**dTI Inverter CAN MANuaL**

FOR HV-500/HV-550/HV-850

Version 2.5

**For developers and integrators**

**For inverter related informations and CAN Pinout please refer to** the controller Technical description. Upon any concerns or questions, please contact us at [info@drivetraininnovation.com](mailto:info@drivetraininnovation.com?subject=[CAN]%20Question%20about%20DTI%20Inverter%20CAN%20Interface)

**Table of contents**

[1. History and related documents 3](#_Toc134688765)

[1.1 Document history 3](#_Toc134688766)

[2. Overview 4](#_Toc134688767)

[2.1 CAN Speed 4](#_Toc134688768)

[2.2 Controller ID (Node ID) 5](#_Toc134688769)

[2.3 ID Ranges 5](#_Toc134688770)

[2.3.1 Standard ID operation 5](#_Toc134688771)

[2.3.2 Extended ID operation 5](#_Toc134688772)

[2.4 General message overview 6](#_Toc134688773)

[3. Transmitted data by inverter 8](#_Toc134688774)

[3.1 Packet IDs 8](#_Toc134688775)

[3.2 Description of transmitted signals by the inverter 9](#_Toc134688776)

[3.6 Example messages for transmitted data by inverter 17](#_Toc134688777)

[4. Commands to the inverter 19](#_Toc134688778)

[4.1 Command overview 19](#_Toc134688779)

[4.2 Commands can be sent to the inverter 20](#_Toc134688780)

[4.3 Drive enable command 27](#_Toc134688781)

[4.3.1 Operation when analogue inputs are used for control 27](#_Toc134688782)

[4.3.2 Operation when CAN2 interface used for control 27](#_Toc134688783)

[4.4 Broadcast message 28](#_Toc134688784)

[4.5 Example messages for commands 28](#_Toc134688785)

[5. List of tables 34](#_Toc134688786)

# History and related documents

## Document history

|  |  |
| --- | --- |
| 07/2018 | **V1.0** Document created; basic specifications added. |
| 01/2019 | **V1.1** Fixed typos |
| 03/2019 | **V1.2** Fixed typos |
| 04/2019 | **V1.3** Fixed typos |
| 08/2020 | **V2.0** Major revision and actualizing for firmware version 4.1 |
| 10/2020 | **V2.1** Modified IDs for commands. |
| 10/2020 | **V2.11** Fixed MSB / LSB order |
| 10/2020 | **V2.2** Updated document according to bugfix in the HV-500 regarding Standard CAN ID addressing |
| 08/2021 | **V2.3** Added new messages and drive enable control |
| 10/2022 | **V2.4** Rearranged packet IDs, added “set Digital out” command |
| 02/2023 | **V2.4 REV2** Wrong transmitted signal packet IDs fixed |
| 04/2023 | **V2.4 REV3** Added operation range for signals |
| 05/2023 | **V2.5** Added 3 new messages for support multi-controller config. |

* 1. Related documents
* HV-500/HV-550/HV-850 Manual
* DTI CAN Tool manual (to be released)
* DTI Firmware upgrade manual

# Overview

The scope of this manual is the CAN2 of DTI Inverters. The CAN1 is reserved for DTI Can Tool, Firmware updates and DTI devices.

The motor controller includes a software timeout. If the controller does not receive control command for the set time interval (in DTI CAN Tool: APP Settings / General / Timeout), the motor will be in free running and the control will stop until the next command. Therefore, CAN control messages should be sent faster (at least half period of the timeout) than the set time interval for continuous control.

**DTI Inverters are CAN 2.0B Compliant.**

## CAN Speed

The default CAN Speed is 500Kbit/s. This can be adjusted in DTI Can Tool under APP Settings / CAN. This parameter can be adjusted individually for CAN1 and CAN2.

Available speeds:

* 125 Kbit/s
* 250 Kbit/s
* 500 Kbit/s
* 1Mbit/s

## Controller ID (Node ID)

The Controller ID can be set via the DTI CAN tool in APP Settings / CAN.

By default, this ID is the last two digits of the inverter serial number in decimal.

## ID Ranges

The inverter supports both Standard ID and Extended ID CAN message ID formats. The integrator must choose according to their requirement which method is used for addressing. Both addressing method executes the same functions.

### Standard ID operation

With Standard CAN ID configuration the available ID range is between 1 and 30. The ID 31 is reserved for broadcast messages. Messages with node ID 0 will be rejected. This operation uses 5 bits to address nodes on the bus.

For more information, refer to 2.4 chapter.

### Extended ID operation

When using Extended CAN ID, the settable ID range is between 1-254. The ID 255 is reserved for broadcast messages. Messages with node ID 0 will be rejected. In that mode 8 bit is used for addressing nodes on the bus.

For more information, refer to 2.4 chapter.

## General message overview

The messages should contain the data in **BIG ENDIAN** (Motorola) byte order format.

Every message has a **fixed length of 8 bytes**. Bytes which are not used filled with 0xFF.

* + 1. CAN message ID

The standard CAN message ID is represented as below:

|  |  |
| --- | --- |
| **Standard ID (11 bits)**  **MSB:LSB** | |
| Packet ID | Node ID |
| 0x 21 | 4 |
| 10:5 bits (6 bits) | 4:0 bits (5 bits) |
| It can represent a transmitted “AC Current, DC Current” frame by the inverter with node ID 8. | |

The extended CAN message ID is represented as below:

|  |  |
| --- | --- |
| **Extended ID (29 bits)**  **MSB:LSB** | |
| Packet ID | Node ID |
| 0x 00 00 21 | 4 |
| 28:8 bits (21 bits) | 7:0 bits (8 bits) |
| It can represent a transmitted “AC Current, DC Current” frame by the inverter with node ID 34. | |

The following tables represents a general CAN bus message with standard and extended formats:

Table Standard ID example message

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Message ID** | | **Data bytes** | | | | | | | |
| **Position** | 10:5 bits | 4:0 bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Message** | 0x 00 | 4 | 00 | 00 | 24 | 5E | FF | FF | FF | FF |
| **Desc.** | Packet ID | Node ID | ERPM bytes [3:0]  MSB:LSB (9310d) | | | | Not used, filled with FF | | | |

In case of standard ID you must shift the CAN ID by 5 to get the packet id:

Packet ID = (CANID >> 5)

To get the Node ID you must mask it by 0x1F:

Node ID = CANID & 0x1F

Table Extended ID example message

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Message ID** | | **Data bytes** | | | | | | | |
| **Position** | 28:8 bits | 7:0 bits | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **Message** | 0x 00 00 00 | 4 | 00 | 00 | 24 | 5E | FF | FF | FF | FF |
| **Desc.** | Packet ID | Node ID | ERPM bytes [3:0]  MSB:LSB (9310d) | | | | Not used, filled with FF | | | |

In case of extended ID you must shift the CAN ID by 8 to get the packet id:

Packet ID = (CANID >> 8)

To get the Node ID you must mask it by 0xFF:

Node ID = CANID & 0xFF

# Transmitted data by inverter

In order to enable the transmission from the inverter, the following parameters should be enabled in the DTI CAN Tool:

* Enable CAN2: *Enabled*
* Send CAN2 Status: *Enabled*
* CAN2 map version: *V24*
* Specify each message broadcasting period

## Packet IDs

Packet IDs are identifying the information content of each message. In the table below you can find which kind of information available on the CAN bus:

Table : Transmitted messages

|  |  |
| --- | --- |
| **Packet ID** | **Message description** |
| **0x1F** | Control mode, Target Iq, motor position, isMotorStill |
| **0x20** | ERPM, Duty, Input Voltage |
| **0x21** | AC Current, DC Current |
| **0x22** | Controller Temp., Motor Temp., Fault code |
| **0x23** | Id, Iq values |
| **0x24** | Throttle signal, Brake signal, Digital I/Os, Drive enable, Limit status bits, CAN map version |
| **0x25** | configured max AC current, available max AC current, configured min AC current, available min AC current |
| **0x26** | configured max DC current, available max DC current, configured min DC current, available min DC current |

## Description of transmitted signals by the inverter

Table Transmitted CAN bus messages

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/Max data range (post-scale)** | **Min/Max operational range** | **Scale** | **Unit** |
| 0x1F | Length: 8 bytes | | General data 6 |  | | | | |
|  | 0 | 0 | Control mode | Describes the control mode of the inverter:  **1: CONTROL\_MODE\_SPEED**  **2: CONTROL\_MODE\_CURRENT**  **3: CONTROL\_MODE\_CURRENT\_BRAKE**  **4: CONTROL\_MODE\_POS**  **7: CONTROL\_MODE\_NONE**  **0, 5, 6: NOT USED**  Mainly used in multi-controller configurations for internal communication. | 0 - 7 | 0 - 7 | 1 | # |
| 1 - 2 | 8 | Target Iq | The value represent how much Iq current the inverter is targeted to reach. This value excludes limits. For ex. if the target Iq is 50 A and temperature limit is hit, the values keep 50 A in any case, not including the deration of the temp. limit. This is useful in multi-inverter configuration to let know the secondary inverter the target Iq. | -3276,8 - 3276,7 | -850 - 850 | 10 | Apk |
| 3 – 4 | 24 | Motor position | Motor position expressed in degrees. | 0 - 359 | 0 - 359 | 10 | degree |
| 5 | 40 | isMotorStill | Represents if the motor in still position or not. **1: still, 0: rotating** | 0 - 1 | 0 - 1 | 1 | # |
| 6 - 7 | 48 | RESERVED | Filled with FF’s. For future use. | - | - | - | - |
| 0x20 | Length: 8 bytes | | General data 1 |  | | | | |
|  | 0 – 3 | 0 | ERPM | Electrical RPM  Equation: ERPM = Motor RPM \* number of the motor pole pairs. | -2147483648-2147483647 | -120 000 – 120 000 | 1 | ERPM |
| 4 - 5 | 32 | Duty cycle | The controller duty cycle. The sign of this value will represent whether the motor is running(positive) current or regenerating (negative) current. | -3276,8 - 3276,7 | -100 – 100 | 10 | % |
| 6 - 7 | 48 | Input voltage | Input voltage is the DC voltage. | -32768 - 3276,7 | 0 – 1000 | 1 | V |
| 0x21 | Length: 8 bytes | | General data 2 |  | | | | |
|  | 0 - 1 | 0 | AC current | The motor current. The sign of this value represents whether the motor is running(positive) current or regenerating (negative) current. | -3276,8 - 3276,7 | -850 - 850 | 10 | Apk |
| 2 – 3 | 16 | DC current | DC Current: Current on DC side. The sign of this value represents whether the motor is running(positive) current or regenerating (negative) current. | -3276,8 - 3276,7 | -850 - 850 | 10 | Adc |
|  | 4 - 7 | 32 | RESERVED | Filled with FF’s. For future use. | - | - | - | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x22 | Length: 8 bytes | | General data 3 |  | | | | |
|  | 0 – 1 | 0 | Controller temperature | Temperature of the inverter semiconductors. | -3276,8 - 3276,7 | -55 - 200 | 10 | °C |
| 2 – 3 | 16 | Motor temperature | Temperature of the motor measured by the inverter | -3276,8 - 3276,7 | -55 - 200 | 10 | °C |
| 4 | 32 | Fault code | **0x00 : NO FAULTS**  **0x01 : Overvoltage** - The input voltage is higher than the set maximum.  **0x02 : Undervoltage** - The input voltage is lower than the set minimum.  **0x03 : DRV** - Transistor or transistor drive error  **0x04 : ABS. Overcurrent** - The AC current is higher than the set absolute maximum current.  **0x05 : CTLR Overtemp.** - The controller temperature is higher than the set maximum.  **0x06 : Motor Overtemp.** - The motor temperature is higher than the set maximum.  **0x07 : Sensor wire fault** - Something went wrong with the sensor differential signals.  **0x08 : Sensor general fault** - An error occurred while processing the sensor signals  **0x09 : CAN Command error** - CAN message received contains parameter out of boundaries  **0x0A : Analog input error** – Redundant output out of range | 0 | 255 | 1 | # |
| 5-7 | 40 | RESERVED | Filled with FF’s. For future use. | - | - | - | - |
| 0x23 | Length: 8 bytes | | General data 4 |  | | | | |
|  | 0 – 3 | 0 | Id | FOC algorithm component Id. | -2147483,648 - 2147483,647 | -850 – 850 | 100 | Apk |
| 4 – 7 | 32 | Iq | FOC algorithm component Iq | -2147483,648 - 2147483,647 | -850 – 850 | 100 | Apk |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x24 | Length: 8 bytes | | General data 5 |  | | | | |
|  | 0 | 0 | Throttle signal | Throttle signal derived from analog inputs or CAN2 | -128 – 127 | 0 - 100 | 1 | % |
| 1 | 8 | Brake signal | Brake signal derived from analog inputs or CAN2 | -128 - 127 | 0 - 100 | 1 | % |
| 2 | 16 | Digital input 1 | 1: Digital input is active  0: Digital input is inactive | 0 | 1 | 1 | # |
| 17 | Digital input 2 | 0 | 1 | 1 | # |
| 18 | Digital input 3 | 0 | 1 | 1 | # |
| 19 | Digital input 4 | 0 | 1 | 1 | # |
| 20 | Digital output 1 | 1: Digital output is active  0: Digital output is inactive | 0 | 1 | 1 | # |
| 21 | Digital output 2 | 0 | 1 | 1 | # |
| 22 | Digital output 3 | 0 | 1 | 1 | # |
| 23 | Digital output 4 | 0 | 1 | 1 | # |
| 3 | 24 | Drive enable | 1: Drive enabled  0: Drive disabled  Drive can be enabled/disbled by the digital input or/and via CAN2 interface | 0 | 1 | 1 | # |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 4 | 32 | Capacitor temp limit | 1: Capacitor temperature limit active  0: Capacitor temperature limit inactive  The inverter can limit the output power to not to overheat the internal capacitors. (only valid HW version 3.6 or newer) | 0 | 1 | 1 | # |
| 33 | DC current limit | 1: DC current limit active  0: DC current limit inactive | 0 | 1 | 1 | # |
| 34 | Drive enable limit | 1: Drive enable limit active  0: Drive enable limit inactive  Indicates whether the drive enable limitation is active or inactive. Used for software development purposes. **For true indication of the drive state please use byte 3, bit 24 of this message.** | 0 | 1 | 1 | # |
| 35 | IGBT acceleration temperature limit | 1: IGBT acceleration limit active  0: IGBT acceleration limit inactive | 0 | 1 | 1 | # |
| 36 | IGBT temperature limit | 1: IGBT temperature limit active  0: IGBT temperature limit inactive | 0 | 1 | 1 | # |
| 37 | Input voltage limit | 1: Input voltage limit active  0: Input voltage limit inactive | 0 | 1 | 1 | # |
| 38 | Motor acceleration temperature limit | 1: Motor acceleration temperature limit active  0: Motor acceleration temperature limit inactive | 0 | 1 | 1 | # |
| 39 | Motor temperature limit | 1: Motor temperature limit active  0: Motor temperature limit inactive | 0 | 1 | 1 | # |
| 5 | 40 | RPM min limit | 1: RPM min limit active  0: RPM min limit inactive | 0 | 1 | 1 | # |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 | 41 | RPM max limit | 1: RPM max limit active  0: RPM max limit inactive | 0 | 1 | 1 | # |
| 42 | Power limit | 1: Power limit by configuration active  0: Power limit by configuration inactive | 0 | 1 | 1 | # |
| 43 - 47 | RESERVED | Set to 0. | 0 | 1 | 1 | # |
| 6 | 48 | RESERVED | Filled with FF’s. For future use. | - | - | - | - |
| 7 | 56 | CAN map version | Indicates the CAN map version. For ex: 25 -> 2,5 (V2,5) | 0 | 255 | 1 | # |
| 0x25 | Length: 8 bytes | | Configured and available AC currents | | | | | |
|  | 0 – 1 | 0 | Max AC current | Max AC current configured with the DTI CAN tool. | -3276,8 - 3276,7 | 0 - 850 | 10 | Apk |
| 2 – 3 | 16 | Av. Max AC current | Available max AC current. This value affected by the limitation functions (igbt temp, motortemp etc.) Defines how much current available from the configured one. | -3276,8 - 3276,7 | 0 - 850 | 10 | Apk |
| 4 – 5 | 32 | Min AC current | Min AC current configured with the DTI CAN tool. | -3276,8 - 3276,7 | -850 – 0 | 10 | Apk |
| 6- 7 | 48 | Av. Min AC current | Available min AC current. This value affected by the limitation functions (igbt temp, motortemp etc.) Defines how much current available from the configured one. | -3276,8 - 3276,7 | -850 - 0 | 10 | Apk |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x26 | Length: 8 bytes | | Configured and available DC currents | | | | | |
|  | 0 – 1 | 0 | Max DC current | Max DC current configured with the DTI CAN tool. | -3276,8 - 3276,7 | 0 - 850 | 10 | Adc |
| 2 – 3 | 16 | Av. Max DC current | Available max DC current. This value affected by the limitation functions (igbt temp, motortemp etc.) Defines how much current available from the configured one. | -3276,8 - 3276,7 | 0 - 850 | 10 | Adc |
| 4 – 5 | 32 | Min DC current | Min DC current configured with the DTI CAN tool. | -3276,8 - 3276,7 | -850 - 0 | 10 | Adc |
| 6- 7 | 48 | Av. Min DC current | Available min DC current. This value affected by the limitation functions (igbt temp, motortemp etc.) Defines how much current available from the configured one. | -3276,8 - 3276,7 | -850 - 0 | 10 | Adc |

## Example messages for transmitted data by inverter

Table : Example message: ERPM, Duty, Input Voltage (0x20)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Can message: ERPM, Duty, Input Voltage **(0x20)** | | | | | |
|  | **Message ID** | | **Data segment** | | |
| **Message** | 0x 00 00 20 | 22 | 00 00 24 5E | 00 71 | 01 86 |
| **Data** | PID | Node ID | ERPM  [3:0] | Duty Cycle  [1:0] | Input voltage [1:0] |
| **Length** |  |  | 4- byte | 2- byte | 2- byte |
| **Scale** | 1 | 10 | 1 |
| **HEX Value** | 0x0000245E | 0x0071 | 0x0186 |
| **DEC Value** | 9310 | 113 | 390 |
| **Real value** | 9310 RPM | 11.3% | 390 V |
| **Raw MSG:** | 00 00 20 22 00 00 24 5E 00 71 01 86 (hexadecimal) | | | | |

Table : Example message: AC Current, DC Current, (0x21)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Can message: AC Current, DC Current **(0x21)** | | | | |  |
|  | **Message ID** | | **Data segment** | |
| **Message** | 0x 00 00 21 | 22 | 00 5C | 00 11 |
| **Data** | PID | Node ID | AC Current  [1:0] | DC Current  [1:0] |
| **Length** |  |  | 2- byte | 2- byte |
| **Scale** | 10 | 10 |
| **HEX Value** | 0x005C | 0x0011 |
| **DEC Value** | 92 | 17 |
| **Real value** | 9.2 AAC | 1.7 ADC |
| **Raw MSG:** | 00 00 21 22 00 5C 00 11 FF FF FF FF (hexadecimal) | | | |

Table : Example message: CTLR Temp, Motor Temp, Fault code (0x22)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Can message: Controller Temp, Motor Temp, Fault code **(0x22)** | | | | | |
|  | **Message ID** | | **Data segment** | | |
| **Message** | 0x 00 00 22 | 22 | 01 53 | 01 17 | 00 |
| **Data** | PID | Node ID | CTLR Temp  [1:0] | Motor Temp  [1:0] | Fault Code |
| **Length** |  |  | 2- byte | 2- byte | 1- byte |
| **Scale** | 10 | 10 | 1 |
| **HEX Value** | 0x0153 | 0x0117 | 0x00 |
| **DEC Value** | 339 | 279 | 0 |
| **Real value** | 33.9 °C | 27.9 °C | None. |
| **Raw MSG:** | 00 00 22 22 01 53 01 17 00 FF FF FF (hexadecimal) | | | | |

Table : Example message: Id, Iq Values (0x23)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Can message: Id, Iq Values **(0x23)** | | | | |
|  | **Message ID** | | **Data segment** | |
| **Message** | 0x 00 00 23 | 22 | 00 00 00 64 | 00 00 02 92 |
| **Data** | PID | Node ID | Id  [3:0] | Iq  [3:0] |
| **Length** |  |  | 4- byte | 4- byte |
| **Scale** | 100 | 100 |
| **HEX Value** | 0x00000064 | 0x00000292 |
| **DEC Value** | 100 | 658 |
| **Real value** | 1 | 6.58 |
| **Raw MSG:** | 00 00 23 22 00 00 00 64 00 00 02 92 (hexadecimal) | | | |

# Commands to the inverter

The inverter can receive commutation related and limiting commands on the CAN2. The following overview shows the possible methods for commutating via CAN and setting limits.

## Command overview

The following commands can be sent to the inverter:

Table : Control commands

|  |  |  |
| --- | --- | --- |
| **Packet ID** | **Command** | **Function** |
| **0x01** | Set Current |  |
| **0x02** | Set Brake current |
| **0x03** | Set ERPM |
| **0x04** | Set Position |
| **0x05** | Set Relative current |
| **0x06** | Set relative brake current |
| **0x07** | Set digital output | Sets an output to HIGH or LOW |
| **0x08** | Set maximum AC current | Limiting command |
| **0x09** | Set maximum AC brake current |
| **0x0A** | Set maximum DC current |
| **0x0B** | Set maximum DC brake current |
| **0x0C** | Drive enable |

## Commands can be sent to the inverter

Table Description of inverter commands

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x01 | Length: 8 bytes | | Set AC current |  | | | | |
|  | 0 – 1 | 0 | AC current | This command sets the target motor AC current (peak, not RMS). When the controller receives this message, it automatically switches to current control mode.  This value must not be above the limits of the inverter and must be multiplied by 10 before sending. This is a signed parameter, and the sign represents the direction of the torque which correlates with the motor AC current. (For the correlation, please refer to the motor parameters) | -3276,8 - 3276,7 | -850 - 850 | 10 | Apk |
|  | 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |
| 0x02 | Length: 8 bytes | | Set Brake current |  | | | | |
|  | 0 - 1 | 0 | Target brake current | Targets the brake current of the motor. It will result negative torque relatively to the forward direction of the motor.  This value must be multiplied by 10 before sending, **only positive currents are accepted.** | -3276,8 - 3276,7 | 0 - 850 | 10 | Apk |
| 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x03 | Length: 8 bytes | | Set speed (ERPM) |  | | | | |
|  | 0 – 3 | 0 | Target ERPM | This command enables the speed control of the motor with a target ERPM. This is a signed parameter, and the sign represents the direction of the spinning. For better operation you need to tune the PID of speed control.  Equation: ERPM = Motor RPM \* number of the motor pole pairs. | -2147483648 - 2147483647 | -100 000 – 100 000 | 1 | ERPM |
|  | 4 – 7 | 32 | NOT USED | Not relevant to the command. Fill with FFs or use 4-byte DLC. | - | - | - | - |
| 0x04 | Length: 8 bytes | | Set position |  | | | | |
|  | 0 - 1 | 0 | Target position | This value targets the desired position of the motor in degrees. This command is used to hold a position of the motor.  This feature is enabled only if encoder is used as position sensor. The value has to be multiplied by 10 before sending. | -3276,8 - 3276,7 | 0 - 359 | 10 | degree |
| 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x05 | Length: 8 bytes | | Set relative current |  | | | | |
|  | 0 – 1 | 0 | Target relative AC current | This command sets a relative AC current to the minimum and maximum limits set by configuration. This achieves the same function as the “Set AC current” command. Gives you a freedom to send values between -100,0% and 100,0%. You do not need to know the motor limit parameters.  This value **must be between -100 and 100** and must be multiplied by 10 before sending. | -3276,8 - 3276,7 | -100 - 100 | 10 | % |
|  | 2 – 7 | 32 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |
| 0x06 | Length: 8 bytes | | Set relative brake current |  | | | | |
|  | 0 - 1 | 0 | Target relative brake AC current | Targets the relative brake current of the motor. It will result negative torque relatively to the forward direction of the motor. This value must be between 0 and 100 and must be multiplied by 10 before sending Gives you a freedom to send values between 0% and 100,0%. You do not need to know the motor limit parameters.  This value must be between 0 and 100 and has to be multiplied by 10 before sending | -3276,8 - 3276,7 | 0 - 100 | 10 | % |
| 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x07 | Length: 8 bytes | | Set digital output | Sets a digital output to a desired state | | | | |
|  | 0 | 0 | Digital output 1 | Sets the digital output 1 to HIGH (1) or LOW (0) state | 0 | 1 | 1 | # |
| 0 | 1 | Digital output 2 | Sets the digital output 2 to HIGH (1) or LOW (0) state | 0 | 1 | 1 | # |
| 0 | 2 | Digital output 3 | Sets the digital output 3 to HIGH (1) or LOW (0) state | 0 | 1 | 1 | # |
| 0 | 3 | Digital output 4 | Sets the digital output 4 to HIGH (1) or LOW (0) state | 0 | 1 | 1 | # |
| 0x08 | Length: 8 bytes | | Set max. AC current |  | | | | |
|  | 0 – 1 | 0 | Maximum AC current | This value determines the maximum allowable drive current on the AC side. With this function you are able maximize the maximum torque on the motor.  The value must be multiplied by 10 before sending. | -3276,8 - 3276,7 | 0 - 850 | 10 | Apk |
|  | 2 – 7 | 32 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x09 | Length: 8 bytes | | Set max brake current |  | | | | |
|  | 0 - 1 | 0 | Maximum brake AC current | This value sets the maximum allowable brake current on the AC side.  This value must be multiplied by 10 before sending, **only negative currents are accepted.** | -3276,8 - 3276,7 | -850 - 0 | 10 | Apk |
| 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Packet ID** | **Byte** | **Bit start** | **Msg name** | **Message description** | **Min/max data range (post-scale)** | **Min/max operational range** | **Scale** | **Unit** |
| 0x0A | Length: 8 bytes | | Set max. DC current |  | | | | |
|  | 0 – 1 | 0 | Maximum DC current limit | This value determines the maximum allowable drive current on the DC side. With this command the BMS can limit the maximum allowable battery discharge current.  The value has to be multiplied by 10 before sending. | -3276,8 - 3276,7 | 0 - 850 | 10 | A |
|  | 2 – 7 | 32 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |
| 0x0B | Length: 8 bytes | | Set max. DC brake current |  | | | | |
|  | 0 - 1 | 0 | Maximum brake DC current | This value determines the maximum allowable brake current on the DC side. With this command the BMS can limit the maximum allowable battery charge current.  The value has to be multiplied by 10 before sending. **Only negative currents are accepted.** | -3276,8 - 3276,7 | -850 - 0 | 10 | % |
| 2 - 7 | 16 | NOT USED | Not relevant to the command. Fill with FFs or use 2-byte DLC. | - | - | - | - |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0x0C | Length: 8 bytes | | Drive enable |  |  |  |  |  |
|  | 0 | 0 | Drive enable | 0: Drive not allowed  1: Drive allowed  Only 0 and 1 values are accepted. Must be sent periodically to be enabled. **Refer to chapter 4.3** | 0 | 255 | 1 | # |
|  | 1 - 7 | 8 | NOT USED | Not relevant to the command. Fill with FFs or use 1-byte DLC. | - | - | - | - |

## Drive enable command

The drive enable command let you allow the drive of the inverter, even if analogue inputs are used for control.

To activate this function, you should check / APP settings / CAN / “Drive enable via CAN2” button. If the function is not activated, then commands via CAN2 interface will be not accepted.

### Operation when analogue inputs are used for control

The drive enable signal will be “FALSE” by default, thus not allowing drive until the command message (0x24) not received periodically. If you use analogue control, you will not be able to drive the motor without sending the drive enable command periodically.

The sending period must be at least half of the value set in the configuration / App settings / General / Timeout.

**For example:**

If the timeout is 1000 ms, you should send at least 500 ms or quicker (250 ms recommended) to avoid unexpected timeouts.

### Operation when CAN2 interface used for control

The drive enable signal will be “FALSE” by default, thus not allowing drive until the command message (0x24) not received periodically. When you are using CAN2 interface for control the inverter, the drive enable message can be sent periodically, but not mandatory. It is enough to send a drive enable message at first and keep sending a control command (like AC current, speed control, brake etc.) which will reset timeout. That means that when a control message not received periodically, a timeout will happen, which will disable the drive enable state. After that event, the drive enable message should be sent again, to allow the drive.

## Broadcast message

Broadcast messages can be sent to all the inverters on the same bus in order to set a common limit or commutation request.

* With Standard CAN message, the **node ID** should be replaced with **0x1F**
* With Extended CAN message, the **node ID** should be **0xFF**.

## Example messages for commands

Table : Example message: Set AC Current (0x01)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set AC Current **(0x01)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 01 | 22 | 00 64 |
| **Data** | PID | Node ID | Set AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 10A |
| **Scaled value** | 100 |
| **Hex value** | 0x0064 |
| **Raw MSG:** | 00 00 01 22 00 64 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set Brake Current (0x02)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set Brake Current **(0x02)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 02 | 22 | 00 64 |
| **Data** | PID | Node ID | Set Brake AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 10A |
| **Scaled value** | 100 |
| **Hex value** | 0x0064 |
| **Raw MSG:** | 00 00 02 22 00 64 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set Brake Current (0x03)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set ERPM **(0x03)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 03 | 22 | 00 00 01 F4 |
| **Data** | PID | Node ID | Set ERPM  [3:0] |
| **Length** |  |  | 4- byte |
| **Scale** | 1 |
| **Real (Target) value** | 500 ERPM |
| **Scaled value** | 500 |
| **Hex value** | 0x000001F4 |
| **Raw MSG:** | 00 00 03 22 00 00 01 F4 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set position (0x04)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set Position **(0x04)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 04 | 22 | 03 E8 |
| **Data** | PID | Node ID | Set position  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 100° |
| **Scaled value** | 1000 |
| **Hex value** | 0x03E8 |
| **Raw MSG:** | 00 00 04 22 03 E8 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set Relative current (0x05)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set Relative Current (**0x05)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 05 | 22 | 00 64 |
| **Data** | PID | Node ID | Set relative AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 10% |
| **Scaled value** | 100 |
| **Hex value** | 0x0064 |
| **Raw MSG:** | 00 00 05 22 00 64 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set relative brake current (0x06)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set Relative brake current **(0x06)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 06 | 22 | 00 64 |
| **Data** | PID | Node ID | Set relative brake Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 10% |
| **Scaled value** | 100 |
| **Hex value** | 0x0064 |
| **Raw MSG:** | 00 00 06 22 00 64 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set maximum AC current (0x08)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set maximum AC current (**0x08)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 08 | 22 | 03 E8 |
| **Data** | PID | Node ID | Set maximum AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 100 AAC |
| **Scaled value** | 1000 |
| **Hex value** | 0x03E8 |
| **Raw MSG:** | 00 00 08 22 03 E8 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set maximum AC brake current (0x09)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set maximum AC Brake current **(0x09)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 09 | 22 | FC 18 |
| **Data** | PID | Node ID | Set maximum AC Brake Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | -100AAC |
| **Scaled value** | -1000 |
| **Hex value** | 0xFC18 |
| **Raw MSG:** | 00 00 09 22 FC 18 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set maximum DC current (0x0A)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set maximum DC current **(0x0A)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 0A | 22 | 00 C8 |
| **Data** | PID | Node ID | Set maximum AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | 20ADC |
| **Scaled value** | 200 |
| **Hex value** | 0x00C8 |
| **Raw MSG:** | 00 00 0A 22 00 C8 00 00 00 00 00 00 (hexadecimal) | | |

Table : Example message: Set maximum DC brake current (0x0B)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Set maximum DC brake current **(0x0B)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 0B | 22 | FF 38 |
| **Data** | PID | Node ID | Set maximum AC Current  [1:0] |
| **Length** |  |  | 2- byte |
| **Scale** | 10 |
| **Real (Target) value** | -20 ADC |
| **Scaled value** | -200 |
| **Hex value** | 0xFF38 |
| **Raw MSG:** | 00 00 0B 22 FF 38 00 00 00 00 00 00 (hexadecimal) | | |

Table Example message: Drive enable command (0x0C)

|  |  |  |  |
| --- | --- | --- | --- |
| Can message: Drive enable **(0x0C)** | | | |
|  | **Message ID** | | **Data segment** |
| **Message** | 0x 00 00 0C | 22 | 01 |
| **Data** | PID | Node ID | Drive enable |
| **Length** |  |  | 1- byte |
| **Scale** | 1 |
| **Real (Target) value** | 1: TRUE enables drive  0: FALSE disables drive |
| **Scaled value** | 1 |
| **Hex value** | 0xFF38 |
| **Raw MSG:** | 00 00 0C 22 01 00 00 00 00 00 00 00 (hexadecimal) | | |

# List of tables

[Table 1 Standard ID example message 7](#_Toc128389541)

[Table 2 Extended ID example message 7](#_Toc128389542)

[Table 3: Transmitted messages 8](#_Toc128389543)

[Table 4 Transmitted CAN bus messages 9](#_Toc128389544)

[Table 5: Example message: ERPM, Duty, Input Voltage (0x20) 17](#_Toc128389545)

[Table 6: Example message: AC Current, DC Current, (0x21) 17](#_Toc128389546)

[Table 7: Example message: CTLR Temp, Motor Temp, Fault code (0x22) 18](#_Toc128389547)

[Table 8: Example message: Id, Iq Values (0x23) 18](#_Toc128389548)

[Table 9: Control commands 19](#_Toc128389549)

[Table 10 Description of inverter commands 20](#_Toc128389550)

[Table 11: Example message: Set AC Current (0x01) 28](#_Toc128389551)

[Table 12: Example message: Set Brake Current (0x02) 29](#_Toc128389552)

[Table 13: Example message: Set Brake Current (0x03) 29](#_Toc128389553)

[Table 14: Example message: Set position (0x04) 30](#_Toc128389554)

[Table 15: Example message: Set Relative current (0x05) 30](#_Toc128389555)

[Table 16: Example message: Set relative brake current (0x06) 31](#_Toc128389556)

[Table 17: Example message: Set maximum AC current (0x08) 31](#_Toc128389557)

[Table 18: Example message: Set maximum AC brake current (0x09) 32](#_Toc128389558)

[Table 19: Example message: Set maximum DC current (0x0A) 32](#_Toc128389559)

[Table 20: Example message: Set maximum DC brake current (0x0B) 33](#_Toc128389560)

[Table 21 Example message: Drive enable command (0x0C) 33](#_Toc128389561)