# The OrbitWatch

***An Energy-Efficient Bluetooth Wrist Interface based on Motion Analysis***

**Narendra Anand Ahmad Rahmati Lin Zhong**
**nanand@rice.edu rahmati@rice.edu lzhong@rice.edu**
**Efficient Computing Group**
**Department of Electrical & Computer Engineering**
**Rice University, Houston, TX 77005**

## 1. Introduction

Mobile phones have provided people ubiquitous connectivity with the world and access to a vast array of rich and dynamic information. However, our phones are not usually readily accessible, staying in our pocket or bags. Wristwatches represent a form factor that is much more accessible, especially for daily information. Digital system designers have attempted to explore the accessibility of watches in the past, without much success. The creations have ranged from calculators, cameras, GPS receivers, and even PDA's, all built into a wristwatch; however, these devices failed to gain main stream popularity because of the difficulty of operating such a wide array of functions from such a small form factor. Their extra peripherals also increased the cost and deprived them of a decent battery life. The sheer number of functions with respect to the small viewing area also contributes to the lack of aesthetic appeal for these devices, which further diminishes their social acceptance. To solve this problem, we have designed and prototyped the OrbitWatch, a TI-MSP430 based Bluetooth wrist interface that avoids the problems of previous designs while still being feature-rich and supporting a variety of applications. The OrbitWatch functions as a secondary user interface for a mobile phone. It leverages the computing capacity of the mobile phone through Bluetooth and provides a friendly user interface through motions sensors and tactile buttons.

The rest of the paper is organized as follows. In Section 2, we present the design of the OrbitWatch. In Section 3, we describe our OrbitWatch prototype and its user interface design. In Section 4, we provide details for the OrbitWatch software and hardware design. We conclude by discussing future possibilities in Section 5.

## 2. Proposed Design

Previous multi-function wristwatch designs have attempted to pack too many features onto the wrist, thus creating confusing, unsightly devices that are unable to permeate all walks of life. Instead, the OrbitWatch is a wrist interface with a regular watch appearance that provides access to information without any superfluous, power-hungry features.

### 2.1. Function

Our proposal is to leverage the widespread use of mobile phones, which is how the OrbitWatch can be so "feature rich" with only a TI-MSP430 microcontroller. The MSP430 controls the LCD and the Bluetooth transceiver, displaying pertinent information retrieved from a Bluetooth enabled mobile phone in the user's pocket while still providing a battery life of several days on a coin-sized Lithium Ion cell. Its clean and natural user interface ensures a simple learning curve in order to help promote social acceptance. In addition, the design is based on commercial off-the-shelf components, such as the inexpensive MSP430 microcontroller, which significantly reduces the design cost.

### 2.2. Sensor-based user interfaces

Our design incorporates different ultra low-power sensors into the OrbitWatch system. A three-axis accelerometer enriches the user interface by allowing for motion based operation of the OrbitWatch. For example, the user can avoid interrupting their current activity and navigate through messages and alerts using simple hands-free wrist motions. A temperature sensor on the watch is an additional tool for data collection for the application designer. Again, all of these peripherals are controlled by the MSP430 in order to ensure efficient operation.

### 2.3. Aesthetic

The simplicity of the OrbitWatch design allows for incorporation into a regular digital or analog, watch without much alteration to the polished watch appearance. Such a watch would have one or two lines of embedded text display in the watch face, providing the user with information without appearing "geeky." We believe this is extremely important for any "information" watches to enjoy wide social acceptance. Figure 1 illustrates this possible design.


**Figure 1: Information Watch based on the OrbitWatch**

### 2.4. Potential Applications

The OrbitWatch allows for an application designer to access any text based information that exists on a Bluetooth mobile device. Such a design basically allows for you to have an RSS feed of your life on your wrist. Currently, people use their PDA's to access stock quotes, weather updates, sports scores and other information that now may be easily scrolled across the wrist. The watch can also display notifications and alerts that cell phones commonly produce, such as text messages, Caller ID, or alarms, without the user having to remove the

device from the pocket. Why is such an immediate and discreet method of information access useful? The following examples will illustrate.

A sick patient, whose life is run by a vast array of medications, may now receive discreet reminders when to take the next dose. The advantage of the Orbit Platform over a regular watch alarm is that the schedule is edited and stored on the PDA, which makes it much easier to create and maintain than directly entering data into the watch. In addition, the patient's physician can provide an electronic copy of the dosage schedule that can be easily downloaded to the PDA while in the doctor's office.

A businessman, whose life is run by his Blackberry, is in a meeting. He is completely connected with the world because he constantly receives phone calls, text messages, emails, and daily schedule reminders. During the meeting, however, his connection is severed. What polite way is there to fumble with a PDA when a colleague is halfway through a presentation? With the Orbit Platform, text messages, email subjects, reminders for the next event in his workday, and even Caller ID information can be displayed on his wrist. The question as to whether or not that vibrating in your pocket is something urgent can now be discretely answered.

A doctor, whose life is run by a pager, is tending to a patient. Suddenly, a page, sent to the doctor's PDA, interrupts the examination and is displayed on the watch. However, what if the doctor's hands were occupied? The accelerometer built into the OrbitWatch will allow for hands free operation so pages can be browsed without interrupting the procedure.

## 3. OrbitWatch Design and Prototype

We have designed and prototyped the proposed OrbitWatch, configured as seen in the following figure. The unit consists of a 2x8 character display surrounded by three buttons, which control all aspects of the watch's operation. It also contains a three-axis accelerometer from Kionix and a temperature sensor for motion-sensing based operation and context-aware computing.

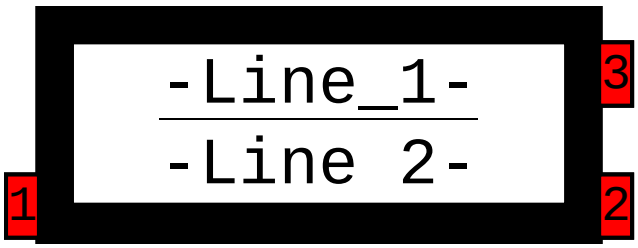


**Figure 2: OrbitWatch user interface**

The OrbitWatch utilizes a Bluetooth transceiver for wireless communication with a mobile phone and employs a TI-MSP430F1612 microcontroller for computing with minimal power consumption. The OrbitWatch PCB design and a complete prototype are shown in Figure 3.

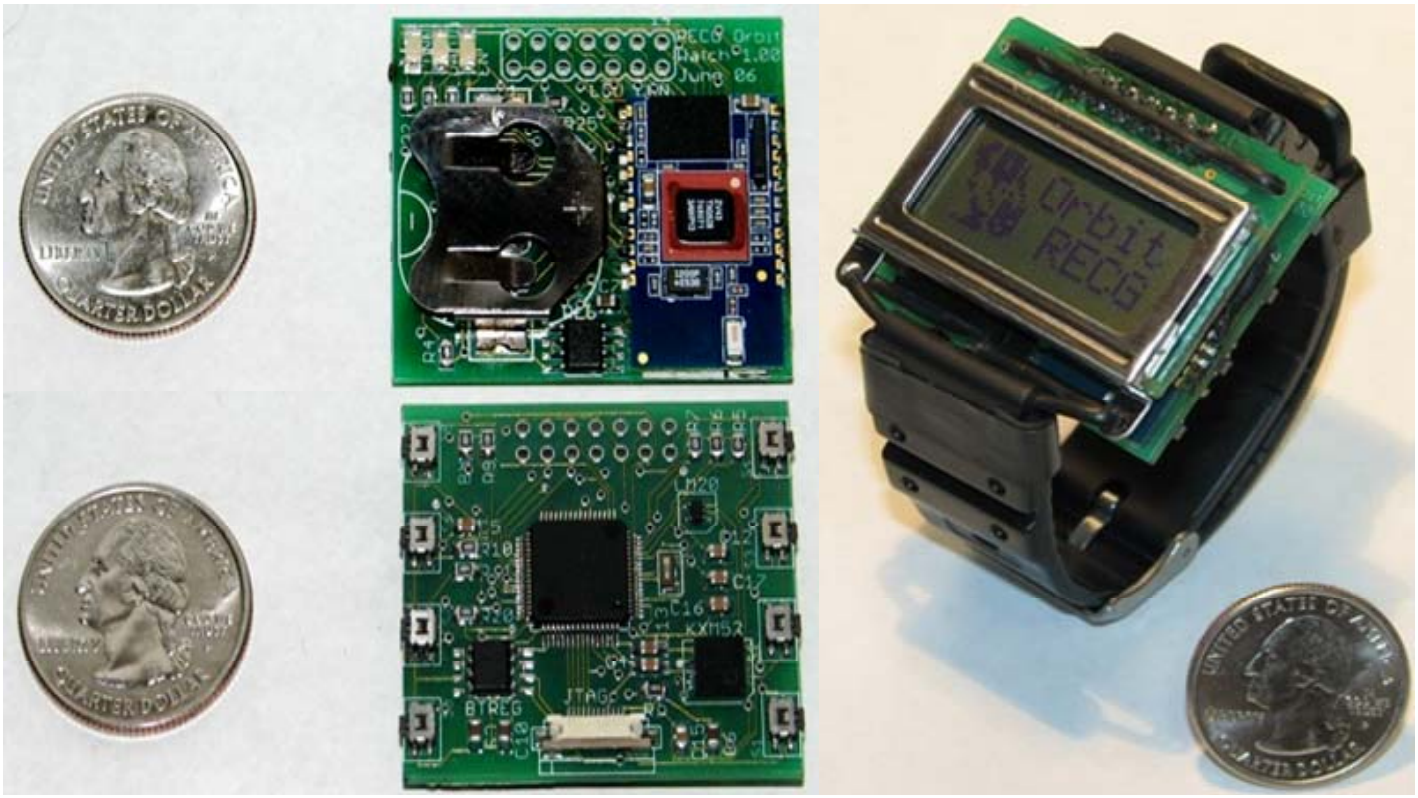


**Figure 3: OrbitWatch PCB design and complete prototype**

We also designed a debugging board for application developers to explore different features of the OrbitWatch as shown in Figure 4.

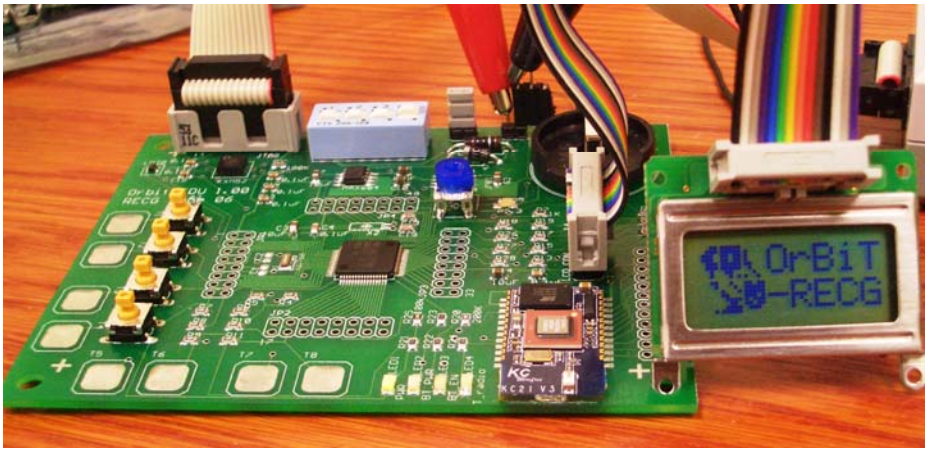


**Figure 4: Orbit Debugging Board**

### 3.1. OrbitWatch User Interface

We next explain the OrbitWatch operation using a state machine whose states are navigated through button presses or wrist motions. The two main operating states are Auto and Manual modes, which describe the user control over the watch display.

In Auto mode, text messages stored in the watch are cycled through and scroll across the top line of the display while the time is displayed on the second.

In Manual mode, the upper line pauses at the current message and allows for the user to cycle through the message character by character, to proceed to following messages, or to return to Auto mode. The display of the second line is determined by the type of the currently displayed message.

A message can either be simple text or a question, which saves a user inputted answer. If the current message were simple text, then the second line of the display in manual mode would still display the current time. If the watch user opts to delete the message, the second line changes to a prompt which verifies this selection. If the current message were, on the other hand, a question, the second line would automatically prompt for a yes or no answer, an answer to which would delete the message from the display cycle. In any of these sub-states, the user may still advance character by character, cycle through to the next message, or return to Auto mode. Also, a user specified timeout where no button activity occurs in Manual Mode will cause the system to revert back to Auto Mode. In addition to the buttons, the accelerometer will detect intentional “jolts” and watch tilt to emulate button push operations.

Figure 5 catalogs the button pushes and motion inputs required for traversal between the different modes and states.

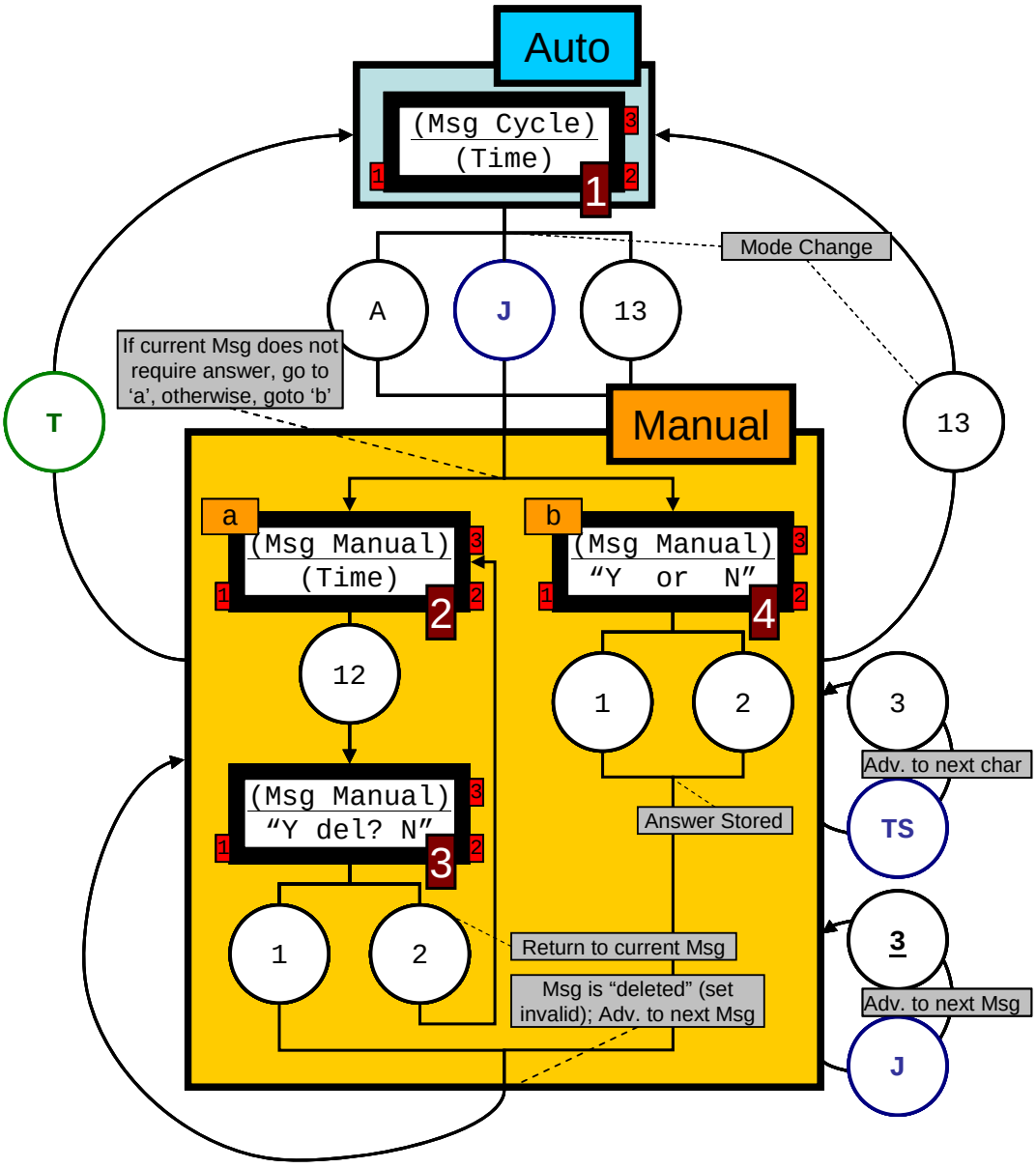


**Figure 5: The State Machine**

## Legend:

- Buttons -

1: Button one press
2: Button two press
3: Button three press
12: Button one and two press
13: Button one and three press
**1**: Button one hold (etc.)
A: Any button press
**T**: Timeout
**J**: Jolt Motion
**TS**: Tilt Motion - Side

- General Map -

13: Mode Switch
12: Delete (normal text message)
3: Scroll (button click character advance)
**3**: Next message (in manual mode)
1: Display dedicated button (yes)
2: Display dedicated button (no)

## 4. System Architecture

We next present the hardware and software design of the OrbitWatch.

### 4.1 Hardware

A conceptualization of the hardware layout is as shown in the following diagram.

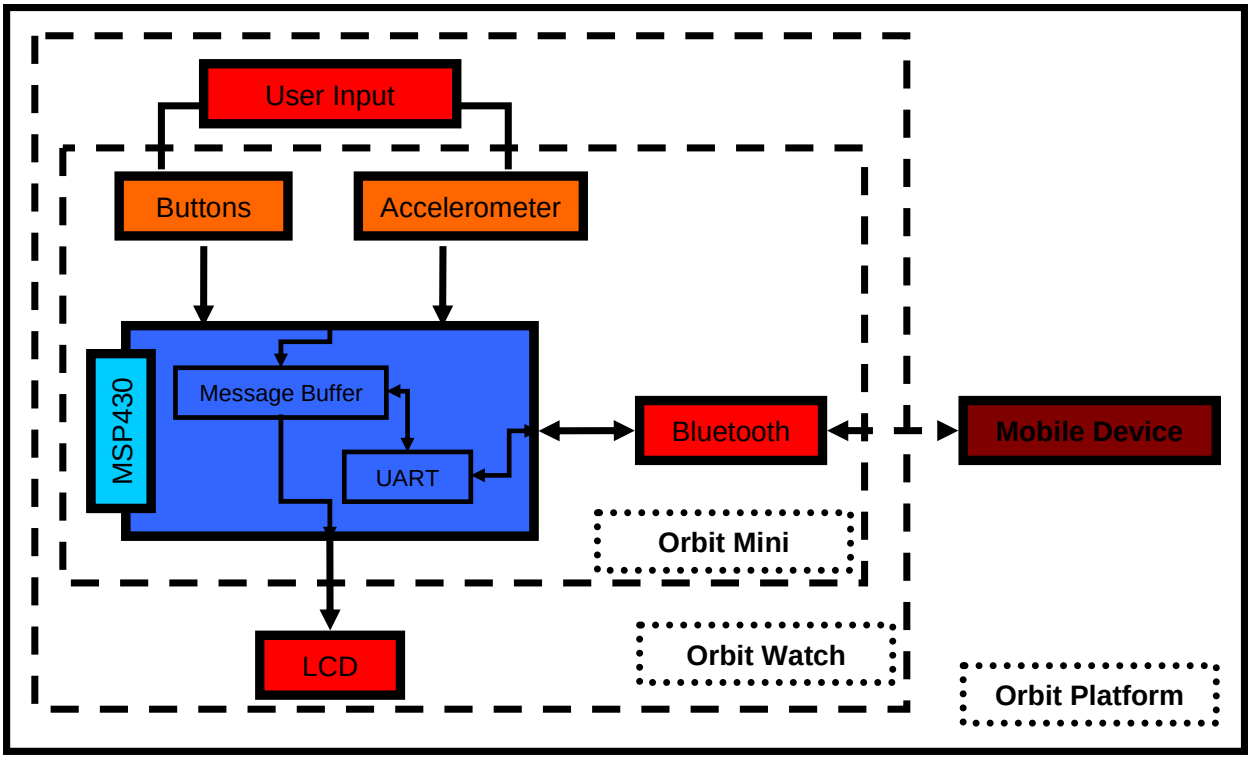


**Figure 6: Hardware Design**

*MSP430*
The MSP430F1612 is an ultra low-power microcontroller that drives the OrbitWatch. Several General Purpose Input/Output (GPIO) pins run the LCD and buttons, the Analog to Digital Converters (ADC) processes the information from the accelerometer and temperature sensor, and the Universal Asynchronous Receiver/Transmitter (UART) sends data through the Bluetooth module. The MSP430 is adaptable, inexpensive, and energy efficient so it is perfect for such an application.

*LCD*
The LCD, a simple 2x8 character display, is controlled by twelve GPIO pins that transmit the information to be displayed every timer interrupt. The expansive display capability of the peripheral coupled with its inherently simple design make the Orbit Watch platform adaptable to a wide array of styles and configurations.

*Buttons*
The Orbit Watch uses three buttons, connected to three GPIO pins, for user operation. The buttons are polled for once every timer interrupt to determine whether the user's depression of a button is a single click or a hold.

*Accelerometer*
The three axis Accelerometer is connected to three ADC inputs for processing. Again, these values are sampled and processed once every timer interrupt to detect jolts and tilts that mimic button press operations.

*Bluetooth*
We have used the KCWirefree KC21 Bluetooth module. It is relatively small and energy efficient, and comes with a built-in antenna. Bluetooth connectivity enables the phone to periodically synchronize with the watch. The MSP430 is connected to the KC21 through a UART interface. Bluetooth transmissions account for over 95% of the OrbitWatch energy consumption.

*Battery and Battery lifetime*
The current OrbitWatch contains a LR2032 coin cell Lithium Ion rechargeable battery with a capacity of 40mAh. With a Bluetooth synchronization interval of 5 minutes, the OrbitWatch can last for about two days. It would have a much longer battery lifetime if a larger battery or more integrated design were used.

**4.2 Software**

The driver for the prototype is constructed around the Message Buffer, which stores all of the incoming messages from the Bluetooth source. Application designers can simply store messages into this buffer through Bluetooth under the proper specifications and the driver will take care of the display.

*Message Buffer*
The Message Buffer is a 16x32 two dimensional array that stores up to 16 messages with 30 characters each. Two additional characters per message are used for message configuration data such as its length, blink status, validity, question status, and its priority. The valid bit of each message indicates whether or not the message has been deleted. When a user opts to delete a message through the watch interface, the message is not actually cleared from memory but rather is invalidated. Thus, the system does not display invalid messages. Only the Bluetooth source has the power to overwrite/delete messages from the memory. Also, the valid bit is checked each time a new character is displayed so if the Bluetooth source deletes a message while it is being displayed (and properly sets the invalid bit before overwriting the data), then the message stops mid-scroll and the next message is selected. If there are no messages in the Message Buffer, then "none" will be displayed on the top line.

The following are specifications for the Message Buffer:

- Message Buffer: "`char MSG_BUF[a][b]`"
    - a: Message Index
    - b: Message Data
- Each message consists of 32 characters
    - Char 0 – 29: The message for display (30 character maximum)
    - Char 30: Message length (character interpreted as unsigned 8-bit integer)
    - Char 31: Message configuration byte (most to least significant bit)

1. Blink Bit (Hi: Blink, Lo: Normal)
2. Valid Bit (Hi: Valid, Lo: Invalid) – Only valid messages are displayed
3. Question Bit (Hi: Message requires answer, Lo: Simple Message)
4. Answered Bit (Hi: Question has been answered, Lo: Question not answered)
5. Answer Bit (Hi: Yes, Lo: No)
6. Priority Bit
7. Priority Bit (8 priority levels)
8. Priority Bit

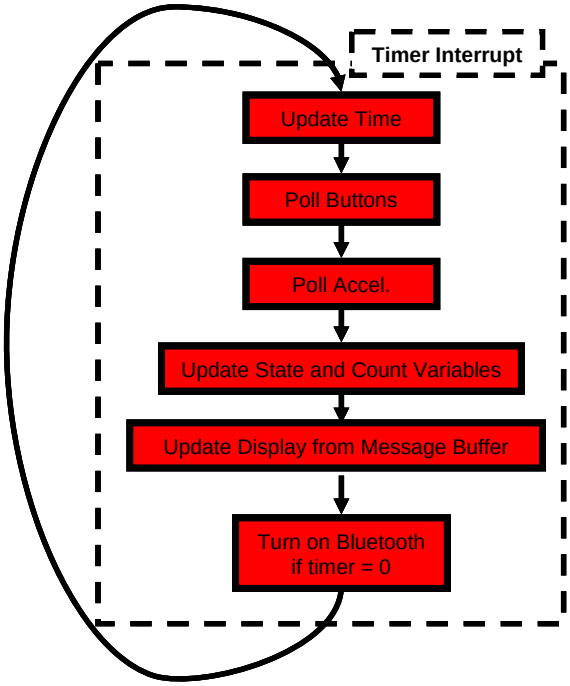


**Figure 7: Timer Interrupt**

*Timer Interrupt*

The timer interrupt occurs 16 times each second. All watch operation occurs through functions that are repeatedly called in this interrupt. This means that buttons are polled, states are switched, characters are displayed, and times are updated by calling the same set of functions over and over again. Current information is stored in a set of global variables that indicate the current state, the current message being displayed, the current time etc. By storing current and previous interrupt (or system tick) information, the system can deduce whether or not a button was pressed or held or whether or not a wrist motion was an intentional jolt or title. The scrolling message across the display is also accomplished this way by displaying eight character sections of the messages every specified number of system ticks. Organizing almost the entirety of the watch operation in this single interrupt allows for minimal interrupt nesting and thus more reliable operation.

*Bluetooth/UART Interrupt*

The MSP430 communicates with the Bluetooth module using one of the two UART ports. Data transfer is interrupt-driven. The OrbitWatch and the mobile phone communicate using an application-layer protocol similar to Hayes modem commands. All commands begin with “AT*” and end with a “CR+LF.”

Responses are also in this format. Commands must be sent within a specific timeout period, or they will be disregarded. Anything that is not a command or a response is also disregarded.

To minimize power usage, the Bluetooth module is completely powered down whenever possible. This is enabled by a separate regulator for the Bluetooth module, which is controlled by the MSP430. After synchronization, the phone (master) sends a command to the watch to turn the Bluetooth module off for a specific time length. The Bluetooth module is turned on after that time period, and the connection is re-established.

To save the battery, for example in the case the phone is no longer near the watch, we have implemented a Bluetooth connection timeout. If the connection cannot be re-established after this timeout, the Bluetooth module is turned off, and only turned on periodically for a short length of time. This enables connection re-establishment, albeit with slight latency, while enjoying great energy savings.

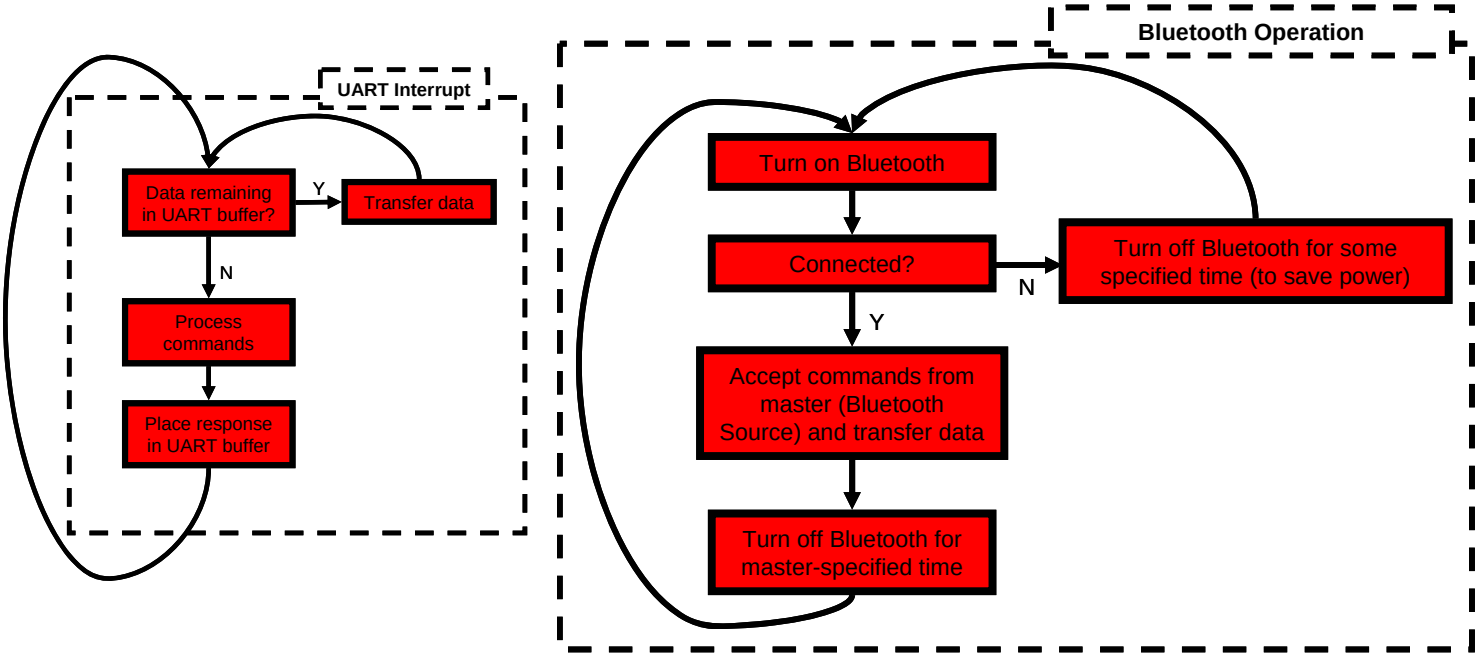


**Figure 9: UART interrupt handler for Bluetooth (left) and Bluetooth power management (right).**

## 5. Conclusions and Future Work

We present the OrbitWatch, a simple, flexible, prototype design of a wrist-worn user interface for convenient information access. The OrbitWatch is based on the TI MSP430F1612 microcontroller and a Bluetooth transceiver for use with mobile phones. It functions as an energy-efficient secondary, readily-accessible user interface for a mobile phone. Unlike previous information watch designs, the OrbitWatch enjoys a very simple, low-power design that leverages the computing capacity of the mobile phone. The OrbitWatch also has a simple interface with three buttons and sensors for motion and temperature that support hand-free operation. We intend for the OrbitWatch to be incorporated into future information watches with an additional text display

embedded into the regular watch face. We believe such information watches will gain wider social acceptance than existing complicated, power-hungry, digital watches with a tech-heavy appearance.

We plan to build the battery charging interface into the next version of OrbitWatch so users do not have to remove the battery for recharging. We also plan to conduct user studies on the current OrbitWatch user interface and evaluate user satisfaction and social acceptance. Although the current OrbitWatch power consumption is low, we believe there is still room for improvement, especially with the Bluetooth interface. We will develop new power management policies for even lower power Bluetooth communication.