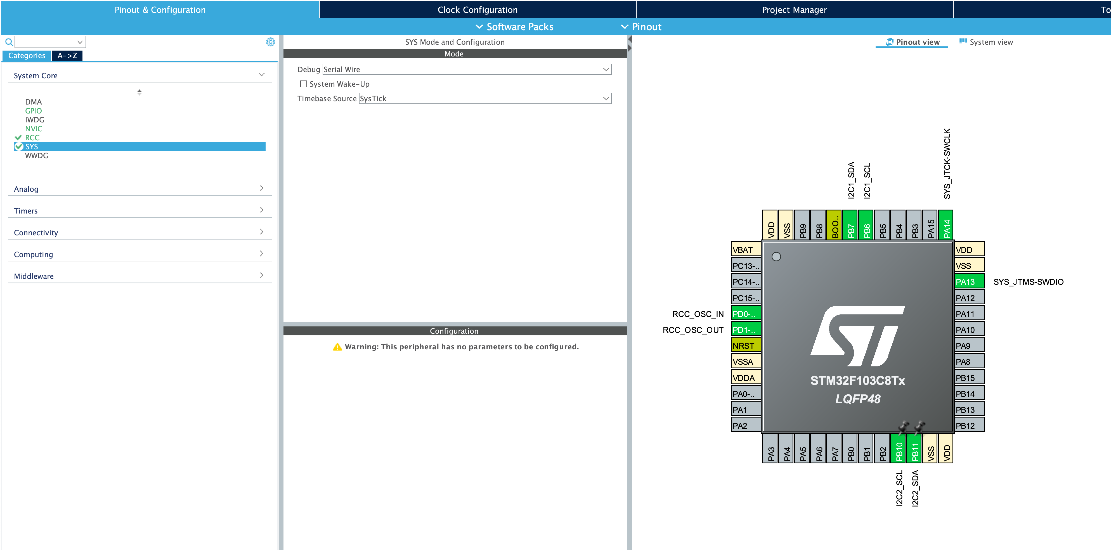
# Opsætning af gyroskop, Accel og temp.

For at opsætte gyroskopet skal du først kører den generelle opsætning

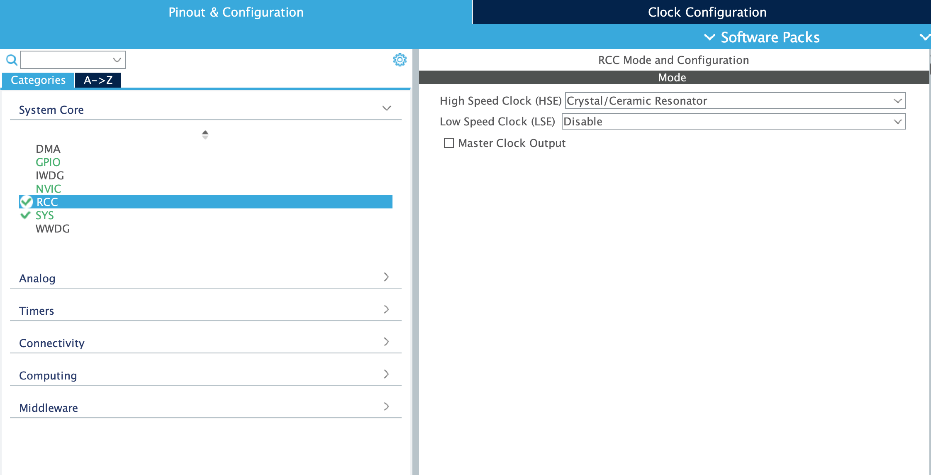
For at finde databladet så åben dette link:

<https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Register-Map1.pdf>

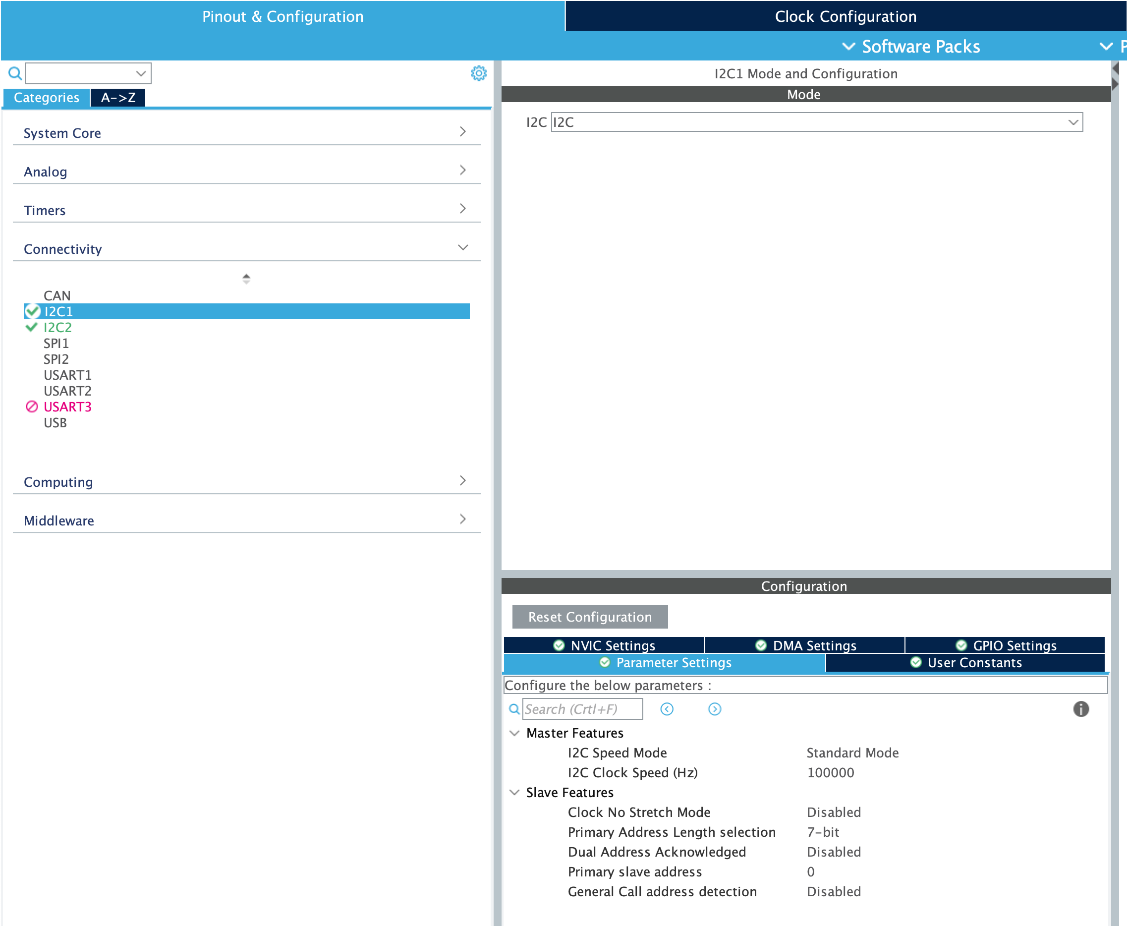
Det første vi gør, er at sætte serial wire op, så der er mulighed for at debug, som vist på billedet.



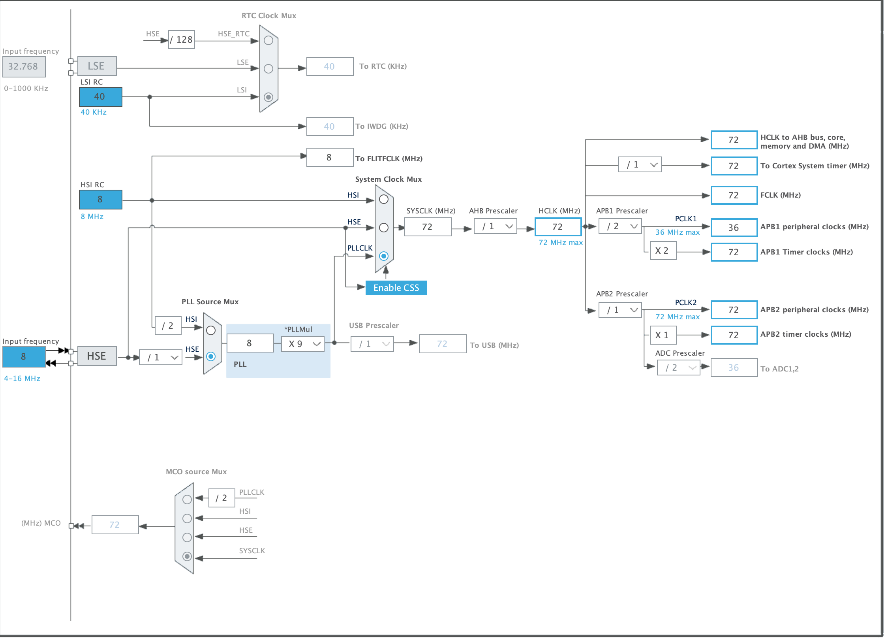
Herefter går vi ind på System core -> RCC -> HSE -> Crystal /Ceramic resonator. Det gør vi for at få STM til at give os mulighed for at indstille system clock til 72 MHz.



Vi sætter desuden I2C op, som MCU6050 kommunikere op:



Herefter skal vi ind og opsætte clk.



Det meste her er standard, men vi sætter HCLK (High CLK) til 72 MHz.

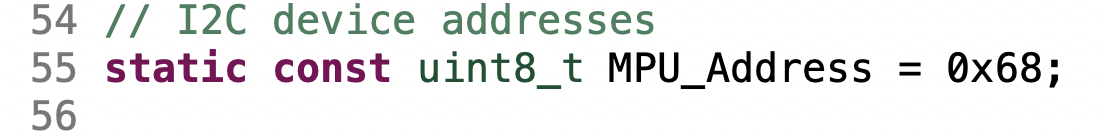
Herefter trykker vi filer->save og så ja tak til at autogenere kode.

Vi skal inkludere følgende biblioteker:

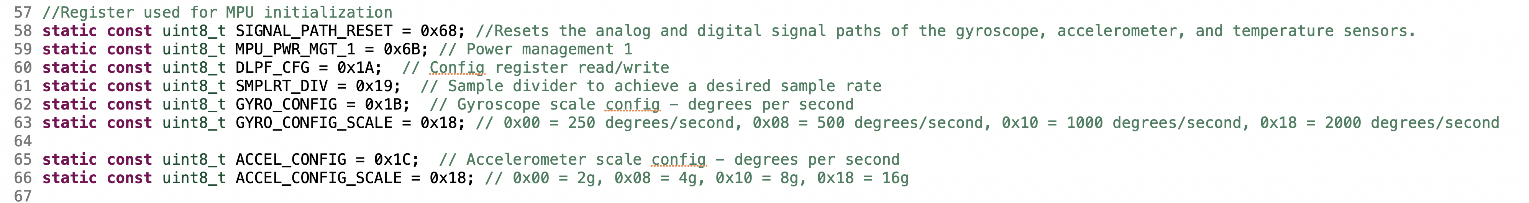
#include string.h //For at skrive strings

#include stdio.h //sprinf (printe ting ud)

I2C er en bus hvor alle enheder har en bestemt adresse på 7-bits. Hver gang der skal kommunikeres med en enhed på bussen så skal enhedens adresse transmiteres som det første. Kun den ene adresse lytter så på beskeden. MPU6050 har to adresser alt efter om Ad0 er sat høj eller lav. Ad0 lav = 0x68 og Ad0 høj = 0x69. Vi anvender Ad0 som er høj og derfor skriver en variabel, som er lig 0x68, som vist på billedet.



For at kunne kommunikere med MPU, så skal den konfigureres efter behov. Den konfigureres ved at man skriver til forskellige adresser. Følgende adresser er brugt til konfigureringen, som er fundet i databladet:



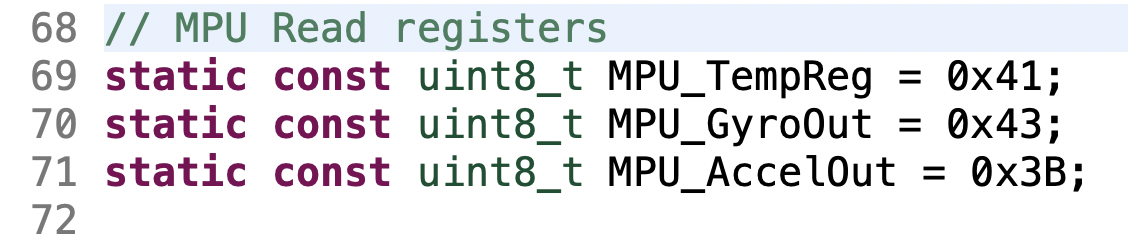
*Signal path reset evt. filter.*

*Slår Low-pass filter til eller fra(fjerner støj og spikes fra målinger) og definering af sample rate på gyro.*

*Gyro\_config og Accel\_config er hvilke registrer vi skal snakke til.*

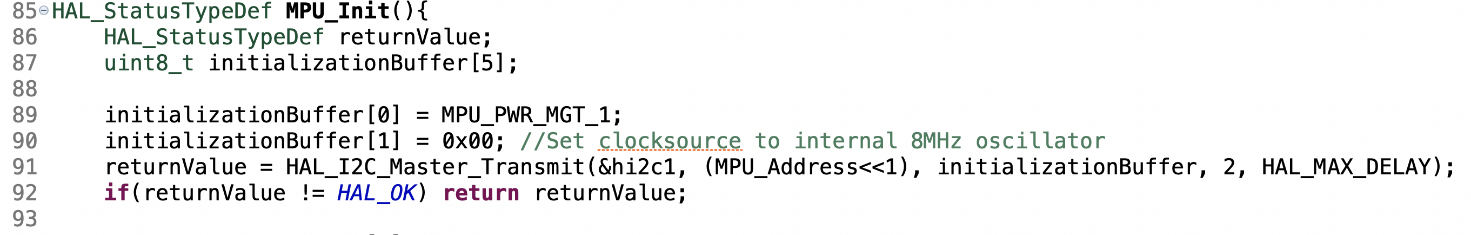
*Gyro\_config\_scale og Accel\_config\_scale er hvilken følsomhed/spænd vi ønsker (Se databladet for mere information)*

Herefter laver vi opsætning af hvilke adresse, som vi ønsker at læse fra:



Alle værdier vi ønsker at læse er delt over 2 registre, hvor der er 8-bit i hver register, men vi behøver blot at oplyse den først da MPU selv inkrementere register efter hver byte (Eksempel på dette kommer i kode senere).

Herefter definere vi en funktion, som konfiguere MPU’en.



Som det ses på linje 89 så puttes register adressen ind på første plads i bufferen. Herefter indsættes de 8-bit vi ønsker at sende på den efterfølgende plads i bufferen.

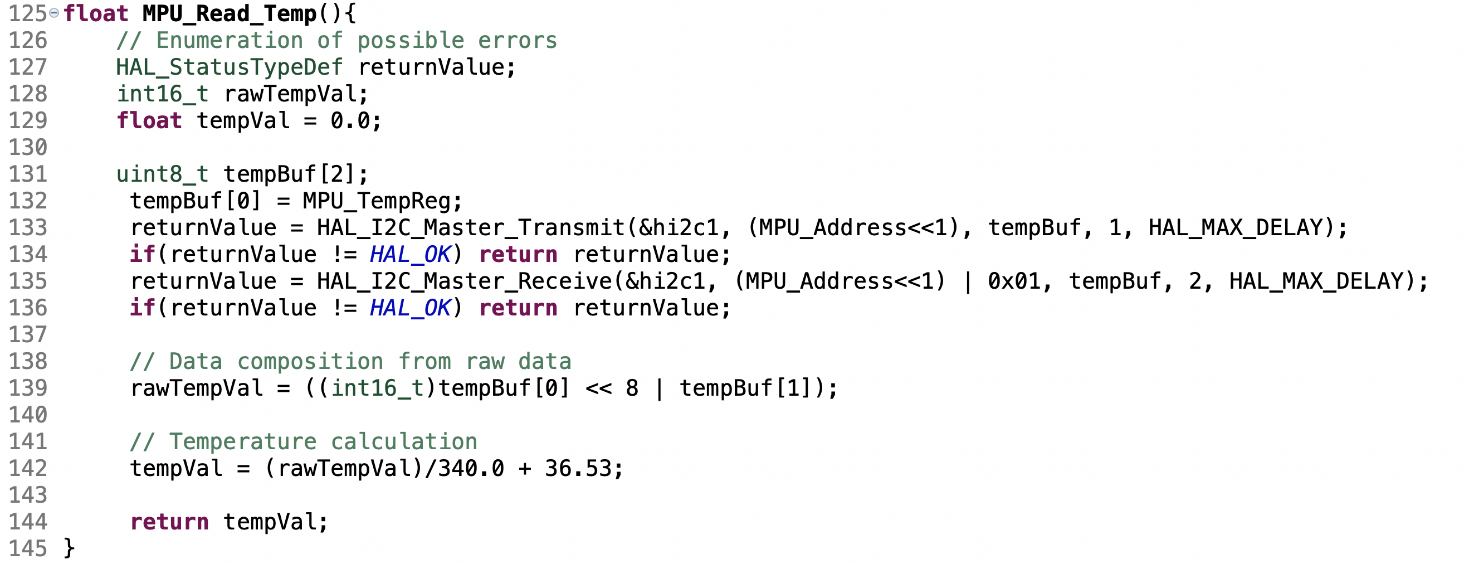
På linje 91 transmittere vi til MPU’en. Den første parameter er vores I2C handle, som er autogenereret. Næste parameter er adressen på enheden, som vi ønsker at skrive til. OBS. Denne skal bitshiftes en gang til venstre da adresser kun er 7-bit lange og den sidste bit indikere om vi vil læse eller skrive til enheden (bitten længst til højre bliver sat til 0). 0 i dette tilfælde gør at vi vil skrive til MPU’en. Den tredje parameter fortæller hvor det vi vil transmittere ligger henne. Den anden sidste er mængden af bytes vi vil transmittere og den sidste er hvor lang tid vi vil på reaktion fra slave-enheden, her HAL\_MAX\_DELAY, som er rigtig lang tid.

Det er vigtig at huske HAL\_I2C\_Master\_Transmit er en blokerende funktion, så den optager CPU’en indtil kommunikationen er overstået.

Dette er så færdig for Power management og herefter skal det også gøres på samme måde for alle registre nævnt i konfigurationen.

Nu er alt opsat og vi går i gang med funktioner, som giver os værdier fra de respektive sensorer.

## Vi laver nu en funktion til at læse temperaturen:



*Linje 132: Første plads i bufferen sættes til det register vi ønsker at læse fra.*

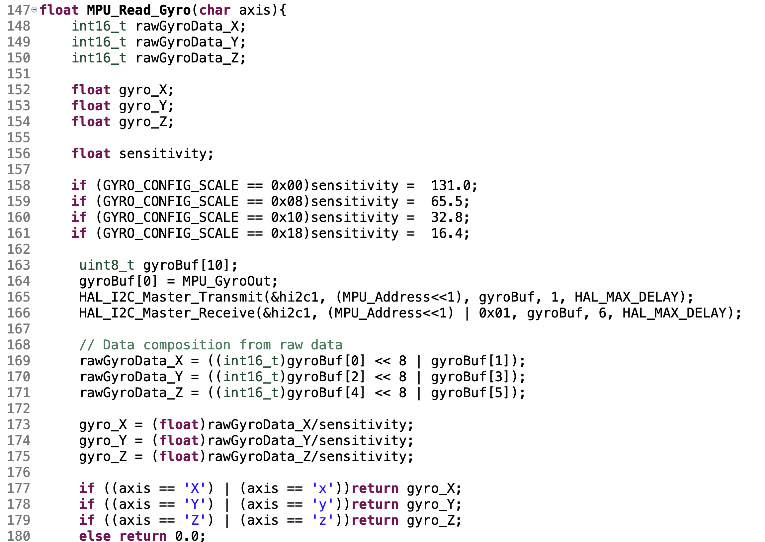
*135: HAL\_I2C\_Master\_Recieve fungere meget ligesom Transmit, hvor anden sidste plads er hvor meget vi forventer at modtage og pladsen i bufferen inkrementeres automatisk efter hver byte.*

*OBS. 2 parameter i recieve-funktionen er ”and” med et et-tal. Dette er for at den sidste bit indikerer at vi ønsker at læse.*

*Linje 139: Vi sammensætter de første to pladser i bufferen til en int 16 værdi.*

*Linje 142: Her omregnes til temperatur i grader-celsius jf. databladet.*

## Vi laver nu funktion for Gyro:



På 158-161 definere vi variablen sensitivity, som er en omregningsfaktor på baggrund af hvilken følsomhed/spænd, som er definere tidligere i programmet. (Se Gyro\_Config\_Scale).

Den første plads i bufferen defineres som adressen på Gyroen ligesom i temperaturen.

Linje 165 og 166 følger samme procedure som for temperatur-sensoren.

Linje 169-171 er også det samme blot med 3 variabler.

173-175 sker der en omregning på baggrund af den valgte følsomhed/spænd.

177-179 returnerer data fra en akse på baggrund af hvilken input-parameter, som er givet til funktionen.

## Procedure for accelerometer:

Dette er præcis det samme som for gyroen.