

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:	17321E	DATE: 06/20/23



Table of Contents

1. Overview	2
2. Software Installation	3
3. Configuring and Programming the Module.....	3
3.1 Connecting to the Module with the Programming Tool.....	6
3.2 Updating the Base Software.....	6
3.3 Configuring the Module.....	7
3.3.1 XML Mode	11
3.3.1.1 Configuration.....	12
3.3.1.2 Logic.....	13
3.3.1.2.1 Operators.....	15
3.3.1.2.2 Module Parameters and Variables.....	19
3.3.1.2.3 Example.....	22
3.3.2 CAN I/O Mode	24
3.3.3 Hardware I/O Mode.....	28
3.3.4 Keypad Mode.....	31
3.4 Loading the User Program	37
4. LED Module	39

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.

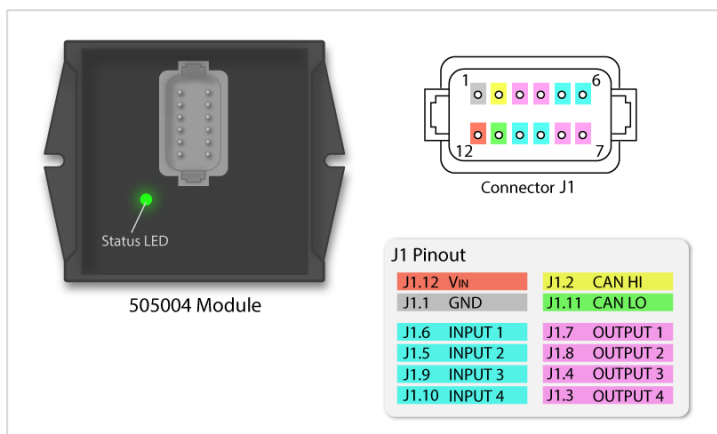


Figure 1

The input types supported by each of the input pins 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	✗	✓	✗	✗

The output types supported by each of the Each output pin 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 PWN – 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 on 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.

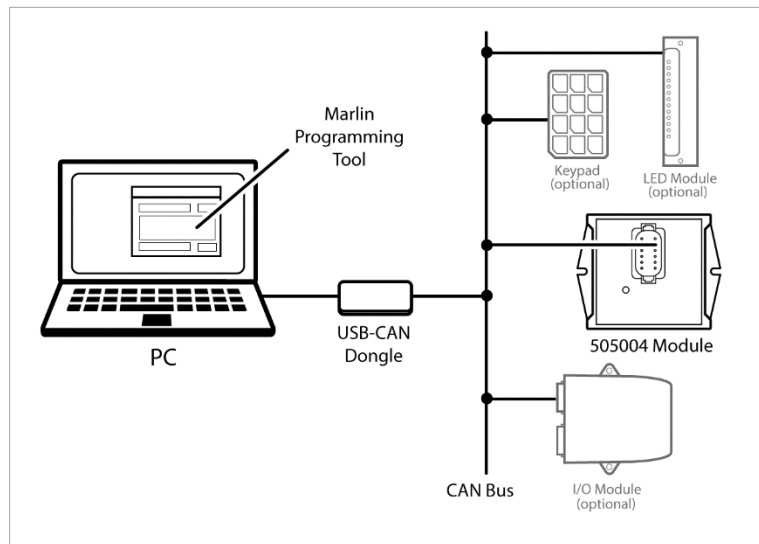


Figure 2

Programming Tool

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

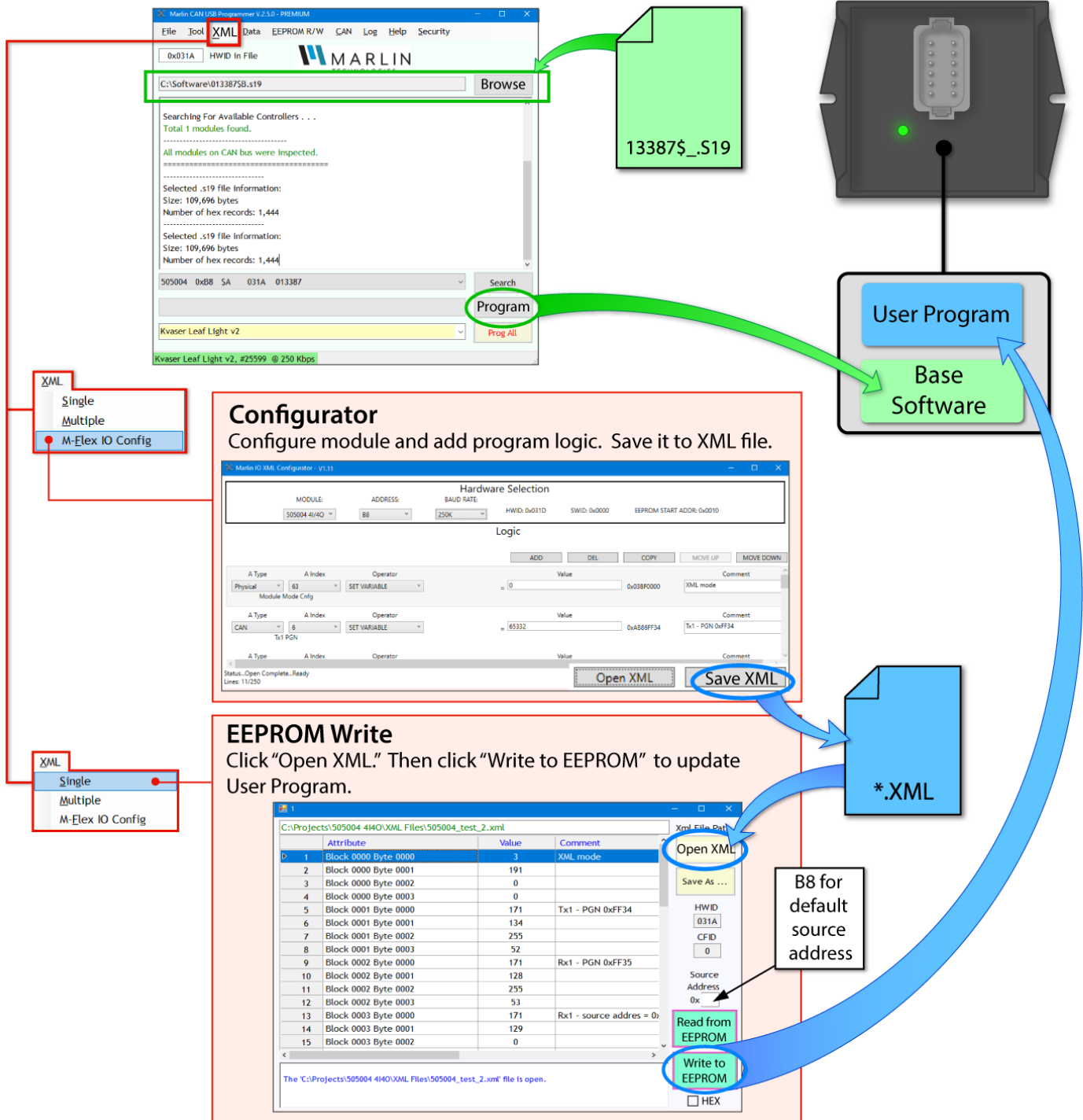


Figure 3

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*.

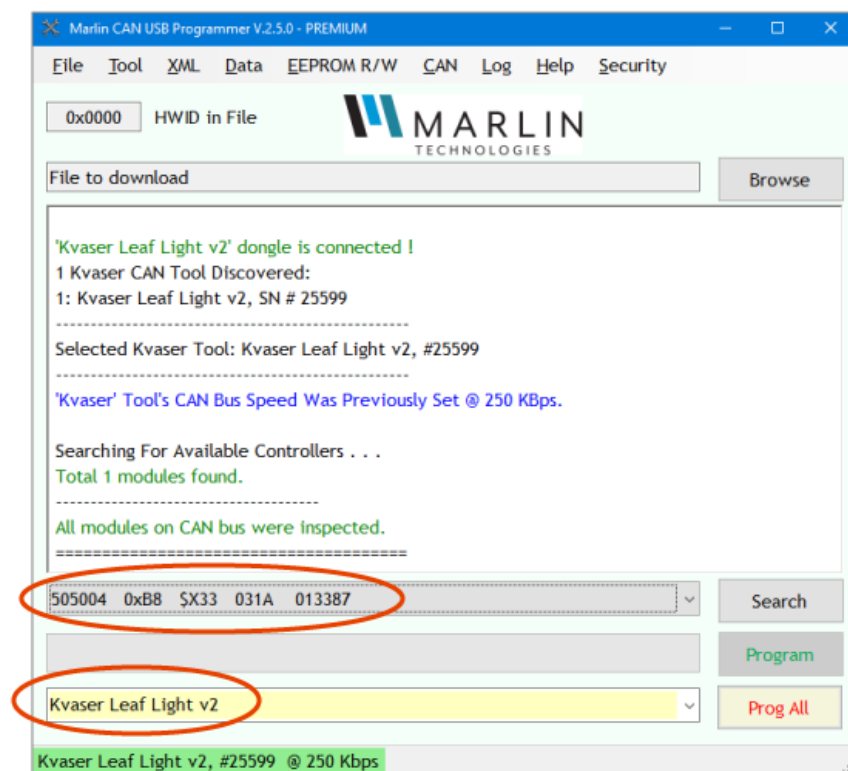


Figure 4

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.

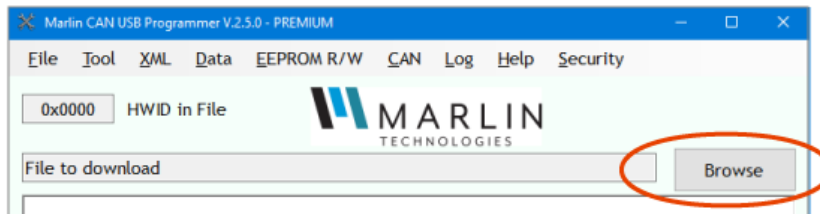


Figure 5

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.

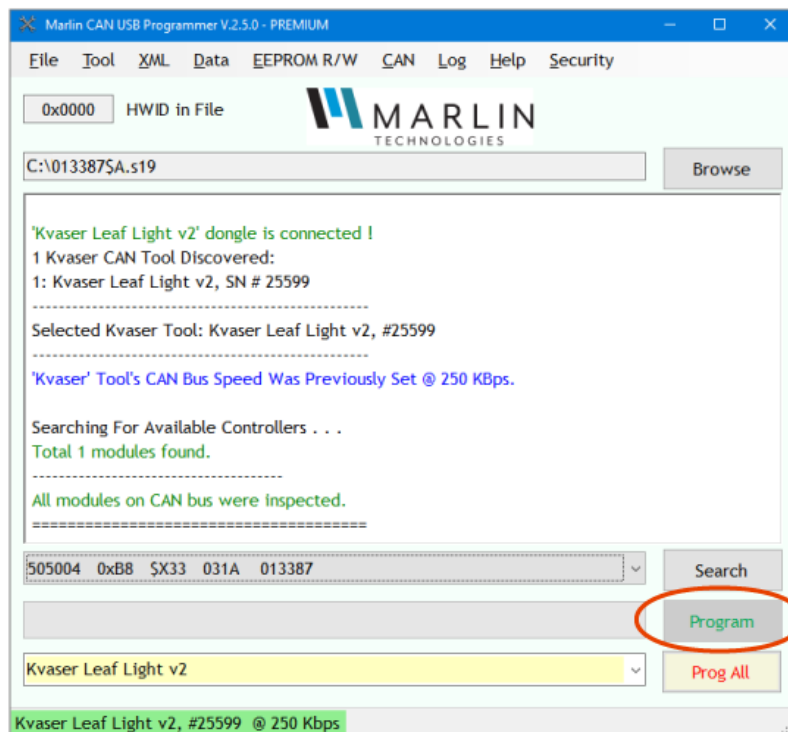


Figure 6

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 "XML\M-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.

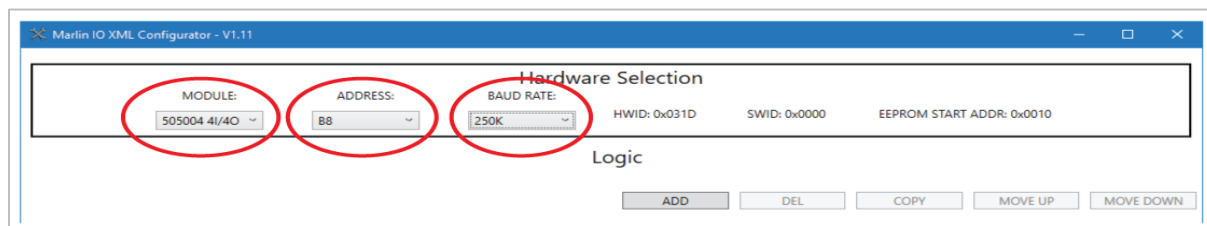


Figure 8

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.

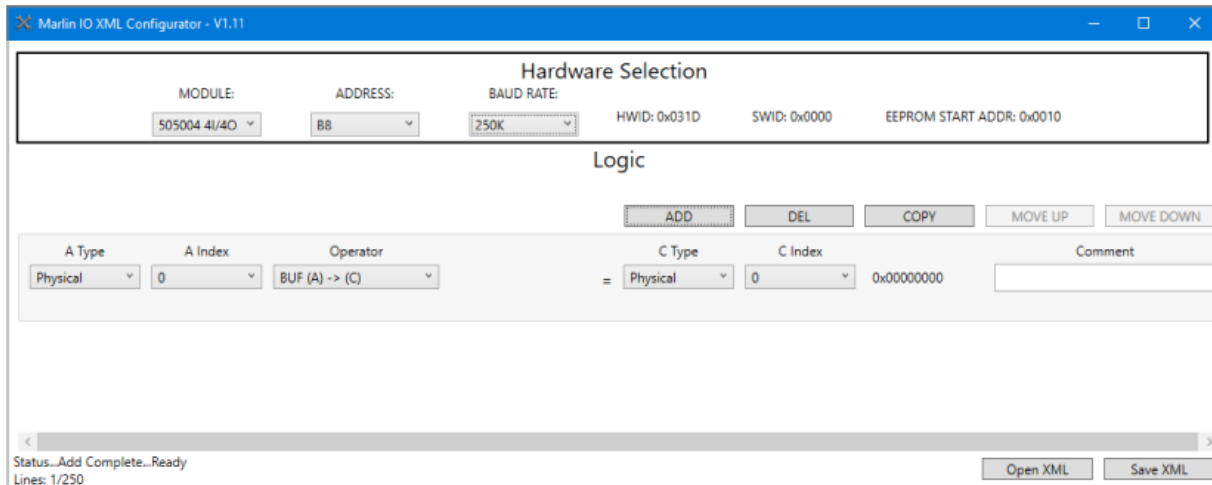


Figure 9

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.

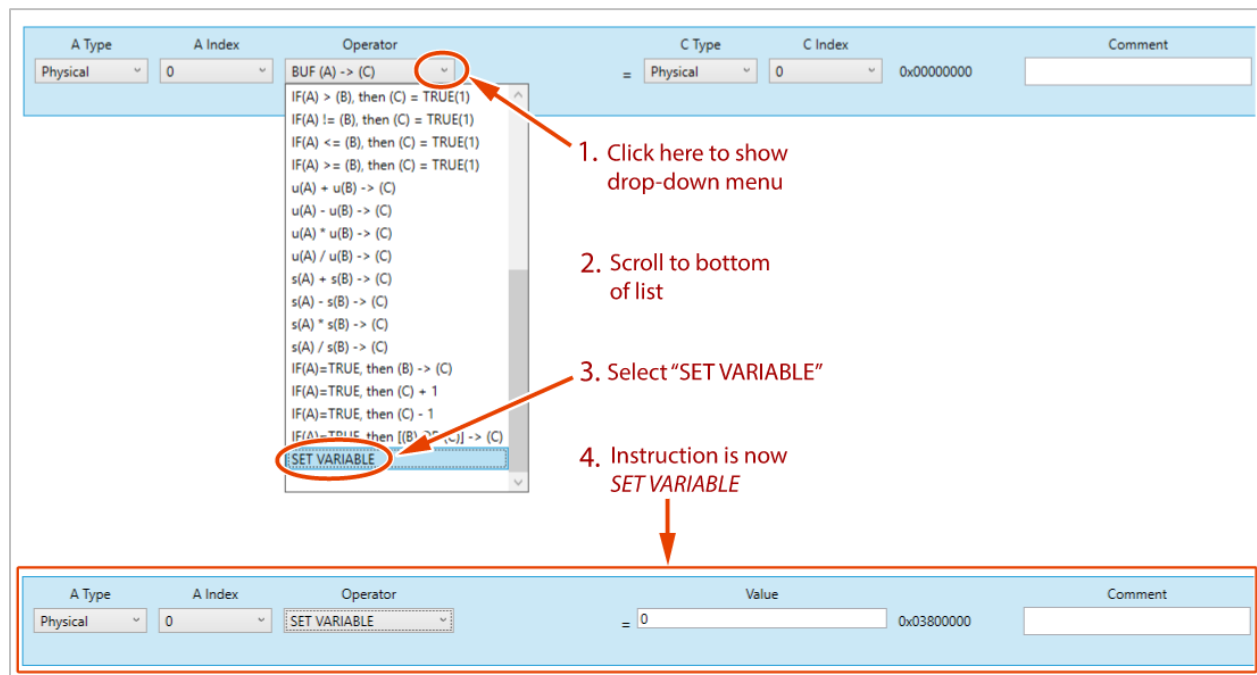


Figure 10

The instructions available in the Configurator consist of an operator that takes inputs, performs 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

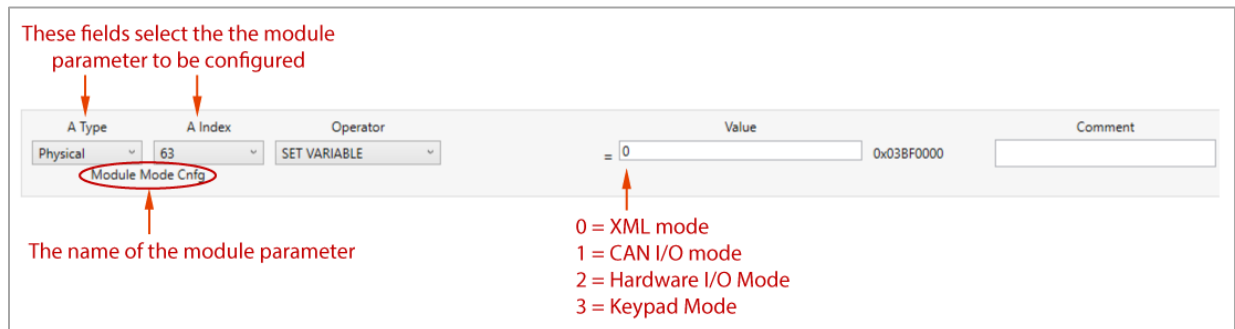


Figure 11

to "63":

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 [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

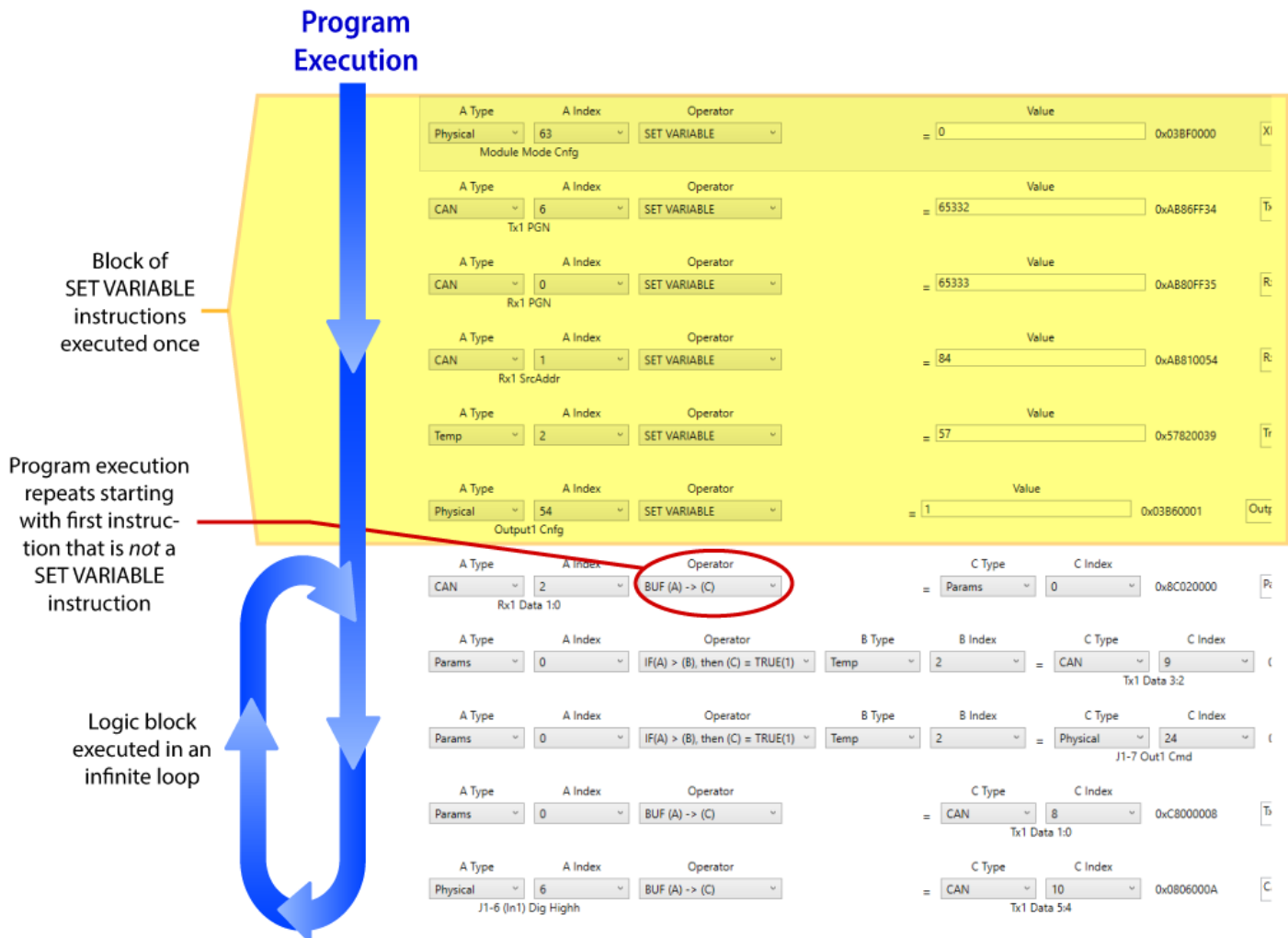


Figure 12

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 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.

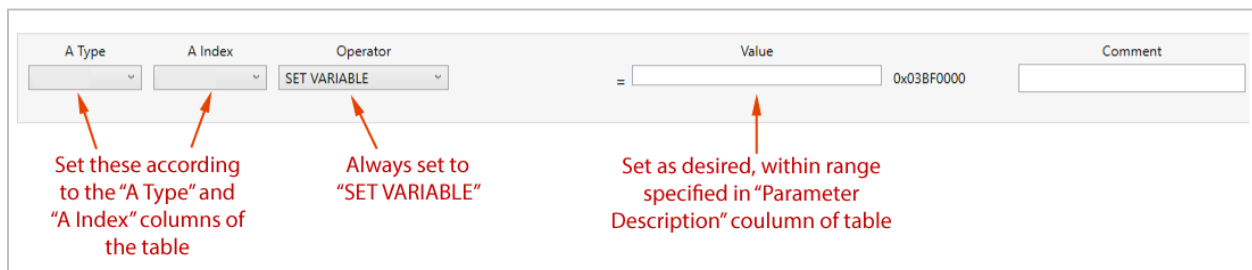


Figure 13

List of Configuration Parameters Used in XML Mode

A Type	A Index	Parameter Description
Physical	63	Module Mode. Must be 0 for XML Mode.
Physical	53	Set Base Frequency of all PWM Outputs [30-1000Hz, 1Hz/bit]
Physical	54	Output 1 Type ¹
Physical	55	Output 2 Type ¹
Physical	56	Output 3 Type ¹
Physical	57	Output 4 Type ¹
Physical	64	Input 1 Type ²
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]

A Type	A Index	Parameter Description
Physical	39	Output 4 Current Limit [1 mA/bit, 0-3000mA]
Physical	58	Output 1 Keypad Config
Physical	59	Output 2 Keypad Config
Physical	60	Output 3 Keypad Config
Physical	61	Output 4 Keypad Config
Physical	62	Keypad Source Address
CAN	63	Cmd Message Timeouts [10mS/bit, 0=NoTimeout]

- ¹
- 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)

- ²
- 0 = Digital/Analog/PWM [Default]
 - 1 = Current
 - 2 = Resistance
 - 3 = Quadrature X1
 - 4 = Quadrature X2
 - 5 = Quadrature X4

Configuration Example for XML Mode

Configurator Instruction				Result
Operator	A Type	A Index	Value	
SET VARIABLE	Physical	63	0	Module Mode = XML Mode
SET VARIABLE	Physical	54	1	Output 1 is a high side digital output
SET VARIABLE	Physical	55	2	Output 2 is a low side digital output
SET VARIABLE	Physical	56	4	Output 3 is a high side PWM output
SET VARIABLE	Physical	57	6	Output 4 is a low side PWM output
SET VARIABLE	Physical	53	100	Base frequency of all PWM outputs is 100 Hz

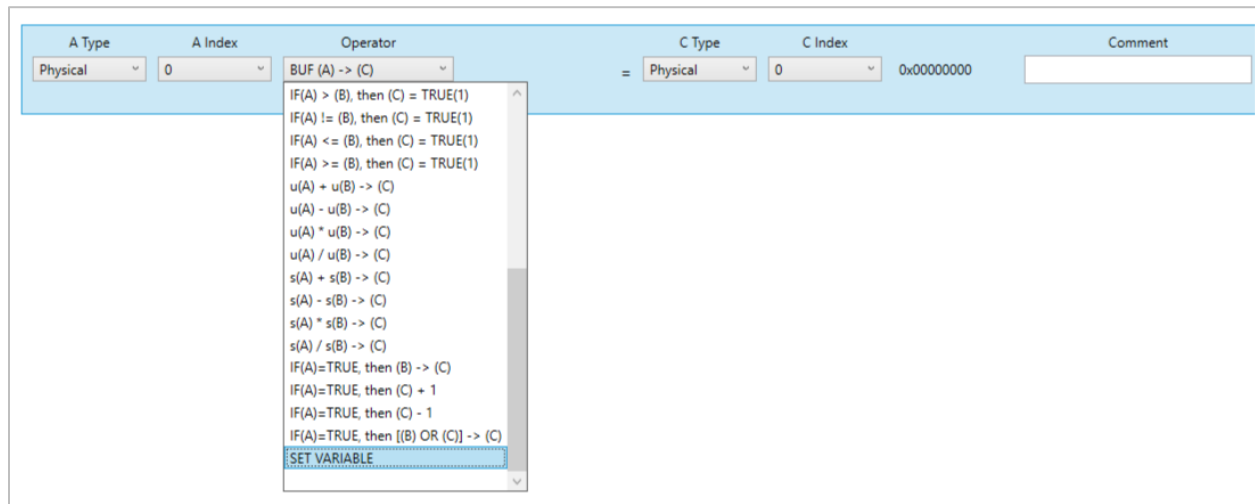
3.3.1.2 Logic

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

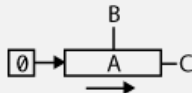
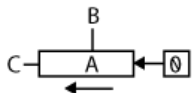
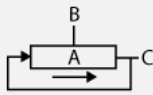
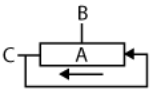
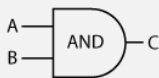
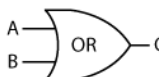

IF(A) > (B), then (C) = TRUE(1)
IF(A) != (B), then (C) = TRUE(1)
IF(A) <= (B), then (C) = TRUE(1)
IF(A) >= (B), then (C) = TRUE(1)
u(A) + u(B) -> (C)
u(A) - u(B) -> (C)
u(A) * u(B) -> (C)
u(A) / u(B) -> (C)
s(A) + s(B) -> (C)
s(A) - s(B) -> (C)
s(A) * s(B) -> (C)
s(A) / s(B) -> (C)
IF(A)=TRUE, then (B)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

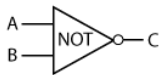
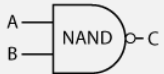

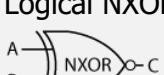
Figure 14

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

Name	Menu Item	Comment
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
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.

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

MATHEMATICAL OPERATORS (Signed Values)

Name	Menu Item	Comment
Add	$s(A) + s(B) \rightarrow s(C)$	Value A, B, and C must fit within a range of -32768 to 32767.
Subtract	$s(A) - s(B) \rightarrow s(C)$	Value A, B, and C must fit within a range of -32768 to 32767.
Multiply	$s(A) * s(B) \rightarrow s(C)$	Value A, B, and C must fit within a range of -32768 to 32767.
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	$IF(A)=TRUE, \text{ then } (B) \rightarrow (C)$	Otherwise C is left unchanged.
Conditional INCREMENT	$IF(A)=TRUE, \text{ the } (C) + 1$	Otherwise C is left unchanged.
Conditional DECREMENT	$IF(A)=TRUE, \text{ the } (C) - 1$	Otherwise C is left unchanged.
Conditional OR	$IF(A)=TRUE, \text{ the } [(B) \text{ OR } (C)] \rightarrow C$	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).

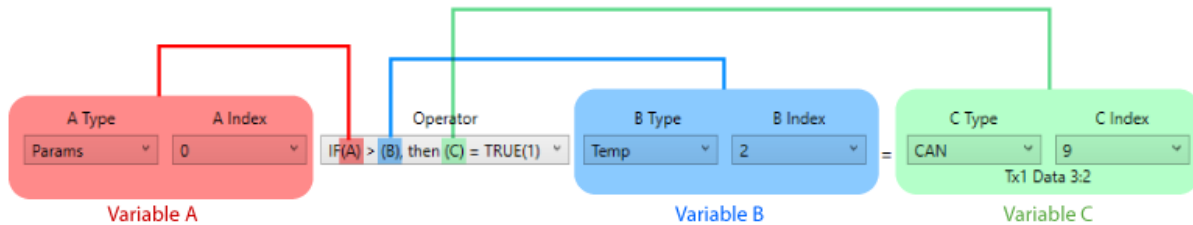


Figure 15

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 within the initial block of *SET VARIABLE* operations that precede the program's infinite loop. 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 are meant to be used inside the programs infinite loop to store calculated values, control states, etc.

The 505004 module does not distinguish between Temp and Params variables; they can be used any way the user sees fit. However, the intended use of the two variable types described above is suggested for the sake of organization.

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 Parameters

Index	Description	
0	DO NOT USE	
1	Input	Battery Voltage [0-43,554 mV, 1mV/bit]
2	Input	Input 1 Analog Voltage/Count [0-43,554 mV, 1mV/bit] / [0-65535 1cnt/bit, Unsigned]
3	Input	Input 2 Analog Voltage [0-43,554 mV, 1mV/bit] ¹
4	Input	Input 3 Analog Voltage [0-43,554 mV, 1mV/bit]
5	Input	Input 4 Analog Voltage [0-43,554 mV, 1mV/bit]
6	Input	Input 1 Digital High [1=Active High, 0=otherwise]
7	Input	Input 1 Digital Low [1=Active High, 0=otherwise]
8	Input	Input 2 Digital High [1=Active High, 0=otherwise]
9	Input	Input 2 Digital Low [1=Active High, 0=otherwise]
10	Input	Input 3 Digital High [1=Active High, 0=otherwise]
11	Input	Input 3 Digital Low [1=Active High, 0=otherwise]
12	Input	Input 4 Digital High [1=Active High, 0=otherwise]
13	Input	Input 4 Digital Low [1=Active High, 0=otherwise]
14	Input	Output 1 Current [0-4200mA, 1mA/bit]
15	Input	Output 2 Current [0-4200mA, 1mA/bit]
16	Input	Output 3 Current [0-4200mA, 1mA/bit]
17	Input	Output 4 Current [0-4200mA, 1mA/bit]
18	Input	Digital Input 1 Duty Cycle [0-100%, 0.1%/bit] ¹
19	Input	Digital Input 1 Period [0-65535uS, 1 uS/bit] ¹
20	Input	Digital Input 1 Frequency [32-10,000Hz, 0.1 Hz/bit] ¹
21	Input	Digital Input 2 Duty Cycle [0-100%, 0.1%/bit] ¹
22	Input	Digital Input 2 Period [0-65535uS, 1 uS/bit] ¹
23	Input	Digital Input 2 Frequency [32-10,000Hz, 0.1 Hz/bit] ¹
24	Output	Output 1 Digital [1=On, 0=Off]
25	Output	Output 2 Digital [1=On, 0=Off]
26	Output	Output 3 Digital [1=On, 0=Off]
27	Output	Output 4 Digital [1=On, 0=Off]
28	Output	Output 1 DutyCycle/Current [1 mA or 0.1% /bit]
29	Output	Output 2 DutyCycle/Current [1 mA or 0.1% /bit]
30	Output	Output 3 DutyCycle/Current [1 mA or 0.1% /bit]
31	Output	Output 4 DutyCycle/Current [1 mA or 0.1% /bit]

¹ These values are always 0 if input 1 is configured as a quadrature input.

Index #	CAN Parameters (XML Mode)		
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	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	Msg #3	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
63	Cnfg	TimeOut	Time Cmd Message Timeouts [10mS/bit, 0=NoTimeout]

(*) NOTE: 11-bit Std_ID: PGN is Message ID
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.

Temporary Variables

Index	Description	
0	FALSE	0 value
1	TRUE	1 value
2	Variables available to user to store values	
3		
.		
.		
.		
.		
61		
62		
63		

Params Variables

Index	Description
0	Variables available to user to store values
1	
2	
3	
.	
.	
.	
.	
61	
62	
63	

3.3.1.2.3 Example

Figure 16 shows an example of a simple vehicle application using logic symbols. Following that is the same implementation using Configurator instructions. The blue numbers in Figure 16 correlate to the blue numbers in the Configurator implementation, to show equivalent operations between the two implementations. The application checks if there are any E-Stop signals (emergency stop) from various sources and then, if necessary, activates an E-Stop indicator light, sends a CAN message commanding the TCU (traction control unit) to neutral, sends a CAN message commanding the ECU (engine control unit) to stop the engine, and kills the throttle signal.

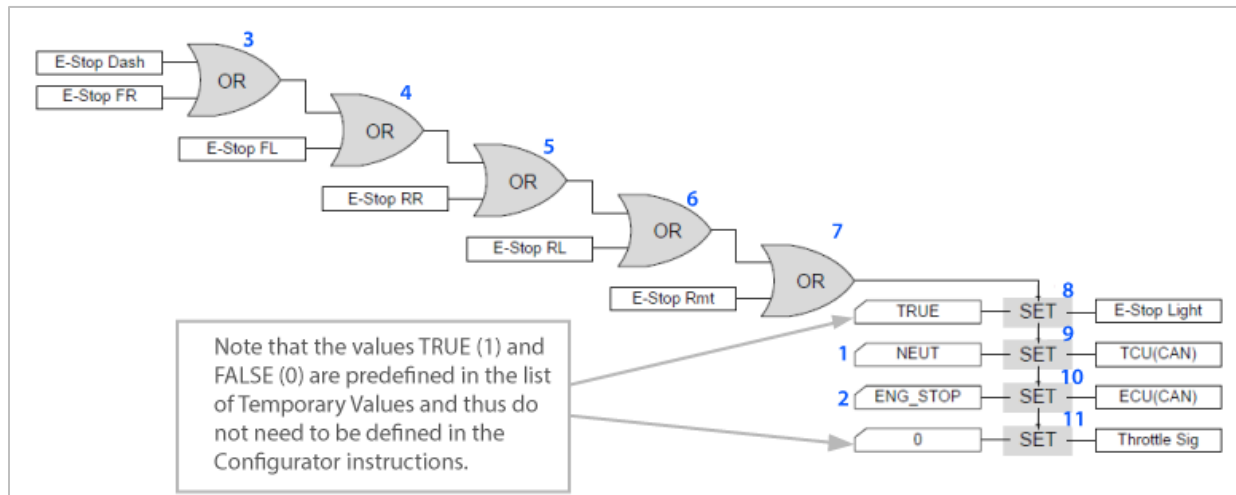


Figure 16

Configurator Implementation

	Variable A	Operator	Variable B		Variable C (or fixed value)	Comment
	Physical #63	SET VARIABLE		=	0	XML mode
1	Temp #2	SET VARIABLE		=	NEUT	Neutral value
2	Temp #3	SET VARIABLE		=	ENG_STOP	Engine stop
3	E-Stop Dash	Logical OR	E-Stop FR	=	PARAMS #1	Check e-stops.
4	PARAMS #1	Logical OR	E-Stop FL	=	PARAMS #1	
5	PARAMS #1	Logical OR	E-Stop RR	=	PARAMS #1	
6	PARAMS #1	Logical OR	E-Stop RL	=	PARAMS #1	
7	PARAMS #1	Logical OR	E-Stop Rmt	=	PARAMS #1	
8	PARAMS #1	IF(A)=TRUE, then (B) -> (C)	Temp #1 (TRUE/1)	=	E-Stop Light	Activate E-stop indicator light.
9	PARAMS #1	IF(A)=TRUE, then (B) -> (C)	Temp #2 (NEUT)	=	TCU(CAN)	Command TCU to neutral.
10	PARAMS #1	IF(A)=TRUE, then (B) -> (C)	Temp #3 (ENG_STOP)	=	ECU(CAN)	Command ECU to stop engine.
11	PARAMS #1	IF(A)=TRUE, then (B) -> (C)	Temp #0 (FALSE/0)	=	ThrottleSig	Kill throttle signal.

For the sake of simplicity, parameters in the orange cells are not shown to be referenced using a variable type and index number, nor is the configuration related to such parameters (CAN messages, output type, etc) shown in the configuration portion of the program. In an actual implementation, the E-Stop parameters may be read from bytes of a received CAN message. The E-Stop light may driven by a digital or PWM output. And so on.

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).

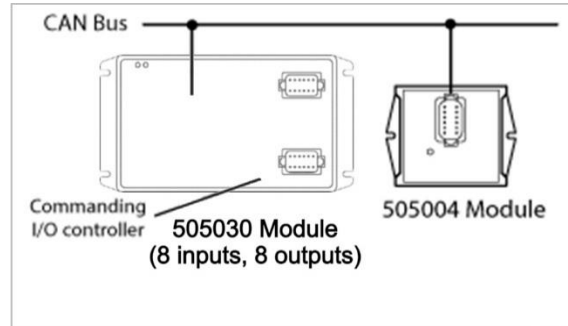


Figure 17

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]									

Received CAN messages:

	PGN	Data Bytes							
		LSB				MSB			
Cmd Msg: Outputs	18EFsa xx	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
- See the 13544S Marlin CAN-Bus I/O Module Protocol specification for further details.

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 18 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 18

List of CAN I/O Mode 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]
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 ¹
Physical	55	Output 2 Type ¹
Physical	56	Output 3 Type ¹
Physical	57	Output 4 Type ¹
Physical	64	Input 1 Type ²
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	0	Tx Rate of Msg FF40 [10mS/bit, 0=Off]
CAN	1	Tx Rate of Msg FF41 [10mS/bit, 0=Off]
CAN	2	Tx Rate of Msg FF42 [10mS/bit, 0=Off]
CAN	3	Tx Rate of Msg FF60 [10mS/bit, 0=Off]
CAN	4	Tx Rate of Msg FF61 [10mS/bit, 0=Off]
CAN	5	Tx Rate of Msg FF63 [10mS/bit, 0=Off]
CAN	6	Tx Rate of Msg FF64 [10mS/bit, 0=Off]
CAN	63	Cmd Message Timeouts [10mS/bit, 0=No Timeout]

- ¹ 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)

- ² 0 = Digital/Analog/PWM [Default]
1 = Current
2 = Resistance
3 = Quadrature X1
4 = Quadrature X2
5 = Quadrature X4

When finished adding instructions in the Configurator, click "Save XML" in the bottom-right corner of the window (as seen in Figure 9) 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*.

3.3.2.3 CAN I/O Mode Configuration Example

Configuration Example for CAN I/O Mode

Configurator Instruction				Result
Operator	A Type	A Index	Value	
SET VARIABLE	Physical	63	1	Module Mode = CAN I/O Mode
SET VARIABLE	Physical	54	1	Output 1 is a high side digital output
SET VARIABLE	Physical	55	2	Output 2 is a low side digital output
SET VARIABLE	Physical	56	4	Output 3 is a high side PWM output
SET VARIABLE	Physical	57	6	Output 4 is a low side PWM output
SET VARIABLE	Physical	53	100	Base frequency of all PWM outputs is 100 Hz

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 19), 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*

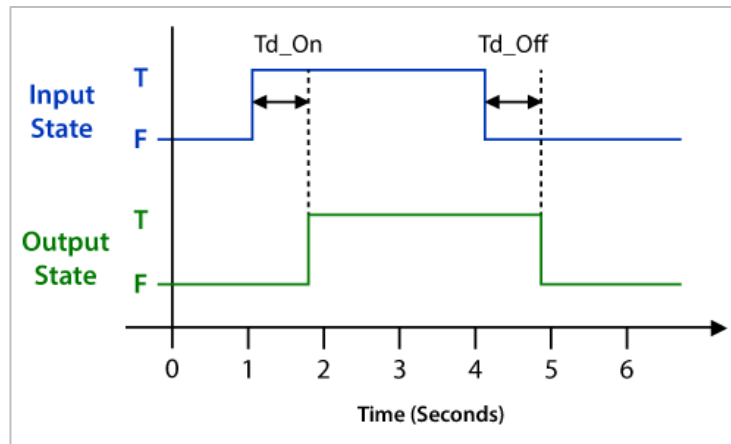
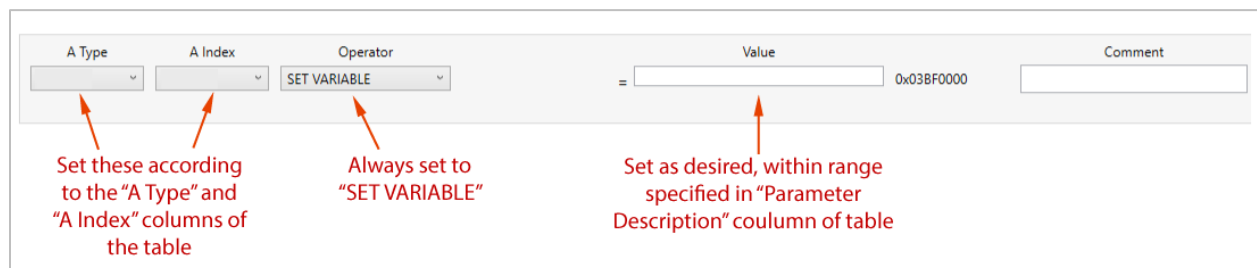


Figure 19

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 19), the output state then changes to FALSE.

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 20 as a guide for filling out the *SET VARIABLE* instructions. If



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 20

necessary, see section *3.3 Configuring the Module* for more details about adding *SET VARIABLE* instructions.

List of Hardware I/O Mode 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 ¹
Physical	55	Output 2 Type ¹
Physical	56	Output 3 Type ¹
Physical	57	Output 4 Type ¹
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] ²
Physical	29	Output 2 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
Physical	30	Output 3 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
Physical	31	Output 4 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
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]

- ¹ 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)

- ² 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.

When finished adding instructions in the Configurator, click "Save XML" in the bottom-right corner of the window (as seen in Figure 9) 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*.

3.3.3.3 Hardware I/O Mode Configuration Example

Configuration Example for Hardware I/O Mode

Configurator Instruction				Result
Operator	A Type	A Index	Value	
SET VARIABLE	Physical	63	2	Module Mode = Hardware I/O Mode
SET VARIABLE	Physical	54	1	Output 1 is a high side digital output
SET VARIABLE	Physical	55	2	Output 2 is a low side digital output
SET VARIABLE	Physical	56	4	Output 3 is a high side PWM output
SET VARIABLE	Physical	57	6	Output 4 is a low side PWM output
SET VARIABLE	Physical	53	100	Base frequency of all PWM outputs is 100 Hz
SET VARIABLE	Physical	30	1000	Output 3 Duty Cycle is 100% (when output state is TRUE)
SET VARIABLE	Physical	31	1000	Output 4 Duty Cycle is 100% (when output state is TRUE)
SET VARIABLE	Physical	45	100	Output 1 On-Delay is 1.0 seconds.
SET VARIABLE	Physical	49	150	Output 1 Of-Delay is 1.5 seconds.
SET VARIABLE	Physical	46	200	Output 2 On-Delay is 2.0 seconds.
SET VARIABLE	Physical	50	250	Output 2 Of-Delay is 2.5 seconds.
SET VARIABLE	Physical	47	300	Output 3 On-Delay is 3.0 seconds.
SET VARIABLE	Physical	51	350	Output 3 Of-Delay is 3.5 seconds.
SET VARIABLE	Physical	48	400	Output 4 On-Delay is 4.0 seconds.
SET VARIABLE	Physical	52	450	Output 4 Of-Delay is 4.5 seconds.

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:

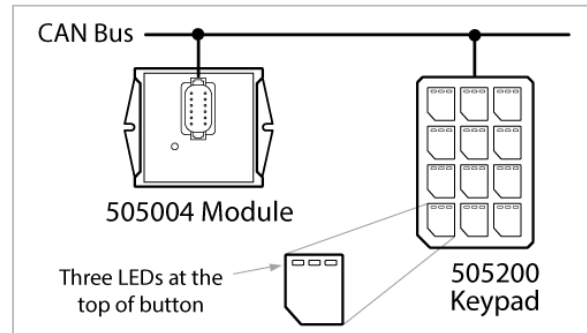


Figure 21

- **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 22.

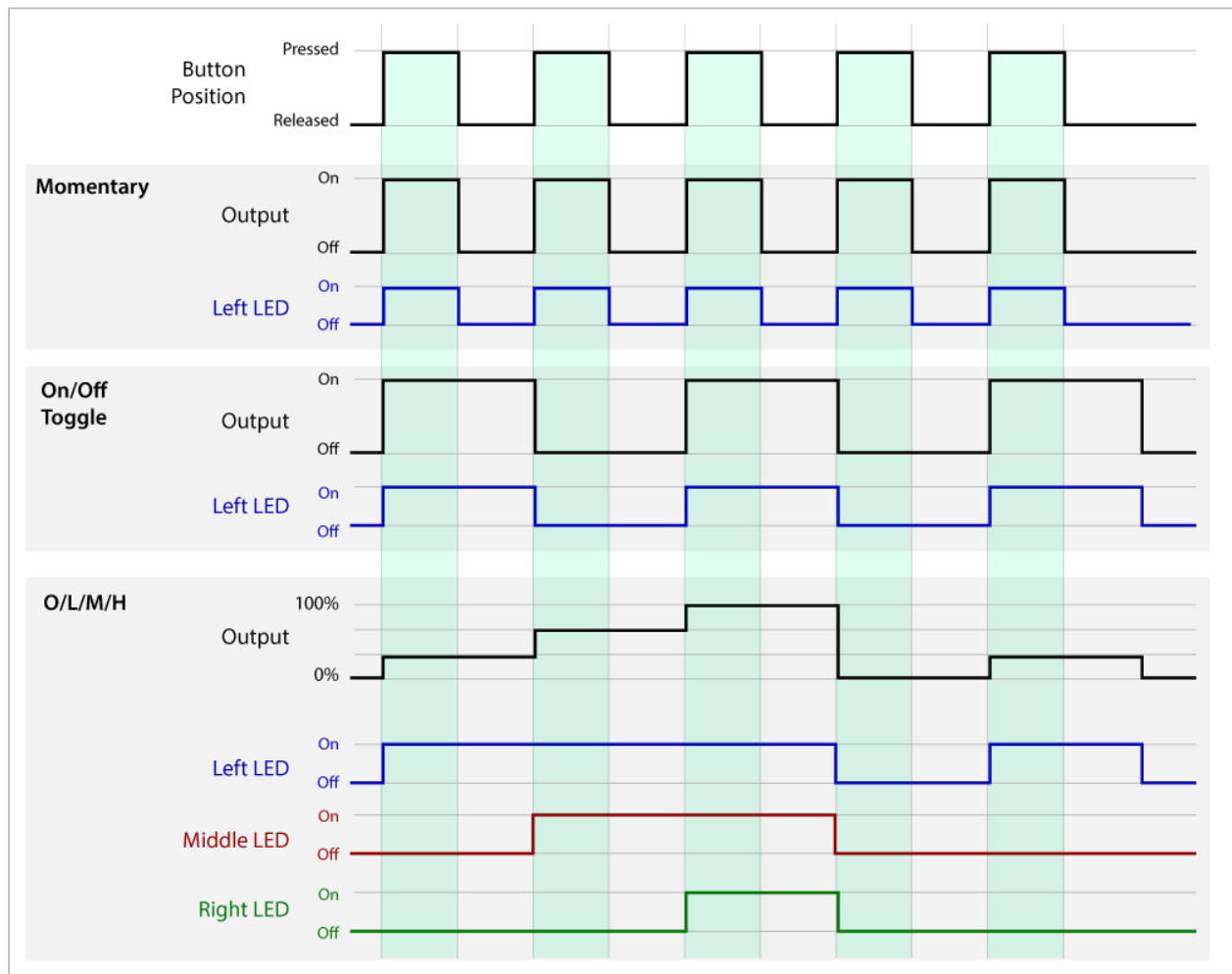


Figure 22

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.

Transmitted CAN messages:

		Data Bytes							
		LSB				MSB			
Indicator LED Msg	18A7kksa	LED1-4	LED5-8	LED9-12	LED13-16	LED17-20	LED21-24	LED25-28	LED29-32
		2-bit flags [00=Off, 01=On, 10=Blink 2Hz, 11=No Change]							
Indicator LED Msg	18A6kksa	LED33-36	LED37-40	LED41-44	LED45-48	LED49-52	LED53-56	LED57-60	LED61-64
		2-bit flags [00=Off, 01=On, 10=Blink 2Hz, 11=No Change]							

Received CAN messages:

		Data Bytes							
		LSB				MSB			
Button Status Msg	18FF40kk	Btn1-4	Btn5-8	Btn9-12	Btn13-16	Btn17-20	Btn21-24	Btn25-28	Btn29-32
		2-bit Status Flags[00=False, 01=True, 10=Error, 11=Not Applicable]							

Notes:

- Indicator LED Status Messages are sent every 100mS.
- sa = Source Address of the 505004 Module
- kk = Source Address of the Keypad Module
- See the 11713S specification for Marlin Keypad Modules

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. Refer to Figure 23 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.

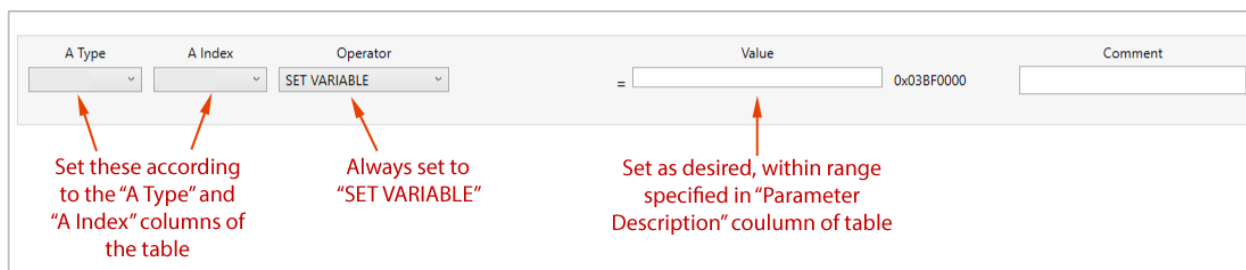


Figure 23

List of Keypad Mode 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 ¹
Physical	55	Output 2 Type ¹
Physical	56	Output 3 Type ¹

A Type	A Index	Parameter Description
Physical	57	Output 4 Type ¹
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] ²
Physical	29	Output 2 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
Physical	30	Output 3 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
Physical	31	Output 4 Duty Cycle/Current [0.1%/bit or 1 mA/bit] ²
Physical	58	Keypad Button Index and Button Type for Output 1 ³
Physical	59	Keypad Button Index and Button Type for Output 2 ³
Physical	60	Keypad Button Index and Button Type for Output 3 ³
Physical	61	Keypad Button Index and Button Type for Output 4 ³
Physical	62	Keypad Source Address
CAN	63	Cmd Message Timeouts [10mS/bit, 0=NoTimeout]

- ¹
- 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)

- ²
- 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.

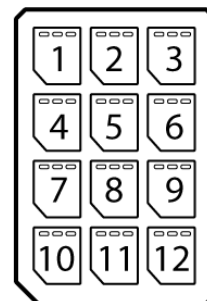
- 3 This parameter a) identifies which button on the keypad drives the given output (i.e. its Index Value), and b) specifies the button's functionality (i.e. its Type). The parameter's value is in the following 16-bit format:

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)



When finished adding instructions in the Configurator, click "Save XML" in the bottom-right corner of the window (as seen in Figure 9) 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*.

3.3.4.3 Keypad I/O Mode Configuration example

Configuration Example for Keypad Mode

Configurator Instruction				Result
Operator	A Type	A Index	Value	
SET VARIABLE	Physical	63	3	Module Mode = Keypad Mode
SET VARIABLE	Physical	54	1	Output 1 is a high side digital output
SET VARIABLE	Physical	55	2	Output 2 is a low side digital output
SET VARIABLE	Physical	56	4	Output 3 is a high side PWM output
SET VARIABLE	Physical	57	6	Output 4 is a low side current output
SET VARIABLE	Physical	53	100	Base frequency of all PWM outputs is 100 Hz
SET VARIABLE	Physical	30	1000	Output 3 Duty Cycle is 100% (when output is On)
SET VARIABLE	Physical	31	1000	Output 4 Duty Cycle is 100% (when output is On)
SET VARIABLE	Physical	62	64	Keypad source address is 64 (or 0x40)

Configurator Instruction				Result
Operator	A Type	A Index	Value	
SET VARIABLE	Physical	58	3	Keypad Button 3 drives Output 1 on 505004 module
SET VARIABLE	Physical	59	1	Keypad Button 1 drives Output 2 on 505004 module
SET VARIABLE	Physical	60	2	Keypad Button 2 drives Output 3 on 505004 module
SET VARIABLE	Physical	61	4	Keypad Button 4 drives Output 4 on 505004 module

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 24.



Figure 24

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 25. 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 24, 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 25). If there are

errors, consult the Programming Tool user guide mentioned in section 2. *Software Installation* to troubleshoot.

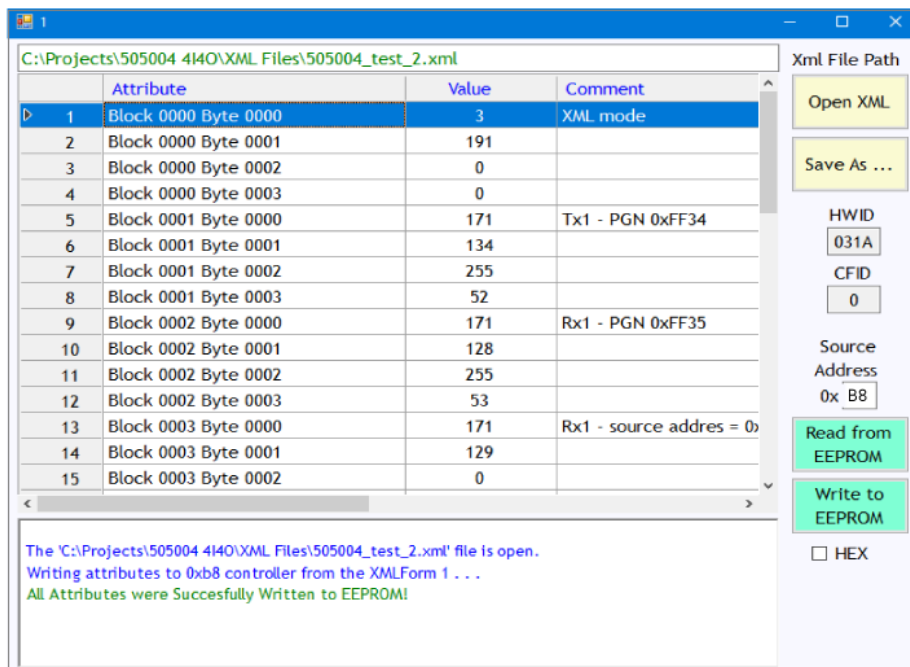


Figure 25

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. No configuration is necessary to enable the LED module to work with the 505004 module.

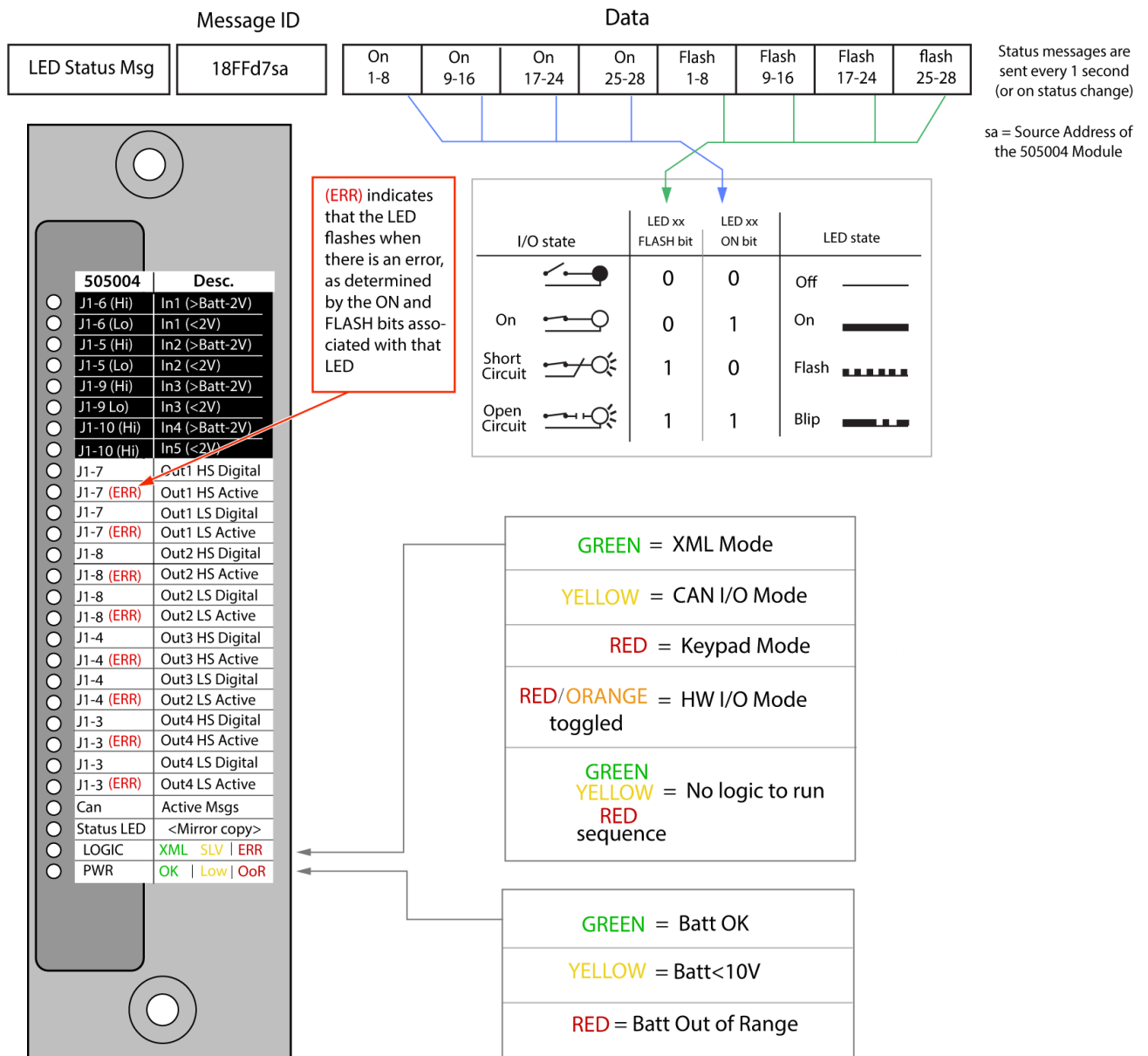


Figure 26