Freescale Semiconductor
Application Note

Document Number: AN4693
Rev. 0, 11/2012

Using the Xtrinsic FXLS8471CQ Transient-Acceleration Function

by: Talat Ozyagcilar
Applications Engineer

1 Introduction

This application note demonstrates the Transient-Acceleration function of the FXLS8471CQ with use-case examples.

Two use cases identified for the Transient-Acceleration function are included:

- Detection of shake along an axis (using the function with high-pass filter enabled).
- Detection of linear acceleration or tilt along an axis (using the function with high-pass filter disabled).

This application note makes references to the FXLS8471CQ data sheet.

1.1 Keywords

Transient Acceleration, Gesture, Flick, Tap, Shake, Pedometer, Motion Detection, Tilt.

Contents

1 Introduction 1
1.1 Keywords 1
1.2 Related Documentation 2
2 Overview 2
2.1 Use case 1: Detecting shake along an axis 3
2.2 Use case 2: Detect linear acceleration or tilt along an axis 5

© 2013 Freescale Semiconductor, Inc. All rights reserved.

freescale Xtrinsic

1.2 Related Documentation

The FXLS8471CQ device features and operations are described in a variety of reference manuals, user guides, and application notes. To find the most-current versions of these documents:

1. Go to the Freescale homepage at:

   http://www.freescale.com/

2. In the Keyword search box at the top of the page, enter the device number FXLS8471CQ.
3. In the Refine Your Result pane on the left, click on the Documentation link.

2 Overview

The main purpose of the Transient-Acceleration function is to detect sudden changes in acceleration along a particular axis. This function can be used to sense tap, flick, and shake events.

When the high-pass filter is bypassed, the functionality becomes similar to the motion-detection function; in this mode, acceleration greater than a programmable threshold is detected (along an axis). Motion or tilt can be detected in this mode of operation.

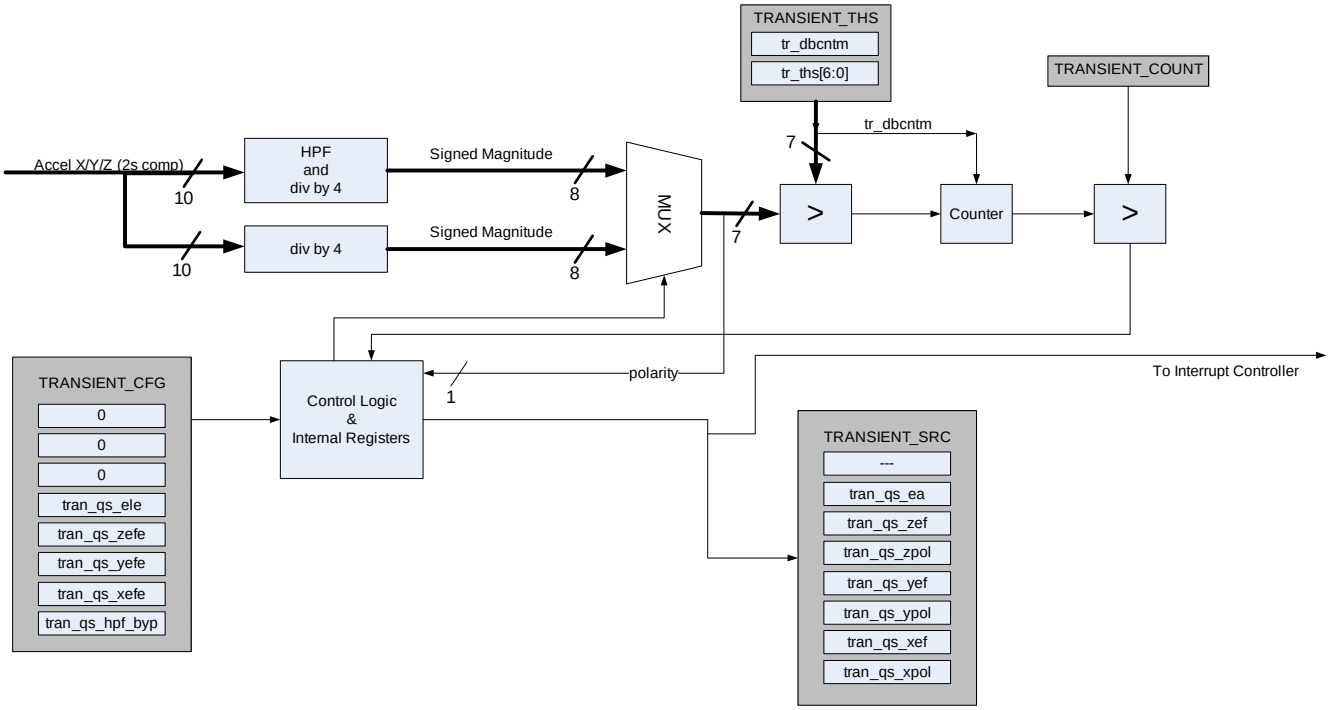

Figure 1. Transient-Acceleration function block diagram

2.1 Use case 1: Detecting shake along an axis

This use case utilizes the transient-detection function with the high-pass filter enabled. In this mode of operation, the function will detect an “instantaneous” *change* in acceleration that exceeds the programmed threshold (set via TRANSIENT_THS register) for the programmed time period (set via TRANSIENT_COUNT register) on a particular axis. An “instantaneous” change in acceleration really means the change in the acceleration observed by the part between two consecutive sample acquisitions; this depends on the current ODR (set via CTRL_REG1, see Equation 1).

$$\left|G_axis_{HPF}(nT)\right| = \left|G_axis(nT) - G_axis((n-1)T)\right| > r \qquad \textbf{Eqn. 1}$$

where $T = 1/ODR$, r is the programmable threshold and n represents the number of samples acquired since the part was transitioned into ACTIVE mode.

The python code snippet in Example 1 shows how to program the motion-detection function to detect an “instantaneous” acceleration change exceeding r = 315 mg for a minimum period of 40 ms on either the X or Y axes.

Key points to note are:

- Setting of the ODR to 12.5 Hz, OSR to normal (default), and putting the part into ACTIVE mode using CTRL_REG1 (0x2A).
- Setting of TRANSIENT_THS (0x1F) register to 0x85, specifying the behavior of the debounce counter (0x80, clear counter when acceleration change goes below threshold) and setting the threshold to 5 x 63 mg = 315 mg (0x05)
- Setting of TRANSIENT_COUNT (0x20) register to 0x02, making the minimum period for the acceleration change being over the threshold 40 ms (see Table 117 in data sheet).
- Enabling of X and Y axes to participate in the detection function and raise an interrupt when the threshold is exceeded via the TRANSIENT_CFG (0x1D) register.

Once the part is set up in this way, an acceleration change of greater than 315 mg between consecutive samples, for a minimum duration of 40 ms, will cause the INT1 pin to be asserted (active low). At this point, the user application can read register TRANSIENT_SRC (0x1E) in an interrupt service routine, clearing the interrupt as well obtaining information on the nature of the event detected (i.e. polarity and axis). This example uses polling for checking the state of the INT1 pin.

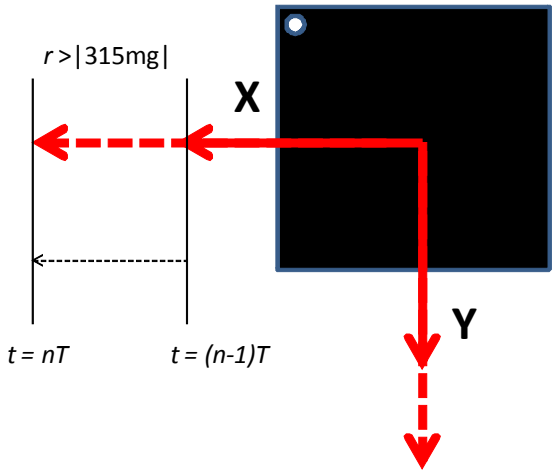

Figure 2. Detect an “instantaneous” acceleration change in X and Y axes

Example 1. Python Code snippet

```
# TRANSIENT_THS: 63mg x 5 = 315mg, debounce behavior : clear when condition not true

write_byte( [0x1F, 0x85] )  # write a byte 0x85 to address 0x1F

# TRANSIENT_COUNT = 40ms (based on ODR, see Table 117 in data sheet)

write_byte( [0x20, 0x02] )

# A_TRAN_INIT_MSB... initial reference set to 0g for all axes

write_byte( [0x79, 0x00] )
write_byte( [0x7A, 0x00] )
write_byte( [0x7B, 0x00] )
write_byte( [0x7C, 0x00] )

# TRANSIENT_CFG:
# tran_qs_ele = 1 => event latching enabled
# tran_qs_zefe = 0 => do not detect events on z-axis
# tran_qs_yefe = 1 => y-axis event detection enabled
# tran_qs_xefe = 1 => x-axis event detection enabled
# tran_qs_hpf_byp = 0 => high-pass filter is NOT bypassed

write_byte( [0x1D, 0b00010110] )

# enable interrupts for the feature using CTRL_REG4

write_byte( [0x2D, 0x20] )

# route interrupts to INT1 pin using CTRL_REG5

write_byte( [0x2E, 0x20] )
```

```
# Enable device with ODR = 12.5Hz using CTRL_REG1,
# OSR=NORMAL (def)

write_byte( [0x2A, 0x29] )


# Wait for INT1 to assert and clear interrupt by reading register TRANSIENT_SRC (0x1E)

while( True ):
   transition = aa_gpio_change( handle, 100 )
   if (transition & INT1_PIN ) == INT1_PIN:
      print "No interrupt..."
      continue
   (count, dataIn) = read_dev( 0x1E, 1 )
   print "TRAN_SRC = 0x%X" % dataIn[0]
   if( (dataIn[0] & 0x40) == 0x40 ):
      print "Event detected!!!"
```

2.2 Use case 2: Detect linear acceleration or tilt along an axis

This use case configures the function with the high-pass filter bypassed. In this mode, the function works in a similar manner to the Freefall/Motion detection feature; an event is raised when the *static* acceleration exceeds the programmable threshold for the programmable time period (debounce time). In other words, when the acceleration goes above 315 mg on an enabled axis (X or Y in this example) for longer than 40 ms, an event flag is raised which can be used to generate an interrupt event (mapped to either the INT1 or INT2 pins).

In this use-case setting r = 315 mg results in a tilt-angle threshold of ±18.4 degrees from the horizontal (Equation 3, Figure 3).

$$|G_{axis}| = |g\,sin(\alpha)| > r \qquad \textit{Eqn. 2}$$

$$|\alpha| > sin^{-1}\left(\frac{r}{g}\right) \qquad \textit{Eqn. 3}$$

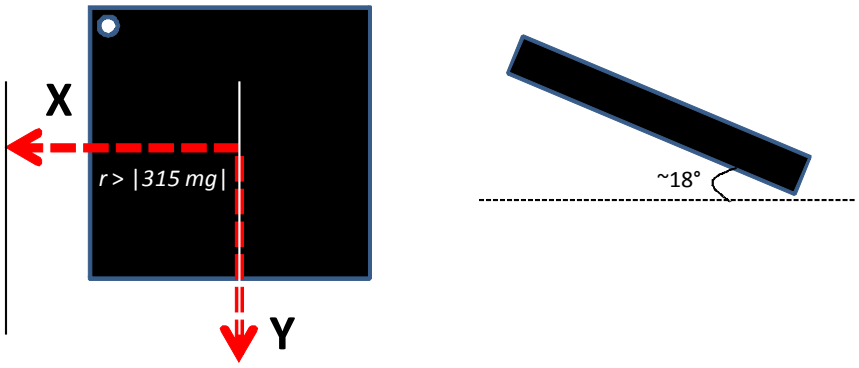

Figure 3. Detect static acceleration level or tilt in X and Y axes

Example 2. Python Code snippet (only differences from initial example code are shown)

```
…
…

# TRANSIENT_CFG:
# tran_qs_ele = 1 => event latching enabled
# tran_qs_zefe = 0 => do not detect events on z-axis
# tran_qs_yefe = 1 => y-axis event detection enabled
# tran_qs_xefe = 1 => x-axis event detection enabled
# tran_qs_hpf_byp = 1 => high-pass filter is bypassed

write_byte( [0x1D, 0b00010111] )
…
…
```

How to Reach Us:

Home Page:
www.freescale.com

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

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

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale 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 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 does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: http://www.reg.net/v2/webservices/Freescale/Docs/TermsandConditions.htm.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. Xtrinsic is a trademark of Freescale Semiconductor, Inc.

All other product or service names are the property of their respective owners.

© 2013 Freescale Semiconductor, Inc. All rights reserved.

AN4693
Rev. 0
11/2012