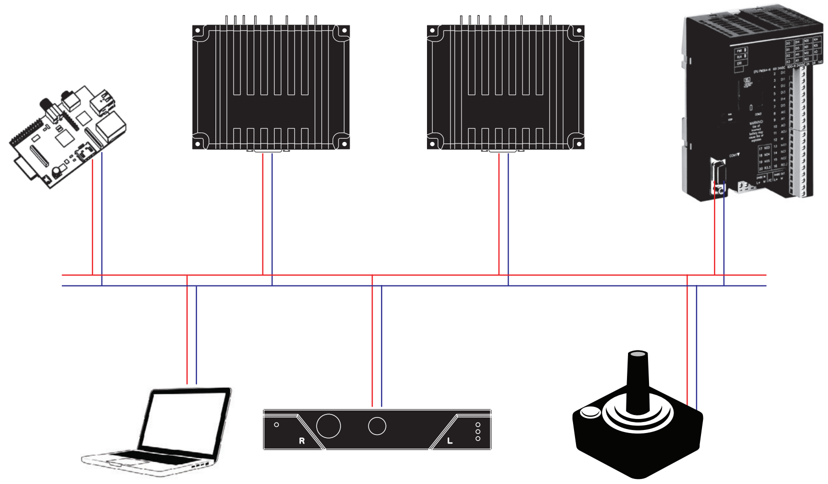
CANBus



CANOpen
RawCAN
MiniCAN
RoboCAN

User & Reference Manual

V2.1a, February 10, 2022

visit www.roboteq.com to download the latest revision of this manual

Revision History

Date	Version	Changes
July 8, 2019	2.0	Extracted from main User Manual
December 3, 2020	2.1	Add cyclic Synchronous Modes. Miscellaneous updates in order to conform to firmware v2.1
February 10, 2022	2.1a	Miscellaneous updates in order to conform to firmware v2.1a

SECTION 3

SECTION 4

Introduction

Refer to the Datasheet for Hardware-Specific Issues

This manual is the companion to your controller's datasheet. All information that is specific to a particular controller model is found in the datasheet. These include:

- Number and types of I/O
- Connectors pin-out
- Wiring diagrams
- Maximum voltage and operating voltage
- Thermal and environmental specifications
- Mechanical drawings and characteristics
- Available storage for scripting
- Battery or/and Motor Amps sensing
- Storage size of user variables to Flash or Battery-backed RAM

User Manual Structure and Use

The user manual discusses issues that are common to all controllers inside a given product family. Except for a few exceptions, the information contained in the manual does not repeat the data that is provided in the datasheets.
The manual is divided in 3 sections organized as follows:

SECTION 1 CAN Networking on Roboteq Controllers

This section describes the RawCAN and MiniCAN operating modes available on CAN-enabled Roboteq controllers.

SECTION 2 RoboCAN Networking

This section describes the RoboCAN protocol: a simple and efficient meshed network scheme for Roboteq devices

SECTION 3 CANopen Interface

This section describes the configuration of the CANopen communication protocol and the commands accepted by the controller operating in the CANopen mode.

SECTION 4 DS402 Implementation on Roboteq Motor Controllers

This section will describe the implementation of CiA DS402 standard on Roboteq motor controllers. DS402 Objects are not applicable to film-ware for Induction and sensor-less motor controllers.

SECTION 1

CAN Networking on Roboteq Controllers

Some controller models are equipped with a standard CAN interface allowing up to 127 controllers to work together on a single twisted pair network at speeds up to 1Mbit/s.

Supported CAN Modes

Four CAN operating modes are available on Roboteq controllers:

1 - RawCAN
2 - MiniCAN
3 - CANopen
4 - RoboCAN

RawCAN is a low-level operating mode giving total read and write access to CAN frames. It is recommended for use in low data rate systems that do not obey to any specific standard. CAN frames are typically built and decoded using the MicroBasic scripting language.

MiniCAN is greatly simplified subset of CANopen, allowing, within limits, the integration of the controller into an existing CANopen network. This mode requires MicroBasic scripting to prepare and use the CAN data.

CANopen is the full Standard from CAN in Automation (CIA), based on the DS402 specification. It is the mode to use if full compliance with the CANopen standard is a primary requisite.

RoboCAN is a Roboteq proprietary meshed networking scheme allowing multiple Roboteq devices to operate together as a single system. This protocol is extremely simple and lean, yet practically limitless in its abilities. It is the preferred protocol to use by users who just wish to make multiple controllers work together with the minimal effort.

This section describes the RawCAN and MiniCAN modes.

Detailed descriptions of CANopen and RoboCAN can be found in specific sections of this manual.

Connecting to CAN bus

A CAN bus network is made of a stretch of two wires. A device can be put on a CANbus network by simply connecting it's CAN-High and CAN-Low lines to those of other devices on the network.

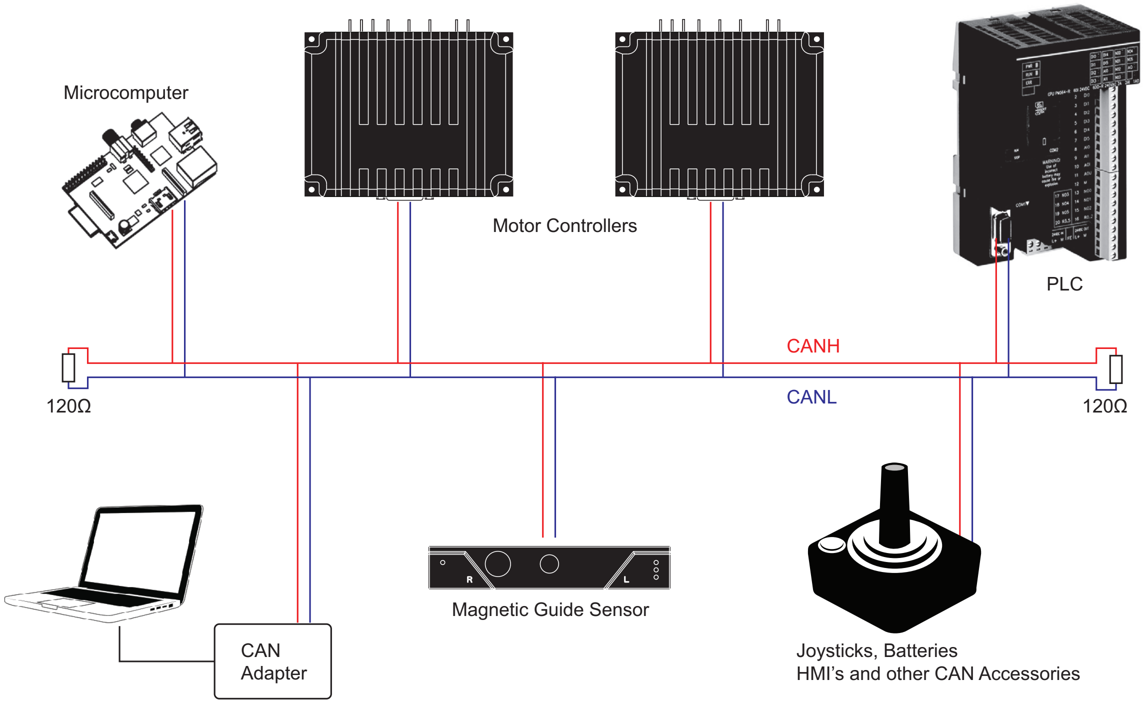


Figure 1-1: CAN Network topology

Resistors should be 120 ohm and located at each end of the cable. However, on a short network communication will take place with a single resistor of 100 to 200 ohm located anywhere on the network. Communication will not work if no resistor is present, or if its value is too high.

No ground connection is necessary in between nodes. However, the ground potential of each node must be within a few volts of each other. If all devices on the network are powered from the same power source, this can be expected to be the case.

CANbus will be operational upon enabling the desired CAN protocol and speed using the PC utility.

Important Warning

A ground difference up to around 10V is acceptable. A difference of 30V or higher can cause damage to one or more nodes. CANbus isolators must be used if a similar ground level cannot be guaranteed between nodes.

Introduction to CAN Hardware signaling

CANbus uses differential signals, which is where CAN derives its robust noise immunity and fault tolerance. The two signal lines of the bus, CANH and CANL, are biased to around 2.5 V. A logical “1” (also known as the dominant state) on the bus takes CANH around 1 V higher to around 3.5 V, and takes CANL around 1 V lower to 1.5 V, creating a typical 2V differential signal as shown in Figure 1-2.

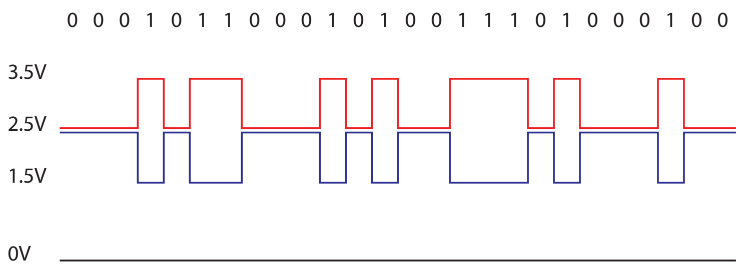


Figure 1-2: CANbus signaling

Differential signaling reduces noise coupling and allows for high signaling rates over twisted-pair cable. The High-Speed CANbus specifications (ISO 11898 Standard) are given for a maximum signaling rate of 1 Mbps with a bus length of 40 m with a maximum of 30 nodes. It also recommends a maximum unterminated stub length of 0.3 m.

CAN Bus Pinout

Depending on the controller model, the CAN signals are located on the 9-pin, 15-pin or 25-pin DSub connector. Refer to datasheet for details.

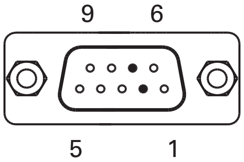


FIGURE 1-3. DB9.Connector pin locations

The pins on the DB9 connector are mapped as described in the table below.

TABLE 1-1. CAN Signals on DB9 connector

Pin Number	Signal	Description
2	CAN_L	CAN bus low
7	CAN_H	CAN bus high

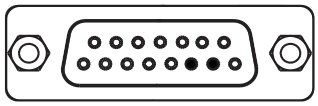


FIGURE 1-4. DB15 Connector pin locations

The pins on the DB15 connector are mapped as described in the table below.

TABLE 1-2. CAN Signals on DB15 connector

Pin Number	Signal	Description
6	CAN_L	CAN bus low
7	CAN_H	CAN bus high

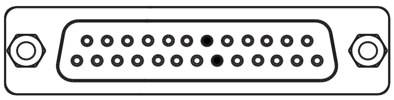


FIGURE 1-5. DB25 pin locations

The pins on the DB25 connector are mapped as described in the table below.

TABLE 1-3. CAN Signals od DB25 connector

Pin Number	Signal	Description
8	CAN_L	CAN bus low
20	CAN_H	CAN bus high

CAN and USB Limitations

On some controller models CAN and USB cannot operate at the same time. On controllers equipped with a USB connector, if simultaneous connection is not allowed, the controller will enable CAN if USB is not connected.

The controller will automatically enable USB and disable CAN as soon as the USB is connected to the PC. The CAN connection will then remain disabled until the controller is restarted with the USB unplugged.

See the controller model datasheet to verify whether simultaneous CAN and USB is supported.

Basic Setup and Troubleshooting

CANbus is very easy to setup: Simply connect the CANH and CANL to a pair of wires with at least one resistor somewhere along the cable. Enable the desired CAN protocol and speed using the PC utility.

If communication cannot be established, it can be difficult to determine the source of the problem. Here are a few ways to diagnose:

Cable polarity, integrity and termination resistor

Verify that the controller's CANH and CANL are connected to the CANH and CANL wire. Check cable continuity to every node. Verify the presence of a least one resistor and that its value is 120ohm (a value of 60 to 200 ohm would be acceptable)

Check CANbus activity using a voltmeter

The presence of CAN data traffic can be checked using a simple voltmeter and measuring the voltage between GND and CANH, and between GND and CANL. When CAN is disabled, both lines should have approximately the same voltage around 2.5V. When CAN is enabled with RoboCAN or MiniCAN protocol selected, the controller will send a continuous stream of data frames. This will cause the CANH voltage to rise above, and the CANL voltage to drop below, the 2.5V midpoint. If the idle and active voltages do not match the above, try again on the controller alone disconnected from the network but with a 100 to 200 ohm resistor across its CANH and CANL pins.

The CANOpen and RawCAN protocol should not be used for this test as these do not generate data traffic on their own and will not cause measurable voltage changes.

Check CANbus activity using a CAN sniffer

When working on a CAN system, it is highly recommended to make the acquisition of a USB to CAN adapter such as the PCAN-USB from Peak Systems. Connect the adapter to the CANH and CANL and run the sniffer software with the correct bit rate selected. The figure below shows the expected received data when a Roboteq device is on the network with MiniCAN protocol enabled.

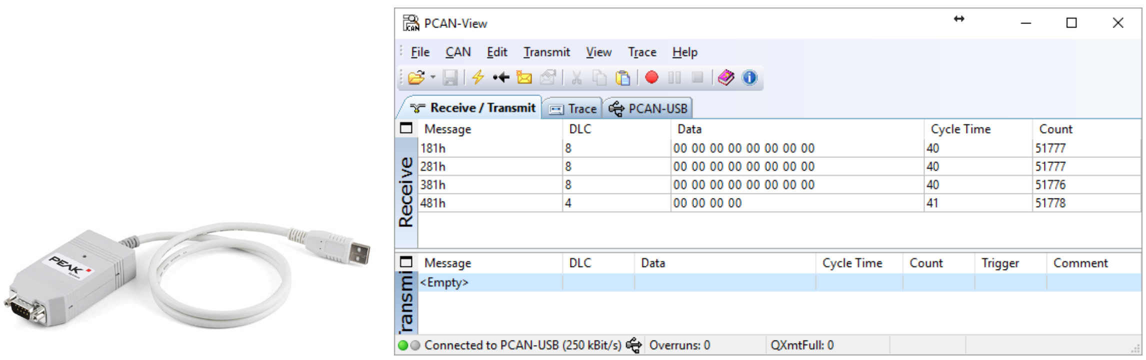


Figure 1-6. USB to CAN adapter and MiniCAN frame capture

Mode Selection and Configuration

Mode selection is done using the CAN menu in the RoborunPlus PC utility.

Common Configurations

CAN Mode:	Used to select one of the 4 operating modes. Off disables all CAN receive and transmit capabilities.
Node ID:	CAN Node ID used for transmission from the controller. Value may be between 1 and 126 included.
Bit Rate:	Selectable bit rate. Available speeds are 1000, 800, 500, 250, and 125 kbit/s. Default is 125kbit and is the recommended speed for RawCAN and MiniCAN modes.
Heartbeat:	Period at which a Heartbeat frame is sent by the controller. The frame is CANopen compatible 0x700 + NodeID, with one data byte of value 0x05 (Status: Operational). The Heartbeat is sent in any of the selected modes. It can be disabled by entering a value of 0.

MiniCAN Configurations

ListenNodeID:	Filters to accept only packets sent by a specific node.
SendRate:	Period at which data frames are sent by the controller. Frames are structured as standard CANopen Transmit Process Data Objects (TPDOs). Transmission can be disabled by entering a value of 0.

RawCAN Configurations

In the RawCAN mode, incoming frames may be filtered or not by changing the ListenNodeID parameter that is shared with the MiniCAN mode. A value of 0 will capture all incoming frames and it will be up to the user to use the ones wanted. Any other value will cause the controller to capture only frames from that sender.

Using RawCAN Mode

In the RawCAN Mode, received unprocessed data packets can be read by the user. Likewise, the user can build a packet with any content and send it on the CAN network. A FIFO buffer will capture up to 16 frames.

CAN packets are essentially composed by a header and a data payload. The header is an 11 bit number that identifies the sender's address (bits 0 to 6) and a packet type (bits 7 to 10). Data payload can be 0 to 8 bytes long.

Checking Received Frames

Received frames are first loaded in the 16-frame FIFO buffer. Before a frame can be read, it is necessary to check if any frames are present in the buffer using the **?CF** query. The query can be sent from the serial/USB port, or from a MicroBasic script using the getvalue(_CF) function. The query will return the number of frames that are currently pending, and copy the oldest frame into the read buffer, from which it can then be accessed. Sending **?CF** again, copies the next frame into the read buffer.
The query usage is as follows:

Syntax:	**?CF**
Reply:	**CF**=number of frames pending

Reading Raw Received Frames

After a frame has been moved to the read buffer, the header, bytecount and data can be read with the **?CAN** query. The query can be sent from the serial/USB port, or from a MicroBasic script using the getvalue(_CAN, n) function. The query usage is as follows:

When the query is sent from serial or USB, without arguments, the controller replies by outputting all elements of the frame separated by colons.

Syntax: **?CAN [ee]**

Reply: **CAN**=header:bytecount:data0:data1: .... :data7

Where: **ee** = frame element
1 = header
2 = bytecount
3 to 10 = data0 to data7

Examples: Q: **?CAN**
R: **CAN=5:4:11:12:13:14:0:0:0:0**

Q: **?CAN 3**
R: **CAN=11**

Notes: Read the header to detect that a new frame has arrived. If header is different than 0, then a new frame has arrived and you may read the data.

After reading the header, its value will be 0 if read again, unless a new frame has arrived.

New CAN frames will not be received by the controller until a ?CAN query is sent to read the header or any other element.

Once the header is read, proceed to read the other elements of the received frame without delay to avoid data to be overwritten by a new arriving frame.

Transmitting Raw Frames

RawCAN Frames can easily be assembled and transmitted using the CAN Send Command !CS. This command can be used to enter the header, bytecount, and data, one element at a time. The frame is sent immediately after the bytecount is entered, and so it should be entered last.

Syntax: **!CS ee nn**

Where: **ee** = frame element
1 = header
2 = bytecount
3 to 10 = data0 to data7
nn = value

Examples:

!CS 1 5	Enter 5 in header
!CS 3 2	Enter 2 in Data 0
!CS 4 3	Enter 3 in Data 1
!CS 2 2	Enter 2 in bytecount. Send CAN data frame

Using MiniCAN Mode

MiniCAN is greatly simplified subset of CANopen. It only supports Heartbeat, and fixed map Received Process Data Objects (RPDOs) and Transmit Process Data Objects (TPDOs). It does not support Service Data Objects (SDOs), Network Management (NMT), SYNC or other objects.

Transmitting Data

In MiniCAN mode, data to be transmitted is placed in one of the controller's available Integer or Boolean User Variables. Variables can be written by the user from the serial/USB using !VAR for Integer Variables, or !B for Boolean Variables. They can also be written from MicroBasic scripts using the setcommand(_VAR, n) and setcommand(_B, n) functions. The value of these variables is then sent at a periodic rate inside four standard CANopen TPDO frames (TPDO1 to TPDO4). Each of the four TPDOs is sent in turn at the time period defined in the SendRate configuration parameter.

Header:

TPDO1: 0x180 + NodeID
TPDO2: 0x280 + NodeID
TPDO3: 0x380 + NodeID
TPDO4: 0x480 + NodeID

Data:

TABLE 1-4. TPDO default mapping

	Byte1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
TPDO1	VAR1				VAR2			
TPDO2	VAR3				VAR4			
TPDO3	VAR5		VAR6		VAR7		VAR8	
TPDO4	BVar 1-8	BVar 9-16	BVar 17-24	BVar 25-32				

Byte and Bit Ordering:

Integer Variables are loaded into a frame with the Least Significant Byte first. Example 0x12345678 will appear in a frame as 0x78 0x56 0x34 0x12.

Boolean Variables are loaded in a frame as shown in the table above, with the lowest Boolean Variable occupying the least significant bit of each byte. Example Boolean Var 1 will appear in byte as 0x01.

Receiving Data

In MiniCAN mode, incoming frames headers are compared to the Listen Node ID number. If matched, and if the other 4 bits of the header identify the frame as a CANopen standard RPDO1 to RPDO4, then the data is parsed and stored in Integer or Boolean Variables according to the map below. The received data can then be read from the serial/USB using the ?VAR or ?B queries, or they can be read from a MicroBasic script using the getvalue(_VAR, n) or getvalue(_B, n) functions.

Header:

RPDO1: 0x200 + NodeID
RPDO2: 0x300 + NodeID
RPDO3: 0x400 + NodeID
RPDO4: 0x500 + NodeID

Data:

TABLE 1-5. RPDO default mapping

<table>
<tr><th></th><th>Byte1</th><th>Byte2</th><th>Byte3</th><th>Byte4</th><th>Byte5</th><th>Byte6</th><th>Byte7</th><th>Byte8</th></tr>
<tr><td>RPDO1</td><td colspan="4">VAR9</td><td colspan="4">VAR10</td></tr>
<tr><td>RPDO2</td><td colspan="4">VAR11</td><td colspan="4">VAR12</td></tr>
<tr><td>RPDO3</td><td colspan="2">VAR13</td><td colspan="2">VAR14</td><td colspan="2">VAR15</td><td colspan="2">VAR16</td></tr>
<tr><td>RPDO4</td><td>BVar 33-40</td><td>BVar 41-48</td><td>BVar 49-56</td><td>BVar 57-64</td><td></td><td></td><td></td><td></td></tr>
</table>

Byte and Bit Ordering:

Integer Variables are loaded from frame with the Least Significant Byte first. Example, a frame with data as 0x78 0x56 0x34 0x12 will load in an Integer Variable as 0x12345678.

Boolean Variables are loaded from a frame as shown in the table above, with the lowest Boolean Variable occupying the least significant bit of each byte. Example a received byte of 0x01 will set Boolean Var 33 and clear Vars 34 to 40.

MiniCAN Usage Example

MiniCAN can only be used with the addition of MicroBasic scripts that will give a meaning to the general variables in which the CAN data are stored. The following simple script uses VAR1 that is transported in RPDO1 as the incoming motor command and puts the Motor Amp VAR9 so that it is sent in TPDO1.

```
top:
speed = getvalue(_VAR, 9)
setcommand(_G, 1, speed)
motor_amp = getvalue(_A, 1)
setcommand(_VAR, 1, motor_amp)
wait(10)
goto top:
```

Note: This script does not check for loss of communication on the CAN bus. It is provided for information only.

SECTION 2

RoboCAN Networking

RoboCAN is a Roboteq proprietary meshed networking scheme allowing multiple Roboteq products to operate together as a single system. This protocol is extremely simple and lean, yet practically limitless in its abilities. It is the preferred protocol to use by a user who just wishes to make multiple controllers work together with minimal effort.

In RoboCAN, every controller can send commands to, and can read operational data from, any other node on the network. One or more controller can act as a USB to CAN or Serial to CAN gateway, allowing several controllers to be thus managed from a single PC or microcomputer.

Using a small set of dedicated Microbasic function, scripts can be written to exchange data between controllers in order to create automation systems without the need for a PLC or external computer.

In addition, RoboCAN includes support for processing raw can data as defined in the RawCAN specification (See page 14), in order to incorporate simple CAN compatible 3rd party devices in the network.

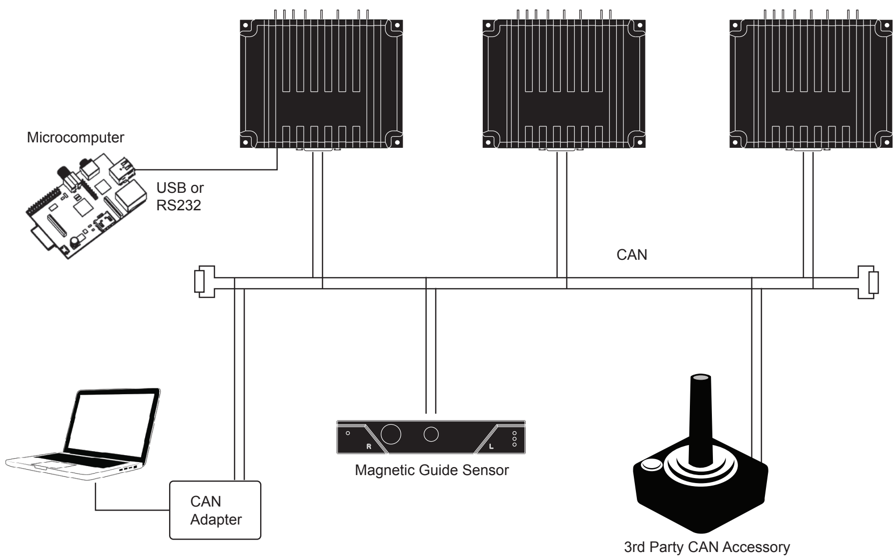


FIGURE 2-1. RoboCAN Network

Network Operation

RoboCAN requires only that a controller has a unique node number (other than 0) assigned and that the RoboCAN mode is selected and enabled. All nodes must be configured to operate at the same bit rate. Each enabled node will emit a special heartbeat at a set and unchangeable rate of 128ms so that each node can create and maintain a map of all nodes alive in the network.

RoboCAN via Serial & USB

Important notice: On many controller models, CAN and USB cannot be operated at the same time. Please see product datasheet to verify if this is the case on the model used. In case USB is not available, this section only applies to RS232 connections.

RoboCAN commands and queries can be sent from a USB or serial port using a modified syntax of the normal serial protocol: By simply adding the @ character followed by the node as a 2 digit hex address, a command or query is sent to the desired node. This scheme works with every Command (! Character), Query (?), Configuration setting (^), Configuration read (~), and most Maintenance commands (%)

Runtime Commands

Below is a Command example:

!G 1 500

This is the normal command for giving a 50% power level command to motor 1 of the controller that is attached to the computer.

@04!G 1 500

This will send the same 50% command to motor 1 of the controller at node address 4.

The reply to a local command is normally a + or - sign when a command is acknowledged or rejected in normal serial mode.

When a command is sent to a remote node, the reply is also a + or – sign. However, in addition, the reply can be a * sign to indicate that the destination node does not exist or is not alive. Note that the + sign only indicates that the command syntax is valid and that the destination node is alive.

Broadcast Command

Node address 00 is used to broadcast a command simultaneously to all the nodes in the network. For example

@00!G 1 500

Will apply 50% power to all motor 1 at all nodes, including the local node

Realtime Queries

Queries are handled the same way but the reply to a query includes the responding node's address. Below is a Query example:

?V 2

This is the normal query for reading the battery voltage of the local controller. The controller will reply V=123

@04?V 2

This will send the same query to node address 4

The reply of the remote node is @04 V=123

Replies to remote nodes queries are identical to these to a local controller with the exception of an added latency. Since the reply must be retrieved from the remote node depending on the selected bit rate, the reply may come up to 10ms after the query was sent.

Remote Queries restrictions

Remote queries can only return a single value whereas local queries can be used to read an array of values. For example

?AI

Is a local query that will return the values of all analog capture channels in a single string as

AI=123:234:345:567

@04?AI

Is a remote query and it will return only the first analog capture channel as

@04 AI=123

Remote queries are not being added in the Query history.

Broadcast remote queries are not supported. For example @00?V 1 will not be executed.

Queries that return strings, such as ?FID or ?TRN are not supported. They will return the value 0

See the Command Reference section in the manual for the complete list and description of available queries

Configurations Read/Writes

Configuration settings, like Amp Limit or Operating Modes can be read and changed on a remote node via the CAN bus. For example

@04^ALIM 1 250 will set the current limit of channel 1 of node 4 at 25.0A

@04~OVL will read the Overvoltage limit of node 4.

Note that changing a configuration via CAN only makes that change temporary until the remote controller is powered down. The %EESAV maintenance command must be send to the remote node to make the configuration change permanent.

A configuration write can be broadcast to all nodes simultaneously by using the node Id 00. For example

@00^OVL 250

Will set the overvoltage limit of all nodes at 25.0 Volts

Configuration reads cannot be broadcast.

See the Commands Reference section for the complete list and description of available configurations

Remote Configurations Read restrictions

Remote Configuration Reads can only return a single value whereas local Configuration Reads can be used to read an array of parameters. For example

~AMOD

Will return the operating mode of all analog capture channels in a single string as

AI=01:01:00:01:02

@04~AMOD

Will return only the mode first analog capture channel as

@04 AI=01

Configuration reads cannot be broadcast.

Remote Maintenance Commands

Maintenance Commands are not supported in RoboCAN.

Self Addressed Commands and Queries

For sake of consistency commands sent to the local node number are executed the same way as they would be on a remote node. However the no CAN frame is sent to the network. For example if node 04 receive the command

@04!G 1 500

No data will be sent on the network and it will be interpreted and executed the same way as

!G 1 500

RoboCAN via MicroBasic Scripting

A set of functions have been added to the MicroBasic language in order to easily send commands to, and read data from any other node on the network. Functions are also available to read and write configurations at a remote node. Maintenance commands are not supported.

Sending Commands and Configuration

Sending commands or configuration values is done using the functions

SetCANCommand(id, cc, ch, vv)

SetCANConfig(id, cc, ch, vv).

Where:

id is the remote Node Id in decimal

cc is the Command code, eg _G

ch is the channel number. Put 1 for commands that do not normally require a channel number

vv is the value

Example:

SetCANCommand(04, _G, 1, 500)

Will apply 50% power to motor 1 of node 4

SetCANConfig(0, _OVL, 1, 250)

Will set the overvoltage limit of all nodes to 25.0V. Note that even though the Overvoltage is set for the controller and does not normally require that a Channel, the value 1 must be put in order for the instruction to compile.

Script execution is not paused when one of these function is used. The frame is sent on the CAN network within one millisecond of the function call.

Reading Operating values Configurations

When reading an operating value such as Current Counter or Amps, or a configurations such as Overvoltage Limit from another node, since the data must be fetched from the network, and in order to avoid forcing a pausing of the script execution, data is accessed in the following manner:

1. Send a request to fetch the node data
2. Wait for data to be received
3. Read the data

The wait step can be done using one of the 3 following ways

1. Pause script execution for a few milliseconds using a wait() instruction in line.
2. Perform other functions and read the results a number of loop cycles later
3. Monitor a data ready flag

The following functions are available in microbasic for requesting operating values and configurations from a remote node.

FetchCANValue(id, cc, ch)

FetchCANConfig(id, cc, ch)

Where:

id is the remote Node Id in decimal
cc is the Command code, eg _G
cc is the channel number. Put 1 for commands that do not normally require a channel number

The following functions can be used to wait for the data to be ready for reading:

IsCANValueReady()

IsCANConfigReady()

These functions return a Boolean true/false value. They take no argument and apply to the last issued FetchCANValue or FetchCANConfig function

The retrieved value can then be read using the following functions:

ReadCANValue()

ReadCANConfig()

These functions return an integer value. They take no argument and apply to the last issued FetchCANValue or FetchCANConfig function

Below is a sample script that continuously reads and print the counter value of node 4

```
top:
FetchCANValue(4, _C, 1) ' request data from remote node
while(IsCANValueReady = false) ' wait until data is received
end while
Counter = ReadCANValue() ' read value
print (Counter, "\r") ' print value followed by new line
goto top ' repeat forever
```

Continuous Scan

In many applications, it is necessary to monitor the value of an operating parameter on a remote node. A typical example would be reading continuously the value of a counter. In order to improve efficiency and reduce overhead, a technique is implemented to automatically scan a desired parameter from a given node, and make the value available for reading without the need to send a Fetch command.

A function is provided to initiate the automatic sending of a value from the remote node, at a specific periodic rate, and to be stored to user selected location in a receive buffer.

The remote node will then send the data continuously without further commands.

A function is then provided to detect the arrival of a new value in that buffer location, and another to read the value from that location.

Since the scan rate is known, the execution of the script can be timed so that it is not necessary to check the arrival of a new value.

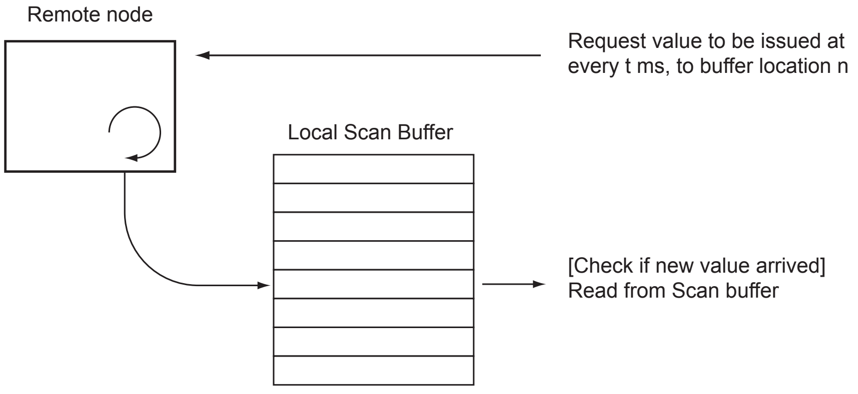


FIGURE 2-2. RoboCAN Continuous Scan

A scan is initiated with the function:

ScanCANValue(id, cc, ch, tt, bb)

Where:

id is the remote Node Id in decimal

cc is the Query code, eg _V

ch is the channel number. Put 1 for queries that do not normally require a channel number

tt is the scan rate in ms

bb is the buffer location

The scan rate can be up to 255ms. Setting a scan rate of 0 stops the automatic sending from this node.

Unless otherwise specified, the buffer can store up to 32 values.

The arrival of a new value is checked with the function

IsScannedCANReady(aa)

Where

aa is the location in the scan buffer.

The function returns a Boolean true/false value

The new value is then read with the function

ReadScannedCANValue(aa)

Where

aa is the location in the scan buffer.

The function returns an integer value. If no new value was received since the previous read, the old value will be read.

The following example shows the use of the Scan functions

```
' Initiate scan of counter every 10ms from node 4 and store to
buffer location 0
ScanCANValue(4, _C, 1, 10, 0)
' initiate scan of voltage every 100ms from node 4 and store to
buffer location 1
ScanCANValue(5, _V, 1, 100, 1)

top:

wait(10) ' Executer loop every 10 ms

' check if scanned volts arrived
if(IsScannedCANReady(1))
   ' read and print volts
   Volts = ReadScannedCANValue(1)
   print (Volts,"\r")
end if

' No need to check if counter is ready since scan rate = loop cy-
cle
Counter = ReadScannedCANValue(0)
print (Counter,"\r")

goto top ' Loop continuously
```

Checking the presence of a Node

No error is reported in MicroBasic if an exchange is initiated with a node that does not exist. A command or configuration sent to a non-existent node will simply not be executed. A query sent to a non existing or dead node will return the value 0. A function is therefore provided for verifying the presence of a live node. A live node is one that sends the distinct RoboCAN heartbeat frame every 128ms. The function syntax is:

IsCANNodeAlive(id)

Where:

id is the remote Node Id in decimal
The function returns a Boolean true/false value.

Self Addressed Commands and Queries

Functions addressed to the local node have no effect. The following function **will not work** if executed on node 4

SetCANCommand(04, _G, 1, 500)

The regular function must be used instead

SetCommand(_G, 1, 500)

Broadcast Command

Node address 00 is used to broadcast a command, or a configuration write simultaneously to all the nodes in the network.

The local node, however, will not be reached by the broadcast command.

Remote MicroBasic Script Download

RoboCAN includes a mechanism for loading MicroBasic scripts into any node in the network. Use the "To Remote" button in the Scripting Tab of the Roborun PC utility. A window will pop-up asking for the destination node Id. Details of the command used to enter the download mode and transferring scripts is outside the scope of this manual.

SECTION 3

CANopen Interface

This section describes the configuration of the CANopen communication protocol and the commands accepted by the controller using the CANopen protocol. It will help you to enable CANopen on your Roboteq controller, configure CAN communication parameters, and ensure efficient operation in CANopen mode.

The section contains CANopen information specific to Roboteq controllers. Detailed information on the physical CAN layer and CANopen protocol can be found in the DS402 section.

Use and benefits of CANopen

CANopen protocol allows multiple controllers to be connected into an extensible unified network. Its flexible configuration capabilities offer easy access to exposed device parameters and real-time automatic (cyclic or event-driven) data transfer.

The benefits of CANopen include:

- Standardized in EN50325-4
- Widely supported and vendor independent
- Highly extensible
- Offers flexible structure (can be used in a wide variety of application areas)
- Suitable for decentralized architectures
- Wide support of CANopen monitoring tools and solutions

CAN Connection

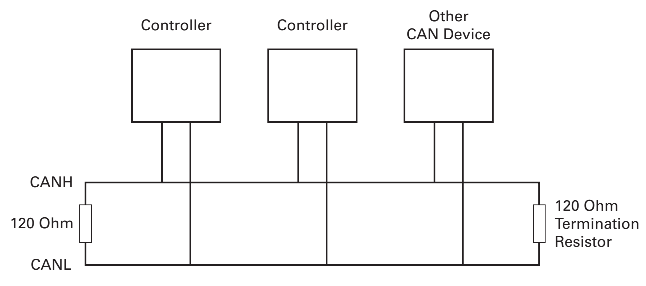


FIGURE 3-1. CAN connection

Connection to a CAN bus is as simple as shown on the diagram above. 120 Ohm Termination Resistors must be inserted at both ends of the bus cable. CAN network can be up to 1000m long. See CAN specifications for maximum length at the various bit rates.

CAN Bus Configuration

To configure communication parameters via the RoborunPlus PC utility, your controller must be connected to a PC via an RS232/RS485/TCP/USB port

Use the CAN menu in the Configuration tab in order to enable the CANopen mode. Additionally, the utility can be used to configure the following parameters:

- Node ID
- Bit rate
- Heartbeat (ms)
- Autostart
- TPDO Enable and Send rate

Node ID

Every CANopen network device must have a unique Node ID, between 1 and 127. The value of 0 is used for broadcast messaging and cannot be assigned to a network node.

Bit Rate

The CAN bus supports bit rates ranging from 10Kbps to 1Mbps. The default rate used in the current CANopen implementation is set to 125kbps. Valid bit rates supported by the controller are:

- 1000K
- 800K
- 500K
- 250K
- 125K

Heartbeat

A heartbeat message is sent to the bus in millisecond intervals, according to the value of the object 0x1017. Heartbeats are useful for detecting the presence or absence of a node on the network and determining the NMT state. The default value is set to 0ms (deactivated).

Autostart

When AutoStart is enabled, the controller automatically enters the Operational mode of the NMT state machine of the CANopen. The controller AutoStart is disabled by default. Disabling the parameter, will prevent the controller from entering operational state automatically, after the reset occurs. When Autostart is disabled, after power on or reset, the NMT state will not automatically do the transition and will remain to pre-operational.

Consumer Heartbeat Lost Action

The consumer heartbeat monitors if the producer heartbeat device is active on the network. Up to four devices can be monitored. The consumer heartbeat lost action can be configured through the Roborun+, by choosing either Safety Stop or Emergency Stop. When the consumer heartbeat is lost the watchdog is considered expired.

Commands Accessible via CANopen

Practically all of the controller's real-time queries and real-time commands that can be accessed via Serial/USB communication can also be accessed via CANopen. The meaning, effect, range, and use of these commands is explained in detail in Commands Reference section of the Roboteq Controllers User Manual.

All supported commands are mapped in a table, or Object Dictionary that is compliant with the CANopen specification. See "Object Dictionary" on page 39 for a complete set of commands.

NMT machine

In Figure 4.1 the NMT state diagram of a CANopen device is specified. CANopen devices enter the NMT state Pre-operational directly after finishing the CANopen devices initialization. During this NMT state CANopen device parameterization and CAN-ID-allocation via SDO. Then the CANopen devices switched directly into the NMT state Operational.

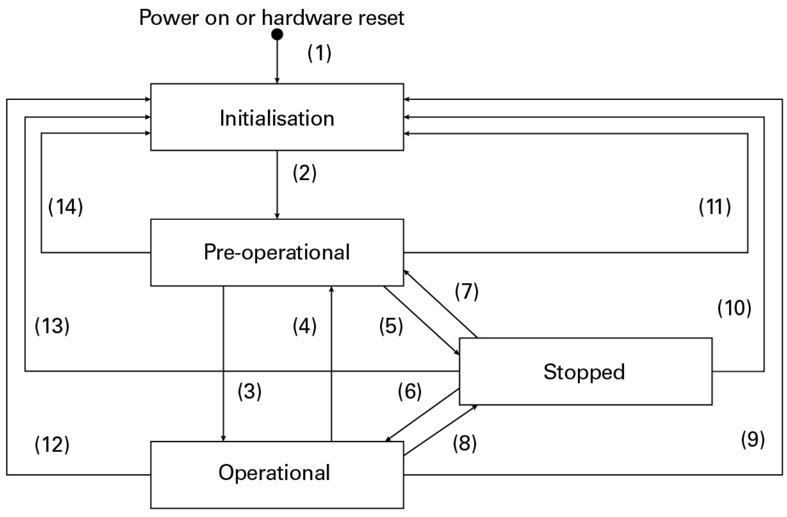


FIGURE 4.1. NMT state diagram of a CANopen device.

The table 3-1 presents the transitions of NMT machine. For more information about NMT operation modes you can find in CiA 301 documentation.

TABLE 3-1. Transitions of NMT machine.

(1)	At Power on the NMI initialisation is entered autonomously
(2)	NMT state initialisation finished - enter NMT state Pre-operational automatically
(3)	NMT service start remote node indication or by local control
(4), (7)	NMT service enter pre-operational indication
(5), (8)	NMT service stop remote node indication
(6)	NMT service start remote node indication
(9), (10), (11),	NMT service reset node indication
(12), (13), (14),	NMT service reset communication indication

The module control protocol is used by the NMT master to change the state of the devices. The CAN-frame's CAN-ID of this protocol is always 0, meaning that it has a function code 0 and node ID 0, which means that every node in the network will process this message. The actual node ID, to which the command is meant to, is given in the data part of the message (at the second byte). This can also be 0, meaning that all the devices on the bus should go to the indicated state. At table 3-2 is presented the CAN frame where the Requested state can be selected from the Table 3-3.

TABLE 3-2. Can frame to chance the NMT state.

COB-ID	Data Byte 0	Data Byte 1
0x000	Requested state	Addressed node

TABLE 3-3. Command code for requested NMT state

NMT command code	Meaning
0x01	Go to 'operational'
0x02	Go to 'stopped'
0x80	Go to 'pre-operational'
0x81	Go to 'reset node'
0x82	Go to 'reset communication'

TABLE 3-4. NMT states and communication objects

NMT states	Pre-operational	Operational	Stopped
PDO	X		
SDO	X	X	
SYNC	X	X	
TIME	X	X	
EMCY	X	X	
Node control and error control	X	X	X

CANopen Message Types

The controller operating in the CANopen mode can accept the following types of messages:

- Service Data Objects, or SDO messages to read/write parameter values
- Process Data Objects, or PDO mapped messages to automatically transmit parameters and/or accept commands at runtime
- Network Management, or NMT as defined in the CANopen specification

Service Data Object (SDO) Read/Write Messages

Runtime queries and runtime commands can be sent to the controller in real-time using the expedited SDO messages.

SDO messages provide generic access to Object Dictionary and can be used for obtaining parameter values on an irregular basis due to the excessive network traffic that is generated with each SDO request and response message.

The list of commands accessible with SDO messages can be found in the "Object Dictionary" on page 39.

Transmit Process Data Object (TPDO) Messages

Transmit PDO (TPDO) messages are one of the two types of PDO messages that are used during operation.

TPDOs are runtime operating parameters that are sent automatically on on a synchronous or asynchronous way, depending on the transmission type, from the controller to one or multiple nodes. TPDOs do not alter object data; they only read internal controller values and transmit them to the CAN bus.

TPDOs are identified on a CANopen network by the bit pattern in the 11-bit header of the CAN frame.

4 bits	7 bits
Object Type	NodeID

TPDO1: 0x180 + Node ID
TPDO2: 0x280 + Node ID
TPDO3: 0x380 + Node ID
TPDO4: 0x480 + Node ID
TPDO5: 0x000
TPDO6: 0x000
TPDO7: 0x000
TPDO8: 0x000

Roboteq Products allow up to eight TPDOs for any node ID. Unless otherwise specified in the product datasheet, by default, TPDO1 to TPDO8 are used to transmit user variables which may be loaded with any operating parameters using MicroBasic scripting. Each of the 8 TPDO can be mapped with any mappable SDO query. For more details see chapter PDO Mapping below.

Each of the 8 TPDOs can be configured to be sent based on the transmission type (See "CTPS - CANOpen Transmission Type" in "Roboteq Controllers User Manual") or through object 0x1800 05. If the default transmission type is selected (255) then the TPDOs are sent at user defined periodic intervals. This is done using the CTPS parameter (See "CTPS - CANOpen TPDO Send Rate" in "Roboteq Controllers User Manual") or through object 0x1800 02.

TABLE 3-5. Commands mapped on TPDOs

TPDO	Object Index-Sub	Size	Default Object Mapped
TPDO1	0x2106-1	S32	User VAR 1
	0x2106-2		User VAR 2
TPDO2	0x2106-3	S32	User VAR 3
	0x2106-4		User VAR 4
TPDO3	0x2106-5	S32	User VAR 5
	0x2106-6		User VAR 6
TPDO4	0x2106-7	S32	User VAR 7
	0x2106-8		User VAR 8
TPDO5	0x2106-25	S32	User VAR 25
	0x2106-26		User VAR 26
TPDO6	0x2106-27	S32	User VAR 27
	0x2106-28		User VAR 28
TPDO7	0x2106-29	S32	User VAR 29
	0x2106-30		User VAR 30
TPDO8	0x2106-31	S32	User VAR 31
	0x2106-32		User VAR 32
S32: signed 32-bit word			

Receive Process Data Object (RPDO) Messages

RPDOs are configured to capture runtime data destined to the controller.

RPDOs are CAN frames identified by their 11-bit header.

4 bits	7 bits
Object Type	NodeID

RPDO1: 0x200 + Node ID
RPDO2: 0x300 + Node ID
RPDO3: 0x400 + Node ID
RPDO4: 0x500 + Node ID
RPDO5: 0x000
RPDO6: 0x000
RPDO7: 0x000
RPDO8: 0x000

Roboteq CANopen implementation supports RPDOs. By default, data received using RPDOs are stored in user variables from where they can be processed using MicroBasic scripting. Each of the 8 RPDO can be mapped with any mappable object. For more details see chapter PDO Mapping below.

TABLE 3-6. Commands mapped on RPDOs

RPDO	Object Index-Sub	Size	Default Object Mapped
RPDO1	0x2005-9	S32	User VAR 9
	0x2005-10		User VAR 10
RPDO2	0x2005-11	S32	User VAR 11
	0x2005-12		User VAR 12
RPDO3	0x2005-13	S32	User VAR 13
	0x2005-14		User VAR 14
RPDO4	0x2005-15	S32	User VAR 15
	0x2005-16		User VAR 16
RPDO5	0x2005-17	S32	User VAR 17
	0x2005-18		User VAR 18
RPDO6	0x2005-19	S32	User VAR 19
	0x2005-20		User VAR 20
RPDO7	0x2005-21	S32	User VAR 21
	0x2005-22		User VAR 22
RPDO8	0x2005-23	S32	User VAR 23
	0x2005-24		User VAR 24
S32: signed 32-bit word			

PDO Mapping

The Process Data Object (PDO) service allows exchanging one or several process variables in one single CAN message. The PDO mapping parameter describes which objects in the CANopen object dictionary are transmitted by the producer. The PDO consumer uses also a PDO mapping parameter, which specifies where to write the received process data in the CANopen object dictionary. The PDO mapping parameter of the producer and the consumer may use different pointers (16-bit index and 8-bit sub-index) depending on the CANopen profile (in the Roboteq controllers dynamic mapping is supported).

In some simple devices, the user does not have the possibility to configure the PDO mapping parameters. This is called static PDO mapping, but our controllers provide variable PDO mapping. This means the system designer can re-configure the PDO mapping. Normally, this is done in the NMT pre-operational state, when the PDOs are disabled. Of course, the user can also reconfigure the PDO mapping in the NMT operational state, but then it is necessary to avoid inconsistencies in the PDO mapping on the producer and the consumer side. To avoid this, the PDO must not be produced until the entire reconfiguration is finished.

The CiA 301 application layer specification requires a dedicated re-mapping procedure:

1. "Destroy" the PDO by setting the valid bit to 1_b of sub-index 01_h of the PDO communication parameter.
2. Disable PDO mapping by setting the sub-index 00h of the PDO mapping parameter to 00_h.

3. Modify PDO mapping by changing the values of the corresponding sub-indices of the PDO mapping parameters.
4. Enable PDO mapping by setting the sub-index 00_h to the number mapped process data.
5. "Create" a PDO by setting the valid bit to 0_b of sub-index 01_h of the PDO communication parameter.

If the controller detects that the index and subindex of the mapped object does not exist or the object cannot be mapped during step 3, the controller responds with the SDO abort transfer service (abort code: 06020000_h or 06040041_h). If the controller detects that the RPDO mapping is not valid or not possible during step 4, the controller responds with the SDO abort transfer service (abort code: 06020000_h or 06040042_h).
In the example below, the PDO producer device has the node-id 01.

1. Destroy the TPDO1 by setting the invalid bit of COB-ID (0x1800-01):

TABLE 3-7. Destroy TPDO1

Frame	23	00 18	01	81 01	00 C0
Description	write request for 4 bytes	TPDO1 communication parameter	Sub-index	Object id	Destroy TPDO

2. Disable TPDO1 mapping by setting the number of mapping parameter entries to 00

TABLE 3-8. Disable TPDO1

Frame	23	00 1A	00	00 00	00 00
Description	write request for 4 bytes	TPDO1 mapping parameter			

3. Set objects for TPDO1:

- $0x2100_{(01)}$: Motor amps for channel 1 (S16).

TABLE 3-9. Set Objects for TPDO1

Frame	23	00 1A	01	10	01	00 21
Description	write request for 4 bytes	TPDO1 mapping parameter	Mapping order Length of object Sub index			Object id

- $0x2100_{02}$: Motor amps for channel 2 (S16).

TABLE 3-10. Map Motor amps for channel 2

Frame	23	00 1A	02	10	02	00 21
Description	write request for 4 bytes	TPDO1 mapping parameter	Mapping order	Length of object	Sub index	Object id

- $0x210D_{01}$: Internal voltage (U16).

TABLE 3-11. Map Internal Voltage

Frame	23	00 1A	03	10	01	0D 21
Description	write request for 4 bytes	TPDO1 mapping parameter	Mapping order	Length of object	Sub index	Object id

- $0x210F_{01}$: MCU temperature (S8).

TABLE 3-12. Map MCU Temperature

Frame	23	00 1A	04	10	01	0F 21
Descrip-tion	write request for 4 bytes	TPDO1 mapping parameter	Mapping order	Length of object	Sub index	Object id

- Set TPDO1 mapping parameters to the number of entries (04)

TABLE 3-13. Set TPDO1 number of entries

Frame	2F	00 1A	00	04	00 00 00
Descrip-tion	write request for 1 byte	TPDO1 mapping parameter	Sub index	Number of objects for remapping	

4. Enable TPDO1.

TABLE 3-14. Enable TPDO1

Frame	23	00 18	01	81 01	00 40
Descrip-tion	write request for 4 bytes	reset request for TPDO	Sub-index	CAN-ID	Object id

The following example explains the procedure of a RPDO remapping. Set object 0x6071 to RPDO1.

1. Destroy RPDO1 by setting the invalid bit of COB-ID (1400_{01}):

TABLE 3-15. Destroy RPDO1

Frame	23	00 14	01	01 02	00 80
Description	write request for 4 bytes	RPDO1 communication parameter	Sub-index	CAN-ID	Destroy RPDO

2. Disable RPDO1 mapping by setting number of mapping parameter entries to 00

TABLE 3-16. Disable RPDO1

Frame	23	00 16	00	00 00	00 00
Description	write request for 4 bytes	RPDO1 mapping parameter	set number of objects to 0		

3. Set objects for RPDO1:
   - $0x6071_{00}$: DS402 Target torque for channel 1 (S16).

TABLE 3-17. Map target torque for channel 1

Frame	23	00 16	01	10	00	71 60
Description	write request for 4 bytes	RPDO1 mapping parameter	Mapping order Length of object			Object id

4. Set RPDO1 mapping parameters to the number of entries (01):

TABLE 3-18. Set RPDO1 number of entries

Frame	2F	00 16	00	01	00 00 00
Description	write request for 1 byte	RPDO1 mapping parameter		Number of objects for remapping	

5. Create RPDO1 by setting the Valid bit of COB-ID to 0 (0x6071).

TABLE 3-19. Enable RPDO1

Frame	23	00 14	01	01 02	00 00
Description	write request for 4 bytes	reset request for RPDO	Sub-index	CAN-ID	Create RPDO

PDO Transmission Type

The transmission type of a PDO can be set via the second sub-index.

TABLE 3-20. PDO Transmission Types

0(1)	The Transmit PDO is synchronous. Which specific SYNC Object occurrence triggers the transmission is given in the device profile.
1 – 240	The Transmit PDO is synchronous. It is transmitted after every nth SYNC Object within the Synchronous Window Length, where n is the transmission type. For example, when using transmission type 34, the PDO is transmitted after every 34th SYNC Object.
241 – 251(1)	Reserved.
252(1)	The data for the PDO is updated on reception of a SYNC Object, but the PDO is not transmitted. The PDO is only transmitted on reception of a Remote Transmission Request.
253(1)	The data for the PDO is updated and the PDO is transmitted on reception of a Remote Transmission Request.
254(2)	The conditions that cause the Transmit PDO to be transmitted are manufacturer specific.
255	The Transmit PDO is asynchronous. The transmission is triggered at defined send rate.

(1) Not supported in Roboteq controllers.
(2) In Roboteq controllers, it behaves exactly like value 255.

First it is necessary to distinguish between synchronous and asynchronous PDOs:
Asynchronous PDOs are event-controlled and represent the normal transmission type of PDOs. For this, the values 255 or 254 are to be entered as PDO type.

Synchronous PDOs are only transmitted after prior reception of a synchronization message (Sync Object). PDO transmission is thus carried out synchronously in the entire network, more or less at the same time. But what is much more important is that all device inputs must be sampled on the arrival of the sync object, so that a uniform snapshot of the process results. With the next sync-message, the recorded data are then sent in the synchronous PDOs. Therefore, there is a delay here corresponding to the cycle time of the Sync message, as the consumers receive the process variables at the time of the previous Sync message. In output direction the synchronous PDOs received by a node only become valid on arrival of the next Sync message.
In order that the bus is not blocked up by a large number of synchronous PDOs, which are all sent with every Sync message, the values 1-240 of the cyclic synchronous PDO type are used as dividers for the transmission interval. Accordingly, [18xxsub02] = 4 means that the synchronous PDO is only sent with every fourth Sync message.

Object Dictionary

The CANopen dictionary shown in this section is subject to change. The CANopen EDS file is included in the firmware download folder.

The Object Dictionary given in the table below contains the runtime queries and runtime commands that can be accessed with SDO/PDO messages during controller operation.

Communication Profile

TABLE 3-21. Communication Profile Objects

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x1000	00	Device Type	U32	RO	No	
0x1001	00	Error Register	U8	RO	No	
0x1008	00	Manufacturer Device Name	STR	CONST	No	
0x1009	00	Manufacturer Hardware Version	STR	CONST	No	
0x100A	00	Manufacturer Software Version	STR	CONST	No	
0x100C	00	Guard Time	U16	RW	No	
0x100D	00	Life Time Factor	U8	RW	No	
0x1016	01-04	Consumer Heartbeat Time	U32	RW	No	
0x1017	00	Producer Heartbeat Time	U16	RW	No	
0x1018	01	Identity Object – Vendor ID	U32	CONST	No	
0x1400	01	Communication parameter for RPDO 1 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 1 Transmission Type	U8	RW	No	
0x1401	01	Communication parameter for RPDO 2 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 2 Transmission Type	U8	RW	No	
0x1402	01	Communication parameter for PDO 3 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 3 Transmission Type	U8	RW	No	
0x1403	01	Communication parameter for PDO 4 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 4 Transmission Type	U8	RW	No	
0x1404	01	Communication parameter for PDO 5 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 5 Transmission Type	U8	RW	No	
0x1405	01	Communication parameter for PDO 6 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 6 Transmission Type	U8	RW	No	
0x1406	01	Communication parameter for PDO 7 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 7 Transmission Type	U8	RW	No	
0x1407	01	Communication parameter for PDO 8 COB-ID	U32	RW	No	
	02	Communication parameter for RPDO 8 Transmission Type	U8	RW	No	

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x1600	01	Mapping Parameter for RPDO 1 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 1 - PDO Mapping Entry	U32	RW	No	
0x1601	01	Mapping Parameter for RPDO 2 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 2 - PDO Mapping Entry	U32	RW	No	
0x1602	01	Mapping Parameter for RPDO 3 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 3 - PDO Mapping Entry	U32	RW	No	
0x1603	01	Mapping Parameter for RPDO 4 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 4 - PDO Mapping Entry	U32	RW	No	
0x1604	01	Mapping Parameter for RPDO 5 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 5 - PDO Mapping Entry	U32	RW	No	
0x1605	01	Mapping Parameter for RPDO 6 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 6 - PDO Mapping Entry	U32	RW	No	
0x1606	01	Mapping Parameter for RPDO 7 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 7 - PDO Mapping Entry	U32	RW	No	
0x1607	01	Mapping Parameter for RPDO 8 - PDO Mapping Entry	U32	RW	No	
	02	Mapping Parameter for RPDO 8 - PDO Mapping Entry	U32	RW	No	
0x1800	01	Communication Parameter for TPDO 1 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 1 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 1 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 1 - Event Timer	U16	RW	No	
0x1801	01	Communication Parameter for TPDO 2 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 2 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 2 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 2 - Event Timer	U16	RW	No	
0x1802	01	Communication Parameter for TPDO 3 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 3 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 3 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 3 - Event Timer	U16	RW	No	

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x1803	01	Communication Parameter for TPDO 4 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 4 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 4 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 4 - Event Timer	U16	RW	No	
0x1804	01	Communication Parameter for TPDO 5 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 5 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 5 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 5 - Event Timer	U16	RW	No	
0x1805	01	Communication Parameter for TPDO 6 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 6 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 6 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 6 - Event Timer	U16	RW	No	
0x1806	01	Communication Parameter for TPDO 7 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 7 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 7 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 7 - Event Timer	U16	RW	No	
0x1807	01	Communication Parameter for TPDO 8 - COB-ID	U32	RW	No	
	02	Communication Parameter for TPDO 8 - Transmission Type	U8	RW	No	
	03	Communication Parameter for TPDO 8 - Inhibit Time	U16	RW	No	
	05	Communication Parameter for TPDO 8 - Event Timer	U16	RW	No	
0x1A00	01	Mapping Parameter - PDO Mapping for TPDO1 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO1 Entry - Qry	U32	RW	No	
0x1A01	01	Mapping Parameter - PDO Mapping for TPDO2 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO2 Entry - Qry	U32	RW	No	
0x1A02	01	Mapping Parameter - PDO Mapping for TPDO3 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO3 Entry - Qry	U32	RW	No	

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x1A03	01	Mapping Parameter - PDO Mapping for TPDO4 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO4 Entry - Qry	U32	RW	No	
0x1A04	01	Mapping Parameter - PDO Mapping for TPDO5 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO5 Entry - Qry	U32	RW	No	
0x1A05	01	Mapping Parameter - PDO Mapping for TPDO6 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO6 Entry - Qry	U32	RW	No	
0x1A06	01	Mapping Parameter - PDO Mapping for TPDO7 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO7 Entry - Qry	U32	RW	No	
0x1A07	01	Mapping Parameter - PDO Mapping for TPDO8 Entry-Qry	U32	RW	No	
	02	Mapping Parameter - PDO Mapping for TPDO8 Entry - Qry	U32	RW	No	

Runtime Commands

TABLE 3-22. Runtime Commands Objects

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x2000	01-mm(1)	Set Motor Command	S32	WO	Yes	CG
0x2001	01-mm(1)	Set Position	S32	WO	Yes	P
0x2002	01-mm(1)	Set Velocity	S32	WO	Yes	S
0x2003	01-ee(2)	Set Encoder Counter	S32	WO	Yes	C
0x2004	01-mm(1)	Set Brushless Counter	S32	WO	Yes	CB
0x2005	01-vv(3)	Set User Integer Variable	S32	WO	Yes	VAR
0x2006	01-mm(1)	Set Acceleration	S32	WO	Yes	AC
0x2007	01-mm(1)	Set Deceleration	S32	WO	Yes	DC
0x2008	00	Set All Digital Out bits	U8	WO	Yes	DS
0x2009	00	Set Individual Digital Out bits	U8	WO	Yes	D1
0x200A	00	Reset Individual Digital Out bits	U8	WO	Yes	D0
0x200B	01-ee(2)	Load Home Counter	U8	WO	Yes	H
0x200C	00	Emergency Shutdown	U8	WO	Yes	EX
0x200D	00	Release Shutdown	U8	WO	Yes	MG
0x200E	00	Stop in all modes	U8	WO	Yes	MS
0x200F	01-mm(1)	Set Pos Relative	S32	WO	Yes	PR

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x2010	01-mm(1)	Set Next Pos Absolute	S32	WO	Yes	PX
0x2011	01-mm(1)	Set Next Pos Relative	S32	WO	Yes	PRX
0x2012	01-mm(1)	Set Next Acceleration	S32	WO	Yes	AX
0x2013	01-mm(1)	Set Next Deceleration	S32	WO	Yes	DX
0x2014	01-mm(1)	Set Next Velocity	S32	WO	Yes	SX
0x2015	01-bb(4)	Set User Boolean Variable	U32	WO	Yes	B
0x2016	01-rr(5)	Set RC Pulse Out	S32	WO	Yes	RS
0x2017	00	Save Config to Flash	U8	WO	Yes	EES
0x2018	00	Run MicroBasic Script	U8	WO	Yes	R
0x201F	01-si(6)	Set Absolute SSI Counter	S32	WO	Yes	CSS
0x202C	01-mm(1)	Safety Stop	U8	WO	Yes	SFT
0x202D	01-mm(1)	Motor Sensor Setup	U8	WO	Yes	MSS
0x2034	01-mm(1)	Brake Override	U8	WO	Yes	BRK

(1) mm: Maximum number of motors.
(2) ee: Maximum number of encoders.
(3) vv: Maximum number of integer variables.
(4) bb: Maximum number of boolean variables.
(5) rr: Maximum number of RC pulse output.
(6) si: Maximum number of SSI encoders.

Runtime Queries

TABLE 3-23. Runtime Queries Objects

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x2100	01-mm(1)	Read Motor Amps	S16	RO	Yes	A
0x2101	01-mm(1)	Read Actual Motor Command	S32	RO	Yes	M
0x2102	01-mm(1)	Read Applied Power Level	S16	RO	Yes	P
0x2103	01-ee(2)	Read Encoder Motor Speed	S32	RO	Yes	S
0x2104	01-ee(2)	Read Absolute Encoder Counter	S32	RO	Yes	C
0x2105	01-mm(1)	Read Absolute Brushless	S32	RO	Yes	CB
0x2106	01-vv(3)	Read User Integer Variable	S32	RO	Yes	VAR
0x2107	01-ee(2)	Read Relative Encoder Motor Speed	S16	RO	Yes	SR
0x2108	01-ee(2)	Read Encoder Count Relative	S32	RO	Yes	CR
0x2109	0-mm(1)	Read Brushless Count Relative	S32	RO	Yes	BCR
0x210A	01-mm(1)	Read BL Motor Speed in RPM	S16	RO	Yes	BS
0x210B	01-mm(1)	Read Relative BL Motor Speed	S16	RO	Yes	BSR
0x210C	01-mm(1)	Read Battery Amps	S16	RO	Yes	BA
0x210D	01	Read Internal Voltages	U16	RO	Yes	V
	02	Read Internal Voltages (Battery)	U16	RO	Yes	V
	03	Read Internal Voltages (5Vout)	U16	RO	Yes	V

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x210E	00	Read All Digital Inputs	U32	RO	Yes	D
0x210F	01-tt(5) + 1	Read MCU temperature (01) and each transistor temperature (02, 03, …).	S8	RO	Yes	T
0x2110	01-mm(1)	Read Feedback	S32	RO	Yes	F
0x2111	00	Read Status Flags	U16	RO	Yes	FS
0x2112	00	Read Fault Flags	U16	RO	Yes	FF
0x2113	00	Read Current Digital Outputs	U16	RO	Yes	DO
0x2114	01-mm(1)	Read Closed Loop Error	S32	RO	Yes	E
0x2115	01-bb(4)	Read User Boolean Variable	U32	RO	Yes	B
0x2116	01-mm(1)	Read Internal Serial Command	S32	RO	Yes	CIS
0x2117	01-mm(1)	Read Internal Analog Command	S32	RO	Yes	CIA
0x2118	01-mm(1)	Read Internal Pulse Command	S32	RO	Yes	CIP
0x2119	00	Read Time	U32	RO	Yes	TM
0x211A	01-kk(6)	Read Spektrum Radio Capture	U16	RO	Yes	K
0x211B	01-mm(1)	Destination Pos Reached Flag	U8	RO	Yes	DR
0x211C	01-ma(7)	Read Field Oriented Control Motor Amps	S32	RO	Yes	MA
0x211D	01-mg(8)	Read Magsensor Track Detect	U8	RO	Yes	MGD
0x211E	01-3×mg(8)	Read Magsensor Track Position (Left, Right, and Active Track)	S16	RO	Yes	MGT
0x211F	01-2×mg(8)	Read Magsensor Markers (Left and Right)	U8	RO	Yes	MGM
0x2120	01-mg(8)	Read Magsensor Status	U16	RO	Yes	MGS
0x2121	01-mg(8)	Read Magsensor Gyroscope	S16	RO	Yes	MGY
0x2122	01-mm(1)	Read Motor Status Flags	U16	RO	Yes	FM
0x2123	01-mm(1)	Read Hall Sensor States	U8	RO	Yes	HS
0x2124	00	Read Lock Status	U8	RO	Yes	LK
0x2125	01-mm(1)	Read Destination Tracking	S32	RO	Yes	TR
0x2132	01-mm(1)	Read Rotor Angle	S16	RO	Yes	ANG
0x2133	00	Read Script Checksum	U32	RO	Yes	SCC
0x2134	00	Read Node Is Alive	U8	RO	Yes	ICL
0x2135	01-mm(1)	Read FOC Angle Correction	S16	RO	Yes	FC
0x2136	01-ii(9)	Read AC Induction Slip	S16	RO	Yes	SL
0x2137	01	Read Firmware Version	U16	RO	Yes	FIN
	02	Read Firmware Month	U16	RO	Yes	FIN
	03	Read Firmware Day	U16	RO	Yes	FIN
	04	Read Firmware Year	U16	RO	Yes	FIN
0x2138	01-mg(8)	Read MagSensor Cross Tape	U8	RO	Yes	MGX
0x213A	00	Read BMS Battery's State of Charge	U8	RO	Yes	BSC
0x213C	01-si(10)	Read SSI Sensor Motor Speed	S32	RO	Yes	SS

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x213D	01-si(10)	Read Relative SSI Sensor Motor Speed	S16	RO	Yes	SSR
0x213E	01-si(10)	Read SSI Absolute Counter	S32	RO	Yes	CSS
0x213F	01-si(10)	Read SSI Relative Counter	S32	RO	Yes	CSR
0x2141	00	Read BMS State of Charge	U8	RO	Yes	BMC
0x2142	00	Read BMS Status Flags	U8	RO	Yes	BMF
0x2143	00	Read BMS Operational State	U8	RO	Yes	BMS
0x2145	01-13	Read Digital Inputs	BOOL	RO	Yes	DI
0x2146	01-di(11)	Read Analog Inputs	S16	RO	Yes	AI
0x2147	01-di(11)	Read Analog Inputs Converted	S16	RO	Yes	AIC
0x2148	01-pi(11)	Read Pulse Inputs	U16	RO	Yes	PI
0x2149	01-pi(11)	Read Pulse Inputs Converted	S16	RO	Yes	PIC
0x214A	01-fs(11)	Read FlowSensor	S32	RO	Yes	FLW
0x2161	01-mm(1)	Brake Override	U8	RO	Yes	BRK

(1) mm: Maximum number of motors.
(2) ee: Maximum number of encoders.
(3) vv: Maximum number of integer variables.
(4) bb: Maximum number of boolean variables.
(5) tt: Maximum number of internal temperature sensors.
(6) kk: Maximum number of spectrum radio.
(7) ma: Maximum number of Amps Sensors.
(8) mg: Maximum number of magnetic sensors.
(9) ii: Maximum number of AC induction motors.
(10) si: Maximum number of SSI sensors.
(11) di: Maximum number of digital inputs.
(12) pi: Maximum number of pulse inputs.
(13) fs: Maximum number of flow sensors.

DS402 Profile

TABLE 3-24. DS402 Objects

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x6040	00	Control Word CH1	U16	RW	Yes	CW
0x6041	00	Status Word CH1	U16	RO	Yes	SW
0x6042	00	Target Velocity CH1	S16	RW	Yes	S16
0x6043	00	VL Velocity Demand CH1	S16	RO	Yes	RMP
0x6044	00	VL Velocity Actual Value CH1	S16	RO	Yes	SPE
0x6046	01	VL Velocity Min Amount CH1	U32	RW	Yes	SPL
	02	VL Velocity Max Amount CH1	U32	RW	Yes	SPL
0x6048	01	Velocity Acceleration Delta Speed CH1	U32	RW	Yes	SAC
	02	Velocity Acceleration Delta Time CH1	U16	RW	Yes	SAC
0x6049	01	Velocity Deceleration Delta Speed CH1	U32	RW	Yes	SDC
	02	Velocity Deceleration Delta Time CH1	U16	RW	Yes	SDC

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x6060	00	Modes of Operation CH1	S8	RW	Yes	ROM
0x6061	00	Modes of Operation Display CH1	S8	RO	No	AOM
0x6064	00	Position Actual Value CH1	S32	RO	Yes	PST
0x6065	00	Following Error Window CH1	U32	RW	Yes	FEW
0x6066	00	Following Error Time Out CH1	U16	RW	Yes	FET
0x6069	00	Velocity Sensor Actual Value CH1	S32	RO	Yes	VSA
0x606B	00	Velocity Demand Value CH1	S32	RO	Yes	VDV
0x606C	00	Velocity Actual Value CH1	S32	RO	Yes	SPE
0x6071	00	Target Torque CH1	S16	RW	Yes	TC
0x6077	00	Torque Actual Value CH1	S16	RO	Yes	TRQ
0x607A	00	Target Position CH1	S32	RW	Yes	POS
0x607D	01	Software Position Limit (min value) CH1	S32	RW	Yes	PLT
	02	Software Position Limit (max value) CH1	S32	RW	Yes	PLT
0x6080	00	Max Motor Speed CH1	U32	RW	Yes	MSL
0x6081	00	Profile Velocity CH1	U32	RW	Yes	PSP
0x6083	00	Profile Acceleration CH1	U32	RW	Yes	PAC
0x6084	00	Profile Deceleration CH1	U32	RW	Yes	PDC
0x6085	00	Quick Stop Deceleration	U32	RW	Yes	EDC
0x6087	00	Torque Slope CH1	U32	RW	Yes	TSL
0x60B0	00	Position Offset CH1	S32	RW	Yes	POF
0x60B1	00	Velocity Offset CH1	S32	RW	Yes	VOF
0x60B2	00	Torque Offset CH1	S32	RW	Yes	TOF
0x60C2	01	Interpolation Time Period Value CH1	U8	RW	Yes	INT
	02	Interpolation Time Index CH1	S8	RW	Yes	INT
0x60FF	00	Target Profile Velocity CH1	S32	RW	Yes	SPC
0x6502	00	Supported Drive Modes CH1	U32	CONST	No	SDM
0x67FE	00	Version Number CH1	U32	CONST	No	VNM
0x6840	00	Control Word CH2	U16	RW	Yes	CW
0x6841	00	Status Word CH2	U16	RO	Yes	SW
0x6842	00	Target Velocity CH2	S16	RW	Yes	S16
0x6843	00	VL Velocity Demand CH2	S16	RO	Yes	RMP
0x6844	00	VL Velocity Actual Value CH2	S16	RO	Yes	SPE
0x6846	01	VL Velocity Min Amount CH2	U32	RW	Yes	SPL
	02	VL Velocity Max Amount CH2	U32	RW	Yes	SPL
0x6848	01	Velocity Acceleration Delta Speed CH2	U32	RW	Yes	SAC
	02	Velocity Acceleration Delta Time CH2	U16	RW	Yes	SAC
0x6849	01	Velocity Deceleration Delta Speed CH2	U32	RW	Yes	SDC
	02	Velocity Deceleration Delta Time CH2	U16	RW	Yes	SDC
0x6860	00	Modes of Operation CH2	S8	RW	Yes	ROM
0x6861	00	Modes of Operation Display CH2	S8	RO	No	AOM
0x6864	00	Position Actual Value CH2	S32	RO	Yes	PST

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x6865	00	Following Error Window CH2	U32	RW	Yes	FEW
0x6866	00	Following Error Time Out CH2	S32	RW	Yes	FET
0x6869	00	Velocity Sensor Actual Value CH2	S32	RO	Yes	VSA
0x686B	00	Velocity Demand Value CH2	S32	RO	Yes	VDV
0x686C	00	Velocity Actual Value CH2	S32	RO	Yes	SPE
0x6871	00	Target Torque CH2	S16	RW	Yes	TC
0x6877	00	Torque Actual Value CH2	S16	RO	Yes	TRQ
0x687A	00	Target Position CH2	S32	RW	Yes	POS
0x687D	01	Software Position Limit CH2 (min value)	S32	RW	Yes	PLT
	02	Software Position Limit CH2 (max value)	S32	RW	Yes	PLT
0x6880	00	Max Motor Speed CH2	U32	RW	Yes	MSL
0x6881	00	Profile Velocity CH2	U32	RW	Yes	PSP
0x6883	00	Profile Acceleration CH2	U32	RW	Yes	PAC
0x6884	00	Profile Deceleration CH2	U32	RW	Yes	PDC
0x6885	00	Quick Stop Deceleration CH2	U32	RW	Yes	EDC
0x6887	00	Torque Slope CH2	U32	RW	Yes	TSL
0x68B0	00	Position Offset CH2	S32	RW	Yes	POF
0x68B1	00	Velocity Offset CH2	S32	RW	Yes	VOF
0x68B2	00	Torque Offset CH2	S32	RW	Yes	TOF
0x68C2	01	Interpolation Time Period Value CH2	U8	RW	Yes	INT
	02	Interpolation Time Index CH2	S8	RW	Yes	INT
0x68FF	00	Target Profile Velocity CH2	S32	RW	Yes	SPC
0x6D02	00	Supported Drive Modes CH2	U32	CONST	No	SDM
0x6FFE	00	Version Number CH2	U32	CONST	No	VNM
0x7040	00	Control Word CH3	U16	RW	Yes	CW
0x7041	00	Status Word CH3	U16	RO	Yes	SW
0x7042	00	Target Velocity CH3	S16	RW	Yes	S
0x7043	00	VL Velocity Demand CH3	S16	RO	Yes	RMP
0x7044	00	VL Velocity Actual Value CH3	S16	RO	Yes	SPE
0x7046	01	VL Velocity Min Amount CH3	U32	RW	Yes	SPL
	02	VL Velocity Max Amount CH3	U32	RW	Yes	SPL
0x7048	01	Velocity Acceleration Delta Speed CH3	U32	RW	Yes	SAC
	02	Velocity Acceleration Delta Time CH3	U16	RW	Yes	SAC
0x7049	01	Velocity Deceleration Delta Speed CH3	U32	RW	Yes	SDC
	02	Velocity Deceleration Delta Time CH3	U16	RW	Yes	SDC
0x7060	00	Modes of Operation CH3	S8	RW	Yes	ROM
0x7061	00	Modes of Operation Display CH3	S8	RO	No	AOM
0x7064	00	Position Actual Value CH3	S32	RO	Yes	PST

Index	Sub (hex)	Entry Name	Type	Access	PDO	Command
0x7065	00	Following Error Window CH3	U32	RW	Yes	FEW
0x7066	00	Following Error Time Out CH3	S32	RW	Yes	FET
0x7069	00	Velocity Sensor Actual Value CH3	S32	RO	Yes	VSA
0x706B	00	Velocity Demand Value CH3	S32	RO	Yes	VDV
0x706C	00	Velocity Actual Value CH3	S32	RO	Yes	SPE
0x7071	00	Target Torque CH3	S16	RW	Yes	TC
0x7077	00	Torque Actual Value CH3	S16	RO	Yes	TRQ
0x707A	00	Target Position CH3	S32	RW	Yes	POS
0x707D	01	Software Position Limit CH3 (min value)	S32	RW	Yes	PLT
	02	Software Position Limit CH3 (max value)	S32	RW	Yes	PLT
0x7080	00	Max Motor Speed CH3	U32	RW	Yes	MSL
0x7081	00	Profile Velocity CH3	U32	RW	Yes	PSP
0x7083	00	Profile Acceleration CH3	U32	RW	Yes	PAC
0x7084	00	Profile Deceleration CH3	U32	RW	Yes	PDC
0x7085	00	Quick Stop Deceleration CH3	U32	RW	Yes	EDC
0x7087	00	Torque Slope CH3	U32	RW	Yes	TSL
0x70B0	00	Position Offset CH3	S32	RW	Yes	POF
0x70B1	00	Velocity Offset CH3	S32	RW	Yes	VOF
0x70B2	00	Torque Offset CH3	S32	RW	Yes	TOF
0x78C2	01	Interpolation Time Period Value CH3	U8	RW	Yes	INT
	02	Interpolation Time Index CH3	S8	RW	Yes	INT
0x70FF	00	Target Profile Velocity CH3	S32	RW	Yes	SPC
0x7502	00	Supported Drive Modes CH3	U32	CONST	No	SDM
0x77FE	00	Version Number CH3	U32	CONST	No	VNM

SDO Construction Details

CANOpen SDO frames can easily be created manually and used to send commands and queries to a Roboteq device. The directives below are a simplified description of the CANOpen SDO protocol. For more details please advise the CANOpen standard.

A CANOpen command/query towards a Roboteq device can be analyzed as shown below:

TABLE 3-25. SDO Request

		Payload					
		Byte0					
Header	DLC	bits 4-7	bits2-3	bits0-1	Byte1-2	Byte 3	Bytes4-7
0x600+nd	8	css	n	xx	index	subindex	data

- nd is the destination node id.
- ccs is the Client Command Specifier, if 2 it is command if 4 it is query.
- n is the Number of bytes in the data part, which do not contain data

- xx not necessary for basic operation. For more details advise CANOpen standard.
- index is the object dictionary index of the data to be accessed
- subindex is the subindex of the object dictionary variable
- data contains the data to be uploaded.

The Response from the roboteq device is as shown below:

TABLE 3-26. SDO Response

		Payload					
		Byte0					
Header	DLC	bits 4-7	bits2-3	bits0-1	Byte1-2	Byte 3	Bytes4-7
0x580+nd	8	css	n	xx	index	subindex	Data

- nd is the source node id.
- ccs is the Client Command Specifier, if 4 it is query response, 6 it is a successful response to command, 8 is an error in message received.
- n is the Number of bytes in the data part, which do not contain data
- xx not necessary for the simplistic way. For more details advise CANOpen standard.
- index is the object dictionary index of the data to be accessed.
- subindex is the subindex of the object dictionary variable
- data contains the data to be uploaded. Applicable only if css=4.

SDO Example 1: Set Encoder Counter 2 (C) of node 1 value 10

- nd = 1, since the destination's node id is 1.
- ccs = 2, since it is a command.
- n = 0 since all 4 bytes of the data are used (signed32).
- index = 0x2003 and subindex = 0x02 according to object dictionary.

TABLE 3-27. Set Encoder Counter 2 Request

		Payload					
		Byte0					
Header	DLC	bits 4-7	bits2-3	bits0-1	Byte1-2	Byte 3	Bytes4-7
0x600+1	8	2	0	0	0x2003	0x02	0x0A
601h	8	20			03 20	02	0A 00 00 00

The respective response will be:

TABLE 3-28. Set Encoder Counter 2 Response

		Payload					
		Byte0					
Header	DLC	bits 4-7	bits2-3	bits0-1	Byte1-2	Byte 3	Bytes4-7
0x580+1	8	6	0	0	0x2003	0x02	0x00
581h	8	60			03 20	02	00 00 00 00

SDO Example 2: Activate emergency shutdown (EX) for node 12

- nd = 12, since the destination's node id is 12.
- ccs = 2, since it is a command.
- n = 3 since only one byte of the data is used (unsigned8).
- index = 0x200C and subindex = 0x00 according to object dictionary.

TABLE 3-29. Activate Emergency Shutdown Request

		Payload					
		Byte0					
Header	**DLC**	**bits 4-7**	**bits2-3**	**bits0-1**	**Byte1-2**	**Byte 3**	**Bytes4-7**
0x600+12	8	2	3	0	0x200C	0x00	0x01
601Ch	8	2C			0C 20	00	01 00 00 00

The respective response will be:

TABLE 3-30. Activate Emergency Shutdown Response

		Payload					
		Byte0					
Header	**DLC**	**bits 4-7**	**bits2-3**	**bits0-1**	**Byte1-2**	**Byte 3**	**Bytes4-7**
0x580+1	8	6	0	0	0x200C	0x00	0x00
58Ch	8	60			0C 20	00	00 00 00 00

SDO Example 3: Read Battery Volts (V) of node 1.

- nd = 1, since the destination's node id is 1.
- ccs = 4, since it is a query.
- n = 2 since 2 bytes of the data are used (unsigned16).
- index = 0x210D and subindex = 0x02 according to object dictionary.

TABLE 3-31. Read Battery Volts Request

		Payload					
		Byte0					
Header	**DLC**	**bits 4-7**	**bits2-3**	**bits0-1**	**Byte1-2**	**Byte 3**	**Bytes4-7**
0x600+1	8	4	2	0	0x210D	0x02	0x00
601h	8	48			0D 21	02	0A 00 00 00

The respective response will be:

- nd = 1, since the source node id is 1.
- ccs = 4, since it is a query response.
- n = 2 since 2 bytes of the data are used (unsigned16).
- index = 0x210D and subindex = 0x02 according to object dictionary.
- data = 0x190 = 400 = 40 Volts.

TABLE 3-32. Read Battery Volts Response

Header	DLC	Payload					
		Byte0					
		bits 4-7	bits2-3	bits0-1	Byte1-2	Byte 3	Bytes4-7
0x580+1	8	4	2	xx	0x210D	0x02	0x190
581h	8	48			0D 21	02	90 01 00 00

SDO Example 4: Send Motor Command (CG)

- nd = 1, since the destination's node id is 1.
- ccs = 2 since it is a command.
- n = 3 since no bytes of the data are used (signed 32).
- index = 0x2100 and subindex = 0x00 according to object dictionary.

TABLE 3-33. Send Motor Command Request

Header	DLC	Payload					
		Byte0					
		bits 4-7	bits2-3	bits0-1	byte 1-2	byte 3	bytes 4-7
0x600+nd	8	2	0	0	0x2000	0x01	0x64
601	8	23			00 20	01	64 00 00 00

The respective response will be:

TABLE 3-34. Send Motor Command Response

Header	DLC	Payload					
		Byte0					
		bits 4-7	bits2-3	bits0-1	byte 1-2	byte 3	bytes 4-7
0x580+1	8	6	0	0	0x2000	0x00	0x00
581h	8	60			00 20	0x00	00 00 00 00

The following table shows the summary selection of bits 0-8 of Byte 0.

TABLE 3-35. Selection of bits 0-8 of Byte 0

Command	Access to Data 1- Data 4			
	4 byte data (5th - 8th byte)	2 byte data (5th - 6th byte)	1 byte data (5th byte)	Block
	hex	hex	hex	hex
Write request (Send Parameters to drive)	23	2B	2F	Writing not possible
Write Response (Controller response to the write request (acknowledgement))	60	60	60	

Command	Access to Data 1- Data 4			
	4 byte data (5th - 8th byte)	2 byte data (5th - 6th byte)	1 byte data (5th byte)	Block
	hex	hex	hex	hex
Read Request (Request to read a parameter from the drive)	40	40	40	40
Read Response (Response to the read request with an actual value)	43	4B	4F	41
Error Response (The controller indicates a communication error)	80	80	80	80

SECTION 4

DS402 Implementation on Roboteq Motor Controllers

Abbreviations

C	Constant
CiA	CAN in Automation
FSA	Finite State Automation
NMT	Network Management
PDS	Power Drive System
PP	Profile Position Mode
PV	Profile Velocity Mode
RO	Read Only
RW	Read Write
SDO	Service Data Object
TQ	Torque Mode
VL	Velocity Mode
CSV	Cyclic Synchronous velocity Mode
CSP	Cyclic Synchronous Position Mode
CST	Cyclic Synchronous Torque
U8	Unsigned byte
U16	Unsigned 2 bytes
U32	Unsigned 4 bytes
S8	Signed byte
S16	Signed 2 bytes
S32	Signed 4 bytes

Introduction

This documentation will describe the implementation of CiA DS402 standard on Roboteq motor controllers.

What is DS402

DS402 is an open standard, that is designed specifically for motion control. There are a number of CANOpen SDOs with which one can control the motor by commanding the motor controller.

The standard describes all the required SDOs, as long as the actions the motor controller should take upon receiving these SDOs. Additionally the standard describes a Finite State Machine (FSA) which should run on motor controller.

Implementation

The implementation has been directed under standard version 4.1.0.

Index Range & Channel Selection

The index range per channel is:

- 0x6000 - 0x67FF, for channel 1.
- 0x6800 - 0x6FFF, for channel 2.
- 0x7000 - 0x77FF, for channel 3.

There are Roboteq motor controllers with up to three channels available.

Modes of Operation

Roboteq Controllers support the following operation Modes:

A. Open Loop
B. Closed Loop Speed, controls Speed using Speed as feedback.
C. Closed Loop Speed Position, controls Speed using Position as feedback.
D. Closed Loop Count Position, controls Position.
E. Closed Loop Position Relative, controls Position within specific boundaries.
F. Closed Loop Position Tracking, controls Position within specific boundaries, with abrupt transition.

In order to conform the above operation modes to the operation modes described, the DS402 modes of operation supported by Roboteq are shown in Table 4-1 - Operation Table 1. Any other mode described in DS402 standard is not supported by Roboteq controllers.

TABLE 4-1. Operation Modes

Value	Definition	Roboteq Operation Mode
-4[1]	Velocity Mode	Closed Loop Speed Position
-3[1]	Profile Velocity Mode	Closed Loop Speed Position
-2[1]	Profile Position Mode	Closed Loop Position Tracking Mode[2]
-1[1]	Profile Position Mode	Closed Loop Position Relative Mode[2]

Value	Definition	Roboteq Operation Mode
0	No Mode	Open Loop Mode
1	Profile Position Mode	Closed Loop Count Position Mode
2	Velocity Mode	Closed Loop Speed Mode
3	Profile Velocity Mode	Closed Loop Speed Mode
4	Torque Profile Mode	Closed Loop torque Mode
8	Cyclic Synchronous Position Mode	Closed Loop Count Position Mode
9	Cyclic Synchronous Velocity Mode	Closed Loop Speed Mode
10	Cyclic Synchronous Torque Mode	Closed Loop Torque Mode
[1]Roboteq Specific Modes [2]Not all Profile Position features can be supported with this mode.		

Supported SDOs

Table 4-2 shows the SDOs described in DS402 standard and supported by Roboteq Motor Controllers.

TABLE 4-2. Supported SDO

	Description	Roboteq Command	Profile Position	Velocity	Profile Velocity	Torque Profile	Cyclic Sync Velocity	Cyclic Sync Torque	Cyclic Sync Position
6040_{00}	Control Word	CW	✓	✓	✓	✓	✓	✓	✓
6041_{00}	Status Word	SW	✓	✓	✓	✓	✓	✓	✓
6042_{00}	Target velocity (vl)	S16		✓					
6043_{00}	vl velocity demand	RMP		✓					
6044_{00}	vl velocity actual value	SPE		✓					
6046_{xx}	vl velocity min max amount	SPL		✓					
6048_{xx}	vl velocity acceleration	SAC		✓					
6049_{xx}	vl velocity deceleration	SDC		✓					
6060_{00}	Modes of Operation	ROM	✓	✓	✓	✓	✓	✓	✓
6061_{00}	Modes of Operation Display	AOM	✓	✓	✓	✓	✓	✓	✓
6064_{00}	Position actual value	PST	✓						✓
6065_{00}	Following error window	FEW							✓
6066_{00}	Following error time out	FET							✓

	Description	Roboteq Command	Profile Position	Velocity	Profile Velocity	Torque Profile	Cyclic Sync Velocity	Cyclic Sync Torque	Cyclic Sync Position
6069_{00}	Velocity sensor actual value	VSA			✓		✓		✓
$606B_{00}$	Velocity demand value	VDV			✓		✓		
$606C_{00}$	Velocity actual value	F			✓		✓		✓
6071_{00}	Target torque	TC				✓		✓	
6077_{00}	Torque actual value	TRQ				✓	✓	✓	✓
$607A_{00}$	Target position	POS	✓						✓
$607D_{00}$	Software position limits	PLT	✓						✓
6080_{00}	Max motor speed	MSL	✓		✓	✓	✓	✓	✓
6081_{00}	Profile velocity	PSP	✓						
6083_{00}	Profile acceleration	PAC	✓		✓				
6084_{00}	Profile deceleration	PDC	✓		✓				
6085_{00}	Quick stop deceleration	EDC	✓	✓	✓		✓	✓	✓
6087_{00}	Torque slope	TSL				✓			
$60B0_{00}$	Position Offset	POF							✓
$60B1_{00}$	Velocity Offset	VOF						✓	✓
$60B2_{00}$	Torque Offset	TOF					✓	✓	✓
$60C2_{00}$	Interpolation time period	INT					✓	✓	✓
$60FF_{00}$	Target velocity (pv)	SPC			✓		✓		
6502_{00}	Supported Drive Modes	SDM	✓	✓	✓	✓	✓	✓	✓
$67FE_{00}$	Version Number	VNM	✓	✓	✓	✓	✓	✓	✓

PDS FSA

The standard requires the implementation of a specific finite state machine called FSA. The FSA is designed not only to react to CANOpen commands (Controlword and Statusword), but also to local commands (in this case the use of CW command and SW query). For backward compatibility reasons, the FSA is not active by default. It can be activated by using a special configuration command (^FSA 1, see Figure 4-2).

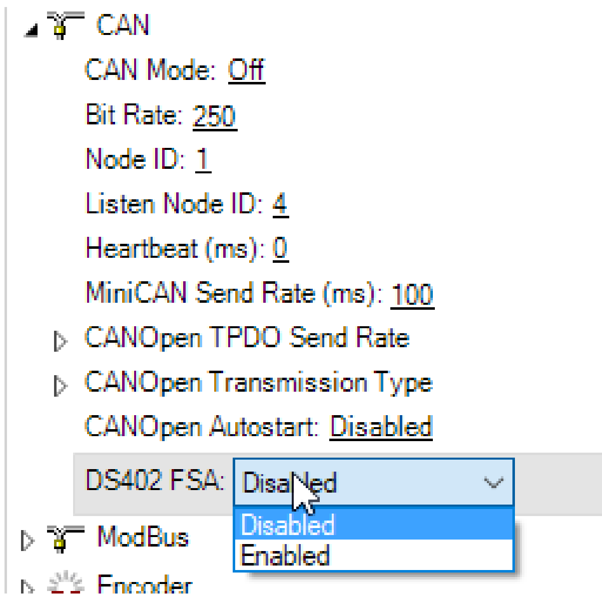


FIGURE 4-2. FSA Configuration

Figure 4-2 describes The states and the transitions of the finite state machine, while Table 4-6 describes the actions and the events of the transitions.

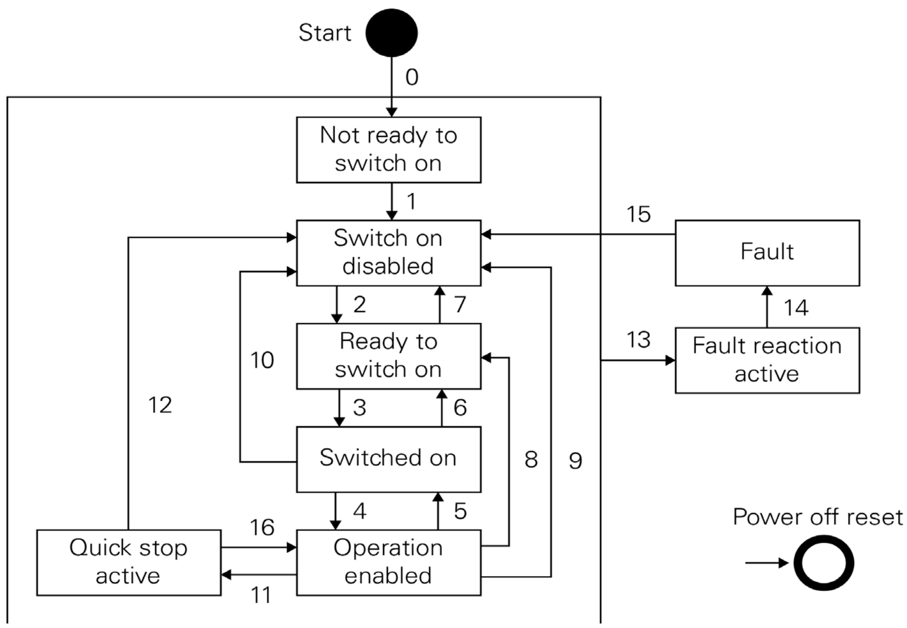


FIGURE 4-3. Power Drive System Finite State Automation

TABLE 4-3. Transition Events and Actions

Transition	Event(s)	Action(s)
0	Automatic Transition after power on or reset application (if ^FSA 1), or when ^FSA is set from 0 to 1.	None
1	Automatic transition	None
2	Shutdown Command	None
3	Switch On Command	None
4	Enable Operation Command	The drive function shall be enabled and all internal set-points cleared.
5	Disable Operation Command	The drive function shall be disabled
6	Shutdown Command	None
7	Quick Stop or Disable Voltage Command	None
8	Shutdown Command	The drive function shall be disabled
9	Disable Voltage Command	The drive function shall be disabled
10	Quick Stop or Disable Voltage Command	None
11	Quick Stop Command or local signal	Quick Stop process is initiated
12	Disable Voltage Command	The drive function shall be disabled
13	Fault Signal or local signal	None
14	Automatic Transition	The drive function shall be disabled
15	Fault Reset Command	The drive function shall be enabled
Note: Roboteq controllers during quick stop state function slow down on quick stop ramp and stay in quick stop active.		

Object Description

0x6040 - Control Word

TABLE 4-4. Control Word

Sub-Index	00	Type	U16	Access	RW	PDO
Value Range	Discrete			Default	Operation mode specific	
RoboCommand	CW					
Description	The received command in order to control the PDS FSA.					

Table 4-4 gives a short description of the object, Table 4-5 the mapping of the respective variable and Table 4-6 the usage of the bits that are independent to operation mode.

TABLE 4-5. Control Word Mapping

15				11	10	9	8	7	6		4	3	2	1	0
R					R	OMS	H	FR	OMS			EO	QS	EV	SO
MSB															LSB
R → Reserved, OMS → Operation mode specific, H → Halt, FR → Fault reset, EO → Enable operation QS → Quick stop, EV → Enable voltage, and SO → Switch on															

TABLE 4-6. Command Coding

Command	Bits of the Control Word					Transition
	Bit 7	Bit 3	Bit 2	Bit 1	Bit 0	
Shutdown	0	X	1	1	0	2,6,8
Switch On	0	0	1	1	1	3
Switch On + Enable Operation	0	1	1	1	1	3+4
Disable Voltage	0	X	X	0	X	7, 9,10,12
Quick Stop	0	X	0	1	X	7,10,11
Disable Operation	0	0	1	1	1	5
Enable Operation	0	1	1	1	1	4,16
Fault Reset	0->1	X	X	X	X	15

Bits 9, 6, 5, and 4 of the ControlWord are operation mode specific. The halt function (bit 8) behavior is operation mode specific. If the bit is 1, the commanded motion shall be interrupted, After releasing the halt function, the commanded motion shall be continued if possible, see Table 4-7.

TABLE 4-7. Halt bit (bit 8)

Bit	Value	Definition
8	0	Positioning shall be executed or continued
	1	Axis shall be stopped. Slow down on quick stop ramp and stay in operation enabled

Profile Position Mode

TABLE 4-8. control word mapping in profile position mode

15 10	9	8	7	6	5	4	3 0
see Table 4-5	Not supported	Halt	see Table 4-5	Abs/ rel	Change Set Immediately	New Set Point	see Table 4-5
MSB							LSB

In Profile Position Mode the operation specific bits are mapped in Table 4-8. With bits 4 and 5, user can define when the command for next Position (0x607A - POS) will be processed. Bit 6 defines whether the command is absolute or relative to the current position.

Table 4-9. Definition of bits 4,5 and 6 in profile position Mode

Bit 5	Bit 4		Definition
0	0->1		Positioning shall be completed (target reached) before the next one gets started.
1	0->1		Next positioning shall be started immediately
Bit	**Value**	**Definition**	
6	0	Target position shall be an absolute value	
	1	Target position shall be a relative value. Positioning moves shall be performed relative to the preceding (internal absolute) target position	

Velocity Mode

TABLE 4-10. control word mapping in Velocity Mode

15 … 9	8	7	6	5	4	3 … 0
see Table 4-5	Halt	see Table 4-5	Reference Ramp	Unlock Ramp	Enable Ramp	see Table 4-5
MSB						LSB

In Velocity Mode the operation specific bits are mapped on Table 4-10. With bits 4, 5 and 6, user can configure the available ramp related options as shown in Table 4-11.

TABLE 4-11. Definition of bits 4, 5, 6 and 8 in Velocity Mode

Bit	Value	Definition
4	0	Motor shall be halted. Slow down on deceleration ramp and stay in operation enabled
	1	Velocity demand value shall accord with ramp output value
5	0	Ramp output value shall be locked to current output value
	1	Ramp output value shall follow ramp input value
6	0	Ramp input value shall be set to zero
	1	Ramp input value shall accord with ramp reference
8	0	No command
	1	Motor shall be stopped
Note: Bit 4 has got higher priority than bit 5.		

Other Modes

Those modes use some bits of the controlword. Table 4-12 shows the structure of the controlword. Table 4-13 defines the values for bit 8 of the controlword.

TABLE 4-12. Controlword for profile torque mode

15 … 9	8	7	6 … 4	3 … 0
See Table 4-5	Halt	See Table 4-5	reserved	See Table 4-5
MSB				LSB

TABLE 4-13. Definition of bit 8

Bit	Value	Definition
8	0	The motion shall be executed or continued
	1	Axis shall be stopped according to the halt option code (605Dh)
Note: At cyclic synchronous modes Halt function is not active		

0x6041 - Status Word

Table 4-14 gives a short description of the object, Table 4-15 the mapping of the respective variable and Table 4-16 the usage of the bits that are independent to operation mode.

TABLE 4-14. Status Word

Sub-Index	00	Type	U16	Access	RO	PDO
Value Range	Discrete			Default	-	
RoboCommand	SW					
Description	The status of the PDS FSA.					

TABLE 4-15. Status Word Mapping

15 14	13 12	11	10	9	8	7	6	5	4	3	2	1	0
NU	OMS	ILA	TR	RM	MS	W	SOD	QS	VE	F	OE	SO	RTSO
MSB													LSB

NU → Not Used, OMS → Operation mode specific, ILA → Internal limit active
TR → Target reached, RM → Remote, W → Warning, SOD → Switch on disabled,
QS → Quick stop, VE → Voltage enabled, F → Fault, OE → Operation Enabled,
SO → Switch on RTSO → Ready to switch on.

If bit 4 (voltage enabled) of the status word is always 1. If bit 5 (quick stop) of the status word is 0, this shall indicate that the PDS is reacting on a quick stop request (quick stop mode is always 6). Bit 7 (warning) is always 0. Bit 9 (remote) of the status word is always 1. If bit 10 (target reached) of the status word is 1, this shall indicate that the PDS has reached the set-point. Bit 10 shall also be set to 1, if the operation mode has been changed. The change of a target value by software shall alter this bit. If halt occurred and the PDS has halted then bit 10 shall be set to 1, too. If the same internal value is commanded then bit 10 shall not alter, if bit 10 is supported (see Table 4-17). If bit 11 (internal limit active) of the statusword is 1, this shall indicate that, an current limit has been reached or the motor command is out of limits.

TABLE 4-16. State Coding

Status Word	**PDS FSA state**
xxxx xxxx x0xx 0000_b	Not ready to switch on
xxxx xxxx x1xx 0000_b	Switch on disabled
xxxx xxxx x01x 0001_b	Ready to switch on
xxxx xxxx x01x 0011_b	Switched on
xxxx xxxx x01x 0111_b	Operation enabled
xxxx xxxx x00x 0111_b	Quick stop active
xxxx xxxx x0xx 1111_b	Fault reaction active
xxxx xxxx x0xx 1000_b	Fault

TABLE 4-17. Definition of bit 10

Bit	**Value**	**Definition**
10	0	Halt (bit 8 in controlword) = 0: Speed or Position Target not reached Halt (bit 8 in controlword) = 1: Axis decelerates
	1	Halt (bit 8 in controlword) = 0: Speed or Position Target reached Halt (bit 8 in controlword) = 1: Velocity of axis is 0
Note: In Roboteq controllers, Halt operation mode is to slow down on slow down ramp and stay in operation enable.		

Profile Position Mode

TABLE 4-18. Status word mapping in profile position mode

15 14	13	12	11	10	9 0
see Table 4-15	Following Error	Set-Point Acknowledge	see Table 4-15	Target Reached	see Table 4-15
MSB					LSB

In Profile Position Mode the operation specific bits are mapped in Table 4-18. With bits 10 and 12 user can acknowledge the status of the controller as shown in Table 4-17 and Table 4-19. Bit 13 is always 0.

TABLE 4-19. Definition of bits 10, 12 and 13 in Profile Position Mode

Bit	Value	Definition
10	0	Hallf (bit 8 in controlword) =0: Target position not reached Half (bit 8 in controlword) = 1: Axis decelerates
	1	Half (bit 0 in controlword) = 0: Target position reached Half (bit 0 in controlword) = 1: Velocity of axis is 0
12	0	Previous set-point already processed, waiting for new set-point
	1	Previous set-point still in process, set-point overwriting shall be accepted
13	0	No following error
	1	Following error

Profile Torque Mode

The profile torque mode uses some bits of the statusword for mode specific purposes. Table 4-20 shows the structure of the status word. Target torque reached is defined Table 4-21.

TABLE 4-20. Statusword for profile torque mode

15 14	13 12	11	10	9 0
see Table 4-15	reserved	see Table 4-15	Target reached	see Table 4-15
MSB				LSB

TABLE 4-21. Definition of bit 8

Bit	Value	Definition
10	0	Halt (bit 8 in controlword) = 0: Target torque not reached Halt (bit 8 in controlword) = 1: Axis decelerates
	1	Halt (bit 8 in controlword) = 0: Target torque reached Halt (bit 8 in controlword) = 1: Velocity of axis is 0

Velocity Mode

The Velocity mode uses some bits of the statusword for mode specific purposes. Table 4-22 shows the structure of the status word.

TABLE 4-22. Statusword for velocity mode

15 – 14	13 – 12	11	10	9 – 0
see Table 4-15	reserved (0)	see Table 4-15	reserved (0)	see Table 4-15
MSB				LSB

Cyclic Synchronous Position Mode

The cyclic synchronous position mode uses three bits of the statusword for mode-specific purposes. Table 4-23 shows the structure of the statusword. Table 4-24 defines the values for bit 10, 12, and 13 of the statusword.

TABLE 4-23. Statusword for profile cyclic synchronous position mode

15 – 14	13	12	11	10	9 – 0
see Table 4-15	Following error	Drive follows the command value	see Table 4-15	reserved	see Table 4-15
MSB					LSB

TABLE 4-24. Definition of bit 10, bit 12, and bit 13

Bit	Value	Definition
10	0	Reserved
	1	Reserved
12	0	Drive does not follow the command value – Target position ignored
	1	Drive follows the command value – Target position used as input to position control loop
13	0	No following error
	1	Following error

Cyclic Synchronous Velocity Mode and Cyclic Synchronous Torque Mode

The Cyclic synchronous velocity and Cyclic synchronous torque mode use some bits of statusword. Table 4-25 shows the structure of the statusword. Table 4-26) defines the values for bit 10, 12, and 13 of the statusword.

TABLE 4-25. Statusword for profile cyclic synchronous velocity mode

15 – 14	13	12	11	10	9 – 0
see Table 4-15	reserved	Drive follows the command value	see Table 4-15	reserved	see Table 4-15
MSB					LSB

TABLE 4-26. Definition of bit 10, bit 12, and bit 13

Bit	Value	Definition
10	0	Reserved
	1	Reserved
12	0	Target velocity or torque ignored
	1	Target velocity or torque used as input to velocity or torque control loop.
13	0	Reserved
	1	Reserved

Profile Velocity Mode

TABLE 4-27. Status Word Mapping in Profile Velocity Mode

15 … 14	13	12	11	10	9 … 0
see Table 4-15	Not Used	Speed	see Table 4-15	Target Reached	see Table 4-15
MSB					LSB

In Profile Velocity Mode the operation specific bits are mapped in Table 4-27. With bits 10 and 12 user can acknowledge the status of the controller as shown in Table 4-26 and Table 4-28. Bit 13 is always 0.

TABLE 4-28. Definition of bit 12 in Profile Velocity Mode

Bit	Value	Definition
12	0	Speed is not equal 0
	1	Speed is equal 0

0x6042 - VL Target Velocity (VL)

Table 4-29 gives a short description of the object.

TABLE 4-29. Target Velocity

Sub-Index	00	Type	S16	Access	RW	PDO
Value Range				Default	0	
RoboCommand	S16					
Description	This object shall indicate the required velocity of the system in RPM. Positive values shall indicate forward direction and negative values shall indicate reverse direction.					

0x6043 - VL Velocity Demand (VL)

Table 4-30 gives a short description of the object.

TABLE 4-30. Velocity Demand

Sub-Index	00	Type	S16	Access	RO	PDO
Value Range				Default	No	
RoboCommand	RMP					
Description	This object shall provide the instantaneous velocity in RPM generated by the ramp function. It is an internal object of the drive device. Positive values shall indicate forward direction and negative values shall indicate reverse direction.					

0x6044 - VL Velocity Actual Value (VL)

Table 4-31 gives a short description of the object.

TABLE 4-31. Velocity Actual Value

Sub-Index	00	Type	S16	Access	RO	PDO	T
Value Range				Default	No		
RoboCommand	RMP						
Description	This object shall provide the velocity in RPM at the motor spindle or load. Depending on the implementation the drive shall provide the appropriate image of the actual velocity derived for example from velocity demand or a sensor signal.						

0x6046 - VL Velocity Min Max Amount (VL)

Table 4-32 gives a short description of the object.

TABLE 4-32. Velocity Min Max Amount

Sub-Index	01	Type	U32	Access	RW	PDO
Value Range				Default	0	
RoboCommand	SPL					
Description	VL velocity min amount.					
Sub-Index	02	Type	U32	Access	RW	PDO
Value Range				Default	1000	
RoboCommand	SPL					
Description	VL velocity max amount.					

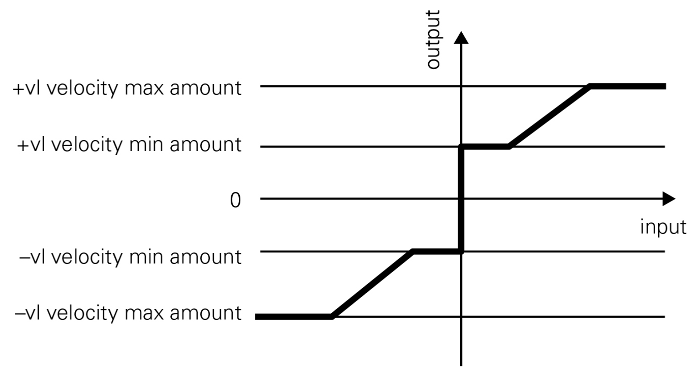


Figure 4-9. Velocity Min Max Amount

This object shall indicate the configured minimum and maximum amount of velocity in RPM. The vl velocity max amount sub-object shall be mapped internally to the vl velocity max positive and vl velocity max negative values. The vl velocity min amount sub-object shall be mapped internally to the vl velocity min positive and vl velocity min negative values. as shown Figure 4-9.

0x6048 - VL Velocity Acceleration (VL)

Table 4-33 gives a short description of the object.

TABLE 4-33. Velocity Acceleration

Sub-Index	01	Type	U32	Access	RW	PDO
Value Range				Default	MAC(20000)	
RoboCommand	SAC					
Description	Delta speed in RPM*10.					
Sub-Index	02	Type	U16	Access	RW	PDO
Value Range				Default	1	
RoboCommand	SAC					
Description	Delta time in seconds.					

0x6049 - VL Velocity Deceleration (VL)

Table 4-34 gives a short description of the object.

TABLE 4-34. Velocity Deceleration

Sub-Index	01	Type	U32	Access	RW	PDO
Value Range				Default	MDEC(20000)	
RoboCommand	SDC					
Description	Delta speed in RPM*10.					
Sub-Index	02	Type	U16	Access	RW	PDO
Value Range				Default	1	
RoboCommand	SDC					
Description	Delta time in seconds.					

0x6060 - Modes of Operation

Table 4-35 gives a short description of the object and Table 4-1 shows the available modes.

TABLE 4-35. Modes of Operation

Sub-Index	00	Type	S8	Access	RW	PDO
Value Range				Default	MMOD(0)	
RoboCommand	ROM					
Description	The requested operation mode.					

0x6061 - Modes of Operation Display

Table 4-36 gives a short description of the object and Table 4-1 shows the available modes.

TABLE 4-36. Modes of Operation Display

Sub-Index	00	Type	S8	Access	RO	PDO
Value Range				Default	MMOD(0)	
RoboCommand	AOM					
Description	The actual operation mode.					

0x6064 - Position Actual Value (PP)

Table 4-37 gives a short description of the object.

TABLE 4-37. Position Actual Value

Sub-Index	00	Type	S32	Access	RO	PDO	T
Value Range				Default	No		
RoboCommand	PST						
Description	This object shall provide the actual value of the position measurement device.						

The position unit are in sensor counts in Closed Loop Count Position mode. In Closed Loop Position Relative mode and in Closed Loop Tracking Position mode the position unit is in range -1000 to 1000 scaled by the minimum and maximum sensor value.

0x6065 - Following Error Window (CSP)

This object indicates the configured range of the tolerated position values symmetrically to the position demand value. If the position actual value is out of the following error window, a following error occurs. If the value of the following error window is FFFF FFFFh, the following control is disabled. The table 4-38 specifies the object description.

TABLE 4-38. Position Demand Value 6065

Sub-Index	00	Type	U32	Access	RW	PDO
Value Range				Default	FFFF FFFFh	
RoboCommand	FEW					
Description	Following Error Window					

0x6066 - Following Error Time Out (CSP)

This object shall indicate the configured time for a following error condition, after that the bit 13 of the statusword shall be set to When the following error occurs, the controller goes to quick stop operation mode. The value is given in ms. Table 4-39 specifies the object description.

TABLE 4-39. Following error time out 6066

Sub-Index	00	Type	U16	Access	RW	PDO
Value Range	0-65535			Default	0	
RoboCommand	FET					
Description	Following error time out (ms)					

0x6069 - Velocity Sensor Actual Value (PV, VL, CSV, CSP)

This object provides the value read from the velocity sensor. The value is given in increments per second. The table 4-40 specifies the object description.

TABLE 4-40. Velocity sensor actual value 6069 (pv,csv,csp)

Sub-Index	00	Type	S32	Access	RO	PDO
Value Range				Default	No	
RoboCommand	VSA					
Description	Velocity sensor actual value inc/s					

0x606B - Velocity Demand Value (PC, CSV)

This object provides the output value of the trajectory generator. Table 4-41 specifies the object description.

TABLE 4-41. Velocity demand value 606B

Sub-Index	00	Type	S32	Access	RO	PDO
Value Range				Default	No	
RoboCommand	VDV					
Description	Velocity demand value (rpm/sec)					

0x606C - Velocity Actual Value (PV)

Table 4-42 gives a short description of the object.

TABLE 4-42. Velocity Actual Value

Sub-Index	00	Optional	CND	Type	S32	Access	RO	PDO	T
Value Range						Default	No		
RoboCommand	SPE								
Description	This object shall provide the actual velocity value, in RPM, derived either from the velocity sensor or the position sensor.								

0x6071 - Target Torque (TQ)

Table 4-43 gives a short description of the object.

TABLE 4-43. Target Torque

Sub-Index	00	Optional	CND	Type	S16	Access	RW	PDO
Value Range						Default	0	
RoboCommand	TC							
Description	This object shall indicate the configured input value, in Nm*100, for the torque controller in profile torque mode.							

0x6077 - Torque Actual Value (TQ)

Table 4-44 gives a short description of the object.

TABLE 4-44. Torque Actual Value

Sub-Index	00	Optional	CND	Type	S16	Access	RO	PDO	T
Value Range						Default	No		
RoboCommand	TRQ								
Description	This object shall provide the actual value of the torque, in Nm*100. It shall correspond to the instantaneous torque in the motor.								

0x607A - Target Position (PP)

Table 4-45 gives a short description of the object.

TABLE 4-45. Target Position

Sub-Index	00	Optional	CND	Type	S32	Access	RW	PDO
Value Range						Default	0	
RoboCommand	POS							
Description	The commanded position that the drive should move to in position profile mode using the current settings of motion control parameters such as velocity, acceleration, deceleration, motion profile type etc. The value of this object shall be interpreted as absolute or relative depending on the abs/rel flag in the ControlWord.							

The position unit are in sensor counts in Closed Loop Count Position mode. In Closed Loop Position Relative mode and in Closed Loop Tracking Position mode the position unit is in range -1000 to 1000 scaled by the minimum and maximum sensor value.

0x607D - Software Position Limit (PP, CSP)

This object indicates the configurated maximal and minimal software position limits. These parameters define the absolute position limits for the position demand value and the position actual value. Every new target has been checked against these limits. Table 4-46 specifies the object description. To disable the software position limits, the min position limit (sub-index 01h) and the max position limit (sub-index 02h) shall be set to 0. The positions limits is given in same position units as the target position.

TABLE 4-46. Software position limit 607D

Sub-Index	01	Type	S32	Access	RW	PDO
Value Range				Default	0	
RoboCommand	PLT					
Description	Software position limit (min value) (counts)					
Sub-Index	02	Type	S32	Access	RW	PDO
Value Range				Default	0	
RoboCommand	PLT					
Description	Software position limit (max value) (counts)					

0x6081 - Profile Velocity (PP)

Table 4-47 gives a short description of the object.

TABLE 4-47. Profile Velocity

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	MVEL(1000)	
RoboCommand	PSP							
Description	This object shall indicate the configured velocity, in RPM, normally attained at the end of the acceleration ramp during a profiled motion and shall be valid for both directions of motion.							

0x6083 - Profile Acceleration (PP)

Table 4-48 gives a short description of the object.

TABLE 4-48. Profile Acceleration

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	MAC(20000)	
RoboCommand	PAC							
Description	This object shall indicate the configured acceleration, in (RPM*10)/ second.							

0x6084 - Profile Deceleration (PP)

Table 4-49 gives a short description of the object.

TABLE 4-49. Profile Deceleration

Sub-Index	00	Optional	Y	Type	U32	Access	RW	PDO
Value Range						Default	MDEC(20000)	
RoboCommand	PAC							
Description	This object shall indicate the configured deceleration, in (RPM*10)/ second.							

0x6080 - Max Motor Speed (PP,PV,TQ,CSV,CST,CSP)

This object shall indicate the configured maximal allowed speed for the motor in either direction. It is used to protect the motor and is taken from the motor data sheet. The value is given in rotations per minute (r/min) or user-defined velocity units. Table 4-50 specifies the object description.

TABLE 4-50. Profile Deceleration

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	(MVEL or MXRPM) 1000	
RoboCommand	MSL							
Description	Max motor speed (rpm/sec)							

0x6085 - Quick Stop Deceleration (PP, VI,PV,CSV,CST,CSP)

Table 4-51 gives a short description of the object.

TABLE 4-51. Profile Deceleration

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	EDEC(20000)	
RoboCommand	EDC							
Description	This object shall indicate the configured fault Deceleration, in (RPM*10)/ second.							

0x6087 - Torque Slope (TSL)

Table 4-52 gives a short description of the object.

TABLE 4-52. Profile Deceleration

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	MAC(20000)*	
RoboCommand	TSL							
Description	This object shall indicate the configured rate of change of torque, in (miliNm*10)/second. *As long as the Torque Constant (TNM) is 1000 miliNm/A.							

0x60B0 - Position Offset (POF)

Table 4-53 gives a short description of the object.

TABLE 4-53. Torque Offset

Sub-Index	00	Optional	Y	Type	S32	Access	RW	PDO
Value Range							Default	0
RoboCommand	POF							
Description	This object shall indicate the configured position offset in sensor counts.							

0x60B1 - Torque Offset (TOF)

Table 4-54 gives a short description of the object.

TABLE 4-54. Torque Offset

Sub-Index	00	Optional	Y	Type	S32	Access	RW	PDO
Value Range							Default	0
RoboCommand	TOF							
Description	This object shall indicate the configured torque offset in miliNm.							

0x60B2 - Velocity Offset (VOF)

Table 4-55 gives a short description of the object.

TABLE 4-55. Torque Offset

Sub-Index	00	Optional	Y	Type	S32	Access	RW	PDO
Value Range							Default	0
RoboCommand	VOF							
Description	This object shall indicate the configured velocity offset in RPM.							

0x60C2 - Interpolation Time Period (CSP, CSV, CST)

This object indicates the configured interpolation cycle time. The interpolation time period (sub-index 01h) value is given in $10^{(\text{interpolation time index})}$ s(seconds). The interpolation time index (sub index 02h) is dimensionless. The table 4-56 specifies the object description. The default value of the interpolation time is 0.001 s (1 millisecond).

TABLE 4-56. Interpolation time period 60C2

Sub-Index	00	Type	U8	Access	RW	PDO
Value Range	0-127			Default	1	
RoboCommand	INT					
Description	Interpolation time period value					
Sub-Index	02	Type	S8	Access	RW	PDO
Value Range	-3, +7			Default	-3	
RoboCommand	INT					
Description	Interpolation time index					

0x60FF - Target Velocity (PV,CSV)

Table 4-57 gives a short description of the object.

TABLE 4-57. Target Velocity

Sub-Index	00	Optional	CND	Type	U32	Access	RW	PDO
Value Range						Default	0	
RoboCommand	SPC							
Description	This object shall indicate the configured target velocity, in RPM, and shall be used as input for the trajectory generator.							

0x6502 - Supported Drive Modes

Table 4-58 gives a short description of the object.

TABLE 4-58. Supported Drive Modes

Sub-Index	00	Optional	REQ	Type	U32	Access	C	PDO	N
Value Range						Default	0x0000038F		
RoboCommand	SDM								
Description	The supported drive modes.								

Roboteq Controllers support:

- Profile Position Mode (PP).
- Velocity Mode (VL).
- Profile Velocity Mode (PV).
- Torque Mode (TQ).
- Cyclic Synchronous Position (CSP)
- Cyclic Synchronous Velocity (CSV)
- Cyclic Synchronous Torque (CST)

0x67FE - Version Number

Table 4-59 gives a short description of the object.

TABLE 4-59. Version Number

Sub-Index	00	Optional	REQ	Type	U32	Access	C	PDO	N
Value Range						Default	0x00040100		
RoboCommand	VNM								
Description	This object shall provide the version number of the CiA 402 profile, which is implemented in the device.								

References

1. CiA® 402 Draft Standard Proposal, v4.1.0, https://www.can-cia.org/can-knowledge/canopen/cia402/