# Improving Performance and Power with Sensor Hubs

Bryon Moyer, Convergence Promotions LLC

Sensors have become a regular feature in many systems. They reside alongside numerous other peripherals to add to the range of environmental inputs that can be incorporated into the job being done by the system. However, in some systems, the number of sensors threatens to overwhelm the other more traditional peripherals, and may influence performance more than the other peripherals.

Nowhere is this more of an issue than with smartphones. From one or two sensors with dedicated functions, mobile platforms have proliferated the number of sensors dramatically and opened up the sensor data to enterprising developers who have dreamed up new things to do. The architects probably never envisioned these applications when they incorporated the sensors.

As a result, the sensors – including touch sensors for the screens – have become more of a burden when directly connected to the application processor (AP). With sensors for ambient light, proximity, acceleration, rotation, and even magnetic field, pressure, humidity, radiation, and chemical sensing in the offing, it can make sense to offload the management of the sensors from the AP.

If the AP needed every piece of sensor data, there might be no choice but to flood the AP with that data. However, sensors, if they are doing their job, simply report what they see. At their most basic, they do not ascribe meaning to that data – that is what the processor does. Sensors have become more intelligent over the last few years, but there is still a limited range of decisions that the sensor can make. After that, some processor must decide what, if anything, to do with the data.

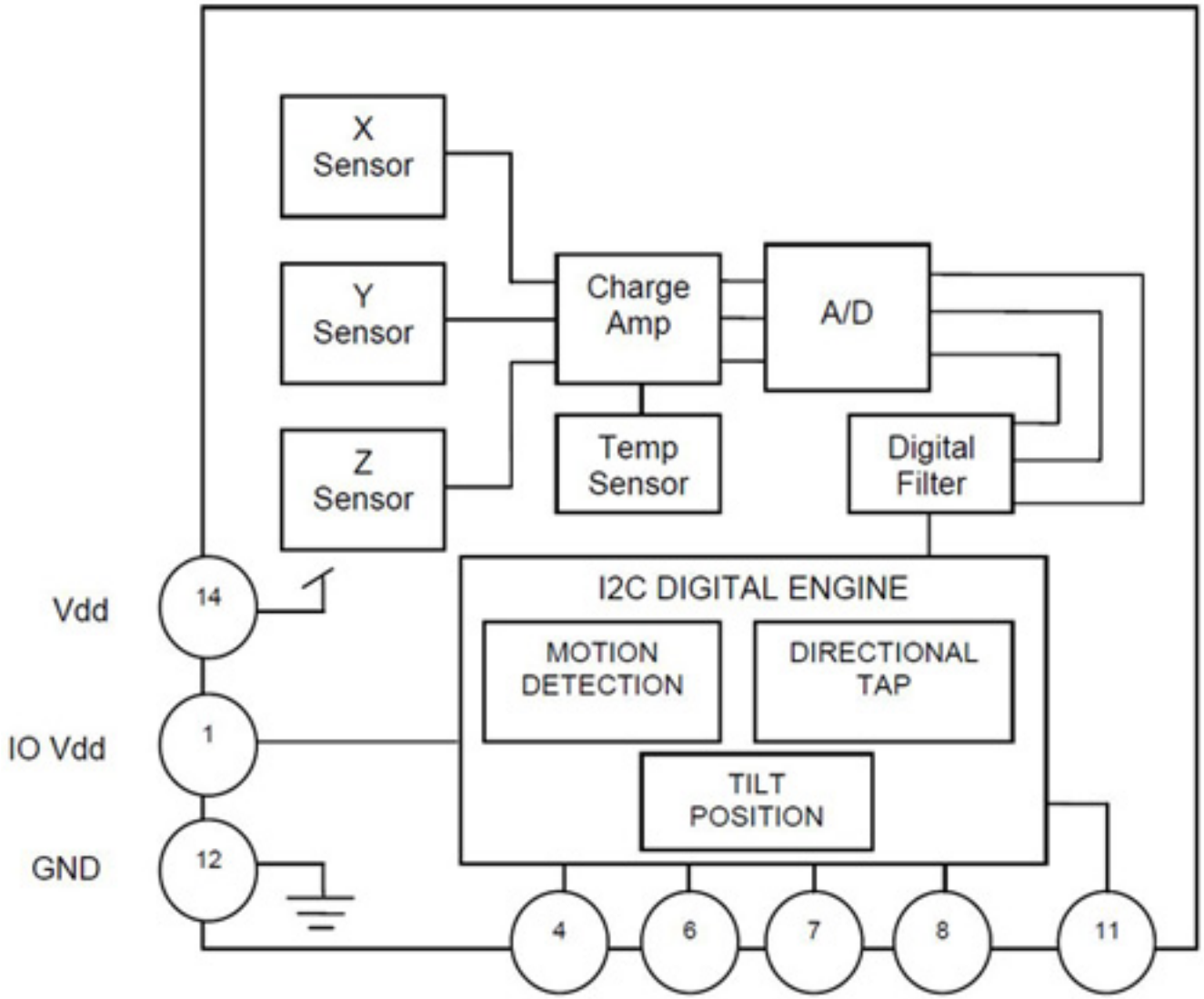
While the AP is ultimately likely to use relevant data, it can be a fair bit of work just to keep track of the data and decide what is relevant. Just as dedicated circuitry handles the radio, video, graphics, and other intensive functions, so a sensor hub can free up processor bandwidth by shielding the AP from sensor events that are likely to be ignored. By allowing the AP to sleep, a sensor hub can also lengthen battery life, even while sensors may still be active.

### Sensor Interfaces

Sensors have a variety of interfaces for communicating their data with the rest of the system. Most use a peripheral bus scheme like SPI or I²C for the delivery of that data, but that is where much of the commonality ends. We will look at a few examples, which can serve to illustrate why a sensor hub may be warranted.

One example is an accelerometer from Kionix [1], the KXTIK-1004 [2], which demonstrates both basic and more complex interactions. It happens to have an I²C

interface, although there are SPI variants. It is a three-axis accelerometer, able to detect accelerations in the x, y, and z directions. It is a physically small device, delivered in a 3 x 3 mm² LGA package. It has a programmable range, allowing a +/- 2, 4, or 8 g range. Its block diagram is shown in **Figure 1**.




Like most sensors, the KXTIK-1004 samples an internal state – in this case, relative capacitances – and makes available a running set of data. This used to be the only way data was delivered on older sensors, and it required constant listening. The problem is that the sensor is fundamentally reporting the current acceleration, and it is up to the listener to decide whether that acceleration has changed enough to be meaningful and to decide what the meaning is.

The KXTIK-1004 still allows the raw data to be accessed. The sensor stores all of the sampled data in 8-bit registers that can be accessed through the I²C port. The data resolution can be 12-bit, which requires the use of two registers, or simply 8-bit, cutting the read time in half; a control setting specifies the resolution. The register value, or “count,” can be translated to an actual acceleration value using **Figure 2**.

| 12-bit Register Data (2's complement) | Equivalent Counts in decimal | Range = +/-2g | Range = +/-4g | Range = +/-8g |
|---|---|---|---|---|
| 0111 1111 1111 | 2047 | +1.999g | +3.998g | +7.996g |
| 0111 1111 1110 | 2046 | +1.998g | +3.996g | +7.992g |
| ... | ... | ... | ... | ... |
| 0000 0000 0001 | 1 | +0.001g | +0.002g | +0.004g |
| 0000 0000 0000 | 0 | 0.000g | 0.000g | 0.000g |
| 1111 1111 1111 | -1 | -0.001g | -0.002g | -0.004g |
| ... | ... | ... | ... | ... |
| 1000 0000 0001 | -2047 | -1.999g | -3.998g | -7.996g |
| 1000 0000 0000 | -2048 | -2.000g | -4.000g | -8.000g |

| 8-bit Register Data (2's complement) | Equivalent Counts in decimal | Range = +/-2g | Range = +/-4g | Range = +/-8g |
|---|---|---|---|---|
| 0111 1111 | 127 | +1.984g | +3.968g | +7.936g |
| 0111 1110 | 126 | +1.968g | +3.936g | +7.872g |
| ... | ... | ... | ... | ... |
| 0000 0001 | 1 | +0.016g | +0.032g | +0.064g |
| 0000 0000 | 0 | 0.000g | 0.000g | 0.000g |
| 1111 1111 | -1 | -0.016g | -0.032g | -0.064g |
| ... | ... | ... | ... | ... |
| 1000 0001 | -127 | -1.984g | -3.968g | -7.936g |
| 1000 0000 | -128 | -2.000g | -4.000g | -8.000g |

Because the acceleration is not read directly, the reader must perform the conversion if an absolute value is required using this method. If this were the only way to access the data, then the processor to which this was connected would need to poll constantly, watching for useful changes in the value.

The polling frequency matters as well. The register values are updated on a regular basis. If the polling rate is too slow, then you will only sample occasional readings, missing interim ones. With this particular sensor, you can set the output data rate (ODR) to match more closely the polling rate. Timing could, in theory, be an issue when using 12-bit resolution, which requires the reading of two registers: you do not want the data updated after you have read one register but before reading the second register. Fortunately, in this case, reads are protected to ensure that both registers read reflect the same data sample.

The need to poll has been alleviated by interrupt circuitry that not only can inform the processor of when new data is available, but can also identify several common types of acceleration event. The burden on the processor is reduced, not only because it does not have to calculate the events, but also because, when the event occurs, an interrupt is sent. This means that, if these events are all that matter, polling is not required.

The KXTIK-1004 can identify four general events: motion, a tap, a change in tilt, and

a “watermark” event (more on that later). There is a single interrupt pin, pin 11 (INT). The pin can be programmed to fire due to any of these events. When the interrupt fires, there are status registers that can be read to determine what event occurred and gather any relevant data.

Let us look at “tilt” as an example. This feature creates an interrupt that can instruct the system to rotate the screen on a phone. There are a number of parameters that can be set in the control registers. These include the tilt ODR (which can be different from the ODR for other functions), the length of time the tilt is in place before deciding that the phone is in a new state, hysteresis, and the point at which it decides the phone is “flat” (i.e., there is no way to tell which way to rotate the screen). Given these parameters, when the sensor detects that the phone has entered a new tilt state, it can fire an interrupt.

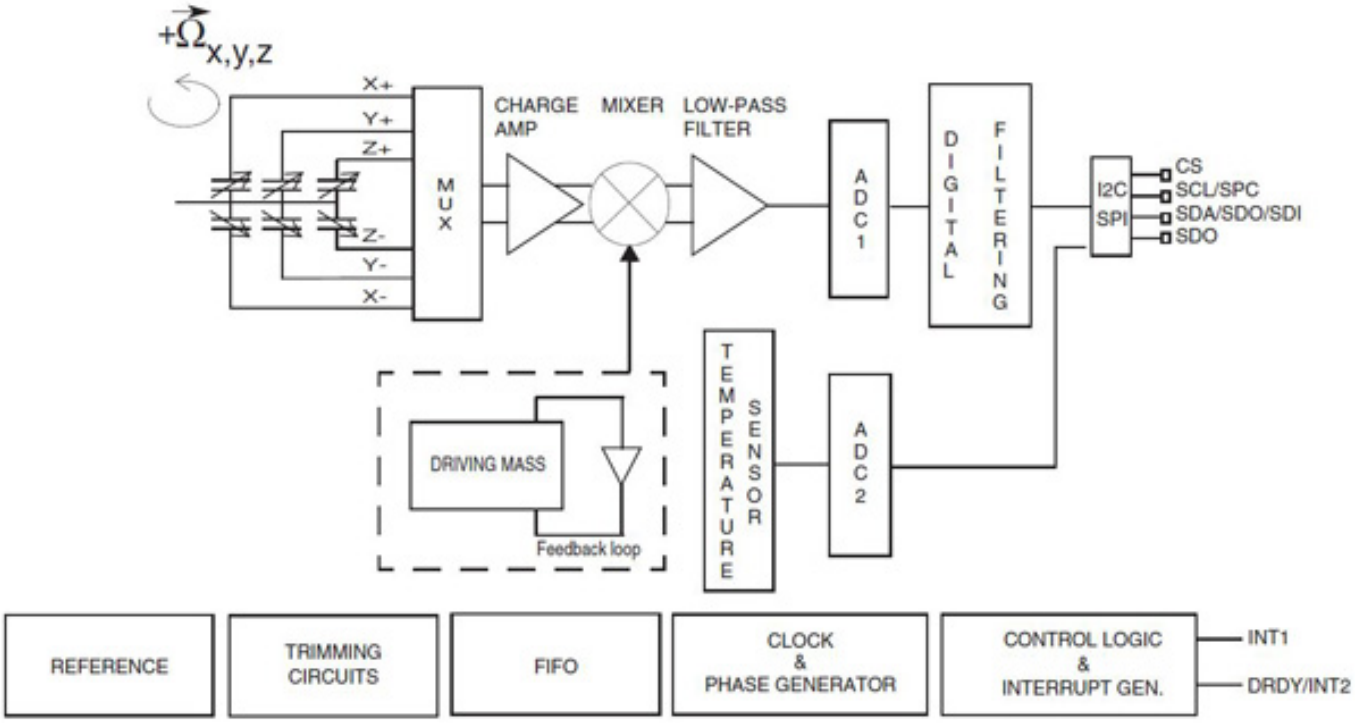
When the interrupt occurs, the processor can read two registers, one of which provides the prior tilt position, the other providing the new tilt position. The tilt position is expressed as a state, and each of the registers has six bits that identify the appropriate state, as shown in **Figure 3**.

| Bit | Description |
|---|---|
| LE | Left State (X-) |
| RI | Right State (X+) |
| DO | Down State (Y-) |
| UP | Up State (Y+) |
| FD | Face-Down State (Z-) |
| FU | Face-Up State (Z+) |

Likewise, single and double taps can be identified, as can motion events to wake the unit up, all of which have a variety of parameters that can be set. Data buffering can also be handled by a FIFO that can be used in FIFO mode (once full, no new data is written – FILO is also available) or streaming mode (when full, the oldest data is overwritten by new data). “Watermark” levels can be defined indicating when a particular fill level has been reached; an interrupt can be fired when the watermark is hit.

In this sensor, there are numerous options for reducing the amount of work that the processor has to do – and yet work remains to manage the interrupt events and sort through their sources so that appropriate action can be taken.

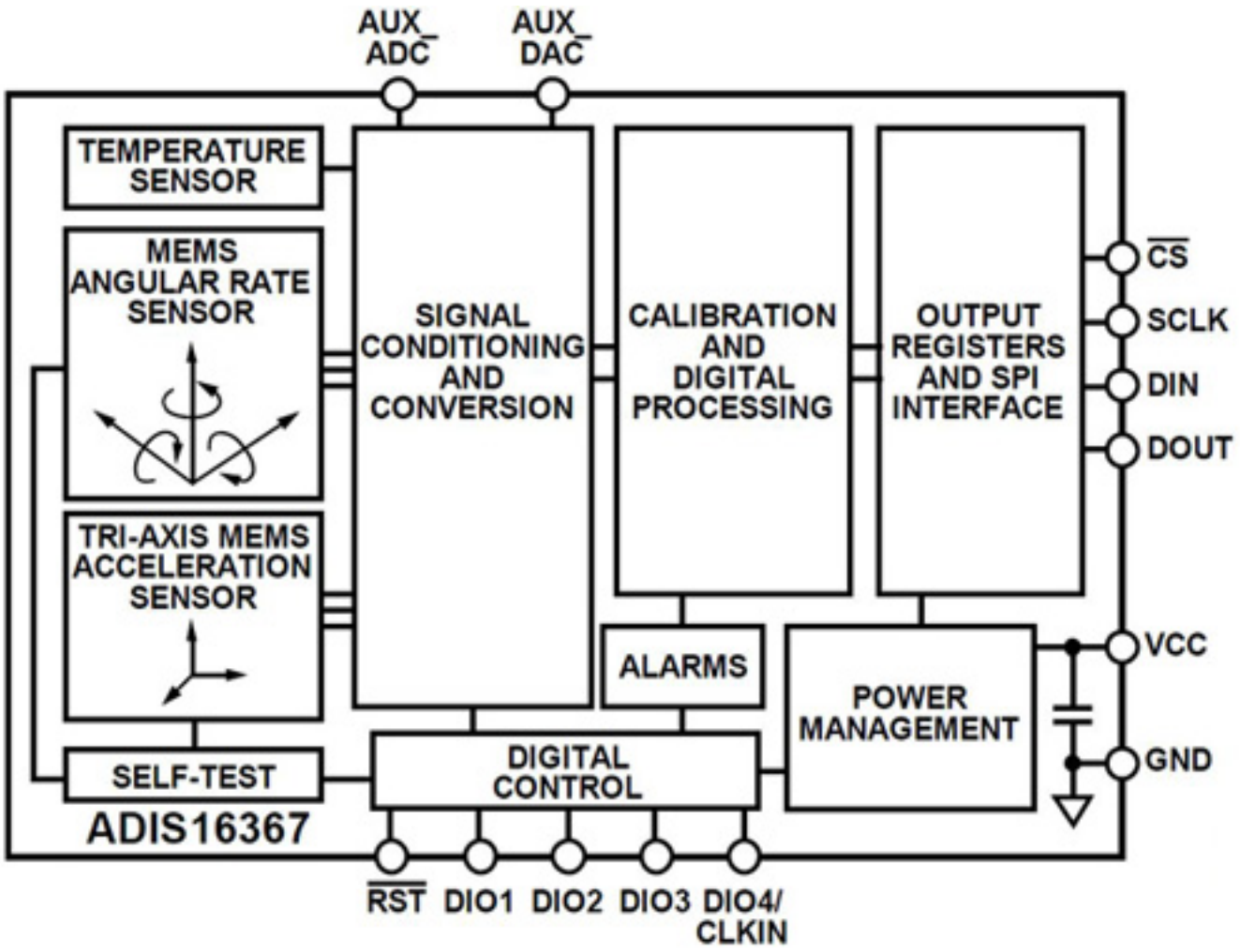
Another sensor example is STMicroelectronics’ [3] L3G4200D [4] gyroscope (see **Figure 4**). It allows communication over I²C or an SPI interface. It also has a basic DRY (data ready) interrupt that signals a new sample ready for reading, and it has a FIFO that can operate in FIFO or streaming mode.




Whether simply reading data or using the FIFO, the read rate is important. If you read too slowly, you might miss samples or you might read X data from one sample, then Y data from the next sample, and then Z data from yet a third sample, meaning you would never have all of the information for a single sample. (A BDU feature allows you to read both halves of the sample data before it will be overwritten.) When using the FIFO in FIFO mode, reading too slowly will cause blocking; in streaming mode, it will cause overflow.

That puts the burden on the processor to keep paying attention to the gyro when using these modes. However, the device does provide interrupts for watermark, FIFO overflow, movement (for wake-up), no-move, and selective axis activity. These interrupts reduce some of that reading work.

Some of the work of monitoring multiple sensors can be alleviated with “combo” sensors. For example, Analog Devices’ [5] ADIS16367 [6] (see **Figure 5**) combines an accelerometer and a gyroscope. It has a signal that can be used as a data-ready interrupt, so the data read over the SPI interface can provide samples from both sensors.




The final example is Atmel's [6] AT42QT1110 [7] touch controller. This device works with a touchpad to signal when the capacitance of a particular key has changed, signaling a touch. It can work with 7- or 11-key pads, and it can communicate over SPI.

In 7-key mode, it has a pin for each key to indicate touches. In 11-key mode, reading is done periodically either using an internal clock or by providing an external SYNC signal (reducing the number f pads detectable to 10). The device also has a CHANGE pin that signals when a key has changed (either being touched or released). This can act as an interrupt, reducing the number of reads to coincide with actual touch or release events.

**Managing Sensor Data Acquisition and Interrupts**

In all of these cases, there are options for reading the raw data directly or for responding to any of a number of interrupts. If the interrupts provide exactly the events of interest, and if those events need to be processed by the AP, then perhaps a direct sensor-to-AP connection is warranted. However, there are several reasons why such a direct connection might not be advised.

- If raw data is required, the timing needed for reading may require too much attention from the AP.
- Even with interrupts, some sensors – especially touch-screens that allow swipes and multi-touch – can generate a huge number of interrupts.
- If additional processing of data is required for further event qualification or fusion, then this becomes a burden on the AP.
- Some lower-level responses to an event – say the rotation of the screen – might not need the AP to complete, meaning that some lower-level entity could capture and execute that event without involving the AP.
- Finally, and perhaps most importantly, the AP consumes a significant amount of power, it is important that the AP sleep when it is not needed. If the AP bears sole responsibility for managing the sensors, it cannot be put to sleep. (In fact, sensor interrupts may be used to wake the AP.)

For these reasons, there is an increasing tendency to put a device between the sensors and the AP; such a device is typically called a “sensor hub.” Its duties are to maintain the low-level dialog with all of the sensors, monitor events, execute those events within its lower-level purview, handle low-level fusion, and pass along to the AP only those events that are both relevant and necessary uses of the AP’s attention.

One of two kinds of devices can take on the role of sensor hub. FPGAs make logical sense, but only very small ones have the cost structure necessary to be considered for an application like a mobile device. The other choice, which is more typical, is a microcontroller. The critical resources for monitoring sensors boil down to the availability of I²C or SPI, and how the microcontroller handles external interrupts.

For example, Atmel’s ATmega48A [8] is an 8-bit RISC microcontroller with 4 K bytes of FLASH, 256 K bytes of EEPROM, and 512 K bytes of RAM; it can communicate over both I²C and SPI. This device has two types of interrupts: “pin-change” interrupts and “external” interrupts. The former allow any of 24 pins to trigger an interrupt via three interrupt flags (with each flag corresponding to a different interrupt vector). Each of the flags is associated with eight of the 24 pins.

There are also two level- or edge-triggered external interrupt pins called INT0 and INT1. Each of these has its own interrupt vector. This yields a total of 26 pins with five different vectoring destinations, leaving room for numerous sensors and other sources of possible external interrupt.

Microchip’s [9] PIC18F2520 [10], a 16-bit microcontroller with 32 K of FLASH, 1536 bytes of SRAM, and 256 bytes of EEPROM, can also communicate over both interface types. It has three edge-sensitive external interrupt pins with programmable priority that can be high or low for INT1 and INT2 (priority is fixed at high for INT0). In addition, pins 7:4 of PORTB can be used as pin-change interrupts; their priority can also be programmed as high or low.

Finally, Freescale Semiconductor’s [11] MCF51QE32 [12], a 32-bit ColdFire microcontroller with 32 K bytes of FLASH and 8 K bytes of RAM, has two SPI and two I²C ports. It has a single IRQ external interrupt pin and an elaborate priority scheme.

There are seven “levels” of interrupt, and each one has seven levels of priority. The various interrupts (mostly internally generated) occupy different positions in the resulting matrix (see **Figure 6**). The external interrupt pin operates at the highest level, midpoint priority. This single-pin scheme may work well for a multi-sensor combination having a single interrupt for all of the sensor data updates.

| Level | Priority within Level | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | | 6 | | 5 | | 4 | | Midpoint | | 3 | | 2 | | 1 | | 0 | |
| 7 | — | | — | | — | | — | | IRQ_pin | | Low_voltage | | — | | — | | force_lvl7 | |
| | | | | | | | | | 0 | 64 | 1 | 65 | | | | | FRC[32] | 96 |
| 6 | remapped | | remapped | | TPM1_ch0 | | TPM1_ch1 | | — | | TPM1_ch2 | | — | | TPM1_ovfl | | force_lvl6 | |
| | PL6P7 | * | PL6P6 | * | 2 | 66 | 3 | 67 | | | 4 | 68 | | | 5 | 69 | FRC[33] | 97 |
| 5 | TPM2_ch0 | | TPM2_ch1 | | TPM2_ch12 | | — | | — | | — | | — | | TPM2_ovfl | | force_lvl5 | |
| | 6 | 70 | 7 | 71 | 8 | 72 | | | | | | | | | 9 | 73 | FRC[34] | 98 |
| 4 | SPI2 | | SPI1 | | SCI1_err | | SCI1_rx | | — | | SCI1_tx | | — | | — | | force_lvl4 | |
| | 10 | 74 | 11 | 75 | 12 | 76 | 13 | 77 | | | 14 | 78 | | | | | FRC[35] | 99 |
| 3 | IICx[1] | | KBIx[2] | | ADC | | ACMPx[3] | | — | | — | | — | | — | | force_lvl3 | |
| | 15 | 79 | 16 | 80 | 17 | 81 | 18 | 82 | | | | | | | | | FRC[36] | 100 |
| 2 | — | | — | | SCI2_err | | SCI2_rx | | — | | SCI2_tx | | — | | RTC | | force_lvl2 | |
| | | | | | 19 | 83 | 20 | 84 | | | 21 | 85 | | | 22 | 86 | FRC[37] | 101 |
| 1 | TPM3_ch0 | | TPM3_ch1 | | TPM3_ch2 | | TPM3_ch3 | | — | | TPM3_ch4 | | TPM3_ch5 | | TPM3_ovfl | | force_lvl1 | |
| | 32 | 87 | 24 | 88 | 25 | 89 | 26 | 90 | | | 27 | 91 | 28 | 92 | 29 | 93 | FRC[38] | 102 |

[1] There are two $I^2C$ modules on-chip. They share a common interrupt vector.
[2] The keyboard features are available on GPIO ports B and D. The two modules share a common interrupt vector.
[3] There are two analog comparator modules on-chip. They share a common interrupt vector.

## Summary

With the number of sensors being integrated into a wide variety of embedded systems, the application processor becomes a poor choice for managing them both because of the cycles required and that doing so means that the AP cannot sleep, removing a key opportunity for power savings. Using a small, lower-power microcontroller to aggregate the sensor signals allows the main processor either to focus on things that are more important or to sleep, being interrupted only when there is something important that it should address.

The examples shown in this article are but a few of the many sensors and microcontrollers available from Digi-Key. By combining your sensor requirements with the other system duties that your microcontroller may perform, over and above being a sensor hub, you have numerous options for meeting your function, performance, and power requirements.

Further Reading:

The Critical Need for Accuracy: Getting it Right with Today’s (and Tomorrow’s) Sensors [13], Carolyn Mathas
Combining the Input of Multiple Sensors to Produce Better Overall Results [14], Carolyn Mathas

Inertial Sensor Overview-Performance, Size, and Power Advantages [15], Carolyn Mathas
The Five Senses of Sensors—Touch [16], Carolyn Mathas
Advances in Accelerometers Broaden User Choices [17], Richard Comerford
Using Integrated Sensors to Simultaneously Detect Multiple Parameters [18], Richard Comerford

**Source URL (retrieved on *03/15/2014 - 5:09am*):**
http://www.wirelessdesignmag.com/articles/2013/03/improving-performance-and-power-sensor-hubs?qt-blogs=0

**Links:**
[1] http://www.digikey.com/Suppliers/us/kionix.page?lang=en
[2] http://www.digikey.com/product-detail/en/KXTIK-1004/1191-1003-1-ND/3137349?cur=USD
[3] http://www.digikey.com/us/en/techzone/sensors/supplier/STMicroelectronics__497.html
[4] http://www.digikey.com/product-detail/en/L3G4200DTR/497-11071-1-ND/2587903?cur=USD
[5] http://www.digikey.com/us/en/techzone/sensors/supplier/Analog_Devices_Inc__505.html
[6] http://www.digikey.com/product-detail/en/ADIS16367BMLZ/ADIS16367BMLZ-ND/2272223?cur=USD
[7] http://www.digikey.com/product-detail/en/AT42QT1110-MU%20QS470/AT42QT1110-MU%20QS470CT-ND/2346840?cur=USD
[8] http://www.digikey.com/product-detail/en/ATMEGA48A-AU/ATMEGA48A-AU-ND/2271038?cur=USD
[9] http://www.digikey.com/us/en/techzone/sensors/supplier/Microchip_Technology__150.html
[10] http://www.digikey.com/product-detail/en/PIC18F2520-I%2FSO/PIC18F2520-I%2FSO-ND/613219?cur=USD
[11] http://www.digikey.com/us/en/techzone/sensors/supplier/Freescale_Semiconductor__375.html
[12] http://www.digikey.com/product-detail/en/MCF51QE32CLH/MCF51QE32CLH-ND/1791396?cur=USD
[13] http://www.digikey.com/us/en/techzone/sensors/resources/articles/the-critical-need-for-accuracy.html
[14] http://www.digikey.com/us/en/techzone/sensors/resources/articles/combining-the-input-of-multiple-sensors.html
[15] http://www.wirelessdesignmag.com/Inertial%2520Sensor%2520Overview-Performance%2C%2520Size%2C%2520and%2520Power%2520Advantages
[16] http://www.digikey.com/us/en/techzone/sensors/resources/articles/The-Five-Senses-of-SensorsTouch.html
[17] http://www.digikey.com/us/en/techzone/sensors/resources/articles/advances-in-accelerometers.html
[18] http://www.wirelessdesignmag.com/Using%2520Integrated%2520Sensors%2520to%2520Simultaneously%2520Detect%2520Multiple%2520Parameters