Electrical/Firmware requirements specification

for

MApC02 Gen2

Version 0.1

Prepared by Matt Casari

PMEL/EDD

August 2, 2017

Table of Contents

[1 Introduction 5](#_Toc490820051)

[1.1 Purpose 5](#_Toc490820052)

[1.2 Document Conventions 5](#_Toc490820053)

[1.2.1 Essential Definitions, Acronyms and Abbreviations 5](#_Toc490820054)

[1.3 Intended Audience and Reading Suggestions 5](#_Toc490820055)

[1.4 Project Scope 6](#_Toc490820056)

[1.5 References 6](#_Toc490820057)

[1.5.1 Documentation References 6](#_Toc490820058)

[1.5.2 Project Websites 6](#_Toc490820059)

[2 Product description 7](#_Toc490820060)

[2.1 Product Perspective 7](#_Toc490820061)

[2.2 Product Functions 7](#_Toc490820062)

[2.3 User Classes and Characteristics 7](#_Toc490820063)

[2.4 Operating Environment 8](#_Toc490820064)

[2.5 Design and Implementation Constraints 8](#_Toc490820065)

[2.5.1 Functional Requirements 8](#_Toc490820066)

[2.5.2 Hardware requirements 9](#_Toc490820067)

[2.5.3 Specific Technologies and Tools 9](#_Toc490820068)

[2.5.4 Language Requirements 10](#_Toc490820069)

[2.5.5 Security considerations 10](#_Toc490820070)

[2.5.6 Design Conventions 10](#_Toc490820071)

[2.6 User Documentation 11](#_Toc490820072)

[2.6.1 Basic Documentation 11](#_Toc490820073)

[2.6.2 Product repository 11](#_Toc490820074)

[2.7 Assumptions and Dependencies 13](#_Toc490820075)

[3 External interface Requirements 14](#_Toc490820076)

[3.1 User Interfaces 14](#_Toc490820077)

[3.2 Hardware Interfaces 14](#_Toc490820078)

[3.3 Software Interfaces 15](#_Toc490820079)

[3.4 Communications Interfaces 15](#_Toc490820080)

[3.4.1 Connected Communications 15](#_Toc490820081)

[4 System Features 16](#_Toc490820082)

[4.1 Main Control System (MCS) 16](#_Toc490820083)

[4.1.1 Description and Priority 16](#_Toc490820084)

[4.1.2 Stimulus/Response Sequences 16](#_Toc490820085)

[4.1.3 Functional Requirements 16](#_Toc490820086)

[4.2 Flow Control System (FCS) 17](#_Toc490820087)

[4.2.1 Description and Priority 17](#_Toc490820088)

[4.2.2 Stimulus/Response Sequences 17](#_Toc490820089)

[4.2.3 Functional Requirements 17](#_Toc490820090)

[4.3 Communications Control System (CCS) 18](#_Toc490820091)

[4.3.1 Description and Priority 18](#_Toc490820092)

[4.3.2 Stimulus/Response Sequences 18](#_Toc490820093)

[4.3.3 Functional Requirements 18](#_Toc490820094)

[4.4 Serial Sensor Subsystem (SSS) 19](#_Toc490820095)

[4.4.1 Description 19](#_Toc490820096)

[4.4.2 Stimulus/Response Sequences 19](#_Toc490820097)

[4.4.3 Functional Requirements 19](#_Toc490820098)

[4.5 Data Storage System (DSS) 20](#_Toc490820099)

[4.5.1 Description 20](#_Toc490820100)

[4.5.2 Stimulus/Response Sequences 20](#_Toc490820101)

[4.5.3 Functional Requirements 20](#_Toc490820102)

[4.6 Battery Management System (BMS) 21](#_Toc490820103)

[4.6.1 Description and Priority 21](#_Toc490820104)

[4.6.2 Stimulus/Response Sequences 21](#_Toc490820105)

[4.6.3 Functional Requirements 21](#_Toc490820106)

[4.7 Graphical User Interface (GUI) 22](#_Toc490820107)

[4.7.1 Description and Priority 22](#_Toc490820108)

[4.7.2 Stimulus/Response Sequences 22](#_Toc490820109)

[4.7.3 Functional Requirements 22](#_Toc490820110)

[4.8 System Feature 1 23](#_Toc490820111)

[4.8.1 Description and Priority 23](#_Toc490820112)

[4.8.2 Stimulus/Response Sequences 23](#_Toc490820113)

[4.8.3 Functional Requirements 23](#_Toc490820114)

[4.9 System Feature 1 23](#_Toc490820115)

[4.9.1 Description and Priority 23](#_Toc490820116)

[4.9.2 Stimulus/Response Sequences 23](#_Toc490820117)

[4.9.3 Functional Requirements 23](#_Toc490820118)

[4.10 System Feature 1 24](#_Toc490820119)

[4.10.1 Description and Priority 24](#_Toc490820120)

[4.10.2 Stimulus/Response Sequences 24](#_Toc490820121)

[4.10.3 Functional Requirements 24](#_Toc490820122)

[4.11 24](#_Toc490820123)

[4.12 System Feature 2 (and so on) 24](#_Toc490820124)

[5 Other NonFunctional Requirements 26](#_Toc490820125)

[5.1 Performance Requirements 26](#_Toc490820126)

[5.2 Safety Requirements 26](#_Toc490820127)

[5.3 Security Requirements 26](#_Toc490820128)

[5.4 Software Quality Attributes 26](#_Toc490820129)

[6 Other Requirements 27](#_Toc490820130)

[6.1 Revision Control System 27](#_Toc490820131)

[6.2 Version Numbers 27](#_Toc490820132)

[6.3 Reuse/adaptation of Code 28](#_Toc490820133)

[Appendix A: Glossary 29](#_Toc490820134)

[Appendix B: Analysis Models 32](#_Toc490820135)

[Appendix C: To Be Determined List 33](#_Toc490820136)

REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **DATE** | **CHANGES** | **VERSION** |
| M.Casari | 8/2/2017 | Initial Release | 0.1 |

# Introduction

## Purpose

The purpose of this document is to present an electrical & firmware systems overview for the proposed second generation Moored Autonomous Partial Pressure Carbon Dioxide (MApCO2) sensor. This document covers all requirements for essential functionality of the new sensor to preserve, at minimum, the functionality of the first generation MApCO2 sensor.

## Document Conventions

### Essential Definitions, Acronyms and Abbreviations

The following definitions, acronyms and abbreviations are considered important to the general reading of this document. A more comprehensive list of terms can be found in Appendix A: Glossary.

Table 1. Definition of important terminology.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| NOAA | National Oceanic and Atmospheric Administration |
| PMEL | Pacific Marine Environmental Laboratory |
| DOC | Department of Commerce |
| MApCO2 | Moored Autonomous Partial Pressure Carbon Dioxide |
| GUI | Graphical User Interface |
| User | System or subsystem user, could be human or machine |
| Firmware | Embedded system (microcontroller) code |
| i-Data | Iridium Formatted Data (PMEL Format) |
| s-Data | Standard Formatted Data (PMEL Format) |
|  |  |

Table 2. Requirements terminology.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| SHALL | Hard Requirement, considered critical to functionality of the system/project. |
| WILL | Intermediate requirement, to be implemented if time permits. |
| SHOULD | Lowest priority. Concept is not complete at the time of writing this document and further investigation may determine this feature is unnecessary. |

## Intended Audience and Reading Suggestions

* System developers, such as engineers, technicians, scientists and administrators.
* Typical Users, such as scientists, engineers and technicians in the NOAA PMEL organization.
* External users, such as scientists and engineers in partnering institutions or industry.
* Programmers and data analysts who will interface with the system remotely.

## Project Scope

The second generation of MApCO2 sensors (MApCO2Gen2) is an electrical hardware and firmware upgrade to the first generation of pCO2 sensors. This document will cover both hardware and software requirements for the system, as well as practices to be used during development.

This document will not cover the specific interface specification to communicate with the system. See the TBD for more information regarding communications. At minimum, the system shall have the basic functionality of the Battelle Interface Specification [4].

## References

### Documentation References

1. Musielewicz,S. (2015). *MApCO2 User Manual*, NOAA/PMEL, Settle, WA
2. Battelle. (2009). *System Manual for Seaology pCO2 Monitor*, Battelle, Columbus, OH.
3. Battelle. (2010). *User Manual for the pCO2 System Software Version 03.18*, Battlelle, Columbus, OH.
4. Battelle. (2009). *Interface Specification for the pCO2 Sensor System*, Battelle, Columbus, OH.
5. “Battelle Seaology® pCO2 Monitoring System.” pCO2 Monitoring System, Battelle, [www.battelle.org/government-offerings/national-security/maritime-systems-technologies/sensors/pco2-monitoring-system](http://www.battelle.org/government-offerings/national-security/maritime-systems-technologies/sensors/pco2-monitoring-system).
6. Preston-Werner, Tom. “Semantic Versioning 2.0.0.” Semantic Versioning, Github, semver.org/
7. Patel, Amit, et al. “Google Python Style Guide.” Google Python Style Guide, Google, google.github.io/styleguide/pyguide.html.

### Project Websites

The project repository for this development can be found at <https://github.com/NOAA-PMEL/EDD-MApCO2>.

# Product description

## Product Perspective

The system being developed is the second generation of moored autonomous partial carbon dioxide (MApCO2Gen2) system, focusing on the upgrade and replacement of obsolete electronics found in the first generation systems. The new system will utilize all existing housings and sensors used by the previous system, allowing for a drop in replacement as the older system electronics degrade over time. The new system will follow a modular design paradigm, allowing for quick replacement of individual sensors as well as provide integration of newer sensors without major revisions to the systems firmware. The system has capabilities for measuring pCO2 from seawater and air using a custom designed flow system and a LiCor 820/840. Interfaces for CTD, Oxygen optode and pH sensors will be available. User will have a graphical user interface to program sampling schedules. Data measurements will be stored internally while statistical sampling will be transmitted via Iridium or cellular modem.

## Product Functions

The MApCO2Gen2 has the following functions:

* Collect, dry and measuring surface air samples; measure pCO2 concentrations.
* Collect and process (sea)water samples; measure pCO2 concentrations.
* Measure (sea)water conductivity and temperature.
* Measure (sea)water pH via external sensor.
* Measure dissolved oxygen in (sea)water.
* Perform sampling of sensors and control of flow systems on a user defined schedule.
* Record sensor and system data on removable storage drive.
* Connect and save data subsets to user network storage via Iridium or cellular modem.

The configuration GUI shall have the following functions:

* Master controller scheduling configuration.
* Subsystem parameters, configurations and settings interface.
* Monitor and display data from sensors and systems in real-time for bench testing.

## User Classes and Characteristics

*<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>*

## Operating Environment

The MApCO2Gen2 system shall be built on custom printed circuit boards (PCBs) with custom firmware. Where possible, the subsystems shall reuse code from other subsystems to provide a consistent code base. A real-time operating system (RTOS) may be used if it meets the timing and performance requirements discussed in this document. Preference shall be given to an open-source RTOS and other tools.

The GUI system shall be developed on the Windows operating system. Where possible, the code should be written to be useable cross-platform. The GUI source code shall be open to other users who wish to port it to other operating systems.

## Design and Implementation Constraints

### Functional Requirements

#### Basic Requirements to Meet Generic Specifications

The MApCO2Gen2 system shall meet the following minimum general functional requirements of the first generation MApCO2 system (modified from system performance specification on the Battelle pCO2 webpage [5]):

1. Provide time series measurements of ocean carbon and air sea exchange to monitor carbon cycle variability.
2. Feature a measurement range of 100 to 600 ppm or greater (with an extended range available) and a precision of about 1ppm.
3. Offer ease of operation, set up and deployment.
4. Functions continuously for more than one year without intervention.
5. Feature two-way satellite data transmission capabilities that control the system and receive data in near real-time.

#### System Performance Requirements

The system shall conform to the following minimum requirements

1. The system shall be capable of measuring a pCO2 range between 100 and 600 ppm.
2. The system shall have a precision of approximately 1 ppm
3. The system shall have a maximum average current draw of 24mA over 24 hour (not including Iridium/Communications).
4. Iridium/Communications shall have a maximum average current draw of 3mA over 24 hours.
5. The system shall have a means of (limited) two-way communication over Iridium or other wireless communication systems.
6. The system shall be easy to operate, set up and deploy.
7. The system communications interface shall be backwards compatible with the first generation MApCO2 system.
8. The new system shall be designed to fit into all first generation housings.
9. The new system shall utilize the same connector and external wiring as the first generation system.

### Hardware requirements

#### Memory Constraints

* There is no specified memory constraints for the embedded system, provided that all tasks and schedules can be fulfilled to their minimum requirements.
* For the GUI and other desktop/laptop programs used to interface to the embedded system shall have a minimum memory requirement of TBD.

#### Timing Requirements

* All measurements on the CO2 embedded system happen at a 1Hz rate maximum. The firmware shall be designed to sample/operate at 10Hz or better to capture any future use cases for the system.
* For the desktop/laptop programs, any modern computer (circa 2010 and newer) shall be sufficient.

#### Communication

* The user shall have the ability to connect to the system via RS-232 or USB virtual COM port communications.
* The system shall be capable of serial communications at baud rates of 9600 to 115200 (Standard baud rates only)

### Specific Technologies and Tools

* A list of tools and associated software versions shall be maintained in a README file located in the main directory of the system project repository.

#### EMBEDDED SYSTEM TOOLS

* For all embedded (sub)systems, Texas Instruments MSP430 or ARM Cortex-M microcontrollers shall be used.
* Any MSP430 microcontroller used shall be chosen from the MSP430FR family (ferromagnetic RAM) for their extreme low-power capabilities and extreme lifetime during multiple read/write cycles.
* For MSP430 devices, the IAR Embedded Workbench for MSP430 v7.10 and associated compiler shall be used.
* For ARM Cortex-M devices (if used), the IAR Embedded Workbench for ARM v8.11 and associated compiler shall be used.
* For embedded unit testing, Unity, Ceedling and CMock shall be used.

#### SCHEMATICS AND PRINTED CIRCUIT BOARD DESIGN TOOLS

* PMEL uses Capilano Systems DesignWorks tool for schematic capture.
* MentorGraphics PADS is used for PCB layout design.
* Other tools may be used if they are compatible with either of the tools previously mentioned.

#### SCRIPTING AND GRAPHICAL USER INTERFACE TOOLS

* Python shall be used as the primary language for any scripting. Python 3.6 or later shall be used.
* Any Graphical User Interface shall utilize open-source technologies; specifically Python, PyQt5, pyqtgraph and PySerial.

### Language Requirements

* All embedded firmware shall be written in the C language, Assembly language (specific to the microcontroller) or a combination of both.
* GUI’s shall be written in Python using PyQt5 unless higher performance is determined necessary. C++ and Qt5 will be used for higher performance for those cases.

### Security considerations

* NOAA and DOC security requirements shall be adhered to. Passwords, personally identifiable information (PII) or any other sensitive information SHALL NOT be included in any external documentation, files, repositories, etc.
* Any security techniques used in the development of the system shall be documented but not disseminated outside of PMEL without authorization from PMEL EDD.

### Design Conventions

* The embedded source code shall be written to the ISO/IEC 9899:1999 (C99) standard. This requirement should be enforced by utilizing a lint program to identify adherence violations prior to a version release.
* Embedded systems should conform to the MISRA-C standard (MISRA-C 2004)
* All firmware source code shall use embedded commenting in the Doxygen convention so that Application Program Interface (API) documentation can be easily generated using the Doxygen tool.
* All Python scripts shall conform to the Google Python Style

## User Documentation

### Basic Documentation

Documentation delivered with this project will include:

* System User Manual
* System Wiring Diagrams
* System Bill of Materials
* System Communication Protocol
* Subsystem Schematics
* Subsystem Printed Circuit Board Gerber files
* Subsystem Bill of Materials
* Subsystem Datasheets
* Firmware reference documentation

### Product repository

In addition to the aforementioned documentation, an online repository will be kept to provide a central location for development and dissemination of information. Github is currently used by PMEL EDD, and will be used for this system development. Only PMEL EDD personnel and those granted authorization by PMEL EDD will have access to the repository. See 6.1 for more detail regarding Revision Control.

At the time of project completion this repository shall contain the following file structure with all documents included and completed (**Bold** indicates folder, *italic* indicates file):

* **Documentation**
  + *Readme*
  + *System User Manual*
  + *System Interface Communication Document*
  + **Subsystem A**
    - *User Guide*
    - *Communication Protocol*
    - *Calibration Instructions*
    - *Calibration Checklist*
* **Electrical**
  + *Readme*
  + *System Wiring Diagram*
  + *System Bill of Material*
  + **Subsystem A**
    - *Schematic*
    - *PCB Gerbers*
    - *Bill of Materials*
    - *Datasheets*
  + **Subsystem N**  
    …
* **Firmware**
  + **Binaries**
    - *Binary Files*
    - *How-to (install) readme*
  + **Source** Code
    - *All Source Code Files*
  + **Test**
    - *Readme*
    - Embedded Test
      * *Firmware Test Files*
    - System Test
      * *System Test Files*
  + **Documentation**
    - Embedded API Documentation
* **GUI**
  + *Readme*
  + **Source Code**
    - *Source Code File*
  + **Test**
    - *Readme*
    - *Test Files*
* **Mechanical**
  + *Readme*
  + *Mechanical Drawings*
  + *Bill of Materials*
  + *Solid models TBD*

## Assumptions and Dependencies

It is assumed that the end-user or manufacturer of these system will have access to all tools required to flash (install) the binary code onto all microcontrollers in the system. Similarly, the user must have access to a computer with a USB port that can run the GUI.

Although it will be possible to configure the MApCO2 to accept new sensors, it is assumed that users outside of PMEL EDD shall work with PMEL EDD engineers to do so.

# External interface Requirements

## User Interfaces

A graphical user interface (GUI) shall be provided with the final delivered system. The GUI shall be able to connect to the system hardware via a serial (RS-232, RS-485, etc.) interface in an intuitive manner. If possible, the GUI shall have an “auto-connect” feature which scans all active COM ports on the GUI computer for the MApCO2 system.

The GUI shall provide an easy-to-use method of updating system parameters and settings to the user’s requirements. An import menu shall be available to load previous system configurations. Similarly, and export menu shall be available to save current configurations.

The GUI configuration shall have a simple method of adding/removing sensors from the current system. If possible, this shall be accomplished by a graphical means. At minimum, there shall be a drop-down menu based on known sensors to choose from.

The GUI shall include a data visualization function which can plot data in real-time or in bursts (typical Iridium transmit mode). The data visualization shall have the ability to plot multiple data streams simultaneously. There shall be a method to differentiate data streams by selecting color or other unique identifier for each plot.

The GUI shall have the option to log all data received from the system. The GUI default shall be to log all data to a file. The user shall have the option to rename the log file.

The GUI shall be able to auto-configure its basic layout/configuration based on the firmware version provided by the hardware system.

Keyboard shortcuts shall be identified in the drop-down menus by underlining the special keys to access the shortcut.

## Hardware Interfaces

The system shall have the option to connect to the software console either via a RS-232 connection (3-wire Rx, Tx, GND) external to the enclosure (bulkhead connector) or a USB-to-serial “Virtual COM” port. These serial ports shall be capable of data transfer from 4800 to 115,200 baud, at typical baud rates.

The first revision hardware interface between the master controller and the subsystem controllers shall be a RS-232 or UART connections on a serial multiplexor or serial bridge. Alternatively, a RS-485 multi-drop system could be utilized if it was demonstrated that there would be no collisions or catastrophic failure should one subsystem fail. Subsequent hardware revisions should attempt to move toa system bus architecture to communicate commands and data.

The interface between master controllers or subsystem and any sensor or subsystem external to the housing should be electrically isolated, if possible

All hardware interfaces between low power electronics and electromechanical systems (solenoids, motors, etc.) should be electrically isolated, if possible.

## Software Interfaces

*< Describe the connection between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shard across software components. If the data sharing mechanism must be implement in a specific way (for example, use of a global data area in multitasking operating system), specify this as an implementation constraint>.*

## Communications Interfaces

*<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocol, electronics forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>*

The MApCO2 system must be capable of two separate communications paradigms; “Connected” and “Deployed”.

### Connected Communications

“Connected” communications refers to all communications with the MApCO2 system while the system is tethered directly to the user (computer, etc.). In this mode, RS-232 communications shall be employed with a custom communication protocol, described a TBD interface control document, and backwards compatible with the previous Battelle Interface Specification [4].

### Deployed Communications

“Deployed” communications refers to all communications that occur while the system is untethered and communicating via Iridium modem, Cellular modem, or other techniques. This mode shall adhere to the communication protocol described in the TBD interface control document in addition to the Battelle Interface Specification [4].

# System Features

This section documents all of the system and subsystem features of the MApCO2Gen2 system, their requirements and optional features.

*Requirements are considered essential.   
Options are considered “nice-to-have” but will only be implemented if time permits.  
Priorities are scaled between a low of 1 and a high of 9.*

## Main Control System (MCS)

### Description and Priority

The MCS provides all communication and scheduling between subsystems (individual sensors and flow control systems) and the data storage system. Additionally, the MCS routes all data to/from the communications control system.

### Stimulus/Response Sequences

The MCS shall have the following stimulus/response sequences:

1. MCS Power up shall provide indication of a stable ready state.
2. MCS system parameters shall be available at user request.
3. MCS status parameters shall be available at the user request.
4. MCS failures and error log shall be available at the user request.
5. MCS system options shall be available at the user request.
6. MCS shall have “pass-through” mode to allow user to directly communicate with sub-systems.
7. MCS shall have a system reset state which puts all sub-systems in the lowest, stable, power state.
8. MCS shall have features to set the system time for itself and all sub-systems.

### Functional Requirements

**REQUIREMENTS:**

1. The console/user interface protocol **shall** be backwards compatible with the first generation pCO2 system. *Priority = 9*
2. The MCS **shall** use a microcontroller commonly used by PMEL OR a microcontroller with an architecture expected to remain non-obsolete for 10 (ten) years. *Priority = 5*
3. The MCS **shall** manage scheduling for all subsystems. Where scheduling can be offloaded to a specific subsystem (i.e. subsystems with RTC), the MCS MUST ensure time synchronization at intervals of six (6) hours or better. *Priority = 9*
4. The MCS **shall** include a unique ID chip which provides a unique 16-bit (or better) serial number.

**OPTIONS:**

1. The console/user interface **should** have the option of connecting to the user over a serial port OR USB port.
2. The MCS **should** be designed for the ability to move to a communication bus such as Controller Area Network bus (CANbus) or Local Interconnect Network (LIN) as a replacement for point-to-point serial (RS-232 or TTL) connections.
3. If the MCS implements CANbus for intercommunication, NMEA2000 **should** be considered as a high-level communications protocol.

## Flow Control System (FCS)

### Description and Priority

The FCS provides the system the ability to switch between CO2 purge mode, span gas mode, air CO2 measurement mode and ocean CO2 measurement mode. It handles all switching of solenoid valves, motor control and any feedback of those electromechanical devices.

### Stimulus/Response Sequences

The FCS shall have the following stimulus/response sequences:

1. FCS Power up **shall** provide indication of a stable ready state.
2. FCS system parameters **shall** be available at user request.
3. FCS status parameters **shall** be available at the user request.
4. FCS failures and error log **shall** be available at the user request.
5. FCS system options **shall** be available at the user request.
6. FCS **shall** have user ability to control all valves either individually or as a group.
7. FCS **shall** have user pump control ability.
8. FCS **should** have predefined routines and schedules which can be called by the user.

### Functional Requirements

**REQUIREMENTS:**

1. The FCS **shall** use latching valves which are capable of operating in low (TBD) temperatures. *Priority = 9*
2. The FCS **shall** be capable of driving the six (6) latching valves at the required voltage and current for a duration that is at least 500% over the typical duration required to switch the valves (time TBD). *Priority = 9*
3. The FCS **shall** be capable of driving a TBD pump at the required voltage and current for a duration TBD. *Priority = 9*
4. The FCS **shall** have protective fuses on the input to the drive electronics to prevent any
5. The FCS **shall** monitor the state of the input drive fuses to alert the user to any failure conditions. *Priority = 7*
6. The FCS **shall** have an onboard real-time clock (RTC) to run configurable profiles in sync with the main control system. Priority = 8
7. The FCB **shall** be capable of accepting battery voltages between TBD and providing voltage regulation to the valve and pump, if needed.
8. The FCB **shall** utilize electrical isolation between the driver circuitry and the logic control subsystems. *Priority = 8*

**OPTIONS:**

1. The FCS **should** be capable of monitoring the current profile the pump while energized.
2. The FCS **should** be capable of monitoring the current state of each individual valve.
3. The FCS **should** monitor the temperature of the valve block and provide an extended energized period during colder temperatures.
4. The FCS **should** be capable of monitoring the flow rate of the pumped fluid.

## Communications Control System (CCS)

### Description and Priority

The communication control system is responsible for all rou

### Stimulus/Response Sequences

The CCS shall have the following stimulus/response sequences:

1. CCS Power up **shall** provide indication of a stable ready state.
2. CCS system parameters **shall** be available at user request.
3. CCS status parameters **shall** be available at the user request.
4. CCS failures and error log **shall** be available at the user request.
5. CCS system options **shall** be available at the user request.
6. CCS **shall** provide the GPS location at the users request.
7. CCS **shall** provide the Iridium signal strength at the users request.
8. CCS **shall** provide the ability to test Iridium communications at the users request.

### Functional Requirements

1. The CCS **shall** have a command interface which is backwards compatible with pCO2g1. *Priority = 9*
2. The CCS **shall** be capable of communications of the IRIDIUM network. *Priority = 9*
3. The CCS **shall** be capable of providing GPS locations at regular intervals (TBD). *Priority = 9*
4. The CCS **shall** be capable of providing GPS fix when externally requested. *Priority = 9*
5. The CCS **shall** return zeros when the GPS fails to fix (or NaN). *Priority = 5*
6. The CCS **shall** have two configurable sampling modes, at minimum; Report every sample or Report every day. *Priority = 8*

**OPTIONS**

1. The CCS **should** be designed to communicate over IRIDIUM or Cellular Modems.

## Serial Sensor Subsystem (SSS)

### Description

The SSS is a module which commands and interprets sensors with serial communications. The SSS can interpret commands from the user (MCS or other), and based on configuration flies, provide data from one of the many sensors the MApCO2 system monitors. The SSS converts the native data format of the sensor into a PMEL format (e.g., i-Data, s-Data, etc.) before returning the data to the user.

### Stimulus/Response Sequences

The SSS shall have the following stimulus/response sequences:

1. SSS Power up **shall** provide indication of a stable ready state.
2. SSS system parameters **shall** be available at user request.
3. SSS status parameters **shall** be available at the user request.
4. SSS failures and error log **shall** be available at the user request.
5. SSS system options **shall** be available at the user request.
6. SSS **shall** provide a sensor “pass-throught” mode at the user request.
7. SSS **shall** provide the i- and s-Data at the user request.

### Functional Requirements

1. The SSS **shall** be capable of interfacing with serial (RS-232, RS-485, TTL) sensors at typical baud rates. *Priority = 9*
2. The SSS **shall** be capable of communicating with the following devices (AT MINIMUM):
   * LI-COR 820: All versions
   * Seabird SBE16: All versions
   * Seabird SeaFET: All versions
   * Sunburst Sensors SAMI2
   * Honeywell Durafet: All versions
   * Aanderaa 4831 Oxygen Optode
   * Aanderaa 4330 Oxygen Optode

**OPTIONS:**

1. Change the sys in the bullet list to reflect the acronym of abbreviated name of the system feature.

## Data Storage System (DSS)

### Description

The DSS provides all data storage for the MApCO2 system. This critical system provides the storage of all raw data from each of the sensors which the MCC controls.

### Stimulus/Response Sequences

The DSS shall have the following stimulus/response sequences:

1. DSS Power up **shall** provide indication of a stable ready state.
2. DSS system parameters **shall** be available at user request.
3. DSS status parameters **shall** be available at the user request.
4. DSS failures and error log **shall** be available at the user request.
5. DSS system options **shall** be available at the user request.
6. DSS Save Data mode **shall** be available at user request.
7. DSS Retrieve Data mode **shall** be available at the user request.
8. DSS disk space **shall** be available at the user request.

### Functional Requirements

**REQUIREMENTS:**

1. The DSS **shall** be able to record the data from each subsystem at size of up to 1MB per record. *Priority = 4*
2. The DSS **shall** save data in a format consistent with:

* Folder with the NAME of the sensor
* File name with the date of the data collection plus the serial number of the sensor
  + e.g. LICOR\_381210
* Data will be timestamped according to when it was sampled

1. The DSS **shall** be able to interface over a serial connection at typical baud rates up to 115,200 baud. *Priority = 5*
2. The DSS **shall** use microSD cards as storage media. *Priority = 6*
3. The DSS **shall** be capable of using microSD cards exceeding 32GB in size, therefore requiring exFAT formatting. *Priority = 7*

**OPTIONS:**

1. The DSS should have redundancy when integrated into the MCS.

## Battery Management System (BMS)

### Description and Priority

The BMS is a dedicated system to monitor the battery packs in the MApCO2Gen2 system. At minimum, it will monitor the state of each pack and switch off packs.

### Stimulus/Response Sequences

1. BMS Power up **shall** provide indication of a stable ready state.
2. BMS system parameters **shall** be available at user request.
3. BMS status parameters **shall** be available at the user request.
4. BMS failures and error log **shall** be available at the user request.
5. BMS **should** provide the current state of the battery pack(s) at the user request.
6. BMS **should** be able to switch battery pack on and off at the user request.
7. BMS **should** provide indication of fuse state at the user request.

### Functional Requirements

**REQUIREMENTS:**

1. The BMS **shall** have the ability to monitor both battery packs to determine the current charge level. *Priority = 9*
2. The BMS **shall** provide fusing to prevent any short circuits from occurring. *Priority = 7*
3. The BMS **shall** be capable of switching battery connections to open if measurements indicate a failing or failed battery pack. *Priorty = 1*
4. The BMS **shall** be capable of switching battery connections by a user command (including commands by the control system). *Priority = 9*

**OPTIONS:**

1. The BMS **should** be designed for forward compatibility by offloading battery monitoring by type (e.g., Lithium, Alkaline, etc.) onto a dedicated module. Ability to monitor/manage ALL battery types is preferred, but not required.
2. The BMS **should** be designed for the ability to integrate energy harvesting techniques (e.g. Solar) in future revisions.
3. The BMS **should** monitor power statistics (i.e. average current, peak current, etc.) if it will not significantly affect battery life (power draw less than TBD).

## Graphical User Interface (GUI)

### Description and Priority

The Graphical User Interface **shall** be the primary tool used by system operators to connect to and configure the MApCO2 system. It

### Stimulus/Response Sequences

1. The GUI **shall** auto-connect with the system at the user request.
2. The GUI **shall** configure the system at the user request.
3. The GUI **should** auto-generate or retrieve a unique serial number at the user request.

### Functional Requirements

**REQUIREMENTS:**

1. The GUI **shall** be easy-to-use and have intuitive functions.
2. The GUI **shall** have a help menu.
3. The GUI **shall** have an import menu which retrieves previously saved configurations.
4. The GUI **shall** have an export menu to save the current configuration.
5. The GUI **shall** have the ability to plot pCO2 data.
6. The GUI **shall** graphically identify the current system state.
7. The GUI **shall** have a system auto-connect feature.

**OPTIONS:**

1. The GUI **should** have the ability to link all current configurations to a database.
2. The GUI **should** have the ability to control each sub-system individually.

## System Feature 1

*<Don’t really say “System Feature 1.” State the feature name in just a a few words.>*

### Description and Priority

*<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>*

### Stimulus/Response Sequences

*<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>*

### Functional Requirements

*<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to the anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>*

*<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>*

**REQUIREMENTS:**

1. Change the sys in the bullet list to reflect the acronym of abbreviated name of the system feature.

**OPTIONS:**

1. Change the sys in the bullet list to reflect the acronym of abbreviated name of the system feature.

# Other NonFunctional Requirements

## Performance Requirements

*< If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify timing relationships for real-time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>*

## Safety Requirements

The design of the system shall adhere to all safety regulations enforced by NOAA and the DOC.

## Security Requirements

Any documentation, source code, or other files which could be exposed to the public (i.e. stored on cloud services) shall not contain:

* Passwords
* Personally identifiable information (PII)
* Sensitive information as regulated by DOC and NOAA regulations

## Software Quality Attributes

*<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>*

# Other Requirements

## Revision Control System

This project shall maintain revision control using Git, with the primary repository located on Github. All active developer participants in the project shall have administrative rights. Occasional contributors shall have read/write permissions. All other users shall have read permissions only. Access to a higher permission level shall be granted on an as-needed basis.

Active developers shall commit to the Github repository on a basis TBD by the project manager. Merge conflicts that cannot be easily resolved shall involve the project manager before committing to the project repository.

## Version Numbers

Code versioning shall follow the rules of the Semantic Versioning 2.0.0 [5], as restated below:

**Given a version number MAJOR.MINOR.PATCH, increment the:**

1. **MAJOR version when you make incompatible API changes,**
2. **MINOR version when you add functionality in a backwards-compatible manner, and**
3. **PATCH version when you make backwards-compatible bug fixes.**

**Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.**

See the Sematic Versioning document for more details.

For MAJOR versions less than 2 (beta), builds should be indicated by the build number and git commit SHA value (short format). For example:

**1.3.12-10-5d12c6f**

Where:

* **1.3.12** is the MAJOR.MINOR.PATCH version
* -**10** is the BUILD number
* **-5d12c6f** is the git commit SHA value

Build number and git commit SHA value shall not be included on releases 2 or greater.

Version numbering in source code shall be automated by means of a script run at compilation or runtime.

MAJOR and MINOR versions shall be decided by the project/product manager based on completion of project tasks and validation by test.

## Reuse/adaptation of Code

Where possible, all embedded firmware code should be developed as modularly as possible to allow for reuse in other (sub) systems. The interface requirements between sub-systems shall remain consistent after the release of the beta version (Major release 2.0.0), requiring all new modules at that time to be backwards compatible.

# Appendix A: Glossary

*<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>*

**A**

**B**

**C**

**D**

**E**

**F**

**G**

**H**

**I**

**J**

**K**

**L**

**M**

**N**

NOAA – National Oceanic and Atmospheric Administration

**O**

**P**

PMEL – Pacific Marine Environmental Laboratory

**Q**

**R**

RTOS – Real-time operating system

**S**

**T**

**U**

**V**

**W**

**X**

**Y**

**Z**

# Appendix B: Analysis Models

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>*

# Appendix C: To Be Determined List

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*