|  |
| --- |
|  |
| Uitwerkingen Practicum |
| Microcontrollers |
|  |
| **Mauro de Lyon & Arthur Brink** |
| **15-02-2015** |

|  |
| --- |
| --- |

Inhoudsopgave

[Track 1 2](#_Toc443298369)

[**B.1** 2](#_Toc443298370)

[**B.2** 3](#_Toc443298371)

[**B.3** 3](#_Toc443298372)

[**B.4** 4](#_Toc443298373)

[**B.5** 5](#_Toc443298374)

[**B.6** 6](#_Toc443298375)

[**B.7a** 7](#_Toc443298376)

Track 1

**B.1** Uit de literatuur

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

128kB

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

Het adres van PORTE is $03($23) en DDRE is $02($22)

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

Op pagina 80 van de instruction set wordt beschreven dat de IN instructie 2 bytes is.

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

Op het bord zijn 2 RS232 poorten aanwezig

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

De ADC zit op pin 60

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

64 Kb

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

Op pagina 20 staat dat het bord 64 ports hebben

* *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?*

De standaard is Pull downs

**B.2**

*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. Ontwikkel de applicatie in de simulator en programmeer daarna het board (gaat veel sneller!)*

Zoals de opdracht al zegt moet je applicatie eerst PORTD instellen op OUTPUT en vervolges om de beurt twee LED’s op aan en uit zetten met een interval van 500 ms. Dit gebeurt met de volgende code (zie het bijbehorende commentaar):

DDRD = 0b11111111; //Set PORTD to OUTPUT

while (1)

{

PORTD = 0x80; //Turn on only PORTD7

wait(500); //Wait for 500ms

PORTD = 0x40; //Turn on only PORTD7

wait(500); //Wait for 500ms

}

Link naar repository:

<https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B2>

**B.3**

*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.*

Zoals de opdracht al noemde moet je applicatie bij deze opdracht een LED laten knipperen als reactie op het indrukken van een drukknop. Dit doen wij door eerst PORTC op INPUT te zetten en PORTD op output te zetten. Vervolgens checkt de applicatie met een interval van 250 ms wat de status is van PINC en of het eerste bit aan staat (wat betekend dat de benodigde drukknop zou zijn ingedrukt). Zo ja, dan zal de LED(s) op PORTD getoggled worden. Mocht het eerste bit niet aanstaan zal PORTD uitgezet worden (zie commentaar). Hieronder de essentiele code van deze opdracht:



Link naar repository:

<https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B3>

**B.4**

*Implementeer een looplicht applicatie op de LED’s van PORTD. Tussen elke verandering van output zit 50ms (milliseconden). Hoe zou je dit kunnen meten?. Om een eenvoudig looplicht te maken kun je gebruik maken van de shift operatoren in C (de >> en de < <). Dit heb je ook al gedaan in periode TI-1.1 op het GUI board.*

Zoals hierboven al is uitgelegd moet de applicatie van deze opdracht een “Looplicht” maken met de LED’s van PORTD. Dit gebeurd door eerst PORTD op OUTPUT te zetten en deze een standaard waarde te geven (om deze te kunnen bitshiften verderop in de code). Vervolgens checkt de applicatie om de 50 ms eerst of de animatie al ten einde is. Zo ja, dan zal de applicatie PORTD resetten naar 0x1 en weer 50 ms wachten. Is dit echter niet het geval dan zal de applicatie de waarde/status van PORTD gaan bitshiften. Dit zorgt ervoor dat het eerstvolgende LED op de strip die is aangesloten op PORTD zal aangaan (de rest van de LED’s is dan uit) waarna er weer 50 ms zal worden gewacht (zie ook het commentaar bij de onderstaande code).

De code die bij deze opdracht hoort:



Link naar repository:

<https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B4>

**B.5**

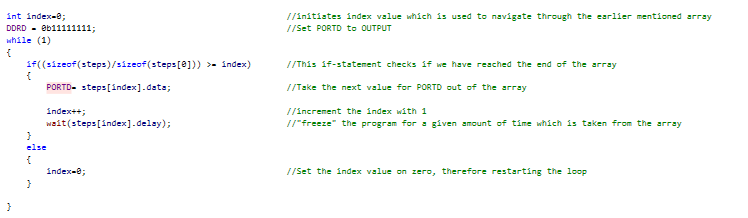
*Een looplicht kun je implementeren met een schuifoperatie. Als het gewenste patroon niet zo eenvoudig is kun je e.a. met een grote if-then-else of switch-case constructie implementeren. Dit levert, in het algemeen, slecht onderhoudbare en starre implementaties op. Beter is om een lichtpatroon te sturen vanuit een datastructuur, bijvoorbeeld een C array.*

*Enig idee hoe dit moet? Zie ook het voorbeeld in de code repository. Implementeer een lichteffect met behulp van deze techniek.*

Zoals in de opdracht is vermeld moet je een animatie aansturen op een strip met LED’s. Zoals in de opdracht is voorgesteld hebben wij de animatie opgeslagen in een array van structs om grote switches en if-then-else statements te voorkomen. Hierdoor kan de animatie in kwestie ook snel worden aangepast. De desbetreffende struct (genaamd ‘step’) bevat een code waarin staat welke LED’s aan of uit moeten en een value die aangeeft wat de pause moet zijn tussen deze en de volgende stap. Voor de uitwerking van deze array en deze struct verwijs ik naar de repository.

Als de applicatie draait zal PORTD eerst op OUTPUT worden ingesteld. Verder word ook een value(index) geinitialiseerd die bij gaat houden bij welke stap in de array we zijn. Hierna checkt de applicatie om de zoveel tijd (aangegeven bij de huidge step) of we aan het einde van de animatie zijn gekomen. Zo nee, dan zal PORTD de waarde van de huidige step krijgen en zal de index waarde met 1 worden verhoogd (in andere woorden de volgende step wordt geselecteerd). Vervolgens zal er gewacht worden voor een bepaalde tijd die door de huidige step wordt gedefinieerd. Mocht de animatie ten einde zijn dan zal de index weer op 0 gezet worden zodat de animatie opnieuw begint (zie het commentaar bij de onderstaande code).

De essentiele code bij deze opdracht:



Link naar repository:

<https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B5>

**B.6**

*Toestanden. 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.*

Zoals in de opdracht al uitgelegd moet deze applicatie de toestand van een LED (de LED van PORTD.7 om precies te zijn) veranderen op basis van het indrukken van een drukknop die is aangesloten op PORTC (PORTC.0 op precies te zijn).

In onze applicatie gebeurt dat door eerst PORTD op OUTPUT te zetten en PORTC op INPUT en word PORTD op 0x0 gezet. Ook worden er eerst twee values geinitialiseerd (de toggle value die zal bijhouden of het LED op 1 of 4 Hz moet knipperen en de “Time to wait” value T2W. Hierna zal de applicatie om de zoveel tijd (aangegeven door T2W) controleren of de drukknop die is aangesloten op PORTC.0 is ingedrukt. Zo ja, dan zal gecontroleerd worden wat de huidige instelling is van de LED en zal deze worden aangepast (zie if-then-else statement van de onderstaande code). Op het einde van de while loop zal de waarde van de LED aangesloten op PORTD (PORTD.7) worden omgedraaid (zie commentaar bij de onderstaande code).

De essentiele code van deze opdracht:



Link naar repository:

<https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B6>

**B.7a**

*Implementeer onderstaande ‘eindige toestandsmachine’ (eng: finite state machine of fsm). Een fsm is de basis van bijna elke embedded applicatie. Koffiemachines, televisies, pacemakers, ABS computers, alarmsystemen enz. zijn voorbeelden van applicaties waar de main-loop vaak bestaat uit een (ingewikkelde) eindige toestandsmachine.*

Bij het maken van de opgave is gekozen voor een switch case met daarin verscheidene if-statements. Er is ook een enumerator aangemaakt die ervoor moet zorgen dat er duidelijke states zijn waar de switch doorheen gaat.



Ondanks deze code werkte het programma niet op de bedoelde manier en ging hij regelmatig naar de end state.

Link naar repository: <https://github.com/MaurodeLyon/Microcontrollers/tree/master/Track1/B7a>

Opmerking: indien er bij deze opgaves bewijs nodig is horen wij het graag, maar voor nu hopen dat deze simpele code voorbeelden voldoen.

(In de toekomst worden er voor bewijzen gezorgd.)