DESIGN OF A LOCATION TRACKING DEVICE FOR REMOTE PATIENT MONITORING

Chika Ezeh
B.S., California State University, Sacramento, 2001

PROJECT

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

SUMMER 2011
DESIGN OF A LOCATION TRACKING DEVICE FOR REMOTE PATIENT MONITORING

A Project

by

Chika Ezeh

Approved by:

__________________________________, Committee Chair
Warren D. Smith, Ph.D.

__________________________________, Second Reader
Anita Bagley, Ph.D.

__________________________________
Date

ii
Student: Chika Ezeh

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator       ________________
Preetham Kumar, Ph.D.       Date

Department of Electrical and Electronic Engineering
Abstract

of

DESIGN OF A LOCATION TRACKING DEVICE FOR REMOTE PATIENT MONITORING

by

Chika Ezeh

Shriners Hospitals for Children, Northern California provides treatment to improve the mobility of children with cerebral palsy. The hospital wants to monitor falls and activity levels of the children under treatment. A separate project developed a Fall Detection Device to log falls and activity levels in children during two weeks of everyday living. However, the Fall Detection Device cannot identify where falls occur.

This project fills the gap of identifying where falls occur by developing a mobile Location Tracking Device to work together with the Fall Detection Device. The project focuses on design, integration, and testing of the hardware and software that make up the Location Tracking Device. The overall design was driven by specific requirements defined by Shriners Hospitals for Children on the types of data to be collected, how often the data need to be collected, how long the data need to be collected, and how the collected data will be analyzed.
The project successfully met all pre-defined objectives, except for the useful service life of the battery. The useful service life of the battery used in the design was calculated to be 13.6 days, while the design objective was 14 days. The report discusses areas of improvement, including design changes that will help increase the service life of the battery beyond 14 days.

_______________________, Committee Chair
Warren D. Smith, Ph.D.

_______________________
Date
ACKNOWLEDGMENTS

I would like to thank Dr. Warren Smith for his guidance and support in this project. He inspired me to develop the design ideas and methodologies that are critical to the success of this project. I would also like to thank Dr. Anita Bagley for suggesting this topic, setting the project goals, and providing valuable feedback on the project report.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>ix</td>
</tr>
<tr>
<td>List of Figures</td>
<td>x</td>
</tr>
<tr>
<td>Chapter</td>
<td></td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. BACKGROUND AND DESIGN METHODS</td>
<td>4</td>
</tr>
<tr>
<td>2.1. GPS</td>
<td>4</td>
</tr>
<tr>
<td>2.2. GPS Module</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1. Power Mode</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2. Communication Protocol</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3. Communication Interface</td>
<td>10</td>
</tr>
<tr>
<td>2.3. Microcontroller</td>
<td>11</td>
</tr>
<tr>
<td>2.3.1. Flash Memory Module</td>
<td>13</td>
</tr>
<tr>
<td>2.3.2. Clock Module</td>
<td>13</td>
</tr>
<tr>
<td>2.3.3. Timer Module</td>
<td>14</td>
</tr>
<tr>
<td>2.3.4. DMA Controller</td>
<td>14</td>
</tr>
<tr>
<td>2.3.5. Operating Mode</td>
<td>15</td>
</tr>
<tr>
<td>2.3.6. UART Interface</td>
<td>16</td>
</tr>
<tr>
<td>2.4. USB-UART Interface Module</td>
<td>16</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1. Table 1 Sample GPRMC message structure .................................................... 11
2. Table 2 GPS module average power consumption ............................................. 31
3. Table 3 Memory required to store GPS data without compression .................... 32
4. Table 4 Preamble data representation ................................................................. 34
5. Table 5 Worst-case memory requirement with data compression .................... 36
6. Table 6 Battery service life estimation................................................................. 38
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Figure 1 A constellation of satellites orbiting the earth</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>Figure 2 Determining signal travel time from SV to GPS receiver</td>
<td>6</td>
</tr>
<tr>
<td>3.</td>
<td>Figure 3 u-blox LEA-5 based GPS module</td>
<td>8</td>
</tr>
<tr>
<td>4.</td>
<td>Figure 4 NMEA 0183 protocol message structure</td>
<td>10</td>
</tr>
<tr>
<td>5.</td>
<td>Figure 5 MSP430F169 microcontroller development board</td>
<td>12</td>
</tr>
<tr>
<td>6.</td>
<td>Figure 6 Current consumption vs. operating mode</td>
<td>15</td>
</tr>
<tr>
<td>7.</td>
<td>Figure 7 USB to UART interfacing module</td>
<td>17</td>
</tr>
<tr>
<td>8.</td>
<td>Figure 8 MAX884 pin configuration</td>
<td>18</td>
</tr>
<tr>
<td>9.</td>
<td>Figure 9 Device hardware integration block diagram</td>
<td>20</td>
</tr>
<tr>
<td>10.</td>
<td>Figure 10 Power control module schematic</td>
<td>21</td>
</tr>
<tr>
<td>11.</td>
<td>Figure 11 Program flow diagram</td>
<td>24</td>
</tr>
<tr>
<td>12.</td>
<td>Figure 12 HyperTerminal window showing UART settings</td>
<td>39</td>
</tr>
<tr>
<td>13.</td>
<td>Figure 13 HyperTerminal window showing device initialization</td>
<td>40</td>
</tr>
<tr>
<td>14.</td>
<td>Figure 14 HyperTerminal window showing valid GPS data in memory</td>
<td>41</td>
</tr>
<tr>
<td>15.</td>
<td>Figure 15 HyperTerminal window showing no GPS data in memory</td>
<td>43</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

Remote Patient Monitoring (RPM) refers to a range of technologies specifically designed to aid in remotely monitoring some health conditions effectively [1]. Remote tracking devices and point-of-care monitoring devices can be used to form a powerful health data collection infrastructure for the health system. Such a remote monitoring system can allow valuable patient health information to be collected while the patients are miles away from the doctors’ offices, during normal day to day activities. The system can help doctors and caregivers gain more insight into how the patients are responding to doctors’ treatments, allowing doctors to adjust treatment regimens for the optimal results. This benefit of RPM technology has caught the attention of hospitals across the globe. One such hospital interested in the use of Remote Patient Monitoring technologies to improve the quality of healthcare services is Shriners Hospitals for Children, Sacramento California.

Shriners Hospitals for Children has requested the development of a Remote Patient Monitoring system to be used in the treatment of children with cerebral palsy. The system will consist of two parts – A Fall Detection Device to identify falls and a Location Tracking Device to identify where falls occur. The knowledge of where a fall occurs can help the clinician better understand the nature of the fall. For instance, a fall in the playground at play time might be looked at differently from a fall at the dining table during dinner. Therefore, location tracking becomes an integral part of an
effective monitoring system for children under treatment for cerebral palsy. This project focuses on the development of a functional prototype for the Location Tracking Device.

The Location Tracking Device will be battery operated, and it needs to be small and lightweight enough to be worn by a child during normal daily activities. The device needs to be capable of operation on a single charge for a duration of two weeks. The device’s operation will be autonomous during the monitoring period with no human intervention required. A set of time, position, and speed data will be collected by the device once every 5 min. Blackout periods of no data collection are acceptable at night when the monitored child is sleeping and when the child enters buildings that adversely affect the reception of signals necessary to determine location. The device will be capable of storing the collected data in a non-volatile medium, and the device also will be capable of transferring the collected data to a Personal Computer (PC) for further analysis. The data collected by the Location Tracking Device include date, time, longitude, latitude, and speed.

To meet these specifications, the device has been designed around a Global Positioning System (GPS) module, a microcontroller unit, and a Universal Serial Bus (USB) interface. The GPS module is used to collect the necessary data. The microcontroller controls the operation of the GPS module, processes and stores the acquired data, and transfers the data to a PC. The details of hardware and software
design, integration, and testing to meet the project’s objectives are discussed in the remainder of this report.

Chapter 2 of the report provides a background on the hardware components and methods used in the design. It also provides a high-level overview of the GPS system, mode of operation, and related terminologies. Chapter 3 presents the hardware integration and software development for the project. Chapter 4 discusses device testing and test results. In Chapter 5, the summary and conclusions of the report are presented, as well as recommendations for improvement.
Chapter 2

BACKGROUND AND DESIGN METHODS

2.1. GPS

Satellite Navigation is a method that uses a Global Navigation Satellite System (GNSS) to determine position and time anywhere on earth [2]. A GPS system developed by the U.S. Department of Defense is a form of GNSS that consists of a constellation of satellites orbiting the earth at very high altitudes. An illustration of a constellation of satellites orbiting the earth is shown in Figure 1. These Satellite Vehicles (SVs) periodically transmit electromagnetic signals corresponding to certain SV orbit information. The signals also include the time of signal transmission, which is derived from highly accurate atomic clocks on board the SVs.

A compatible GPS receiver can receive the transmitted signals and determine, through mathematical computations, the location of the GPS receiver on earth. The computations require signals from at least three SVs. The SV orbit information helps to determine the position of the SV, while the time from an on-board atomic clock helps to determine the signal travel time from the SV to the receiver. With the transmitted electromagnetic signals travelling through space at the speed of light, if the travel time is known, the relative distance of the receiver from the SV can be computed. Figure 2 shows how the signal travel time from an SV to a GPS receiver can be determined. Some of the signals transmitted by the SV to determine position include almanac, ephemeris, and time.
Almanac data transmitted by a GPS satellite include orbit information on all the satellites, satellite clock correction, and atmospheric delay parameters [3]. In order to receive signals transmitted by the SVs, a line of sight is required from GPS receivers to SVs in orbit. With valid almanac data, GPS receivers are configured to receive signals from SVs that are most likely to be above the horizon. Without valid almanac data, GPS receivers search blindly for SV signals. This search can last for several minutes.

While almanac data are used to approximate the position of SVs in orbit, the ephemeris data are what determine the position of the SV in orbit with high
Figure 2. Determining signal travel time from SV to GPS receiver [1, p. 15]

The effect of having valid almanac or ephemeris data is evident in the different modes in which GPS receivers start up. Depending on the availability and validity of the ephemeris or almanac data, GPS receivers start up in either Cold Start mode, Warm Start mode, or Hot Start mode. Often referred to as Time To First Fix (TTF), the length of time it takes a GPS receiver from start-up to a valid position solution is a reflection of the start-up mode.

The start-up mode of a GPS receiver with no valid almanac, ephemeris, and time is Cold Start [3]. Cold Start has the longest TTF due to the lack of valid almanac data in this mode.

Warm Start is a start mode of a GPS receiver when almanac data are valid, but ephemeris data are invalid [3]. Almanac data facilitate approximations that enable
faster signal acquisitions from SVs. Warm Start mode has a better TTF than Cold Start mode.

Hot Start is the start mode of a GPS receiver when current position, time, and SV positions are all available [3]. In this mode, the receiver uses the available valid ephemeris data to accurately compute a position. Hot Start mode has the fastest TTF.

The TTFs of GPS modules often are considered by developers when choosing modules for new design. Other characteristics considered include power consumption, operating voltage, and features that enable flexibility in design.

2.2. GPS Module

The GPS module of choice for this project is based on the LEA-5H-0-007 GPS receiver produced by u-blox AG, Switzerland. “The LEA-5 module series is a family of stand-alone GPS receivers featuring the high performance u-blox 5 positioning engine. These versatile receivers feature an extensive and flexible range of functionality, connectivity and cost savings options” [4, p. 1]. The module operating voltage can range from 2.7 V to 3.3 V. Some of the GPS module’s functionalities critical to the development of this project include programmable power mode, configurable communication protocol, and configurable input/output communication interface. The module also features a non-volatile memory that allows its configurations to be retained even after the module has been completely shut down. A picture of the LEA-5 based GPS module is shown in Figure 3.
2.2.1. Power Mode

The GPS module can be programmed to operate in three different power modes – Maximum Performance mode, Eco mode, and Power Save mode. In Maximum Performance mode, the acquisition engine of the GPS module runs at its highest performance level when searching for satellites and downloading almanac data [4]. The Maximum Performance mode achieves the best TTF at the expense of power. A less power-consuming mode of operation is the Eco mode. This mode strikes a balance between power consumption and performance by sacrificing better TTF for lower power consumption.

Power Save mode is best suited for designs where power is more critical than high performance. For battery powered designs where less frequent updates are acceptable over a long period of time, Power Save mode is preferred. In this mode, the
GPS module can be programmed to periodically switch itself on and off at configurable intervals, thereby reducing the average tracking current [4]. Given that this project aims at producing a device that can operate for two weeks on a single charge while updating location data once every 5 min., the Power Save mode becomes the mode of choice.

2.2.2. Communication Protocol

Two communication protocols are supported in the u-blox 5 GPS module – the UBX communication protocol and the NMEA communication protocol. The UBX protocol is a ublox AG proprietary communication protocol that uses binary data format to communicate between a ublox AG GPS module and a host computer. On the other hand, the NMEA protocol is a standardized protocol developed by the National Marine Electronics Association (NMEA).

National Marine Electronics Association is a U.S. standards committee that defines the data message criteria that allow GPS receivers to communicate with other electronic equipment [3]. One of the data communication protocols defined by NMEA is the NMEA 0183 protocol, and this is the communication protocol of choice for the project. The NMEA 0183 data format consists of specific GPS information grouped together in data sets known as sentences. These data sets are transmitted in American Standard Code for Information Interchange (ASCII) data format and have comma delimited fields. Each field represents specific GPS information relevant to the group or sentence. Figure 4 shows the structure of the NMEA protocol message.
Of all the several data sets or sentences defined by the NMEA 0183 standard, the GPRMC data set is the sentence with all the GPS information needed for the project. This data set contains information on longitude, latitude, date, time, speed, system status, and course [2]. The time data are given in Coordinated Universal Time (UTC) time, while the speed data are given in knots. The Coordinated Universal Time is Greenwich Mean Time (GMT). Table 1 shows a sample GPRMC sentence with corresponding field descriptions.

### 2.2.3. Communication Interface

The GPS receiver supports up to four communication interfaces. Of the supported interfaces, the Universal Asynchronous Receiver/Transmitter (UART) was
the preferred communication interface for the project. This interface was chosen because it was also supported by other devices used in the project.

Table 1. Sample GPRMC message structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Example</th>
<th>Format</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$GPRMC</td>
<td>String</td>
<td>Message header for GPRMC data set</td>
</tr>
<tr>
<td>1</td>
<td>083559.00</td>
<td>hhmmss.ss</td>
<td>Time of position fix (Coordinated Universal Time)</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>Character</td>
<td>Data set status – A: Valid, V: Invalid</td>
</tr>
<tr>
<td>3</td>
<td>4717.11437</td>
<td>ddm.mmmm</td>
<td>Latitude in degrees, minutes, and fractional minutes</td>
</tr>
<tr>
<td>4</td>
<td>N</td>
<td>Character</td>
<td>Hemisphere (N/S)</td>
</tr>
<tr>
<td>5</td>
<td>00833.91522</td>
<td>ddddm.mmmm</td>
<td>Longitude in degrees, minutes, and fractional minutes</td>
</tr>
<tr>
<td>6</td>
<td>E</td>
<td>Character</td>
<td>E/W</td>
</tr>
<tr>
<td>7</td>
<td>0.004</td>
<td>Float</td>
<td>Speed over ground in knots</td>
</tr>
<tr>
<td>8</td>
<td>77.52</td>
<td>Float</td>
<td>Course over ground in degrees</td>
</tr>
<tr>
<td>9</td>
<td>091202</td>
<td>ddmmyy</td>
<td>Date</td>
</tr>
<tr>
<td>10</td>
<td>-</td>
<td>Numeric</td>
<td>Magnetic Variation – data not calculated by receiver</td>
</tr>
<tr>
<td>11</td>
<td>-</td>
<td>Character</td>
<td>Magnetic variation E/W – data not calculated by receiver</td>
</tr>
<tr>
<td>12</td>
<td>-</td>
<td>Character</td>
<td>Mode indicator – data not calculated by receiver</td>
</tr>
<tr>
<td>13</td>
<td>*57</td>
<td>Character and hex</td>
<td>End of data indicator and checksum value</td>
</tr>
<tr>
<td>14</td>
<td>-</td>
<td>Character</td>
<td>Carriage Return and Line Feed</td>
</tr>
</tbody>
</table>

2.3. Microcontroller

The microcontroller used for the project is the MSP430F169 microcontroller manufactured by Texas Instruments, Dallas, Texas. The microcontroller is packaged in a 64-pin Quad Flat Pack (QFP) package, and it can operate at a voltage in the range of
1.8 V to 3.6 V. This operating voltage range makes it inherently compatible with the voltage levels of the selected GPS Module without the need for a voltage level shifter in the GPS module microcontroller interface. For faster prototyping, a microcontroller development board is used. The development board comes with the microcontroller already soldered on to the board and with the microcontroller input/output ports routed to pin header connectors for easy connectivity. The board also comes with a few components already attached to it, including a Liquid Crystal Display (LCD) interface, a joystick controller, and a 32-kHz crystal oscillator. From the list of pre-attached components, the 32-kHz crystal oscillator is the component used in this project. Figure 5 shows the microcontroller development board with the MSP430F169 microcontroller attached.

Figure 5. MSP430F169 microcontroller development board [6]
The MSP430F169 has over a dozen built-in features and peripheral modules. Only those features and modules relevant to this project are described in this report. These features and modules include a flash memory module, a flexible clock module, timer modules, Direct Memory Access (DMA) feature, low-power feature, and UART communication interfaces.

2.3.1. Flash Memory Module

The microcontroller flash memory is partitioned into two memory sections – the 60-KB main memory section and the 256-B information memory section. These memory sections are split up into segments. The main memory section has a segment size of 512 B, while the information memory section has a segment size of 128 B. Single bits, bytes, or words can be written to the flash memory, but a flash segment is the smallest size memory that can be erased [7]. Write and erase operations are controlled by the flash timing generator, which has to operate at a frequency in the range of 257 kHz to 476 kHz.

2.3.2. Clock Module

The MSP430F169 microcontroller has a flexible clock module that can be configured for ultra-low power application, medium power and performance application, or high performance application. Of the different clock configurations supported by the module, the low-frequency Auxiliary Clock (ACLK) and the high-speed Digitally Controlled Oscillator (DCO) are the clock sources used in the design. The ACLK is driven by a 32-kHz watch crystal, and it is the clock source of choice for
low power applications. At 32 kHz, the ACLK cannot be used to directly drive the flash timing generator when flash write/erase access is desired. The configurable DCO can be used to source the main clock or peripherals that require higher operating frequency.

2.3.3. Timer Module

Two timer modules – Timer_A and Timer_B – are available with the microcontroller. These modules are 16-bit configurable counters that can be used in several different design applications that require event timing. When used as counters with interrupts enabled, the timer modules can count to pre-programmed values and generate interrupts that can be used to either wake the microprocessor up from sleep or re-direct program flow. In this project, Timer_A module is used to keep track of the amount of time that elapses between when the GPS module is powered down at night and when it is powered back up in the morning by the microcontroller.

2.3.4. DMA Controller

The microcontroller has a DMA controller that can be used for memory data transfer without CPU intervention. Using the DMA controller can increase the throughput of peripheral modules. It also can reduce system power consumption by allowing the CPU to remain in a low-power state without having to awaken to move data to or from a peripheral [7]. Two of the three configurable DMA channels are used in this project.
2.3.5. Operating Mode

The configurable operating mode feature of the microcontroller is crucial to the development of this project. The microcontroller can be put into six different operating modes that include one active mode (AM) and five low-power (LP) modes – LPM0 to LPM4. In AM, the Central Processing Unit (CPU), all peripherals, and all clocks are on. In LP modes, some peripherals and clock sources are turned off, depending on the LP mode. Figure 6 shows a graph of current consumption vs. operating mode. In the figure, ICC/µA @ 1 MHz indicates the current consumed by the microcontroller in microamperes when operating at a frequency of 1 MHz. For this project, the active mode and the LPM3 mode are used. In LPM3 mode, the CPU, master clock, and DCO are all disabled while the ACLK is enabled.

Figure 6. Current consumption vs. operating mode [7, p. 2-14]
2.3.6. UART Interface

The microcontroller has two configurable Universal Synchronous/Asynchronous Receiver Transmitter (USART) interfaces that can be configured to operate in UART mode. In asynchronous mode, a USART interface connects the microcontroller to an external system via two pins – URXD and UTXD [7]. In this project, one of the UART interfaces connects the microcontroller to the GPS module communication port, while the other interface is used to connect it to a PC via a USB-UART Interface module.

2.4. USB-UART Interface Module

With modern PCs equipped with Universal Serial Bus (USB) communication ports rather than RS232 compatible communication ports, a device was needed to interface the UART configured devices in the project with a PC. The FT232R USB to UART module was used to accomplish this goal. The module is based on the single-chip FT232R device produced by Future Technology Devices International, Ltd., Glasgow, United Kingdom. A picture of the USB to UART module is shown in Figure 7. In addition to data format conversion, the FT232R also acts as a voltage level shifter to shift the higher voltage levels of the USB signal to the lower voltage levels of the devices in the project, and vice versa.

2.5. Voltage Regulator

The project requires a constant supply voltage of 3.3 V to be delivered to the microcontroller and GPS modules as the voltage at the battery terminals drop with
Figure 7. USB to UART interfacing module [8]

device usage. To accomplish this task, the MAX884 linear voltage regulator
developed by MAXIM Integrated Products, Inc. Sunnyvale, California, was used. This
voltage regulator maintains an output voltage of 3.3 V for an input voltage in the range
of 3.5 V to 11.5 V. Below an input voltage of 3.5 V, the output voltage of the regulator
drops linearly as the input voltage drops, and a voltage difference of 0.5 V is
maintained between the input and output voltages. The regulator has a 200-mA output
current capability and features an on/off signal to enable or disable output voltage. The
on/off signal is used in the design to power down the GPS module at night and to
power it back up in the morning, in order to conserve power. The pin configuration of
the MAX884 Integrated Circuit (IC) is shown in Figure 8.
Figure 8. MAX884 pin configuration [9, p. 1]
3.1. Hardware

A block diagram of the Location Tracking Device is shown in Figure 9. This diagram shows the major components of the design – the power control module, the microcontroller module, and the GPS module. The power control module delivers power to the microcontroller and GPS modules respectively through ports P1 and P2 as shown in the diagram. The power to the GPS module can be turned on or off independently from the power to the microcontroller. The block diagram also shows two serial communication interfaces – UART0 and UART1 – connected to the microcontroller module. To communicate with the GPS module, the microcontroller uses UART0 interface, while UART1 interface is used to communicate with a PC through the USB-UART interface module. Although the block diagram shows a USB connector between the USB-UART interface module and the PC, this connection is not present at all times. The communication with a PC is needed only during device initialization, when the Location Tracking Device is first started up, and during data retrieval at the end of the location tracking period. At other times, the device operates independently and does not need to be connected to a PC. The acquired GPS data are logged in the internal flash memory of the microcontroller, thereby eliminating the need for a separate data storage device. The Location Tracking Device is turned on or off using an on/off sliding switch.
3.1.1. Power Control Module Design

The power control module is made up of two power blocks – Power Block 1 and Power Block 2 – designed around two MAX884 ICs. The schematic for the power control module is shown in Figure 10. In this figure, two rectangular boxes with dashed lines are used to denote the power blocks. The input voltage into each block comes from an on-board 9 V battery, and the output voltage from the each power block is a regulated 3.3 V. Each power block has an independent active-low input, “OFF” to disable the output voltage of that block. In the schematic, the OFF pin of
Power Block 1 is connected to the positive terminal of the battery, via the on/off switch, to enable power delivery to the microcontroller at start-up. The enable signal of Power Block 2 is controlled by the microcontroller.

![Power control module schematic](image)

**Figure 10.** Power control module schematic

### 3.1.2. Hardware Integration and Configuration

Port P6.7 of the microcontroller is connected to the OFF (STBY) pin of Power Block 2, and the microcontroller is configured to turn off the GPS module at 10:00 p.m. and turn it back on at 5:00 a.m. local times. The microcontroller relies on the data received by the GPS module to determine local time.

The microcontroller’s UART0 module is configured to use ports P3.4 and P3.5, which are connected respectively to the TX and RX ports of the USB-UART interface. Similarly, the microcontroller’s UART1 module is configured to use ports P3.6 and P3.7, which are connected respectively to the TX and RX ports of the GPS.
module. To enable data transfer between modules, the microcontroller and GPS modules are both configured to communicate in UART mode at 9600 baud using a start bit, an 8-bit character, and one stop bit.

The GPS module is configured to operate in Power Save mode, in which it goes into low power state after every valid position fix, stays in low power state for 5 min., and comes out of low power state to search for satellite signals needed for the next position fix. The search for signals is limited to 15 min., after which the GPS receiver goes back into low power state. If adequate satellite signals are acquired before the 15-min. limit expires, the acquired signal is tracked for a specified time of 10 s before going into low power state. Otherwise, the GPS receiver goes into low power state for 30 min. before performing another search. Tracking the signal for 10 s before transitioning into low power state allows enough time for the GPS module to write the ephemeris and almanac data into its internal memory, thereby enabling a faster signal re-acquisition time on the next transition out of low power state.

This transition to and from low power state is autonomous and independent of controls from the microcontroller. The typical signal re-acquisition time was observed to be only 10 s. When operating in Power Save mode, the GPS module consumes 70 mA of current during satellite data acquisition and 40 mA of current during satellite tracking. In low power state, the current consumed by the module is below 0.001 µA. The GPS module was configured using u-center, version 6.10, a GPS evaluation computer program developed by u-blox AG to support their GPS modules.
3.2. Software

Microcontroller software was developed to control the hardware used in the project and process the data received by the GPS module. A data compression and decompression scheme is implemented in the software design to enable over two weeks of acquired data to fit in the flash memory of the microcontroller, thereby eliminating the need for a separate data storage module.

3.2.1. Software Development Tool

The software for the project was developed using Code Composer Essentials (CCE), version 3.2. Code Composer Essentials is a software development tool designed by Texas Instruments to support code development in the C programming language for their products, including the MSP430 family of microcontrollers. This development tool has built-in functions to facilitate code development for controlling the microcontroller peripherals, such as the clock, UART, and flash memory modules.

3.2.2. Program Flow

A high-level flow diagram of the microcontroller software implementation is shown in Figure 11. Appendix A contains the complete code listing. The first block in the flow diagram represents a required connection between the Location Tracking Device and a PC. Before the device is turned on, it is expected to be connected to a PC running a version of the HyperTerminal program. HyperTerminal is a program developed by Microsoft Corporation, Redmond, Washington, that enables serial communication between a PC.
Figure 11. Program flow diagram
and an attached device. The next block in the flow diagram represents the switching on of the device. When the device is turned on using the on/off switch, the microcontroller first initializes its peripheral modules. The clock module is configured to use the ACLK as the main program clock, while the DCO is configured to drive the flash timing generator. In addition to the clock module, the UART modules also are configured, and the UART1 module is enabled in order to allow communication between the device and the attached PC. At this point in the program flow, the GPS and UART0 modules are disabled. Once this initialization sequence is complete, the device puts a message on the HyperTerminal window requesting the user to enter a command. The microcontroller goes into low power state after the request for command is sent to the HyperTerminal program. These initialization and configuration steps listed above are represented by the first five blocks in the flow diagram.

The next two blocks in the flow diagram show how the user interacts with the device by issuing commands to it. To enter a command, the user types a command word on the computer keyboard followed by a carriage return. The only two words recognized by the device are “RESET” and “REPLAY”. The words are case sensitive. The word “RESET” is entered when the user wants to initiate new tracking, while the word “REPLAY” is entered when the user wants to retrieve any GPS data stored in memory. If any other word is entered, the microcontroller ignores it and remains in low power state.
The next four blocks following the “RESET” command in the flow diagram represent steps taken in preparation for receiving, processing, and storing location data from the GPS module. When a command to start new tracking is entered by the user, the microcontroller initializes its flash memory module and erases data in the memory cells designated for storing acquired data. The microcontroller also puts out a message on the HyperTerminal window requesting the user to enter a UTC time adjustment information. The user is expected to enter the time difference in hours between Greenwich Mean Time (GMT) and the local time. For example, in Sacramento, California, during Pacific Daylight Time (PDT), the local time is 7 hrs behind GMT, so the user is expected to enter -7. This time difference value is used by the microcontroller to convert the UTC time data received by the GPS module to local time. The time difference between GMT and local time is available online [10]. Once the time difference value is entered by the user, the microcontroller disables UART1, enables UART0, and drives a logic value of 1 on the disable signal of Power Block 2 in order to enable the GPS module.

Once the GPS module is powered on, the device operates independently and can be disconnected from the attached PC. The second block after the “RESET” command in the flow diagram shows that the device can be disconnected from the PC as soon as the user enters the UTC time adjustment data. At this time in the program flow, the device is in data acquisition phase. In this phase, the microprocessor stays in low power state until incoming data from the GPS module are detected on the UART0.
interface. The GPS module sends acquired GPRMC data to the microcontroller every 5 min., regardless of the validity of the data. Once the GPRMC data are received, the microcontroller determines the number of fields in the received GPRMC sentence and compares it to the number of fields in the standard GPRMC sentence as defined by NMEA 0183 protocol. If a mismatch is detected in the comparison, the microcontroller discards the entire sentence and goes into low power mode until the next set of data is received. If a mismatch is not detected in the comparison, the microcontroller checks the status field of the GPRMC sentence to determine if the transmitted data are valid. If the data are determined to be invalid, the microcontroller also discards the sentence and goes into low power state while waiting for the next transmission. If the data are determined to be valid, the microcontroller transitions into the data processing phase. The data acquisition and error checking processes are represented in the flow diagram in the fifth and sixth blocks following the “RESET” command.

In the data processing phase, the microcontroller separates the fields of the GPRMC sentence by referencing the comma delimiters in the sentence. The data in each field then are converted from ASCII to decimal format using built in functions of the software development tool. Once in decimal format, the speed data are converted from knots to miles per hour by multiplying the decimal value by 6076 and dividing the result by 5280. The result of the computation is rounded up to the nearest whole number in order to conserve storage space. At the beginning of the project, it was
decided that rounding the speed up to the nearest whole number was adequate for the purpose of the project.

The UTC time is converted to local time by adding the time difference value entered by the user during device initialization to the hour value of UTC time. If the addition results in a number bigger than 24, then 24 is subtracted from the result to determine local time, and the day value is incremented by 1. If the addition result is negative, then 24 is added to the result to determine local time, and the day value is decremented by 1. If the result of the addition is greater than 0 and less than, the resulting hour value and day value are left unchanged. The accuracy of this time adjustment scheme is limited to only when UTC time and local time are in the same month. There is no algorithm designed to support daylight savings time – for example, if the transition from Pacific Daylight Time to Pacific Standard Time occurs during tracking, the device does not adjust the time for such change.

Once data processing completes, the data are compressed and stored in the flash memory of the microcontroller. Prior to every data storage process, the microcontroller compares the flash memory pointer to a pre-determined value, in order to ascertain if the allocated storage memory is full. If storage memory is determined full, the microcontroller shuts down the GPS module and goes into the low power state. In the flow diagram, the data processing and storage process are represented by a single block. This block follows the error checking block.
After every flash memory write, the microcontroller compares the computed local time to 10:00 p.m. If the comparison shows that local time is earlier than 10:00 p.m., the microcontroller goes into the low power state and waits for the next set of GPRMC data from the GPS module. If the comparison shows that the local time is 10:00 p.m. or later, but not later than 5:00 a.m., the microcontroller disables the GPS module and enables Timer_A module to track the amount of time the GPS module is turned off. The GPS “off time” is determined by computing the number of hours between current time and 5:00 a.m. For example, if local time is 10:00 p.m., GPS “off time” is 7 hr, but if local time is 1:00 a.m., GPS “off time” is 4 hr. Once the GPS “off time” elapses, an interrupt signal is generated by Timer_A. Upon receiving the signal from Timer_A, the microcontroller enables the GPS module, disables Timer_A module, and goes into the low power state while waiting for the next set of data from the GPS module. This power management process is represented in the flow diagram by the last two blocks following the “RESET” command. The data acquisition, processing, and storage cycle are repeated until the device is turned off, the designated memory for data storage is full, or the battery voltage falls too low.

The GPS module requires a minimum voltage level of 3 V in order to acquire and track satellite signals. On the other hand, the microcontroller requires a minimum voltage level of 2.7 V for the flash write operation to work correctly. As the battery voltage drops with usage, the GPS module stops sending out valid GPS data first before the voltage level drops to the point where a flash write operation cannot be
performed accurately. This sequence makes the device inherently immune to writing invalid data into memory as a result of voltage drop. However, the impact of manually shutting the device down using the on/off switch at the same time the microcontroller writes to flash memory is unknown.

This implementation helped to achieve a significant reduction in overall power consumption in this project. Although the measured values of current consumed by the GPS module in acquisition and tracking modes are 73 mA and 46 mA, respectively, the daily average current consumption rate calculated with this implementation is 3.6 mA – a reduction of over 92% in GPS module current consumption during tracking. Table 2 shows the calculated average power consumption of the GPS module using the program flow discussed in this chapter.

3.2.3. Data Compression and Decompression

The microcontroller used for the project has a total of 60 KB of flash memory. The flash memory is shared among code, data, and information needed to support code execution. In this project, 20 KB of memory was reserved for the developed code, while 0.25 KB of memory was used for information. The remaining 39.75 KB of flash memory is for GPS data storage. One of the design goals in the project is to fit two weeks of received GPS data into the available 39.75 KB.

The data set transmitted by the GPS module has 15 fields. Out of the 15 fields, only seven contain data relevant to this project that need to be stored in flash memory. This data stream from the GPS module is transmitted in ASCII format. Table 3 shows
Table 2. GPS module average power consumption

<table>
<thead>
<tr>
<th>Average power requirement calculation for normal condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Power-up time</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>Current (mA)</th>
<th>On time per cycle (s)</th>
<th>Cycles per day</th>
<th>Average current per day (mA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold Start acquisition</td>
<td>73</td>
<td>900</td>
<td>1</td>
<td>0.760</td>
</tr>
<tr>
<td>Hot Start acquisition</td>
<td>73</td>
<td>10</td>
<td>204</td>
<td>1.724</td>
</tr>
<tr>
<td>Tracking</td>
<td>46</td>
<td>10</td>
<td>204</td>
<td>1.086</td>
</tr>
<tr>
<td>Sleep</td>
<td>0.001</td>
<td>280</td>
<td>204</td>
<td>0.001</td>
</tr>
<tr>
<td><strong>Total Average Current</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>3.571</strong></td>
</tr>
</tbody>
</table>

the list of GPS data values required by this project. It also shows the corresponding memory requirement for two weeks of data if stored in ASCII format. Based on the calculation results shown in Table 3, storing these data in ASCII format would require more memory than is available for data storage in the microcontroller. A conversion from ASCII to decimal format was implemented, but the resulting reduction in data size still did not allow the GPS data in the available flash memory of the microcontroller. Table 3 also shows the total memory requirement for storing two weeks of GPS data in decimal format.
To meet the data storage needs of the project with the available memory, a data compression scheme was used that relies on the fact that all captured data are not likely to change with every sample. At the sample rate of one data set every 5 min.,

Table 3. Memory required to store GPS data without compression

<table>
<thead>
<tr>
<th>Data</th>
<th>Stored as integer</th>
<th>Stored as ASCII</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Max value</td>
<td>Required memory (B)</td>
</tr>
<tr>
<td>Year</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>Month</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>Day</td>
<td>31</td>
<td>1</td>
</tr>
<tr>
<td>Hour</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>Minutes</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Latitude degrees</td>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>Latitude minutes</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Latitude seconds</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Latitude fractional seconds</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>Latitude hemisphere</td>
<td>N/S</td>
<td>1</td>
</tr>
<tr>
<td>Longitude degrees</td>
<td>180</td>
<td>1</td>
</tr>
<tr>
<td>Longitude minutes</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Longitude seconds</td>
<td>59</td>
<td>1</td>
</tr>
<tr>
<td>Longitude fractional minutes</td>
<td>99</td>
<td>1</td>
</tr>
<tr>
<td>Longitude hemisphere</td>
<td>E/W</td>
<td>1</td>
</tr>
<tr>
<td>Speed</td>
<td>200</td>
<td>1</td>
</tr>
<tr>
<td>Commas</td>
<td>16</td>
<td>0</td>
</tr>
</tbody>
</table>

Total memory per sample     | 16        | 47               |
Number of samples in two weeks | 2856     | 2856             |
Memory requirement for two weeks | 45696    | 134232           |

the year and month sections of the “Date” field can only change once during the entire tracking period. Similarly, the “Longitude” and “Latitude” fields change only slowly.
In fact, only the minute section of the “Time” field is guaranteed to change with every sample. Because all stored data do not change with every sample, storing the data once, tracking the changes in sampled data, and storing only changed data result in a significant amount of data compression. To help optimize this data compression scheme, the captured data are split up into sections that can change independently. For instance, the “Date” field is divided into year, month, and day sections, and the “Time” field is divided into hours and minutes sections. The “Longitude” and “Latitude” sections each are divided into degrees, minutes, seconds, and fractional seconds sections.

To track the changes between samples, one bit is used to represent data change from the previous sample to the current sample for each section. Two more bits are used to represent East/West hemisphere and North/South hemisphere, respectively, for the “Longitude” and “Latitude” fields. To work correctly, these change indication bits, referred to as the Preamble in this report, need to precede each set of stored data in memory. In the first Preamble written to memory, all change indication bits are written as ones, because all data are new in the first set. The first Preamble is always written to a predetermined memory location, and the first set of data is written in its entirety. For the first and subsequent writes, the compression algorithm uses the content of the Preamble to advance the flash memory pointer appropriately to avoid data overwrite. The Preamble as implemented in this project is 16 bits wide. Table 4 shows the representation of each bit of the Preamble.
Table 4. Preamble data representation

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Longitude</th>
<th>Latitude</th>
<th>Hemisphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Day</td>
<td>Month</td>
<td>Year</td>
<td>Hour</td>
<td>Minutes</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

To retrieve the stored GPS data, the user connects the device to a PC running a HyperTerminal program, turns the device on, and types in the command word “REPLAY” through the HyperTerminal window. Once the command to retrieve stored data is entered by the user, the decompression algorithm reads the first set of captured data into a read buffer and uses it as the basis for determining the value of subsequent reads. If the current Preamble indicates that a section of the data did not change, the value in the read buffer is retained as the current value. If the current Preamble indicates that a section of the stored data changed, the value in memory is used to update the read buffer with data for that section. The data in the read buffer are put out on the HyperTerminal window of the attached PC. Based on the number of changed
bits in the current Preamble, the decompression algorithm advances the flash memory pointer to the next Preamble location in memory.

In addition to allowing for efficient data compression and decompression, this algorithm detects the end of valid data in memory without using any additional memory elements. Each data set written to memory is structured such that the first 32 bits of every data set never yield a value of 0xFFFF under normal operating condition. However, the flash memory of the microcontroller erases to a logic value of 1 for each storage element. With every read from flash memory, the decompression algorithm reads the first 32 bits of data at the current flash memory pointer, prior to reading the 16-bit Preamble. When no GPS data are in memory, or when the end of data is reached during data retrieval, a value of 0xFFFF is read at the beginning of the current data set, and the microcontroller puts out a message on the HyperTerminal window to indicate the end of data.

With this compression and decompression scheme, the project was able to meet its data storage requirements using the available flash memory from the microcontroller. The amount of memory required to store two weeks of GPS data using this compression scheme is shown in Table 5. The data in Table 5 reflect a worst-case condition calculation for using the device in the United States of America. One of the two assumptions used to define the worst-case condition is that the device is in constant motion of 100 miles per hour throughout the entire tracking period. The second worst-case condition assumption is that the device is used in Alaska where the
distance for a degree change in longitude is the shortest, approximately 21 miles. The table shows that the worst-case memory requirement of 30342 is less than the allocated memory size of 39.75KB.

Table 5. Worst-case memory requirement with data compression

<table>
<thead>
<tr>
<th>Field</th>
<th>Byte</th>
<th>Limit</th>
<th>Change/day</th>
<th>Change/14 days</th>
<th>Memory (Byte)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>1</td>
<td>99</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Month</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Day</td>
<td>1</td>
<td>31</td>
<td>1</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Hour</td>
<td>1</td>
<td>24</td>
<td>17</td>
<td>238</td>
<td>238</td>
</tr>
<tr>
<td>Minutes</td>
<td>1</td>
<td>59</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Long: Deg</td>
<td>1</td>
<td>180</td>
<td>85</td>
<td>1190</td>
<td>1190</td>
</tr>
<tr>
<td>Long: Mins</td>
<td>1</td>
<td>59</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Long: Sec</td>
<td>1</td>
<td>59</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Long: Fractional sec</td>
<td>1</td>
<td>99</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Lat: Deg</td>
<td>1</td>
<td>90</td>
<td>24</td>
<td>336</td>
<td>336</td>
</tr>
<tr>
<td>Lat: Mins</td>
<td>1</td>
<td>59</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Lat: Sec</td>
<td>1</td>
<td>59</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Lat: Frac. Seconds</td>
<td>1</td>
<td>99</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Speed</td>
<td>1</td>
<td>255</td>
<td>204</td>
<td>2856</td>
<td>2856</td>
</tr>
<tr>
<td>Preable</td>
<td>2</td>
<td>16 bits</td>
<td>204</td>
<td>2856</td>
<td>5712</td>
</tr>
</tbody>
</table>

Total Worst Case Memory Requirement 30342
4.1. Battery Service Life

The microcontroller development board used for this project came with attached peripheral components not used in the design. These extra components consumed more current than what the manufacturer specified in the data sheet for the microcontroller. Due to the additional current consumed by the peripheral components on the development board, the actual service life of the 9-V battery used in the design could not be determined by testing. However, the service life of the battery was estimated based on the battery capacity, the measured current consumption of the GPS module, and the documented current consumption of the microcontroller.

To determine the current consumed by the microcontroller development board, it was disconnected from the power block and GPS module, and it was connected to a 3.3-V power supply with built-in displays for voltage and current values. The power supply was turned on, and the current drawn by the development board was recorded. The microcontroller data sheet specifies a current consumption of 340 µA in active mode and 2 µA in LPM3 mode for a supply voltage of 3.3 V. A current of 5 mA was measured when the microcontroller development board was connected to a supply voltage of 3.3 V.

The input/output current characteristic of the voltage regulator also was considered in estimating the useful service life of the battery. To determine this
input/output current characteristic, the input voltage port of the regulator was connected to a variable voltage source with built-in displays for voltage and current values, and the output voltage from the voltage regulator was used to drive the microcontroller development board. The variable power supply was turned on, and the output voltage of the variable supply was swept from 2 V to 9 V. The current delivered by the power supply was recorded, and the output voltage of the regulator under load also was recorded. This test showed that the current going into the regulator under load is approximately the current delivered by the regulator to the attached load, regardless of the voltage value at the input of the regulator. In other words, the efficiency of the regulator decreased as the input voltage increased.

Table 6 shows the estimated service life of the 9-V battery based on average daily current consumption of the device. This estimation assumes that a circuit board with only necessary components is fabricated for the project.

<table>
<thead>
<tr>
<th>Average daily current (mA)</th>
<th>Battery capacity (mAh)</th>
<th>Estimated service life</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS module</td>
<td>Microcontroller</td>
<td>Total for device</td>
</tr>
<tr>
<td>3.57</td>
<td>0.17</td>
<td>3.74</td>
</tr>
</tbody>
</table>

4.2. Tracking, Recording, and Playback

The Location Tracking Device was tested to determine whether it acquires GPS signals, processes and stores the received GPS data in flash memory, and
displays the stored data in HyperTerminal program of a PC. To perform this test, a PC first was configured to enable communication between the PC and the device. Figure 12 shows the Microsoft Windows XP HyperTerminal settings needed to communicate with the tracking device.

Figure 12. HyperTerminal window showing UART settings

After configuring the PC, the device was connected to an available USB port on the PC using an appropriate USB cable. The HyperTerminal program was launched, and then the tracking device was turned on using the on/off switch of the device. An initialization message was displayed on the HyperTerminal window,
followed by a prompt to either enter tracking mode or data retrieval mode. To initiate tracking, the word “RESET” was entered in the command prompt. The device then prompted the user for the time difference between UTC and local time, and a value of “-6” was entered. At this point, the device initialization sequence was completed, and the device was disconnected from the PC and left to operate autonomously for several hours before manually turning it off. Figure 13 shows a screen capture of the HyperTerminal window during tracking mode initialization.

![HyperTerminal window showing device initialization](image)

Figure 13. HyperTerminal window showing device initialization

To check for stored GPS data, the tracking device was connected to a PC using a USB cable. Once connected to a PC, the HyperTerminal program was launched, and
the tracking device was turned on using the on/off switch of the device. After the
prompt to select between data retrieval mode and tracking mode appeared, the word
“REPLAY” was entered in the command prompt to initiate data retrieval.

In data retrieval mode, the device displayed the stored GPS data in tabular
format in the HyperTerminal window of an attached PC. These data showed that the
device successfully tracked date, time, position, and speed. The time reported by the
device was 6 hours behind UTC time as expected, and the location data were mapped
on Google Maps to confirm the accuracy of the data capture by the device. Figure 14
is a screen capture of the HyperTerminal window showing valid GPS data displayed.

Figure 14. HyperTerminal window showing valid GPS data in memory
4.3. End of GPS Data Detection

The device also was tested to confirm that it can detect when there are no GPS data present in memory. To perform this test, the device was connected to a PC and turned on, and the command to initiate new tracking was issued. Immediately after initialization, the device was turned off manually before any GPS data could be acquired. Because device initialization erases the memory location designated for storing acquired data, no GPS data were expected to be displayed when the device was put into data retrieval mode.

To check for GPS data in memory, the device was turned back on while still connected to the PC, and the command to retrieve stored data was issued. The device successfully detected no GPS data in memory and displayed a corresponding message in the HyperTerminal window of the attached PC. Figure 15 shows a screen capture of the HyperTerminal window when a data retrieval attempt was made with no GPS data in memory.

4.4. Everyday-Living Recording

A one-day recording was completed to simulate device usage under everyday living conditions. The recording started on June 21, 2011, and completed on June 22, 2011. For this recording, I attached a strap to the device and wore it loosely around my neck during the recording period.

On the first day of the recording, I turned on and initialized the device at 7:30 a.m. in my house. I left for work at 8:10 a.m., drove an approximate distance of 22
miles, and arrived at work at 8:55 a.m. I entered into the building at my workplace and walked to my office. Due to the nature of my work, I could not wear the device around my neck while at work, so I returned the device to the car and set it on the passenger seat. I left my office at 11:30 a.m. and drove to a nearby restaurant for lunch.

![HyperTerminal window showing no GPS data in memory](image)

**Figure 15.** HyperTerminal window showing no GPS data in memory

I returned back to my office at 12:35 p.m. During the drive and while at the restaurant, I wore the device around my neck, but I left it in the car once I returned back from lunch. I left the office at 5:45 p.m. and drove home. During the drive home,
and while at home that evening, I wore the device around my neck. I removed the device from around my neck and set it down on a table in my office that night at 8:25 p.m. before going to bed. I turned the device off at 7:45 a.m. the next morning.

A review of the recorded data (Appendix B) shows that the first position fix for the test was recorded 13 min. after the device was turned on. It also shows that 36 min. elapsed between the data recorded at 9:03 a.m. and the subsequent recorded data on the first day of the test. This time corresponds to the time I walked into my office with the device around my neck, and later returned it to the car. This lapse in time occurred because the GPS receiver is configured to go into low power state for 30 min. after searching for satellite signal for a specified period. The conditions inside my office building adversely affected the reception of signals necessary to determine location. Another observation made from the recorded data is that there is a 6-minute delay between recordings at 12:04 p.m. and 12:15 p.m. on the first day of recording. This time corresponds to the time I was in the restaurant for lunch. Although the device successfully determined my location while in the restaurant, the signal acquisition time appeared to be slightly longer while in the restaurant. The recorded data also show that the device successfully put the GPS receiver into the low power state after 10:01 p.m. and put it back into active mode 7 hr later. To validate the location data recorded by the device, I mapped the first data recorded by the device on the day of the recording to Google Maps, and the data corresponded to my home location. During the drive to work on the first day of the recording, I reached a
maximum speed of 75 mph at 8:36 a.m. The device captured a speed of 73 mph at 8:38 a.m. on the same day.

The data memory consumption report generated by the device shows that 812 bytes of memory were used to store the data from the one-day recording. Based on calculations from Table 3, the same data would require 3.264 KB of memory if stored in integer format without the implemented compression scheme.
Chapter 5

SUMMARY, CONCLUSIONS, AND RECOMMENDATIONS

5.1. Summary

A location tracking device was developed to track the location of someone wearing the device for a duration of two weeks while logging the data in non-volatile memory. The device was designed to be portable and to operate autonomously on a single battery charge during the two-week tracking period.

To help achieve the required power conservation, the device was designed to disable tracking and data logging when the data are not needed. A data compression and decompression scheme was used that enabled the expected data to fit in the available memory. A Personal Computer is used in conjunction with the device only during device initialization before each tracking session and during data retrieval at the end of tracking. At other times, the device operates independently.

5.2. Conclusions

Test results showed that the features built into the project all functioned as designed. A test to determine the useful service life of the battery could not be performed; however, the service life of the battery was estimated by calculation to be 13.6 days, assuming a dedicated circuit board was designed for the project. This service life was approximately half a day less than the design target.
5.3. Recommendations

In this project, prefabricated microcontroller development, GPS module, and USB-UART interface boards were used for faster prototyping. These pre-fabricated boards made the overall size of the final project larger than necessary. The extra components of the microcontroller board also added to the overall current consumption of the device. Designing and fabricating a single board specifically for the project will reduce the overall size of the final product, as well as the current consumed by the device. The projected size of the device with a dedicated circuit board is 3 in. x 2 in. x 0.75 in.

Another area of the design that can be improved upon is the choice of voltage regulator. The efficiency of the regulator decreases as the input voltage into the regulator increases. Considering that the design uses a 9-V battery, a more efficient regulator will help extend the service life of the battery beyond 14 days.

Although the design inherently is immune to corrupting data in memory caused by voltage drop from device usage, the behavior of the device when it is shut down at the same time it is writing to flash memory is unknown. To prevent the microcontroller from writing to memory at the time the device is shut down, a voltage level monitoring scheme could be implemented to detect device shut down and halt write access to memory.

Furthermore, the user interface has room for improvement. Rather than a HyperTerminal window, a LabVIEW program can be developed specifically for
communicating with the device using computer mouse clicks. To help simplify post-
processing of collected data, the LabVIEW program can be designed such that the
logged GPS data can be mapped automatically to an electronic atlas such as Google
Maps. The LabVIEW program also can be developed to handle the time conversion
from UTC time to local time, rather than doing the conversion in the microcontroller
program.
APPENDIX A

Flow Control Code

```c
#include <msp430x16x.h>
#include <string.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>

#define SAME_DAY        (0x8000)
#define SAME_MONTH       (0x4000)
#define SAME_YEAR         (0x2000)
#define SAME_HOUR        (0x1000)
#define SAME_MINS        (0x0800)
#define SAME_LONG_DEG   (0x0400)
#define SAME_LONG_MINS  (0x0200)
#define SAME_LONG_SEC   (0x0100)
#define SAME_LONG_SEC_F (0x0080)
#define SAME_LAT_DEG    (0x0040)
#define SAME_LAT_MINS   (0x0020)
#define SAME_LAT_SEC     (0x0010)
#define SAME_LAT_SEC_F  (0x0008)
#define SAME_SPEED       (0x0004)
#define NORTH            (0x0002)
#define EAST             (0x0001)

#define gps             P6OUT
#define on              (0x80)
#define off             (0x00)
#define max_memory_limit (0xF9FF)
#define shutdown_time    22
#define startup_time     5

unsigned short time_mins, date_year, date_month, date_day,
             speed  = 0;
unsigned short long_deg, long_mins, long_sec, long_sec_fract = 0;
unsigned short lat_deg, lat_mins, lat_sec, lat_sec_fract = 0;

unsigned short prev_time_mins, prev_date_year, prev_date_month,
             prev_date_day, prev_speed = 0;
unsigned short prev_long_deg, prev_long_mins, prev_long_sec,
             prev_long_sec_fract = 0;
unsigned short prev_lat_deg, prev_lat_mins, prev_lat_sec,
             prev_lat_sec_fract = 0;
short time_hour, prev_time_hour = 0;
```
unsigned short initial_flag;
unsigned int data_config;
int time_adjustment_hrs = -7;
unsigned int gps_timer = 0;

char rx_buffer[85];
char *field[13];
char *ram_ptr = rx_buffer;
char *flash_ptr = (char *) 0x6200;
char *flash_base = (char *) 0x6200;

unsigned short write_size;
unsigned short field_counter;
unsigned short mode = 0;
volatile unsigned char *uart_rx_buf;

void initialize (void);
void display (char message[]);
void clock_delay (unsigned int delay);
void process_input (void);
void flash_replay (void);
void device_reset (void);
void extract_data (void);
void write_data (void);
void flash_write (void);
void gps_shutdown(void);
void gps_startup(void);

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;
P6DIR |= 0x80;
    initialize();

    _BIS_SR(LPM3_bits + GIE); // Enter LPM3 w/
    interrupt
}

#pragma vector=DACDMA_VECTOR
__interrupt void DACDMA_ISR (void)
{
    if (*uart_rx_buf == '\n')
    {
        ram_ptr = rx_buffer;
        if (strncmp(ram_ptr,"REPLAY",6) == 0) flash_replay();
        else if (strncmp(ram_ptr,"RESET",5) == 0) device_reset();
        else if (mode == 1) device_reset();
    }
else if (strncmp(ram_ptr,"$GPRMC",6) == 0)
{
    process_input();
    if (field_counter != 12)
    {
        DMA1DA = (unsigned int)&rx_buffer;
    }
    else if (*field[2] != 'A')
    {
        DMA1DA = (unsigned int)&rx_buffer;
    }
    else
    {
        extract_data();
        write_data();
    }
}
DMA1DA = (unsigned int)&rx_buffer;

DMA1CTL &= ~DMAIFG;
DMA1CTL |= DMAEN;
if ((time_hour >= shutdown_time) || (time_hour < startup_time))
gps_shutdown();

// Timer B0 interrupt service routine
#pragma vector TIMERB0_VECTOR
__interrupt void Timer_B (void)
{
    if (gps_timer == 0) gps_startup();
    else {
        gps_timer --;
        TBCTL = TBCLR;
        TBCCTL0 = CCIE;
        TBCCRO = 0xF000;
        TBCTL = 0x01F2;
    }
}

void initialize (void)
{
    gps = off;
}
UCTL1 = SWRST;
ME2 &= 0xCF;
UCTL0 = SWRST;
ME1 &= 0x3F;
P3SEL &= 0x00;
P3DIR &= 0x00;
P3SEL = 0x30;
ME1 |= URXE0 + UTXE0;
UCTL0 |= CHAR;
UTCTL0 |= SSELO;
UBR00 = 0x03;
UBR10 = 0x00;
UMCTL0 = 0x4A;
UCTL0 &= ~SWRST;
DMA1CTL &= 0xFFEF;
DMA0CTL &= 0xFFEF;
DMACTL0 = DMA1TSEL_3 + DMA0TSEL_4;

DMA0SA = (unsigned int)flash_base;
DMA0DA = (unsigned int)&U0TXBUF;
DMA0SZ = 0x01;
DMA0CTL = DMASRCINCR_3 + DMASBDB + DMALEVEL;

DMA1SA = (unsigned int)&U0RXBUF;
DMA1DA = (unsigned int)&rx_buffer;
DMA1SZ = 0x01;
DMA1CTL &= ~DMAIFG;
DMA1CTL = DMADSTINCR_3 + DMASBDB + DMAIE + DMAEN;
DMA1CTL &= ~DMAIFG;

uart_rx_buf = &U0RXBUF;

FCTL2 = FWKEY + FSSEL0 + FN0;
field_counter = 0;
initial_flag = 1;

clock_delay(50000);

display ("\r\n---Initialization Complete. Waiting for
Command---\n\n"");
display ("REPLAY: Retrieve stored tracking data\n\nRESET :
Reset and initialize device for new tracking\n\n"));

mode = 0;
}

void display (char message[])
unsigned int index, message_length;
char *display_ptr = (char*)message;
message_length = strlen(message);

for (index = 0; index <= message_length; index++)
{
    DMA0SA = (unsigned int)display_ptr;
    DMA0CTL |= DMAEN;
    clock_delay (800);
    display_ptr ++;
}

void clock_delay (unsigned int delay)
{
    if (delay > 0)
    {
        do (delay--);
        while (delay != 0);
    }
}

void process_input (void)
{
    field_counter = 0;
    field[field_counter] = ram_ptr;
    while (*ram_ptr != '\n')
    {
        if (*ram_ptr == ',')
        {
            *ram_ptr = '\0';
            field_counter++;
            field[field_counter] = ram_ptr + 1;
        }
        ram_ptr ++;
    }
    *ram_ptr = '\0';
}

void flash_replay (void)
{
    char write_buffer[12];
    char temp_buffer[9];
    unsigned long end_check;
    unsigned int *config_ptr ;
    unsigned long *end_check_ptr;
flash_ptr = flash_base;
end_check_ptr = (unsigned long *) &temp_buffer;
config_ptr = (unsigned int *) &temp_buffer;

display("\n\r");

strncpy(temp_buffer,flash_ptr,8);
end_check = *end_check_ptr;

if (end_check == 0xFFFFFFFF)
{
    display ("--No valid data found in memory--
\n\n\r");
    return;
}

display("DATE-D/M/Y\tTIME\tLONGITUDE\t	LATITUDE\t	MPH\n\r");
data_config = *config_ptr;
flash_ptr += 2;
while (end_check != 0xFFFFFFFF)
{
    if (((data_config & SAME_DAY) != 0) // changed
    {
        date_day = (unsigned short) *flash_ptr;
        flash_ptr ++;
    }
    if (((data_config & SAME_MONTH) != 0) // changed
    {
        date_month = (unsigned short) *flash_ptr;
        flash_ptr ++;
    }
    if (((data_config & SAME_YEAR) != 0) // changed
    {
        date_year = (unsigned short) *flash_ptr;
        flash_ptr ++;
    }
    if (((data_config & SAME_HOUR) != 0) // changed
    {
        time_hour = (unsigned short) *flash_ptr;
        flash_ptr++;
    }
    if (((data_config & SAME_MINS) != 0) // changed
    {
        time_mins = (unsigned short) *flash_ptr;
        flash_ptr++;
    }
}
if ((data_config & SAME_LONG_DEG) != 0) // changed
{
    long_deg = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LONG_MINS) != 0) // changed
{
    long_mins = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LONG_SEC) != 0) // changed
{
    long_sec = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LONG_SEC_F) != 0) // changed
{
    long_sec_fract = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LAT_DEG) != 0) // changed
{
    lat_deg = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LAT_MINS) != 0) // changed
{
    lat_mins = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LAT_SEC) != 0) // changed
{
    lat_sec = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_LAT_SEC_F) != 0) // changed
{
    lat_sec_fract = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
if ((data_config & SAME_SPEED) != 0) // changed
{
    speed = (unsigned short) *flash_ptr;
    flash_ptr ++;
}
sprintf(write_buffer, "%d/", date_day);
display (write_buffer);

sprintf(write_buffer, "%d/", date_month);
display (write_buffer);

sprintf(write_buffer, "%d\t\t", date_year);
display (write_buffer);

sprintf(write_buffer, "%d:\", time_hour);
display (write_buffer);

sprintf(write_buffer, "%d\t", time_mins);
display (write_buffer);

sprintf(write_buffer, "%d ", long_deg);
display (write_buffer);

sprintf(write_buffer, "%d ", long_mins);
display (write_buffer);

sprintf(write_buffer, "%d.", long_sec);
display (write_buffer);

sprintf(write_buffer, "%d", long_sec_fract);
display (write_buffer);

if ((data_config & EAST) != 0) display("E\t\t");
else display("W\t\t");

sprintf(write_buffer, "%d ", lat_deg);
display (write_buffer);

sprintf(write_buffer, "%d ", lat_mins);
display (write_buffer);

sprintf(write_buffer, "%d.", lat_sec);
display (write_buffer);

sprintf(write_buffer, "%d", lat_sec_fract);
display (write_buffer);

if ((data_config & NORTH) != 0) display("N\t\t");
else display("S\t\t");

sprintf(write_buffer, "%d\n\r", speed);
display (write_buffer);
strncpy(temp_buffer,flash_ptr,8);
end_check = *end_check_ptr;
data_config = *config_ptr;

flash_ptr += 2;
}
data_config = (flash_ptr - flash_base);
data_config --;
sprintf (write_buffer,"\n\r%d ", data_config);
display(write_buffer);
display ("Bytes used\n\r");
clock_delay(500000);
initialize();
}

void device_reset (void)
{
char buffer_1[40];
short time_adj_sign = 1;

if (mode == 0)
{
    display("\n\rEnter UTC Time Adjustment in hours. Example: 3, -7, 22, etc\n\r>");
    mode = 1;
    return;
}
else if (mode == 1)
{
    ram_ptr = rx_buffer;
    if (*ram_ptr == '-')
    {
        ram_ptr ++;
        time_adj_sign = -1;
    }
    time_adjustment_hrs = atoi(ram_ptr);
time_adjustment_hrs *= time_adj_sign;
if (time_adjustment_hrs >= 24)
{
display ("Invalid value entered. Enter a UTC Time Adjustment value between -23 to +23\r\n");
return;
}

if (time_adjustment_hrs <= -24)
{
    display ("Invalid value entered. Enter a UTC Time Adjustment value between -23 to +23\r\n");
    return;
}
ram_ptr = rx_buffer;
else return;

display ("---Reseting Device...---\n\n");
flash_ptr = flash_base;
FCTL3 = FWKEY;
while ((unsigned int)flash_ptr < max_memory_limit)
{
    FCTL1 = FWKEY + ERASE;
    *flash_ptr = 0;
    flash_ptr = flash_ptr + 512;
}
flash_ptr = flash_base;
FCTL1 = FWKEY;
FCTL3 = FWKEY + LOCK;
sprintf(buffer_1," Start-up Time : %d:00\n\r",startup_time);
display (buffer_1);
sprintf(buffer_1," Shut-down Time : %d:00\n\r",shutdown_time);
display (buffer_1);
sprintf(buffer_1," UTC Time Adjustment: %d hrs\n\n\r",time_adjustment_hrs);
display (buffer_1);

display ("---GPS Module enabled---\n\r");

UCTL0 = SWRST;
ME1 &= 0x3F;
UCTL1 = SWRST;
ME2 &= 0xCF;
P3SEL &= 0x00;
P3DIR &= 0x00;
P3SEL = 0xC0;
ME2 |= URXE1 + UTXE1;
UCTL1 |= CHAR;
UTCTL1 |= SSEL0;
UBR01 = 0x03;
UBR11 = 0x00;
UMCTL1 = 0x4A;
UCTL1 &= ~SWRST;

DMA1CTL &= 0xFFEF;
DMA0CTL &= 0xFFEF;

DMACTL0 = DMA1TSEL_9 + DMA0TSEL_10;
DMA0SA = (unsigned int)flash_base;
DMA0DA = (unsigned int)&UTXBUF;
DMA0SZ = 0x01;
DMA0CTL = DMASRCINCR_3 + DMASBDB + DMALEVEL;
DMA1SA = (unsigned int)&U1RXBUF;
DMA1DA = (unsigned int)&rx_buffer;
DMA1SZ = 0x01;
DMA1CTL &= ~DMAIFG;
DMA1CTL = DMADSTINCR_3 + DMASBDB + DMAIE + DMAEN;

uart_rx_buf = &U1RXBUF;
mode = 2;
gps = on;
}

void extract_data (void)
{
    unsigned long temp_int;
    short temp_int2;
    unsigned int temp_int3;
    float temp_float;
    char* token;
    data_config = 0;
    temp_int = 0;
    temp_int = atol(field[1]);
    time_hour = (temp_int / 10000);
    temp_int = (temp_int % 10000);
    time_mins = (temp_int / 100);
    token = strtok (field[3], ".");
    temp_int3 = atoi(token);
    lat_deg = (temp_int3/100);
    lat_mins = (temp_int3 % 100);
    token = strtok (NULL, ".");

    if (token == NULL)
\{
    lat_sec = 0;
    lat_sec_fract = 0;
\}

else {
    temp_int = atol(token);
    temp_int = (temp_int * 60);
    lat_sec = (temp_int / 100000);
    temp_int = (temp_int % 100000);
    lat_sec_fract = (temp_int / 1000);
}

if (*field[4] == 'N') data_config |= NORTH;

    token = strtok (field[5], ".");
    temp_int3 = atoi(token);
    long_deg = (temp_int3/100);
    long_mins = (temp_int3 % 100);
    token = strtok (NULL, ".");

if (token == NULL)
{
    long_sec = 0;
    long_sec_fract = 0;
}

else {
    temp_int = atol(token);
    temp_int = (temp_int * 60);
    long_sec = (temp_int / 100000);
    temp_int = (temp_int % 100000);
    long_sec_fract = (temp_int / 1000);
}

if (*field[6] == 'E') data_config |= EAST;

    temp_float = atof(field[7]);
    temp_float = (temp_float*6076/5280);
    speed = (unsigned short) (temp_float + 0.5);
    temp_int = atol(field[9]);
    date_day = (temp_int / 10000);
    temp_int = (temp_int % 10000);
    date_month = (temp_int / 100);
    date_year = (temp_int % 100);
    temp_int2 = time_hour + time_adjustment_hrs;

if (temp_int2 < 0)
{
    \{
time_hour = (24 + temp_int2);
date_day --;
}
else if (temp_int2 > 23)
{
    time_hour = (temp_int2 % 24);
    date_day ++;
}
else time_hour += time_adjustment_hrs;

void write_data (void)
{
    if (mode != 2) return;

    if (date_day != prev_date_day) data_config |= SAME_DAY;
    if (date_month != prev_date_month) data_config |= SAME_MONTH;
    if (date_year != prev_date_year) data_config |= SAME_YEAR;
    if (time_hour != prev_time_hour) data_config |= SAME_HOUR;
    if (time_mins != prev_time_mins) data_config |= SAME_MINS;
    if (long_deg != prev_long_deg) data_config |= SAME_LONG_DEG;
    if (long_mins != prev_long_mins) data_config |= SAME_LONG_MINS;
    if (long_sec != prev_long_sec) data_config |= SAME_LONG_SEC;
    if (long_sec_fract != prev_long_sec_fract) data_config |= SAME_LONG_SEC_F;
    if (lat_deg != prev_lat_deg) data_config |= SAME_LAT_DEG;
    if (lat_mins != prev_lat_mins) data_config |= SAME_LAT_MINS;
    if (lat_sec != prev_lat_sec) data_config |= SAME_LAT_SEC;
    if (lat_sec_fract != prev_lat_sec_fract) data_config |= SAME_LAT_SEC_F;
    if (speed != prev_speed) data_config |= SAME_SPEED;

    if (initial_flag != 0)
    {
        data_config |= SAME_YEAR;
        data_config |= SAME_MONTH;
        data_config |= SAME_DAY;
        data_config |= SAME_HOUR;
        data_config |= SAME_MINS;
        data_config |= SAME_LONG_DEG;
        data_config |= SAME_LONG_MINS;
        data_config |= SAME_LONG_SEC;
        data_config |= SAME_LONG_SEC_F;
        data_config |= SAME_LAT_DEG;
        data_config |= SAME_LAT_MINS;
        data_config |= SAME_LAT_SEC;
        data_config |= SAME_LAT_SEC_F;
data_config |= SAME_SPEED;
initial_flag = 0;
}

ram_ptr = (char*)&data_config;
write_size = 2;
flash_write();

if ((data_config & SAME_DAY) != 0)
{
    ram_ptr = (char *)&date_day;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_MONTH) != 0)
{
    ram_ptr = (char *)&date_month;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_YEAR) != 0)
{
    ram_ptr = (char *)&date_year;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_HOUR) != 0)
{
    ram_ptr = (char *)&time_hour;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_MINS) != 0)
{
    ram_ptr = (char *)&time_mins;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LONG_DEG) != 0)
{
    ram_ptr = (char *)&long_deg;
    write_size = 1;
flash_write();
}

if ((data_config & SAME_LONG_MINS) != 0)
{
    ram_ptr = (char *)&long_mins;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LONG_SEC) != 0)
{
    ram_ptr = (char *)&long_sec;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LONG_SEC_F) != 0)
{
    ram_ptr = (char *)&long_sec_fract;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LAT_DEG) != 0)
{
    ram_ptr = (char *)&lat_deg;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LAT_MINS) != 0)
{
    ram_ptr = (char *)&lat_mins;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LAT_SEC) != 0)
{
    ram_ptr = (char *)&lat_sec;
    write_size = 1;
    flash_write();
}

if ((data_config & SAME_LAT_SEC_F) != 0)
ram_ptr = (char *)&lat_sec_fract;
write_size = 1;
flash_write();
}

if ((data_config & SAME_SPEED) != 0)
{
    ram_ptr = (char *)&speed;
    write_size = 1;
    flash_write();
}

prev_time_hour = time_hour;
prev_time_mins = time_mins;
prev_date_year = date_year;
prev_date_month = date_month;
prev_date_day = date_day;

prev_long_deg = long_deg;
prev_long_mins = long_mins;
prev_long_sec = long_sec;
prev_long_sec_fract = long_sec_fract;

prev_lat_deg = lat_deg;
prev_lat_mins = lat_mins;
prev_lat_sec = lat_sec;
prev_lat_sec_fract = lat_sec_fract;

prev_speed = speed;
ram_ptr = rx_buffer;
}

void flash_write (void)
{
if ((unsigned int)flash_ptr >= max_memory_limit)
{
    time_hour = 0;
gps = off;
return;
}

FCTL1 = FWKEY + WRT;
FCTL3 = FWKEY;
while (write_size > 0)
{

*flash_ptr = *ram_ptr;
flash_ptr++;
ram_ptr++;
write_size--;
}
FCTL1 = FWKEY;
FCTL3 = FWKEY + LOCK;
}

void gps_shutdown(void)
{
if (mode == 2)
{
    gps = off;
    time_hour = 0;
    mode = 3;
    gps_timer = 0;
    TBCTL = TBCLR;
    TBCCTL0 = CCIE;
    TBCCR0 = 0xF000;
    TBCTL = 0x01F2;
    if (time_hour > shutdown_time)
    {
        gps_timer = 24 - shutdown_time;
        gps_timer += startup_time;
    }
    else
    {
        gps_timer = startup_time - time_hour;
    }
    gps_timer *= 240;
    gps_timer -= 1;
}
}

void gps_startup(void)
{
    mode = 2;
    TBCTL = TBCLR;
    TBCCTL0 = 0;
    gps = on;
}
**APPENDIX B**

Everyday-Living Recording Results

---Initialization Complete. Waiting for Command---

REPLAY: Retrieve stored tracking data

RESET: Reset and initialize device for new tracking

> REPLAY

<table>
<thead>
<tr>
<th>DATE-D/M/Y</th>
<th>TIME</th>
<th>LONGITUDE</th>
<th>LATITUDE</th>
<th>MPH</th>
</tr>
</thead>
<tbody>
<tr>
<td>21/6/11</td>
<td>7:43</td>
<td>121 22 26.32 W</td>
<td>38 26 26.32 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>7:48</td>
<td>121 22 26.32 W</td>
<td>38 26 26.44 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>7:53</td>
<td>121 22 26.23 W</td>
<td>38 26 26.59 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>7:58</td>
<td>121 22 26.23 W</td>
<td>38 26 26.59 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8: 3</td>
<td>121 22 26.23 W</td>
<td>38 26 26.59 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8: 8</td>
<td>121 22 26.38 W</td>
<td>38 26 26.35 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:13</td>
<td>121 22 26.71 W</td>
<td>38 26 26.77 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:18</td>
<td>121 22 26.68 W</td>
<td>38 26 26.55 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:23</td>
<td>121 21 6.40 W</td>
<td>38 27 9.55 N</td>
<td>50</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:28</td>
<td>121 20 5.37 W</td>
<td>38 29 44.28 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:33</td>
<td>121 20 7.26 W</td>
<td>38 32 58.65 N</td>
<td>51</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:38</td>
<td>121 17 8.10 W</td>
<td>38 35 30.79 N</td>
<td>73</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:43</td>
<td>121 11 58.72 W</td>
<td>38 38 19.69 N</td>
<td>36</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:48</td>
<td>121 9 57.90 W</td>
<td>38 38 44.23 N</td>
<td>5</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:53</td>
<td>121 9 53.16 W</td>
<td>38 38 40.11 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>8:58</td>
<td>121 9 52.66 W</td>
<td>38 38 40.59 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9: 3</td>
<td>121 9 52.53 W</td>
<td>38 38 40.87 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9:39</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9:44</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9:49</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9:54</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>9:59</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10: 4</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10: 9</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:14</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:19</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:24</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:29</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:34</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:39</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:44</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:49</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:54</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>10:59</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11: 4</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11: 9</td>
<td>121 9 52.81 W</td>
<td>38 38 40.75 N</td>
<td>0</td>
</tr>
<tr>
<td>Date</td>
<td>Time</td>
<td>Longitude</td>
<td>Latitude</td>
<td>Weather</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:14</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:19</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:24</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:29</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:34</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:39</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:44</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>11:49</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:04</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:09</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:14</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:19</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:24</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:29</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:34</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:39</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:44</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>12:49</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:04</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:09</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:14</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:19</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:24</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:29</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:34</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:39</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:44</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:49</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:54</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>13:59</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:04</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:09</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:14</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:19</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:24</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:29</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:34</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:39</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:44</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>14:49</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:04</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:09</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:14</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:19</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:24</td>
<td>121° 21'</td>
<td>38° 40.75''</td>
<td>0</td>
</tr>
<tr>
<td>Date</td>
<td>Time</td>
<td>Latitude</td>
<td>Longitude</td>
<td>Temperature</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:06</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:11</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:16</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:21</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:26</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:31</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:36</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:41</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:46</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:51</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>15:56</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:01</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:06</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:11</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:16</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:21</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:26</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:31</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:36</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:41</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:46</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:51</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>16:56</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:01</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:06</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:11</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:16</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:21</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:26</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:31</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:36</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:41</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:46</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:51</td>
<td>121°9'</td>
<td>50°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>17:56</td>
<td>121°12'</td>
<td>18°30' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:01</td>
<td>121°11'</td>
<td>17°11.82' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:06</td>
<td>121°20'</td>
<td>10°96' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:11</td>
<td>121°20'</td>
<td>5°95' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:16</td>
<td>121°20'</td>
<td>26°58' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:21</td>
<td>121°22'</td>
<td>18°60' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:26</td>
<td>121°22'</td>
<td>26°64' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:31</td>
<td>121°22'</td>
<td>26°64' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:36</td>
<td>121°22'</td>
<td>26°64' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:41</td>
<td>121°22'</td>
<td>26°31' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:46</td>
<td>121°22'</td>
<td>26°49' W</td>
<td>38</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:51</td>
<td>121°22'</td>
<td>26°49' W</td>
<td>38</td>
</tr>
<tr>
<td>Date</td>
<td>Time</td>
<td>Longitude</td>
<td>Latitude</td>
<td>Temperature</td>
</tr>
<tr>
<td>------</td>
<td>-------</td>
<td>------------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>21/6/11</td>
<td>18:56</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:01</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:06</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:11</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:16</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:21</td>
<td>121 22 26.49 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:26</td>
<td>121 22 26.41 W</td>
<td>38 26 26.44 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:31</td>
<td>121 22 26.46 W</td>
<td>38 26 26.40 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:36</td>
<td>121 22 26.38 W</td>
<td>38 26 26.56 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:41</td>
<td>121 22 26.50 W</td>
<td>38 26 26.56 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:46</td>
<td>121 22 26.50 W</td>
<td>38 26 26.56 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:51</td>
<td>121 22 26.28 W</td>
<td>38 26 26.50 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>19:56</td>
<td>121 22 26.46 W</td>
<td>38 26 26.44 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:01</td>
<td>121 22 26.46 W</td>
<td>38 26 26.44 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:06</td>
<td>121 22 26.47 W</td>
<td>38 26 26.50 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:11</td>
<td>121 22 26.59 W</td>
<td>38 26 26.67 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:16</td>
<td>121 22 26.38 W</td>
<td>38 26 26.47 N</td>
<td>1</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:21</td>
<td>121 22 26.29 W</td>
<td>38 26 26.41 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:26</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:31</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:36</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:41</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:46</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:51</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>20:56</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:01</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:06</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:11</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:16</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:21</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:26</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:31</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:36</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:41</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:46</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:51</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>21:56</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>21/6/11</td>
<td>22:01</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:12</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:17</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:22</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:27</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:32</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:37</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:42</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
<tr>
<td>22/6/11</td>
<td>5:47</td>
<td>121 22 26.41 W</td>
<td>38 26 26.64 N</td>
<td>0</td>
</tr>
</tbody>
</table>
22/6/11 5:52 121 22 26.41 W 38 26 26.64 N 0
22/6/11 5:57 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:02 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:07 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:12 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:17 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:22 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:27 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:32 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:37 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:42 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:47 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:52 121 22 26.41 W 38 26 26.64 N 0
22/6/11 6:57 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:02 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:07 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:12 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:17 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:22 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:27 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:32 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:37 121 22 26.41 W 38 26 26.64 N 0
22/6/11 7:42 121 22 26.41 W 38 26 26.64 N 0

812 Bytes used

---Initialization Complete. Waiting for Command---
REPLAY: Retrieve stored tracking data
RESET : Reset and initialize device for new tracking
>


REFERENCES


