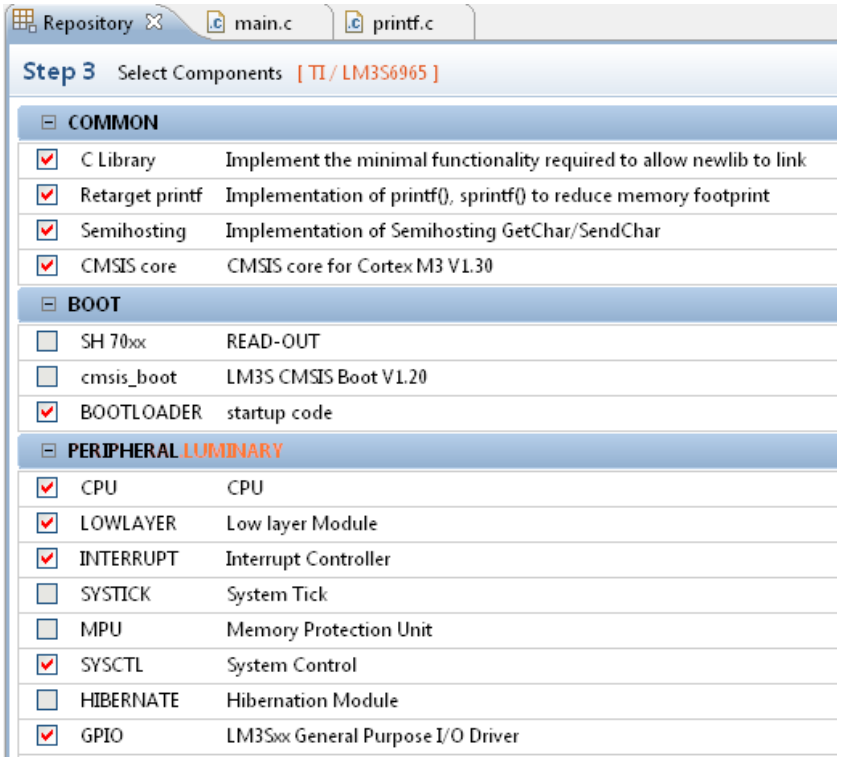
Hands on 3: deel 1- Communicatie via semihosting

Maak een nieuw project aan.
Via de repository voeg je volgende dingen toe:



De printf functie maakt onderliggend gebruik van de functie `void PrintChar(char c)` om een karakter naar buiten te sturen (zie printf.c). Breidt deze functie uit zodat de karakters via semihosting naar het scherm van de debugger worden gestuurd (zie screenshot). Je zal hiervoor de functies in semihosting.c moeten bestuderen en semihosting.h moeten includen in printf.c.

Schrijf een functie om een string in te lezen. De gebuiker geeft zijn string in via semihosting en sluit af met enter (= carriage return, linefeed = \r\n). De enter mag niet in de outputstring terecht komen. De string moet afgesloten worden met '\0'. De functie ziet er verder als volgt uit:

```
int32_t leesNString(char * output, int32_t max)
```

Output moet worden ingevuld met de ingelezen string.
max bevat de maximale lengte dat mag ingelezen worden.
de return waarde is een getal dat aangeeft hoeveel karakters er zijn ingelezen.

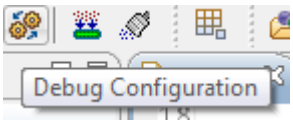
schrijf nu in main.c een applicatie die eerst uw naam vraagt via printf.
Dan via leesNString de naam inleest.

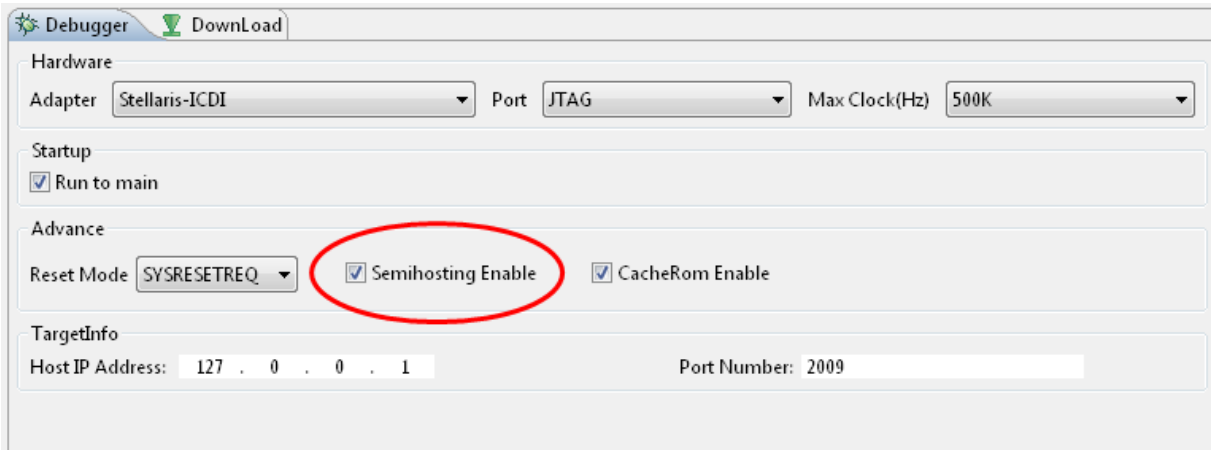
En dan een verwelkoming afdrukt in de vorm "Dag *[naam]*!" vb: Dag Bart!

Om dit programma te runnen en de output te kunnen zien heb je support voor semihosting nodig! (zie screenshots hieronder)

Instellen semihosting support

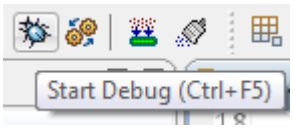
Via debug configuration krijg je een venster waar je een vinkje plaatst bij “Semihosting Enable” (zie hieronder)



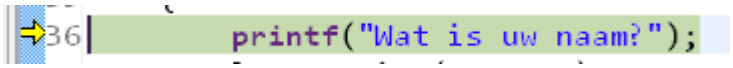


Debug run starten om via semihosting te communiceren

Voor je een debug run kan starten moet uw project foutloos gecompileerd zijn.
Indien dit zo is kan je op de knop “start debug” klikken

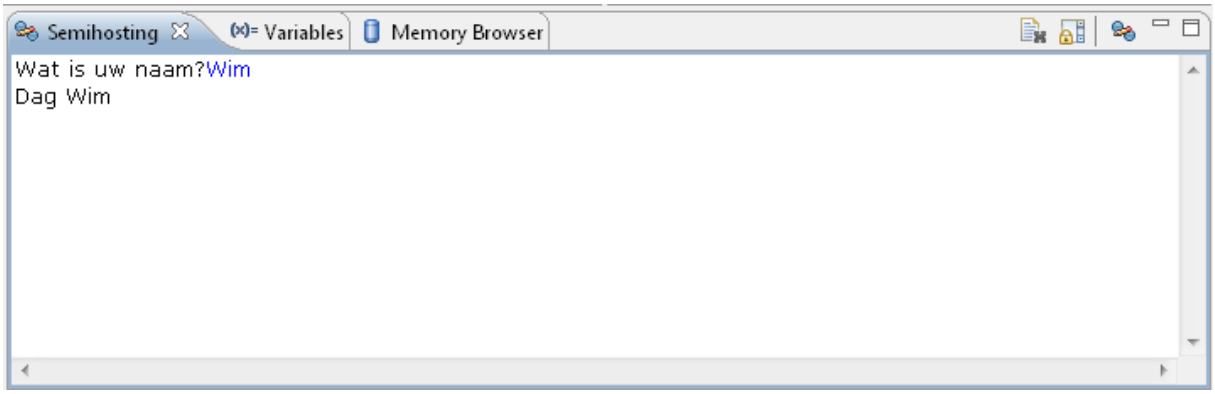


Het programma is wat groter nu dus duurt het ook wat langer om te laden.
Als het programma geladen is dan merk je dat aan een gele pijl die op een bepaalde lijn staat zoals hieronder is aangeduid.



Start nu het programma door op run te duwen.

In het semihosting venster (standaard onderaan rechts) kan je nu de vraag zien “Wat is uw naam?” Daar kan je ook uw naam ingeven en bevestigen met enter. Als alles correct werkt krijg je ook de verwelkoming te zien (zie screenshot).



Opmerking: indien de output niet is wat je verwacht kan je het document "Printf en optimalisaties.pdf" (zie trac) nalezen ivm bugs in de printf lib.

Hands on 3: deel 2- Communicatie via seriële poort

Begin met een nieuw project te maken. Bedenk goed wat je selecteert in de repository.

Tip: 

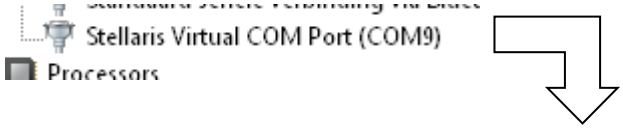
Uw programma moet juist hetzelfde doen in deel 1 maar nu via de seriële poort (bekijk ook de voorbeelden).

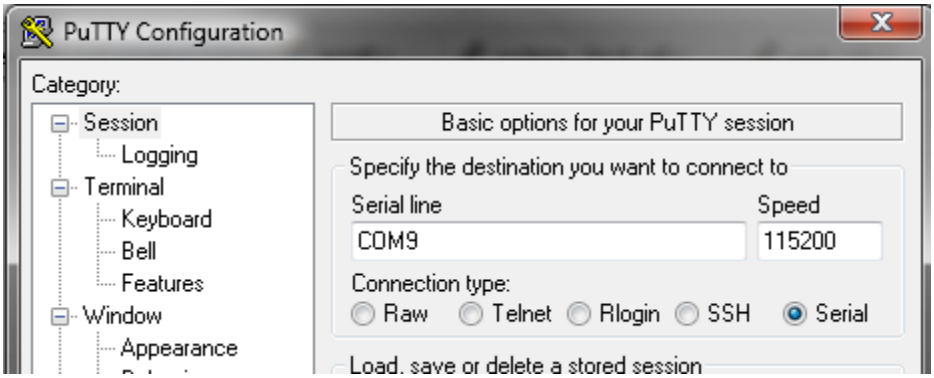
De seriële poort op je microcontroller configureren doe je in volgende stappen:

1. Stel de clock van je µC in via SysCtlClockSet.
2. Enable GPIO poort A via SysCtlPeripheralEnable.
3. Enable uart0 via SysCtlPeripheralEnable.
4. Stel de GPIO pinnen PA0 en PA1 in volgens hun alternate function via GPIOPinTypeUART.
5. Stel de baudrate in via UARTConfigSetExpClk.
6. Enable de uart via UARTEnable.

Uiteraard moet je nog een karakter kunnen sturen en ontvangen. Hiervoor heb je UARTCharPut en UARTCharGet.

Via apparaatbeheer kan je te weten komen welke com poort er moet gebruikt worden.





Hands on 3: deel 3- zowel over de seriële poort als via semihosting.

Voeg deel 1 en 2 samen en maak een nieuwe printf 'functie' die als eerste argument bepaalt waar de string terecht komt. Ofwel naar de debugger via semihosting ofwel naar de seriële poort.

Je kan met macros werken of met functies met variabele argumenten.

Voorbeeld met macro's:

```
#define ERRORPRINT(s, v...) fprintf(stderr,"Error at line %d: "s"\n", LineNumber, ## v)
```

Te gebruiken als:

```
ERRORPRINT("Waarde is %s",str);
```

Voor functie met variabele argumenten verwijs ik naar:

http://c-faq.com/varargs/varargs1.html