

AT90CAN32, hoofdstuk 6

E. Gernaat (ISBN 978-90-79302-06-2)

1 Extra programma's

1.1 De stappenmotor

Het volgende programma demonstreert de werking van een stappenmotor. Hiervoor dienen we de print met de stappenmotor op de 34-polige connector aan te sluiten (fig. 1). Zie verder de bijlage (het laatste hoofdstuk) voor het elektrische aansluitschema. Zoals bekend mag worden verondersteld wordt een stappenmotor aangestuurd door een stappenpatroon. Alleen het hoofdprogramma is voor ons interessant. Na bestudering van het hoofdprogramma mag de werking niet al te veel problemen meer opleveren. Voer vervolgens de bij het programma behorende opdrachten uit.



Figuur 1: Foto van de stappenmotorprint

```

/*Programma naam: STAPPENMOTOR.ASM
Programma voor de AT90CAN32 Elektor-Timloto print.
Stappenmotor gemonteerd op een aparte print.
Poort F output. Stappenmotor aangesloten op PF1, PF3, PF5, PF7.
Poort A output en geeft het stappenpatroon op de leds weer.
Schakelaars op poort C worden hier niet gebruikt.
Er wordt hier gebruik gemaakt van AVR Studio 4.
Het programma draait vanuit het flash-geheugen.*/

.DEVICE AT90CAN32

.INCLUDE "can32def.inc"           ; definitie poorten in aparte file

RJMP RESET                       ;spring naar het begin adres

/*INITIALISATIE*/

RESET:    LDI R16,0xFF             ;zet alle pinnen van poort A en F op output
          OUT DDRF,R16
          OUT DDRA,R16

          LDI R16,high(RAMEND)
          OUT SPH,R16
          LDI R16,low(RAMEND)
          OUT SPL,R16             ;stackpointer initialisatie nodig voor subroutine

          ;Op poort C zitten de schakelaars, dus op input
          LDI R16,0xFF           ;activeer de pull-up weerstanden
          OUT PORTC,R16         ;door enen te schrijven naar de inputpoort
          LDI R17,0x00          ;zet alle pinnen van poortC op input
          OUT DDRC,R17         ;niet strikt noodzakelijk (default waarde)
          NOP                   ;No Operation

/* HOOFDPROGRAMMA*/

BEGIN:    LDI R17,0b00100010     ;0101 stap1
          OUT PORTF,R17         ;naar stappenmotor
          LDI R21,0b00000101    ;stap 1
          OUT PORTA,R21        ;naar leds
          RCALL WACHT1

          LDI R17,0b10000010    ;1001 stap2
          OUT PORTF,R17

```

```

LDI R21,0b00001001 ;stap 2
OUT PORTA,R21 ;naar leds
RCALL WACHT1

LDI R17,0b10001000 ;1010 stap3
OUT PORTF,R17
LDI R21,0b00001010 ;stap 3
OUT PORTA,R21 ;naar leds
RCALL WACHT1

LDI R17,0b00101000 ;0110 stap4
OUT PORTF,R17
LDI R21,0b00000110 ;stap 4
OUT PORTA,R21 ;naar leds
RCALL WACHT1

RJMP BEGIN

```

```
/*SUBROUTINE WACHT*/
```

```

WACHT1: LDI R20,0x0F ;0F (01 voor debugger)
WACHT: LDI R18,0xFF ;0x77 (01 voor debugger)
WEER: LDI R19,0xFF ;0xFF (01 voor debugger)
LUS: SUBI R19,0x01
BRNE LUS
SUBI R18,0x01
BRNE WEER
SUBI R20,0x01
BRNE WACHT
RET ; keer terug naar het hoofdprogramma

```

*/*Vragen en opgaven*

1. Gebruik de stappenmotorprint geschikt voor de Elektor-Timloto print. Sluit deze met de flat-cable aan. Sluit de voeding nog niet aan! Download het programma en controleer de werking. De leds laten nu het stappenpatroon zien. Sluit nu de (5-10V) voeding aan en controleer of de stappenmotor daadwerkelijk draait. Zet de voedingsspanning niet hoger dan strikt noodzakelijk.
2. Laat de stappenmotor sneller en langzamer draaien door de wachtlus aan te passen.
3. Zoek het maximale toerental op waarbij de motor nog net draait.
4. Verander de draairichting van de stappenmotor. Dus stap 4 wordt stap 3, stap 3 wordt 2 etc.

Extra opdracht:

Betrek een schakelaar bij het programma waardoor de motor naar keuze linksom of rechtsom kan draaien.

Vraag:

Is het, gezien het feit dat zowel de potmeter als de stappenmotor gebruik maken van poort F, mogelijk om met de potmeter de snelheid van de stappenmotor te regelen? Verklaar het antwoord. /*

2 Analooq-digitaal omzetting

We laten nu zien hoe we analoge signalen in kunnen lezen en digitaal weer kunnen geven. Poort F kan digitaal maar ook analoog worden gebruikt. Poort F is gekoppeld aan een Analooq-Digitaal omzetter (ADC). Het omzetten van een analoog naar een digitaal signaal gaat in stappen. Dit betekent dat het een aantal klokpulsen duurt voordat de omzetting gereed is. We maken in eerste instantie gebruik van de potmeter op de controllerprint. De toegepaste AD-omzetter van de AT90CAN32-processor heeft maximaal een 10 bits resolutie, een 8 bits resolutie kan echter worden ingesteld. Dit betekent dat een spanning van 0-5 V in stapjes van $5V / 1024 (2^{10}) = 0,005 V$ of $5V / 256 (2^8) = 0,002 V$ kan worden weergegeven. De enkelvoudige ADC is verbonden met een 8 kanaals multiplexer waardoor 8 individuele analoge ingangen ontstaan. Deze zgn. 'single ended inputs' refereren aan 0 V (GND). Wanneer de A_{ref} pin met 5 V wordt verbonden, dan ontstaan 8 analoge inputs die een spanning kunnen verwerken tussen de 0 en 5 V. Dit is het geval op de Elektor-Timloto print.

Het multiplex principe verbindt de geselecteerde pin met de AD-omzetter. Een andere mode maakt het ook mogelijk om een verschilspanning te digitaliseren. Wij maken hier echter geen gebruik van. De instelling van de ADC geschiedt in eerste instantie met behulp van het ADMUX register. We geven een paar programmeer-voorbeelden:

- 00000000 in het ADMUX register selecteert de ADC0 pin en geeft een 10 bits resolutie;
- 00100000 in het ADMUX register selecteert de ADC0 pin en geeft een 8 bits resolutie;
- 00100001 in het ADMUX register selecteert de ADC1 pin en geeft een 8 bits resolutie;

In de practicum-opdrachten gebruiken we alleen de ADC0 en de ADC1 pin met een 8-bits resolutie. Het tweede register is het zgn. 'ADC control en status register' afgekort tot ADCSRA. Hierin kunnen we de ADC aan of uit zetten en de conversie (omzetting) starten.

- 10000000 in het register zet de ADC aan en
- 11000000 in het register zet de ADC aan en start het omzettingsproces.

We dienen hier de enkelvoudige conversie-mode en de free running conversie-mode te onderscheiden. Bij de enkelvoudige conversie moet elke keer weer een logische 1 naar bit 6 worden geschreven. In de enkelvoudige conversie mode wordt bit 6 weer gereset (wordt 0) zodra de omzetting is voltooid. We moeten dit bit controleren voordat we het resultaat van de omzetting kunnen uitlezen. De resultaat van de conversie wordt naar het 'ADC data register' geschreven afgekort tot ADCL en ADCH. Wanneer we voor een 8-bits precisie hebben gekozen dan is het voldoende om alleen ADCH uit te lezen. Dit doen we in de programma's. Het voorbeeldprogramma ANALOOG1 leest de analoge spanning van de potmeter op de print in waarna het wordt gedigitaliseerd. Het resultaat hiervan wordt op de leds gezet. Hier volgt het programma.

```

/*Programma naam: ANALOOG1.ASM
Programma voor de AT90CAN32 Elektor-Timloto print.
Het programma zet de stand van potmeter digitaal op de leds van de print.
Er wordt hier gebruik gemaakt van AVR Studio4.
Potmeter op de print is aangesloten op PFO (AD0). Denk om de jumper!
Programma en data in het Flash geheugen */

.DEVICE AT90CAN32

.INCLUDE "can32def.inc"           ; definitie poorten in aparte file

R JMP RESET                      ;spring naar het begin adres

/*INITIALISATIE POORTEN*/

RESET:    LDI R16,0xFF            ;zet alle pinnen van poortA op output
          OUT DDRA,R16           ;DDRA=Data-Directie-register

/*INITIALISATIE STACKPOINTER*/

          LDI R16,high(RAMEND)
          OUT SPH,R16
          LDI R16,low(RAMEND)
          OUT SPL,R16

/*INITIALISATIE ADC */

;selecteer het eerste kanaal (bit 0 t/m 4) door de MUX bits te setten
;kanaal 0 is 00000
;set de ADLAR bit, bit 5 (left adjusted, nodig voor 8-bits precisie)

```

```
LDI R16,0x20          ;binair 0b00100000
STS ADMUX,R16
```

*;enable ADC door ADEN bit (bit7) te zetten
;start single bit conversion, set bit ADSC (bit6)
;bit6 wordt 0 als de conversie is beëindigd*

```
LDI R16,0b11000000
STS ADCSRA,R16
```

;lees alleen ADCH uit voor een 8 bits resolutie

*/*HOOFDPROGRAMMA*/*

```
BEGIN:  LDI R16,0b11000000  ;zet startbit weer op 1
        STS ADCSRA,R16     ;start single conversion
```

```
CONVERS: LDS R17,ADCSRA    ;controleer of conversie gereed is?
         SBRC R17,ADSC     ;skip als bit is nul wordt
         RJMP CONVERS
```

```
LDS R16,ADCH              ;alleen ADCH uitlezen voor 8 bits resolutie
OUT PORTA,R16
```

```
RJMP BEGIN
```

/ Vragen en opgaven*

1. Download het programma en draai aan de potmeter en zie of de leds oplichten.
2. Zet een voltmeter over de PF0 pin (pin 25 van de 34-polige connector) en de min (pin 34) en draai de potmeter zodanig dat deze steeds met 0,5 V verhoogd wordt. Vul dan de tabel in:

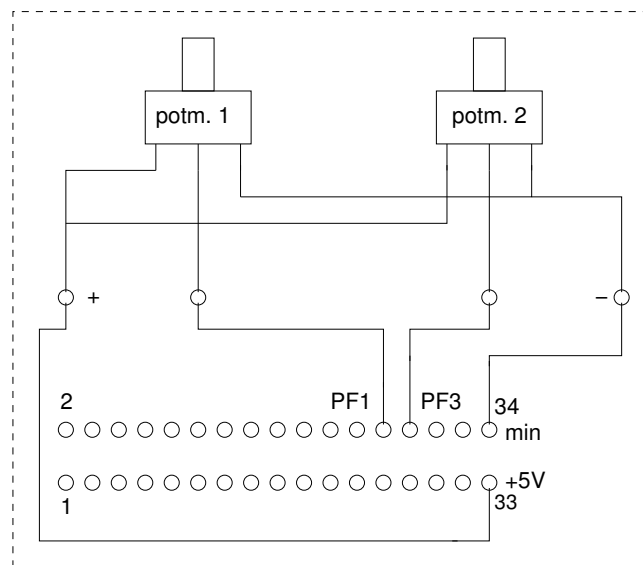
<i>potmeter stand:</i>	<i>binair stand leds</i>	<i>decimale waarde</i>
0,0 V
0,5 V
1,0 V
1,5 V
2,0 V
2,5 V
3,0 V

3,5 V
4,0 V
4,5 V
5,0 V

3. Wat is nu de resolutie in volts van deze AD-omzetter?
Dus: 1 bit komt overeen metV.
4. Wat is de resolutie in volts van een 10 bits ADC? (hulp: 8 bits komt overeen met 256, 10 bits komt overeen met) */

2.0.1 Het tweede ADC-programma

Het tweede programma over AD-omzetting maakt gebruik van een externe print die op de 34-polige connector kan worden aangesloten. Op deze print zijn twee potmeters bevestigd, een lineaire en een logaritmische. Om de analoge spanning te kunnen meten zijn op de print ook een aantal meetpennen aangebracht (fig. 2). Sluit de potmeterprint aan op de 34-polige connector van de AT90CAN32. Sluit vervolgens een voltmeter aan op de middelste pin (loper)



Figuur 2: Schema van de potmeterprint. Met behulp van een voltmeter kan de spanning op de lopers worden gemeten.

van een potmeter en de min-pin. Assembleer en download nu het volgende programma (ANALOG2) en beantwoord de vragen. Verdraai eerst potmeter1 en vervolgens potmeter2 in 10 verschillende standen. Vul tijdens het draaien de gemeten analoge waarden in (kolom 2 van de tabellen). Zie fig. 3.

/*Programma naam: ANALOOG2.ASM

Het programma wordt gebruikt voor de print met een lineaire en een logaritmische potmeter (de potmeterprint). De gewenste potmeter kan worden geselecteerd door schakelaar S0. Potmeters zijn aangesloten op PF1 en PF3 (AD1 en AD3). De potmeters worden ingelezen en de digitale waarde wordt op de leds gezet. Het programma is ook geschikt om de gasklepstand van bijv. het Monomotronic gasklephuis op de leds te zetten.

Er wordt hier gebruik gemaakt van AVR Studio 4. Programma en data in het flash geheugen */

```
.DEVICE AT90CAN32
```

```
.INCLUDE "can32def.inc" ; definitie poorten in aparte file
```

```
RJMP RESET ;spring naar het begin adres
```

```
/*INITIALISATIE POORTEN*/
```

```
RESET:    LDI R16,0xFF          ;zet alle pinnen van poortA op output
           OUT DDRA,R16      ;DDRA=Data-Directie-register

           LDI R16,0xFF      ;activeer de pull-up weerstanden
           OUT PORTC,R16     ;door enen te schrijven naar de inputpoort
           LDI R17,0x00      ;zet alle pinnen van poort C op input
           OUT DDRC,R17     ;niet strikt noodzakelijk (default waarde)
           NOP
```

```
/*INITIALISATIE STACKPOINTER*/
```

```
           LDI R16,high(RAMEND)
           OUT SPH,R16
           LDI R16,low(RAMEND)
           OUT SPL,R16

BEGIN:    IN R16,PINC        ;lees schakelstand in
           ANDI R16,0x01     ;selecteer schakelaar S0, AND R16 met 0x01
           BRNE KANAAL3     ;antwoord ongelijk 0, dan naar KANAAL3
```

```
/*INITIALISATIE ADC */
```

```
;selecteer het eerste kanaal (bit 0 t/m 4) door MUX bits te zetten
;kanaal 0 is 00000
;kanaal 1 is 00001
;kanaal 3 is 00011
```

;set de ADLAR bit, bit 5 (left adjusted, nodig voor 8 bits precisie)

*;enable ADC door ADEN bit (bit7) te zetten
;start single bit conversion, set bit ADSC (bit6)
;bit 6 wordt 0 als de conversie is beëindigd
;lees alleen ADCH uit voor een 8 bits resolutie*

```
LDI R16,0x21          ; 0b00100001 selecteer kanaal 1
STS ADMUX,R16
```

```
LDI R16,0b11000000
STS ADCSRA,R16
RJMP CONVERSIE
```

```
KANAAL3: LDI R16,0x23          ; 0b00100011 selecteer kanaal 3
          STS ADMUX,R16
```

```
LDI R16,0b11000000
STS ADCSRA,R16
RJMP CONVERSIE
```

*/*DE EIGENLIJKE CONVERSIE*/*

```
CONVERSIE: LDI R16,0b11000000 ;zet start bit weer op 1
           STS ADCSRA,R16      ;start single conversion
```

```
CONVERS:  LDS R17,ADCSRA      ;controleer of conversie gereed is?
           SBRC R17,ADSC      ;skip als bit is nul wordt
           RJMP CONVERS
```

```
LDS R16,ADCH          ;alleen ADCH uitlezen voor 8-bits resolutie
OUT PORTA,R16
RJMP BEGIN
```

/ Vragen en opgaven*

PRINT MET DE LINEAIRE EN LOGARITMISCHE POTMETER

- 1. Download het programma en verdraai een potmeter en zie of de leds oplichten. Bedien de schakelaar S0 en verdraai de andere potmeter en zie of de leds oplichten.*
- 2. Met behulp van de schakelaar S0 kan de gewenste potentiometer worden geselecteerd. Vul de nu volgende tabellen in.*
- 3. Verklaar welke van de potmeters de lineaire resp. logaritmische is.*

1	2	3	4	5
stand potm.	U looper (v)	binair	hex.	dec.
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

1	2	3	4	5
stand potm.	U looper (v)	binair	hex.	dec.
0				
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				

Figuur 3

Op deze wijze kunnen natuurlijk ook analoge autotechnische sensoren worden uitgelezen. Te denken valt aan gasklephuizen met dubbele potentiometers.

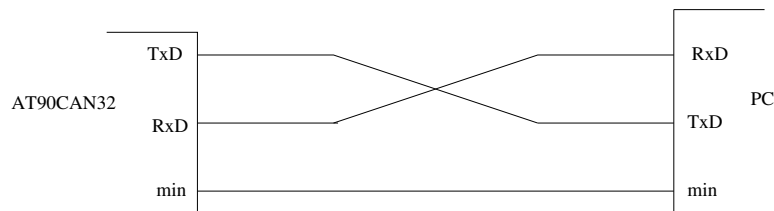
PRINT MET GASKLEPHUIS

- 1. Download het programma en verdraai de gasklep en zie of de leds oplichten. Bedien de schakelaar S0 en verdraai weer de gasklep en zie of de leds oplichten. Noteer en verklaar het verschil.*
- 2. Dus met behulp van de schakelaar S0 kan de gewenste gaskleppotentiometer worden geselecteerd. Indien aanwezig: voer nu opdracht uit van de praktijkopdracht 'Gasklephuis Monomotronic' op de AT90CAN32 processor.* /*

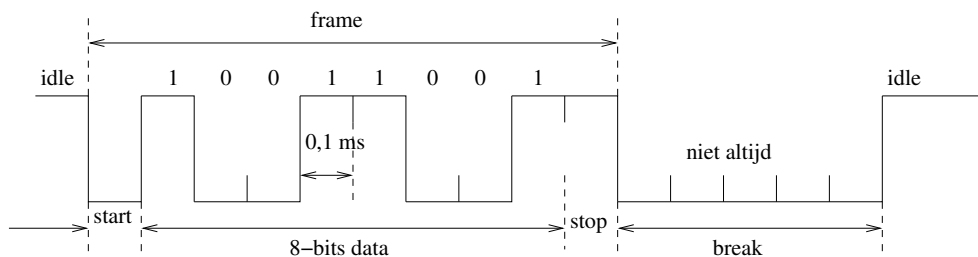
3 Seriële communicatie

De volgende twee programma's geven voorbeelden van seriële communicatie. Tot nu toe ging het om parallelle in- en output. Acht datalijnen zorgden voor de communicatie. Nu kunnen we de bits ook na elkaar in lezen en/of versturen. Dit wordt seriëel genoemd. Om het begin en het einde van een 8-bits databyte te onderscheiden wordt er een start- en een stopbit toegevoegd. Een belangrijk gegeven bij seriële communicatie is de baudrate. De baudrate van een seriëel signaal is het aantal bits dat per seconde verstuurd wordt. De baudrate kan door de programmeur worden ingesteld. In onze voorbeeld-programma's maken we gebruik van 1 startbit, 8 databits en een stopbit. Er wordt geen 'break' gegeven, d.w.z. de communicatie gaat continu door. De ingestelde baudrate bedraagt hier 9600 bits/sec. De AT90CAN32-microprocessor gebruikt hiervoor poort D. Twee pinnen zijn hiervoor gereserveerd, de TxD-(transmit)pin en de RxD-(receive)pin. Er worden dus voor het zenden en ontvangen twee verschillende aansluitingen gebruikt. Op de auto maakt de k-lijn gebruik van

deze communicatie mogelijkheid. Er wordt alleen een ééndraads structuur toegepast. Het zenden en ontvangen gaat dan door dezelfde draad. Om van tweedraads naar een ééndraads over te gaan (en omgekeerd) is een interface-schakeling nodig. Fig. 4 laat zien op welke wijze twee computers volgens de tweedraads structuur seriëel met elkaar verbonden moeten worden. Fig. 5 laat het principe van de seriële datastroom zien. Let erop dat het bitje met de laagste waarde normaal gesproken geheel rechts staat, maar omdat de oscilloscoop van links naar rechts schrijft komt het bitje met de laagste waarde op het scherm van de oscilloscoop geheel links te staan.



Figuur 4: Seriële transmissie tussen twee computers. De draden moeten worden gekruist.



Figuur 5: Opbouw van een dataframe met start en stopbit, een break is optioneel. Bij een baudrate van 9600 bits/s is de bittijd 0,1 ms.

*/*Programma naam: SERIEEL1.ASM*

Het programma zet de stand van de schakelaars op de leds en stuurt de schakelstand seriëel via de TxD(1) pin uit.

Oscilloscoop op pin 24 (TxD1) en pin 34 (min) van de 34-polige connector. USART1 wordt gebruikt.

*Er wordt hier gebruik gemaakt van AVR Studio 4. Programma en data in het Flash geheugen */*

`.DEVICE AT90CAN32`

`.INCLUDE "can32def.inc"` ; definitie poorten in aparte file

`RJMP RESET` ;spring naar het begin adres

*/*INITIALISATIE POORTEN*/*

```
RESET:      LDI R16,0xFF          ;zet alle pinnen van poortA op output
            OUT DDRA,R16        ;DDRA=Data-Directie-register

            LDI R16,0xFF        ;activeer de pull-up weerstanden
            OUT PORTC,R16       ;door enen te schrijven naar de inputpoort
            LDI R17,0x00        ;zet alle pinnen van poortC op input
            OUT DDRC,R17       ;niet strikt noodzakelijk (default waarde)
            NOP
```

*/*INITIALISATIE STACKPOINTER*/*

```
            LDI R16,high(RAMEND)
            OUT SPH,R16
            LDI R16,low(RAMEND)
            OUT SPL,R16
```

*/*INITIALISATIE UART1 */*

```
;init baudrate
;baudrate afhankelijk van klokfrequentie (hier 8 MHz)
;stel baudrate in op 9600 bits/s
```

```
            LDI R17,0b00000000
            LDI R16,0b01010001    ;51 hex volgens tabel boek
            STS UBRR1H,R17        ;baudrate register
            STS UBRR1L,R16
```

```
;set frame format, frame format door:
;USCR A/B/C Status Control Register
;USCROC laden met 0b00000110 start bit, 8 bits data, 1 stopbit
```

```
            LDI R16,0b00000110
            STS UCSR1C,R16
;activeer receiver en transmitter in UCSROB, laadt met 00011000
```

```
            LDI R16,0b00011000
            STS UCSR1B,R16
```

*/*HOOFDPROGRAMMA*/*

```

TRANSMIT:  LDS R17,UCSR1A      ;controleer of zendregister (Control Status R)
           SBRS R17,UDRE1      ;leeg is (DRE=data register empty)
                                           ;Skips next instruction if Bit in Register is Set

           RJMP TRANSMIT

           IN R16,PINC          ;lees schakelstand in
           OUT PORTA,R16       ;schakelstand op de leds
           STS UDR1,R16        ;data register
                                           ;nu naar Transmit Shift register als deze leeg is

           RJMP TRANSMIT

```

Vragen en opgaven

1. Download het programma en sluit een oscilloscoop aan op de TxD1-pin en de min. Zet de schakelaar in een willekeurige stand en bepaal de baudrate van het signaal.
2. Zet de schakelaars in de stand 00000000 en verklaar het oscilloscoopbeeld. Teken het signaalbeeld na en geef aan waar het stop- en het startbit zich bevindt.
3. Zet de schakelaars in de stand 00000001 en verklaar het oscilloscoopbeeld. Teken het signaalbeeld en geef aan waar bit0 zich bevindt.
4. Zet de schakelaars in de stand 11111111 en verklaar het oscilloscoopbeeld. Teken het signaalbeeld en geef aan waar het start- en stopbit zich bevindt.
5. Zet de schakelaars in de stand 10011011. Teken het signaalbeeld en geef aan waar bit0 en bit7 zich bevinden (Let op: de bits op het oscilloscoopscherm verschijnen van links naar rechts).*/

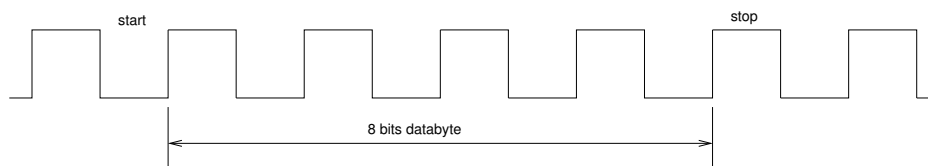
3.0.2 Toelichting

We dienen te controleren wanneer een byte in zijn geheel is verzonden voordat we opnieuw een byte willen gaan versturen. Dit gebeurt door het UDRE1 bitje (bit 5) in het UCSR1A-register te controleren. Hiervoor wordt de SBRS instructie gebruikt. Deze instructie (Skip if Bit in Register is Set) verlaat de lus wanneer bit 5 logisch 1 is geworden en de data byte is verzonden.

3.0.3 PROGRAMMA SERIEEL2

Het tweede seriële programma laat zien hoe we de seriële aangeboden data kunnen inlezen en vervolgens op de leds kunnen weergeven. We maken hiervoor gebruik van de functiegenerator die we instellen op een bloksignaal op TTL-niveau met een frequentie van 4800 Hz. Het signaal van de functiegenerator doet de leds om en om oplichten (fig. 6). We sluiten de functiegenerator aan op pin 23 en 34 (min) van de 34-polige connector.

/*Programma naam: SERIEEL2.ASM



Figuur 6: Het 4800 Hz signaal van de functiegenerator geeft een baudrate van 9600 bits/sec. en omvat ook de start- en stopbit.

Het programma leest een TTL-puls van de functiegenerator in en zet de verzonden bytes op de leds. We maken gebruik van een baudrate van 9600 bits/s (k-lijn VAG). We zetten hiervoor de frequentie van de functiegenerator op 4800 Hz. We sluiten de TTL-uitgang van de functiegenerator aan op pin 23 (RxD1) en pin 34 (min) van de 34-polige connector. Er wordt hier gebruik gemaakt van AVR Studio 4. Programma en data in het Flash geheugen./*

```
.DEVICE AT90CAN32
```

```
.INCLUDE "can32def.inc" ; definitie poorten in aparte file
```

```
RJMP RESET ;spring naar het begin adres
```

```
/*INITIALISATIE POORTEN*/
```

```
RESET: LDI R16,0xFF ;zet alle pinnen van poortA op output
        OUT DDRA,R16 ;DDRA=Data-Directie-register
```

```
/*INITIALISATIE STACKPOINTER*/
```

```
LDI R16,high(RAMEND)
OUT SPH,R16
LDI R16,low(RAMEND)
OUT SPL,R16
```

```
/*INITIALISATIE UART1 */
```

```
;init baudrate, stel baudrate in op 9600 bits/s
;baudrate afhankelijk van klokfrequentie (hier 8 MHz)
```

```
LDI R17,0b00000000
LDI R16,0b01010001 ;51 hex volgens tabel boek
STS UBRR1H,R17 ;baudrate register
STS UBRR1L,R16
```

```
;set frame format, frame format door:  
;USCR A/B/C Status Control Register  
;USCROC laden met 0b00000110 start bit, 8 bits data, 1 stopbit
```

```
LDI R16,0b00000110  
STS UCSR1C,R16
```

```
;activeer receiver en transmitter in UCSROB, laadt met 00011000
```

```
LDI R16,0b00011000  
STS UCSR1B,R16
```

```
/*HOOFDPROGRAMMA*/
```

```
RECEIVE: LDS R17,UCSR1A ;controleer of ontvangstregister vol is  
SBRS R17,RXC1 ;slaat volgende instructie over  
RJMP RECEIVE ;als bit in register is 1 (set)
```

```
LDS R16,UDR1 ;lees de databyte uit  
OUT PORTA,R16 ;zet de databyte op de leds
```

```
RJMP RECEIVE
```

```
/* Vragen en opgaven
```

```
Stel eerst met de oscilloscoop de frequentie van de functiegenerator nauwkeurig in op 4800 Hz.
```

- 1. Verklaar hoe het komt dat de frequentie van 4800 Hz overeenkomt met een baudrate van 9600 bits/s.*
- 2. Assembleer en download het programma en controleer of de leds om en om oplichten. Welk getal wordt nu door de functiegenerator uitgezonden? Verklaar dit.*
- 3. Verander de frequentie van de functiegenerator en noteer de frequentie waarbij geen goede communicatie meer mogelijk is (Gebruik de oscilloscoop).*/*

4 Externe interrupten

Wanneer bijv. de koelvloeistoftemperatuur te hoog wordt dan dient in elk geval de bestuurder te worden gewaarschuwd. Het lopende motormanagementprogramma moet dan even worden onderbroken, een waarschuwinglampje moet worden aangezet, waarna het programma kan worden vervolgd. Zo'n programma onderbreking noemt men een interrupt en het stukje software dat het lampje aanzet noemt men een 'interrupt service routine'. Dit is een eenvoudig

voorbeeld. In werkelijkheid maken veel sensoren gebruik van deze methode. Men onderscheidt interne en externe interrupten.

4.1 De interruptstructuur van de AT90CAN32

In een computerprogramma worden de instructies één voor één na elkaar uitgevoerd. Hiervoor wordt de programma-counter (PC) elke keer met 1 verhoogd. De programma-counter kan weliswaar door de diverse CALL-instructies worden beïnvloed, maar het programmaverloop blijft sequentiëel en volgt in principe zijn eigen instructievolgorde. In veel situaties zal het nodig zijn dat op verzoek van bijvoorbeeld een sensor een ander programmadeel wordt uitgevoerd. Deze verzoeken zijn a-synchroon met het hoofdprogramma en worden zoals reeds opgemerkt 'interrupts' genoemd. Het programma dat vervolgens wordt uitgevoerd, de interrupt service routine (ISR) noemt men ook wel een interrupt-handler. De krukhoeksensor van een auto zou ook een voorbeeld kunnen zijn. Op het moment dat het referentiepunt de sensor passeert, wordt een interrupt aangevraagd; de bijbehorende interrupt-handler bepaalt vervolgens het nieuwe ontstekingstijdstip. Nadat de interrupt-handler is uitgevoerd, wordt het onderbroken hoofdprogramma weer vervolgd.

Interrupt-handlers lijken veel op subroutines met dit verschil dat ze ook door de hardware kunnen worden opgeroepen. Ook worden tegelijk met de programma-counter de inhoud van de registers op de STACK gezet. In plaats van een RET-instructie aan het einde van een subroutine, eindigt een interrupt handler met een RETI-instructie (ReTurn Interrupt).

Interrupts kunnen in de software aan- en uitgezet worden (enable en disable). Dit dient te gebeuren met de SEI (enable) en de CLI (clear) instructie. De interrupt-structuur werkt met behulp van vectoren. Een vector wijst een adres aan waarin zich het eerste adres van de interrupt handler bevindt. De vectoren zijn genummerd en hebben een vaste bron. De Reset instructie kan worden beschouwd als een bijzondere hardware interrupt. De adressen van de Interrupt Service Routines bevinden zich in de eerste adressen van het programmeergeheugen (flash-geheugen). In het handboek van de AT90CAN32 bevindt zich de complete tabel van de interrupt vectoren. De volgende interrupten worden bij de AT90CAN onderscheiden:

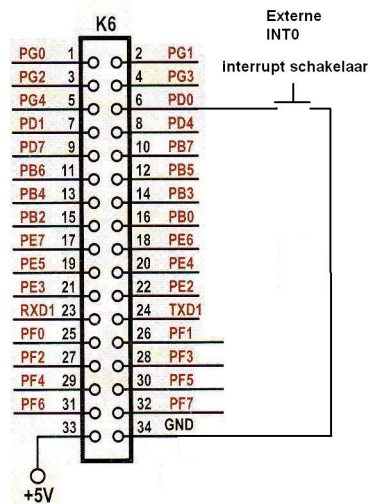
1. de externe hardware interrupts (RESET, INT0 t/m INT7);
2. de interne hardware interrupts (timer, seriële communicatie, AD conversie, CAN e.d.);

We geven een programma voorbeeld van een externe hardware interrupt. De AT90CAN32 kent er acht: INT0 t/m INT7. De vectoradressen 1 t/m 7 bevinden zich in de adressen 0002 t/m 0010h. Deze adressen moeten dus gevuld worden met de eerste adressen van de interrupt service handlers. Hier volgt een

overzicht van de externe hardware interrupten. De overige interrupten worden behandeld waar dat nodig is.

Vector	Adress	Code Label	handler van..
1	0x0000	jmp RESET	;Reset Handler
2	0x0002	jmp EXT_INT0	;IRQ0 Handler
3	0x0004	jmp EXT_INT1	;IRQ1 Handler
4	0x0006	jmp EXT_INT2	;IRQ2 Handler
5	0x0008	jmp EXT_INT3	;IRQ3 Handler
6	0x000A	jmp EXT_INT4	;IRQ4 Handler
7	0x000C	jmp EXT_INT5	;IRQ5 Handler
8	0x000E	jmp EXT_INT6	;IRQ6 Handler
9	0x0010	jmp EXT_INT7	;IRQ7 Handler

Aan het begin van het programma kunnen we de vectoradressen toewijzen. De interrupt handler dient dan het bijbehorende label te krijgen. De interrupt pinnen komen via de 34-polige connector naar buiten. De externe interrupten zijn de derde functie van poort D en poort E. In ons voorbeeld programma maken we gebruik van een schakelaar die we op pin 6 van de 34-polige connector aansluiten (fig. 7). Het geheel werkt dan als het volgt: wanneer we



Figuur 7: Een interrupt-drukschakelaar kan worden aangesloten op pin 6 van de 34-polige connector van de Timloto-Elektor controllerset. Pin 6 van de connector is verbonden met pin 25 van de AT90CAN-controller.

de drukschakelaar bedienen dan wordt pin 25 van de controller kortstondig

aan de massa gelegd. Hierop volgt een interrupt. Het programma wordt onderbroken en in de Program Counter wordt het adres geladen dat zich in adres 0002 en 0003 bevindt. Het terugkeeradres wordt met de andere registers op de STACK gezet. In het begin van het programma wordt het vectoradres geladen. Dit gebeurt met de JMP-instructie. Is de interrupt handler uitgevoerd dan wordt het programma vervolgd. Zie programma INTERRUPT1.

*/*Programma naam: INTERRUPT1.ASM*

Poort D input voor interrupt en poort A output (Poort C wordt hier niet gebruikt). Het programma laat led0 knipperen. Wanneer op de interrupt-schakelaar wordt gedrukt gaat led7 gedurende 1 sec. aan. Hierna knippert Led0 weer. De drukschakelaar verbindt pin 25 (PDO INTO) van de controller kortstondig met de min. Er wordt hier gebruik gemaakt van AVR Studio4./*

.DEVICE AT90CAN32

.INCLUDE "can32def.inc" ;definitie poorten in aparte file

*/*INITIALISATIE INTERRUPTEN*/*

JMP RESET ;spring naar het beginadres (interrupt)
JMP EXT_INT0 ;pointer naar ISR (Interrupt handler)

*/*INITIALISATIE*/*

RESET: LDI R16,high(RAMEND)
OUT SPH,R16
LDI R16,low(RAMEND)
OUT SPL,R16 ;stackpointer initialisatie nodig voor subroutine
;en interrupts
LDI R16,0xFF ;zet alle pinnen van poortA op output
OUT DDRA,R16 ;DDRA=Data-Directie-register

LDI R16,0xFF ;activeer de pull-up weerstanden
OUT PORTD,R16 ;door enen te schrijven naar de inputpoort
LDI R17,0x00 ;zet alle pinnen van poortD op input
OUT DDRD,R17 ;niet strikt noodzakelijk (default waarde)
NOP ;interrupt ook op Output

*/*Sta interrupt toe*/*

LDI R16,0x01
OUT EIMSK,R16 ;stel poort D bit 0 in op Externe Interrupt
STS EICRA,R16 ;interrupt vindt nu plaats op een logische verandering

```

                SEI                                ;enable interrupts (sets I-bit in SREG)
                                                ;SEI na interrupt poortinitialisatie!

/*HOOFDPROGRAMMA*/

BEGIN:         LDI R16,0x00
                OUT PORTA,R16                    ;zet leds uit

                CLI                              ;sta geen interrupt toe tijdens wachtlus
                RCALL WACHT1
                SEI                              ;sta interrupt weer toe

                LDI R16,0x01
                OUT PORTA,R16                    ;zet led aan

                CLI                              ;clears interrupt (resets I-bit in SREG)
                RCALL WACHT1
                SEI                              ;sta interrupt weer toe

                RJMP BEGIN

/*SUBROUTINE WACHT*/

WACHT1:        LDI R20,0x0F                      ;0x0F (01 voor debugger)
WACHT:         LDI R18,0xFF                      ;0xFF (01 voor debugger)
WEER:          LDI R19,0xFF                      ;0xFF (01 voor debugger)
LUS:           SUBI R19,0x01
                BRNE LUS
                SUBI R18,0x01
                BRNE WEER
                SUBI R20,0x01
                BRNE WACHT
                RET                              ;keer terug naar het hoofdprogramma

/* INTERRUPT HANDLER of Interrupt Service Routine */

EXT_INT0:      LDI R16,0x80
                OUT PORTA,R16                    ;zet led7 aan

                LDI R20,0x0F                      ;wachtlus
WAIT:          LDI R18,0xFF
AGAIN:         LDI R19,0xFF
LOOP:         SUBI R19,0x01

```

```

BRNE LOOP
SUBI R18,0x01
BRNE AGAIN
SUBI R20,0x01
BRNE WAIT

RETI                ;return van Interrupt

```

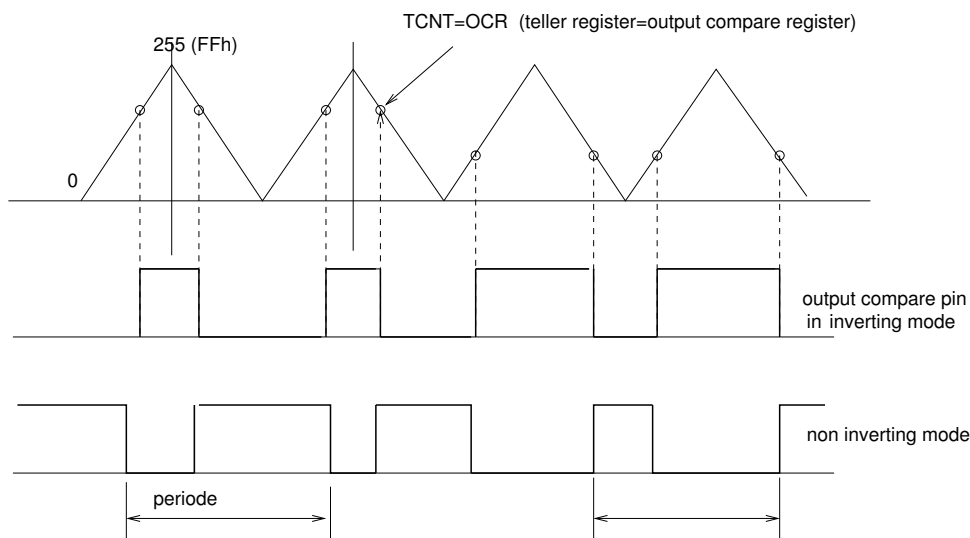
/ Vragen en opgaven*

1. *Wat zou het verschil zijn tussen een hardware en een software interrupt?*
2. *Met wat voor een interrupt hebben we in het programma te maken?*
3. *Wat zouden we verstaan we onder een pointer-adres?*
4. *Het EICRA register wordt met 01 geladen. Wat stelt men hiermee in?*
5. *CLI en SEI staan voor en achter de 'CALL' naar 'WACHT1' Wat is de bedoeling hiervan?*/*

5 Timer-Counters

5.1 De 8-bit Timer/Counter0 in PWM-mode

Timer/counters maken gebruik van een interne of externe klok (oscillator). In het programmeervoorbeeld wordt gebruik gemaakt van de interne I/O klok. Een timer telt geheel buiten de processor om tot een bepaalde waarde (tot 255 voor een 8 bits of tot 65535 voor een 16 bits timer) en begint dan weer overnieuw of begint terug te tellen. De processor kan bijv. het telregister gedurende het tellen uitlezen of dat vergelijken met een waarde in een ander register. De timer/counter kent verschillende modes. We beschrijven hier de 'PWM, fase correct' mode. In de 'fase correct mode' telt de timer tot de maximale 8-bits waarde (FFh) en begint dan weer terug te tellen tot de minimum waarde (00h). Grafisch gezien ontstaat dan een zaagtandvorm (fig. 8). Op het moment dat de counter de waarde bereikt die in het Output Compare Register (OCR0A) is geschreven slaat het niveau van de outputpin (OC0A-pin) om. Men onderscheidt dan de invertering en non invertering mode. Deze PWM-methode wordt dual slope genoemd omdat beide flanken actief zijn. De inhoud van het OCR0A-register wordt dus voortdurend vergeleken met TCNT0-register. Er kan ook een Output Compare Interrupt worden opgewekt. Dit laten we hier buiten beschouwing. De frequentie van het signaal wordt bepaald door de kloksnelheid. De maximale snelheid is gelijk aan de systeemklok. De klokfrequentie kan echter ook worden teruggebracht. De kloksnelheid kan dan door 8, 64, 256 of 1024 worden gedeeld. Dit gebeurt door een zgn. prescaler.



Figuur 8: De PWM-mode van de AT90CAN-controller. Het niveau van de outputpin slaat om op het moment dat een bepaalde tellerwaarde wordt bereikt. Er is een invertering en een non invertering mode.

5.1.1 Het programma

De uitgangspin is pin 17 (bit 7 van poort B) van de controller en pin 10 van de 34-polige connector. Poort B wordt dus in de CounterTimer mode gezet. De pin krijgt dan de naam: OCR0A (Output compare Channel A). De betreffende pin van poort B moet dan als outputpoort worden gedefiniëerd. De belangrijkste 8-bits register zijn:

- TCNT0 Timer/counter register (hierin vindt het tellen plaats);
- OCR0A Het vergelijkingsregister (compare).

We zullen echter eerst de klok moeten gaan instellen. Dit gebeurt door de Clock Select bits (CS0) in het in het Timer/Counter Control Register (TCCR0A). De CS-bits zijn de eerste drie bits (0 t/m 2) van het register. De betekenis is als het volgt:

CS02 CS01 CS00 beschrijving

0	0	0	Timer /counter stopt
0	0	1	Interne klok geen prescaling (23,6 kHz)
0	1	0	Interne klok / 8 (prescaling)
0	1	1	Interne klok / 64 (prescaling)
1	0	0	Interne klok / 256 (prescaling)
1	0	1	Interne klok / 1024 (prescaling)
1	1	0	Externe klok op T0 (klok op neergaande flank)
1	1	1	Externe klok op T0 (klok op opgaande flank)

We zetten de prescaler op 256 (100b op de bits). Met bit 3 en 6 kunnen de de PWM mode 'fase correct' instellen. We schrijven hiervoor '10'. Met bit 5 en 4 stellen we Compare mode in. We kunnen 10 of 11 gebruiken. Bit 7 kan altijd 0 zijn. Dus:

FOCA	WGM00	COM0A1	COM0A0	WGM01	CS02	CS01	CS00
0	1	1	1	0	1	0	0

Resteert het Ouput Compare register (OCR0A). Hier kunnen we een getal tussen de 00h en FFh inzetten. Met deze waarde bepalen we de duty-cycle. De frequentie kan ingesteld worden met behulp van de CS0-bits. Zie het voorbeeld programma PWM1.

*/*Programma naam: PWM1.ASM*

*Poort A output, poort C input (default). Wordt echter hier niet gebruikt. Op bit7 van Poort B kan een PWM-sigitaal met een oscilloscoop worden gemeten. Scoop aansluiten op pin 10 en 34 van de 34-polige connector. Om de duty-cycle in te stellen moet de waarde in het OCR0A-register worden veranderd. Er wordt hier gebruik gemaakt van AVR Studio 4. */*

.DEVICE AT90CAN32

.INCLUDE "can32def.inc" ;definitie poorten in aparte file
.INCLUDE "timlotodef.inc" ;eigen initialisaties

*/*INITIALISATIE */*

```

LDI R16,0xFF           ;zet alle pinnen van poort B op output
OUT DDRB,R16          ;DDRA=Data-Directie-register
                       ;GTCCR in default waarde

```

/ HOOFDPROGRAMMA */*

```

LDI R16,0x80           ;waarde tussen 00h en FFh
OUT OCR0A,R16         ;duty-cycle wordt hierdoor ingesteld

LDI R16,0b01110100    ;instelling timer 01110100 Zie theorie.
OUT TCCR0A,R16        ;PWM-mode ingesteld, interne I/O klok/256

```

*/*Vragen en opgaven*

1. *Sluit de oscilloscoop aan tussen pin 10 en 34. Assembleer en download het programma en bepaal de duty-cycle (positief actief) en de frequentie van het signaal.*

2. Stel nu de duty-cycle in op 10% resp. 90% door de waarde in het OCROA-register te wijzigen.
3. De drie meest rechtse bits die in het TCCROA register staan bepalen de frequentie van het signaal. Van hoogste frequentie naar laagste frequentie: 001, 010, 011, 100, 101. Wat is nu de laagste en de hoogste frequentie van het signaal? Opm: Verander alleen de laagste drie bits!!
4. De duty cycle kan worden geïnverteerd door bit 4 en 5 te veranderen in TC-CROA. In het programma wordt 11b gebruikt. Verander deze waarde maar eens in 10b en zie wat er gebeurt. Dus veranderen in: 01 10 0100 */

Programma PWM2 maakt dat het PWM-signaal met de potmeter kan worden ingesteld. De programma's worden nu wat gecompliceerder omdat we diverse zaken gaan combineren. We zouden nu de AT90CAN32 kunnen gaan gebruiken om PWM-kleppen aan te sturen. We moeten dan natuurlijk wel gebruik maken van een driver (transistor). Hier volgt het programma.

*/*Programma naam: PWM2.ASM*

Programma voor de AT90CAN32 Elektor-Timloto print. Het programma leest de stand van potmeter in. Deze waarde is een maat voor de duty-cycle van het PWM-signaal. De waarde van de potmeter wordt tevens op de leds weergegeven. Er wordt hier gebruik gemaakt van AVR Studio 4. Potmeter op de print is aangesloten op PF0 (AD0). Denk om de jumper! De oscilloscoop wordt aangesloten pin 10 (PB7) en pin 34 (min) van de 34-polige connector om het PWM-signaal te bestuderen. Programma en data in het Flash-geheugen./*

.DEVICE AT90CAN32

.INCLUDE "can32def.inc" ;definitie poorten in aparte file

RJMP RESET ;spring naar het begin adres

*/*INITIALISATIE POORTEN*/*

*RESET: LDI R16,0xFF ;zet alle pinnen van poortA op output
OUT DDRA,R16 ;DDRA=Data-Directie-register
OUT DDRB,R16*

*/*INITIALISATIE ADC */*

*;selecteer het eerste kanaal (bit 0 t/m 4) door MUX bits te setten
;kanaal 0 is 00000
;set de ADLAR bit, bit 5 (left adjusted, nodig voor 8 bits precisie)*

LDI R16,0x20 ;0b00100000

STS ADMUX,R16

*;enable ADC door ADEN bit (bit7) te setten
;start single bit conversion, set bit ADSC (bit6)
;bit 6 wordt 0 als de conversie is beëindigd*

LDI R16,0b11000000
STS ADCSRA,R16

;lees alleen ADCH uit voor een 8 bits resolutie

*/*HOOFDPROGRAMMA*/*

BEGIN: LDI R16,0b11000000 ;zet start bit weer op 1
STS ADCSRA,R16 ;start single conversion

CONVERS: LDS R17,ADCSRA ;controleer of conversie gereed is?
SBRC R17,ADSC ;skip als bit is nul wordt
RJMP CONVERS

LDS R16,ADCH ;alleen ADCH uitlezen voor 8 bits resolutie
OUT PORTA,R16 ;leuk om te zien

/ HOOFDPROGRAMMA TIMER */*

OUT OCR0A,R16 ;duty-cycle wordt hierdoor ingesteld
LDI R16,0b01110100 ;instelling timer 01110100 Zie theorie
OUT TCCR0A,R16 ;PWM-mode ingesteld, interne I/O klok/256

RJMP BEGIN

/ Vragen en opgaven*

1. Sluit de oscilloscoop aan tussen pin 10 en 34 van de 34-polige connector. Maak een grafiek die de waarde van de potmeter (0-255) uitzet tegen de duty cycle in procenten (0-100%). Maak stappen van 10%. Gebruik hiervoor de oscilloscoop. */
2. Verklaar hoe het hoofdprogramma werkt.