Sluttest

for

Embedded Controller II

**1½.Time**

Navn: Daniel Simonsen

Hold : h3pd041121

Alt udleveret materiale må benyttes. Egne opgaver/løsninger må benyttes. Egen PC med adgang til internet må benyttes. **Mail, SMS og øvrig kommunikation med øvrige elever er ikke tilladt!!!!**

**Alle opgaver i denne test, omhandler ATMega168**  
ANSI C opgaver

1. **Skriv den nødvendige C kode, for at sætte hele PORTB til output**  
   PORTB |= (1<<0xFF);
2. **Skriv den nødvendige C kode, for at sætte hele PORTD til INPUT**  
   PORTD &= ~(1<<0xFF);
3. **Hvilken** **funktion har følgende registre i USARTEN:**  
   UBRR0H og UBRR0L = Det er dem, der styrer vores baud rate.

UBRR0H: USART Baud Rate Register High

UBRR0L: USART Baud Rate Register Low

1. **UCSR0B registeret i USARTEN sættes til værdien 0X18 i de eksempler du har arbejdet med**   
   **Hvorfor:** For at enable recieve & transmit
2. **Hvilket register i USARTEN benyttes til udveksling af data:**

UDR0: USART I/O Data Register

1. **Hvilken funktion har følgende kodelinje, og hvilke af de funktioner som du har benyttet, har været afhængig af den:  
     
   #define F\_CPU 16000000UL**F\_CPU (clockspeed) bliver sat til 16MHz, så vi kan beregne delays i sekunder.
2. **ADC’en har en opløsning på 10 bit, og en referencespænding på 5 Volt.  
    Hvilket tal vil man få fra ADC’en, hvis man aflæser 4,0Volt på en Analog indgang:**

|  |
| --- |
| 10b = 5V  10b / 5 = 5V / 5  2b = 1V  4 • 2b = 4 • 1V  8b = 4V |

1. **Hvis man har en embedded processer med en ADC, der har en opløsning på 12 bit, hvor mange digitale værdier kan man så aflæse på dem Amaloge indgang?**

1111 1111 1111 => decimal = 4.095

1. **Hvilket register i ADC’en indeholder resultatet af en ADC måling:**ADCL & ADCH
2. **Skriv den nødvendige kode, for at tænde Bit2 på PORTB. (Bit2 = 1)  
   Det forudsættes at hele porten er sat som udgang. Din kode må ikke påvirke resten af bittene på porten.**PORTB |= (1<<PB2);
3. **Skriv den nødvendige kode, for at slukke Bit2 på PORTB. (Bit2 =0)   
   Det forudsættes at hele porten er sat som udgang. Din kode må ikke påvirke resten af bittene på porten.**PORTB &= ~(1<<PB2);
4. **Skriv en korrekt if-sætning, som reagerer på, at Bit2 på PORTC, er sat. (Bit2 =1)  
   Det forudsættes at PORTC er sat op som indgang**if ( 1 == PC2 )

{

printf(”Nu kører det !!!”);

}

1. **Skriv en korrekt if-sætning, som reagerer på, at Bit3 på PORTC, ikke er sat. (Bit3 =0)  
   Det forudsættes at PORTC er sat op som indgang:**if ( 1 == PC3 )

{

printf(”Det kører stadigvæk !!?”);

}

1. **Forklar med dine egne ord, hvad en pointer i C (og andre sprog) er:**

En pointer peger på en variabel i rammen. Ved at bruge pointere, behøver rammen ikke at copy/paste variablen i sig selv igen, når variablen for eksempel bliver brugt i en metode. Dette betyder også, at hvis man ændrer på værdien af variablen i en metode, som bruger en pointer parameter i stedet for en normal variabel parameter, bliver den originale variabel ændret, hvorimod hvis det ikke var en pointer, ville de ændringer *ikke* blive gemt!

1. **Hvad skal man altid huske at gøre med en pointer, før man kan bruge den til noget. Dette skal man altid huske, ellers kan de få katastrofale konsekvenser for ens program!!!!**

Man skal altid huske at få den til at pege på en variabel. Ellers vil den, by default, pege på 0x00 i rammen, hvilke kan lede til nogle sjove/underlige/frustrerende problemer…

1. **Hvor meget hukommelse (i bytes) fylder en pointer i AtMega verdenen, der peger på en 8 bit størrelse (én byte). Altså en pointer der er defineret som følger: uint8\_t \*p\_uint8\_t; Det der spørges efter, er altså hvor mange bytes i hukommelsen pointerens selv fylder.**

uint: 4 bits

uint8 = 4 bits • 8 = 32 bits

Pointer: 4 bytes = 8bits • 4 = 32bits

unit8 (32 bits) + Pointer (32 bits) = 64 bits ⇄ 8 bytes

1. **Hvor meget hukommelse (i bytes) fylder en pointer i AtMega verdenen, der peger på en 16 bit størrelse (2 bytes størrelse). Altså en pointer der er defineret som følger: uint16\_t \*p\_uint16\_t; Det der spørges efter, er altså hvor mange bytes i hukommelsen pointerens selv fylder.**

uint: 4 bits

uint16 = 4bits • 16 = 64 bits

Pointer: 32 bits

Resultat = Pointer (32 bits) + uint16 (64 bits) = 96 bits ⇄ 12 bytes

1. **Hvor meget hukommelse (i bytes) fylder en pointer i AtMega verdenen, der peger på en 32 bit størrelse (4 bytes størrelse). Altså en pointer der er defineret som følger: uint32\_t \*p\_uint32\_t; Det der spørges efter, er altså hvor mange bytes i hukommelsen pointerens selv fylder.**

uint: 4 bits

uint32 = 4 bits \* 32 = 128 bits

Pointer = 32 bits

Resultat = Pointer (32 bits) + uint32 (128 bits) = 160 bits ⇄ 20 bytes

1. **Hvor meget hukommelse (i bytes) fylder en pointer i AtMega verdenen, der peger på en struct med en størrelse på 100 bytes. Altså en pointer der er defineret som følger: My100BytesStruct \*p\_ My100BytesStruct; Det der spørges efter, er altså hvor mange bytes i hukommelsen pointerens selv fylder.**

My100ByteStruct: 100 bytes = 800 bits

Pointer = 32 bits

Resultat = My100ByteStruct (800 bits) + Pointer (32 bits) = 832 bits ⇄ 104 bytes

1. **Hvor stort et adresseområde (i bytes) kan man adressere med en pointer, hvor pointeren selv fylder 8 bits (en byte) i hukommelsen. Spørgsmålet her er et generelt pointerspørgsmål og retter sig ikke specifikt mod ATMega-verdenen!!!**

Indtil vi ikke har mere plads i rammen

1. **Hvor stort et adresseområde (i bytes) kan man adressere med en pointer, hvor pointeren selv fylder 16 bits (to bytes) i hukommelsen. Spørgsmålet her er et generelt pointerspørgsmål og retter sig ikke specifikt mod ATMega verdenen!!!**

Indtil vi ikke har mere plads i rammen

1. Giv 3 argumenter for, at man burger pointere I C

**Argument 1:** Det fylder mindre i rammen

**Argument 2:** Hvis du vil ændre på en variabel i en metode, behøver du ikke at returnere den variabel du fik med ind i parameteren. Du kan bare tage imod en pointer og ændre på variablen, og stadig beholder dine ændringer, når du er kommet ud af metoden.

**Argument 3:** Nem måde at lave callback metoder på

1. Givet følgende kode:

**uint8\_t Number1 = 100;**

**uint8\_t \*p\_uint8\_t;**

**p\_uint8\_t = &Number1;**

**\*p\_uint8\_t = 50;**

**Hvad er værdien af Number1, efter at koden er udført.**

50

1. Givet følgende kode

**Typedef struct Koordinatsaets**

**{**

**uint8\_t x;**

**uint8\_t y;**

**} Koordinatsaet;**

**Koordinatsaet MineKoordinater;**

**Koordinatsaet \*p\_Koordinater = &MineKoordinater;**

**MineKoordinater.x = 5;**

**MineKoordinater.y = 8;**

**p\_Koordinater->x = 12;**

**Hvilken værdi vil x og y koordinaterne af MineKoordinater have, efter at koden er udført.**

0 og 0, da MineKoordinater ikke er blivet initialiseret og derfor peger på 0x00.

1. Givet følgende kode

**Typedef struct Koordinatsaets**

**{**

**uint8\_t x;**

**uint8\_t y;**

**} Koordinatsaet;**

**Koordinatsaet MineKoordinater;**

**uint8\_t \*Byte\_Ptr = &MineKoordinater.x;**

**MineKoordinater.x = 5;**

**MineKoordinater.y = 8;**

**\*Byte\_Ptr += 5;**

**MineKoordinater.y \*= \*Byte\_Ptr;**

Hvilken værdi vil x og y koordinaterne af MineKoordinater have, efter at koden er udført.

0 og 0, for MineKoordinater er stadig ikke blevet initialiseret.

1. Givet følgende kode

**uint16\_t Number = 0x1234;**

**uint8t \*Byte\_Ptr = &Number;**

**\*Byte\_Ptr \*= 2;**

**Hvilken værdi (i hex) vil Number have, efter at koden er udført.**

0x2468

1. **Giv eksempler på hvad man kan bruge functions pointerer til I C**

Du kan bruge dem til callback metoder, som f. eks. i interrupts.

1. **Hvad er betegnelsen for det, der i C# svarer til en function pointer i C**

I C# bruger vi **ref** keywordet for at **ref**erere til et sted i rammen.

1. **Hvor mange ledninger skal man bruge i alt, hvis man vil koble for eksempel et I2C bus display til sin mikroprocesser.**

2 ledninger – 1 til strøm og 1 til ground

1. **Hvor mange signalledninger er der i en I2C busforbindelse**

2 – en til clock SCL og data SDA