

5 De microcontroller programmeren

Je hebt in een vorig hoofdstuk al gezien hoe je een hex bestand kunt inladen in de microcontroller van de SMU. In dit hoofdstuk leer je om zelf een eenvoudig programma te schrijven.

5.1 De microcontroller

De term is al een paar keer gevallen maar wat is nou eigenlijk een microcontroller? Een microcontroller is een klein computertje op een enkele chip. Het bevat in ieder geval een CPU (processorkern), geheugen en input/output randapparatuur. Je zou het kunnen vergelijken met je laptop, maar alles is wat compacter uitgevoerd. Ter vergelijking:

	Laptop	Microcontroller
RAM geheugen	8 Gb	16 kB
Schijf (installeren programma's)	SSD 256 Gb	Flash 128 kB
Kloksnelheid	2,85 Ghz	24 MHz

Je kunt je afvragen of je wel iets kunt als je deze specificaties ziet: de microcontroller heeft duidelijk minder geheugen en is langzamer dan je laptop. Je zult dus ook geen Windows op deze microcontroller kunt installeren. Waarvoor wordt de microcontroller dan wel gebruikt? Over het algemeen gebruiken we de microcontroller om systemen uit te lezen en aan te sturen, en daar is deze heel geschikt voor. Voorbeelden van systemen waarin de microcontroller wordt gebruikt zijn: een wasmachine, je telefoon, de auto, een robotgrasmaaier, kortom bijna alle moderne apparaten. Voor het voorbeeld van de wasmachine krijgt de microcontroller signalen binnen (input) van een aantal sensoren: temperatuur, snelheid van rotatie, deursensor, etc. Vervolgens verwerkt de microcontroller die signalen en gaat er andere onderdelen mee aansturen (output): de motor bijvoorbeeld, of het verwarmingselement. Wat er tijdens het verwerken gebeurt wordt bepaald door de microcontroller te programmeren. In de programma's die je voor de microcontroller schrijft kan je precies aangeven op basis van welke input welke output aangestuurd moet gaan worden. Het uitlezen van ingangen en het schrijven naar uitgangen noemen we ook wel I/O of GPIO (General Purpose I/O) aansturing.

THUISOPDRACHT 1:

Bekijk nog eens het schema van de SMU in hoofdstuk 1.

- Welke onderdelen zijn een input voor de microcontroller (geven informatie aan de microcontroller)?
- Welke onderdelen worden aangestuurd door de microcontroller en zijn dus een output?

Bekijk nog eens de datasheet van de microcontroller (hoofdstuk 2: Pinout)

- Welke pinnen van de microcontroller kan je gebruiken om in- en outputs op aan te sluiten?
- Wat betekent 'digital function only' (datasheet 2.1)?
- Wat betekent 'analog function'?

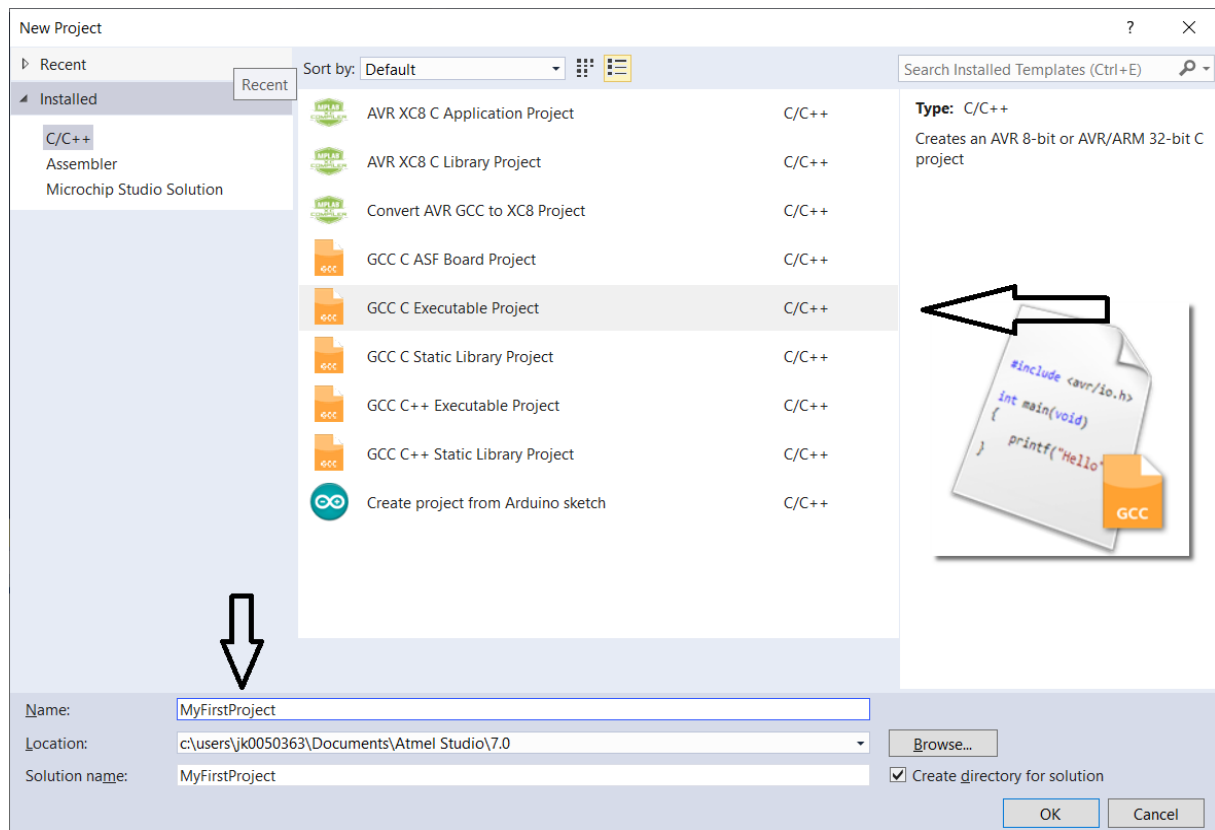
5.2 Een basis C Programma

We moeten een aantal stappen doorlopen om een programma in de microcontroller te krijgen:

1. Intypen broncode ("het programma")
2. Compileren
3. Inladen in de microcontroller

Voor het intypen van het programma gaan we Microchip Studio gebruiken.

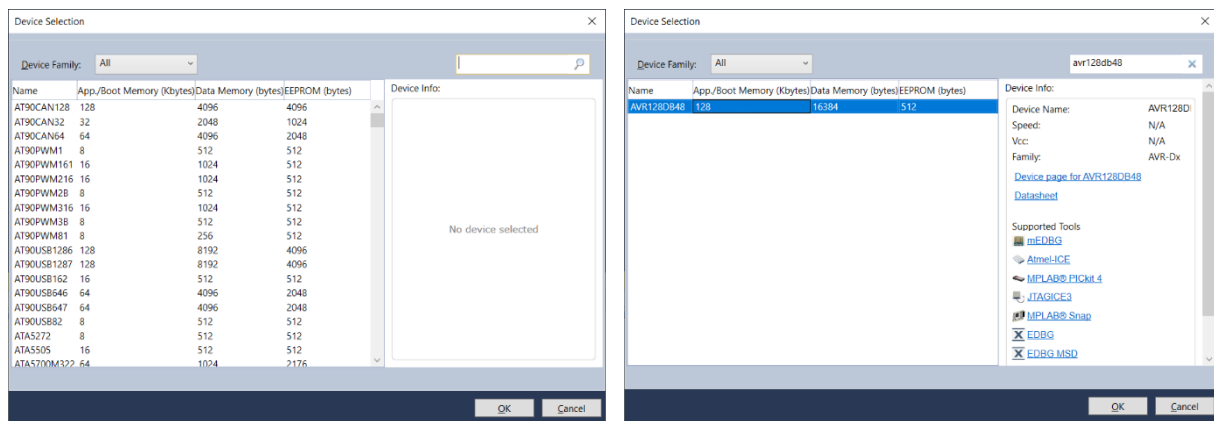
We gaan het programma schrijven in de taal C. Om een programma te maken gebruiken we de project wizard in Microchip Studio. Kies File -> New -> Project.



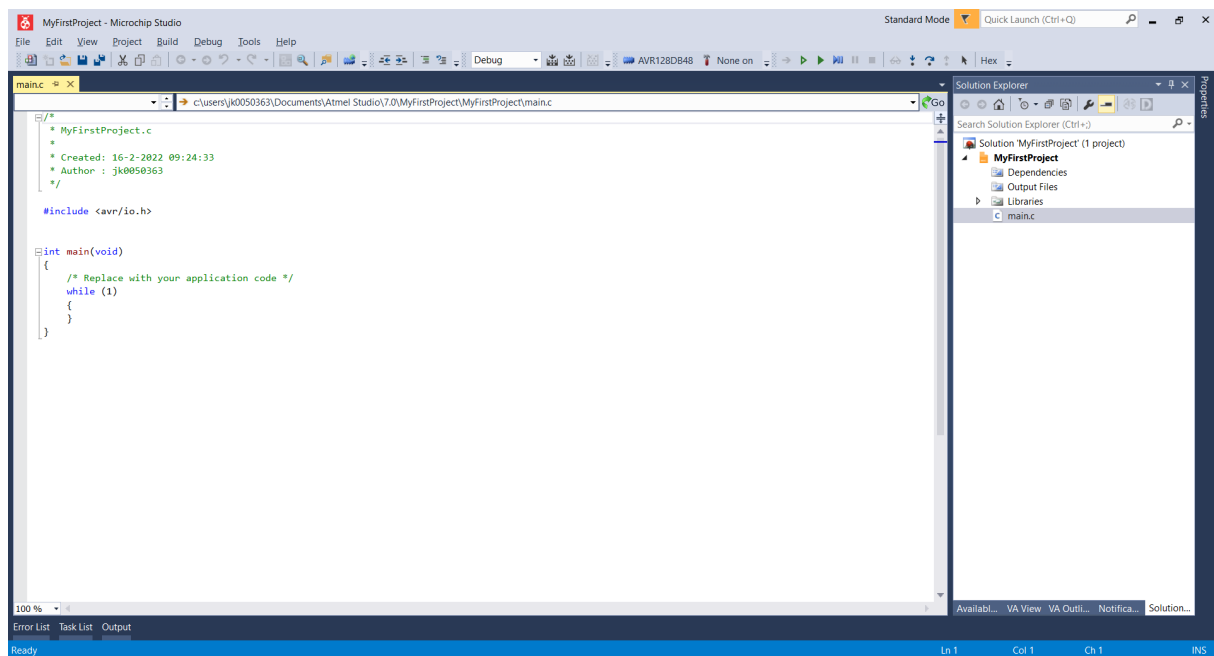
Kies voor een GCC C Executable Project. Geef je project een geschikte naam. Kijk ook goed naar de Location: daar worden de bestanden van je project opgeslagen. Handig als je die nog eens terug moet vinden om te versturen of om een backup te maken! Kies vervolgens OK.

In het volgende scherm moet je aangeven voor welk type processor je een programma wilt maken. Dit typenummer staat vaak aangegeven op de bovenkant van de chip. Kijk goed wat het typenummer is en vul het in.

De reden dat je het type chip aan moet geven is dat elk type chip een verschillend aantal pinnen heeft, deze anders noemt, en ze voor andere functionaliteiten geschikt zijn. Zodra je hebt aangegeven welk type chip je gebruikt, wordt er een 'header'-bestand (.h) aan je code toegevoegd waarin al dit soort eigenschappen en functies zijn beschreven zodat je vervolgens makkelijker kunt programmeren.



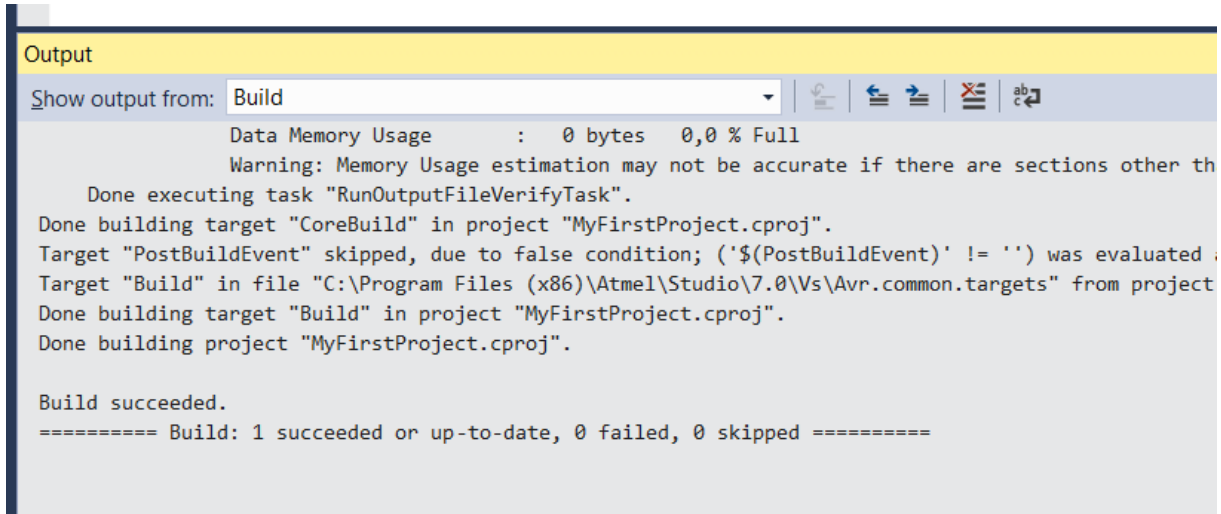
Nu het type bekend is wordt het eerste programma gemaakt:



Zonder teveel details te geven is het wel goed om te weten wat er nu gemaakt is.

- Er is een project aangemaakt met daarin een *source* bestand (main.c). In dit source bestand staat de code die je programmeert.
- In het source bestand is al wat code gemaakt. Dit is de standaard structuur van een programma voor een microcontroller:
 - o Een include om bij de *registers* in de microcontroller te kunnen komen. De registers zijn locaties in het geheugen van de microcontroller. Er zijn bijvoorbeeld registers waarin aangegeven staat of een pin als in- of een output wordt gebruikt.
 - o Een main functie: hier begint je programma met uitvoeren zodra de microcontroller wordt aangezet.
 - o Een while(1) loop, waarbinnen code staat die steeds opnieuw uitgevoerd moet worden.

Dit c programma kan niet meteen in de controller worden ingeladen. Daarvoor moet het eerst worden *gecompileerd* waarbij de c-code wordt omgezet naar een hex bestand, welke de microcontroller kan uitvoeren. Om te compileren kies je voor Build -> Build solution. In het output window kun je zien wat het resultaat is van de compilatie:



The screenshot shows the 'Output' window in AVR Studio. The 'Show output from:' dropdown is set to 'Build'. The output text is as follows:

```
Data Memory Usage      : 0 bytes  0,0 % Full
Warning: Memory Usage estimation may not be accurate if there are sections other th
Done executing task "RunOutputFileVerifyTask".
Done building target "CoreBuild" in project "MyFirstProject.cproj".
Target "PostBuildEvent" skipped, due to false condition; ('$(PostBuildEvent)' != '') was evaluated :
Target "Build" in file "C:\Program Files (x86)\Atmel\Studio\7.0\Vs\Avr.common.targets" from project
Done building target "Build" in project "MyFirstProject.cproj".
Done building project "MyFirstProject.cproj".

Build succeeded.
===== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped =====
```

Als het compileren is gelukt is er een hex file gemaakt die je in de microcontroller kunt laden. Kies daarvoor Debug -> Start without debugging. De hex file wordt nu in de microcontroller geladen en meteen uitgevoerd.

5.3 I/O settings

Het programma dat door de wizard is gemaakt bevat weinig code, het doet zelfs nog helemaal niets, maar geeft alleen de standaard structuur van een programma. Om inputs te lezen en outputs te sturen hebben we meer code nodig. **Hoofdstuk 18 van de datasheet** beschrijft wat we moeten toevoegen om een input te kunnen lezen en een output te kunnen schrijven. We beginnen met het instellen van de input en output pinnen. De code die hiervoor nodig is moet binnen de *main*, maar buiten de *while* komen te staan. We willen dit immers maar één keer instellen als de microcontroller wordt aangezet. Wat er precies getypt moet worden volgt in de volgende paragrafen.

```
* MyFirstProject.c
*
* Created: 16-2-2022 09:24:33
* Author : jk0050363
*/

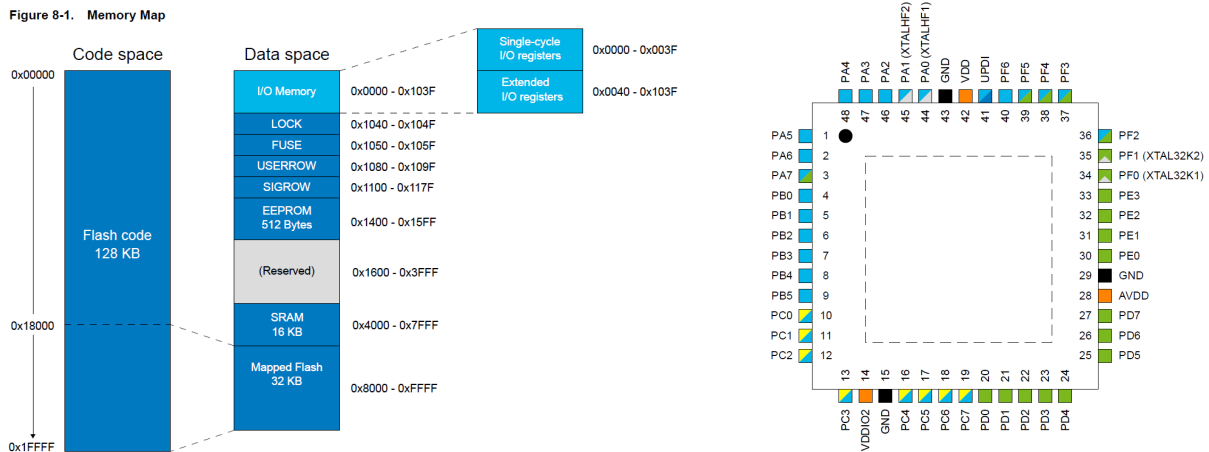
#include <avr/io.h>

int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Hier moeten de instellingen voor in en output pinnen komen te staan.

5.3.1 Registers

Om inputs te kunnen lezen en outputs te kunnen schrijven moeten we ergens in het geheugen van de microcontroller instellen dat een pin als in- of output gebruikt gaat worden. Ook zijn er een aantal optionele functionaliteiten die ingesteld kunnen worden. De plek waar dit allemaal gebeurt is het I/O memory. De gegevens in dit deel van het geheugen zijn georganiseerd in “registers” van 8 bits (dus een stukje geheugen van 8 enen en/of nullen dat bij elkaar hoort). Zo’n groep van 8 bits wordt overigens ook wel een byte genoemd.



Alle I/O pinnen worden gedefinieerd in de PORTx registers. Er is een PORTA, PORTB, PORTC, etc. (vandaar PORTx). Welke pin in welk register zit kan je afleiden uit de pinnaam (zie pinout in figuur). Pin 1 bijvoorbeeld, heet PA5. Dit betekent dat de pin in het PORTA register zit en met bit nummer 5 uit dat register correspondeert. Merk op dat de nummering begint bij 0, wat overeenkomt met het meest rechter bit in het register.

5.3.2 Hexadecimale getallen

Het hexadecimale talstelsel is bedacht om overzichtelijk in bytes te kunnen rekenen. Een byte is 8 bits breed en kun je eenvoudig met twee hexadecimale cijfers aanduiden. Dit komt van pas als we met de registers gaan programmeren.

Hexadecimaal betekent letterlijk zestientallig. Het is een talstelsel waarbij met zestien cijfers wordt gewerkt. De cijfers 0 t/m 9 worden daarom uitgebreid met ‘A’ (= 10) t/m ‘F’ (= 15). Net als bij andere talstelsels komt elke positie overeen met een macht van het grondtal (16 in dit geval). Om aan te geven dat het een hexadecimaal getal is staat er vaak ‘0x’ voor het getal. Hexadecimaal wordt vaak afgekort tot hex.

De omrekening van hexadecimaal naar decimaal verloopt op dezelfde manier als bij de binaire getallen, alleen is het grondtal nu 16 in plaats van 2:

Hexadecimaal getal	Macht	Decimaal getal
0x01	16^0	1
0x02	2×16^0	2
0x10	16^1	16
0xC0	12×16^1	192
0xF8	$15 \times 16^1 + 8 \times 16^0$	248

Binaire getallen zijn gemakkelijk om te zetten in hexadecimale getallen door ze op te delen in groepjes van 4 bits (nibbles). Vervolgens zet je elk groepje van 4 bits om naar hexadecimaal:

Het binaire getal 10001101 wordt hexadecimaal:

- Opsplitsen in groepjes van 4 bits: 1000 1101
- Per groepje omzetten naar decimaal: 8 en 13
- Omzetten naar hexadecimaal: 0x8D

Bestaat je binaire getal niet precies uit een veelvoud van 4 bits? Voeg dan nullen toe aan het begin van het getal tot dit wel het geval is:

Het binaire getal 100001 wordt hexadecimaal:

- Opsplitsen in groepjes van 4 bits: 0010 0001
- Per groepje omzetten naar decimaal: 2 en 1
- Omzetten naar hexadecimaal: 0x21

Deze hexadecimale getallen ga je zometeen gebruiken bij het instellen van de in- en output pinnen.

THUISOPDRACHT 2:

Zet om van hexadecimaal naar decimaal:

- a) 0xA5
- b) 0x3F
- c) 0x13C

Zet om van binair naar hexadecimaal:

- d) 11001010
- e) 101110
- f) 11011

Zet om van hexadecimaal naar binair:

- g) 0x40
- h) 0x7B
- i) 0xE28

5.3.3 Een pin als input gebruiken

Een pin van de microcontroller is standaard geconfigureerd als input. Om een pin als input te gebruiken hoeven we dus niets meer in te stellen. Waar we wel over na moeten denken is of een pin nog een pull-up weerstand nodig heeft of niet. De microcontroller beschikt namelijk over een interne pull-up weerstand die aan of uitgezet kan worden. Is er in de elektronische schakeling al een pull-up of pull-down weerstand aanwezig, dan hoeven we hem softwarematig niet aan te zetten. Is in de elektronische schakeling een schakelaar direct op een pin aangesloten zonder pull-up of pull-down, dan moet je wel softwarematig de pull-up instellen (om storingen te voorkomen). Hoe je dat doet staat in H18.3: met het bit PULLUPEN (pull-up enable) in het PORTx.PINnCTRL register.

In praktijk werkt dat als volgt: om voor pin PA7 de pull-up aan te zetten typ je in de code:

```
PORTA.PIN7CTRL = PORTA.PIN7CTRL | PORT_PULLUPEN_bm;
```

THUISOPDRACHT 3:

Stel, we willen een schakelaar (SW1) van het SMU bord als input gebruiken voor de microcontroller.

- Aan welke pin van de microcontroller is SW1 aangesloten?
- Welke poort en bit uit het register hoort daarbij?
- Heeft SW1 een externe pull-up weerstand (dus fysiek op de printplaat)?
- Wat moet er dus worden ingesteld in de software om de pin van SW1 als input te kunnen gebruiken?
- Maak een waarheidstabel voor deze schakelaar. Wanneer geeft hij een '0' of een '1' aan de microcontroller?

5.3.4 Een pin als output gebruiken

Aangezien een pin standaard als input gedefinieerd is, moeten we zeker code gaan toevoegen om deze als output te gaan gebruiken. In H18.3.2.1 van de datasheet staat dat, als je een pin wilt configureren als output, je het bijbehorende bit in het PORTx.DIR register op '1' moet zetten. Je gebruikt daarvoor een bitoperatie: de OR (in code getypt als: |). Deze werkt hetzelfde als de OR poort die je al hebt gezien in het vorige hoofdstuk.

Om pin PD5 als output te gebruiken typ je in:

```
PORTD.DIR = PORTD.DIR | PIN5_bm;
```

Of:

```
PORTD.DIR = PORTD.DIR | 0x20;
```

Hier staat: de waarde in het DIR register wordt de waarde in het DIR register waarbij het bit PIN5_bm hoog wordt gemaakt (de hexadecimale waarde van PIN5_bm is 0x20).

Wanneer je meerdere pinnen van één poort hoog wilt maken kan je dat doen door een OR te doen met het bijbehorende hexadecimale getal. Bijvoorbeeld: we willen pin 2 en 5 van PORTA hoog maken. De tweede en vijfde bit uit de byte moet dus 1 worden: "00100100". Zetten we dit om naar hexadecimaal dan vinden we: 0x24. We kunnen de twee pinnen tegelijk hoog maken door:

```
PORTA.DIR = PORTA.DIR | 0x24;
```

THUISOPDRACHT 4:

We willen LED2 van het SMU bord als output gebruiken voor de microcontroller.

- Aan welke pin van de microcontroller is LED2 aangesloten?
- Welke poort en bit uit het register hoort daarbij?
- Wat moet er dus worden ingesteld in de software om de pin van LED2 als output te kunnen gebruiken?
- Waar moet dit in de code komen te staan? (geef een snapshot van je totale code)
- Bij welk signaal op de pin (hoog of laag) gaat de LED aan?

5.4 Functionaliteit van de code

Nu we de schakelaar hebben ingesteld als input en de LED als output, kunnen we code gaan toevoegen die het gedrag van het systeem beschrijft. In dit geval dus: bij welke standen van de schakelaar de LED aan of uit is. Dit deel van de code komt binnen de *while* loop te staan:

```
* MyFirstProject.c
*
* Created: 16-2-2022 09:24:33
* Author : jk0050363
*/
```

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

```
int main(void)
{
    /* Replace with your application code */
    while (1)
    {
    }
}
```

Hier moeten de instellingen voor in- en outputpinnen komen te staan.

Hier moet komen te staan wat de in- en outputpinnen moeten doen.

5.4.1 Pinnen uitlezen en aansturen

Uitlezen

Om te weten wanneer de LED aan moet, moeten we ten eerste weten wat de stand van de schakelaar is. Daarvoor lezen we de inputpin van de schakelaar uit. Dit doe je door te typen (zie H18.2 datasheet):

```
PORTA.IN & PIN7_bm
```

We kijken dan eigenlijk of er een '1' of een '0' op pin PA7 staat.

Aansturen

Om een pin aan te sturen (bijvoorbeeld de pin waaraan de LED is aangesloten) moet je het bijbehorende bit hoog of laag maken in PORTx.OUT. Om dat hoog en laag maken te vereenvoudigen bestaan er twee registers: PORTx.OUTSET en PORTx.OUTCLR. Als je in deze registers een bit schrijft wordt de output respectievelijk geset (op '1' gezet) of gecleared (op '0' gezet).

Dus, zodra we hebben ingesteld dat PA7 een outputpin is kunnen we hem hoog maken (LED gaat dan aan) door te typen:

```
PORTA.OUTSET = PIN7_bm
```

En we kunnen pin PA7 laag maken (LED gaat dan uit) door te typen:

```
PORTA.OUTCLR = PIN7_bm
```

5.4.2 Het if-statement

Een voor de hand liggende functionaliteit voor de schakelaar en de LED zou zijn: als ik op de knop druk moet de LED aan zijn, als ik niet op de knop druk is de LED uit. Aan de beschrijving hoor je het al: dit is een "als... , dan ..." constructie. In programmeertaal heet dit een if-statement. In C ziet dat statement er als volgt uit:

<pre>If (test expression) { //code } Else { //code }</pre>	<pre>If (toestand schakelaar) { LED uit } Else { LED aan }</pre>
--	--

Het werkt als volgt: als de test expression tussen de haakjes WAAR is (of '1'), dan wordt de code tussen de accolades onder *if* uitgevoerd. Is de test expression NIET WAAR (of '0'), dan wordt het stuk code onder *else* uitgevoerd.

Oftewel, voor onze schakelaar met LED: als de schakelaar wordt ingedrukt geeft deze een '0' aan de microcontroller (volgt uit het schema). De test expressie tussen de haakjes is dan NIET WAAR, dus in het stuk onder *else* moet komen te staan dat de LED aan gaat. Als de schakelaar niet wordt ingedrukt geeft deze een '1' aan de microcontroller. De LED moet dan uitgaan.

In C-code wordt dit:

```
if (PORTF.IN & PIN0_bm)          // hier is de pin hoog, schakelaar niet ingedrukt
{
    PORTA.OUTCLR = PIN7_bm;    // LED uit
}
else
{
    PORTA.OUTSET = PIN7_bm;    // LED aan
}
```

De totale code, inclusief het instellen van de pinnen en de functionaliteit wordt nu:

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

int main()
{
    PORTA.DIR = PORTA.DIR | PIN7_bm;

    while(1)
    {
        if (PORTF.IN & PIN0_bm)
        {
            PORTA.OUTCLR = PIN7_bm;
        }
        else
        {
            PORTA.OUTSET = PIN7_bm;
        }
    }
}
```

LABOPDRACHT 1:

Maak een programma waarbij de LED de “inverse van” de schakelaar aangeeft: schakelaar ingedrukt = LED uit, schakelaar niet ingedrukt = LED aan. Gebruik hiervoor schakelaar 2 en LED2 op de SMU.

- Maak in Microchip Studio een nieuw project aan.
- Zoek in het schema van de SMU op met welke pin schakelaar 2 verbonden is. Configureer deze als ingang in de setup functie.
- Zoek in het schema van de SMU op met welke pin de LED verbonden is. Configureer deze als uitgang in de setup functie.
- Zorg dat je de waarde van de schakelaar uitleest in de *while* loop en deze schrijft naar de LED. Als je de schakelaar indrukt moet de LED dus uit gaan.
- Test je programma.

5.4.3 Logische poorten in code

We hebben in hoofdstuk 4 al kennis gemaakt met de verschillende logische poorten. Deze logische basisfuncties kun je ook gebruiken in je programma. Je gebruikt hiervoor de *logische operatoren*:

De AND functie schrijf je als: `&&`

De OR functie schrijf je als: `||`

De NOT functie schrijf je als: `!`

Voorbeeld:

```
led = a && b && !c;
```

Let op: de prioriteit van de `&&` en de `||` operatie is gelijk in je programma. Het kan zijn dat je dus haakjes moet toepassen terwijl die niet in de schakelalgebra staan.

LABOPDRACHT 2:

Maak een programma waarbij de LED een combinatorische aansturing van schakelaars is volgens de volgende waarheidstabel:

A (SW1)	B (SW2)	C (SW3)	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Schrijf de schakelfunctie op aan de hand van de waarheidstabel.
- Vereenvoudig de functie middels schakelalgebra.
- Programmeer de vereenvoudigde schakelfunctie (let op: de switches zijn geïnverteerd).
- Test je programma.

LABOPDRACHT 3:

Bij deze opdracht ga je de LED toggelen (dus steeds inverteren) en meet je de knipperfrequentie.

- Inverteer de LED elke keer in de loop functie.
- Meet met de oscilloscoop de frequentie waarmee de LED knippert.
- Wat zegt deze frequentie over de snelheid van de microcontroller?
- Klopt dat met de specificaties uit de datasheet?

Naast de logische operatoren is het handig om nog wat meer programmeerfuncties tot onze beschikking te hebben. Deze worden hieronder toegelicht.

5.4.4 De delay

De loop functie wordt steeds opnieuw aangeroepen. Soms wil je dat de microcontroller even stopt. Daarvoor kun je een wachtfunctie aanroepen. Deze heet binnen Microchip Studio `_delay_ms()`. Je geeft dan de tijd in milliseconden mee aan de functie, bijvoorbeeld:

```
_delay_ms(500);
```

Let op: als de microcontroller in de wachtfunctie zit zie je dus ook geen knoppen die ingedrukt worden! Wees daarom wel voorzichtig met deze functie. Om deze functie te kunnen gebruiken heb je een extra include file nodig genaamd "delay.h". Ook moet je vertellen hoe snel de klok van jouw processor is zodat de delay functie de juiste wachttijd kan bepalen. Standaard draait de door onze processor op 4 MHz. Dat resulteert in de volgende code:

```
#define F_CPU 4000000UL

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

5.4.5 Variabelen

Om een getal op te slaan in de microcontroller gaan we gebruik maken van variabelen. Een variabele bestaat uit twee delen: het datatype en de naam. Het datatype is de soort data die je wilt gaan opslaan. Voor gehele getallen gebruiken we het datatype `int`, en voor kommagetallen het datatype `float`. Later in de opleiding komen daar nog andere datatypes bij. De naam van de variabele mag je zelf kiezen, maar zorg wel voor een logische naam. Je mag geen spaties in de naam van de variabele gebruiken. Een voorbeeld kan zijn:

```
int teller;
```

Je kunt een variabele aanmaken op verschillende plekken in je programma:

- Binnen een functie. Dat noemen we een *lokale* variabele. De variabele is dan alleen binnen de functie te benaderen.
- Buiten een functie. Dat noemen we een *globale* variabele. De variabele is dan alleen vanuit alle plekken in je programma te benaderen. Dit lijkt handig maar kan de leesbaarheid van je code wel lastig maken. Mijn voorkeur zou dan ook hebben om zoveel mogelijk lokale variabelen te gebruiken.

Als je een variabele wilt aanpassen geef je deze een nieuwe waarde, bijvoorbeeld:

```
teller = teller + 1;
```

5.5 Het 7-segmentsdisplay aansturen met de microcontroller

Vorige week heb je de werking van het 7-segmentsdisplay uitgezocht met behulp van een schakeling op het breadboard. Met de kennis die je nu hebt over programmeren, kan je het 7-segmentsdisplay ook via de microcontroller gaan aansturen.

LABOPDRACHT 4:

Bij deze opdracht ga je het 7-segmentsdisplay aansturen met de microcontroller.

- a) Maak een lijstje van de pinnen van de decoder die door de microcontroller worden aangestuurd en de bijbehorende pinnen van de microcontroller. Let op: alle ingangspinnen van de decoder moeten worden aangestuurd!

Pin decoder	Verbonden pin microcontroller

- b) Gebruik hexadecimale getallen om in 2 regels alle benodigde outputpinnen van de microcontroller in te stellen als output.
- c) Schrijf een programma dat het cijfer 3 op het 7-segmentsdisplay doet verschijnen.
- d) Pas het programma zo aan dat het cijfer op het display aangeeft hoeveel knoppen er tegelijk worden ingedrukt.

VOORTGANGSOPDRACHT 4:

Zodra je er klaar voor bent kan je je docent vragen om de vierde voortgangsofdracht die afgetekend moet worden om mee te mogen doen aan de eindtoets.

Voor deze opdracht moet je laten zien dat je een LED kan aansturen met switches.