

**Motion Protocol CPR-CAN-V2  
for  
Closed Loop motor controller**

Version 1.5  
July 18, 2022  
© Commonplace Robotics GmbH

## Contents

<b>1 About this document</b>	<b>2</b>
1.1 Contact	2
1.2 Target group and qualification	3
1.3 Symbols Used	3
<b>2 Introduction</b>	<b>4</b>
2.1 Brief description of the protocol	4
2.2 Communication procedure	4
2.3 Communication parameters	5
2.4 Configuration software ModuleCtrl	5
2.5 Demo software with sources	5
<b>3 CAN Messages</b>	<b>6</b>
3.1 Motion commands	6
3.1.1 Position CMD 0x14	6
3.1.2 Velocity CMD 16bit 0x25	6
3.1.3 Torque CMD 0x16	7
3.2 Response to motion commands	8
3.3 Process CMD 0x01	10
3.3.1 Reset Error 0x06	10
3.3.2 Set position to 0 0x08	10
3.3.3 Enable Motor 0x09	11
3.3.4 Disable Motor 0x0A	11
3.3.5 Referencing 0x0B	12
3.3.6 Rotor alignment 0x0C	12
3.3.7 Ping 0xCC	13
3.3.8 EEPROM write enable 0xCD	13
3.4 Parameter V2 0x94 & 0x96	14
3.4.1 Save parameters 0x94	14
3.4.2 Read parameters 0x96	15
3.5 Special CAN messages	15
3.5.1 Startup message	16
3.5.2 Environmental parameters 0x12	16
3.5.3 Extended error messages	16
3.5.4 Output encoder position	17

## 1 About this document

### 1.1 Contact

Commonplace Robotics GmbH  
Gewerbepark 9-11  
Im Innovationsforum  
D-49143 Bissendorf

Tel.: +49(0)5402 / 968929-0  
Fax: +49(0)5402 / 968929-9  
E-Mail: [info@commonplacrobotics.de](mailto:info@commonplacrobotics.de)

Internet: <https://cpr-robots.com>



## 1.2 Target group and qualification

This protocol documentation is aimed at technically trained professionals such as:

- Programmers
- Development Engineers

Experience with the following is required:

- Hardware-related programming e.g. with C++
- Knowledge of the CAN fieldbus
- ESD procedures when working with electronics

This documentation supplements the product documentation of the motor controllers. Knowledge of this documentation is assumed.

## 1.3 Symbols Used

All notes in this document follow a consistent form and are structured according to the following classes.



**The WARNING notice alerts the reader to possible dangerous situations.**

Disregarding a warning can **possibly** result in moderate injury to the user.

- Within a warning, this describes ways to avoid hazards.



**This note indicates possible incorrect operation of the product.**

Failure to comply with this notice may **possibly** result in damage to this product or other products.

## 2 Introduction

### 2.1 Brief description of the protocol

The CPR-CAN-V2 protocol is used for communication between a controller and the drive modules. The controller sends position setpoints or speeds to the motor controllers via the CAN field bus, and these respond with their current status.

The calculation of acceleration ramps and the interpolation between target points takes place in the controller, not in the motor controller. The controller sends the target position to the motor controller at short intervals of 10 to 50 ms.

The operation is thus comparable to the CANopen operation mode IPO or CSP. An operation mode like CANopen ProfilePosition is not supported.

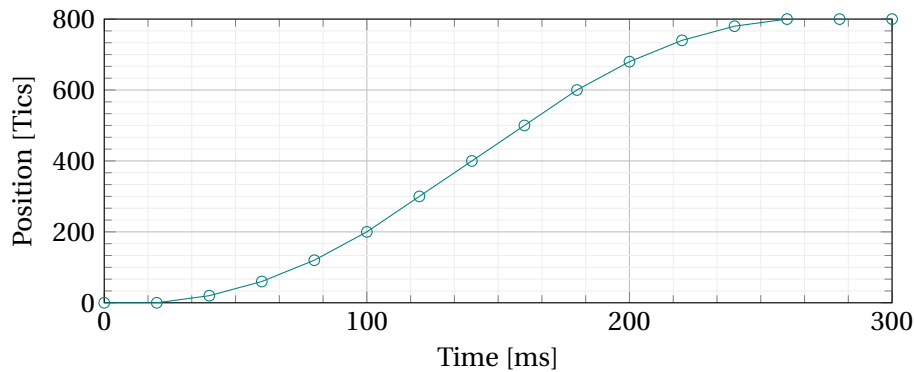
In addition to transmitting the position or speed specifications, the controllers can be parameterized via the protocol.

### 2.2 Communication procedure

This example is intended to illustrate the communication sequence in positioning mode.

1. Starting the motor controller
2. Start of a cyclically called function which sends a position specification every 20 ms (chapter 3.1.1)
3. Aligning the positions of motorcontroller and master software. There are two options:
  - Reset the current position to 0 (Chapter 3.3.2). Initialization of the setpoint position in the master software also with 0.  
This option is recommended for the first tests.
  - Reading the current motor position from the CAN message answers (Chapter 3.2) and using this position as current setpoint position in the master software.  
This option is recommended for real applications.
4. Reset error on the axis. (Chapter 3.3.1)
5. Enable axis (chapter 3.3.3) so that error byte in response shows 0x00 (chapter 3.2).
6. Slowly change the target position in the cyclically sent position specification. The axis now follows the preset.

The following graphic shows the movement from 0 to 800 tics. The interpolation with acceleration and deceleration ramps takes place in the controller, which sends the target positions marked by points cyclically to the motor controller.



### 2.3 Communication parameters

CAN baud rate                    500kBaud  
Communication frequency    20 bis 100 hz

### 2.4 Configuration software ModuleCtrl

The ModuleControl configuration software is available to the user for simple control of all axis functions. This supports the bootloader and the parameter interface.

[https://wiki.cpr-robots.com/index.php/Config\\_Software\\_ModuleCtrl](https://wiki.cpr-robots.com/index.php/Config_Software_ModuleCtrl)



### 2.5 Demo software with sources

For a quick start in developing your own control, a sample program is available on GitHub:

<https://github.com/CommonplaceRobotics/CANV2ProtocolDemoClient>



Included is a VisualStudio2019 project in CSharp that connects to and drives a motor controller. All source files are included, these can be used as a starting point for your own development. To run the code and test your own applications you need the following components:

1. Visual Studio 2019 or later
2. PCAN-USB CAN to USB Interface from PEAK-System Technik GmbH
3. Motor controller with motor
4. Connection cable depending on motor controller

## 3 CAN Messages

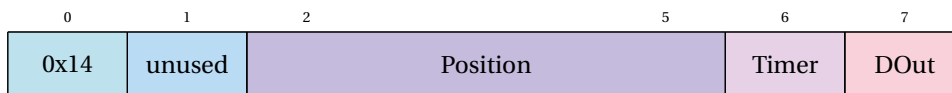
### 3.1 Motion commands

Motion commands fall into the category of cyclic commands. These commands must be sent repeatedly from the motion master to the motor controller in a fixed time. The basic cycle is 50ms. If the motion master does not keep to this time and exceeds a specified time, the motor controller goes into an error state. When changing between two motion commands, the axis interrupts the execution and stops the motion. Only after reset and enable a new movement in the new mode is possible.

#### 3.1.1 Position CMD 0x14

The Position CMD puts the axis into the positioning mode. This mode is comparable with the CANopen operating mode IPO or CSP. The specification of the position is done in Encodertics. The motor controller tries to reach the target position as fast as possible, no path with target speed and acceleration ramps is calculated. If the target position is too far away from the current position, this can lead to unexpected behavior or a shutdown.

##### 1. Message to the motor controller



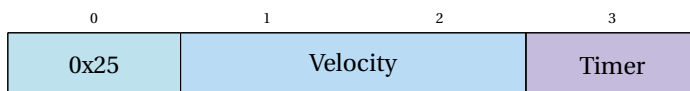
- Position 32-bit position of the motor in [Tics]
- Timer Value for checking the correct order of messages
- DOut If the motor controller has outputs, they can be set here

##### 2. Response from motor controller Standard response CANV2 (chapter 3.2)

#### 3.1.2 Velocity CMD 16bit 0x25

This mode is comparable to the CANopen operating mode ProfileVelocity.

##### 1. Message to the motor controller



- Velocity Preset speed with 16bit in [RPM] as int16
- Timer Value for checking the correct order of messages

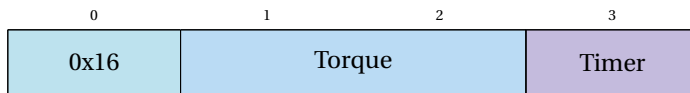
##### 2. Response from motor controller Standard response CANV2 (chapter 3.2)



### 3.1.3 Torque CMD 0x16

This mode is comparable to the CANopen operating mode ProfileTorque.

1. Message to the motor controller



Torque Specification of the Desired torque in engineering units between [-1024-1024]

Timer Value for checking the correct order of messages

2. Response from motor controller  
Standard response CANV2 (chapter 3.2)

### 3.2 Response to motion commands

The motor controllers respond to all motion commands with a uniform response.

CAN-ID = BoardID + 1



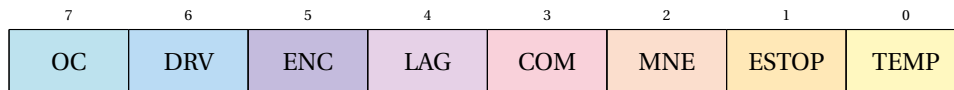
errorCode    Error byte see below for more information

Position     32-bit current axis position in [Tics].

Current      Current RMS current in [mA]

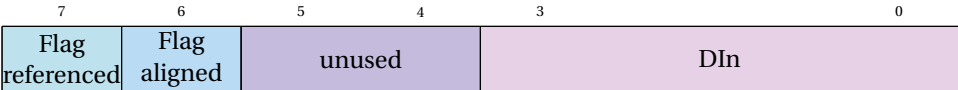
DIn + Flags    The inputs of the motor controllers, as well as the flags are explained below

The motor controller sends an error byte in every CANV2 response. This can define various error states.



TEMP	Over temperature	The temperature of the motor controller or the motor is above the defined value in the parameters.
ESTOP	Emergency stop / no voltage	The voltage at the motor controller is below the set limit value. This may indicate a defective fuse or emergency stop.
MNE	Motor not activated	The movement of the motor is not enabled, it is not in control.
COM	Communication failure	The motor controller requires CAN messages at regular intervals. If the distance between the messages is too large or the messages do not arrive, the motor controller stops the movement.
LAG	Following error	The motor controller monitors the following error, if this is greater than the value set in the parameters, the motor controller stops the movement.
ENC	Encoder error	The motor controller has detected an encoder error. Errors can be triggered by both the motor or the output encoder.
DRV	Driver error	A driver error can have various causes. One possible cause is exceeding the maximum speed from the parameters. With the closedloop motor controllers, the error also occurs with problems with the initial rotor position.
OC	Over current	The RMS current in the motor controller was above the allowed value in the parameters.

The byte field for the inputs as well as the referencing flags of the motor controller is differentiated in the nibble. The low nibble contains the inputs, the high nibble the referencing flags.



### 3.3 Process CMD 0x01

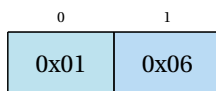
This category includes control instructions which are not sent cyclically to the board.

#### 3.3.1 Reset Error 0x06

This message resets all errors on the board.

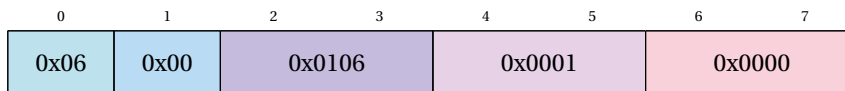
Depending on the board version the board has irreversible errors, these cannot be reset. An example is an error in the motor wiring.

1. Motor controller message



2. Response from motor controller

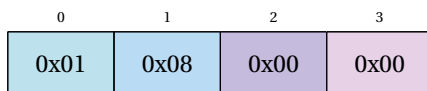
CAN-ID = BoardID + 2



#### 3.3.2 Set position to 0 0x08

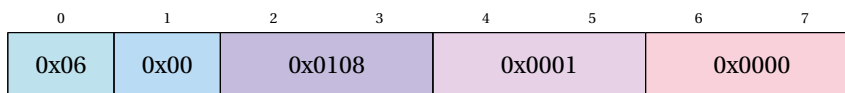
This message must be sent to the controller 2 times within 20 ms. This command sets the Current to 0. The controller then goes into an error state.

1. Motor controller message



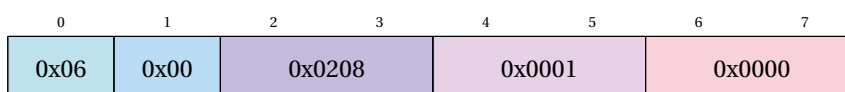
2. 1. Response from motor controller CMD received

CAN-ID = BoardID + 2

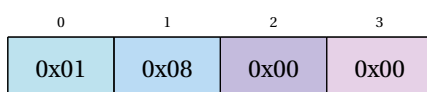


3. 2. Response from motor controller CMD ok

Only sent if CMD is ok. CAN-ID = BoardID + 2

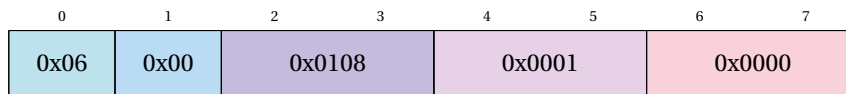


4. 2. Motor controller message



## 5. 3. Response from motor controller CMD received

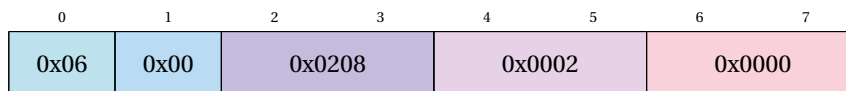
CAN-ID = BoardID + 2



## 6. 4. Response from motor controller CMD ok

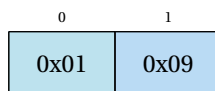
Only sent if CMD is ok. After this response, the motor controller has set the position to 0.

CAN-ID = BoardID + 2

**3.3.3 Enable Motor 0x09**

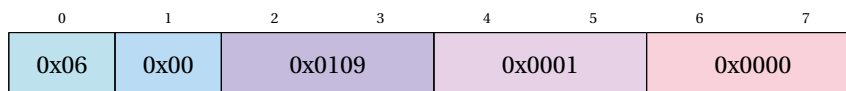
Activates the motor. Only possible if no error is present.

## 1. Motor controller message



## 2. Response from motor controller

CAN-ID = BoardID + 2

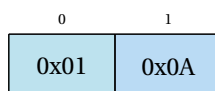
**3.3.4 Disable Motor 0x0A**

Deactivate motor. The power output stage switches off the current to the motor.

**No Safe Stop!**

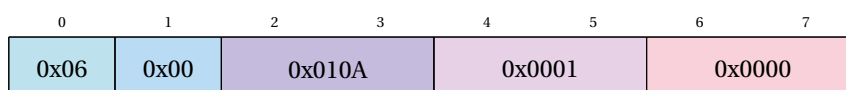
This function does not represent a safe stop. It is possible that the motor continues to be supplied with voltage.

## 1. Motor controller message



## 2. Response from motor controller

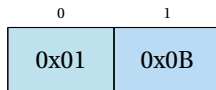
CAN-ID = BoardID + 2



### 3.3.5 Referencing 0x0B

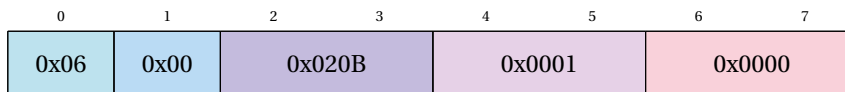
Request for referencing. The message must be sent to the motor controller several times.

1. Motor controller message

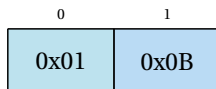


1. Response from motor controller

CAN-ID = BoardID + 2

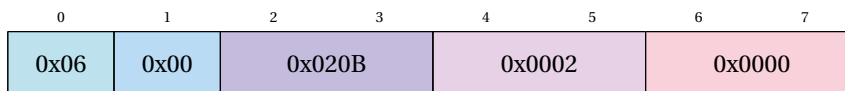


2. Motor controller message



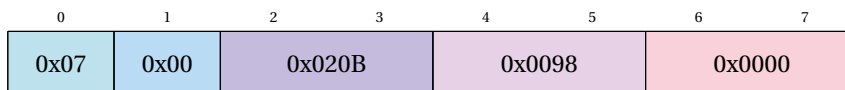
2. Response from motor controller

CAN-ID = BoardID + 2



5. Error from motor controller Referencing active. Referencing of the axis is already active.

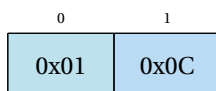
CAN-ID = BoardID + 2



### 3.3.6 Rotor alignment 0x0C

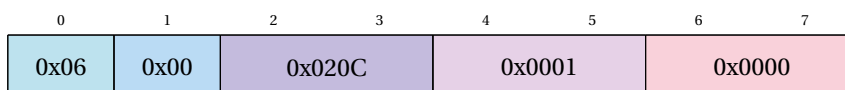
or the operation of closed loop motor controllers it is necessary that they know the initial rotor orientation. This can be recalibrated by the following command. This command must also be sent twice.

1. Motor controller message

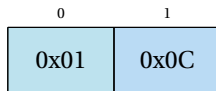


1. Response from motor controller

CAN-ID = BoardID + 2

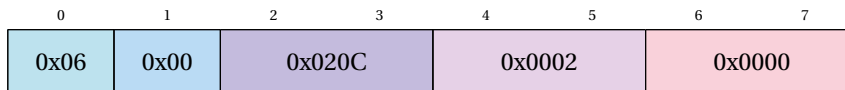


## 3. 2. Motor controller message



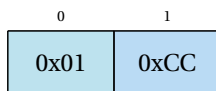
## 4. 2. Response from motor controller

CAN-ID = BoardID + 2

**3.3.7 Ping 0xCC**

The controller responds to this message with the startup message.

## 1. Motor controller message

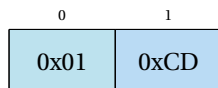


## 2. Response from motor controller

The CAN message sent corresponds to the startup message. (chapter 3.5.1)

**3.3.8 EEPROM write enable 0xCD**

Before parameters can be stored on the motor controller, the EEPROM must be enabled with this message. The motor controller does not respond to this message.



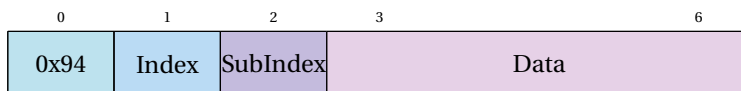
### 3.4 Parameter V2 0x94 & 0x96

The following parameter interface is available for the closed loop motor controller. This uses a structure consisting of index and subindex.

The description of the corresponding parameters with index and subindex can be found in the product documentation of the motor controllers.

#### 3.4.1 Save parameters 0x94

1. Enabling the EEPROM (chapter 3.3.8)
2. Message to motor controller

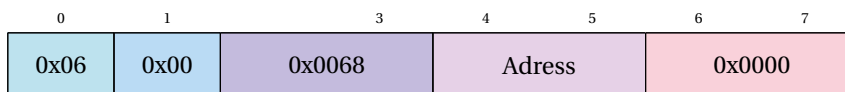


Index      category of the parameter

SubIndex   parameter in the category

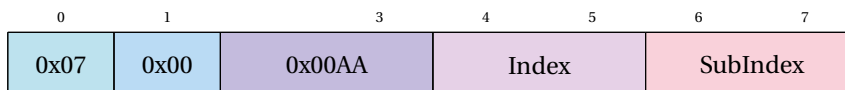
Data      32bit value of the data

3. Message from motor controller: Transmission ok  
CAN-ID = BoardID + 2



Address    Calculated EEPROM Address

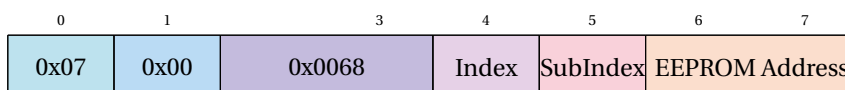
4. Error from motor controller: EEPROM not enabled  
The EEPROM was not enabled before, see chapter 3.3.8  
CAN-ID = BoardID + 2



Index      category of the parameter

SubIndex   parameter in the category

5. Error from motor controller: Not stored  
The data could not be stored in the EEPROM.  
CAN-ID = BoardID + 2





Index                    category of the parameter  
 SubIndex                parameter in the category  
 EEPROM Address    EEPROM Address to be written

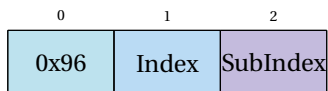
### 3.4.2 Read parameters 0x96



#### Interruption of the cyclic messages

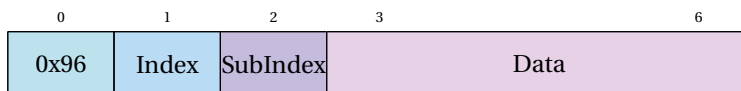
For transmitting the parameters the cyclic CAN messages must be interrupted. The response for reading the parameters and the standard response (chapter 3.2) share the response ID BoardID + 1.

#### 1. Message to motor controller



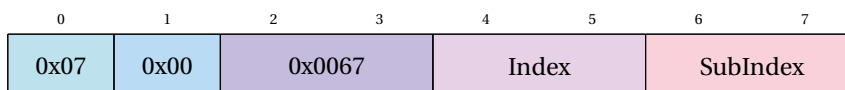
Index                    category of the parameter  
 SubIndex                parameter in the category

#### 2. Message from motor controller: Transmission ok CAN-ID = BoardID + 1



Index                    category of the parameter  
 SubIndex                parameter in the category  
 Data                    32bit value of the data

#### 3. Error from motor controller: Error while reading CAN-ID = BoardId + 2



Index                    category of the parameter  
 SubIndex                parameter in the category

## 3.5 Special CAN messages

This section contains the description of special CAN messages, e.g. the motor controllers send various status information cyclically without request.

### 3.5.1 Startup message

When the motor controllers are started, they send a defined message. This message helps to identify the motor controller.

CAN-ID = BoardID + 2



HW-Ident Board identification

Version Major and minor version of the firmware

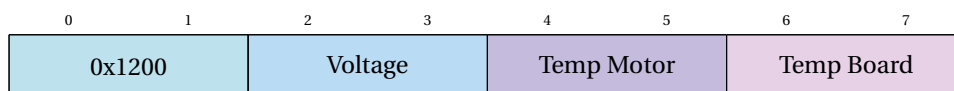
The following values are implemented for the identification of the board:

DIN-RAIL CL BLDC	0x50	Closed Loop BLDC motor controller for DIN rail mounting with CPR backplane bus
DIN-RAIL CL STEPPER	0x51	Closed Loop stepper motor controller for DIN rail mounting with CPR backplane bus
ON-AXIS CL BLDC	0x52	Closed Loop integrated motor controller for on axis mounting
DIN-RAIL CL DC	0x53	Closed Loop DC motor controller for DIN rail mounting with CPR backplane bus
dsPIC33 Boot loader	0x89	Boot loader for motor controller dsPIC33

### 3.5.2 Environmental parameters 0x12

The motor controller sends a message once per second with the current voltage, as well as the motor and board temperature.

CAN-ID = BoardID + 3



Voltage Current voltage at motor controller [mV]

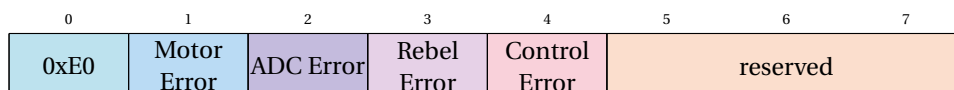
Temp Motor Motor Temperature [m°C]

Temp Board Temperature Motor Controller [m°C]

### 3.5.3 Extended error messages

he motor controller sends a message with the detailed status information once per second. For an evaluation of the individual bytes please use the configuration software ModuleControl.

CAN-ID = BoardID + 2



Motor Error	Byte with errors of the motor
ADC Error	byte with errors of the AD converter
Rebel Error	byte with rebel encoder errors
Controll Error	byte with control errors

### 3.5.4 Output encoder position

The Igus Rebel is equipped with on-axis motor controllers, the axes have an output encoder. If the output encoder is parameterized, the motor controller sends the output position once per second.

CAN-ID = BoardID + 2

0	1	2	3	4	5	6	7
0xEF	0x0000		0x7E	Encoder Position			

Encoder Position    Position of the output drive in  $\frac{Degree}{100}$