AN-1025
APPLICATION NOTE

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

Utilization of the First In, First Out (FIFO) Buffer in Analog Devices, Inc. Digital Accelerometers

by Christopher J. Fisher, Tomoaki Tsuzuki, and James Lee

INTRODUCTION

With an increasing demand for user interaction, gesture recognition, and power savings in many of today's products, inertial sensors are quickly being adopted into a wide variety of applications. By sensing movement of a device, features can be turned off to save power during periods of inactivity. Being able to differentiate between the axis of a tap and its direction allows for new and unique ways of providing user input.

Analog Devices offers a wide range of accelerometers, including analog and digital, in 1-, 2-, and 3-axis variants. The ADXL345 ultralow power, 3-axis digital accelerometer includes the flexibility of a first in, first out (FIFO) buffer. Utilizing the FIFO extends even further the ability to differentiate user gestures, provide further power savings, and increase system performance while decreasing the need for host processor interaction.

This application note discusses the FIFO technology of digital accelerometers from Analog Devices by providing a description of the FIFO and its modes of operation, focusing on the ADXL345. Example configurations for each mode are provided, and some examples of how to use the FIFO for signal processing and power savings are also discussed.

TABLE OF CONTENTS

Introduction 1
FIFO Description 3
Retrieving Data from the Data Registers 4
Retrieving Data from the FIFO 5
Communication Speed, Data Rate, and Termination Time ... 6
Monitoring FIFO Status 6
FIFO Configuration 7
FIFO Modes 7
Examples of FIFO Applications 11
Power Savings 11
Signal Processing and Filtering 12

FIFO DESCRIPTION

The FIFO is capable of holding up to 32 sample sets of data. Each sample set of data consists of one x-axis sample, one y-axis sample, and one z-axis sample. One x-axis sample is the data normally held in the DATAX0 and DATAX1 registers, with a sample of y- and z-axis data corresponding to their appropriate register. In addition, one more sample set of data is held in the output filter of the accelerometer, effectively creating a 33rd level to the FIFO. Figure 1 shows the representation of the FIFO.

After sensing and digitizing the acceleration data, the sample is released from the output filter. This data then is placed into the closest available spot to the data registers in the FIFO. When the data registers are read, the data in FIFO[0] is obtained and then removed from the FIFO, allowing the rest of the stack to shift one level closer to the data registers. That is, after reading FIFO[0], the sample in FIFO[1] is shifted into FIFO[0], the sample in FIFO[2] is shifted into FIFO[1], and so on. If there is no new sample available to be shifted into FIFO[0] after a read, the old data remains until new data is available in the output filter and then replaces the sample in FIFO[0]. The operation of the output filter when the FIFO is full depends on the mode of operation.

	X-AXIS		Y-AXIS		Z-AXIS	
OUTPUT FILTER	DATAX0	DATAX1	DATAY0	DATAY1	DATAZ0	DATAZ1
FIFO[31]	DATAX0[31]	DATAX1[31]	DATAY0[31]	DATAY1[31]	DATAZ0[31]	DATAZ1[31]
FIFO[30]	DATAX0[30]	DATAX1[30]	DATAY0[30]	DATAY1[30]	DATAZ0[30]	DATAZ1[30]
⋮	⋮	⋮	⋮	⋮	⋮	⋮
FIFO[2]	DATAX0[2]	DATAX1[2]	DATAY0[2]	DATAY1[2]	DATAZ0[2]	DATAZ1[2]
FIFO[1]	DATAX0[1]	DATAX1[1]	DATAY0[1]	DATAY1[1]	DATAZ0[1]	DATAZ1[1]
FIFO[0]	DATAX0[0]	DATAX1[0]	DATAY0[0]	DATAY1[0]	DATAZ0[0]	DATAZ1[0]
DATA REGISTER (0x32 TO 0x37)						

08234-001

Figure 1. FIFO Buffer Representation

RETRIEVING DATA FROM THE DATA REGISTERS

It is recommended that all reads from the data registers be done as a multiple-byte read. A multiple-byte read is a read in which the communication does not end until the last byte in the multiple-byte read is obtained. This is done to ensure that the data read from the registers corresponds to the same sample. If single-byte reads are performed, there is a chance that data may change between reads, causing the read data to be mixed between different samples.

For SPI 3- or 4-wire communication, a multiple-byte read is accomplished by setting the multiple-byte bit and then continuing to send sets of eight clock pulses for each byte. With the multiple-byte bit set, the register pointer shifts to the next value after each set of eight clock pulses, as shown in Figure 2. Refer to the ADXL345 data sheet for a description of the location of the multiple-byte bit.

For I²C communication, a multiple-byte read is performed in a similar manner as the multiple-byte read for SPI, taking into account the necessary I²C protocol for performing a read. After initiating a read, continuing to send sets of nine clock pulses (a ninth clock pulse is required in I²C communication for the acknowledge bit) shifts the register pointer to the next value. More information about performing a read in I²C communication can be found in the ADXL345 data sheet or the *UM10204 I²C-Bus Specification and User Manual*, Rev. 03—19 June 2007, available from NXP Semiconductors.

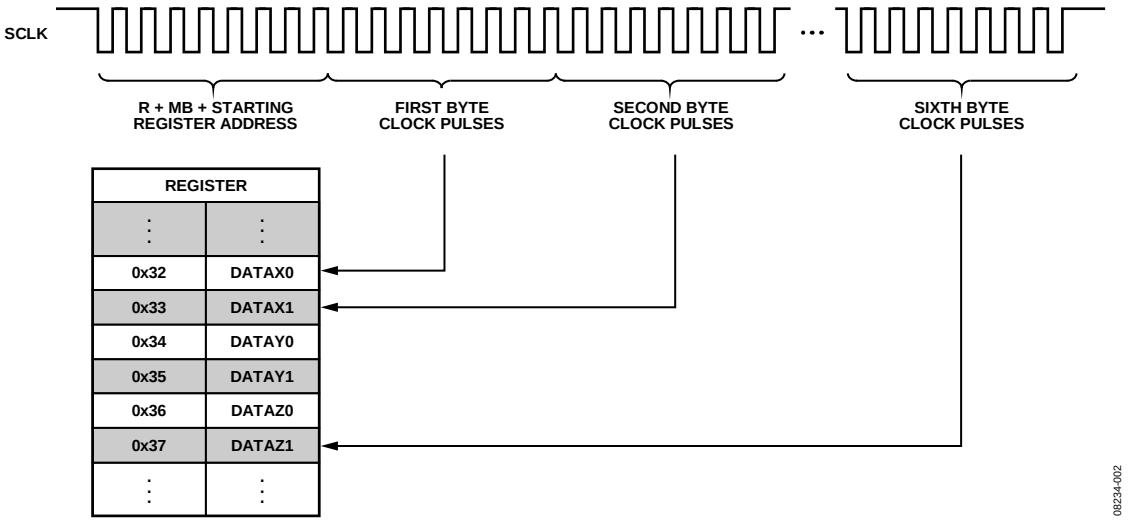

Figure 2. Register Pointer Shift During Multiple-Byte SPI Read in the ADXL345

RETRIEVING DATA FROM THE FIFO

When the FIFO is in use, reading the data registers returns the sample stored at FIFO[0]. To initiate the process of removing the old FIFO[0] value and shifting the samples down, also known as popping the FIFO, the last clock pulse of the DATAZ1 register must be received or communication terminated. Termination of communication during a SPI transmission is performed by deasserting the $\overline{CS}$ pin. The master device, by issuing a stop command, terminates I²C communication. To retrieve the next sample, perform the multiple-byte read again, including the necessary register addressing, starting at the initial data register as shown in Figure 3.

Similar to reading data from the data registers, it is required that a multiple-byte read be performed if more than one byte of data per sample is retrieved from the FIFO. Performing a single-byte read when using the FIFO causes data to shift between reads and mix samples from different sets of data because the end of a read operation causes the FIFO to pop. For instance, using a single-byte read to read the LSB of the x-axis and then another single-byte read to retrieve the MSB of the x-axis results in the FIFO popping between reads and returning bytes from adjacent samples, not from the same sample (see Figure 4).

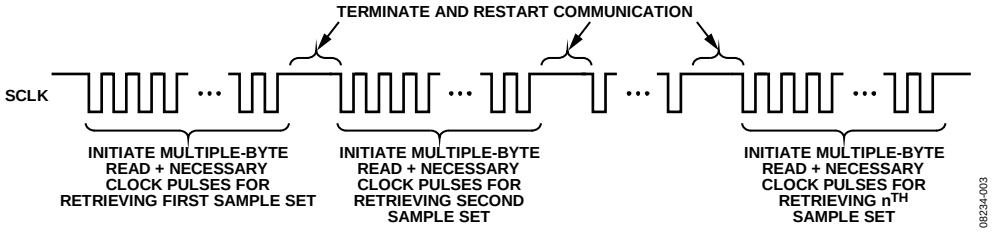

Figure 3. Termination and Restart of Communication Between FIFO Reads to Pop the FIFO

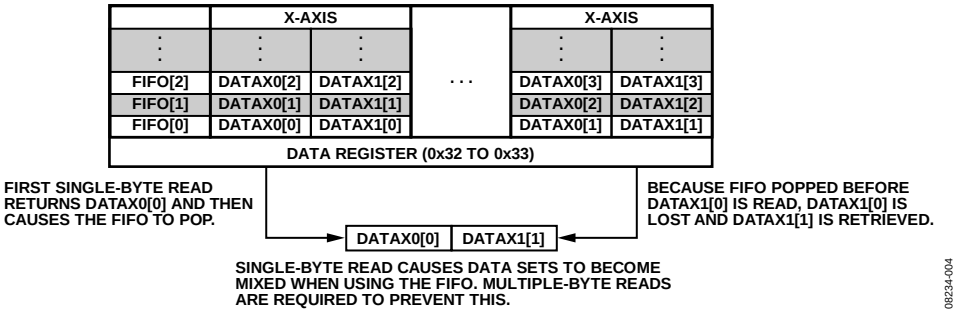

Figure 4. Sample Mixing Due to Single-Byte Reads When Using the FIFO

COMMUNICATION SPEED, DATA RATE, AND TERMINATION TIME

Because the rate at which data can be read from the FIFO depends on the communication speed, take care to select an appropriate data rate such that data does not fill the FIFO faster than it can be read. I²C protocol requires a start, slave address plus write, address, restart, and then a slave address plus read just to initiate the multiple-byte read. The minimum time necessary just for beginning the multiple-byte read for fast mode I²C is roughly 70 µs. If all six bytes of data are retrieved, the minimum amount of time to get a FIFO sample is roughly 207 µs, including all minimum start and stop setup and hold times and the minimum cycle period. Based on this, data can still be read out of the FIFO faster than it is put into the FIFO at the maximum data rate of 3200 Hz, corresponding to a period of 312.5 µs, but if a slower speed of I²C communication is used, the data rate may need to be adjusted. The efficiency of the host controller in I²C communication also has an impact on the maximum data rate that can be used and should be accounted for when determining this value.

Because SPI does not have the same requirements for initiating a read, speeds slower than I²C can be used at the same data rate to successfully retrieve the data. The system designer should verify that an appropriate data rate is selected for the bus speed.

The amount of time required between sequential sample reads (the terminate and restart communication time shown in Figure 3) depends on the method of communication and the speed. This occurs because the FIFO must pop between each sample and requires a minimum amount of time, roughly 5 µs, between the end of reading DATAZ1 or deasserting the $\overline{CS}$ pin and starting to read the next set of samples. For I²C, the minimum amount of time in fast mode to initiate a read (70 µs) provides sufficient time to meet this 5 µs requirement.

For SPI communication at 5 MHz, the initiation of a read only requires roughly 1.6 µs, requiring that the $\overline{CS}$ pin be deasserted for a minimum of 3.4 µs. Similarly, with a SPI communication speed of 1.6 MHz it takes at least 5 µs to initiate a read, suggesting that for SPI speeds below 1.6 MHz, only the minimum amount of time the $\overline{CS}$ pin must be deasserted, $t_{CS,DIS}$, must be met.

MONITORING FIFO STATUS

The FIFO_STATUS register is used to monitor the status of the FIFO. The FIFO_TRIG bit is discussed in the Trigger Mode section. The entries bits correspond to how many samples are currently in the FIFO. Although entries is six bits long, the maximum possible value is 0x20 or 32 decimal. Note that if the FIFO_STATUS register is read before the FIFO is allowed to completely pop, the value in entries is incorrect because it does not update until after the FIFO finishes shifting. If a multiple-byte read is performed starting with the data registers and intended to end after reading the FIFO_STATUS register, a minimum of 5 µs to allow the FIFO to pop is required between finishing the read of DATAZ1 and beginning the read of FIFO_STATUS. This can be done by clock stretching, or stopping the clock until the 5 µs minimum is met, and then continuing communication.

Table 1. FIFO_STATUS Register (Read Only)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_TRIG	X[1]	Entries					

[1] X = don't care.

FIFO CONFIGURATION

Configuration of the FIFO is done through the FIFO_CTL register. Through the FIFO_CTL register, the mode of operation for the FIFO is determined and configured by the FIFO_MODE bits. The trigger interrupt for trigger mode is also configured in the FIFO_CTL register as well as the samples bits, which operate differently depending on the selected mode.

Table 2. FIFO_CTL Register (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
FIFO_MODE		Trigger	Samples				

To prevent the generation of false interrupts or the unintentional filling of the FIFO, configure the FIFO and the triggering interrupt(s) for trigger mode or watermark before enabling any corresponding interrupts and before placing the part into measurement mode. As a general rule, configure the digital accelerometers in the following order:

1. Set data parameters such as data rate, measurement range, data format, and offset adjustment.
2. Configure interrupts (do not enable): thresholds and timing values, and map interrupts to pins.
3. Configure FIFO (if in use): mode, trigger interrupt if using trigger mode, and samples bits.
4. Enable interrupts: INT_ENABLE register.
5. Place part into measurement mode: POWER_CTL register.

Following this order prevents false watermark interrupts when the FIFO is empty and the samples bits are configured to its default value of 0, as well as false interrupts due to interrupts being enabled with default values.

FIFO MODES

The FIFO has four modes of operation: bypass mode, FIFO mode, stream mode, and trigger mode. Each mode is configured through the FIFO_MODE bits in the FIFO_CTL register. Refer to the ADXL345 data sheet for which values correspond to which mode.

Bypass Mode

In bypass mode, the FIFO is effectively disabled. Each time a new sample is ready at the output filter, the value stored in the data registers or FIFO[0] is immediately replaced by the new data. If the data registers are read again before new data is ready, the old data remains until a new sample is available and then immediately replaces the old data in FIFO[0] or the data registers. If the current sample of data is being read when new data is available, a double-buffered output allows the read to finish first and then places the new data into the output filter, preventing the loss of data due to a read.

In this mode, the DATA_READY interrupt is normally used to signal to the master device that new data is ready to be retrieved. The DATA_READY interrupt is configured by configuring the DATA_READY bit in the INT_MAP register to the interrupt pin of choice and then setting the DATA_READY bit in the INT_ENABLE register. The interrupt that DATA_READY is mapped to triggers when new data is available and is cleared by reading the data in the data registers. The polarity of the DATA_READY interrupt is active high by default. DATA_READY and all other interrupts can be configured to active low by setting the INT_INVERT bit in the DATA_FORMAT register.

Table 3. INT_MAP/INT_ENABLE Registers (Read/Write)

D7	D6	D5	D4
DATA_READY	SINGLE_TAP	DOUBLE_TAP	Activity
D3	**D2**	**D1**	**D0**
Inactivity	FREE_FALL	Watermark	Overrun

FIFO Mode

In FIFO mode, the FIFO collects new samples until full and then discards new samples until room is made available in the FIFO. When room is made available by reading samples, the newest data available in the output filter begins to fill the FIFO again. If the FIFO fills completely and samples are not read quickly enough, there is a disconnect in the data due to samples being discarded between those stored in the FIFO when it filled and the new ones shifted in after space was made available. This is demonstrated in Figure 5. The normal method of operation in FIFO mode is to disable DATA_READY (if enabled) and, instead, use the watermark interrupt. In FIFO mode, the watermark interrupt is triggered when the number of samples stored in the FIFO, corresponding to the value in entries of the FIFO_STATUS register, is equal to the value written into the samples bits of the FIFO_CTL register. Using the watermark interrupt instead of the DATA_READY interrupt allows the FIFO to fill to the desired level (stored in the samples bits) and then, in a few sequential multiple-byte reads, empty the FIFO, reducing how often the accelerometer burdens the host processor. To configure the watermark interrupt, write a nonzero value to the samples bits. The value used is a system level choice and depends on how quickly the interrupt can be handled when it occurs. If it is handled quickly and reading of the FIFO can begin before a new sample is ready, a value close to 32 can be selected. If there is latency between the interrupt occurring and being serviced, select a lower value to prevent the FIFO from discarding samples. After configuring the value in the samples bits, map the watermark interrupt to either INT1 or INT2 via the INT_MAP register and then enable it by setting the watermark bit in the INT_ENABLE register.

Similar to the DATA_READY interrupt, the watermark interrupt remains set until the condition causing it is eliminated. This means that the watermark interrupt remains triggered as long as the value in entries is greater than or equal to the value in the samples bits. To clear the watermark interrupt, read the FIFO (via the data registers) until the number of samples in the FIFO is below the value stored in the samples bits; however, reading more than the minimum number of data samples is recommended to prevent the watermark interrupt from triggering too often, thus eliminating the benefit of using the FIFO.

An example of how the FIFO operates in FIFO mode with the watermark interrupt is shown in Figure 6. In this example, the samples bits are configured to a value of 6, causing the watermark interrupt to trigger when a sample is pushed into FIFO[5]. After a few more samples the interrupt is serviced, and six samples are read from the FIFO, bringing the number of entries in the FIFO to below 6 and clearing the watermark interrupt.

Note that placing the part into bypass mode from any other mode of FIFO operation causes the FIFO to be emptied. Therefore, retrieve the data in the FIFO before placing the part into bypass mode. Placing the part into standby mode from measure mode preserves the contents of the FIFO, but does not cause new samples to fill the FIFO. This could cause a disconnect in data samples if the FIFO is allowed to partially fill and is then placed into standby mode without first clearing the FIFO before returning to measure mode.

	FIFO CONTENTS							
OUTPUT FILTER	DATA[0]	DATA[1]	. . .	DATA[31]	DATA[32]	DATA[33]	DATA[34]	DATA[35]
FIFO[31]	X	X	. . .	DATA[31]	DATA[31]	DATA[31]	DATA[31]	DATA[35]
FIFO[30]	X	X	. . .	DATA[30]	DATA[30]	DATA[30]	DATA[30]	DATA[34]
FIFO[29]	X	X	. . .	DATA[29]	DATA[29]	DATA[29]	DATA[29]	DATA[31]
FIFO[28]	X	X	. . .	DATA[28]	DATA[28]	DATA[28]	DATA[28]	DATA[30]
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
FIFO[2]	X	X	. . .	DATA[2]	DATA[2]	DATA[2]	DATA[2]	DATA[4]
FIFO[1]	X	DATA[1]	. . .	DATA[1]	DATA[1]	DATA[1]	DATA[1]	DATA[3]
FIFO[0]	DATA[0]	DATA[0]	. . .	DATA[0]	DATA[0]	DATA[0]	DATA[0]	DATA[2]

TIME . . .

MULTIPLE-BYTE READ OF 2 SAMPLE SETS, DATA[0] AND DATA[1], CAUSES FIFO TO SHIFT. DATA[32] AND DATA[33] ARE LOST BECAUSE ROOM WAS NOT AVAILABLE UNTIL AFTER DATA[34] WAS SAMPLED. BECAUSE DATA[34] IS STILL THE NEWEST DATA WHEN SPACE IS MADE AVAILABLE IN FIFO, IT IS SHIFTED INTO FIFO[30] BEFORE DATA[35] IS MADE AVAILABLE.

08234-005

Figure 5. Fill Order of FIFO in FIFO Mode with Data Lost Due to Overrun

Stream Mode

When the FIFO operates in stream mode, new samples of data are constantly collected. If the FIFO is full, the oldest sample in FIFO[0] is discarded, the remaining samples are shifted down, and the new sample is pushed into FIFO[31]. The oldest samples continue to be discarded until the FIFO is read and room is made available for new samples. This process is shown in Figure 7.

Stream mode targets applications where an external stimulus, that is, a stimulus not from the accelerometer, is used to determine when to read acceleration data. This could be a button press or a host processor requesting the information from the accelerometer at a certain time. Because of this, interrupts are generally not used with stream mode. However, the watermark interrupt operates the same way in stream mode as in FIFO mode, triggering when the number of samples in the FIFO equals the value stored in the samples bits.

The feature most relevant to stream mode is the overrun bit in the INT_SOURCE register. When the FIFO fills and then has to discard a sample when a new sample is available, the overrun bit is set to signify that data has been lost due to the FIFO being full. This operates the same way in all modes of FIFO operation, but in FIFO mode and trigger mode data is typically read before the FIFO fills to prevent the loss of data. In Figure 7, the overrun bit is set when DATA[32] is available in the output filter, as DATA[0] was discarded. The bit clears after the FIFO is read, shifting the samples down and creating room in the FIFO. This prevents the loss of further data until the FIFO becomes full again.

Table 4. INT_SOURCE Register (Read Only)

D7 DATA_READY	**D6** SINGLE_TAP	**D5** DOUBLE_TAP	**D4** Activity
D3 Inactivity	**D2** FREE_FALL	**D1** Watermark	**D0** Overrun

Trigger Mode

In trigger mode, the FIFO begins operation similar to stream mode, collecting samples until full and then discarding the oldest samples to make room for new samples. After a triggering event occurs, the newest samples, with a total number equal to the value stored in the samples bits, are preserved with the remaining oldest samples discarded. The FIFO then begins to operate similar to FIFO mode, collecting new samples only until full and then discarding new samples until room is made available in the FIFO. This behavior is demonstrated in Figure 8 with a samples value of 15.

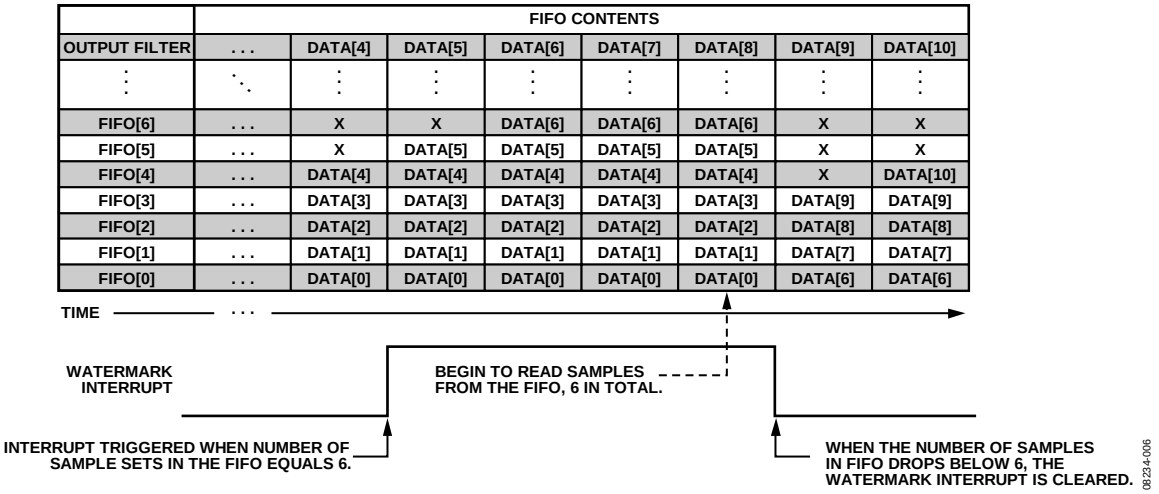

Figure 6. Example of FIFO Mode Operation Using Watermark Interrupt, Samples Value of 6

	FIFO CONTENTS							
OUTPUT FILTER	DATA[0]	DATA[1]	. . .	DATA[31]	DATA[32]	DATA[33]	DATA[34]	DATA[35]
FIFO[31]	X	X	. . .	DATA[31]	DATA[32]	DATA[33]	DATA[34]	X
FIFO[30]	X	X	. . .	DATA[30]	DATA[31]	DATA[32]	DATA[33]	X
FIFO[29]	X	X	. . .	DATA[29]	DATA[30]	DATA[31]	DATA[32]	DATA[35]
FIFO[28]	X	X	. . .	DATA[28]	DATA[29]	DATA[30]	DATA[31]	DATA[34]
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
FIFO[2]	X	X	. . .	DATA[2]	DATA[3]	DATA[4]	DATA[5]	DATA[8]
FIFO[1]	X	DATA[1]	. . .	DATA[1]	DATA[2]	DATA[3]	DATA[4]	DATA[7]
FIFO[0]	DATA[0]	DATA[0]	. . .	DATA[0]	DATA[1]	DATA[2]	DATA[3]	DATA[6]

TIME . . .

MULTIPLE-BYTE READ OF 3 SAMPLE SETS, DATA[3], DATA[4], AND DATA[5], CAUSES FIFO TO SHIFT, CREATING ROOM FOR NEW SAMPLES. DATA[0], DATA[1], AND DATA[2] ARE LOST BECAUSE ROOM WAS NOT AVAILABLE UNTIL AFTER DATA[34] WAS SAMPLED, CAUSING THEM TO BE DISCARDED TO MAKE ROOM FOR DATA[32], DATA[33], AND DATA[34].

08234-007

Figure 7. FIFO Operation in Stream Mode with the Oldest Samples Discarded

	FIFO CONTENTS							
OUTPUT FILTER	DATA[0]	DATA[1]	. . .	DATA[31]	DATA[32]	DATA[33]	DATA[34]	DATA[35]
FIFO[31]	X	X	. . .	DATA[31]	DATA[32]	DATA[33]	X	X
FIFO[30]	X	X	. . .	DATA[30]	DATA[31]	DATA[32]	X	X
FIFO[29]	X	X	. . .	DATA[29]	DATA[30]	DATA[31]	X	X
FIFO[28]	X	X	. . .	DATA[28]	DATA[29]	DATA[30]	X	X
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
FIFO[16]	X	X	. . .	DATA[16]	DATA[17]	DATA[18]	X	DATA[35]
FIFO[15]	X	X	. . .	DATA[15]	DATA[16]	DATA[17]	DATA[34]	DATA[34]
FIFO[14]	X	X	. . .	DATA[14]	DATA[15]	DATA[16]	DATA[33]	DATA[33]
⋮	⋮	⋮	⋱	⋮	⋮	⋮	⋮	⋮
FIFO[2]	X	X	. . .	DATA[2]	DATA[3]	DATA[4]	DATA[21]	DATA[21]
FIFO[1]	X	DATA[1]	. . .	DATA[1]	DATA[2]	DATA[3]	DATA[20]	DATA[20]
FIFO[0]	DATA[0]	DATA[0]	. . .	DATA[0]	DATA[1]	DATA[2]	DATA[19]	DATA[19]

TIME . . .

TRIGGER MODE OPERATES SIMILAR TO STREAM MODE BEFORE TRIGGER EVENT OCCURS

TRIGGER EVENT OCCURS, NEWEST 15 SAMPLES RETAINED IN FIFO

AFTER TRIGGER EVENT, FIFO OPERATES SIMILAR TO FIFO MODE.

08234-008

Figure 8. Operation of FIFO in Trigger Mode With a Samples Value of 15

The triggering event for trigger mode is any of the interrupts native to the accelerometer. If multiple interrupts are selected, the first interrupt to occur acts as the triggering event. The interrupt(s) of interest are configured (refer to the ADXL345 data sheet), then mapped to the appropriate interrupt in the INT_MAP register, and enabled in the INT_ENABLE register. Once the interrupt is configured, the trigger bit in the FIFO_CTL register is configured to correspond to the interrupt that should trigger the FIFO; a value of 0 selects INT1, whereas a value of 1 selects INT2 for the ADXL345. When an event occurs that causes an interrupt selected by the trigger bit to occur, the FIFO retains the newest samples determined by the value in the samples bits and then undergoes the transition between acting in stream mode and acting in FIFO mode. It is recommended that DATA_READY, watermark, and overrun interrupts not be used as triggering events for trigger mode.

After the triggering interrupt occurs, read the FIFO until empty. If the FIFO continues to operate in FIFO mode, the value in the samples bits can be adjusted to minimize how often the host processor must pull data from the accelerometer and the watermark interrupt must be configured, as described in the FIFO Mode section. Alternatively, the watermark interrupt can be configured from the start, mapped to the other interrupt pin, and the host processor configured to ignore that interrupt until after the triggering event occurs and the FIFO has been emptied for the first time. The value in the samples bits can then be adjusted and the host processor configured to respond to the watermark interrupt.

Due to the data in the FIFO having to adjust after the triggering event, a minimum of 5 µs is required between the triggering event occurring and the start of reading the FIFO. After 5 µs, the FIFO properly discards the oldest samples and shifts the retained samples.

To reset trigger mode so that additional trigger events can be recognized, first read the FIFO completely if data is needed. Then place the part into bypass mode in the FIFO_CTL register, which causes all remaining data in the FIFO to be discarded and resets the trigger. Place the part back into trigger mode by configuring the appropriate bits in the FIFO_CTL register. After this, the device is configured to respond to the next triggering interrupt.

EXAMPLES OF FIFO APPLICATIONS

POWER SAVINGS

Along with reducing how often the host processor needs to communicate with the accelerometer, the FIFO can be used for power savings in applications where the device is waiting for an inertial event to occur or logging data. This could include monitoring shock on a package being shipped or a pedometer, where the output does not need to update with every step, but can be allowed to update every few steps.

Given the already ultralow power consumption of the Analog Devices digital accelerometers, the biggest power savings come from being able to put the host processor and other peripherals to sleep when not in use. By using the FIFO, acceleration data can be collected continuously and the host processor woken up only when the FIFO is close to full, determined by the value in the samples bits, using the watermark interrupt. During the remaining time, when the FIFO is collecting data and the processor is not needed, the processor can go to sleep, significantly reducing power consumption of the system. Additionally, if other peripheral devices are present, they too can be turned off until needed by the system.

To demonstrate the power savings capabilities of the digital accelerometer's FIFO, Figure 9 shows a typical flow of data collection and current consumption. The current consumption of the ADXL345 is used at 100 Hz output data rate with a current and wake-up time profile comparable to the Analog Devices line of ARM7 microcontrollers. The FIFO is configured to generate a watermark interrupt when 30 samples are available in the FIFO and data is read via SPI at 5 MHz, which is roughly 500 µs to read 30 complete samples.

In the Figure 9 example, it takes roughly 3.5 ms to wake up the host processor, retrieve the 30 samples of data, and then go back to sleep. The average current consumption by using the FIFO falls to only 460 µA, compared to the over 11 mA required if the host processor were left on the whole time. This is a power savings of close to 96%.

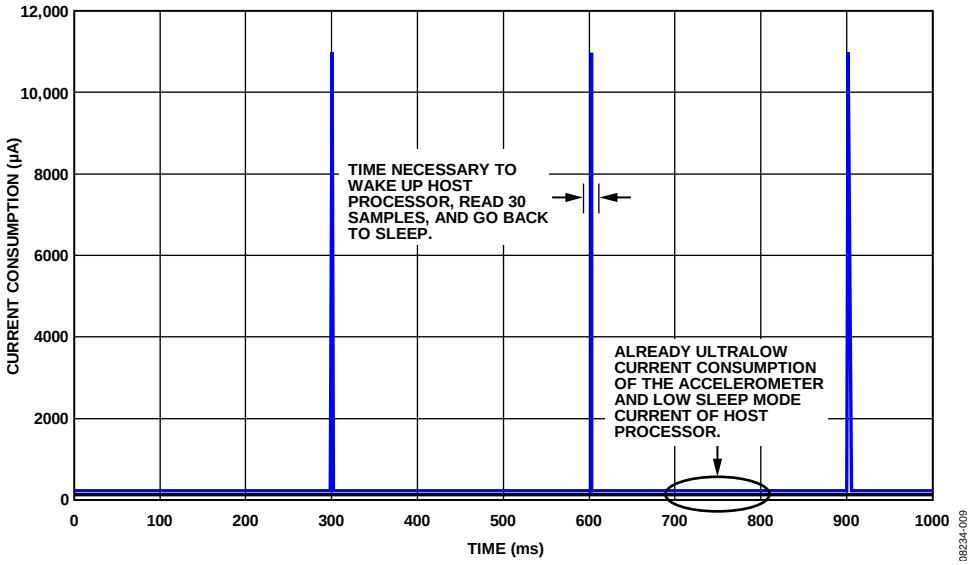

Figure 9. Resulting Power Savings by Placing Host Processor into Sleep Mode While Using FIFO

SIGNAL PROCESSING AND FILTERING

In some applications, such as tilt sensing, noise reduction and greater resolution may be required. Due to the nature of the current line of digital accelerometers from Analog Devices, operation below 100 Hz output data rate does not improve the noise performance below that of the noise at 100 Hz output data rate. If lower noise is desired, filter the output externally. The 3.9 mg/LSB scale factor corresponds to roughly a 0.25° resolution in tilt, which is sufficient for most applications.

If lower noise or greater resolution is desired, filtering or oversampling techniques can be used. In both cases, multiple samples of data must be processed. For these applications the FIFO can be allowed to fill until the required number of samples needed is reached, removing the need for the host processor to continually retrieve and store the data while waiting for enough data to be available for processing. Once the FIFO contains all the necessary data, all of the samples can be quickly pulled from the FIFO and processed, reducing the burden on the host processor.

For a simple averaging filter, expect a reduction in noise approximately equal to the square root of the number of samples averaged. For instance, if 100 samples are taken with a root-mean-squared (rms) noise of 1 LSB, using a simple four-sample average filter reduces that noise to approximately 0.5 LSB rms. To demonstrate this, 100 data points can be taken using the ADXL345 at an output data rate of 100 Hz, and then every four samples averaged into a single sample, using the FIFO to trigger when each group of four samples is ready. This yields a visible reduction in noise and an effective output data rate of 25 Hz because the host processor only needs to retrieve data from the part once for every four samples, as shown in Figure 10.

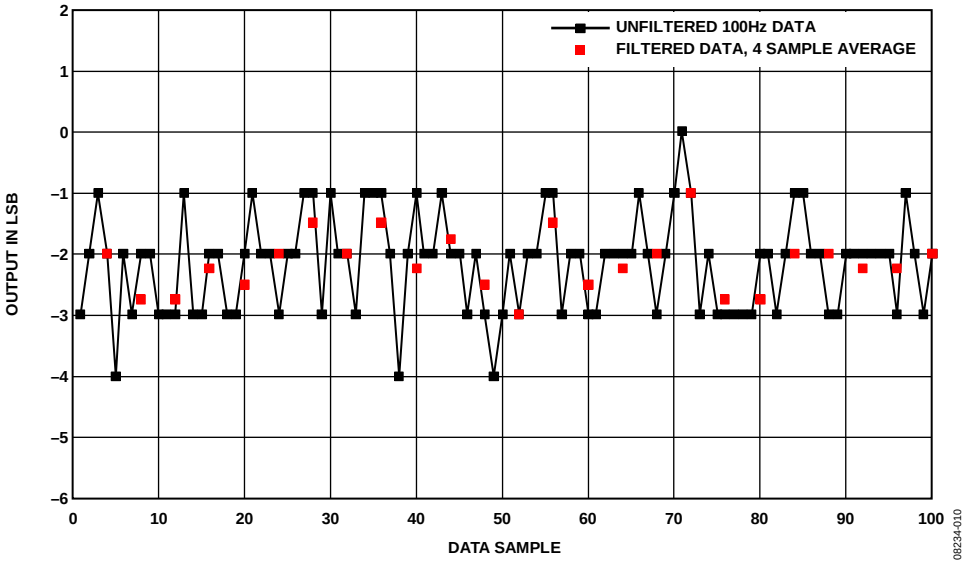

Figure 10. Noise Reduction with a Four-Sample Averaging Filter Utilizing FIFO

©2009 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.
AN08234-0-6/09(0)