4-7-2017

Nick van Gils

AVANS BREDA

A6 – jaar 2

2099527

Groeidocument

Microcontrollers

Inhoudsopgave

[Week 1 4](#_Toc479325243)

[Week 2 8](#_Toc479325244)

[Week 3 12](#_Toc479325245)

[Week 4 15](#_Toc479325246)

[Week 6 17](#_Toc479325247)

[Bronvermelding (handige links) 21](#_Toc479325248)



* Alle gemaakte code is te vinden onder <https://github.com/nickvgils/Microcontrollers.git> gestructureerd in de verschillende opgaves.

# Week 1

B1.

**Vraag:** Hoe groot is het program memory van de ATmega128?

**Antwoord:** 128K bytes

**Vraag:** Wat is het adres van Data direction register van PORTE (DDRE)?

**Antwoord:** $03

**Vraag:** Uit hoeveel byte bestaat de instructie ‘IN R3, PORTA’ ?

**Antwoord:** 2 bytes

**Vraag:** Hoeveel RS232 poorten zitten er op het BIGAVR6 development board?

**Antwoord:** Twee

**Vraag:** Op welke pin van de microcontroller zit de ingang voor Analog digitaalconverter, channel 1?

**Antwoord:** 61

**Vraag:** Hoe groot is het data geheugen van de microcontroller maximaal?

**Antwoord:** 8KB

**Vraag:** Hoeveel I/O-registers zijn er op de ATmega128?

**Antwoord:** 6

**Vraag:** De pinnen van PORTA kunnen met een weerstand naar 0 V (pull-down) of met een weerstand naar de +5V verbonden worden (pull-up). Hoe is dat standaard ingesteld op het BIGAVR6 development board?

**Antwoord:** 0 V (pull-down)

B2.

**Opdracht:** Maak een nieuwe applicatie die beurtelings de LED op PORTD, pin 7 (PORTD.7) en de LED op PORTD, pint 6 (PORTD.6) om de 500ms laat oplichten.

**Code:**

DDRD = 0b11111111; // All pins PORTD are set to output

while (1)

{

PORTD = 0b10000000; // PortD: Pin 7 of set to on, rest off

wait( 500 );

PORTD = 0b01000000; // PortD: Pin 6 set to on, rest off

wait( 500 );

}

B3.

**Opdracht:** Maak een applicatie die de led op PORTD.7 laat knipperen als drukknop PORTC.0 laag (0) is (ingedrukt) en stopt bij het loslaten van de drukknop.

**Code:**

DDRD = 0b11111111; // All pins PORTD are set to output

DDRC = 0b11111110; // All pins PORTC are set to output, except pin 0

while (1)

{

if(PINC & 1) // If Pin 0 (0b00000001) is pressed

{

PORTD = 0b10000000; // PortD: Pin 7 of set to on, rest off

wait( 500 ); // Wait 0,5 sec

PORTD = 0b00000000; // PortD: all pins are off

wait( 500 ); // Wait 0,5 sec

}

}

B4.

**Opdracht:** Implementeer een looplicht applicatie op de LED’s van PORTD. Tussen elke verandering van output zit 50ms (milliseconden).

**Code:**

DDRD = 0b11111111; // All pins PORTD are set to output

while (1)

{

if(PORTD == 0b10000000 || PORTD == 0b00000000) // PortD: If it is the //first time or the last pin in the row (7) 🡪

{

PORTD = 0b00000001; // PortD: Turn pin 0 on

}

else{

PORTD = PORTD << 1; // PortD is bitshifting one to the left

}

wait( 2000 ); // Wait 2 sec

}

B5.

**Opdracht:** Implementeer een patroon in de LEDS door gebruik te maken van een struct structuur die herhaaldelijk kan worden afgespeeld.

**Code:**

typedef struct {

unsigned char data;

unsigned int delay ;

} PATTERN\_STRUCT; // A struct has been defined as an binary data element and a preferred // delay. Saved as PATTERN\_STRUCT

PATTERN\_STRUCT pattern[] = {

{0b10000001, 2000}, {0b01000010, 2000}, {0b00100100, 2000}, {0b00011000, 2000}

}; // A new PATTERN\_STRUCT has been made with a pattern of different binaries.

DDRD = 0b11111111; // All pins PORTD are set to output

int index = 0; // Counter

while (1)

{

if(index > 4) // If the counter has reached the last element

{

index = 0; // index back to zero to display the first element

}

// Write data to PORTD

PORTD = pattern[index].data; // portD is set to the data field of the // pattern index

wait(pattern[index].delay); // Wait is set to the delay field of the // pattern index

index++; // increment for next round

}

B6.

**Opdracht:** Maak een applicatie die de led op PORTD.7 laat knipperen met een frequentie van circa 1Hz (1 keer per seconde). Als nu PORTC.0 kort wordt ingedrukt gaat (en blijft) de led sneller knipperen (bijvoorbeeld 4H). Bij nogmaals kort drukken gaat (en blijft) de led weer knipperen met een frequentie van 1Hz.

**Code:**

DDRD = 0b11111111; // All pins PORTD are set to output

DDRC = 0b11111110; // All pins PORTC are set to output, except pin 7

int delay = 1000; // Saves delay variable

while (1)

{

if(PINC & 0b01) // If Pin 0 (0b00000001) is pressed

{

if(delay == 1000) // If delay is equal to 1000

{

delay = 4000; // delay set to 4000

}

else{ // Otherwhise (delay is then 4000)

delay = 1000; //delay set to 4000

}

}

PORTD = 0b10000000; // PortD turn pin 0 on

wait( delay ); // Wait “delay” ms

PORTD = 0b00000000; // PortD turn pin 0 off

wait( delay ); // Wait “delay” ms

}

# Week 2

A

**Opdracht:** Elliot Williams bespreekt op hackaday.com ‘Interrupts, the good, the bad and the ugly’ in 3 weblogs. Bestudeer deze artikelen en beschrijf in enkele zinnen de kerngedachten uit de 3 weblogs. Geef ook enkele voorbeelden.

**Antwoord:**

B.1

**Opdracht:** Download ioisr.c uit de repository en executeer deze op het BIGAVR board. Verklaar de werking.

**Antwoord:**

In de main loop wordt eerst voor poort D pin 0 t/m 3 input gezet en 4 t/m 7 output. Vervolgens wordt de interupt hardware instellingen gedaan. Er wordt gekozen voor een rising edge voor pin 0 en een falling edge voor pin 1 (drukknoppen/input). Daarna worden pin 0 en 1 gezet als interrupt hardware en wordt het interrupt systeem nog aan gezet.

Pin 7 wordt continu getoggled in de while loop.

Als er op drukknop/pin 0 wordt gedrukt triggerd deze meteen vanwege de rising edge instelling en de processor voert als hoogste prioriteit taak de ISR( INT0\_vect ) methode uit. Deze methode laat pin 5 altijd branden wanneer deze aangeroepen wordt.

Als er op drukknop/pin 1 wordt gedrukt triggerd deze pas nadat de knop wordt losgelaten, dit vanwege de falling edge. De processor voert dan als hoogste prioriteit taak de ISR( INT1\_vect ) methode uit. Deze methode laat pin 5 altijd uitzetten wanneer deze aangeroepen wordt.

B.2

**Opdracht:** Implementeer een looplicht applicatie waarbij de main() bestaat uit een initialisatie gedeelte met daarna een lege while(true) loop. In ISR\_INT1 en ISR\_INT2 maakt het looplicht steeds 1 stap. Het looplicht wordt dus gestuurd vanuit de ISR’s, niet vanuit de main().

**Code:**

// Init ISR

ISR( INT0\_vect ) // called when input 0 changes

{

wait(3000); // wait 3 sec

PORTD = 0b00000010; // turn on pin 1 (LED)

}

ISR( INT1\_vect ) // called when input 1 changes

{

wait(3000); // wait 3 sec

PORTD = 0b00000100; // turn on pin 2 (LED)

}

ISR( INT2\_vect ) // called when input 2 changes

{

wait(3000); // wait 3 sec

PORTD = 0b00000001; // turn on pin 0 (LED)

}

// Init I/O

DDRD = 0b11111111; // PORTD(0:7) output

// Init Interrupt hardware

EICRA |= 0b00111111; // INT2, INT1, INT0 rising edge

EIMSK |= 0b00000111; // Enable INT1 & INT0

// Enable global interrupt system

//SREG = 0x80; // Of direct via SREG of via wrapper

sei();

PORTD = 0b00000001; // pin 0 triggers a rising edge

while (1)

{

wait( 10000 ); // wait 1 sec, does nothing in the while loop

}

return 1;

B3.

**Opdracht:** Implementeer een functie waarmee een 7 segment display kan worden aangestuurd. Bijvoorbeeld: void display(int digit){…}. Als 0 <= digit <= 15 dan wordt ‘digit’ hexadecimaal weergegeven op het 7 segment display, als ‘digit’ > 15 verschijnt op het display de ‘E’ van Error.

Implementeer ook de functionaliteit dat de waarde op het display steeds met 1 wordt verhoogd / verlaagd bij het indrukken van toetsen (up / down). Als je op beide toetsen tegelijk drukt wordt het display gereset naar ‘0’.

**Code:**

const unsigned char nummers[16] =

{

0b00111111, // 0

0b00000110, // 1

0b01011011, // 2

0b01001111, // 3

0b01100110, // 4

0b01101101, // 5

0b01111101, // 6

0b00000111, // 7

0b01111111, // 8

0b01101111, // 9

0b01110111, // A

0b01111111, // B

0b00001111, // C

0b01011110, // D

0b01111001, // E

0b01110001 // F

};

void display(int digit){

if(digit <= 15) // if digit below 16

{

DDRB = nummers[digit]; // digit is being displayed, number from lookup table

}else{

DDRB = nummers[14]; // else error sign(E), number from lookup table

}

}

// Init I/O

PORTB = 0b11111111; // PORTB input

PORTC = 0b00000000; // PORTC output

int counter = 0; // hexdecimal counter

display(counter); //display a hex number, calles display method

while(1)

{

if(PINC & 1) // if PORTC pin 0 (button) is pressed

{

counter++;

display(counter); // displays the counter

}if(PINC == 0b00000010){ // if PORTC pin 1 (button) is pressed

counter--; // decrease the counter with 1

display(counter); // displays the counter

}if(PINC == 0b00000011){ // if PORTC pin 0 and pin 1 (buttons) are pressed

counter = 0; // counter reset to 0

display(counter); // displays the 0

}

wait(1000); // waits for 1 sec

B4. (<https://youtu.be/57MF4V3HS1w>)

**Opdracht:** Bestudeer de code uit de repository uit lesweek 1 (lookup.c) waarin met behulp van een lookup-table een patronen op leds worden afgebeeld. Implementeer een ‘7 segment display lichteffect’ met behulp van deze techniek.

**Code:**

PATTERN\_STRUCT pattern[] = {

{0b00000001, 5000}, {0b00000010, 5000}, {0b00000100, 5000}, {0b00001000, 5000},{0b00010000, 5000},{0b00100000, 4000}

}; // circle pattern, walking light

// Init I/O

PORTB = 0b11111111; // PORTB input

int index = 0; // Counter

while (1)

{

// Write data to PORTB

DDRB = pattern[index].data; // portB is set to the data field of the // pattern index

wait(pattern[index].delay); // Wait is set to the delay field of the // pattern index

if(pattern[index].delay == 4000) // If the counter has reached an element with a delay of 4000

{

index = 0; // index back to zero to display the first element

}else{

index++; // increment for next round

}

}

# Week 3

A.1

void init(void){ //initializes the LCD

DDRC = 0b11111111; // All the pins of port C to output mode

PORTC = 0b00000000; // All pins are off (0V)

// return home

lcd\_command( 0x20 );

// mode: 4 bits interface data, 2 lines, 5x8 dots

lcd\_command(0x28);

// display: on, cursor off, blinking off

lcd\_command(0x0F);

////// entry mode: cursor to right, no shift

lcd\_command(0x06);

////// RAM adress: 0, first position, line 1

Lcd\_command(0x80);

}

A2.

**Vraag:** Verklaar de functie van de D0-D7 pinnen, de E pin en de RS pin

**Antwoord:**

De D0-D7 staan voor de 8bits pinnen die de verschillende data kunnen verzenden. Voor 4-bits mode is alleen D4-D7 aangesloten.

De E(nable) pin ontvangt de data op de D0-D7 pinnen wanneer deze hoog gezet wordt.

De RS pin, RS = on 🡪 verwacht data

RS = off 🡪 verwacht commands

A3.

**Vraag:** Het afbeelden van enkele karakters op het display. Welke data, commando’s en/of instellingen moet je maken? Vertaal deze handelingen in een C functie.

void lcd\_write\_data(unsigned char byte) //needs a character

{

// first 4 bits of the character writing

PORTC = byte; // sets PORTC to the character

PORTC |= 0b00000100; //enables the RS pin in data mode

strobe(); //sets the enable pin rising and back into falling to send the message

PORTC = (byte << 4); // sets the last 4 bits PORTC to the character

PORTC |= 0b00000100;

strobe();//sets the enable pin rising and back into falling to send the message

}

A4.

**Vraag:** Het schuiven van alle tekst naar rechts. Welke data, commando’s en/of instellingen moet je maken? Vertaal deze handelingen in een C functie.

**Antwoord:**

Lcd\_command(0b0011 1000*of*0x38);

B1. *(zie A1, A3 voor meer code (init en lcd\_write\_data) );*

**Vraag:** Ontwerp een LCD ‘C module’ (dus een \*.c en een \*.h file, bijvoorbeeld lcd.c en lcd.h). Implementeer (in een aparte file) een test programma om bovenstaande te testen.

**Antwoord:**

void display\_text(char \*str){

//loops through all the characters and calls the lcd\_write\_data function for each one

for(;\*str; str++){

lcd\_write\_data(\*str);

}

}void set\_cursor(int position){

////// RAM adress: 0, first position, line 1

if(position < 68){ // check if the prefered cursor position will fit into the screen

position |= 0b10000000; //to set the DDRAM Adress, DB7 has to be on.

lcd\_command(position); // writes the first 4 bit command to the LCD

lcd\_command(position << 4); //writes the last 4 bit command to the LCD

}

}

init(); //calls the lcd.h include method

clear(); //clears the screen

set\_cursor(0); // sets cursor to 0 position

display\_text("joel"); //displays text on the screen , in this case "joel"

set\_cursor(10); //changes the cursor position

display\_text("hoi"); //displays text on the screen , in this case "hoi"

while(1)

{

wait(100); // wait 0,1 sec, does nothing in the while

}



B2. (<https://youtu.be/Xg4lnXr3WeI>)

**Vraag:** Realiseer met AVR T/C-2 (counter mode) een ‘teller’ die het aantal keren ‘toets indrukken’ telt en weergeeft op het LCD.

**Antwoord:**

#include <stdio.h>

init(); //initialize the LCD

DDRD = 0x00; // PORTD PIN 7 op input, for the counter

TCCR2 = 0x07; // TCCR2 on a rising edge, (pin 7 rising)

char buffer[20]; //text buffer

while(1)

{

clear(); // clear screen

set\_cursor(0); //set cursor on position 0

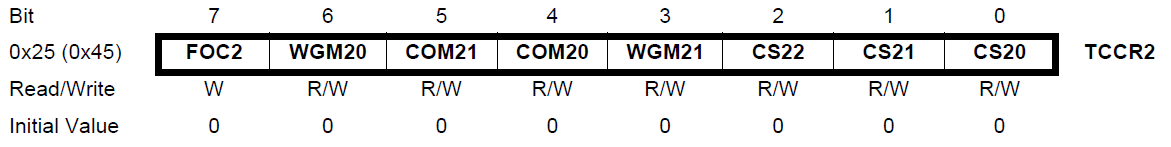
*sprintf*(buffer, "%d", TCNT2); //a binairy number casting to a char array buffer

display\_text(buffer); //displays the counter buffer

wait(2000); //wait 2 sec

}

B3.



ISR( TIMER2\_COMP\_vect )

{

PORTD ^= BIT(7); // Toggle bit 0 van PORTC

if(PORTD == 0){

TCNT2 = 0;

OCR2 = 195.3;

}else{

TCNT2 = 0;

OCR2 = 117;

}

}

void timer2Init( void )

{

OCR2 = 117; // Compare value of counter 2

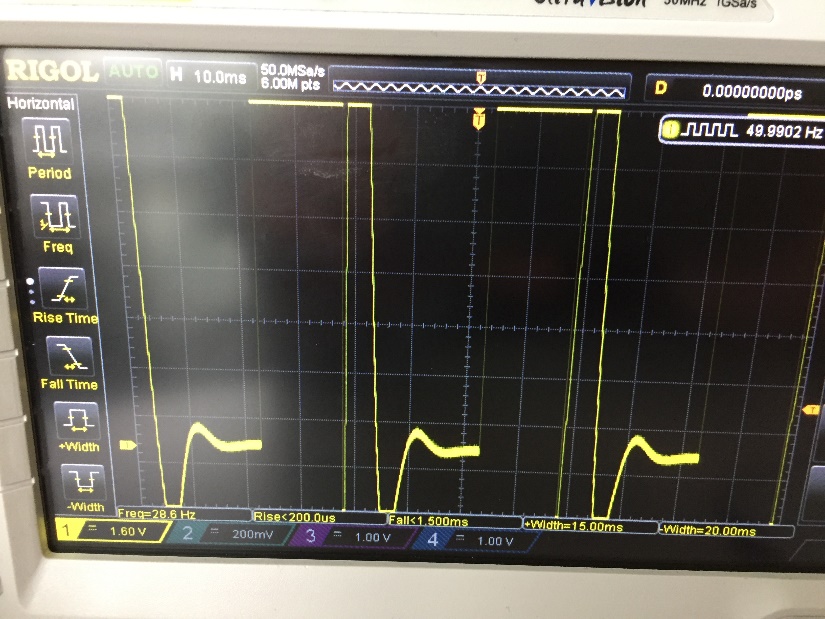
TIMSK |= BIT(7); // T2 compare match interrupt enable

SREG |= BIT(7); // turn\_on intr all

TCCR2 = 0b00001101; // Initialize T2: timer, prescaler=32,

// compare output disconnected,CTC,RUN

}



**Vraag:** Zet de prescaler, in deze opdracht, op 1024. Wat betekent dit?

**Antwoord:** de normale klokpulsen zijn 1/8 microseconde gestuurd.

**Vraag:** Met welke frequentie wordt Timer2 dan aangestuurd?

**Antwoord:** 8/1024 = 7812.5 Hz (1/128 MHz)

**Vraag:** Met welk tijdsinterval telt Timer2?

**Antwoord:** Wanneer prescaler 1024 wordt gezet is 8/1024 = 78.125 pulsen per seconde.

**Vraag:** Wat is de waarde in TCNT2 die je moet instellen om 15ms af te tellen?

**Antwoord:** 117

**Vraag:** Kun je dan een interrupt genereren en daar een I/O pin toggelen?

# Week 4

B1.

**Opdracht:** Laat de 10 bits AD-waarde van kanaal 1 op de led’s van Poort A en B zien. Bij Vin=0V zijn alle led’s uit, bij Vin = Vref zijn alle leds aan!

**Antwoord:**

ADMUX = 0b01100001;

ADCSRA = 0b11100110;

// input 1: PF1

// single ended

// prescaler = 64 (fosc = 8MHz à 125 kHz <200 kHz)

// continuous mode

// no interrupt

// 10 bits, left positioned

// Vref = 5V (intern Vcc)

lcd\_init(); // initialize lcd

while (1)

{

PORTB = ADCL; //low register output of the adc converter shown at B leds

PORTA = ADCH; //high register output of the adc converter shown at A leds

char str[15];

*sprintf*(str, "%d", ADCH);

lcd\_writeLine(str); // show the high register output on the lcd

wait(300);

lcd\_clear();

B2.

**Opdracht:** Verander het programma (voor kanaal 3 en voor 8 bits) zodat het alleen een AD-conversie uitvoert als jij dat wilt, dus op aanvraag.

**Antwoord:**

ADMUX = 0b01100011;

ADCSRA = 0b10000110;

// input 3: PF3

// single ended

// prescaler = 64 (fosc = 8MHz à 125 kHz <200 kHz)

// conversion on request

// no interrupt

// 10 bits, left positioned

// Vref = 5V (intern Vcc)

while (1)

{

ADCSRA |= BIT(6); // Start ADC

while ( ADCSRA & BIT(6) ) ; // Wait for completion

PORTB ^= 0x01;

PORTA = ADCH; // Show MSB (bit 9:2) of ADC

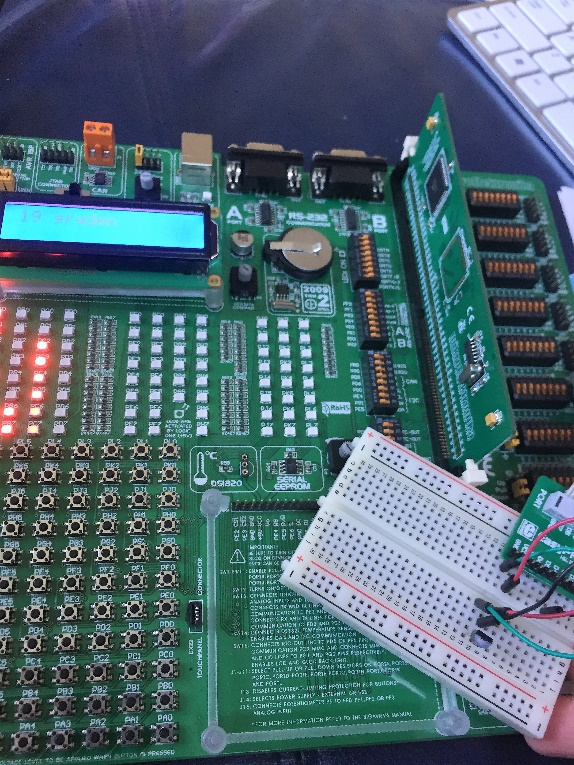
wait(500); //wait 0,5 sec for a new conversion

}

B3.

**Opdracht:**

Laat de waarde van ADCH op de LED’s van een poort zien. • Laat de waarde van temperatuur op het LCD-display verschijnen.

**Antwoord:**

ADMUX = 0b11100011;

ADCSRA = 0b11100110;

// input 3: PF3, thermometer on this pin

// single ended

// prescaler = 64 (fosc = 8MHz à 125 kHz <200 kHz)

// free running mode

// no interrupt

// 10 bits, left positioned

// Vref = 2,5V (intern Vcc)

lcd\_init();

while (1)

{

PORTB = ADCL; //low register output of the adc converter shown at B leds

PORTA = ADCH; //high register output of the adc converter shown at A leds

char str[15];

*sprintf*(str, "%d graden", ADCH);

lcd\_writeLine(str); // show the high register output on the lcd

wait(300);

lcd\_clear();

}

# Week 6

**Opzoekvragen**

**Vraag:** Hoe heet de adresseermode in: ldi yh,$04

**Antwoord:** immediate adressing

**Vraag:** Wat is het verschil tussen: mov r1,r2 en ld r1,Z

**Antwoord:** Mov kopieert register r1 naar register r2 en ld laad de waarde Z in register r1.

**Vraag:** Hoeveel klokpulsen zijn nodig voor: adc r5,r7

**Antwoord:** 1 klokpuls

**Vraag:** Hoeveel verschillen de ‘load’ - instructies zijn er?

**Antwoord:** 15

**Vraag:** Schrijf een instructie/ instructies om een geheugenplaats waarvan het adres in register r12 zit een waarde 88d te geven.

**Antwoord:** LDI R12, $88d

**Coding**

b.

**vraag:** Uit hoeveel bytes bestaat de code van dit programma.

**Antwoord:**

|  |  |  |  |
| --- | --- | --- | --- |
| **instructie** | **#bytes** | **#cycles** | **Total** |
| **.ORG $0200** | **200** | **1** | **200** |
| **LDI YH, $04** | **1** | **1** | **201** |
| **LDI YL, $00** | **1** | **1** | **202** |
| **LDD R8, Y+0** | **1** | **2** | **203** |
| **LDD R9 Y+1** | **1** | **2** | **204** |
| **LDD R6, Y+2** | **1** | **2** | **205** |
| **MOV R4,R8** | **1** | **2** | **206** |
| **ADD R4,R6** | **1** | **1** | **207** |
| **MOV R5,R7** | **1** | **1** | **209** |
| **ADC R5, R9** | **1** | **1** | **20A** |
| **STD Y+4, R4** | **1** | **1** | **20B** |
| **STD Y+5, R5** | **1** | **2** | **20C** |

d.

**Vraag:** Zet een breakpoint aan het einde, laat het programma runnen in de debugger (‘go’), klopt het aantal cycles met b)

**Antwoord:** Ja klopt

**Coding vervolg**

1.

**Vraag:** Schrijf een stukje assembler programma om de geheugenplaatsen in het datageheugen met adres 540H tot en met 548H de waarde 55H te geven.

**Antwoord:**

main:

ldi R17, $55;

STS $540, R17

STS $541, R17

STS $542, R17

STS $543, R17

STS $544, R17

STS $545, R17

STS $546, R17

STS $547, R17

STS $548, R17

no\_end:

2.

**Vraag:** Maak een (assembler) programma dat het 1-complement bepaalt van de geheugenplaatsen in het datageheugen tussen adres 0a60H en 0a70H.

**Antwoord:**

start:

lds R17, $60

com R17

STS $60, R17

end:

rjmp end

3.

**Vraag:** Maak een (assembler) programma dat waarden van de geheugenplaatsen in het datageheugen van adres 660H t/m 670H kopieert naar de geheugenplaatsen 8B0 t/m 8C0H.

**Antwoord:**

start:

lds R16, 10

ldi xh, $16

ldi xl, 00

ldi yh, $8B

ldi yl, 00

loop:

ld R17, X+

ST Y+, R17

dec R16

brne loop

end:

rjmp end

4.

**Vraag:** Bepaal uit hoeveel geheugenplaatsen het programma van 6.5 bestaat en hoeveel klokpulsen dit programma in beslag neemt. Laat een berekening zien in een duidelijke tabel.

**Antwoord:**

5.

**Vraag:** Maak een (assembler) programma dat een 32-bits optelling van twee 32-bits getallen doet (Add32, dit is een uitbreiding van Add16 met telkens 4 bytes).

**Antwoord:**

start:

lds yh, $04

ldi yl, $00

ldd r14, Y+0

ldd r15, Y+1

ldd r12, Y+2

ldd r13, Y+3

ldd r10, Y+4

ldd r11, Y+5

ldd r8, Y+6

ldd r9, Y+7

mov r4, r10

add r4, r15

mov r5, r11

adc r5, r14

mov r6, r8

adc r6, r12

mov r7, r13

adc r7, r9

std Y+4, r4

std Y+5, r5

std Y+6, r6

std Y+7, r7

no\_end:

rjmp no\_end

6.

**Vraag:** Maak een programma RunLight voor de LED’s van PORTA. Laat de LED’s heen en weer lopen (van 0 ◊ 7 en weer van 7 ◊ 0).

**Antwoord:**

start:

ldi R16, 0

ldi R17, 7

out PORTA, R16

loop:

inc R16

out PORTA, R16

dec R17

brne loop

back:

ldi R17, 7

secondloop:

dec R16

out PORTA, R16

dec R17

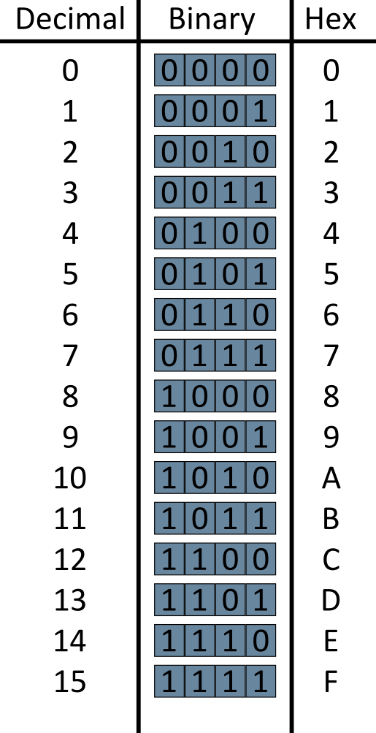
brne secondloop

end:

rjmp start

# Bronvermelding (handige links)

Figuur 1 Binair naar Hexadecimaal naar Decimaal



<http://www.cse.unsw.edu.au/~cs9032/slides/wk7.pdf>