```
1.  //  技术文档未公布的寄存器    主要用于官方  DMP操作
2.  #define MPU6050_RA_XG_OFFS_TC       0x00          //[bit7] PWR_MODE, [6:1] XG_OFFS_TC, [bit 0]
    OTP_BNK_VLD
3.  #define MPU6050_RA_YG_OFFS_TC       0x01          //[7] PWR_MODE, [6:1] YG_OFFS_TC, [0] OTP_BNK_VLD
4.  //bit7      的定义 , 当设置为  1, 辅助 I2C  总线高电平是  VDD。当设置为  0, 辅助  I2C  总线高电平是  VLOGIC
5.
6.  #define MPU6050_RA_ZG_OFFS_TC       0x02          //[7] PWR_MODE, [6:1] ZG_OFFS_TC, [0] OTP_BNK_VLD
7.  #define MPU6050_RA_X_FINE_GAIN      0x03          //[7:0] X_FINE_GAIN
8.  #define MPU6050_RA_Y_FINE_GAIN      0x04          //[7:0] Y_FINE_GAIN
9.  #define MPU6050_RA_Z_FINE_GAIN      0x05          //[7:0] Z_FINE_GAIN
10.
11. #define MPU6050_RA_XA_OFFS_H        0x06          //[15:0] XA_OFFS        两个寄存器合在一起
12. #define MPU6050_RA_XA_OFFS_L_TC     0x07
13.
14. #define MPU6050_RA_YA_OFFS_H        0x08          //[15:0] YA_OFFS        两个寄存器合在一起
15. #define MPU6050_RA_YA_OFFS_L_TC     0x09
16.
17. #define MPU6050_RA_ZA_OFFS_H        0x0A          //[15:0] ZA_OFFS        两个寄存器合在一起
18. #define MPU6050_RA_ZA_OFFS_L_TC     0x0B
19.
20. #define MPU6050_RA_XG_OFFS_USRH     0x13          //[15:0] XG_OFFS_USR        两个寄存器合在一起
21. #define MPU6050_RA_XG_OFFS_USRL     0x14
22.
23. #define MPU6050_RA_YG_OFFS_USRH     0x15          //[15:0] YG_OFFS_USR        两个寄存器合在一起
24. #define MPU6050_RA_YG_OFFS_USRL     0x16
25.
26. #define MPU6050_RA_ZG_OFFS_USRH     0x17          //[15:0] ZG_OFFS_USR        两个寄存器合在一起
27. #define MPU6050_RA_ZG_OFFS_USRL     0x18
28.
29.  / * 陀螺仪的采样频率  */
30.  / * 传感器的寄存器输出  ,FIFO   输出 ,DMP 采样、运动检测、
31.    * 零运动检测和自由落体检测都是基于采样率。
32.    * 通过  SMPLRT_DIV 把陀螺仪输出率分频即可得到采样率
33.    * 采样率 =陀螺仪输出率  / (1 + SMPLRT_DIV)
34.    * 禁用  DLPF 的情况下 (DLPF_CFG = 0    或 7)   ,   陀螺仪输出率  = 8 khz
```

```
35.  * 在启用 DLPF( 见寄存器 26) 时，陀螺仪输出率 = 1 khz
36.  * 加速度传感器输出率是 1 khz 。这意味着 , 采样率大于 1 khz 时,
37.  * 同一个加速度传感器的样品可能会多次输入到 FIFO 、DMP和传感器寄存器 */
38. #define MPU6050_RA_SMPLRT_DIV       0x19        //[0-7]       陀螺仪输出分频采样率
39.
40. / * 配置外部引脚采样和 DLPF 数字低通滤波器 */
41. #define MPU6050_RA_CONFIG           0x1A
42. / /bit5-bit3          一个连接到 FSYNC端口的外部信号可以通过配置 EXT_SYNC_SET来采样
43. / /                   也就是说 , 这里设置之后 ,FSYNC 的电平 0 或 1 进入最终数据寄存器 , 具体如下
44. / /                   0 不使用 1 FSYNC 电平进入所有数据寄存器 2 FSYNC 电平进入 GYRO_XOUT_L 3 FSYNC电平进入
    GYRO_YOUT_L
45. / /                   4 FSYNC电平进入 GYRO_ZOUT_L5 FSYNC电平进入 ACCEL_XOUT_L6 FSYNC电平进入 ACCEL_YOUT_L
46. / /                   7 FSYNC 电平进入 SYNC_ACCEL_ZOUT_L
47. / /bit2-bit0          数字低通滤波器 用于滤除高频干扰 高于这个频率的干扰被滤除掉
48. / * 对应关系如下
49.  * *                  |       加速度传感器 |                  陀螺仪
50.  * * DLPF_CFG |          带宽 |       延迟 |        带宽 |       延迟 |     采样率
51.  * ------------+--------+-------+--------+------+-------------
52.  * 0              | 260Hz     | 0ms    | 256Hz   | 0.98ms | 8kHz
53.  * 1              | 184Hz     | 2.0ms  | 188Hz   | 1.9ms  | 1kHz
54.  * 2              | 94Hz      | 3.0ms  | 98Hz    | 2.8ms  | 1kHz
55.  * 3              | 44Hz      | 4.9ms  | 42Hz    | 4.8ms  | 1kHz
56.  * 4              | 21Hz      | 8.5ms  | 20Hz    | 8.3ms  | 1kHz
57.  * 5              | 10Hz      | 13.8ms | 10Hz    | 13.4ms | 1kHz
58.  * 6              | 5Hz       | 19.0ms | 5Hz     | 18.6ms | 1kHz
59.  * 7              | Reserved  | Reserved | Reserved
60.  * */
61.
62.
63. / * 陀螺仪的配置，主要是配置陀螺仪的量程与自检 ( 通过相应的位 7 6 5 开启自检 )*/
64. #define MPU6050_RA_GYRO_CONFIG      0x1B
65. / /bit4-bit3          量程设置如下
```

66. / / 0 = +/- 250 度 / 秒

67. / / 1 = +/- 500 度 / 秒

68. / / 2 = +/- 1000 度/ 秒

69. / / 3 = +/- 2000 度/ 秒*/

70.

71. / * 加速度计的配置，主要是配置加速度计的量程与自检 （通过相应的位 7 6 5 开启自检)

72. * 另外 , 还能配置系统的高通滤波器 */

73. #define MPU6050_RA_ACCEL_CONFIG 0x1C

74. / /bit7 启动 X 自检 加速度计的自检

75. / /bit6 启动 Y 自检

76. / /bit5 启动 Z 自检

77. / /bit4-bit3 加速度传感器的量程配置

78. / / 0 = +/- 2g

79. / / 1 = +/- 4g

80. / / 2 = +/- 8g

81. / / 3 = +/- 16g*/

82. / /bit0 到 bit2 加速度传感器的高通滤波器

83. / *DHPF 是在路径中连接于运动探测器 （自由落体 , 运动阈值 , 零运动) 的一个滤波器模块。

84. * 高通滤波器的输出值不在数据寄存器中

85. * 高通滤波器有三种模式：

86. * 重置 : 在一个样本中将滤波器输出值设为零。 这有效的禁用了高通滤波器。 这种模式可以快速切换滤波器的设置

    模式。

87. * 开启 : 高通滤波器能通过高于截止频率的信号

88. * 持续 : 触发后 , 过滤器持续当前采样。过滤器输出值是输入样本和持续样本之间的差异

89. * 设置值如下所示

90. * ACCEL_HPF | 高通滤波模式 | 截止频率

91. * ----------+-------------+-----------------

92. * 0 | Reset | None

93. * 1 | On | 5Hz

```
94.  * 2        | On        | 2.5Hz
95.  * 3        | On        | 1.25Hz
96.  * 4        | On        | 0.63Hz
97.  * 7        | Hold      | None
98.  */
99.
100.        #define MPU6050_RA_FF_THR         0x1D
101.        /*  自由落体加速度的阈值
102.         * 这个寄存器为自由落体的阈值检测进行配置。
103.        *FF_THR  的单位是 1LSB = 2mg  。当加速度传感器测量而得的三个轴的绝对值
104.        *  都小于检测阈值时，就可以测得自由落体值。这种情况下，  ( 加速度计每次检测到就 +1 以下, 所以还要依
    靠加速度采样率 )
105.        *  自由落体时间计数器计数一次  (  寄存器 30) 。当自由落体时间计数器达到
106.        *FF_DUR  中规定的时间时，自由落体被中断  ( 或发生自由落体中断  )
107.        **/
108.
109.        #define MPU6050_RA_FF_DUR         0x1E
110.        /*
111.         * 自由落体加速度的时间阈值
112.        *  这个寄存器为自由落体时间阈值计数器进行配置。
113.        *  时间计数频率为  1 khz,  因此 FF_DUR的单位是 1 LSB = 1    毫秒。
114.        *  当加速度器测量而得的绝对值都小于检测阈值时,
115.        *  自由落体时间计数器计数一次。当自由落体时间计数器
116.        *  达到该寄存器的规定时间时，自由落体被中断。
117.        * (  或发生自由落体中断  )
118.        * */
119.
120.        #define MPU6050_RA_MOT_THR        0x1F
121.        /*
122.         * 运动检测的加速度阈值
123.         * 这个寄存器为运动中断的阈值检测进行配置。
124.         *MOT_THR 的单位是  1LSB = 2mg   。
```

```
125.        * 当加速度器测量而得的绝对值都超过该运动检测的阈值时,
126.        * 即可测得该运动。这一情况下，运动时间检测计数器计数一次。
127.        * 当运动检测计数器达到 MOT_DUR 的规定时间时，运动检测被中断。
128.        *  运动中断表明了被检测的运动 MOT_DETECT_STATUS (Register 97)  的轴和极性。
129.        */
130.
131.       #define MPU6050_RA_MOT_DUR          0x20
132.       /*
133.       * 运动检测时间的阈值。
134.       * 这个寄存器为运动中断的阈值检测进行配置。
135.       * 时间计数器计数频率为 1 kHz ， 因此 MOT_THR 的单位是 1LSB = 1ms 。
136.       * 当加速度器测量而得的绝对值都超过该运动检测的阈值时  (Register 31)  ,
137.       * 运动检测时间计数器计数一次。当运动检测计数器达到该寄存器规定的时间时,
138.       * 运动检测被中断。
139.        **/
140.
141.       #define MPU6050_RA_ZRMOT_THR        0x21
142.       /*
143.       * 零运动检测加速度阈值。
144.       *  这个寄存器为零运动中断检测进行配置。
145.       * ZRMOT_THR 的单位是 1LSB = 2mg 。
146.       *  当加速度器测量而得的三个轴的绝对值都小于检测阈值时,
147.       *  就可以测得零运动。这种情况下，零运动时间计数器计数一次  ( 寄存器 34) 。
148.       *  当自零运动时间计数器达到 ZRMOT_DUR (Register 34)  中规定的时间时，零运动被中断。
149.       *  与自由落体或运动检测不同的是， 当零运动首次检测到以及当零运动检测不到时， 零运动检测都被中断。
150.       *  当零运动被检测到时 ，其状态将在 MOT_DETECT_STATUS 寄存器 ( 寄存器 97) 中显示出来。
151.       *  当运动状态变为零运动状态被检测到时 , 状态位设置为 1。当零运动状态变为运动状态被检测到时 ,
152.       *  状态位设置为 0。
153.        **/
```

```
154.
155.        #define MPU6050_RA_ZRMOT_DUR        0x22
156.        /*
157.        * 零运动检测的时间阈值
158.        *  这个寄存器为零运动中断检测进行时间计数器的配置。
159.        *  时间计数器的计数频率为  16 Hz,  因此  ZRMOT_DUR的单位是  1 LSB = 64 ms    。
160.        *  当加速度器测量而得的绝对值都小于检测器的阈值    (Register 33)       时,
161.        *  运动检测时间计数器计数一次。当零运动检测计数器达到该寄存器规定的时间时,
162.        *  零运动检测被中断。
163.         **/
164.
165.
166.        /*
167.         * 设备的各种  FIFO 使能 , 包括温度   加速度   陀螺仪   从机
168.         * 将相关的数据写入   FIFO  缓冲中
169.         **/
170.        #define MPU6050_RA_FIFO_EN          0x23
171.        //bit7      温度 fifo     使能
172.        //bit6      陀螺仪  Xfifo     使能
173.        //bit5      陀螺仪  Yfifo     使能
174.        //bit4      陀螺仪  Zfifo     使能
175.        //bit3      加速度传感器   fifo     使能
176.        //bit2      外部从设备   2fifo     使能
177.        //bit1      外部从设备   1fifo     使能
178.        //bit0      外部从设备   0fifo     使能
179.
180.        #define MPU6050_RA_I2C_MST_CTRL     0x24
181.        //  配置单主机或者多主机下的    IIC    总线
182.        //bit7      监视从设备总线 , 看总线是否可用   MULT_MST_EN设置为  1 时,MPU-60X0   的总线仲裁检测逻辑被打
    开
```

183. //bit6 延迟数据就绪中断，直达从设备数据也进入主机再触发 相当于数据同步等待

184. //bit5 当设置为 1 时, 与 Slave3 相连的外部传感器数据 (寄存器 73 到寄存器 96) 写入 FIFO 缓冲中, 每次都写入

185. //bit4 主机读取一个从机到下一个从机读取之间的动作 为 0 读取之间有一个 restart, 为 1 下一次读取前会有一个重启，然后

186. // 一直读取直到切换写入或者切换设备

187. //bit3-bit0 配置 MPU作为 IIC 主机时的时钟，基于 MPU内部 8M 的分频

188. /* I2C_MST_CLK | I2C 主时钟速度 | 8MHz 时钟分频器

189. * ------------+----------------------+------------------

190. * 0 | 348kHz | 23

191. * 1 | 333kHz | 24

192. * 2 | 320kHz | 25

193. * 3 | 308kHz | 26

194. * 4 | 296kHz | 27

195. * 5 | 286kHz | 28

196. * 6 | 276kHz | 29

197. * 7 | 267kHz | 30

198. * 8 | 258kHz | 31

199. * 9 | 500kHz | 16

200. * 10 | 471kHz | 17

201. * 11 | 444kHz | 18

202. * 12 | 421kHz | 19

203. * 13 | 400kHz | 20

204. * 14 | 381kHz | 21

205. * 15 | 364kHz | 22

206. * */

207.

208.

209.

210. /**********MPU 链接 IIC 从设备控制寄存器，没使用从机连接的基本不用考虑这些**********/

211. /* 指定 slave (0-3) 的 I2C 地址

212. * 注意 Bit 7 (MSB) 控制了读 / 写模式。如果设置了 Bit 7, 那么这是一个读取操作 ，

213. * 如果将其清除 , 那么这是一个编写操作。其余位 (6-0) 是 slave 设备的 7-bit 设备地址。

214. * 在读取模式中 , 读取结果是存储于最低可用的 EXT_SENS_DATA寄存器中。

```
215.        * MPU-6050   支持全 5 个 slave ，但 Slave 4   有其特殊功能 (getSlave4*      和 setSlave4*)    。
216.        *  如寄存器 25 中所述， I2C 数据转换通过采样率体现。用户负责确保   I2C 数据转换能够
217.        *  在一个采样率周期内完成。
218.        * I2C slave      数据传输速率可根据采样率来减小。
219.        *  减小的传输速率是由  I2C_MST_DLY( 寄存器 52) 所决定的。
220.        * slave    数据传输速率是否根据采样率来减小是由   I2C_MST_DELAY_CTRL (  寄存器 103) 所决定的。
221.        * slave    的处理指令是固定的。 Slave  的处理顺序是 Slave 1, Slave 2, Slave 3          和 Slave 4    。
222.        *  如果某一个 Slave  被禁用了，那么它会被自动忽略。
223.        *  每个 slave   可按采样率或降低的采样率来读取。在有些    slave   以采样率读取有些以减小
224.        *  的采样率读取的情况下，  slave   的读取顺序依旧不变。然而，
225.        *  如果一些 slave   的读取速率不能在特定循环中进行读取，那么它们会被自动忽略
226.        *  更多降低的读取速率相关信息 ，请参阅寄存器 52。
227.        * Slave    是否按采样率或降低的采样率来读取由寄存器    103 得 Delay Enable     位来决定
228.         **/
229.
230.        //  从机 0 设置相关
231.        #define MPU6050_RA_I2C_SLV0_ADDR    0x25
232.        //bit7      当前 IIC   从设备 0 的操作 ,1  为读取 0  写入
233.        //bit6-bit0         从机设备的地址
234.        /*   要读取或者要写入的设备内部的寄存器地址   , 不管读取还是写入 */
235.        #define MPU6050_RA_I2C_SLV0_REG     0x26
236.        /*iic     从机系统配置寄存器 */
237.        #define MPU6050_RA_I2C_SLV0_CTRL    0x27
238.        //bit7      启动或者禁止这个设备的   IIC  数据传送过程
239.        //bit6      当设置为 1 时 , 字节交换启用。当启用字节交换时   , 词对的高低字节即可交换
240.        //bit5      当 I2C_SLV0_REG_DIS   置 1 ， 只能进行读取或者写入数据。当该位清    0 ， 可以再读取
241.        //              或写入数据之前写入一个寄存器地址。当指定从机设备内部的寄存器地址进行发送或接收
```

```
242.        //              数据时，该位必须等于  0
243.        //bit4        指定从寄存器收到的字符对的分组顺序。当该位清    0 ，寄存器地址
244.        //              0 和 1，  2  和 3   的字节是分别成对（甚至，奇数寄存器地址   ），作为一个字符对。当该位置 1 ，
245.        //              寄存器地址 1   和 2 ，  3   和 4   的字节是分别成对的，作为一个字符对
246.        //bit3-bit0           指定从机  0   发送字符的长度。由  Slave 0    转换而来和转换至  Slave 0    的字节数 ,(IIC    一
            次传输的长度 )
247.        //                    该位清 0 ， I2C_SLV0_EN  位自动置 0.
248.
249.        /*IIC SLAVE1     配置寄存器，与 0 相同*/
250.
251.        #define MPU6050_RA_I2C_SLV1_ADDR    0x28
252.        #define MPU6050_RA_I2C_SLV1_REG     0x29
253.        #define MPU6050_RA_I2C_SLV1_CTRL    0x2A
254.
255.        /*IIC SLAVE2     配置寄存器，与 0 相同*/
256.        #define MPU6050_RA_I2C_SLV2_ADDR    0x2B
257.        #define MPU6050_RA_I2C_SLV2_REG     0x2C
258.        #define MPU6050_RA_I2C_SLV2_CTRL    0x2D
259.
260.        /*IIC SLAVE3     配置寄存器，与 0 相同*/
261.        #define MPU6050_RA_I2C_SLV3_ADDR    0x2E
262.        #define MPU6050_RA_I2C_SLV3_REG     0x2F
263.        #define MPU6050_RA_I2C_SLV3_CTRL    0x30
264.
265.        /*slave4      的 I2C  地址  IIC4    与前几个的寄存器定义有所不同   */
266.        #define MPU6050_RA_I2C_SLV4_ADDR    0x31                //  与 IIC SLAVE1    类似
267.        #define MPU6050_RA_I2C_SLV4_REG     0x32                /*slave4      的当前内部寄存器 */
268.        #define MPU6050_RA_I2C_SLV4_DO      0x33
269.          /*  写于 slave4    的新字节这一寄存器可储存写于    slave4    的数据。
270.          *    如果 I2C_SLV4_RW 设置为 1（设置为读取模式），那么该寄存器无法执行操作     */
271.        #define MPU6050_RA_I2C_SLV4_CTRL    0x34
272.        //  当设置为 1 时，此位启用了  slave4    的转换操作。当设置为   0 时，则禁用该操作
273.        #define MPU6050_I2C_SLV4_EN_BIT         7
```

```
274.        //  当设置为 1 时，此位启用了  slave4   事务完成的中断信号的生成。
275.        //   当清除为 0 时，则禁用了该信号的生成。这一中断状态可在寄存器     54 中看到。
276.        #define MPU6050_I2C_SLV4_INT_EN_BIT    6
277.        //  当设置为 1 时, 只进行数据的读或写操作。当设置为    0 时,
278.        //   在读写数据之前将编写一个寄存器地址。当指定寄存器地址在      slave   设备中时
279.        //   ，这应该等于 0，而在该寄存器中会进行数据处理。
280.        #define MPU6050_I2C_SLV4_REG_DIS_BIT    5
281.        //  采样率延迟 , 这为根据采样率减小的   I2C slaves     传输速率进行了配置。
282.        //   当一个 slave   的传输速率是根据采样率而降低的   , 那么该 slave   是以每 1 / (1 + I2C_MST_DLY)       个
        样本进行传输。
283.        //   这一基本的采样率也是由   SMPLRT_DIV ( 寄存器 25)  和 DLPF_CFG ( 寄存器 26) 所决定的的。
284.        // slave      传输速率是否根据采样率来减小是由    I2C_MST_DELAY_CTRL ( 寄存器 103) 所决定的
285.        #define MPU6050_I2C_SLV4_MST_DLY_BIT    4          //[4:0]
286.        #define MPU6050_I2C_SLV4_MST_DLY_LENGTH 5
287.        /*slave4     中可读取的最后可用字节 */
288.        #define MPU6050_RA_I2C_SLV4_DI     0x35
289.
290.        /*
291.         * IIC     辅助从机系统中断状态
292.         **/
293.        #define MPU6050_RA_I2C_MST_STATUS   0x36
294.        //bit7      此位反映了一个与  MPU-60X0 相连的外部设备的  FSYNC中断状态。
295.        //              当设置为 1 且在 INT_PIN_CFG( 寄存器 55) 中断言 FSYNC_INT_EN 时，中断产生。
296.        //bit6      当 slave4   事务完成时，设备会自动设置为   1  如果定义了 INT_ENABLE 中的 I2C_MST_INT_EN 则
        产生中断
297.        //bit5     I2C 主机失去辅助 I2C 总线（一个错误状态）的仲裁，此位自动设置为   1. 如果断言了 INT_ENABLE
        寄存器
298.        //              （寄存器 56）中的 I2C_MST_INT_EN 位，则中断产生
299.        //bit4         slave4   的 NACK状态
```

```
300.        //bit3  slave3        的 NACK状态
301.        //bit2  slave2        的 NACK状态
302.        //bit1  slave1        的 NACK状态
303.        //bit0  slave0        的 NACK状态
304.
305.
306.        /*  中断引脚配置寄存器  */
307.        #define MPU6050_RA_INT_PIN_CFG      0x37
308.        //bit7        中断的逻辑电平模式 , 高电平时，设置为  0；低电平时，设置为  1
309.        //bit6        中断驱动模式 , 推拉模式设置为  0，开漏模式设置为  1.
310.        //bit5        中断锁存模式 .50us-pulse      模式设置为  0，latch-until-int-cleared                模式设置为  1
311.        //bit4        中断锁存清除模式  status-read-only          状态设置为  0，any-register-read          状态设置为  1.
312.        //bit3  FSYNC       中断逻辑电平模式  0=active-high, 1=active-low
313.        //bit2  FSYNC       端口中断启用设置设置为  0 时禁用，设置为  1 时启用
314.        //bit1  I2C         支路启用状态 , 此位等于  1 且 I2C_MST_EN (  寄存器 106   位[5])   等于  0 时 , 主机应用程序处
        理器能够直接访问  MPU-60X0 的辅助 I2C  总线
315.        //                  否则无论如何都不能直接访问
316.        //bit0        当此位为  1 时，CLKOUT端口可以输出参考时钟。当此位为    0 时，输出禁用
317.
318.
319.        /*  部分中断使能 */
320.        #define MPU6050_RA_INT_ENABLE       0x38
321.        //bit7        自由落体中断使能
322.        //bit6        运动检测中断使能
323.        //bit5        零运动检测中断使能
324.        //bit4  FIFO        溢出中断使能
325.        //bit3  IIC         主机所有中断源使能
```

```
326.        //bit0        数据就绪中断使能
327.
328.
329.        /*DMP  中断使能 */
330.        #define MPU6050_RA_DMP_INT_STATUS   0x39
331.        //  不知道这些位的具体作用是什么  ，官方语焉不详，但是的确存在
332.        #define MPU6050_DMPINT_4_BIT            4
333.        #define MPU6050_DMPINT_3_BIT            3
334.        #define MPU6050_DMPINT_2_BIT            2
335.        #define MPU6050_DMPINT_1_BIT            1
336.        #define MPU6050_DMPINT_0_BIT            0
337.
338.        /*DMP  中断配置 */
339.        #define MPU6050_RA_INT_STATUS       0x3A
340.        //DMP  中断位之一使能
341.        #define MPU6050_INTERRUPT_PLL_RDY_INT_BIT   2
342.        //DMP  中断位之二使能
343.        #define MPU6050_INTERRUPT_DMP_INT_BIT       1
344.
345.        /*  加速度 X 输出*/
346.        #define MPU6050_RA_ACCEL_XOUT_H     0x3B
347.        #define MPU6050_RA_ACCEL_XOUT_L     0x3C
348.
349.        /*  加速度 Y 输出*/
350.        #define MPU6050_RA_ACCEL_YOUT_H     0x3D
351.        #define MPU6050_RA_ACCEL_YOUT_L     0x3E
352.
353.        /*  加速度 Z 输出*/
354.        #define MPU6050_RA_ACCEL_ZOUT_H     0x3F
355.        #define MPU6050_RA_ACCEL_ZOUT_L     0x40
356.
357.        /*  温度值输出 */
358.        #define MPU6050_RA_TEMP_OUT_H       0x41
359.        #define MPU6050_RA_TEMP_OUT_L       0x42
360.
361.        /*  陀螺仪 X 输出*/
362.        #define MPU6050_RA_GYRO_XOUT_H      0x43
363.        #define MPU6050_RA_GYRO_XOUT_L      0x44
```

```
364.
365.        /*  陀螺仪 Y 输出*/
366.        #define MPU6050_RA_GYRO_YOUT_H      0x45
367.        #define MPU6050_RA_GYRO_YOUT_L      0x46
368.
369.        /*  陀螺仪 Z 输出*/
370.        #define MPU6050_RA_GYRO_ZOUT_H      0x47
371.        #define MPU6050_RA_GYRO_ZOUT_L      0x48
372.
373.        /*  从 IIC   从机上获取到的数据  */
374.        #define MPU6050_RA_EXT_SENS_DATA_00 0x49
375.        #define MPU6050_RA_EXT_SENS_DATA_01 0x4A
376.        #define MPU6050_RA_EXT_SENS_DATA_02 0x4B
377.        #define MPU6050_RA_EXT_SENS_DATA_03 0x4C
378.        #define MPU6050_RA_EXT_SENS_DATA_04 0x4D
379.        #define MPU6050_RA_EXT_SENS_DATA_05 0x4E
380.        #define MPU6050_RA_EXT_SENS_DATA_06 0x4F
381.        #define MPU6050_RA_EXT_SENS_DATA_07 0x50
382.        #define MPU6050_RA_EXT_SENS_DATA_08 0x51
383.        #define MPU6050_RA_EXT_SENS_DATA_09 0x52
384.        #define MPU6050_RA_EXT_SENS_DATA_10 0x53
385.        #define MPU6050_RA_EXT_SENS_DATA_11 0x54
386.        #define MPU6050_RA_EXT_SENS_DATA_12 0x55
387.        #define MPU6050_RA_EXT_SENS_DATA_13 0x56
388.        #define MPU6050_RA_EXT_SENS_DATA_14 0x57
389.        #define MPU6050_RA_EXT_SENS_DATA_15 0x58
390.        #define MPU6050_RA_EXT_SENS_DATA_16 0x59
391.        #define MPU6050_RA_EXT_SENS_DATA_17 0x5A
392.        #define MPU6050_RA_EXT_SENS_DATA_18 0x5B
393.        #define MPU6050_RA_EXT_SENS_DATA_19 0x5C
394.        #define MPU6050_RA_EXT_SENS_DATA_20 0x5D
395.        #define MPU6050_RA_EXT_SENS_DATA_21 0x5E
396.        #define MPU6050_RA_EXT_SENS_DATA_22 0x5F
397.        #define MPU6050_RA_EXT_SENS_DATA_23 0x60
398.
399.        //  运动检测的状态
400.        #define MPU6050_RA_MOT_DETECT_STATUS    0x61
401.        //bit7  x       轴反向运动检测中断状态
402.        //bit6  x       轴正向运动检测中断状态
403.        //bit5  Y       轴反向运动检测中断状态
404.        //bit4  Y       轴正向运动检测中断状态
```

```
405.        //bit3  Z       轴反向运动检测中断状态
406.        //bit2  Z       轴正向运动检测中断状态
407.        //bit1
408.        //bit0        零运动检测中断状态
409.        //
410.
411.
412.        /*  写入到 IIC   从机中的数据，指定的 slv   数据输出容器 */
413.        #define MPU6050_RA_I2C_SLV0_DO      0x63
414.        #define MPU6050_RA_I2C_SLV1_DO      0x64
415.        #define MPU6050_RA_I2C_SLV2_DO      0x65
416.        #define MPU6050_RA_I2C_SLV3_DO      0x66
417.
418.        /*  外部影子寄存器的配置 ，这个寄存器用于指定外部传感器数据影子的时间
419.        * 当启用了某一特定的   slave  ，  其传输速率就会减小。
420.        * 当一个  slave    的传输速率是根据采样率而降低的   ，那么该 slave    是以
421.        * 每 1 / (1 + I2C_MST_DLY)          个样本进行传输。
422.        *     1 / (1 + I2C_MST_DLY) Samples
423.        *  这一基本的采样率也是由   SMPLRT_DIV (  寄存器  25)  和 DLPF_CFG (  寄存器  26) 所决定的的。  */
424.        #define MPU6050_RA_I2C_MST_DELAY_CTRL   0x67
425.        //DELAY_ES_SHADOW设置为 1, 跟随外部传感器数据影子将会延迟到所有的数据接收完毕。
426.        #define MPU6050_DELAYCTRL_DELAY_ES_SHADOW_BIT   7
427.        //slv4-0        的配置
428.        #define MPU6050_DELAYCTRL_I2C_SLV4_DLY_EN_BIT   4
429.        #define MPU6050_DELAYCTRL_I2C_SLV3_DLY_EN_BIT   3
430.        #define MPU6050_DELAYCTRL_I2C_SLV2_DLY_EN_BIT   2
431.        #define MPU6050_DELAYCTRL_I2C_SLV1_DLY_EN_BIT   1
432.        #define MPU6050_DELAYCTRL_I2C_SLV0_DLY_EN_BIT   0
433.
434.        /*  用于陀螺仪，加速度计，温度传感器的模拟和数字信号通道的复位。
435.        复位会还原模数转换信号通道和清除他们的上电配置     */
436.        #define MPU6050_RA_SIGNAL_PATH_RESET    0x68
437.        //bit2        重置陀螺仪的信号路径
438.        //bit1        重置加速度传感器的信号路径
```

```
439.        //bit0        重置温度传感器的信号路径
440.
441.
442.        /*  获取加速度传感器启动延迟    还有滤波器的一些配置
443.        *   加速度传感器数据路径为传感器寄存器、运动检测、
444.        *   零运动检测和自由落体检测模块提供样本。在检测模块开始操作之前,
445.        *   包含过滤器的信号路径必须用新样本来启用。
446.        *   默认的 4 毫秒唤醒延迟时间可以加长   3 毫秒以上。在  ACCEL_ON_DELAY中规定
447.        *   这个延迟以  1 LSB = 1     毫秒为单位。除非   InvenSense    另行指示,
448.        *   用户可以选择任何大于零的值。    */
449.        #define MPU6050_RA_MOT_DETECT_CTRL      0x69
450.        //  具体的有效控制位
451.        //bit5-bit4 [5:4]1-4ms                延时时间  1-4ms  选择
452.        //bit3-bit2           自由落体检测计数器的减量配置。
453.        //                    当指定数量的样本的加速度测量都满足其各自的阈值条件时,
454.        //                    检测结果存储于自由落体检测模块中。当满足阈值条件时,
455.        //                    相应的检测计数器递增   1。用户可通过  FF_COUNT配置不满足阈值条件来减量。
456.        //                    减量率可根据下表进行设置：
457.                            /* FF_COUNT |      计数器减量
458.                            * ---------+-----------------
459.                            * 0                       |   重置
460.                            * 1                       | 1
461.                            * 2                       | 2
462.                            * 3                       | 4
463.                            *   当 FF_COUNT配置为 0(  复位) 时, 任何不合格的样品都将计数器重置为     0*/
464.        //bit1-bit0           运动检测计数器的减量配置。
465.        //                    当指定数量的样本的加速度测量都满足其各自的阈值条件时,
466.        //                    检测结果存储于运动检测模块中。当满足阈值条件时，相应的检测计数器递增        1。
```

467. // 用户可通过 MOT_COUNT配置不满足阈值条件来减量。减量率可根据下表进行设置：

468. // MOT_COUNT | 计数器减量

469. /* ----------+-----------------

470. * 0 | 重置

471. * 1 | 1

472. * 2 | 2

473. * 3 | 4

474. * 当 MOT_COUNT配置为 0(复位) 时, 任何不合格的样品都将计数器重置为 0*/

475.

476.

477. /* 这个寄存器允许用户使能或使能 FIFO 缓冲区,

478. *I2C 主机模式和主要 I2C 接口。FIFO 缓冲

479. 区， I2C 主机，传感器信号通道和传感器寄存器也可以使用这个寄存器复位 */

480. #define MPU6050_RA_USER_CTRL 0x6A

481. //bit7 DMP 禁止

482. //bit6 当此位设置为 0,FIFO 缓冲是禁用的

483. //bit5 当这个模式被启用 ,MPU-60X0 即成为辅助 I2C 总线上的外部传感器 slave 设备的 I2C 主机

484. // 当此位被清除为 0 时, 辅助 I2C 总线线路 (AUX_DA and AUX_CL) 理论上是由 I2C 总线

485. // (SDA 和 SCL)驱动的。这是启用旁路模式的一个前提

486. //bit4 I2C 转换至 SPI 模式 (只允许 MPU-6000)

487. //bit3 重置 DMP模式, 官方文档未说明的寄存器

488. //bit2 重置 FIFO 当设置为 1 时，此位将重置 FIFO 缓冲区，此时 FIFO_EN 等于 0。触发重置后，此位将

自动清为 0

489. //bit1 重置 I2C 主机当设置为 1 时，此位将重置 I2C 主机，此时 I2C_MST_EN 等于 0。触发重置后，此

位将自动清为 0

490. //bit0 重置所有传感器寄存器和信号路径 如果只重置信号路径 （不重置传感器寄存器） ，请使用寄存器

104

491.

492.

493. /* 允许用户配置电源模式和时钟源。还提供了复位整个设备和禁用温度传感器的位 */

494. #define MPU6050_RA_PWR_MGMT_1 0x6B

495. //bit7 触发一个设备的完整重置。 触发重置后，一个 ~ 50 毫秒的小延迟是合理的

496. //bit6 寄存器的 SLEEP 位设置使设备处于非常低功率的休眠模式。

497. //bit5 唤醒周期启用状态当此位设为 1 且 SLEEP 禁用时 . 在休眠模式和唤醒模式间循环，以此从活跃的

传感器中获取数据样本

498. //bit3 温度传感器启用状态控制内部温度传感器的使用

499. //bit2-bit0 设定时钟源设置，一个频率为 8 mhz 的内部振荡器，基于陀螺仪的时钟或外部信息源都可以被

选为 MPU-60X0 的时钟源

500. /* CLK_SEL | 时钟源

501. * --------+-----------------------------------

502. * 0 | 内部振荡器

503. * 1 | PLL with X Gyro reference

504. * 2 | PLL with Y Gyro reference

505. * 3 | PLL with Z Gyro reference

506. * 4 | PLL with external 32.768kHz reference

507. * 5 | PLL with external 19.2MHz reference

508. * 6 | Reserved

509. * 7 | Stops the clock and keeps the timing generator in reset

510. * */

511.

512.

513. /* 这个寄存器允许用户配置加速度计在低功耗模式下唤起的频率。也允许用户让加速度计和

514. 陀螺仪的个别轴进入待机模式。 */

515. #define MPU6050_RA_PWR_MGMT_2 0x6C

516. //bit7-bit6 Accel-Only 低电量模式下的唤醒频率

517. /* 通过把 Power Management 1 寄存器（寄存器 107）中的 PWRSEL设为 1,

518. * MPU-60X0 可以处于 Accerlerometer Only 的低电量模式。在这种模式下 ,

519. 设备将关闭除了原 I2C 接口以外的所有设备，只留下 accelerometer 以固定时间

520. 间隔醒来进行测量。唤醒频率可用 LP_WAKE_CTRL进行配置，如下表所示：

521. * LP_WAKE_CTRL | 唤醒频率

```
522.                * ------------+-----------------
523.                * 0          | 1.25 Hz
524.                * 1          | 2.5 Hz
525.                * 2          | 5 Hz
526.                * 3          | 10 Hz
527.                * */
528.        //bit5      备用的 x 轴加速度传感器启用状态，也就是进入待机模式
529.        //bit4      备用的 Y 轴加速度传感器启用状态
530.        //bit3      备用的 Z 轴加速度传感器启用状态
531.        //bit2      备用的 x 轴陀螺仪启用状态
532.        //bit1      备用的 Y 轴陀螺仪启用状态
533.        //bit0      备用的 Z 轴陀螺仪启用状态
534.
535.        /* 设定 DMP模式下的 bank*/
536.        #define MPU6050_RA_BANK_SEL         0x6D
537.        //DMP 内存配置
538.        #define MPU6050_BANKSEL_PRFTCH_EN_BIT       6
539.        #define MPU6050_BANKSEL_CFG_USER_BANK_BIT   5
540.        #define MPU6050_BANKSEL_MEM_SEL_BIT         4
541.        #define MPU6050_BANKSEL_MEM_SEL_LENGTH      5
542.        //dmp 内存地址设置
543.        #define MPU6050_DMP_MEMORY_BANKS        8
544.        #define MPU6050_DMP_MEMORY_BANK_SIZE    256
545.        #define MPU6050_DMP_MEMORY_CHUNK_SIZE   16
546.
547.        /* 设定 DMP模式下的起始地址 */
548.        #define MPU6050_RA_MEM_START_ADDR   0x6E
549.        /* 一个字节的 dmp数据缓存 */
550.        #define MPU6050_RA_MEM_R_W          0x6F
551.        /*DMP 配置寄存器 1*/
552.        #define MPU6050_RA_DMP_CFG_1        0x70
553.        /*DMP 配置寄存器 2*/
554.        #define MPU6050_RA_DMP_CFG_2        0x71
555.
556.        /* 当前 FIFO 缓冲区大小
```

```
557.        *  这个值表明了存储于  FIFO 缓冲区的字节数。
558.        *  而这个数字也是能从  FIFO 缓冲区读取的字节数,
559.        *  它与存储在  FIFO( 寄存器 35 和 36) 中的传感器数据组所提供的可用样本数成正比。
560.        *  两个寄存器一起构成一个  16 位数据*/
561.        #define MPU6050_RA_FIFO_COUNTH      0x72
562.        #define MPU6050_RA_FIFO_COUNTL      0x73
563.
564.        /*  这个寄存器用于从  FIFO 缓冲区中读取和编写数据。数据在寄存器编号  ( 从低到高 ) 的指
565.         *  令下编写入数据写入  FIFO。如果所有的  FIFO 启用标志 ( 见下文 ) 都被启用了且
566.         *  所有外部传感器数据寄存器 ( 寄存器 73 至寄存器 96) 都与一个 slave  设备相连
567.         *,  那么寄存器 59 到寄存器 96 的内容都将在采样率的指令下编写。
568.        *  当传感器数据寄存器（寄存器  59 到寄存器 96）的相关 FIFO 启用标志在 FIFO_EN 寄存
569.        *  器 35) 中都设为 1 时，它们的内容将被写入  FIFO 缓冲区。在 I2C_MST_CTRL (  寄存器 36)
570.        *  中能找到一个与  I2C Slave 3  相连的额外的传感器数据寄存器标志。
571.        *  如果 FIFO 缓冲区溢出 , 状态位 FIFO_OFLOW_INT 自动设置为 1。
572.        *  此位位于 INT_STATUS (  寄存器 58) 中。当 FIFO 缓冲区溢出时，最早的数据将会丢失
573.        *  而新数据将被写入  FIFO。如果 FIFO 缓冲区为空，  读取将返回原来从  FIFO 中读取的
574.        *  最后一个字节，直到有可用的新数据。用户应检查  FIFO_COUNT, 以确保不在 FIFO 缓冲为空时读取。 */
575.        #define MPU6050_RA_FIFO_R_W         0x74
576.
577.        /*  寄存器是用来验证设备的身份的  默认值是 0X34*/
578.        #define MPU6050_RA_WHO_AM_I         0x75
579.        //bit6-bit1         设备身份验证  0x34  最高位和最低位都剔除掉
580.
581.
```