List of public functions of the ICM20948_WE library

Function	Parameters	what it does
bool init( )	none	Init() first resets and then initiates the ICM20948 with some default register values. Returns true if the ICM20948 has responded.
void autoOffsets ( )	none	Measures acceleration and gyroscope values and calculates offset values. The ICM20948 should be positioned flat in its xy-plane.
void enableAcc(true/false)	true / false	Enables / disables the accelerometer.
void enableGyr(true/false)	true / false	Enables / disables the gyrometer.
void setAccOffsets (min/max values)	xMin, xMax, yMin, yMax, zMin, zMax (all float)	A more accurate method to set offsets. You need to determine the min/max raw acceleration values for the axes manually (2g range).
void setGyrOffsets (x,y,z-Offsets)	xOffset, yOffset, zOffset	A method to set gyroscope offsets. You need to determine the raw gyroscope values, when only gravity acts on the sensor.
uint8_t whoAmI( )	none	Returns the ID of the ICM20948, which should be 0xEA.
void setGyrDLPF(level)	ICM20948_DLPF_0 ..... ICM20948_DLPF_7	Sets the digital low pass filter for the gyroscope to reduce noise. You can choose from 8 levels. You find more information in the example sketches.
void setAccSampleRateDivider(divider)	0...4095	Divides the sample rate of the accelerometer by (1+divider). It can only be applied if the corresponding DLPF is enabled and 0 < DLPF < 7.
void setGyrSampleRateDivider(divider)	0...255	Divides the sample rate of the gyroscope by (1+divider). It can only be applied if the corresponding DLPF is enabled and 0 < DLPF < 7.
void setGyrRange(range)	ICM20948_GYRO_RANGE_250, ICM20948_GYRO_RANGE_500, ICM20948_GYRO_RANGE_1000, ICM20948_GYRO_RANGE_2000	Sets the gyroscope range in degrees / second. The higher the range, the lower is the resolution. Default is 250.
void setAccRange(range)	ICM20948_ACC_RANGE_2G, ICM20948_ACC_RANGE_4G, ICM20948_ACC_RANGE_8G, ICM20948_ACC_RANGE_16G	Sets the range for the accelerometer in g. You can set it to +/2, +/-4, +/-8 or +/- 16 g. The higher the range, the lower is the resolution. Default is 2g.
void setAccDLPF(level)	ICM20948_DLPF_0 ..... ICM20948_DLPF_7 ICM20948_OFF	Sets the digital low pass filter to reduce noise. You can choose from 8 levels. You find more information in the sketches. ICM20948_OFF disables DLPF.
void setTempDLPF(level)	ICM20948_DLPF_0 ..... ICM20948_DLPF_7 ICM20948_OFF	Sets the digital low pass filter for the thermometer to reduce noise. You can choose from 8 levels. You find more information in the sketches. ICM20948_OFF disables DLPF.
void setI2CMstSampleRate(level)	0....15	setI2CMstSampleRate sets the rate of the devices controlled by the I2C master, i.e. the magnetometer. It is not the internal sample rate of the magnetometer, but the output rate of the I2C master. Allowed values are x = 0...15. The sample rate is 1.1 kHz / (2^x). Example: x = 13 => Sample rate = 1.1 kHz / 8192 = ~0.1343 Hz, or: data output every ~7.45 seconds.
void readSensor( )	none	Sensors are read and raw values are written into data registers.
xyzFloat getAccRawValues( )	none	Returns a set (x,y,z) of raw acceleration values. xyzFloat is a struct which consists of three floats: x,y,z.
xyzFloat getCorrectedAccRawValues( )	none	Returns the "calibrated" raw values for acceleration.
xyzFloat getGValues( )	none	Returns g values which are based on the corrected raw acceleration values.
xyzFloat getAccRawValuesFromFifo( )	none	Returns acceleration raw values (one set of x,y,z values) from the Fifo.
xyzFloat getCorrectedAccRawValuesFromFifo( )	none	Returns corrected (calibrated) raw values from the Fifo.
xyzFloat getGValuesFromFifo( )	none	Return values from the Fifo as g values. These are calculated from the corrected raws.
float getResultantG(xyzFloat g-value)	g values as xyzFloat	Returns the resulting g value of the three axes (sum of the vectors which is not the sum of the x,y,z g values). If only gravity acts on the ICM20948, it should always return 1 g.
float getTemperature( )	none	Returns the temperature measured by the temperature sensor of the ICM20948.
xyzFloat getGyrRawValues( )	none	Returns the raw gyroscope values. xyzFloat is a struct which consists of three floats: x,y,z.
xyzFloat getCorrectedGyrRawValues( )	none	Returns the calibrated raw gyroscope values.
xyzFloat getGyrValues( )	none	Returns gyroscope values in degrees/second. Based on calibrated raws.
xyzFloat getGyrValuesFromFifo ( )	none	Returns gyroscope values (one set of x,y,z values) from the Fifo in degrees/second. Based on calibrated raws.
xyzFloat getMagValues( )	none	Returns the magnetic flux density for the x,y and z-axis in µTesla.
xyzFloat getAngles( )	none	Returns the angles of the x,y and z axis vs. the horizontal. It only works if only gravity acts on the ICM20948. The method works well below 60°, then the deviations increase. It's just the arcsin of the g values of the axes.
ICM20948_orientation getOrientation( )	none	Returns the axis with the highest positive acceleration. The return value is an enum called ICM20948_orientation. For "translation" have a look into ICM20948.h.
String getOrientationAsString( )	none	This function also returns the orientation, but - better to understand - as a string: "x up", "x down", "y up", "y down", "z up" or "z down".

float getPitch( )	none	Returns the pitch tilt angle. The calculation is based on x, y and z and therefore better at higher angles than the getAngles method. The latter has a higher precision at small angles.
float getRoll( )	none	Returns the roll tilt angle. Otherwise same comments as for getPitch.
void sleep(true/false)	true / false	Enables / disables sleep mode. After disabling it takes some time before you will measure correct acceleration and gyroscope values. This depends on mode / DLPF.
void enableCycle(mode)	ICM20948_NO_CYCLE ICM20948_GYR_CYCLE ICM20948_ACC_CYCLE ICM20948_ACC_GYR_CYCLE ICM20948_ACC_GYR_I2C_MST_CYCLE	Enables / disables cycle modes. The ICM20948 toggles between active and sleep mode. The frequency depends on the sample. If you enable cycle for several sensors, the priority order for the frequency is: gyroscope -> accelerometer -> magnetometer (I2C_MST_CYCLE).
void setGyrAverageInCycleMode (number)	ICM20948_GYR_AVG_x x= 1,2,4,8,16,32,64,128	Number of gyroscope samples to be averaged in cycle mode.
void setAccAverageInCycleMode (number)	ICM20948_ACC_AVG_4 ICM20948_ACC_AVG_8 ICM20948_ACC_AVG_16 ICM20948_ACC_AVG_32	Number of accelerometer samples to be averaged in cycle mode.
void enableLowPower(true/false)	true / false	The low power mode only affects the digital circuitry, it helps to reduce the digital current when sensors are in cycle mode. Sensors in cycle mode and low power mode together help to reduce overall current. Enabling low power has no effect when the sensors are in low-noise mode.
void setFSyncIntPolarity(polarity)	ICM20948_ACT_HIGH ICM20948_ACT_LOW	Sets the polarity which shall trigger an interrupt.
void setIntPinPolarity(polarity)	ICM20948_ACT_HIGH ICM20948_ACT_LOW	Sets the interrupt pin polarity active-high (default) or active-low.
void enableIntLatch(true/false)	true, false	If latch is enabled the interrupt pin level is held until the interrupt status is cleared. If latch is disabled the interrupt pulse is ~50µs (default).
void enableClearIntByAnyRead(true/false)	true, false	The interrupt can be cleared by any read (true) or it will only be cleared if the interrupt status register is read (false = default).
void enableInterrupt(type)	ICM20948_FSYNC_INT ICM20948_WOM_INT ICM20948_DATA_READY_INT ICM20948_FIFO_OVF_INT	Enables an interrupt type. The library has implemented four types: FSYNC pin polarity, new data is ready to be read, fifo overflow or wake-on-motion interrupt. The latter is triggered by acceleration data which exceeds a defined threshold. If you want to enable more than one interrupt type, then call the function several times.
void disableInterrupt(type)	ICM20948_FSYNC_INT ICM20948_WOM_INT ICM20948_DATA_READY_INT ICM20948_FIFO_OVF_INT	Should be self-explaining (see also enableInterrupt)
bool checkInterrupt(source, type)	source (ICM20948_intType), interrupt type	If an interrupt occurred you might want to check if it was data ready, fifo overflow or wake-on-motion. readAndClearInterrupts() returns the source, but as an enum: ICM20948_intType. Either you look up in ICM20948_WE.h how it is defined or you check with this function. The disadvantage is that you need to check one by one.
uint8_t readAndClearInterrupts( )	none	Returns which interrupt occurred as ICM20948_intType and clears the interrupt.
void setWakeOnMotionThreshold (thresh, mode)	threshold (1…255) ICM20948_WOM_COMP_DISABLE/ ICM20948_WOM_COMP_ENABLE	Sets the threshold for the wake-on-motion interrupt. The LSB is 4 mg (= milli-g), i.e. 1 equals 4 mg, 255 equals 1020 mg. Enables/disables the compare mode. In compare mode the current acceleration value is compared with the last measured value. If compare mode is disabled the baseline is the starting value, when the WOM interrupt was enabled.
void startFifo(type)	ICM20948_FIFO_ACC, ICM20948_FIFO_GYR, ICM20948_FIFO_ACC_GYR	If called, the ICM20948 starts writing data into the Fifo. Fifo must be enabled before. I have implemented three options: you can write acceleration data (max 85 x,y,z sample sets), gyroscope data (max 85 x,y,z sample sets) or acceleration and gyroscope data (max 42 sets) into the Fifo.
void stopFifo( )	none	Stops writing data into the Fifo.
void enableFifo(true/false)	true / false	Enables / disables the Fifo function.
void resetFifo( )	none	Sets the Fifo counter to zero.
int16_t getFifoCount( )	none	Returns the number of bytes in the Fifo. According to the data sheet the FIFO should be max. 512 bytes. However I found 4096. This could be related with the DMP which I have not implemented.
void setFifoMode(mode)	ICM20948_CONTINUOUS, ICM20948_STOP_WHEN_FULL	Sets continuous or stop-when-full mode. In continuous mode new data is continuously written into the Fifo. If full, the oldest data is replaced by new data. In the other mode, no new data is written to the Fifo is full.
int16_t getNumberOfFifoDataSets( )	none	Returns the number of complete data sets in the Fifo. E.g. a complete set of acceleration and gyroscope data consists of 2 (acc & gyro) x 3 (x,y,z) x 2 (2 bytes) = 12 bytes. If Fifo is full it contains 512 byte or 4096 bytes -> 512 / 12 = 42 complete sets, rest is 8. -> 4096 / 12 = 341 complete sets, rest is 4.
void findFifoBegin( )	none	In the stop-when-full mode the Fifo will start at the beginning of a set. The last set will be incomplete. In the continuous it will end with a complete set. That means in continuous mode you (or the library) has to calculate at which byte (fifo count) the first complete set starts.
bool initMagnetometer( )	none	Initiates the magnetometer and reads adjustment factors from the ROM (which have been added there by the manufacturer).
void setMagOpMode(mode)	AK09916_PWR_DOWN AK09916_TRIGGER_MODE AK09916_CONT_MODE_10HZ AK09916_CONT_MODE_20HZ AK09916_CONT_MODE_50HZ AK09916_CONT_MODE_100HZ	Sets the operational mode for the magnetometer (AK8963): power down, triggered mode (saves power vs. continuous mode), continuous mode at 10, 20, 50 or 100 Hz.
int16_t whoAmIMag( )	none	Returns the ID of the AK09916, wich should be 0x4809.