

# 505004 M-Flex 4-Input/ 4-Output CAN Module User Guide

CREATED:	C. PAWLAK	DATE:	11/24/2020
CHECKED:	S. JOHNSON	DATE:	12/28/2020
APPROVED:	S. JOHNSON	DATE:	01/08/2021
ECN:	19141E	DATE:	12/05/2025



## Table of Contents

1. Overview .....	2
2. Software Installation.....	4
3. Configuring and Programming the Module .....	4
3.1 Connecting to the Module with the Programming Tool .....	6
3.2 Updating the Base Software.....	7
3.3 Configuring the Module.....	9
3.3.1 XML Mode .....	14
3.3.1.1 Configuration .....	15
3.3.1.2 Logic.....	15
3.3.1.2.1 Operators.....	15
3.3.1.2.2 Module Parameters and Variables .....	19
3.3.2 CAN I/O Mode .....	26
3.3.2.1 Mode Description.....	26
3.3.2.2 Configuring for CAN I/O Mode.....	28
3.3.3 Hardware I/O Mode .....	30
3.3.3.1 Mode Description.....	30
3.3.3.2 Configuring for Hardware I/O Mode .....	30
3.3.4 Keypad Mode .....	33
3.3.4.1 Mode Description.....	33
3.3.4.2 Configuring for Keypad I/O Mode .....	35
3.3.4.3 Configuring the Marlin M-FLEX Keypad.....	38
3.4 Loading the User Program.....	39
4. LED Module .....	42
5. Low Power Mode .....	43
6. Quadrature Inputs.....	44

## 1. Overview

The 505004 Module is a configurable CAN I/O controller with four inputs, four outputs, and programmable logic. The module also has optional built-in support for external Marlin devices including other I/O controllers, keypads, and LED module. The module has an LED (see Figure 1) that indicates the module's power status. In CAN I/O mode, the module flashes the LED as a beacon to identify itself.

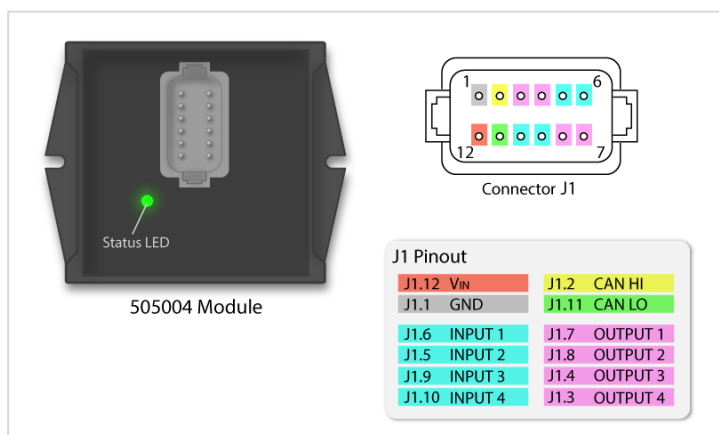
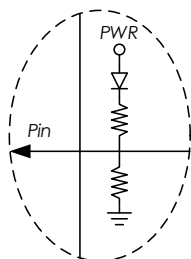


Figure 1

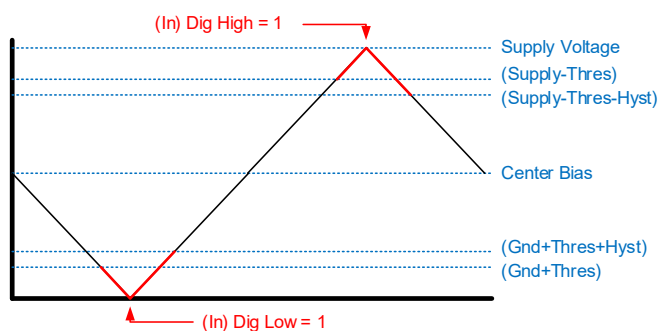
The input types supported by each of the input pins, as shown in Figure 1, are as follows:

	INPUT 1	INPUT 2	INPUT 3	INPUT 4
Digital – Active High	✓	✓	✓	✓
Digital – Active Low	✓	✓	✓	✓
Analog (0-37)	✓	✓	✓	✓
Frequency	✓	✓	✗	✗
PWM	✓	✓	✗	✗
Resistance	✗	✗	✗	✗
Current (0-20mA)	✗	✗	✗	✗
Quadrature X1	✓	✗	✗	✗
Quadrature X2	✓	✗	✗	✗
Quadrature X4	✓	✗	✗	✗
Quadrature B	✗	✓	✗	✗

### (\* NOTE: Input Circuit and Input Circuit Best Practices)



- Generated Analog Signal
- Analog Potentiometer (5k or less)
- On/Off Switch to Power (No Pull-Down)
- On/Off Switch to Gnd (No Pull-Up)
- 3-way Switch to Pwr/Gnd (No PU/PD)
- Frequency Input ( 0-5V, LS\_Output )
- PWM Duty Cycle ( 0-5V, LS\_Output )
- Quadrature Input ( 0-5V, LS\_Output )



The output types supported by each output pin, as shown in Figure 1, are as follows:

	OUTPUT 1	OUTPUT 2	OUTPUT 3	OUTPUT 4
High Side Digital – Active High	✓	✓	✓	✓
Low Side Digital – Active Low	✓	✓	✓	✓
High Side PWM – Closed Loop (Current)	✓	✓	✓	✓
High Side PWM – Open Loop	✓	✓	✓	✓
Low Side PWM – Closed Loop (Current)	✓	✓	✓	✓
Low Side PWM – Open Loop	✓	✓	✓	✓
Bi-Directional (Half Bridge) – Active Hi/Lo	✓	✓	✓	✓

The module can be configured to run in one of four modes:

- **XML Mode** The module executes user-defined logic, allowing for highly customizable functionality.
- **CAN I/O Mode** The 505004 module is commanded by another Marlin controller via CAN communication.
- **Hardware I/O Mode** The module functions like a relay module, in which input signals drive corresponding output signals.
- **Keypad Mode** The module's outputs are driven by button presses from a Marlin keypad.

## 2. Software Installation

In order to program the 505004 module, the Marlin Programming Tool application, as well as the driver for the chosen USB-CAN dongle must be installed on a PC. Run the setup.exe file included with the Programming Tool installation files and follow the directions on the screen to complete installation. When installation is complete, open the Programming Tool's user guide located at

C:\Program Files (x86)\Marlin Technologies\MarlinProgTool\UserGuide.pdf

Note: The exact file path and file name may vary, depending on the version of the Programming Tool and where it was installed. Follow the directions in the Programming Tool user guide to install the driver for the appropriate USB-CAN dongle.

## 3. Configuring and Programming the Module

The 505004 module runs on two pieces of embedded software, both of which can be loaded onto the module using the Programming Tool:

- Base Software.** This software serves as an interface between the user program and the module's hardware. It carries out the user's instructions, handling the more technical details of low-level software and hardware interaction. Base software comes pre-loaded on the 505004 module. Updates, if available, are provided by Marlin Technologies in the form of a .S19 file.
- User Program.** This software is what gives the 505004 module its customized functionality. The user program is created within the Configurator window of the Programming Tool application. This is where I/O is configured, and functional logic is written. The Configurator saves the user program to an .XML file. Data from the .XML file is then loaded to the module via the EEPROM Read/Write window of the Programming tool.

To program the module with either type of software, the setup shown in Figure 2 is required. Connect the USB-CAN dongle to the USB port of the PC. Then connect the CAN lines between the dongle and the 505004 module. See the pinout in Figure 1 for the CAN HI and CAN LO pins on the 505004 module. The configuration and programming process using the Programming Tool is summarized in Figure 3. Details of the process are covered in the sections that follow.

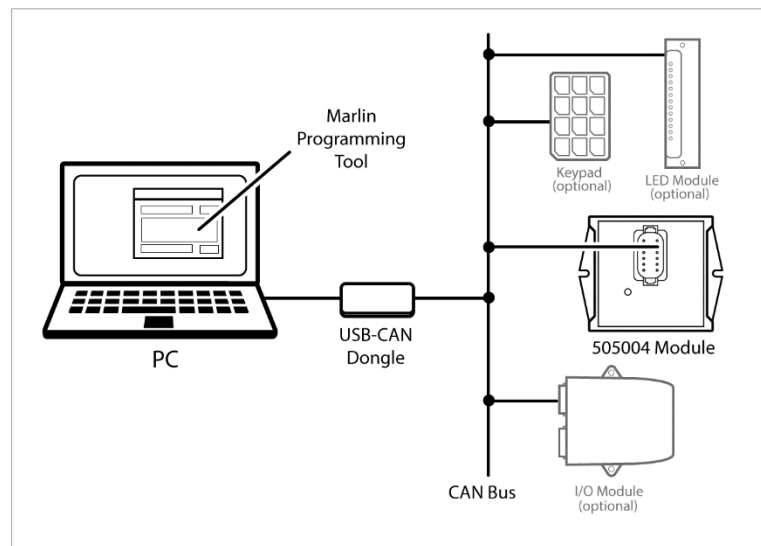


Figure 2

## Programming Tool

Click "Browse" to open S19 file. Then click "Program" update base software.

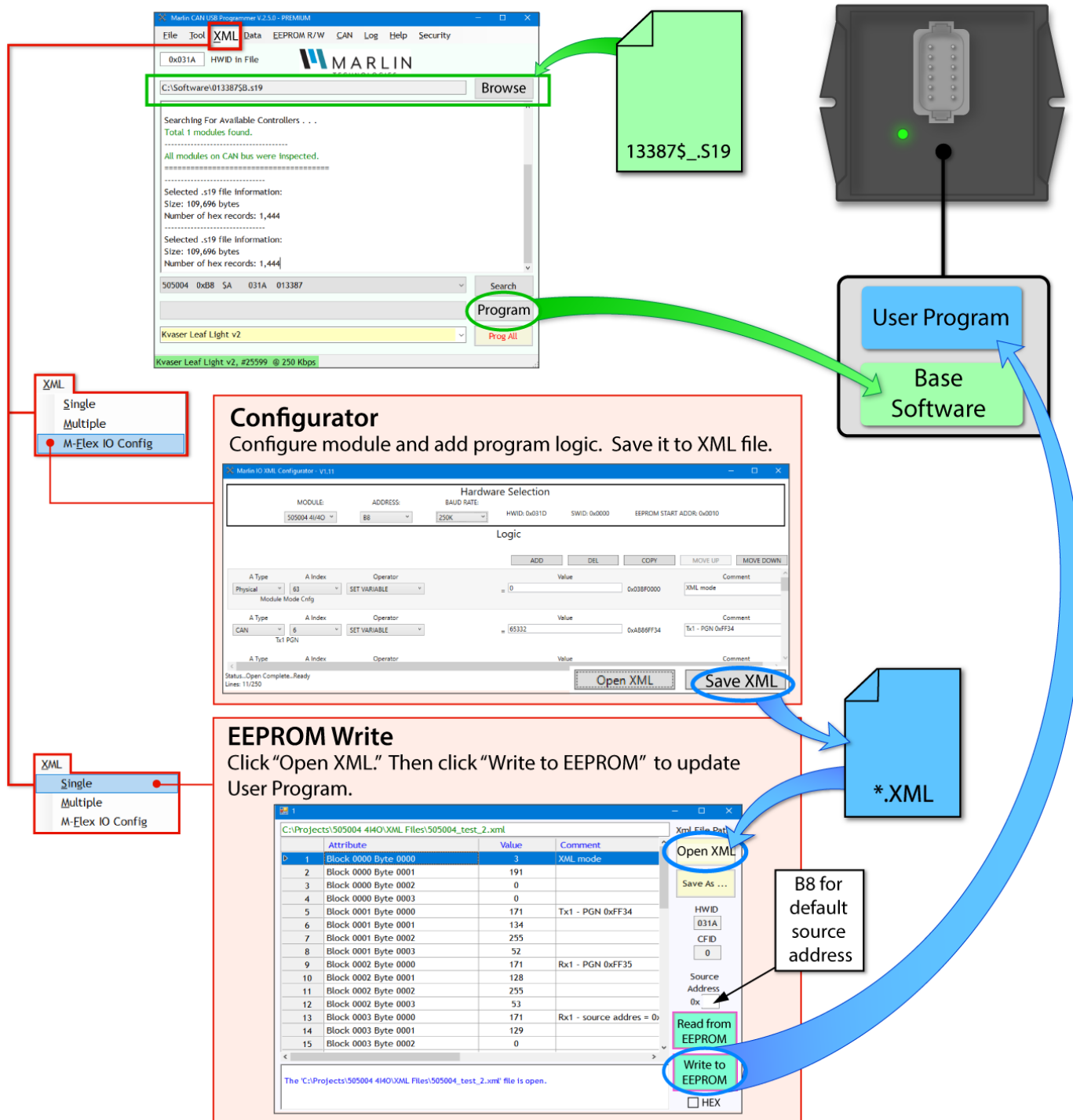


Figure 3

---

### **3.1 Connecting to the Module with the Programming Tool**

After installing the Programming Tool, there should be an icon for it on the PC desktop. Double-click the icon to launch the Programming Tool app. If all devices are set up properly, as shown in Figure 2, information regarding the USB-CAN dongle and the 505004 module should automatically appear near the bottom of the Programming Tool window (see the circled text in Figure 4). If the module information does not appear, click on the “Search” button to prompt the Programming Tool to search again for the module. If the module or dongle information fail to appear, or if there are any other problems encountered with the Programming Tool, see the Programming Tool user guide mentioned in section

## 2. Software Installation.

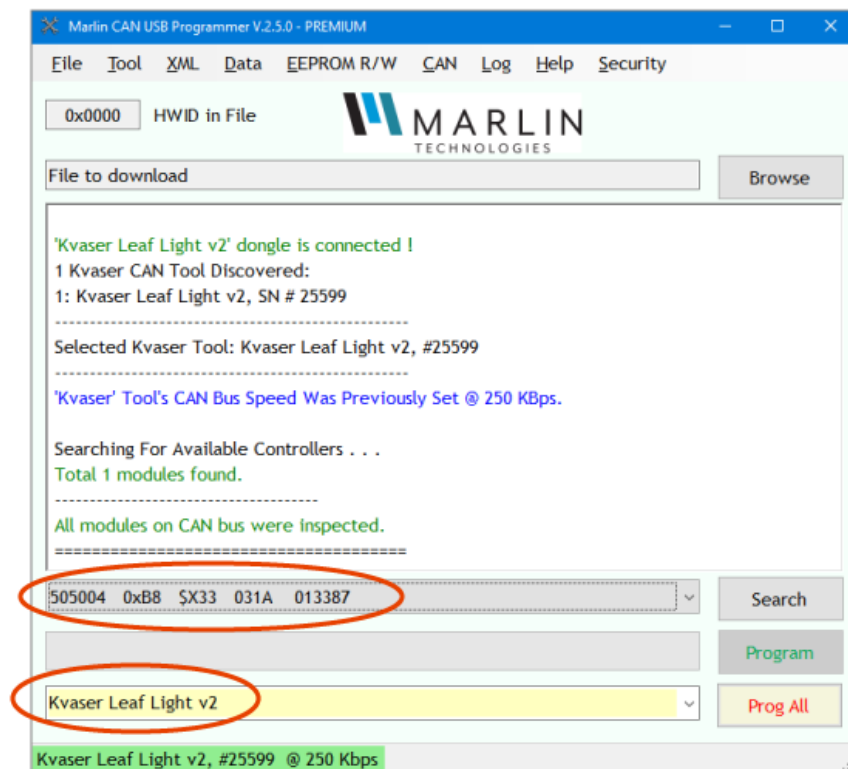


Figure 4

### 3.2 Updating the Base Software

Since the 505004 module comes with the base software pre-loaded, it should not be necessary to load the base Software unless a new revision of the software has been released. In that case, an .S19 file containing the updated software can be obtained from Marlin Technologies.

With the Program Tool running and connected to the module (see section 3.1 *Connecting to the Module with the Programming Tool* for details), click on the “Browse” button, as seen circled in Figure 5.



Figure 5

In the file explorer that pops up, browse for the appropriate .S19 file and select it. Then press the “Program” button, shown circled in Figure 6.

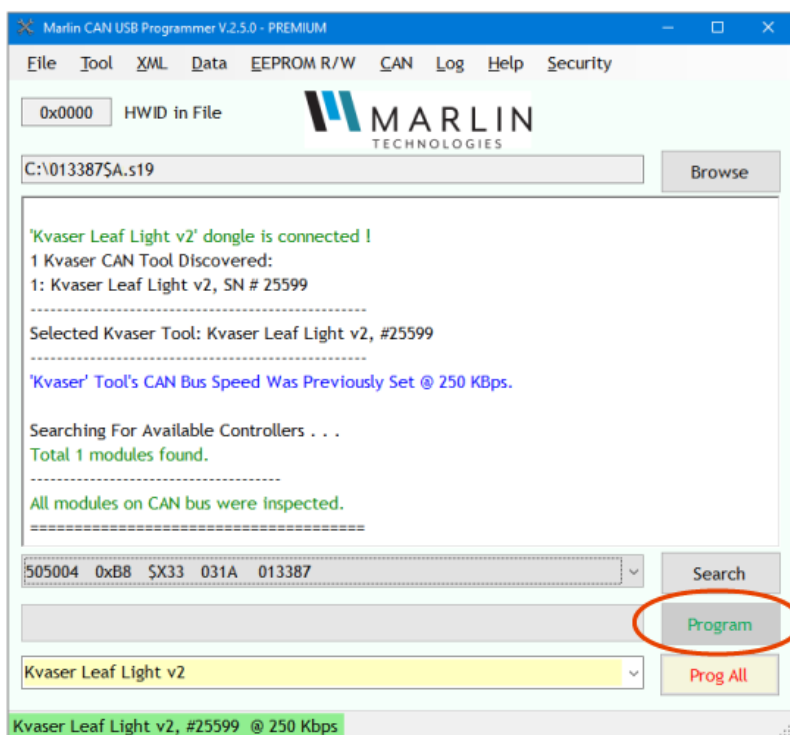


Figure 6

While programming, a green progress bar will appear to the left of the “Program” button. Programming is complete once the progress bar reaches 100% and a message indicating “successful programming” is displayed. If programming is not successful, see the Programming Tool user guide mentioned in section



## 2. Software Installation to troubleshoot.

### 3.3 Configuring the Module

The Configurator window of the Programming Tool allows the user to configure the 505004 module's mode, its I/O, and (if applicable) create its functional logic. Configuration instructions created in the Configurator can be saved to an .XML file, which can be later opened by the Programming Tool's EEPROM-writing utility and used to program the module (see Figure 3 for an overview of this process).

To open a Configurator window, launch the Programming Tool app, and then select the “XMLM-Flex IO Config” menu option, shown circled in Figure 7.

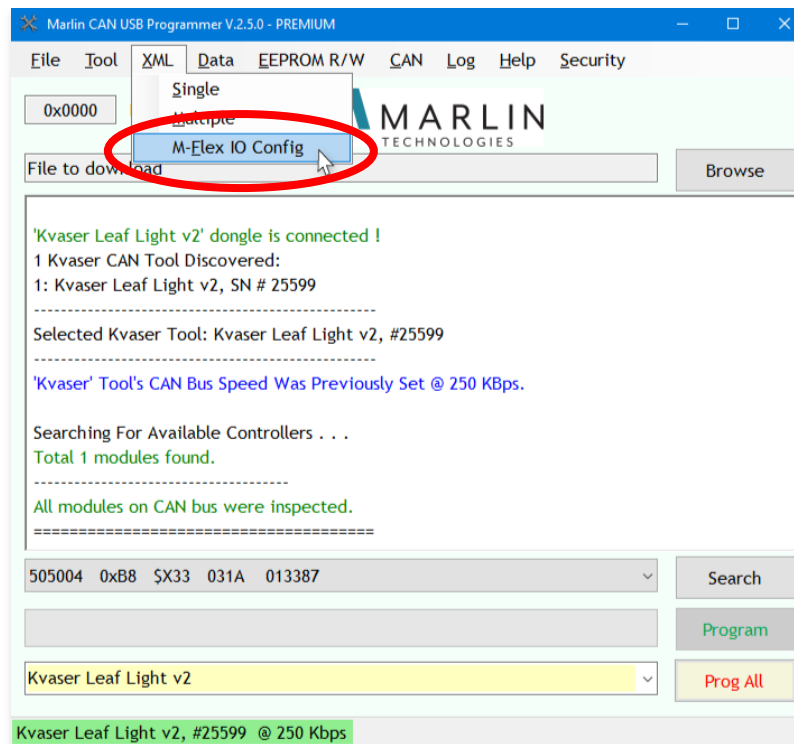


Figure 7

This will open a Configurator window, separate from the existing main Programming Tool window. In the Configurator window, using the drop-down menus circled in Figure 8, select “505004 4I/4O” for the module type, the desired source address for CAN messages, and the desired baud rate for CAN messages.

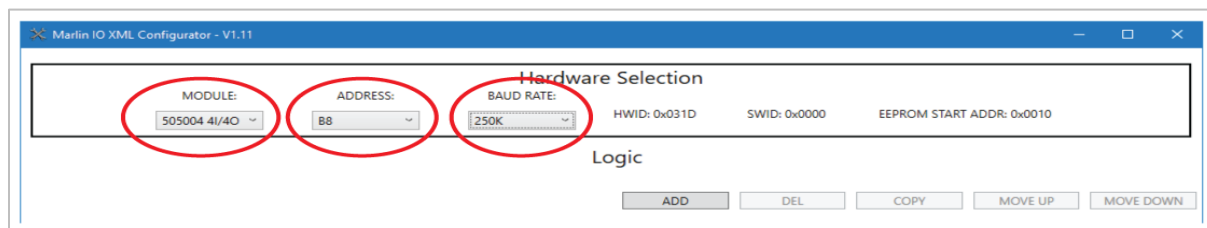


Figure 8

Now instructions for configuring the module can be added. Click the “ADD” button to add an instruction. The default instruction that appears is shown Figure 9. The type of operation that instruction performs is indicated by the “Operator” field. For configuration, only *SET VARIABLE* instructions need be used.

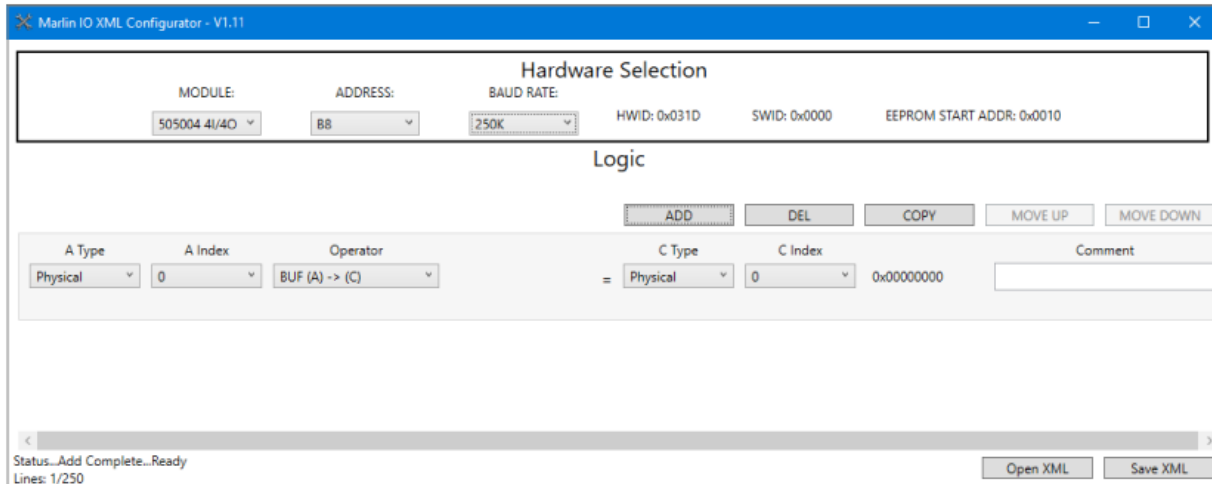


Figure 9

So, to change the recently added instruction to a *SET VARIABLE* instruction, select “SET VARIABLE” from the drop-down menu of the Operator field, as shown in Figure 10. The instructions available in the Configurator consist of an operator that takes inputs, performs

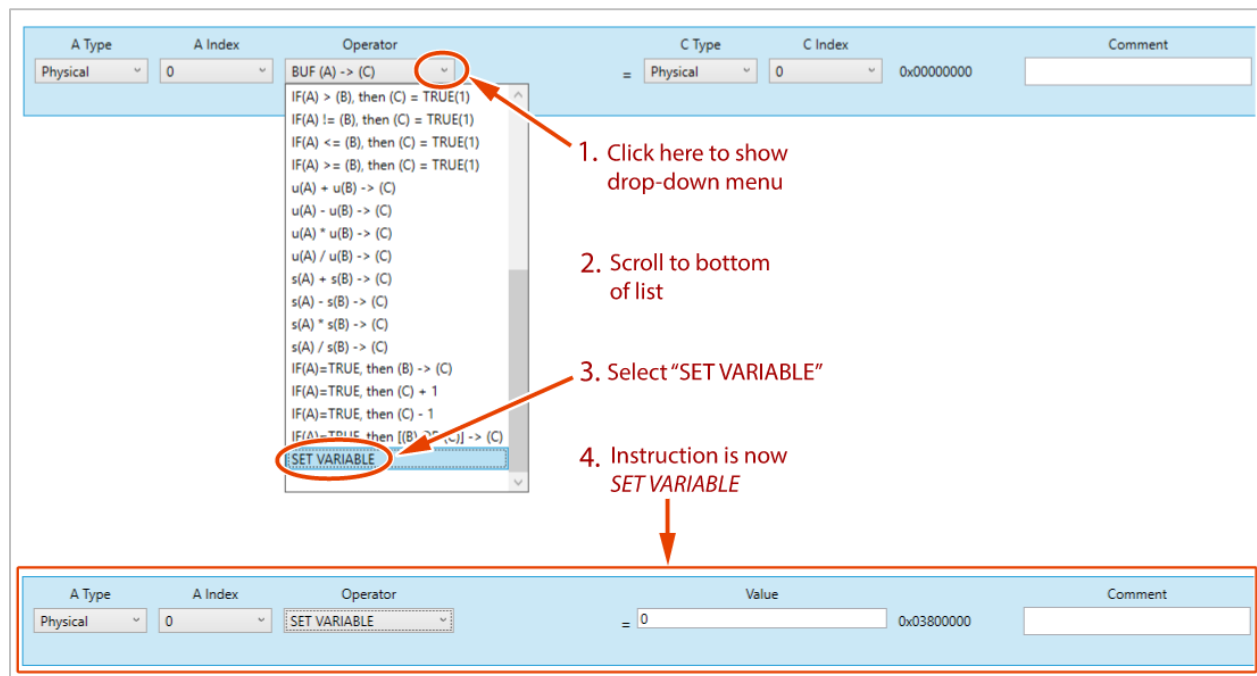


Figure 10

an operation, and puts the result in an output variable. Inputs may be a fixed value or a variable. The variables used are A, B, and C. *SET VARIABLE* is a simple instruction that takes a fixed value

and assigns it to variable A, hence the two fields: “A Type” and “A Index.” In this case, variable A represents the module parameter that the instruction will configure.

Since the module mode dictates the other parameters that need to be configured, setting the module mode is a good choice for the first instruction. To select the module mode as the parameter to be configured, use the drop-down menus to set the Type to “Physical” and Index to “63”:

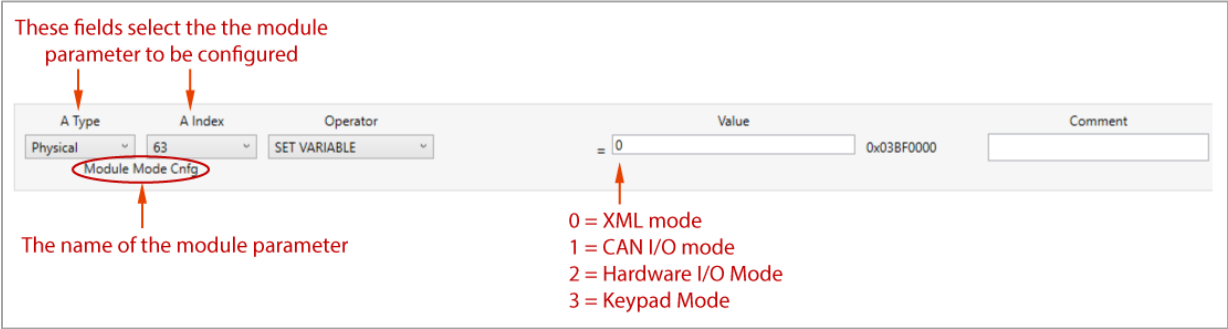


Figure 11

Notice that when A Type and A Index is changed, the name of the module parameter is updated automatically.

Now additional instructions can be added pertaining module mode that has been selected:

- For **XML Mode**, see section [3.3.1 XML Mode](#). Use XML Mode for highly customized functionality. In XML Mode, the module executes user-defined logic.
- For **CAN I/O Mode**, see section

Parameters Variables Type (11b) – Index List of Values		
0	<Blank>	Variables are open and available for the User
1	<Blank>	
2	<Blank>	
3	<Blank>	
4	<Blank>	
5	<Blank>	
126	<Blank>	
127	<Blank>	

## Example XML Configuration

Operator	Variable	Value	
Set_Var	Phy 63	0	Mode = XML
Set_Var	Phy 54	1	Output1 = HS_Dig
Set_Var	Phy 55	2	Output2 = LS_Dig
Set_Var	Phy 56	4	Output3 = HS_DC
Set_Var	Phy 57	6	Output4 = LS_DC
Set_Var	Phy 53	100	PWM Freq = 100Hz

XML Logic

## Example XML for Custom Keypad Behavior

Operator	Variable	Value	
Set_Var	Phy 63	0	Mode = XML
Set_Var	Phy 62	192	Keypad SA = 0xC0
Set_Var	Phy 58	2	Output 1 controlled by Button 2; Type = Momentary
Set_Var	Phy 59	521	Output 2 controlled by Button 9; Type = Tri-State
Set_Var	Phy 54	1	Output1 = Dig Active Hi
Set_Var	Phy 55	1	Output2 = Dig Active Hi
Set_Var	Tmp 3	68	Constant for LED command (Reference Pg. 31)
Set_Var	Tmp 4	2	Constant for Button State that turns on output (Reference Pg. 13 & 16)

### XML Logic

Operator	A Variable	B Variable	C Variable	
Buf	Can 63		Phy 24	Output 1 state = Button 2 state
Buf	Tmp 0		Tmp 5	Clear Flag before use
IF(A = B)-> C=1	Can 64	Tmp 4	Tmp 5	If button 9 state is 2, then set flag
IF(A = 1) B->C	Tmp 5	Tmp 1	Phy 25	If flag is set, then turn ON output 2
Buf	Tmp 0		Can 67	Clear Button 1 LED command
If A=true; B->C	Can 63	Tmp 3	Can 67	If Button 1 is pressed, Send custom LED command

\*Button 9 will use standard LED command, not a custom one, so it doesn't need to be set.

- 
- [3.3.2 CAN I/O Mode](#). Use CAN I/O mode if the 505004 is to be commanded by another controller via CAN communication.
  - For **Hardware I/O Mode**, see section [3.3.3 Hardware I/O Mode](#). Use Hardware I/O Mode if the module is to function as a relay module, in which input signals drive corresponding output signals.
  - For **Keypad Mode**, see section [3.3.4 Keypad Mode](#). Use Keypad Mode if the module's outputs are to be driven by button presses on a keypad.

### 3.3.1 XML Mode

In XML Mode, the 505004 module executes user-defined logic that allows for highly customizable functionality. With the Programming Tool's Configurator utility, the user creates a program that consists of two main groups of instructions: the initialization block and the logic block. When the module powers up, the initialization block is executed once, then the logic block is executed, repeatedly, in an infinite loop.

Instructions are executed in the order they appear in the Configurator, from top to bottom. Initialization instructions utilize only the *SET VARIABLE* operator, and must be grouped together in the top rows of the program. The first instruction that does *not* use the *SET VARIABLE* operator represents the beginning of the logic block. It is this instruction that program execution loops back to after then the last logic instruction is executed. The path of program execution is visualized in Figure 12.

The initialization block consists of instructions that configure module parameters such as

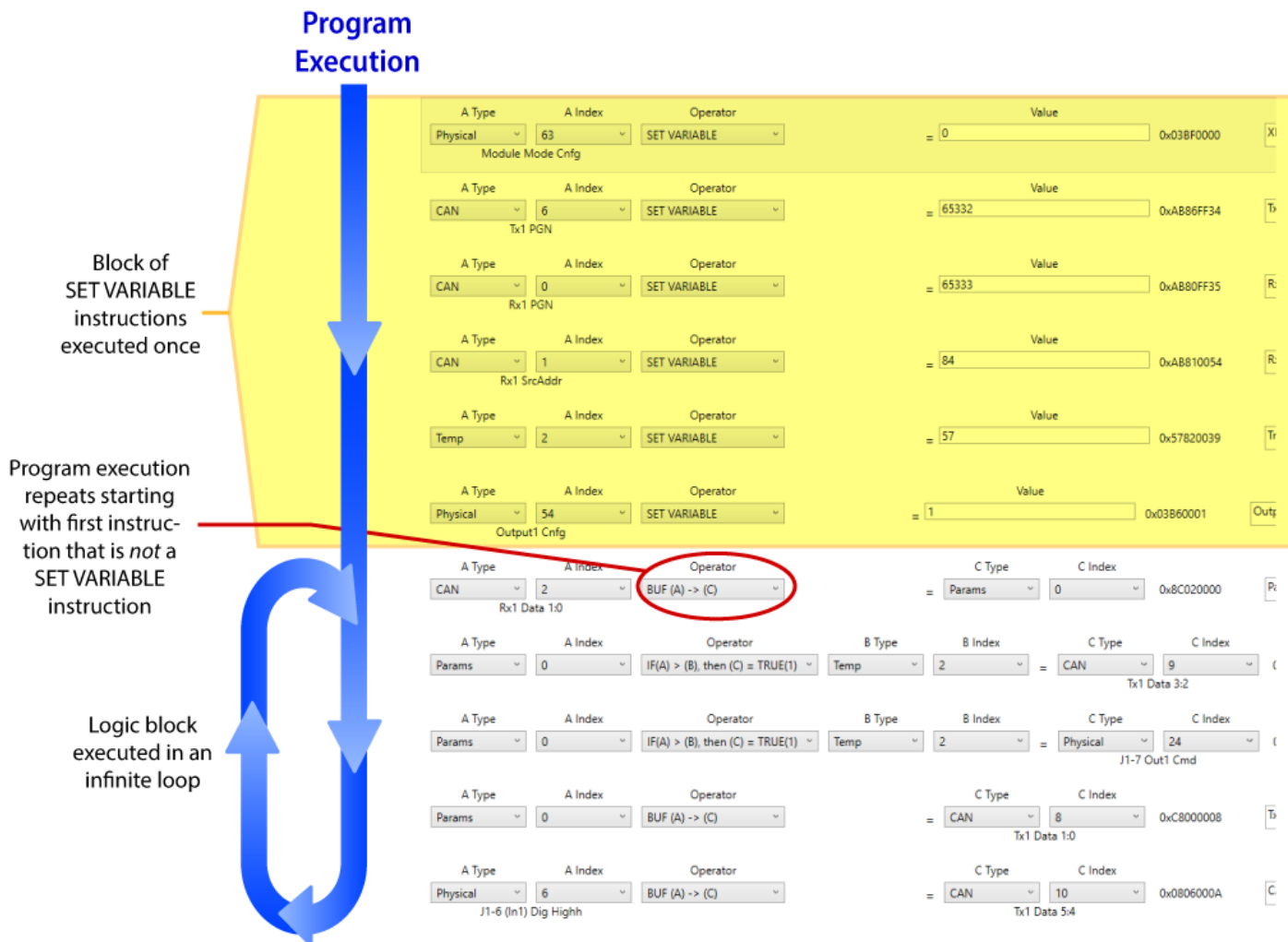


Figure 12

output/input types, current limits, setpoints, constants, initialized variables, CAN PGN definitions, and CAN transmit intervals to name a few.

### 3.3.1.1 Configuration

Use the ADD button to add *SET VARIABLE* instructions for each parameter listed in the *List of Configuration Parameters Used in XML Mode* below. Refer to Figure 13 as a guide for filling out the *SET VARIABLE* instructions. If necessary, see section 3.3 *Configuring the Module* for more details about adding *SET VARIABLE* instructions.

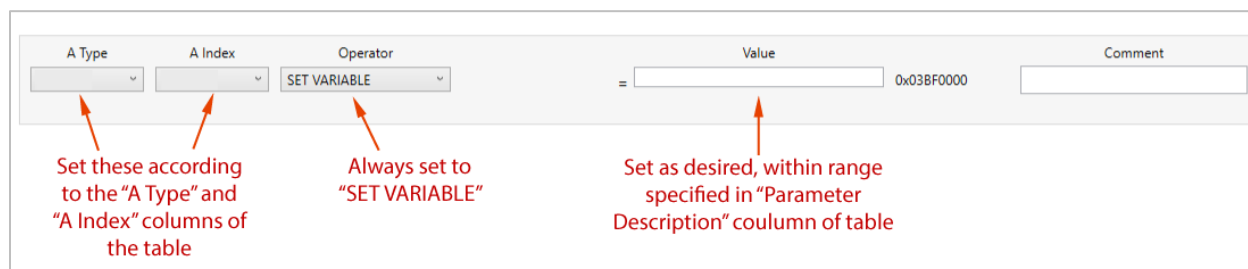


Figure 13

There are few restrictions on which configuration variables are used/useful in XML mode. Availability of configuration values is based purely on the desired application. The only configuration values that are not usable in XML mode are the output ON and OFF delays. To achieve an ON/OFF delay, the use of a counter in the user defined XML logic block is required.

### 3.3.1.2 Logic

Logic instructions are to be added *below* the block of initialization instructions that use the *SET VARIABLE* instruction. Unlike initialization instructions, logic instructions can utilize a wide variety of operators. The various operators are discussed in section

3.3.1.2.1 Operators, and the parameters and variables that the operators act upon are discussed in section 3.3.1.2.2 *Module Parameters and Variables*.

#### 3.3.1.2.1 Operators

Each instruction in the Configurator performs a single operation. An operation consists of an Operator, which acts upon variables A, B, and C. The number of variables involved varies from operator to operator. Variables A and B typically serve as inputs to an operation, while variable C is mainly used to store the operation's output. (The only exception is the *SET VARIABLE* operator, which uses variable A, instead of C, for its output.) The operators below are listed in the order they appear in the drop-down menu of the Operator field. The one exception is that the *SET VARIABLE* appears first in the list below, but appears last in the drop down menu (see Figure 14). This is for the sake of organization.

A Type	A Index	Operator	C Type	C Index	Comment
Physical	0	BUF (A) -> (C)	= Physical	0	0x00000000

IF(A) > (B), then (C) = TRUE(1)  
 IF(A) != (B), then (C) = TRUE(1)  
 IF(A) <= (B), then (C) = TRUE(1)  
 IF(A) >= (B), then (C) = TRUE(1)  
 u(A) + u(B) -> (C)  
 u(A) - u(B) -> (C)  
 u(A) \* u(B) -> (C)  
 u(A) / u(B) -> (C)  
 s(A) + s(B) -> (C)  
 s(A) - s(B) -> (C)  
 s(A) \* s(B) -> (C)  
 s(A) / s(B) -> (C)  
 IF(A)=TRUE, then (B) -> (C)  
 IF(A)=TRUE, then (C) + 1  
 IF(A)=TRUE, then (C) - 1  
 IF(A)=TRUE, then [(B) OR (C)] -> (C)  
**SET VARIABLE**

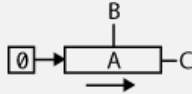
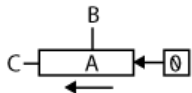
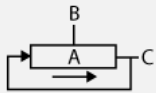
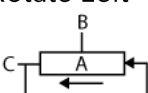
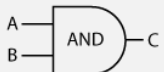
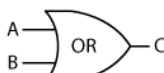
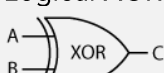
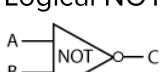

Figure 14

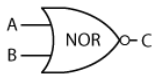



### ASSIGNMENT OPERATORS

Name	Menu Item	Comment
Set Variable	SET VARIABLE	Sets variable A to a given fixed value.
Copy	BUF (A) -> (C)	Copies variable A's value to variable C.

### LOGICAL OPERATORS

Name	Menu Item	Comment
Shift Right 	(A) >> (B) -> (C)	The value in A is shifted bitwise to the right B times, with zero shifted into the most significant bit. The result is placed in output C.
Shift Left 	(A) << (B) -> (C)	The value in A is shifted bitwise to the left B times, with zero shifted into the least significant bit. The result is placed in output C.
Rotate Right 	ROT RGHT (A) BY (B)-> (C)	The value in A is rotated bitwise to the right B times, with the least significant bit rotated into the most significant bit. The result is placed in output C.
Rotate Left 	ROT LEFT (A) BY (B)-> (C)	The value in A is rotated bitwise to the right B times, with the most significant bit rotated into the least significant bit. The result is placed in output C.
Logical AND 	(A) AND (B) -> (C)	The value in A is AND'd with the value in B and placed in the output C. Example: 1100 AND 1010 = 1000
Logical OR 	(A) OR (B) -> (C)	The value in A is OR'd with the value in B and placed in the output C. Example: 1100 OR 1010 = 1110
Logical XOR 	(A) XOR (B) -> (C)	The value in A is XOR'd with the value in B and placed in the output C. Example: 1100 XOR 1010 = 0110
Logical NOT 	NOT (A) -> (C)	The value in A is inverted and placed in the output C. Example: NOT 1010 = 0101
Logical NAND 	(A) NAND (B) -> (C)	The value in A is AND'd with the value in B, then inverted and placed in the output C. Example: 1100 NAND 1010 = 0111

Name	Menu Item	Comment
Logical NOR 	(A) NOR (B) -> (C)	The value in A is OR'd with the value in B, then inverted and placed in the output C. Example: 1100 NOR 1010 = 0001
Logical NXOR 	(A) XNOR (B) -> (C)	The value in A is XOR'd with the value in B, then inverted and placed in the output C. Example: 1100 XNOR 1010 = 1001

### COMPARATIVE OPERATORS

Name	Menu Item	Comment
If Equal	IF(A) = (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.
If Less Than	IF(A) < (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.
If Greater Than	IF(A) > (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.
If Not Equal	IF(A) != (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.
If Less Than or Equal To	IF(A) <= (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.
If Greater Than or Equal To	IF(A) >= (B) -> (C) = TRUE(1)	Otherwise C is left unchanged.

### MATHEMATICAL OPERATORS (Unsigned Values)

Name	Menu Item	Comment
Add	u(A) + u(B) -> u(C)	Value A, B, and C must fit within a range of 0 to 65535.
Subtract	u(A) - u(B) -> u(C)	Value A, B, and C must fit within a range of 0 to 65535.
Multiply	u(A) * u(B) -> u(C)	Value A, B, and C must fit within a range of 0 to 65535.
Divide	u(A) / u(B) -> u(C)	Value A, B, and C must fit within a range of 0 to 65535.

### MATHEMATICAL OPERATORS (Signed Values)

Name	Menu Item	Comment
Add	s(A) + s(B) -> s(C)	Value A, B, and C must fit within a range of -32768 to 32767.
Subtract	s(A) - s(B) -> s(C)	Value A, B, and C must fit within a range of -32768 to 32767.
Multiply	s(A) * s(B) -> s(C)	Value A, B, and C must fit within a range of -32768 to 32767.

Name	Menu Item	Comment
Divide	$s(A) / s(B) \rightarrow s(C)$	Value A, B, and C must fit within a range of -32768 to 32767.

### CONDITIONAL OPERATORS

Name	Menu Item	Comment
Conditional Copy	IF(A)=TRUE, then (B) $\rightarrow$ (C)	Otherwise C is left unchanged.
Conditional INCREMENT	IF(A)=TRUE, the (C) + 1	Otherwise C is left unchanged.
Conditional DECREMENT	IF(A)=TRUE, the (C) - 1	Otherwise C is left unchanged.
Conditional OR	IF(A)=TRUE, the [(B) OR (C)] $\rightarrow$ C	Otherwise C is left unchanged.

#### 3.3.1.2.2 Module Parameters and Variables

In a Configurator instruction, the module parameter that Variable A, B, or C represents is specified by a Type field and an Index Number field. Thus, each variable appearing in an operation has two drop-down menus associated with it. For example, in Figure 15, the user has selected *IF (A) > (B), then (C) = TRUE(1)* as the Operator. As a result, the row was populated with drop down menus for variables A (highlighted in red), B (highlighted in blue), and C (highlighted in green).

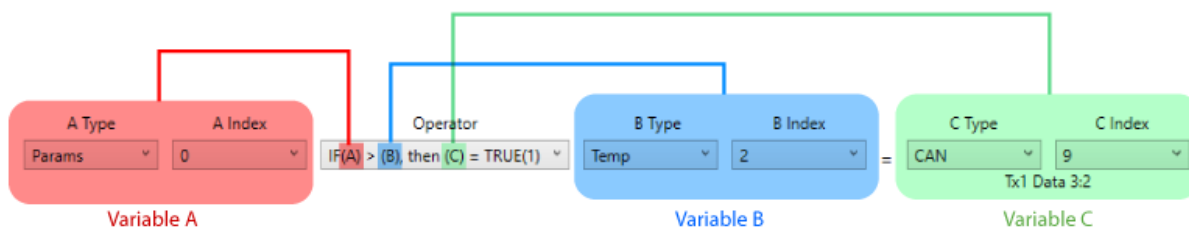


Figure 15

The options available in the Type menus are:

- **Physical** – These parameters often pertain to the module’s I/O.
- **Temp** – These are variables intended for use as temporary or constant values. For example, they might be used to store gains, timer reload values, CAN masks, or values to compare to. All Temp variables can be set to values of the user’s choosing except for TRUE and FALSE, which are to remain at fixed values of 1 and 0, respectively.
- **CAN** – These parameters pertain to the settings and data of received and transmitted CAN messages
- **Params** – These are variables that are meant to store non-volatile values. For example, counts and status flags that are meant to be remembered after a power cycle. **Non-Volatile behavior is not implemented at present time.**

The module parameters available through the Type and Index drop-down menus in Figure 15 are presented on the pages below. Note that each parameter value is 16 bits.

When finished adding instructions in the Configurator, click “Save XML” in the bottom-right corner of the window (as seen in Figure 10) to save the data to an .XML file. To load this data on to the 505004 module, follow the instructions in [section 3.4 Loading the User Program](#).

Physical Variables Type (00b) – Index List of Values				
0	NULL	DO NOT USE!		
1	J1-12	Input	Analog	Battery Voltage [0-43,554 mV, 1mV/bit]
2	J1-6	Input	Analog	Input 1 Voltage [0-43,554 mV, 1mV/bit] / Count [1cnt/bit, Unsigned]
3	J1-5	Input	Analog	Input 2 Voltage [0-43,554 mV, 1mV/bit]
4	J1-9	Input	Analog	Input 3 Voltage [0-43,554 mV, 1mV/bit]
5	J1-10	Input	Analog	Input 4 Voltage [0-43,554 mV, 1mV/bit]
6	J1-6	Input	Digital	Input 1 Digital High [1=Active High, 0=otherwise]
7	J1-6	Input	Digital	Input 1 Digital Low [1=Active Low, 0=otherwise]
8	J1-5	Input	Digital	Input 2 Digital High [1=Active High, 0=otherwise]
9	J1-5	Input	Digital	Input 2 Digital Low [1=Active Low, 0=otherwise]
10	J1-9	Input	Digital	Input 3 Digital High [1=Active High, 0=otherwise]
11	J1-9	Input	Digital	Input 3 Digital Low [1=Active Low, 0=otherwise]
12	J1-10	Input	Digital	Input 4 Digital High [1=Active High, 0=otherwise]
13	J1-10	Input	Digital	Input 4 Digital Low [1=Active Low, 0=otherwise]
14	J1-7	Input	Analog	Output 1 Current [0-4200mA, 1mA/bit]
15	J1-8	Input	Analog	Output 2 Current [0-4200mA, 1mA/bit]
16	J1-4	Input	Analog	Output 3 Current [0-4200mA, 1mA/bit]
17	J1-3	Input	Analog	Output 4 Current [0-4200mA, 1mA/bit]
18	J1-6	Input	D.C.	Digital Input 1 Duty Cycle [0-100%, 0.1%/bit]
19	J1-6	Input	Period	Digital Input 1 Period [0-65535uS, 1 uS/bit]
20	J1-6	Input	Freq	Digital Input 1 Frequency [32-10,000Hz, 0.1 Hz/bit]
21	J1-5	Input	D.C.	Digital Input 2 Duty Cycle [0-100%, 0.1%/bit]
22	J1-5	Input	Period	Digital Input 2 Period [0-65535uS, 1 uS/bit]
23	J1-5	Input	Period	Digital Input 2 Frequency [32-10,000Hz, 0.1 Hz/bit]
24	J1-7	Output	Digital	Output 1 Digital Cmd [1=On, 0=Off]
25	J1-8	Output	Digital	Output 2 Digital Cmd [1=On, 0=Off]
26	J1-4	Output	Digital	Output 3 Digital Cmd [1=On, 0=Off]
27	J1-3	Output	Digital	Output 4 Digital Cmd [1=On, 0=Off]
28	J1-7	Output	DC/mA	Output 1 DutyCycle/Current Cmds (Mode based) [1 mA or 0.1% /bit]
29	J1-8	Output	DC/mA	Output 2 DutyCycle/Current Cmds (Mode based) [1 mA or 0.1% /bit]
30	J1-4	Output	DC/mA	Output 3 DutyCycle/Current Cmds (Mode based) [1 mA or 0.1% /bit]
31	J1-3	Output	DC/mA	Output 4 DutyCycle/Current Cmds (Mode based) [1 mA or 0.1% /bit]
32	Cnfg	Load	mΩ	Output 1 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
33	Cnfg	Load	mΩ	Output 2 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
34	Cnfg	Load	mΩ	Output 3 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
35	Cnfg	Load	mΩ	Output 4 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
36	Cnfg	Limit	mA	Output 1 Current Limit [1 mA /bit, 0-3000mA]
37	Cnfg	Limit	mA	Output 2 Current Limit [1 mA /bit, 0-3000mA]
38	Cnfg	Limit	mA	Output 3 Current Limit [1 mA /bit, 0-3000mA]
39	Cnfg	Limit	mA	Output 4 Current Limit [1 mA /bit, 0-3000mA]
40	Cnfg	Thres	factor	Threshold Voltage for Digital inputs to be set to Active [65,535mV, 1mV/bit]
41	Cnfg	Hyst	factor	Hysteresis Voltage for Digital inputs to return to Inactive [65,535mV, 1mV/bit]
42	Cnfg	P-term	factor	P-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
43	Cnfg	I-term	factor	I-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
44	Cnfg	D-term	factor	D-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]

45	Cnfg	OnDly1	Time	Output 1 HW_InOut Mode On Delay [0-65,535mS, 1mS/bit]
46	Cnfg	OnDly2	Time	Output 2 HW_InOut Mode On Delay [0-65,535mS, 1mS/bit]
47	Cnfg	OnDly3	Time	Output 3 HW_InOut Mode On Delay [0-65,535mS, 1mS/bit]
48	Cnfg	OnDly4	Time	Output 4 HW_InOut Mode On Delay [0-65,535mS, 1mS/bit]
49	Cnfg	OffDly1	Time	Output 1 HW_InOut Mode Off Delay [0-65,535mS, 1mS/bit]
50	Cnfg	OffDly2	Time	Output 2 HW_InOut Mode Off Delay [0-65,535mS, 1mS/bit]
51	Cnfg	OffDly3	Time	Output 3 HW_InOut Mode Off Delay [0-65,535mS, 1mS/bit]
52	Cnfg	OffDly4	Time	Output 4 HW_InOut Mode Off Delay [0-65,535mS, 1mS/bit]
53	Cnfg	PWMs	Freq	Set Base Frequency of all PWM Outputs [30-1000Hz, 1Hz/bit]
54	Cnfg	Out1	Type	Output 1 [0=Off, 1=HS, 2=LS, 3=HS_mA, 4=HS_DC, 5=LS_mA, 6=LS_DC, 7=BiDir]
55	Cnfg	Out2	Type	Output 2 [0=Off, 1=HS, 2=LS, 3=HS_mA, 4=HS_DC, 5=LS_mA, 6=LS_DC, 7=BiDir]
56	Cnfg	Out3	Type	Output 3 [0=Off, 1=HS, 2=LS, 3=HS_mA, 4=HS_DC, 5=LS_mA, 6=LS_DC, 7=BiDir]
57	Cnfg	Out4	Type	Output 4 [0=Off, 1=HS, 2=LS, 3=HS_mA, 4=HS_DC, 5=LS_mA, 6=LS_DC, 7=BiDir]
58	Cnfg	Kypd	B1	Output 1 – Button Index# [Refer to Spec 11713S__]
59	Cnfg	Kypd	B2	Output 2 – Button Index# [Refer to Spec 11713S__]
60	Cnfg	Kypd	B3	Output 3 – Button Index# [Refer to Spec 11713S__]
61	Cnfg	Kypd	B4	Output 4 – Button Index# [Refer to Spec 11713S__]
62	Cnfg	Kypd	SA	Keypad Source Address
63	Cnfg	Module	Mode	Config Module [ 0=XML, 1=CAN_I/O, 2=HW_I/O, 3=HW_Keypad ]
64	Cnfg	In1	Type	Input 1 [0=Analog/Digital/Freq, 1=Current, 2=Resistance, 3=QuadX1, 4=QuadX2, 5=QuadX4]
65	Undefined at the current time of publication Reserved for future Phy Configurations			
66				
67				
68	Cnfg	Sleep	See page 19	

CAN Variables Type (10b) – Index List of Values			
0	Rx1 PGN	Rx CAN Msg #1	PGN of CAN Message to be received (*)
1	Rx1 SrcAddr		Source Address of CAN Message to be received (**)
2	Rx1 Data 1:0		Data Bytes 1[MSB] & 0[LSB] of message Received
3	Rx1 Data 3:2		Data Bytes 3[MSB] & 2[LSB] of message Received
4	Rx1 Data 5:4		Data Bytes 5[MSB] & 4[LSB] of message Received
5	Rx1 Data 7:6		Data Bytes 7[MSB] & 6[LSB] of message Received
6	Tx1 PGN	Tx CAN Msg #1	PGN of CAN Message to be Transmitted (*)
7	Tx1 MsgRate		Std/Ext ID Flag : Message Transmit Rate [10mS/bit, 0-32,000mS] (***)
8	Tx1 Data 1:0		Data Bytes 1[MSB] & 0[LSB] to send
9	Tx1 Data 3:2		Data Bytes 3[MSB] & 2[LSB] to send
10	Tx1 Data 5:4		Data Bytes 5[MSB] & 4[LSB] to send
11	Tx1 Data 7:6		Data Bytes 7[MSB] & 6[LSB] to send
12	Rx2 PGN	Rx CAN Msg #2	PGN of CAN Message to be received (*)
13	Rx2 SrcAddr		Source Address of CAN Message to be received (**)
14	Rx2 Data 1:0		Data Bytes 1[MSB] & 0[LSB] of message Received
15	Rx2 Data 3:2		Data Bytes 3[MSB] & 2[LSB] of message Received
16	Rx2 Data 5:4		Data Bytes 5[MSB] & 4[LSB] of message Received
17	Rx2 Data 7:6		Data Bytes 7[MSB] & 6[LSB] of message Received
18	Tx2 PGN	Tx CAN Msg #2	PGN of CAN Message to be Transmitted (*)
19	Tx2 MsgRate		Std/Ext ID Flag : Message Transmit Rate [10mS/bit, 0-32,000mS] (***)
20	Tx2 Data 1:0		Data Bytes 1[MSB] & 0[LSB] to send
21	Tx2 Data 3:2		Data Bytes 3[MSB] & 2[LSB] to send
22	Tx2 Data 5:4		Data Bytes 5[MSB] & 4[LSB] to send
23	Tx2 Data 7:6		Data Bytes 7[MSB] & 6[LSB] to send
24	Rx3 PGN	Rx CAN Msg #3	PGN of CAN Message to be received (*)
25	Rx3 SrcAdr		Source Address of CAN Message to be received (**)
26	Rx3 Data 1:0		Data Bytes 1[MSB] & 0[LSB] of message Received
27	Rx3 Data 3:2		Data Bytes 3[MSB] & 2[LSB] of message Received
28	Rx3 Data 5:4		Data Bytes 5[MSB] & 4[LSB] of message Received
29	Rx3 Data 7:6		Data Bytes 7[MSB] & 6[LSB] of message Received
30	Tx3 PGN	Tx CAN Msg #3	PGN of CAN Message to be Transmitted (*)
31	Tx3 MsgRate		Std/Ext ID Flag : Message Transmit Rate [10mS/bit, 0-32,000mS] (***)
32	Tx3 Data 1:0		Data Bytes 1[MSB] & 0[LSB] to send
33	Tx3 Data 3:2		Data Bytes 3[MSB] & 2[LSB] to send
34	Tx3 Data 5:4		Data Bytes 5[MSB] & 4[LSB] to send
35	Tx3 Data 7:6		Data Bytes 7[MSB] & 6[LSB] to send
36	Rx4 PGN	Rx CAN Msg #4	PGN of CAN Message to be received (*)
37	Rx4 SrcAddr		Source Address of CAN Message to be received (**)
38	Rx4 Data 1:0		Data Bytes 1[MSB] & 0[LSB] of message Received
39	Rx4 Data 3:2		Data Bytes 3[MSB] & 2[LSB] of message Received
40	Rx4 Data 5:4		Data Bytes 5[MSB] & 4[LSB] of message Received
41	Rx4 Data 7:6		Data Bytes 7[MSB] & 6[LSB] of message Received
42	Tx4 PGN	Tx CAN Msg #4	PGN of CAN Message to be Transmitted (*)
43	Tx4 MsgRate		Std/Ext ID Flag : Message Transmit Rate [10mS/bit, 0-32,000mS] (***)
44	Tx4 Data 1:0		Data Bytes 1[MSB] & 0[LSB] to send
45	Tx4 Data 3:2		Data Bytes 3[MSB] & 2[LSB] to send
46	Tx4 Data 5:4		Data Bytes 5[MSB] & 4[LSB] to send
47	Tx4 Data 7:6		Data Bytes 7[MSB] & 6[LSB] to send

59	Tx1 Priority	Tx CAN Msg Pri	Tx1 Msg Priority bits [0-7]
60	Tx2 Priority		Tx2 Msg Priority bits [0-7]
61	Tx3 Priority		Tx3 Msg Priority bits [0-7]
62	Tx4 Priority		Tx4 Msg Priority bits [0-7]

81	Rx5 PGN	Rx CAN Msg #5	PGN of CAN Message to be received (*)
82	Rx5 SrcAddr		Source Address of CAN Message to be received (**)
83	Rx5 Data 1:0		Data Bytes 1[MSB] & 0[LSB] of message Received
84	Rx5 Data 3:2		Data Bytes 3[MSB] & 2[LSB] of message Received
85	Rx5 Data 5:4		Data Bytes 5[MSB] & 4[LSB] of message Received
86	Rx5 Data 7:6		Data Bytes 7[MSB] & 6[LSB] of message Received
87	Tx5 Priority	Priority	Tx5 Msg Priority bits [0-7]
88	Tx5 PGN	Tx CAN Msg #5	PGN of CAN Message to be Transmitted (*)
89	Tx5 MsgRate		Std/Ext ID Flag : Message Transmit Rate [10mS/bit, 0-32,000mS] (***)
90	Tx5 Data 1:0		Data Bytes 1[MSB] & 0[LSB] to send
91	Tx5 Data 3:2		Data Bytes 3[MSB] & 2[LSB] to send
92	Tx5 Data 5:4		Data Bytes 5[MSB] & 4[LSB] to send
93	Tx5 Data 7:6		Data Bytes 7[MSB] & 6[LSB] to send

48	Cnfg	Tx Rate	Time	CAN Msg FF40 Tx Rate [10mS/bit, 0=Off]
49	Cnfg	Tx Rate	Time	CAN Msg FF41 Tx Rate [10mS/bit, 0=Off]
50	<Blank>			
51	Cnfg	Tx Rate	Time	CAN Msg FF43 Tx Rate [10mS/bit, 0=Off]
52	<Blank>			
53	Cnfg	Tx Rate	Time	CAN Msg FF60 Tx Rate [10mS/bit, 0=Off]
54	Cnfg	Tx Rate	Time	CAN Msg FF61 Tx Rate [10mS/bit, 0=Off]
55	Cnfg	Tx Rate	Time	CAN Msg FF63 Tx Rate [10mS/bit, 0=Off]
56	Cnfg	Tx Rate	Time	CAN Msg FF64 Tx Rate [10mS/bit, 0=Off]
57	Undefined at the current time of publication Reserved for future CAN message information			
58	<Blank>			

63	Kypd	Input	Digital	Keypad Input 1 [0=Up,1=Down,2=Err,3=No Key]
64	Kypd	Input	Digital	Keypad Input 2 [0=Up,1=Down,2=Err,3=No Key]
65	Kypd	Input	Digital	Keypad Input 3 [0=Up,1=Down,2=Err,3=No Key]
66	Kypd	Input	Digital	Keypad Input 4 [0=Up,1=Down,2=Err,3=No Key]
71	Kypd	Output	6-bit	Keypad Out1 LEDs [0=Off,1=On,2=Blink,3=NoChg]
72	Kypd	Output	6-bit	Keypad Out2 LEDs [0=Off,1=On,2=Blink,3=NoChg]
73	Kypd	Output	6-bit	Keypad Out3 LEDs [0=Off,1=On,2=Blink,3=NoChg]
74	Kypd	Output	6-bit	Keypad Out4 LEDs [0=Off,1=On,2=Blink,3=NoChg]



79	Cnfg	TimeOut	Time	Cmd Message Timeouts [10mS/bit, 0=NoTimeout]
80	Cnfg	Tx Rate	Time	LED status Message Transmit Rate [10mS/bit, 0=No Transmit]

93	<Blank>	J1939 CAN Or Pre- Define Rx Values	Undefined at the current time of publication Reserved for future CAN message information
94	<Blank>		
...	...		
127	<Blank>		
128	<Blank>		

(\*) NOTE: 11-bit Std\_ID: PGN is Message ID (Full Std ID range 0-2047 [0x000-0x7FF])  
29-bit Ext\_ID: PGN is PDU:Format and PDU:Specific of Message ID  
Priority is ignored on Received messages, and default of 6 for Transmitted messages.

(\*\*) NOTE: 11-bit Std\_ID: Not Applicable, set as 65535.  
29-bit Ext\_ID: SourceAddress of Message ID

(\*\*\*) NOTE: Add 32768 to the desired transmit rate to send the PGN value as a Standard ID.

Keypad functionality can be used in XML mode in a similar way to Keypad I/O Mode. Keypad configuration is identical to Keypad I/O Mode. The difference being that in XML mode the buttons do not directly control the outputs. The user is required to translate the buttons states into action via the XML logic loop.

The keypad button states are stored in CAN 63-66. The state is dependent on the button configuration and the keypad LEDs can be controlled either automatically or by writing to CAN 71-78. See section 3.3.4 Keypad Mode for more information on how to initialize keypad values.


Tmp Variables Type (01b) – Index List of Values			
0	FALSE	Binary	FALSE / 0 value
1	TRUE	Binary	TRUE / 1 value
2	<Blank>		Variables are open and available for the User
3	<Blank>		
...	...		
126	<Blank>		
127	<Blank>		

Parameters Variables Type (11b) – Index List of Values			
0	<Blank>		Variables are open and available for the User
1	<Blank>		
2	<Blank>		
3	<Blank>		
4	<Blank>		
5	<Blank>		
...	...		
126	<Blank>		
127	<Blank>		



## Example XML Configuration

Operator	Variable	Value	
Set_Var	Phy 63	0	Mode = XML
Set_Var	Phy 54	1	Output1 = HS_Dig
Set_Var	Phy 55	2	Output2 = LS_Dig
Set_Var	Phy 56	4	Output3 = HS_DC
Set_Var	Phy 57	6	Output4 = LS_DC
Set_Var	Phy 53	100	PWM Freq = 100Hz


**XML Logic**

## Example XML for Custom Keypad Behavior

Operator	Variable	Value	
Set_Var	Phy 63	0	Mode = XML
Set_Var	Phy 62	192	Keypad SA = 0xC0
Set_Var	Phy 58	2	Output 1 controlled by Button 2; Type = Momentary
Set_Var	Phy 59	521	Output 2 controlled by Button 9; Type = Tri-State
Set_Var	Phy 54	1	Output1 = Dig Active Hi
Set_Var	Phy 55	1	Output2 = Dig Active Hi
Set_Var	Tmp 3	68	Constant for LED command (Reference Pg. 31)
Set_Var	Tmp 4	2	Constant for Button State that turns on output (Reference Pg. 13 & 16)

### XML Logic

Operator	A Variable	B Variable	C Variable	
Buf	Can 63		Phy 24	Output 1 state = Button 2 state
Buf	Tmp 0		Tmp 5	Clear Flag before use
IF(A = B)-> C=1	Can 64	Tmp 4	Tmp 5	If button 9 state is 2, then set flag
IF(A = 1) B->C	Tmp 5	Tmp 1	Phy 25	If flag is set, then turn ON output 2
Buf	Tmp 0		Can 67	Clear Button 1 LED command
If A=true; B->C	Can 63	Tmp 3	Can 67	If Button 1 is pressed, Send custom LED command

\*Button 9 will use standard LED command, not a custom one, so it doesn't need to be set.

### 3.3.2 CAN I/O Mode

#### 3.3.2.1 Mode Description

In CAN I/O mode, the 505004 module is commanded by another controller via CAN communication. The 505004 module reports its input statuses, while the commanding module commands the 505004 module to turn its outputs on or off. The CAN messages involved in this functionality are included below for reference. While the 505004 module automatically handles these CAN messages, some CAN functionality is configurable (e.g. the transmit rate for each message, and whether they are transmitted at all).

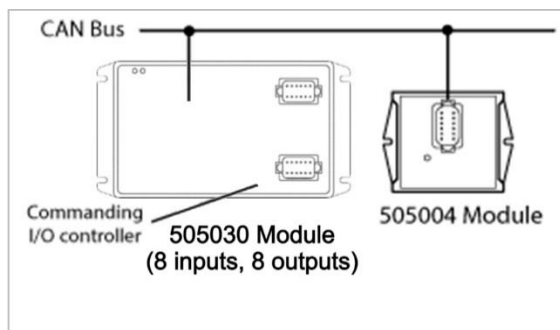


Figure 16

#### Transmitted CAN messages:

	PGN	Data Bytes							
		LSB				MSB			
Digital Status Msg	18FF40sa	In1-4(Hi)	In1-4(Lo)	0x00	0x00	0x00	0x00	0x00	0x00
2-bit Status Flags[00=False, 01=True, 10=Error, 11=Not Applicable]									
Analog Status Msg	18FF41sa	Input 1 Analog		Input 2 Analog		Input 3 Analog		Input 4 Analog	
Voltage: 0-42000 [0-42,000mV, 1mV/bit] / Quad Count: 0-65535 [1cnt/bit]									
Analog Status Msg	18FF43sa	Input 1 Frequency		Input 2 Frequency		Input 1 Duty Cycle		Input 2 Duty Cycle	
Frequency: 32-10,000Hz [0.1Hz/bit]      Duty Cycle: 0-100% [0.1%/bit]									
Module Status Msg	18FF60sa	Supply Voltage		0xFFFF		Supply Raw ADC		Module HWID	
Output Fault Msg	18FF61sa	Out1Err	Out2Err	Out3Err	Out4Err	0xFF	0xFF	0xFF	0xFF
Output Setpoint Msg	18FF63sa	Out 1 Cmd (16-bit)		Out 2 Cmd (16-bit)		Out 3 Cmd (16-bit)		Out 4 Cmd (16-bit)	
Output Feedback Msg	18FF64sa	Output 1 Current		Output 2 Current		Output 3 Current		Output 4 Current	
Current: 0-4200 [0-4,200mA, 1mA/bit]									

#### Notes:

- sa = Source Address of the 505004 Module
- xx = Source Address of the controller commanding the 505004 Module
- Beacon: Flashes the status LED so that the commanded module can be identified

### Binary Active Hi/Lo Status – Detailed View

In1-4(Hi)	In1-4(Lo)	0x00	0x00	0x00	0x00	0x00	0x00
-----------	-----------	------	------	------	------	------	------

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Input 4		Input 3		Input 2		Input 1	

Note: Each input has a 2-bit flag: 00 – Input is Not Active  
 (Ref Pg.3) 01 – Input is Active  
 1X – Error

### Fault Status – Detailed View

Out1Err	Out2Err	Out3Err	Out4Err	0xFF	0xFF	0xFF	0xFF
---------	---------	---------	---------	------	------	------	------

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-------	-------	-------	-------	-------	-------	-------	-------

*Undefined   Undefined   Undefined   Undefined   Undefined   Undefined   Undefined   Over Current Fault*

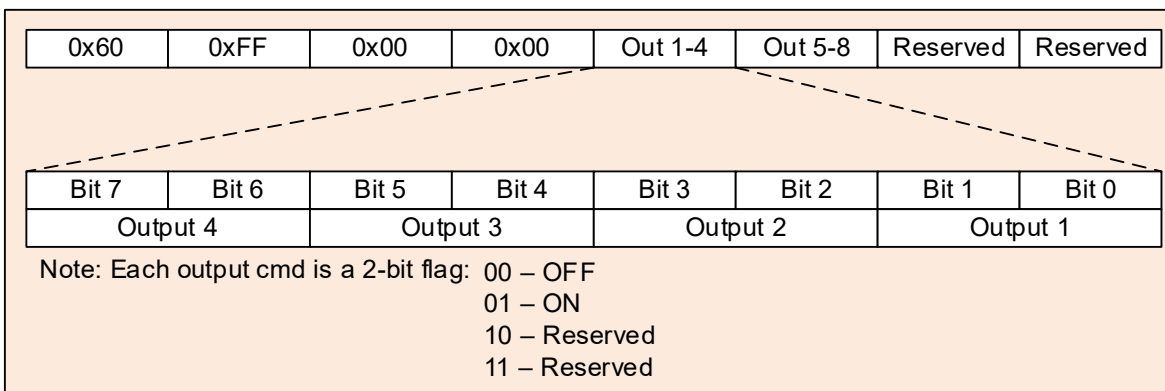
**Received CAN messages:**

	PGN	Data Bytes							
		LSB				MSB			
Cmd Msg: Outputs	18EFsaxx	0x60	0xFF	0x00	0x00	Out 1-4	0x00	0x00	0x00
		0x61	0xFF	0x00	0x00	Out1 Cmd	Out2 Cmd	Out3 Cmd	Out4 Cmd
		Duty Cycle: 0-250 [0-100.0%, 0.4%/bit]							
		0x62	0xFF	0x00	0x00	Out 1 Cmd (16-bit)	Out 2 Cmd (16-bit)		
					0x01	Out 3 Cmd (16-bit)	Out 4 Cmd (16-bit)		
		Duty Cycle: 0-1000 [0-100.0%, 0.1%/bit] mA: 0-3000 [0-3000mA, 1mA/bit]							
		0x70	0xFF	0x00	0x00	In 1 Quad Cnt Cmd	0x00	0x00	
		Count Setpoint [0-65535]							
Cmd Msg: Beacon On	18EFsaF9	0x15	0xFF	0xF9	0xB0	0x44	0x55	0x66	0x77
Cmd Msg: Beacon Off	18EFsaF9	0x16	0xFF	0xF9	0xB0	0x44	0x55	0x66	0x77

## Notes:

- sa = Source Address of the 505004 Module
- xx = Source Address of the controller commanding the 505004 Module
- Beacon: Flashes the status LED so that the commanded module can be identified

### Binary Active Hi/Lo Control – Detailed View



Outputs (and inputs in certain circumstances) are commanded by PGN 0xEF00 when the module is configured for CAN I/O Mode. The command operation is specified by the messages first data byte.

The command operation 0x70 acts as means to reset or initialize the quadrature count for compatible inputs.

#### 3.3.2.2 Configuring for CAN I/O Mode

To configure the various module parameters involved in CAN I/O Mode, open a Configurator window as described in section 3.3 *Configuring the Module*. Use the ADD button to add **SET VARIABLE** instructions for each parameter listed in the *List of CAN I/O Mode Parameters* below. Refer to Figure 17 as a guide for filling out the **SET VARIABLE** instructions. If necessary, see section 3.3 *Configuring the Module* for more details about adding **SET VARIABLE** instructions.

A Type	A Index	Operator	Value	Comment
<div style="border: 1px solid black; height: 20px; width: 100%;"></div>	<div style="border: 1px solid black; height: 20px; width: 100%;"></div>	SET VARIABLE	= <div style="border: 1px solid black; width: 150px;"></div> 0x03BF0000	<div style="border: 1px solid black; height: 20px; width: 100%;"></div>

Set these according to the "A Type" and "A Index" columns of the table

Always set to "SET VARIABLE"

Set as desired, within range specified in "Parameter Description" column of table

Figure 17

#### List of CAN I/O Mode Related Parameters

A Type	A Index	Parameter Description
Physical	63	Module Mode. Must be 1 for CAN I/O Mode.
Physical	40	Threshold Voltage for Digital inputs to be set to Active [65.535mV, 1mV/bit]
Physical	41	Hysteresis Voltage for Digital inputs to return to Inactive [65.535mV, 1mV/bit]
Physical	42	P-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	43	I-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	44	D-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	32	Output 1 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	33	Output 2 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]

A Type	A Index	Parameter Description
Physical	34	Output 3 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	35	Output 4 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	53	Set Base Frequency of all PWM Outputs [30-1000Hz, 1Hz/bit]
Physical	54	Output 1 Type <sup>1</sup>
Physical	55	Output 2 Type <sup>1</sup>
Physical	56	Output 3 Type <sup>1</sup>
Physical	57	Output 4 Type <sup>1</sup>
Physical	64	Input 1 Type <sup>2</sup>
Physical	36	Output 1 Current Limit [1 mA/bit, 0-3000mA]
Physical	37	Output 2 Current Limit [1 mA/bit, 0-3000mA]
Physical	38	Output 3 Current Limit [1 mA/bit, 0-3000mA]
Physical	39	Output 4 Current Limit [1 mA/bit, 0-3000mA]
CAN	48	Tx Rate of Msg FF40 [10mS/bit, 0=Off]
CAN	49	Tx Rate of Msg FF41 [10mS/bit, 0=Off]
CAN	51	Tx Rate of Msg FF42 [10mS/bit, 0=Off]
CAN	53	Tx Rate of Msg FF60 [10mS/bit, 0=Off]
CAN	54	Tx Rate of Msg FF61 [10mS/bit, 0=Off]
CAN	55	Tx Rate of Msg FF63 [10mS/bit, 0=Off]
CAN	56	Tx Rate of Msg FF64 [10mS/bit, 0=Off]
CAN	79	Cmd Message Timeouts [10mS/bit, 0=No Timeout]

- <sup>1</sup> 0 = Off  
1 = High Side Digital (Active High)  
2 = Low Side Digital (Active Low)  
3 = High Side Current (mA)  
4 = High Side PWM (Duty Cycle)  
5 = Low Side Current (mA)  
6 = Low Side PWM (Duty Cycle)  
7 = Bi-Directional (Half-Bridge)
- <sup>2</sup> 0 = Digital/Analog/PWM [Default]  
1 = Current  
2 = Resistance  
3 = Quadrature X1  
4 = Quadrature X2  
5 = Quadrature X4

## Example XML Configuration

Operator	Variable	Value	
Set_Var	Phy 63	1	Mode = CAN_I/O
Set_Var	Phy 54	1	Output1 = HS_Dig
Set_Var	Phy 55	2	Output2 = LS_Dig
Set_Var	Phy 56	4	Output3 = HS_DC
Set_Var	Phy 57	6	Output4 = LS_DC
Set_Var	Phy 53	100	PWM Freq = 100Hz
Set_Var	Can 1	10	CAN Msg 0xFF41 Tx Rate = 100ms
Set_Var	Can 63	100	CAN Cmd Timeout = 1s

### 3.3.3 Hardware I/O Mode

#### 3.3.3.1 Mode Description

In Hardware I/O Mode, the module functions like a relay module, in which input signals drive corresponding output signals. When a given input is connected to power or ground, the input state is set to TRUE. Then, after the user-specified On-Delay time ( $Td_{On}$  in Figure 18), the state of the corresponding output changes to TRUE (for digital outputs, that means the output turns on; for PWM outputs or current outputs, that means the output is set to the specified output Duty Cycle or Output Current, respectively (see Physical parameters 36-39 in *List of Hardware I/O Mode Parameters* below). When power or ground is removed from the input, the input floats at half the system voltage, which sets the input state to FALSE. After the user-specified Off-Delay time ( $Td_{Off}$  in Figure 18), the output state then changes to FALSE.

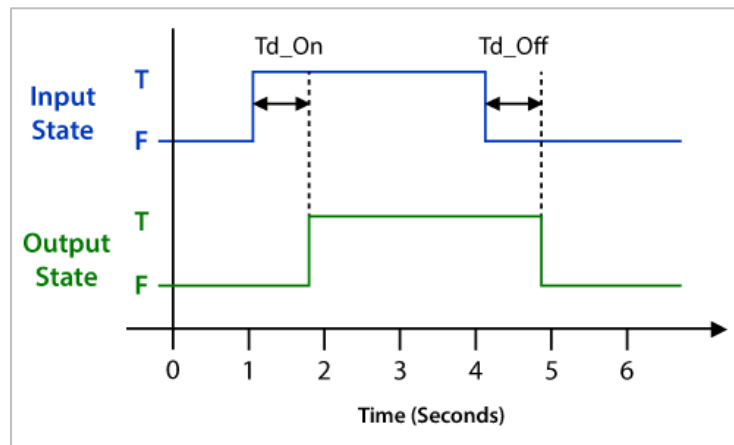


Figure 18

#### 3.3.3.2 Configuring for Hardware I/O Mode

To configure the various module parameters involved in Hardware I/O Mode, open a Configurator window as described in section 3.3 *Configuring the Module*. Use the ADD button to add *SET VARIABLE* instructions for each parameter listed in the *List of Hardware I/O Mode Parameters* below. Refer to Figure 19 as a guide for filling out the *SET VARIABLE* instructions. If necessary, see section 3.3 *Configuring the Module* for more details about adding *SET VARIABLE* instructions.

A Type	A Index	Operator	Value	Comment
<input type="text"/>	<input type="text"/>	SET VARIABLE	= <input type="text"/> 0x03BF0000	<input type="text"/>

Set these according to the "A Type" and "A Index" columns of the table

Always set to "SET VARIABLE"

Set as desired, within range specified in "Parameter Description" column of table

Figure 19

#### List of Hardware I/O Mode Related Parameters

A Type	A Index	Parameter Description
Physical	63	Module Mode. Must be 2 for Hardware I/O Mode.
Physical	53	Set Base Frequency of all PWM Outputs [30-1000Hz, 1Hz/bit]
Physical	54	Output 1 Type <sup>1</sup>
Physical	55	Output 2 Type <sup>1</sup>
Physical	56	Output 3 Type <sup>1</sup>
Physical	57	Output 4 Type <sup>1</sup>
Physical	32	Output 1 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	33	Output 2 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	34	Output 3 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	35	Output 4 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	40	Threshold Voltage for Digital inputs to be set to Active [65.535mV, 1mV/bit]
Physical	41	Hysteresis Voltage for Digital inputs to return to Inactive [65.535mV, 1mV/bit]
Physical	42	P-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	43	I-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	44	D-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	36	Output 1 Current Limit [1 mA/bit, 0-3000mA]
Physical	37	Output 2 Current Limit [1 mA/bit, 0-3000mA]
Physical	38	Output 3 Current Limit [1 mA/bit, 0-3000mA]
Physical	39	Output 4 Current Limit [1 mA/bit, 0-3000mA]
Physical	28	Output 1 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	29	Output 2 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	30	Output 3 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	31	Output 4 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	45	Relay-Style On-Delay for Output 1 [0-655,350mS, 10mS/bit]
Physical	46	Relay-Style On-Delay for Output 2 [0-655,350mS, 10mS/bit]
Physical	47	Relay-Style On-Delay for Output 3 [0-655,350mS, 10mS/bit]
Physical	48	Relay-Style On-Delay for Output 4 [0-655,350mS, 10mS/bit]
Physical	49	Relay-Style Off-Delay for Output 1 [0-655,350mS, 10mS/bit]
Physical	50	Relay-Style Off-Delay for Output 2 [0-655,350mS, 10mS/bit]
Physical	51	Relay-Style Off-Delay for Output 3 [0-655,350mS, 10mS/bit]
Physical	52	Relay-Style Off-Delay for Output 4 [0-655,350mS, 10mS/bit]

<sup>1</sup> 0 = Off

1 = High Side Digital (Active High)

2 = Low Side Digital (Active Low)

3 = High Side Current (mA)

4 = High Side PWM (Duty Cycle)

5 = Low Side Current (mA)

6 = Low Side PWM (Duty Cycle)

7 = Bi-Directional (Half-Bridge)

<sup>2</sup> This parameter only applies if the corresponding output is configured as Current or PWM, per Physical  
It determines the output level when a keypad button press commands the output to its ON state.

# Example XML Configuration

Operator	Variable		Value	
Set_Var	Phy	63	2	Mode = HW_I/O
Set_Var	Phy	54	1	Output1 = HS_Dig
Set_Var	Phy	55	2	Output2 = LS_Dig
Set_Var	Phy	56	4	Output3 = HS_DC
Set_Var	Phy	57	6	Output4 = LS_DC
Set_Var	Phy	53	100	PWM Freq = 100Hz
Set_Var	Phy	28	1000	OutCmd1 = 100%
Set_Var	Phy	29	1000	OutCmd2 = 100%
Set_Var	Phy	30	1000	OutCmd3 = 100%
Set_Var	Phy	31	1000	OutCmd4 = 100%
Set_Var	Phy	45	100	Out1 On Delay = 1.0s
Set_Var	Phy	49	150	Out1 Off Delay = 1.5s
Set_Var	Phy	46	200	Out2 On Delay = 2.0s
Set_Var	Phy	50	250	Out2 Off Delay = 2.5s
Set_Var	Phy	47	300	Out3 On Delay = 3.0s
Set_Var	Phy	51	350	Out3 Off Delay = 3.5s
Set_Var	Phy	48	400	Out4 Onf Delay = 4.0s
Set_Var	Phy	52	450	Out4 Off Delay = 4.5s



### 3.3.4 Keypad Mode

#### 3.3.4.1 Mode Description

In Keypad mode, the 505004 Module's outputs are driven by button presses on a Marlin 5052xx keypad. LEDs on each button are turned on or off, giving the operator visual feedback when a button is pressed.

Individual keypad buttons can be assigned to drive individual outputs on the 505004 module. The type of output (digital, current, PWM) can also be assigned. Furthermore, the button's functionality, can be configured to be one of three types:

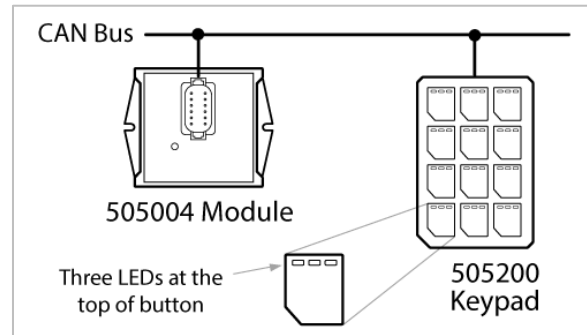


Figure 20

- **Momentary:** The output is set to its On state for as long as the button is held down. When the button is released, the output returns to its Off state. The button's left LED is On when the output is On, and Off when the output is Off. The middle and right LEDs are not used in this mode.
  - Note: for outputs configured as PWM or Current, the On state is defined by the output's respective *Output Duty Cycle/Current* parameter. See Physical parameters 28-31 in the *List of Keypad Mode Parameters* below.
- **On/Off Toggle:** The output is toggled between its On and Off state each time the button is pushed downwards. For example, with the output initially off, the operator pushes the button down and thus causes the output to turn On. The output remains On as the operator releases the button. The output is turned Off when the operator again presses the button down, and remains Off as the button is released, and so on. The button's left LED is On when the output is On, and Off when the output is Off. The middle and right LEDs are not used in this mode.
  - Note: for outputs configured as PWM or Current, the On state is defined by the output's respective *Output Duty Cycle/Current* parameter. See Physical parameters 28-31 in the *List of Keypad Mode Parameters* below.
- **Off/Low/Medium/High (O/L/M/H):** This functionality applies only to outputs configured as PWM or Current (i.e. not Digital). As with On/Off Toggle functionality, the output state changes only when the button is pressed down (and not when the button is released). But instead of the output simply turning on or off, a button press causes the output signal to cycle between four different states: Off (0% of the fully-on state), Low (33% of the fully-on state), Medium (66% of the fully-on state), and High (100% of the fully-on state). The "fully-on state" is defined by the output's respective *Output Duty Cycle/Current* parameter. See Physical parameters 28-31 in the *List of Keypad Mode Parameters* below.

The three types of button functionality are represented visually in Figure 21.

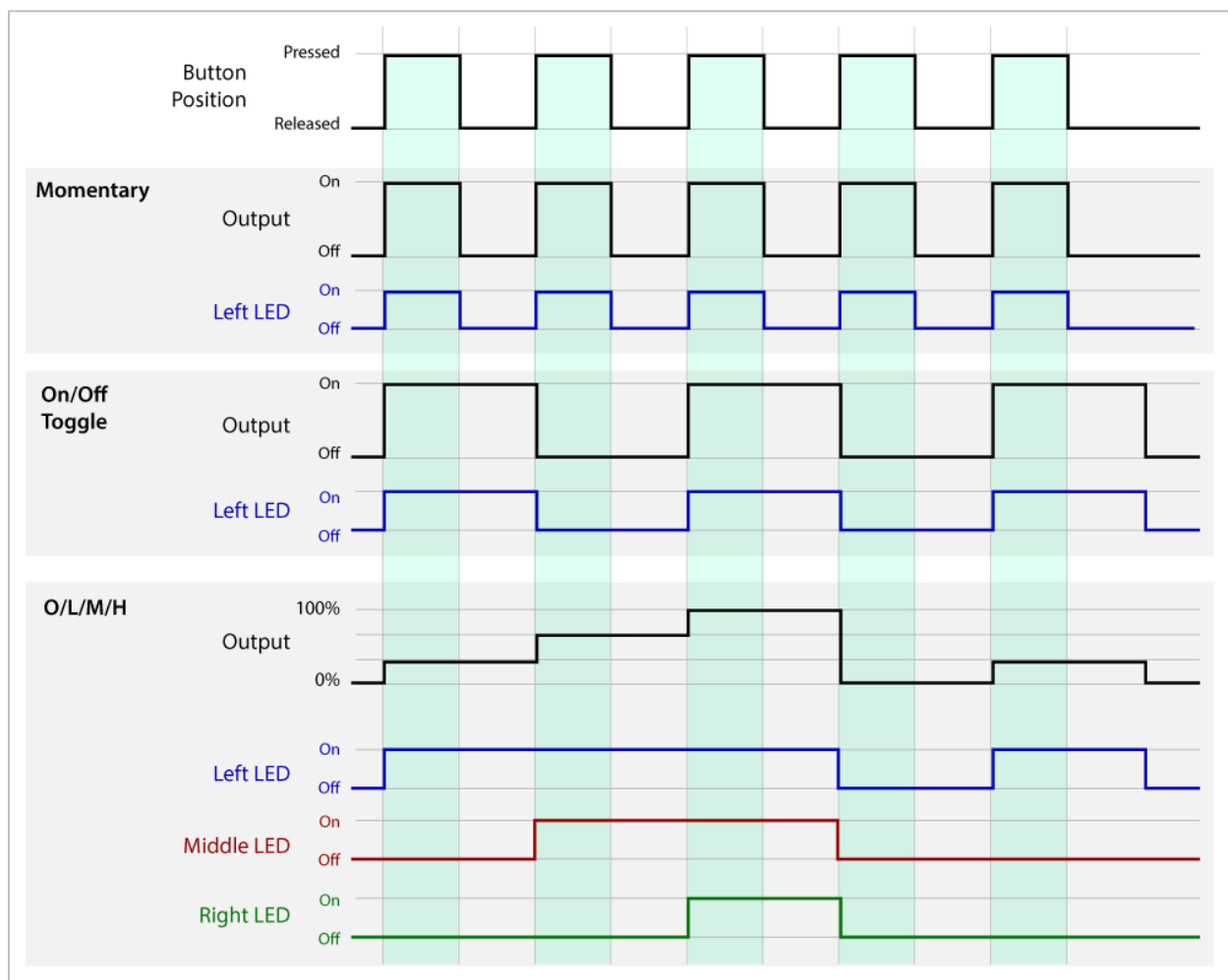
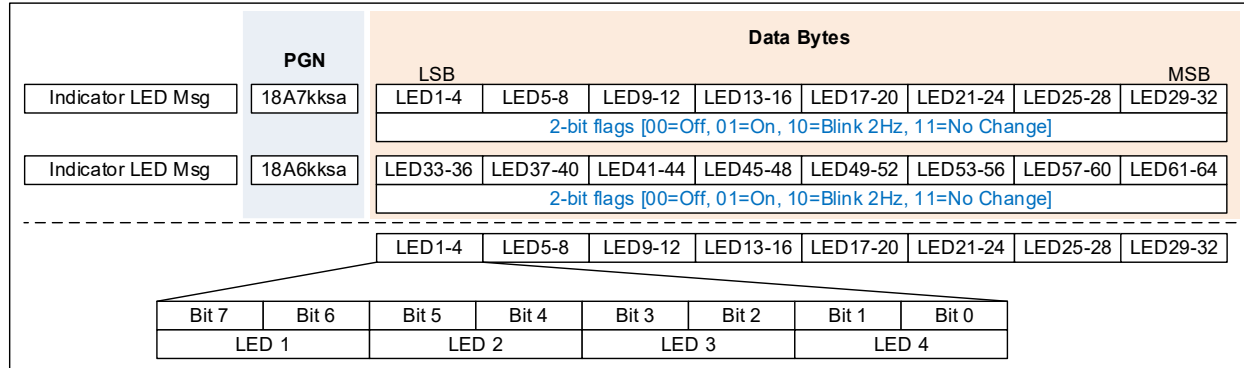


Figure 21

The 505004 module receives CAN messages from the keypad indicating when buttons are pressed. In turn, the 505004 module turns outputs on or off, and transmits CAN messages that turns LEDs on or off on the keypad's buttons. The CAN messages involved are included below for reference. While the 505004 module automatically handles these CAN messages, some CAN functionality is configurable.

### CAN messages Transmitted by 505004:



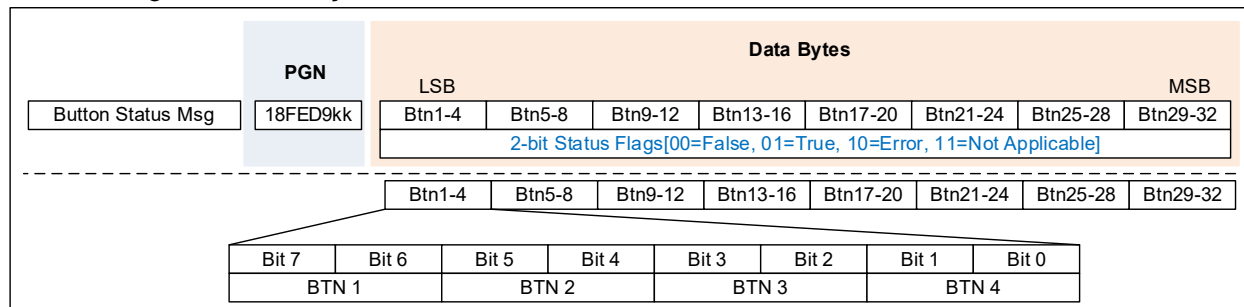
#### Keypad LED Command Byte Format

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Left LED		Middle LED		Right LED			

#### Note1)

Each LED cmd is a 2-bit flag 0=Off, 1=On, 2=Blink 2Hz, 3=No Change

### CAN messages Received by 505004:



#### Notes:

- Indicator LED Status Messages are sent every 100mS.
- sa = Source Address of the 505004 Module
- kk = Source Address of the Keypad Module

### 3.3.4.2 Configuring for Keypad I/O Mode

To configure the various module parameters involved in Keypad Mode, open a Configurator window as described in section 3.3 *Configuring the Module*. Use the ADD button add SET VARIABLE instructions for each parameter listed in the *List of Keypad Mode Parameters* below.

A Type	A Index	Operator	Value	Comment
<input type="text"/>	<input type="text"/>	SET VARIABLE	= <input type="text"/> 0x03BF0000	<input type="text"/>

Set these according to the "A Type" and "A Index" columns of the table

Always set to "SET VARIABLE"

Set as desired, within range specified in "Parameter Description" column of table

Figure 22

Refer to Figure 22 as a guide for filling out the *SET VARIABLE* instructions. If necessary, see section 3.3 *Configuring the Module* for more details about adding *SET VARIABLE* instructions.

#### List of Keypad Mode Related Parameters

A Type	A Index	Parameter Description
Physical	63	Module Mode. Must be 3 for Keypad Mode .
Physical	53	Set Base Frequency of all PWM Outputs [30-1000Hz, 1Hz/bit]
Physical	54	Output 1 Type <sup>1</sup>
Physical	55	Output 2 Type <sup>1</sup>
Physical	56	Output 3 Type <sup>1</sup>
Physical	57	Output 4 Type <sup>1</sup>
Physical	32	Output 1 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	33	Output 2 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	34	Output 3 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	35	Output 4 Closed Loop PWM Nominal Load Resistance [10mΩ/bit]
Physical	42	P-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	43	I-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	44	D-term for Closed-Loop of all outputs [0-655.35%, 0.01%/bit]
Physical	36	Output 1 Current Limit [1 mA/bit, 0-3000mA]
Physical	37	Output 2 Current Limit [1 mA/bit, 0-3000mA]
Physical	38	Output 3 Current Limit [1 mA/bit, 0-3000mA]
Physical	39	Output 4 Current Limit [1 mA/bit, 0-3000mA]
Physical	28	Output 1 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	29	Output 2 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	30	Output 3 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	31	Output 4 Duty Cycle/Current [0.1%/bit or 1 mA/bit] <sup>2</sup>
Physical	45	Output 1 On Delay [0-65535ms, 10ms/bit]
Physical	46	Output 2 On Delay [0-65535ms, 10ms/bit]
Physical	47	Output 3 On Delay [0-65535ms, 10ms/bit]
Physical	48	Output 4 On Delay [0-65535ms, 10ms/bit]
Physical	49	Output 1 Off Delay [0-65535ms, 10ms/bit]
Physical	50	Output 2 Off Delay [0-65535ms, 10ms/bit]
Physical	51	Output 3 Off Delay [0-65535ms, 10ms/bit]
Physical	52	Output 4 Off Delay [0-65535ms, 10ms/bit]
Physical	58	Keypad Button Index and Button Type for Output 1 <sup>3</sup>
Physical	59	Keypad Button Index and Button Type for Output 2 <sup>3</sup>
Physical	60	Keypad Button Index and Button Type for Output 3 <sup>3</sup>
Physical	61	Keypad Button Index and Button Type for Output 4 <sup>3</sup>
Physical	62	Keypad Source Address
CAN	63	Cmd Message Timeouts [10mS/bit, 0=NoTimeout]

<sup>1</sup> 0 = Off

- 1 = High Side Digital (Active High)
- 2 = Low Side Digital (Active Low)
- 3 = High Side Current (mA)
- 4 = High Side PWM (Duty Cycle)
- 5 = Low Side Current (mA)
- 6 = Low Side PWM (Duty Cycle)
- 7 = Bi-Directional (Half-Bridge)

<sup>2</sup> This parameter only applies if the corresponding output is configured as Current or PWM, per Physical. It determines the output level when a keypad button press commands the output to its On state.

- 3 This parameter a) identifies which button on the keypad drives the given output (i.e. its Index Value), and b) specifies the button's functionality (i.e. its Type). The parameter's value is in the following 16-bit format:

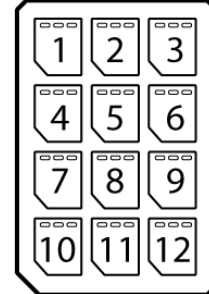
Bits 15:10	Bits 9:8	Bits 7:0
Not Used	Button Type	Button Index Number

**Button Index Number:** Buttons on a keypad are typically numbered in order from left to right, top row to bottom as follows, as shown in the example to the right. Thus, the Index Number of the lower-left button on a the 3x4 keypad in the example would be 12.

**Button Type:** This following values select the types of button functionality as follows:

- 0 = Momentary
- 1 = On/Off Toggle
- 2 = Off/Low/Medium/High (for PWM and Current outputs only)

See above (including Figure 21) for details about the types of button functionality.



## Example XML Configuration

Operator	Variable		Value	
Set_Var	Phy	63	3	Mode = Keypad
Set_Var	Phy	54	1	Output1 = HS_Dig
Set_Var	Phy	55	2	Output2 = LS_Dig
Set_Var	Phy	56	4	Output3 = HS_DC
Set_Var	Phy	57	6	Output4 = LS_DC
Set_Var	Phy	53	100	PWM Freq = 100Hz
Set_Var	Phy	28	1000	OutCmd1 = 100%
Set_Var	Phy	29	1000	OutCmd2 = 100%
Set_Var	Phy	30	1000	OutCmd3 = 100%
Set_Var	Phy	31	1000	OutCmd4 = 100%
Set_Var	Phy	62	64	SourceAddr = 0x40(64)
Set_Var	Phy	58	1	Output1 controlled by Button1; Type momentary
Set_Var	Phy	59	2	Output2 controlled by Button2; Type momentary
Set_Var	Phy	60	3	Output3 controlled by Button3; Type momentary
Set_Var	Phy	61	4	Output4 controlled by Button4; Type momentary

### 3.3.4.3 Configuring the Marlin M-FLEX Keypad

Configuring Keypad Source Address										
		Byte 1				Byte 8				
1	Enter Config Mode	18EFF9kk	0x10	0xFF	0xF9	kk	0x44	0x55	0x66	0x77
2	R/W Source Address	18EFF9kk	0x20	0x10/0x11	0xFF	New sa	0xFF	0xFF	0xFF	0xFF
3	Exit Config Mode	18EFF9kk	0x11	0xFF	0xF9	kk	0x44	0x55	0x66	0x77

Configuring Keypad LED Brightness										
		Byte 1				Byte 8				
LED Brightness CMD	18D0sakk	CMD*	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF	0xFF

\* 0-100 percent (0.4% per bit)

Notes:

- sa = Source Address of the commanding Module
- kk = Current Source Address of the Keypad Module
- When in config mode, keypad will echo CAN message with data byte 2 containing the success/Failure response (0x12 = success, others = failure)

### 3.4 Loading the User Program

Once a user program has been created in the Programming Tool's Configurator utility and saved to an .XML file, it can then be loaded to the 505004 module using the Programming Tool's EEPROM Read/Write window. With the Program Tool running and connected to the module (see section 3.1 *Connecting to the Module with the Programming Tool* for details), click the menu item "XML", then click the item "Single," both shown circled in Figure 23.



Figure 23

A file explorer will pop up in which the desired .XML file can be browsed and selected. After the file is selected, the EEPROM Read/Write window will pop up, as shown in Figure 24. If a different .XML file is desired, click on the yellow "Open XML" button near the top right corner and select a new file. Otherwise, locate the source address field on the right side of the window, just above the green buttons. Fill in the field with the current source address of the 505004 module. (If the source address is unknown, use the Search button in the Programming Tool window, as shown in Figure 23, to request the module's information. The source address in the figure is 0xB8.) Click the "Write to EEPROM" button. A pop-up warning will be displayed indicating the controller's EEPROM data will be overwritten. Click "OK" to proceed. If the data from the .XML file is programmed successfully, the text "All Attributes were Successfully Written to EEPROM!" will appear (in green) in the text box at the bottom of the window (see Figure 24).

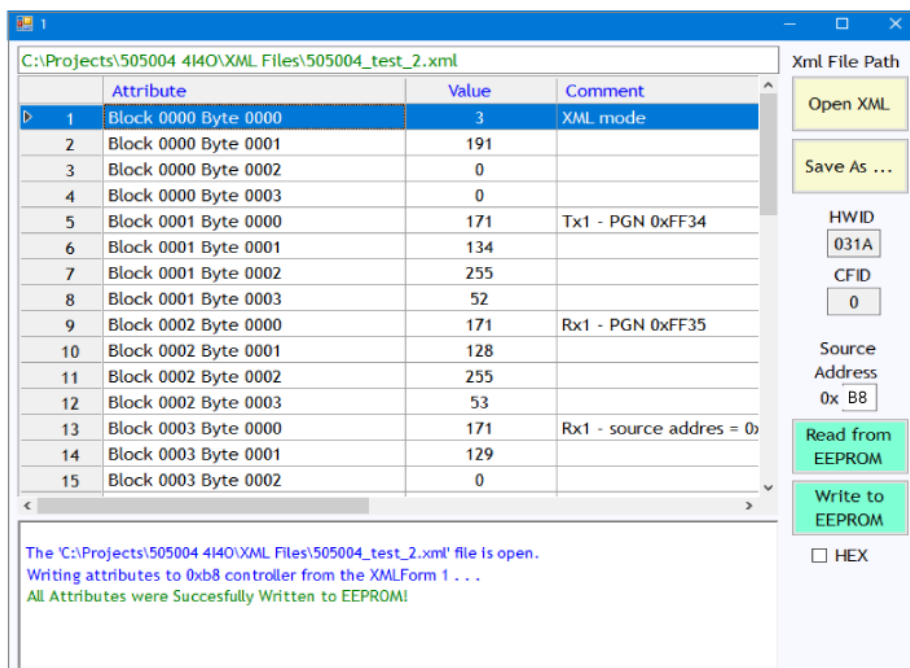


Figure 24

If there are errors, consult the Programming Tool user guide mentioned in section



---

2. *Software* Installation to troubleshoot.

## 4. LED Module

The 505004 module has built-in support for the Marlin 5010xx LED Module, which can be used to monitor the module's I/O for diagnostic purposes. The module's I/O statuses are transmitted to the LED module via a CAN message. This CAN message is enabled by default, but can be either disabled or its transmitting frequency changed. By default, it is enabled at a transmission rate of 100ms.

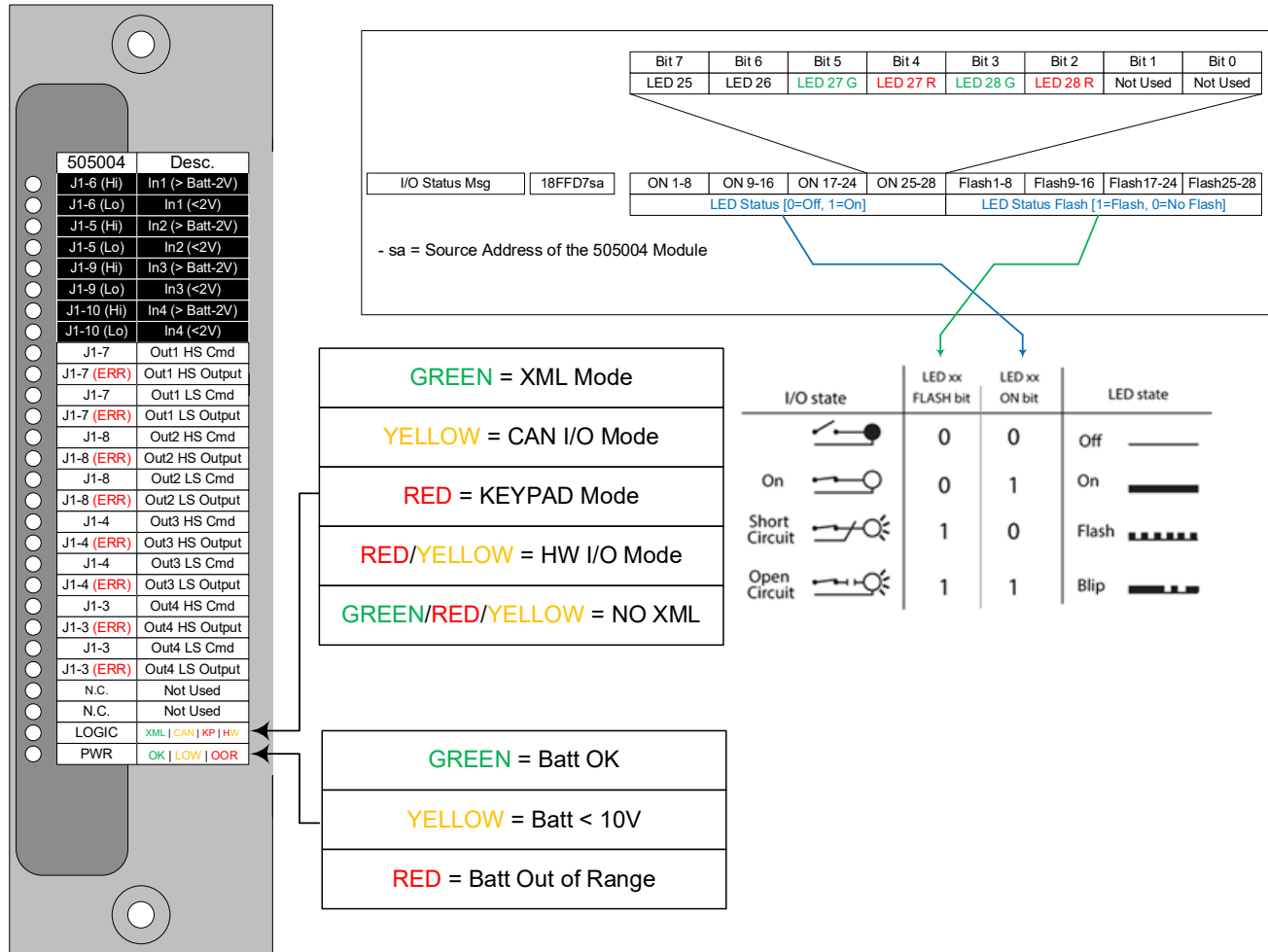


Figure 25

## 5. Low Power Mode

The 505004 supports a low power mode in every mode of operation. Low power mode is disabled by default and requires configuration to be able to use. Configuration is set via index 68 in the Phy list of config values. This configuration value is able to be changed dynamically in XML mode if needed.

A Type	A Index	Parameter Description
Physical	68	Low Power configuration.

Below is the breakdown of how to set the low power mode configuration.

### Config Breakdown

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Enable L-P Mode	Not Used	Not Used	CAN Wakeup	In2 Pos. Edge	In2 Neg. Edge	In1 Pos. Edge	In1 Neg. Edge

\*Bits 15-8 are not used

Note: If bit 7 of the config value is not set, then low power mode will remain disabled regardless of the state of the other bits.

The module enters low power mode via a CAN message as described below.

CAN msg received by 505004								
Cmd Msg: Low Power Mode	18FF28xx	sa	Cmd	D.C.	D.C.	D.C.	D.C.	D.C.

#### Note:

sa = Source address of 505004 module  
xx = Source address of commanding module  
DC = Don't care

#### Cmd:

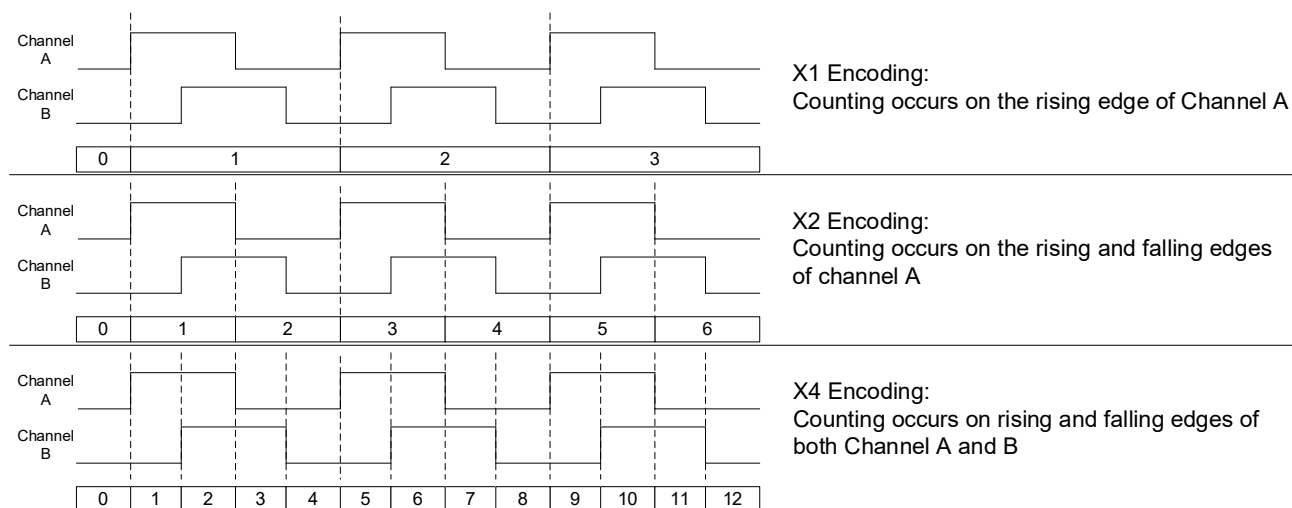
0 = Wake up (No effect if already awake)  
1 = Enter low power mode (No effect if in low power mode)  
Others = Ignored

During low power mode, the module current draw will be **approximately 9mA**. This may vary slightly depending on the states of the inputs and/or outputs, and the active bus load if wake via CAN is enabled (Ref notes below).

#### Other Notes:

- If low power mode is enabled and used without enabling any wake-up functions, the module will not be able to exit sleep mode except via a power cycle.
- When CAN wake-up is enabled, the module will briefly wake-up for every received CAN message. If the received message is not the wake-up message as described above, the module will go back into low power mode after approximately 10ms.
- Voltage thresholds for waking on edge detection are different than those configured via Physical values 40 & 41. The actual voltage values are as follows:
  - High-to-low transition (Negative Edge):  $V_{in} < 0.8V$
  - Low-to-High transition (Positive Edge):  $V_{in} > 2.0V$

## 6. Quadrature Inputs



When configured in CAN I/O mode, the total quadrature count and the interval count are transmitted in place of Input 'n' & Input 'n+1' Analog data respectively. The time between count samples is equal to the transmit rate of the 0xFF41 Input Status message.

Analog Status Msg	18FF41sa	Input 1 Total Count	Input 1 Interval Count	N/A	N/A
-------------------	----------	---------------------	------------------------	-----	-----

When configured for CAN I/O mode, the total quadrature count can be commanded to a specified value using the output command PGN 0xEF00 with the command operation of 0x70. This is useful for initializing the count after power up or zeroing of the count without power cycling the module. The interval count is not resettable and will reflect any change in count associated with this command in the next 0xFF41 transmission.

Cmd Msg: Outputs	18EFsaxx	0x70	0xFF	0x00	0x00	In 1 Quad Cnt Cmd	Reserved (leave as 0x00)
							Count Setpoint [0-65535]