METAWATCH

META WATCH

SYSTEM OVERVIEW

Pre-release document version A

METAWATCH

Contents

METAWATCH

1 Introduction

This document lists the commands used to talk to the Apollo watch using the Bluetooth Serial Port profile.

1.1 Revision History

Date	Change Description	Revision

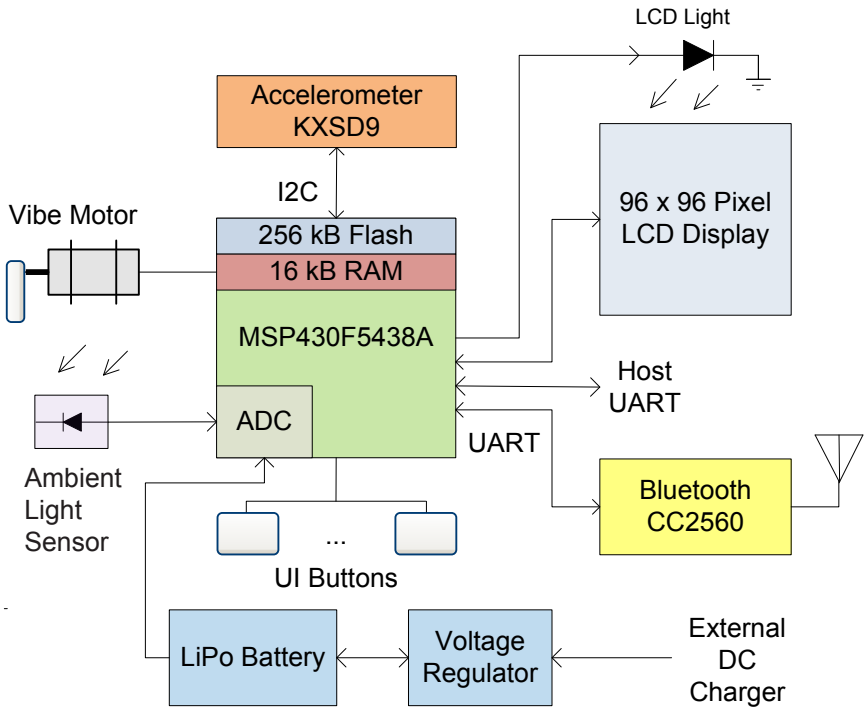
Table 1: Revision History

2 Meta Watch Platform Architecture

2.1 Hardware Platform

The Meta Watch platform is based on the TI MSP430F5438A microprocessor and CC2560 Bluetooth radio.

2.1.1 Digital Watch Block Diagram



2.1.2 Analog / Digital Watch Block Diagram

Diagram TBD. It's the same as the digital except for the display and analog motor connections.

2.2 Software Platform

A block diagram of the overall software architecture is shown in Figure 1. The operating system is FreeRTOS (from freertos.org). It is a small preemptive RTOS commonly used in embedded applications. The number of tasks in the system is not fixed, however due to the limited RAM in the MSP430, the typical architecture is recommended.

Communication between tasks is via message queues. There is a system buffer pool to support alloc / free of messages. Events are also sent as messages. Each task has an input message queue.

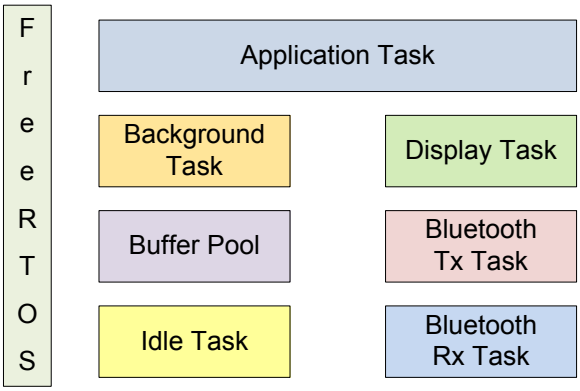


Figure 1 Software Block Diagram

2.2.1 Application Task

It is possible for watch functions to be controlled completely through messages from the phone. However, the application task provides a way to customize functionality of the watch, and provide functionality when disconnected from the phone. The application task is responsible for managing the Bluetooth pairing of the watch with a phone.

2.2.2 Display Task

The display task is responsible for managing updates to the LCD (or OLED) display. Updates may be in response to a message from the Application Task, a message from the Phone or an internal event such as time update.

The Display Task maintains an internal buffer to allow rapid updates of images transferred from the phone or created by the application task. New images are written to the display buffer. When a complete buffer has been written, to buffer data is transferred to the LCD display.

2.2.3 Background Task

The background task handles the system control and status functions, such as ambient light measurement, battery charging, button status and monitoring/control of the vibration motor.

2.2.4 Buffer Pool

Events and inter-task communication are performed using messages. A buffer pool is supplied to assist in managing message flow. Tasks are responsible for allocating a buffer when sending a message and freeing (or forwarding) a message placed on their queue.

2.2.5 Idle Task

The idle task is responsible for power management of the MSP430. When no other tasks need to run, the processor is placed in a low power sleep mode. The lowest power sleep mode we can run to support time keeping and Bluetooth communications is MSP430 LPM3 (Low Power Mode 3).

2.2.6 Bluetooth Rx/Tx Tasks

Bluetooth communication is managed with a pair of tasks. Most of the communication does not require the involvement of the other tasks. Messages are sent to the phone by placing them on the Bluetooth transmit queue. Messages received from the phone are placed on the queue of the task(s) that has registered for them.

The application task is responsible for handling some Bluetooth events, such as pairing and (re)connection. Bluetooth API calls are provided to assist in managing the link.

METAWATCH

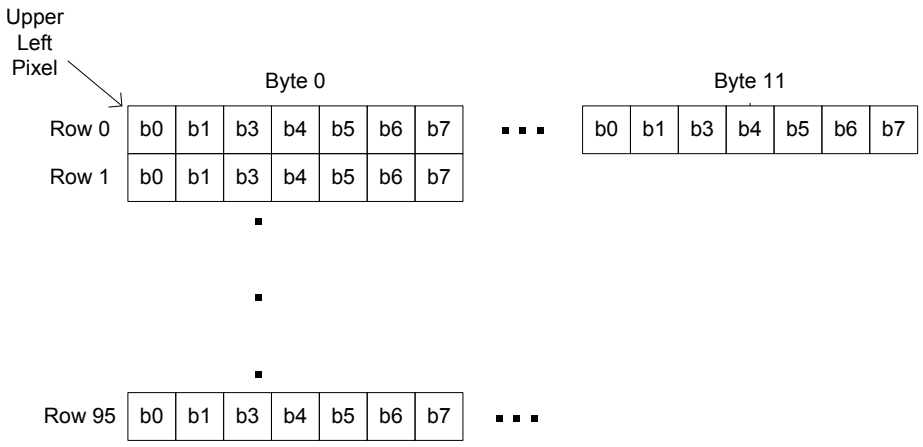
3 Watch Systems

3.1 DIGITAL DISPLAYS

Each type of digital displays has a buffer structure in memory. There are two types of digital displays supported by the platform today, a 96x96 pixel TFT LCD and a 16x80 pixel OLED. The Display task accepts messages for controlling the displays routed from the local system or from the wireless connection.

3.1.1 LCD Display Buffers

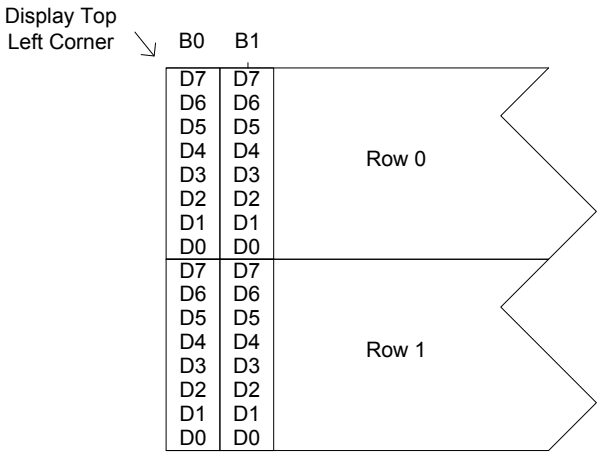
The LCD display is 96x96 pixel monochrome. Each LCD buffer is arranged as 96 lines of 12 bytes. A block diagram of the pixel layout is shown in **Error! Reference source not found.**.



3.1.2 OLED Display Buffers

The OLED displays are both 16x80 monochrome. All buffers are arranged as two eight pixel high rows (lines), so the display can be used either as one line (16 x 80 pixels) or two lines (8 x 80 pixels each).

Byte zero in the buffer is located at the left most position on the display. The byte to pixel orientation in the display is most significant bit is at the top of the display.



3.2 ANALOG WATCH HANDS

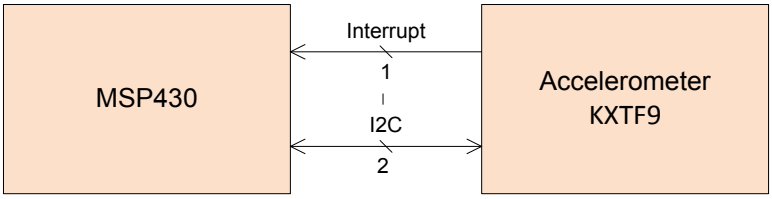
The motor that controls the analog watch hands is connected to the microcontroller. Normal time keeping function is a system function, but it is possible to move the hands electronically. Movement of the hands is only clockwise and absolute position can not be read from the motor. Any movement will be made relative to the current position. Keep this in mind.

3.3 ACCELEROMETER

A Kionix KXTF9 is accelerometer is included in the watch hardware. Motion events can be detected by setting it to wake up on orientation change, tap detection, or programming the accelerometer's motion thresholds.

The usage of the accelerometer is highly application dependent. The amount of power used by the accelerometer would exceed the rest of the watch if it were left on continuously. For that reason, control of the accelerometer is left to the application.

The accelerometer has its own dedicated I2C port so the application task is not blocked for a long period of time waiting to access the I2C bus.



3.4 AMBIENT LIGHT SENSOR

The ambient light sensor is managed by the background task. Measurements are made at the default interval or the application may change the settings.

3.5 BATTERY MONITOR

The battery monitor is part of the background task. Battery charging is handled automatically.

3.6 BUTTON INPUTS

Button inputs are monitored by the background task. Button inputs are interrupt driven so that the button event is sent to the application and/or phone immediately after the button debounce time.

3.7 TIME

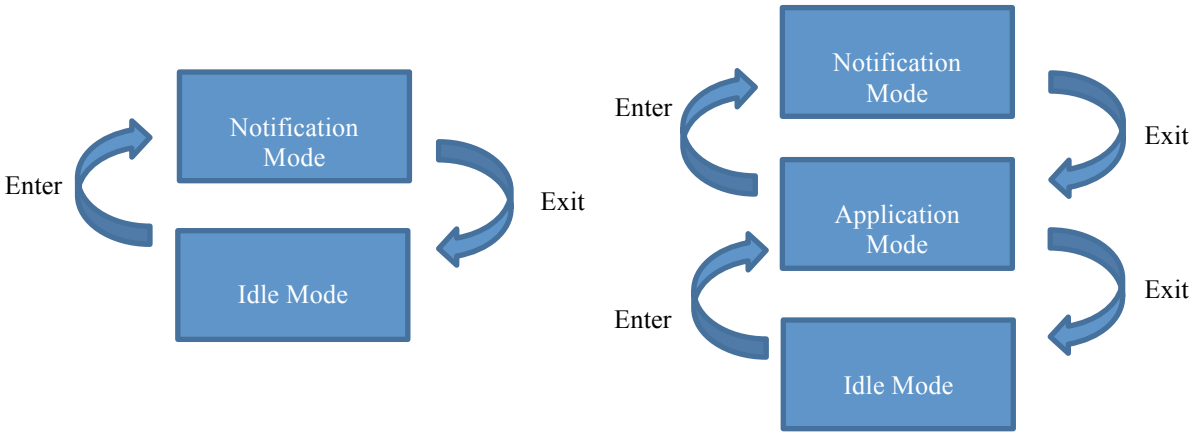
The real time clock time, day and date is set by sending a message to the background task which makes the appropriate updates to the internal RTC registers. The alarm time is updated with a similar message.

3.8 VIBRATION

Vibration is enabled/disabled by sending a message to the background task.

4 Watch Modes

The watch supports three basic running modes to add some structure to the interaction with the phone. The modes are Idle, Notification and Application Mode, each mode has its own state machine, options, and display buffer. Idle Mode is the most basic mode and any timeouts will ultimately return here. Notification mode can interrupt either Idle mode or Application mode, but when Notification Mode is exited, the watch will return to the last active mode.



4.1 Idle Mode

Idle mode is the default mode of the watch. The digital watch displays the Idle mode buffer constantly. The analog watch displays are OFF during Idle mode. To conserve power the watch should strive to be in Idle mode as often as possible and the phone should update Idle mode buffers only the minimum needed to keep the display information up to date.

By default, the digital display for Idle mode is shared with a watch controlled system area that keeps time and date. This operation can be changed so the entire display is available for the remote application.

By default the watch reserves all button input for built-in user interface functions. Idle mode buttons can be overridden by a remote application using the Configure Mode message.

4.2 Notification mode

Notification mode is for important transient notifications. You can think of notification mode as interrupting the current mode for a short time and then dropping back again, like an SMS alert, or a Tweet. Notification mode can interrupt both Idle mode and Application mode. Notification mode is explicitly entered and exited using a protocol message. Notification Mode will timeout after a period of inactivity. The mode that was active before the interruption is the mode the watch will return to.

The entire screen is available during this mode and by default all button press events are sent to the phone while in Notification mode.

4.3 APPLICATION MODE

Application mode is a mode used when control over the screen and buttons is required for an extended period of time. This “dumb terminal” mode can be used for an application to tailor a specific type of interface.

Application mode is explicitly entered and exited using a protocol message.

During Application mode, the watch will periodically send a keep-alive message that must be responded to or Application mode will time out (TBD).

METAWATCH

5 Bluetooth Interface

5.1 Overview

Communication between the watch and the phone is over the Bluetooth interface. The upper layers of the Bluetooth protocol stack are implemented in code on the MSP430. To shorten descriptions, the upper layers of the Bluetooth stack running on the MSP430 will be referred to as just "the stack" since they are what the code described here interacts with.

Communication uses the Bluetooth serial port profile (SPP) which is supported by most smart phones. While SPP can implement the serial port RTS and CTS signals, it is common for only the serial data stream to be supported.

As a serial data stream, SPP does not support any kind of framing of the data or checking of the serial stream. SPP is normally configured to not time out so it will not drop bytes. However it is still possible for the data stream to be corrupted it the Bluetooth connection is lost. In this application, it is preferable to set a packet timeout to make the Watch / Phone interface more responsive.

5.2 Connections and Paired Data

It would be very difficult for the watch to manage image data from more than one source, particularly because the devices that the watch connects to can change its mode. For this reason, Meta Watch only supports one active Bluetooth connection at a time. At this time, Meta Watch will only store data for one paired device at a time.

5.3 Bluetooth Control and Status

Most of the Bluetooth communication is handled automatically by the stack. Control and configuration of the link with the phone requires user input.

To avoid adding a separate task to manage the Bluetooth connection, a set Bluetooth API functions allows the application to perform link control and monitor link status.

METAWATCH

6 Remote Protocol Message Structure

The Meta Watch Remote Protocol is a lightweight, bi-directional message based command protocol that can be used to remotely configure and control the watch. These protocol messages are sent via the Bluetooth SPP wireless link.

6.1 PACKET FORMAT

The protocol has a standard packet format. Packets are based on a 32 byte maximum message length.

Start	Length	MsgType	Options	Data	CRC
B_0	$B_1 = N$	B_2	B_3	$B_{3+1} \ldots B_{N-2}$	B_{N-1} , B_N

Start: ASCII <SOF> = 0x01

Length: Total number of bytes in the packet including start and CRC

Type: Packet type, typically a command

Options: Additional information about the type or command

Data: Packet data, 0 to 26 bytes

Check: CRC-CCITT All packet bytes (up to the CRC)

The CRC matches the settings in the MSP430 it is CRC-CCITT (0xFFFF) with reverse input bit order.

CRC({0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37, 0x38, 0x39}) = 0x89F6