Low Power Modes and Auto-Wake/Sleep Using the MMA8450Q

by: Kimberly Tuck
Applications Engineer

1.0 Introduction

Accelerometers are commonly used in hand held electronics and/or battery operated electronic devices. Consumption of current in the entire system is a critical feature of the product design. Users do not want to be inconvenienced by continually recharging or changing out batteries. When designing in the accelerometer, battery power usage is often a critical feature which concerns many customer-users. Therefore, current consumption of the sensor as well as of the entire system should be paramount design considerations. If the system processor is used often only for processing data from the accelerometer, then it is ideal to embed the intelligence in the sensor to avoid burdening the system processor from running continually. The flexibility of embedded interrupt driven functions and selectable data rates with trade-offs for resolution, response time, and current are the types of intelligent features in the MMA8450Q.

This application note aims to explain the following:

- The Low Power Mode bit and the trade off with noise in the accelerometer
- Noise vs. current consumption at all different Output Data Rates
- The comparisons of the different current consumption levels running the device at 6 different available sample rates and the trade off with resolution vs. current consumption
- Importance of embedded functions and how the overall current consumption of the system can be lowered compared to algorithm driven functions analyzing XYZ data alone
- Reducing overall system power using the FIFO
- An explanation of the Auto-Wake/Sleep feature, explaining the configuration procedure with example register settings and code

TABLE OF CONTENTS

1.1 Key Words

Accelerometer, Output Data Rate, Current, Standby Current, Power Down Mode Current, Low Power Mode, Noise, Auto-Wake/Sleep, Sleep Timer, Power Cycling, FIFO, MCU, Processor, Sensor

1.2 Summary

A. When there is a need for low power over high resolution the MMA8450Q is capable of reducing the current consumption of the part at all ODRs resulting in significant overall system power savings.

B. The effective number of bits in Normal Mode and Low Power Mode are given with the current consumption for each condition.

C. The embedded functions allow the system MCU or processor to go to sleep and wait for an interrupt from the accelerometer. The processor does not need to continuously access and monitor data. This has significant benefits over continuous polling of XYZ data and can save over 96% of the total current consumption allowing wireless products to last much longer on a battery.

D. The FIFO has tremendous power savings potential for applications requiring data-logging or when waiting for an event to view the exact data that triggered the event. Instead of accessing the data per every sample, the processor/MCU can go to Sleep Mode and wake up only to flush the data when the FIFO is full or if an interrupt has occurred. Current consumption savings can range from 78% up to 96% or higher depending on conditions of the MCU and ODR chosen.

E. The MMA8450Q can be used to cycle between different ODRs, which results in overall lower current consumption of the device. This can be achieved from 5 programmable functions.

2.0 MMA8450Q Consumer 3-axis Accelerometer 3 x 3 x 1 mm

The MMA8450Q has a selectable dynamic range of ±2g, ±4g and ±8g with sensitivities of 1024 counts/g, 512 counts/g and 256 counts/g respectively. The device offers either 8-bit or 12-bit XYZ output data for algorithm development. The chip shot and pinout are shown in Figure 1.

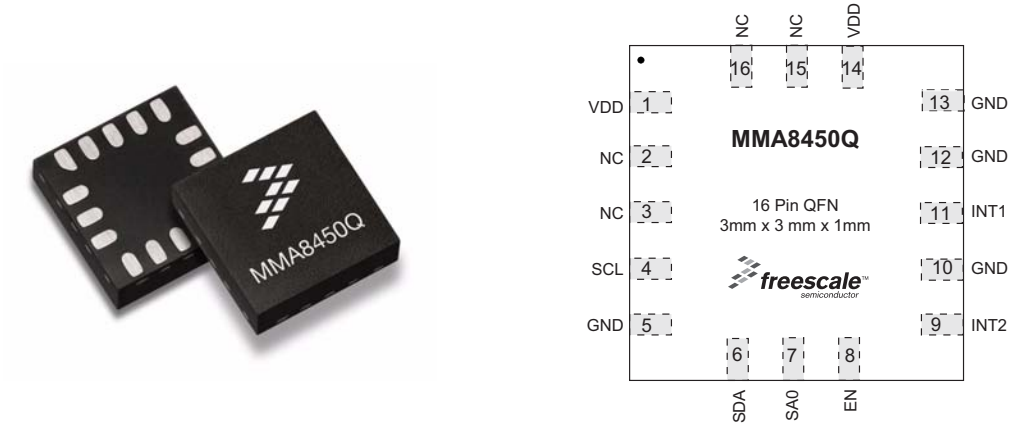

Figure 1. MMA8450Q Consumer 3-axis Accelerometer 3 x 3 x 1 mm

2.1 Key Features of the MMA8450Q

1. Shutdown Mode: Typical < 1 µA, Standby Mode 3 µA
2. Low Power Mode current consumption ranges from 27 µA (1.56 - 50 Hz) to 120 µA (400 Hz)
3. Normal Mode current consumption ranges from 42 µA (1.56 - 50 Hz) to 225 µA (400 Hz)
4. I^2C digital output interface (operates up to 400 kHz Fast Mode)
5. 12-bit and 8-bit data output, 8-bit high pass filtered data output
6. Post Board Mount Offset < ±50 mg typical
7. Self Test X, Y and Z axes

2.2 Two (2) Programmable Interrupt Pins for 8 Interrupt Sources

1. Embedded 4 channels of Motion detection
   a. Freefall or Motion detection: 2 channels
   b. Tap detection: 1 channel
   c. Transient detection: 1 channel
2. Embedded orientation (Portrait/Landscape) detection with hysteresis compensation
3. Embedded automatic ODR change for auto-wake-up and return-to-sleep
4. Embedded 32 sample FIFO
5. Data Ready Interrupt

2.3 Application Notes for the MMA8450Q

The following is a list of Freescale Application Notes written for the MMA8450Q:

- **AN3915**, *Embedded Orientation Detection Using the MMA8450Q*
- **AN3916**, *Offset Calibration of the MMA8450Q*
- **AN3917**, *Motion and Freefall Detection Using the MMA8450Q*
- **AN3918**, *High Pass Filtered Data and Transient Detection Using the MMA8450Q*
- **AN3919**, *MMA8450Q Single/Double and Directional Tap Detection*
- **AN3920**, *Using the 32 Sample First In First Out (FIFO) in the MMA8450Q*
- **AN3921**, ***Low Power Modes and Auto-Wake/Sleep Using the MMA8450Q***
- **AN3922**, *Data Manipulation and Basic Settings of the MMA8450Q*
- **AN3923**, *MMA8450Q Design Checklist and Board Mounting Guidelines*

3.0 Low Power Mode Compared to Normal Mode

Table 1 shows the different current consumption levels at different selectable Output Data Rates. The Low Power Mode is set in Register 0x39 System Control Register 2, bit 0. If this bit is cleared, the device is in the Normal Mode. When this bit is set, the device is in the Low Power Mode. Notice that in Low Power Mode the current consumption drops but this advantage comes at the expense of higher noise. The typical noise calculated in mg RMS is given for all different available sample rates. The equivalent effective number of noise free bits of the 12-bit data is given for each based on tested data. The Low Power Mode reduces the current consumption by internally sleeping longer and averaging the data less. The change in effective number of bits is approximately 0.6 to 0.7 bits. For an application requiring the highest resolution with the lowest current consumption, the trade-off will need to be made.

Also note that when comparing the current consumption at different sample rates, the current consumption remains the same from 1.56 Hz to 50 Hz. This is a trade off between current consumption and noise. At the lower sample rates, the device is averaging data to improve the noise performance. At 1.56 Hz the device averages 32 more samples than at 50 Hz; this improves the noise performance. At 50 Hz in Normal Mode, the device typically has 7.8 effective (noise free) bits and at 1.56 Hz, the device has 10.2 effective (noise free) bits.

Table 1. Current Consumption for Different Sample Rates Normal and Low Power Mode

Symbol	Parameter	Test conditions	Typ Noise mg RMS	Effective Bits*	Typ Current (μA)
I_{dd}LP	Low Power Mode $39 CTRL_REG2: MOD[0] = 1	EN = 1, ODR = 1.563 Hz	1.5	9.6	27
		EN = 1, ODR = 12.5 Hz	4.0	8.2	27
		EN = 1, ODR = 50 Hz	8.0	7.2	27
		EN = 1, ODR = 100 Hz	8.0	7.2	42
		EN = 1, ODR = 200 Hz	8.0	7.2	72
		EN = 1, ODR = 400 Hz	8.0	7.2	120
I_{dd}	Normal Mode $39 CTRL_REG2: MOD[0]=0	EN =1, ODR = 1.563 Hz	1.0	10.2	42
		EN =1, ODR = 12.5 Hz	2.5	8.9	42
		EN =1, ODR = 50 Hz	5.2	7.8	42
		EN =1, ODR = 100 Hz	5.2	7.8	72
		EN =1, ODR = 200 Hz	5.2	7.8	132
		EN =1, ODR = 400 Hz	5.2	7.8	225
I_{dd}Sdn	Current consumption in Shutdown Mode	EN = 0	—	—	1
I_{dd}Stby	Supply Current Drain in Standby Mode	EN = 1 and FS[1:0] = 00	2	—	3

*Note the noise values in the chart are based on real data on the MMA8450Q Demo Board, taken at a typical workstation, not in an isolated noise free environment.

4.0 Power Savings of the MMA8450Q in the End System Application

The consideration of power savings is often specific to the application. Most applications for the accelerometer are in portable devices using a battery power supply. Battery life is paramount and the ability to minimize power consumption depends upon the performed operations required in the application. In most scenarios, the preference should be to shut down everything and only wake up and do what is required as quickly and as efficiently as possible. Often, this will depend upon a user display and how long the display needs to be “on” as well as how the unit is capable of being woken up.

Sometimes, if the processor needs to be up and running continuously, it may be possible to save power by “gear shifting” the bus clock speed, that is, switching between fast and slow clocking modes rather than jumping between run and stop. The embedded FIFO is a proven benefit as it limits how often the processor needs to read the data. The FIFO is also an advantage in non-battery powered applications as it can improve computational throughput, again, by not needing to interrupt the processor every time there is a new sample.

Most, if not all, MCUs/processors can leave a sleep state via an external interrupt, which is how the MMA8450Q could be used for “wake on shake”, or “wake on tilt”, etc. This is where the advanced features of the MMA8450Q prove to be beneficial. Several MCUs/processors can also wake up via an internal interrupt, usually based upon a timer interval – i.e. wake up every 100 ms, etc. This can be required in order to perform some regular housekeeping functions (such as time of day, etc.) which could include scanning the accelerometer and processing its data via software. Switching off the power to the MCU rarely makes any sense compared to waking up from sleep because this is always faster than a cold start. Wake up times can vary considerably depending on the MCU or processor. For example, some Freescale 8-bit MCUs have the ability to wake up from sleep/stop mode in 6 µs, while other processors may take around 3 ms. Very fast wake-up times on the MCU/processor make it very efficient for switching between sleep and wake states.

The MMA8450Q has many embedded functions in the device which relieve the host processor from having to continuously sample the XYZ data and run various algorithms for motion detection, orientation detection, freefall, or quick jolts. The device has the intelligence internally to recognize any of these embedded events and can change the sample rate upon detection. For example, in a remote controller application, much of the time the remote will sit on the table motionless while no one is using it. The MMA8450Q can be configured to be at a low sample rate (50 Hz) when in sleep mode and then when the user picks up the remote the accelerometer will switch over into a faster sample rate (400 Hz) in wake mode ready to recognize faster moving motion signatures. The embedded blocks to keep the device awake must be enabled and configured. For example, the orientation detection can be configured to wake the device along with motion detection. All changes in orientation or motions will keep the device at the higher sample rate. When the device stops moving it will go back to the sleep state to conserve power. The details of how to configure the Auto-Wake/Sleep are described in Section 5.0.

4.1 Power Savings Using the FIFO Data Logging

The FIFO is very beneficial for saving overall system power by putting the processor into sleep mode until it needs to process data from the accelerometer. The idea is to configure the MMA8450Q to monitor a desired interrupt, putting the processor in a low power mode until it needs to respond to the accelerometer. This maximizes the time that the processor spends in a sleep or low power mode and ultimately will minimize the system’s overall power consumption, increasing the life of the battery. The FIFO allows the processor to sleep longer while samples are being collected inside the sensor. This also minimizes the traffic across the I^2C bus.

The timing of the data rate and the bus speed should be chosen with care. As an example the accelerometer is put in Low Power Mode sampling at 50 Hz (20 ms) with the FIFO running in Fill Mode and the FIFO interrupt enabled. The interrupt would be used to trigger the processor to wake up, service the interrupt, and flush the 32 samples. New data cannot be stored into the FIFO while it is being flushed. Therefore the processor must wake up, service the interrupt and flush the data within 20 ms before the next sample is available.

The FIFO overflow is asserted every 32 samples. The user has the option of flushing either the 12-bit data or the 8-bit data. For the 12-bit data each sample consists of three 12-bit values, each stored as 2 bytes. Therefore, when full, the FIFO will contain 192 bytes. For the 8-bit data each sample consists of three 8-bit values, each stored as 1 byte. Therefore in this case when full the FIFO will contain 96 bytes. An I^2C burst access has about 3 extra bytes of “overhead”, for a total of 195 bytes in the 12-bit data flush and 99 bytes in total for the 8-bit data flush. Also the Start, Stop and Repeat Start I^2C transactions take a minimum of 0.6 µs. A 1 µs value will be added for each of these. Assuming that each I^2C byte requires a 10-bit transfer window (8 for data, 1 for the acknowledge and 1 for bus idle), the time required to perform an I^2C burst read of N samples can be calculated as follows:

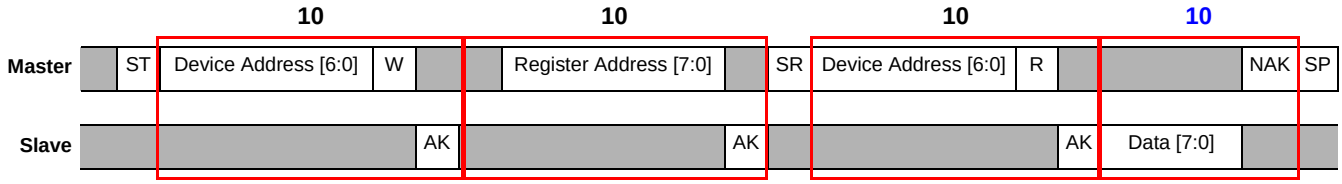

Figure 2. I^2C Single Byte Read Transaction

12- bit Data Flush Calculations

FIFORead(N) = ((((N · 3 · 2) + 3) · 10) / I^2C bit rate)

FIFORead(32) = 1950 / I^2C bit rate

For an I^2C bit rate of 400 kHz, FIFORead(32) + 3 µs = 4.878 ms.

For an I^2C bit rate of 400 kHz, FIFORead(16) + 3 µs = 2.478 ms.

8-bit Data Flush Calculations

FIFORead(N) = ((((N · 3) + 3) · 10) / I^2C bit rate)

FIFORead(32) = 990 / I^2C bit rate

For an I^2C bit rate of 400 kHz, FIFORead(32) + 3 µs = 2.478 ms.

Note that bursting out 32 samples of 12-bit XYZ data consecutively takes 1950 bits to perform the transaction. By bursting XYZ 12-bit data each time new data is ready requires 2880 bits, as this requires 32 iterations. The start, stop and repeat start transactions calculated at 1 µs each start to add up over 32 iterations.

DataReadyRead(32) = (((3 · 2)+ 3) · 10) · 32 = 2880 bits/I^2C bit rate

For an I^2C bit rate of 400 kHz, DataReadyRead(32) = 7.2 ms + 3 µs · 32 = 7.296 ms

It is seen that using the FIFO to pull out all 32 samples at one time saves on the overhead. This allows the application processor to do other things or to remain in a low power mode for longer.

Example conditions are given for a processor with the wake timing and current consumption values in Table 2. In Wake Mode the example processor uses a total of 12 mA, while in sleep mode it only uses 0.5 mA. It take 3 ms to wake the processor from sleep and read the FIFO status. As shown above, a 12-bit data flush from the accelerometer FIFO takes close to 5 ms while an 8-bit flush of the FIFO takes 2.5 ms. With these example conditions, the average current consumption and the percentage of saved current consumption can be calculated.

Note: current consumption and wake times on different processors/MCUs will vary but this same methodology applies. The next several sections will show the analysis of flushing the FIFO at different sample rates using the assumed conditions from Table 2.

Table 2. Example Conditions

Wake-Up Time	12-bit Flush	8-bit Flush	Sleep Mode	Wake Mode
3 ms	5 ms	2.5 ms	0.5 mA	12 mA

4.1.1 Flushing the FIFO at 100 Hz ODR and Below

At 100 Hz (or less) output data rate the processor can wake up and flush the FIFO without missing any samples. The following is a timing diagram typical of how the FIFO and processor would be configured for sample rates 100 Hz or less. The FIFO collects data until the overflow flag interrupt is asserted. Then the processor wakes up and flushes all the data out of the FIFO before the next sample is ready. From sample 1 to sample 32 the processor is in sleep mode. **Note:** The processor will also be asleep during the later part of the interval before the first sample is ready. The details of the sleep to wake timing are captured below in Figure 3 and Table 3. The total current is calculated assuming 0.5 mA in sleep mode and 12 mA in active mode.

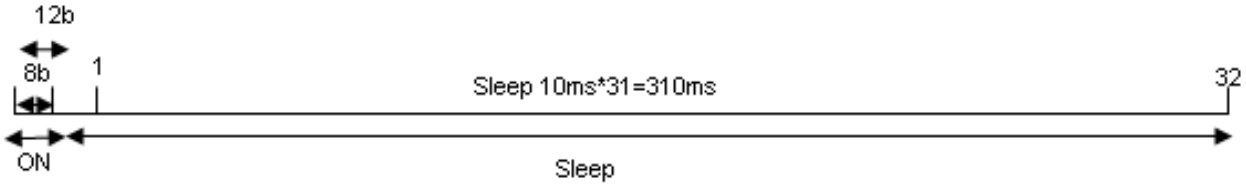

Figure 3. Timing of the FIFO at 100 Hz ODR Showing Sleep and Wake Timing

Table 3. Wake to Sleep Timing at 100 Hz ODR

Data	ODR	Total Time	Sleep Time	Wake Time	Watermark	Wake/Total	Sleep/Total	Total Current	Current Savings
12-bit	100	320 ms	312 ms	8 ms	32	8/320	312/320	0.7875 mA	93.4%
8-bit	100	320 ms	314.5 ms	5.5 ms	32	5.5/320	314.5/320	0.6978 mA	94.2%

4.1.2 Flushing the FIFO at 200 Hz ODR

When the data rate is set to 200 Hz the processor can be triggered by the watermark set at 31 samples, giving 5 ms to turn on, which is more than enough time. Then when the overflow flag asserts the FIFO is flushed, which takes almost 5 ms for flushing the 12-bit data and 2.5 ms for flushing the 8-bit data. The FIFO can be flushed at 200 Hz ODR without missing any samples by waking up from the Watermark interrupt set at sample 31 as shown in Figure 4. If the 12-bit data flush takes longer than the 5 ms then the first sample of the next data set will be missed. The results of the sleep to wake timing and current drain are captured in Table 4.

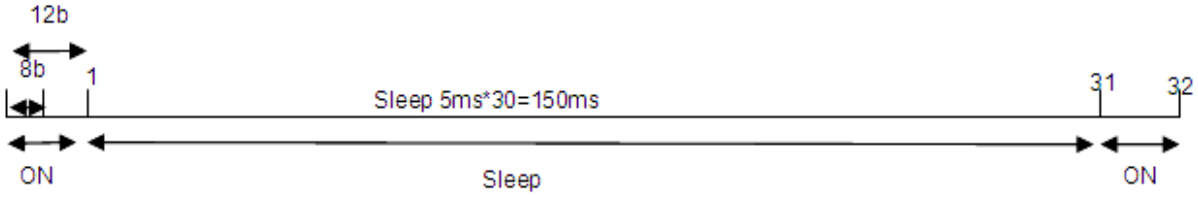

Figure 4. Timing of the FIFO at 200 Hz ODR Showing Sleep and Wake Timing

Table 4. Wake to Sleep Timing at 200 Hz ODR

Data	ODR	Total Time	Sleep Time	Wake Time	Watermark	Wake/Total	Sleep/Total	Total Current	Current Savings
12-bit	200	160 ms	150 ms	10 ms	31	10/160	150/160	1.218 mA	89.8%
8-bit	200	160 ms	152.5 ms	7.5 ms	31	7.5/160	152.5/160	1.039 mA	91.3%

4.1.3 Flushing the FIFO at 400 Hz ODR

When sampling at 400 Hz, there is a new sample every 2.5 ms, which does not allow a lot of time to wake and flush without missing samples. At 400 Hz the best way to configure the FIFO to avoid losing data is to set the Watermark for 30 samples. This is the trigger to interrupt the processor to wake up. Then, when the overflow flag is asserted, a 16 sample (12-bit data) flush occurs, which takes 2.475 ms. Next, the processor will go immediately to sleep and continue cycling through this pattern, waking up at the watermark then flushing the last 16 samples when the overflow flag asserts. When flushing 8-bit samples the FIFO should have enough time to flush the entire buffer. Figure 5 shows the timing for flushing 12-bit data at 400 Hz ODR. Figure 6 shows the timing for the 8-bit data flush at 400 Hz.

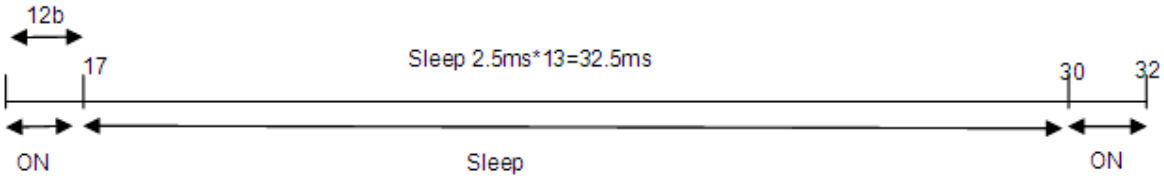

Figure 5. Timing of the FIFO at 400 Hz ODR Flushing 12-bit data Showing Sleep and Wake Timing

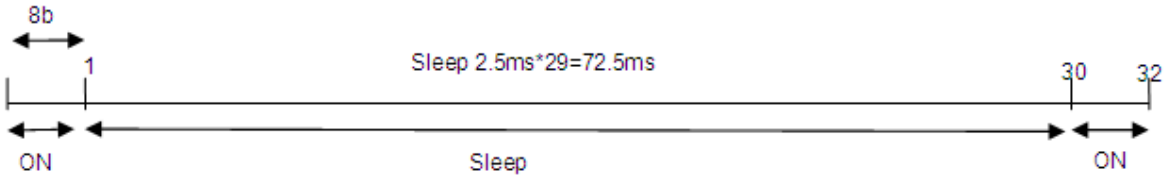

Figure 6. Timing of the FIFO at 400 HZ ODR Flushing 8-bit data Showing Sleep and Wake Timing

Table 5 presents all the calculations at 400 Hz flushing 12-bit data and 8-bit data without missing samples.

Table 5. Wake to Sleep Timing at 400 Hz ODR

Data	ODR	Total Time	Sleep Time	Wake Time	Watermark	Wake/Total	Sleep/Total	Total Current	Current Savings
12-bit	400	40 ms	32.5 ms	7.5 ms	14	7.5/40	32.5/40	2.656 mA	77.9%
8-bit	400	80 ms	72.5 ms	7.5 ms	30	7.5/80	72.5/80	1.578 mA	86.9%

Table 6 summarizes the wake and sleep timing for all sample rates of the MMA8450Q. The total current consumed per cycle and the current savings as a percentage are calculated based on the amount of time the processor is in wake vs. sleep.

Table 6. Power Savings Using FIFO at Different Data Rates

ODR	Time between Samples	Sleep/Total Ratio 12-bit	Current Consumption 12-bit Data Flush mA	Current Savings 12-bit Data (%)	Sleep/Total Ratio 8-bit	Current Consumption 8-bit Data Flush mA	Current Savings 8-bit Data (%)
1.56 Hz	641 ms	99.96%	0.505	95.8%	99.97%	0.503	95.8%
12.5 Hz	80 ms	99.69%	0.536	95.5%	99.79%	0.524	95.6%
50 Hz	20 ms	98.75%	0.644	94.6%	99.14%	0.599	95.0%
100 Hz	10 ms	97.50%	0.788	93.4%	98.28%	0.698	94.2%
200 Hz	**5 ms**	**93.75%**	**1.219**	**89.8%**	**95.31%**	**1.039**	**91.3%**
400 Hz	**2.5 ms**	**81.25%**	**2.656**	**77.9%**	**90.63%**	**1.578**	**86.9%**

From Table 6, these values can be related to the amount of time that a typical lithium ion battery for a cell phone would last. This gives a representation of power savings related to battery life time. The percentage saved for current consumption is for the application processor only. Table 7 incorporates the current consumption of the processor and the accelerometer in full power mode to give the average total current consumption. An example lithium-ion cell phone battery stores 1200 mA hours. Based on this information a comparison is made. This shows the total current consumption (processor + accelerometer) at all sample rates when the processor is continuously polling data and therefore always in the wake state.

Table 7. Example Li-Ion Battery Life Calculations without the FIFO to Data Log Data

ODR LP-Low Power N-Normal	Processor Current Consumption	MMA8450Q Current Consumption	Total Consumption	AA Li-Ion Battery Life 1200 mAh (1200 mAh/Total mA) Time (h)	Time (Days)
1.56 Hz (LP)	12	0.027	12.027	99.77	4.16
1.56 Hz (N)	12	0.042	12.042	99.65	4.16
12.5 Hz (LP)	12	0.027	12.027	99.77	4.16
12.5 Hz (N)	12	0.042	12.042	99.65	4.16
50 Hz (LP)	12	0.027	12.027	99.77	4.16
50 Hz (N)	12	0.042	12.042	99.65	4.16
100 Hz (LP)	12	0.042	12.042	99.65	4.15
100 Hz (N)	12	0.072	12.072	99.40	4.14
200 Hz (LP)	12	0.072	12.072	99.40	4.14
200 Hz (N)	12	0.132	12.132	98.91	4.12
400 Hz (LP)	12	0.120	12.120	99.01	4.13
400 Hz (N)	12	0.225	12.225	98.16	4.09

When the processor is continuously running, the accelerometer current consumption has a small effect on the battery life because the processor uses much more current than the accelerometer. The ability to use the accelerometer to put the processor in a sleep mode can have a significant impact on the battery life (Table 8). The current consumption of the processor is based on the 12-bit data that was explained from Table 6. Note in Table 8 the far column on the right displays the battery life improvement by using the FIFO to data log the data. This shows that at the highest sampling rate in Normal Mode the battery life improves 4.2x what it would by polling the data with the processor continually running. At the lowest sample rate in Low Power Mode, the savings is 22.6x longer battery life.

Table 8. Example Li-Ion Battery Life Calculations Using the FIFO to Data Log 12-bit Data

ODR LP-Low Power N-Normal	Processor Current Consumption 12-bit Data Flush mA	MMA8450Q Current Consumption 12-bit Data Flush mA	Total Consumption 12-bit Data Flush mA	Li-Ion Battery 1200 mAh (1200 mAh/Total mA) Time (h)	Time (Days)	Battery Life Improvement (x Longer)
1.56 Hz (LP)	0.505	0.027	0.532	2255.64	93.98	22.6
1.56 Hz (N)	0.505	0.042	0.547	2193.78	91.41	22.0
12.5 Hz (LP)	0.536	0.027	0.563	2131.44	88.81	21.4
12.5 Hz (N)	0.536	0.042	0.578	2076.12	86.51	20.8
50 Hz (LP)	0.644	0.027	0.671	1788.38	74.52	17.9
50 Hz (N)	0.644	0.042	0.686	1749.27	72.89	17.6
100 Hz (LP)	0.788	0.042	0.830	1445.78	60.24	14.5

Table 8. Example Li-Ion Battery Life Calculations Using the FIFO to Data Log 12-bit Data

100 Hz (N)	0.788	0.072	0.860	1395.35	58.14	14.0
200 Hz (LP)	1.219	0.072	1.291	929.51	38.73	9.4
200 Hz (N)	1.219	0.132	1.351	888.23	37.01	9.0
400 Hz (LP)	**2.656**	**0.120**	**2.776**	**432.28**	**18.01**	**4.4**
400 Hz (N)	**2.656**	**0.225**	**2.881**	**416.52**	**17.36**	**4.2**

Note: A similar analysis can be done for the 8-bit data using the FIFO.

4.2 Power Savings Using the FIFO to Collect the History Leading up to an Event Trigger

Another use for the FIFO is the ability to analyze the data that occurred right up to the point of an interrupt triggering event. After the interrupt flag of the event is set, the FIFO (configured in Circular Mode) can be flushed to extract the previous 32 samples of data leading up to the event. If it is desirable for the FIFO to hold the data in the FIFO after the interrupt, then this can only be done when there is a transition from Wake to Sleep Mode only. Otherwise the FIFO must be flushed after the event to store the data in the processor for further analysis. This technique is discussed in AN3919 for analyzing directional tap. The single tap is configured and the FIFO is configured for Circular Buffer Mode to run at 400 Hz. When the tap interrupt flag is set, the FIFO is read within 15 ms of the interrupt to collect the full signature of the tap to analyze the data leading up to the event, and the data during the event. This technique can be particularly important when tracking events over a long period of time. The MCU or processor can remain asleep until the event has triggered and it can add up to substantial power savings.

5.0 Configuring the MMA8450Q into Auto-Wake/Sleep Mode

The MMA8450Q can be configured to transition between different sample rates (different current consumption) based on different selected events. Enabling this feature can be accomplished by enabling the Sleep Mode and setting a time-out period. Then the functions of interest must be set to trigger the device to wake. The advantage of using the Auto-Wake/Sleep is that the system can automatically transition to a higher sample rate (higher current consumption) when needed but spends the majority of the time in the Sleep Mode (lower current) when the device does not require higher sampling rates. This can all be triggered on selected events. The Low Power Mode bit (Reg 0x39 bit 0) can be used as well with this feature to minimize the total current consumption. Figure 7 shows transition states.

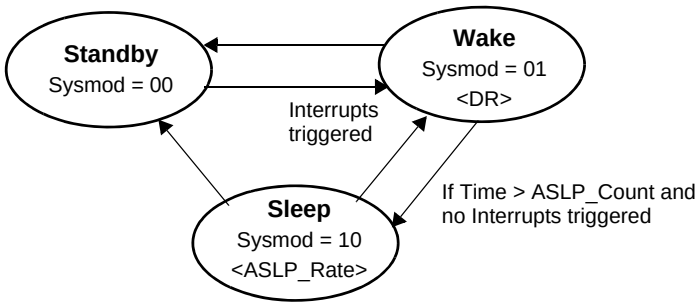

Figure 7. System Modes

The selected embedded functions must be enabled and the same corresponding functions must be set to “Wake From Sleep” if they are to be used to wake the device.

All enabled functions will still function in Sleep Mode at the sleep ODR. Only the functions that have been selected for “Wake From Sleep” will wake the device.

This section reviews the different registers involved in configuring the device for auto-wake/sleep.

1. Register 0x39 bit 1 – **SLPE** Enable Sleep bit
2. Register 0x38 Sleep Sample Rate and Wake Sample Rate
3. Register 0x37 Time Out Counter
4. Register 0x3B Enable the Interrupts for the Selected Functions
5. Register 0x3C Route the Interrupts to INT1 or INT2
6. Register 0x3A Enable the Wake from Sleep Interrupts

5.1 Set the Sleep Enable Bit

If the Sleep Enable bit (Reg. 0x39 bit 1) is not enabled then the device can only toggle between Standby and Wake Mode by writing to the FS0 and FS1 bits in Register 0x38. When the Sleep Enable bit is enabled the device can transition between Standby, Wake, and Sleep. The **SLPE** bit is shown as bit 1 in Table 9.

Table 9. 0x39 CTRL_REG2 Register (Read/Write) and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ST	BOOT	0	0	0	0	**SLPE**	MODS

5.2 Configure the Sleep Sample Rate

It is important to note that when the device is in Sleep Mode, the system ODR and the data rate for all the system functions are overwritten by the data rate set by the **ASLP_RATE** field in the **CTRL_REG1** register (0x38). The Sleep Sample Rate and the Normal Mode Sample Rate are found in Table 10. The different bit settings for the Sleep Sample Rate can be found in Table 11.

Table 10. 0x38 CTRL_REG1 Register (Read/Write) and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ASLP_RATE1	ASLP_RATE0	0	DR2	DR1	DR0	FS1	FS0

Table 11. Sleep Mode Sample Rate Description

ASLP_RATE1	ASLP_RATE0	Frequency (Hz)
0	0	50
0	1	25
1	0	12.5
1	1	1.56

Example:

- Sleep Period = 1/ASLP_RATE
- If ASLP_RATE = 50 Hz, Sleep Period = 20 ms

5.3 Set the Time Out Counter

The **ASLP_COUNT** register sets the minimum time period of inactivity required to change current ODR value from the value specified in the **DR[2:0]** to **ASLP_RATE** (Reg 0x38) value provided the **SLPE** bit is set to a logic ‘1’ in the **CTRL_REG2** register.

Table 12. 0x37 ASLP_COUNT Register (Read/Write) and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
D7	D6	D5	D4	D3	D2	D1	D0

D7-D0 defines the minimum duration time to change current ODR value from **DR** to **ASLP_RATE**. Time step and maximum value depend on the ODR chosen. See Table 13.

Table 13. ASLP_COUNT Relationship with ODR

Output Data Rate (ODR)	Duration	Step
400	0 to 81s	320 ms
200	0 to 81s	320 ms
100	0 to 81s	320 ms
50	0 to 81s	320 ms
12.5	0 to 81s	320 ms
1.56	0 to 325.125s	640 ms

5.4 Enable the Interrupts to be used in the System and Route to INT1 or INT2

The functions must be enabled in Register 0x3B per Table 14 for the event to trigger the Auto-Wake/Sleep.

Table 14. Register 0x3B CTRL_REG4 Register (Read/Write) and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INT_EN _ASLP	INT_EN _FIFO	INT_EN _TRANS	INT_EN _LNDPRT	INT_EN _PULSE	INT_EN_FF_MT_1	INT_EN_FF_MT_2	INT_EN_DRDY

The corresponding interrupt enable bit allows the function to route its event detection flag to the interrupt controller. The interrupt controller routes the enabled interrupt to the INT1 or INT2 pin. By default all interrupts are routed to INT2 and the corresponding configuration register bit value is 0. To route a functional block to INT1 instead of the default, set the corresponding configuration register bit to 1. The configuration register bit settings are shown in Table 15.

Table 15. Register 0x3C CTRL_REG5 Register (Read/Write) and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INT_CFG_ASLP	INT_CFG_FIFO	INT_CFG_TRANS	INT_CFG_LNDPRT	INT_CFG_PULSE	INT_CFG_FF_MT_1	INT_CFG_FF_MT_2	INT_CFG_DRDY

5.5 Enable the Interrupt Sources that Wake the Device

The register to control which interrupts will wake the device are configured in Register 0x3A shown below in Table 16. **There are six (6) functional blocks that can be used to keep the sensor from falling asleep if they are enabled.** These are the Transient, Orientation, Tap, Motion/FF1 and Motion/FF2 and the FIFO. **There are only five (5) functions used to wake the device.** The FIFO will not wake the device from sleep. Also note the auto-wake/sleep interrupt and the data ready interrupt do not affect the wake/sleep.

Table 16. Register 0x3A CTRL_REG3 Interrupt Control Register and Description

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FIFO_GATE	WAKE_TRANS	WAKE_LNDPRT	WAKE_PULSE	WAKE_FF_MT_1	WAKE_FF_MT_2	IPOL	PP_OD

Note: The FIFO is flushed whenever the system ODR changes in order to prevent mixing the FIFO data from different time domains unless the FIFO_GATE (bit 7) is set. Also, the FIFO cannot wake the device from sleep but can prevent the device from going to sleep. Details of the functionality of the FIFO is captured in Table 17.

Table 17. Behavior of FIFO under Wake/Sleep Conditions

FIFO INT ENABLED	Wake from Sleep Enabled	Result (Assuming that the FIFO is set up to accept samples in either Fill or Circular Mode)
NO	NO	• FIFO will fall asleep when the sleep timer times out and no other interrupt wakes the system. • There is an AUTOMATIC flush and the FIFO starts refilling at the Sleep ODR from 0. • If another functional block causes the device to wake, the FIFO will FLUSH itself again and start filling at Wake ODR.
YES	NO	• With the interrupt enabled, the FIFO can be read and flushed (clearing the interrupt) to keep the device from falling asleep. This is dependant on the sleep time out value and how fast the FIFO is clearing the interrupt. • If the system does fall asleep, *(and no interrupts occur during the time-out period)*, the FIFO AUTOMATICALLY flushes and starts refilling at the Sleep ODR from 0. • When the device wakes up again by an interrupt, the FIFO AUTMOATICALLY flushes and starts from 0 and stores at the Wake ODR.
NO	YES	• FIFO will fall asleep if no wake events occur within the time out period. • Last data remains in the FIFO until it is flushed. • Once the FIFO is flushed, it will start collecting the new data at the current ODR.
YES	YES	• With interrupt enabled, the FIFO can be read and flushed (clearing the interrupt) to keep the device from falling asleep. • If the system does fall asleep, *(and no interrupts occur during the time out period)*, then the FIFO will stop collecting any data. • The last data will be held in the FIFO. • Once the FIFO is flushed, it will start collecting the new data at the current ODR.

6.0 Example Configuration for the Auto-Wake/Sleep Function

The following are the steps to configure the Auto-Wake/Sleep function with the registers of importance in Table 18. In this example the data rate will be set to 400 Hz in Active Mode and 12.5 Hz in Sleep Mode. The time-out period will be set to 20 seconds. The wake triggers will be tap and motion. There may be other interrupts that are enabled in the system including orientation detection, but these will not wake the device in this example.

Table 18. Registers used for Auto-Wake/Sleep Functionality

Reg	Name	Definition	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
14	SYSMOD	System Mode R	PERR	FGERR	0	0	0	0	SYSMOD1	SYSMOD0
15	INT_SOURCE	Interrupt Status R	SRC_ASLP	SRC_FIFO	SRC_TRANS	SRC_LNDPRT	SRC_PULSE	SRC_FF_MT_1	SRC_FF_MT_2	SRC_DRDY
37	ASLP_COUNT	Auto-Sleep Counter R/W	D7	D6	D5	D4	D3	D2	D1	D0
38	CTRL_REG1	Control Reg 1 R/W	ASLP_RATE1	ASLP_RATE0	0	DR2	DR1	DR0	FS1	FS0
39	CTRL_REG2	Control Reg 2 R/W	ST	RST	0	0	0	0	SLPE	MODS
3A	CTRL_REG3	Control Reg3 R/W (Wake Interrupts from Sleep)	FIFO_GATE	WAKE_TRANS	WAKE_LNDPRT	WAKE_PULSE	WAKE_FF_MT_1	WAKE_FF_MT_2	IPOL	PP_OD
3B	CTRL_REG4	Control Reg4 R/W (Interrupt Enable Map)	INT_EN _ASLP	INT_EN _FIFO	INT_EN _TRANS	INT_EN _LNDPRT	INT_EN _PULSE	INT_EN _FF_MT_1	INT_EN _FF_MT_2	INT_EN_DRDY
3C	CTRL_REG5	Control reg5 R/W (Interrupt Configuration)	INT_CFG_ASLP	INT_CFG_FIFO	INT_CFG_TRANS	INT_CFG_LNDPRT	INT_CFG_PULSE	INT_CFG_FF_MT_1	INT_CFG_FF_MT_2	INT_CFG_DRDY

6.1 Example Procedure for Configuring the Auto-Wake/Sleep Function of the MMA8450Q

- Sleep Time-out period = 20 seconds
- Wake triggers = Tap and Motion
- Wake Sample Rate = 400 Hz, Sleep Sample Rate = 12.5 Hz
- 8g Mode

Step 1: Put the device in Standby Mode

Register 0x38 CTRL_REG1

- CTRL_REG1Data = IIC_RegRead(0x38);
- CTRL_REG1Data& = 0xDC;
- IIC_RegWrite(0x38,CTRL_REG1Data);

Step 2: To enable the Auto-Wake/Sleep set bit 1 in Register 39, the **SLPE** bit.

Register 0x39 CTRL_REG2

- CTRLReg2Data = IIC_RegRead(0x39); // Store value in the Register
- CTRLReg2Data| = 0x02; // Set the Sleep Enable bit
- IIC_RegWrite(0x39, CTRLReg2Data); Write the updated value into CTRL Register 2.

Step 3: The sleep sample rate must be chosen by writing in the corresponding sample rate value to bits 6 and 7 ASLP_RATE0 and ASLP_RATE1 in Register 0x38.

Register 0x38 CTRL_REG1

- CTRL_REG1Data = IIC_RegRead(0x38);
- CTRL_REG1Data& = 0x1C; //50Hz
- CTRL_REG1Data& = 0x5C; //20Hz
- CTRL_REG1Data& = 0x9C; //12.5Hz
- CTRL_REG1Data& = 0xDC; //50Hz
- IIC_RegWrite(0x38,CTRL_REG1Data);

Step 4: The Interrupt for the event to trigger the device to wake-up must be enabled by writing to Register 0x3B, CTRL_Reg4. Bits 1 through 7 will affect the auto-wake sleep. The data ready interrupt doesn't trigger the auto-wake/sleep mechanism.

Example: Set Pulse and Orientation and Motion 1 and Auto-Wake/Sleep Interrupts Enabled in the System

- IIC_RegWrite(0x3B, 0x9C);

Step 5: Route the interrupt chosen and enabled to either INT1 or INT2 in Register 0x3C CTRL_REG5.

Example: Route Pulse, Motion1 and Orientation to Int2 and Auto-Sleep to Int1.

- IIC_RegWrite(0x3C,0x80);

Step 6: Enable the interrupts that will wake the device from sleep. There can be more interrupts enabled in Step 4 than in Step 6. Only interrupts that are Enabled in Step 4 and that have the "Wake From Sleep" bit set in Register 0x3A will actually wake the device.

Example: Choose Pulse and Motion1 to wake the device from sleep

- IIC_RegWrite(0x3A,0x18);

Step 7: Set the Wake Mode Sample Rate and set the full-scale active mode in Register 0x38.

Register 0x38 CTRL_REG1

- CTRL_REG1Data = IIC_RegRead(0x38);
- CTRL_REG1Data& = 0xC0; //Make sure to clear the wake sample rate and set the ODR to 400 Hz
- CTRL_REG1Data| = 0x03;//Set 8g Active Mode
- IIC_RegWrite(0x38,CTRL_REG1Data);

Step 8: Write an Interrupt Service routine to monitor the Auto-Sleep Interrupt

```
Interrupt void isr_KBI (void)
{
      //clear the interrupt flag
      CLEAR_KBI_INTERRUPT;
      //Determine the source of interrupt by reading the system interrupt register
      Int_SourceSystem = IIC_RegRead(0x15);
      // Set up Case statement here to service all of the possible interrupts
      if ((Int_SourceSystem &=0x80)==0x80)
      {
            //Perform an Action since Auto-Sleep Flag has been set
            //Read the System Mode to clear the system interrupt
            Int_SysMod = IIC_RegRead(0x14);
            if (Int_SysMod==0x02)
            {// sleep mode
            }
            else if (Int_SysMod==0x01)
            {//Wake Mode
            }
            else
            {//Error
            }
      }
}
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

AN3921
Rev. 1
04/2010