SPARK MAX USER'S MANUAL

TABLE OF CONTENTS

1 - SPARK MAX OVERVIEW

The REV Robotics SPARK MAX Motor Controller is an all-in-one USB, CAN, and PWM enabled motor controller that can drive both 12 V brushed and 12 V brushless DC motors. SPARK MAX is designed for use in the FIRST Robotics Competition (FRC), incorporating advanced motor control in a small, easy-to-use, and affordable package. Configure and run the SPARK MAX through its built-in USB interface without needing a full control system.

1.1 - FEATURE SUMMARY

- Brushed and sensored-brushless motor control
- PWM, CAN, and USB control interfaces
  - PWM/CAN - Locking and keyed 4-pin JST-PH
  - USB - USB type C
- USB configuration and control
  - Rapid configuration with a PC
- Smart control modes
  - Closed-loop velocity control
  - Closed-loop position control
  - Follower mode
- Encoder port
  - Locking and keyed 6-pin JST-PH
  - 3-phase hall-sensor encoder input
  - Motor temperature sensor input
- Data port
  - Limit switch input
  - Quadrature encoder input with index
  - Multi-function pin
- Mode button
  - On-board motor type and idle behavior configuration
- RGB status LED
  - Detailed mode and operation feedback
- Integrated power and motor wires
  - 12 AWG ultra-flexible silicone wire
- Passive cooling

- 1 - PWM/CAN cable retention clip
- 1 - Data port protection cap

1.3 - SPECIFICATIONS

The following tables provide the operating and mechanical specifications for the SPARK MAX motor controller.

CAUTION

DO NOT exceed the maximum electrical specifications. Doing so will cause permanent damage to the SPARK MAX and will void the warranty.

Table 1.1 - Main Electrical Specifications

Parameter	Min	Typ	Max	Units
Operating voltage range (V_{IN})	5.5	12	24	V
Absolute maximum supply voltage	-	-	30	V
Continuous output current	-	-	60[a]	A
Maximum output current (2 second surge)	-	-	100	A
Output frequency	-	20	-	kHz

a. Continuous operation at 60A may produce high temperatures on the heat sink. Caution should be taken when handling the SPARK MAX if it has been running at higher current level for an extended period of time.

Table 1.2 - PWM Input Specifications

Parameter	Min	Typ	Max	Units
Full-reverse input pulse[a]	-	1000	-	Î¼s
Neutral input pulse[b]	-	1500	-	Î¼s
Full-forward input pulse[c]	-	2000	-	Î¼s
Valid input pulse range	500	-	2500	Î¼s
Input frequency	50	-	200	Hz
Input timeout[d]	-	50	-	ms
Default Input deadband[e]	-	5	-	%

a. Brushed: $-V_{IN}$ between A and B outputs at 100% duty.
Brushless: A->B->C direction at 100% duty.
b. Neutral corresponds to zero output voltage (0 V) and is either braking or coasting depending on the current idle behavior mode.
c. Brushed: $+V_{IN}$ between A and B outputs at 100% duty.
Brushless: C->B->A direction at 100% duty.
d. If a valid pulse isn't received within the timeout period, the SPARK MAX will disable its output.
e. Input deadband is added to each side of the neutral pulse width. Within the deadband,

Digital input-low voltage[a]	-	-	1.36	V
Analog input voltage range[b]	0	-	3.3	V
5V supply current (I_{5V})[c]	-	-	100	mA
3.3V supply current ($I_{3.3V}$)	-	-	30	mA
Total supply current (I_{5V} + $I_{3.3V}$)	-	-	100	mA

a. See section 2.4 - Data Port for more details on the digital pins on the Data Port.
b. See section 2.4 - Data Port for more details on the analog pin on the Data Port.
c. The 5V supply is shared between the Data Port and Encoder Port.

Table 1.4 - Encoder Port Specifications

Parameter	Min	Typ	Max	Units
Digital input voltage range[a]	0	-	5	V
Digital input-high voltage[a]	1.85	-	-	V
Digital input-low voltage[a]	-	-	1.36	V
Analog input voltage range[b]	0	-	3.3	V
5V supply current (I_{5V})[c]	-	-	100	mA
3.3V supply current ($I_{3.3V}$)	-	-	30	mA
Total supply current (I_{5V} + $I_{3.3V}$)	-	-	100	mA

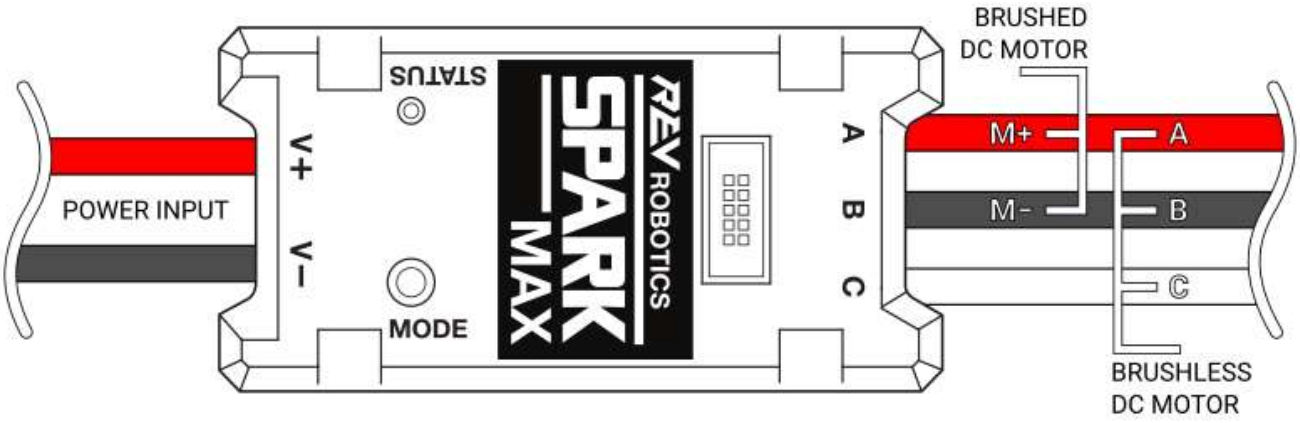
a. See section 2.3 - Encoder Port for more details on the digital pins on the Data Port.
b. See section 2.3 - Encoder Port for more details on the analog pin on the Data Port.
c. The 5V supply is shared between the Data Port and Encoder Port.

Table 1.5 - Mechanical Specifications

Parameter	Min	Typ	Max	Units
Body length	-	70	-	mm
Body width	-	35	-	mm
Body height	-	25.5	-	mm
Weight	-	113.3	-	g
Power and motor wire gauge	-	12	-	AWG
Power and motor wire length	-	15	-	cm

2 - FEATURE DESCRIPTION

12AWG ultra-flexible silicone-coated wire. Each wire runs approximately 15cm from the end faces of the controller. Be sure to take care when cutting and stripping the wires as not to cut them too short. The figure below shows these connections in detail.



CAUTION

As with any electrical component, make all connections with power turned off. Connecting the SPARK MAX to a powered system may result in unexpected behavior and may pose a safety risk.

2.1.1 - MOTOR OUTPUT

Motor output wires are labeled as A, B, and C with red, black, and white wires. Brushed motors must be connected to the A and B wires, while brushless motors must be connected to all three. **It is critical that the order of the brushless motor wires match the SPARK MAX or the motor will not spin and could be damaged.** Additional details are below in Table 2.1.

Table 2.1 - Motor Connections

Motor Type	Motor Wires	SPARK MAX Wires
Brushed	Red / M+	Red / A
	Black / M-	Black / B
Brushless (NEO Brushless Motor)	Red / A	Red / A
	Black / B	Black / B
	White / C	White / C

SPARK MAX cannot detect which motor type it is connected to. Be sure to configure the SPARK MAX to run the type of motor you have connected. See the Motor Type - Brushed/Brushless Mode section for more details

DO NOT reverse V+ and V- or swap motor and power connections. Doing so will cause permanent damage to the SPARK MAX and will void the warranty.

CAUTION

DO NOT exceed the maximum supply voltage of 30V. Doing so will cause permanent damage to the SPARK MAX and will void the warranty.

When using high current motors, it is recommended to use a power source that is capable of handling large surge currents, e.g. a 12V lead-acid battery. If the supply voltage drops below 5.5V the SPARK MAX will brown out, resulting in unexpected behavior. It is also highly recommended to incorporate a fuse or circuit-breaker in series with the SPARK MAX between it an the power source to prevent exceeding the maximum current rating.

CAUTION

DO NOT exceed the maximum current ratings of 60A or 100A for 2 seconds. Doing so will cause permanent damage to the SPARK MAX and will void the warranty.

2.2 - CONTROL CONNECTIONS

The SPARK MAX can be controlled by three different interfaces, servo-style PWM, controller area network (CAN), and USB. The following sections describe the physical connections to these interfaces in detail. For details on the operation and protocols of the PWM, CAN, and USB interfaces, please see the Section 3.3 - Control Interfaces.

2.2.1 - CAN/PWM PORT

The CAN/PWM Port is located on the power input side of the SPARK MAX. This port can be connected to either a servo-style PWM signal or a CAN bus with other devices. Connector details can be found below.

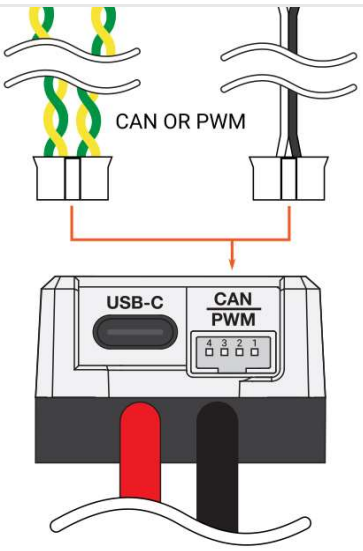
Table 2.2 - CAN/PWM Port Connector Information

Connector Pin	CAN Function	PWM Function
1	CAN High	Signal
2	CAN Low	Ground
3	CAN High	Signal
4	CAN low	Ground

Mating Connector Information				
Description	**Manufacturer**	**Part Number**	**Vendor**	**Vendor P/N**
JST-PH 4-pin Housing	JST	PHR-4	DigiKey	455-1164-ND
JST-PH Contact	JST	SPH-002T-P0.5L	DigiKey	455-2148-1-ND

2.2.2 - USB-C PORT

The USB-C Port is located on the power input side of the SPARK MAX. It supports USB 2.0 and 5V power for the SPARK MAX's internal microcontroller. While you can configure the SPARK MAX without main power, you will not be able to spin a motor.



2.3 - ENCODER PORT

Located on the motor output side of the SPARK MAX is a 6-pin Encoder Port. This port is designed to accept the built-in hall-encoder from the NEO Brushless Motor, but it can also connect to other external encoders when running in Brushed Mode. The connector details can be found below.

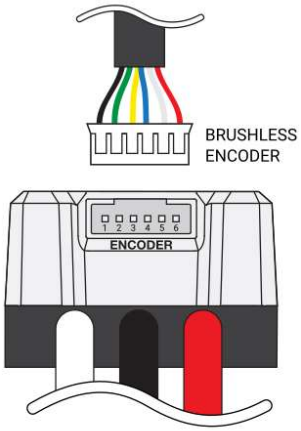


Table 2.3 - Encoder Port Connector Information

Connector Pin	Pin Type	Pin Function
1	Power	Ground
2	Digital	Encoder C / Index
3	Digital	Encoder B
4	Digital	Encoder A
5	Analog	Motor Temperature
6	Power	+5V

Mating Connector Information				
Description	**Manufacturer**	**Part Number**	**Vendor**	**Vendor P/N**
JST-PH 6-pin Housing	JST	PHR-6	DigiKey	455-1162-ND
JST-PH Contact	JST	SPH-002T-P0.5L	DigiKey	455-2148-1-ND
Recommended Crimping Tool	IWISS	SN-2549	Amazon	SN-2549

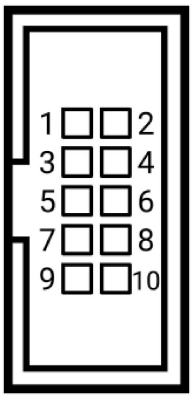


Table 2.4 - Data Port Connector Information

Connector Pin	Pin Type	Pin Function
1	Power	+3.3V
2	Power	+5V
3	Analog	Analog Input
4	Digital	Forward Limit Switch Input
5	Digital	Encoder B
6	Digital	Multi-function Pin
7	Digital	Encoder A
8	Digital	Reverse Limit Switch Input
9	Digital	Encoder C / Index
10	Ground	Ground

2.4.1 - LIMIT SWITCH INPUTS

SPARK MAX has two limit switch inputs that, when triggered, can independently prevent motion in both the forward and reverse directions. By default, when the pin for the corresponding direction is grounded, SPARK MAX will override any input commands for that direction and force the output into the neutral state. Input commands for the opposite direction will still be processed unless the corresponding limit signal is also triggered.

The default polarity is compatible with Normally Open (NO) style limit switches, who's contacts are shorted together when the switch is pressed. The Limit Switch Inputs can be configured for the opposite polarity using the USB or CAN interfaces. When configured for the opposite polarity, Normally Closed (NC), the limit will be triggered when the pin is left disconnected from ground. In other words, connecting the pin to ground will release the limit. The following table shows these configurations in detail:

Table 2.5 - Limit Switch Operation

Normally Closed	Normally Open (default)	Pressed	Normal operation	
	Normally Closed	Released	Normal operation	*
		Pressed	Triggered	

REV Robotics will be releasing a selection of breakout boards and data cables that are compatible with the SPARK MAX Data Port. Stay tuned for updates.

2.4.2 - QUADRATURE ENCODER INPUT

The Quadrature Encoder Input on the Data Port is compatible with standard quadrature encoder signals, usually labeled as channel A, channel B, and Index. SPARK MAX shares these signals with the Encoder Port on the output side of the controller, therefore the Index signal is shared with the third brushless encoder signal C. When in Brushless Mode, these Data Port pins cannot be used with an external encoder. See Alternate Encoder Mode for information on how to configure the SPARK MAX to accept an alternative encoder source when running in Brushless Mode.

When in Brushed Mode, an external encoder can be connected through either the Data Port or the Encoder Port.

The SPARK MAX encoder signals are not pulled high internally. This is to ensure the maximum compatibility with different types of encoders.

2.4.3 - ANALOG INPUT

The Analog Port on the SPARK MAX can measure voltages up to 3.3V with 12-bit resolution. The SPARK MAX Data Port Breakout includes a 5V to 3.3V amplifier circuit so that 5V signals can be sensed with the Analog Input pin.

Analog input is supported on firmware versions 1.4.0 and newer.

2.4.4 - MULTI-FUNCTION PIN

This pin is reconfigured when the SPARK MAX is configured in Alternate Encoder Mode.

2.4.5 - POWER RAILS

The SPARK MAX Data Port can provide both 3.3V and 5V power to connected devices. Please check Table 1.3 - Data Port Specifications for details on the supply current capabilities of both rails.

3.1 - MOTOR TYPE - BRUSHED/BRUSHLESS MODE

Brushed and brushless DC motors require different motor control schemes based on the differences in their technology. It is possible to damage the SPARK MAX, the motor, or both if the appropriate motor type isn't configured properly. At the moment, the NEO Brushless Motor and the NEO 550 Brushless Motor are the only FRC-legal brushless motors compatible with the SPARK MAX, so choosing the correct operating mode should be straightforward.

Brushed or brushless motor types can be configured using the Mode Button, CAN, and USB interfaces.

3.1.1 - CONFIGURATION WITH MODE BUTTON

Follow the steps below to switch motor types with the Mode Button. It is recommended that the motor be left disconnected until the correct mode is selected.

1. Connect the SPARK MAX to main power, not just USB Power.
2. The Status LED will indicate which motor type is configured by blinking yellow or blue for Brushed Mode, or blinking magenta or cyan for Brushless Mode.

   Please see the Status LED Colors and Patterns in the SPARK MAX Quick Start Guide for more details on the Brushed and Brushless Mode colors.

3. Press and hold the Mode Button for approximately 3 seconds.

   Use a small screwdriver, straightened paper clip, pen, or other small implement to press the button. Do not use any type of pencil as the pencil lead can break off inside the SPARK MAX.

4. After the button has been held for enough time, the Status LED will change and indicate the different motor type.
5. Release the mode button.

3.1.2 - CONFIGURATION WITH USB

Follow the steps below to switch motor types with the USB and the SPARK MAX Client application. Be sure to download and install the SPARK MAX Client application from the SPARK MAX Software Resources page before continuing.

1. Connect the SPARK MAX to your computer using a USB-C cable.
2. Open the SPARK MAX Client application and verify that the application is connected to your SPARK MAX.
3. On the **Basic** tab, select the appropriate motor type under the **Select Motor Type** menu.

3.2 - IDLE MODE - BRAKE/COAST MODE

When the SPARK MAX is receiving a neutral command the idle behavior of the motor can be handled in two different ways: Braking or Coasting.

When in Brake Mode, the SPARK MAX will effectively short all motor wires together. This quickly dissipates any electrical energy within the motor and brings it to a quick stop.

When in Coast Mode, the Spark MAX will effectively disconnect all motor wires. This allows the motor to spin down at its own rate.

The Idle Mode can be configured using the Mode Button, CAN, and USB interfaces.

3.2.1 - CONFIGURATION WITH MODE BUTTON

Follow the steps below to switch the Idle Mode between Brake and Coast with the Mode Button.

1. Connect the SPARK MAX to main power, not just USB Power.
2. The Status LED will indicate which Idle Mode is currently configured by blinking blue or cyan for Brake and yellow or magenta for Coast depending on the motor type.

   Please see the Status LED Colors and Patterns in the SPARK MAX Quick Start Guide for more details on the Brushed and Brushless Mode colors.

3. Press and release the Mode Button

   Use a small screwdriver, straightened paper clip, pen, or other small implement to press the button. Do not use any type of pencil as the pencil lead can break off inside the SPARK MAX.

4. You should see the Status LED change to indicate the selected Idle Mode.

3.2.2 - CONFIGURATION WITH USB

Follow the steps below to switch the Idle Mode between Brake and Coast with the USB and the SPARK MAX Client application. Be sure to download and install the SPARK MAX Client application from the SPARK MAX Software Resources page before continuing.

1. Connect the SPARK MAX to your computer using a USB-C cable.
2. Open the SPARK MAX Client application and verify that the application is connected to your SPARK MAX.
3. On the **Basic** tab, select the desired mode with the **Idle Mode** switch.
4. Click **Update Configuration** and confirm the change.

3.3 - CONTROL INTERFACES

The SPARK MAX can be controlled by three different interfaces, servo-style PWM, controller area network (CAN), and USB. The following sections describe the operation and protocols of these interfaces. For more details on the physical connections, see Section 2.2 - Control Connections.

3.3.1 - PWM INTERFACE

The SPARK MAX can accept a standard servo-style PWM signal as a control for the output duty cycle. Even though the PWM port is shared with the CAN port, SPARK MAX will automatically detect the incoming signal type and respond accordingly. For details on how to connect a PWM cable to the SPARK MAX, see Section 2.2.1 - CAN/PWM Port.

The SPARK MAX responds to a factory default pulse range of 1000Âµs to 2000Âµs. These pulses correspond to full-reverse and full-forward rotation, respectively, with 1500Âµs (Â±5% default input deadband) as the neutral position, i.e. no rotation. The input deadband is configurable with the SPARK MAX Client Application or the CAN interface. The table below describes how the default pulse range maps to the output behavior.

Table 3.1 - PWM Pulse Mapping

	Output duty cycle and direction (default deadband)				
	Full Reverse	**Proportional Reverse**	**Neutral**	**Proportional Forward**	**Full Forward**
Output duty cycle, D (%)	D = 100	100 < D < 0	D = 0	0 < D < 100	D = 100
Input pulse width, p (Âµs)	p â‰¤ 1000	1000 < p < 1475	1475 â‰¤ p â‰¤ 1525	1525 < p < 2000	2000 â‰¤ p
Maximum pulse range, p (Âµs)	500 â‰¤ p â‰¤ 2500				
Valid pulse frequency, f (Hz)	50 â‰¤ f â‰¤ 200				

If a valid signal isn't received within a 60ms window, the SPARK MAX will disable the motor output and either brake or coast the motor depending on the configured Idle Mode. For details on the Idle Mode, see Section 3.2 - Idle Mode - Brake/Coast Mode.

3.3.2 - CAN INTERFACE

The SPARK MAX can be connected to a robot CAN network. CAN is a bi-directional communications bus that enables advanced features within the SPARK MAX. SPARK MAX must be connected to a CAN network that has the appropriate termination resistors at both endpoints. Please see the FIRST Robotics Competition Robot Rules for the CAN bus wiring requirements. Even though the CAN port is shared with the PWM port, SPARK MAX will automatically detect the incoming signal type and respond accordingly. SPARK MAX uses standard

Each device on the CAN bus must be assigned a unique CAN ID number. Out of the box, SPARK MAX is assigned a device ID of 0. It is highly recommended to change all SPARK MAX CAN IDs from 0 to any unused ID from 1 to 62. CAN IDs can be changed by connecting the SPARK MAX to a Windows computer and using the SPARK MAX Client Application. For details on other SPARK MAX configuration parameters, see Appendix A - Configuration Parameters.

Additional information about the CAN accessible features and how to access them can be found in the SPARK MAX API Information section of the SPARK MAX Software Resources page.

3.3.2.1 - Periodic Status Frames

The SPARK MAX sends data periodically back to the roboRIO. Frequently accessed data, like motor position and temperature, can be accessed using several APIs. Data is broken up into several CAN "frames" which are sent at a periodic rate. This rate can be changed manually in code, but unlike other parameters, this setting **does not persist** through a power cycle. The rate can be set anywhere from a minimum 1ms to a maximum 65535ms period. The table below describes each status frame and its available data.

Table 3.3 - Periodic Status Frames

Periodic Status 0 - Default Rate: 10ms	
Available Data	**Description**
Applied Output	The actual value sent to the motors from the motor controller. The frame stores this value as a 16-bit signed integer, and is converted to a floating point value between -1 and 1 by the roboRIO SDK. This value is also used by any follower controllers to set their output.
Faults	Each bit represents a different fault on the controller. These fault bits clear automatically when the fault goes away.
Sticky Faults	The same as the Faults field, however the bits do not reset until a power cycle or a 'Clear Faults' command is sent.
Is Follower	A single bit that is true if the controller is configured to follow another controller.
Periodic Status 1 - Default Rate: 20ms	
Available Data	**Description**
Motor Velocity	32-bit IEEE floating-point representation of the motor velocity in RPM using the selected sensor.
Motor Temperature	8-bit unsigned value representing: Firmware version 1.0.381 - Voltage of the temperature sensor with 0 = 0V and 255 = 3.3V. Current firmware versions - Motor temperature in Â°C for the NEO Brushless Motor.
Motor Voltage	12-bit fixed-point value that is converted to a floating point voltage value (in Volts) by the roboRIO SDK. This is the input voltage to the controller.
Motor Current	12-bit fixed-point value that is converted to a floating point current value (in Amps) by the roboRIO SDK. This is the raw phase current of the motor.
Periodic Status 2 - Default Rate: 20ms	
Available Data	**Description**

The user can change this rate to 10ms by calling:

Pseudocode

```
setPeriodicFrameRate(PeriodicFrame.kStatus2, 10);
```

HIGH CAN UTILIZATION

A user has many connected CAN devices and wishes to minimize the CAN bus utilization. They do not need any telemetry feedback, and have several follower devices that are only checked for faults.

The user can set the telemetry frame rates low, and set the Periodic Status 0 frame rate low on the follower devices:

Pseudocode

```
leader.setPeriodicFrameRate(PeriodicFrame.kStatus1, 500);
leader.setPeriodicFrameRate(PeriodicFrame.kStatus2, 500);
follower.setPeriodicFrameRate(PeriodicFrame.kStatus0, 100);
follower.setPeriodicFrameRate(PeriodicFrame.kStatus1, 500);
follower.setPeriodicFrameRate(PeriodicFrame.kStatus2, 500);
```

FASTER FOLLOWER BANDWIDTH

The user wants the follower devices to update at a faster rate: 200Hz (every 5ms).

The Periodic Status 0 frame can be increased to achieve this.

Pseudocode

```
leader.setPeriodicFrameRate(PeriodicFrame.kStatus0, 5);
```

3.3.3 - USB INTERFACE

The SPARK MAX can be configured and controlled through a USB connection to a computer running the SPARK MAX Client Application. The USB interface utilizes a standard CDC (USB to Serial) driver. The command interface is similar to CAN, using the same ID and data structure, but always sends and receives a full 12-byte packet. The CAN ID is omitted (DNC) whn talking directly to the device. However, the three MSB of the ID allow selection of alternate commands:

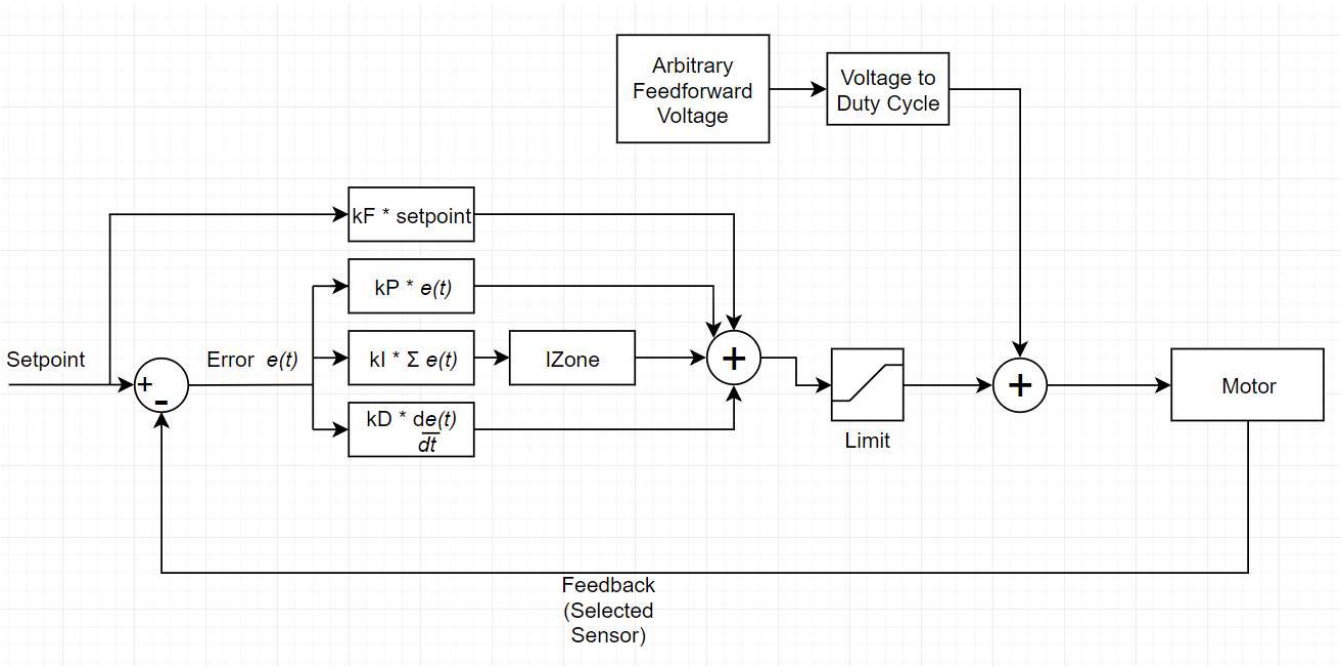
- 0b000 - Standard command - CAN ID omitted (DNC)

Table 3.5 - USB Non-standard Commands

Command	API Class	API Index
Enter DFU Bootloader (will also disconnect USB interface)	0	1

3.4 - CLOSED-LOOP CONTROL

SPARK MAX can operate in several closed-loop control modes, using sensor input to tightly control the motor velocity or position. The internal control loop follows a standard PID algorithm with a feed-forward (F) term to compensate for known system offsets. Below is a diagram and the firmware implementation of the internal SPARK MAX PIDF.




```
//Synchronous PID, call at desired frequency
float pid_run(pid_instance_t* pid, float setpoint, float pv,
        const pid_constants_t* constants)
{
    float error = setpoint - pv;

    float p = error * constants->kP;

    if(fabsf(error) <= constants->iZone || constants->iZone == 0.0f) {
        pid->iState = pid->iState + (error * constants->kI);
    } else {
        pid->iState = 0;
    }
```

```
    return output;
}
```

For more information on utilizing the built-in closed-loop control modes, please take a look at our SPARK MAX Code Examples.

3.5 - RECOVERY MODE

When updating the firmware on the SPARK MAX, it is possible for the process to be interrupted or for the firmware to be corrupted by a bad download. In this state, the Status LED will be dark and the SPARK MAX will fail to operate. SPARK MAX has a built-in recovery mode that can force it to accept new firmware even if the controller seems to be bricked. The following procedure requires a small tool, like a straightened paper clip, to press the Mode Button, a USB C cable, and a computer with the SPARK MAX Client Application installed:

1. With the SPARK MAX powered off completely, press and hold the Mode Button.
2. While still holding the Mode Button, connect the SPARK MAX to the computer using the USB cable. The Status LED **will not** illuminate, this is expected.
3. Wait a few seconds for the computer to recognize the connected device, then release the Mode Button.
4. Open the SPARK MAX Client Application. The SPARK MAX will remain dark and it **will not** connect to the Client, this is expected.
5. Navigate to the **Network** tab and click the **Rescan** arrows at the top of the window.
6. The SPARK MAX will be listed under *Devices in Recovery Mode.* Click the checkbox next to the device.
7. Click the **Load Firmware** button.
8. Select the latest firmware file and click **Open**.
9. The firmware should load successfully and the SPARK MAX will now connect to the Client.

3.6 - ALTERNATE ENCODER MODE

The SPARK MAX can be configured to run in Alternate Encoder Mode, which reconfigures the Data Port on the top of the controller to accept an alternative quadrature encoder, separate from the default encoder inputs shared between the front Encoder Port and the default quadrature encoder Data Port pins. Analog input is not affected by Alternate encoder mode.

This feature is designed for use in low-RPM mechanisms such as drivetrains, arms, and other manipulators. For high RPM applications it is recommended to use the built-in motor sensor for brushless motors or the default encoder inputs for brushed motors.

Table 3.6 - Alternate Encoder Specifications

Parameter	Specification
Encoder Output Voltage Level	3.3V or 5.0V[a]

a. **Before connecting a sensor with 5V output**, the SPARK MAX **must first be configured** into Alternate Encoder Mode and have its configuration **must be saved** on the SPARK MAX. This can be done through the SPARK MAX Client or the software APIs. b. Index pulses are not currently supported.

When configured for Alternate Encoder Mode, a quadrature encoder connected to the reconfigured Data Port pins can be used as a feedback device by the SPARK MAX. **Please note, the limit switch inputs cannot be used at the same time as an alternate encoder.** The limit switch pins are repurposed for the alternate encoder and are thus disabled. Please see 3.6.1 - Connecting an Alternate Encoder for for more information.

3.6.1 - CONNECTING AN ALTERNATE ENCODER

Connecting an alternate encoder will likely require a custom wiring harness to connect the necessary encoder power, ground, and signals to the reconfigured Data Port. When configured in Alternate Encoder Mode, the Data Port has the following pinout:

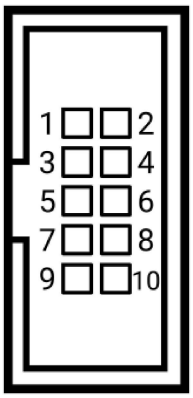


Table 3.7 - Data Port Pinout in Alternate Encoder Mode

Connector Pin	Pin Type	Pin Function
1	Power	+3.3V
2	Power	+5V
3	Analog	Analog Input
4	Digital	**Alternate Encoder Index**[a]
5	Digital	Encoder B
6	Digital	**Alternate Encoder A**
7	Digital	Encoder A
8	Digital	**Alternate Encoder B**
9	Digital	Encoder C / Index
10	Ground	Ground

Port Breakout Board

Breakout Board Pad Label	Alternate Encoder Function
Limit - F	Index[a]
P6 (P5 in older batches)	A
Limit - R	B
3.3V or 5.0V	Encoder Power
GND	Encoder Ground

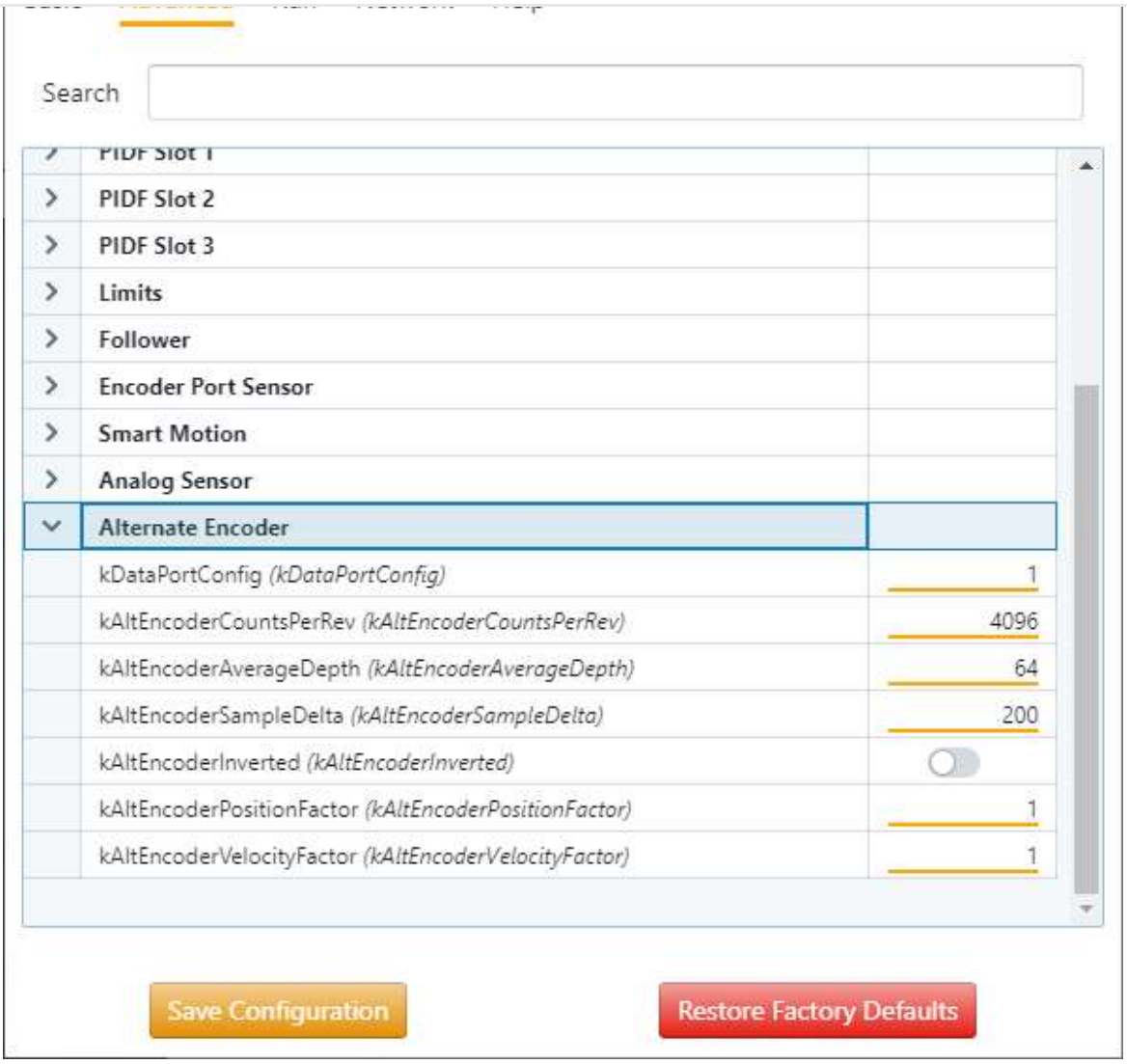
a. Index pulses are not currently supported.

3.6.2 - CONFIGURING AND USING THE ALTERNATE ENCODER MODE

Below you will find the steps required to set up and use the Alternate Encoder Mode on the SPARK MAX, starting with configuration through either the SPARK MAX Client or the SPARK MAX APIs.

3.6.2.1 - Configuration Using the SPARK MAX Client

Using the SPARK MAX Client, navigate to the Advanced Tab and scroll to the Alternate Encoder parameter section. Enable the alternate encoder by setting the *kDataPortConfig* parameter to '1'. You can also set the other Alternate Encoder parameters at this time.



3.6.2.2 - Configuring Using the SPARK MAX APIs

If using the SPARK MAX APIs, the Alternate Encoder is automatically configured when the Alternate Encoder object is instantiated. An Alternate Encoder is created the same as a *CANEncoder*, either by directly using the constructor or calling *GetALternateEncoder()* on a previously constructed *CANSparkMax*.

```
static constexpr int kCanID = 1;
static constexpr auto kMotorType = rev::CANSparkMax::MotorType::kBrushless;
static constexpr auto kAltEncType = rev::CANEncoder::AlternateEncoderType::kQuadrature;
static constexpr int kCPR = 8192;

// initialize SPARK MAX with CAN ID
rev::CANSparkMax m_motor{kCanID, kMotorType};

/**
 * An alternate encoder object is constructed using the GetAlternateEncoder()
 * method on an existing CANSparkMax object. If using a REV Through Bore
```

3.6.2.3 - Configuration Conflicts

Since the alternate encoder inputs and the default digital inputs are shared on the Data Port, the user cannot use both the alternate encoder and a digital inputs in code. Therefore, a **std::invalid_argument** (C++), **IllegalArgumentException** (Java), or an **Error on the Error Out terminal** (LabVIEW) will be thrown if a user tries to construct both types objects in code simultaneously.

3.6.2.4 - Closed-Loop Control

The alternate encoder can be used with the different closed-loop control modes available on the SPARK MAX. The feedback device used by a *CANPIDController* must be set to use the alternate encoder through *SetFeedbackDevice().*

```
/**
 * By default, the PID controller will use the Hall sensor from a NEO for its
 * feedback device. Instead, we can set the feedback device to the alternate
 * encoder object
 */
m_pidController.SetFeedbackDevice(m_alternateEncoder);
```

3.6.2.5 - Initial Bring-up

Unlike the built-in sensor on the NEO Brushless motors, the 'phase' of the alternate encoder is unknown to the SPARK MAX. Before enabling any closed-loop control, it is critical that the phase is configured correctly. To verify:

1. Configure and connect the sensor as a quadrature alternate encoder, but **do not** run a closed-loop mode.
2. Plot the output signal of the motor using *GetAppliedOutput()* and the output of the encoder using *altEncoder.GetVelocity()*. Confirm that the sensor is behaving as expected. This can be done on the SmartDashboard:
```
frc::SmartDashboard::PutNumber("Alt Encoder Velocity", m_alternateEncoder.GetVelocity());
frc::SmartDashboard::PutNumber("Applied Output", m_motor.GetAppliedOutput());
```
3. Verify that the sign of the sensor is correct relative to the motor direction when driving it forward and backward. If it is not, the sensor must be inverted by calling *altEncoder.SetInverted(true).*

APPENDIX A - CONFIGURATION PARAMETERS

Below is a list of all the configurable parameters within the SPARK MAX. Parameters can be set through the CAN or USB interfaces. The parameters are saved in a different region of memory from the device firmware and persist through a firmware update.

kInputMode	1	Input Mode	-	0	mode is detected by the firmware automatically. 0 - PWM 1 - CAN 2 - USB
kMotorType	2	Motor Type	-	BRUSHLESS	Motor type: 0 - Brushed 1 - Brushless **This parameter persists through a normal firmware update.**
Reserved	3	-	-		Reserved
kSensorType	4	Sensor Type	-	HALL_EFFECT	Sensor type: 0 - No Sensor 1 - Hall Sensor 2 - Encoder **This parameter persists through a normal firmware update.**
kCtrlType	5	Ctrl Type	-	CTRL_DUTY_CYCLE	Control Type, this is a read only parameter of the currently active control type. The control type is changed by calling the correct API. 0 - Duty Cycle 1 - Velocity 2 - Voltage 3 - Position
kIdleMode	6	Idle Mode	-	IDLE_COAST	State of the half bridge when the motor controller commands zero output or is disabled. 0 - Coast 1 - Brake **This parameter persists through a**

					update.
Reserved	8	-	-	-	Reserved
Reserved	9	-	-	-	Reserved
kPolePairs	10	uint	-	7	Number of pole pairs for the brushless motor. This is the number of poles/2 and can be determined by either counting the number of magnets or counting the number of windings and dividing by 3. This is an important term for speed regulation to properly calculate the speed.
kCurrentChop	11	float32	Amps	115	If the half bridge detects this current limit, it will disable the motor driver for a fixed amount of time set by kCurrentChopCycles. This is a low sophistication 'current control'. Set to 0 to disable. The max value is 125.
kCurrentChopCycles	12	uint	-	0	Number of PWM Cycles for the h-bridge to be off in the case that the current limit is set. Min = 1, multiples of PWM period (50Î¼s). During this time the current will be recirculating through the low side MOSFETs, so instead of 'freewheeling' the diodes, the bridge will be in brake mode during this time.
kP_0	13	float32	-	0	Proportional gain constant for gain slot 0.
kI_0	14	float32	-	0	Integral gain constant for gain slot 0.
kD_0	15	float32	-	0	Derivative gain constant for gain slot 0.

kDFilter_0	18	float32	-	0	constant for gain slot 0.
kOutputMin_0	19	float32	-	-1	Max output constant for gain slot 0. This is the max output of the controller.
kOutputMax_0	20	float32	-	1	Min output constant for gain slot 0. This is the min output of the controller.
kP_1	21	float32	-	0	Proportional gain constant for gain slot 1.
kI_1	22	float32	-	0	Integral gain constant for gain slot 1.
kD_1	23	float32	-	0	Derivative gain constant for gain slot 1.
kF_1	24	float32	-	0	Feed Forward gain constant for gain slot 1.
kIZone_1	25	float32	-	0	Integrator zone constant for gain slot 1. The PIDF loop integrator will only accumulate while the setpoint is within IZone of the target.
kDFilter_1	26	float32	-	0	PIDF derivative filter constant for gain slot 1.
kOutputMin_1	27	float32	-	-1	Max output constant for gain slot 1. This is the max output of the controller.
kOutputMax_1	28	float32	-	1	Min output constant for gain slot 1. This is the min output of the controller.
kP_2	29	float32	-	0	Proportional gain constant for gain slot 2.
kI_2	30	float32	-	0	Integral gain constant for gain slot 2.
kD_2	31	float32	-	0	Derivative gain constant for gain slot 2.
kF_2	32	float32	-	0	Feed Forward gain constant for gain slot 2.
					Integrator zone constant for gain slot

					controller.
kOutputMax_2	36	float32	-	1	Min output constant for gain slot 2. This is the min output of the controller.
kP_3	37	float32	-	0	Proportional gain constant for gain slot 3.
kI_3	38	float32	-	0	Integral gain constant for gain slot 3.
kD_3	39	float32	-	0	Derivative gain constant for gain slot 3.
kF_3	40	float32	-	0	Feed Forward gain constant for gain slot 3.
kIZone_3	41	float32	-	0	Integrator zone constant for gain slot 3. The PIDF loop integrator will only accumulate while the setpoint is within IZone of the target.
kDFilter_3	42	float32	-	0	PIDF derivative filter constant for gain slot 3.
kOutputMin_3	43	float32	-	-1	Max output constant for gain slot 3. This is the max output of the controller.
kOutputMax_3	44	float32	-	1	Min output constant for gain slot 3. This is the min output of the controller.
Reserved	45	-	-	-	Reserved
Reserved	46	-	-	-	Reserved
Reserved	47	-	-	-	Reserved
Reserved	48	-	-	-	Reserved
Reserved	49	-	-	-	Reserved
kLimitSwitchFwdPolarity	50	bool	-	0	Forward Limit Switch polarity. 0 - Normally Open 1 - Normally Closed
kLimitSwitchRevPolarity	51	bool	-	0	Reverse Limit Switch polarity. 0 - Normally Open 1 - Normally Closed

kRampRate	56	float32	V/s	0	modes in % output per second, a value of 0 disables this feature. All APIs take the reciprocal to make the unit 'time from 0 to full'.
kFollowerID	57	uint	-	0	CAN EXTID of the message with data to follow
kFollowerConfig	58	uint	-	0	Special configuration register for setting up to follow on a repeating message (follower mode). CFG[0] to CFG[3] where CFG[0] is the motor output start bit (LSB), CFG[1] is the motor output stop bit (MSB). CFG[0] - CFG[1] determines endianness. CFG[2] bits determine sign mode and inverted, CFG[3] sets a preconfigured controller (0x1A = REV, 0x1B = Talon/Victor style as of 2018 season)
kSmartCurrentStallLimit	59	uint	A	80	Smart Current Limit at stall, or any RPM less than kSmartCurrentConfig RPM.
kSmartCurrentFreeLimit	60	uint	A	20	Smart current limit at free speed
kSmartCurrentConfig	61	uint	-	10000	Smart current limit RPM value to start linear reduction of current limit. Set this > free speed to disable.
Reserved	62	-	-	-	Reserved
Reserved	63	-	-	-	Reserved
Reserved	64	-	-	-	Reserved
Reserved	65	-	-	-	Reserved
Reserved	66	-	-	-	Reserved
Reserved	67	-	-	-	Reserved
Reserved	68	-	-	-	Reserved
kEncoderCountsPerRev	69	uint	-	4096	Number of encoder counts in a single revolution, counting

kEncoderAverageDepth	70	uint	-	64	to average for velocity data based on quadrature encoder input. This value can be between 1 and 64.
kEncoderSampleDelta	71	uint	per 500us	200	Delta time value for encoder velocity measurement in 500Î¼s increments. The velocity calculation will take delta the current sample, and the sample x * 500Î¼s behind, and divide by this the sample delta time. Can be any number between 1 and 255
Reserved	72	-	-	-	Reserved
Reserved	73	-	-	-	Reserved
Reserved	74	-	-	-	Reserved
kCompensatedNominalVoltage	75	float32	V	0	In voltage compensation mode mode, this is the max scaled voltage.
kSmartMotionMaxVelocity_0	76	float32	-	0	
kSmartMotionMaxAccel_0	77	float32	-	0	
kSmartMotionMinVelOutput_0	78	float32	-	0	
kSmartMotionAllowedClosedLoopError_0	79	float32	-	0	
kSmartMotionAccelStrategy_0	80	float32	-	0	
kSmartMotionMaxVelocity_1	81	float32	-	0	
kSmartMotionMaxAccel_1	82	float32	-	0	
kSmartMotionMinVelOutput_1	83	float32	-	0	
kSmartMotionAllowedClosedLoopError_1	84	float32	-	0	
kSmartMotionAccelStrategy_1	85	float32	-	0	
kSmartMotionMaxVelocity_2	86	float32	-	0	
kSmartMotionMaxAccel_2	87	float32	-	0	
kSmartMotionMinVelOutput_2	88	float32	-	0	
kSmartMotionAllowedClosedLoopError_2	89	float32	-	0	
kSmartMotionAccelStrategy_2	90	float32	-	0	
kSmartMotionMaxVelocity_3	91	float32	-	0	
kSmartMotionMaxAccel_3	92	float32	-	0	
kSmartMotionMinVelOutput_3	93	float32	-	0	

kSlot3Placeholder2_1	102	float32	-	0	
kSlot3Placeholder3_1	103	float32	-	0	
kIMaxAccum_2	104	float32	-	0	
kSlot3Placeholder1_2	105	float32	-	0	
kSlot3Placeholder2_2	106	float32	-	0	
kSlot3Placeholder3_2	107	float32	-	0	
kIMaxAccum_3	108	float32	-	0	
kSlot3Placeholder1_3	109	float32	-	0	
kSlot3Placeholder2_3	110	float32	-	0	
kSlot3Placeholder3_3	111	float32	-	0	
kPositionConversionFactor	112	float32	-	1	
kVelocityConversionFactor	113	float32	-	1	
kClosedLoopRampRate	114	float32	DC/sec	0	
kSoftLimitFwd	115	float32	-	0	Soft limit forward value
kSoftLimitRev	116	float32	-	0	Soft limit reverse value
Reserved	117	-	-	-	Reserved
Reserved	118	-	-	-	Reserved
kAnalogPositionConversion	119	float32	rev/volt	1	Conversion factor for position from analog sensor. This value is multiplied by the voltage to give an output value.
kAnalogVelocityConversion	120	float32	vel/v/s	1	Conversion factor for velocity from analog sensor. This value is multiplied by the voltage to give an output value.
kAnalogAverageDepth	121	uint	-	0	Number of samples in moving average of velocity.
kAnalogSensorMode	122	uint	-	0	0 Absolute: In this mode the sensor position is always read as voltage * conversion factor and reads the absolute position of the sensor. In this mode setPosition() does not have an effect. 1 Relative: In this

					sensor value is 3.3V - voltage. In relative mode the direction is reversed.
kAnalogSampleDelta	124	uint	-	0	Delta time between samples for velocity measurement
Reserved	125	-	-	-	Reserved
Reserved	126	-	-	-	Reserved
kDataPortConfig	127	uint	-	0	0: Default configuration using limit switches 1: Alternate Encoder Mode - limit switches are disabled and alternate encoder is enabled. **This parameter persists through a normal firmware update.**
kAltEncoderCountsPerRev	128	uint	-	4096	Number of encoder counts in a single revolution, counting every edge on the A and B lines of a quadrature encoder. (Note: This is different than the CPR spec of the encoder which is 'Cycles per revolution'. This value is 4 * CPR.
kAltEncoderAverageDepth	129	uint	-	64	Number of samples to average for velocity data based on quadrature encoder input. This value can be between 1 and 64.
kAltEncoderSampleDelta	130	uint	-	200	Delta time value for encoder velocity measurement in 500Î¼s increments. The velocity calculation will take delta the current sample, and the sample x * 500Î¼s behind, and divide by this the sample delta time. Can be any number between 1 and 255

					encoder for position.
kAltEncoderVelocityFactor	133	float32	-	1	Value multiplied by the native units (rotations) of the encoder for velocity.