SQ-SEN-200

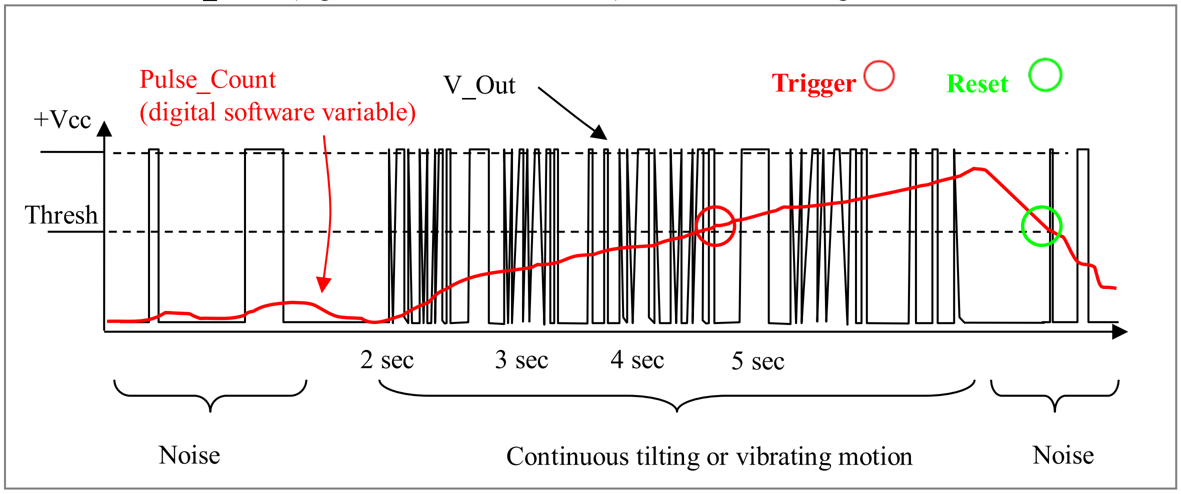
DIGITAL FILTER AND MOTION ESTIMATION ALGORITHM

OVERVIEW

To interpret the pulse train output from the sensor, many customers use a combination of a "software one-shot" and a "software integrator". This is normally done in firmware, but sometimes in hardware if a microcontroller is not being used. These simple, low power algorithms eliminate the effect of sensor to sensor sensitivity differences that naturally occur and enable easy adjustment of triggering sensitivity.
This application note describes a compact algorithm to condition the raw pulse train output of the V_Out (V_Raw or V_Filt signal seen in "SQ-SEN-200 Application Circuits", into a quantitative signal representing the amount of motion experience over a given period of time. This is the preferred method of enhancing the output signal from the SQ-SEN-200 for robust, false-alarm free, movement detection and wake-up functionality. The firmware routines are superior to an analog low pass filter for a variety of reasons.

IMPLEMENTATION BRIEF

- **Circuit:** See application note titled "SQ-SEN-200 Application Circuits".
- **Input**: Set an interrupt or digital input on the microcontroller to edge triggering. Connect this to the sensor's output. If edge triggering is not available you may use the AC Coupling circuit shown in "SQ-SEN-200 Application Circuits". It is also possible to poll the sensor's signal, but there will be a chance of missing pulses if the polling rate is low. Some adjustment to the notes below will be necessary.
- **Software one-shot:** While remaining in a low power mode i.e. running on a 32 KHz watch crystal, look for an interrupt pulse. After the first interrupt, disable further interrupts for ½ second, thereby only allowing one pulse per ½ second time-slice to be recognized. Do not wake up the processor at this time.
- **Software integrator:** If the one-shot is tripped, increment a pulse counter variable by 5. Every ½ seconds, decrement the pulse counter variable by 1 to drain the count value over time. Do not switch to a high power mode yet.
- **Threshold:** If the pulse counter value exceeds a threshold of 25 (5 seconds of motion) then the device is moving. Now, switch to a high power state, alarm etc.
- **Notes:** Be sure to stop incrementing the pulse counter variable at some maximum value (try 50), so it doesn't continue to grow with continuous motion and cause a numerical overflow of the variable. Be sure to stop decrementing the pulse counter at zero so a numerical underflow doesn't occur.
- **OUTPUT EXAMPLE**
- The V_Out signal (from sensor) will look something like the graph below in black.
- The Pulse_Count (digital software filter variable) will look like the signal below in red.



THEORY OF OPERATION

When the one-shot is tripped (interrupt from sensor), a constant value is added to the Pulse_Count variable. At a regular interval, a constant value is subtracted from the Pulse_Count variable thereby "leaking" off some of its value. This ensures that pulse counts do not accumulate over time. After a period without motion, the Pulse_Count will decay to zero.

- The software one-shot disables additional input pulses from being recognized for a given time window after a triggering pulse is received. This prevents a noise signal from contributing a large count value to the Pulse_Count variable as the sensor settles to rest. This also prevents the processor from triggering thousands of interrupts for a given movement signal.

INITIALIZATIONS AND DEFINITIONS

- Pulse_Count - counter accumulator variable
- Trigger" - used to signal a wake-up or alarm state
- Dec_val - constant value to be subtracted at a regular interval
- Inc_val - constant value to be added when the software one-shot is tripped
- Max_Count - used to prevent numerical overflow
- Thresh - threshold for comparison against Pulse_Count
- Initialize Pulse_Count = 0
- Initialize Trigger = False
- **PROGRAM**

```
On interrupt signal
    Disable further interrupts for ½ second         // this is the software one-shot
    If Pulse_Count  <  (Max_Count – Inc_val )       // check for possible overflow
        Pulse_Count = Pulse_Count + Inc_val         // increments Pulse_Count each transition
    Else
        Do nothing                                  // no increment is done to avoid overflow
Do every ½ second
    Re-enable interrupts                            // this is the software one-shot
    If Pulse_Count  > Dec_val                       // check for possible underflow
        Pulse_Count = Pulse_Count – Dec_val         // decrements Pulse_Count when no motion
    Else
        Do nothing                                  // no decrement is done to avoid underflow
    If Pulse_Count  > Thresh
        Trigger = True                              // wake up now, alarm etc.
    Else
        Trigger = False
```