Erebus Labs

# *STEM Sensors*

# Design Specification

Version 1*.*5

*3/31/2014*

# scott lawson

# Bryan Button

# Chris clary

# max cope

# VERSION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Reason** |
| 1.5 | Scott Lawson | 3/31/2014 | Added interface mode flowchart  Added error messages |
| 1.4 | Scott Lawson | 3/30/2014 | Updated 3.1 IRQ information  Updated 3.2 Interface Mode details  Adjusted page numbering to start on Table of Contents |
| 1.3 | Scott Lawson | 3/3/2014 | Added 3.2.3 Virtual COM Port  Added 3.2.4 Interface Mode  Updated 3.1.5 with VBUS details |
| 1.2 | Scott Lawson | 2/1/2014 | Fixed header typos  Added Engineering Requirements to Introduction |
| 1.1 | Scott Lawson | 1/30/2014 | Changed data sample format  Changed Appendix A name to “Glossary” from “Terminology” |
| 1.0 | Scott Lawson | 1/28/2014 | Initial Release  Converted from Software Plan V1.0 |

**NOTE TO READER**

This is a template obtained from:

<http://www2.cdc.gov/cdcup/library/templates/default.htm>

Template Name: Product Design

**UP Template Version:** 12/31/07

# TABLE OF CONTENTS

1 Introduction 2

1.1 Purpose of The Document 2

1.2 Overview 2

Objective Statement 2

Theory of Operation 2

1.3 Engineering Requirements 3

2 Hardware Plan 4

2.1 Overview 4

2.2 Block Diagrams 4

Base Unit Level 0 Block Diagram 4

Base Unit Level 1 Block Diagram 5

Base Unit Level 2 Block Diagram 6

PSoC3 Block Diagram 7

2.3 Implementation 8

Microcontroller 8

Sensor Design 8

Power Supply 8

3 Software Plan 9

3.1 Firmware 9

Firmware Overview 9

Initialization Tasks 10

Interrupt Service Routines 10

LowV\_IRQ ISR 12

VBus\_IRQ ISR 13

Data Samples 14

3.2 Computer Interface 16

User Interface 16

Program Installer 16

Virtual COM Port 16

Interface Mode 17

Appendix A: Glossary 20

Acronyms 20

System Architecture 21

# Introduction

## Purpose of The Document

This document describes plans for both the hardware and software components of the Erebus Labs STEM Sensor. It does not include any design files such as schematics, layout files, or source code. Rather, it is intended to guide the team members in their creation of such files and describe how various system components interact with each other.

## Overview

### Objective Statement

Encourage an interest in STEM in K-12 students by delivering a working prototype of an affordable, simple and flexible device to collect environmental data.

### Theory of Operation

The Erebus Labs STEM Sensor system is an open-source electronic device for collecting environmental data over a period of time and presenting it for analysis. The system is comprised of the following components:

**Base Unit**

The central device that manages power, communication, and data storage, and has one or more sensors attached to it.

**Sensor**

The individual data collection devices such as VOC detectors and thermometers that are attached to the base unit.

**User Interface**

The program that will be run on a laptop or desktop computer that allows the user to view and interact with the data collected.

The base unit will have one sensor attached to it and will passively collect data without being attached to a computer system. The data collection site will be chosen by the user. The user interface will be a simple GUI for displaying collected data and exporting the data to a CSV file for analysis with a third-party program.

## Engineering Requirements

See proposal for marketing requirements.

|  |  |  |
| --- | --- | --- |
| **Marketing Requirements** | **Engineering Requirements** | **Justification** |
| **1, 2, 3, 4** | All sensors must use the same interface to connect to the base unit | Minimizes cost and complexity for users while increasing versatility |
| **4** | The user interface must provide a method for the user to access the raw data collected | Allows advanced users to perform their own data analysis |
| **7** | A publicly-accessible repository must be used for code and documentation hosting | Encourages exploration and experimentation by students |
| **2, 7** | If third-party software is used, it must be open-source | Encourages exploration and experimentation by students, minimizes cost |
| **5, 12** | The base unit with sensors attached must operate when exposed to temperatures between -10°C and +70°C | Temperature range required for outdoor operation |
| **2, 3** | If the system is does not use a rechargeable power source, it must not use proprietary battery types | Using widely available batteries minimizes cost |
| **2** | BOM for base unit should not exceed $20.00 each | Necessary for adoption by K-12 classrooms with limited budgets |
| **2** | BOM for sensors should not exceed $5.00 each | Necessary for adoption by K-12 classrooms with limited budgets |
| **3, 4, 9** | The base unit should identify the sensor(s) attached and configure itself appropriately | Simplifies operation for younger users |
| **1, 4, 6** | The system should be able to collect data points at rates between 1 Hz and 1 per day | Accommodates a wide variety of data collection applications |
| **4, 10** | The system should be able to coordinate data collection between 6 base units simultaneously | Accommodates a wide variety of data collection applications |
| **5, 12** | The base units and sensors should be operational after a 1.5m drop-test | The system needs to survive daily use by K-12 students |
| **5, 12** | The base unit with sensors attached should operate when exposed to temperatures between -20°C and +80°C | Temperature range suggested for outdoor operation |
| **3, 8** | The base unit should be able to collect data points for 90 days without user interaction | Simplifies operation for all users |
| **11** | A wireless data dump interface should be utilized by the base unit | Provides a convenient method for users to retrieve data |
| **11** | If a wireless data dump interface is utilized by the base unit, it should not require the user to be closer to the base unit than 3 meters | Provides a convenient method for users to retrieve data |
| **5, 12** | The base and sensors may be constructed with a water-resistant case | The system needs to survive daily use by K-12 students |
| **2, 4, 14** | The base unit may contain multiple attachment points to enable multiple sensors to be used simultaneously | Enhances versatility for advanced data collection |
| **3, 13** | The base unit may use sockets and connectors to attach the controller, power, and communications devices to the PCB | Further modularity provides hardware interactivity and learning opportunities for younger users |

# Hardware Plan

## Overview

The base unit will be comprised of a Cypress PSoC 3 microcontroller, a voltage regulator, a sensor interface and a USB bus, along with miscellaneous passive and capacitive components.

## Block Diagrams

### Base Unit Level 0 Block Diagram



### Base Unit Level 1 Block Diagram



### Base Unit Level 2 Block Diagram



### 2014-01-29-020249_3280x1080_scrotPSoC3 Block Diagram

## Implementation

### Microcontroller

The PSoC3 microcontroller from Cypress Semiconductor was selected because of its balance between flexibility and cost. It contains several embedded programmable logic blocks that can be used to implement a Full-Speed USB controller and real-time clock. Additionally, the programmable blocks also provide an interface for utilizing the chip’s on-board Flash memory in place of an external EEPROM chip. Therefore, the PSoC3 provides a one-chip solution.

|  |  |
| --- | --- |
| Manufacturer | Cypress Semiconductor |
| Family | PSoC3 |
| Model Number | CY8C3245PVI-150 |
| Architecture | 8-bit 8051 |
| Clock Speed | 48MHz |
| Operating Voltage | 1.71V – 5.5V |
| Current Draw | 0.8mA@3MHz, 1.2mA@6MHz, 6.6mA@48MHz |
| ADC | 12-bit Delta-Sigma |
| Program Memory | 32KB Flash |
| EEPROM | 1KB |
| UDBs | 20 |
| Package | 48-pin SSOP |

### Sensor Design

No sensor data available at this time.

### Power Supply

No power supply data available at this time.

# Software Plan

## Firmware

### Firmware Overview

The Erebus Labs STEM Sensors device firmware will be written in standard C compliant with ISO/IEC 9899:2011 and developed in Cypress’ PSoC Creator 3.0. Upon reset, the controller will operate as follows:



The STEM Sensor’s MAIN function will be a loop that only contains a command that puts the chip to sleep. All functionality will be interrupt-driven. The system will remain in sleep mode except when executing an interrupt service routine.

### Initialization Tasks

The following initialization tasks are performed upon reset:

Allocate array in flash for data samples

Place flash array tail pointer at end of previously collected data block

Start EEPROM component

Retrieve and apply user variables from EEPROM

Enable VBus\_IRQ, LowV\_IRQ, SampleStart\_IRQ, and Hibernate\_IRQ interrupts individually

Enable Global interrupts

### Interrupt Service Routines

All functionality will be handled by the following interrupt service routines:

|  |  |  |
| --- | --- | --- |
| **Name** | **Trigger** | **Action** |
| VBus\_IRQ | Logic High voltage on VBUS pin | Activate USB Component, eumerate device on host, receive and store user settings, dump data samples to host, software reset on exit |
| TakeSample\_IRQ | Periodic interrupt | Activate ADC component, take a sample of the ADC voltage, store in Flash, deactivate ADC, strobe “Sampling” signal on LED |
| LowV\_IRQ | Low voltage powering chip | Store Low Power flag in Flash (at location of next data sample), activate low-power LED signal |
| SampleStart\_IRQ | “Start Data Collection” button held for 3 seconds | Activate TakeSample\_IRQ, store time stamp in Flash, activate “Data Sample Start” signal on LED |
| SampleEnd\_IRQ | “Stop Data Collection” button held for 3 seconds | Disable TakeSample\_IRQ, activate “Data Sample End” signal on LED |
| Hibernate\_IRQ | “Power Off” button held for 3 seconds | If in active state: disable all IRQs except Hibernate\_IRQ, enter hibernate  If in hibernate state: enter sleep state, reactivate other IRQs |
| Reset\_IRQ | “Reset” button held for 3 seconds | Perform software reset |

**TakeSample\_IRQ ISR**

The PSoC3 employs a 1-kHz free-running clock to periodically wake the system up from sleep. The free running clock increments an interval counter that can be set as high as 4096, resulting in the counter rolling over every ~4.096 seconds and generating an interrupt, waking the system from sleep. This value will be adjusted based on the user’s desired data sampling frequency.

The firmware will keep a software counter in memory, which will be decremented every time the system is woken from sleep. The value of the counter will be equal to Sample\_Period / Wake\_Period. See Pg 148 of the PSoC3 TRM for the Central Timewheel information.



### LowV\_IRQ ISR

The power input to the PSoC3 will be monitored for battery drain. A low power warning will be presented to the user via a signal on an LED if a low power condition is detected, and the system will shut down if power is critically low. The system shutdown is necessary to prevent data corruption in Flash and EEPROM.



### VBus\_IRQ ISR

The system will enter an interface mode when plugged into a computer. The ISR will be initiated by a rising edge on the VBUS monitor pin of the USBFS PSoC3 component. More details about interface mode can be found in the Interface Mode section of this document.



### Data Samples

Data samples will be stored in Flash memory along with program code by utilizing the EEPROM emulator component available for the PSoC3. Each section of data samples will begin with an 8-byte header. The header contains information about the sensor used (SE), sample period (SP) and a date/time stamp (DT) when data collection begins.

1 Byte – Mark Header and Identify Sensor Type (From EEPROM)

SE[7:4] 0x2

SE[3:0] <Sensor ID>

1. 3 Bytes – Sample Period Information (From EEPROM)
   1. SP[23:22] 0x0 (Unused)
   2. SP[21:17] Day of Month
   3. SP[16:12] Hour
   4. SP[11:6] Minute
   5. SP[5:0] Second
2. 4 Bytes – Initial Date/Time Stamp for the beginning of data collection (From RTC)
   1. DT[31] 0x0 (Unused)
   2. DT[30:26] Year offset from 2014
   3. DT[25:22] Month
   4. DT[21:17] Day of Month
   5. DT[16:12] Hour
   6. DT[11:6] Minute
   7. DT[5:0] Second

**Data Samples**

The data samples themselves will be recorded in a 2-byte bit-field (DA) comprised of the 12-bit ADC output and a 4-bit message signature:

* 1. DA[15:12] Message
  2. DA[11:0] Raw ADC output

**Data Sample Messages**

**The data sample Message section, DA[15:12], is used to specify error conditions or other details about data samples:**

|  |  |  |
| --- | --- | --- |
| **Bit Pattern** | **Meaning** | **Consumer** |
| **0x0** | **No message – normal sample** | **Host/Sensor** |
| **0x8** | **Marks the end of data sample transmission to host** | **Host** |
| **0x4** | **Pad byte – used to fill 64 byte packet for USB transmission** | **Host** |
| **0x2** | **Identifies Header** | **Host** |
| **0x1** | **Error in Sample – DA[11:0] contains error code for this sample** | **Host** |
| **0xF** | **Marks end of valid sample data in Flash** | **Sensor** |

**Sample Error Messages**

**If DA[15:12] is 0b0001, then DA[11:0] contains information about an error that occurred when a data sample capture was attempted.**

|  |  |
| --- | --- |
| **Bit Pattern** | **Meaning** |
| **0x001** | **Low Battery Warning** |
| **0x002** | **Data could not be retrieved from ADC** |
| **0x003** | **Flash is full – no more data samples can be stored** |

## Computer Interface

### User Interface

The host computer user interface will be a simple GUI that performs three functions:

1. Provide a method for the user to change data collection settings or reset the base unit.
2. Allow the user export data from the device. When this occurs, the GUI will be responsible for translating the data from the 40-bit fields described in 3.1.6 into human-readable data.
3. Allow the user to display simple graphs of previously collected and exported data when the base unit is not connected to the computer.

The interface will be written in Python, compatible with version 3.3 or later. Python provides a platform-independent framework for a GUI and for interacting with the sensor.

### Program Installer

The user interface software will be packaged in an installer that allows the user’s system to recognize the base unit when attached, and launch the GUI.

### Virtual COM Port

The PSoC3 microcontroller USBFS\_UART component is used for communication with the host computer. The Python library pySerial is used to communicate over the serial port with the base unit. Communicating using a virtual serial port provides a cross-platform method of enumerating on and communicating with a host without the need for additional drivers.

### Interface Mode

While plugged into the computer, the sensor will remain in Interface Mode. During this time, the PSoC3 will continually monitor its input buffer for commands from the host computer and respond to them accordingly. The microcontroller will exit interface mode when a falling edge is detected on the VBUS monitor pin of the USBFS PSoC3 component.

All commands to the sensor from the host are packaged in a 5-byte packet (CO):

* 1. CO[39:24] Value, if applicable – padded with zeros otherwise
     + - 1. CO[23:8] Target, if applicable – padded with zeros otherwise (See table below)
  2. CO[7:0] Command Bit Pattern

All replies from the sensor are comprised of a single byte ASCII character. Data dumps from the sensor are comprised of the sample block header, described in the Data Samples section of this document, as well as the data samples themselves. The end of the data transmission is marked by a 4-byte trailer:

TR[31:16] Number of data points transmitted

TR[15:8] 0x0

TR[7:0] 0x8

All command and data packets use Big-Endian formatting.

**Available Commands**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Command** | **Mnemonic** | **Bit Pattern** | **Action** | **Target/Value** |
| IDENTIFY | I | 0x49 | Sensor replies with identifier | N/A |
| DUMP\_DATA | D | 0x44 | Sensor dumps sampled data points to host | N/A |
| CHANGE\_SETTING | C | 0x43 | Sensor stores new setting in EEPROM | Target = Setting to Modify  Value = New value of Setting |
| SUCCESS | Y | 0x59 | Sensor resumes waiting for next command | N/A |
| FAILURE | N | 0x4E | Sensor reattempts last action | N/A |

**Incoming Messages from Sensor**

|  |  |  |  |
| --- | --- | --- | --- |
| **Messages** | **Mnemonic** | **Bit Pattern** | **Meaning** |
| IDENTIFIER | E | 0x45 | Differentiates Erebus Labs Sensor from other potential serial devices |
| SUCCESS | Y | 0x59 | Last action was successful, ready for next command |
| FAILURE | N | 0x4E | Last action failed, reattempt command |



# Appendix A: Glossary

## Acronyms

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| ADC | Analog-to-Digital Converter |
| BOM | Bill of Materials |
| CO | Carbon Monoxide |
| CSV | Comma-separated-value formatted file |
| EEPROM | Electrically Erasable Programmable Read-Only Memory |
| EPL | The Portland State University Engineering and Prototyping Lab |
| I2C | The Inter-Integrated Circuit communication protocol |
| ISR | Interrupt Service Routine |
| K-12 | Kindergarten through 12th grade school |
| LED | Light Emanating Diode |
| PCB | Printed Circuit Board |
| PSoC | Programmable System On Chip |
| SI | Silicon |
| SPI | Serial Peripheral Interface Bus |
| STEM | Science, Technology, Engineering and Math |
| TRM | Technical Reference Manual |
| USB | Universal Serial Bus |

## System Architecture

**Base Unit**

The central device that manages power, communication, and data storage, and has one or more sensors attached to it.

**Sensor**

The individual data collection devices such as VOC detectors and thermometers that are attached to the base unit.

**User Interface**

The program that will be run on a laptop or desktop computer that allows the user to view and interact with the data collected.

**System**

The operational product comprised of base units with attached sensors and a user interface.