WEEK 3

ONDERWERPEN

- de structuur van C-programma's
- ATMEL studio en AVR libc
- typen, constanten en operatoren
- AVR register access in C
- control statements
- functies & stackframe
- visibility scope
- arrays & strings
- struct & enum

GROEN GEMARKEERD = AFTEKEN OPDRACHT

OPMERKING

Meestal wanneer je begint met C programmeren dan werk je op een besturingssysteem (Windos, OSX of linux). De standaard library geeft je toegang tot toetsenbord, scherm en het filesysteem. Op het Arduino bordje is geen OS, we werken alleen met AVR GCC, en gebruiken C als een soort high-level assembly). Het gevolg hiervan is dat we wel wat beperkt zijn in het type opgaven, printf() en scan() zijn bijvoorbeeld niet standaard beschikbaar (nauwkeuriger : stdin, stdout en stderr moet de programmeur zelf definiëren).

OPDRACHT 1 : OPERATOREN

Leg uit wat in C het verschil is tussen de operatoren :

a) & en &&
b) | en ||

OPDRACHT 2: HEADERFILES EN SCOPE

Je kunt C programma's in meerdere C en H bestanden opknippen.

a) Waarom zou je een (groter) programma in een H en C bestand opknippen ? (Denk eens aan #include <avr/io.h>, #include <avr/interrupt.h>, etc.)

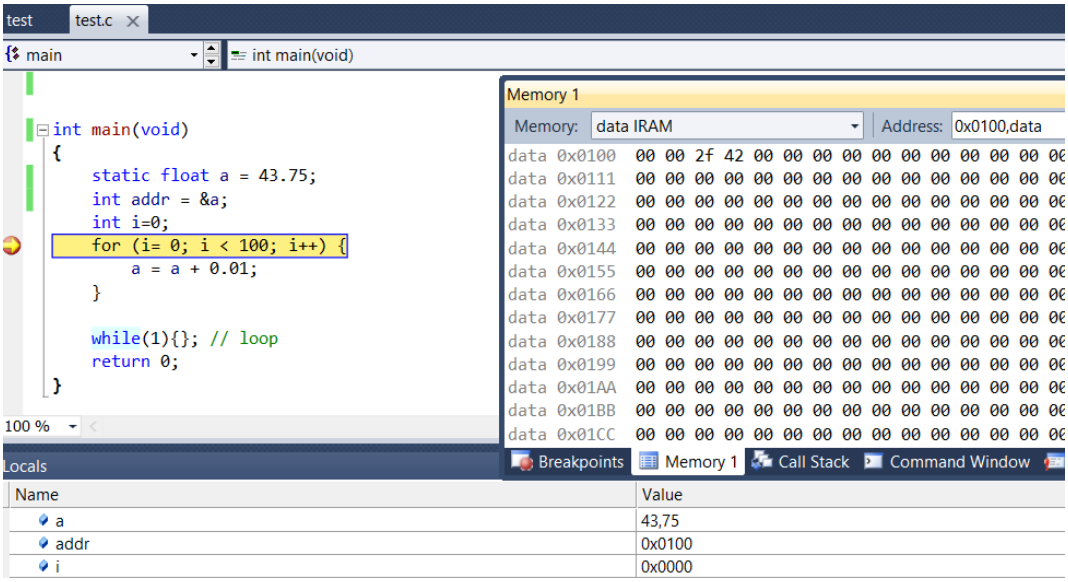
b) Wat is het fundamentele verschil tussen een **#include** in C en een **import** in Java of **using** in C# ?

c) Wat is een functieprototype ?

D) Een functieprototype wordt ook wel eens in een C bestand gebruikt. Waarom ? Hint: hoe leest de compiler het C bestand?

OPDRACHT 3 : EEN C PROGRAMMA (SIMULATOR)

Type het onderstaande C-programma over :



a) Wat is de storage size van een float in avr-gcc ?

b) Zorg ervoor dat de compiler niet optimaliseert. Dit kun je door : Project properties > Toolchain > Compiler > **`Optimization level = None`** te zetten. Leg uit waarom we dit moeten doen.

c) Laad het programma in de simulator, 'bouw' het programma en plaats een breakpoint op de for-statement. Start de debugger en druk 1x op continue. Met Debug>Windows>Locals kun je de (lokale) variabelen bekijken. Wat is het adres van a ?

d) Hoeveel bytes staan er op één enkel adres in het datasegment ?

e) Met Debug>Windows>Memory kun je het datageheugen bekijken. Wat de hexadecimale representatie van de floating point waarde 43,75 ? (Zie de sheets CS week 1.) Klopt dit met wat je ziet ?

f) Is de AVR-architectuur big of little endian ?

g) Plaats een breakpoint op de while-lus. Wat is na afloop van de for-lus de waarde van a ? Klopt dit met wat je verwacht ?

OPDRACHT 4 : REGISTERS EN BITWISE OPERERATOREN (SIMULATOR)

Maak de onderstaande functies en laat zien hoe je ze aan kunt roepen van uit de functie main(). Test de functies in de simulator. Je kan er in de debugger door heen stappen, maar dan moet je wel de compiler optimalisatie uitzetten. Met Debug>Windows>Locals kun je variabelen bekijken. (Met de rechter muisknop kun je de weergave op hexadecimaal zetten).

a) Een functie die een bit set (1 maakt) in een byte met het volgend prototype :

```
uint8_t set_bit(uint8_t byte, uint8_t bit);
```

Voorbeeld : set_bit(0xF0, 2) geeft als resultaat 0xF4.

b) Een functie die alle bits inverteert in een byte met het volgend prototype :

```
uint8_t invert(uint8_t byte);
```

Voorbeeld : invert(0xAA) geeft als resultaat 0x55.

c) Een functie het aantal bits = 1 van een gegeven 16-bit integer terug geeft met het volgende prototype :

```
uint8_t count_ones(uint16_t word);
```

Voorbeeld : count_ones(0xAAAA) geeft als resultaat 8.

OPDRACHT 5: SCOPE

Gegeven twee versies (a) en (b) van een C programma. Wat is aan het einde van het programma (dus voor de return 0) de waarde van z ?

```
// versie a
int z;

void f(int x)
{
	z+=x;
}
int main()
{
	z = 5;
	f(z);
	return 0; ////10
}
```

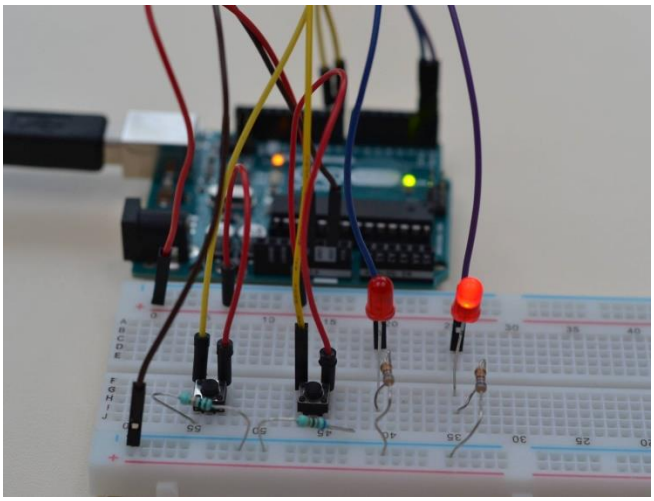
```
// versie b
int z;

void f(int x)
{
	int z = 0;
	z+=x;
}
int main()
{
	z = 5;
	f(z);
	return 0; ///5
}
```

OPDRACHT 6 : KNOPJES LEZEN MET GEHEUGEN

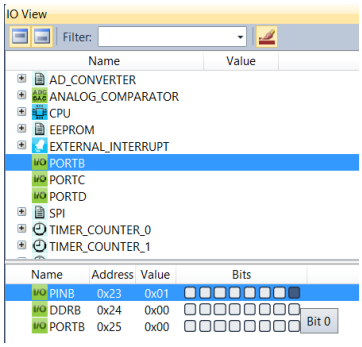
Deze opdracht is vergelijkbaar met een opdracht van de vorige week, alleen nu moet het programma in C worden geschreven.

Verbind op het breadboard eerst een blauwe min-lijn met de ground en een rode plus-lijn met de 5V. Verbind daarna twee LED's met port D (PD0 en PD1), elk via een weerstand naar de ground. Plaats de LED's in de goede richting. Verbind dan twee schakelaars met port B (PB0 en PB1) elk via een weerstand met de ground, en via een schakelaar met de 5V. (Zie ook de sheets van week 2 hierover).



Maak een programma dat een LED laat branden wanneer een knopje wordt ingedrukt. Wanneer knop 0 wordt ingedrukt, moet LED 0 gaan branden. Wanneer knop 1 wordt ingedrukt, moet LED 1 gaan branden. Zorg er voor dat de LED blijft branden ook wanneer je de knop loslaat.

TIp : Je kan je programma ook debuggen met de simulator. In de simulator kan je ook de waarde van register PINB aanpassen via Debug > WIndows > IO view.



OPDRACHT 7 : PINCODE

Hieronder staat een programma pincode.c dat nog niet helemaal af is. De functies init_ports() is af, maar aan het hoofdprogramma main() ontbreekt nog wel iets. Op de ontbrekende ... moet nog jou code komen !

Verbind vier switches met poort B0..B3 en vier LED's met poort D0..D3. (Gebruik bij de switches wel pull-down weerstanden).

Het programma moet de pincode die de gebruiker intoetst controleren. Aangezien er 4 switches zijn, zijn er 16 mogelijke pincodes. Voert de gebruiker de juiste pincode in, dan wordt dit aangegeven door de 4 LEDs.

Misschien handig om het programma eerst te testen in de simulator. Via de I/O view kun je poort B waarden geven. Vergeet dan niet (tijdelijk) de compiler optimalisatie uit te zetten. (Wil je programma flashen, zet dan code optimalisatie weer aan, anders werkt de delay functie niet meer optimaal).

```
#include <avr/io.h> // ATmega328P specific I/O
#define F_CPU 16E6  // used in _delay_ms, 16 MHz
#include <util/delay.h>

#define OFF 0x00
#define ON  0x01

// connect 4 switches with port B0..B3 (total 16 pin-codes)
// connect 4 LEDs with port D0..D3

// set the pincode
// switches are numbered 0..3
const int8_t pincode[4] = {ON, OFF, ON, OFF}; // matching B0..B3

void init_ports(void);

int main(void)
{
   uint8_t input, i, match;
   init_ports();
   while(1) {
      // read B0..B3

      // check if code == pincode
      match = 0;
      for (i=0; i<3; i++) {
      // check is there is a match
      ....

      } // for
      if (match) {
         // all four switches match pincode, unlock door
         PORTD=0x0f; // LEDs on
         // delay 2 sec
         _delay_ms(2000);
         PORTD=0x00; // LEDs off
      } // if

      // delay 100 ms to avoid switch bouncing
      _delay_ms(100);
   } // while(1)
}

void init_ports(void)
{
   DDRB = 0x00; // set port B0..B3 as input
   DDRD = 0xff; // set port D as output
   PORTD = 0x00; // LEDs off
}
```