#include <stdio.h>

#include "boards.h"

#include "app\_util\_platform.h"

#include "app\_error.h"

#include "nrf\_drv\_twi.h"

#include "nrf\_delay.h"

#include "nrf\_log.h"

#include "nrf\_log\_ctrl.h"

#include "nrf\_log\_default\_backends.h"

#include "bme680.h"

#include "bme680\_defs.h"

struct bme680\_dev gas\_sensor;

int8\_t rslt = BME680\_OK;

/\* TWI instance ID. \*/

#define TWI\_INSTANCE\_ID 0

/\* TWI instance. \*/

static const nrf\_drv\_twi\_t m\_twi = NRF\_DRV\_TWI\_INSTANCE(TWI\_INSTANCE\_ID);

int8\_t user\_i2c\_read (uint8\_t dev\_id, uint8\_t reg\_addr, uint8\_t \*data, uint16\_t len)

{

ret\_code\_t err;

uint8\_t var[30]={0};

int size=0, result = 0, j=0;

uint8\_t reg[1]={reg\_addr};

nrf\_drv\_twi\_tx(&m\_twi, dev\_id, reg, 1,false);

err = nrf\_drv\_twi\_rx(&m\_twi, dev\_id, var, len);

if(err==0)

{

size=1;

for(int i = 0 ;i<30;i++)

{

if(var[i]!=0) size=i+1;

}

}

else

{

size=0;

}

if(size ==15)

size=len;

if(len<=size)

{

for (int i=0;i<len;i++)

{

data[i]=var[i];

}

}

else

{

result=5;

}

return result;

}

int8\_t user\_i2c\_write (uint8\_t dev\_id, uint8\_t reg\_addr, uint8\_t \*data, uint16\_t len)

{

int result=0;

uint8\_t reg[]={reg\_addr,\*data};

nrf\_drv\_twi\_tx(&m\_twi, dev\_id, reg, len+1, false);

for(int i = 0;i<len;i++)

return result;

}

void user\_delay\_ms (uint32\_t period)

{

nrf\_delay\_ms(period);

}

void twi\_init (void)

{

ret\_code\_t err\_code;

const nrf\_drv\_twi\_config\_t twi\_config = {

.scl = 27,

.sda = 26,

.frequency = NRF\_DRV\_TWI\_FREQ\_100K,

.interrupt\_priority = APP\_IRQ\_PRIORITY\_HIGH,

.clear\_bus\_init = false

};

err\_code = nrf\_drv\_twi\_init(&m\_twi, &twi\_config, NULL, NULL);

APP\_ERROR\_CHECK(err\_code);

nrf\_drv\_twi\_enable(&m\_twi);

}

void init\_bme (void)

{

uint8\_t set\_required\_settings;

/\* Set the temperature, pressure and humidity settings \*/

gas\_sensor.tph\_sett.os\_hum = BME680\_OS\_2X;

gas\_sensor.tph\_sett.os\_pres = BME680\_OS\_4X;

gas\_sensor.tph\_sett.os\_temp = BME680\_OS\_8X;

gas\_sensor.tph\_sett.filter = BME680\_FILTER\_SIZE\_3;

/\* Set the remaining gas sensor settings and link the heating profile \*/

gas\_sensor.gas\_sett.run\_gas = BME680\_ENABLE\_GAS\_MEAS;

/\* Create a ramp heat waveform in 3 steps \*/

gas\_sensor.gas\_sett.heatr\_temp = 300; /\* degree Celsius \*/

gas\_sensor.gas\_sett.heatr\_dur = 100; /\* milliseconds \*/

/\* Select the power mode \*/

/\* Must be set before writing the sensor configuration \*/

gas\_sensor.power\_mode = BME680\_FORCED\_MODE;

/\* Set the required sensor settings needed \*/

set\_required\_settings = BME680\_OST\_SEL | BME680\_OSP\_SEL | BME680\_OSH\_SEL | BME680\_FILTER\_SEL| BME680\_GAS\_SENSOR\_SEL;

/\* Set the desired sensor configuration \*/

rslt = bme680\_set\_sensor\_settings(set\_required\_settings,&gas\_sensor);

/\* Set the power mode \*/

rslt = bme680\_set\_sensor\_mode(&gas\_sensor);

}

int main(void)

{

ret\_code\_t err\_code;

APP\_ERROR\_CHECK(NRF\_LOG\_INIT(NULL));

NRF\_LOG\_DEFAULT\_BACKENDS\_INIT();

NRF\_LOG\_INFO("TWI scanner started.");

NRF\_LOG\_FLUSH();

twi\_init();

gas\_sensor.dev\_id = BME680\_I2C\_ADDR\_SECONDARY;

gas\_sensor.intf = BME680\_I2C\_INTF;

gas\_sensor.read = user\_i2c\_read;

gas\_sensor.write = user\_i2c\_write;

gas\_sensor.delay\_ms = user\_delay\_ms;

gas\_sensor.amb\_temp = 25;

rslt = bme680\_init(&gas\_sensor);

init\_bme();

NRF\_LOG\_INFO("chip\_id : %d :",gas\_sensor.chip\_id);

NRF\_LOG\_FLUSH();

NRF\_LOG\_INFO("Rslt : %d",rslt);

NRF\_LOG\_FLUSH();

if(rslt!=0)

{

while(1);

}

uint8\_t data[1]={140};

uint8\_t regt[1]={0x74};

uint8\_t regh[1]={0x72};

uint8\_t regs[1]={0x71};

uint8\_t addr = 0xE0;

uint8\_t val = 0xB6;

bme680\_set\_regs(&addr, &val, 1, &gas\_sensor);

bme680\_set\_regs(regt, data, 1, &gas\_sensor);

bme680\_set\_regs(regh, data, 1, &gas\_sensor);

bme680\_set\_regs(regs, data, 1, &gas\_sensor);

//NRF\_LOG\_INFO("Register1 0x74 : %d",dat[0]);

nrf\_delay\_ms(500);

//NRF\_LOG\_INFO("Register2 0x72 : %d",dat[0]);

nrf\_delay\_ms(500);

NRF\_LOG\_FLUSH();

while (1)

{

uint16\_t meas\_period;

bme680\_get\_profile\_dur(&meas\_period, &gas\_sensor);

struct bme680\_field\_data data;

char temperature[20];

char pressure[20];

char humidity[20];

while(1)

{

nrf\_delay\_ms(meas\_period);

rslt = bme680\_get\_sensor\_data(&data, &gas\_sensor);

//NRF\_LOG\_INFO("Read Result: %d",rslt);

NRF\_LOG\_FLUSH();

//NRF\_LOG\_INFO("%d",data.temperature);

nrf\_delay\_ms(2000);

NRF\_LOG\_FLUSH();

NRF\_LOG\_ERROR( "TEMPERATURE : " NRF\_LOG\_FLOAT\_MARKER "\r\n", NRF\_LOG\_FLOAT(data.temperature /100.0));

NRF\_LOG\_FLUSH();

nrf\_delay\_ms(2000);

NRF\_LOG\_ERROR( "PRESSURE : " NRF\_LOG\_FLOAT\_MARKER "\r\n", NRF\_LOG\_FLOAT(data.pressure/1000.0));

NRF\_LOG\_FLUSH();

nrf\_delay\_ms(2000);

NRF\_LOG\_ERROR( "HUMIDITY: " NRF\_LOG\_FLOAT\_MARKER "\r\n", NRF\_LOG\_FLOAT(data.humidity /1000.0));

NRF\_LOG\_FLUSH();

nrf\_delay\_ms(2000);

//NRF\_LOG\_ERROR( "GAS: " NRF\_LOG\_FLOAT\_MARKER "\r\n", NRF\_LOG\_FLOAT(data.gas\_resistance /1000.0));

//NRF\_LOG\_FLUSH();

//nrf\_delay\_ms(500);

if (data.status & BME680\_GASM\_VALID\_MSK)

{

NRF\_LOG\_ERROR( "GAS RESISTANCE : " NRF\_LOG\_FLOAT\_MARKER "\r\n", NRF\_LOG\_FLOAT(data.gas\_resistance /1000.0));

NRF\_LOG\_FLUSH();

nrf\_delay\_ms(500);

}

}

}

}