|  |  |
| --- | --- |
| Afbeeldingsresultaat voor atmega128  Rapportage document  Microcontrollers  4-02-2020 | Noah Walsmits & Vincent de Rooij  Periode 2.3 – microcontrollers |

Contents

[Rapportage week 1 2](#_Toc32411465)

[Opdracht A1 2](#_Toc32411466)

[Opdracht A2 3](#_Toc32411467)

[Opdracht B1 4](#_Toc32411468)

[Opdracht B2 8](#_Toc32411469)

[Opdracht B3 9](#_Toc32411470)

[Opdracht B4 9](#_Toc32411471)

[Opdracht B5 10](#_Toc32411472)

[Opdracht B6 11](#_Toc32411473)

[Opdracht B7a 12](#_Toc32411474)

[Rapportage week 2 13](#_Toc32411475)

[Opdracht A1 13](#_Toc32411476)

[Opdracht A2 13](#_Toc32411477)

[Opdracht B1 14](#_Toc32411478)

[Opdracht B2 14](#_Toc32411479)

[Opdracht B3 15](#_Toc32411480)

[Opdracht B4 15](#_Toc32411481)

[Opdracht B5 16](#_Toc32411482)

# Rapportage week 1

## Opdracht A1

**Essentie:**

De materialen verzamelen en klaarmaken. En door de user guide bladeren.

**Oplossing:**

Materiaal verzameld en opgezet.

**Reflectie:**

Ik had wat moeite met de MCU kaart maar voor de rest ging het wel goed met de hulp van de handleiding.

**Bewijs:**

<https://youtu.be/jI10g7l5yBI>

**Volledige code:**

N.V.T.

## Opdracht A2

**Essentie:**

Installeer AVR Studio, maak een nieuw project en gebruik de simulator.

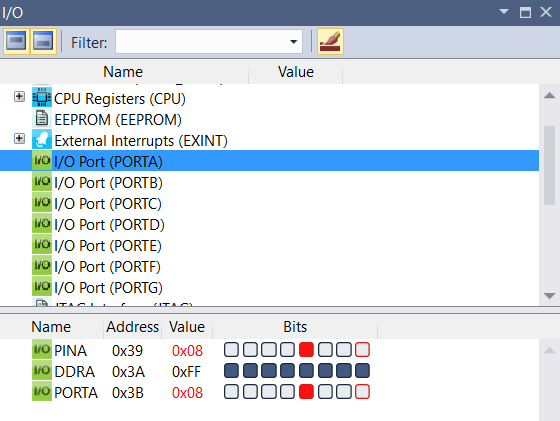
**Oplossing:**

Project gemaakt voor ATmega 128A, omdat die naam op de handleiding stond die kwam met het bord. Ik heb code van blackboard gekopieerd naar main.c. De simulator lijkt de code te draaien. Je kan de aansturing van de I/O bekijken met een knop bovenin. Uiteindelijk heeft het ook gewerkt op het echte bord.

**Reflectie:**

Atmel Studio werkt niet op een andere drive dan C:, dus ik heb het opnieuw moeten installeren. De I/O knop had ik niet zelf kunnen vinden.

**Bewijs:**



**Volledige code:**

Test code gebruikt van BlackBoard.

## Opdracht B1

**Essentie:**

Beantwoord deze vragen:

* Hoe groot is het program memory van de ATmega128?
* Wat is het adres van Data direction register van PORTE (DDRE)?
* Uit hoeveel byte bestaat de instructie ‘IN R3, PORTA’ ?
* Hoeveel RS232 poorten zitten er op het BIGAVR6 development board?
* Op welke pin van de microcontroller zit de ingang voor Analog digitaalconverter, channel 1?
* Hoe groot is het data geheugen van de microcontroller maximaal?
* Hoeveel I/O-registers zijn er op de ATmega128?
* 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?

**Oplossing:**

1: Het flash program memory is 128 kilobytes.

2: Onder de naam DDRE, het adres is 0x22.

3: De instructie bestaat uit één byte. De meeste register instructies worden in één cycle uitgevoerd. Op een 8-bit systeem kan die instructie dan niet meer dan een byte zijn.

4: Er zitten twee RS232 poorten op het bord.

5: De ingang voor analoog digitaal converter kanaal 1 zit op pin 60 of PF1.

6: De maximale grootte van het data geheugen is 64 kilobytes.

7: Er zijn 64 I/O registers.

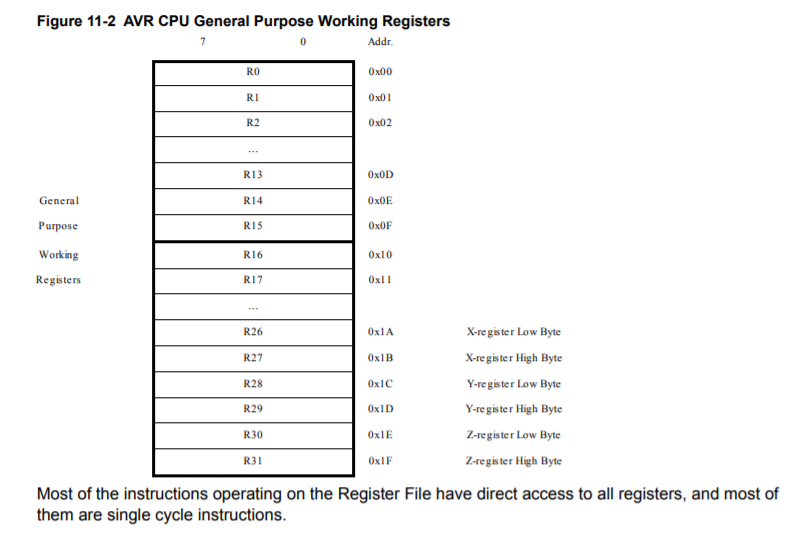
8: Pull-up en pull-down staan standaard uit.

**Reflectie:**

De meeste antwoorden waren niet zo moeilijk om te vinden.

**Bewijs:**

2:

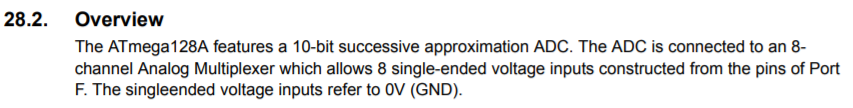


4:

A circuit board

Description automatically generated

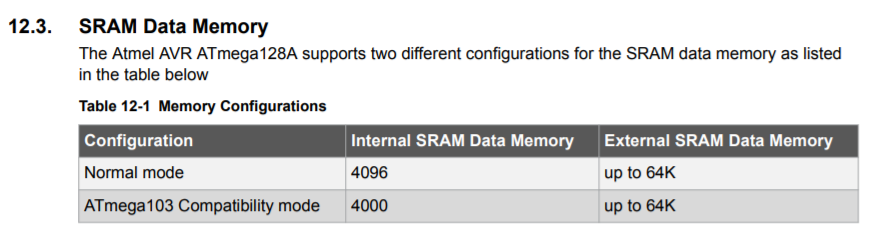
5:



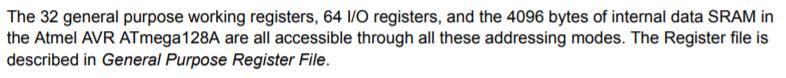
A close up of text on a white background

Description automatically generated

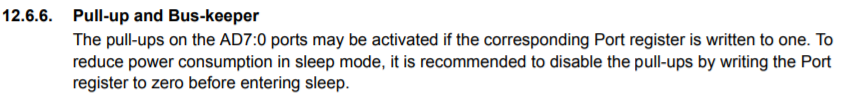
6:



7:



8:



**Bron:**

<https://www.tme.eu/Document/554724703930dfa21e190a7257787ec0/ATmega128A-DTE.pdf>

## Opdracht B2

**Essentie:**

Maak een applicatie die twee beurtelings twee LEDs om de 500ms laat knipperen.

**Oplossing:**

We passen de waarde van PORTD om de 500ms aan. Om een enkele bit op een positie te zetten hebben we een define gebruikt om dat sneller te doen

#define BIT(x) (1<<(x))

**Reflectie:**

Het gebruik van BIT(x) zorgt ervoor dat we makkelijk een pin of port op het bord aan kunnen geven. Port A5 is dan BIT(5) in plaats van 0b00100000.

**Bewijs:**

A circuit board

Description automatically generated

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB2( )

## Opdracht B3

**Essentie:**

Maak een applicatie die een LED laat knipperen als een knop in is gedrukt.

**Oplossing:**

Eerst wordt er gekeken of de knop in is gedrukt door naar de waarde van pin c te kijken. Als dat waar is wordt een LED aan en uit gezet.

if (PINC == 0x80)

**Reflectie:**

De waarde van DDRC is belangrijk als er met de knoppen gewerkt wordt, het veranderen hiervan hadden we eigenlijk in de methode van de opdracht moeten doen.

**Bewijs:**

Volgende versie

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB3( )

## Opdracht B4

**Essentie:**

Implementeer een looplicht met 50ms tussen elke verandering. Geef ook aan hoe er gemeten kan worden of er daadwerkelijk 50ms tussen iedere verandering zit.

**Oplossing:**

Met een for loop wordt iedere LED om de beurt even aangezet met steeds 50ms ertussen. Je kan de timing meten door op het begin van de cycle een timer te beginnen en die te stoppen aan het einde van de cycle. Door de tijd te delen door het aantal veranderingen is te vinden of er tussen iedere verandering 50ms zit.

**Reflectie:**

Om de wait methode goed te laten met de hardware hebben we de wachttijd vermenigvuldigd met 8 in de wait methode zelf.

**Bewijs:**

<https://youtu.be/jI10g7l5yBI?t=25>

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB4( )

## Opdracht B5

**Essentie:**

Implementeer een looplicht met een datastructuur zoals een C array.

**Oplossing:**

We hebben het patroon uit het voorbeeld gebruikt. We probeerden het patroon te gebruiken in een for loop maar we hadden moeite met het bepalen van de lengte van de array. Uiteindelijk volgden we het voorbeeld en hadden we een while loop gebruikt.

**Reflectie:**

Het bepalen van de lengte van een array zal waarschijnlijk binnenkort in de embedded programming lessen voorkomen.

**Bewijs:**

<https://youtu.be/jI10g7l5yBI?t=35>

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB5( )

## Opdracht B6

**Essentie:**

Maak een applicatie die een LED met 1 Hz laat knipperen. Als er op een knop gedrukt wordt moet de LED met sneller knipperen. Als er weer op de knop gedrukt wordt gaat de LED weer terug naar de oude snelheid.

**Oplossing:**

Wij hebben een delay variabele die eerst op 1000 wordt gezet. Terwijl het programma aan staat checkt het of de knop in is gedrukt en wacht het daarna met de tijd die de delay variabele aangeeft. Als de knop in is gedrukt en het delay 1000 is, wordt het delay op 250 gezet. Als de knop in gedrukt is en het delay niet 1000 is, wordt het delay weer op 1000 gezet.

if (PINC == 0b00000001) {

if (delay == 1000) {

delay = 250;

} else {

delay = 1000;

}

}

**Reflectie:**

De toggle werkt niet terwijl het programma wacht. Dit zou op te lossen zijn door de tijd te meten en met een if statement het knipperen uit te voeren.

**Bewijs:**

Volgende versie

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB6( )

## Opdracht B7a

**Essentie:**

Implementeer de applicatie zodat het voldoet aan het state machine diagram.

**Oplossing:**

Wij hebben twee enums gemaakt om de states en events te representeren. Voor iedere enum is ook een variabele om de huide waarde weer te geven.

Eerst gaat het programma door een switch case om de huidige waarde van het event te veranderen als de juiste knop in is gedrukt. Daarna gaat het programma door een tweede switch case om de acties van de huidige state uit te voeren en deze ook mogelijk te veranderen. Aan het einde is een wait methode gebruikt omdat het porgramma anders door meerdere states heen gaat als de knop even ingedrukt is. Dit maakt het wel moeilijker om te bedienen.

case START:

if (btnpressed == D5) {

currentstate = STATE2;

} else if (btnpressed == D6) {

currentstate = STATE1;

}

PORTA = 0b00000001;

break;

**Reflectie:**

Een switch case is misschien niet de meest onderhoudbare oplossing, maar in deze situatie heeft het goed gewerkt.

**Bewijs:**

<https://youtu.be/jI10g7l5yBI?t=47>

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week1/src/main.c> opdrachtB7( )

# Rapportage week 2

## Opdracht A1

**Essentie:**

**Oplossing:**

**The good:** Met interrupts kan je bepaalde gebeurtenissen afhandelen zodra ze gebeuren, zonder dat die gebeurtenissen afgewacht moeten worden door het programma. Interrupts kunnen gebruikt worden als de timing belangrijk is. Als de timing minder belangrijk is kan polling gebruikt worden. De prioriteit van een interrupt is in te stellen via de hardware.

**The bad:** Als je te veel interrupts gebruikt dan kan je ervoor zorgen dat het we werking van je programma stoort omdat je programma niet goed werkt tussen de interrupt calls. Want als je te veel code hebt in je interrupt zorgt het ervoor dat het programma wacht totdat de interrupt is voltooid. Dit zijn dan te lange interrupts. Ook kan je te veel kleine interrupts hebben wat voor hetzelfde effect zorgt.

**The ugly:** Wanneer variabelen in de code tussentijds veranderd worden door interrupts, kan dat voor problemen zorgen stel, je hebt een if statement die controlleert op een GETAL variabele dat niet ‘0’ mag zijn. Momenteel is de globale variabel waarde van GETAL = 8, dus de if statement is true en de code doorloopt de code van de if-statement, op dat moment is er een interrupt, waardoor de code wordt gepauzeerd. De globale variabel van GETAL wordt veranderd naar ‘0’, de code wordt vervolgens vervolgd, maar wel met de ‘0’ waarde die niet in de code mocht voorkomen. Dit kan gebeuren, mits je interrupts niet op de juiste manier afhandelt, dit kan je oplossen door een locale variabele te introduceren die de globale variabele check en overneemt aan het begin van de code. Hierdoor kan de code niet tussentijds veranderd worden.

**Reflectie:**

**Bewijs:**

**Volledige code:**

N.V.T

## Opdracht A2

**Essentie:**

**Oplossing:**

0x20 om het in 4 bit modus te zetten.

0x00 voor de hoge nibble

0x0f voor de lage nibble om het scherm aan te zetten

0x00

0x02 cursor home

2: D0/D7 wordt gebruikt om de bits van het commando door te geven. E wordt gebruikt om de input te verzenden. RS is gebruikt om van command modus naar character modus te veranderen.

3: Om een character te ‘’printen’’ op het LCD moet eerst de modus op RS gezet worden de upper bits verstuurd worden, daarna de lower bits, zo kan je elke willekeurige character maken.

4: // later invullen

**Reflectie:**

**Bewijs:**

**Volledige code:**

## Opdracht B1

**Essentie:**

De werking verklaren en onderzoeken van interrupts d.m.v. demo code te bestuderen.

**Oplossing:**

Er zijn twee interrupts gedefinieerd. Een interrupt zet het licht uit en de andere zet het licht aan. De interrupts gaan in werking als de juiste poort een signaal krijgt. Deze interrupts voeren code uit terwijl de main methode bezig is.

**Reflectie:**

**Bewijs:**

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week2/src/main.c>

## Opdracht B2

**Essentie:**

Het werken met interrupts, d.m.v. het besturen van een lampje, hierdoor is het makkelijker te visualiseren.

**Oplossing:**

Er is een globale variable de de index bijhoud van het looplicht. Als ISR0 afgaat gaat de lamp op die index aan en de index gaat omhoog met 1 of teruggezet naar 0, als die al te hoog is.

**Reflectie:**

**Bewijs:**

**Volledige code:**

## Opdracht B3

**Essentie:**

Onderzoeken hoe een 7-segmenten display werkt samen met knoppen/interrupts

**Oplossing:**

**Reflectie:**

**Bewijs:**

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week2/src/main.c>

## Opdracht B4

**Essentie:**

Een leuke tussen opdracht met het 7-segmenten display.

**Oplossing:**

Door een Array van waardes te doorlopen d.m.v. een for loop kan elk bitje aangestuurd/veranderd worden. Hierdoor kan er een looplicht gemaakt worden.

const unsigned char

PATTERN[] =

{

// dP g f e d c b a

0b00000001,

0b00000010,

0b00000100,

0b00001000,

0b00010000,

0b00100000,

0b00010000,

0b00001000,

0b00000100,

0b00000010,

0b00000001,

void \_7Segment\_Pattern()

{

DDRC = 0b11111111; // PORTC all output

while (1)

{

for (int i = 0 ; i < sizeof(PATTERN) / sizeof(PATTERN[0]); i++ )

{

PORTC = PATTERN[i];

wait(100);

}

}

}

**Reflectie:**

Het werken met het 7-segmenten display ging in het begin niet zonder problemen, eerst hadden we problemen met de juiste aansluiting van de draden, hierdoor kregen we verkeerde waardes. Uiteindelijk is dit opgelost en hebben we de opdrachten met het 7-segmenten display goed kunnen afronden.

**Bewijs:**

--- Plak hier de link naar het filmpje/foto’s

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week2/src/main.c>

## Opdracht B5

**Essentie:**

Het begrijpen van het LCD scherm en de onderliggende aansturing d.m.v. bit/hex waardes

**Oplossing:**

**Reflectie:**

**Bewijs:**

**Volledige code:**

<https://github.com/VincentDeRooij/Microcontrollers-Avans2.3/blob/master/Microcontrollers/week2/src/main.c>