Hi!

I don’t understand your answers.

“The register 0x3D contains the working mode of sensor and we expect all setting changes can only happen in "Config mode". ***The register write happened under non "config mode****" will have different values when you read back.”*

In your reply you claim that the register write was not made in “config mode”. Please indicate below which writes you are referring to.

The value of register 0x3d is marked in **green** before writing to any other register.

Regarding the delays when shifting to/from Config Mode I was referring to the difference between your BNO055 driver and the data-sheet.

In the function: ’bno055_set_operation_mode’ 600ms and 20ms are given, please see below highlighted with **blue**.

Finally your comment about the 7ms is not relevant. Have a look in Table 3.6 of the data-sheet.

Looking forward to your reply!

Regards,
Göran

########################## Test case begin ##################

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x02
Write to register: 0x3e Cnt in write: 1 Byte(s): 0x00
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x00
Power Mode value: 0x00 <---------- Power Mode = NORMAL

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Write to register: 0x3d Cnt in write: 1 Byte(s): 0x1c
Read from register: 0x3d Cnt in read: 1 Byte(s): 0x1c
Operation Mode value: 0x0c <---------- Operation Mode = OPERATION_MODE_NDOF

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x1c
Read from register: 0x3d Cnt in read: 1 Byte(s): 0x1c
Read from register: 0x3d Cnt in read: 1 Byte(s): 0x1c
Write to register: 0x3d Cnt in write: 1 Byte(s): 0x10
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x00
Write to register: 0x3e Cnt in write: 1 Byte(s): 0x02
Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Write to register: 0x3d Cnt in write: 1 Byte(s): 0x1c
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x02
Power Mode value: 0x02 <---------- Power Mode = SUSPEND

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x02
Write to register: 0x3e Cnt in write: 1 Byte(s): 0x00
Read from register: 0x3e Cnt in read: 1 Byte(s): 0x00
Power Mode value: 0x00 <---------- Power Mode = NORMAL

Read from register: 0x3d Cnt in read: 1 Byte(s): 0x10
Operation Mode value: 0x00 <---------- Operation Mode = CONFIG_MODE (would have expected OPERATION_MODE_NDOF)

########################## Test case end ##################

```
/*!   @brief This API used to write the operation mode
 *    from register from 0x3D bit 0 to 3
 *
 *    @param operation_mode_u8 : The value of operation mode
 *
 *  operation_mode_u8  |      result                        | comments
 * ---------|-----------------------------------|-------------------------
 *  0x00    | BNO055_OPERATION_MODE_CONFIG       | Configuration mode
 *  0x01    | BNO055_OPERATION_MODE_ACCONLY      | Reads accel data alone
 *  0x02    | BNO055_OPERATION_MODE_MAGONLY      | Reads mag data alone
 *  0x03    | BNO055_OPERATION_MODE_GYRONLY      | Reads gyro data alone
 *  0x04    | BNO055_OPERATION_MODE_ACCMAG       | Reads accel and mag data
 *  0x05    | BNO055_OPERATION_MODE_ACCGYRO      | Reads accel and gyro data
 *  0x06    | BNO055_OPERATION_MODE_MAGGYRO      | Reads accel and mag data
 *  0x07    | OPERATION_MODE_ANY_MOTION          | Reads accel mag and
 *                          |          -                                    | gyro data
 *  0x08    | BNO055_OPERATION_MODE_IMUPLUS      | Inertial measurement unit
 *   -      |                                    | Reads accel,gyro and
 *          |       -                            | fusion data
 *  0x09    | BNO055_OPERATION_MODE_COMPASS      | Reads accel, mag data
 *   -      |       -                            | and fusion data
 *  0x0A    | BNO055_OPERATION_MODE_M4G          | Reads accel, mag data
 *    -     |       -                            | and fusion data
 *  0x0B    | BNO055_OPERATION_MODE_NDOF_FMC_OFF | Nine degrees of freedom with
 *   -      |       -                            | fast magnetic calibration
 *   -      |       -                            | Reads accel,mag, gyro
 *   -      |       -                            | and fusion data
 *  0x0C    | BNO055_OPERATION_MODE_NDOF         | Nine degrees of freedom
 *   -      |       -                            | Reads accel,mag, gyro
 *   -      |       -                            | and fusion data
 *
 *    @return results of bus communication function
```

```
 *    @retval 0 -> BNO055_SUCCESS
 *    @retval 1 -> BNO055_ERROR
 *
 *    @note In the config mode, all sensor and fusion data
 *    becomes zero and it is mainly derived
 *    to configure the various settings of the BNO
 *
 */
BNO055_RETURN_FUNCTION_TYPE bno055_set_operation_mode(u8 operation_mode_u8)
{
BNO055_RETURN_FUNCTION_TYPE com_rslt = BNO055_ERROR;
u8 data_u8r = BNO055_INIT_VALUE;
u8 prev_opmode_u8 = BNO055_OPERATION_MODE_CONFIG;
s8 stat_s8 = BNO055_ERROR;
/* Check the struct p_bno055 is empty */
if (p_bno055 == BNO055_INIT_VALUE) {
      return BNO055_E_NULL_PTR;
      } else {
            /* The write operation effective only if the operation
            mode is in config mode, this part of code is checking the
            current operation mode and set the config mode */
            stat_s8 = bno055_get_operation_mode(&prev_opmode_u8);
            if (stat_s8 == BNO055_SUCCESS) {
                  /* If the previous operation mode is config it is
                        directly write the operation mode */
                  if (prev_opmode_u8 == BNO055_OPERATION_MODE_CONFIG) {
                        com_rslt = p_bno055->BNO055_BUS_READ_FUNC
                        (p_bno055->dev_addr,
                        BNO055_OPERATION_MODE_REG,
                        &data_u8r, BNO055_GEN_READ_WRITE_LENGTH);
                        if (com_rslt == BNO055_SUCCESS) {
                              data_u8r =
```

```
                BNO055_SET_BITSLICE(data_u8r,
                BNO055_OPERATION_MODE,
                operation_mode_u8);
                com_rslt +=
                p_bno055->BNO055_BUS_WRITE_FUNC
                (p_bno055->dev_addr,
                BNO055_OPERATION_MODE_REG,
                &data_u8r,
                BNO055_GEN_READ_WRITE_LENGTH);
                /* Config mode to other
                operation mode switching
                required delay of 600ms*/
                p_bno055->delay_msec(
                BNO055_MODE_SWITCHING_DELAY);
        }
} else {
        /* If the previous operation
        mode is not config it is
         write the config mode */
        com_rslt = p_bno055->BNO055_BUS_READ_FUNC
        (p_bno055->dev_addr,
        BNO055_OPERATION_MODE_REG,
        &data_u8r, BNO055_GEN_READ_WRITE_LENGTH);
        if (com_rslt == BNO055_SUCCESS) {
                data_u8r =
                BNO055_SET_BITSLICE(data_u8r,
                BNO055_OPERATION_MODE,
                BNO055_OPERATION_MODE_CONFIG);
                com_rslt += bno055_write_register(
                BNO055_OPERATION_MODE_REG,
                &data_u8r,
                BNO055_GEN_READ_WRITE_LENGTH);
```

```
        /* other mode to config mode switching
        required delay of 20ms*/
        p_bno055->delay_msec(
        BNO055_CONFIG_MODE_SWITCHING_DELAY);
}
/* Write the operation mode */
if (operation_mode_u8 !=
BNO055_OPERATION_MODE_CONFIG) {
        com_rslt =
        p_bno055->BNO055_BUS_READ_FUNC
        (p_bno055->dev_addr,
        BNO055_OPERATION_MODE_REG,
        &data_u8r,
        BNO055_GEN_READ_WRITE_LENGTH);
        if (com_rslt == BNO055_SUCCESS) {
                data_u8r = BNO055_SET_BITSLICE
                (data_u8r,
                BNO055_OPERATION_MODE,
                operation_mode_u8);
                com_rslt +=
                p_bno055->BNO055_BUS_WRITE_FUNC
                (p_bno055->dev_addr,
                BNO055_OPERATION_MODE_REG,
                &data_u8r,
                BNO055_GEN_READ_WRITE_LENGTH);
                /* Config mode to other
                operation mode switching
                required delay of 600ms*/
                p_bno055->delay_msec(
                BNO055_MODE_SWITCHING_DELAY);
        }
}
```

```
            }
        } else {
        com_rslt = BNO055_ERROR;
        }
}
return com_rslt;
```