

MPU6050-BB

Version 1.0



Table of Contents

1	Introduction.....	2
2	Board Features.....	2
3	Specifications.....	2
3.1	PCB Details.....	2
4	Hardware Connections.....	3
5	Pseudo Code.....	5
5.1	Register Addresses and Configuration Parameters	5
5.2	Accessing Registers and Configuring Device.....	6
5.3	Reading Acceleration, Rotation and Temperature Values	7
6	Reference	8





1 Introduction

The MPU6050-BB is a breakout board for MPU6050 which is a small integrated 6-axis motion tracking device that combines 3-axis accelerometer and 3-axis gyroscope. MPU6050 is a very advanced chip which has an inbuilt accelerometer, a gyro and a temperature sensor. MPU6050 can also control external magnetometer and other sensors. The on-chip Digital Motion Processor™ (DMP™) is capable of processing complex 9-axis Motion Fusion algorithms. Hence system processor can offload intensive motion processing computations, like cross-axis alignment, to MPU6050. This minimizes the need for frequent polling of the motion sensor output. MPU6050 motion processing unit is the world's first motion processing solution with integrated 9-axis sensor fusion using its field-proven and proprietary Motion Fusion™ engine. When connected to a 3-axis magnetometer, the MPU6050 delivers a complete 9-axis Motion Fusion output to its primary I2C interface. The MPU6050 combines acceleration, rotational motion and heading information into a single data stream for applications. Additional features include an embedded temperature sensor and an on-chip oscillator with $\pm 1\%$ variation over the operating temperature range. The module can be used in motion gaming application and also in wearable sensors for health fitness and sports.

2 Board Features

- Small size, easy mounting and ease of use
- Optional crystal (on-board) of 32.768 KHz
- I2C interface
- On-board 3.3V regulator
- 3 pin berg connector for auxiliary I2C pin
- 10 pin connector for I2C pins, power supply and interrupt

3 Specifications

- Crystal frequency: 32.768 KHz
- Input power supply: 3.3V
- Gyroscope operating current: 3.6 mA
- Accelerometer normal operating current: 500 μ A
- Tri-axis accelerometer with a programmable full scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$ and $\pm 16g$
- Tri-axis angular rate sensor (gyro) with a sensitivity scaling up to 131 LSB/dps and a full-scale range of ± 250 , ± 500 , ± 1000 and ± 2000 dps

3.1 PCB Details

- PCB size: 25 mm x 25 mm
- PCB type: FR4
- Board thickness: 1.6 mm
- Solder mask: Black
- Surface finish: Immersion gold



4 Hardware Connections

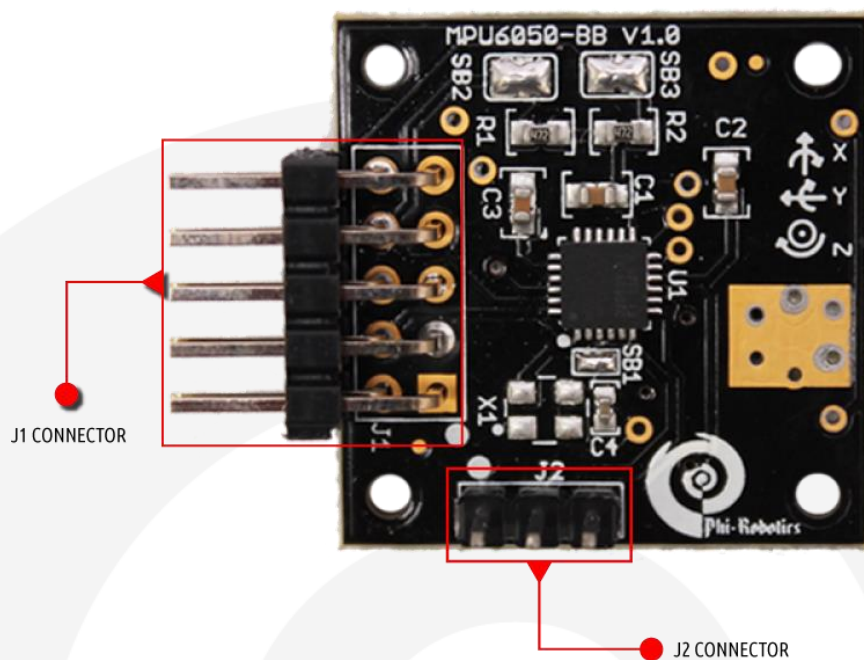


Figure 1 - MPU6050-BB top view

MPU6050-BB has two connectors. The J1 is a 10 pin header for interfacing MPU6050-BB with microcontroller using I2C bus. And J2 is a 3 pin header, provided for communication with an external 3-axis digital output magnetometer or other sensors using auxiliary I2C bus. JP1 jumper is used for selecting the I2C slave address of the MPU6050. When JP1 is connected, the slave address is 0xD0 and if it is not connected, the address is 0xD2.

J1 header contains supply pins, I2C communication pins, interrupt and synchronization pins. FSYNC pin is used for frame synchronization with camera or GPS. Figure 2 below shows pin layout of J1.

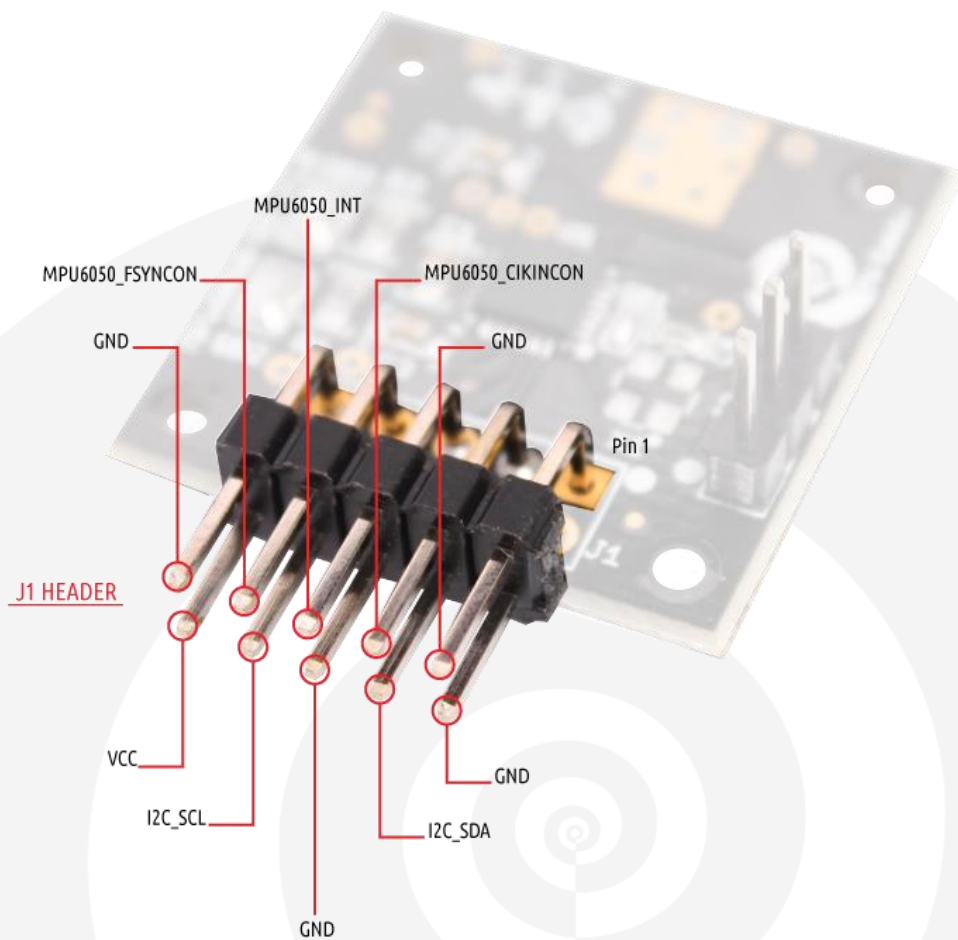


Figure 2 - MPU6050-BB J1 pin layout

J2 header is used to interface external sensors using MPU6050 auxiliary I2C bus. Figure 3 below shows pin layout for J2.

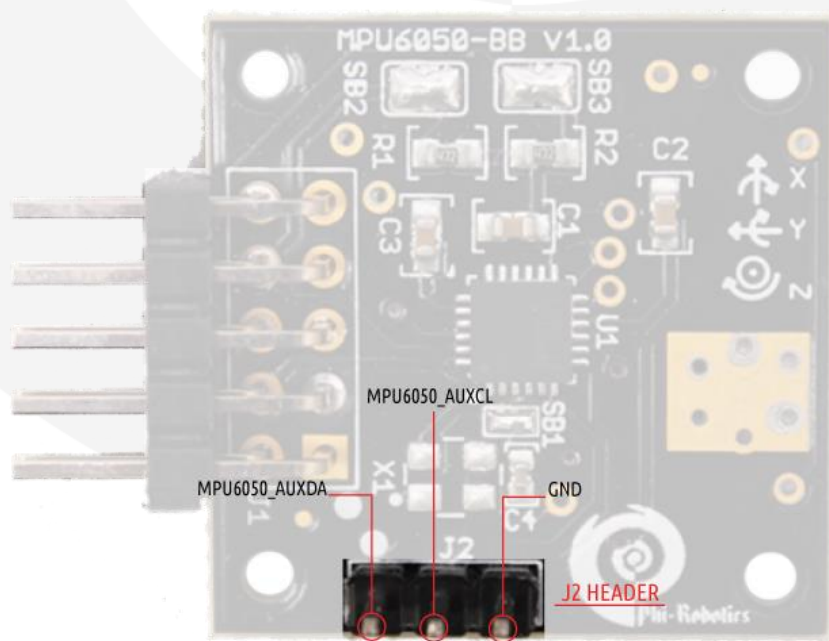


Figure 3 - MPU6050-BB J2 pin layout

5 Pseudo Code

5.1 Register Addresses and Configuration Parameters

```
// MPU6050 I2C slave address
MPU_SLAVE_ADDR = 0xD2

// MPU6050 register addresses
MPU_REG_SMPLRT_DIV = 0x19
MPU_REG_CONFIG = 0x1A
MPU_REG_GYRO_CONFIG = 0x1B
MPU_REG_ACCEL_CONFIG = 0x1C
MPU_REG_INT_PIN_CFG = 0x37
MPU_REG_INT_ENABLE = 0x38
MPU_REG_USER_CTRL = 0x6A
MPU_REG_SIGNAL_PATH_RESET = 0x68
MPU_REG_PWR_MGMT_1 = 0x6B
MPU_REG_ACCEL_XOUT_H = 0x3B
MPU_REG_GYRO_XOUT_H = 0x43
MPU_REG_TEMP_OUT_H = 0x41

// MPU6050 configuration parameters
// sample rate divider, to set the sample rate of the sensor
MPU_SAMPLE_RATE_DIV = 0x07 // to generate the desired Sample Rate for MPU
// external FSYNC pin sampling
MPU_EXT_SYNC = 0
// digital low pass filter bandwidth
MPU_DLP_BW = 0
// gyroscope full scale range
MPU_GYRO_FS_RANGE = 0x18 // full scale range =  $\pm 1000$  °/s
// accelerometer full scale range
MPU_ACC_FS_RANGE = 0x18 // full Scale Range =  $\pm 16$  °/s
// interrupt status bit clear by read operation
MPU_INT_STAT_CLEAR = 0x10 // enable
// set FSYNC pin active logic level
MPU_FSYNC_LOGIC_LEVEL = 0x80 // active low
// set aux I2C bus access for host
MPU_I2C_BYPASS = 0x20 // enable
// enable interrupts
MPU_INT_ENABLE = 0x59 // enabled interrupts: motion detection,
// FIFO overflow,
// I2C master,
// data ready

// clock selection
MPU_CLK_SEL = 0 // internal 8MHz oscillator

// gyroscope scaling factor. This depends on MPU_GYRO_FS_RANGE
MPU_GYRO_SCALE_FACTOR = 0.060975
// accelerometer scaling factor. This depends on MPU_ACC_FS_RANGE
MPU_ACC_SCALE_FACTOR = 0.488281
```



5.2 Accessing Registers and Configuring Device

```
uint8_t mpuRegRead(uint8_t regAddr)
{
    uint8_t data;
    // I2C start signal
    i2cStart();
    // send MPU I2C slave address, R/W bit set as 0
    i2cWriteByte(MPU_SLAVE_ADDR);
    // send register address to read
    i2cWriteByte(regAddr);
    // I2C repeated start signal
    i2cStart();
    // send MPU I2C address, R/W bit set as 1
    i2cWriteByte(MPU_SLAVE_ADDR | 0x01);
    // read a byte from I2C
    data = i2cReadByte(1);
    // I2C stop signal
    i2cStop();
    return data;
}

void mpuRegWrite(uint8_t regAddr, uint8_t data)
{
    i2cStart();
    // send MPU I2C slave address, R/W bit set as 1
    i2cWriteByte(MPU_SLAVE_ADDR | 0x01);
    // send register address to write
    i2cWriteByte(regAddr);
    // write data byte
    i2cWriteByte(data);
    i2cStop();
}

void mpuInit()
{
    // set sampling rate
    mpuRegWrite(MPU_REG_SMPLRT_DIV, MPU_SAMPLE_RATE_DIV);
    // FSYNC and digital low pass filter settings
    mpuRegWrite(MPU_REG_CONFIG, (MPU_EXT_SYNC | MPU_DLP_BW) );
    // set gyroscope full scale range
    mpuRegWrite(MPU_REG_GYRO_CONFIG, MPU_GYRO_FS_RANGE);
    // set accelerometer full scale range
    mpuRegWrite(MPU_REG_ACCEL_CONFIG, MPU_ACC_FS_RANGE);

    // MPU control functions
    // set interrupt clear option, FSYNC logic level, aux bus access
    mpuRegWrite(MPU_REG_INT_PIN_CFG, (MPU_INT_STAT_CLEAR
                                      | MPU_FSYNC_LOGIC_LEVEL | MPU_I2C_BYPASS));

    // enable interrupts
    mpuRegWrite(MPU_REG_INT_ENABLE, MPU_INT_ENABLE);
    // configure MPU hardware FIFO
    mpuRegWrite(MPU_REG_USER_CTRL, 0x40);
    // reset the analog and digital signal paths of all on-chip sensors
    mpuRegWrite(MPU_REG_SIGNAL_PATH_RESET, 0x07);
    // CLKSEL is a 3-bit unsigned value specifying
    // the clock source of the device
    mpuRegWrite(MPU_REG_PWR_MGMT_1, MPU_CLK_SEL);
}
```





5.3 Reading Acceleration, Rotation and Temperature Values

```
void mpuGetAcceleration(int32_t* x, int32_t* y, int32_t* z)
{
    uint8_t data[6]={0};
    uint32_t temp;

    /*
    all 6 acceleration data registers are in sequence,
    these registers should be read in a single read sequence
    to avoid change in value of an axis while other axis data
    is being read
    */
    i2cStart();
    i2cWriteByte(MPU_SLAVE_ADDR);
    i2cWriteByte(MPU_REG_ACCEL_XOUT_H);
    i2cStart();
    i2cWriteByte(MPU_SLAVE_ADDR | 0x01);
    //read 6 data registers
    data = i2cReadByte(6);
    i2cStop();

    // calculate acceleration values
    temp = ((data[0] << 8) | data[1]);
    *x = temp * MPU_ACC_SCALE_FACTOR;

    temp = ((data[2] << 8) | data[3]);
    *y = temp * MPU_ACC_SCALE_FACTOR;

    temp = ((data[4] << 8) | data[5]);
    *z = temp * MPU_ACC_SCALE_FACTOR;
}

void mpuGetRotation(int32_t* x, int32_t* y, int32_t* z)
{
    uint8_t data[6]={0};
    uint32_t temp;


    // read all gyroscope register in a single read sequence
    i2cStart();
    i2cWriteByte(MPU_SLAVE_ADDR);
    i2cWriteByte(MPU_REG_GYRO_XOUT_H);
    i2cStart();
    i2cWriteByte(MPU_SLAVE_ADDR | 0x01);
    data = i2cReadByte(6);
    i2cStop();

    // calculate axis rotation values in °/sec
    temp = (data[0] << 8) | data[1];
    *x = ((*x) * MPU_GYRO_SCALE_FACTOR);

    temp = (data[2] << 8) | data[3];
    *y = temp * MPU_GYRO_SCALE_FACTOR;

    temp = (data[4] << 8) | data[5];
    *z = temp * MPU_GYRO_SCALE_FACTOR;
}
```





```
int16_t mpuGetTemperature()
{
    int16_t temperature;
    int16_t calc = 0;
    uint8_t data[2]={0};

    data[0] = mpuRegRead(MPU_REG_TEMP_OUT_H);
    data[1] = mpuRegRead(MPU_REG_TEMP_OUT_L);

    calc = (data[0] << 8) | dataBuf[1];
    //calculate temperature in °C
    temperature = (calc / 340.0) + 36.53 ;


    return temperature;
}
```

6 Reference

MPU6050 Datasheet: <http://invensense.com/mems/qyro/documents/PS-MPU-6000A-00v3.4.pdf>

MCP1826 Datasheet: <http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en531455>





Phi Robotics Research Pvt. Ltd.

www.phi-robotics.com