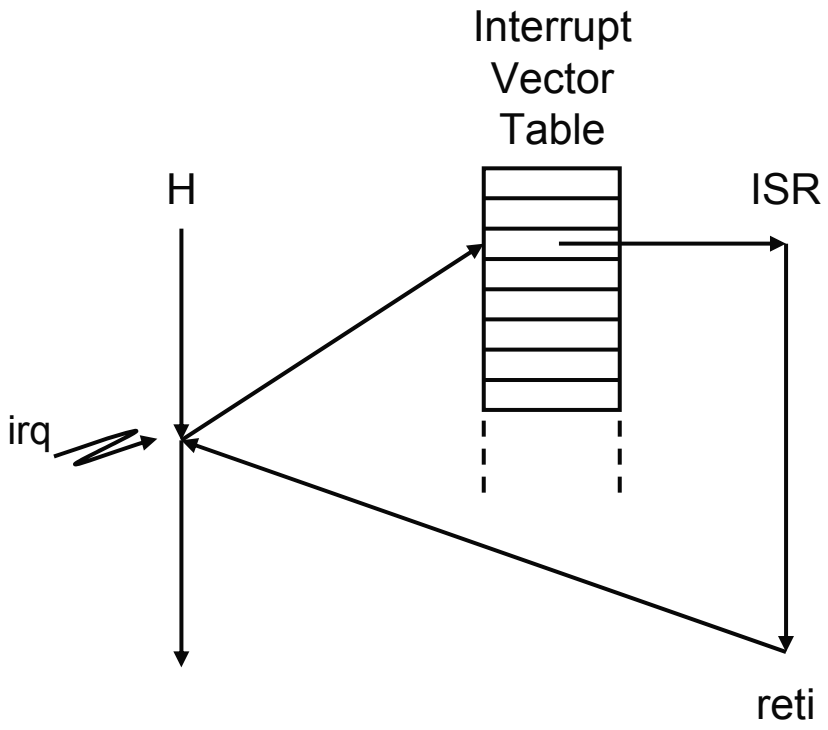
DE HAAGSE HOGESCHOOL

Interrupt Vector

- Aan elke interrupt is een *Interrupt Vector* gekoppeld. Dit is een adres waar naartoe gesprongen wordt als een interrupt is herkend. Elke vector is twee words (4 bytes) groot.
- De *Interrupt Vector Table* bevat alle Interrupt Vectors en ligt aan het begin van de Flash-ROM (adres 0x000 – 0x029 bij ATmega32).
- Na het herkennen van de interrupt wordt de PC dus geladen met het adres van de Interrupt Vector die bij de interrupt hoort.
- Meestal wordt hier een `jmp` naar de eigenlijke ISR geplaatst.

Interrupt Vector Table

- Het uitvoeren van een ISR verloopt in twee stappen: eerst een sprong naar de interrupt vector en daarna naar de eigenlijke ISR.



I ← 0, PC op stack

I ← 1, PC van stack

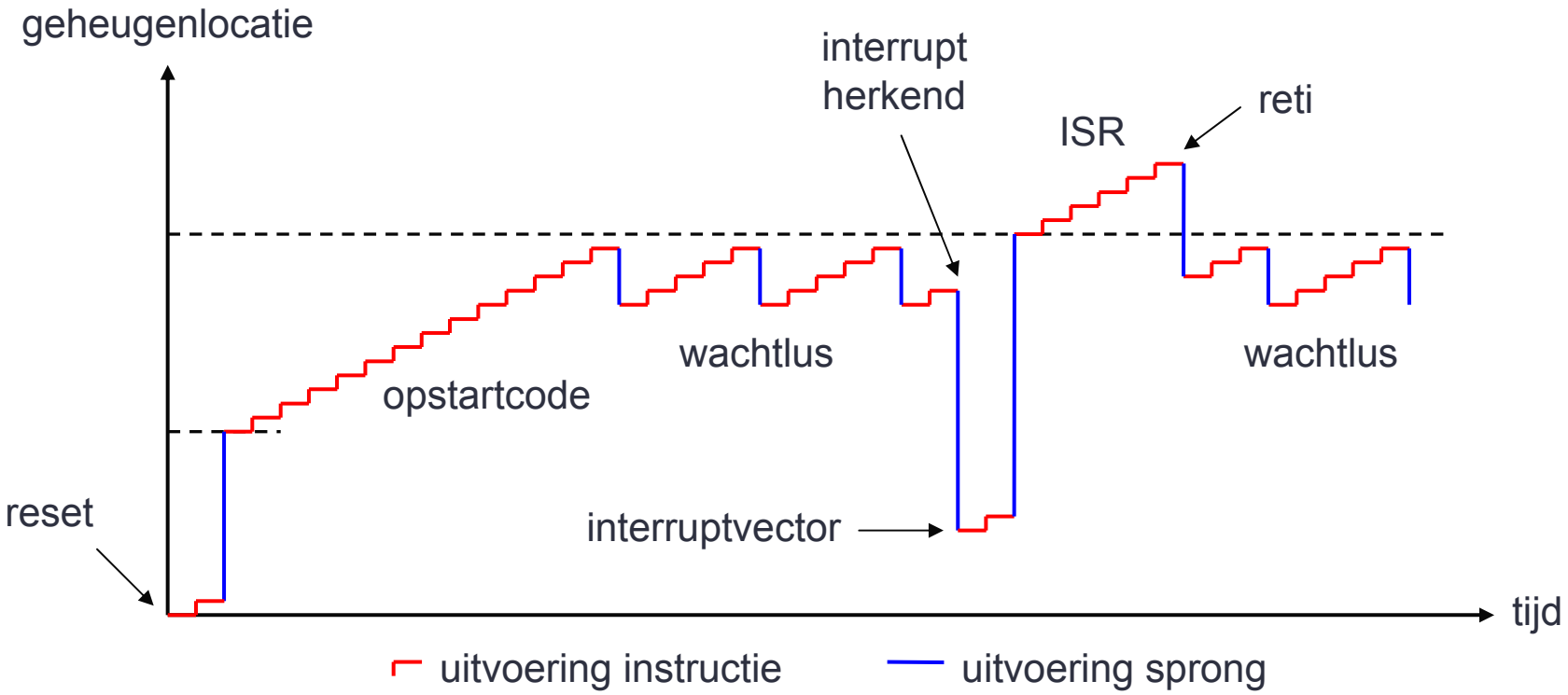
DE HAAGSE HOGESCHOOL

Interrupt Vector Table

- Merk op dat reset ook in deze tabel staat.
- Reset heeft de hoogste prioriteit, daarna INT0, etc.
- Eerste vrije ROM-adres is 0x02a.

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$002	INT0	External Interrupt Request 0
3	$004	INT1	External Interrupt Request 1
4	$006	INT2	External Interrupt Request 2
5	$008	TIMER2 COMP	Timer/Counter2 Compare Match
6	$00A	TIMER2 OVF	Timer/Counter2 Overflow
7	$00C	TIMER1 CAPT	Timer/Counter1 Capture Event
8	$00E	TIMER1 COMPA	Timer/Counter1 Compare Match A
9	$010	TIMER1 COMPB	Timer/Counter1 Compare Match B
10	$012	TIMER1 OVF	Timer/Counter1 Overflow
11	$014	TIMER0 COMP	Timer/Counter0 Compare Match
12	$016	TIMER0 OVF	Timer/Counter0 Overflow
13	$018	SPI, STC	Serial Transfer Complete
14	$01A	USART, RXC	USART, Rx Complete
15	$01C	USART, UDRE	USART Data Register Empty
16	$01E	USART, TXC	USART, Tx Complete
17	$020	ADC	ADC Conversion Complete
18	$022	EE_RDY	EEPROM Ready
19	$024	ANA_COMP	Analog Comparator
20	$026	TWI	Two-wire Serial Interface
21	$028	SPM_RDY	Store Program Memory Ready

Uitvoering interrupt in tijd en ROM-locatie



Noot: interruptresponstijd niet meegenomen

DE HAAGSE HOGESCHOOL

Interrupts

- Omdat een interrupt op een willekeurig moment kan plaatsvinden, zullen de vlaggen en registers die in de ISR gebruikt worden, moeten worden opgeslagen op de stack.

```
ISR:   push   r16           ; push R16
       in     r16,SREG      ; SREG in I/O memory…
       push   r16           ; … so push flags via R16
       …
       pop    r16           ; pop flags via R16
       out    SREG,r16
       pop    r16           ; pop R16
       reti                 ; return
```

Algemene opbouw programma

```
.org 0x000                              ; reset vector
     rjmp    reset
.org 0x016                              ; timer0 overflow IV
     rjmp    timer_isr
.org 0x02a                              ; first address after IVT

reset:
     ldi     r16,low(RAMEND)            ; Set up stackpointer
     out     SPL,r16
     ldi     r16,high(RAMEND)
     out     SPH,r16
     … specifieke hardware setup zoals timers, USART, …
     sei                                ;enable interrupts globally
     … hoofdprogramma

timer_isr:                              ; ISR code
     reti                               ; return from interrupt
```

Interrupt en opbouw programma

- Zorg ervoor dat het uitvoeren van de ISR weinig tijd in beslag neemt! (waarom?)

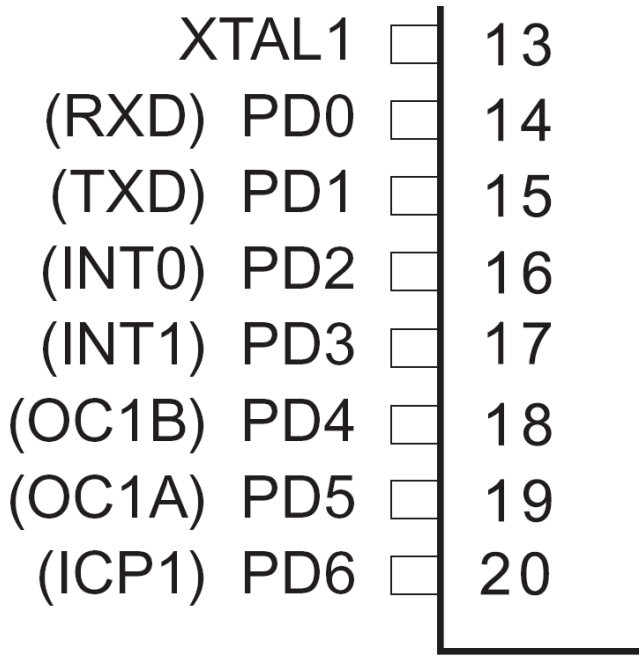
Kritieke code

- Als een stuk code zonder onderbreking moet worden uitgevoerd, zogenaamde kritieke code, dan moet deze code worden afgeschermd door de interrupts te blokkeren.

```
.....
cli
lds    r16,0x0060    ; kritieke code
inc    r16           ; verhoog inhoud
sts    0x0060,r16    ; geheugenplaats
sei
....
```

Externe interrupts

- INT0, 1 en 2 zijn extern te activeren interrupts.
- Ze kunnen reageren op een niveau of een flank (INT2 alleen op flank).
- Om ze te gebruiken moeten diverse I/O registers geprogrammeerd worden.
- Alleen INT0 en INT1 worden besproken.



General Interrupt Control Register

- De General Interrupt Control Register (GICR) heeft drie bits waarmee de INT's kunnen worden geactiveerd.
- Een logische 1 in het betreffende bit activeert de INT.
- De I-flag moet 1 zijn om interrupt ook daadwerkelijk te herkennen.

Bit	7	6	5	4	3	2	1	0
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE
Read/Write	R/W	R/W	R/W	R	R	R	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

MCU Control Register

- De MCU Control Register (MCUCR) bepaalt onder andere hoe op een interrupt moet worden gereageerd.

- De bits ISC11 en ISC10 geven dit aan voor INT1, de bits ISC01 en ISC00 voor INT0.

Bit	7	6	5	4	3	2	1	0
	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
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

DE HAAGSE HOGESCHOOL

ISC bits

- In onderstaande tabel zijn de combinaties voor INT0 gegeven, voor INT1 zijn deze hetzelfde.

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Voorbeeldprogramma voor gebruik INT0

```
.org 0x000                ; reset vector
   rjmp   reset
.org 0x002                ; ISR vector for INT0
   rjmp   int0_isr
.org 0x02a                ; main program

   ; set up stack pointer
reset:
   ldi    temp,low(RAMEND)
   out    SPL,temp
   ldi    temp,high(RAMEND)
   out    SPH,temp
```

Instellen registers

```
    ldi    temp,0b00000011     ; set up rising edge
    out    MCUCR,temp

    ldi    temp,0b01000000     ; set up INT0
    out    GICR,temp

    sei                        ; enable interrupts

loop:
    rjmp   loop                ; forever....
```

ISR voor INT0

```
int0_isr:
    push   temp                 ; save regs and flags
    in     temp,SREG
    push   temp

    nop                         ; do something

    pop    temp                 ; restore regs and flags
    out    SREG,temp
    pop    temp
    reti                        ; return from interrupt
```

General Interrupt Flag Register

- De General Interrupt Flag Register (GIFR) geeft aan of er een IRQ *pending* (hangende) is. Dit werkt alleen bij flankgevoelige interrupts.

- Een flag wordt gereset als de bijbehorende ISR wordt uitgevoerd. Een flag kan worden gereset door een logische 1 te schrijven naar het betreffende bit.

Bit	7	6	5	4	3	2	1	0
	INTF1	INTF0	INTF2	–	–	–	–	–
Read/Write	R/W	R/W	R/W	R	R	R	R	R
Initial Value	0	0	0	0	0	0	0	0

DE HAAGSE HOGESCHOOL

Timer

- In veel applicaties moet gewacht worden, bijvoorbeeld elke seconde een meting doen of een waterklep 1 seconde openen.
- Het meten van deze tijd (wachten) is op te lossen door middel van de bekende wachtlus.
- De processor kan dan echter geen andere taken uitvoeren en dat is in veel situaties wél gewenst. Denk hierbij aan Real Time systemen en besturingssystemen.
- Beter is deze tijd hardwarematig te meten.

Timer

- Een *timer* kan dan uitkomst bieden.
- Een timer is niets anders dan een teller die op iedere klokpuls, bijvoorbeeld van de oscillator, wordt verhoogd.
- Na een bepaald aantal klokpulsen heeft de timer de hoogste stand bereikt en begint weer opnieuw.
- Er wordt dan een overflow-signaal afgegeven. Dit signaal genereert, indien geactiveerd, een interrupt.

Counter

- Dezelfde schakeling kan ook gebruikt worden om extern aangeboden pulsen te tellen.

- Dit wordt dan een *counter* genoemd.

Timer/Counter 0

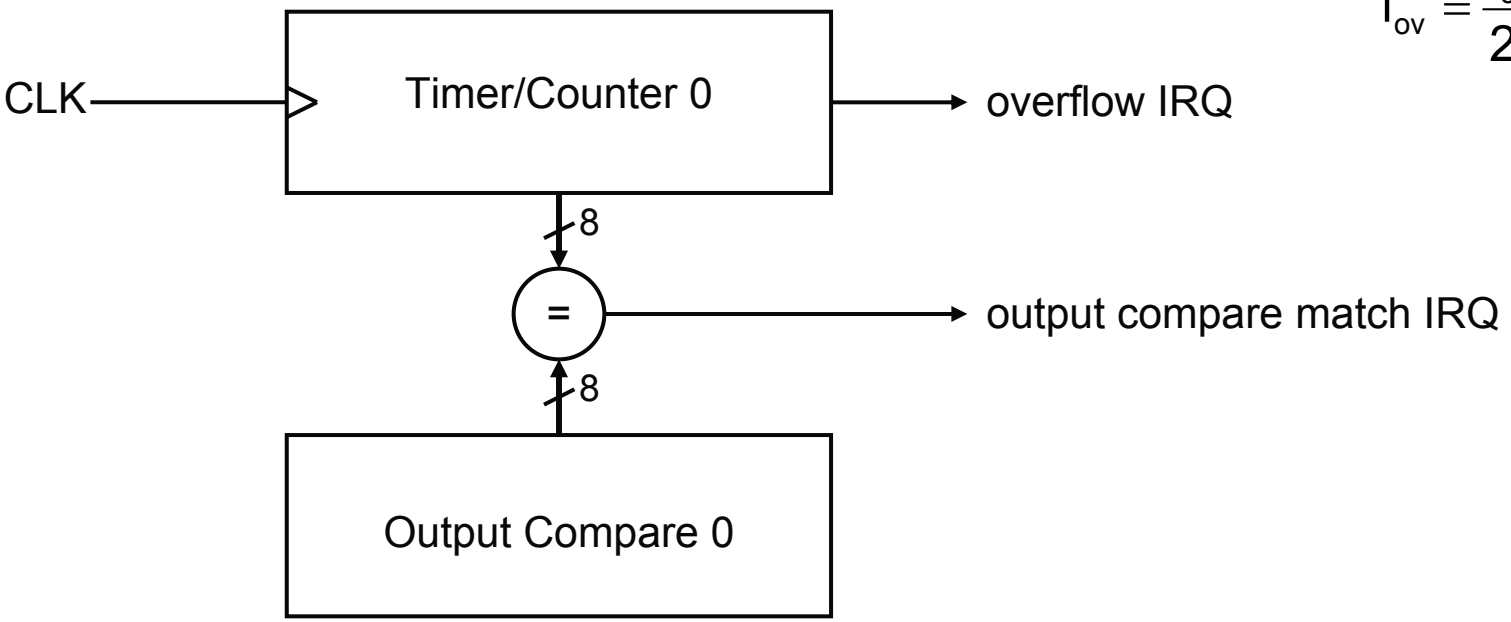
- Alleen Timer/Counter 0 (T/C0) wordt besproken.
- Dit is een 8 bits timer met de volgende werkmodi:
  - Normaal: telt van 0 tot 255 en gaat dan weer beginnen bij 0
  - CTC (Clear Timer on Compare Match): telt van 0 tot een bepaald maximum en gaat dan weer beginnen bij 0
  - Fast PWM: snelle pulsbreedtemodulatie, wordt niet besproken *)
  - Phase Correct PWM: fasecorrect pulsbreedtemodulatie, wordt niet besproken *)

*) wordt bij MICPRG besproken.

Timer/Counter 0

- Timer/Counter 0 telt de klokpulsen. Als T/C0 van 255 naar 0 gaat, wordt een overflow IRQ gegenereerd. Als T/C0 gelijk is aan de OCR, wordt een output compare match IRQ gegenereerd.

$$f_{ov} = \frac{f_{clk}}{2^8}$$

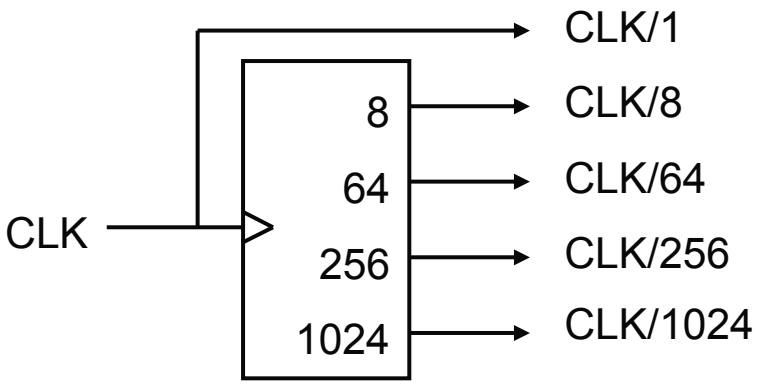


Timer/Counter 0

- Aangezien de klokfrequentie vrij groot is en het aantal timer-bits klein, zal T/C0 zeer snel op het maximum zitten en genereert zodanig heel vaak een IRQ.

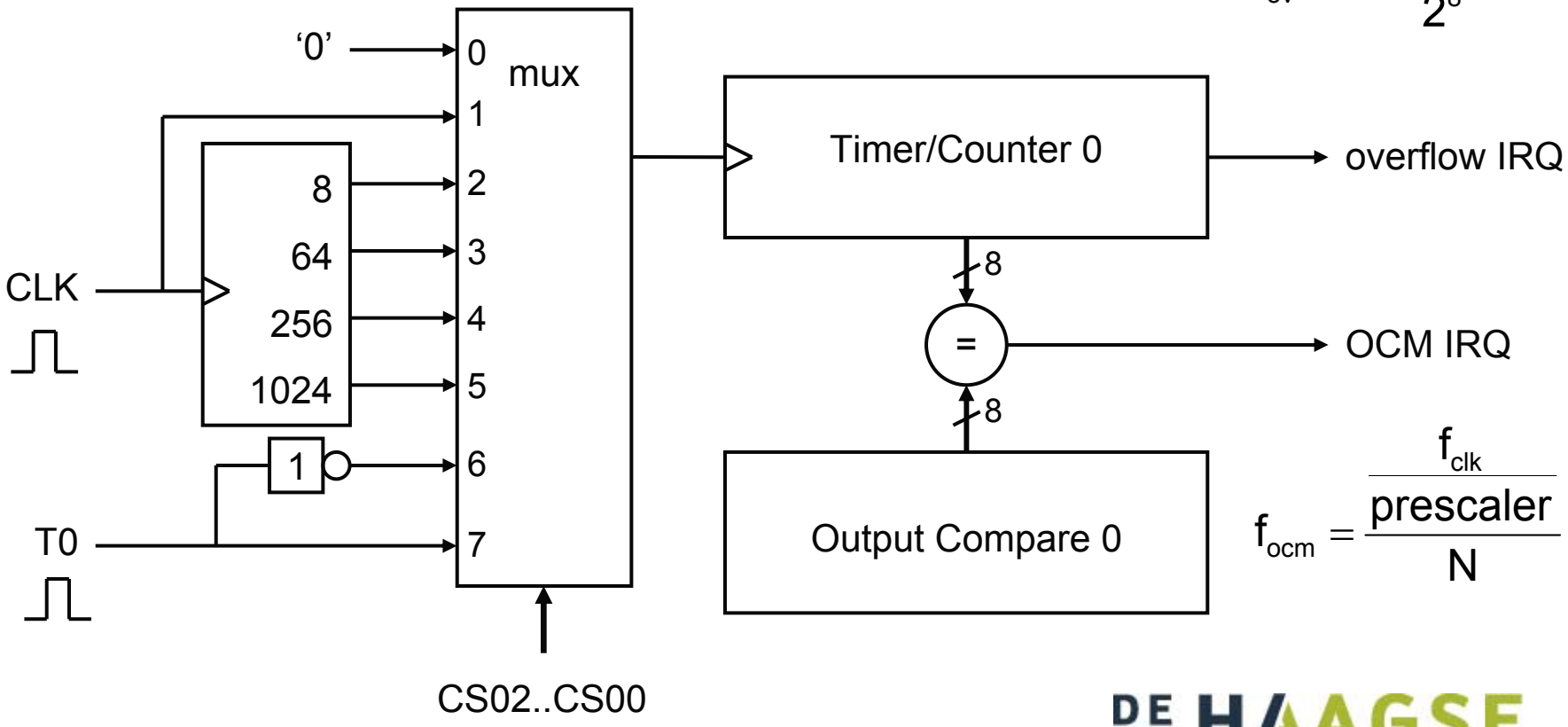
- Een *prescaler* is een digitale schakeling die het aangeboden (klok-) signaal ‘deelt’ door een getal, meestal een macht van 2.

- Hierdoor telt de T/C0 op een (veel) lagere frequentie.



Timer/Counter 0

- Hieronder het complete systeem (vereenvoudigd).



Timer/Counter Control Register

- De TCCR0 bestuurt T/C0. Hierin wordt onder andere de werkmodus, compare match output mode en klokbron opgegeven.

Bit	7	6	5	4	3	2	1	0
	FOC0	**WGM00**	**COM01**	**COM00**	**WGM01**	**CS02**	**CS01**	**CS00**
Read/Write	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

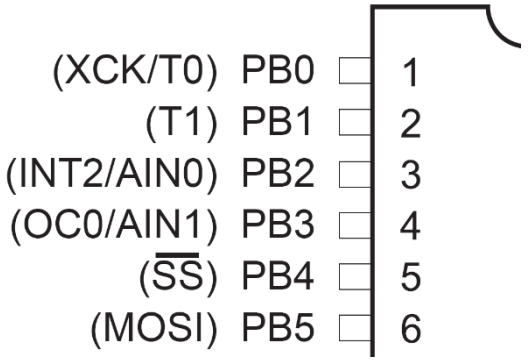
Waveform Generation Mode

- De bits WGM01 en WGM00 geven de werkmodus aan. Alleen Normal en CTC zijn nu van belang.

Mode	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
0	0	0	Normal	0xFF	Immediate	MAX
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	1	0	CTC	OCR0	Immediate	MAX
3	1	1	Fast PWM	0xFF	TOP	MAX

Compare Output Mode

- De bits COM01 en COM00 besturen de manier waarop output OC0 (PB3 bij de ATmega32) wordt aangestuurd.



COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Toggle OC0 on compare match
1	0	Clear OC0 on compare match
1	1	Set OC0 on compare match

Noot: deze tabel is voor Normal en CTC-werkmodus.

Noot: pin PB3 moet als output gedefinieerd worden.

DE HAAGSE HOGESCHOOL

Clock Source

- De bits CS02, CS01 en CS00 selecteren de klokbron.

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.

TCNT0 – OCR0

- TCNT0 bevat de huidige telstand.
- OCR0 bevat de waarde voor vergelijken bij CTC-werkmodus.
- Beide registers zijn te lezen en te schrijven.

Bit	7	6	5	4	3	2	1	0
	TCNT0[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

Bit	7	6	5	4	3	2	1	0
	OCR0[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

DE HAAGSE HOGESCHOOL

Timer/Counter Interrupt Mask Register

- Het TIMSK register bevat de bits OCIE0 en TOIE0.
- Indien OCIE0 op 1 wordt gezet, zal een interrupt worden gegenereerd indien er een Compare Match optreedt.
- Indien TOIE0 op 1 wordt gezet, zal een interrupt worden gegenereerd indien er een overflow optreedt.

Bit	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
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

Rekenvoorbeeld overflow

- Gegeven: f_{clk} = 3,6864 MHz
  prescaler = 1024
- Gevraagd: frequentie van de overflow
- Oplossing:

$$f_o = \frac{\frac{f_{clk}}{prescaler}}{256} = \frac{\frac{3686400}{1024}}{256} = 14,06\,25Hz$$

DE HAAGSE HOGESCHOOL

Rekenvoorbeeld OCM

- Genereer 100 Hz (= 10 ms), bijvoorbeeld voor een stopwatch of *time slice* voor een besturingssysteem.

- Gegeven: f_{clk} = 3,6864 MHz
  f_{ocm} = 100 Hz

- Gevraagd: de waarde voor de prescaler
  de waarde van OCR0

Rekenvoorbeeld OCM

- Eerst de formules herschrijven:

$$f_{ocm} = \frac{\frac{f_{clk}}{prescaler}}{N} = \frac{f_{clk}}{prescaler \cdot N}$$

$$prescaler \cdot N = \frac{f_{clk}}{f_{ocm}} = \frac{3686400}{100} = 36864$$

- Er is nu te kiezen tussen twee prescaler-waarden: 1024 en 256

Rekenvoorbeeld OCM

- Bij prescalerwaarde 1024 volgt:

$$N_{1024} = \frac{36860}{1024} = 36 \qquad OCR0_{1024} = N_{1024} - 1 = 35$$

- Bij prescalerwaarde 256 volgt:

$$N_{256} = \frac{36860}{256} = 144 \qquad OCR0_{256} = N_{256} - 1 = 143$$

- Welke is nauwkeuriger?

Voorbeeldcode

```
.org 0x000
           rjmp   reset

; The interrupt vector for T/C0 OCM
.org 0x014
           rjmp   tc0_ocm_isr

.org 0x02a
           ; Set up stack pointer
reset:     ldi          temp,low(RAMEND)
           out          SPL,temp
           ldi          temp,high(RAMEND)
           out          SPH,temp
```

Voorbeeldcode

```
; Set up OCR0 with value of 143
ldi          temp,143
out          OCR0,temp

; Set up TCCR0: CTC-waveform, 256 prescaler
ldi          temp,0b00001100
out          TCCR0,temp

; Set up OCM interrupt (OCIE0, bit 2)
ldi          temp,0b00000010
out          TIMSK,temp
```

Voorbeeldcode

```
            ; Free interrupts!!!
            sei

            ; Sit and wait
loop:       rjmp   loop

            ; The T/C0 OCM ISR
tc0_ocm_isr:
            nop
            reti
```

Screenshot code

```
.def    temp = r16

.org 0x000
        rjmp    reset

.org 0x014
        rjmp    tc0_ocm_isr

.org 0x02a
        ; Set up stack pointer
reset:  ldi     temp,low(RAMEND)
        out     SPL,temp
        ldi     temp,high(RAMEND)
        out     SPH,temp

        ; Set up OCR0 with value of 143
        ldi     temp,143
        out     OCR0,temp

        ; Set up TCCR0: CTC-waveform, 256 prescaler
        ldi     temp,0b00001100
        out     TCCR0,temp

        ; Set up OCM interrupt (OCIE0, bit 1)
        ldi     temp,0b00000010
        out     TIMSK,temp

        ; Free interrupts!!!
        sei;

        ; Sit and wait
loop:   rjmp    loop

        ; The T/C0 OCM ISR
tc0_ocm_isr:
        nop
        reti
```

Name	Value
Program Counter	0x00000014
Stack Pointer	0x085D
X Register	0x0000
Y Register	0x0000
Z Register	0x0000
Status Register	I T H S V N Z C
Cycle Counter	73732
Frequency	3,686 MHz
Stop Watch	20.001,09 µs
Registers	
R00	0x00
R01	0x00
R02	0x00
R03	0x00
R04	0x00
R05	0x00
R06	0x00
R07	0x00
R08	0x00
R09	0x00
R10	0x00
R11	0x00
R12	0x00
R13	0x00
R14	0x00
R15	0x00
R16	0x02
R17	0x00
R18	0x00
R19	0x00
R20	0x00
R21	0x00
R22	0x00
R23	0x00
R24	0x00
R25	0x00
R26	0x00
R27	0x00

Stand processor na twee interrupts

Zelf proberen

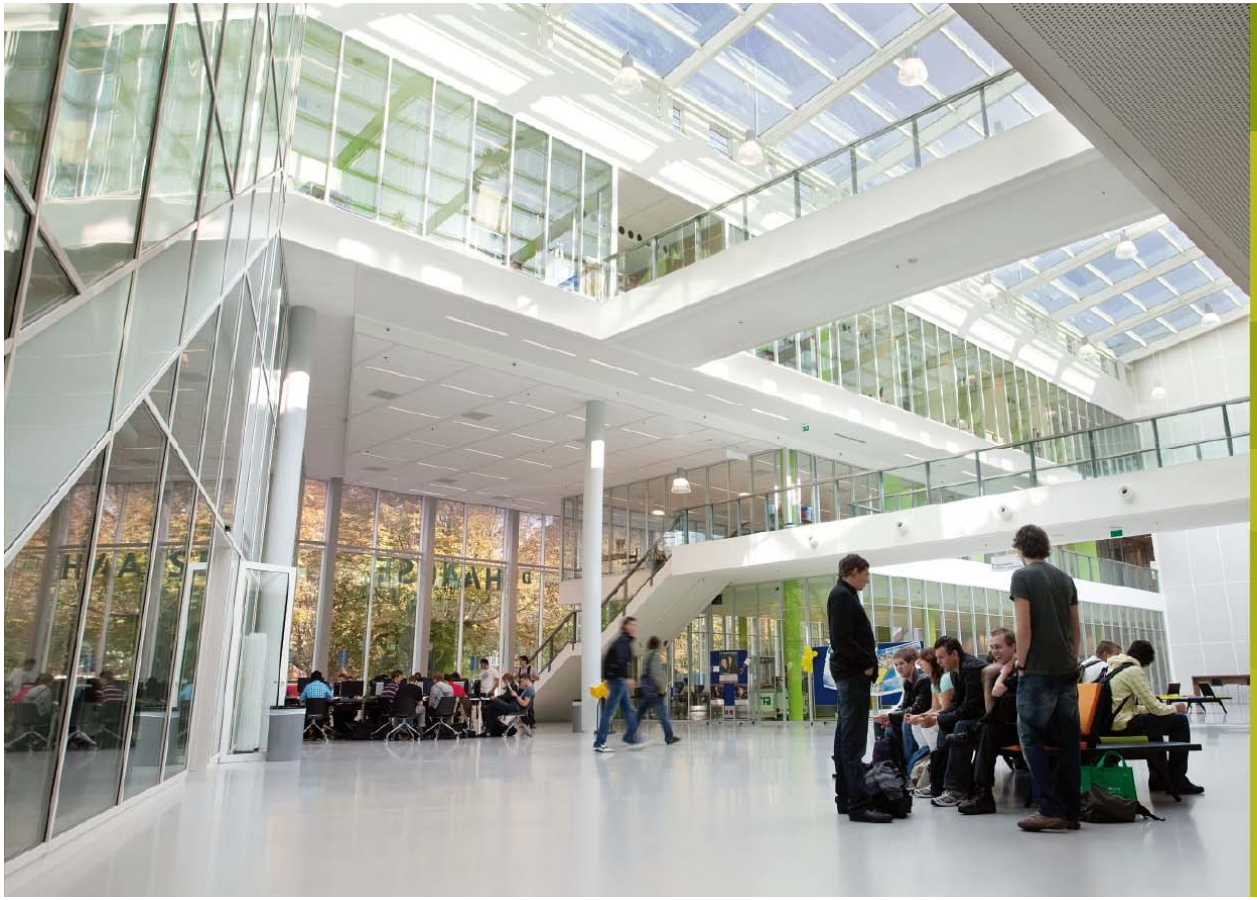
- Nu zelf eens proberen

- Gegeven: f_{clk} = 1,000 MHz
  f_{ocm} = 100 Hz

- Gevraagd: de waarde voor de prescaler
  de waarde van OCR0
  afwijking berekende en gebruikte waarden

Literatuur/leeswerk

- H11S1, H11S3, H11S4, H10S1 (t/m pag. 331), H10S2, H11S2
- Amerikaanse editie: H10S1, H10S3, H10S4, H9S1 (t/m pag. 331), H9S2, H10S2
- Amerikaanse editie: Benodigd voor practicumopdracht 6: H9S1 en H9S2 (in het bijzonder pag. 335)



Academie voor Technology, Innovation & Society Delft

Academie voor ICT & Media

De Haagse Hogeschool, Delft
015-2606311
J.E.J.opdenBrouw@hhs.nl
www.dehaagsehogeschool.nl

DE HAAGSE HOGESCHOOL