Freescale Semiconductor
Application Note

AN4073
Rev 0, 09/2010

Using the 32 Sample First In First Out (FIFO) in the MMA8451Q

by: **Kimberly Tuck**
Applications Engineer

1.0 Introduction

The MMA8451Q has a built-in 32 sample first in, first out buffer capable of storing 14-bit data or 8-bit data. The data can be high pass filtered or not depending on whether the HPF_OUT bit is set in the part. 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 MMA8451Q to monitor a desired interrupt, putting the processor in a low power mode until it needs to respond to the accelerometer. This minimizes the system's overall power consumption, increasing the life of the battery. The embedded FIFO is a proven benefit as it limits how often the processor needs to read the data. The FIFO allows the processor to sleep longer while samples are being collected inside the sensor.

Higher sample rate data can be captured in the FIFO and accessed at a reasonable update time without increasing computational throughput by accessing every sample individually.

1.1 Key Words

Accelerometer, Output Data Rate (ODR), 32 Sample FIFO, 12-bit Data, 8-bit Data, I^{2}C Bus, Flushing, Algorithm Development, Circular Buffer Mode, Fill Buffer Mode, Trigger Buffer Mode, Watermark, Overflow, Sensor, Power Savings, Multi-read, Processor, MCU

TABLE OF CONTENTS

1.2 Summary

A. The embedded FIFO is highly beneficial for system power savings and minimizing traffic across the I^2C bus.

B. The FIFO allows for remarkable power savings of the system by allowing the host processor/MCU to go into a sleep mode while the accelerometer independently stores the data, up to 32 samples.

C. The FIFO can be configured to be in a circular buffer mode, trigger mode or a fill buffer mode, which is application dependent.

D. The FIFO is very useful for further enhancing embedded algorithms to extract more details from what caused the interrupt, with the ability to read previous history.

E. The FIFO relieves the host processor from continuously polling data and relieves bus congestion.

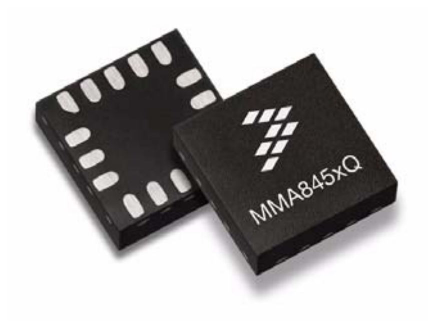
F. The FIFO in the MMA8451Q is a bit different than the MMA8450Q and allows for data to be written into the device and read out at the same time. This allows for even further current consumption savings.

G. The FIFO can be used to store 14-bit or 8-bit data. The data can be high pass filtered or not.

2.0 MMA8451, 2, 3Q Consumer 3-axis Accelerometer 3 x 3 x 1 mm

The MMA8451, 2, 3Q has a selectable dynamic range of ±2g, ±4g, ±8g. The device has 8 different output data rates, selectable high pass filter cut-off frequencies, and high pass filtered data. The available resolution of the data and the embedded features is dependant on the specific device.

Note: The MMA8450Q has a different memory map and has a slightly different pin-out configuration.



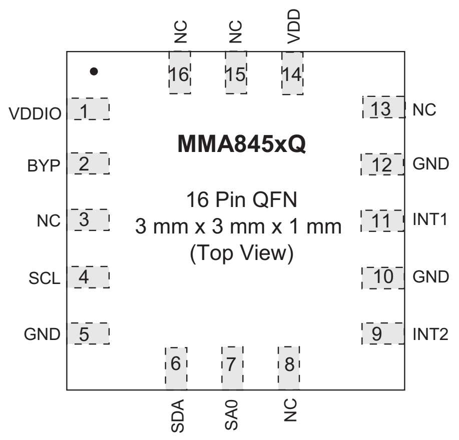


Figure 1. MMA8451, 2, 3Q Consumer 3-axis Accelerometer 3 x 3 x 1 mm

2.1 Output Data, Sample Rates and Dynamic Ranges of all Three Products

2.1.1 MMA8451Q

1. **14-bit data**
   2g (4096 counts/g = 0.25 mg/LSB) **4g** (2048 counts/g = 0.5 mg/LSB) **8g** (1024 counts/g = 1 mg/LSB)
2. **8-bit data**
   2g (64 counts/g = 15.6 mg/LSB) **4g** (32 counts/g = 31.25 mg/LSB) **8g** (16 counts/g = 62.5 mg/LSB)
3. **Embedded 32 sample FIFO (MMA8451Q)**

2.1.2 MMA8452Q

1. **12-bit data**
   2g (1024 counts/g = 1 mg/LSB) **4g** (512 counts/g = 2 mg/LSB) **8g** (256 counts/g = 3.9 mg/LSB)
2. **8-bit data**
   2g (64 counts/g = 15.6 mg/LSB) **4g** (32 counts/g = 31.25 mg/LSB) **8g** (16 counts/g = 62.5 mg/LSB)

2.1.3 MMA8453Q Note: No HPF Data

1. **10-bit data**
   2g (256 counts/g = 3.9 mg/LSB) **4g** (128 counts/g = 7.8 mg/LSB) **8g** (64 counts/g = 15.6 mg/LSB)
2. **8-bit data**
   2g (64 counts/g = 15.6 mg/LSB) **4g** (32 counts/g = 31.25 mg/LSB) **8g** (16 counts/g = 62.5 mg/LSB)

2.2 Application Notes for the MMA8451, 2, 3Q

The following is a list of all the application notes available for the MMA8451, 2, 3Q:

- **AN4068**, *Embedded Orientation Detection Using the MMA8451, 2, 3Q*
- **AN4069**, *Offset Calibration of the MMA8451, 2, 3Q*
- **AN4070**, *Motion and Freefall Detection Using the MMA8451, 2, 3Q*
- **AN4071**, *High Pass Filtered Data and Functions Using the MMA8451, 2,3Q*
- **AN4072**, *MMA8451, 2, 3Q Single/Double and Directional Tap Detection*
- **AN4073, *Using the 32 Sample First In First Out (FIFO) in the MMA8451Q***
- **AN4074**, *Auto-Wake/Sleep Using the MMA8451, 2, 3Q*
- **AN4075**, *How Many Bits are Enough? The Trade-off Between High Resolution and Low Power Using Oversampling Modes*
- **AN4076**, *Data Manipulation and Basic Settings of the MMA8451, 2, 3Q*

3.0 Applications Using the FIFO

The FIFO is used typically for the following functions:

- Data logging- flushing once every 32 samples for power savings and to relieve bus contention
- Storing the previous history leading up to an event, for power savings and relives bus contention
- The FIFO can be programmed to stop on an event trigger if the device was in sleep mode and the sample rate changes when the event occurs
- The FIFO can be programmed to stop on an event trigger for tap, Portrait/Landscape, transient or motion. The watermark can be set to store a certain amount of the data leading up to the event and a certain amount after the event.
- The FIFO can be used to store High Pass Filtered data for gesture analysis involving data transitions. This simplifies the processing required on the host.

3.1 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 MMA8451Q 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^{2}C 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 14-bit data or the 8-bit data. For the 14-bit data each sample consists of three 14-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^{2}C 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^{2}C transactions take a minimum of 0.6 µs. A 1 µs value will be added for each of these. Assuming that each I^{2}C 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^{2}C burst read of N samples can be calculated as follows:

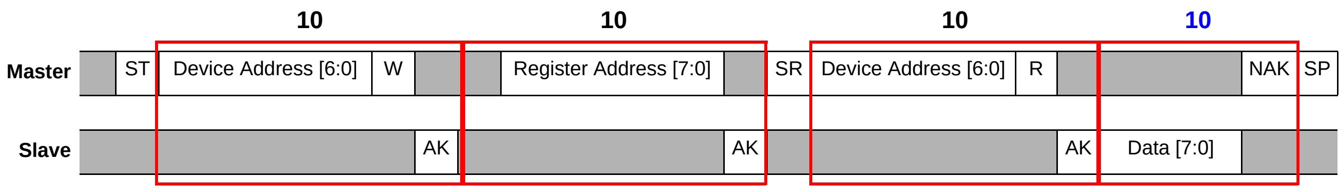


Figure 2. I^{2}C Single Byte Read Transaction

14-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 14-bit XYZ data consecutively takes 1950 bits to perform the transaction. By bursting XYZ 14-bit data each time new data is ready requires 2880 bits, as this requires 32 iterations. The start, stop and repeat start trans- actions 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. Note that the timing to read the FIFO can be minimized further by increasing the bus speed if this is possible by the processor.

Note: The MMA8451Q is capable of I^2C bus speeds up to 4.75 MHz.

Example conditions are given for a processor with the wake timing and current consumption values in Table 1. In Wake Mode the example processor uses a total of 12 mA, while in sleep mode it only uses 0.5 mA. It takes 3 ms to wake the processor from sleep and read the FIFO status. As shown above, a 14-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 1.

Table 1. Example Conditions Processor

Wake-Up Time	I^2C Bit Rate	14-bit Flush	8-bit Flush	Sleep Mode	Wake Mode
3 ms	400 kHz	5 ms	2.5 ms	0.5 mA	12 mA

3.1.1 Flushing the FIFO at 100 Hz ODR and Below

At all data rates 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. 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. At 100 Hz and below this occurs before the next sample is ready. The details of the sleep to wake timing are captured in Figure 3 and Table 2.

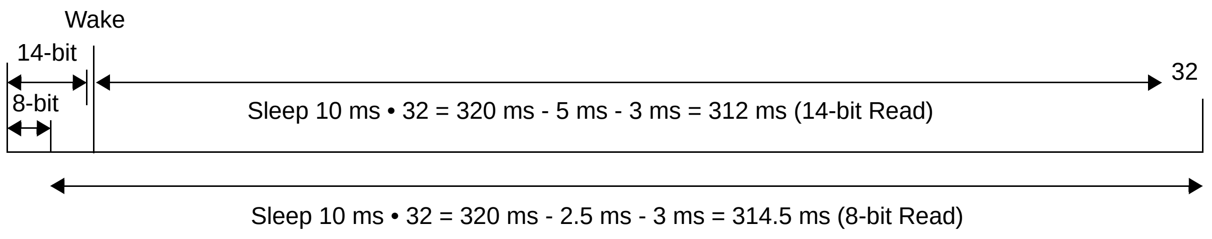

Wake
14-bit
8-bit
32
Sleep 10 ms • 32 = 320 ms - 5 ms - 3 ms = 312 ms (14-bit Read)
Sleep 10 ms • 32 = 320 ms - 2.5 ms - 3 ms = 314.5 ms (8-bit Read)

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

Table 2. Wake and Sleep Timing at 100 Hz ODR

Data	ODR	LP Current	Total Time	Sleep Time	Wake Time	Sleep/Total	Wake/Total	Sleep Current	Wake Current	Total Current	Current Savings
14-bit	100 Hz	0.024 mA	320 ms	312 ms	8 ms	312/320	8/320	0.524 mA	12.024 mA	0.812 mA	93.3%
8-bit	100 Hz	0.024 mA	320 ms	314.5 ms	5.5 ms	314.5/320	5.5/320	0.524 mA	12.024 mA	0.722 mA	94.0%

3.1.2 Flushing the FIFO at 200 Hz ODR

When the data rate is set to 200 Hz the time between samples is 5 ms. The processor can be put in sleep mode until the overflow flag is asserted. Then the data is flushed. This is done without missing any samples. The results of the Sleep to Wake timing and current drain are captured in Table 3.

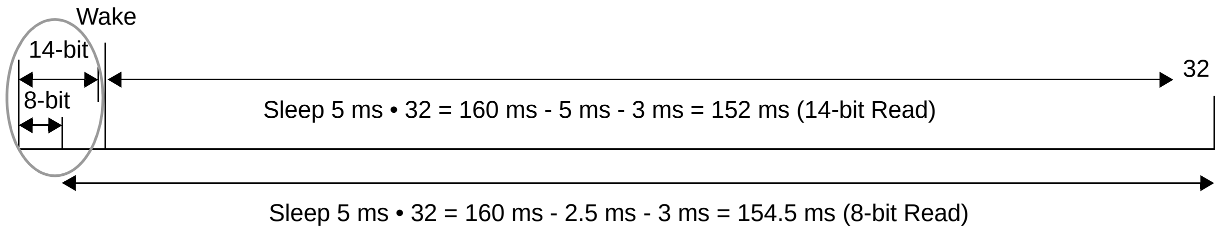

Wake
14-bit
8-bit
32
Sleep 5 ms • 32 = 160 ms - 5 ms - 3 ms = 152 ms (14-bit Read)
Sleep 5 ms • 32 = 160 ms - 2.5 ms - 3 ms = 154.5 ms (8-bit Read)

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

Table 3. Wake and Sleep Timing at 200 Hz ODR

Data	ODR	LP Current	Total Time	Sleep Time	Wake Time	Sleep/Total	Wake/Total	Sleep Current	Wake Current	Total Current	Current Savings
14-bit	200 Hz	0.044 mA	160 ms	152 ms	8 ms	152/160	8/160	0.544 mA	12.044 mA	1.119 mA	90.7%
8-bit	200 Hz	0.044 mA	160 ms	154.5 ms	5.5 ms	154.5/160	5.5/160	0.544 mA	12.044 mA	0.939 mA	92.2%

3.1.3 Flushing the FIFO at 400 Hz ODR

When sampling at 400 Hz, there is a new sample every 2.5 ms. Because the FIFO can read and write at the same time, there is no conflict with missing samples to read out the data fast enough. The timing diagram illustrating the Wake and Sleep intervals are shown in Figure 5.

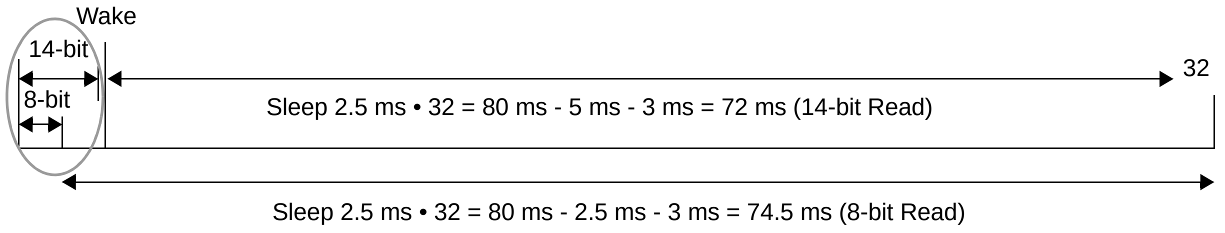

Wake
14-bit
8-bit
32
Sleep 2.5 ms • 32 = 80 ms - 5 ms - 3 ms = 72 ms (14-bit Read)
Sleep 2.5 ms • 32 = 80 ms - 2.5 ms - 3 ms = 74.5 ms (8-bit Read)

Figure 5. Timing of the FIFO at 400 Hz ODR Showing Sleep and Wake Timing

Table 4 presents all the calculations at 400 Hz flushing 14-bit data and 8-bit data without missing samples.

Table 4. Wake and Sleep Timing at 400 Hz ODR

Data	ODR	LP Current	Total Time	Sleep Time	Wake Time	Sleep/Total	Wake/Total	Sleep Current	Wake Current	Total Current	Current Savings
14-bit	400 Hz	0.085 mA	80 ms	72 ms	8 ms	72/80	8/80	0.585 mA	12.085 mA	1.735 mA	85.6%
8-bit	400 Hz	0.085 mA	80 ms	74.5 ms	5.5 ms	74.5/80	5.5/80	0.585 mA	12.085 mA	1.376 mA	88.6%

3.1.4 Flushing the FIFO at 800 Hz ODR

When sampling at very high speeds the value of the FIFO is very apparent. The processor no longer needs to access the data every 1.25 ms at 800 Hz mode to collect all the data. The processor can go to sleep and wait until the overflow flag asserts, flushing the data every 4 ms. Because the FIFO can read and write at the same time there is no conflict with missing samples to read out the data fast enough. The timing diagram illustrating the Wake and Sleep intervals are shown in Figure 6.

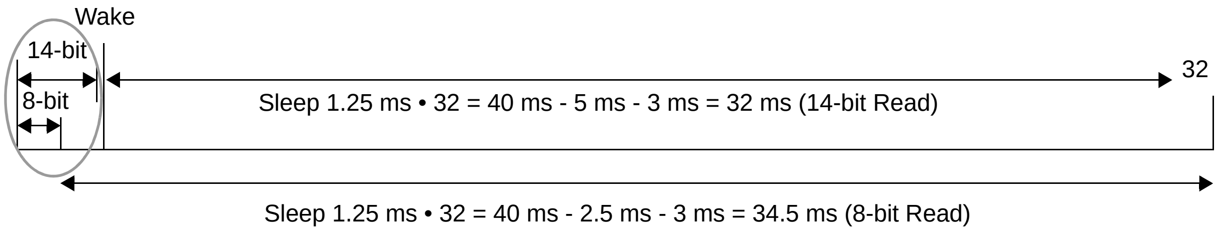

Wake
14-bit
8-bit
32
Sleep 1.25 ms • 32 = 40 ms - 5 ms - 3 ms = 32 ms (14-bit Read)
Sleep 1.25 ms • 32 = 40 ms - 2.5 ms - 3 ms = 34.5 ms (8-bit Read)

Figure 6. Timing of the FIFO at 800 Hz ODR Showing Sleep and Wake Timing

Table 5. Wake and Sleep Timing at 800 Hz ODR

Data	ODR	LP Current	Total Time	Sleep Time	Wake Time	Sleep/Total	Wake/Total	Sleep Current	Wake Current	Total Current	Current Savings
14-bit	800 Hz	0.165 mA	40 ms	32 ms	8 ms	32/40	8/40	0.665 mA	12.165 mA	2.965 mA	75.6%
8-bit	800 Hz	0.165 mA	40 ms	34.5 ms	5.5 ms	34.5/40	5.5/40	0.665 mA	12.165 mA	2.246 mA	81.5%

Table 6 summarizes the Wake and Sleep timing for all sample rates of the MMA8451Q. 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 All Data Rates

ODR	Time Between Samples	Sleep/Total Ratio 14-bit	Total Current mA	Current Savings 14-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%	0.510 mA	95.75%	99.98%	0.509 mA	95.76%
6.25 Hz	160 ms	99.84%	0.524 mA	95.64%	99.89%	0.518 mA	95.68%
12.5 Hz	80 ms	99.69%	0.542 mA	95.49%	99.79%	0.531 mA	95.58%
50 Hz	20 ms	98.75%	0.658 mA	94.53%	99.14%	0.613 mA	94.90%
100 Hz	10 ms	97.50%	0.812 mA	93.25%	98.28%	0.722 mA	94.00%
200 Hz	5 ms	95.00%	1.119 mA	90.71%	96.56%	0.939 mA	92.20%
400 Hz	2.5 ms	90.00%	1.736 mA	85.64%	93.13%	1.376 mA	88.62%
800 Hz	1.25 ms	80.00%	2.965 mA	75.63%	86.25%	2.246 mA	81.54%

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. 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. The current consumption values for the accelerometer are taken in the Low Power mode, which gives the best case scenario. This analysis can be done for the other 3 oversampling modes (High Resolution, Low Noise + Low Power, and Normal)

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

ODR LP-Low Power	Total Consumption	AA Li-Ion Battery Life 1200 mAh (1200 mAh/Total mA) Time (h)	Time (Days)
1.56 Hz (LP)	12.006	99.95	4.16
6.25 Hz (LP)	12.006	99.95	4.16
12.5 Hz (LP)	12.006	99.95	4.16
50 Hz (LP)	12.014	99.88	4.16
100 Hz (LP)	12.024	99.80	4.16
200 Hz (LP)	12.044	99.63	4.15
400 Hz (LP)	12.085	98.30	4.14
800 Hz (LP)	12.165	98.64	4.11

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 store data and put the processor in a sleep mode can have a significant impact on the battery life (Table 8 and Table 9). In Table 8 and Table 9 the far column on the right displays the battery life improvement by using the accelerometer FIFO to data log the data, putting the processor into a sleep mode until the FIFO is full. This is compared to having the processor and accelerometer running all the time, which is the method required to data log without a FIFO. This shows that at the highest sampling rate in Low Power Mode the battery life improves 4 - 5x what it would by polling the data with the processor continually running. At the lowest sample rate in Low Power Mode, the savings is 23.6x longer battery life.

Table 8. Example Li-Ion Battery Life Calculations Using the FIFO to Data Log 14-bit Data

ODR LP-Low Power	Total Consumption 14-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)	0.510 mA	2350.71	97.95	23.52
6.25 Hz (LP)	0.524 mA	2290.21	95.43	22.91
12.5 Hz (LP)	0.542 mA	2214.28	92.26	22.15
50 Hz (LP)	0.658 mA	1824.4	76.02	18.27
100 Hz (LP)	0.812 mA	1478.74	61.61	14.82
200 Hz (LP)	1.119 mA	1072.39	44.68	10.76
400 Hz (LP)	1.735 mA	691.64	28.82	6.93
800 Hz (LP)	2.965 mA	404.72	16.86	4.10

Table 9. Example Li-Ion Battery Life Calculations Using the FIFO to Data Log 8-bit Data

ODR LP-Low Power	Total Consumption 8-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)	0.509 mA	2357.18	98.22	23.58
6.25 Hz (LP)	0.518 mA	2315.02	96.46	23.16
12.5 Hz (LP)	0.531 mA	2261.13	94.21	22.62
50 Hz (LP)	0.613 mA	1958.13	81.59	19.60
100 Hz (LP)	0.722 mA	1662.84	69.29	16.66
200 Hz (LP)	0.939 mA	1277.53	53.23	12.82
400 Hz (LP)	1.376 mA	872.33	36.35	8.79
800 Hz (LP)	2.246 mA	534.22	22.26	5.42

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

One of the application uses 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 Trigger Mode) can be flushed to extract the data leading up to the event. The event trigger watermark can be set to store any number of samples before the event occurred and then the remaining samples after the event occurred, as long as the total number of samples is equal to 32. For example, the FIFO can be set to store the first 20 samples leading up to the event by setting the watermark to 20. Then after the event the FIFO will fill to 32 collecting the next 12 samples after the event and then will stop and assert to overflow flag. This technique can be used for a variety of gestures. For example, directional tap can be analyzed by configuring the single tap event and setting the FIFO to Trigger Buffer Mode enabling the Tap Trigger. The tap event must be configured to all the desired threshold settings. For this application the data rate should be set to 800 Hz. The processor can go into a low power or sleep mode until the FIFO overflow flag is asserted. Then the data can be flushed to analyze the data leading up to the event and right after the event. The ability to extract the data of interest efficiently while the processor is in sleep mode has huge advantages. Even if the processor is not in a sleep mode this saves the processor from having to continually analyze irrelevant data. The accelerometer is smart enough to be configured to store the data of interest and let the processor know when that data is available.

3.3 FIFO Behavior During Wake to Sleep Transitions

Table 10 describes the different behaviors of the FIFO under the wake/sleep conditions.

Table 10. Behavior of FIFO under Wake/ Sleep Conditions

FIFO INT Enabled	Wake From Sleep Enabled	Result
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 the Wake ODR.
YES	NO	With the interrupt enabled the FIFO can be read and flushed clearing the interrupt. The system is kept from falling asleep by reading the status after the interrupt is set. The FIFO does not have to be flushed to keep the device in wake mode as long as the FIFO status is read continuously after the FIFO interrupt is enabled. If the system falls asleep (and no new interrupts occur during the time-out period), the FIFO AUTOMATICALLY flushes and starts refilling at the Sleep ODR 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 here 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) . Note: Reading the FIFO status will keep the system 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.

When the FIFO is configured and the Auto-Wake/Sleep is configured with the FIFO Wake from Sleep bit set (Reg 0x2C bit 7) the data in the FIFO is held until the FIFO is flushed. If a new sample arrives before the FIFO is flushed a FIFO Gate Error occurs indicating at least one sample has been missed. The FIFO Gate Error bit is set by the system in register 0x0B, bit 7. This bit will clear when the FIFO is flushed. This can be useful to see the data up to the point before the device changes ODR. The FG_Time bits (in 0x0B) indicate the number of samples or time period that has elapsed since the FIFO Gate Error was asserted. A value of up to 31 can be stored.

Table 11. 0x0B SYSMOD: System Mode Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FGERR	FGT_4	FGT_3	FGT_2	FGT_1	FGT_0	SYSMOD1	SYSMOD0

4.0 Embedded Settings of the FIFO

The following section discusses the different registers involved in configuring the FIFO.

1. Register 0x09 bit 7, 6 – Set **F_Mode to 01, 10 or 11** and set Watermark Value
2. Register 0x0A Interrupt Triggers for FIFO trigger mode
3. Register 0x01 Points to the FIFO buffer and is used to access 8-bit or 14-bit data.
4. Register 0x0E XYZ_DATA_CFG Register for enabling high pass filtered data and dynamic range
5. Register 0x00 FIFO Status Register
6. Register 0x2D Interrupt Enable Register bit 6
7. Register 0x2E Interrupt Configuration Register bit 6, Route INT1 or INT2

4.1 Register 0x09: F_SETUP FIFO Set-Up Register

The **F_SETUP** register configures the FIFO mode and the watermark value. When F_MODE is a value greater than zero (00) the FIFO is enabled. When F_MODE is 00 the FIFO is disabled.

Table 12. 0x09 F_SETUP: FIFO Setup Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F_MODE1	F_MODE0	F_WMRK5	F_WMRK4	F_WMRK3	F_WMRK2	F_WMRK1	F_WMRK0

FMODE = 01 Circular Buffer: In this mode the FIFO contains the most recent samples when overflowed. The oldest sample is discarded to be replaced by the new sample. This mode is useful when waiting for an event and wanting to collect the latest 32 samples up to that event.

FMODE = 10 Fill Buffer: In this mode the FIFO stops accepting new samples when overflowed. This is a good mode to be in for data logging.

FMODE = 11 Trigger Buffer: In this mode the FIFO will be in a circular mode up to the number of samples set in the watermark. After the trigger occurs (tap, P/L, motion, transient) the FIFO will fill and collect samples until the buffer if full. This allows the data to be collected both before and after the trigger event, defined by the watermark. Note that the FIFO mode can be switched between the three operational modes (01, 10, 11) in Active Mode. The mode must first be changed to 00 then it can be modified.

To change Modes while in Active try the following sequence:

A. Set the Watermark 1 to 32 counts
B. Set **F_Mode** bit 6 and 7 to “Fill” 01
C. Set Active Mode
D. Set **F_Mode** bit 6 and 7 to “Disable” 00
E. Set **F_Mode** bit 6 and 7 to “Circular” 10

The watermark is used to trigger a watermark interrupt. If the FIFO watermark remains at 0 this will disable the FIFO watermark event flag generation. A FIFO sample count exceeding the watermark event does not stop the FIFO from accepting new data. The FIFO update rate is dictated by the selected system ODR. In active mode the ODR is set by the **DR** bits in the **CTRL_REG1** register (0x2A) and when auto-sleep is active the ODR is set by the **ASLP_RATE** field in the **CTRL_REG1** register (0x2A).

When a byte is read from the FIFO buffer the oldest sample data in the FIFO buffer is returned and also deleted from the front of the FIFO buffer, while the FIFO sample count is decremented by one. It is assumed that the host application shall use the I^2C multi-read transaction to dump the FIFO.

4.2 Register 0x0A Interrupt Triggers for Trigger Mode

The Trigger Configuration register is used with the Trigger Buffer Mode (FMODE = 11). The event to trigger the FIFO to switch from circular to fill and stop is selected in register 0x0A. The function settings for the selected interrupt must be configured but the system function interrupt does not need to be enabled.

Table 13. 0x0A: TRIG_CFG Trigger Configuration Register (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
—	—	Trig_TRANS	Trig_LNDPRT	Trig_PULSE	Trig_FF_MT	—	—

4.3 Accessing the FIFO Buffer Data

If F_Mode > 00 in Reg 0x09 then the data points to the head of the FIFO buffer at Register 0x01 which is the X_MSB byte. This is the start point for both the 14-bit or the 8-bit data. When accessing the 8-bit data the F_READ bit in Register 0x2A is set which modifies the auto-incrementing to skip over the LSB data. When F_READ is cleared the 14-bit data is read accessing all 6 bytes sequentially per sample. Registers 0x02, 0x03, 0x04, 0x05 and 0x06 will return a value of zero when read.
Note: F_MODE must be equal to 00 when changing the F_READ bit to change between 8-bit or 14-bit data. Also the part must be in Standby mode.

4.4 Setting HPF Data through the FIFO

In order to store HPF data in the FIFO the HPF_OUT bit must be set in Register 0x0E. When the HPF_OUT bit is not set the data will not be high pass filtered. The dynamic range is also set in this register with the FS0 and FS1 bits.

Table 14. Register 0x0E: XYZ_DATA_CFG (Read/Write)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	HPF_OUT	0	0	FS1	FS0

4.5 Register 0x00: F_STATUS FIFO Status Register

The Status Register shown in Table 15, is shared between the real time status when F_MODE = 00, and the FIFO Status when F_MODE > 00. The status is used to retrieve information about the FIFO. This register has a flag for the overflow and watermark. It also has a counter that can be checked to review the number of samples stored in the buffer.

Table 15. Register 0x00_STATUS: FIFO STATUS Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
F_OVF	F_WMRK_FLAG	F_CNT5	F_CNT4	F_CNT3	F_CNT2	F_CNT1	F_CNT0

The **F_OVF** and **F_WMRK_FLAG** flags remain asserted while the event source is still active, but the user can clear the FIFO interrupt bit flag in the interrupt source register **(INT_SOURCE Reg 0x0C**) by reading the **F_STATUS** register (0x00).

The **F_OVF** bit flag will assert when the FIFO has overflowed and the **F_WMRK_FLAG** bit flag will assert when the **F_CNT** value is greater than then **F_WMRK** value. These interrupts remain asserted until at least one sample (X,Y,Z) is read. Both the Watermark flag and the Overflow flag cause a System Interrupt for the FIFO in Register 0x0C when the FIFO interrupt is enabled.

4.6 Configuring the FIFO to an Interrupt Pin

In order to set up the system to route to a hardware interrupt pin, the System Interrupt (bit 6 in Reg 0x2D) must be enabled. The MMA8451Q allows for seven (7) separate types of interrupts. One (1) of these is reserved for the FIFO.

Step 1: Enable the Interrupt in Register 0x2D.

Table 16. 0x2D CTRL_REG4 Register (Read/Write)

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	—	INT_EN_DRDY

The **INT_EN_FIFO** interrupt enable bit allows the FIFO function to route its event detection flag to the interrupt controller of the system. The interrupt controller routes the enabled function to either the INT1 or INT2 pin.

To enable the FIFO, set bit 6 in Register 0x2D as follows:

Code Example: IIC_RegWrite (0x2D, 0x40);

Step 2: Route the interrupt to INT1 or to INT2. This is done in register 0x2E.

Table 17. 0x2E CTRL_REG5 Register (Read/Write)

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	—	INT_CFG_DRDY

Note: To route the FIFO to INT1 set bit 6 in register 0x2E. Clear bit 6 to set the FIFO to INT2.

Code Example: IIC_RegWrite (0x2E,0x40); //Set to INT1

4.7 Reading the System Interrupt Status Source Register

In the interrupt source register, the status of the various embedded features can be determined. This is shown in Table 18. The bits that are set (logic ‘1’) indicate which function has asserted an interrupt; conversely, the bits that are cleared (logic ‘0’) indicate which function has not asserted or has de-asserted an interrupt. The interrupts are rising edge sensitive. The bits are set by a low to high transition and are cleared by reading the appropriate interrupt source register.

Table 18. 0x0C INT_SOURCE: System Interrupt Status Register (Read Only)

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SRC_ASLP	SRC_FIFO	SRC_TRANS	SRC_LNDPRT	SRC_PULSE	SRC_FF_MT	—	SRC_DRDY

5.0 Example Code Using the FIFO

The following are three examples for configuring the FIFO. Table 19 shows all the registers of importance for using the FIFO.

Table 19. Registers of Importance for the FIFO

Reg	Name	Definition	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
00	F_STATUS	FIFO Status R	F_OVF	F_WMRK_FLAG	F_CNT5	F_CNT4	F_CNT3	F_CNT2	F_CNT1	F_CNT0
09	F_SETUP	FIFO Setup R/W	F_MODE1	F_MODE0	F_WMRK5	F_WMRK4	F_WMRK3	F_WMRK2	F_WMRK1	F_WMRK0
0A	TRIG_CFG	FIFO Triggers R/W	—	—	Trig_TRANS	Trig_LNDPRT	Trig_PULSE	Trig_FFMT	—	—
0B	SYSMOD	System Mode R	FGERR	FGT_4	FGT_3	FGT_2	FGT_1	FGT_0	SYSMOD1	SYSMOD0
0C	INT_SOURCE	Interrupt Status R	SRC_ASLP	SRC_FIFO	SRC_TRANS	SRC_LNDPRT	SRC_PULSE	SRC_FF_MT	—	SRC_DRDY
2A	CTRL_REG1	Control Reg1 R/W	ASLP_RATE1	ASLP_RATE0	DR2	DR1	DR0	LNOISE	F_READ	ACTIVE
2C	CTRL_REG3	Control Reg3 R/W (Wake Interrupts from Sleep)	FIFO_GATE	WAKE_TRANS	WAKE_LNDPRT	WAKE_PULSE	WAKE_FF_MT	—	IPOL	P_OD
2D	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	—	NT_EN_DRDY
2E	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	—	INT_CFG_DRDY

5.1 Power Minimization Example: Data Logger Collecting 14-bit HPF Data 100 Hz, 4g

Step 1: Go to Standby Mode, set Sample Rate to 100 Hz, Standby (Active = 0),14-bit (F_READ = 0),

IIC_RegWrite(0x2A, 0x18); //CRTL_REG1: 100 Hz mode DR = 011, Active = 0, F_Read = 0

Step 2: Set the FIFO to Fill Buffer Mode in Register 0x09 F_Setup

IIC_RegWrite(0x09, 0x80); //FIFO Set to Fill Mode

Step 3: Set the FIFO Interrupt Pin to Int1 to flush the data every 32 samples when the overflow flag asserts

IIC_RegWrite(0x2D, 0x40); //Enable the interrupt Pin for the FIFO

IIC_RegWrite(0x2E, 0x40); //Set the interrupt to route to INT1

Step 4: Turn on HPF for Data Out and set 4g Mode

IIC_RegWrite(0x0E, 0x11); //HPF_OUT set, 4g mode

Step 5: Go to Active Mode

CRTL_REG1_Data = IIC_RegRead(0x2A);

CRTL_REG1_Data| = 0x01;

IIC_RegWrite(CRTL_REG1_Data);

Step 6: Write an Interrupt Service Routine to Clear the interrupt and Flush the data from the FIFO.

```
Interrupt void isr_KBI (void)
{
      WAKE_MCU;
      //clear the interrupt flag
      CLEAR_KBI_INTERRUPT;

      //Determine the source of the interrupt by reading the system interrupt
      Int_SourceSystem = IIC_RegRead(0x0C);
      // Set up Case statement here to service all of the possible interrupts
      if ((Int_SourceSystem&0x40)==0x40)
      {
            // 1. Read the Status Register to Clear the Overflow Flag Interrupt

            IIC_RegRead(0x00);

            // 2. This would be used to burst out 14 bit data
            ComObj.ReadDataBurst(CurDeviceAddress, 0X01, ref databuf14, 192);
            //This would be used to burst out 8 bit data
            ComObj.ReadDataBurst(CurDeviceAddress, 0X01, ref databuf8, 96);
      }
}
```

5.2 Event Detection Trigger on a Tap Event to Flush the Data for Further Analysis 800 Hz ODR, 8g Mode, 8-bit data

Step 1: Go to Standby Mode, set Sample Rate to 800 Hz, Standby (Active = 0),8-bit (F_READ = 1),

IIC_RegWrite(0x2A, 0x02); //CRTL_REG1: 800 Hz mode DR = 000, Active = 0, F_Read = 1

Step 2: Set the FIFO to Trigger Buffer Mode in Register 0x09 F_Setup

IIC_RegWrite(0x09, 0xD4); //FIFO Set to Trigger, Set Watermark to 20

Step 3: Set the Trigger for Tap Detection

IIC_RegWrite(0x0A,0x08); //Enable the Tap to Trigger the FIFO

Step 4: Enable X and Y and Z Single Pulse

IIC_RegWrite(0x21, 0x15)

Step 5: Set Threshold 1.575g on X and 2.52g on Z

Note: Every step is 0.063g

1.575g/ 0.063g = 25 counts

2.52g/0.063g = 40 counts

IIC_RegWrite(0x23, 0x19); //Set X Threshold to 1.575g

IIC_RegWrite(0x24, 0x19); //Set Y Threshold to 1.575g

IIC_RegWrite(0x25, 0x28); //Set Z Threshold to 2.52g

Step 6: Set Time Limit for Tap Detection to 50 ms No LPF Enabled

Note: 800 Hz ODR, Time step is 0.625 ms per step

50 ms/0.625 ms = 80 counts

IIC_RegWrite(0x26,0x50); //50 ms

Step 7: Set Latency Timer to 300 ms

Note: 800 Hz ODR, Time step is 1.25 ms per step

300 ms/1.25 ms = 240 counts

IIC_RegWrite(0x27,0xF0); //300 ms

Step 8: Set the Tap Interrupt Pin to INT1 to Alert to MCU to wake up and flush the FIFO

IIC_RegWrite(0x2D, 0x08); //Enable the interrupt Pin Tap

IIC_RegWrite(0x2E, 0x08); //Set the interrupt to route to INT1

Step 9: Put Device in 8g Mode, Storing HPF Data

IIC_RegWrite(0x0E, 0x12); //8g HPF Data

Step 10: Put Device in Active Mode

CRTL_REG1_Data = IIC_RegRead(0x2A);

CRTL_REG1_Data| = 0x01;

IIC_RegWrite(CRTL_REG1_Data);

Step 11: Write an Interrupt Service Routine to Clear the interrupt and Flush the data from the FIFO The interrupt service routine will wake the MCU and then flush the data. The data will contain 20 samples of data leading up to the tap event and 12 samples after the event. Sometimes the rebound acceleration data can be very insightful to understanding an event. Therefore it is a good idea to capture the data right before the event, during the event and maybe a few samples after the event. All of this can be accomplished with the FIFO.

```
Interrupt void isr_KBI (void)
{
      WAKE_MCU;
      //clear the interrupt flag
      CLEAR_KBI_INTERRUPT;
      //Determine the source of the interrupt by first reading the system interrupt
      register
      Int_SourceSystem = IIC_RegRead(0x0C);
      // Set up Case statement here to service all of the possible interrupts
      if ((Int_SourceSystem&0x40)==0x40)
      {
            // 1. Read the Status Register to Clear the Tap Status Flag
            TapStatus = IIC_RegRead(0x22);
            // 2. 8 bit data only required. The FIFO interrupt is cleared when
            // flushing the data. Therefore no requirement to read the FIFO Status
            ComObj.ReadDataBurst(CurDeviceAddress, 0X01, ref databuf8, 96);
      }
}
```

5.3 Auto-Wake Sleep Trigger Using the FIFO to Hold the Data that Saved Before the ODR Changed: Circular Buffer Mode

Step 1: Put the device in Standby Mode, ASLEEP = 50 Hz, Wake DR = 800 Hz

IIC_RegWrite (0x2A, 0x02); //Sleep 50 Hz, Wake 800 Hz, F_READ = 1 8-bit data

Step 2: Enable Auto Sleep: To enable the Auto-Wake/Sleep set bit 2 in Register 0x2B, the SLPE bit.

Note: Register 0x2B CRTL_REG2

CRTL_REG2_Data = IIC_RegRead(0x2B); //Store value in the Register

CRTL_REG2_Data| = 0x04; //Set the Sleep Enable Bit

IIC_RegWrite(0x2B, CRTL_REG2_Data); Write the updated value in CRTL_REG2.

Step 3: Set the Sleep Timer for 20 seconds to time out

Note: Time step at 800 Hz ODR = 320 ms steps
20s/ 0.32s = 62.5, rounded to 63 counts

IIC_RegWrite(0x29, 0x3F);

Step 4: Set the FIFO Gate and the Motion Block 1 in the Wake From Sleep Register

WakeRegData = IIC_RegRead(0x2C); //Store Register contents
WakeRegData | = 0x88; Set the Wake from Motion and the FIFO Gate
IIC_RegWrite(0x2C, WakeRegData);

Step 5: Configure the Data to be Regular Data, not HPF data, 2g mode in Register 0x0E

IIC_RegWrite(0x0E, 0x00);

Step 6: Put the FIFO in Circular Buffer Mode

IIC_RegWrite(0x09, 0x40); //FIFO Set to Circular Buffer Mode

Step 7: Set up the Motion Freefall Block (INT2), and Auto Sleep (INT1)

IIC_RegWrite(0x2D, 0x84); //Enable Motion Block, Auto Sleep
IIC_RegWrite(0x2E, 0x80); //Set INT1 to AutoSleep, Set INT2 to Motion

Step 8: Enable X, Y, Z with “OR” Condition and latch enabled for Motion Detection

IIC_RegWrite(0x15, 0xF8);

Step 9: Set the Threshold to 2g

Note: Each count is 0.063g per count

2g/0.063g/count = 32 counts

IIC_RegWrite(0x17, 0x20);

Step 10: Set the debounce counter to eliminate false readings at 800 Hz, will need to adjust at Sleep ODR

IIC_RegWrite(0x18, 0x30); //60 ms at 800 Hz

Step 11: Put the device in Active Mode

CRTL_REG1_Data = IIC_RegRead(0x2A);

CRTL_REG1_Data| = 0x01;

IIC_RegWrite(CRTL_REG1_Data);

Step 12: Set up the Interrupt Service Routine

```
Interrupt void isr_KBI (void)
{
      WAKE_MCU;
      //clear the interrupt flag
      CLEAR_KBI_INTERRUPT;
      //Determine the source of the interrupt by first reading the system interrupt
      register
      Int_SourceSystem = IIC_RegRead(0x0C);
      // Set up Case statement here to service all of the possible interrupts
      if ((Int_SourceSystem&0x80)==0x80)
      {
            //Perform an Action since AutoSleep Flag has been set
            //Read the System Mode to clear the system interrupt
            Int_SysMod = IIC_RegRead(0x0B);
            if (Int_SysMod==0x02)
            {
                  // sleep mode
                  //Flush Data to MCU and store
                  ComObj.ReadDataBurst(CurDeviceAddress, 0X01, ref databuf8, 96);

            }
            else if (Int_SysMod==0x01)
            {
                  //Wake Mode
                  //Flush Data to MCU and store
                  ComObj.ReadDataBurst(CurDeviceAddress, 0X01, ref databuf8, 96);

            }
            else
            {
                  //Error
            }
      }
}
```

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. The Energy Efficiency Solutions Logo and Xtrinsic are trademarks of Freescale Semiconductor, Inc.
All other product or service names are the property of their respective owners.

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:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor

AN4073
Rev. 0
09/2010