# MAX21002 USER GUIDE

*Revision 2.5, May 2015*

maxim integrated™

©2015 Maxim Integrated Products, Inc.
All rights reserved.

No part of this documentation may be reproduced nor distributed in any form or by any means, graphic, electronic, or mechanical, including but not limited to photocopying, scanning, recording, taping, e-mailing, or storing in information storage and retrieval systems without the written permission of Maxim Integrated Products, Inc. (hereafter, “Maxim”). Products that are referenced in this document such as Microsoft Windows® may be trademarks and/or registered trademarks of their respective owners. Maxim makes no claim to these trademarks. While every precaution has been taken in the preparation of this document, individually, as a series, in whole, or in part, Maxim, the publisher, and the author assume no responsibility for errors or omissions, including any damages resulting from the express or implied application of information contained in this document or from the use of products, services, or programs that may accompany it. In no event shall Maxim, publishers, authors, or editors of this guide be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Rev. 2.5, May 2015

# CONTENTS

**1 Revision History ..... 7**

**2 Introduction ..... 8**

**3 Nomenclature ..... 9**

**4 MAX21002 Description ..... 10**

**5 Pin Description ..... 11**

**6 $I^2C$ Interface ..... 12**

6.1 $I^2C$ Protocol ..... 13
6.2 Slave Address ..... 13
6.3 Acknowledge ..... 13
6.4 Register Address ..... 14
6.5 $I^2C$ Operations ..... 14
*6.5.1 Write One Byte ..... 14*
*6.5.2 Write a Burst of Data ..... 14*
*6.5.3 Read One Byte ..... 15*
*6.5.4 Read a Burst of Data ..... 16*

**7 SPI Interface ..... 17**

7.1 SPI Protocol ..... 17
7.2 Register Address ..... 18

**8 Interrupts ..... 20**

8.1 Interrupt Flags ..... 20
8.2 Interrupt Lines ..... 20
8.3 Rate Interrupts ..... 20

**9 Reading Data from MAX21002 and FIFO Operation ..... 22**

9.1 Synchronous Reading ..... 22
9.2 Asynchronous Reading ..... 22
9.3 FIFO Modes Description ..... 22
*9.3.1 FIFO OFF Mode ..... 22*
*9.3.2 Normal Mode ..... 22*
*9.3.3 Interrupt Mode ..... 24*
*9.3.4 Snapshot Mode ..... 27*
9.4 Example of FIFO Read/Write Pointers Evolution ..... 27

**10 Programming Example ..... 29**

10.1 Simple Read-Out Sequence, No FIFO, No Interrupts ..... 30
10.2 Simple Read-Out Sequence, FIFO Normal Mode, No Interrupts ..... 31
10.3 Simple Read-Out Sequence, Normal Mode, Data-Ready Interrupts ..... 32

**11 DSYNC ..... 33**

11.1 Power Mode Switching Using DSYNC ..... 33
*11.1.1 Example Register Settings for Power Mode Switching ..... 33*
11.2 Filling FIFO with DSYNC Edge ..... 33
*11.2.1 Example Register Settings for FIFO Filling with DSYNC ..... 34*
11.3 Map DSYNC on LSb ..... 34

*11.3.1 Example Register Settings for Mapping DSYNC level on LSB ..... 34*
11.4 Example of DSYNC Application: Generation of Non-Standard ODR ..... 35

**12 Register Map ..... 36**

12.1 COMMON BANK ..... 36
*12.1.1 WHO_AM_I ..... 37*
*12.1.2 BANK_SELECT ..... 37*
*12.1.3 SYSTEM_STATUS ..... 38*
*12.1.4 GYRO_X_H ..... 38*
*12.1.5 GYRO_X_L ..... 39*
*12.1.6 GYRO_Y_H ..... 39*
*12.1.7 GYRO_Y_L ..... 39*
*12.1.8 TEMP_H ..... 40*
*12.1.9 TEMP_L ..... 40*
*12.1.10 HP_RST ..... 40*
*12.1.11 FIFO_COUNT ..... 41*
*12.1.12 FIFO_STATUS ..... 42*
*12.1.13 FIFO_DATA ..... 43*
*12.1.14 PAR_RST ..... 43*
12.2 USER BANK #0 (*bank_sel* = 0000) ..... 44
*12.2.1 POWER_CFG ..... 45*
*12.2.2 SENSE_CFG1 ..... 46*
*12.2.3 SENSE_CFG2 ..... 47*
*12.2.4 SENSE_CFG3 ..... 48*
*12.2.5 DR_CFG ..... 49*
*12.2.6 IO_CFG ..... 49*
*12.2.7 I2C_CFG ..... 50*
*12.2.8 ITF_OTP ..... 51*
*12.2.9 FIFO_TH ..... 51*
*12.2.10 FIFO_CFG ..... 52*
*12.2.11 DSYNC_CFG ..... 53*
*12.2.12 DSYNC_CNT ..... 53*
12.3 USER BANK #1 (*bank_sel* = 0001) ..... 54
*12.3.1 INT_REF_X ..... 55*
*12.3.2 INT_REF_Y ..... 55*
*12.3.3 INT_DEB_X ..... 55*
*12.3.4 INT_DEB_Y ..... 56*
*12.3.5 INT_MSK_X ..... 56*
*12.3.6 INT_MSK_Y ..... 57*
*12.3.7 INT_MSK_AO ..... 57*
*12.3.8 INT_CFG1 ..... 58*
*12.3.9 INT_CFG2 ..... 58*
*12.3.10 INT_TMO ..... 59*
*12.3.11 INT_STS_UL ..... 59*
*12.3.12 INT1_STS ..... 60*
*12.3.13 INT2_STS ..... 60*
*12.3.14 INT1_MSK ..... 61*
*12.3.15 INT2_MSK ..... 61*
*12.3.16 OTP_STATUS ..... 62*

*12.3.17 SILICON_REV_OTP ..... 62*
*12.3.18 SERIAL_[0:5] ..... 62*

**13 Definitions ..... 63**

# TABLES

Table 1: Pin Description ..... 11
Table 2: I²C External Component Properties ..... 12
Table 3: I²C Device Addresses ..... 13
Table 4: SPI External Component Properties ..... 17
Table 5: Register Settings for Switching Power Mode ..... 33
Table 6: Register Settings for FIFO Filling with DSYNC ..... 34
Table 7: Register Settings for Mapping LSB ..... 34
Table 8: Common Bank ..... 36
Table 9: User Bank 0 ..... 44
Table 10: Power Mode Configuration ..... 45
Table 11: Bandwidth Configuration ..... 46
Table 12: HPF Cut-Off Frequencies ..... 48
Table 13: User Bank 1 ..... 54

# FIGURES

Figure 1: Block Diagram ..... 10
Figure 2: I²C Interface Connection to an Application Processor ..... 12
Figure 3: START (S), STOP (P), and Repeated START (Sr) Conditions ..... 13
Figure 4: I²C Write One Byte ..... 14
Figure 5: I²C Write a Burst of Data ..... 15
Figure 6: I²C Read One Byte ..... 15
Figure 7: I²C Read a Burst of Data ..... 16
Figure 8: SPI Interface Connection to an Application Processor ..... 17
Figure 9: SPI Protocol ..... 18
Figure 10: Conditional Ranges ..... 21
Figure 11: FIFO Normal Mode, Stop on Full ..... 23
Figure 12: FIFO Normal Mode, Overwrite ..... 24
Figure 13: FIFO Interrupt Mode, Stop on Full ..... 25
Figure 14: FIFO Interrupt Mode, Overwrite ..... 26
Figure 15: FIFO Snapshot Mode ..... 27
Figure 16: FIFO Read/Write Pointer Evolution ..... 28
Figure 17: Simple Read-Out Sequence, No FIFO, No Interrupts ..... 30
Figure 18: Simple Read-Out Sequence, FIFO Normal, No Interrupts ..... 31
Figure 19: Simple Read-Out Sequence, Normal Mode, w/ Data Ready Interrupts and No FIFO ..... 32

Figure 20: Timing Diagram to Generate a Nonstandard ODR .....35
Figure 21: FIFO Flags .....42

# 1 Revision History

| Revision | Date | Description |
|---|---|---|
| 1.0 | 12/12/2012 | Inital Release |
| 2.0 | 3/4/2015 | Initial release with new format |
| 2.1 | 3/5/2015 | First review with new format |
| 2.2 | 3/6/2015 | Misspellings and mis-references are fixed |
| 2.3 | 3/6/2015 | Registers are renamed |
| 2.4 | 4/13/2015 | DSYNC section is added<br>Bit fields are updated<br>Detailed endian information is included<br>Detailed IF_Parity information is included<br>Added new programming examples |
| 2.5 | 5/27/2015 | Re-fomatting is done<br>Broken hyperlinks are fixed<br>Register definitions are updated |

# 2 Introduction

MEMS sensors are revolutionizing the way people interact with everyday technology, making it easier and more user-friendly. Maxim can leverage its analog integration expertise to develop and manufacture new breakthrough MEMS sensors being smaller, lower power and more accurate than ever.

Owning the entire supply chain, Maxim brings its customers complete, reliable and cost-effective solutions, ensuring prompt time-to-volume and time-to-market to effectively address high-volume applications in consumer and industrial market segments.

Thanks to its leadership in analog integration and its manufacturing experience in MEMS, Maxim is capable of high-volume production to meet the market's demands. Maxim’s manufacturing expertise and highest quality standards also guarantee high performance and product reliability.

Every MEMS sensor is tested and trimmed in factory so that for most consumer applications, no additional sensor calibrations are required. The end user can quickly verify the sensor’s operation without physically tilting or rotating the sensor thanks to the built-in self-test feature, which allows accelerating the time-to-market for mass production.

This User Guide will provide a clear picture of the guidelines for its use in consumer applications and a comprehensive description of his unique features. The final section of this guide will present the structure of the register file, the purpose of each field or every register, including two examples about typical programming sequences.

# 3 Nomenclature

| | |
|---|---|
| ODR | Output Data Rate |
| BW | Bandwidth |
| FS | Fulllscale |
| UI | User Interface |
| OIS | Optical Image Stabilization |
| MSB | Most Significant Bit or Byte |
| LSB | Least Significant Bit or Byte |
| HPF | Highpass Filter |
| LPF | Lowpass Filter |
| dps | Degrees per seconds |
| RFU | Reserved for future uses |

# 4 MAX21002 Description

The MAX21002 is a low-power, low-noise, two-axis (XY) angular rate sensor able to offer unprecedented accuracy and sensitivity over temperature and time.

It is capable of working with a supply voltage as low as 1.71V for minimum power consumption. It includes a sensing element and an IC interface capable of providing the measured angular rate to the external world ($I^2C$/SPI).

The MAX21002 has a configurable full scale of ±31.25/±62.5/±125/±250/±500/±1000dps and is capable of measuring rates with a finely tunable user-selectable bandwidth. The high output data rate (ODR) and the large bandwidth (BW), together with the low phase delay, make the MAX21002 suitable for both user interface (UI) and optical image stabilization (OIS) applications.

The MAX21002 is a highly integrated solution requiring only two external capacitors, available in a compact 3mm x 3mm x 0.9mm plastic land grid array (LGA) package and can operate within a temperature range of -40°C to +85°C.

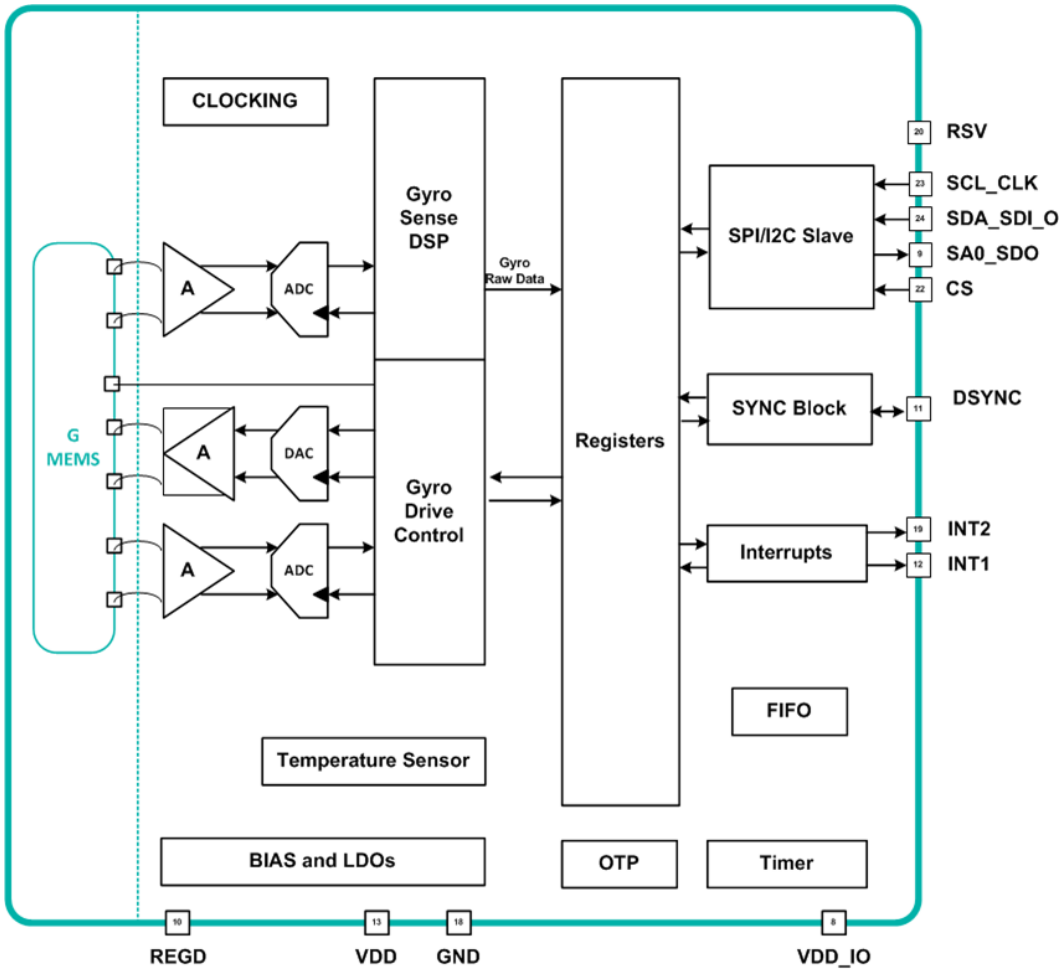



***Figure 1: Block Diagram***

# 5 Pin Description

***Table 1: Pin Description***

| PIN | NAME | FUNCTION |
|---|---|---|
| 1 | $V_{DDIO}$ | Interface and Interrupt Pad Supply Voltage. Same range of $V_{DD}$. $V_{DDIO} \leq V_{DD}$ (diode). |
| 2 | N.C. | Not Internally Connected |
| 3 | N.C. | Not Internally Connected |
| 4 | SCL_CLK | SPI and $I^2C$ Clock. When in $I^2C$ mode, the IO has selectable anti-spike filter and delay to ensure correct hold time. |
| 5 | GND | Power-Supply Ground |
| 6 | SDA_SDI_O | SPI In/Out Pin and $I^2C$ Serial Data. When in $I^2C$ mode, the IO has selectable anti-spike filter and delay to ensure correct hold time. |
| 7 | SA0_SDO | SPI Serial Data Out or $I^2C$ Slave Address LSB |
| 8 | CS | SPI Chip Select/Serial Interface Selection |
| 9 | INT2 | Interrupt Line #2 |
| 10 | RESERVED | Must be connected to GND |
| 11 | INT1 | Interrupt Line #1 |
| 12 | DSYNC | Data Synchronization Pin to wake up MAX21002 from power down/standby or to synchronize data with an external device. |
| 13 | RESERVED | Leave unconnected |
| 14 | $V_{DD}$ | Analog Power Supply Pin: Bypass to GND with a 0.1µF capacitor and one 10µF capacitor. |
| 15 | $V_{DD}$ | Must be tied to $V_{DD}$ in the application. |
| 16 | N.C. | Not Internally Connected |

# 6 I²C Interface

To connect a MAX21002 device to an I²C master, the SDA_SDI_O pin of the MAX21002 device must be connected to the SDA pin of the I²C master and the SCL_CLK pin of the MAX21002 device must be connected to the SCL pin of the I²C master. Both SDA and SCL lines must be connected to a pullup resistor. The SA0_SDO pin must be connected to VDD or GND to configure the MAX21002 I²C slave address (see Table 3: I²C Device Addresses).

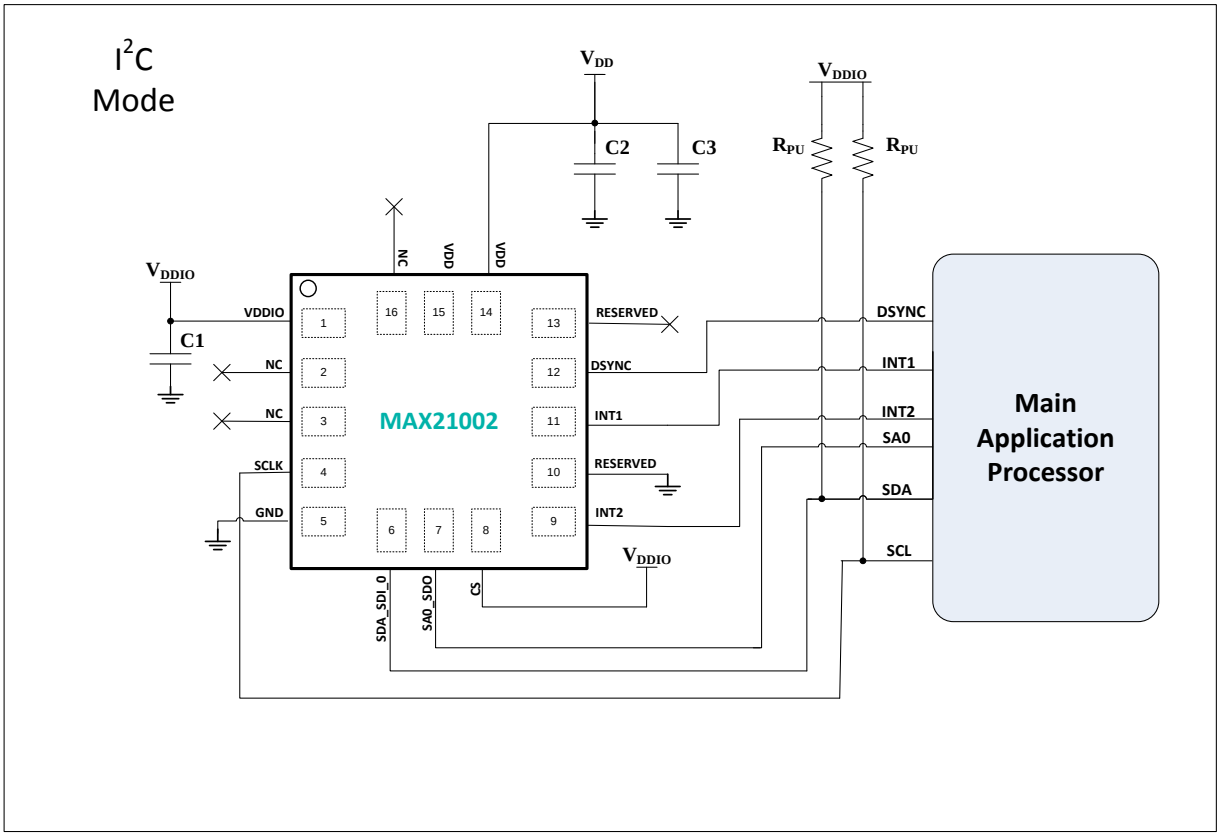



***Figure 2: I²C Interface Connection to an Application Processor***

***Table 2: I²C External Component Properties***

| COMPONENT | LABEL | SPECIFICATION | QUANTITY |
|---|---|---|---|
| $V_{DDIO}$ /$V_{DD}$ Bypass Capacitor | C1,C2 | Ceramic, X7R, 100nF ±10%, 4V | 2 |
| $V_{DD}$ Bypass Capacitor | C3 | Ceramic, X7R, 1uF ±10%, 4V | 1 |
| Pullup Resistor (I²C Mode only) | $R_{PU}$ | 1.1kΩ - 10kΩ (min - max) | 2 |

## 6.1 I²C Protocol

To start an I²C request, the master sends a START condition (S), followed by the MAX21002's I²C address. Then, the master sends the address of the register to be programmed. The master then terminates the communication by issuing a STOP condition (P) to relinquish the control of the bus, or a repeated START condition (Sr) to keep controlling it.

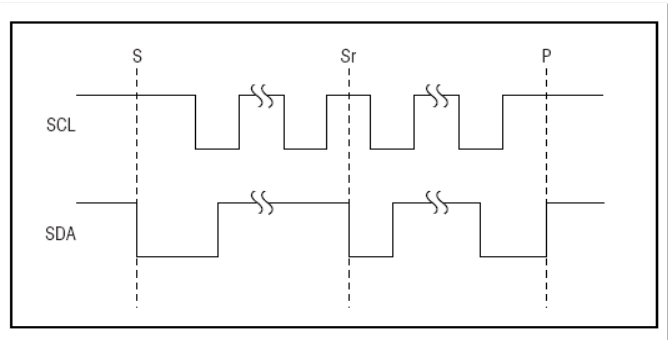



***Figure 3: START (S), STOP (P), and Repeated START (Sr) Conditions***

## 6.2 Slave Address

The slave address is used to identify the MAX21002 device in I²C communications. The address is defined as the seven most significant bits followed by the read/write bit. Set the read/write bit to 1 to request a read operation, or 0 to request a write operation (see Table 3).

***Table 3: I²C Device Addresses***

| I²C Base Address | SA0_SDO pin | R/W bit | Resulting Address |
|---|---|---|---|
| 0x2C (6bit) | 0 | 0 | 0xB0 |
| 0x2C | 0 | 1 | 0xB1 |
| 0x2C | 1 | 0 | 0xB2 |
| 0x2C | 1 | 1 | 0xB3 |

## 6.3 Acknowledge

The acknowledge bit is sent after every byte. This bit allows the receiver to notify the transmitter that the byte has been received correctly and another byte may be sent. The master generates all clock pulses, including the acknowledge's ninth clock pulse (8 bits of data + ACK).

To allow the receiver to send the acknowledge, the transmitter releases the SDA line during the acknowledge pulse, so that the receiver can pull the SDA line LOW during the ninth clock pulse to signal an Acknowledge (ACK), or release the SDA line (HIGH) to signal a Not Acknowledge (NACK).

The NACK is sent if the device is busy or a system fault occurs. It is also used by the master to signal the end of the transfer during a read operation.

## 6.4 Register Address

The I²C register address for the MAX21002 is composed of 6 bits of address and 1 bit (if parity) whose meaning can be configured as:

| | |
|---|---|
| ***Auto-increment:*** | if 0, in case of burst operation the initial register address is auto-incremented after every data byte; if 1, the operation is executed always on the same register; |
| ***Even parity*** | this bit represents the even parity computed on the 6 bits of the register address; |
| ***Odd Parity*** | this bit represents the odd parity computed on the 6 bits of the register address; |

## 6.5 I²C Operations

### 6.5.1 Write One Byte

To write one byte, the following steps must be executed:

1: The master sends a START condition.
2: The master sends the 7 bits slave ID plus a write bit (low)
3: The addressed slave asserts an ACK on the data line.
4: The master sends 8 bits of the Register Address.
5: The slave asserts an ACK on the data line ***only if the address is valid (NACK if not)***.
6: The master sends 8 bits of data.
7: The slave asserts an ACK on the data line.
8: The master generates a STOP condition.

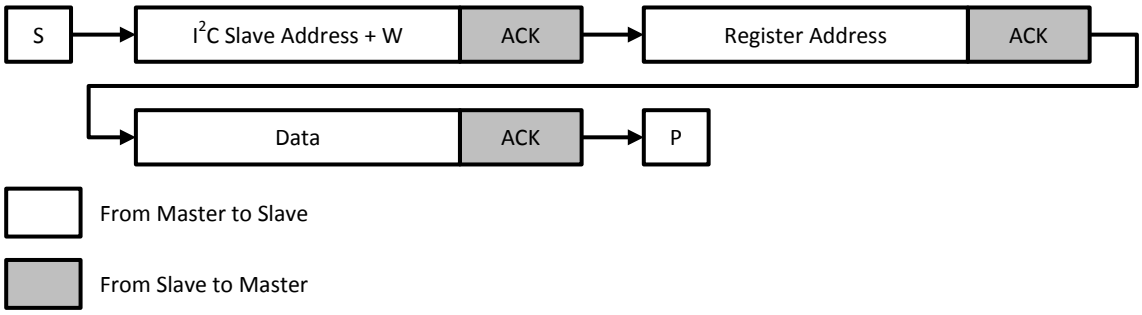



***Figure 4: I²C Write One Byte***

### 6.5.2 Write a Burst of Data

To execute a write of a burst of data, the following steps must be executed:

1: The master sends a START condition.
2: The master sends the 7 bits slave ID plus a write bit (low).
3: The addressed slave asserts an ACK on the data line.
4: The master sends 8 bits of the Register Address.
5: The slave asserts an ACK on the data line ***only if the address is valid (NACK if not)***.
6: The master sends 8 bits of data.
7: The slave asserts an ACK on the data line.
8: Repeat 6 and 7 as long as needed.

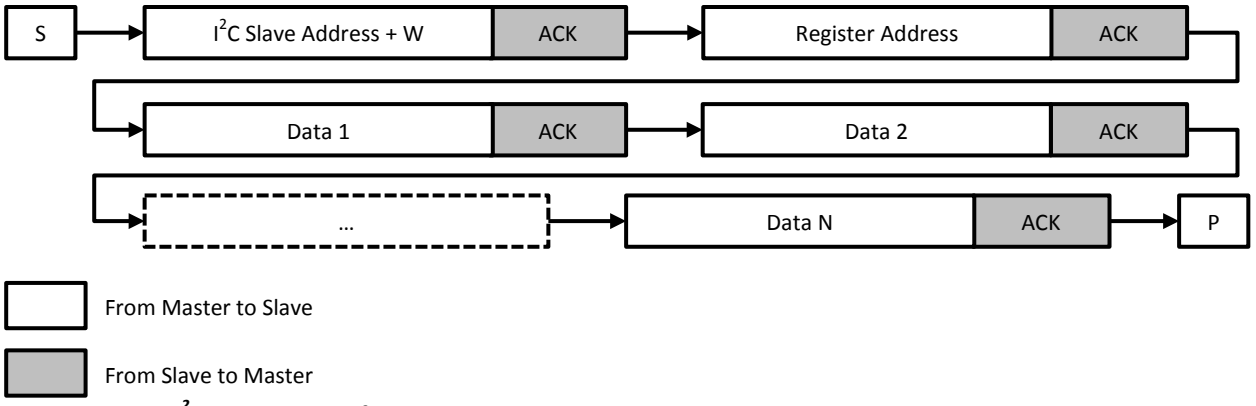



***Figure 5: I²C Write a Burst of Data***

### 6.5.3 Read One Byte

To read one byte, the following steps must be executed:

1: The master sends a START condition.
2: The master sends the 7 bits slave ID plus a write bit (low).
3: The addressed slave asserts an ACK on the data line.
4: The master sends 8 data bits.
5: The active slave asserts an ACK on the data line ***only if the address is valid (NACK if not)***.
6: The master sends a restart condition.
7: The master sends the 7 bits slave ID plus a read bit (high).
8: The addressed slave asserts an ACK on the data line.
9: The slave sends 8 data bits.
10: The master asserts a NACK on the data line.
11: The master generates a STOP condition.

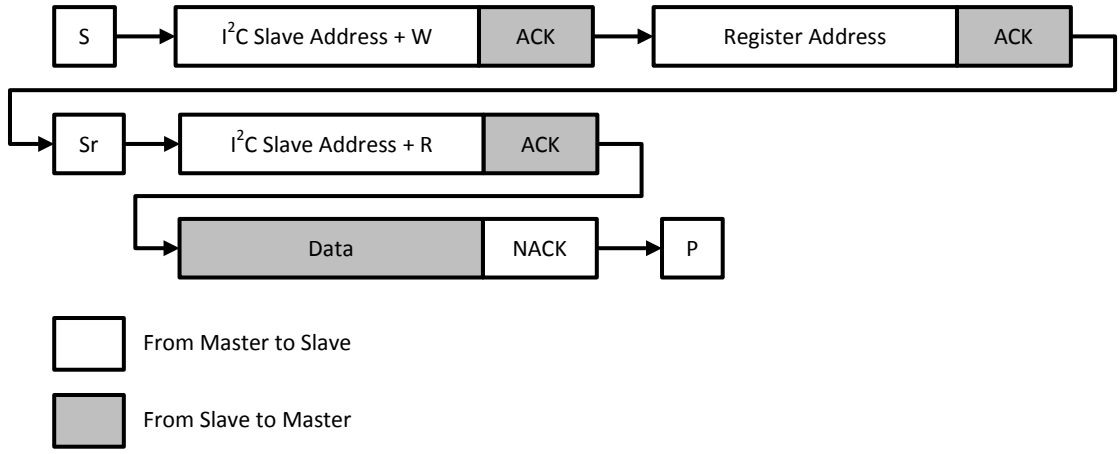



***Figure 6: I²C Read One Byte***

### 6.5.4 Read a Burst of Data

To execute a read of a burst of data, the following steps must be executed:

1: The master sends a START condition.
2: The master sends the 7 bits slave ID plus a write bit (low).
3: The addressed slave asserts an ACK on the data line.
4: The master sends 8 bits of the Register Address.
5: The slave asserts an ACK on the data line **only if the address is valid (NACK if not)**.
6: The master sends a repeated START condition.
7: The master sends the 7 bits slave ID plus a read bit (high).
8: The slave asserts an ACK on the data line.
9: The slave sends 8 bits of data.
10: The master asserts an ACK on the data line.
11: Repeat 9 and 10 as long as needed.
12: The master generates a STOP condition.

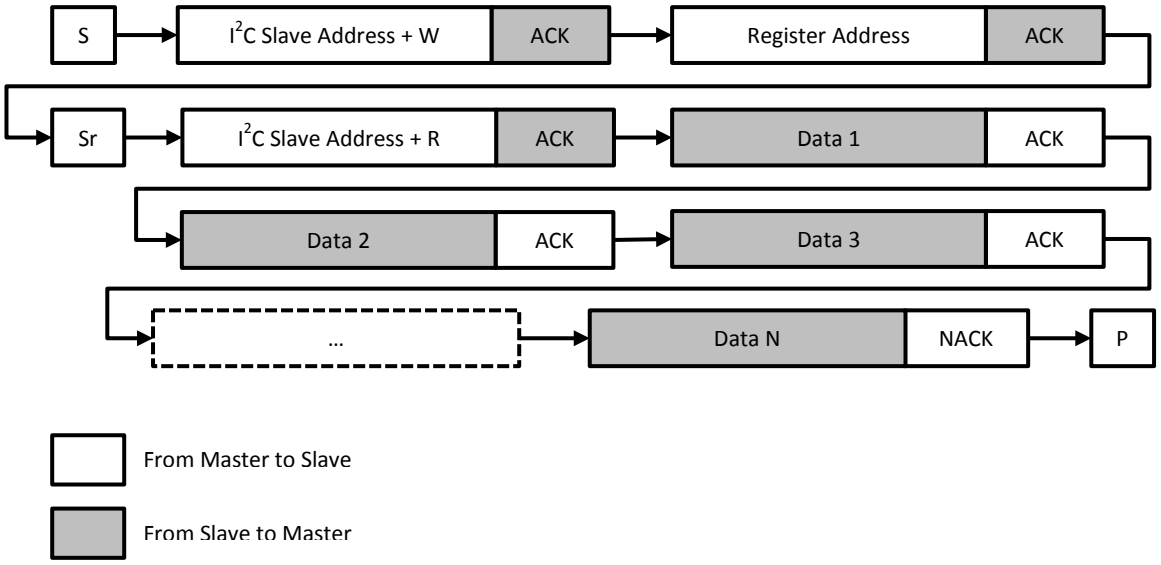



***Figure 7: I²C Read a Burst of Data***

# 7 SPI Interface

To connect a MAX21002 device to an SPI master, the CS pin of the MAX21002 device must be connected to the CSn pin of the SPI master, the SDA_SDI_O pin of the MAX21002 device must be connected to the MOSI pin of the SPI master, the SA0_SDO pin of the MAX21002 device must be connected to the MISO pin of the SPI master and the SCL_CLK pin of the MAX21002 device must be connected to the SCLK pin of the SPI master. For external component parameters, refer to Table 4: SPI External Component Properties.

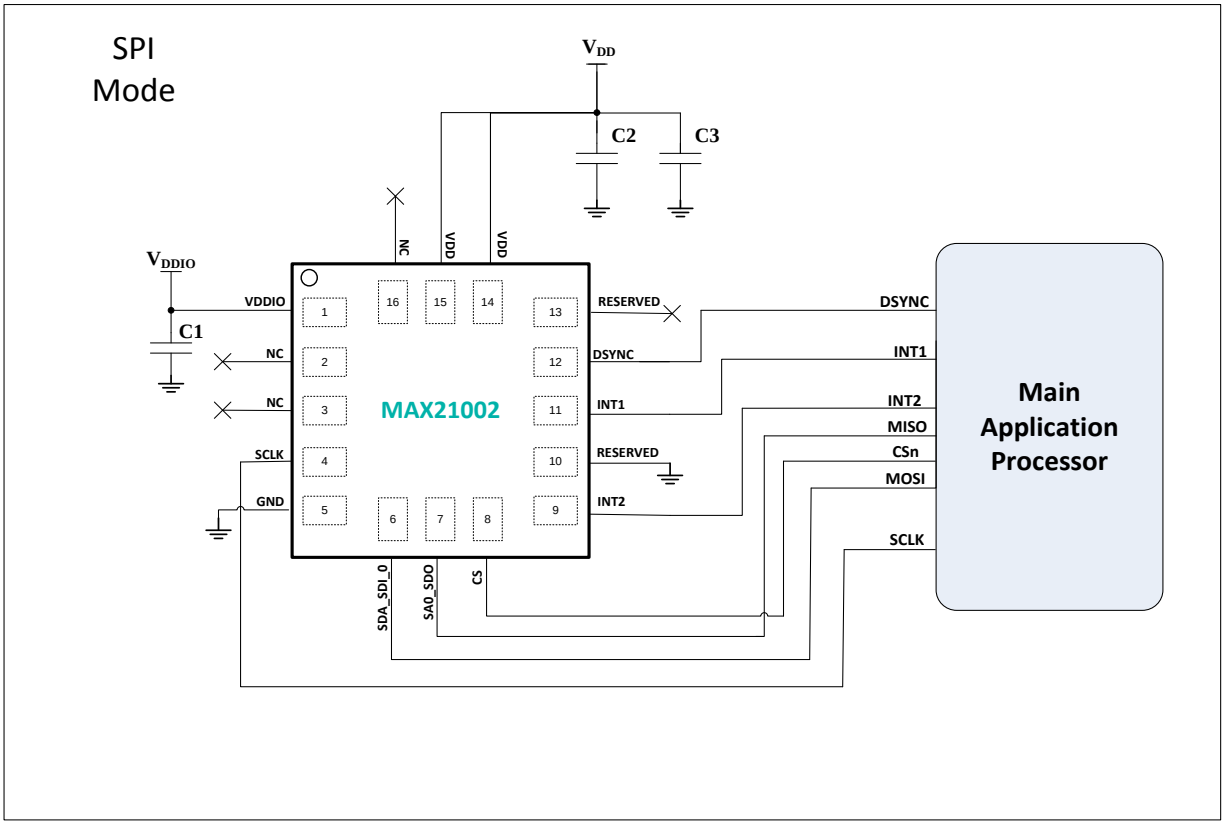



***Figure 8: SPI Interface Connection to an Application Processor***

***Table 4: SPI External Component Properties***

| COMPONENT | LABEL | SPECIFICATION | QUANTITY |
|---|---|---|---|
| $V_{DDIO}$ /$V_{DD}$ Bypass Capacitor | C1,C2 | Ceramic, X7R, 100nF ±10%, 4V | 2 |
| $V_{DD}$ Bypass Capacitor | C3 | Ceramic, X7R, 1µF ±10%, 4V | 1 |

## 7.1 SPI Protocol

CSn is the serial port enable and is controlled by the SPI master. It goes low at the start of the transmission and returns to high at the end.

SCLK is the serial port clock and is controlled by the SPI master. It is kept high when CSn is high (no transmission). MISO and MOSI are, respectively, the serial port data input and output. These lines are driven at the falling edge of SCLK and are sampled at the rising edge of SCLK.

Both the read register and write register commands are completed in 16 clock pulses, or in multiples of 8 in case of multiple read/write bytes. Bit duration is the time between two falling edges of SCLK.

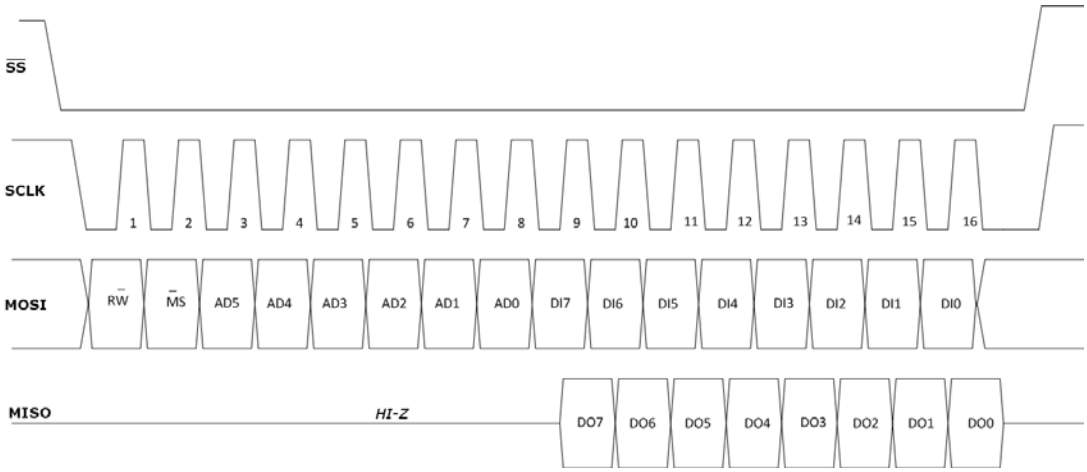



***Figure 9: SPI Protocol***

## 7.2 Register Address

The SPI register address for the MAX21002 is composed by 6 bits of address, 1 bit to select the direction of the operation (Read/Write) and 1 bit whose meaning can be configured as:

| | |
|---|---|
| ***Auto-increment:*** | if 0, in case of burst operation the initial register address is auto-incremented after every data byte; if 1, the operation is executed always on the same register; |
| ***Even parity*** | this bit represents the even parity computed on the 6 bits of the register address; |
| ***Odd Parity*** | this bit represents the odd parity computed on the 6 bits of the register address; |

parity_error and if_parity are used to manage the parity bit during the SPI communication. The parity bit is an additional bit added to the end of a digital word and it indicates whether the number of bits in the word with the value one is even or odd. Parity bit is used to verify if there was a communication error. According to the datasheet, during a SPI communication, the first byte is the register address you want to read/write:

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| R/W | MS/Parity | A5 | A4 | A3 | A2 | A1 | A0 |

| | |
|---|---|
| ***Bit 7:*** | Is used to define if you want to read or write a register. 0 = write, 1 = read; |
| ***Bit 6:*** | Can have 2 different functionalities, in according with the if_parity (see below) |
| ***Bit [5:0]*** | Address of the register you want to read or write; |
| ***If_parity = '00':*** | Bit 6 is used to set the **multi-addressing standard mode** (MS).<br>MS = 0 -> the address is auto incremented in multiple read/write command<br>MS = 1 -> the address remains unchanged in multiple read/write commands; |
| ***if_parity = '01':*** | Bit 6 is used to check the **even parity** with the register address (A[5:0]); |
| ***if_parity = '10':*** | Bit 6 is used to check the **odd parity** with the register address (A[5:0]); |

Here are shown some examples:

| Address | count of 1 bits | 8 bits including parity | |
|---|---|---|---|
| | | Even | Odd |
| 000000 (0x00) | 0 | x0000000 | x1000000 |
| 100000 (0x20) | 1 | x1100000 | x0100000 |
| 100011 (0x23) | 3 | x1100011 | x0100011 |
| 111111 (0x3F) | 6 | x0111111 | x1111111 |

**Bit 7 – indicated with x, can be 0 or 1, depending on if you want to write or read the correspondent register**

For further explanation:

- to **write** register **0x00**, using **odd parity,** you have to send from your MCU the byte '**01000000**' (0x40);
- to **read** register **0x20**, using **even parity,** you have to send from your MCU the byte '**11100000**' (0xE0);
- to **read** register **0x23**, using **odd parity,** you have to send from your MCU the byte '**10100011**' (0xA3);
- to **read** register **0x3F**, using **even parity,** you have to send from your MCU the byte '**10111111**' (0xBF);

When the device receives the above bytes from the MCU, it will try to calculate the parity bit, in according with the rule set in the ***if_parity*** field. If the parity calculated by the device is the same of the parity bit, the communication was good and the content of ***parity_error*** bit will be 0. If instead the parity calculated by the device is different from the parity bit, the content of ***parity_error*** bit is set to '1', reporting that there was an error in the serial communication.

# 8 Interrupts

The MAX21002 is equipped with an interrupt module to control a set of interrupt flags and two interrupt lines (INT1 and INT2).

This module allows to:

1: Configure the behavior of the interrupt lines (INT1 and INT2)
2: Map each interrupt flag to one of both the interrupt lines
3: Create a conditional interrupt (Rate interrupt) based on four different thresholds

## 8.1 Interrupt Flags

The interrupt module provides several interrupt flags in the INT_STS_UL register. Each flag reports the occurrence of a significant event inside the device.

## 8.2 Interrupt Lines

An interrupt line is a dedicated pin where a notification to an external application processor can be provided. The MAX21002 is equipped with two interrupt lines that can be configured independently.

For each interrupt line, it is possible to (refer to INT_CFG2):

- Enable/disable them
- Set the active level to low
- Set the output type to push-pull or open drain configuration

To map an interrupt flag to an interrupt line it is necessary to enable it through the INT1_MSK for INT1 and INT2_MSK for INT2: by setting its corresponding bit to 1 on the bit-mask, the interrupt flag is mapped to the related interrupt line. The output of the two INT1 and INT2 interrupt lines is then computed by applying the OR operator to all the enabled interrupt flags contained in the INT1_STS and INT2_STS registers, respectively. Please note that the enable is not applied to the **INT1_STS** and **INT2_STS** registers, so those registers contain also the value of the interrupt flags that are not enabled.

An interrupt line can be configured in order to keep the status until the master requests to clear it (latched) or after a timeout. Those modes can be selected through the ***int1_latch_mode*** and ***int2_latch_mode*** register fields (INT_TMO). The only exception is the ***gyro_dr*** flag that is always unlatched regardless of the ***int1_latch_mode*** and ***int2_latch_mode*** register fields setting.

The lines can be configured also to auto-clear its status after a period of time where the duration to clear the interrupt is programmable (see Timed Mode at INT_TMO).

## 8.3 Rate Interrupts

The *rate interrupt* allows generating an interrupt event when the gyroscope data falls inside a range of values defined by a set of thresholds. By setting the absolute value of the threshold (|TH|) for each gyroscope axis, four ranges are defined:




***Figure 10: Conditional Ranges***

For each range, it is possible to select if it contributes to the generation of the rate interrupt for each axis. Then, the rate interrupts are computed as follow:

- **int_and:** Active if the defined conditions are satisfied for all the gyroscope axes at the same time;
- **int_or:** Active if the defined conditions are satisfied for at least one of the gyroscope axes.

The threshold absolute value can be set by writing the **INT_REF_X** and **INT_REF_Y** registers. Those registers represent the most significant byte of the real threshold; so, to compute the actual absolute value of the threshold, the register value must be multiplied by 256. If the application requires a better resolution, it is possible to specify a 16-bit threshold by setting the int_single_ref register field; in this case, all the axes share the same threshold that is defined as the combination of INT_REF_X and INT_REF_Y where the first one is the most significant byte and the last one is the least significant byte of the threshold.

Once the thresholds are defined, the INT_MSK_X and INT_MSK_Y permits to select which range contributes to the generation of the rate interrupts for each *axis* as follow:

- **int_*axis*_high_pos_en:** if enabled, the condition is satisfied if the data of the *axis* falls in the *high_pos* range;
- **int_*axis*_high_neg_en:** if enabled, the condition is satisfied if the data of the *axis* falls in the *high_neg* range;
- **int_*axis*_low_pos_en:** if enabled, the condition is satisfied if the data of the *axis* falls in the *low_pos* range;
- **int_*axis*_low_neg_en:** if enabled, the condition is satisfied if the data of the *axis* falls in the *low_neg* range.

Then, the desired axes must be enabled by setting to 1 the corresponding bit of the int_mask_xyx_and for the **int_and** and int_mask_xyz_or for the **int_or**.

Through the INT_MSK_X and INT_MSK_Y registers, it is also possible to read the current value of the conditions, regardless of the content of the int_mask_xyz_or and int_mask_xyz_and register fields. Each of these values can be configured as latched by setting the int_freeze register field. For the rate interrupts, it is possible also to define a de-bounce value by defining the number of samples the axis data has to satisfy the condition before asserting the corresponding interrupt. Those values can be set in the INT_DEB_X and INT_DEB_Y registers. If the required value is greater than 255, it is possible to define a 16-bit debounce value by setting int_single_deb register field; in this case, all the conditions share the same de-bounce value that is defined as the combination of INT_DEB_X and INT_DEB_Y, where the first one is the most significant byte and the last one is the least significant byte of the debounce value.

# 9 Reading Data from MAX21002 and FIFO Operation

MAX21002 sensor output data can be read by the processor in two different mechanisms: synchronous (DATA_READY interrupt) and asynchronous (polling) reading methods.

## 9.1 Synchronous Reading

In this mode, the processor reads the data set (e.g. 6 bytes for a 2 axes configuration) generated by the MAX21002 every time that gyro_dr flag is set. The processor must read the output data only once the gyro_dr flag is set in order to avoid data inconsistencies. Benefits of using this approach include the perfect reconstruction of the signal coming from the sensors and minimization of the data traffic at the interface.

## 9.2 Asynchronous Reading

In this mode, the processor reads the data generated by the MAX21002 regardless the status of the gyro_dr flag. To minimize the error caused by different samples being read a different number of times, the access frequency to be used must be higher than the selected ODR.

## 9.3 FIFO Modes Description

The MAX21002 also embeds a 256-slot of a 16-bit data FIFO for each of the two output channel; X and Y. This allows a consistent power saving for the system since the host processor does not need to continuously poll data from the sensor, but it can wake up only when needed and burst the data out from the FIFO. When configured in Snapshot mode, it offers the ideal mechanism to capture the data following a Rate Interrupt event. The data order in FIFO depends on the endian setting (endian):

**Big Endian:** *GYRO_X_H, GYRO_X_L, GYRO_Y_H, GYRO_Y_L*
**Little Endian:** *GYRO_X_L, GYRO_X_H, GYRO_Y_L, GYRO_Y_H*

The FIFO buffer can work according to four main different modes (see FIFO_CFG): off, normal, interrupt and snapshot. Both normal and Interrupt modes can be optionally configured to operate in overrun Mode, depending on whether, in case of buffer under-run, newer or older data are lost.

### 9.3.1 FIFO OFF Mode

In this mode, the FIFO is turned off; data are stored only in the data registers. No data are available from FIFO if read. When the FIFO is turned OFF, there are two options to use the device: synchronous/asynchronous reading.

### 9.3.2 Normal Mode

The behavior of the FIFO in Normal mode varies depending on the **Overrun** settings (***fifo_overrun*** register field). The following paragraphs show a description of the behavior for both settings of the overrun. For the next sections, the following descriptions are useful for the user's understanding of FIFO:

**Write Pointer (WP):** *Write pointer is the address number where the next data will be written in FIFO, and whenever there is a new data is written, the write pointer is incremented by 1,*
**Read Pointer (RP):** *Read pointer is the address number from where the first data in FIFO is read, and whenever there is a new data is read, the read pointer is incremented by 1,*
**Level** *Level tells the difference between Write pointer and Read pointer, which also gives you the total number of data available in FIFO*

#### 9.3.2.1 Stop on Full

- FIFO is turned on.
- FIFO is filled with the data at the selected Output Data Rate (ODR).
- When FIFO is full, an interrupt can be generated.
- When FIFO is full, all the new incoming data will be discharged. Reading only a subset of the data already stored into the FIFO keeps locked the possibility for new data to be written.
- Only if all the data are read the FIFO will restart saving data. If the communication speed is high, data loss can be prevented.
- In order to prevent a FIFO full condition, the required condition is to complete the reading of the data set before the next DATA_READY occurs.
- If this condition is not guaranteed, data can be lost.

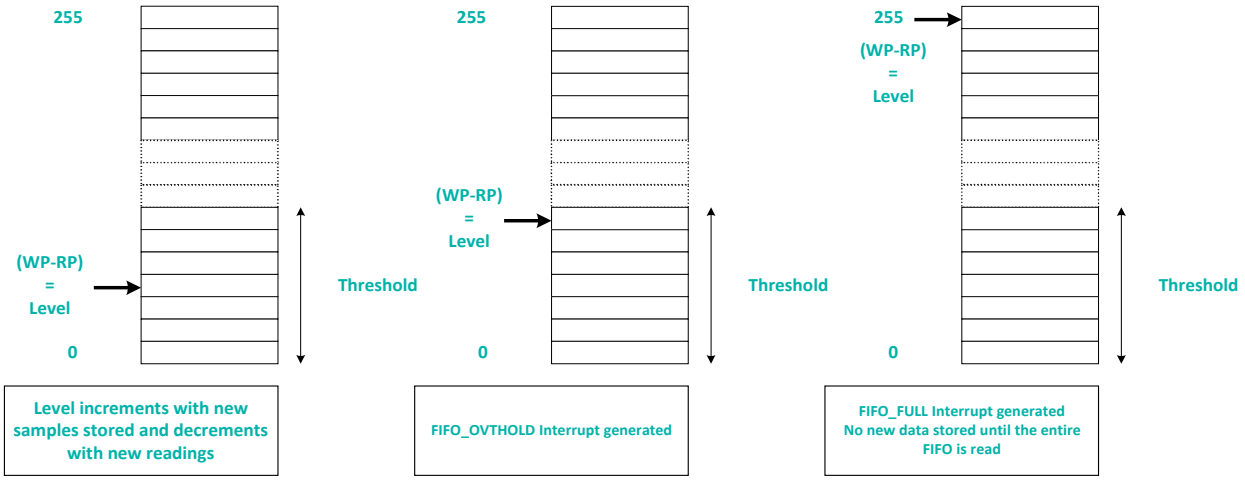



*Figure 11: FIFO Normal Mode, Stop on Full*

#### 9.3.2.2 Overwrite

- FIFO is turned on.
- FIFO is filled with the data at the selected ODR.
- When FIFO is full, an interrupt can be generated.
- When FIFO is full, the oldest data will be overwritten with the new ones.
- If communication speed is high, data integrity can be preserved.
- In order to prevent a data lost condition, the requirement is to complete the reading of the data set before the next DATA_READY occurs.
- If this condition is not guaranteed, data can be overwritten.
- When an overrun condition occurs the Reading pointer is forced to Writing Pointer -1, to make sure only older data are discarded and newer data have a chance to be read.

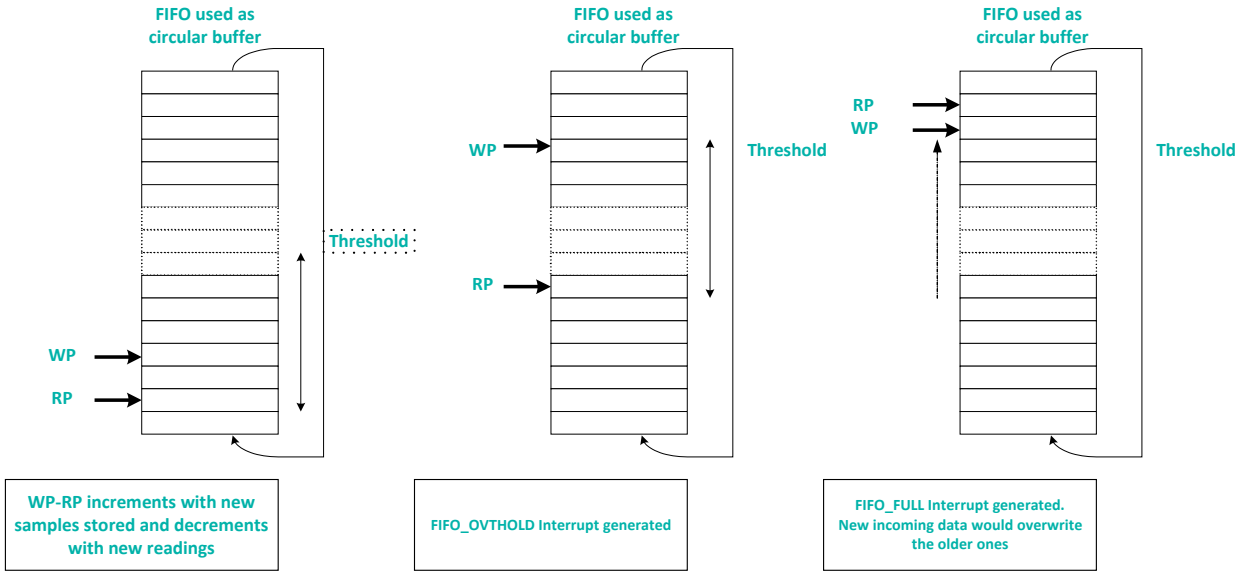



***Figure 12: FIFO Normal Mode, Overwrite***

### 9.3.3 Interrupt Mode

The behavior of the FIFO in Interrupt mode varies depending on the **Overrun** settings (***fifo_overrun*** register field). The following paragraphs show a description of the behavior for both settings of the overrun.

#### 9.3.3.1 Stop on Full

- FIFO is initially disabled. Data is stored only in the data registers.
- When a Rate Interrupt (either OR or AND) is generated, the FIFO is turned on automatically. It stores the data at the selected ODR. Rate interrupt are documented starting from INT_REF_X.
- When FIFO is full, all the new incoming data will be discharged. Reading only a subset of the data already stored into the FIFO keeps locked the possibility for new data to be written.
- Only if all the data are read the FIFO will restart saving data.
- If communication speed is high, data loss can be prevented.
- In order to prevent a FIFO full condition, the required condition is to complete the reading of the data set before the next DATA_READY occurs.
- If this condition is not guaranteed, data can be lost.

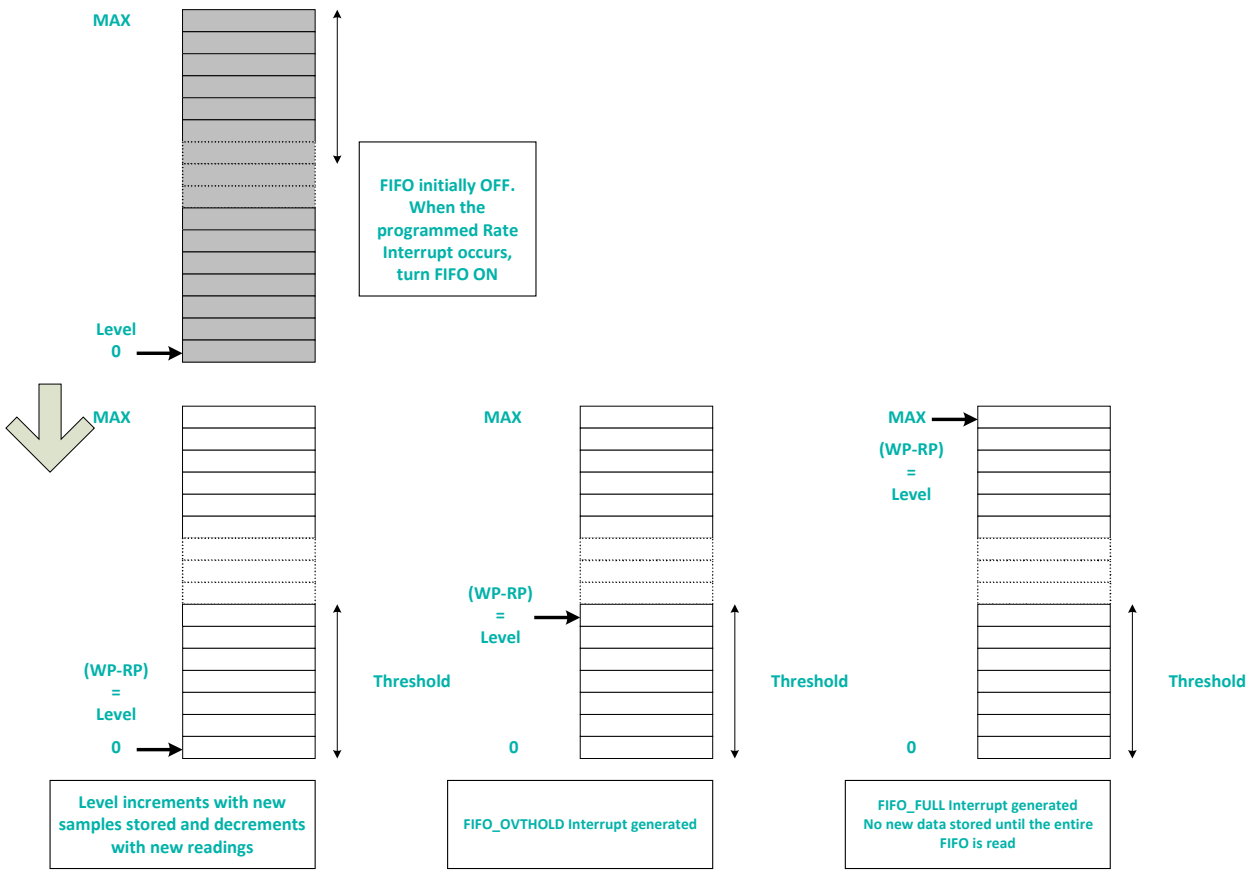



***Figure 13: FIFO Interrupt Mode, Stop on Full***

### 9.3.3.2 Overwrite

- FIFO is initially disabled. Data is stored only in the data registers.
- When a Rate Interrupt (either OR or AND) is generated, the FIFO is turned on automatically. It stores the data at the selected ODR.
- When FIFO is full, an interrupt can be generated.
- When FIFO is full, the oldest data will be overwritten with the new ones.
- If communication speed is high, data integrity can be preserved.
- In order to prevent a data lost condition, the required condition is to complete the reading of the data set before the next DATA_READY occurs.
- If this condition is not guaranteed, data can be overwritten.
- When an overrun condition occurs the Reading pointer is forced to Writing Pointer -1, to make sure only older data are discarded and newer data have a chance to be read.

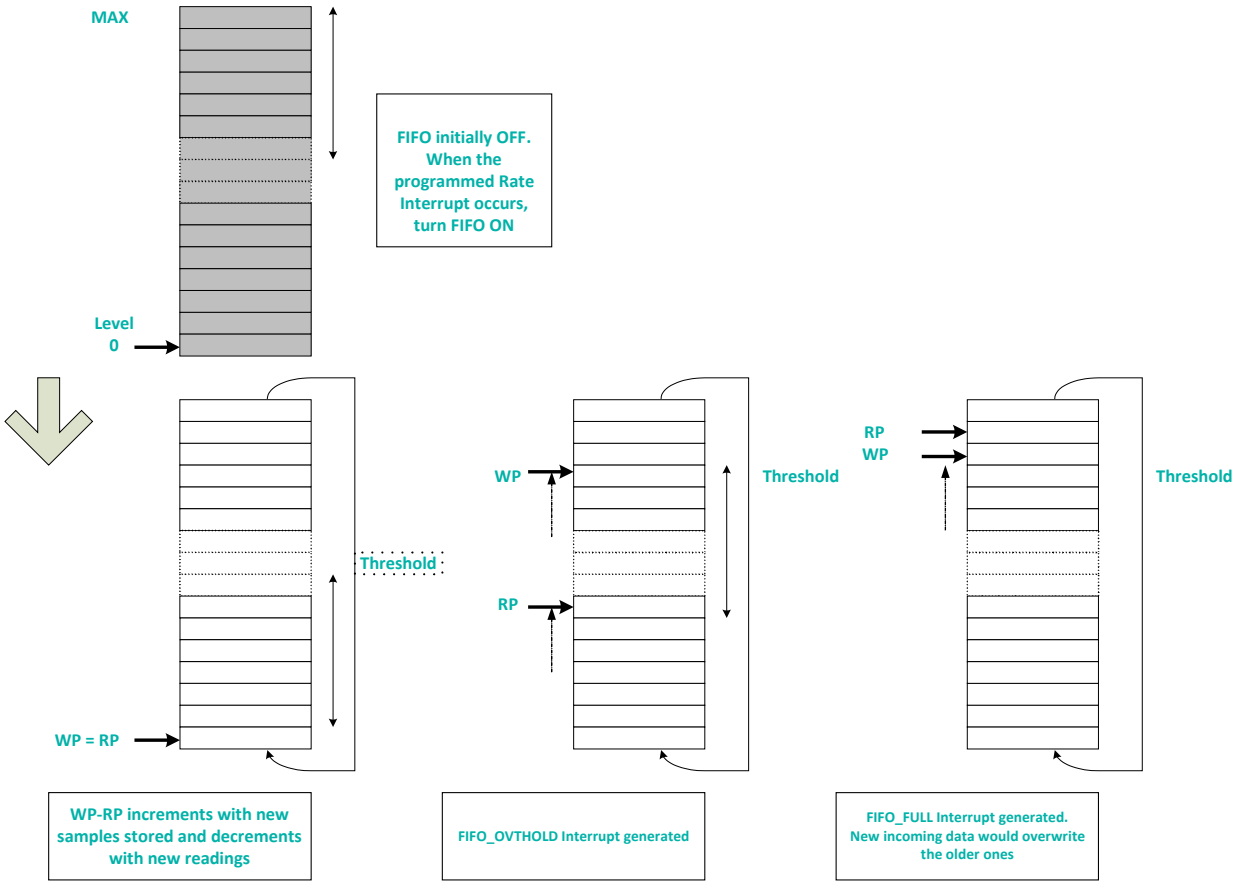



***Figure 14: FIFO Interrupt Mode, Overwrite***

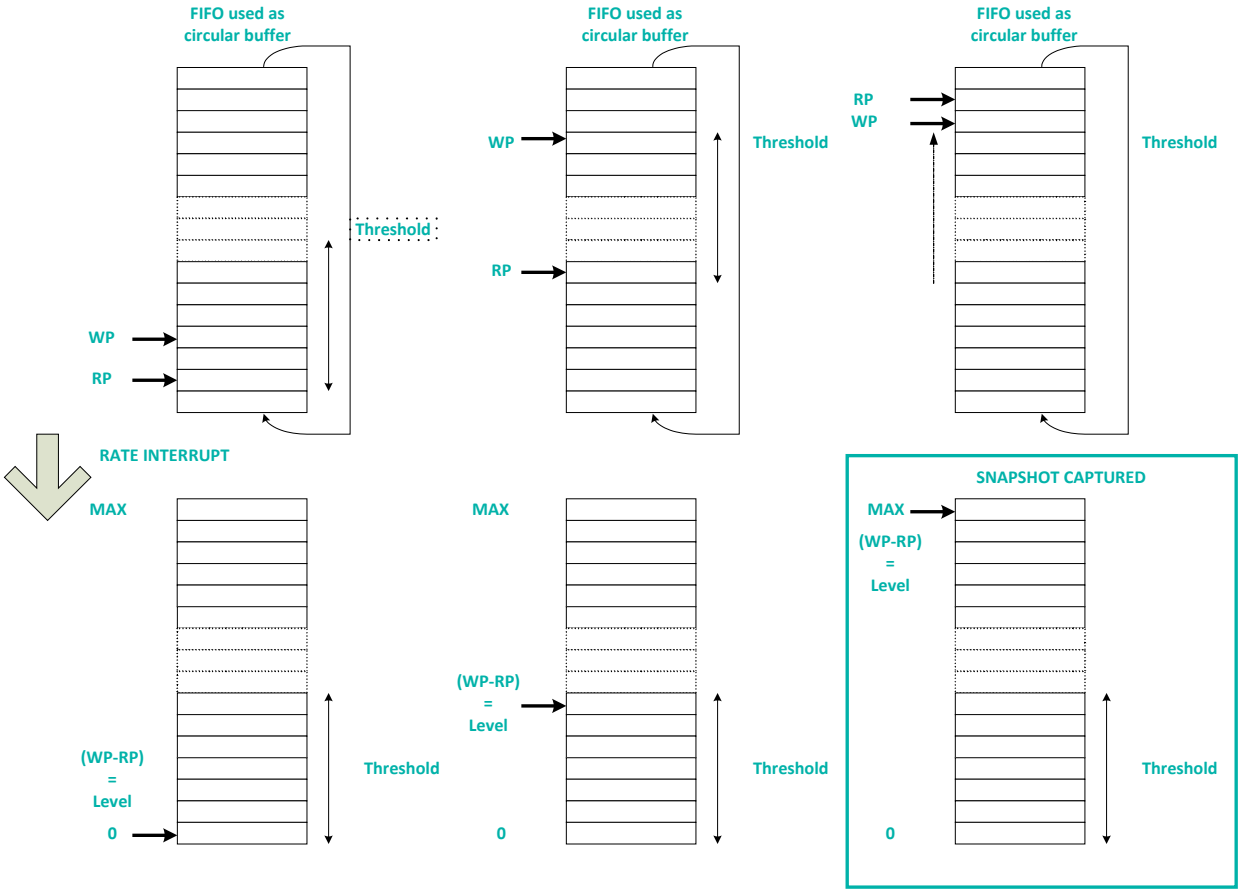



***Figure 15: FIFO Snapshot Mode***

### 9.3.4 Snapshot Mode

- FIFO is initially in normal mode with overrun enabled.
- When a Rate Interrupt (either OR or AND) is generated, the FIFO switches automatically to not-overrun mode. It stores the data at the selected ODR until the FIFO becomes full.
- When FIFO is full, an interrupt can be generated.
- When FIFO is full, all the new incoming data will be discharged. Reading only a subset of the data already stored into the FIFO keeps locked the possibility for new data to be written.
- Only if all the data are read the FIFO will restart saving data.
- If communication speed is high, data loss can be prevented.
- In order to prevent a FIFO full condition, the required condition is to complete the reading of the data set before the next DATA_READY occurs.
- If this condition is not guaranteed, data can be lost.

## 9.4 Example of FIFO Read/Write Pointers Evolution

The following drawing assumes:

- A reading frequency roughly twice the writing frequency (ODR);
- A FIFO threshold = 126
- FIFO is in normal mode

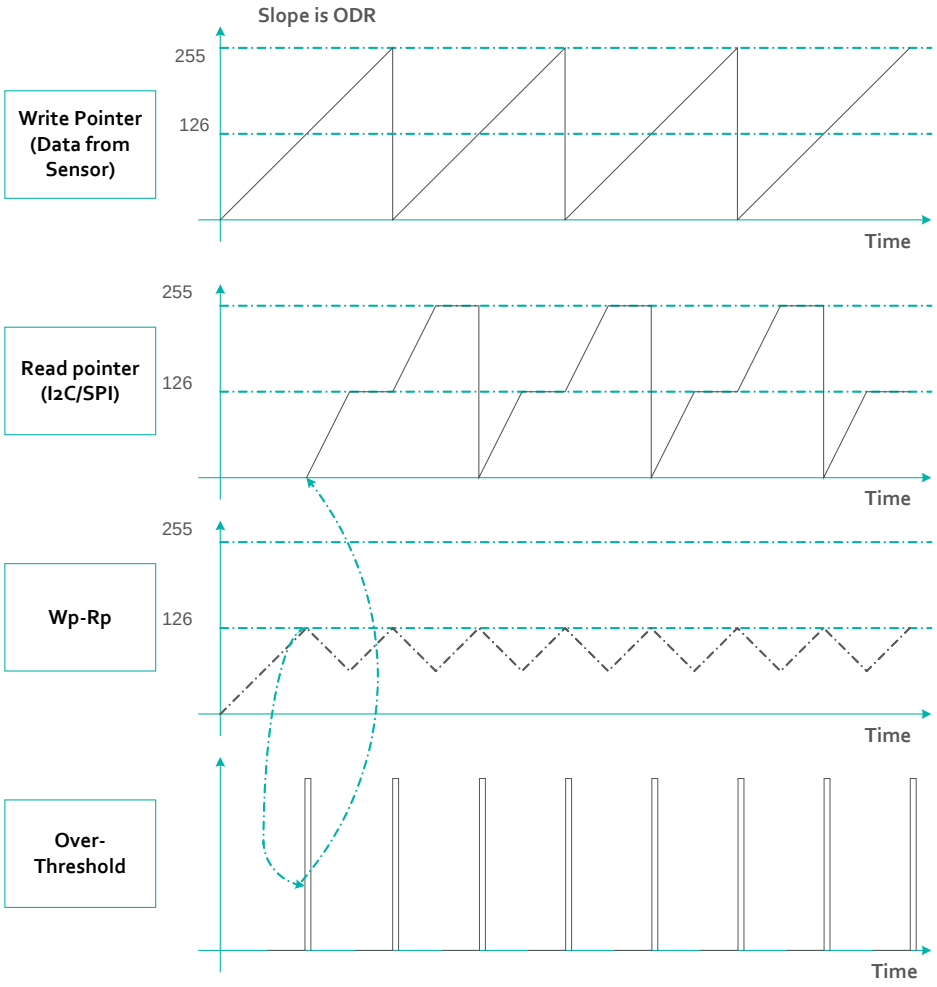



***Figure 16: FIFO Read/Write Pointer Evolution***

# 10 Programming Example

This chapter shows some examples on how to execute some operations on the device. Two functions are defined to abstract the SPI/$I^2C$ communication:

- *Write(<Register Address>, <Value>)*: Writes the *<Value>* to the register at the *<Register Address>;*
- *Read(<Register Address>)*: Returns the value of the register at the *<Register Address>*;
- *ReadBurstNoInc(<Register Address>, <count>)*: Returns an array of *count* elements with the result of a burst read with no auto-increment on *<Register Address>* register;

## 10.1 Simple Read-Out Sequence, No FIFO, No Interrupts

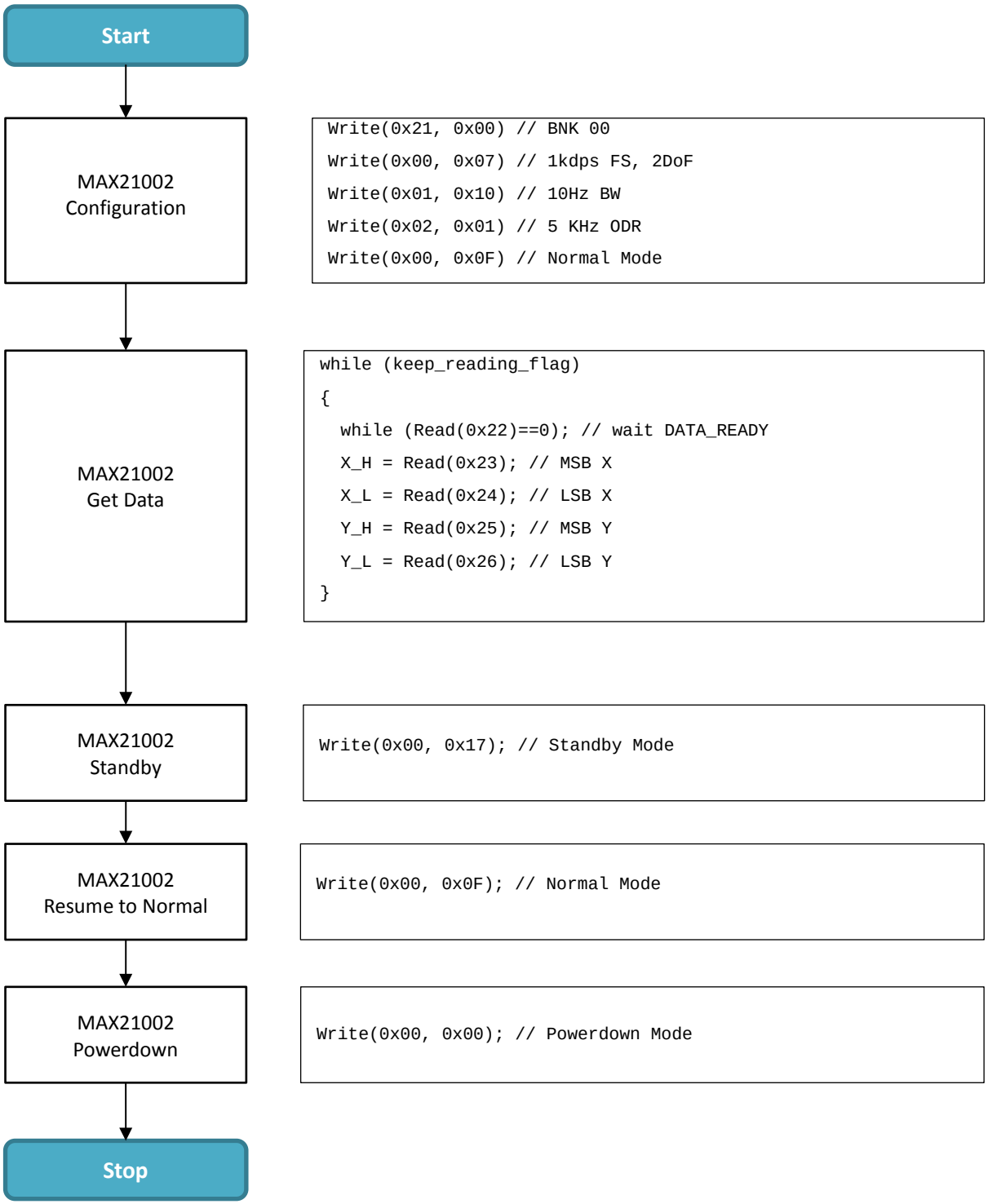



***Figure 17: Simple Read-Out Sequence, No FIFO, No Interrupts***

## 10.2 Simple Read-Out Sequence, FIFO Normal Mode, No Interrupts

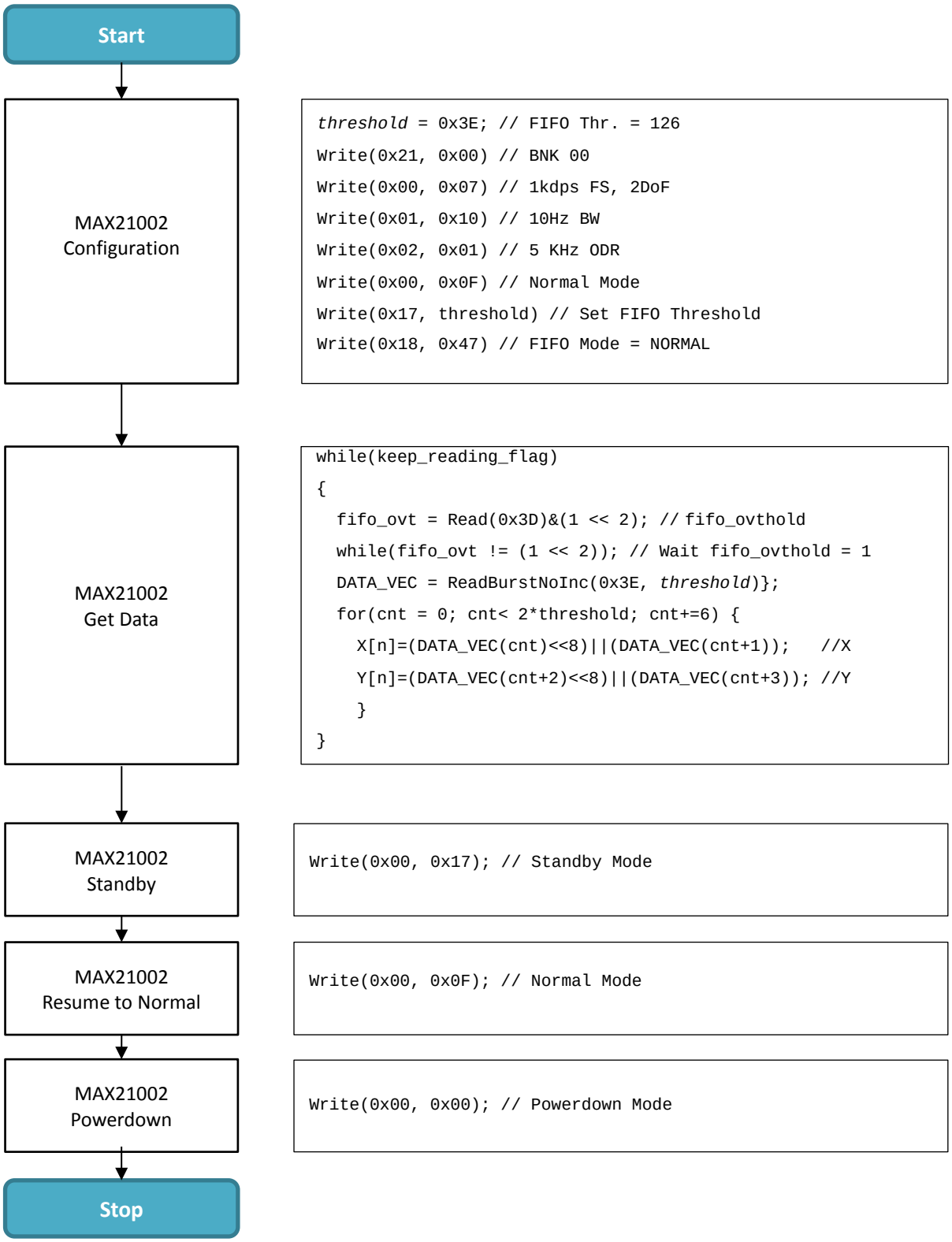



***Figure 18: Simple Read-Out Sequence, FIFO Normal, No Interrupts***

## 10.3Simple Read-Out Sequence, Normal Mode, Data-Ready Interrupts

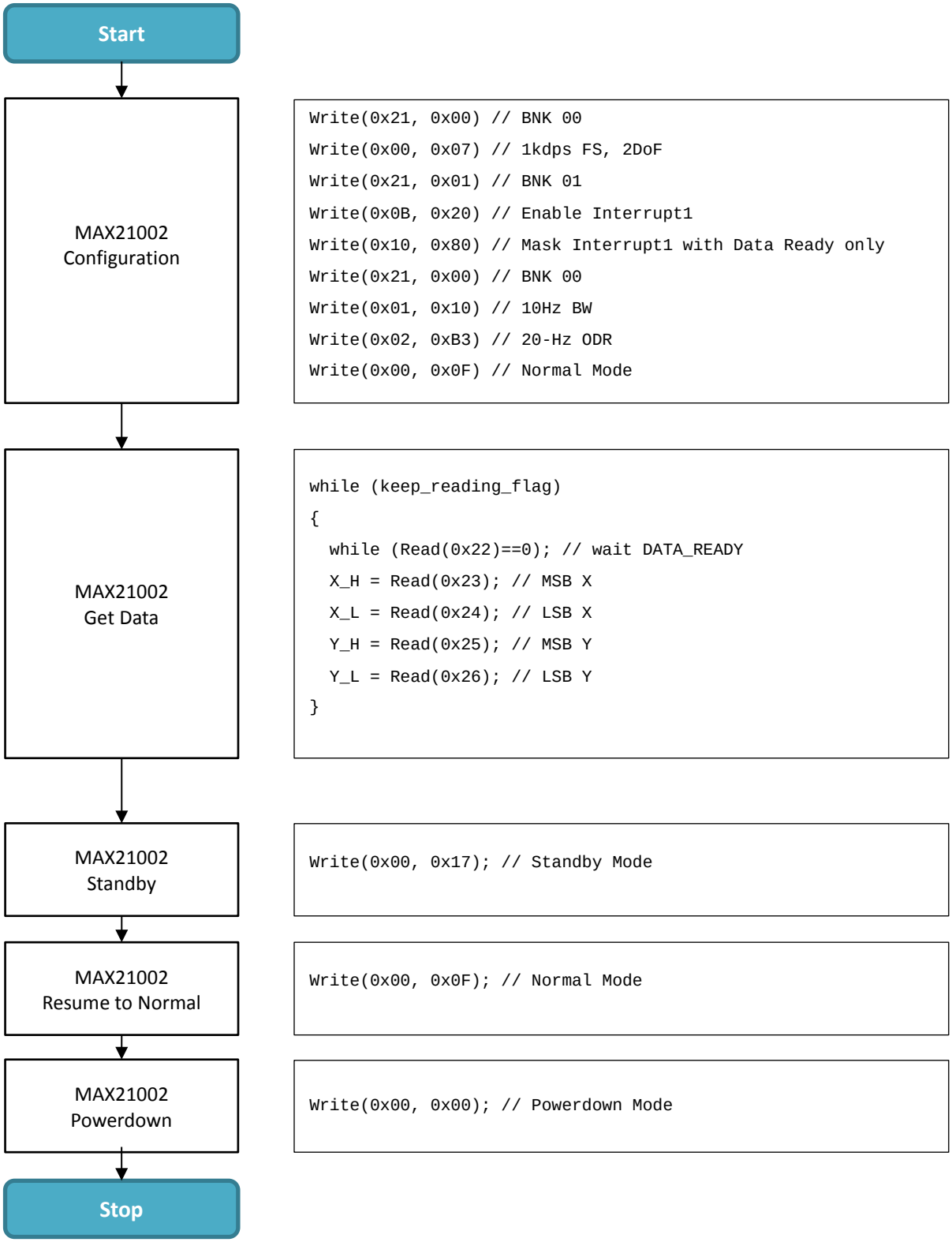



***Figure 19: Simple Read-Out Sequence, Normal Mode, w/ Data Ready Interrupts and No FIFO***

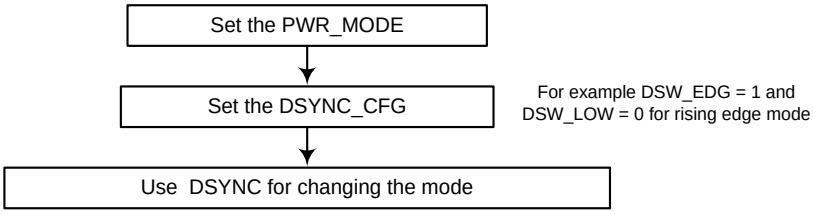
# 11 DSYNC

Data synchronization pin of MAX21002 can be used in various use cases such as controlling the power mode of MAX21002 through an external device, capturing the data in FIFO with an event triggered by camera or GPS, and lastly mapping the DSYNC level in LSB of the sensor output data in order to compare the data timing with respect to the DSYNC edge timing.

The DSYNC operation modes are described in three different sub-sections:

## 11.1 Power Mode Switching Using DSYNC

DSYNC pin can be configured to trigger the switching from a power mode to another. In order to achieve this, the following flow has to be executed:




In Power mode table, the power mode transitions with respect to DSYNC active level have been described. In order to enable power mode transition with DSYNC level, DSYNC_CFG register following bit have to be set properly:

***DSW_EDG:*** 0: DSYNC is active on level, 1: DSYNC is active on edge

***DSW_LOW:*** When 1, DSYNC is an active low level control to wake up. When 0, DSYNC is an active high level to wake up. This bit affects both the edge and the level modes

### 11.1.1 Example Register Settings for Power Mode Switching

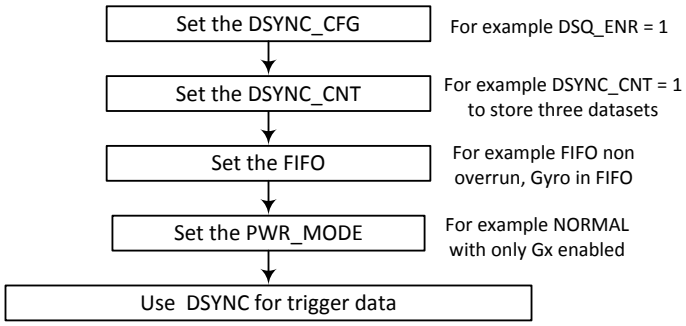
***Table 5: Register Settings for Switching Power Mode***

| Device | R/W | Bank | Register | Value | Comment |
|---|---|---|---|---|---|
| MAX21002 | W | 0x00 | 0x00 | 0x3F | FS=1000-dps (make sure that 0x01 [bit 0] is 0, Power Mode by DSYNC, transition from power down to normal mode (all axes are enabled) |
| MAX21002 | W | 0x00 | 0x1A | 0x20 | DSYNC active ‘on rising edge’ |

## 11.2 Filling FIFO with DSYNC Edge

With this function turned on, DSYNC signal is used to capture data and fill the FIFO with a selected number of data. It is enabled with DSYNC_CFG register and number of samples is configured using the DSYNC_CNT register.

In order to configure data filling with FIFO, the following procedure has to be performed:




The number of dataset stored in the FIFO is given by: ***N_data_set_stored_in_FIFO = DSYNC_CNT + 1***

A dataset is composed by the numbers of the axes enabled for each sensor (2 for Gyro). The DSYNC frequency must be: ODR / (DSYNC_CNT + 1) if one edge only is selected. Minimum DSYNC pulse duration is 500ns.

### 11.2.1 Example Register Settings for FIFO Filling with DSYNC

***Table 6: Register Settings for FIFO Filling with DSYNC***

| Device | R/W | Bank | Register | Value | Comment |
|---|---|---|---|---|---|
| MAX21002 | W | 0x00 | 0x1A | 0x80 | Enable data filling on rising edge |
| MAX21002 | W | 0x00 | 0x1B | 0x04 | DSYNC_CNT=4 FIFO filled with 2*4+1=9 datasets |
| MAX21002 | W | 0x00 | 0x18 | 0x47 | FIFO in normal mode, and Gyro X, Y are stored |
| MAX21002 | W | 0x00 | 0x00 | 0x0F | FS=1000-dps, Power mode normal, all axes are enabled |

## 11.3 Map DSYNC on LSb

When this function is turned on, the DSYNC signal is mapped into the LSb of all the sensors that are enabled for this functionality. It is configured through DSYNC_CFG register, and the following have to be set accordingly:

***DSM_ENB:*** When 1, the DSYNC signal is mapped onto the Gyro LSB

***DS_TEMP*** When 1, the DSYNC signal is mapped onto the temperature LSB

In this configuration, the DSYNC is active only on the level. To map the DSYNC signal in the LSb correctly, it is suggested to use an ODR that is bigger than the DSYNC frequency.

### 11.3.1 Example Register Settings for Mapping DSYNC level on LSB

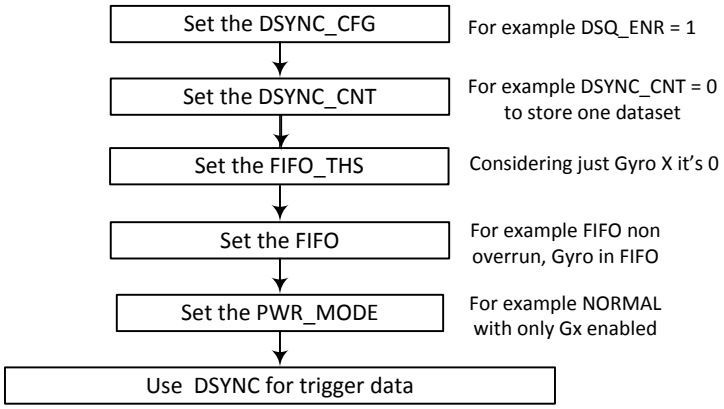
***Table 7: Register Settings for Mapping LSB***

| Device | R/W | Bank | Register | Value | Comment |
|---|---|---|---|---|---|
| MAX21002 | W | 0x00 | 0x00 | 0x0F | FS=1000dps (make sure that 0x01 [bit 0] is 0, Power Mode is set normal |
| MAX21002 | W | 0x00 | 0x1A | 0x08 | DSYNC mapped on Gyro |

## 11.4 Example of DSYNC Application: Generation of Non-Standard ODR

One of the possible uses of the DSYNC functionality is to generate the output data at a particular rate that it is not defined in the list of the possible ODRs. Using the “data queuing in FIFO with DSYNC” function and the FIFO over-threshold interrupt, it is possible to generate a data-ready signal at a specific data rate. In order to minimize the ODR jitter, the max ODR of the device should be selected, so that the max jitter is equal to ± $1/(2*ODR_{max})$.

For configuring this functionality the following procedure is suggested:




Based on this procedure, the following operation described with a timing diagram can be obtained:

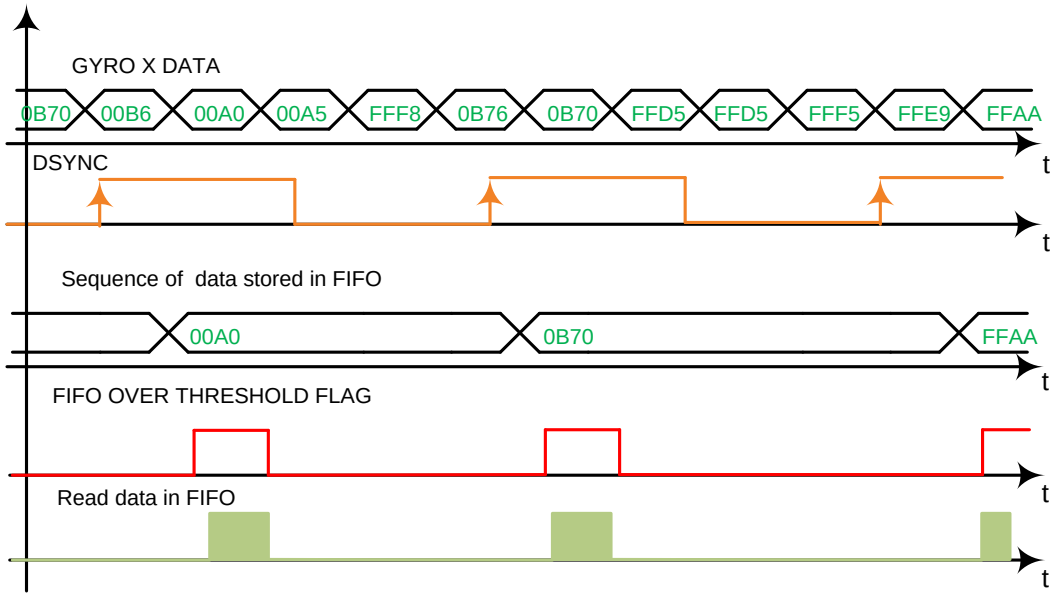



***Figure 20: Timing Diagram to Generate a Nonstandard ODR***

# 12 Register Map

## 12.1 COMMON BANK

*Table 8: Common Bank*

| Name | Register Address | Type | Default Value | Comment |
|---|---|---|---|---|
| WHO_AM_I | *0x20* | R | 1011 0001 | Device ID (0xB1) |
| BANK_SELECT | *0x21* | R/W | 0000 0000 | Register Bank Selection |
| SYSTEM_STATUS | *0x22* | R | 0000 0000 | System Status Register |
| GYRO_X_H | *0x23* | R | Data | Bits [15:8] of gyroscope X output if big endian<br>Bits [7:0] of gyroscope X output if little endian |
| GYRO_X_L | *0x24* | R | Data | Bits [7:0] of gyroscope X output if big endian<br>Bits [15:8] of gyroscope X output if little endian |
| GYRO_Y_H | *0x25* | R | Data | Bits [15:8] of gyroscope Y output if big endian<br>Bits [7:0] of gyroscope Y output if little endian |
| GYRO_Y_L | *0x26* | R | Data | Bits [7:0] of gyroscope Y output if big endian<br>Bits [15:8] of gyroscope Y output if little endian |
| TEMP_H | *0x29* | R | Data | Bits [15:8] of gyroscope Z output if big endian<br>Bits [7:0] of gyroscope Z output if little endian |
| TEMP_L | *0x2A* | R | Data | Bits [7:0] of gyroscope Z output if big endian<br>Bits [15:8] of gyroscope Z output if little endian |
| **RFU** | *0x2B:0x3A* | R | 0000 0000 | |
| HP_RST | *0x3B* | W | 0000 0000 | Highpass Filter Reset |
| FIFO_COUNT | *0x3C* | R | 0000 0000 | Available number of FIFO samples |
| FIFO_STATUS | *0x3D* | R | 0000 0000 | FIFO Status Flags |
| FIFO_DATA | *0x3E* | R/W | Data | FIFO data register |
| PAR_RST | *0x3F* | W | 0000 0000 | Parity Reset (Reset on Write) |

### 12.1.1 WHO_AM_I

| Register Address | Bank COMMON - 0x20 (Hex) - 32 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 10110001 (0xB1) | | | | | | | |

#### *Description*

WHO_AM_I identifies the MAX21002 with the value of 0xB1.

### 12.1.2 BANK_SELECT

| Register Address | Bank COMMON - 0x21 (Hex) - 31 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Fields | RFU | | | | ***bank_sel*** | | | |
| Type | R | | | | R/W | | | |
| Default Value | 0000 | | | | 0000 | | | |

#### *Description*

There are total 48 registers in the User Space of MAX21002: Bank 0, Bank 1 and Common Bank. ***bank_sel*** is used to access the registers in two different banks : Bank 0, Bank 1, whereas Common Bank registers can be accessed independent of the ***bank_sel*** value. Bank 0 and Bank 1 have total 32 registers (address range from 0x00 to 0x1F) and Common Bank has the register address range from 0x20 to 0x3F.

#### *Fields*

***bank_sel:*** Selects the total number of 16 banks each of which has total 32 bytes (registers). For the user, the valid values of BANK_SEL are:

| Values | Bank Selection |
|---|---|
| 0000 | Bank 0 (User Bank) |
| 0001 | Bank 1 (Interrupt Bank) |
| 0010-1111 | Reserved Banks |

### 12.1.3 SYSTEM_STATUS

| Register Address | Bank COMMON - 0x22 (Hex) - 34 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | RFU | | | | | | ***gyro_err*** | ***gyro_dr*** |
| **Type** | R | | | | | | R | R |
| **Default Value** | 000000 | | | | | | 0 | 0 |

#### *Description*

System Status register reports two fundamental flags necessary to properly manage the communication with the MAX21002. Ideally, every new data-reading operation from the MAX21002 should only take place when at least a new DATA_READY (***gyro_dr***) event occurs. Failure to read data at every DATA_READY event may result in either reading the same data more than once or missing at least one ouput data. That is particularly true when the FIFO is disabled. The way the ***gyro_dr*** flag is reset can be configured using register DR_CFG (0x13).

The ***gyro_err*** flag indicates the occurence of either one of the events described above. If the FIFO is used, multiple data can be read safely. In order to set up FIFO and burst read the data loaded on the FIFO (refer to FIFO section).

#### *Fields*

***gyro_err:*** Active high when a new data is generated before or during data reading

***gyro_dr:*** Active high when a new set of gyroscope data is available

### 12.1.4 GYRO_X_H

| Register Address | Bank COMMON - 0x23 (Hex) - 35 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | | | | | | | | |
| **Type** | R | | | | | | | |
| **Default Value** | 00000000 | | | | | | | |

#### *Description*

GYRO_X_H stores the MSB (bit[15:8]) of the most recent X-axis gyroscope output when endian bit is set to 0, or the LSB (bit[7:0]) when endian bit is set to 1.

### 12.1.5 GYRO_X_L

| Register Address | Bank COMMON - 0x24 (Hex) - 36 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

GYRO_X_L stores the LSB (bit[7:0]) of the most recent X-axis gyroscope output when endian bit is set to 0, or the MSB (bit[15:8]) when endian bit is set to 1.

### 12.1.6 GYRO_Y_H

| Register Address | Bank COMMON - 0x25 (Hex) - 37 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

GYRO_Y_H stores the MSB (bit[15:8]) of the most recent Y-axis gyroscope output when endian bit is set to 0, or the LSB (bit[7:0]) when endian bit is set to 1.

### 12.1.7 GYRO_Y_L

| Register Address | Bank COMMON - 0x26 (Hex) - 38 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

GYRO_Y_L stores the LSB (bit[7:0]) of the last Y-axis gyroscope output when endian bit is set to 0, or the MSB (bit[15:8]) when endian bit is set to 1.

### 12.1.8 TEMP_H

| Register Address | Bank COMMON - 0x29 (Hex) - 41 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

TEMP_H stores the MSB (bit[15:8]) of the most recent temperature sensor output when endian bit is set to 0, or the LSB (bit[7:0]) when endian bit is set to 1.

### 12.1.9 TEMP_L

| Register Address | Bank COMMON - 0x2A (Hex) - 42 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

TEMP_L stores the LSB (bit[7:0]) of the most recent temperature sensor output when endian bit is set to 0, or the MSB (bit[15:8]) when endian bit is set to 1.

### 12.1.10 HP_RST

| Register Address | Bank COMMON - 0x3B (Hex) - 59 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

Reading HP_RST register resets the Highpass filter output.

### 12.1.11 FIFO_COUNT

| Register Address | Bank COMMON - 0x3C (Hex) - 60 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

FIFO_COUNT provides the total number of FIFO 16-bits words available in FIFO.

### 12.1.12 FIFO_STATUS

| Register Address | Bank COMMON - 0x3D (Hex) - 61 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | ***fifo_cnt_msb*** | ***RFU*** | | ***fifo_data_lost*** | ***fifo_read_empty*** | ***fifo_ovthold*** | ***fifo_full*** | ***fifo_empty*** |
| **Type** | R | R | | R | R | R | R | R |
| **Default Value** | 0 | 00 | | 0 | 0 | 0 | 0 | 0 |

#### *Description*

FIFO_STATUS register provides the status of all the potential FIFO events.

#### *Fields*

***fifo_cnt_msb:*** The MSb of the FIFO count (bit 7 of the FIFO_COUNT register)

***fifo_data_lost:*** At least one data was lost while the FIFO was full

***fifo_read_empty:*** At least one read has occured while the FIFO was empty

***fifo_ovthold:*** the number of data in FIFO exceeds the threshold

***fifo_full:*** FIFO is full

***fifo_empty:*** FIFO is empty

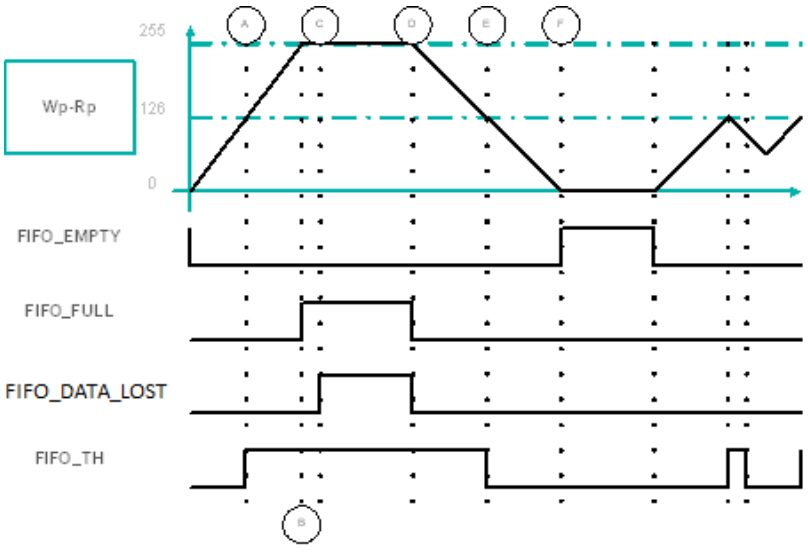



***Figure 21: FIFO Flags***

***(A)*** Wp-Rp = Programmed threshold

***(B)*** FIFO is full, next write operation will cause data to be lost

***(C)*** At least one data has been lost

***(D)*** Read access clears FIFO_FULL and FIFO_WR_FULL flags

***(E)*** Wp-Rp < Programmed threshold

***(F)*** FIFO is empty and all the available new data have been read

### 12.1.13 FIFO_DATA

| Register Address | Bank COMMON - 0x3E (Hex) - 62 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

FIFO_DATA register is used to read and write data on the FIFO buffer. The contents of the sensor data registers are written into the FIFO buffer when their corresponding FIFO enable flags are set to 1 through the FIFO_CFG register. When the FIFO buffer has overflowed, the oldest data will be lost and new data will be written to the FIFO if the override bit is set in the FIFO_CFG register, otherwise the new data is discarded. If the FIFO buffer has overflowed, the status bit ***fifo_data_lost*** (bit 4 of FIFO_STATUS register) is automatically set to 1.

If the FIFO buffer is empty, reading this register will return the last byte that was previously read from the FIFO until new data is available. The user should check FIFO_COUNT to ensure that the FIFO buffer is not read when empty.

### 12.1.14 PAR_RST

| Register Address | Bank COMMON - 0x3F (Hex) - 63 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

Reading PAR_RST register clears the ***parity_error*** flag (ITF_OTP).

## 12.2 USER BANK #0 (*bank_sel* = 0000)

***Table 9: User Bank 0***

| Name | Register Address | Type | Default Value | Comment |
|---|---|---|---|---|
| **POWER_CFG** | *0x00* | R/W | 0000 0111 | Power mode configuration |
| **SENSE_CFG1** | *0x01* | R/W | 0010 1000 | Sense configuration : LP and OIS |
| **SENSE_CFG2** | *0x02* | R/W | 0001 0011 | Sense configuration : ODR |
| **SENSE_CFG3** | *0x03* | R/W | 0000 0000 | Sense configuration : HP |
| **RFU** | *0x04:0x12* | R | 0000 0000 | |
| **DR_CFG** | *0x13* | R/W | 0000 0001 | Data Ready configuration |
| **IO_CFG** | *0x14* | R/W | 0000 0000 | Input/Output configuration |
| **I2C_CFG** | *0x15* | R/W | 0000 0100 | $I^2C$ configuration |
| **ITF_OTP** | *0x16* | R/W | 0000 0000 | Interface and OTP configuration |
| **FIFO_TH** | *0x17* | R/W | 0000 0000 | FIFO Threshold configuration |
| **FIFO_CFG** | *0x18* | R/W | 0000 0000 | FIFO Mode configuration |
| **RFU** | *0x19* | R | 0000 0000 | |
| **DSYNC_CFG** | *0x1A* | R | 0000 0000 | DSYNC Configuration |
| **DSYNC_CNT** | *0x1B* | R | 0000 0000 | DSYNC Counter |
| **RFU** | *0x1C* | R | 0000 0000 | |
| **RFU** | *0x1D* | R | 0000 0000 | |
| **RFU** | *0x1E* | R | 0000 0000 | |
| **RFU** | *0x1F* | R | 0000 0000 | |

### 12.2.1 POWER_CFG

| Register Address | Bank 0 - 0x00 (Hex) - 0 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | ***sns_dout_fsc*** | | ***pwr_mode*** | | | RFU | ***sns_en_y*** | ***sns_en_x*** |
| **Type** | R/W | | R/W | | | R/W | R/W | R/W |
| **Default Value** | 00 | | 000 | | | 1 | 1 | 1 |

#### *Description*

Gyroscope Full scale, power mode and axis configuration register.

#### *Fields*

***sns_dout_fsc:*** Full scale configuration bits.
When ***sns_ois*** = 0: { b00: 1000 dps; b01: 500 dps; b10: 250 dps; b11: 125 dps}
When ***sns_ois*** = 1: { b00: 125 dps; b01: 62.5 dps; b10: 31.25 dps; b11: Reserved}

***pwr_mode:*** Power mode configuration register (see Table 10)

***sns_en_y:*** Gyroscope Y-axis enable bit (0: disabled; 1: enabled)

***sns_en_x:*** Gyroscope X-axis enable bit (0: disabled; 1: enabled)

***Table 10 Power Mode Configuration***

| ***pwr_mode*** | **DSYNC (pin)** | **Power Mode** | **Description** |
|---|---|---|---|
| b000 | X | Power-Down mode | In Power-Down mode, the IC is configured to minimize the power consumption. In power-down mode, registers can still be read and written, but the gyroscope cannot generate new data. Compared to the standby mode, it takes longer to activate the IC and to start collecting data from the gyroscope. |
| b001 | X | Normal mode | In Normal mode, the IC is operational with minimum noise level. |
| b010 | X | Standby mode | To reduce power consumption and have a shorter turn-on time, the IC features a standby mode. In Standby mode, the IC does not generate data, as a significant portion of the signal processing resources is turned off to save power. Still, this mode enables a much quicker turn-on time. |
| b011 | X | Eco mode | The Eco mode reduces power consumption with the same sensor accuracy at the price of a higher rate noise density. This unique feature can be activated with four ODRs: 25Hz, 50Hz, 100Hz, and 200Hz. |
| b100 | 0 → 1 | Standby mode → Eco mode | If DSYNC pin is LOW, the power mode is set to Standby Mode; if DSYNC pin is HIGH, the power mode is set to Eco Mode. |
| b101 | 0 → 1 | Power-Down mode → Eco mode | If DSYNC pin is LOW, the power mode is set to Power-Down Mode; if DSYNC pin is HIGH, the power mode is set to Eco Mode. |
| b110 | 0 → 1 | Standby mode → Normal mode | If DSYNC pin is LOW, the power mode is set to Standby Mode; if DSYNC pin is HIGH, the power mode is set to Normal Mode. |
| b111 | 0 → 1 | Power-Down mode → Normal mode | If DSYNC pin is LOW, the power mode is set to Power-Down Mode; if DSYNC pin is HIGH, the power mode is set to Normal Mode. |

### 12.2.2 SENSE_CFG1

| Register Address | Bank 0 - 0x01 (Hex) - 1 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | ***self_test*** | | ***sns_lpf_bnd*** | | | | RFU | ***sns_ois*** |
| **Type** | R | | R/W | | | | R | R |
| **Default Value** | 00 | | 1010 | | | | 0 | 0 |

#### *Description*

Low Pass filter, OIS and Self Test configuration register. When a self-test mode is selected, an offset is generated on the digital output whose amount depends on the selected full-scale. The output of this parameter is affected by a strong spread, in the order of +/- 50%. This test allows detecting both electrical and mechanical issues.

#### *Fields*

***self_test:*** ***self_test[0]***: Self Test enable bit:
- 0: disabled
- 1: enabled

***self_test[1]***: Self Test output sign inversion bit
When ***self_test[1]*** = 0, Self Test output { X, Y} are:
- FS = 1000 dps: { 225, -225, 225 } dps
- FS = 500 dps: { 110, -110, 110 } dps
- FS = 250 dps: { 55, -55, 55 } dps

Output sign is inverted when ***self_test[1]*** = 1.

***sns_lpf_bnd:*** Low pass filter bandwidth value (see Table 11)

***sns_ois*** For Optical Image Stabilization (OIS) applications the main requirement is the resolution. So, when this mode is enabled (***sns_ois*** = 1), the available full-scale settings are: 125 dps, 62.5 dps and 31.25 dps (see POWER_CFG register)

***Table 11: Bandwidth Configuration***

| ***sns_lpf_bnd*** | **Bandwidth** |
|---|---|
| b0000 | 2Hz |
| b0001 | 4Hz |
| b0010 | 6Hz |
| b0011 | 8Hz |
| b0100 | 10Hz |
| b0101 | 14Hz |
| b0110 | 22Hz |
| b0111 | 32Hz |
| b1000 | 50Hz |
| b1001 | 75Hz |
| b1010 | 100Hz (default) |
| b1011 | 150Hz |
| b1100 | 200Hz |
| b1101 | 250Hz |
| b1110 | 300Hz |
| b1111 | 400Hz |

### 12.2.3 SENSE_CFG2

| Register Address | Bank 0 - 0x02 (Hex) - 2 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00010011 | | | | | | | |

#### *Description*

Output Data Rate configuration register selects the preferred Output Data Rate (ODR) according to the description below.

In Normal mode:
ODR = 10kHz/(n+1) for n=0..99 (100Hz<=ODR<=10kHz)
ODR = 10kHz/(100+5*(n-99)) for n=100..179 (20Hz<=ODR<100Hz)
ODR = 10kHz / (500+20*(n-179) ) for n=180..255 (4.95Hz<=ODR<20Hz)
where n is the decimal value written in ODR register.

In ECO Mode:
SENSE_CFG2[1:0] = b00 → ODR = 200Hz
SENSE_CFG2[1:0] = b01 → ODR = 100Hz
SENSE_CFG2[1:0] = b10 → ODR = 50Hz
SENSE_CFG2[1:0] = b11 → ODR = 25Hz

### 12.2.4 SENSE_CFG3

| Register Address | Bank 0 - 0x03 (Hex) - 3 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | RFU | | ***sns_byp_lpf*** | ***sns_dout_cfg*** | ***sns_hpf_co*** | | | |
| **Type** | R | | R/W | R/W | R/W | | | |
| **Default Value** | 00 | | 0 | 0 | 0000 | | | |

#### *Description*

Highpass filter configuration register comprises 3 fields. The least significant 4 bits can be used to select the cut-off frequency of the highpass filter. Bit[5:4] is used to bypass the lowpass filter and highpass filter, respectively.

#### *Fields*

***sns_byp_lpf:*** Bypass lowpass filter:
- 0: LPF enabled (default)
- 1: LPF bypassed

***sns_dout_cfg:*** Bypass highpass filter:
- 0: HPF bypassed (default);
- 1: HPF enabled

***sns_hpf_co:*** Setting HPF cut-off frequency (see Table 12)

***Table 12: HPF Cut-Off Frequencies***

| ***SNS_HPF_CO*** | **HPF f-3dB** |
|---|---|
| b0000 | 0.1Hz (default) |
| b0001 | 0.2Hz |
| b0010 | 0.3Hz |
| b0011 | 0.5Hz |
| b0100 | 0.7Hz |
| b0101 | 1.0Hz |
| b0110 | 1.7Hz |
| b0111 | 3.0Hz |
| b1000 | 4.5Hz |
| b1001 | 7.0Hz |
| b1010 | 11Hz |
| b1011 | 17Hz |
| b1100 | 26Hz |
| b1101 | 40Hz |
| b1110 | 64Hz |
| b1111 | 100Hz |

### 12.2.5 DR_CFG

| Register Address | Bank 0 - 0x13 (Hex) - 19 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | RFU | | ***dr_rst_mode*** | | RFU | | ***coarse_temp*** | ***temp_en*** |
| **Type** | R | | R/W | | R | | R/W | R/W |
| **Default Value** | 00 | | 00 | | 00 | | 0 | 1 |

#### *Description*

Data Ready reset and temperature sensor settings.

#### *Fields*

***dr_rst_mode:*** Controls the way Data Ready is reset; 3 available modes are available:
- **ALL** (b00): DATA_READY is cleared after all the active channels are read. Data are not updated until the clear operation is accomplished.
- **ANY** (b01): DATA_READY is cleared when at least a byte of one of the active channels is read. Data are updated independently from the clear operation.
- **STATUS** (b10): DATA_READY is cleared when status register is read. Data are not updated until the clear operation is accomplished.

***coarse_temp:*** **Fine** (b0): Temperature data is updated only when temperature data registers are read (TEMP_H, TEMP_L).
**Coarse** (b1): Temperature data is updated only when the MSB of the temperature data is read (TEMP_H).

***temp_en:*** Temperature sensor enable (0: Enabled; 1: Disabled)

### 12.2.6 IO_CFG

| Register Address | Bank 0 - 0x14 (Hex) - 20 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | ***dsync_pd_en*** | ***dsync_pu_en*** | ***int1_pd_en*** | ***int1_pu_en*** | ***int2_pd_en*** | ***int2_pu_en*** | RFU | RFU |
| **Type** | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| **Default Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

I/O Configuration Register controls the pullup and pulldown resistors of the pins DSYNC, SCL, SDA, INT1 and INT2.

#### *Fields*

***dsync_pd_en:*** The internal pulldown of the pad is (0: disconnected; 1: connected)
***dsync_pu_en:*** The internal pullup of the pad is (0: disconnected; 1: connected)
***int1_pd_en:*** The internal pulldown of the pad is (0: disconnected; 1: connected)
***int1_pu_en:*** The internal pullup of the pad is (0: disconnected; 1: connected)
***int2_pd_en:*** The internal pulldown of the pad is (0: disconnected; 1: connected)
***int2_pu_en:*** The internal pullup of the pad is (0: disconnected; 1: connected)

### 12.2.7 I2C_CFG

| Register Address | Bank 0 - 0x15 (Hex) - 21 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | RFU | ***if_setting*** | | | ***drive*** | | RFU | ***i2c_off*** |
| **Type** | R | R/W | | | R/W | | R | R/W |
| **Default Value** | 0 | 000 | | | 01 | | 0 | 0 |

#### *Description*

I2C_CFG sets for $I^2C$ bus speed and output drive strength.

#### *Fields*

***if_setting:*** 5 settings are available:
- b000: $I^2C$ Fast Mode without anti-spike filter
- b001: $I^2C$ Fast Mode standard configuration
- b011: $I^2C$ High Speed without anti-spike filter
- b100: $I^2C$ Fast Mode without filters and delays
- b101: SPI interface recommended
- b110: Reserved
- b111: Reserved

***drive:*** IO output current
- b00: 3 mA;
- b01: 6 mA;
- b10: 6 mA;
- b11:12 mA

***i2c_off:*** $I^2C$ interface is 0: enabled; 1: disabled

### 12.2.8 ITF_OTP

| Register Address | Bank 0 - 0x16 (Hex) - 22 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | RFU | ***parity_error*** | ***spi_3_wire*** | ***if_parity*** | | ***endian*** | ***otp_downloading*** | ***restart*** |
| **Type** | R | R | R/W | R/W | | R/W | R | R/W |
| **Default Value** | 0 | 0 | 0 | 00 | | 0 | 0 | 0 |

#### *Description*

Interface and OTP configuration register.

#### *Fields*

***parity_error:*** {0: No error; 1:Error} in SPI/$I^2C$ address

***spi_3_wire:*** 0: 4-Wire SPI, 1:3-Wire SPI

***if_parity:*** Interface Bit 6 configuration setting.

- b00: Bit 6 indicates if the register address has to be incremented after each data byte when a burst operation is requested. If bit 6 is LOW, the register address is auto-incremented; if bit 6 is HIGH, the register address is kept unchanged (default).
- b01: Bit 6 represents the even parity bit. When this mode is selected and a burst operation is requested, the register address is incremented automatically after each data byte of the burst.
- b10: Bit 6 represents the odd parity bit. When this mode is selected and a burst operation is requested, the register address is incremented automatically after each data byte of the burst.
- b11: Don't care

when ***if_parity*** is not b00, then the burst is auto-incremental by default (see SPI Interface chapter).

***endian***

- 0: Big Endian (MS Byte, LS Byte); 1: Little Endian (LS Byte, MS Byte)

***endian*** bit affects the data stored into both the registers and the FIFO

***otp_downloading:*** Flag indicating that the OTP is being downloaded

***restart:*** Trigger a new download of the OTP. This bit is automatically reset when the operation is completed.

### 12.2.9 FIFO_TH

| Register Address | Bank 0 - 0x17 (Hex) - 23 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | | | | | | | | |
| **Type** | R/W | | | | | | | |
| **Default Value** | 00000000 | | | | | | | |

#### *Description*

FIFO Threshold configuration register. When the number of 16-bits samples stored in FIFO is above the threshold, the ***fifo_ovthold*** interrupt is generated. In this case, a sample is the entire set of axes; for example, if the threshold is set to 5 and all the axes are stored in FIFO, the interrupt is generated when the FIFO contains 5 sets of {X, Y} gyroscope data. This value must be different from 0.

### 12.2.10 FIFO_CFG

| Register Address | Bank 0 - 0x18 (Hex) - 24 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Bit #** | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| **Fields** | ***fifo_mode*** | | ***fifo_int_mode*** | ***fifo_overrun*** | RFU | RFU | ***fifo_store_y*** | ***fifo_store_x*** |
| **Type** | R/W | | R/W | R/W | R | R/W | R/W | R/W |
| **Default Value** | 00 | | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

FIFO configuration register determines which sensor measurements are loaded into the FIFO buffer and selects the desired FIFO behavior.

Data stored inside the sensor data registers will be loaded into the FIFO buffer if a sensor's respective ***fifo_store_****n* bit is set to 1 in this register. The behavior of FIFO writes, when the FIFO buffer is full, can be configured with the ***fifo_mode*** bit. In order to read the data in the FIFO buffer, the ***fifo_mode*** must be set to a value diffferent than 00.

The sensors are written into the FIFO at the Output Data Rate defined in SENSE_CFG2 register.

#### *Fields*

***fifo_mode:***
- b00: OFF Mode — FIFO is disabled and the data are exposed only through the sensor registers
- b01: Normal Mode — FIFO starts collecting the data immediately
- b10: Interrupt Mode — FIFO starts collecting the data right after a rate interrupt event (either OR or AND) is generated.
- b11: Snapshot Mode — FIFO starts collecting the data immediately, overwriting the oldest data in case of FIFO full. When a rate interrupt event (either OR or AND) is generated, data are collected until the FIFO is full, then the data collection is stopped without overwriting the oldest values.
  This mechanism is useful in case a post-processing of the data that generated the rate interrupt event is requested to better classify the event itself.

***fifo_int_mode:*** When Interrupt Mode is selected:
- 0: User OR mask;
- 1: User AND mask

***fifo_overrun:*** FIFO overrun mode:
- 0: When the FIFO is full, no more data are collected and newer data are discarded;
- 1: When the FIFO is full, oldest data are replaced by newer ones

***fifo_store_y:*** {0: Does Not Store; 1: Stores} the gyroscope Y-axis output in FIFO

***fifo_store_x:*** {0: Does Not Store; 1: Stores} the gyroscope X-axis output in FIFO

### 12.2.11 DSYNC_CFG

| Register Address | Bank 0 - 0x1A (Hex) - 26 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ds_queue_en_r | ds_queue_en_f | ds_wakeup_act_edg | ds_wakeup_act_low | ds_gyro_map_en | RFU | | ds_temp_map_en |
| Type | R/W | R/W | R/W | R/W | R/W | R | | R/W |
| Default Value | 0 | 0 | 0 | 0 | 0 | 00 | | 0 |

#### *Description*

DSYNC configuration register has to be used to configure the way the MAX21002 manages events occurring on the DSYNC pin. Multiple different actions can be taken simultaneously, like changing the power mode, mapping the DSYNC pin value onto the gyroscope LSB data and concurrently triggering the capture of new data.

When the DSYNC pin is configured as active on edge and a dynamic power mode is configured, only the active edge determines the transition. The opposite transition must be done within SW or by reversing the active edge.

#### *Fields*

| | |
|---|---|
| ***ds_queue_en_r:*** | Enable the data queueing when a rising edge on the DSYNC pin occurs |
| ***ds_queue_en_f:*** | Enable the data queueing when a falling edge on the DSYNC pin occurs |
| ***ds_wakeup_act_edg:*** | 0: DSYNC is active on level, 1: DSYNC is active on edge |
| ***ds_wakeup_act_low:*** | When 1, DSYNC is an active low level control to wake up. When 0, DSYNC is an active high level to wake up. This bit affects both the edge and the level modes |
| ***ds_gyro_map_en:*** | When 1, the DSYNC signal is mapped onto the Gyro LSB |
| ***ds_temp_map_en*** | When 1, the DSYNC signal is mapped onto the temperature LSB |

### 12.2.12 DSYNC_CNT

| Register Address | Bank 0 - 0x1B (Hex) - 27 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

This register sets the number of words to be filled in FIFO after a DSYNC pin event occurs. DSYNC counter configuration can be used to track the evolution of the rate signal from the gyroscope immediately after an external event captured on the DSYNC pin. The number of dataset stored in the FIFO is given by the following formula:

*N_data_set_stored_in_FIFO = DSYNC_CNT + 1*

## 12.3 USER BANK #1 (*bank_sel* = 0001)

***Table 13: User Bank 1***

| Name | Register Address | Type | Default Value | Comment |
|---|---|---|---|---|
| **INT_REF_X** | *0x00* | RW | 0000 0000 | Interrupt Reference for X-axis |
| **INT_REF_Y** | *0x01* | RW | 0000 0000 | Interrupt Reference for Y-axis |
| **INT_DEB_X** | *0x03* | RW | 0000 0000 | Interrupt Debounce, X |
| **INT_DEB_Y** | *0x04* | RW | 0000 0000 | Interrupt Debounce, Y |
| **INT_MSK_X** | *0x06* | RW | 0000 0000 | Interrupt Mask, X-axis zones |
| **INT_MSK_Y** | *0x07* | RW | 0000 0000 | Interrupt Mask, Y-axis zones |
| **INT_MASK_AO** | *0x09* | RW | 0000 0000 | Interrupt Masks, AND/OR |
| **INT_CFG1** | *0x0A* | RW | 0000 0000 | Interrupt Configuration #1 |
| **INT_CFG2** | *0x0B* | RW | 0010 0100 | Interrupt Configuration #2 |
| **INT_TMO** | *0x0C* | RW | 0000 0000 | Interrupt Timeout |
| **INT_STS_UL** | *0x0D* | R | 0000 0000 | Interrupt Sources, unlatched |
| **INT1_STS** | *0x0E* | R | 0000 0000 | Interrupt 1 Status, latched |
| **INT2_STS** | *0x0F* | R | 0000 0000 | Interrupt 2 Status, latched |
| **INT1_MSK** | *0x10* | RW | 1000 0000 | Interrupt 1 Mask |
| **INT2_MSK** | *0x11* | RW | 0000 0010 | Interrupt 2 Mask |
| **RFU** | *0x12:0x14* | R | 0000 0000 | |
| **OTP_STATUS** | *0x15* | R | 0000 0000 | OTP Status |
| **RFU** | *0x16:0x18* | R | 0000 0000 | |
| **SILICON_REV_OTP** | *0x19* | R | Variable | Silicon revision. |
| **SERIAL_0** | *0x1A* | R | Variable | Unique Serial Number, Byte 0 |
| **SERIAL_1** | *0x1B* | R | Variable | Unique Serial Number, Byte 1 |
| **SERIAL_2** | *0x1C* | R | Variable | Unique Serial Number, Byte 2 |
| **SERIAL_3** | *0x1D* | R | Variable | Unique Serial Number, Byte 3 |
| **SERIAL_4** | *0x1E* | R | Variable | Unique Serial Number, Byte 4 |
| **SERIAL_5** | *0x1F* | R | Variable | Unique Serial Number, Byte 5 |

### 12.3.1 INT_REF_X

| Register Address | Bank 1 - 0x00 (Hex) - 0 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

MSB of the reference for X-axis. The actual reference is then computed as ***INT_REF_X * 256***. When ***int_single_ref*** is set, the reference will be {INT_REF_X, INT_REF_Y}, where INT_REF_X is the MSB and INT_REF_Y is the LSB.

### 12.3.2 INT_REF_Y

| Register Address | Bank 1 - 0x01 (Hex) - 1 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

MSB of the reference for Y-axis. The actual reference is then computed as ***INT_REF_Y * 256***. When ***int_single_ref*** is set, the reference will be {INT_REF_X, INT_REF_Y}, where INT_REF_X is the MSB and INT_REF_Y is the LSB.

### 12.3.3 INT_DEB_X

| Register Address | Bank 1 - 0x03 (Hex) - 3 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

Rate interrupt duration reference for X-axis. This register determines how long (measured in number of samples) the selected AND/OR rate interrupt configuration has to stay asserted before the corresponding interrupt source bit is set and the interrupt on either INT1 or INT2 is eventually generated. When the selected AND/OR rate interrupt configuration is deasserted (goes to 0) the corresponding interrupt source bit is deasserted immediately, without any delay. The duration in seconds can be computed as ***Number of samples * ODR***.

### 12.3.4 INT_DEB_Y

| Register Address | Bank 1 - 0x04 (Hex) - 4 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R/W | | | | | | | |
| Default Value | 00000000 | | | | | | | |

#### *Description*

Rate interrupt duration reference for Y-axis. This register determines how long (measured in number of samples) the selected AND/OR rate interrupt configuration has to stay asserted before the corresponding interrupt source bit is set and the interrupt on either INT1 or INT2 is eventually generated. When the selected AND/OR rate interrupt configuration is deasserted (goes to 0) the corresponding interrupt source bit is deasserted immediately, without any delay. The duration in seconds can be computed as ***Number of samples * ODR***.

### 12.3.5 INT_MSK_X

| Register Address | Bank 1 - 0x06 (Hex) - 6 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***int_x_high_pos_en*** | ***int_x_low_pos_en*** | ***int_x_high_neg_en*** | ***int_x_low_neg_en*** | ***x_high_pos*** | ***x_low_pos*** | ***x_high_neg*** | ***x_low_neg*** |
| Type | R/W | R/W | R/W | R/W | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

Rate Interrupt, X-axis configuration register comprises 2 fields. The 4 LSBs are one-hot encoded bits which indicate whether the rate is positive or negative and whether it is above or below the rate threshold. The most significant 4 bits enable the interrupt bits when the corresponding condition is met.

#### *Fields*

***int_x_high_pos_en:*** Enable the int_x_high_pos interrupt generation for threshold event detection on X-axis.
***int_x_low_pos_en:*** Enable the int_x_low_pos interrupt generation for threshold event detection on X-axis.
***int_x_high_neg_en:*** Enable the int_x_high_neg interrupt generation for threshold event detection on X-axis.
***int_x_low_neg_en:*** Enable the int_x_low_neg interrupt generation for threshold event detection on X-axis.
***x_high_pos:*** Signal is positive, higher than threshold (see Interrupt Zones)
***x_low_pos:*** Signal is positive, lower than threshold
***x_high_neg:*** Signal is negative, higher than threshold
***x_low_neg:*** Signal is negative, lower than threshold

### 12.3.6 INT_MSK_Y

| Register Address | Bank 1 - 0x07 (Hex) - 7 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | *int_y_high_pos_en* | *int_y_low_pos_en* | *int_y_high_neg_en* | *int_y_low_neg_en* | *y_high_pos* | *y_low_pos* | *y_high_neg* | *y_low_neg* |
| Type | R/W | R/W | R/W | R/W | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

Rate Interrupt, Y-axis configuration register comprises 2 fields. The 4 LSBs are one-hot encoded bits which indicate whether the rate is positive or negative and whether it is above or below the rate threshold. The most significant 4 bits enable the interrupt bits when the corresponding condition is met.

#### *Fields*

***int_y_high_pos_en:*** Enable the int_y_high_pos interrupt generation for threshold event detection on Y-axis.
***int_y_low_pos_en:*** Enable the int_y_low_pos interrupt generation for threshold event detection on Y-axis.
***int_y_high_neg_en:*** Enable the int_y_high_neg interrupt generation for threshold event detection on Y-axis.
***int_y_low_neg_en:*** Enable the int_y_low_neg interrupt generation for threshold event detection on Y-axis.
***y_high_pos:*** Signal is positive, higher than threshold (see Interrupt Zones)
***y_low_pos:*** Signal is positive, lower than threshold
***y_high_neg:*** Signal is negative, higher than threshold
***y_low_neg:*** Signal is negative, lower than threshold

### 12.3.7 INT_MSK_AO

| Register Address | Bank 1 - 0x9 (Hex) - 9 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | RFU | *int_freeze* | *int_mask_xyz_and* | | | *int_mask_xyz_or* | | |
| Type | R | R/W | R/W | | | R/W | | |
| Default Value | 0 | 0 | 000 | | | 000 | | |

#### *Description*

Interrupt AND/OR masks register.

#### *Fields*

***int_freeze:*** Set the interrupt on threshold as latched. When enabled, all the rate interrupt flags are latched. When triggered, the interrupt is latched until the INT_MSK_{X,Y} register is read.
***int_mask_xyz_and:*** Each bit activates an axis. The active axes are ANDed together to generate the AND interrupt.
***int_mask_xyz_or:*** Each bit activates an axis. The active axes are ORed together to generate the OR interrupt.

### 12.3.8 INT_CFG1

| Register Address | Bank 1 - 0x0A (Hex) - 10 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***sns_intp_fsc*** | | ***int1_clk_out*** | ***int2_clk_out*** | ***int_single_deb*** | ***int_single_ref*** | ***sns_intp_cfg*** | |
| Type | R/W | | R/W | R/W | R/W | R/W | R/W | |
| Default Value | 00 | | 0 | 0 | 0 | 0 | 00 | |

#### *Description*

Interrupt 1 configuration register. In this register, the rate interrupt functionality configuration such as FS and data rate for all axes can be separately set from the output data signal path.

#### *Fields*

***sns_intp_fsc:*** Sets the FS used by the rate interrupts:
{ b00: 1000 dps; b01:500 dps; b10: 250 dps, b11: 125 dps}

***int1_clk_out:*** INT1 provides the internal clock (8.8 MHz), {0: Disable ; 1: Enable}
***int2_clk_out:*** INT2 provides the internal clock (8.8 MHz), {0: Disable ; 1: Enable}

***int_single_deb:*** When set, the same duration {INT_DEB_X, INT_DEB_Y} is used for all the axes, where INT_DEB_Y is the LSB
***int_single_ref:*** When set, the same threshold {INT_REF_X, INT_REF_Y} is used for all the axes.

***sns_intp_cfg:*** Configure the filtering and the ODR of the interrupt data:
- b00: data at ODR without HPF;
- b01: data at ODR with HPF;
- b10: data at 10 kHz without HPF;
- b11: data at 10 kHz with HPF

### 12.3.9 INT_CFG2

| Register Address | Bank 1 - 0x0B (Hex) - 11 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | RFU | | ***int1_enable*** | ***int1_active_level*** | ***int1_out_mode*** | ***int2_enable*** | ***int2_active_level*** | ***int2_out_mode*** |
| Type | R | | R/W | R/W | R/W | R/W | R/W | R/W |
| Default Value | 00 | | 1 | 0 | 0 | 1 | 0 | 0 |

#### *Description*

INT_CFG2 register controls both INT1/2 enable bits, push-pull/open-drain configuration and interrupt active levels. When the interrupts are disabled, they will remain inactive regardless of the settings at INT1_MSK and INT2_MSK.

#### *Fields*

***int1_enable:*** {0:Disable ;1: Enable} the INT1 register
***int1_active_level:*** 0: INT1 active high; 1: INT1 active low
***int1_out_mode:*** 0: Push-pull configuration;
1: Open drain configuration

***int2_enable:*** {0:Disable ;1: Enable} the INT2 register
***int2_active_level:*** 0: INT2 active high; 1: INT2 active low
***int2_out_mode:*** 0: Push-pull configuration;
1: Open drain configuration

### 12.3.10 INT_TMO

| Register Address | Bank 1 - 0x0C (Hex) - 12 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***int1_latch_mode*** | | ***int2_latch_mode*** | | ***int_timeout*** | | | |
| Type | R/W | | R/W | | R/W | | | |
| Default Value | 00 | | 00 | | 0000 | | | |

#### *Description*

Interrupt Timeout and Interrupt Mode configuration register allows to configure the interrupt lines to operate as either un-latched, latched or timed. As un-latched, they can be further configured in such a way that interrupt sources (INT1_STS and INT2_STS) can be cleared when they are read or cleared as they are written with a logic 1. Clearing an interrupt source by writing a logic 1 allows clearing single bits rather than the entire register.

#### *Fields*

***int1_latch_mode:***
- b00: INT1 is unlatched;
- b01: INT1 is latched, cleared on Read
- b10: INT1 is latched, cleared on Write (1);
- b11: INT1 is Timed (see ***int_timeout***)

***nt2_latch_mode:***
- b00: INT2 is unlatched;
- b01: INT2 is latched, cleared on Read
- b10: INT2 is latched, cleared on Write (1);
- b11: INT2 is Timed (see ***int_timeout***)

***int_timeout:*** Interrupt timeout period (shared between INT1 and INT2):

| | | | |
|---|---|---|---|
| b0000: 100 µs | b0011: 1 ms | b0110: 10 ms | b1001: 100 ms |
| b0001: 200 µs | b0100: 2 ms | b0111: 20 ms | b1010: 200 ms |
| b0010: 500 µs | b0101: 5 ms | b1000: 50 ms | b1011: 500 ms |

### 12.3.11 INT_STS_UL

| Register Address | Bank 1 - 0x0D (Hex) - 13 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***sts_ul_ data_ready*** | ***sts_ul_ fifo_empty*** | ***sts_ul_ fifo_overrun*** | ***sts_ul_ fifo_ths*** | ***sts_ul_ int_and*** | ***sts_ul_ int_or*** | ***sts_ul_otp_ downloading*** | ***sts_ul_ data_sync*** |
| Type | R | R | R | R | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

INT_STS_UL bits are the un-latched version of the interrupt status registers; as these signals are shared by INT1 and INT2 generator, there is a unique register shared. This register is the actual source for the interrupt lines when the interrupts are configured as un-latched. When the interrupt lines are configured as latched, be it both or just one of them, these bits can be used to keep monitoring the status of an interrupt source after the event.

#### *Fields*

***sts_ul_data_ready:*** DATA_READY status
***sts_ul_fifo_empty:*** FIFO_EMPTY status
***sts_ul_fifo_overrun:*** FIFO_OVERRUN status
***sts_ul_fifo_ths:*** FIFO_THRESHOLD status
***sts_ul_int_and:*** Rate interrupt AND status
***sts_ul_int_or:*** Rate interrupt OR status
***sts_ul_otp_downloading:*** OTP_DOWNLOADING status
***sts_ul_data_sync:*** DSYNC status

### 12.3.12 INT1_STS

| Register Address | Bank 1 - 0x0E (Hex) - 14 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***sts_i1_ data_ready*** | ***sts_i1_ fifo_empty*** | ***sts_i1_ fifo_overrun*** | ***sts_i1_ fifo_ths*** | ***sts_i1_ int_and*** | ***sts_i1_ int_or*** | ***sts_i1_otp_ downloading*** | ***sts_i1_ data_sync*** |
| Type | R | R | R | R | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

INT1_STS bits are the latched interrupt sources; when the latched mode is used, they can be cleared either by reading INT1_STS register, or writing these bits as 1. Bits can be cleared at the same time by forming the proper mask. When INT1 is configured as unlatched or timed, these registers are set as 0.

#### *Fields*

***sts_i1_data_ready:*** DATA_READY status
***sts_i1_fifo_empty:*** FIFO_EMPTY status
***sts_i1_fifo_overrun:*** FIFO_OVERRUN status
***sts_i1_fifo_ths:*** FIFO_THRESHOLD status
***sts_i1_int_and:*** Rate interrupt AND status
***sts_i1_int_or:*** Rate interrupt OR status
***sts_i1_otp_downloading:*** OTP_DOWNLOADING status
***sts_i1_data_sync:*** DSYNC status

### 12.3.13 INT2_STS

| Register Address | Bank 1 - 0x0F (Hex) - 15 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***sts_i2_ data_ready*** | ***sts_i2_ fifo_empty*** | ***sts_i2_ fifo_overrun*** | ***sts_i2_ fifo_ths*** | ***sts_i2_ int_and*** | ***sts_i2_ int_or*** | ***sts_i2_otp_ downloading*** | ***sts_i2_ data_sync*** |
| Type | R | R | R | R | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

INT2_STS bits are the latched interrupt sources; when the latched mode is used, they can be cleared either by reading INT2_STS register, or writing these bits as 1. Bits can be cleared at the same time by forming the proper mask. When INT2 is configured as unlatched or timed, these registers are set as 0.

#### *Fields*

***sts_i2_data_ready:*** DATA_READY status
***sts_i2_fifo_empty:*** FIFO_EMPTY status
***sts_i2_fifo_overrun:*** FIFO_OVERRUN status
***sts_i2_fifo_ths:*** FIFO_THRESHOLD status
***sts_i2_int_and:*** Rate interrupt AND status
***sts_i2_int_or:*** Rate interrupt OR status
***sts_i2_otp_downloading:*** OTP_DOWNLOADING status
***sts_i2_data_sync:*** DSYNC status

### 12.3.14 INT1_MSK

| Register Address | Bank 1 - 0x10 (Hex) - 16 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***msk_i1_ data_ready*** | ***msk_i1_ fifo_empty*** | ***msk_i1_ fifo_overrun*** | ***msk_ i1_ fifo_ths*** | ***msk_ i1_ int_and*** | ***msk_ i1_ int_or*** | ***msk_i1_ otp_ downloading*** | ***msk_i1_ data_sync*** |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Default Value | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

#### *Description*

Interrupt 1 generation, mask register is used to enable selected sources in the INT1_STS register to activate the INT1 interrupt line. Interrupt sources are masked (prevented from generating an interrupt) as long as the corresponding bit in the mask register is 0. Valid configurations are with None, One, Multiple or Every bit set to 1; having multiple possible interrupt sources will require the Interrupt Service routine to identify the correct one(s).

#### *Fields*

***msk_i1_data_ready:*** DATA_READY status
***msk _i1_fifo_empty:*** FIFO_EMPTY status
***msk _i1_fifo_overrun:*** FIFO_OVERRUN status
***msk _i1_fifo_ths:*** FIFO_THRESHOLD status
***msk _i1_int_and:*** Rate interrupt AND status
***msk _i1_int_or:*** Rate interrupt OR status
***msk_i1_otp_downloading:*** OTP_DOWNLOADING status
***msk _i1_data_sync:*** DSYNC status

### 12.3.15 INT2_MSK

| Register Address | Bank 1 - 0x11 (Hex) - 17 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | ***msk_i2_ data_ready*** | ***msk_i2_ fifo_empty*** | ***msk_i2_ fifo_overrun*** | ***msk_ i2_ fifo_ths*** | ***msk_ i2_ int_and*** | ***msk_ i2_ int_or*** | ***msk_i2_ otp_ downloading*** | ***msk_i2_ data_sync*** |
| Type | R | R | R | R | R | R | R | R |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

#### *Description*

Interrupt 2 generation, mask register is used to enable selected sources in the INT2_STS register to activate the INT2 interrupt line. Interrupt sources are masked (prevented from generating an interrupt) as long as the corresponding bit in the mask register is 0. Valid configurations are with None, One, Multiple or Every bit set to 1; having multiple possible interrupt sources will require the Interrupt Service routine to identify the correct one(s).

#### *Fields*

***msk_i2_data_ready:*** DATA_READY status
***msk _i2_fifo_empty:*** FIFO_EMPTY status
***msk _i2_fifo_overrun:*** FIFO_OVERRUN status
***msk _i2_fifo_ths:*** FIFO_THRESHOLD status
***msk _i2_int_and:*** Rate interrupt AND status
***msk _i2_int_or:*** Rate interrupt OR status
***msk_i2_otp_downloading:*** OTP_DOWNLOADING status
***msk _i2_data_sync:*** DSYNC status

### 12.3.16 OTP_STATUS

| Register Address | Bank 1 - 0x19 (Hex) - 24 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | RFU | | | | | | *ecc_stat* | |
| Type | R | | | | | | R | |
| Default Value | 000000 | | | | | | 00 | |

#### *Description*

OTP_STATUS register provides the OTP download status.

#### *Fields*

***ecc_stat:***
- b00: completed successfully
- b01: completed successfully, 1 bit corrected
- b10: completed successfully, after some attempts
- b11: completed with errors

### 12.3.17 SILICON_REV_OTP

| Register Address | Bank COMMON - 0x20 (Hex) - 32 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | 0x2 | | | | | | | |

#### *Description*

SILICON_REV_OTP identifies the MAX21002 silicon revision.

### 12.3.18 SERIAL_[0:5]

| Register Address | Bank 0 - 0x[1A:1F] (Hex) – 26:31 (Dec) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bit # | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Fields | | | | | | | | |
| Type | R | | | | | | | |
| Default Value | N/A | | | | | | | |

#### *Description*

SERIAL_0, SERIAL_1, SERIAL_2, SERIAL_3, SERIAL_4 and SERIAL_5 are 6 registers used to assign a unique identifier to every single MAX21002 sample to enable a complete tracability in terms of LOTs, Assembly history and Test equipment.

# 13 Definitions

**Power supply [V]**: This parameter defines the operating DC power supply voltage range of the MEMS gyroscope. Although it is always a good practice to keep $V_{DD}$ clean with minimum ripple, unlike most of the competitors, who require an ultra-low noise, low-dropout regulator to power the MEMS gyroscope, the MAX21002 can not only operate at 1.71V but that supply can also be provided by a switching regular, to minimize the system power consumption.

**Power supply current [mA]:** This parameter defines the typical current consumption when the MEMS gyroscope is operating in normal mode.

**Power supply current in Standby mode [mA]:** This parameter defines the current consumption when the MEMS gyroscope is in Standby mode. To reduce power consumption and have a faster turn-on time, in Standby mode only an appropriate subset of the sensor is turned off.

**Power supply current in Low Power Mode [mA]:** This parameter defines the current consumption when the MEMS gyroscope is in a special mode named Low Power Mode. Whilst in Low Power Mode, the MAX21002 reduces significantly the power consumption, at the price of a slightly higher RND.

**Power supply current in stand-by mode [µA]:** This parameter defines the current consumption when the MEMS gyroscope is powered down. In this mode, both the mechanical sensing structure and reading chain are turned off. Users can configure the control register through the $I^2C$/SPI interface for this mode. Full access to the control registers through the $I^2C$/SPI interface is guaranteed also in power-down mode.

**Full-scale range [dps]:** This parameter defines the measurement range of the gyroscope in degrees per second (dps). When the applied angular velocity is beyond the full-scale range, the gyroscope output signal will be saturated.

**Zero-rate level [LSBs]:** This parameter defines the zero rate level when there is no angular velocity applied to the gyroscope.

**Sensitivity [mdps/LSB]:** Sensitivity (mdps/LSB) is the relationship between 1 LSB and milli-dps. It can be used to convert a digital gyroscope's measurement in LSBs to angular velocity.

**Sensitivity change vs. Temperature [%/°C]:** This parameter defines the sensitivity change in percentage (%) over the operating temperature range specified in the datasheet.

**Zero-rate level change vs. Temperature [dps/°C]:** This parameter defines the zero rate level change in degree per second (dps/°C) over the operating temperature range.

**Non-linearity [% FS]:** This parameter defines the maximum error between the gyroscope's outputs and the best-fit straight line in percentage with respect to the full-scale (FS) range.

**System bandwidth [Hz]:** This parameter defines the frequency of the angular velocity signal from DC to the built-in bandwidth (BW) that the gyroscopes can measure. A dedicated register can be modified to adjust the gyroscope's bandwidth.

**Rate noise density [dps/√Hz]:** This parameter defines the standard resolution that users can get from the gyroscopes outputs together with the BW parameter.

**Self-test [dps]:** This feature can be used to verify if the gyroscope is working properly or in order to not physically rotate the gyroscope after it is assembled on a PCB. When the self-test is enabled, an internal electrostatic force is generated to move the masses to simulate the Coriolis Effect. If the gyroscope's outputs are within the specified self-test values in the data sheet, then the gyroscope is working properly. Therefore, the self-test feature is an important consideration in a user's end-product mass production line.