Microcontroller Programmeren in C

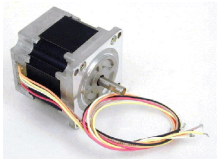
MICPRG Les 6

DE HAAGSE HOGESCHOOL

Pulse generator

- Veel signalen in de buitenwereld zijn pulsvormig.

- Veel actoren kunnen met een pulsvormig signaal worden aangestuurd:
  - Lamp.
  - Motor.
  - Telefoonkiezer.(Puls en toon!)
  - Luidspreker.
  - enz...



- Om deze signalen te kunnen opwekken hebben veel µC's een ingebouwde pulse generator.

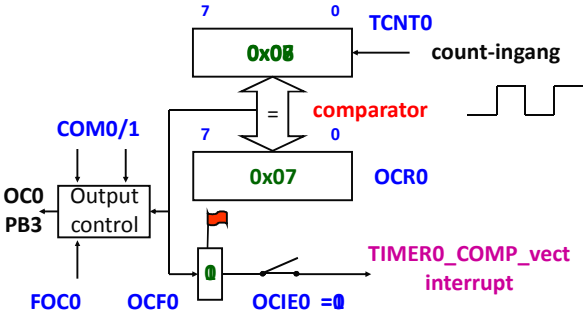
DE HAAGSE HOGESCHOOL 51

Timer/Counter0

- Timer/Counter0 kan een pulsvormig signaal opwekken op pin OC0 = PB3.
  - Output Compare eventueel in CTC mode = Clear Timer on Compare match (50% duty-cycle).
  - Fast PWM (instelbare duty-cycle). PWM = Pulse Width Modulation.
  - Fase correct PWM (instelbare duty-cycle).
- Timer/Counter1 kan pulsvormige signalen op-wekken op pinnen OC1A = PD5 en OC1B = PD4.
- Timer/Counter2 kan een pulsvormig signaal opwekken op pin OC2 = PD7

DE HAAGSE HOGESCHOOL 52

Output compare



DE HAAGSE HOGESCHOOL 53

T/C0 count-ingang

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

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.

DE HAAGSE HOGESCHOOL 54

T/C0 Output Compare

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Waveform Generation Mode Bit Description

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, non-PWM Mode

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

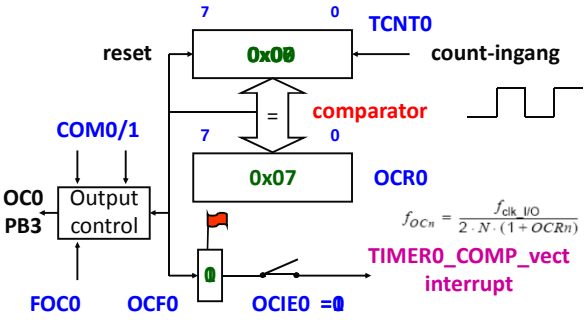
DE HAAGSE HOGESCHOOL 55

T/C0 Output Compare

7	6	5	4	3	2	1	0	
OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	TIFR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

- OCF0 = Output Compare Flag 0.
- OCIE0 = Output Compare Interrupt Enable 0.
- FOC0 = Force Output Compare 0.
  FOC0 = 1 ➔ Actie op pin OC0 = PB3, die bij compare match wordt uitgevoerd, wordt nu meteen uitgevoerd!

CTC Clear Timer on Compare Match



T/C0 CTC

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Waveform Generation Mode Bit Description

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, non-PWM Mode

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

Opdracht

- Zie AVR boek pagina 44 t/m 46.
- 1 seconde wachten bij F_{CLK} = 2.4576 MHz.
- Maak een variant van dit programma met behulp van T/C0 Output Compare in CTC mode.

```
void wacht1sec() {
    uint8_t i;
    TCCR0 &= ~(1<<COM01 | 1<<COM00 | 1<<WGM00 | 1<<CS01);
    TCCR0 |= 1<<WGM01 | 1<<CS02 | 1<<CS00;
    OCR0 = 239;
    TCNT0 = 0;
    for (i = 0; i < 10; i++) {
        TIFR = 1<<OCF0;
        while (~TIFR & 1<<OCF0);
    }
    TCCR0 &= ~(1<<CS02 | 1<<CS01 | 1<<CS00);
}
```

Waarom TIFR = 1<<OCF0
en niet TIFR |= 1<<OCF0 ?

Opdracht

- Stel: Er is een luidspreker aangesloten op pin OC0 = PB3. De frequentie van de µC = 8 MHz.
- Schrijf een programma om een kamertoon op de luidspreker te genereren.
- De kamertoon, in de muziektheorie a' genoemd terwijl de wetenschappelijke benaming A_4 is wordt veel gebruikt voor het stemmen van muziek-instrumenten. De kamertoon hoort een frequentie van 440 Hz te hebben. De meeste stemvorken worden dan ook op 440 Hz gemaakt.

Uitwerking

- 440Hz => periode tijd = 1/440 = sec.
- PD3 elke 1/880 sec inverteren (toggle).
- f_{CPU} = 8 MHz. Prescaler zo laag mogelijk (nauwkeuriger).
- Prescaler=1 => 1 tick T/C0 = 1/8000000 sec. OCR0+1 = 8000000/880 = 9091 Past niet!
- Benodigde prescaler 9091/256 = 36. Kies 64.
- Prescaler=64 => 1 tick T/C0 = 64/8000000 sec. OCR0+1 = 8000000/(880*64) = 142.
- Opgewekte freq = 8000000/(64*142*2) = 440,14

Uitwerking

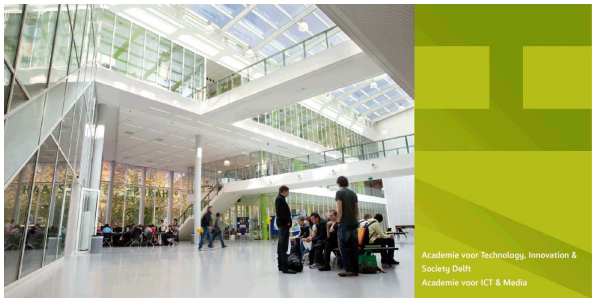
```
#include <avr/io.h>

int main(void) {
    OCR0 = 141;
    TCCR0 = 0x1B;
    DDRB |= 1<<3;
    while (1);
    return 0;
}
```

DE HAAGSE HOGESCHOOL 62

Huiswerk

- Bestudeer AVR Boek pagina 121.
- Bestudeer AVR datasheets hoofdstuk over T/C0 (18 pagina's).

DE HAAGSE HOGESCHOOL 63



Microcontroller Programmeren in C

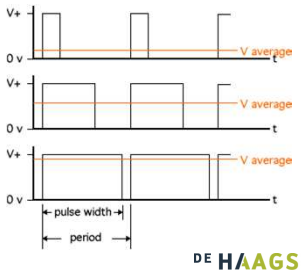
MICPRG Les 7

DE HAAGSE HOGESCHOOL

PWM

- PWM = Puls Width Modulation
- De periodetijd is constant maar de pulsduur varieert.
- Toepassingen:
  - Dimmer.
  - Motorregeling.

Duty Cycle = pulsduur/periodetijd x 100 %



DE HAAGSE HOGESCHOOL 65

PWM Timer/Counter0

- Timer/Counter0 kan een PWM signaal opwekken op pin OC0 = PB3.
  - Fast PWM.
  - Fase correct PWM.
- Timer/Counter1 kan PWM signalen opwekken op pinnen OC1A = PD5 en OC1B = PD4.
  - Timer/Counter1 heeft **12** verschillende PWM modes.
- Timer/Counter2 kan een PWM signaal opwekken op pin OC2 = PD7
  - Timer/Counter2 heeft dezelfde 2 modes als T/C0).

DE HAAGSE HOGESCHOOL 66

Fast PWM Timer/Counter0

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Waveform Generation Mode Bit Description

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, Fast PWM Mode

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP Active high
1	1	Set OC0 on compare match, clear OC0 at TOP Active low

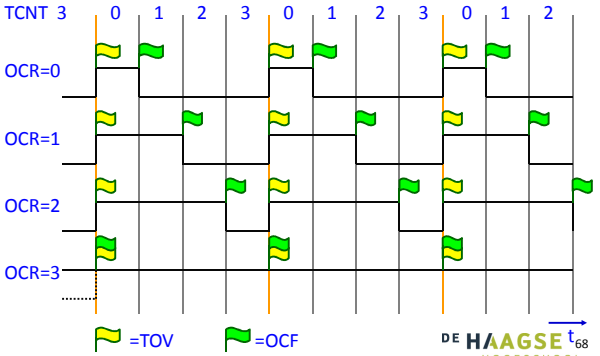
TOP = 255
MAX = 255

Output OC0 veranderd 1 klokperiode na match!

DE HAAGSE HOGESCHOOL 67

Dit is een simpel voorbeeld, T/C0 heeft 8 bits

Fast PWM 2 bits Active high



Fast PWM

- $f_{OC} = f_{CLK} / (N \times (MAX+1))$ N = prescaler
- T/C0: $f_{OC0} = f_{CLK} / (N \times 256)$
  voor f_{CLK} = 3.686 MHz: $f_{OC0,\ max}$ = 14.4 kHz
  voor f_{CLK} = 8.000 MHz: $f_{OC0,\ max}$ = 31.3 kHz
- Duty $Cycle_{OC} = (OCR+1) / (MAX+1) \times 100\%$
- T/C0: Duty $Cycle_{OC0} = (OCR0+1) / 256 \times 100\%$
  instelbaar in 256 stappen 0,39% … 100%

Phase Correct PWM T/C0

7	6	5	4	3	2	1	0	
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Waveform Generation Mode Bit Description

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, Phase Correct PWM Mode

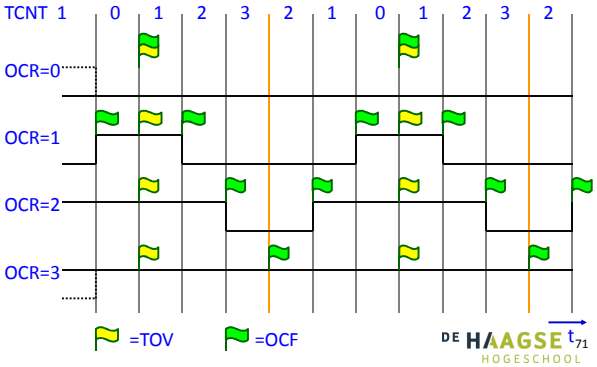
COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting. Active high
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting. Active low

TOP = 255
BOTTOM = 0

Output OC0 veranderd 1 klokperiode na match!

Dit is een simpel voorbeeld, T/C0 heeft 8 bits

Phase Correct PWM 2 bits



Phase Correct PWM

- $f_{OC} = f_{CLK} / (N \times 2 \times MAX)$ N = prescaler
- T/C0: $f_{OC0} = f_{CLK} / (N \times 510)$
  voor f_{CLK} = 3.686 MHz: $f_{OC0,\ max}$ = 7.2 kHz
  voor f_{CLK} = 8.000 MHz: $f_{OC0,\ max}$ = 15.7 kHz
- Duty $Cycle_{OC}$ = OCR / MAX x 100%
- T/C0: Duty $Cycle_{OC0}$ = OCR0 / 255 x 100%
  instelbaar in 256 stappen 0% … 100%

OCR0 register in PWM modes

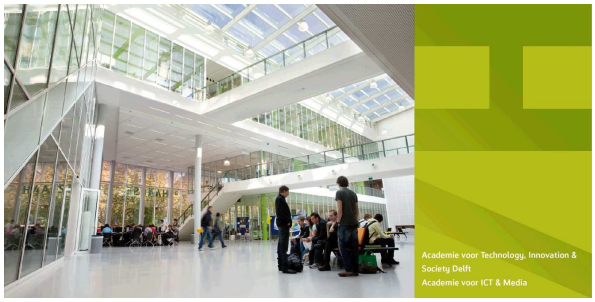
- Het OCR0 register is in de PWM modes dubbel gebufferd (double buffered).
- Schrijf actie naar OCR0 wordt gebufferd (in buffer registers).
- 1 klokcycle nadat TCNT0 == TOP wordt de gebufferde waarde naar het OCR0 register geschreven. Zie oranje lijnen in timing diagrammen.
- Dit voorkomt “vreemde” pulsen als OCR0 aangepast wordt.

Fast versus Phase Correct

- Fast PWM heeft een hogere f_{OC}. Voorkomt irritante pieptoon bij vermogensregeling.
- Phase Correct PWM is symmetrisch. Dit is “beter” bij motorsturingen.
- Duty Cycle is bij Phase Correct PWM regelbaar vanaf 0%.
- Huiswerk:
  - Lees blz. 127 t/m 129 in het AVR boek (AT90S2313)
  - Schrijf een programma om LED3 met een PWM signaal met een duty cycle van 25% aan te sturen.

Uitwerking huiswerk

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

int main(void) {
    TCCR0 = 0x7B;
    OCR0 = 63;
    DDRB = 0x08;
    while (1);
    return 0;
}
```

Microcontroller Programmeren in C

MICPRG Les 8

DE HAAGSE HOGESCHOOL

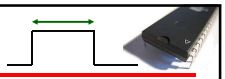
Pulse input

- Veel signalen in de buitenwereld zijn pulsvormig.

- Veel sensoren leveren een pulsvormig signaal:
  - Tachometer (toerenteller).
  - Hartslagsensor.
  - Snelheidsmeter (fietscomputer).
  - Foto-elektrische sensoren.
  - Pedometer, enz...

- Om de pulsduur te kunnen meten hebben veel µC’s een ingebouwde input capture unit.

Meten van pulsduur

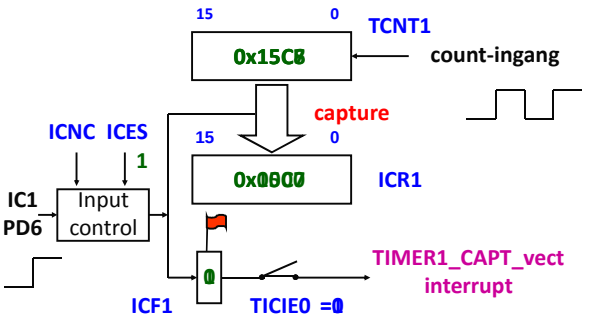


- Algoritme:
  - Wacht tot ingang hoog wordt en reset Timer.
  - Wacht tot ingang laag wordt en lees Timer uit.
  - Tel indien nodig Timer overflows.
  - Pulsduur = (overflows*(MAX+1) + TCNT) * N / f_{clk}
- Waarom is dit niet zo’n goed idee?
  - Als je iets anders aan het doen bent kan het even duren voordat je in de gaten hebt dat ingang veranderd is.
- Interrupt als ingang veranderd?
  - Beter maar: Als je ook andere interrupts gebruikt kan het nog steeds even duren voordat je kan reageren...

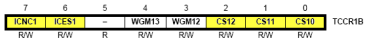
Input capture unit

- Timer/Counter1 van de ATmega32 heeft een speciale Input Capture unit voor het meten van pulsduur.
- Waarde van de TCNT1 wordt gekopieerd in het ICR1 (Input Capture Register 1) bij een bepaalde flank op pin IC1 = PD6.

Input capture



Input capture

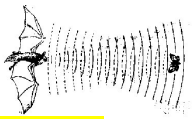


- ICNC1 = Input Capture Noise Canceler 1
  - 0 = Flank wordt meteen herkend.
  - 1 = Flank wordt pas herkend als signaal 4 CPU klokperioden stabiel is. Filtert HF stoorpulsen weg.
- ICES1 = Input Capture Edge Select 1
  - 0 = capture bij
  - 1 = capture bij
- CS1x = Clock Select 1
  - Instelbare prescaler 1, 8, 64, 256, 1024

Lage prescaler = hoge nauwkeurigheid

Meten van pulsduur

- Toepassing:
  - Zend ultrasoon puls en ontvang reflectie
  - verschil = maat voor afstand.



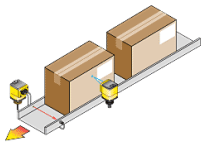
Bij f_{CLK} = 8 MHz nauwkeurigheid +/- 0,04 mm

- Huiswerk:
  - Lees blz. 119 t/m 121 in het AVR boek (AT90S2313).
  - Schrijf een programma om de pulsduur van een positieve puls op pin PD6 te meten. Gegeven:
    - f_{CPU} = 8 MHz.
    - 100 µs <= pulsduur <= 10000 µs.
    - Meting moet op 1 µs nauwkeurig zijn.
    - Programma hoeft verder niets te doen.

Pulse input

- Veel signalen in de buitenwereld zijn pulsvormig.

- Veel sensoren leveren een pulsvormig signaal:
  - Tachometer (toerenteller).
  - Hartslagsensor.
  - Snelheidsmeter (fietscomputer).
  - Foto-elektrische sensoren.
  - Pedometer, enz...



- Om de pulsen te kunnen tellen hebben veel µC's een ingebouwde counter.

Timer/Counter0

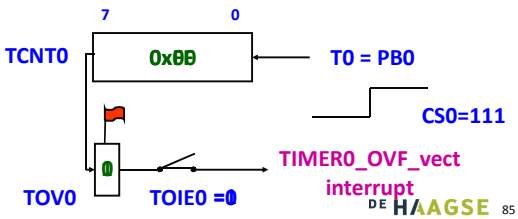
- Counter: kies externe clock T0 = PB0.
  - Instellingen in TCCR0 (Timer/Counter Control Register 0)

Clock Select Bit Description

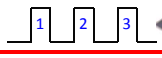
CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

T/C0 overflow interrupt R

- T/C0 kan een interrupt genereren als TCNT0 overloopt.
- Deze interrupt kun je aanzetten met het TOIE0 bit in het TIMSK register.



Tellen van pulsen



- Huiswerk:
  - Schrijf een programma dat telkens na 24 neergaande pulsen op pin PB0 een interrupt geeft (kratje vol).

DE HAAGSE HOGESCHOOL 86

Uitwerking

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

/* inverteer PB7 na 24 neergaande flanken op PB0 */

ISR(TIMER0_COMP_vect) {
    PORTB ^= 0x80;
}

int main(void) {
    TCCR0 = 0x0E;
    OCR0 = 23;
    TIMSK = 0x02;
    DDRB = 0x80;
    sei();
    while (1);
    return 0;
}
```

DE HAAGSE HOGESCHOOL 87

Microcontroller Programmeren in C

MICPRG Les 9

DE HAAGSE HOGESCHOOL

Communicatie

- Veel microcontrollers communiceren met hun omgeving. Redenen voor communicatie zijn:
  - Inlezen sensoren en aansturen actuatoren.
  - Gedistribueerde besturingen.
    - Het systeem bevat meerdere µC's die samen het systeem besturen. B.v. auto, robot enz.
  - Onderhoud.
    - Programmeerinterface, diagnose interface, enz.
  - Gemeten data moet naar verzamelpunt gestuurd worden (data acquisitie systemen).

HAAGSE HOGESCHOOL 89

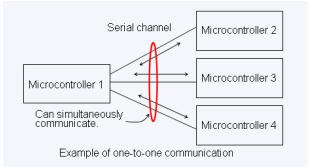
Soorten communicatie

- Communicatie kanaal:
  - Simplex (1 richting). B.v. radio-uitzending.
  - Half Duplex (omschakelbaar). B.v. walkie-talkie.
  - Full Duplex (2 richtingen). B.v. telefoon.
- Communicatie medium:
  - Kabel.
  - Ether (draadloos).
- Communicatie topologie:
  - Punt naar punt.
  - Bus structuur (master-slave).
    - Single master.
    - Multi-master.

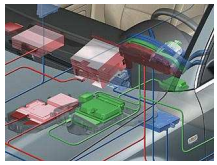
Master neemt initiatief voor data-overdracht en bepaald richting

90

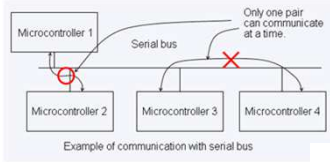
Topologie



Example of one-to-one communication



Bus heeft adressering nodig. Multi-master bus heeft arbitration nodig (bepalen wie de master is).

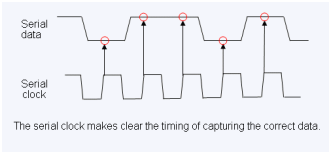


Example of communication with serial bus

91

Soorten communicatie

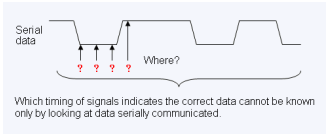
- Parallel versus serieel:
  - Voordelen serieel: goedkoper, geen overspraak tussen parallelle signalen.
- Synchroon versus asynchroon:
  - Synchroon: kloksignaal wordt meegestuurd.



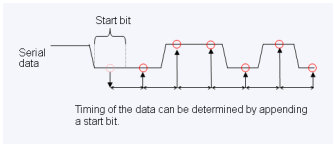
DE HAAGSE HOGESCHOOL 92

Asynchroon

- Kloksignaal wordt niet meegestuurd.



Werken met afgesproken
Baudrate = aantal signaal-wisselingen/seconde.
Bitrate = aantal bits/seconde.



Let op! Baudrate is niet altijd gelijk aan Bitrate.

DE HAAGSE HOGESCHOOL 93

Communicatie standaarden

- IEEE 1284 parallelle poort PC.
- RS232 seriële poort PC.
- RS485 gebouw automatisering, DMX theater verlichting.
- SPI (Serial Peripheral Interface) ISP=In-System Programming. On board peripheral bus (LCD, ADC, RTC=Real Time Clock enz.).
- I^2C (Inter-Integrated Circuit) On board peripheral bus (LCD, ADC, RTC=Real Time Clock enz.).
- JTAG (Joint Test Action Group) Testing, ISP, debugging.
- USB (Universal Serial Bus) Off board peripherals (muis, printer enz.).
- CAN (Controller Area Network) Auto, Machines.
- Ethernet Internet, Machines.
- ...

DE HAAGSE HOGESCHOOL 94

Communicatie standaarden

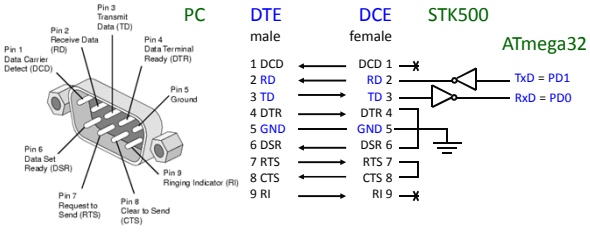
- IEEE 1284 parallel, half duplex, P2P, synchroon.
- RS232 serieel, full duplex, P2P, asynchroon.
- RS485 serieel, half duplex, multi-master, asynchroon.
- SPI serieel, full duplex, single master, synchroon.
- I^2C serieel, half duplex, multi-master, synchroon.
- JTAG serieel, full duplex, single master, synchroon.
- USB serieel, half duplex, single master, synchroon (NRZI met bit-stuffing).
- CAN serieel, half duplex, multi-master, synchroon.
- Ethernet serieel, full duplex, multi-master, synchroon.
- ...

Andrew Tanenbaum: "The nice thing about standards is that there are so many to choose from!"

95

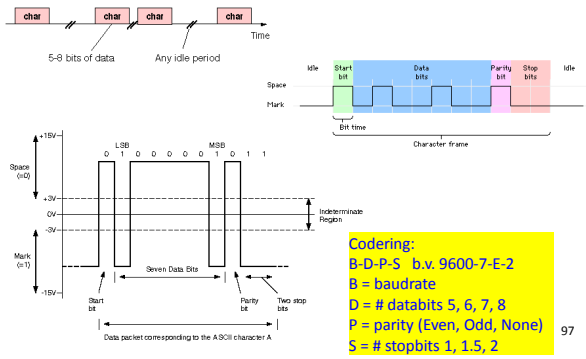
RS232

- DTE: Data Terminal Equipment (PC, Terminal).
- DCE: Data Communications Equipment (Modem).



DE HAAGSE HOGESCHOOL 96

RS232

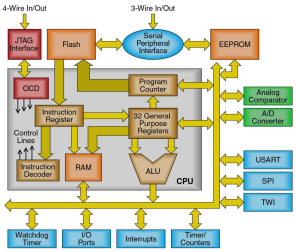


Codering:
B-D-P-S b.v. 9600-7-E-2
B = baudrate
D = # databits 5, 6, 7, 8
P = parity (Even, Odd, None)
S = # stopbits 1, 1.5, 2

97

ATmega32 communicatie

- Parallel. (I/O poorten).
- Serieel Synchroon. (Via software of USART).
- Serieel Asynchroon. (Via software of USART).
- SPI (Serial Peripheral Interface).
- TWI = Two-Wire Interface = I^2C (Inter-Integrated Circuit).
- JTAG (Joint Test Action Group).



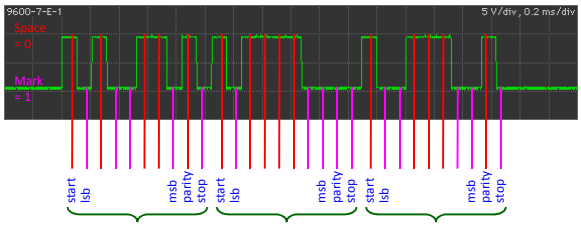
Huiswerk

- Gegeven: 9600-7-E-1. Wat wordt hier in ASCII verstuurd?

- Bestudeer:
  - AVR boek: blz. 129 t/m 134.
  - ATmega32 datasheets.
- Programmeer:
  - Maak een echoput. Elk ontvangen karakter wordt 2x teruggestuurd. Gebruik 9600-7-E-1.

Uitwerking

- Gegeven: 9600-7-E-1. Wat wordt hier in ASCII verstuurd?

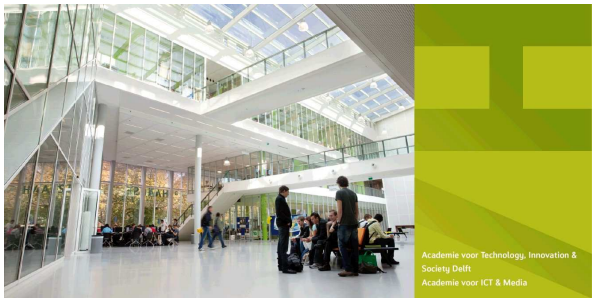


Zie AVR Boek: Appendix G

Karakter 1
1001101
'M'
Parity = OK

Karakter 2
1100001
'a'
Parity = OK

Karakter 3
1100011
'c'
Parity = OK

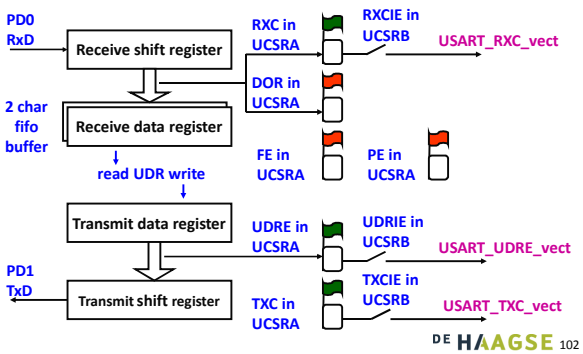


Microcontroller Programmeren in C

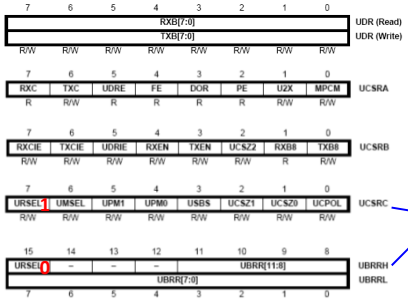
MICPRG Les 10

DE HAAGSE HOGESCHOOL

USART Universal Synchronous Asynchronous Receiver Transmitter



USART ATmega32

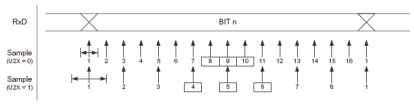


2 registers op 1 I/O adres

2 registers op 1 I/O adres

Zie ATmega32 datasheets USART Register Description.

USART ATmega32 timing



Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal Mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

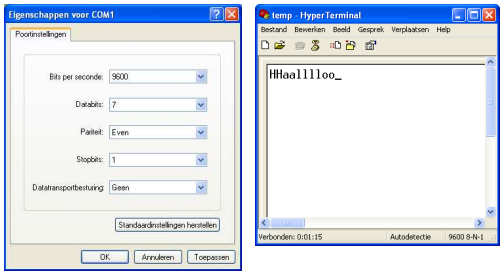
BAUD Baud rate (in bits per second, bps)
f_{OSC} System Oscillator clock frequency
UBRR Contents of the UBRRH and UBRRL Registers, (0 - 4095)

Synchronous: Clock pin = XCK = PB0

Echoput

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

int main(void) {
    char c;
    // 9600-7-E-1
    UCSRA = 0x00; UCSRB = 0x18; UCSRC = 0xA4;
    UBRRH = 0; UBRRL = 23;
    while (1) {
        while (~UCSRA & 1<<RXC);
        if (UCSRA & (1<<FE | 1<<DOR | 1<<PE)) {
            c = UDR; c = '?';
        }
        else {
            c = UDR;
        }
        while (~UCSRA & 1<<UDRE); UDR = c;
        while (~UCSRA & 1<<UDRE); UDR = c;
    }
    return 0;
}
```

Echoput



Echoput Huiswerk

- Wat is er mis als je de volgende uitvoer krijgt bij het intypen van het alfabet?

  aabb??dd????gghh????kk??mmnn??pp????ss??uuvv????yyzz

- Herschrijf het programma zodat zenden en ontvangen met interrupts gebeurd.
  - Gebruik een globale variabele voor dataoverdracht.
  - Zet eerst alleen de ontvangstinterrupt aan.
  - Ontvangstinterrupt schrijft globale variabele en zet zendinterrupt aan en ontvangstinterrupt uit.
  - Zendinterrupt zet na 2x ontvangstinterrupt aan en zendinterrupt uit.

Uitwerking

- Wat is er mis als je de volgende uitvoer krijgt bij het intypen van het alfabet?

  aabb??dd????gghh????kk??mmnn??pp????ss??uuvv????yyzz

7 bit ASCII
a = 1100001
b = 1100010
c = 1100011
d = 1100100
e = 1100101
f = 1100110
g = 1100111

+ even parity
a = 11100001
b = 11100010
c = 01100011
d = 11100100
e = 01100101
f = 01100110
g = 11100111

Conclusie: in plaats van het pariteitsbit wordt een 1 gestuurd.

ATmega32: 9600-7-E-1
PC: 9600-7-N-2

Hyperterminal drukt ? niet af. ? = 0111111 dit wordt verzonden met even parity (0) waardoor een stopbit mist!

Uitwerking (deel 1)

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

char c;

ISR(USART_RXC_vect) {
    if (UCSRA & (1<<FE | 1<<DOR | 1<<PE)) {
        c = UDR;
        c = '?';
    }
    else {
        c = UDR;
    }
    UCSRB &= ~(1<<RXCIE);
    UCSRB |= (1<<UDRIE);
}
```

Uitwerking (deel 2)

```
ISR(USART_UDRE_vect) {
    static uint8_t echo = 0;
    UDR = c;
    echo++;
    if (echo == 2) {
        echo = 0;
        UCSRB |= (1<<RXCIE);
        UCSRB &= ~(1<<UDRIE);
    }
}

int main(void) {
    // 9600-7-E-1
    UCSRA = 0x00; UCSRB = 0x98; UCSRC = 0xA4;
    UBRRH = 0; UBRRL = 23;
    sei();
    while (1);
    return 0;
}
```

Moet de variabele c niet `volatile` zijn?

DE HAAGSE HOGESCHOOL 110