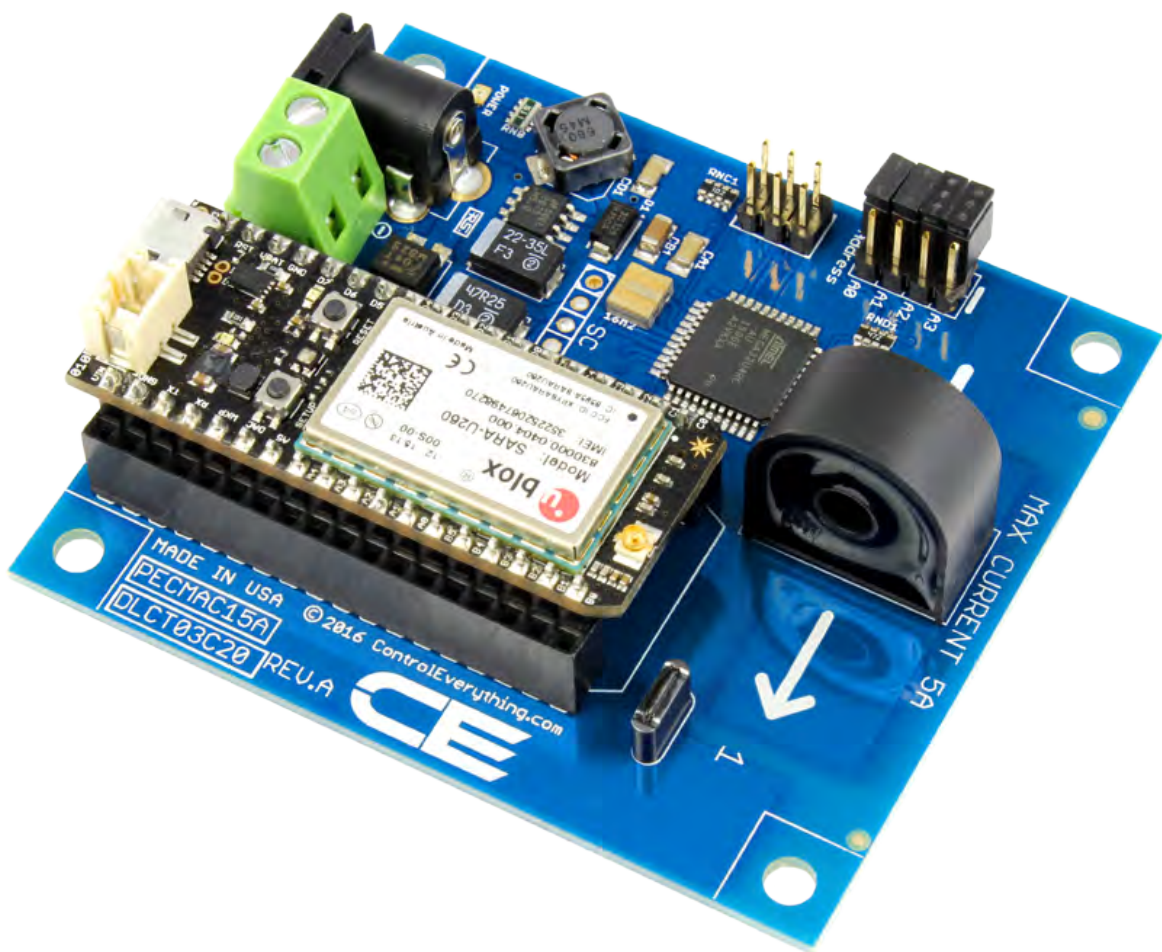


# Current Monitoring Controllers

## Command Reference Guide



# Current Monitoring Controllers Command Reference Guide

---

CONTROLEVERYTHING.COM



ControlEverything.com

© Copyright 2016  
All Rights Reserved

---

# Current Monitoring Controllers Command Reference Guide

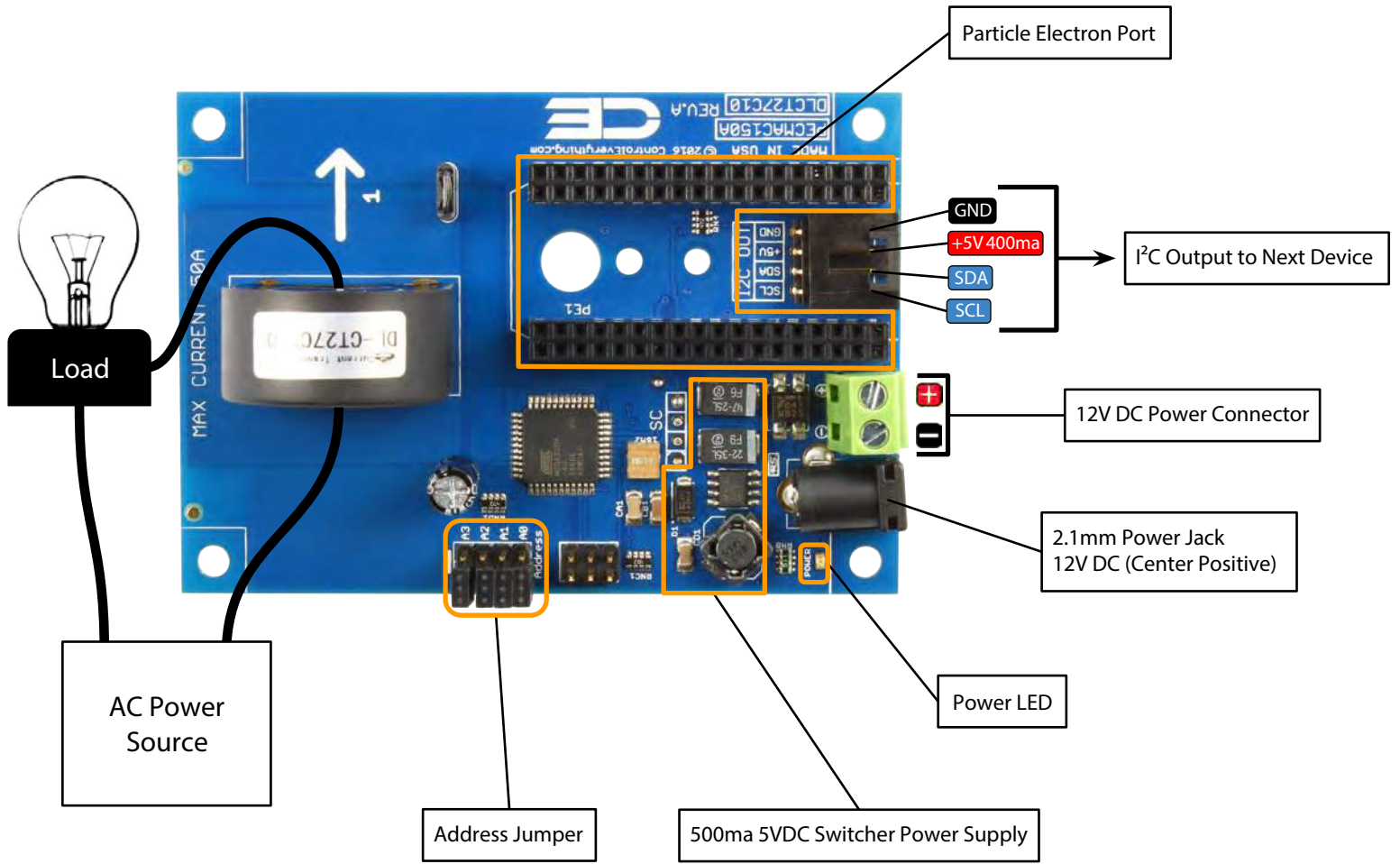
ControlEverything.com

[ControlEverything.com](http://ControlEverything.com) current monitoring controllers provide an easy cross-platform solution for communicating current monitoring data for a wide variety of energy management solutions. We developed a custom CPU that handles the difficult task of current measurement, all you have to do is communicate a simple set of I<sup>2</sup>C commands to ask the controller the amperage draw across each channel. This guide was developed to help you understand the I<sup>2</sup>C commands implemented by our extensive line of current monitoring devices.

## Getting Started:

Before getting started, we wanted to share a few important points about this series of controllers.

- 1) We post our Sample Code on the ControlEverythingCom GitHub repository page. Sample code should be referenced when reading this manual. We have developed code for Arduino, Particle Photon, Particle Electron, and the PC. Sample code shows specific information for I<sup>2</sup>C implementation.
- 2) Current monitoring controllers require a minimum of 100ma to operate properly.
- 3) A 12VDC power supply is required to power the current monitoring controller.
- 4) Only a single wire may passed through the core sensor. Passing more than a single wire will invalidate the data for the associated channel.
- 5) Current Monitoring controllers spend 1/2 second evaluating each channel. Data returned to the user may be up to 6 seconds old when monitoring current on 12-Channel controllers.
- 6) We think you will be very pleased with the accuracy of our sensors. We have been very conservative in our advertised values. We think you will find our controllers to be significantly more accurate than most commercially available clamp meters. However, our test equipment is limited to 20 Amps, therefore, higher current controllers have only been tested and calibrated up to 20 Amps. User calibration beyond 20 Amps may be necessary.



**IMPORTANT USAGE WARNINGS:**  
 This Current Monitoring Device is for use in AC Circuits Only.  
 The Maximum Current Rating Printed on the Circuit Board  
 Must be Followed or Damage to the CPU May Occur.

Up to 16 current monitoring devices can share a single I<sup>2</sup>C data bus. You can easily mix our entire range of 5A to 100A current monitoring controllers in any combination across a single I<sup>2</sup>C bus. A set of 4 jumpers, labeled A0, A1, A2, and A3 define the I<sup>2</sup>C Start Address of Each Device. Please See the Table Below to determine the I<sup>2</sup>C Start Address of your Current Monitoring Controller:

Start Address	A0 Jumper	A1 Jumper	A2 Jumper	A3 Jumper
<b>0x2A</b>	Removed	Removed	Removed	Removed
<b>0x2B</b>	Installed	Removed	Removed	Removed
<b>0x2C</b>	Removed	Installed	Removed	Removed
<b>0x2D</b>	Installed	Installed	Removed	Removed
<b>0x2E</b>	Removed	Removed	Installed	Removed
<b>0x2F</b>	Installed	Removed	Installed	Removed
<b>0x30</b>	Removed	Installed	Installed	Removed
<b>0x31</b>	Installed	Installed	Installed	Removed
<b>0x32</b>	Removed	Removed	Removed	Installed
<b>0x33</b>	Installed	Removed	Removed	Installed
<b>0x33</b>	Removed	Installed	Removed	Installed
<b>0x34</b>	Installed	Installed	Removed	Installed
<b>0x35</b>	Removed	Removed	Installed	Installed
<b>0x36</b>	Installed	Removed	Installed	Installed
<b>0x37</b>	Removed	Installed	Installed	Installed
<b>0x38</b>	Installed	Installed	Installed	Installed

**NOTE:** We will use the I<sup>2</sup>C Start Address of 0x2A in all examples shown in this manual.

## Communications:

[ControlEverything.com](http://ControlEverything.com) current monitoring devices support 4 commands. We have provided samples demonstrating communications in multiple languages, greatly simplifying development. All of our software samples begin with an I<sup>2</sup>C Start Address, and end with a valid checksum. All 4 commands are 8 bytes in length. All current monitoring controllers will ignore commands with an invalid checksum. Similarly, data is returned to the I<sup>2</sup>C Master with a valid checksum.

Our sample software will compute a checksum and compare it to the received checksum. Current monitoring is considered a high-noise application, it is our company policy to provide complete checksum based communications in all custom firmware to help reduce the chances of errant data communications. Our software samples should be consulted as the primary programming resource, as they provide complete working samples for 2-way communications with checksum validation. This guide should be used to help supplement the program examples we have provided.

Current data is represented using 24-bits (3 bytes) at 1ma resolution (the least significant bit = 1ma). Computing the current data is very easy:  $(MSB \times 65536) + (MSB2 \times 256) + LSB = \text{Milliamps}$ . So if the result value is 32,338, the total current draw is 32 amps and 338 ma. Despite this seemingly high resolution, the actual value is usually within 5% or 3% (depending on controller model), which is more accurate than most commercial clamp meters we have tested. Further calibration can yield more accurate results (some of our testing has yielded an accuracy within 10ma of actual), we have pre-calibrated all controllers to match our master designs, we do not calibrate each controller individually.

Crosstalk (or channel bleed) does occur at higher amperages (20A and higher). Typically, the higher the current, the more crosstalk between signals. Crosstalk can account for a significant portion of the total error per channel. However, the crosstalk errors introduced

are typically insignificant when compared to the errors displayed by most commercial clamp meters we have tested. We typically test our controllers up to 20A unless otherwise noted. Any controllers rated for higher currents have not been validated for accuracy, and may require further calibration. Duration testing of several weeks has been performed on our controllers. There tends to be very little drift over time, making our current monitoring series controllers ideal for long-term energy management applications.

### **Matching Controllers to Current**

Every current sensing controller we manufacture is pre-calibrated to measure current over a specified range. For instance, a 20A controller is designed to measure a current over a range of 0 to 20A, but it can measure slightly higher currents to account for over-current conditions. Exceeding the Max current by more than 25% can cause damage to the CPU. For instance, exceeding 20A + 25% (25A) can cause permanent damage to the CPU. This is true of all current monitoring controllers we offer. Please note that we have a 25% safety margin designed into all of our controllers.

Because each controller is pre-tuned for a particular amperage range, we highly recommend using a controller that meets, but does not exceed your amperage range. For instance, if you need to measure current on a 20A circuit, using a 30A controller is not advised. The correct match would be a 20A controller. Never over-rate your controller, doing so will cause an unnecessary loss of resolution across the intended tuning range.

### **Command Summary:**

Command 1: Reading Current

Command 2: Reading Device Information

Command 3: Reading Calibration Values

Command 4: Writing Calibration Values



## **Command 1: Reading Current**

[ControlEverything.com](http://ControlEverything.com) current monitoring controllers are constantly monitoring current. Our custom current monitoring CPU examines each channel and computes a current reading and stores this reading in a buffer. When current data is requested, the CPU instantly pulls the latest reading from the buffer and returns it to the I<sup>2</sup>C Master device. Current Monitoring controllers require 1/2 second to evaluate each channel. One-channel controllers refresh current readings every half second. Twelve-channel controllers refresh current readings every 6 seconds. These delays do not affect communication in any way; however, current readings could be up to 6 seconds old by the time they are communicated to the Master I<sup>2</sup>C controller.

The following section explains each byte of Command 1, followed by a sample implementation.

### **Byte 1: 0x2A I<sup>2</sup>C Start Address**

The I<sup>2</sup>C Start Address is determined by jumpers A0-A3 on the circuit board. Please see the “Addressing” table earlier in this guide.

### **Byte 2 and 3: Header Bytes**

The header bytes are used to identify current monitoring controllers, these bytes are fixed and cannot be changed. These bytes are specific to the current monitoring series of controllers, but have no other purpose.

### **Byte 4: Command 1: Read Current**

This byte 0x01 calls command 1 for reading the current from the controllers.

## **Bytes 5 & 6: Channel Range**

Bytes 5 and 6 of this command indicate the range of channel numbers you would like to read. Valid decimal values are 1 to 12 (0x0 to 0xC) for these parameters. Here are a few examples to show the proper use of bytes 5 and 6:

### **Read Channel 1 Only:**

Byte 5: 1

Byte 6: 1

3 Bytes + Checksum will be Returned

### **Read Channels 1 through 12:**

Byte 5: 1

Byte 6: 12

36 Bytes + Checksum will be Returned

### **Read Channels 11 and 12 Only:**

Byte 5: 11

Byte 12: 12

6 Bytes + Checksum will be Returned

**Byte 7: Reserved, Always Use 0**

**Byte 8: Reserved, Always Use 0**

**Byte 9: Checksum Calculation (See Below)**

## Checksum Calculation:

Checksum calculation is very easy. If you are working with a Microcontroller, Checksum should be defined as a Byte, (avoid word or integer definitions). By defining the Checksum value as a Byte, data is automatically rolled over and truncated to the 8 least significant bits. Checksum is a very easy calculation: It is the 8-Bit sum of Bytes 2-8, Byte 1 is omitted from the Checksum Calculation.

### Example:

Byte 1: 0x2A = 42 Decimal	I <sup>2</sup> C Start Address
Byte 2: 0x92 = 146 Decimal	Header Byte 1
Byte 3: 0x6A = 106 Decimal	Header Byte 2
Byte 4: 0x1 = 1 Decimal	Command Number 1: Read Current
Byte 5: 0x1 = 1 Decimal	Start Channel Number
Byte 6: 0xC = 12 Decimal	Stop Channel Number
Byte 7: 0x0 = 0 Decimal	Reserved Byte
Byte 8: 0x0 = 0 Decimal	Reserved Byte
Byte 9: 0xA	Checksum (Least Sig. Byte, Sum of Bytes 2 to 8)

**Note:** Byte 1 is NOT Included in the Checksum Calculation

146+106+1+1+12+0+0 = 266 Decimal

The 16-Bit Value of 266 in Binary Looks Like this:

MSB:00000001:00001010:LSB

The Least Significant Byte of 266 = 00001010 = 10 (Decimal)

10 Converted to Hexadecimal = 0xA

Byte 9: 0xA (or 10 if you prefer decimal values)

**Note:** This page represents our standard method of calculating checksums for data transmission and reception. When receiving data, the number of bytes received will change, but the checksum calculations remain the same for all communications. The first byte of a received packet will always be included in the checksum calculation.

### Receiving Data from Command 1:

Command 1 always returns 3 Bytes per Channel in consecutive ascending order, one additional byte is also included indicating the 8-bit checksum of all bytes sent. In the sample above, 36 Bytes (+ 1 Checksum) would be returned. The return data will look like this:

```
0x01,0x00,0x05,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x06
```

### Let's break these data into pieces and identify each segment:

```
0x01,0x00,0x05, Channel 1: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 2: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 3: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 4: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 5: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 6: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 7: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 8: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 9: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 10: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 11: MSB,MSB2,LSB
0x00,0x00,0x00,      Channel 12: MSB,MSB2,LSB
0x06                  Checksum
```

### Calculating Current:

Calculating current values for each channel is very easy:

$$\text{Current} = (\text{MSB} * 65536) + (\text{MSB2} * 256) + \text{LSB}$$

$$\text{Current} = (1*65536) + (0*256) + 5$$

$$\text{Current} = 65,541\text{mA} = 65 \text{ Amps } 541\text{ma for Channel 1}$$

Current calculations should be discarded if the received checksum value does not equal the 8-bit sum of bytes for the first 12 channels.

### Command 1 Sample 2:

In this sample, we will demonstrate a 1 channel request using all decimal values:

**Send Bytes:** 42, 146, 106, 1, 1, 1, 0, 0, 255

42 I<sup>2</sup>C Start Address  
146 Header Byte 1  
106 Header Byte 2  
1 Command Number  
1 Start Channel  
1 End Channel  
0 Reserved Byte  
0 Reserved Byte  
255 Checksum (146+106+1+1+1+0+0 = 255)

**Receive Bytes:** 0, 5, 112, 117

0 MSB  
5 MSB2  
112 LSB  
117 Checksum  
0+5+112=117

### Checksum Validation:

117 Calculated = 117 Data Received so Data Packet is Valid  
If these values are not equal, data packet should be rejected

### Converting to a Current Reading:

0 Current MSB  
5 Current MSB2  
112 Current LSB

Current = (MSB x 65536) + (MSB2 x 256) + LSB

Current = (0 x 65536) + (5 x 256) + 112

Current = 1392mA

Current = 1 Amp, 392ma

### Command 1 Sample 3:

In this sample, we will read channels 1,2, and 3 using decimal values.

**Send Bytes:** 42, 146, 106, 1, 1, 3,0,0,2

- 42 I<sup>2</sup>C Start Address
- 146 Header Byte 1
- 106 Header Byte 2
- 1 Command Number
- 1 Start Channel Number
- 3 End Channel Number
- 0 Reserved Byte
- 0 Reserved Byte
- 1 LSB of 257 = Checksum  
((146+106+1+1+3+0+0=257) AND 255)

257 converted to a 8-Bit using the "And" Math function

Checksum = 257 AND 255 (Extracts Lower 8 Bits) = 1

**Receive Bytes:** 0,5,112,0,10,137,0,15,45,68

- 0 Current MSB Channel 1
- 5 Current MSB2 Channel 1
- 112 Current LSB Channel 1
- 0 Current MSB Channel 2
- 10 Current MSB2 Channel 2
- 137 Current LSB Channel 2
- 0 Current MSB Channel 3
- 15 Current MSB2 Channel 3
- 45 Current LSB Channel 3
- 68 Checksum

Current Channel 1 =  $(0 \times 65536) + (5 \times 256) + 112 = 1,392\text{ma}$

Current Channel 2 =  $(0 \times 65536) + (10 \times 256) + 137 = 2,697\text{ma}$

Current Channel 3 =  $(0 \times 65536) + (15 \times 256) + 45 = 3,885\text{ma}$

## Command 2: Read Device Identification Data

This command is used to identify various features of your current monitoring controller. This is a fixed command, there are no parameters for this function:

**Send Bytes:** 42, 146, 106, 2, 0, 0, 0, 0, 254

- 42 is the I<sup>2</sup>C start address
- 146 is the header byte 1
- 106 is the header byte 2
- 2 Command 2: Read Device Data
- 0 Reserved byte
- 0 Reserved byte
- 0 Reserved byte
- 0 Reserved byte
- 254 is the CheckSum (146+106+2+0+0+0+0=254)

**Receive Bytes Sample:** 1, 5, 1, 1, 0, 0, 8

The controller will respond with 7 bytes indicating various parameters of the controller:

- 1 Indicates sensor type, see last page of this guide for a complete list of supported sensors
- 5 Indicates the max current supported by this device, in this case the max current supported by this controller is 5A.
- 1 Indicates the max no of channels supported by this device, in this case the number of channels is 1; however, valid return values include 1, 2, 4, 6, 8, and 12.
- 1 Indicate the firmware revision, in this case the firmware revision is 1.
- 0 Reserved Byte
- 0 Reserved Byte
- 8 Checksum (1+5+1+1+0+0=8)

### Command 3: Read Calibration Value

This command is used to retrieve the calibration values from the controller. Altering the calibration values may improve accuracy. The default values recorded into each controller are based on our master designs, as we do not calibrate each controller individually. All controllers are calibrated up to 20 Amps Max, controllers with ratings above 20 Amps may require manual calibration.

**Send Bytes:** 42,146,106,3,1,1,0,0,1

- 42 I<sup>2</sup>C Start Address
- 146 Header Byte 1
- 106 Header Byte 2
- 3 Command Number 3 Reads Calibration Values
- 1 Start Channel Number (1-12)
- 1 End Channel Number (1-12)
- 0 Reserved Byte
- 0 Reserved Byte
- 1 Checksum (146+106+3+1+0+0=0)

**Receive Bytes:** 00, 155,155

Example Above Reads 3 Bytes for Channel 1. Data is returned as a 16-Bit value + Checksum

- 0 Current Calibration MSB Channel 1
- 155 Current Calibration LSB Channel 1
- 155 Checksum

Calibration Value = (MSB x 256) + LSB

Calibration Value = (0 x 256) + 155

Calibration Value = 155



### Command 3 Example 2:

Reading the calibration value from multiple channels at one time is also very easy, this sample demonstrates how to read calibration data from channels 1, 2, and 3.

**Send Bytes:** 42,146,106,3,1,3,0,0,3

- 42 I<sup>2</sup>C Start Address
- 146 Header Byte 1
- 106 Header Byte 2
- 3 Command Number 3 Reads Calibration Values
- 1 Start Channel Number (1-12 Valid Range)
- 3 End Channel Number (1-12 Valid Range)
- 0 Reserved Byte
- 0 Reserved Byte
- 3 CheckSum  $((146+106+3+1+3+0+0) \text{ AND } 255) = 3$

**Receive Bytes:** 0, 155, 00,155,00,157,211

- 0 Current Calibration Value MSB Channel 1
- 155 Current Calibration Value LSB Channel 1
- 0 Current Calibration Value MSB Channel 2
- 155 Current Calibration Value LSB Channel 2
- 0 Current Calibration Value MSB Channel 3
- 157 Current Calibration Value LSB Channel 3
- 211 Checksum  $((0+155+0+155+0+157) \text{ AND } 255)=211$

Calibration Value Channel 1 =  $((0 \times 256) + 155) = 155$

Calibration Value Channel 2 =  $((0 \times 256) + 155) = 155$

Calibration Value Channel 3 =  $((0 \times 256) + 157) = 157$

#### **Command 4: Write Calibration Data**

This command is used to store current sensor calibration data. Before using this command, we strongly suggest users make only small changes to the current settings, as the settings within the controller should be very close to a usable calibration value. We do not calibrate each controller individually, the values stored in your controller are the values we use in our reference designs. In some cases, calibration at the software level should be considered, as our controllers only allow for linear calibration. Our current monitoring test equipment is limited to 20 Amps, controllers above 20 Amps may benefit from a non-linear calibration scheme, though we have been unable to validate this theory.

**Note:** This command is able to pass only ONE calibration value. This calibration value may be applied to one channel or a range of channels; however, this command cannot be used to store a different calibration value into each channel as a single command. If your controller requires a different calibration value for each channel, this command should be called multiple times using the same Start and End channels. No data is returned with this command.

#### **Choosing a New Calibration Value:**

As we mentioned before, the internal calibration value should be close to the actual reading. Raising the default calibration value will increase the amperage readings from the controller. Similarly, lowering the default calibration value will decrease the amperage readings from the controller.

**Send Bytes:** 42,146,106,4,1,1,0,150,152

This sample demonstrates the storage of calibration value of 150 into Channel Number 1:

- 42 I<sup>2</sup>C Start Address
- 146 Header Byte 1
- 106 Header Byte 2
- 4 Command Number 4 Stores Calibration Data
- 1 Start Channel Number (1-12)
- 1 End channel number (1-12)
- 0 Calibration MSB Value
- 150 Calibration LSB Value
- 152 CheckSum  $((146+106+4+1+1+0+150) \text{ AND } 255) = 152$

**Send Bytes:** 42,146,106,4,1,3,0,150,154

This sample demonstrates the storage of calibration value of 150 into Channel Numbers 1 through 3:

- 42 I<sup>2</sup>C Start Address
- 146 Header Byte 1
- 106 Header Byte 2
- 4 Command Number 4 Stores Calibration Data
- 1 Start Channel Number (1-12)
- 3 End channel number (1-12)
- 0 Calibration MSB Value
- 150 Calibration LSB Value
- 152 CheckSum  $((146+106+4+1+3+0+150) \text{ AND } 255) = 154$

## Sensor Types:

Current monitoring controllers know exactly what sensor they are working with. When querying device information data (found earlier in this guide), the controller will return a "Type" value shown below so your software can identify the type of sensor if necessary.

Supported Sensors:	Type:
DLCT03C20	1
DLCT27C10	2
DLCT03CL20	3
OPCT16AL	4