Pawel Gburzynski

MPU9250 accelerometer/gyro/compass

(preliminary) driver

September 6, 2017

© Copyright 2017, Olsonet Communications Corporation.
All Rights Reserved.

Introduction

MPU9250 is an accelerometer/gyroscope/compass combo manufactured by InvenSense Inc. Documentation available from the manufacturer (data sheet + specification) covers all the technical intricacies. That documentation is not absolutely required to understand how to use the driver and what you can expect from the sensor (under the present driver).

The sensor optionally offers high-level functionality known as DMP (Digital Motion Processing). That functionality is not implemented in the present driver. To implement it requires some reverse engineering, but it can be done, should a need arise. It is a high-power, high-cpu-load function relying on special code loaded into the chip which then requires special processing in software. The manufacturer has provided source code which, I think (with some effort), can be adapted for PicOS.

Another feature that is excluded from the present driver, but can be easily added in the future (this one truly easily) is FIFO access, i.e., high-speed data acquisition, e.g., for high-bandwidth, real-time analysis of motion, gesture recognition, dead reckoning, and so on. The feature is mostly excluded, because it requires a different interface than our standard **`read_sensor`** operation intended for low-bandwidth data extraction, a few bytes at a time.

Summary of the sensor's functionality

The sensor has two basic configurable modes of operation:

1. low power (LP) motion detection mode (triggering motion events)
2. measurement mode (no events, on demand data acquisition)

Only the accelerometer is available in the LP motion detection mode. Besides receiving events on exceeded acceleration thresholds, the application can also read the acceleration values at any time.

In the second mode, the set of sensors is configurable and can comprise any combination of these options:

- acceleration
- gyro
- compass
- temperature

The last option is the free (die) temperature readout that can be obtained from the sensor. For example, when acceleration and temperature have been selected from the above options, the sensor *reading* (as returned by the standard **`read_sensor`** operation) will consists of four numbers: the three-element acceleration vector + the temperature reading. With the complete set of selections, a single sensor reading consists of 10 numbers (3 vectors and 1 scalar)

In the LP motion detection mode, a standard sensor event is triggered when the acceleration (along any of the three axes) exceeds a declared threshold.

For acceleration, the measurement range, aka FSR [Full Scale Range] comes in four options: ±2g, ±4g, ±8g, ±16g. For the gyroscope, the options are: ±250, ±500, ±1000, and ±2000 degrees per second. For the compass (magnetometer) the (immutable)

FSR is ±4800μT (microtesla). The resolution is 15 bits in all cases, i.e., each value is returned as a 16-bit signed integer.

Two more configuration parameters for the driver are:

- low pass filter setting (for damping rapid changes): 8 levels
- wakeup rate in the LP motion-detection mode: 12 levels

The second parameter only applies to the accelerometer operating in the LP motion detection mode. The LPF bandwidth selection applies to acceleration and gyro. For the compass, each readings stands for itself and is taken exactly at the time of issue of the **`read_sensor`** function.

Driver interface

The sensor must be explicitly turned on, which operation determines its mode and parameters. This is accomplished by invoking:

```
mpu9250_on (word options, byte threshold);
```

where **`options`** is a collection of fields composed by or'ring a few constants defined in mpu9250.h. In particular, the LPF setting is determined by one of these constants:

Constant name	Accel (Hz)	Gyro (Hz)	Temperature (Hz)
`MPU9250_LPF_256`	460	250	4000
`MPU9250_LPF_188`	184	184	188
`MPU9250_LPF_98`	92	92	98
`MPU9250_LPF_42`	41	41	42
`MPU9250_LPF_20`	20	20	20
`MPU9250_LPF_10`	10	10	10
`MPU9250_LPF_5`	5	5	5
`MPU9250_LPF_2100`	460	3600	4000

The wakeup rate in LP mode can be any of these:

Constant name	Rate (Hz)
`MPU9250_LPA_02`	0.24
`MPU9250_LPA_05`	0.49
`MPU9250_LPA_1`	0.98
`MPU9250_LPA_2`	1.95
`MPU9250_LPA_4`	3.91
`MPU9250_LPA_8`	7.81
`MPU9250_LPA_16`	15.63
`MPU9250_LPA_32`	31.25
`MPU9250_LPA_64`	62.5
`MPU9250_LPA_128`	125

MPU9250_LPA_256	250
MPU9250_LPA_512	500

The configuration of sensors is described by these constants:

```
MPU9250_SEN_ACCEL
MPU9250_SEN_GYRO
MPU9250_SEN_COMPASS
MPU9250_SEN_TEMP
```

The ranges (FSR) for the accelerometer and the gyro are:

```
MPU9250_ACCEL_RANGE_2
MPU9250_ACCEL_RANGE_4
MPU9250_ACCEL_RANGE_8
MPU9250_ACCEL_RANGE_16

MPU9250_GYRO_RANGE_250
MPU9250_GYRO_RANGE_500
MPU9250_GYRO_RANGE_1000
MPU9250_GYRO_RANGE_2000
```

One more flag specifying LP motion detection is **MPU9250_LP_MOTION_DETECT**.

The **threshold** argument of **mpu9250_on** provides the threshord value for motion detection and is only interpreted when **MPU9250_LP_MOTION_DETECT** is on. The value is in 4mg increments with the range from 0 to 1020 mg. Note that its interpretation doesn't depend on the FSR setting.

For example:

```
mpu9250_on ( MPU9250_LP_MOTION_DETECT     +
             MPU9250_SEN_ACCEL            +
             MPU9250_ACCEL_RANGE_4        +
             MPU9250_LPF_188              +
             MPU9250_LPA_4,
                   32);
```

sets up the device for LP motion detection with the filter bandwidth of 184 Hz, with the wakeup frequency of 3.91 Hz, and the threshold of 32 x 4 = 128 mg. Note that the FSR of the accelerometer is only applicable to readouts of accelereration values, not to the comparison against the threshold. The **MPU9250_LPA_4** selection is superfluous: the LP motion detect flag includes its. Any sensor selections beyond the accelerometer made together with **MPU9250_LP_MOTION_DETECT** would be ignored.

The sensor is stopped by invoking:

```
mpu9250_off ();
```

On the systems where the sensor is powered from a µC pin (e.g., CC1350 SENSORTAG), this will power the sensor down completely. Otherwise, the sensor is put into the sleep mode.

Reading values

The sensor value can be read in the standard way:

```
read_sensor (word st, address val);
```

The state argument is ignored (the sensor is always ready for data extraction). The second argument should point to an array of words whose size depends on the selectted sensors. For example, for the accelerometer alone (e.g., in the LP motion detection mode), the value amounts to a 3-element vector (three words are needed to accommodate it). For the accelerometer + gyro, the size is two vectors, i.e., 6 words. The temperature readout is the only scalar in the set taking just one word.

The ordering of values in the **`val`** array is: accelerometer, gyro, compass, temperature (the ones not selected are skipped). The vector components are signed 2's complement numbers with the granularity of $1/2^{15}$ of the full scale range. I couldn't find the temperature conversion formula anywhere in the data sheet. It only says:

Temp_degC = (TEMP_OUT – RoomTemp_Offset) Temp_Sensitivity) + 21

without defining the constants.

Receiving events

As usual:

```
wait_sensor (SENSOR_MPU9250, TARGET_STATE);
```

Normally, the function immediately blocks (forces release), so it should be the last statement in the current state. The issuing FSM is suspended and will be resumed in **`TARGET_STATE`** when the event (motion) occurs. Needless to say, the FSM can issue other wait requests before calling **`wait_sensor`**.

The event causing **`wait_sensor`** to unblock does not go away until you execute **`read_sensor`** to retrieve the sensor data. This means that a subsequent **`wait_sensor`**, executed before **`read_sensor`**, will fire immediately.