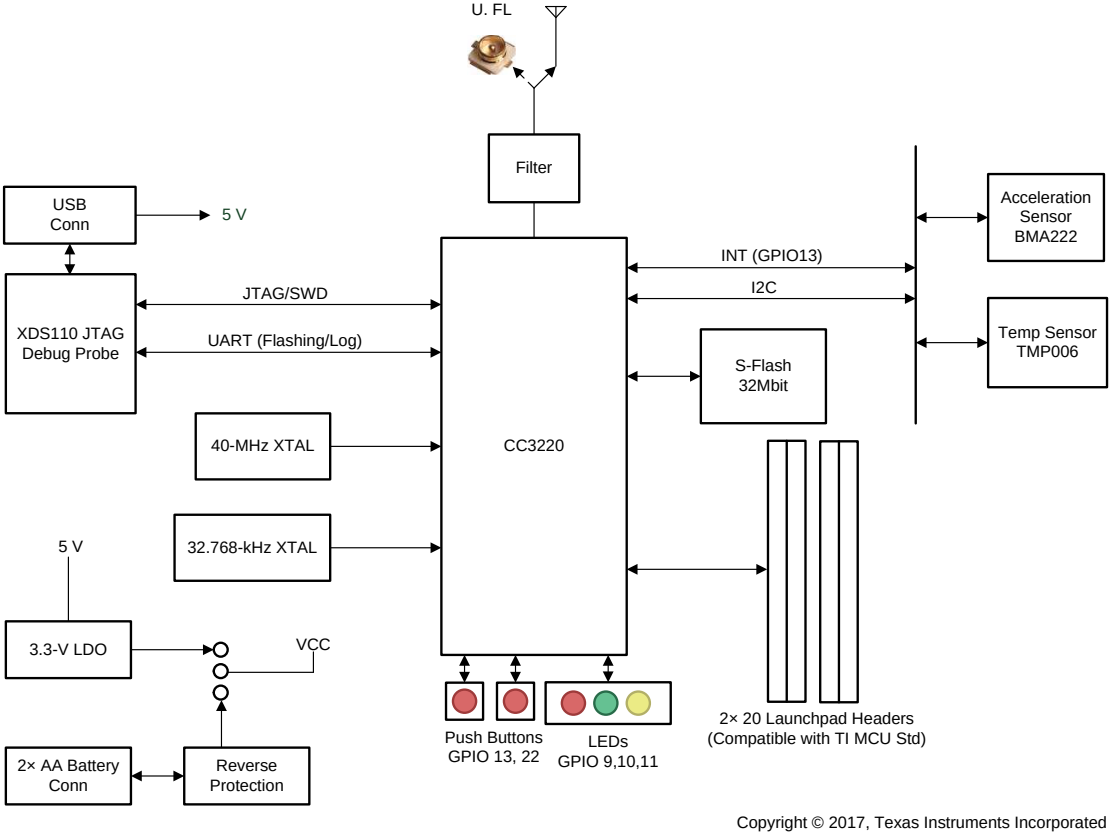
Opdrachten week 3 lab 2 – I²C-driver

Vorige les heb je de GPIO-module van de CC3220S LaunchPad (zie figuur 1) leren kennen. Je hebt de GPIO-module aangesproken via registers en de *TI Drivers*. Ook heb je gezien hoe je de UART-driver kan gebruiken om karakters naar de pc te sturen en van de pc te ontvangen. Deze les gaan we verder met de *TI Drivers* en leer je:

- wat I²C is en hoe je de I²C-driver gebruikt;
- de TMP006 IR-sensor via I²C uit te lezen.

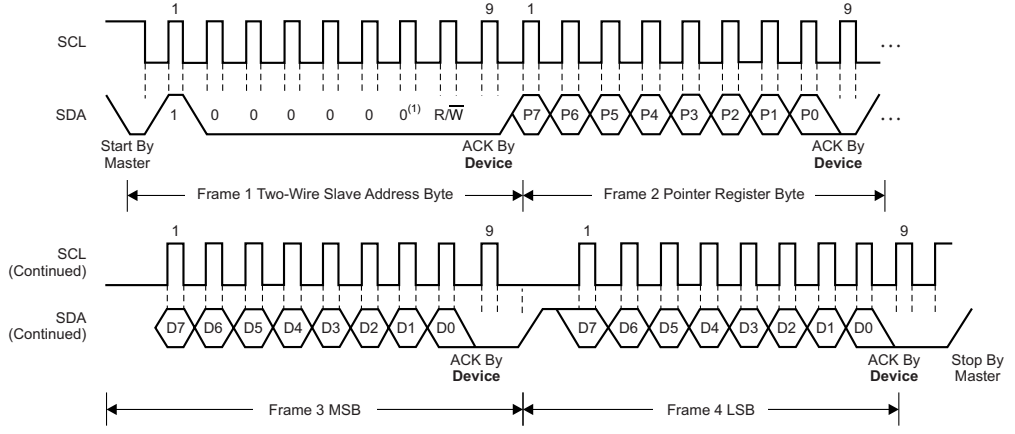


Figuur 1: Opbouw van de TI SimpleLink™ Wi-Fi® CC3220S LaunchPad™.

Voordat we de temperatuursensor kunnen gebruiken is het handig te weten wat I²C is en hoe we daarmee de temperatuursensor kunnen benaderen. I²C staat voor “Inter IC Communication” en is ontwikkeld door Philips. Het is een seriële bus die, door middel van adressen, meerdere chips aan elkaar kan verbinden. Er is altijd één master op de bus, en één

of meerdere slaves. In ons geval is de CC3220S de master en is de TMP006 sensor als slave aangesloten met adres 0x41.

De communicatie op de bus gebeurt via twee signalen: het klok signaal (SCL) en het data signaal (SDA). Door te spelen met flanken en timing van de twee signalen kan de bus arbitratie worden geregeld. Figuur 2 laat zien hoe bijvoorbeeld de communicatie plaatsvindt om 2 bytes naar een specifiek register te schrijven binnen de sensor.



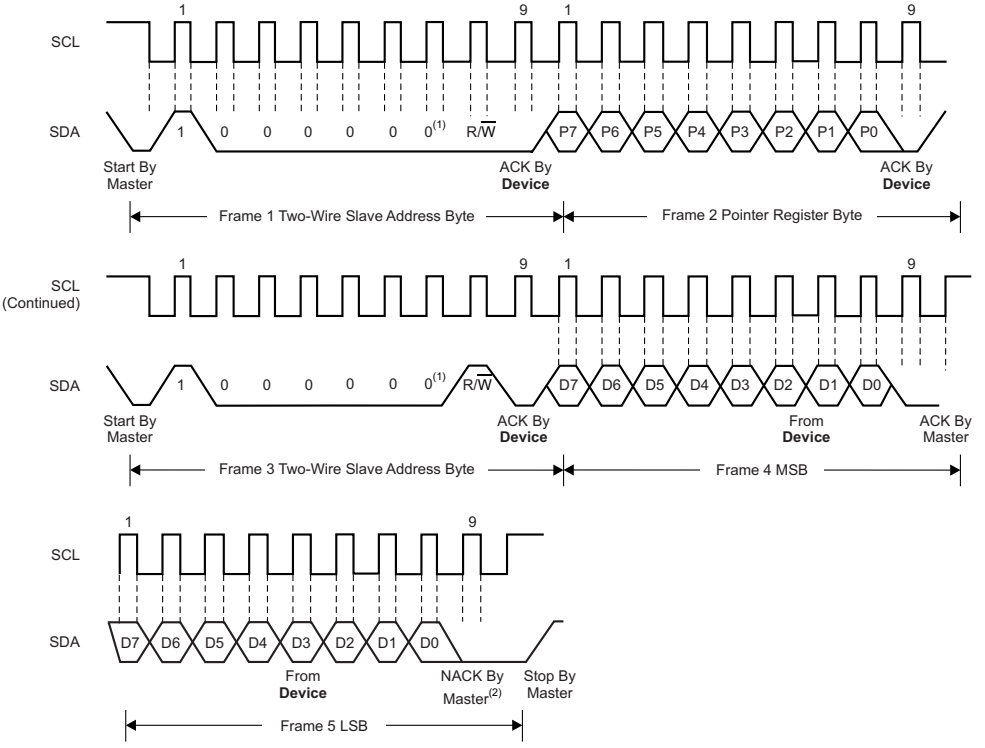
Figuur 2: 2 Bytes schrijven naar een register via I²C. Bron: TMP006-datasheet.

Dingen als ACK(nowledge) en N(ot)ACK(nowledge), die deel uitmaken van de busarbitrage, wordt gelukkig afgehandeld door de peripheral binnen de µC. De I²C-driver handelt ook de overige zaken af, dus het enige wat rest is het opgeven van het slave-adres en zeggen wat je wilt schrijven of lezen. Figuur 3 laat zien hoe twee bytes worden gelezen uit een register. Hiervoor wordt eerst 1 byte geschreven om de sensor te vertellen welk register moet worden gelezen. Daarna worden 2 bytes gelezen die in dat sensorregister staan.

Als eerste is het belangrijk om de documentatie van de drivers beschikbaar te hebben. Die kun je vinden via de resource explorer in CSS, op de harde schijf onder `file:///C:/ti/simplelink_cc32xx_sdk_6_10_00_05/docs/drivers/doxygen/html/index.html` of online via deze link.

De I²C-driver werkt als volgt:

- Met `I2C_open()` kan een specifieke peripheral worden gekoppeld aan een `I2C_Handle` variabele. Bijvoorbeeld module I2C0. De `I2C_Handle` wordt door elke I²C-functie gebruikt.



Figuur 3: 2 Bytes lezen uit een register via I²C. Bron: TMP006-datasheet.

- De driver werkt met transacties. Elke `I2C_Transaction` beschrijft het slave-adres, de schrijf/lees plek en het aantal bytes wat je wilt schrijven/lezen. Zo kun je een `I2C_Transaction` variabele definiëren voor het uitlezen van de temperatuur en nog een andere variabele voor het uitlezen van de spanning.
- De functie `I2C_transfer()` kan vervolgens op basis van de gegeven transactie de communicatie ook werkelijk uitvoeren en afhandelen.

3.2.1

Als eerste gaan we een leeg project aanmaken dat geschikt is om met behulp van de TI Drivers te gaan programmeren.

- Kies Project > Import CCS Projects... en klik op Browse. Navigeer naar het directory: `C:\ti\simplelink_cc32xx_sdk_6_10_00_05\examples\nortos\CC3220S_LAUNCHXL\drivers\gpiointerrupt\ccs`. Klik op Finish.
- Hernoem het project naar `temperatuur_CC3220S_LAUNCHXL_nortos_ccs`.
- Hernoem `gpiointerrupt.c` (op dezelfde wijze) naar `temperatuur.c`.

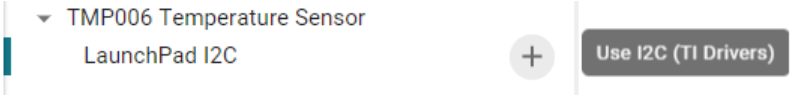
- Hernoem `gpiointerrupt.syscfg` naar `temperatuur.syscfg`.
- Kies Project > Properties en selecteer (indien nodig) CCS General. Klik op de knop Manage Configurations... en maak de Debug configuratie Active. Delete vervolgens de MCU+Image configuratie[1]. Klik tot slot op OK en op Aply and Close.
- Verwijder `main_nortos.c`, `Board.html`, `image.syscfg`, `MCU+Image`, `userFiles`, `README.html` en `README.md` uit het project.
- Vervang de code in `temperatuur.c` door de code die gegeven is in listing 1. Je kunt deze code eenvoudig kopiëren met behulp van deze link: `leeg.c`.
- Debug en start het programma. Als het goed is knippert de rode led.

```c
/*
 * Copyright (c) 2024, Rotterdam University of Applied Sciences
 * All rights reserved.
 */

#include <stdint.h>
// Driver Header files
#include <NoRTOS.h>
#include <ti/drivers/GPIO.h>
// Driver configuration
#include "ti_drivers_config.h"

int main(void) {
    Board_init();
    NoRTOS_start();
    // Turn off red led
    GPIO_write(CONFIG_GPIO_LED_0, CONFIG_GPIO_LED_OFF);

    while (1) { // Loop forever blinking
        volatile int count = 0;
        for (; count < 1000000; count++) /* wait */;
        GPIO_toggle(CONFIG_GPIO_LED_0);
    }
}
```

Listing 1: Basisprogramma voor het gebruik van de TI-drivers, zie `leeg.c`

[1] Deze configuratie is nodig als we het programma in het Flash geheugen willen laden. In dit geval willen we het programma alleen in RAM laden om het te kunnen debuggen (in de Debug configuratie).

3.2.2 Open de file `temperatuur.syscfg` in de system configuration tool. Klik op het knopje *Hardware* (links) en koppel de *TMP006 Temperature Sensor* met de *LaunchPad I2C*, zie figuur 4. Dit geeft een error! Deze error wordt veroorzaakt door het feit dat de I²C-bus gebruik maakt van dezelfde pinnen als waarop de gele en groene led zijn aangesloten. Op dit moment is de groene led verbonden met de GPIO-driver en daarom kan deze pin niet gekoppeld worden aan de I²C-driver. Als je de groene led 'losmaakt' van de GPIO-driver, verdwijnt de error.



Figuur 4: Configureer de I²C-driver.

3.2.3 Open de documentatie van de algemene I²C-driver. Gebruik de voorbeeldcode uit de driver om de I²C-bus met standaard instellingen te initialiseren, te openen en een lege transactie aan te maken met het `slaveAdress` van `0x41` (de adres van de temperatuursensor). De transactie zelf vullen we in de volgende opdracht in.

3.2.4 De TMP006 heeft een "manufacturer register" met de standaard waarde "TI". Als eerste gaan wij dit register uitlezen om te controleren of wij succesvol kunnen verbinden met de TMP006. Stel de eerder aangemaakte `I2C_Transaction` variabele in om het juiste register uit te lezen. Zoals in figuur 3 zichtbaar is, zul je 1 byte moeten sturen om aan te geven dat je het manufacturer register wilt uitlezen, en 2 bytes moeten ontvangen (de inhoud van het register). Gebruik dus figuur 3, de TMP006-datasheet en de voorbeeldcode uit de documentatie, om de juiste instellingen te vinden voor de transactie[2].

Voer de transactie uit en controleer in de debugger of inderdaad "TI" wordt teruggegeven.

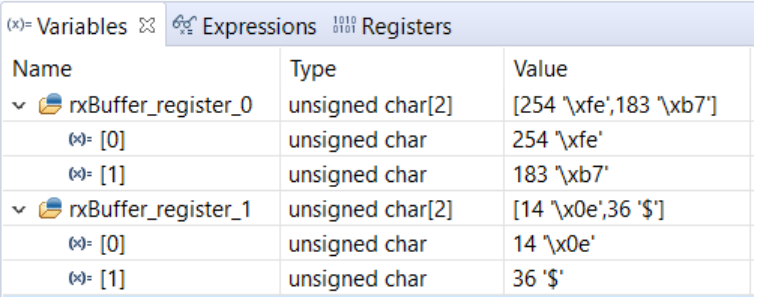
De TMP006 is een IR-sensor[3] die de temperatuur van een object erboven (T_{OBJ}) kan waarnemen door de temperatuur van de sensor (T_{DIE}) te meten en ook de IR-inval op de sensor, die wordt bepaald door een interne spanning in de sensor te meten (V_{OBJ}). Op basis hiervan kan

[2] Als je er niet zelfstandig uitkomt, dan vind je hier enkele tips: https://bitbucket.org/HR_ELEKTRO/ems20/wiki/tipsI2C.md.

[3] IR staat voor Infrarood, zie https://nl.wikipedia.org/wiki/Infrarood.

een schatting worden gemaakt van de temperatuur van het object (T_{OBJ}). Dit is allemaal uitgelegd in de TMP006 User's Guide. Hoofdstuk 5 legt uit wat moet worden berekend.

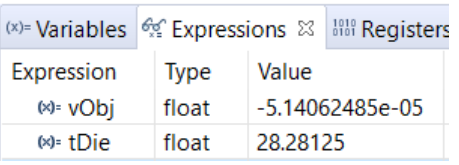
3.2.5 Om V_{OBJ} en T_{DIE} te kunnen berekenen moeten register 0 en register 1 van de TMP006 uitgelezen worden. Maak om dit te doen twee `uint8_t` arrays en twee transactions aan om de spanning en temperatuur uit de betreffende TMP006 registers te lezen. Ook dit staat beschreven in de datasheet van de TMP006. Lees de data in door het uitvoeren van de transacties en bekijk in de debugger welke waarden worden teruggegeven. Zie figuur 5 voor een mogelijk resultaat. Deze waarden zijn natuurlijk in jouw geval anders omdat ze afhankelijk zijn van de temperatuur en de hoeveelheid IR-inval.



Figuur 5: De ingelezen waarden van register 0 en 1 van de TMP006.

3.2.6 Zowel de spanning als de temperatuur staan nu in arrays van unsigned 8-bit integers. In realiteit zijn zowel de spanning als temperatuur signed integer getallen van 16 bits. Maak twee `int16_t` variabelen aan voor de spanning en temperatuur en geef deze de juiste waarden op basis van de ontvangen data in de arrays. Byte 0 van de arrays komt op bitposities 15-8 en byte 1 op bitposities 7-0 van de `int16_t` variabelen. Je kunt hiervoor bit shifting (<<) en de bitwise OR (|) gebruiken. Controleer het resultaat in de debugger.

3.2.7 De temperatuur en de spanning moeten nog geschaald worden. Maak twee **`float`** variabelen aan om de spanning in volts en de temperatuur in graden Celsius in op te slaan. De benodigde schaling is te vinden in de datasheet op pagina 19 respectievelijk pagina 15. Bereken V_{OBJ} en T_{DIE} met behulp van de twee `int16_t` variabelen uit opdracht 3.2.6. Bekijk het resultaat in de debugger. Zie figuur 6 voor een mogelijk resultaat.



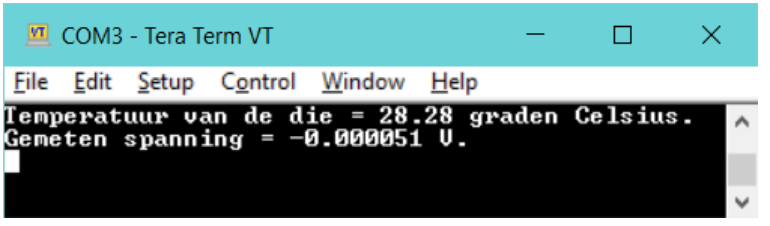
Figuur 6: De berekende waarden van V_{OBJ} en T_{DIE}.

Je gaat nu de gemeten temperatuur T_{DIE} en de gemeten spanning V_{OBJ} versturen naar de pc via een UART-verbinding.

3.2.8 Configureer de UART met behulp van de system configuration tool. Zorg er nu voor dat in de terminal wordt geprint:
`Temperatuur van de die = XX.XX graden Celsius.`
en op de volgende regel:
`Gemeten spanning = X.XXXXXX V.`
Zie figuur 7. Je kunt voor het omzetten van een **`float`** naar een `string`, de volgende code gebruiken:

```
sprintf(string,"mijn tekst %.2f", mijnfloatvariabele)
```

De `%.2f` wordt dan vevangen door de waarde van `mijnfloatvariabele`. De `.2` geeft aan dat er twee cijfers achter de punt afgedrukt moeten worden.



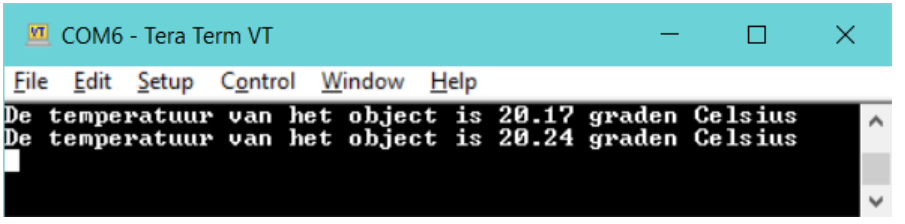
Figuur 7: Voorbeeld uitvoer in de terminal.

De belangrijkste leerdoelen van dit lab heb je met het uitvoeren van de voorgaande opgaven behaald. De volgende opgave behoort niet tot de verplichte stof voor EMS20 en is bedoeld voor studenten die het leuk vinden om de temperatuur van het object dat zich boven de TMP006 bevindt te bepalen.

Je gaat nu met behulp van T_{DIE} en V_{OBJ} de temperatuur van het object (T_{OBJ}) berekenen en het resultaat versturen naar de pc via een UART-verbinding.

3.2.9 Schrijf nu een functie om de spanning (V_{OBJ}) en temperatuur van de chip (T_{DIE}) om te zetten naar de temperatuur van het object (T_{OBJ}). Gebruik de formules die gegeven zijn in paragraaf 5.1 van de TMP006 User's Guide. De waarde van S_0 moet gekalibreerd worden, zoals beschreven in hoofdstuk 6 van de User's Guide. Dat is zoals je kunt lezen een hele klus en je hebt er ook specifieke testapparatuur voor nodig. Ga er in de functie die T_{OBJ} berekent vanuit dat geldt: $S_0 = 6{,}0 \times 10^{-14}$.

3.2.10 Zorg er nu voor dat telkens wanneer in de terminal op een knop wordt gedrukt wordt geprint:
`De temperatuur van het object is xxx.xx graden Celsius`
Zie figuur 8.



Figuur 8: Voorbeeld uitvoer in de terminal.