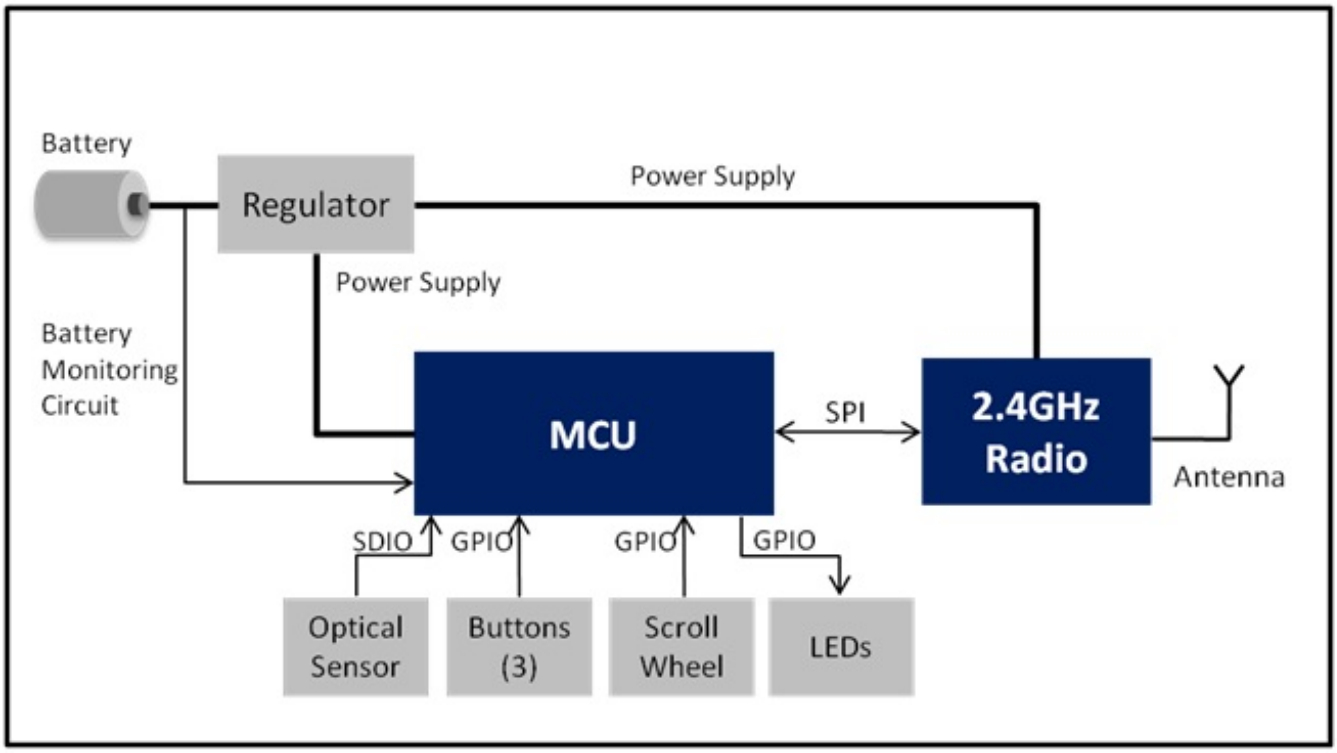
Exclusive: Wireless HID with Touch Sensing – Pros and Cons

Sai Prashanth C, Senior Applications Engineer, Cypress Semiconductor

For many applications, the user experience is the most distinguishing factor in the success of a product. Apple has successfully made the term “user experience” synonymous with touch interfaces in the smart phone segment. With many consumer electronics giants replacing their mechanical actions (buttons, joysticks, scroll bars) with capacitive touch buttons and touch screens, consumers have increased expectations from the wireless human interface device (HID) industry as well. While a touch mouse or touch keyboard may promise to bring about a change in the monotonous features of the normal keyboard and mice, users may find it difficult to switch from legacy gadgets to these more sophisticated ones. This article examines the major system requirements and the implementation details of touch based wireless HID devices and their impact on the user experience.

Legacy Mouse:

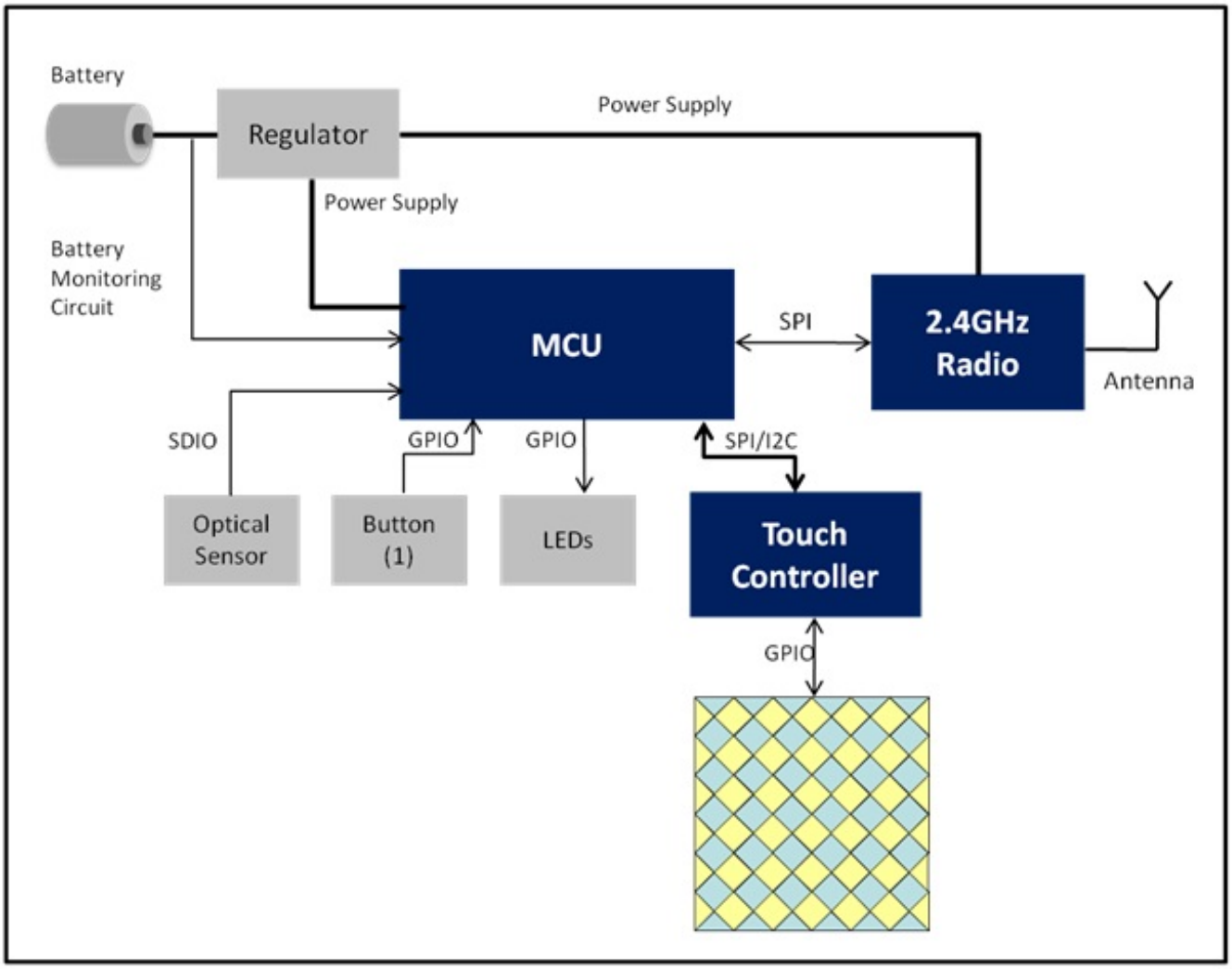
A traditional mouse has several mechanical buttons (typically three – Left Click, Right Click, Middle Click) and a scroll wheel for scrolling the page. It also has an optical sensor for facilitating pointer movement on the screen corresponding to physical mouse movement.



Typically, the MCU controls the 2.4 GHz Radio (1 or 2 Mbps on air data rate), optical sensor through a common interface (SPI, I2C etc), and control LEDs, buttons, and scroll wheel through GPIOs. Some controllers also have the radio integrated inside

the MCU for further integration.

Wireless Touch Mouse:
As the name suggests, a wireless touch mouse has a touch pad on the surface to facilitate gestures and one mechanical button underneath the touch surface to facilitate clicks. The optical sensor is retained from the traditional mouse.



Typically, the main MCU communicates to the 2.4 GHz radio operating at 1 or 2 Mbps on air data rate, touch controller, optical sensor through a standard interface (SPI, I2C, etc) and controls like LEDs and buttons through GPIOs. Today's programmable system-on-chips, such as Cypress PRoC UI (CYRF89535), integrate touch, RF, and MCU functionality on the same device, reduce three ICs to one with corresponding cost reductions.

At a minimum, a touch mouse needs to provide all of the services of a traditional mouse like Left Click, Right Click, Middle Click, and Scroll Wheel. In a wireless touch mouse, the buttons and scroll wheel are replaced by a capacitive touch sensitive area.

Implementing a Click: With a traditional mouse, there are three mechanical buttons placed under the casing. The user places his or her fingers on the casing and presses the buttons to perform the corresponding button click action. Users could also press all the three buttons simultaneously, although there is a function

associated with this action.

Implementing a click on touch mouse is more challenging, as three buttons are not available under the plastic. One can think of adding three buttons for a touch mouse as well and make it work like a traditional mouse. There are three reasons for not choosing to go ahead with three buttons:

- Building a mechanical enclosure to accommodate three buttons and touch pad is challenging.
- Splitting the touch portion of the mouse to make it look like traditional mouse will affect the performance of touch gestures.
- One can leverage the fact that the touch controller can detect the position of the finger/fingers on the touch pad.

Considering these reasons, major touch mouse vendors support only one mechanical button under the casing. When the button is pressed, the respective clicks are processed based on the finger position.

Left Click: Left Click is processed if the finger is placed on the left side of the touch pad and the button is also pressed under the casing.

Right Click: Right Click is processed if the finger is placed on the right side of the touch pad and the button is also pressed under the casing.

Middle Click: Middle click is used for shortcuts like “Open Links in New Tabs in Explorer,” “Close browser Tabs,” etc. in certain applications. There is no global mapping for the middle click. A PC host application can map the user input (middle click) to any operation as a shortcut. For a mouse where the scroll wheel is absent, an application (word, pdf document, webpage, etc.) could be scrolled up/down by pressing middle click and moving the mouse up/down. Typically, there is not much use of the middle click by the average user. Hence vendors can choose to omit this functionality or come up with some specific way to press the mechanical button and report it as middle click. Below is one certain example where both the fingers are placed on the left region and the mechanical button is pressed.

Challenges:

1) What shall happen when both the fingers (forefinger on the left region and the middle finger on the right region) are present and the button is pressed?

The solution to the problem could be finding the pressure of the finger to resolve the intention of the user - whether the press was for left click or right click. However, most vendors chose to assign this scenario to be reported as Left click. To trigger a right click, the user has to place only one finger on the right region and press the button.

This is a deviation from the traditional mouse usage case. The user needs to follow a learning curve to get used to this mouse.

2) What happens if a user moves the finger on the touch pad from the left region to right region without releasing the mechanical button press? Does the system report Left Click or Right Click or a combination of both?

To resolve this ambiguity, vendors chose to send the same click information until the finger is lifted off from the touch pad. Hence with this approach, when the mouse is operated as left click and finger is moved from the left region to the right region of the touch pad without releasing the button press, the system will still report a left click.

Implementing Scroll: With a traditioanl mouse, application scrolling is implemented through a mechanical zWheel. When moved, the zWheel gives a tick that corresponds to one scroll moment. This zWheel is interfaced to the MCU, which reports the scroll wheel data as +1 or -1 depending if the movement is up or down. The zWheel is operated typically with one finger, usually the forefinger.

Implementing scrolling with a touch mouse is achieved with the use of touch gestures. When the finger/fingers are moved up or down, the corresponding action can be converted into scroll data. The touch engine inside the touch controller will have built-in libraries which can detect gestures when finger/fingers are moved up or down as a count of distance.

Challenges:

Scope of enhancements for features:

Distance based scroll: Scroll wheel report data can vary from 0 to 128 for scroll up and -1 to -127 for scroll down. The higher the value reported, the longer the distance scrolled on a page. Users prefer to have the page scroll longer when they move the finger/fingers for longer distances on the touch surface. In the case of a traditional mouse, this feature cannot be implemented as it is driven with a mechanical zWheel.

Inertial scroll: With a touch mouse, users can utilize a feature where they flick vertically and the page scrolls for a longer time and longer distance. This feature makes it easer for users who want to scroll many pages at a time.

Implementing WIN8 gestures: Microsoft introduced the new Metro UI with its Windows 8 operating system. Due to the revolution of touch, the UI has introduced WIN8-specific gestures that makes it more comfortable to use the OS. Below are the gestures that are commonly incorporated.

With a traditional mouse, to switch to the previous application, users often try to use hot keys like ALT + TAB on the keyboard. With touch-enabled devices, Microsoft came up with specific gesture on touchpad to perform a similar action. When the finger is moved from the left edge to center in WIN8 OS, the "Switch to the last Application" action is triggered. This action is triggered basically by sending a hot Key code which the OS can understand; in this case, the Windows logokey + Ctrl +

Backspace.

WIN8 has OS-specific tool bar called “Charms Bars” that pop up when the finger is moved from the right edge to center. Again, this results in a hot key code the OS can understand: Windows logo key + C.

WIN8 has OS-specific tool bars called “Command Menus” that pop up when the finger is moved from the top edge to center. The hot key code for this is the Windows logo key + Z.
All these features provide greater efficiency to users using a touch mouse with WIN8. In other OSes, these hot keys are not recognized and are of no use unless the underlying driver supports them.

Implementing miscellaneous gestures:

Touch mice can be used to implement miscellaneous gestures like Zoom, Rotate, Gesture to revoke, minimize all applications (WIN Key + D), Close Application, etc. Implementation of these features is accomplished in three steps.

- Encode a specific gesture on the touch pad.
- Assign that specific gesture to one of the features.
- Send the HID shortcut key to the PC Host.

Impact of OS: All the above gestures are applicable to Windows OS. However, when the OS is changed, the short cut keys are not valid and hence the gestures shall not be detected by the PC. Alternatively, vendors chose to build their own PC driver that understands these gestures and actuates them on the respective PC applications.

These gestures are easy to access for users to perform an action, but might take some learning to use them on a daily basis. Also, these gestures might be falsely triggered because they are easily accessible. These false triggers would be frustrating if the user is doing some critical actions on the PC.

Source URL (retrieved on *04/19/2014 - 12:54pm*):
http://www.wirelessdesignmag.com/articles/2013/11/exclusive-wireless-hid-touch-sensing-%E2%80%93-pros-and-cons