MMU 6050 interrupt data summary

Functions can be en/disabled via the corresponding Status register/bit.

When interrupt occurs, use **getIntStatus** to determine which int(s) occurred, then for some features you already have the data (eg moving or FreeFall). For other features you need to read the data from other register(s) – eg accel/gyro values or use one of the other helper methods that present the data "nicely".

Invesense documents are often incomplete and can even be misleading. (eg doc bits as "-" when in fact have known and required function!)

Table below tries to lay out the interrupt settings, status and related data access in a logical fashion, but the chip design has many exceptions – tried to highlight these below!

Some of the initialisation values CHANGE during the standard/dmp init process … for dif reasons. Just keep in mind when you delve into settings/defaults!!! Eg I2C slave 0 default address.

		En/Disable features and matching feature interrupts register/bit(s)				**Read status or data item ONLY when matching int register bit indicates ready!**			
Group	**Feature**	**Status Register**	**Bit (s)**	**I2Cdev Lib Set/GetStatus method**	**Notes**	**Data Register**	**Bit (s)**	**I2Cdev Lib GetData method**	**Notes**
General interrupt configuration	INT_LEVEL	**55dec 0x37 INT_PIN_CFG R/W**	7	setInterruptMode	0=active-high, 1=active-low	n/a	n/a	n/a	No data or separate data register. Just read value from interrupt status register, using the matching get method.
	INT_OPEN		6	setInterruptDrive	0=push-pull, 1=open-drain	n/a	n/a	n/a	
	LATCH _INT_EN		5	setInterruptLatch	0=50us-pulse, 1=latch-until-int-cleared	n/a	n/a	n/a	
	INT_RD _CLEAR		4	setInterruptLatchClear	New latch clear mode (0=status-read-only, 1=any-register-read) ie CLEAR!	n/a	n/a	n/a	
	FSYNC_ INT_LEVEL		3	setFSyncInterruptLevel	0=active-high, 1=active-low	n/a	n/a	n/a	
	FSYNC_INT_EN		2	setFSyncInterruptEnabled		n/a	n/a	n/a	
	I2C_BYPASS_EN		1	setI2CBypassEnabled	bit=1 AND I2C_MST_EN (Reg 106 bit[5])=0, host proc. can access auxil MPU-60X0 I2C bus	n/a	n/a	n/a	
	INTCFG_CLKOUT_EN		0	setClockOutputEnabled	1=ref clk out at CLKOUT, 0=no ref clk out	n/a	n/a	n/a	
Raw Data & related interrupts	**Raw & DMP & I2C**	**56dec 0x38 INT_ENABL R/W**	**0-7**	**setIntEnabled**	**<<< BYTE method**	58dec 0x3A INT_STATUS	**0-7**	**getIntStatus**	**<<< BYTE method**
	FF		7	setIntFreefallEnabled			7	getIntFreefallStatus	.... register shows the interrupt status of each interrupt generation source. >>>>Each bit will clear after the register is read. ...This means ALL bits clear if you read ANY bit ... you have to read the whole byte to get a bit.
	MOT		6	setIntMotionEnabled			6	getIntMotionStatus	
	ZMOT		5	setIntZeroMotionEnabled			5	getIntZeroMotionStatus	
	FIFO_OFLOW		4	setIntFIFOBufferOverflowEnabled			4	getIntFIFOBufferOverflowStatus	
	I2C_MST_INT		3	setIntI2CMasterEnabled			3	getIntI2CMasterStatus	
	PLL_RDY		2	setIntPLLReadyEnabled	DMP method		2	***getIntPLLReadyStatus***	
	DMP_INT		1	setIntDMPEnabled	DMP method		1	getIntDMPStatus	SEE DMP Data in next section of this table just BELOW.
	DATA_RDY_EN		0	setIntDataReadyEnabled			0	***getIntDataReadyStatus***	***See discussion on RawData below this table!***
DMP Data & related interrupts ?? all via FIFO??	Unknown function	See 56dec 0x38, bit1	5	En/Disabled via Raw Data & related interrupts – DMP_INT above (**56dec 0x38 INT_ENABL, bit 1 DMP_INT**) There are no methods to en/disable bit level functions.		56dec 0x39 DMP_INT_STATUS	5	getDMPInt5Status	Unknown function
	Unknown function		4				4	getDMPInt4Status	Unknown function
	Unknown function		3				3	getDMPInt3Status	Unknown function
	Unknown function		2				2	getDMPInt2Status	Unknown function
	Unknown function		1		? include FIFO as a reg??		1	getDMPInt1Status	Unknown function
	Unknown function		0				0	getDMPInt0Status	Unknown function

RawData

Raw data can be read direct from corresponding register pairs – see below.

The I2Cdev MMU6050 library ALSO offers 16 bit values & combined x,y,z & combined accel/gyro data methods, FIFO raw data methods and possibly more.

```
#define MPU6050_RA_ACCEL_XOUT_H     0x3B        #define MPU6050_RA_ACCEL_XOUT_L     0x3C
#define MPU6050_RA_ACCEL_YOUT_H     0x3D        #define MPU6050_RA_ACCEL_YOUT_L     0x3E
#define MPU6050_RA_ACCEL_ZOUT_H     0x3F        #define MPU6050_RA_ACCEL_ZOUT_L     0x40

#define MPU6050_RA_TEMP_OUT_H       0x41        #define MPU6050_RA_TEMP_OUT_L       0x42

#define MPU6050_RA_GYRO_XOUT_H      0x43        #define MPU6050_RA_GYRO_XOUT_L      0x44
#define MPU6050_RA_GYRO_YOUT_H      0x45        #define MPU6050_RA_GYRO_YOUT_L      0x46
#define MPU6050_RA_GYRO_ZOUT_H      0x47        #define MPU6050_RA_GYRO_ZOUT_L      0x48        #define MPU6050_RA_EXT_SENS_DATA_00 0x49 ….. 0x60
```