BMS CAN Manual

V2.0, September 3, 2018

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

Table of Contents

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 DS301 documentation.

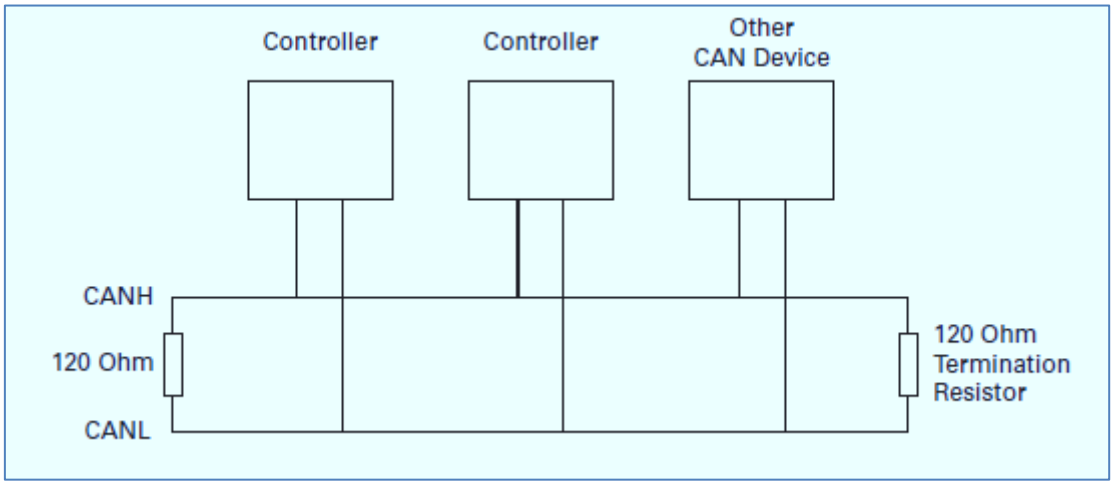
Use and benefits of CANopen

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



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/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 bates supported by the controller are:

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

Heartbeat

A heartbeat message is sent to the bus in millisecond intervals. Heartbeats are useful for detecting the presence or absence of a node on the network. The default value is set to 1000ms.

Autostart

When autostart is enabled, the controller automatically enters the Operational Mode of CANopen. The controller autostart is enabled by default. Disabling the parameter will prevent the controller from starting automatically after the reset occurs. When disabled, the controller can only be enabled when receiving a CANopen management command.

Commands Accessible via CANopen

Almost 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 manual.

All supported commands are mapped in a table, or Object Dictionary that is compliant with the CANopen specification (see Object Dictionary section for details).

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.

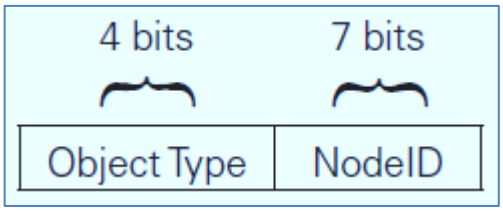
The list of commands accessible with SDO messages can be found in the "Object Dictionary" section.

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 a periodic basis 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.



- TPDO1: 0x180 + Node ID

- TPDO2: 0x280 + Node ID
- TPDO2: 0x380 + Node ID
- TPDO2: 0x480 + Node ID

CANopen allows up to four TPDOs for any node ID. Unless otherwise specified in the product datasheet, TPDO1 to TPDO4 are used to transmit up to 8 user variables which may be loaded with any operating parameters using MicroBasic scripting.

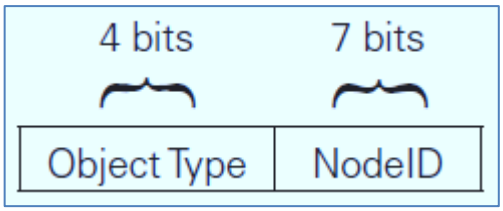
Each of the 4 TPDOs can be configured to be sent at user-defined periodic intervals which can be configured using the PC utility or through CTPS configuration command.

TPDO	Object Index-Sub	Size	Object Mapped
TPDO 1	0x213A-00	U8	BSC – Battery state of charge.
	0x2141-00	U8	BMC – BMS state of charge.
	0x2142-00	U8	BMF – BMS status flags.
	0x2143-00	U8	BMS – BMS operational state.
TPDO 2	0x2106-01	S32	User VAR 1.
	0x2106-02	S32	User VAR 2.
TPDO 3	0x2106-03	S32	User VAR 3.
	0x2106-04	S32	User VAR 4.
TPDO 4	0x2106-05	S32	User VAR 5.
	0x2106-06	S32	User VAR 6.

- U8: unsigned 8-bits.
- S32: signed 32-bits.

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.



- TPDO1: 0x200 + Node ID
- TPDO2: 0x300 + Node ID
- TPDO2: 0x400 + Node ID
- TPDO2: 0x500 + Node ID

Roboteq CANopen implementation supports RPDOs. Unless otherwise specified in the product's datasheet, data received using RPDOs are stored in 4 user variables from where they can be processed using MicroBasic scripting.

RPDO	Object Index-Sub	Size	Object Mapped
RPDO 1	0x2005-09	S32	User VAR 9.
	0x2005-0A	S32	User VAR 10.
RPDO 2	0x2005-0B	S32	User VAR 11.
	0x2005-0C	S32	User VAR 12.
RPDO 3	0x2005-0D	S32	User VAR 13.
	0x2005-0E	S32	User VAR 14.
RPDO 4	0x2005-0F	S32	User VAR 15.
	0x2005-10	S32	User VAR 16.

- S32: signed 32-bits.

Object Dictionary

The CANopen dictionary shown in this section is subject to change. The CANopen EDS file can be downloaded from the roboteq web site.

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.

Runtime Commands

Index	Sub (hex)	Entry Name	Type & Access	Command Name
0x2005	01-vv [(1)]	Set user integer variable.	S32 WO	VAR
0x2008	00	Set all digital out bits.	U8 WO	DS
0x2009	00	Set individual digital out bit.	U8 WO	D1
0x200A	00	Reset individual digital out bit.	U8 WO	D0
0x200C	00	Emergency shutdown.	U8 WO	EX
0x200D	00	Release shutdown.	U8 WO	MG
0X2015	00-bb [(2)]	Set user Boolean variable.	U8 WO	B
0x2017	00	Save configuration to flash.	U8 WO	EES
0x2018	00	Run MicroBasic script.	U8 WO	R
0x201C	00	Switch cell.	U8 WO	CSW
0x201D	00	Switch power.	U8 WO	PSW
0x291E	00	Switch Aux.	U8 WO	ASW

Runtime Queries

Index	Sub (hex)	Entry Name	Type & Access	Command Name
0x2100	01	Read Amps	S16 RO	A
0x2106	01-vv (1)	Read user integer variable.	S32 RO	VAR
0x210D	01-03+cc(3)	Read battery voltage (01), load voltage (02), charge voltage (03) and each cell voltage (04, 05, …)	U16 RO	V
0x210E	00	Read all digital inputs.	U32 RO	D
0x210F	00-01+tt (4)	Read MCU temperature (01) and each transistor temperature (02, 03, …).	S8 RO	T
0x2111	00	Read status flags.	U8 RO	FS
0x2112	00	Read fault flags.	U8 RO	FF
0x2113	00	Read current digital outputs.	U8 RO	DO
0x2115	00-bb (2)	Read user Boolean variable.	U8 RO	B
0x2119	00	Read time.	S32 RO	TM
0x2133	00	Read script checksum.	U32 RO	SCC
0x2134	00	Read if node is alive.	U8 RO	ICL
0x2137	01-04	Read firmware ID (Version, Month, Day, Year).	U16 RO	FIN
0x2139	00	Read capacity runtime.	U16 RO	CRT
0x213A	00	Read battery's state of charge.	U8 RO	BSC
0x213B	00-ss (5)	Read internal switch control.	U8 RO	SWS
0x2140	00	Read the total charge/discharge cycles.	U16 RO	CHD
0x2141	00	Read BMS state of charge.	U8 RO	BMC
0x2142	00	Read BMS status flags.	U8 RO	BMF
0x2143	00	Read BMS operational state.	U8 RO	BMS
0x2144	00	Read battery's state of health.	U8 RO	SOH

(1) vv: maximum number of user integer variables.
(2) bb: maximum number of user Boolean variables.
(3) cc: number of battery cells.
(4) tt: number of internal temperature sensors.
(5) ss: number of internal switches.

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 mechanism. For more details please advise the CANOpen standard.

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

		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:

		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 downloaded. Applicable only if css=4.