KDE Direct, LLC. License Agreement

PLEASE READ THIS LICENSE AND MANUAL IN THEIR ENTIRETY BEFORE USING THE KDE UVC SERIES ESC AND/OR KDECAN PROTOCOL. BY USING THE KDE UVC SERIES ESC AND/OR KDECAN PROTOCOL, YOU ARE ACKNOWLEDGING THAT YOU HAVE READ AND UNDERSTOOD THE CONTENT, INCLUDING ALL WARNINGS AND DISCLAIMERS CONTAINED HEREIN. YOU FURTHER ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD ALL KDE POLICIES REFERRED TO IN THE MANUAL, INCLUDING KDE'S LIABILITY POLICY AVAILABLE AT http://www.kdedirect.com/pages/liability-policy. IF YOU DO NOT UNDERSTAND ANY OF THE WARNINGS AND/OR DISCLAIMERS, DO NOT USE THE KDE UVC SERIES ESC AND/OR KDECAN PROTOCOL.

KDE DIRECT RESERVES THE RIGHT TO CHANGE OR MODIFY THE TERMS OF THE LICENSE AND MANUAL AT ANY TIME WITHOUT PRIOR NOTICE.

1. General. The KDECAN Protocol Software (defined herein) is designed to enable functionality between KDE UVC Series ESC to your third party controllers. This document provides general recommendations to enable functionality between the KDE UVC Series ESC and your third party controller.

As detailed below, KDE Direct, LLC. ("KDE Direct") does not manufacture this third party product and therefore does not support or warranty the safety, use, functionality or compatibility of any third party controllers. Any information as to a third party controller is provided solely as a reference. Please refer to documentation accompanying your third party controller and any updates issued by the third party controller manufacturer for best practices and/or other information related to the safety, use, functionality or capability of your controller.

2. **Intellectual Property.** The KDECAN Protocol Software, documentation, images, content, interfaces, fonts and any other data accompanying this License whether pre-installed on the KDE UVC Series ESC hardware, on disk, in read only memory, on any other media or in any other form ("KDECAN Protocol Software") are licensed, not sold, to you by KDE Direct for use only under the terms of this License. KDE Direct retains ownership of the KDECAN Protocol Software itself and reserve all rights not expressly granted to you. The terms of this License will govern any software upgrades provided by KDE Direct that replace and/or supplement the original KDECAN Protocol Software, unless such upgrade is accompanied by a separate license in which case the terms of that license will govern.

Title and intellectual property rights in and to any content displayed by or hardware accessed through the KDECAN Protocol Software belongs to the respective owner. Such content or hardware may be protected by copyright or other intellectual property laws and treaties, and may be subject to terms of use of the third party providing such content. This License does not grant you any rights to use such content or hardware nor does it guarantee that such content or hardware will continue to be available to you.

3. **License Uses and Restrictions.** The Software is licensed to you and not sold. Subject to the terms of this Agreement, KDE DIRECT hereby grants you a personal, non-exclusive, non-transferable, non-sublicensable, and revocable license to use the Software solely in connection with the related program output for which the software was designed, specifically, with third party controllers.
You acknowledge that the Software is protected intellectual property and KDE DIRECT reserves all such rights with respect to the Software, except for the license expressly granted to you in this Agreement. Except for such express license, no right, title, interest or license in or to the Software, whether by implication, estoppel or otherwise, is granted, assigned or transferred to you. You agree not to take any action that interferes, in any manner, with KDE DIRECT or any of its authorized licensor's rights with respect to the Software. In addition, title, ownership rights and intellectual property rights in and to any content accessed through the Software is the property of the applicable content owner and may be protected by applicable copyright, patent, trademark or other law. This Agreement gives you no rights to such content.

You acknowledge and agree that you will not: (a) reproduce the Software; (b) modify, adapt, translate or create any derivative works of the Software; (c) attempt to circumvent or disable the Software or any technology features or measures in the Software by any means or in any manner; (d) attempt to decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for the Software; (e) distribute, encumber, sell, rent, lease, sublicense, or otherwise transfer, publish or disclose the Software to any third party; (f) remove or alter any trademark, logo, copyright or other proprietary notices, legends, symbols or labels in or on the Software or used in connection with the Software; or (g) use the Software in any manner to aid in the violation of any third party intellectual property rights, including but not limited to another's copyrights, trade secrets, and patents.

4. **Third Party Products and Disclaimer.** The KDECAN Protocol Software may enable, display, include or make available content, data, information, applications, hardware or materials from third parties ("Third Party Products") or make references to certain Third Party Products, including but not limited to the PixHawk Flight Controller. By using the KDECAN Protocol Software, you acknowledge and agree that KDE Direct is not responsible for examining, supporting or evaluating the functionality, content, accuracy, completeness, timeliness, validity, quality, legality, or any other aspect of Third Party Products or associated goods or services. Further, KDE Direct, its officers, affiliates and subsidiaries do not warrant or endorse and do not assume and will not have any liability or responsibility to you or any other person for any Third Party Products, or for any other materials, products, or services of third parties. References to Third Party Products is provided solely as a convenience to you.

5. **Disclaimer of Warranties.** YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT, TO THE EXTENT PERMITTED BY APPLICABLE LAW, USE OF THE KDECAN PROTOCOL SOFTWARE AND ANY FUNCTIONALITY ENABLED BY THE KDECAN PROTOCOL SOFTWARE IS AT YOUR SOLE RISK AND THAT THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH YOU. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE KDECAN PROTOCOL SOFTWARE IS PROVIDED "AS IS" AND "AS AVAILABLE," WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, AND KDE DIRECT HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH RESPECT TO THE KDECAN PROTOCOL SOFTWARE, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND NON-INFRINGEMENT OF THIRD PARTY RIGHTS. KDE DIRECT DOES NOT WARRANT AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE KDECAN PROTOCOL SOFTWARE, THAT THE FUNCTIONS CONTAINED IN, OR SERVICES PERFORMED BY OR PROVIDED BY, THE KDECAN PROTOCOL SOFTWARE WILL MEET YOUR REQUIREMENTS, THAT THE OPERATION OF THE KDECAN PROTOCOL SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, THAT ANY PERFORMANCE OF THE KDECAN PROTOCOL SOFTWARE WILL CONTINUE TO BE MADE AVAILABLE, THE KDECAN PROTOCOL SOFTWARE WILL BE COMPATIBLE OR WORK WITH ANY THIRD PARTY PRODUCTS, SOFTWARE, APPLICATIONS OR THIRD PARTY SERVICES, OR THAT DEFECTS IN THE KDECAN PROTOCOL SOFTWARE WILL BE CORRECTED. INSTALLATION OF THIS SOFTWARE MAY AFFECT THE USABILITY OF THIRD PARTY PRODUCTS, SOFTWARE, APPLICATIONS OR THIRD PARTY SERVICES. YOU ALSO ACKNOWLEDGE THAT USE OF KDE UVC SERIES ESC AND/OR FLIGHT CONTROLLERS REQUIRES ADVANCED WIRING KNOWLEDGE AND YOU ASSUME ALL RESPONSIBILITY ASSOCIATED WITH PROPER WIRING OF THE KDE UVC SERIES ESC AND/OR FLIGHT CONTROLLERS. YOU FURTHER ACKNOWLEDGE THAT THE KDECAN PROTOCOL SOFTWARE IS NOT INTENDED OR SUITABLE FOR USE IN SITUATIONS OR ENVIRONMENTS WHERE THE FAILURE OR TIME DELAYS OF, OR ERRORS OR INACCURACIES IN THE CONTENT, DATA OR INFORMATION PROVIDED BY, THE KDECAN PROTOCOL SOFTWARE COULD LEAD TO DEATH, PERSONAL INJURY OR SEVERE PHYSICAL OR ENVIRONMENTAL DAMAGE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY KDE DIRECT OR AN AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SHOULD THE KDE DIRECT PROTOCOL SOFTWARE PROVE DEFECTIVE, YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SUPPORT, SERVICING, REPAIR OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES OR LIMITATION ON APPLICABLE STATUTORY RIGHTS OF A CUSTOMER, SO THE ABOVE DESCRIPTION MAY NOT APPLY TO YOU.

6. **Limitation of Liability.** AS KDE DIRECT HAS NO CONTROL OVER USE, SETUP, FINAL ASSEMBLY, MODIFICATION, OR MISUSE, NO LIABILITY SHALL BE ASSUMED NOR ACCEPTED FOR ANY RESULTING DAMAGE OR INJURY. BY THE ACT OF USE OF THE KDECAN PROTOCOL SOFTWARE, THE USER ACCEPTS ALL RESULTING LIABILITY. TO THE EXTENT NOT PROHIBITED BY APPLICABLE LAW, IN NO EVENT SHALL KDE DIRECT BE LIABLE FOR PERSONAL INJURY, OR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, ARISING OUT OF OR RELATED TO YOUR USE OR INABILITY TO USE THE KDECAN PROTOCOL SOFTWARE OR THIRD PARTY PRODUCTS USED IN CONJUNCTION WITH THE KDECAN PROTOCOL SOFTWARE, REGARDLESS OF THE THEORY OF LIABILITY (CONTRACT, TORT OR OTHERWISE) AND EVEN IF KDE DIRECT HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH

DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF LIABILITY FOR PERSONAL INJURY, OR OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION MAY NOT APPLY TO YOU. IN NO EVENT SHALL KDE DIRECT'S TOTAL LIABILITY TO YOU FOR ALL DAMAGES (OTHER THAN AS MAY BE REQUIRED BY APPLICABLE LAW IN CASES INVOLVING PERSONAL INJURY) EXCEED THE AMOUNT OF THE INDIVIDUAL PRICE OF THE KDECAN PROTOCOL SOFTWARE ON WHICH LIABILITY IS ASSERTED. THE FOREGOING LIMITATIONS WILL APPLY EVEN IF THE ABOVE STATED REMEDY FAILS OF ITS ESSENTIAL PURPOSE. IF THE USER IS NOT PREPARED TO ACCEPT THE LIABILITY ASSOCIATED WITH THE USE OF THE KDECAN PROTOCOL SOFTWARE, THEY ARE ADVISED TO RETURN THE KDECAN PROTOCOL SOFTWARE IMMEDIATELY TO KDE DIRECT OR AN AUTHORIZED DEALER.

7. **Export Restrictions**. KDE DIRECT makes no representation that the Software is appropriate for use in your country of use. You acknowledge that none of the Software or underlying information or technology may be downloaded or otherwise exported or re-exported into (or to a national or resident of) any countries subject to U.S. trade embargo, or anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. By using the Software, you are agreeing to the foregoing, and are representing and warranting that you are not located in or under the control of a national or resident of any such country or on any such list.

8. **Indemnity.** You hereby agree to indemnify, defend and hold KDE DIRECT and its authorized licensors harmless from and against any and all liabilities, damages, claims, fines and expenses (including reasonable attorneys' fees and costs) arising out of any breach of this Agreement by you.

9. **Termination.** This License is effective until terminated. Your rights under this License will terminate automatically or otherwise cease to be effective without notice from KDE Direct if you fail to comply with any term(s) of this License. Upon termination of the License, you shall cease all use of the KDECAN Protocol Software and destroy all copies, full or partial, of the KDECAN Protocol Software. Sections 1 and 3-8 of this License shall survive any such termination.

10. **Controlling Law and Severability**. This License will be governed by and construed in accordance with the laws of Oregon, without reference to conflict of laws principles. The sole forum for any disputes will be in Multnomah County, Oregon and any objections to jurisdiction are hereby waived. This License shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this License shall continue in full force and effect.

11. **Complete Agreement; Governing Language**. This License constitutes the entire agreement between you and KDE Direct relating to the use of the KDECAN Protocol Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this License will be binding unless in writing and signed by both parties.

Overview

The KDECAN bus firmware update enables KDE UVC ESCs to communicate and actively monitor critical system components through CAN bus. This update allows telemetry and real time monitoring for enhanced safety during operation. Any device with a CAN bus transceiver is capable of communicating with a KDE UVC ESC. This works by having the device send CAN messages to the ESC of which the ESC responds accordingly. This document defines the message structure and protocol necessary to communicate with UVC ESCs through CAN bus.

Contents

List of Figures

KDECAN Capabilities

- **Control Signals (Send)**
  - Throttle Control (50Hz - 500Hz refresh-rate controls)
  - Shutdown Procedure (turn off ESC via command)
  - Restart Procedure (re-arm and enable ESC via command)
- **Live-Telemetry Feedback Signals (Receive)**
  - Drive Voltage (V)
  - Drive Current (A)
  - ESC MCU Temperature (°C)
  - Output Throttle Duty-Cycle (%)
  - Input Throttle Signal (µs)
  - Motor eRPM (rpm)
  - Warning Signals and Errors

CAN Bus Protocol

Selectable Baud Rate: 1000K (default) / 500K / 250K / 125K / 100K bps
Frame Format: Extended Frame Format (CAN 2.0B) 29-bit identifier or
Standard Frame Format (CAN 2.0A) 11-bit identifier
Endianness: Big Endian
KDE ESC series: UVC

ESC minimum firmware version required:	D460115_0404
KDE Device Manager minimum version required:	KDEDevice V134.1
Default CAN master ID:	0x00
Default ESC ID:	0x01

System Configuration

To configure the CAN bus network, the system must be correctly wired with each ESC assigned an ID. Please see the Wiring Schematic and ID Assignment sections below for reference.

KDE UVC Pinout Schematic

The KDE UVC series ESCs come with dual CAN bus ports. These ports use a standard CAN bus 4-pin cable. For compatible cables please visit the KDE Direct website at https://www.kdedirect.com

For manual configuration, please refer to the picture shown below:

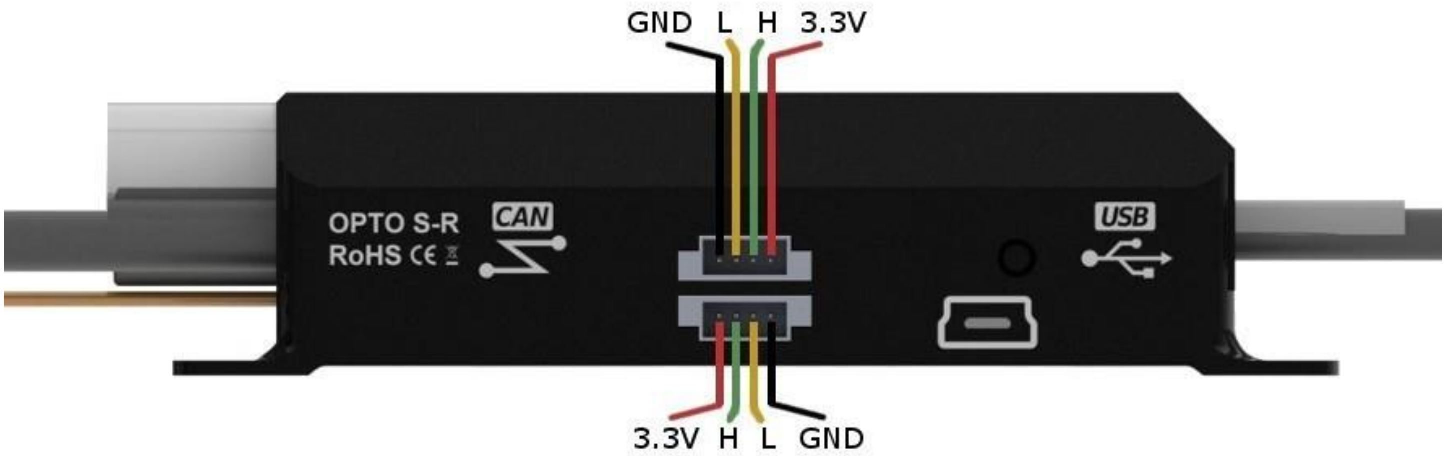


Figure 1: CAN bus port configuration.

Connect multiple ESCs together following the wire setup shown below. To connect additional ESCs, simply connect the two closest CAN bus ports. Wire kits can be found here.

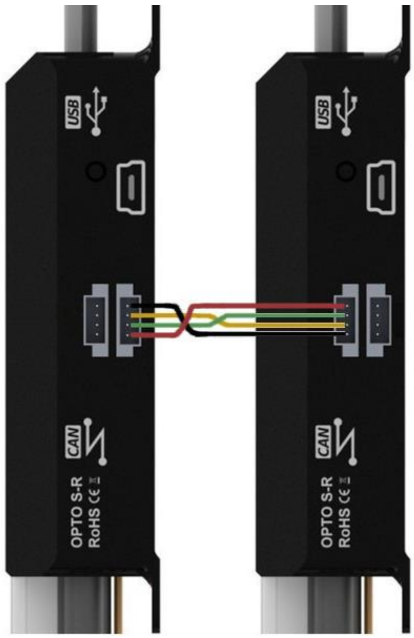


Figure 2: ESC to ESC CAN bus connection.

Wiring Schematic

An example CAN bus network configuration is shown below. Each CAN bus network must contain a master, one or more ESCs that connect to the master, and a 120 ohm terminating resistor. The 120 ohm terminating resistor is required on all CAN bus networks. The Pixhawk has KDECAN abilities implemented however, the Pixhawk is used only as a reference. Flight controllers that are open source or internally developed can incorporate the KDECAN protocol.

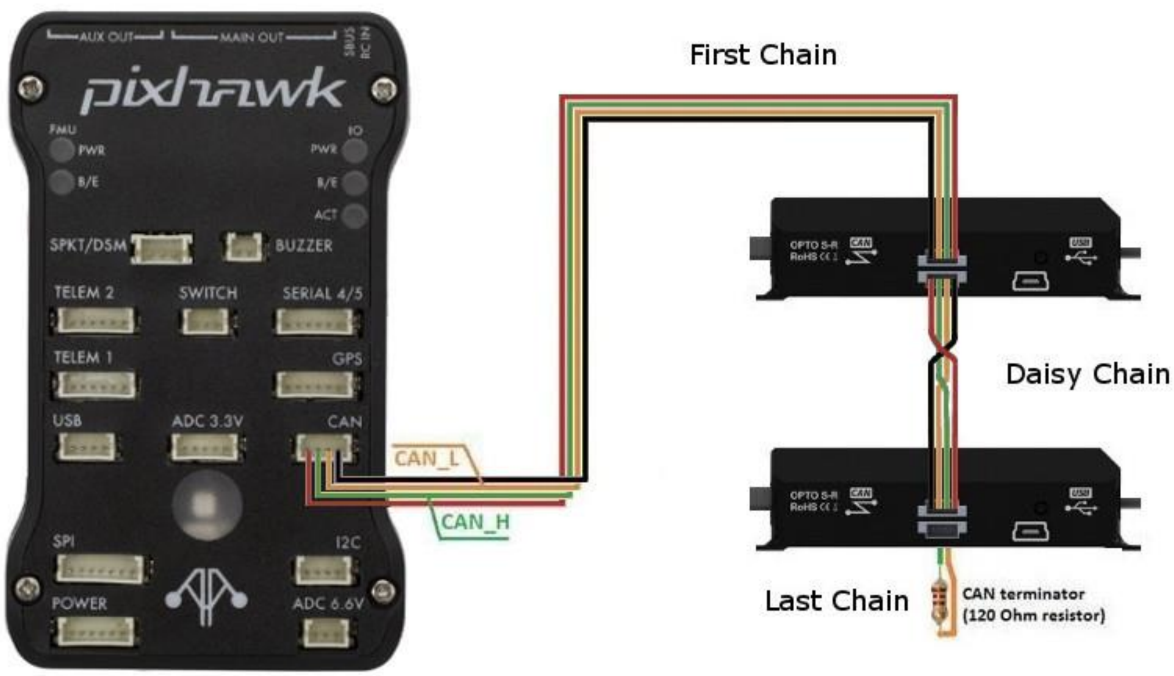


Figure 3: Example CAN bus network.

Note: 3.3V CAN transceivers are fully interoperable with 5V CAN transceivers.

ID Assignment

By default, each ESC has an ID of 0x01 and the master (flight controller or CAN bus analyzer) has an ID of 0x00. This ESC ID is a unique ID used for identifying different ESCs on the CAN BUS network. Each ESC ID must be assigned before it can operate on the network. This can be accomplished through two methods. The first method manually assigns each ESC's ID through the KDE Device Manager. The second method sends an "Enumeration" message followed by an "Update Node Address" message.

Method 1 (Manual ID Assignment)

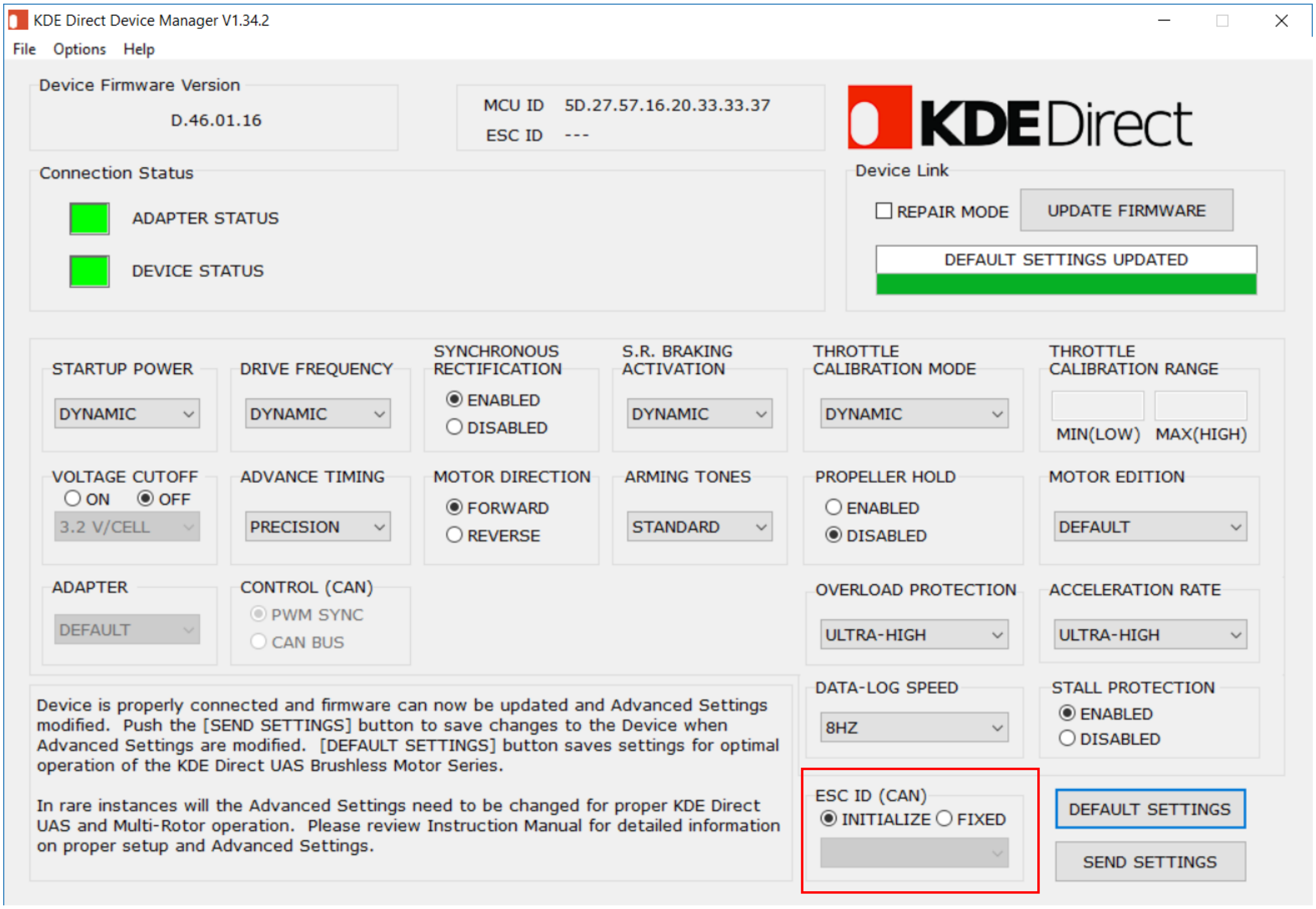


Figure 4: KDE Direct Device Manager (download link)

- The ESC ID can be set in the KDE Device Manager through the ESC ID (CAN) selection.
- To set the ESC ID, press the FIXED checkbox, select the desired ID, and press the SEND SETTINGS button.

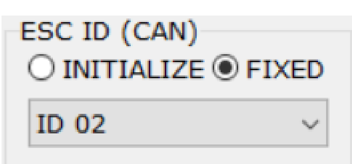


Figure 5: ESC Unique ID Assignment

- The ESC ID will be updated (displayed at the top of the Device Manager window).
- If the ID displayed is "---" then the ID is uninitialized and uses "1" as its ID.
- The MCU ID represents the unique ID of the STM32 CPU in the ESC.

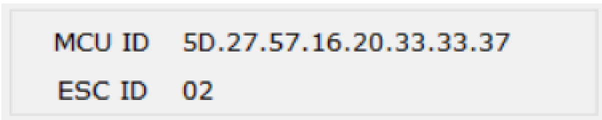


Figure 6: Updated ESC ID

Method 2 (ESC Enumeration)

The ESC enumeration message allows the master to assign the CAN bus ESC ID. This works by having the user manually rotate the motors (1/2 turn) which are connected to the ESCs. Once the master receives the MCU IDs, it can then send a message to set the ESC IDs in the order the motors were rotated. The source and destination bits in the extended frame ID of an ESC enumeration message is filled with a unique number so that a CAN bus error won't occur from messages having nonunique IDs.

Direction	Description	CAN Message (Frame and Data)
Master to ESC (send)	Start ESC Enumeration Rotated motors return ESC MCU #	Frame ID: 00 00 01 0A Data: 27 10
ESC to Master (receive)	Motor connected to ESC A is rotated	Frame ID: 00 05 17 0A Data: FB C1 57 15 20 33 33 37
ESC to Master (receive)	Motor connected to ESC B is rotated	Frame ID: 00 08 73 0A Data: 5D 27 57 16 20 33 33 37
ESC to Master (receive)	Motor connected to ESC C is rotated	Frame ID: 00 0B 22 0A Data: 7C 32 57 13 20 33 33 37
Master to ESC (send)	Set ESC A to ESC ID 2	Frame ID: 00 00 02 09 Data: FB C1 57 15 20 33 33 37
Master to ESC (send)	Set ESC B to ESC ID 3	Frame ID: 00 00 03 09 Data: 5D 27 57 16 20 33 33 37
Master to ESC (send)	Set ESC C to ESC ID 4	Frame ID: 00 00 04 09 Data: 7C 32 57 13 20 33 33 37
Master to ESC (send)	(optional*) ping all ESCs on network	Frame ID: 00 00 01 08 Data: 00

Figure 7: ESC ID assignment example.

Note: The optional command is verification that all ESCs are connected properly.

Communication Definitions and Algorithms

KDECAN uses different data types in the messages that are sent between the ESCs and the master. The CRC algorithm is used to verify that no information is altered within the transmission of the CAN message. The data type descriptions and algorithm are provided below for reference.

Note: Multiple CAN bus analyzers are available on the market. KDE used the CANalyst 2 during development for its features and interface.

Data Types

The following data types used in CAN messages are described below:

HA Hexadecimal ASCII [‘0’... ‘9’, ‘A’... ‘F’]
U8 Unsigned char [0...0xFF]
U16 Unsigned short (High byte – byte 0, Low byte – byte 1), [0… 0xFFFF]
U32 Unsigned int (High byte – byte 0, Low byte – byte 3), [0... 0xFFFFFFFF]
U64 Unsigned long (High byte – byte 0, Low byte – byte 7), [0... 0xFFFFFFFFFFFFFFFF]

CRC Algorithm

The error management as described in the CAN protocol is handled entirely by hardware using a Transmit Error Counter (TEC value, in CAN_ESR register) and a Receive Error Counter (REC value, in the CAN_ESR register), which get incremented or decremented according to the error condition. For detailed information about TEC and REC management, refer to the CAN standard.

For more information please refer to STM32 AN4187

Endianness

KDECAN uses Big Endian for multi-byte values.
Example:

Decimal Value: 1000
Hex Value in Big Endian: 0x03E8

(Byte)	0	1	2	3	4	5	6	7
(Hex)	[03]	[E8]	[xx]	[xx]	[xx]	[xx]	[xx]	[xx]

CAN Bus Extended Frame Structure (CAN 2.0B)

A CAN bus frame consists of an extended frame ID and a data frame. The extended frame ID consists of 5 bits for priority, 8 bits for the source id (sender), 8 bits for the destination id (receiver), and 8 bits for the object address which tells the ESC how to respond to the message.

Extended Frame ID				Data (8Bytes)							
5-bit Priority	8-bit Source	8-bit Destination	8-bit Object	0	1	2	3	4	5	6	7
29-bit Identifier											

Figure 8: CAN Bus frame.

Extended Frame ID				Data							
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x00	0x02	0x00	0x08	MCU ID0	MCU ID1	MCU ID2	MCU ID3	MCU ID4	MCU ID5	MCU ID6	MCU ID7

Figure 9: CAN bus frame example.

CAN Bus Standard Frame Structure (CAN 2.0A)

CAN bus messages using standard frames (11-bit identifier) work the same way as extended frames (29-bit identifier) the only difference is how the bits are interpreted. The standard frame ID consists of 1 bit for the source (0 for master and 1 for the ESC), 5 bits for the ESC ID, and 5 bits for the object address. The ESC will automatically detect the frame type and respond with a standard frame or extended frame based on the received CAN bus message.

Standard Frame ID			Data (8Bytes)							
1-bit Source	5-bit ESC ID	5-bit Object	0	1	2	3	4	5	6	7
11-bit Identifier										

Example:

Direction	Standard Frame ID	Data	Description
Master to ESC	Bin: 0 00010 00000 (Hex: 0x040)		Query FW and HW (obj 0)
ESC to Master	Bin: 1 00010 00000 (Hex: 0x440)	0C 00 01 00 46	ESC ID 2 responds with its FW and HW ver.
Master to ESC	Bin: 0 00011 01000 (Hex: 0x068)		Query MCU ID (obj 8)
ESC to Master	Bin: 1 00011 01000 (Hex: 0x468)	99 70 57 17 20 33 33 37	ESC ID 2 responds with its MCU ID
Master Broadcast	Bin: 0 00001 00011 (Hex: 0x02B)		Broadcast query voltage, current, etc.
ESC ID 2 to Master	Bin: 1 00010 00000 (Hex: 0x44B)	06 27 00 C8 00 00 1B 00	ESC ID 2 responds to message obj 11
ESC ID 3 to Master	Bin: 1 00011 01011 (Hex: 0x46B)	05 f5 01 2C 00 00 19 00	ESC ID 3 responds to message obj 11

CAN Bus Messages

CAN bus messages are sent between the master and the ESCs. These messages define the primary method of communication on the CAN bus network. A properly configured network will consist of a master that has an ID of 0x00 and ESCs that have unique IDs of 0x02~0xFF. An ESC ID 0x02 is used in the following messages because ESC ID 0x01 represents an unassigned ESC. The ESC ID of 0x01 is also used for broadcasting messages. When sending a message from the master, if there are additional bytes specified in the data frame, they are ignored by the ESC.

Object address	*Data length* (Send)	*Data Transmission* (Send)	*Data Transmission* (Receive)	*Data length* (Receive)	*Information Definition* (Send/Receive, Refresh Rate)
0	*0*	*No data*	*ESC information*	*5*	*Receive ESC Programming Information*
1	*2*	*PWM (Throttle)*	*None*	*0*	*Send Throttle Control (0…2200µs)* *50 Hz (20 ms) to 500 Hz (2 ms)*
2	*0*	*No data*	*ESC's Voltage*	*2*	*Receive ESC Input Voltage (V)* *10 Hz (100 ms) to 500 Hz (2 ms)*
3	*0*	*No data*	*ESC's Current*	*2*	*Receive ESC Input Current (A)* *10 Hz (100 ms) to 500 Hz (2 ms)*
4	*0*	*No data*	*ESC's RPM*	*2*	*Receive ESC eRPM (rpm)* *10 Hz (100 ms) to 500 Hz (2 ms)*
5	*0*	*No data*	*ESC's Temperature*	*1*	*Receive ESC MCU Temperature (°C)* *10 Hz (100 ms) to 500 Hz (2 ms)*
6	*0*	*No data*	*ESC's Input Throttle*	*2*	*Receive ESC Input Throttle (0…2200µs)* *10 Hz (100 ms) to 500 Hz (2 ms)*
7	*0*	*No data*	*ESC's Output Throttle*	*1*	*Receive ESC Output Throttle (%)* *10 Hz (100 ms) to 500 Hz (2 ms)*
8	*0*	*No data*	*MCU ID*	*8*	*Receive MCU ID Information*
9	*8*	*MCU ID*	*Node address*	*1*	*Update Node Address*
10	*2*	*Start ESC Enumeration*	*MCU ID*	*8*	*Start ESC Enumeration (send MCU ID back when motor is rotated)*
11	*0*	*No data*	*ESC Voltage, Current, RPM, Temperature, Warn*	*8*	*Receive ESC's Voltage, Current, RPM, Temperature, and Warning (10-500 Hz)*
32	*0*	*No data*	*The Shutdown Procedure is Invoked*	*1*	*Immediately turn off ESC controls (stop all MOSFET channels) and remain idle until restart command is issued*
33	*0*	*No data*	*The Restart Procedure is Invoked*	*1*	*Restart the ESC controls and re-enable arming and throttle control*
34	*0*	*No data*	*Warnings and Errors*	*1*	*Receive ESCs Warning and Errors*

Figure 10: CAN Bus Message Table

Note: Figures left empty are not applicable.

0: Get ESC information (U64)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x00

ESC TO MASTER (Answer)

Extended Frame ID				Data				
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4
0x00	0x02	0x00	0x00	FW0	FW1	HW0	HW1	Mode

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 00	00 00 00 00 00 00 00 00		Master sends request
ESC to Master:	00 02 00 00	0C 00 01 00 46		ESC replies with firmware 0x 0C 00 and hardware 01 00

1: Set PWM (U16)

MASTER TO ESC:

Extended Frame ID				Data	
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x00	0x02	0x01	0~0x08	0~0x98

ESC TO MASTER(Answer)

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x02	0x00	0x01

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 01	05 FC 00 00 00 00 00 00	1532	Send 1532 µs to the ESC
ESC to Master:				The ESC sets the throttle to the specified pulse width.

2: Get Voltage (U16)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x02

ESC TO MASTER(Answer)

Extended Frame ID				Data	
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x02	0x00	0x02	V0	V1

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 02	00 00 00 00 00 00 00 00		Requests ESC voltage.
ESC to Master:	00 02 00 02	06 2D	1581	1581 / 100 = 15.81 V

3: Get Current (U16)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x03

ESC TO MASTER(Answer)

Extended Frame ID				Data	
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x02	0x00	0x03	C0	C1

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description: Current = #/100
Master to ESC:	00 00 02 03	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 03	00 4C	76	76 / 100 = 0.76 A
		00 9D	157	1.57 A
		00 A2	162	1.62 A
		00 AA	170	1.70 A
		00 AE	174	1.74 A

4: Get RPM (U16)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x04

ESC TO MASTER(Answer)

Extended Frame ID				Data	
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x02	0x00	0x04	RPM0	RPM1

Example:

Message Transmission:	Frame Id:	Data:	Data Decimal Equivalent:	Data Description: [Mechanical RPM = eRPM * 60 * 2 / # of magnetic motor poles]
Master to ESC:	00 00 02 04	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 04	01 DD	477	477 * 60 * 2 / 22 = 2,601 rpm
		02 61	609	3,321 rpm
		02 94	660	3,600 rpm
		02 EF	751	4,096 rpm
		02 FA	762	4,156 rpm

5: Get Temperature (U8)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x05

ESC TO MASTER(Answer)

Extended Frame ID				Data
Priority	Source Address	Destination Address	Object Address	Byte0
0x00	0x02	0x00	0x05	Temp0

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 05	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 05	1E	30	30 degrees Celsius

6: Get Input Throttle (U16)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x06

ESC TO MASTER(Answer)

Extended Frame ID				Data	
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x02	0x00	0x06	IT0	IT1

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 06	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 06	05 1C	1308	1308 µs
		05 E6	1510	1510 µs
		06 B2	1714	1714 µs
		07 7D	1914	1914 µs
		08 4B	2123	2123 µs

7: Get Output Throttle (U16)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x07

ESC TO MASTER(Answer)

Extended Frame ID				Data
Priority	Source Address	Destination Address	Object Address	Byte0
0x00	0x02	0x00	0x07	OT0

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 07	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 07	02 22	34	34 %
		02 37	55	55 %
		02 4C	76	76 %
		02 61	97	97 %
		02 64	100	100 %

8: Get All MCU IDs (U64)

MASTER TO ESC:

Extended Frame ID			
Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x01	0x08

ESC TO MASTER(Answer)

Extended Frame ID				Data							
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x00	0x02	0x00	0x08	MCU ID0	MCU ID1	MCU ID2	MCU ID3	MCU ID4	MCU ID5	MCU ID6	MCU ID7

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 01 08	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 08	7d da 57 18 20 33 33 37		All ESCs respond with their MCU ID
ESC to Master:	00 03 00 08	5b 77 57 13 20 33 33 37		All ESCs respond with their MCU ID

9: Update Node Address

MASTER TO ESC:

Extended Frame ID				Data							
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x00	0x00	XX	0x09	MCU ID0	MCU ID1	MCU ID2	MCU ID3	MCU ID4	MCU ID5	MCU ID6	MCU ID7

XX = NEW NODE ADDRESS

ESC TO MASTER(Answer)

Priority	Source Address	Destination Address	Object Address	Byte0
0x00	XX	0x00	0x09	New Node Address

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 09	7d da 57 18 20 33 33 37		
ESC to Master:	00 02 00 09	02		The ESC responds with 2 (its new node address)

10: Start ESC Enumeration

MASTER to ESC:

Priority	Source Address	Destination Address	Object Address	Byte0	Byte1
0x00	0x00	0x01	0x0A	0~0xFF	0~0xFF

ESC to MASTER(Answer)

Extended Frame ID				Data							
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x00	0x02	0x00	0x0A	MCU ID0	MCU ID1	MCU ID2	MCU ID3	MCU ID4	MCU ID5	MCU ID6	MCU ID7

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 01 0a	27 10 00 00 00 00 00 00	10,000	Broadcast Enum message for 10,000 msec
ESC to Master:	00 02 00 0a	7d da 57 18 20 33 33 37		When the motor is rotated, the ESC responds with its MCU ID

11: Get Voltage, Current, RPM, and Temperature, Warning

MASTER to ESC:

Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x0B

ESC to MASTER(Answer)

Extended Frame ID				Data							
Priority	Source Address	Destination Address	Object Address	Byte0	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7
0x00	0x02	0x00	0x0B	V0	V1	C0	C1	RPM0	RPM1	TEMP0	WARN0

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 01 0b	00 00 00 00 00 00 00 00		Broadcast Enum message for 10,000 msec
ESC to Master:	00 01 00 0b	05 E1 00 95 02 10 1F 00	05 E1 represents 1505 00 95 represents 149 02 10 represents 528 1F represents 31 00	15.05 V 1.49 A (528 * 60 * 2 / 22) = 2,880 rpm 31 deg Cel No Warning Messages

32: Turn off ESC (Shut down MOSFET channels)

MASTER to ESC:

Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x20

ESC to MASTER(Answer)

Extended Frame ID				Data
Priority	Source Address	Destination Address	Object Address	Byte0
0x00	0x02	0x00	0x20	0x01

Example:

Message Transmission:	Frame ID:	Data:	Data Decimal Equivalent:	Data Description:
Master to ESC:	00 00 02 20	00 00 00 00 00 00 00 00		
ESC to Master:	00 02 00 20	01	1	Immediately turn off ESC controls

33: Restart ESC

MASTER to ESC:

Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x21

ESC to MASTER(Answer)

Extended Frame ID				Data
Priority	Source Address	Destination Address	Object Address	Byte0
0x00	0x02	0x00	0x21	0x01

Example:

	Frame Id:	Data:	Description:
Master to ESC:	00 00 02 21	00 00 00 00 00 00 00 00	
ESC to Master:	00 02 00 21	01	Restart the ESC controls and re-enable arming and throttle control

34: Get Warning Signals and Errors

MASTER to ESC:

Priority	Source Address	Destination Address	Object Address
0x00	0x00	0x02	0x22

ESC to MASTER(Answer)

Extended Frame ID				Data
Priority	Source Address	Destination Address	Object Address	Byte0
0x00	0x02	0x00	0x22	0x00

Example:

	Frame Id:	Data:	Description:
Master to ESC:	00 00 02 22	00 00 00 00 00 00 00 00	
ESC to Master:	00 02 00 22	00	There are no warnings or errors

Warning Signals and Errors

The warning signals and errors can be viewed through object address 11 or 34. The warning signals and errors are sent in 1 byte, if a protection occurs the corelating bit will be set:

- Stall Protection: BIT 0 DEC 1
- Over Temperature: BIT 1 DEC 2
- Overload Protection: BIT 2 DEC 4
- Over Voltage: BIT 3 DEC 8
- Low Voltage: BIT 4 DEC 16
- Voltage Cutoff (if enabled): BIT 5 DEC 32

If multiple errors occur, the number that appears will be the summation. For example, if Stall Protection, Temperature Protection, and Overload Protection occur the error code will be 7. For more information please refer to the table shown below:

	BIT 7	BIT 6	BIT 5 Voltage Cutoff (if enabled)	BIT 4 Low Voltage	BIT 3 Over Voltage	Bit 2 Overload Protection	Bit 1 Temp Protection	Bit 0 Stall Protection
Bit 7								
Bit 6								
Bit 5 Voltage Cutoff			Binary: 0010 0000 Decimal: 32	Binary: 0011 0000 Decimal: 48	Binary: 0010 1000 Decimal: 40	Binary: 0010 0100 Decimal: 36	Binary: 0010 0010 Decimal: 34	Binary: 0010 0001 Decimal: 33
Bit 4 Low Voltage			Binary: 0011 0000 Decimal: 48	Binary: 0001 0000 Decimal: 16	Binary: 0001 1000 Decimal: 24	Binary: 0001 0100 Decimal: 20	Binary: 0001 0010 Decimal: 18	Binary: 0001 0001 Decimal: 17
Bit 3 Over Voltage			Binary: 0010 1000 Decimal: 40	Binary: 0001 1000 Decimal: 24	Binary: 0000 1000 Decimal: 8	Binary: 0000 1100 Decimal: 12	Binary: 0000 1010 Decimal: 10	Binary: 0000 1001 Decimal: 9
Bit 2 Overload Protection			Binary: 0010 0100 Decimal: 36	Binary: 0001 0100 Decimal: 20	Binary: 0000 1100 Decimal: 12	Binary: 0000 0100 Decimal: 4	Binary: 0000 0110 Decimal: 6	Binary: 0000 0101 Decimal: 5
Bit 1 Temp Protection			Binary: 0010 0010 Decimal: 34	Binary: 0001 0010 Decimal: 18	Binary: 0000 1010 Decimal: 10	Binary: 0000 0110 Decimal: 6	Binary: 0000 0010 Decimal: 2	Binary: 0000 0011 Decimal: 3
Bit 0 Stall Protection			Binary: 0010 0001 Decimal: 21	Binary: 0001 0001 Decimal: 17	Binary: 0000 1001 Decimal: 9	Binary: 0000 0101 Decimal: 5	Binary: 0000 0011 Decimal: 3	Binary: 0000 0001 Decimal: 1

Figure 11: Warning Messages

FAQ

How do I reset my ESC ID?

Currently the ESC ID can be reset to 0x01 with CAN Message 9 or through the KDE Device Manager by pressing the DEFAULT SETTINGS button or by updating firmware (factory reset).

What does the Extended Frame ID represent?

The first byte represents the priority of the CAN message. The second byte represents the source ID. The third byte represents the destination ID. The last byte represents the object address (function call).

What does the data field in a CAN message represent?

The data field represents additional information passed with the CAN message. This additional information can be sent from the master to an ESC and from an ESC to the master.

What is the CAN bus master?

The CAN bus master is the device that controls all of your ESCs. This is typically a flight controller but can also be a CAN Bus analyzer or sniffer.

What is the MCU ID?

The microcontroller unit the UVC series ESCs use is a STM32 CPU. Each MCU has a unique identifier referred to as the MCU ID.

How do I test that all CAN bus connections are wired properly?

To detect all ECSs on network, send a CAN message to frame id: 00 00 01 08. All ESCs will respond with their MCU ID regardless of their ESC ID.

Do I need a terminating resistor on the last node?

Yes, a 120-ohm terminating resistor is required on the last node on all CAN bus setups.

What is the difference between the node ID, node address, and ESC ID?

These all mean the same thing; they represent the unique identifier of a node on the CAN bus network.

CAN bus messages are not working?

Please verify your ESCs are connected to power and are wired correctly.

Is there a time limit on the ESC enumeration message?

The maximum enumeration time limit is 131 seconds.

Why is the limit 131 seconds?

The ESC enumerates its value every two milliseconds. This value is stored in a 16 bit value. (2ms * 2^16 = 131 seconds)

What is the ESC ID if it is not assigned?
ESCs with an unassigned ID (initialized ID) have an ID of 0x01.

What happens if an ESC is not assigned an ID?
ESCs require unique IDs for direct communication with the master. Messages sent with ID 0x01 are broadcast messages sent to all ESCs on a network. The default (unassigned ID) for an ESC is 0x01, and any attempt to direct message this ESC will be broadcasted to all ESCs.

How do broadcast messages work?
All ESCs on the network will respond to messages sent to Extended Frame ID Destination 0x01.

What happens if I assign two or more ESCs to the same ID?
All ESCs receive the message and a potential CAN bus error can occur if the ESCs reply to the message with different data. If they reply with the same data you will only see one message. To avoid this, set the ESCs to have different CAN IDs (through the KDE Device Manager or through ESC Enumeration) or change the data sent back within the CAN message.

I'm only getting 1 CAN message back from 2 ESCs, what am I doing wrong?
Check the ESC IDs to ensure they are not the same. If both ESCs have the same CAN ID and send the same data in a CAN response message, you will only see one message. To avoid this, set the ESCs to have different CAN IDs (through the KDE Device Manager or through ESC Enumeration) or change the data sent back within the CAN message.

How many ESCs can work on the CAN bus network?
Currently KDECAN supports 8 ESCs. Future firmware updates will support extended functions.

Primary Throttle through CAN bus

KDECAN primary throttle control through CAN bus will be a feature available in future firmware updates.

KDECAN Brand Guidelines

Brand guidelines can be found on the KDE Direct website here.

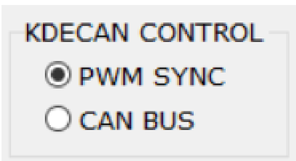
Troubleshooting

- Try sending frames with the extended frame format. Typically, the format needs to be changed from standard to extended (CAN 2.0 B).
- The baud rate is 1Mb/s by default.
- The ESC must be connected to a power supply or LiPo battery. Make sure USB is disconnected.
- Try power cycling (disconnect and reconnect LiPo battery).
- The ESC is correctly wired to the device sending CAN bus messages.
- The ESC will only reply to messages where the destination byte is the ESCs node ID or messages that are broadcasted.
- To use KDECAN, the ESC must be operating on firmware version D460117 or above.

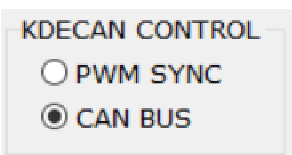
KDECAN Throttle Control BETA Setup:

Warning: CAN bus throttle control is still in BETA and is at your own risk.
This functionality is still currently being tested.
The setup described below is provided as an example.

When KDECAN Control is set to PWM SYNC: Primary throttle control is done through PWM and secondary throttle control is done through CAN bus. In other words, throttle control through CAN bus will work as a backup (if the PWM signal is lost or disconnected) it will use CAN bus. But arming off of CAN bus alone is disabled.



When KDECAN Control is set to CAN bus: Primary throttle control is done through CAN bus - arming and full throttle control will work through KDECAN.



Initial Setup:

First, we recommend updating the UVC ESC firmware to D460224.dfu or above and use KDE Device Manager V1.36.1 or above. To update the ESC firmware, press the Update Firmware button and select the firmware file.

For CAN bus throttle control, we recommend starting with the following ESC settings:

- Throttle Calibration Mode: Range (1100 – 1940)
- ESC ID (KDECAN): FIXED ID 02
- KDECAN CONTROL: CAN BUS
- The Motor Edition should also be selected (if available) to improve motor commutation.
- All other settings left as default

For CAN bus throttle control to work, the ESC expects throttle messages to be sent in 20-2ms intervals to achieve 50-500Hz. This is required because the frequency directly effects the acceleration rate.

Next, connect the ESC to a motor, CAN bus master (device sending messages), and connect the ESC to a LiPo battery or power supply. After connecting the ESC to power and the motor, the motor will start beeping (indicating it is awaiting an arming signal).

You can then send a CAN message to arm the motor:

Frame ID: (ESC ID2 throttle)	Data: (arming pulse 1100µs)
00000201	044c000000000000

Upon receiving an arming frame, the beeping will stop. At this point the ESC may not have completed arming. The ESC requires additional arming messages (for a few seconds) to complete its arming procedure. For example, you can arm the motor by sending 600 arming messages at 5ms (200Hz). The motor will start beeping to indicate the number of cells for the LiPo battery / voltage connected.

You can then send CAN messages to spin the motor:

Frame ID: (ESC ID2 throttle)	Data: (spinning pulse 1175µs)
00000201	0497000000000000

The CAN frames to spin the motor should also be sent at 20-2ms, 600 messages are a good amount to start with. An example is provided here.

Additional information:

- Using lower baud rates may result in data loss, for example:
  A transmission frame is (64 bits + 16 bits data for throttle) * 400 hz * 4 ESCs = 128,000 bps.
  At 100k baud, data would be lost in this example. This is also assuming you are only using 1 bus.
  Lowering the frequency or increasing the baud rate would solve this problem.

- For CAN bus, the microseconds (µs) represents the throttle.
  For throttle calibration RANGE, 1100us represents 0% and 1940us represents 100%.
  You can easily convert the percentage throttle to microseconds and vice versa with the formulas below:
  Output throttle = 1100 + 8.4 * percentage — example: 1100 + 8.4 * 50 = 1520µs
  Percentage = (output throttle - 1100) / 8.4 — example: (1520 - 1100) / 8.4 = 50%