Editing Custom CANBUS Messages

One of the unique features of the Orion BMS product line is that the operator has effectively complete control over what CANBUS messages are transmitted and the contents they contain. This is incredibly helpful when interfacing with existing equipment that may not have configurable CANBUS protocols or structures, as it allows the BMS to adapt to existing standards rather than forcing other existing equipment to change.

Disclaimer: This document assumes the operator has at least a basic understanding of how CANBUS works (both from a physical standpoint and from a protocol standpoint). If that is not the case, there are a number of very good resources available online from various sources that provide excellent overviews of these topics. It is strongly suggested that the operator review some of these resources beforehand in order to get the most out of this document.

Note: Hexadecimal (representation of numbers with 16-base) is very common in this field of work and is used extensively throughout this document. **Please be aware that items prefaced with "0x" refer to them as being represented in hexadecimal.**

Some Orion BMS products allow for more customizable CANBUS messages than others, but at a minimum they all support at least 5 CANBUS messages that can be configured by the operator.

Conceptual Overview

Virtually every aspect of these messages can be configured. This includes the following aspects:

- **Message Identifier:** This is how messages are differentiated on the network. A CANBUS message identifier can either be configured as a "Standard" ID (which is default) that allows for values between 1 and 0x7FF (11 bits). CAN2.0B (which all Orion products support) allows for extended message IDs up to 29 bits (so values 0x1 through 0x1FFFFFFF). Both standard IDs and extended IDs can coexist on the same CANBUS network.

- **Message Length**: This is a value from 0 to 8 that indicates how many bytes are contained in this particular message. All CANBUS messages must have a length specified within this range.
- **Speed:** Also known as transmission frequency, this refers to the period (in milliseconds) of the message, or in other words, the amount of time that transpires between each time the message is transmitted. All custom CANBUS messages on Orion BMS products are transmitted at regular intervals and this is how the operator specifies what that delay period is.
- **Direction:** This specifies whether a message is configured to be received (listened for by the BMS) or transmitted (actively sent out by the BMS). Different data is available for messages depending on whether they are transmitted or received.
- **Message Contents:** This item is more vague, but it refers to the actual contents or parameters that are included in each CANBUS message (such as battery state of charge, pack current, temperatures, etc). Most parameters and data that are calculated or tracked by the BMS can be transmitted.
- **CANBUS Interface:** Some Orion products have 2 CANBUS interfaces (CAN1 and CAN2) whereas others only have 1 CANBUS interface (CAN1). Messages can be configured to be transmitted on either interface (or both).
- **Input Requirements:** Orion BMS products typically have at least 1 multi-purpose input pin as well as 1 or more power input sources. Individual CANBUS messages can be configured to only be transmitted if one of these inputs (or power sources) is energized. This allows for a message to be only transmitted if the BMS is in Charge Mode for example, or using one of the multi purpose input pins to transmit a message whenever it is energized.

In addition to being able to specify the contents of the CANBUS messages, the actual format of those contents can be further customized in a number of ways. These include:

- **Mathematical Operations:** Each message content or field can have additional mathematical operations performed to it prior to transmission (the list of these operations include multiplication, division and addition). This can be very helpful when trying to translate between the default units that the BMS uses and the units that the application is expecting. For example, a “Divide By 2” operation can be specified for the Pack State of Charge parameter so that instead of the BMS transmitting in 0.5% SOC increments, it would then transmit in 1% SOC increments.

  The arithmetic operations are performed in the order they are listed in the utility:

  First multiplication
  Second division
  Third addition

  NOTE: The BMS does not support “floating point” arithmetic operations in CANBUS

messages (meaning, it is not possible to divide by 1.5 or multiply by 2.7). These can still be accomplished though by alternate means. For example, it would be possible to scale a value first by multiplying it by 100 and then dividing it by 15 which would be equivalent to dividing it by 1.5.

- **Maximum / Minimum Values:** Some Orion products give the operator the ability to specify maximum and minimum values for message contents / fields. This can be helpful if a value must never exceed a particular amount (say an inveter voltage limit for example).

- **Byte Order:** Many message fields / parameters can be transmitted as 16 bit values (that is, sent as 2 bytes instead of 1 byte). When doing so, the order of the bytes must be specified so the receiver handles the data properly. This is referred to as “Big Endian” and “Little Endian”, and it dictates whether the bigger (more significant in value) byte is sent first or the smaller is sent first. These are very common terms in the computer industry.

- **Bit Order:** Like the Byte Order above, each individual byte sent can also be represented in 2 ways (with the most significant or highest value bit being sent first, or the least significant or lowest value bit being sent first). These are referred to as “Most Significant Bit” and “Least Significant Bit”, and are very common terms in the computer industry.

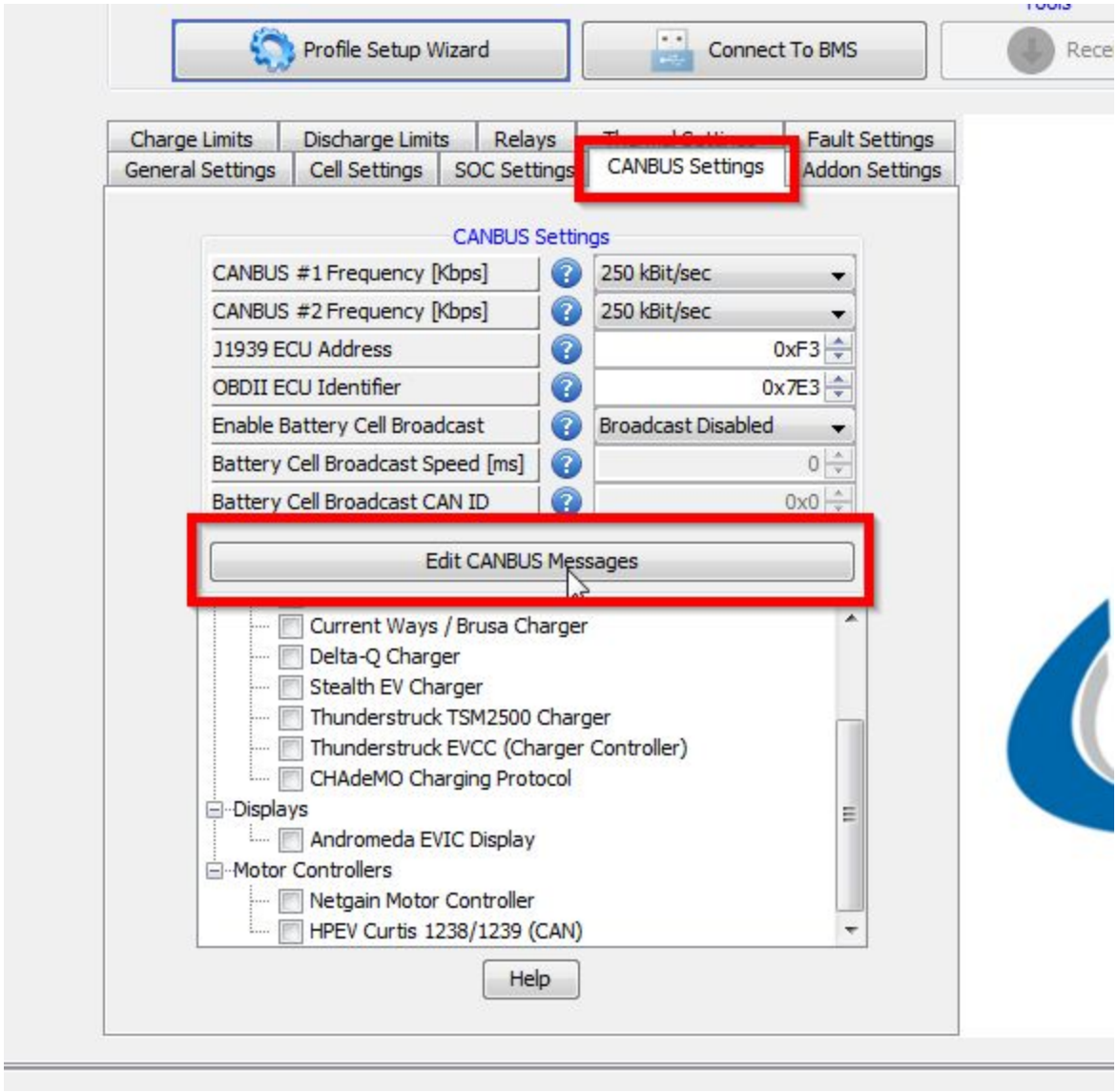
- **Field Length:** As referenced above, some data parameters need to be sent in more than 1 byte (any value over 255 requires more than 8 bits to represent). Specifying the field length (maximum 2 bytes) allows the operator to provide extra space for larger parameters.

- **Signed Value:** This modifier is used to indicate to the BMS whether the parameter in question is expected to potentially be both positive and negative. A good example of this would be temperatures (which can be positive or negative), or pack current. This then allows the “Maximum Value” and “Minimum Value” modifiers to be enforced properly.

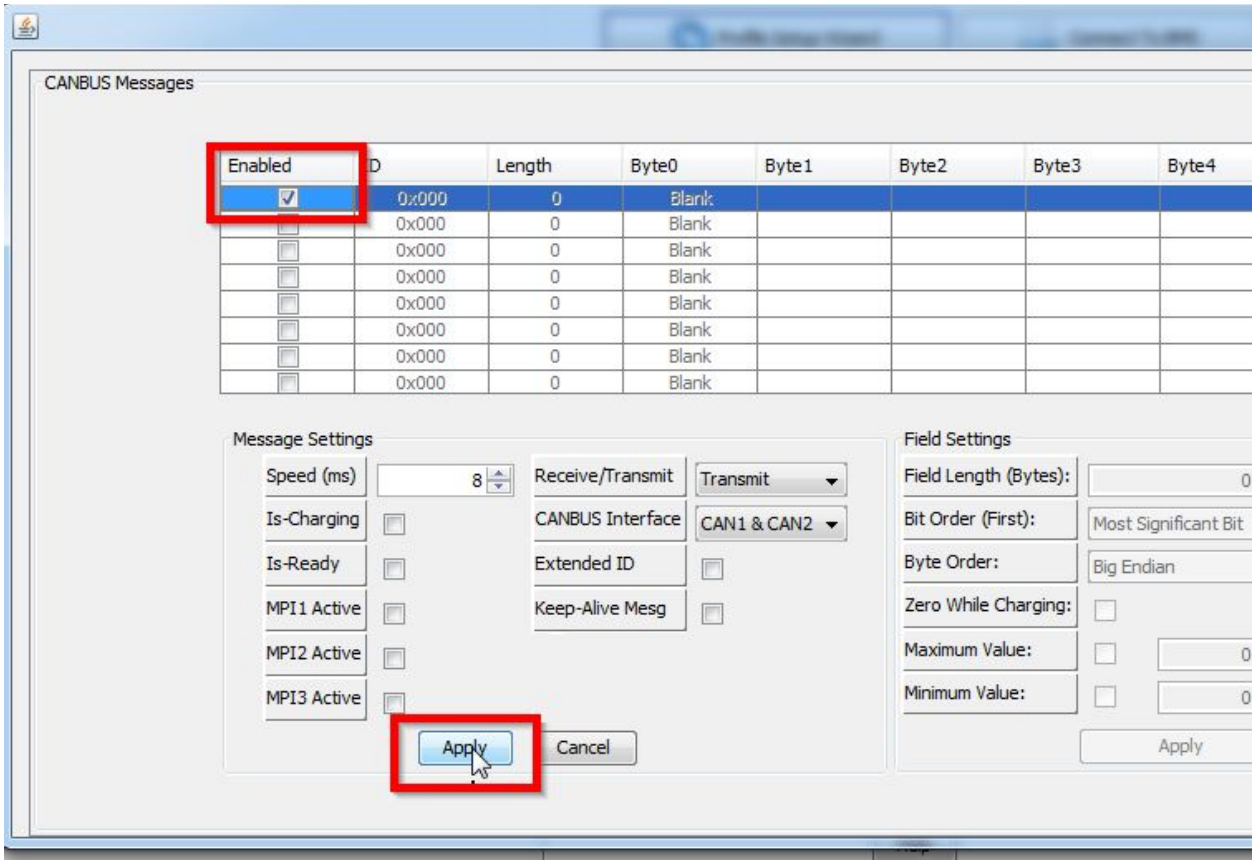
Setting Up a Sample Transmit CANBUS Message

Often the best way to illustrate a process is to demonstrate by means of an example. The following steps go through the process of creating an example CANBUS message that is transmitted by the BMS.

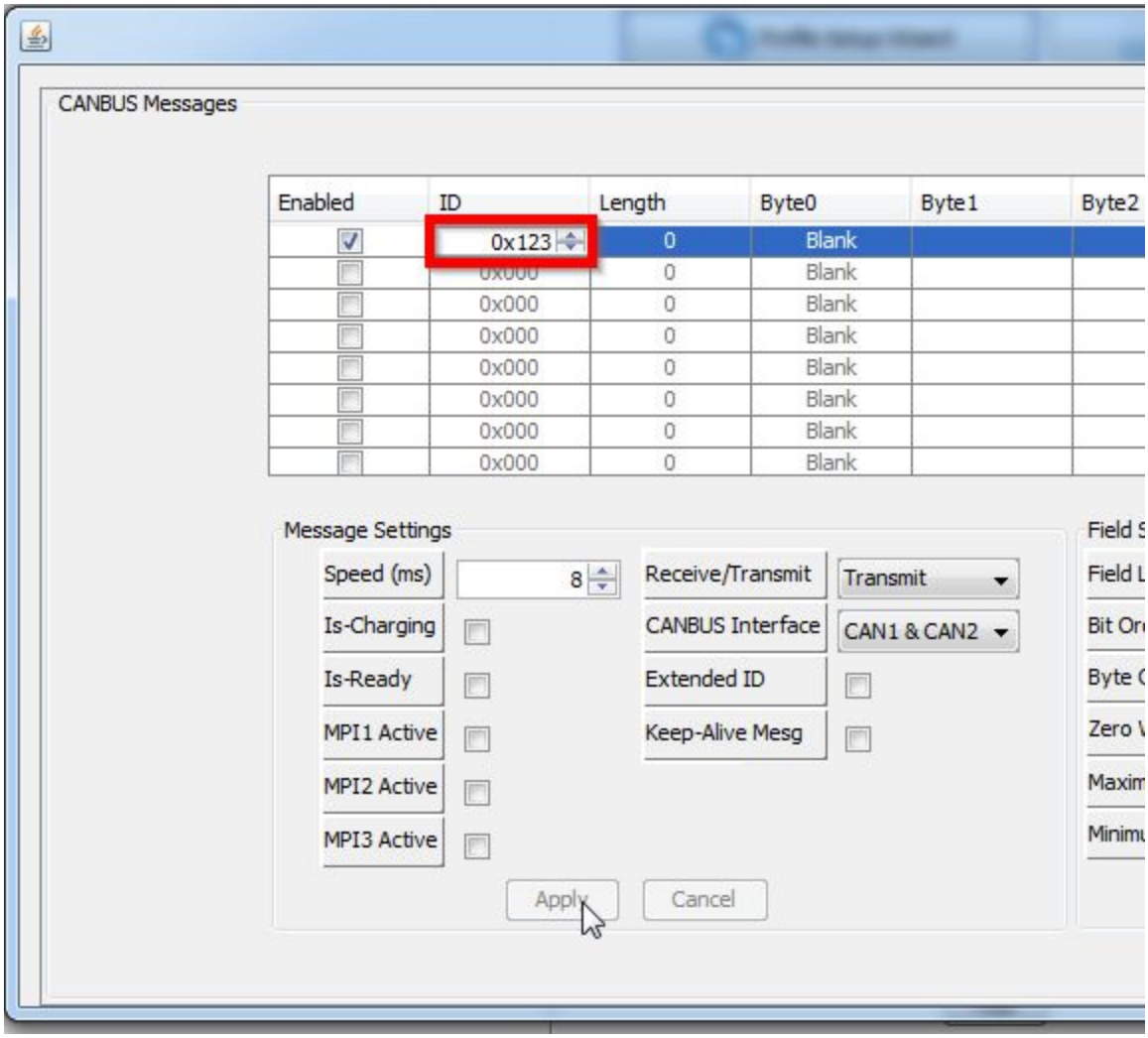
1) Open the Custom CANBUS Message Editor dialog by going to the “CANBUS Settings” tab in the BMS Utiltiy and clicking the button in the middle of the section called “Edit CANBUS Messages”. This should pop up a dialog that contains a number of rows and columns with several sections underneath for settings.



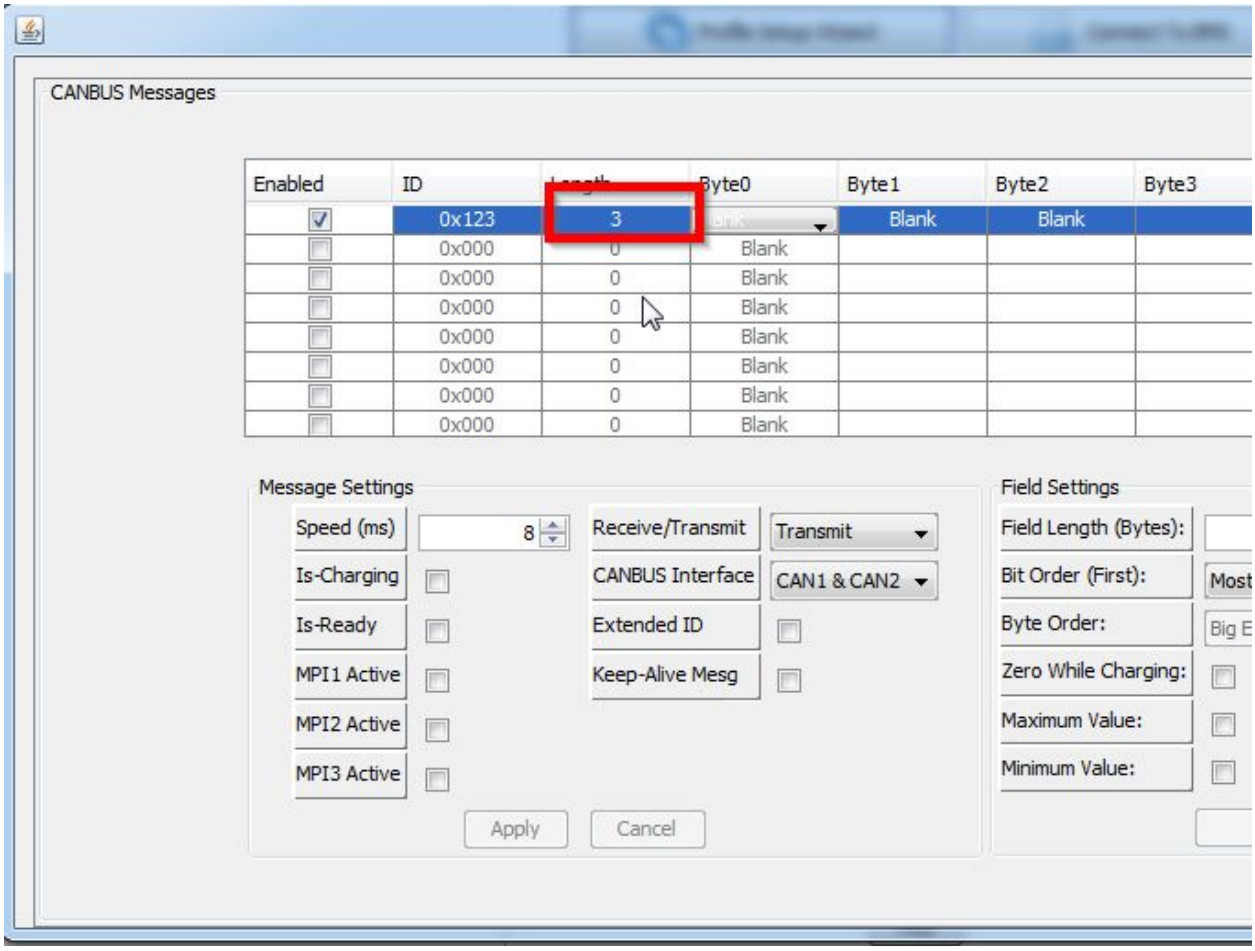
2) Click one of the checkboxes next to a message that is currently not in use, then hit “Apply” in the bottom left corner once it becomes available. This will make the ID field editable so that a Message Identifier can be entered.



3) Type in a Message Identifier into the “ID” column on that row. For this example, enter in ID “0x123”. Press TAB, or click out of the box for the change to take effect, then click “Apply” again in the bottom left corner.

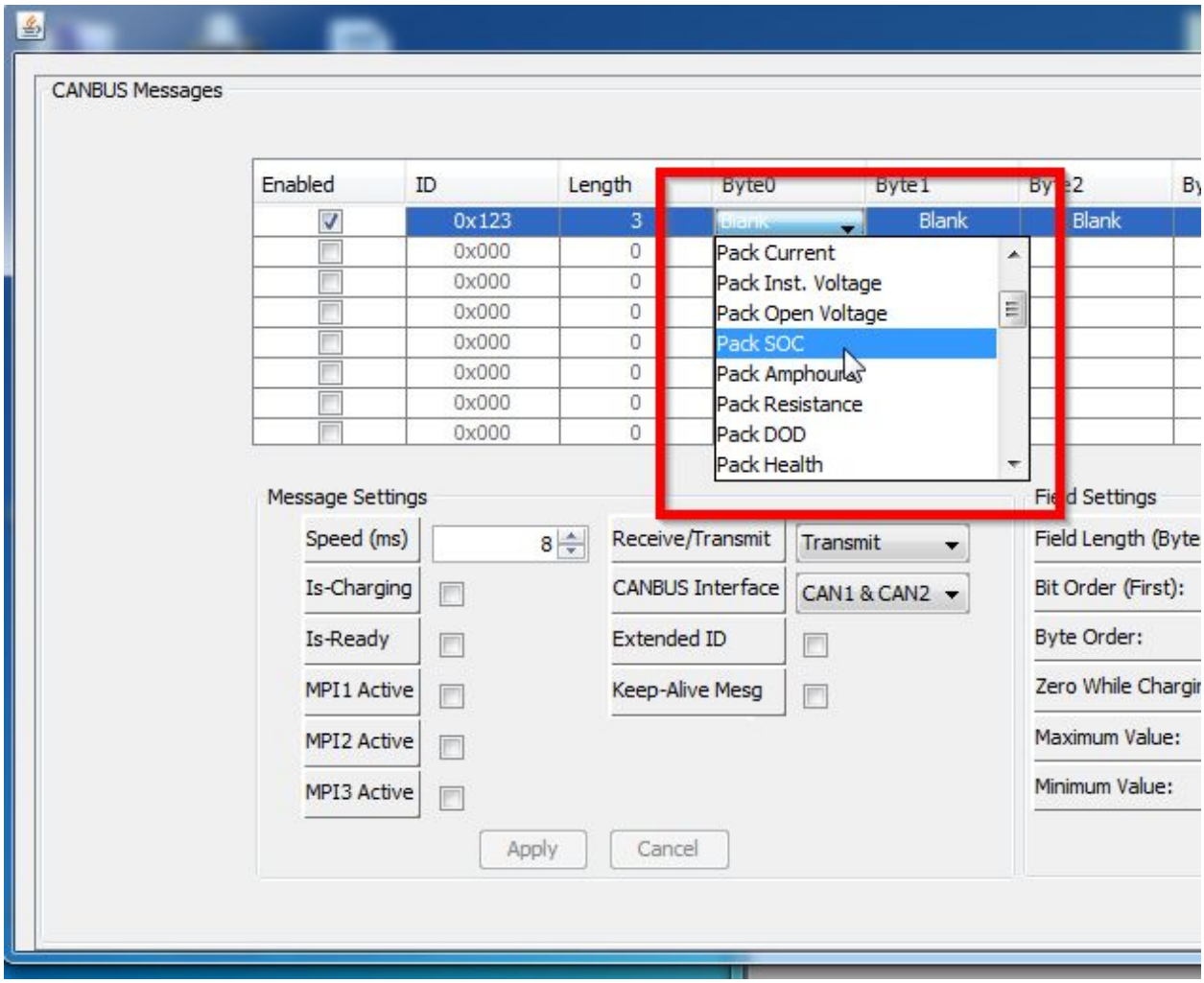


4) Next, set the “Length” column to 3 (an arbitrary value for this particular example). The value can either be typed in manually, or the up/down arrows on that item can be used to increment or decrement the length. Again, press “Apply” after making these changes.

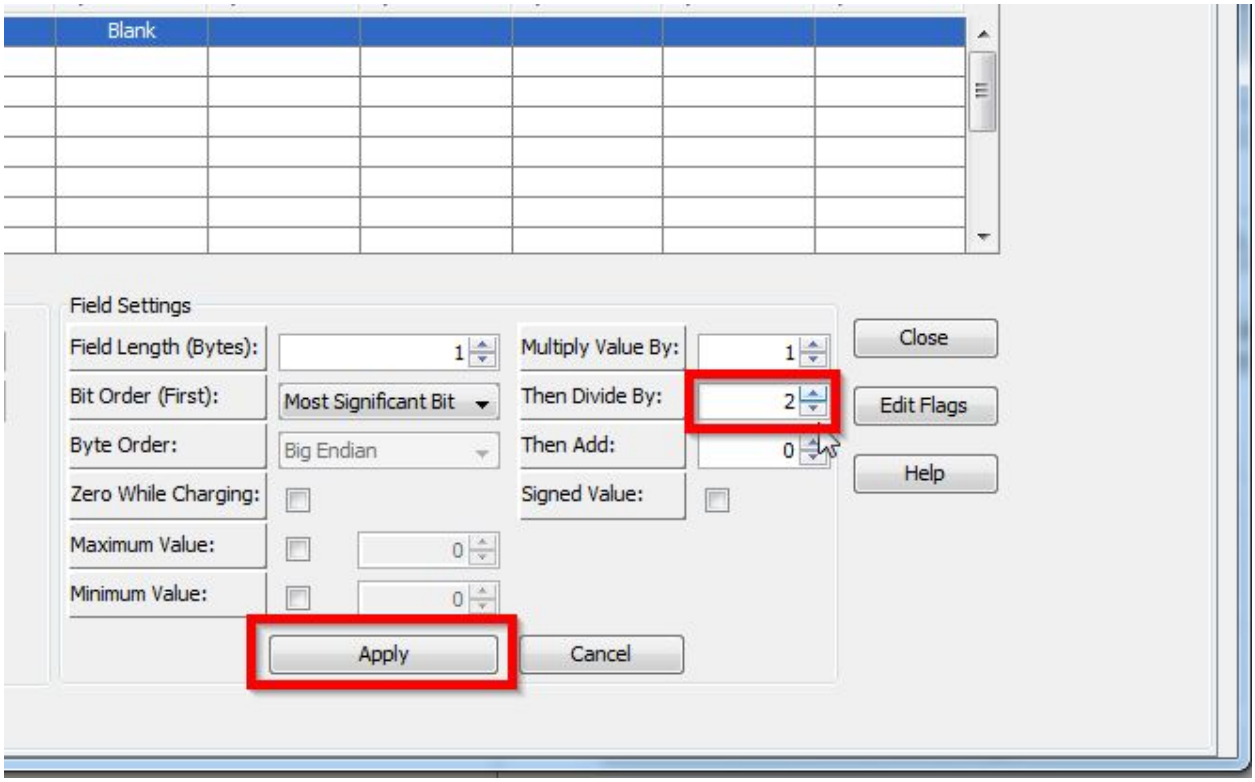


5) At this point the message contents (the “Byte0”, “Byte1”, etc columns) should now be active and editable. Click on the “Byte0” column (immediately to the right of the “Length” field that was just changed) and a drop down menu should now be visible. This menu represents all available parameters or fields that the BMS is able to transmit. The list is quite long, but a more detailed description of what each field represents is available in the Software Utility Manual (which can be accessed by going to “Help” in the top right corner of the utility and clicking “View Help”, then navigating to Appendix B).

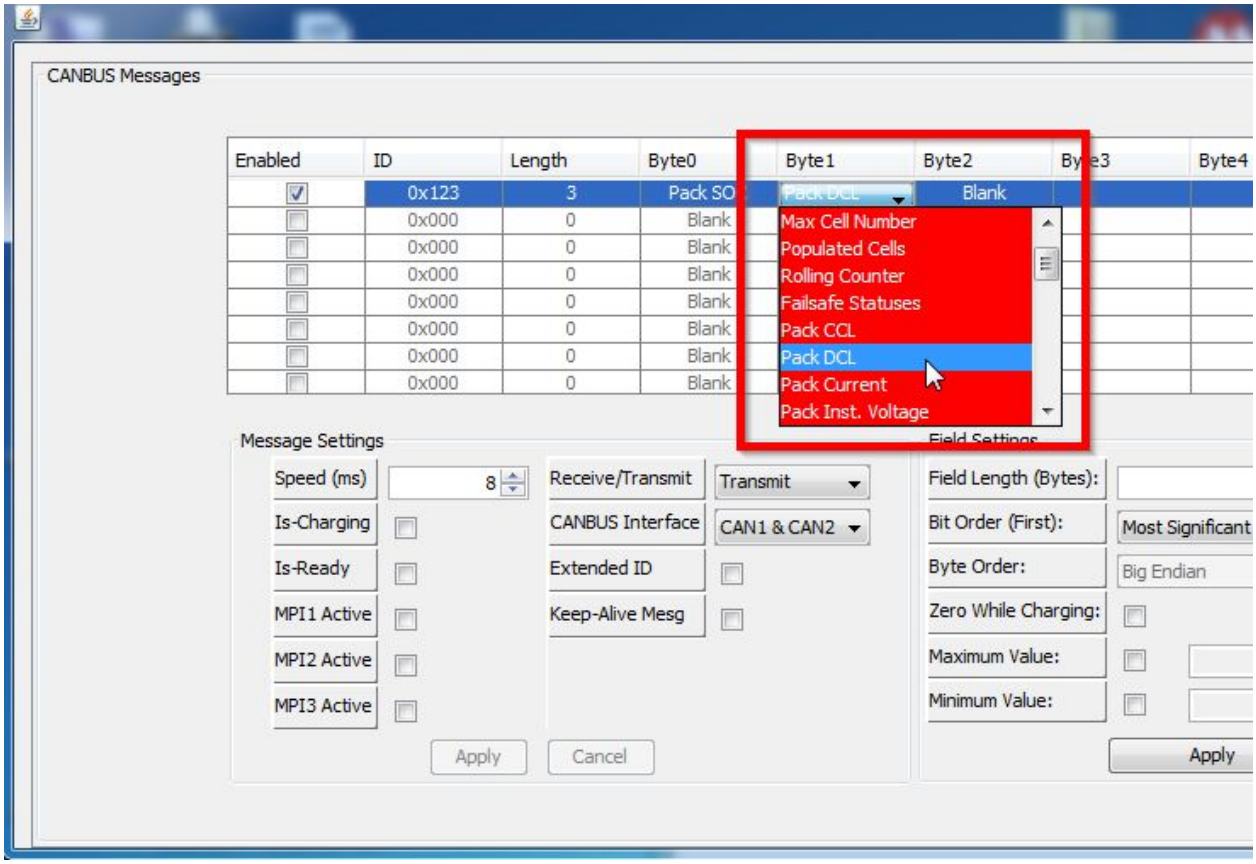
   For the purposes of this example, select “Pack SOC” from the drop down menu (which is the state of charge of the battery pack).



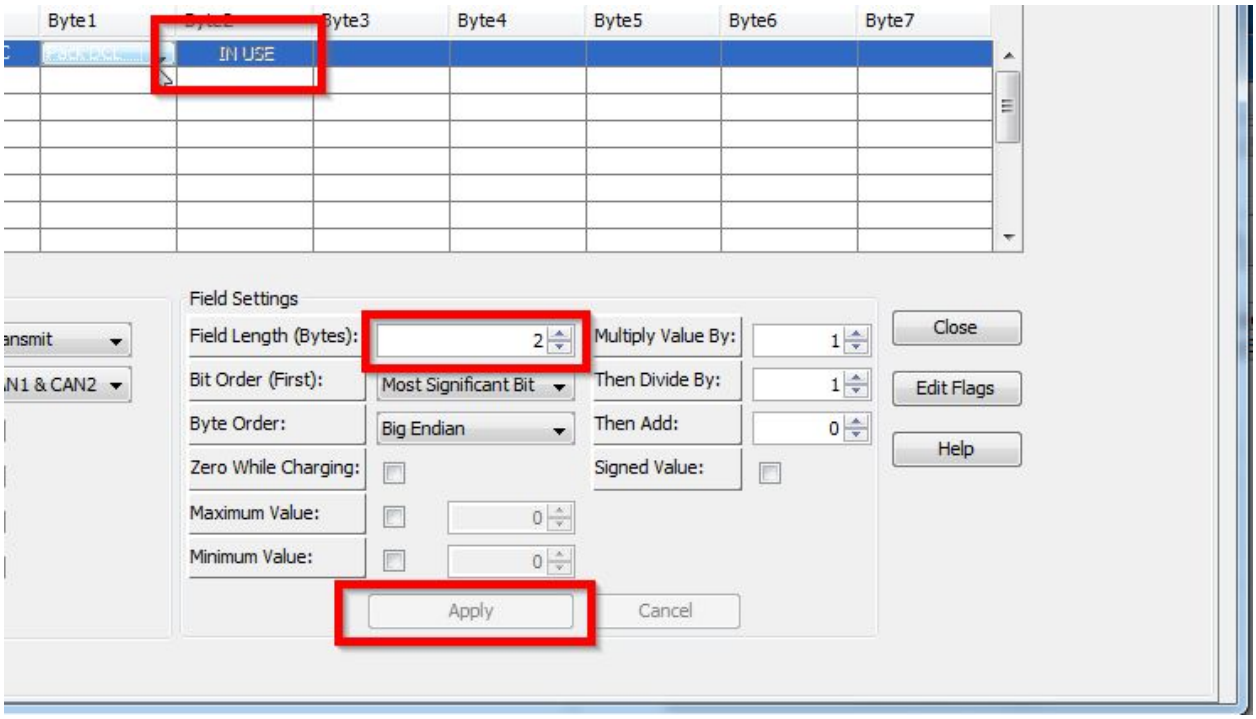
6) By default, the BMS will publish the pack state of charge in 0.5% increments (so an SOC of 75% would actually be represented on the CANBUS as 150). To scale this so it transmits in 1% increments, go to the bottom right corner of the screen where it says “Then Divide By” and enter in a value of 2 (the default is 1, no division). This tells the BMS to divide the parameter field by 2 before actually transmitting it in order to get it to the correct expected units. Then hit apply in the bottom right corner to save the changes.



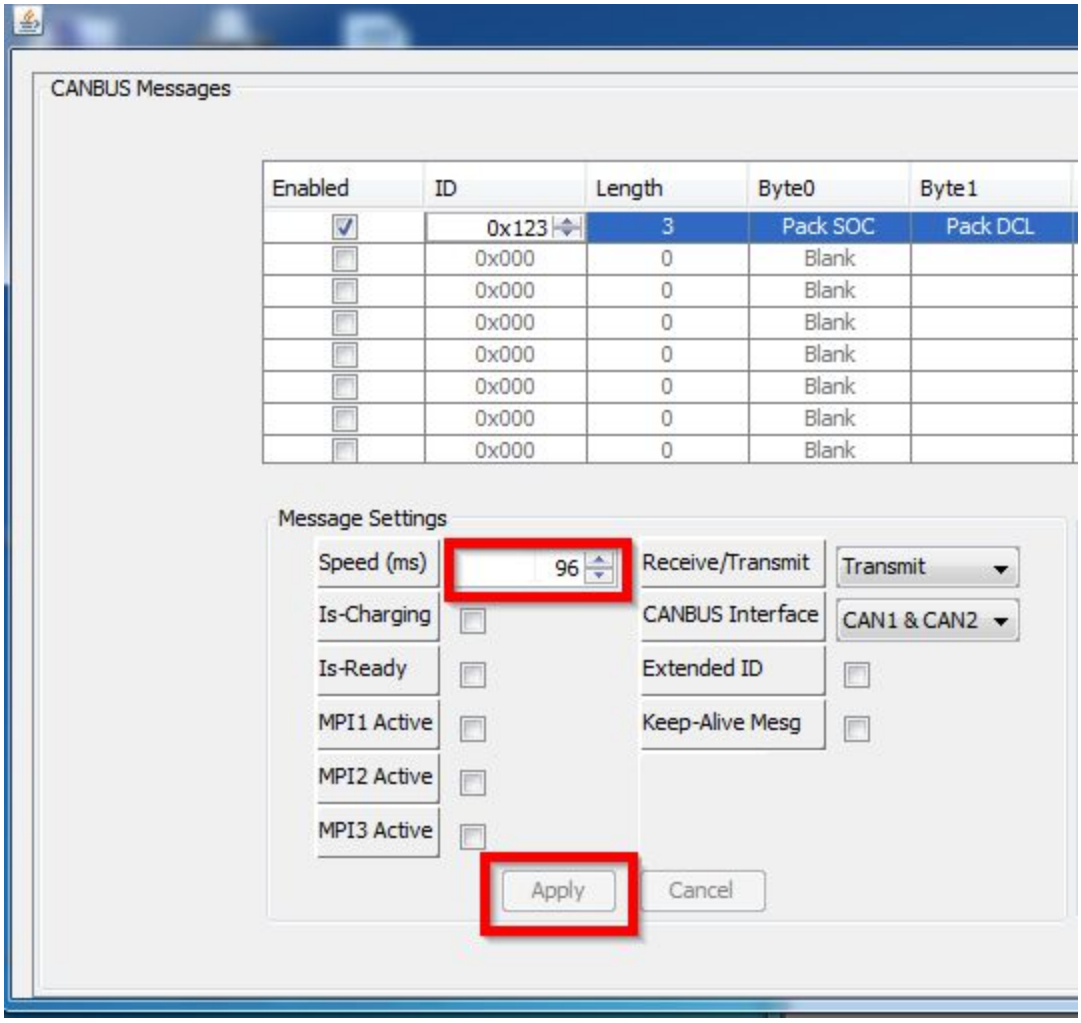
7) Next. click on the adjacent “Byte1” column entry and bring up the pop-up menu again. This time, select “Pack DCL” which is the pack discharge current limit (the amount of current the BMS is allowing to leave the battery pack at this particular moment).



The default units for Pack DCL are 1A, but the value itself needs to be transmitted as a 2 byte value to allow for discharge limits over 255A (more than 8 bits in size). To do this, change the “Field Length (Bytes)” value from 1 to 2 in the “Field Settings” section in the bottom right. This can either be done by manually typing in 2, or using the up arrow to increment the value and then hit apply to save the change. Note how the “Byte2” column now is grayed out and reads “IN USE”. This means that the Byte2 entry is serving as buffer space for the Pack DCL parameter carried over from Byte1, and allows the data to be sent as a 16 bit (2 byte) value.



8) Finally, the speed of the message needs to be set. In the bottom left corner there is a “Speed (ms)” option that can be changed in 8ms increments. For this example, set the value to 96ms and then hit Apply. This means the message will now be transmitted by the BMS once every 96ms.



Now that the message is complete, the CANBUS Message Editor dialog can be closed (by clicking the “Close” button) and then the profile can be transmitted to the BMS. After the profile is successfully uploaded to the BMS, the message set up should now be visible on the CANBUS. The first byte will be the SOC (in 1% increments) and the 2nd and 3rd bytes will be the pack discharge current limit (2 bytes in length). Additionally, the message should be getting transmitted once every 96 milliseconds by the BMS.

NOTE: If using the CANdapter (CANBUS to USB adapter) sold by Ewert Energy Systems, the BMS utility itself allows for it to be used to view live traffic on the CANBUS by going to the “Live CANBUS Traffic” tab in the utility and clicking the “Start” button. This can be a very helpful diagnostic / debugging tool for validating actual CANBUS message output.

Understanding Custom Flags

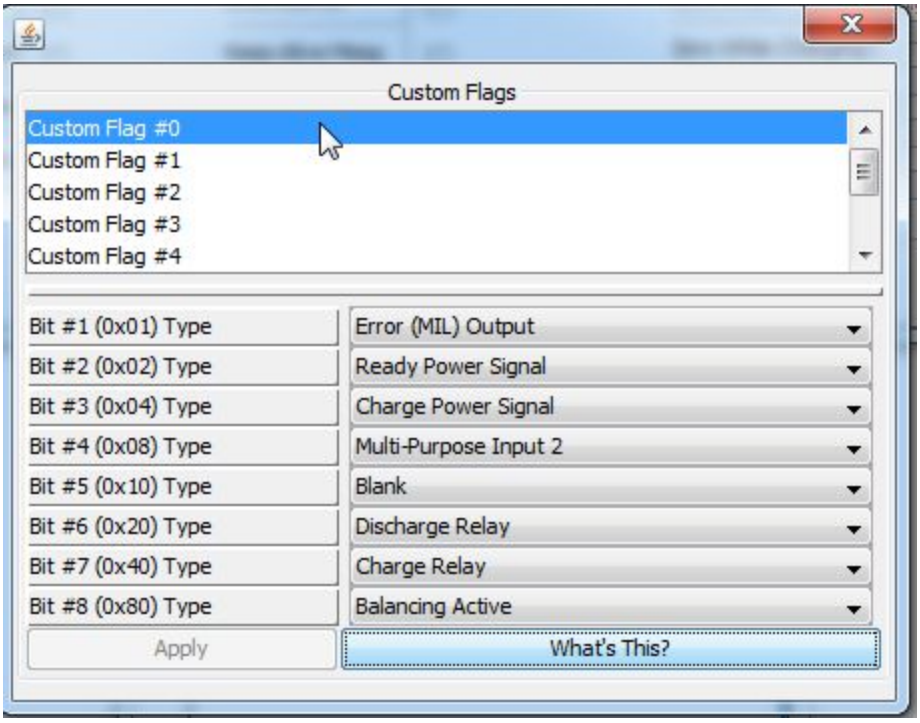
The example above illustrated how to set up a basic CANBUS message for regular transmission using 2 primary parameter fields. In addition to the parameters that are selectable in the drop-down menus for each byte, there are a few special parameters that need to be detailed separately:

- **Blank:** When selecting the “Blank” option, this means that this byte will always be sent as a “0” value. It is strongly suggested that “Blank” be selected for bytes that are not expected to hold a value (such as items that are reserved for future use, or are not yet implemented).

- **Constant Value:** Often times it is beneficial to be able to send a constant (non-changing) value in a particular byte for various reasons. This can be accomplished by selecting “Constant Value” in the drop-down menu for the field. **To specify the actual value that it will transmit, simply edit the “Then Add” value for this field.** For example, setting the field to “Constant Value” and then setting the “Then Add” value to 50 means that this particular field will always be transmitted with a value of 50 (or 0x32 in hex). **This is an incredibly helpful tool.**

- **Custom Flag**: There are many instances where specific applications require a particular bit sequence of flags for certain bytes. To accomplish this, the Orion BMS supports “Custom Flags” that allow the operator to configure what each individual bit in the byte correspond to (each flag has up to 8 bits in it, one byte per flag).

  To edit the Custom Flags, click on the “Edit Flags” button on the Custom CANBUS Message Editor. This will bring up the flag editor dialog:
  - First select the target flag that needs to be edited by clicking on it in the scroll menu at the top of the dialog.
  - Each flag has 8 bits, though bits can be permanently 1 or left blank by setting either “Constant 1” or “Blank” respectively.
  - Each bit has the corresponding hexadecimal value for it in the label to the left of the drop-down menu. This can be helpful for remembering what value each bit represents.
  - After completing edits, click the “Apply” button and close the dialog for the settings to be saved.



To use a custom flag in the custom message contents (fields), select the “Custom Flag” item from the drop-down menu and then hit “Apply” in the bottom right corner. Note that the “Then Add” box has now changed to “Custom Flag #” instead. **Enter in the number of the custom flag (that corresponds to the numbers in the “Custom Flag Editor”) that should be transmitted and then hit Apply again.** This will tell the BMS to transmit that custom flag in that particular byte whenever the message is broadcast.

Setting Up a Sample Receive CANBUS Message

In addition to messages that are transmitted at regular intervals, Orion BMS products also support receive messages. This allows the BMS to accept commands or parameters from third party devices for specific functions.

The process for setting up a Receive message is virtually identical to that of a Transmit message. Please see the steps listed above regarding setting up a transmit CANBUS message for details on how to start this process.

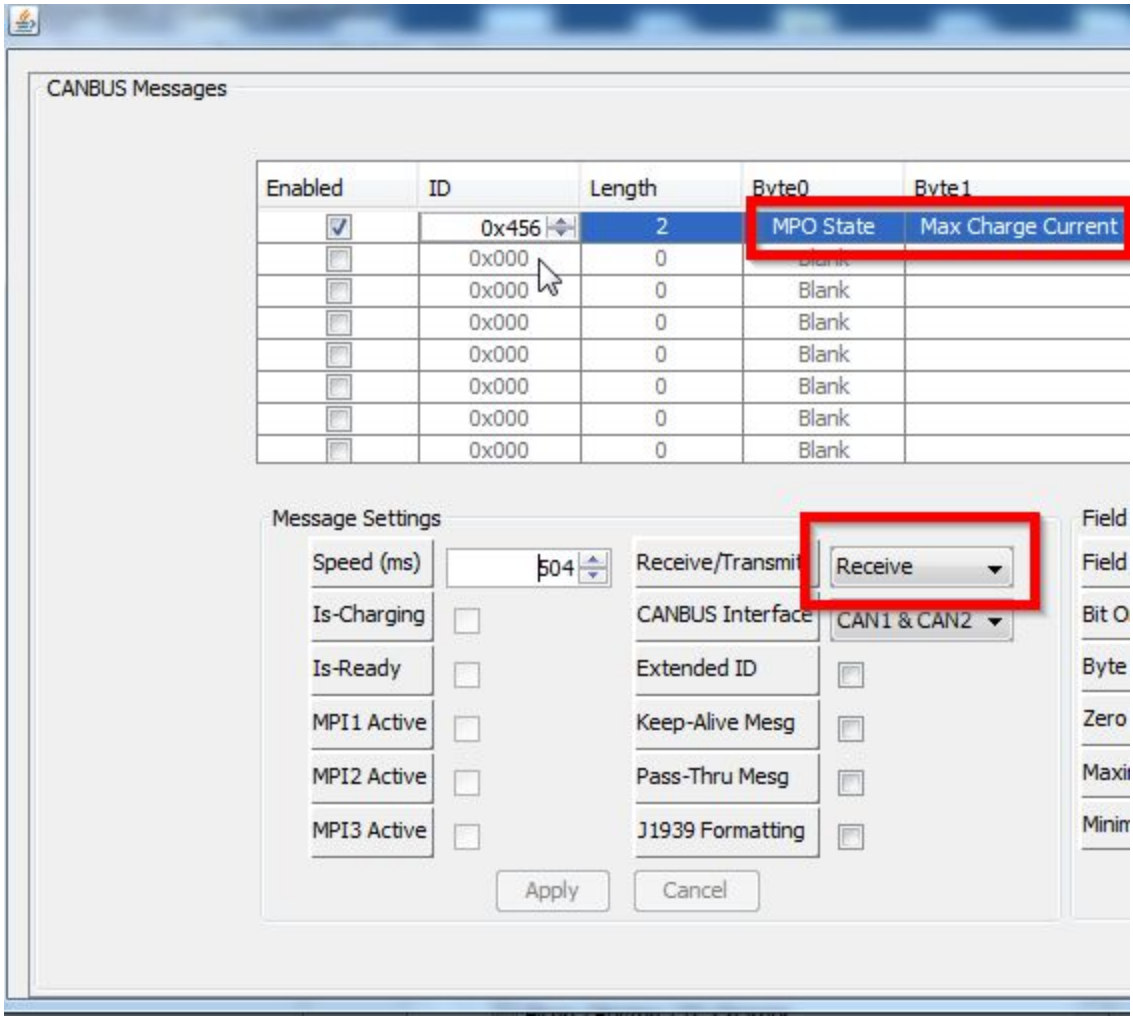
The key difference between Transmit and Receive messages are that Receive messages must be changed from “Transmit” to “Receive” state so the BMS knows to be looking for the ID rather

than transmitting it. This is available in the “Message Settings” section of the CAN message as a drop-down menu item.

Once configured, the BMS will be looking for that particular message on the CANBUS. Please note the following restrictions:

1) The BMS will only be looking for the CANBUS message on the CAN interface(s) that are selected on the message itself. For example, if only CAN1 is enabled for a receive message, it will ignore the message if it shows up on CAN2.
2) The BMS will match both the ID itself, the length and the ID type. Meaning, if a receive message is setup on ID 0x100 with a length of 4, it will ignore 0x100 messages with a different length. Likewise, if a message is sent as an extended ID (29 bit identifier) and the BMS is configured to be expecting a standard ID (11 bit identifier), it will also ignore the message even if the actual ID matches.

The following screenshot illustrates a sample Receive message set up in the CANBUS message editor:



In this example, message 0x456 is configured to be received by the BMS on both CAN1 and CAN2 interfaces as a standard ID (11 bit identifier) with a length of 2 bytes. Of those 2 bytes, byte 1 is being used to control the output state of MPO1 and byte 2 is being used to communicate a maximum allowed charge current limit to the BMS.

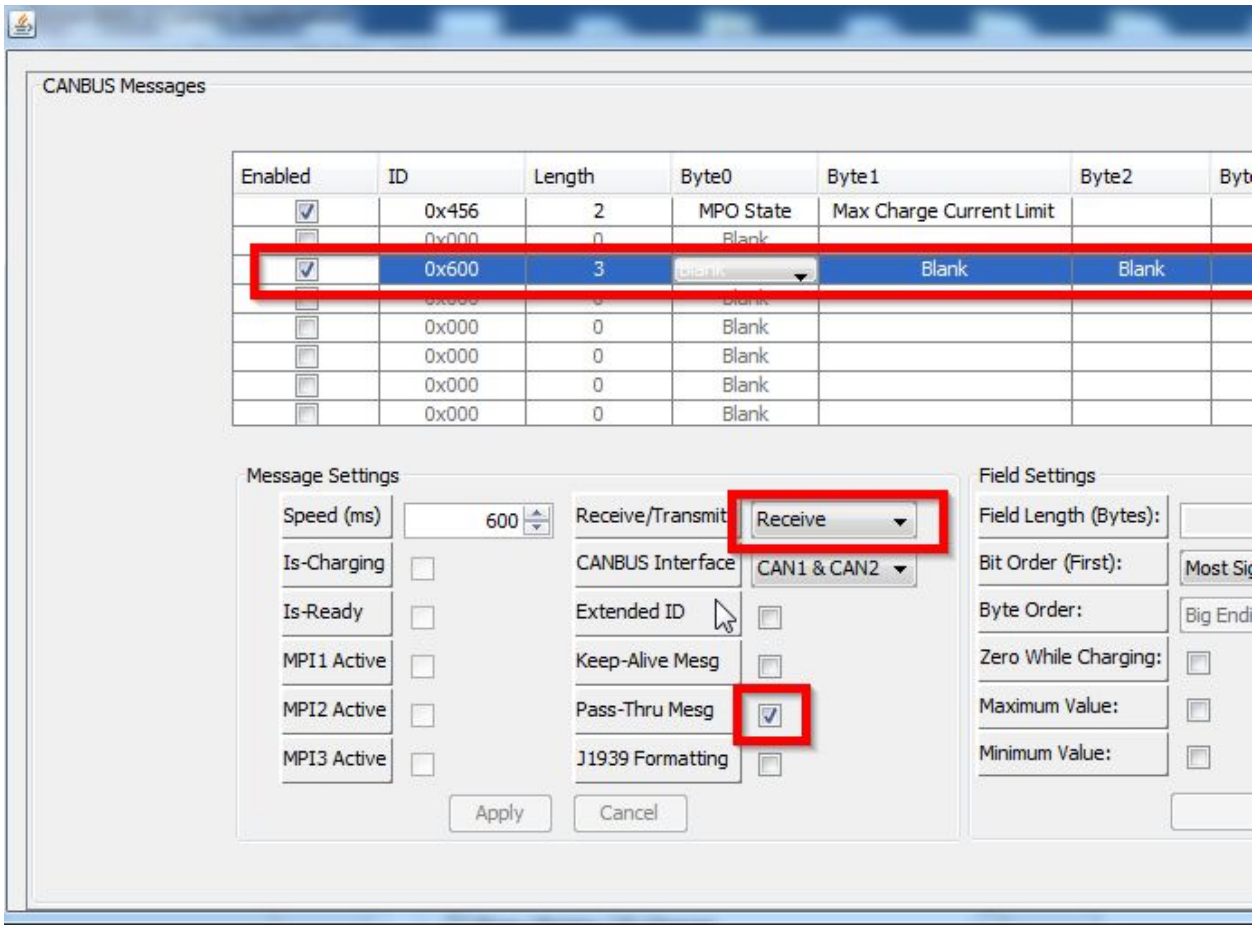
Passing Messages Between CAN Interfaces

On BMS products that have multiple CANBUS interfaces, the custom CANBUS message editor can be used to pass message IDs from one interface to the other, effectively acting as a bridge. This can be very helpful in applications like vehicles where there may be multiple CANBUS networks that the BMS is a part of that need to share information.

To configure a message as a pass-thru ID:

1) Set up a CANBUS Message as described previously in this document.
2) Configure the CANBUS message as “Receive” to indicate that the BMS is looking to receive the message from an external source.
3) Match the ID and length exactly to the data that is expected (it will only pass a message through that matches both the ID and data length provided).
4) Check the “Pass-Thru Mesg” checkbox in the CANBUS message editor to indicate to the BMS that it must propagate the message to the other CANBUS interface.

The example below illustrates a sample pass-thru message:



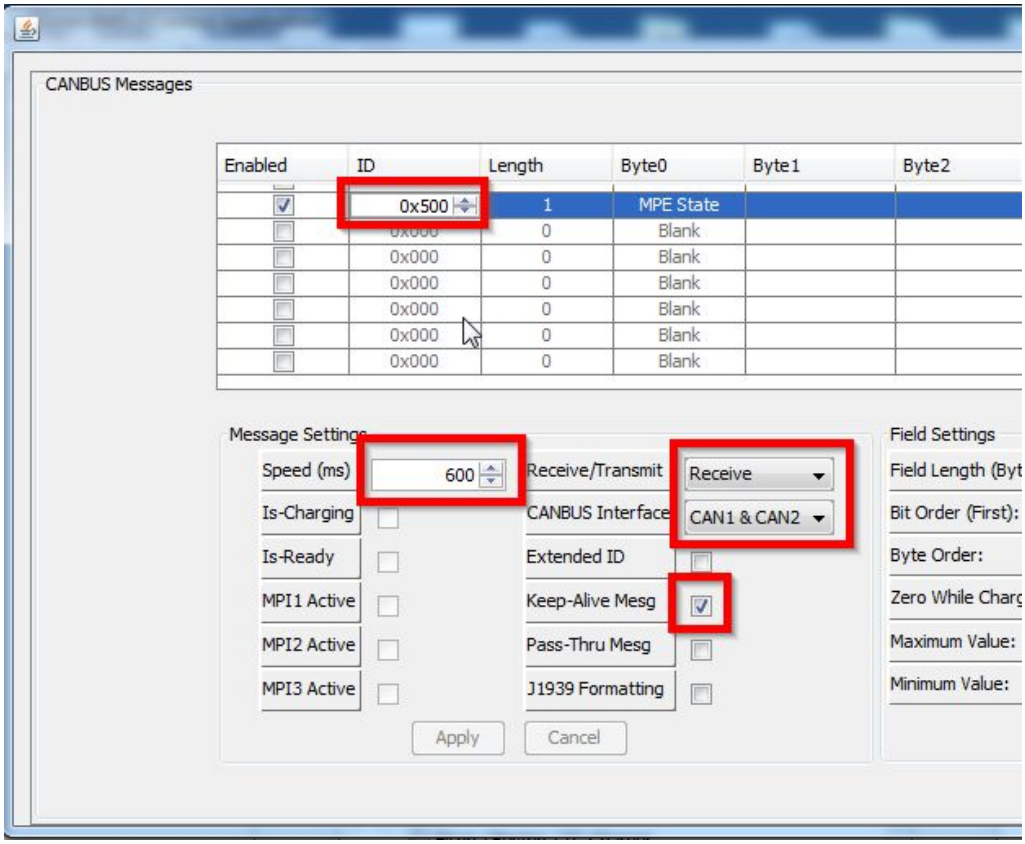
Using Receive Messages As Keep-Alive Signals

Often times it is desirable to set a fault code in the event that CANBUS message is lost between multiple controllers on a CANBUS network. This helps verify that communication is operating correctly, and allows the system to go into a fault mode if that connection is disrupted.

To support this, Receive messages can be configured to have a “Keep Alive” function through the BMS utility. The steps to do this are as follows:

1) Set up a CANBUS Message as described previously in this document.
2) Configure the CANBUS message as “Receive” to indicate that the BMS is looking to receive the message from an external source.
3) Match the ID and length exactly to the data that is expected (it will only pass a message through that matches both the ID and data length provided).
4) Check the “Keep-Alive Mesg” checkbox in the CANBUS message editor to indicate to the BMS that it will treat this as a keep-alive message.
5) Set the “Speed (ms)” value to the desired time that a fault should be generated if the message is not received within.

The following example illustrates how a Keep-Alive message can be set up:



In the above example, message 0x500 with a length of 1 is configured to be a Keep-Alive message that the BMS is expecting to see at least once every 600ms on either CAN1 or CAN2 interfaces. If more than 600ms has past since the last time message 0x500 with length of 1, the BMS will issue an External CANBUS Communication fault to indicate a timeout condition has occurred.

The actual behavior or results from this External CANBUS Communication fault are configurable through the BMS utility profile settings.