1. Introduction

This application note describes pedometer (step counter) engine provided by **KX116/KX126/KX127** accelerometers. Register map structure and its configuration are also explained.

The pedometer engine of KX116/KX126/KX127 is configured, controlled and monitored by configuring various parameters implemented in the registers. In addition, there are controls for data filtering, data buffering, as well as interrupt sources and controls.

2. Data Processing Diagram

Figure 1 shows a general principle for KX116/KX126/KX127 engine data processing. The *Low Current* (engine) averaging runs at the maximum rate of all enabled engine ODRs or at the *Low Current* ODR of the accelerometer (if it is configured to *Low Current*). Further, each engine runs at its own ODR rate and thus ignores extra output data from the averaging algorithm.

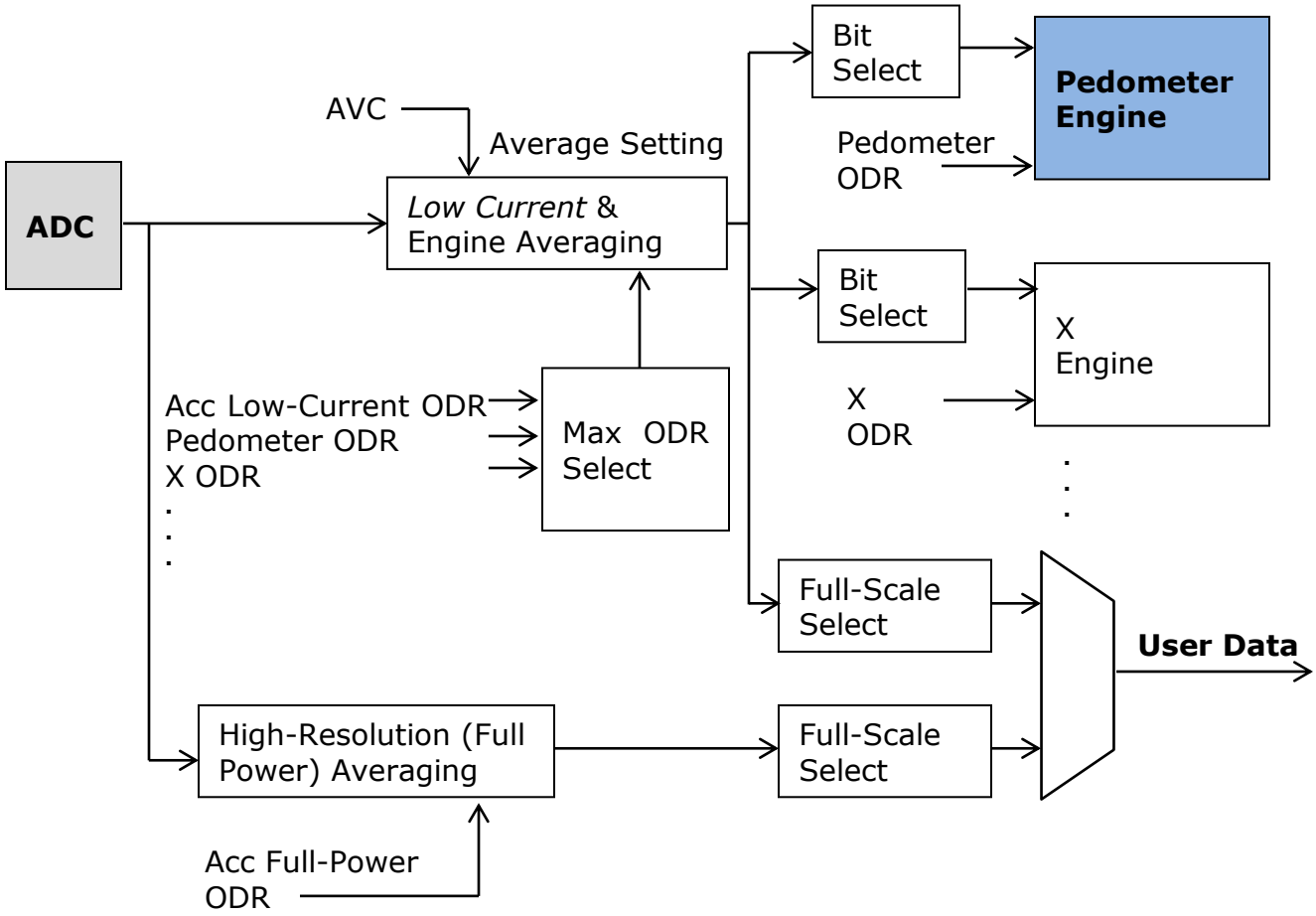

Figure 1: Data processing schematic for KX116/KX126/KX127 engines

36 Thornwood Dr. — Ithaca, NY 14850 USA
Tel: 607-257-1080 — Fax: 607-257-1146
www.kionix.com — info@kionix.com

3. Quick Start Implementation

Here we present basic ways how to initialize, enable and disable pedometer (step counter) function. These cursory solutions are provided as a means for configuring the part to a known operational state. Note that these conditions just provide a starting point, and the values may vary as developers refine their application requirements. Besides the pedometer function, a number of other internal functions (e.g. Tap/Double-Tap™ detection, Wake-up, Tilt position) and output registers are available with the same type of settings.

4. Enable/Disable Pedometer Function

KX116/KX126/KX127 offers a built-in pedometer algorithm for detecting steps and counting the total number of steps over a period of time. Operation of the pedometer is controlled by modifying CNTL1 register at address 0x1A, INC registers at 0x20 – 0x26, and PED_CNTL registers located in addresses 0x41 – 0x4C.

Pedometer function is configured and enabled for the step detection algorithm as below.

a) Write 0x00 to Control Register 1 (CNTL1) to set the accelerometer in stand-by mode (PC1 = 0), and disable the pedometer (step counter) function (PDE = 0).

Register Name	R/W	Address	Value
CNTL1	R/W	0x1A	0x00

As a good practice, it is recommended that the PDE bit 1 in CNTL1 register is also set to zero before modifying the following register values.

b) Write 0x10 and 0x27 (10 000 steps as an example) to Pedometer Step Counter Watermark registers (LSB: PED_STPWM_L, MSB: PED_STPWM_H) to set the 16-bit watermark threshold for step counting. When the threshold value is exceeded, then the watermark interrupt is triggered on the selected physical interrupt pin.

Register Name	R/W	Address	Value
PED_STPWM_L	R/W	0x41	0x10
PED_STPWM_H	R/W	0x42	0x27

A decimal number 10 000 is converted into the hexadecimal format (order: MSB, LSB) as 0x2710.

c) Write 0x2C to Pedometer Control Register 2 (PED_CNTL2) to select 100 Hz output data rate (ODR) for the pedometer engine (PED_ODR<3:0>=1100) and to set the scaling factor for the output high-pass filter to 4 (HPS<2:0>=010).

Register Name	R/W	Address	Value
PED_CNTL2	R/W	0x44	0x2C

Note: For operating pedometer at ODR of 50 Hz, set PED_ODR <3:0> bits to 0x06 in PED_CNTL2 register

d) Write 0x7B to Low Power Control register LP_CNTL to set the number of internal acceleration samples used by pedometer engine to 128* (AVC<2:0>=111).

Register Name	R/W	Address	Value
LP_CNTL	R/W	0x37	0x7B

*Note: Changing this setting will affect the number of sample used for averaging for *all enabled* digital engines (Directional Tap™, Tilt, Wake-Up, Back-to-Sleep, Free fall, Pedometer) both in *High Resolution* and *Low Current* modes and has a direct effect on noise and power consumption. The higher the number of sample used for averaging, the lower the noise and the higher the power consumption.

e) Write 0x61 to Interrupt Control 7 (INC7) to set the step counter overflow interrupt to be reported on the physical interrupt pin INT2 (STPOVI2 = 1), the step counter watermark interrupt to be reported on the physical interrupt pin INT2 (STPWMI2 = 1), and the step counter increment interrupt to be reported on the physical interrupt pin INT1 (STPINCI1 = 1).

Register Name	R/W	Address	Value
INC7	R/W	0x26	0x61

f) Write 0x20 to Interrupt Control Register 1 (INC1) to enable physical interrupt pin INT1 (IEN1 = 1), to set the polarity of the physical interrupt to activate low (IEA1 = 0), and to create a latched interrupt (IEL1 = 0).

Register Name	R/W	Address	Value
INC1	R/W	0x20	0x20

g) Write 0x20 to Interrupt Control Register 5 (INC5) to enable physical interrupt pin INT2 (IEN2 = 1), to set the polarity of the physical interrupt to active low (IEA2 = 0), and to create a latched interrupt (IEL2 = 0).

Register Name	R/W	Address	Value
INC5	R/W	0x24	0x20

h) In order to configure the pedometer functionality, the following values should be written in the Pedometer Control Registers (**ODR = 100 Hz**).

Register Name	R/W	Address	Value
PED_CNTL1	R/W	0x43	0x66
PED_CNTL2	R/W	0x44	0x2C
PED_CNTL3	R/W	0x45	0x17
PED_CNTL4	R/W	0x46	0x1F
PED_CNTL5	R/W	0x47	0x24
PED_CNTL6	R/W	0x48	0x13
PED_CNTL7	R/W	0x49	0x0B
PED_CNTL8	R/W	0x4A	0x08
PED_CNTL9	R/W	0x4B	0x19
PED_CNTL10	R/W	0x4C	0x1C

The above Control Registers adjust a number of pedometer internal features like noise tolerance, step detection sensitivity, characteristics of a step, etc. These values have been tested to provide successful step detection assuming walking and the sensor was placed in the pocket. For additional details, see Section 7 below.

In addition to the basic configuration above (ODR = 100 Hz), pedometer can be employed at **ODR = 50 Hz** with the following settings:

Register Name	R/W	Address	Value
PED_CNTL1	R/W	0x43	0x56
PED_CNTL2	R/W	0x44	0x26
PED_CNTL3	R/W	0x45	0x17
PED_CNTL4	R/W	0x46	0x1F
PED_CNTL5	R/W	0x47	0x0E
PED_CNTL6	R/W	0x48	0x0B
PED_CNTL7	R/W	0x49	0x02
PED_CNTL8	R/W	0x4A	0x03
PED_CNTL9	R/W	0x4B	0x0D
PED_CNTL10	R/W	0x4C	0x10

Advanced notes for parameter settings:

1) STP_TH<2:0> bits in PED_CNTL1 registers are used to allow a successful start for step detection. It is a threshold value for discarding step counting, if not enough steps is coming. The value range is 0,1, …, 7 that refers to 0,1, …, 14 steps. The default value is 6, which means actual 12 steps threshold. A threshold value is written in the PED_CNTL1 as follows:

Register Name	R/W	Address	Value
PED_CNTL1	R/W	0x43	0x66*

*Also sets MAG_SCALE<3:0> scaling factor to 6.

2) Sensitivity of the step detection function can be controlled by Control Register PED_CNTL5. The value range for this threshold parameter value is 0,1, …, 255 with the default value of 36 (0x24) (@ ODR = 100 Hz).

Register Name	R/W	Address	Value
PED_CNTL5	R/W	0x47	0x24

This is an essential parameter value for adjusting how easily the steps will be detected. For example, if the measurement conditions change

significantly (e.g. accelerometer position, specific environment conditions, walking/running style) then a new value for PED_CNTL5 will be necessary. A proper value of PED_CNTL5 should be determined experimentally to produce good step counting results. See Section 7.2 for additional details.

i) Write 0xC2 to Control Register 1 (CNTL1) to set the accelerometer (PC1 = 1), and set the pedometer into operating mode (PDE = 1).

Register Name	R/W	Address	Value
CNTL1	R/W	0x1A	0xC2

Now the pedometer is running using ±8 g range.

The Pedometer function is disabled as follows:

Write 0x80 to Control Register 1 (CNTL1) to disable the Pedometer (step counter) function (PDE = 0).

Register Name	R/W	Address	Value
CNTL1	R/W	0x1A	0x80

5. Monitoring Pedometer Output

After the pedometer function is enabled, the step counting result and the related interrupt sources can be read as below.

The pedometer step counter value is read from the 16-bit pedometer step counter registers PED_STP_L and PED_STP_H:

Register Name	R/W	Address
PED_STP_L	R	0x0E
PED_STP_H	R	0x0F

There are three interrupts reporting on the pedometer status and operation. The actual interrupt source is indicated at the source registers as follows:

a) **Step counter overflow** is reflected in bit 7 of INS1 register (STPOVI bit).

In case of Step counter overflow (the STPOVI bit is set in the INS1 register), an interrupt is reported when the last step to fill up the counter is detected. The step counter is a 16-bit register in which an overflow should occur on

every 65536 counts. The application should keep track of the overflow events.

b) **Exceeding the step counter watermark value** is reflected in bit 6 of INS1 register (STPWMI bit).

c) **Increment in the step counter value** is reflected in bit 1 of INS2 register (STPINCI bit).

In latched mode, the INT_REL register must be read in order to clear the physical interrupt pin. By reading this register, latched interrupt source information (INS1, INS2) is cleared and physical interrupt latched pin (INT1, INT2) is changed to its inactive state. Read value is dummy.

The step count will be reset in the following conditions:

- when either the low byte of the step counter (PED_STP_L) or the high byte of the step counter (PED_STP_H) is read,
- during the device soft reset, i.e. when setting SRST bit to 1 in the CNTL2 register, and
- during the device power up.

Some further notes:

- the step counter function will not be reset during the pedometer disable/enable cycle nor toggling the PC1 bit in CNTL1 register.
- At a time when the step counter watermark value is exceeded the STPWMI bit is set in the INS1 register. In spite of crossing the watermark value, the step counter continues to increment.

6. Step Detection Approach

Step detection is based on the measurement of step impact when touching the ground that is gathered using a 3D accelerometer sensor. With this scenario, walking and running impacts should be periodic and walking impact should also be greater than other general activity's impact.

First, a sum vector of the three acceleration components is calculated on every sample set:

$$ACC(t) = \sqrt{(acc_x(t))^2 + (acc_y(t))^2 + (acc_z(t))^2}$$

The step impact is obtained from the acceleration sum vector ACC(t) by integration within certain time limits. Hereby we define the instantaneous peak area, **A**:

$$A = \int ACC(t)dt$$

In order to figure out the relation between these two quantities, see Figure 2.

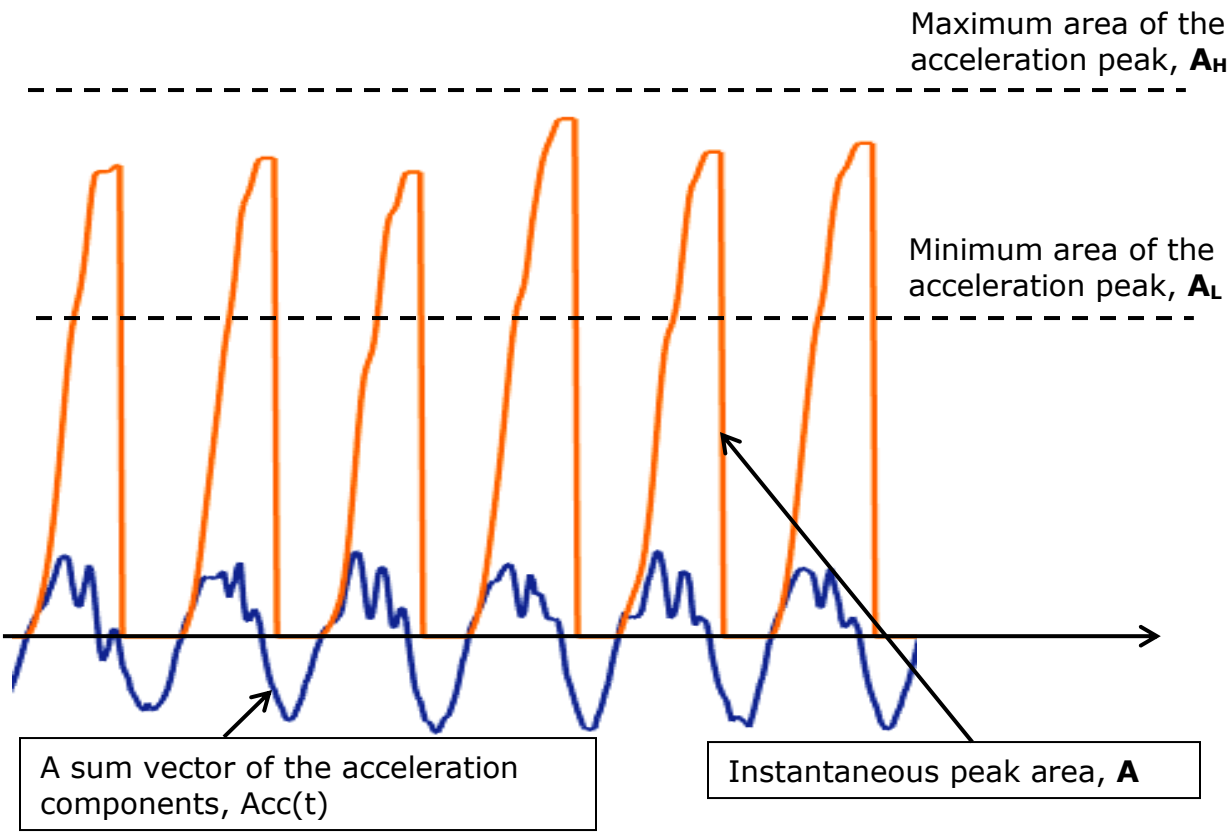

Figure 2: Acceleration sum vector and integration of the peak area

Generally, the step detection principle is based on thresholding the instantaneous peak area of acceleration. The peak area should reach a value between certain minimum and maximum limits (**A_L** and **A_H**) to get a candidate step accepted (see Figure 2).

6.1. Signal Conditioning

For signal conditioning, a two phase procedure is employed:

- IIR-type, high-pass filter (first order, $f_c = 0.25$ Hz with $ODR = 100$ Hz) is applied to the sum vector of the acceleration. This will remove the signal offset (drift) and enables easy thresholding of the instantaneous peak area.
- Canceling high-frequency disturbances in the sum vector of the acceleration is done by a FIR-type (moving average) low-pass filter (length 12 taps, $f_c = 3.70$ Hz with $ODR = 100$ Hz).

As a result, we get a band-pass filtered signal comprised of the most valuable information for step detection.

6.2. Algorithm Parameters

The key algorithm parameters and their constraints are summarized in the following. Figures 3 - 5 also illustrate the meaning of some fundamental algorithm concepts by graphical means.

Instantaneous peak area of acceleration (**A**):

- **A_L** – Minimum area of the acceleration peak
  register: PED_CNTL5
- **A_H** – Maximum area of the acceleration peak
  register: PED_CNTL4_A_H

Width (time interval) of the acceleration peak (**M**):

- **M_L** – Minimum width (time interval) of the peak
  register: PED_CNTL7
- **M_H** – Maximum width (time interval) of the peak
  register: PED_CNTL6

Time period for detecting a step (**T**):

- **T_P** – Minimum time for a single stride
  register: PED_CNTL10
- **T_M** – Time interval to prevent overflow
  register: PED_CNTL9

A parameter to allow a successful start for step detection:

- **step_thr** – Threshold for discarding counting if not enough steps coming
  register: PED_CNTL1_STP_TH

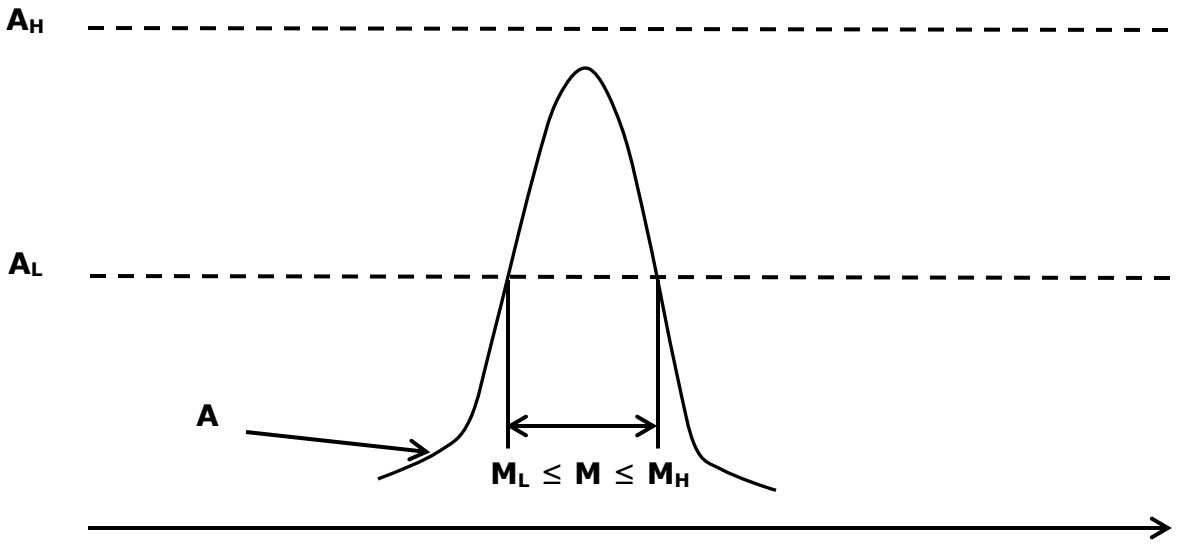

Figure 3: The amplitude of the instantaneous peak area of acceleration (**A**) should stay within $A_L \leq A \leq A_H$, meanwhile its duration (width) should fulfill the condition $M_L \leq M \leq M_H$

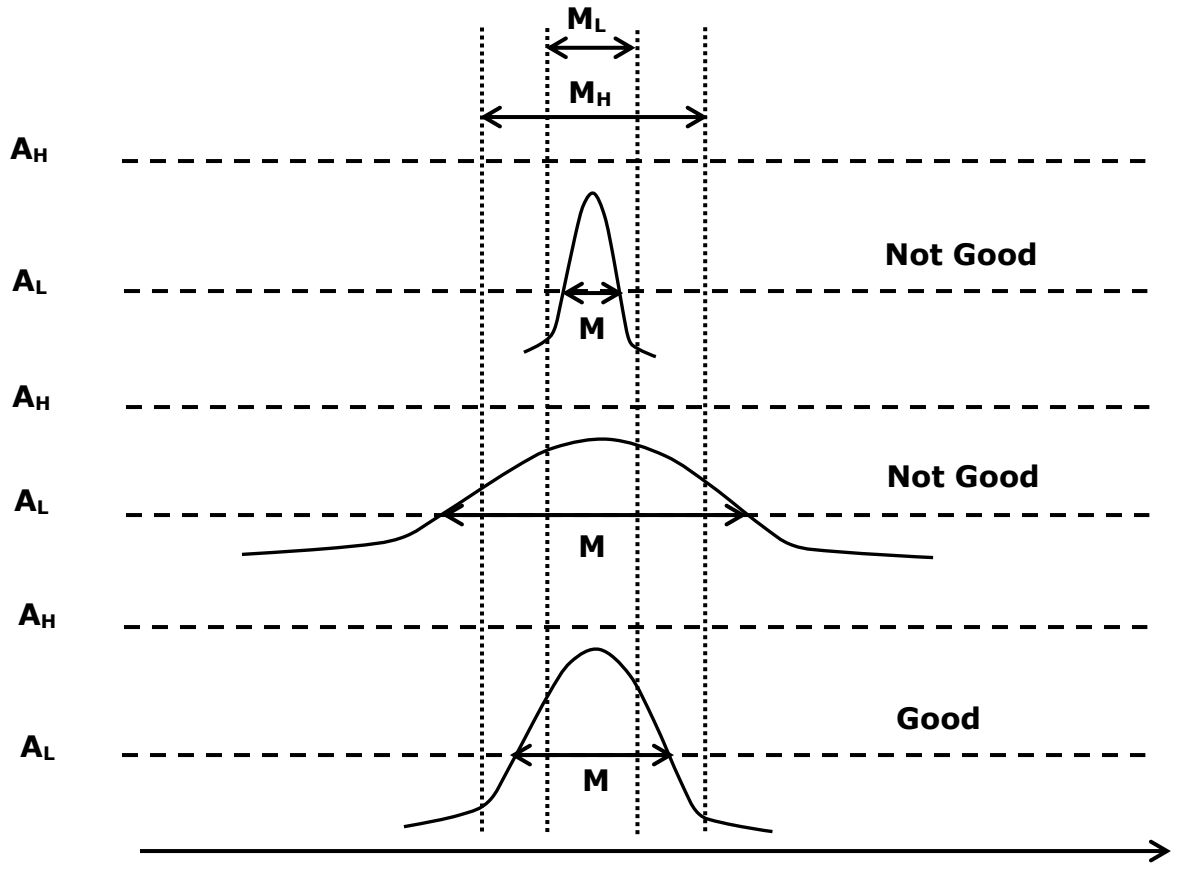

Figure 4: Three options for the width (time interval) of the acceleration peak (**M**)

The objective of the parameter **step_thr** is outlined in Figure 5 (**step_thr = 8**).

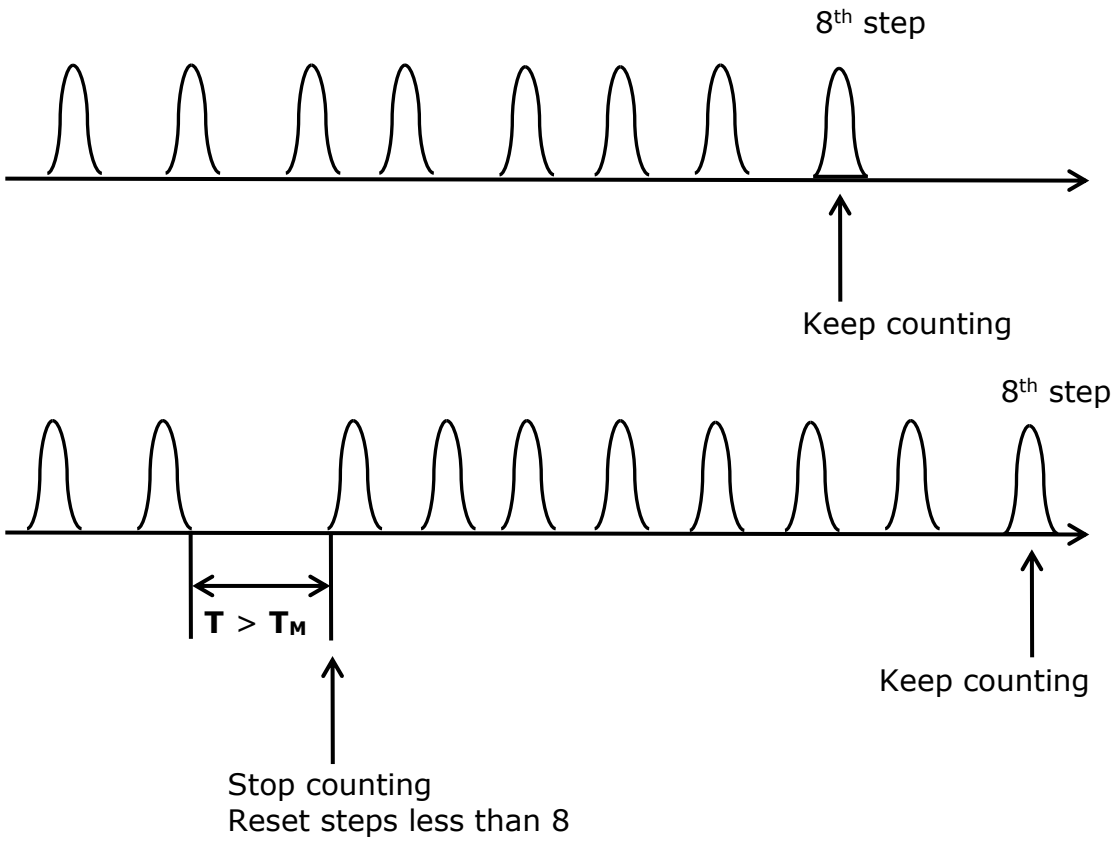

Figure 5: Parameter **step_thr** is discarding counting if too few steps are coming

Please note that a proper operation and accuracy of the pedometer may not be due to a single parameter value but rather often is a result of defining a correct combination of several parameter values. Such a combination may be linked directly to measurement conditions like the location of the accelerometer (wrist, pocket, back bag), specific environment condition (stairs, hardness of the ground), and walking or running style.

7. Timing Requirements

There are several timing requirements that developers should keep in mind when working with the KX116/KX126/KX127.

I²C Clock – The I²C Clock can support Fast Mode up to **400 KHz** and High Speed mode up to **3.4 MHz**.

SPI Clock – The SPI Clock can support up to **10 MHz**.

Enable to Valid Outputs (Start-Up Time) – After the part is enabled (PC1 bit in Control Register 1 is asserted), it takes from **2 ms** to **1300 ms** depending on the ODR and Power Mode setting before the acceleration outputs are valid. (See the relevant Product Specification for details)

Power-Up Time (Time from VDD and IO VDD valid to device boot completion) – After a Power-up, the part takes between **20 ms** to **50 ms** before it is ready for communication.

Software Reset Delay – After a Software Reset, the part takes 2 ms before it is ready for communication.

Standby to Operation Delay – Please allow 1.5/ODR delay time when transitioning from stand-by PC1 = 0 to operating mode PC1 = 1 to allow new settings to load.

8. Interrupt Configuration

There are two (2) available physical interrupts. Each has six (6) possible configurations, based on two states for each of the three customizable variables located in Interrupt Control Register 1 (IEN1, IEA1, IEL1) and Interrupt Control Register 5 (IEN2, IEA2, IEL2):

Enable/Disable (IEN1, IEN2)

- 0 – Disabled – Interrupt conditions will not be reflected on the physical interrupt pin.
- 1 – Enabled – Interrupt conditions will be reflected on the physical pin.

Polarity (IEA1, IEA2)

- 0 – Active Low – The interrupt pin will normally be HIGH, but will transition to LOW when an interrupt is triggered.
- 1 – Active High – The interrupt pin will normally be LOW, but will transition to HIGH when an interrupt is triggered.

Latched/Pulsed (IEL1, IEL2)

- 0 – Latched mode – When an interrupt is triggered, it will remain active on the pin until cleared.
- 1 – Pulse mode – When an interrupt is triggered, it will cause a short (~50 µs) pulse on the pin and clear itself.

For the Pedometer function, interrupt select register INC7 controls the physical interrupt pins INT1 and INT2. This register route the corresponding interrupt to the pins:

STPOVI2[6]	- Step counter overflow interrupt for INT2 pin
STPWMI2[5]	- Step counter watermark interrupt for INT2 pin
STPINCI2[[4]	- Step counter increment interrupt for INT2 pin
STPOVI1[2]	- Step counter overflow interrupt for INT1 pin
STPWMI1[1]	- Step counter watermark interrupt for INT1 pin
STPINCI1[[0]	- Step counter increment interrupt for INT1 pin

By default, all interrupts (noted above) are not routed (disabled).

9. A Few Interrupt Tips

Read the Interrupt Release Register to Clear

In latched mode, the INL register must be read in order to clear the physical interrupt pin. This will also clear the Interrupt Source Registers (INS1, INS2) and the INT bit (0x10) in the Status Register.

Microcontroller/GPIO Interrupt Handling

GPIO configuration is based solely on the connected hardware. The KX116/KX126/KX127 can be configured to issue interrupts depending on how the GPIO is programmed to catch them (if this is not the case, please contact your Kionix Sales Representative). Generally, when an interrupt is triggered, the developer should take the following steps:

1. Disable GPIO interrupt
2. Clear GPIO interrupt and generate desired functionality
3. Enable GPIO interrupt

These steps should be taken without calling any digital communication transactions if done in an interrupt context, because the operating system or kernel will not allow busy-waiting on an I/O operation during an interrupt service routine.

Interrupt Polling

If physical interrupts are not used, a polling mechanism can be devised, which checks the INT bit in the STAT register. This mechanism should then look at INS1 and INS2 registers to determine which engine caused the interrupt and what steps should be taken before clearing the interrupt source information by reading the INL register.

The Kionix Advantage

Kionix technology provides for X, Y, and Z-axis sensing on a single, silicon chip. One accelerometer can be used to enable a variety of simultaneous features including, but not limited to:

Hard Disk Drive protection
Vibration analysis
Tilt screen navigation
Sports modeling
Theft, man-down, accident alarm
Image stability, screen orientation & scrolling
Computer pointer
Navigation, mapping
Game playing
Automatic sleep mode

Theory of Operation

Kionix MEMS linear tri-axis accelerometers function on the principle of differential capacitance. Acceleration causes displacement of a silicon structure resulting in a change in capacitance. A signal-conditioning CMOS technology ASIC detects and transforms changes in capacitance into an analog output voltage, which is proportional to acceleration. These outputs can then be sent to a micro-controller for integration into various applications.

For product summaries, specifications, and schematics, please refer to the Kionix MEMS accelerometer product catalog at:
http://www.kionix.com/parametric/Accelerometers