Revision

1

Rochester institure of technology

Advanced Battery Testing Laboratories

Battery Cycler Operations Manual

Table of Contents

About This Manual x

[An Introduction to the Hardware 1](#_Toc36022999)

Electrical Connections 1

Navigating the Systems Menus 1

Hardware Specifications and Requirements x

An Introduction to the Software x

Overview 3

Profiles, Schedules, and Program Creation 3

Standard Profiles x

User Defined Profiles x

Scaling, BSF, and Units x

Data Logging x

Connecting to the Hardware x

Running Schedules x

Software Specifications and Requirements x

Setup x

System Configuration x

Sizing Wiring x

Calibration x

Safety x

Safety Features x

System Faults x

[Serial](#_Toc36023000) Communication Protocol 8

Electrical Schematics appendix-x

Mechanical Drawings appendix-x

Chapter

1

# An Introduction to the Hardware

t

o save time in the future, print a copy of this document. Click **Print** on the **File** menu, and press ENTER to receive all eight pages of examples and instructions. With the printed document in hand, position yourself in normal view to see the style names next to each paragraph. Scroll through the document, and write the style names next to the paragraphs (press CTRL+HOME to reposition yourself at the beginning of the document).

To create a drop cap for the lead paragraph, like the example above, select the letter T, and then type a new letter.

# Electrical Connections

The “icon key” at left was produced by using the Heading 8 style for the words “icon key” and the List Bullet 5 style for the text below—which uses a Wingdings symbol for the bullet character. To change the bullet symbol, click **Bullets and Numbering** on the **Format** menu. Click **Modify**, and modifty

# Navigating the Systems Menus

MENU SYSTEM:

There are currently 5 different configuration menus.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| E | X | I | T |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| + | + |  |  |  |  |  |  |  |  |  |  |  |  | + | + |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D | I | S | P |  | T | I | M | E |  | O | U | T |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D | I | S | P |  | D | E | L | A | Y |  | T | I | M | E |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F | A | N |  | O | N |  | T | E | M | P |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | W |  | V | E | R | S | I | O | N |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

STATS DISPLAY:

Currently there are 8 status displays; they are shown here in they same order as on the system.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | H | 1 | : |  | A | C | T | I | V | E |  | # | 2 | 5 | 6 |
|  |  |  |  |  | I | D | L | E |  |  |  |  |  |  |  |
|  |  |  |  |  | F | A | U | L | T |  |  | # | 2 | 5 | 6 |
| C | H | 2 | : |  | A | C | T | I | V | E |  | # | 2 | 5 | 6 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | H | 1/2 | : |  |  |  | V | = |  |  | 6 | . | 0 | 0 | 0 |
| W | = | +/- | 1 | 5 | 0 |  | I | = | +/- | 2 | 0 | . | 0 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | H | ½ | : |  |  |  | S | O | C | = | 1 | 0 | 0 | . | 5 |
|  | C | E | L | L |  | T | E | M | P | = | 1 | 5 | 0 | ° | C |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | H | ½ | : |  |  | L | O | A | D | = | 1 | 5 | 0 | ° | C |
|  |  |  |  | C | H | A | R | G | E | = | 1 | 5 | 0 | ° | C |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| C | H | M | B | R |  |  | S | E | T | = | 1 | 5 | 0 | ° | C |
|  |  |  |  |  |  | M | E | A | S | = | 1 | 5 | 0 | ° | C |

# Hardware Specifications and Requirements

This section details the capabilities and needs of the battery cycler hardware.

Table 1: Hardware Specifications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Parameter | Min | Nom | Max | Units |
| Channels |  | 2 |  |  |
| Charge current per channel\* | 0.005 |  | 20 | Amp |
| Load current per channel | 0.005 |  | 20 | Amp |
|  |  |  |  |  |
| Profile step resolution |  | 100 |  | mS |
| Data Sampling Rate |  | 10 |  | Hz |
| Data Recording Rate |  | 1 |  | Hz |
|  |  |  |  |  |
| Commanded current resolution |  | 5 |  | mA |
| Measured current resolution |  | 2.5 |  | mA |
| Measured voltage resolution |  |  |  | mV |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\*Note: the charge current per channel is limited by the charging supply, therefore if only a 20 Amp supply is available the sum of the charge current from each channel cannot exceed 20A.

# An Introduction to the Software

Because all critical timing and profile storage is handled within the hardware of the battery cycler, the responsibility of the host PC software is relatively light. Nevertheless it is still required for system operation. The host PC software is a scheduling and data acquisition application that communicates with the battery cycler hardware. It is responsible for running the battery cycler on a user defined schedule, as well as collecting and saving all data sent back from the hardware during testing.

# Profiles, Schedules and Program Creation

Assuming that you see your paragraph marks, you’ll notice a paragraph mark attached to the lower-right corner of the picture. Click the picture, and notice too, the name of the style—not surprisingly, the Picture style. Pictures attached to paragraph styles make it possible for pictures to act like paragraphs.

Battery Cycler recognized commands:

**(0-32) - Basic commands that can be run while the system is not busy and don’t change any system variables**

0 – Stop

1 – Pause

2 – Clear charging energy

3- Clear load energy

4 –

**(33-64) - Update system variables, can only be done when system is not busy:**

**Chamber temp, BSF, max battery voltage, min batt voltage, max batt temp, min batt SOC?, other safety conditions.**

33 –

How our profiles are defined:

A simple profile is defined as a series of steps that contain 4 variables;

**-8bit mode:** the mode defines how the system is operated and how the limit is interpreted.

**-16bit output value:** the value determines what the output of the system is to be in the particular mode for this particular step.

**-16bit limit value:** will typically be a time limit but could be re-allocated depending on the modes needs. the maximum time in units of seconds that the step is to run for before moving on. Max time limit for a step is 18hours

**- 16bit limit value:** a second value that when reached completes the step, similar to time, could possibly be combined. The relevance of the limit value and time value are dependent on the particular mode of operation.

Note: A limit should be able to be set to its max value or 0 to be ignored…

Modes: (0-256)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mode (0-256) | Output value | Limit value 1 | Limit value 2 | Notes |
| 0 | Channel off | Time limit |  | Resting periods |
| 1 | Constant Current charging(mA) | High cell voltage | Time limit | 1st step charging. |
| 2 | Constant Current load (mA) | Low cell voltage | Time limit | Dirving profiles, CC capacity, |
| 3 | Constant Current load | Low cell voltage | Low SOC | Decreasing SOC (without hitting fault voltage) |
| 4 | Constant Voltage charging | Low current limit | Time limit | 2nd step charging. |
| 5 | Constant Voltage load | Low cell voltage | Time limit |  |
| 6 | Constant Power charging | High cell voltage | Time limit |  |
| 7 | Constant Power load | Low cell voltage | Time limit | CW capacity |
| 8 | Constant power load | Low cell voltage | Low SOC | Decreasing SOC without hitting voltage fault. |



# Safety

###### Written exercise pages 121 - 123 in your workbook.

In print layout view, double-click the header or footer to activate it, or click **Header and Footer** on the **View** menu. You can change or delete the text just as you would regular document text. To specify placement and whether the header or footer should be different on odd and even pages, or different for the first page only, click **Page Setup** on the **File** menu, and then click the **Layout** tab.

# Index

background, 1

border, 3

bullet, 1

caption, 2

color, 2

drawing, 2

drop cap, 1

footer, 3

frame, 3

graphic, 2

group, 2

header, 3

Help, 3

link, 4

margins, 2

normal view, 1

number, 4

picture, 2, 3, 4

print, 1

re-size, 3

section break, 2

shading, 1

style, 1, 2, 3, 4

symbol, 1

Table of Contents, 3

template, 4

ungroup, 2

Wingdings, 1

# Appendix A: Battery Cycler Communications Protocol 0.1

The order that data is transmitted and received is shown in listed in the order that they are received / transmitted.

Communications from the host PC to the battery cycler system and from the battery cycler system to the host PC are very similar. They use the same start byte as shown in , and both end with a 16bit CRC. The data contained in the message is of a fixed length and specific to the type of transmission as detailed below.

Table : Byte order

|  |
| --- |
| Header byte |
| X number of Argument bytes |
|
| Checksum |

Table : Start Byte

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Contents | *Unused* | | | CH# | Type of transmission | | | |

**Types of transmissions (HOST PC)(0-7):**

1. **Basic request:** A basic request is an action request from the host PC that may or may not be completed while the system is busy. The basic request transmission contains one data byte that contains the request number. Requests 0-127 may be processed while the system is busy. Requests 128-255 may not be processed while the system is busy. See for a detailed list of basic requests.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0x1** | | | |
| B2 – Arg 1 | **Basic Request ID** | | | | | | | |
| B3 – Arg 2 | *Unused* | | | | | | | |
| B4 – Arg 3 | *Unused* | | | | | | | |
| B5 – Arg 4 | *Unused* | | | | | | | |
| B6 – Arg 5 | *Unused* | | | | | | | |
| B7 – Arg 6 | *Unused* | | | | | | | |
| B8 – Arg 7 | *Unused* | | | | | | | |
| B9 – Arg 8 | *Unused* | | | | | | | |
| B10 – Arg 9 | *Unused* | | | | | | | |
| B11 – Arg 10 | *Unused* | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

**2) Profile request:** A profile request message has only 1 data byte; it contains the profile ID that is to be started. A profile request can only be made while the system is idle.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0x2** | | | |
| B2 – Arg 1 | **Profile Request ID** | | | | | | | |
| B3 – Arg 2 | *Unused* | | | | | | | |
| B4 – Arg 3 | *Unused* | | | | | | | |
| B5 – Arg 4 | *Unused* | | | | | | | |
| B6 – Arg 5 | *Unused* | | | | | | | |
| B7 – Arg 6 | *Unused* | | | | | | | |
| B8 – Arg 7 | *Unused* | | | | | | | |
| B9 – Arg 8 | *Unused* | | | | | | | |
| B10 – Arg 9 | *Unused* | | | | | | | |
| B11 – Arg 10 | *Unused* | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

1. **Variable update:** A variable update message has 3 data bytes; the first contains the variable ID that is to be updated. Followed by the 16bit integer.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0x3** | | | |
| B2 – Arg 1 | **Variable ID** | | | | | | | |
| B3 – Arg 2 | **Var upper nibble** | | | | | | | |
| B4 – Arg 3 | **Var lower nibble** | | | | | | | |
| B5 – Arg 4 | *Unused* | | | | | | | |
| B6 – Arg 5 | *Unused* | | | | | | | |
| B7 – Arg 6 | *Unused* | | | | | | | |
| B8 – Arg 7 | *Unused* | | | | | | | |
| B9 – Arg 8 | *Unused* | | | | | | | |
| B10 – Arg 9 | *Unused* | | | | | | | |
| B11 – Arg 10 | *Unused* | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

1. **User profile update:** A user profile update message is used to add steps to user defined profiles. The message must contain the profile ID as well as step number for the profile, followed by mode, step, limit 1, and limit 2.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | | **0x4** | | | |
| B2 – Arg 1 | **User profile #** | | | | | | | |
| B3 – Arg 2 | **Profile step #** | | | | | | | |
| B4 – Arg 3 | **Mode upper nibble** | | | | | | | |
| B5 – Arg 4 | **Mode lower nibble** | | | | | | | |
| B6 – Arg 5 | **Output upper nibble** | | | | | | | |
| B7 – Arg 6 | **Output lower nibble** | | | | | | | |
| B8 – Arg 7 | **Limit 1 upper nibble** | | | | | | | |
| B9 – Arg 8 | **Limit 1 lower nibble** | | | | | | | |
| B10 – Arg 9 | **Limit 2 upper nibble** | | | | | | | |
| B11 – Arg 10 | **Limit 2 lower nibble** | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

**Types of transmissions (SYSTEM)(8-15):**

1. **System Idle:** When a channel is Idle its heartbeat transmission will have 4 data bytes containing two temperatures. Cell temp and chamber temp. In units of 0.1 deg C.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0x8** | | | |
| B2 – Arg 1 | **Cell Temp upper nibble** | | | | | | | |
| B3 – Arg 2 | **Cell Temp lower nibble** | | | | | | | |
| B4 – Arg 3 | **Chamber Temp upper nibble** | | | | | | | |
| B5 – Arg 4 | **Chamber Temp lower nibble** | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

1. **System Active:** When a channel is active its heartbeat transmission will have a large number of bytes of data, this data is outlined in table 2. Its current implementation is shown below

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0x9** | | | |
| B2 – Arg 1 | **Profile ID** | | | | | | | |
| B3 – Arg 2 | **Step #** | | | | | | | |
| B4 – Arg 3 | **Step time & 0xF000** | | | | | | | |
| B5 – Arg 4 | **Step time & 0x0F00** | | | | | | | |
| B6 – Arg 5 | **Step time & 0x00F0** | | | | | | | |
| B7 – Arg 6 | **Step time & 0x000F** | | | | | | | |
| B8 – Arg 7 | **Voltage mV upper nibble** | | | | | | | |
| B9 – Arg 8 | **Voltage mV lower nibble** | | | | | | | |
| B10 – Arg 9 | **Current mA upper nibble** | | | | | | | |
| B11 – Arg 10 | **Current mA lower nibble** | | | | | | | |
| B12 – Arg 11 | **Cell Temp upper nibble** | | | | | | | |
| B13 – Arg 12 | **Cell Temp lower nibble** | | | | | | | |
| B14 – Arg 13 | **Chamber Temp upper nibble** | | | | | | | |
| B15 – Arg 14 | **Chamber Temp lower nibble** | | | | | | | |
| B16 – Arg 15 | **Cum Energy upper nibble** | | | | | | | |
| B17 – Arg 16 | **Cum Energy lower nibble** | | | | | | | |
| B18 - Checksum | **Checksum** | | | | | | | |

1. **System Fault:** When a channel is in the fault state its heartbeat transmission will have 1 data byte that contains the specific fault code the channel is experiencing.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | **CH#** | **0xA** | | | |
| B2 – Arg 1 | **Fault ID** | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

1. **Rx ACK:** When the device receives a transmission from the host pc, it checks the checksum of the message. If it is correct it will send an ACK message back to the host PC acknowledging its receipt. An ACK message has only a single data byte, it contains the checksum of the message that it is acknowledging.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B1 – Header | *Unused* | | | | **0xB** | | | |
| B2 – Arg 1 | **Checksum being ACKed** | | | | | | | |
| B12 -Checksum | **Checksum** | | | | | | | |

Specific examples of the communications are shown here:

Table : Data provided in active channel heartbeat

****

Table : List of Basic Requests

|  |  |  |
| --- | --- | --- |
| **Request #** | **Request Name** | **Description** |
| **0** | **Stop** | **Halts testing on the specified channel, sets output to 0. Status becomes idle.** |
| **1** | **Start** | **Will enable the specified channel at whatever step and profile it was stopped at. (ex. Resume from a fault)** |
| **2** | **Reset energy counters** | **Zeros out the Charge, load, and SOC, counts for the specified channel.** |
| **3** |  |  |
| **4** |  |  |
| **5** | **Test mode** | **Outputs ramp data ~~for channels 0 and 1~~ until stopped** |
|  |  |  |

Table : List of updatable variables

|  |  |  |  |
| --- | --- | --- | --- |
| **Variable #** | **Variable Name** | **Units / type (Range)** | **Description** |
| **0** | **Chamber Temp** | **0.1 Deg C / int (+- 3,200)** | **Setpoint for chamber** |
| **1** | **Max cell temperature** | **0.1 Deg C / int (+- 3,200)** | **Fault temp for cell** |
| **2** | **BSF (battery scale factor)** | **- / Unsigned int (64,000)** | **BSF for relative profiles** |
| **3** | **Max cell voltage** | **mV / unsigned int (64V)** | **Battery voltage for high voltage fault** |
| **4** | **Min cell voltage** | **mV / unsigned int (64V)** | **Battery voltage for low voltage fault** |
| **5** | **PID kP** | **TBD** |  |
| **6** | **PID kI** | **TBD** |  |
| **7** | **PID kD** | **TBD** |  |
|  |  |  |  |

Table : List of Profile Requests

|  |  |  |
| --- | --- | --- |
| **Request #** | **Profile Name** | **Description** |
| **0** |  |  |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |
| **6** |  |  |

Table : List of System Faults

|  |  |  |
| --- | --- | --- |
| **Fault #** | **Fault Name** | **Description** |
| **0** |  |  |
| **1** |  |  |
| **2** |  |  |
| **3** |  |  |
| **4** |  |  |
| **5** |  |  |
| **6** |  |  |

**Schedule and Program files:**

**Definition:**

Let’s start with the definition of each term relative to the battery cyclers operation.

A program is a sequential list of commands that is what the scheduler is going to read and transmit to the battery cycler system to be processed. It is processed one line at a time and the next line cannot be run until the previous one has been finished. A battery researcher can use a program to tell the system what tests to run and in what order, when to save a new data file and what to call it, etc. You can also have as many different programs and you desire.

A schedule is very similar to a program. A schedule is simply a list of programs and how many times to run each one. Again this is a list where one program must be completed before the next one can begin.

Why do we need both a schedule and a program if they are almost the same thing? This is because, as the name would imply cycling batteries requires a lot of cycling, or a lot of repetitive actions. So by having a separate schedule file where we can specify that we want one program to be repeated 400 times before we move on we just saved your fingers a lot of copy and pasting and it becomes much easier to modify the length of a test.

**Formatting:**

So how are these schedules and programs formatted? They are both comma delaminated files that are formatted as shown in the examples below. A program being the lower level file list the exact transmission type, the ID of the request #, variable#, or profile# to be run. As well as any additional data necessary; a variable value for updating variables, and a temperature for running profiles.

There is one very special line that can be used in a program file that does not actually send a transmission to the arduino. These commands control the scheduler program on the host pc directly. Currently the only command that does this is transmission type number 16. This specifies to save all the previously collected data that is currently being buffered if ram. It is followed by a description that the file name will end with. The file naming structure is as follows:

<Schedule name>\_<program name>\_<program cycle#>\_<description>

//The program file is written in a comma delaminated format as follows:

// <Transmission Type>, <Request# / Variable# / Profile #>, <Additional data if needed>

//Example Program File

1, 1, 800 // setting max temp

1, 2, 650 // setting the BSF

1, 3, 4300 // setting upper vlim

1, 4, 2200 // setting lower vlim

0, 2 // clearing energy counter

2, 5, 500 // run profile number 5 at a temp of 50 deg C

16, HPPC\_Charge // Setting the file name for following data collected ending with “HPPC\_Charge”

0, 2

2, 13, 500

2, 13, 500

2, 13, 500

2, 13, 500

16, HPPC\_Cycle

The higher level schedule file only lists two things; a programs name, and the number of times to run it.

Note: When a schedule file refers to a program file it should be a relative directory. IE assume the program file is in the same directory as the schedule file so just list its name.

//The schedule file is written in a comma delaminated format as follows:

// <Program Name>, <# of cycles>

//Example Schedule File

Table\_5\_CD, 60

HPCC, 1

HF\_IR, 1

Table\_5\_CD, 60

HPCC, 1

HF\_IR, 1

Table\_5\_CD, 60

HPCC, 1

HF\_IR, 1

**Sending transmissions to the Arduino and error checking:**

Using CRC to check data from host PC and acknowledging receipt of good data:

**Error! Reference source not found.** below shows the flow diagram for sending transmissions to the Arduino. We begin by checking if there is a fault, if there is a fault we do not want to send any more information to the Arduino until the fault is cleared.

If there is not a fault we need to check of the channel we are communicating with is idle or not.

If the channel is idle we can always assume it is ok to send the next line. If the channel is active / busy then we must first check if the next line to be sent is one that can be sent while the system is active / busy. Those commands are basic requests with an ID of 0-127 as outlined earlier.

Once we have been able to send a transmission we do not send another transmission until we receive a ACK flag from the arduino. The ACK flag is set on the start byte of the transmission from the arduino when it has read the transmission and the CRC has checked OK.

If we do not receive a ACK within 5seconds ( 5 heartbeats ) then consider the transmission lost or corrupted and attempt sending the same thing again.

Once we have received an ACK for the transmission the process is finished and we can begin waiting to transmit the next line.



Figure

**Below the process for receiving and verifying Serial data by the battery cycler hardware is defined. This process insures that two safety requirements are met.**

**1 – Errenous data in the serial stream and buffer is ignored.**

**2 – The hardware is able to verify a start byte and insure that it is ‘in sync’ with the host pc.**

**This will prevent the hardware from performing unwanted actions in the event data is corrupted during transmission or the hardware is connect to by a terminal and sent incorrect information.**

**The process also insures that incomplete messages are properly handled. Because the sending and receiving of serial data are two independent processes. It is not guaranteed that all data for a message will be in the FIFO of the UART when the hardware reads it. Therefore the start byte and one data byte may be there, however more data and CRC bytes may still be in transmission. In this case, the hardware will postpone the processing of transmission for one loop, or 100mS. Communicating at 115200Baud this is more than enough time to send even the longest of messages.**



Figure : Serial Data Processing

**Notes about system operation**

Chamber temp setting: When a request for changing a chamber temp is sent in a message to CH-0 or CH-1 it is referring to the same setting as there is only one chamber!

IF one channel requests a temperature change that is not the current set temp, and the other channel is busy. It will wait until the other channel has completed its current cycle and then pause it. Change the temperature and run the original requests profile.

All requests besides profiles can be run while the channel is being held for the different temperature.