Rev A

User Manual

804487-1

CAN Rocker Module

Table of Contents

REVISION LOG 2
Table of Contents 3
1.0 INTRODUCTION 6
1.1 CAN BASICS 6
1.2 DEFINITIONS, ACRONYMS, ABBREVIATIONS 6
2.0 ORDERING INFORMATION 7
2.1 ROCKER HARDWARE 7
2.2 SOFTWARE CONFIGURATION 7
2.2.1 OTTO DEFAULT CONFIGURATION 7
2.2.1.1 DEFAULT VALUES 7
2.2.2 CUSTOMER SPECIFIC 7
2.2.3 CUSTOMER CONFIGURED 8
3.0 MECHANICAL ATTRIBUTES 9
3.1 ISO VIEW 9
3.2 MOUNTING 9
3.3 ROCKER CONNECTION WIRING 10
3.4 TYPICAL CAN NETWORK 11
4.0 CAN ROCKER ATTRIBUTES 11
5.0 ROCKER BASE MODULE 12
5.1 J1939 APPLICATION 12
5.2 CANOPEN APPLICATION 12
6.0 ROCKER EXPANSION MODULE 12
7.0 CAN ROCKER MODULE DATA J1939 AND CANOPEN 12
7.1 SWITCH CONTACT READ 2-BIT DATA 13
7.1.1 ROCKER SWITCH CONTACT STATUS BYTES 13
7.2 LED WRITE 4 BYTE DATA J1939 AND CANOPEN 14
8.0 CAN ROCKER BASE MODULE J1939 INTERFACE 16
8.1 J1939 GENERAL 16
8.2 J1939 DEVICE PROFILE 16
8.2.1 J1939 NAME: 16
8.3 SWITCH STATE PGN(s) 18
8.3.1 SWITCH CONTACTS 18
8.4 LED COLOR AND INTENSITY 19
8.5 CAN MESSAGES 19

8.6 CUSTOMER CONFIGURATION FOR J1939 21
8.6.1 J1939 OBJECT DICTIONARY 21
8.6.2 ROCKER DATA 31
8.6.2.1 SWITCH CONTACT INPUTS 31
8.6.2.2 LED COLOR AND INTENSITY 31
8.6.3 ROCKER MODULE CONFIGURATION MESSAGES 32
8.6.3.1 EXAMPLE SDO (PDU1 PROPRIETARY) MESSAGES 33
8.6.3.2 EXAMPLE INITIATE EXPEDITED SDO DOWNLOAD REQUEST (WRITE) 33
8.6.3.3 EXAMPLE INITIATE EXPEDITED SDO UPLOAD REQUEST (READ) 33
8.6.3.4 EXAMPLE INITIATE SAVE ALL OBJECTS 34
8.6.3.5 ALTERING MAPPING PARAMETERS 34
8.7 ERROR HANDLING 36
8.7.1 SWITCH CONTACT VALUE 36
8.7.2 ERROR OBJECT 36
8.7.3 ERROR MESSAGE 37
8.7.4 EXPANSION MODULE ERRORS 37
9.0 CAN ROCKER BASE MODULE CANOPEN INTERFACE 38
9.1 CANopen GENERAL 38
9.2 NETWORK MANAGEMENT 38
9.3 CANOPEN DS401 DEVICE PROFILE 38
9.4 CUSTOMER CONFIGURATION FOR CANopen 40
9.4.1 CANOPEN OBJECT DICTIONARY 40
9.4.2 ROCKER DATA 50
9.4.2.1 SWITCH CONTACT INPUTS 50
9.4.2.2 LED COLOR AND INTENSITY 50
9.4.3 ROCKER MODULE CONFIGURATION MESSAGES 50
9.4.3.1 EXAMPLE SDO MESSAGES 51
9.4.3.2 EXAMPLE START NODE 51
9.4.3.3 EXAMPLE INITIATE EXPEDITED SDO DOWNLOAD REQUEST (WRITE) 52
9.4.3.4 EXAMPLE INITIATE EXPEDITED SDO UPLOAD REQUEST (READ) 52
9.4.3.5 EXAMPLE INITIATE SAVE ALL OBJECTS 52
9.4.4 ALTERING MAPPING PARAMETERS 53
9.4.5 LSS SERVICES 53
9.4.5.1 EXAMPLE LSS CONFIGURE BIT TIMING 54
9.4.5.2 EXAMPLE LSS SWITCH MODE GLOBAL 55
9.4.5.3 EXAMPLE LSS SET NODE-ID 55
9.4.5.4 EXAMPLE LSS STORE CONFIGURATION 55
9.5 ERROR HANDLING 56
9.5.1 SWITCH CONTACT VALUE 56
9.5.2 ENCODER VALUE 56
9.5.3 ERROR OBJECT 56

9.5.4 ERROR MESSAGE 57
9.5.5 EXPANSION MODULE ERRORS 57
10.0 LED DIMMING 58
11.0 LED FLASH 59
12.0 SLEEP MODE 60
12.1 CAN ROCKER MODULE 60
12.1.1 CANopen 60
12.1.2 J1939 60
12.2 EXPANSION MODULE 60
13.0 CUSTOMER CONFIGURATION SOFTWARE-590342-0500 60
13.1 CAN HARDWARE INSTALLATION 61
13.2 SOFTWARE INSTALLATION 61
13.3 CONFIGURATION SOFTWARE 62
13.3.1 File 62
13.3.2 Help 63
13.3.3 Current baud rate 63
13.4 CONFIGURING A ROCKER 63
13.4.1 Object dictionary CANopen 63
13.4.2 Object dictionary J1939 63
13.4.3 Additional settings 64
13.4.4 LSS settings 64
13.5 CHANGING TRANSMIT MAPPING PARAMETERS 64
13.5.1 CAN ADDRESS 64
13.5.2 CAN DATA 64

1.0 INTRODUCTION

1.1 CAN BASICS

Older vehicles use a simple method to power accessories. The accessory is wired to the battery with a switch in series to turn the accessory on or off. Today's vehicles contain more electronics. Powering many accessories uses many wires, creating large wiring looms. CAN works on a shared communication bus using 4 wires power, ground, CANH, CANL. Switches and accessories connect to processing units (modules). The processing units connect to the CAN bus. When the user flips a switch to turn on an accessory, the module's processing unit sends a message over the CAN bus that every other processing unit (module) is listening to. The processing unit attached to the destination accessory receives the CAN message and acts on it.

Each CAN message has key attributes address and data. A portion of the CAN address allows every processing unit (module) to uniquely identify itself. This is how the accessory processing unit in the previous example knows that the appropriate switch processing unit is trying to communicate a change of state. The CAN message data contains information about whether the switch is on or off.

There are many predefined formats for the CAN messages. These can be thought of as languages spoken on the CAN bus. J1939 or CANopen are popular examples.

1.2 DEFINITIONS, ACRONYMS, ABBREVIATIONS

ADDRESS	Full 11-bit (CANopen) or 29-bit (1939) CAN message header
APPLICATION	Windows executable
BASE	Module connected to customer CAN network
CAN	Controller Area Network
COB-ID	CAN Object ID (CANopen)
CONFIGURE	Alter customer specific objects saved in non-volatile memory
DA	Destination Address 8-bit (J939)
EMCY-object	Emergency-Object, is used to signal of internal devise error
EMI	Electro Magnetic Interference
EXPANSION	Optional module connected to base module via CAN
MODULE	References either the CAN base or CAN expansion assembly
NODE-ID	Portion of ADDRESS used to identify one module, 8-bit.
NMS	Network Management Slave
OD	Object Dictionary
PCB	Printed Circuit Board
PDO	Process Data Object (CANopen)
PDU	Protocol Data Unit, single unit of information (J1939)
PDU1	Message sent to one Destination Address (J1939)
PDU2	Message sent to multiple destinations (J1939)
PROGRAM	Embedded firmware code
RGB	Red, Green and Blue
RPDO	Receive PDO (CANopen)
RTR	Remote Transmission Request
SA	Source Address 8-bit (J1939)
SDO	Service Data Object (CANopen)
SSDO	Server-SDO (CANopen)
SYNC-Object	Synchronization object
TEST	Prompt the operator to verify proper operation of the module under test
TPDO	Transmit PDO (CANopen)

2.0 ORDERING INFORMATION

There are a few key choices to be made regarding ordering an OTTO CAN rocker.

2.1 ROCKER HARDWARE

Consult the OTTO catalog, website or an OTTO customer service representative to specify button legends and other mechanical attributes of the rocker. This information is within the CMK product drawing.

2.2 SOFTWARE CONFIGURATION

The OTTO rocker module may perform as a base module for OTTO rocker expansion modules. See sections 5.0 ROCKER BASE MODULE and 6.0 ROCKER EXPANSION MODULE. A base module needs to be configured to communicate to the specific distribution of expansion modules. Please take this into account when ordering.

There are three implementation options regarding configuration of the rocker module. They are “OTTO default configuration”, “customer specific” and “customer configured”.

2.2.1 OTTO DEFAULT CONFIGURATION

Use this option if you want to evaluate an OTTO rocker module and have not established CAN network communication preferences. This is the “off the shelf” choice for all communication parameters. The only choice is J1939 or CANopen protocol. The module may then be altered using the OTTO customer configuration software. Once communication preferences have been established, future orders could be shipped pre-configured for the customer specific application.

2.2.1.1 DEFAULT VALUES

If an OTTO catalog codable part number is purchased without specifying communication parameters, the module will be shipped with the following summarized CANopen parameters:

- Node ID = 127
- Baud rate = 250Kb
- Rocker module Receive PDO for LEDs
  - COB-ID = 0x27F
  - Data: See 7.2 LED WRITE 4 BYTE DATA J1939 AND CANOPEN
- Rocker module Transmit PDO switch contact data:
  - COB-ID = 0x1FF (change event only)
  - Data: See 7.1 SWITCH CONTACT READ 2-BIT DATA
    - Byte 1: Base module STATUS0
    - Byte 2: Base module STATUS1
    - Byte 3: Base module STATUS2
- Expansion modules are disabled.
  - Error messages will be transmitted if expansion modules are enabled but are not connected.

2.2.2 CUSTOMER SPECIFIC

This is the preferred method of software configuration. OTTO will establish a customer specific part number tailored for every instance of a CAN rocker module used. The rocker module is shipped to the customer pre-configured to drop into a customer CAN network. The module will communicate to other devices on the network immediately. No configuration or device learning is needed. OTTO recommends browsing the details of the “OTTO default configuration”, then request any desired alterations. See the details within sections 8.0 CAN ROCKER BASE MODULE J1939 INTERFACE or 9.0 CAN ROCKER BASE MODULE CANOPEN INTERFACE protocol for details.

2.2.3 CUSTOMER CONFIGURED

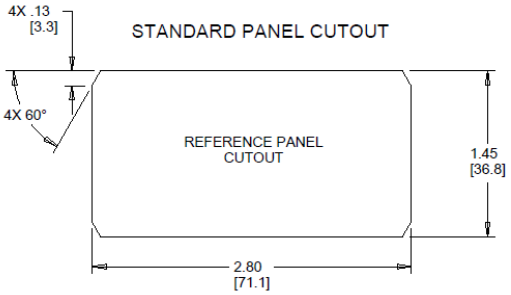
The rocker module messaging is configured by the customer. The CAN addresses, message rate, message contents and other network behavior is established.

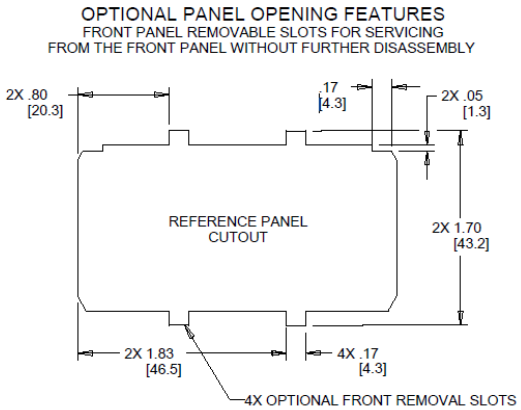
OTTO provides Windows software part number 590342-0500 for the customer to automate the configuration of a rocker. This software requires the use of Kvaser CAN hardware to run. It is also possible for the customer to configure the module with non-Kvaser hardware by manually altering parameters within the modules non-volatile memory. See section 13.0 CUSTOMER CONFIGURATION SOFTWARE for details.

3.0 MECHANICAL ATTRIBUTES

3.1 ISO VIEW

3.2 MOUNTING





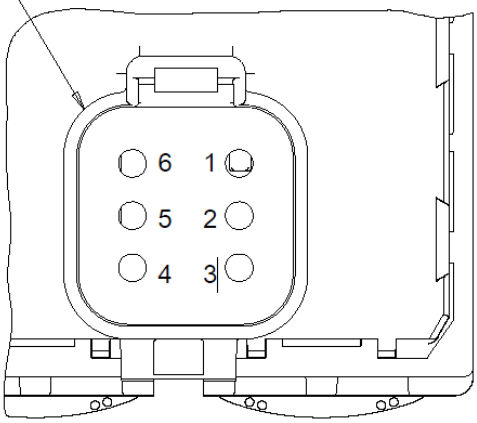
3.3 ROCKER CONNECTION WIRING

MATES WITH DEUTSCH DT06-6S
WITH W6S WEDGE LOCK

ELECTRICAL CONNECTIONS

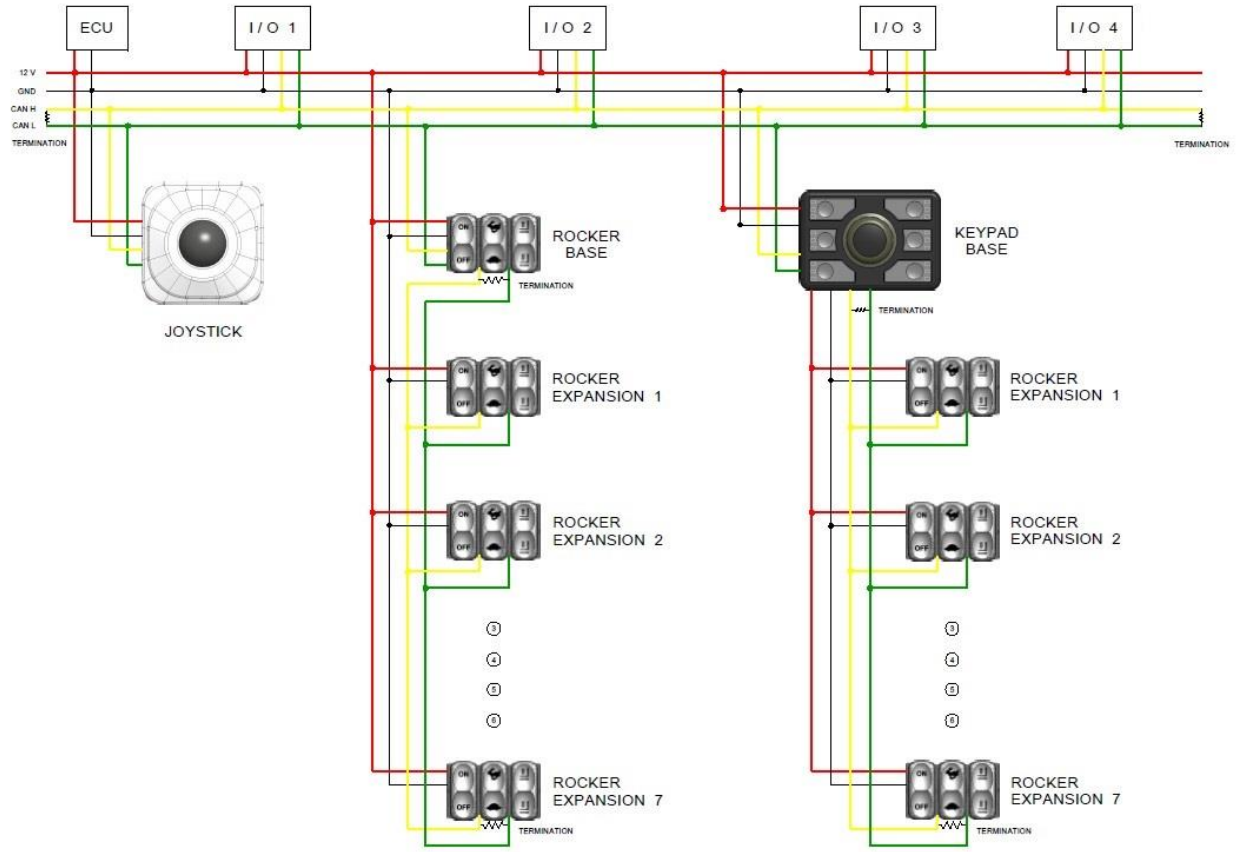
TERMINAL	FUNCTION FOR CONNECTOR
1	POWER (9-32 VDC)
2	CAN HIGH
3	EXPANSION CAN HIGH
4	EXPANSION CAN LOW
5	CAN LOW
6	GROUND

CONNECTOR PIN OUT ORIENTATION



CONNECTOR PIN OUT ORIENTATION SHOWN FOR REFERENCE ONLY. (SWITCH IS NOT MARKED WITH TERMINAL IDENTIFICATION)

3.4 TYPICAL CAN NETWORK



The rocker has an option to place a 120 Ohm CAN bus termination resistor within the module. Since most devices on the network do not have the termination, it is not installed as a default.

4.0 CAN ROCKER ATTRIBUTES

The OTTO CAN rocker module communicates directly with other CAN devices on the customer network. To minimize the number of CAN address used, the rocker module will support up to seven OTTO CAN rocker modules using a secondary CAN expansion bus connection. The customer should install proper CAN bus termination on the base and expansion CAN bus.

The rocker module has three rocker switches. Each rocker switch is configurable for 2 position, 3 position or half action. The positions may be momentary or maintain. Each rocker switch has two full color icon RGB LEDs. The 24-bit color for each RGB LED are set by CAN messages. Other LED features are:

- One 8-bit dimming value for the entire module.
- Synchronized LED flashing.
- Each LED can be individually enabled for flashing.
- A sleep timer will automatically turn off LEDs after four minutes when the module does not observe CAN traffic.

The CAN rocker module is CAN FD tolerant. It will communicate the classical CAN frame format (CAN 2.0A or CAN 2.0B) without corrupting CAN FD messages on the network. It will not transmit CAN FD data.

The CAN module operating parameters and CAN messages are customer configurable. See section 13.0 CUSTOMER CONFIGURATION SOFTWARE.

5.0 ROCKER BASE MODULE

Per ISO11898-1 (2015) section 1, the CAN rocker module will support the Classical CAN frame format. It will tolerate the Flexible Data Rate frame format. The rocker should be configured with the both the nominal and Flexible Data bit rates.

5.1 J1939 APPLICATION

- The module will communicate within J1939 systems. The module is configurable to communicate on either J1939-11 (250K bits/s) or J1939-14 (500K bits/s) networks.
- The CAN module will communicate using proprietary PGN's for switch module status information and setting of the LED indicators and icons.
- Only the 29-bit identifier is used by J1939 application. The application will not interfere with 11-bit CAN identifiers.
- The Source Address will be configurable and stored in non-volatile memory. A Source Address change is only active after a reset. The application will support all 254 available J1939 Source Addresses (SA).
- The J1939 application has a CANopen inspired object dictionary to retain the run time and configurable values. Access to the object dictionary is through PDU1 specific messaging during initial configuration.
- User application configurable objects are stored in non-volatile memory.
- Modules can only be configured using the customer bus (CAN FD tolerant) connection.

5.2 CANOPEN APPLICATION

- All user configurable objects are according to DS401. In addition, some specific configuration and diagnostic data is provided in the manufacturer specific area of the object dictionary. Only the 11-bit identifier is supported by the CANopen interface.
- The CANopen application can be completely remotely configured. The baud rate and node ID are configurable via LSS services. The CAN bus interface is according to ISO11898-2 (High-Speed) and supports baud rates between 50 Kbit/s and 1000 Kbit/s. Once the LSS objects are set (node ID and baud rate), the device can be successfully integrated into a CANopen network.
- User application configurable objects are stored in non-volatile memory.
- Modules can only be configured using the customer bus (CAN FD tolerant) connection.
- The application shall not interfere with 29-bit CAN identifiers

6.0 ROCKER EXPANSION MODULE

- Expansion modules are connected to the base module's expansion CAN bus.
- The CAN rocker base module will support up to 7 CAN rocker expansion modules.
- The CAN expansion modules are CANopen devices. They are not intended to be connected directly to the customer network.
- The Node-ID is used to uniquely distinguish individual CAN expansion modules attached to the base CAN module. Node-ID address ranges are used to distinguish between rocker expansion modules and future option module types. The base module's CANopen Node-ID on the customer CAN bus is unrelated to the Node-ID on the expansion bus.

7.0 CAN ROCKER MODULE DATA J1939 AND CANOPEN

The following tables represent the data format for both the J1939 and CANopen protocols.

The J1939 protocol utilizes a CANopen object dictionary for configuration of the rocker module (CAN messages and data). Once the rocker is configured for the customer application, the module shall communicate per the appropriate protocol with the desired customer CAN messages.

Configuration messages are anticipated to be used only during module initial configuration before installation into a customer network. The eight data bytes of J1939 PDU1 proprietary configuration message match the eight data bytes of the corresponding CANopen SDO configuration message. Only the CAN addressing is different.

The switch contacts and LEDs are independent within the module firmware. The customer CAN controller is responsible for monitoring switch contacts and setting module LEDs as desired. All LEDs within the module will default to the off state.

7.1 SWITCH CONTACT READ 2-BIT DATA

Switch CAN address(es) and message contents are configurable. The customer may choose to use the default mapping, have OTTO preconfigure the module, or configure the CAN messages after delivery.

The values of switch contact data can be mapped into any one of the eight configurable Process Data Objects (CANopen) or PGNs (J1939). Each of the messages is configurable with up to eight bytes of data. The quantity of CAN messages, each CAN address and the data contents are determined by the customer application. Mapping is accomplished by placing one of the object 0x4000 sub-indexes (SWITCH_CONTACTS) into any one of the eight CAN message mapping objects 0x1A00 through 0x1A07 (MAPPING OBJECTS). Any data change within a mapped message shall trigger an immediate transmission of the message. Associated with each CAN message are communication object setting objects 0x1800 through 0x1807. Setting 0x1800 sub-index 0x05 shall establish periodic transmission of the CAN message.

7.1.1 ROCKER SWITCH CONTACT STATUS BYTES

The OTTO CAN rocker is connected to the customer CAN network. Additionally, the OTTO CAN rocker may be connected to a base CAN rocker as an optional expansion module. Connecting OTTO CAN rockers to the base rocker expansion CAN bus will allow the rocker base module to pack switch contact data together. This will increase CAN bus utilization. Additionally, the LEDs on base module and its expansion modules will have synchronized flashing.

The OTTO CAN rocker has three actuators, with configurable momentary and maintain positions. Each switch position has two tactile switches wired in parallel for redundancy. The firmware supports two tactile switches wired independently. Independent tactile switch monitoring may be a future option. The top and bottom actuator positions have full color RGB LEDs. See 7.2 LED WRITE 4 BYTE DATA.

It is possible that the switch contact data will be reported as “Not available or error” during a rocker switch transition. There are six tact switches per rocker actuator, two in parallel per position. If the handle is held between two valid positions, the reported value will be “Not available or error”. The customer application should indicate an error if the “Not available or error” position is maintained. The rocker center position is supported by actual switches. The center position is not inferred by lack of either full travel position.

Reported values for rocker switch contact data are two bits.

00	Middle position
01	Down position
10	Up position
11	Not available or error

Table 7-1 ROCKER SWITCH CONTACT DATA

The values of each switch contact, formatted per the previous table, are mapped into three status bytes. This table corresponds directly to object 0x4000 (SWITCH_CONTACTS). Distinguishing between the rocker

primary and optional redundant switch contacts is bit 6 of each data byte. Since the current hardware has the primary and redundant tact switches wired in parallel, both STATUS0 and STATUS1 should be identical.

Status Byte	Bit	Value
STATUS0	1,0	Left Rocker primary switches
STATUS0	3,2	Center Rocker primary switches
STATUS0	5,4	Right Rocker primary switches
STATUS0	6	0
STATUS0	7	0
STATUS1	1,0	Left Rocker redundant switches
STATUS1	3,2	Center Rocker redundant switches
STATUS1	5,4	Right Rocker redundant switches
STATUS1	6	1
STATUS1	7	0
STATUS2	7..0	0

Table 7-2 ROCKER SWITCH CONTACT STATUS BYTES

7.2 LED WRITE 4 BYTE DATA J1939 AND CANOPEN

LED colors may be modified by the customer application using a base module's RPDO (CANopen) or PDU1 specific (J1939) messages. LEDs are either individually addressable or all LEDs in the module may be updated with a single message. Both the CANopen and J1939 protocols utilize an object dictionary. An object dictionary is basically a configurable look up table containing all the information necessary to communicate on the customer CAN network. There are three objects associated with writing to LEDs. Object index 0x1400 ("Receive PDO Communication Parameter") contains the CAN identifier (address) desired for the customer application. Object 0x1600 contains message data mapping information. Object 0x1600 ("Receive PDO Mapping Parameter") contains a pointer to object 0x2000 and generally should not be changed. Object 0x2000 is the ultimate destination for LED write data.

To write to LEDs, the customer application will send a CAN message to the address specified by object 0x1400 with 8 byte data formatted per Table 7-3 ROCKER MODULE LED WRITE DATA OBJECT 0x2000. The center position indicator is a future option and currently not available. Expansion modules must be enabled with object 0x3000 to allow the CAN module to write LED data. Note that a rocker module may also serve as a base module on the customer CAN network, if properly configured.

Byte	**bit**	**Coding**
0	0-1	00 = Center indicator
		01 = Bottom icon
		10 = Top icon
		11 = Invalid
	2-5	0000 = Left switch
		0001 = Center switch
		0010 = Right switch
		0011 through 1110 = Invalid
		1111 = All LEDs in module (bits 1,0 are ignored)
	6	1= Flash enabled
	7	1 = Ignore color setting
1	0-7	Red intensity (0-255)
2	0-7	Green intensity (0-255)

3	0-7	Blue intensity (0-255)
4	0-2	000 = Base module
		001 = Expansion module 1
		010 = Expansion module 2
		011 = Expansion module 3
		100 = Expansion module 4
		101 = Expansion module 5
		110 = Expansion module 6
		111 = Expansion module 7
5	0-7	0
6	0-7	0
7	0-7	0

Table 7-3 ROCKER MODULE LED WRITE DATA OBJECT 0x2000

8.0 CAN ROCKER BASE MODULE J1939 INTERFACE

8.1 J1939 GENERAL

Supported J1939 features are:

- Eight TX PDU format messages for CAN module switch positions (event and/or time triggered). These messages have dynamic mapping. Dynamic mapping will allow CAN messages to be configured for a specific customer. The TX PDUs will be set before installation in the customer network.
- One RX PDU1 proprietary format messages with dynamic mapping are used for LED write data.
- One TX PDU2 (broadcast) proprietary format message for error handling (event triggered).
- One PDU1 (destination specific) proprietary message for module configuration (request and response). This PDU1 is intended for module configuration prior to installation into the customer CAN network.
- Network management with address claiming functionality.
- The J1939-15 interface runs with 250Kbits/s. The J1939-14 interface runs with 500Kbits/sec.
- The CAN modules are single address capable. The Source Address of each CAN module must be configured before installation into a network. Arbitrary addressing is not supported. Each OTTO rocker contains identical firmware and the NAME function.
- The CAN module is CAN-FD tolerant. No messages are received or transmitted using CAN-FD. The CAN module shall not disrupt a CAN-FD network.
- Proprietary PDU1 configuration messages (SDO's) shall use CANopen abort transfer protocol on mal formed requests within the reply.

8.2 J1939 DEVICE PROFILE

8.2.1 J1939 NAME:

The NAME is sent when the module powers up with address claim PGN60928. The NAME fields are defined as follows:

Byte	Bit	Name	Value	Description / Note
8	8	Arbitrary Address Capable	0b	Single address capable
	7-5	Industry Group	000b	Global
	4-1	Vehicle System Instance	0000b	Default value
7	8-2	Vehicle System	1111111b	Default value: not available
	1	Reserved	0b	
6	8-1	Function	00h	Default value
5	8-4	Function Instance	00000b	Default value
	3-1	ECU Instance	000b	Default value
4	8-1	Manufacturer Code (most sign. 8 Bits)	00011001b	207
3	8-6	Manufacturer Code (least sign. 3 Bits)	111b	
3	5-1	Identity Number (most sig. 5 Bits)	DDDDDb	Manufacturing Code (Firmware value): Y – Year (binary encoded) D – Day (binary encoded)
2	8-1	Identity Number (second Byte)	DDDDDYYYb	
1	8-1	Identity Number (least sig. Byte)	YYYYYYYYb	

Table 8-1 J1939 NAME FIELDS

The 29-bit CAN-ID ADDRESS is sliced into static fields. The first three bits contain the message priority. A value of 000 is the highest priority. The next 18 bits within the CAN-ID ADDRESS are referred to as the Parameter Group Number. The PGN contains the reserved bit, the Data Page bit, the 8-bit PDU Format, and the 8-bit PDU Specific/Group Extension field. The last 8 bits are the Source Address of the transmitting device which is claimed during network initialization. Each Source Address must be unique for every device on the network.

The factory default J1939 Source Address for an OTTO CAN module is 100 (0x64). The PDU specific field /PGN are configurable.

J1939 FRAME FORMAT	SOF	PRIORITY:3	PRIORITY:2	PRIORITY:1	RESERVED	DATA PAGE	PDU FORMAT:8	PDU FORMAT:7	PDU FORMAT:6	PDU FORMAT:5	PDU FORMAT:4	PDU FORMAT:3	SRR	IDE	PDU FORMAT:2	PDU FORMAT:1	PDU SPEC/GE:8	PDU SPEC/GE:7	PDU SPEC/GE:6	PDU SPEC/GE:5	PDU SPEC/GE:4	PDU SPEC/GE:3	PDU SPEC/GE:2	PDU SPEC/GE:1	SOURCE ADDR:8	SOURCE ADDR:7	SOURCE ADDR:6	SOURCE ADDR:5	SOURCE ADDR:4	SOURCE ADDR:3	SOURCE ADDR:2	SOURCE ADDR:1	RTR
J1939 BIT POSITION	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
CAN 29 BIT POS.		28	27	26	25	24	23	22	21	20	19	18			17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
SWITCH STATE PGN 65291 @0x64	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1	1	0	0	0	0	1	0	1	1	0	1	1	0	0	1	0	0	1
PDU1 PGN 61184 (LED @ 0x64)	1				0	0	1	1	1	0	1	1	0	0	1	1	0	1	1	0	0	1	0	0	X	X	X	X	X	X	X	X	1
PDU1 PGN1280 (config. to module 1 @ 0x64)	1				0	0	0	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1	0	0	X	X	X	X	X	X	X	X	1
PDU1 PGN1536 (config. reply module 1 @ 0x64)	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	X	X	X	X	X	X	X	X	0	1	1	0	0	1	0	0	1

Table 8-2 J1939 CAN ADDRESS EXAMPLES

8.3 SWITCH STATE PGN(s)

Switch contact data is transmitted using a proprietary PDU2 message frame. The 64-bit (8 byte) data contained within the frame is the switch contact data. The message shall be sent immediately on change of state of any contact. The message can additionally be transmitted at user defined intervals. Each switch state frame consists of a 29-bit address and up to 8-byte data field. Unused data bytes are transmitted as 0x00.

8.3.1 SWITCH CONTACTS

Reported values for the switch contacts are within Table 7-1 ROCKER SWITCH CONTACT DATA and Table 7-2 ROCKER SWITCH CONTACT STATUS BYTES. Details regarding the messages can be found in the following section 7.0 CAN ROCKER MODULE DATA J1939 AND CA.

Default Parameter Group Number	65291: 0xFF0B (Configurable)
Default priority	6 (Configurable)
Data Length	2 Bytes (Configurable)
Data Page	0 (Configurable)
PDU Format	255 (Configurable)
PDU Specific	0x0B, (Configurable)
Byte 1	(Remappable)
1,0	STATUS0 Left Rocker primary switches
3,2	STATUS0 Center Rocker primary switches
5,4	STATUS0 Right Rocker primary switches
7,6	00
Byte 2	(Remappable)
1,0	STATUS1 Left Rocker redundant switches
3,2	STATUS1 Center Rocker redundant switches
5,4	STATUS1 Right Rocker redundant switches
7,6	01
Byte 3	00 (Remappable or not transmitted)
Byte 4	00 (Remappable or not transmitted)
Byte 5	00 (Remappable or not transmitted)
Byte 6	00 (Remappable or not transmitted)
Byte 7	00 (Remappable or not transmitted)
Byte 8	00 (Remappable or not transmitted)

Table 8-3 CAN ROCKER MODULE SWITCH CONTACT PGN

8.4 LED COLOR AND INTENSITY

LED color and intensities are set using PDU1 proprietary messages for J1939.

Default Parameter Group Number	61184: 0xEF00 (Configurable)
Default priority	(Masked)
Data Length	8 Bytes (Configurable)
Data Page	0 (Configurable)
PDU Format	239 (Configurable)
PDU Specific	0x64, (Module DA) (Configurable)
Byte 1-8	See 7.2 LED WRITE 4 BYTE DATA J1939 AND CANOPEN

8.5 CAN MESSAGES

Object 1200:01 is fixed in firmware and cannot be changed. This PDU1 CAN message is used for SDO customer initial configuration. A configuration tool's Source Address (bits 7-0) will be temporarily retained for the reply SDO. PDU Specific messages will be evaluated by firmware to match the Source Address of the module. PGN 1280 is used for SDO download and PGN 1536 is used for the SDO reply.

Received CAN data message Source Address bits will not be filtered by the base module. The base module shall accept CAN data messages from any Source Address.

Care must be exercised to configure every module with a unique CAN address. Duplication will result in erratic behavior on the network.

Objects 0x1014, 0x1200, 0x1400 and 0x1800-0x1807 have a 32-bit value specify the CAN 2.0B address. Per CANopen DS301 Description of RPDO COB-ID:

Bit 31(valid):

0: PDO exists/is valid

1: PDO does not exist/is not valid

Bit 30 (Reserved-do not care):

Bit 29(frame):

0: 11-bit ID (CAN 2.0A)

1: 29-bit ID (CAN 2.0B)

8.6 CUSTOMER CONFIGURATION FOR J1939

This section is relevant only for customers desiring to configure the rocker module.

8.6.1 J1939 OBJECT DICTIONARY

The J1939 protocol has the following CANopen inspired object dictionary for configuration. Configuration messages are detailed in section 8.6.3 ROCKER MODULE CONFIGURATION MESSAGES

The CAN module J1939 object dictionary has the following object entries:

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
0x1001	0x 00	Error register[1]	ro	y		Unsigned8	0x00
0x1003		pre-defined error field				Unsigned32	
	0x00	number of entries	rw	n	y	Unsigned8	0x00
	0x01	standard error field	ro	n	y	Unsigned32	0x00
	0x02	standard error field	ro	n	y	Unsigned32	0x00
	0x03	standard error field	ro	n	y	Unsigned32	0x00
	0x04	standard error field	ro	n	y	Unsigned32	0x00
	0x05	standard error field	ro	n	y	Unsigned32	0x00
	0x06	standard error field	ro	n	y	Unsigned32	0x00
	0x07	standard error field	ro	n	y	Unsigned32	0x00
	0x08	standard error field	ro	n	y	Unsigned32	0x00
	0x09	standard error field	ro	n	y	Unsigned32	0x00
	0x0A	standard error field	ro	n	y	Unsigned32	0x00
0x1010		store objects				Unsigned32	
	0x00	largest Sidx supported	ro	n		Unsigned8	0x01
	0x01	save all objects	rw	n		Unsigned32	0x00
0x1011		restore objects				Unsigned32	
	0x00	largest Sidx supported	ro	n		Unsigned8	0x01
	0x01	restore all default objects	rw	n		Unsigned32	0x00
0x1014	0x00	CAN Address[1] Emergency	rw	n	y	Unsigned32	0x18FF0000+ (SA)
0x1018		identity object	ro			Identity	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x00	number of entries	ro	n		Unsigned8	0x03
	0x01	Vendor-ID	ro	n		Unsigned32	0x01B3
	0x02	Manufacturer product code	ro	n		Unsigned32	0x0384CA60 (590342-08)
	0x03	Manufacturer revision number	ro	n		Unsigned32	0x00000001 (Increments with each release)
0x1200		server SDO object[1]				SDO Param	
	0x00	number of entries[1]	ro	n		Unsigned8	0x02
	0x01	CAN Address[1] client->server[1]	ro	n		Unsigned32	0x1805(DA)(SA)
	0x02	CAN Address[1] server->client[1]	ro	n		Unsigned32	0x1806(DA)(SA)
0x1400		Receive PDO 1 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x18EF(DA)(SA)
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
0x1600		receive PDO 1 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x01
	0x01	1st object	rw	n	y	Unsigned32	0x20000140
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1800		transmit PDO 1 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x01	CAN (PGN)[1]	rw	n	y	Unsigned32	0x18FF0B(SA)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1801		transmit PDO 2 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x18FF0C(SA)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1802		transmit PDO 3 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF0D(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1803		transmit PDO 4 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF0E(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x64
0x1804		transmit PDO 5 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05

[1] This value has different behavior for CANopen and J1939.

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF0F(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1805		transmit PDO 6 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF10(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1806		transmit PDO 7 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF11(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1807		transmit PDO 8 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	CAN Address[1]	rw	n	y	Unsigned32	0x98FF12(SA) (disabled)
	0x02	Transmission type	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1A00		transmit PDO 1 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x02
	0x01	1st object	rw	n	y	Unsigned32	0x40000108

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x02	2nd object	rw	n	y	Unsigned32	0x40000208
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A01		transmit PDO 2 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A02		transmit PDO 3 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A03		transmit PDO 4 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A04		transmit PDO 5 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A05		transmit PDO 6 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A06		transmit PDO 7 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A07		transmit PDO 8 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
0x2000		Write 32-bit RGB LED data (WRITE_LED_DATA)				Unsigned32	See Table 7-3 ROCKER MODULE LED WRITE DATA OBJECT 0x2000
	0x00	number of entries	ro	n		Unsigned8	0x01
	0x01	LED data	ro	n		Unsigned64	0x0000000000000000
0x2200		LED Dimming level (LED_DIM)				Unsigned8	
	0x00	number of entries	ro	n		Unsigned8	0x2
	0x01	Enable Dimming	rw	y		Unsigned8	0x0 = OFF (range 0 to 1)
	0x02	Dimming scale	rw	y		Unsigned8	0xFF (range 0 to 255)
0x2300		LED Flash (LED_FLASH)				Unsigned8	
	0x00	number of entries	ro	n		Unsigned8	0x3
	0x02	ON interval	rw	y		Unsigned16	0x64 (range 0 to 65535) in 10mS increments
	0x03	OFF interval	rw	y		Unsigned16	0x64 (range 0 to 65535) in 10mS increments
0x2F00		CAN FD Baudrate				Unsigned32	0x000F4240 Valid values: 50000, 100000, 125000, 250000, 500000, 800000, 1000000, 2000000, 4000000,5000000
0x3000		Enable expansion module (ENABLE_EXP)				Unsigned8	0x0 = disabled 0x01 = enabled
	0x00	number of entries	ro	n		Unsigned8	0x07
	0x01	Rocker expansion module 1	rw	n	y	Unsigned8	0x00
	0x02	Rocker expansion module 2	rw	n	y	Unsigned8	0x00
	0x03	Rocker expansion module 3	rw	n	y	Unsigned8	0x00
	0x04	Rocker expansion module 4	rw	n	y	Unsigned8	0x00
	0x05	Rocker expansion module 5	rw	n	y	Unsigned8	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x06	Rocker expansion module 6	rw	n	y	Unsigned8	0x00
	0x07	Rocker expansion module 7	rw	n	y	Unsigned8	0x00
0x3001		Expansion setting (ENABLE_EXP)				Unsigned8	0x0 = Base module, no expansion modules 0x01 = Base module, has expansion modules.
0x3002		Protocol				Unsigned8	0x0 = CANopen 0x01 = J1939

0x4000		read digital input 8-bit (SWITCH_CONTACTS)				Unsigned8	0 (range 0 to 255) See Table 7-1 ROCKER SWITCH CONTACT DATA See **Error! Reference source not found.**
	0x00	number of entries	ro	n		Unsigned8	0x18
	0x01	Base module switch STATUS0	ro	y		Unsigned8	
	0x02	Base module switch STATUS1	ro	y		Unsigned8	
	0x03	Base module switch STATUS2	ro	y		Unsigned8	
	0x04	Expansion module 1 switch STATUS0	ro	y		Unsigned8	
	0x05	Expansion module 1 switch STATUS1	ro	y		Unsigned8	
	0x06	Expansion module 1 switch STATUS2	ro	y		Unsigned8	
	0x07	Expansion module 2 switch STATUS0	ro	y		Unsigned8	
	0x08	Expansion module 2 switch STATUS1	ro	y		Unsigned8	
	0x09	Expansion module 2 switch STATUS2	ro	y		Unsigned8	
	0x0A	Expansion module 3 switch STATUS0	ro	y		Unsigned8	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x0B	Expansion module 3 switch STATUS1	ro	y		Unsigned8	
	0x0C	Expansion module 3 switch STATUS2	ro	y		Unsigned8	
	0x0D	Expansion module 4 switch STATUS0	ro	y		Unsigned8	
	0x0E	Expansion module 4 switch STATUS1	ro	y		Unsigned8	
	0x0F	Expansion module 4 switch STATUS2	ro	y		Unsigned8	
	0x10	Expansion module 5 switch STATUS0	ro	y		Unsigned8	
	0x11	Expansion module 5 switch STATUS1	ro	y		Unsigned8	
	0x12	Expansion module 5 switch STATUS2	ro	y		Unsigned8	
	0x13	Expansion module 6 switch STATUS0	ro	y		Unsigned8	
	0x14	Expansion module 6 switch STATUS1	ro	y		Unsigned8	
	0x15	Expansion module 6 switch STATUS2	ro	y		Unsigned8	
	0x16	Expansion module 7 switch STATUS0	ro	y		Unsigned8	
	0x17	Expansion module 7 switch STATUS1	ro	y		Unsigned8	
	0x18	Expansion module 7 switch STATUS2	ro	y		Unsigned8	

Table 8-4 J1939 OBJECT DICTIONARY

8.6.2 ROCKER DATA

8.6.2.1 SWITCH CONTACT INPUTS

The actual status of the digital inputs is reported with object 0x4000. The values are defined by Table 7-1 ROCKER SWITCH CONTACT DATA. Distinguishing between the primary and redundant rocker switch contacts is bit 6 of each data byte. Object 0x4000 can be customer mapped to any one of eight TPDO's.

8.6.2.2 LED COLOR AND INTENSITY

LED color and intensities are set using mappable PDO's. They may also be set using SDO's.

8.6.3 ROCKER MODULE CONFIGURATION MESSAGES

The configuration objects and additional data is accessible using PDU1 type proprietary messages (PDU format = 239). Configuration is achieved through a dictionary of object indexes and sub-indexes with up to four bytes of data. The PDU1 proprietary messages are used to access a CANopen inspired object dictionary like CANopen expedited SDO's. The addresses are per J1939 PDU1 proprietary messages. The eight data bytes of the PUD1 proprietary message match the CANopen expedited SDO format. The following table has been duplicated from the CANopen Table 9-4 Expedited SDO summary definition.

Data Byte	Bits	Value	Description	Definition
0	7-5	001 010 010 011	Initiate download request Initiate upload request Initiate upload response Initiate download response	Client Command Specifier
0	4	0		
0	3-2		Number of data bytes within bytes 4 to 7 that do NOT contain data	Valid only if "e" & "s" = 1
0	1	0 1	Normal transfer Expedited transfer	"e": transfer type
0	0	0 1	Data size not included Data size included	"s": size indicator
1	7-0		IDX: Multiplexor LSB	
2	7-0		IDX: Multiplexor MSB	
3	7-0		SIDX: Multiplexor	
4-7	7-0		Data	Little Endian

Table 8-5 Expedited SDO summary definition

For J1939, only expedited configuration requests shall be supported:

	BYTE 0 VALUE
Write 1 byte	0x2F
Write 2 bytes	0x2B
Write 3 bytes	0x27
Write 4 bytes	0x23

Table 8-6 Expedited SDO byte 0

8.6.3.1 EXAMPLE SDO (PDU1 PROPRIETARY) MESSAGES

The customer application may alter one of the object dictionaries values within the module. The identifier field is specified by object 0x1200.

8.6.3.2 EXAMPLE INITIATE EXPEDITED SDO DOWNLOAD REQUEST (WRITE)

The customer application may alter one of the object dictionaries values within the CAN module. Within the following examples:
(DA) corresponds to the Destination Address of the rocker module.
(SA) corresponds to the Source Address of the configuration tool.

IDENTIFIER	DATA (in Hex)	
See 0x1200:01 Default: 0x1805(DA)(SA)	2F 00 22 02 80 00 00 00	Write 0x2200:02 (LED_DIM) with value 128

The CAN module shall respond with the following:

IDENTIFIER	DATA (in Hex)	
See 0x1200:02 Default: 0x1806(SA)(DA)	60 00 22 02 00 00 00 00	

8.6.3.3 EXAMPLE INITIATE EXPEDITED SDO UPLOAD REQUEST (READ)

The customer application may read one of the object dictionaries values within the CAN module.

IDENTIFIER	DATA (in Hex)	
See 0x1200:01 Default: 0x1805(DA)(SA)	40 00 1A 01 00 00 00 00	Read 1A00:01 (TPDO1 1st object mapping)

The CAN module shall respond with the following:

IDENTIFIER	DATA (in Hex)	
See 0x1200:02 Default: 0x1806(SA)(DA)	43 00 1A 01 08 01 00 40	(default 0x4000:01 (8b))

8.6.3.4 EXAMPLE INITIATE SAVE ALL OBJECTS

The customer application may request that the CAN module save all/some of the altered Object dictionary values. The new values shall be stored in non-volatile memory for the next restart of the CAN rocker.

IDENTIFIER	DATA (in Hex)	
See 0x1200:01	23 10 10 01 73 61 76 65	Send “save” to 1010:01

Note: depending upon which objects have been altered, please select the appropriate sub-index.: Please allow time for response with before resetting the CAN module.

The CAN module shall respond with the following

IDENTIFIER	DATA (in Hex)	
See 0x1200:02	60 01 1A 02 08 04 00 60	

SDO’s shall use CANopen abort transfer protocol on mal formed requests within the reply.

8.6.3.5 ALTERING MAPPING PARAMETERS

Note: Per CANopen specification, mapping TPDO (0x1A00 through 0x1A07) and RPDO (0x1600) objects must follow a process to change their values. This sequence of steps is automatically handled by the customer configuration application. This process temporarily disables the TPDO while it is being changed.

To manually alter a mapping parameter, follow the recommended steps (shown using TPDO1 default values):

- Recommended: read and retain the communication object 0x1800:01 for later reference if you do not want to subsequently change the CAN address.
  Address is per 0x1200:01 (Default = 0x1805(DA)(SA))..
  The hex data is 40 00 18 01 00 00 00 00.
  The hex reply should be 43 00 18 01 ## ## ## ## (Note first byte, ## ## is the address).
- Set TPDO1 comm object 0x1800:01 to 0x80000000 to disable TPDO.
  Address is per 0x1200:01 (Default = 0x1805(DA)(SA)).
  The hex data is 23 00 18 01 00 00 00 80.
  The hex reply should be 60 00 18 01 00 00 00 00 (Note first byte).
- Clear TPDO1 mapping sub index 0.
  Address is per 0x1200:01.
  The hex data is 2F 00 1A 00 00 00 00 00.
  The hex reply should be 60 00 1A 00 00 00 00 00.
- Set the new mapping parameter.
  This example writes the default for 0x1A00:01 (0x4000:01(8b)).
  Address is per 0x1200:01.
  The hex data is 23 00 1A 01 08 01 00 40.
  The hex reply should be 60 00 1A 01 00 00 00 00.
- Continue with other changes to the same mapping object.
- Restore the mapping number of entries.
  Address is per 0x1200:01.
  The hex data is 2F 00 1A 00 03 00 00 00 (default=3 mapped objects).
  The hex reply should be 60 00 1A 00 00 00 00 00.
- Restore the TDO1 communication object 0x1800:01 with the CAN address.
  Address is per 0x1200:01.

The hex data is 23 00 18 01 ## ## ## ##.
The hex reply should be 60 00 18 01 00 00 00 00

8.7 ERROR HANDLING

The base module is responsible for handling (forward) expansion module errors

8.7.1 SWITCH CONTACT VALUE

When an invalid state of a switch is detected, the reported value is indicated by the value “not available or error”. See Table 7-1 ROCKER SWITCH CONTACT DATA.

If a communication error occurs with an expansion module, the switch contact data value for the expansion module shall be replaced with the value “not available or error”.

8.7.2 ERROR OBJECT

In the Error Register (0x1001), only bits 0, 4 and 7 are used for indicating error states. The bits are set when an emergency is transmitted. If a bit is set once, it shall never be reset while the J1939 CAN module application is in operational state. After power-on, it shall be reset.

The pre-defined error list (0x1003) has a size of 10 entries. The contents are stored in a non-volatile memory (EEPROM). The value of an entry of the pre-defined error list is a 32-bit value and composed in the following way:

MSB	LSB
0x0000	Emergency Error Code

Device situation / Error condition	Set bits in object 0x1001	Emergency Error Code	Manufacturer specific error field (5 Ascii chars)
CAN Overrun (Messages lost)	0 and 4	0x8110	'C','N','O','V','R'
Life Guard or Heartbeat error	0 and 4	0x8130	'L','T','H','B','T'
PDO not processed due to length error	0 and 4	0x8210	'P','D','O','P','E'
PDO length exceeded	0 and 4	0x8220	'P','D','O','L','E'
E2P error	0 and 7	0xFF00	'E','2','P','E','R'
IO Initialization error	0 and 7	0xFF01	'I','I','N','I','T'
Encoder error	0 and 7	0xFF0E	‘E’,’N’,’C’,’E’,’R’
Expansion module 1 error	0 and 7	0xFF11	'E','X','P',’A’,'1'
Expansion module 2 error	0 and 7	0xFF12	'E','X','P',’A’,'2'
Expansion module 3 error	0 and 7	0xFF13	'E','X','P',’A’,'3'
Expansion module 4 error	0 and 7	0xFF14	''E','X','P',’A’,'4'
Expansion module 5 error	0 and 7	0xFF15	'E','X','P',’A’,'5'
Expansion module 6 error	0 and 7	0xFF16	'E','X','P',’A’,'6'
Expansion module 7 error	0 and 7	0xFF17	'E','X','P',’A’,'7'

Table 8-7 J1939 error messages

8.7.3 ERROR MESSAGE

When and invalid value of a module is detected by the firmware, a special error message shall be sent. The error message shall be transmitted using PDU2 proprietary messages (PDU format = 255):

The error message has implemented as follows:

Transmission Repetition Rate: Event triggered

Data Length: 8 Bytes

CAN Address : nn = object 0x1014

An emergency message (EMCY) shall be transmitted when any of the device situations listed in Table 8-7 J1939 error messages occur.

The data bytes mirror the CANopen DS301 definition. The data is defined as follows:

Byte	0	1	2	3	4	5	6	7
	Emergency error code See Table 8-7 J1939 error messages Table 9-7 CANopen error messages		Error register (Object 0x1001)	Manufacturer specific error field				

8.7.4 EXPANSION MODULE ERRORS

The CAN base module firmware may encounter a timeout waiting for a periodic transmission from an expansion module. See section 7.1 SWITCH CONTACT READ 2-BIT DATA. Should a timeout occur, error object 0x1003 shall be created. An error message shall be transmitted. The firmware shall continually retry the request. Until a successful transfer, no more error objects or error messages shall be created.

9.0 CAN ROCKER BASE MODULE CANOPEN INTERFACE

9.1 CANopen GENERAL

Supported CANopen features are:

- 1 Server SDOs expedited and non-expedited
- 8 TX PDOs, dynamic mapping
- 1 RX PDO, dynamic mapping
- Event- and time triggered TX PDOs
- Emergency message (producer)
- Heartbeat producer
- NMT-Slave
- LSS-Slave

Not supported CANopen features

- Client SDOs
- Server SDOs with block transfer

9.2 NETWORK MANAGEMENT

Some objects in the communication, manufacturer and device profile areas can be saved and restored within a non-volatile memory by means of object 0x1010 and 0x1011. Note: the strings “save” and “load” must be used within the data field respectively.

The PDOs support the transmission type255 only.

TX PDOs support event timed transmission.

The CAN module requires the start remote node command to enter the operational state. This can be accomplished by sending COB ID 0x00 with 2 bytes of data. The first byte is 0x01 (command specifier = start). The second data byte is the module address. See DS301, “NMT Services”.

9.3 CANOPEN DS401 DEVICE PROFILE

CAN-ID/ COB-ID			
FROM	TO	Communication Objects	Comment
0x0h	--	NMT Service	From NMT Master
0x01		Reserved	
0x80	--	SYNC Message	From Sync Producer
0x81	0xFF	Emergency Message	From nodes 1 to 127
0x100	--	Time Stamp Message	From time stamp producer
0x101	0x180	Reserved	
0x181	0x1FF	1st Transmit PDO	From nodes 1 to 127

0x201	0x27F	1st Receive PDO	For nodes 1 to 127
0x281	0x2FF	2nd Transmit PDO	From nodes 1 to 127
0x301	0x37F	2nd Receive PDO	Not used
0x381	0x3FF	3rd Transmit PDO	From nodes 1 to 127
0x401	0x47F	3rd Receive PDO	Not used
0x481	0x4FF	4th Transmit PDO	From nodes 1 to 127
0x501	0x57F	4th Receive PDO	Not used
0x581	0x5FF	Transmit SDO	From nodes 1 to 127
0x601	0x67F	Receive SDO	For nodes 1 to 127
0x691		Sleep mode[2]	
0x6E0		Reserved	
0x701	0x77F	NMT Error Control	From nodes 1 to 127
0x780	0x7FF	Reserved	

Table 9-1 Default CANopen COB-ID's

The default address of the OTTO CAN module is 127 (0x7F)

The default address shall use the following pre-defined COB IDs:

PDO	COB-ID	Default objects
TPDO1	0x1FF	IDX:0x4000 SIDX:0x01 CAN rocker – STATUS0 IDX:0x4000 SIDX:0x02 CAN rocker – STATUS1
TPDO2	0x80000000	
TPDO3	0x80000000	
TPDO4	0x80000000	
TPDO5 (Not standard)	0x80000000	
TPDO6 (Not standard)	0x80000000	

[2] Per CIA 320 specification

TPDO7 (Not standard)	0x80000000	
TPDO8 (Not standard)	0x80000000	

Table 9-2 Default rocker TPDOs and RPDOs

9.4 CUSTOMER CONFIGURATION FOR CANopen

This section is relevant only for customers desiring to configure the rocker module.

9.4.1 CANOPEN OBJECT DICTIONARY

The CAN module object dictionary has the following object entries:

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
0x1000	0x00	device type[1]	ro	n		Unsigned32	0x00010191
0x1001	0x00	error register	ro	y		Unsigned8	0x00
0x1003		pre-defined error field				Unsigned32	
	0x00	number of entries	rw	n	y	Unsigned8	0x00
	0x01	standard error field	ro	n	y	Unsigned32	0x00
	0x02	standard error field	ro	n	y	Unsigned32	0x00
	0x03	standard error field	ro	n	y	Unsigned32	0x00
	0x04	standard error field	ro	n	y	Unsigned32	0x00
	0x05	standard error field	ro	n	y	Unsigned32	0x00
	0x06	standard error field	ro	n	y	Unsigned32	0x00
	0x07	standard error field	ro	n	y	Unsigned32	0x00
	0x08	standard error field	ro	n	y	Unsigned32	0x00
	0x09	standard error field	ro	n	y	Unsigned32	0x00
	0x0A	standard error field	ro	n	y	Unsigned32	0x00
0x1008	0x00	manufacturer device name[1]	ro	n		Vis-String	CANopen_Rocker
0x1010		store objects				Unsigned32	
	0x00	largest Sidx supported	ro	n		Unsigned8	0x01
	0x01	save all objects	rw	n		Unsigned32	0x00
0x1011		restore objects				Unsigned32	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x00	largest Sidx supported	ro	n		Unsigned8	0x01
	0x01	restore all default objects	rw	n		Unsigned32	0x00
0x1014	0x00	COB-ID Emergency[1]	rw	n	y	Unsigned32	0x80+Node-ID
0x1017	0x00	producer heartbeat time[1]	rw	n	y	Unsigned16	0x0000
0x1018		identity object	ro			Identity	
	0x00	number of entries	ro	n		Unsigned8	0x03
	0x01	Vendor-ID	ro	n		Unsigned32	0x01B3
	0x02	Manufacturer product code	ro	N		Unsigned32	0x0384CA60 (590342-08)
	0x03	Manufacturer revision number	ro	n		Unsigned32	0x00000001 (Increments with each release)
0x1029	0x00	Error Behaviour[1]	ro	n		Unsigned8	0x02
	0x01	Communication Error[1]	ro	n		Unsigned8	1
	0x02	Device Profile or manufacturer specific Error[1]	ro	n		Unsigned8	1
0x1200		server SDO object[1]				SDO Param	
	0x00	number of entries[1]	ro	n		Unsigned8	0x02
	0x01	COB-ID client->server[1]	ro	n		Unsigned32	600 + Node-ID
	0x02	COB-ID server->client[1]	ro	n		Unsigned32	580 + Node-ID
0x1400		Receive PDO 1 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	200h + Node-ID
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
0x1600		receive PDO 1 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x01
	0x01	1st object	rw	n	y	Unsigned32	0x20000140

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1800		transmit PDO 1 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	180h + Node-ID
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1801		transmit PDO 2 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	280h + Node-ID
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFE
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1802		transmit PDO 3 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	380h + Node-ID
	0x02	transmission type[1]	rw	n	y	Unsigned8	0x01
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1803		transmit PDO 4 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	480h + Node-ID
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1804		transmit PDO 5 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	0x000001DF
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1805		transmit PDO 6 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	0x000002DF
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1806		transmit PDO 7 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	0x000003DF
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1807		transmit PDO 8 comm. object				PDO CommPar	
	0x00	number of entries	ro	n		Unsigned8	0x05
	0x01	COB-ID used by PDO[1]	rw	n	y	Unsigned32	0x000004DF
	0x02	transmission type[1]	rw	n	y	Unsigned8	0xFF
	0x03	Inhibit time (in 100uS)	rw	n	y	Unsigned16	0xC8
	0x05	event timer	rw	n	y	Unsigned16	0x00
0x1A00		transmit PDO 1 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x02
	0x01	1st object	rw	n	y	Unsigned32	0x40000108
	0x02	2nd object	rw	n	y	Unsigned32	0x40000208
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A01		transmit PDO 2 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A02		transmit PDO 3 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A03		transmit PDO 4 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A04		transmit PDO 5 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A05		transmit PDO 6 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A06		transmit PDO 7 mapping object				PDO Mapping	
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00
0x1A07		transmit PDO 8 mapping object				PDO Mapping	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x00	number of entries	ro	n	y	Unsigned8	0x00
	0x01	1st object	rw	n	y	Unsigned32	0x00
	0x02	2nd object	rw	n	y	Unsigned32	0x00
	0x03	3rd object	rw	n	y	Unsigned32	0x00
	0x04	4th object	rw	n	y	Unsigned32	0x00
	0x05	5th object	rw	n	y	Unsigned32	0x00
	0x06	6th object	rw	n	y	Unsigned32	0x00
	0x07	7th object	rw	n	y	Unsigned32	0x00
	0x08	8th object	rw	n	y	Unsigned32	0x00

0x2000		Write 32-bit RGB LED data (WRITE_LED_DATA)				Unsigned32	See Table 7-3 ROCKER MODULE LED WRITE DATA OBJECT 0x2000
	0x00	number of entries	ro	n		Unsigned8	0x01
	0x01	Base module	rw	n		Unsigned32	0x0000000000000000
0x2200		LED Dimming level (LED_DIM)				Unsigned8	
	0x00	number of entries	ro	n		Unsigned8	0x2
	0x01	Enable Dimming	rw	y		Unsigned8	0x0 = OFF (range 0 to 1)
	0x02	Dimming scale	rw	y		Unsigned8	0xFF (range 0 to 255)
0x2300		LED Flash (LED_FLASH)				Unsigned8	
	0x00	number of entries	rw	n		Unsigned8	0x3
	0x02	ON interval	rw	y		Unsigned16	0x64 (range 0 to 65535) in 10mS increments
	0x03	OFF interval	rw	y		Unsigned16	0x64 (range 0 to 65535) in 10mS increments

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
0x2F00		CAN FD Baudrate				Unsigned32	0x000F4240 Valid values: 50000, 100000, 125000, 250000, 500000, 800000, 1000000, 2000000, 4000000,5000000
0x3000		Enable expansion module (ENABLE_EXP)				Unsigned8	0x0 = disabled 0x01 = enabled
	0x00	number of entries	ro	n		Unsigned8	0x07
	0x01	Rocker expansion module 1	rw	n	y	Unsigned8	0x00
	0x02	Rocker expansion module 2	rw	n	y	Unsigned8	0x00
	0x03	Rocker expansion module 3	rw	n	y	Unsigned8	0x00
	0x04	Rocker expansion module 4	rw	n	y	Unsigned8	0x00
	0x05	Rocker expansion module 5	rw	n	y	Unsigned8	0x00
	0x06	Rocker expansion module 6	rw	n	y	Unsigned8	0x00
	0x07	Rocker expansion module 7	rw	n	y	Unsigned8	0x00
0x3001		Expansion setting (ENABLE_EXP)				Unsigned8	0x0 = Base module, no expansion modules 0x01 = Base module, has expansion modules.
0x3002		Protocol				Unsigned8	0x0 = CANopen 0x01 = J1939

4000		read digital input 8-bit (SWITCH_CONTACTS)				Unsigned8	0 (range 0 to 255) See Table 7-1 ROCKER SWITCH CONTACT DATA. See Table 7-2 ROCKER SWITCH CONTACT STATUS BYTES
	0x00	number of entries	ro	n		Unsigned8	0x18
	0x01	Base module switch STATUS0	ro	y		Unsigned8	
	0x02	Base module switch STATUS1	ro	y		Unsigned8	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x03	Base module switch STATUS2	ro	y		Unsigned8	
	0x04	Expansion module 1 switch STATUS0	ro	y		Unsigned8	
	0x05	Expansion module 1 switch STATUS1	ro	y		Unsigned8	
	0x06	Expansion module 1 switch STATUS2	ro	y		Unsigned8	
	0x07	Expansion module 2 switch STATUS0	ro	y		Unsigned8	
	0x08	Expansion module 2 switch STATUS1	ro	y		Unsigned8	
	0x09	Expansion module 2 switch STATUS2	ro	y		Unsigned8	
	0x0A	Expansion module 3 switch STATUS0	ro	y		Unsigned8	
	0x0B	Expansion module 3 switch STATUS1	ro	y		Unsigned8	
	0x0C	Expansion module 3 switch STATUS2	ro	y		Unsigned8	
	0x0D	Expansion module 4 switch STATUS0	ro	y		Unsigned8	
	0x0E	Expansion module 4 switch STATUS1	ro	y		Unsigned8	
	0x0F	Expansion module 4 switch STATUS2	ro	y		Unsigned8	
	0x10	Expansion module 5 switch STATUS0	ro	y		Unsigned8	
	0x11	Expansion module 5 switch STATUS1	ro	y		Unsigned8	
	0x12	Expansion module 5 switch STATUS2	ro	y		Unsigned8	
	0x13	Expansion module 6 switch STATUS0	ro	y		Unsigned8	

Idx	Sidx	Name (Reference)	Attrib	Map-able	EE data	DataType	Default Value, Range
	0x14	Expansion module 6 switch STATUS1	ro	y		Unsigned8	
	0x15	Expansion module 6 switch STATUS2	ro	y		Unsigned8	
	0x16	Expansion module 7 switch STATUS0	ro	y		Unsigned8	
	0x17	Expansion module 7 switch STATUS1	ro	y		Unsigned8	
	0x18	Expansion module 7 switch STATUS2	ro	y		Unsigned8	

Table 9-3 CANOPEN OBJECT DICTIONARY

9.4.2 ROCKER DATA

9.4.2.1 SWITCH CONTACT INPUTS

The actual status of the digital inputs is reported with object 0x4000. The values are defined by Table 7-1 ROCKER SWITCH CONTACT DATA. Distinguishing between the primary and redundant rocker switch contacts is bit 6 of each data byte. Object 0x4000 can be customer mapped to any one of eight TPDO's.

9.4.2.2 LED COLOR AND INTENSITY

LED color and intensities are set using mappable PDO's. They may also be set using SDO's.

9.4.3 ROCKER MODULE CONFIGURATION MESSAGES

The configuration objects and additional data is accessible using SDO upload and download requests. Configuration is achieved through a dictionary of object indexes and sub-indexes with up to four bytes of data.

There are a few different SDO types defined for CANopen. Typically, expedited SDO's can be used to re-configure the object dictionary of the module. Expedited SDO's contain the command specifier and object data within the 8-byte CAN message. A receive SDO COB-ID is defined from 0x601 to 0x67F for nodes 1-127. A transmit SDO COB-ID is defined from 0x581 to 0x5FF for nodes 1-127. The 8-byte data payload within an expedited SDO is defined as follows:

Data Byte	Bits	Value	Description	Definition
0	7-5	001 010 010 011	Initiate download request Initiate upload request Initiate upload response Initiate download response	Client Command Specifier
0	4	0		

0	3-2		Number of data bytes within bytes 4 to 7 that do NOT contain data	Valid only if “e” & “s” = 1
0	1	0 1	Normal transfer Expedited transfer	“e”: transfer type
0	0	0 1	Data size not included Data size included	“s”: size indicator
1	7-0		IDX: Multiplexor LSB	
2	7-0		IDX: Multiplexor MSB	
3	7-0		SIDX: Multiplexor	
4-7	7-0		Data	Little Endian

Table 9-4 Expedited SDO summary definition

	BYTE 0 VALUE
Write 1 byte1	0x2F
Write 2 bytes	0x2B
Write 3 bytes	0x27
Write 4 bytes	0x23

Table 9-5 Expedited SDO byte 0

9.4.3.1 EXAMPLE SDO MESSAGES

The customer application may alter one of the object dictionaries values within the module. The identifier field is specified by object 0x1200.

9.4.3.2 EXAMPLE START NODE

The CAN module shall power up, then transmit a boot up event. The first data byte corresponds to the NMT state of the CAN module.

COB-ID	DATA (in Hex)	
0x700 + Node ID	00	NMT state = Boot up event

If the heartbeat is turned on, the CAN module shall send a heartbeat message with the data byte corresponding to the NMT state of pre-operational. This message shall be sent continually at a rate determined by the heartbeat setup.

COB-ID	DATA (in Hex)	
0x700 + Node ID	7F	NMT state = pre-oper

The bus master initiates a start node command.

COB-ID	DATA (in Hex)	
0x00	01 0A	Start Node 0x0A

0x00	01 00	Start all nodes

If the heartbeat is turned on, the CAN module shall send a heartbeat message with the data byte corresponding to the NMT state of operational. This message shall be sent continually at a rate determined by the heartbeat setup.

COB-ID	DATA (in Hex)	
0x700 + Node ID	05	NMT state = operational

9.4.3.3 EXAMPLE INITIATE EXPEDITED SDO DOWNLOAD REQUEST (WRITE)

The NMT master may alter one of the object dictionaries values within the CAN module. COB-ID 0x60A corresponds to the receive SDO of the CAN module at node address 0x0A.

COB-ID	DATA (in Hex)	
See 0x1200:01 0x600 + Node ID	2F 00 22 02 80 00 00 00	Write 0x2200:02 (LED_DIM) with value 128

The CAN module shall respond with the following:

COB-ID	DATA (in Hex)	
See 0x1200:02 0x580 + Node ID	60 00 22 02 00 00 00 00	

9.4.3.4 EXAMPLE INITIATE EXPEDITED SDO UPLOAD REQUEST (READ)

The NMT master may read one of the object dictionaries values within the CAN module. COB-ID 0x60A corresponds to the receive SDO of the CAN module at node address 0x0A.

COB-ID	DATA (in Hex)	
0x600 + Node ID	40 00 1A 01 00 00 00 00	Read 1A00:01 (TPDO1 1st object mapping)

The CAN module shall respond with the following:

COB-ID	DATA (in Hex)	
0x580 + Node ID	43 00 1A 01 08 01 00 40	(default 0x4000:01 (8b))

9.4.3.5 EXAMPLE INITIATE SAVE ALL OBJECTS

The NMT master may request that the CAN module save all/some of the altered object dictionary values. The new values shall be stored in non-volatile memory for the next restart of the CAN module. COB-ID 0x60A corresponds to the receive SDO of the CAN module at node address 0x0A.

COB-ID	DATA (in Hex)	
0x600 + Node ID	23 10 10 01 73 61 76 65	Send “save” to 1010:01

Note: depending upon which objects have been altered, please select the appropriate sub-index. Please allow time for response with COB-ID 0x58A before resetting the CAN module.

The CAN module shall respond with the following:

COB-ID	DATA (in Hex)	
0x580 + Node ID	60 10 10 01 00 00 00 00	

9.4.4 ALTERING MAPPING PARAMETERS

Note: Per CANopen specification, mapping TPDO (0x1A00 through 0x1A07) and RPDO (0x1600) objects must follow a process to change their values. This sequence of steps is automatically handled by the customer configuration application. This process temporarily disables the TPDO while it is being changed.

To manually alter a mapping parameter, follow the recommended steps (shown using TPDO1 default values):

- Recommended: read and retain the communication object 0x1800:01 for later reference if you do not want to subsequently change the CAN address.
  Address is per 0x1200:01 (Default = 0x600 + Node-ID).
  The hex data is 40 00 18 01 00 00 00 00.
  The hex reply should be 43 00 18 01 ## ## 00 00 (Note first byte, ## ## is the COB-ID).
- Set TPDO1 comm object 0x1800:01 to 0x80000000 to disable TPDO.
  Address is per 0x1200:01 (Default = 0x600 + Node-ID).
  The hex data is 23 00 18 01 00 00 00 80.
  The hex reply should be 60 00 18 01 00 00 00 00 (Note first byte).
- Clear TPDO1 mapping sub index 0.
  Address is per 0x1200:01.
  The hex data is 2F 00 1A 00 00 00 00 00.
  The hex reply should be 60 00 1A 00 00 00 00 00.
- Set the new mapping parameter.
  This example writes the default for 0x1A00:01 (0x4000:01(8b))
  Address is per 0x1200:01.
  The hex data is 23 00 1A 01 08 01 00 40.
  The hex reply should be 60 00 1A 01 00 00 00 00.
- Continue with other changes to the same mapping object.
- Restore the mapping number of entries.
  Address is per 0x1200:01.
  The hex data is 2F 00 1A 00 03 00 00 00 (default=3 mapped objects).
  The hex reply should be 60 00 1A 00 00 00 00 00.
- Restore the TDO1 communication object 0x1800:01 with the CAN address.
  Address is per 0x1200:01.
  The hex data is 23 00 18 01 ## ## 00 00.
  The hex reply should be 60 00 18 01 00 00 00 00.

9.4.5 LSS SERVICES

The module supports CAN FD hardware that needs to be initialized. The standard baud rate shall be established by LSS communication.

The LSS protocol is executed between 1 LSS master and 1 LSS slave. Objects such as the CAN baud rate and CANopen node-ID are configured with the LSS. The LSS master uses COB-ID 0x7E5 to send commands

to the LSS slave. The LSS slave may use 0x7E4 to return values to the master. The first data byte of the command is referred to as the Command Specifier.

The LSS slave (CAN module) supports the following services as specified in [3]:

- Switch Mode Global
- Switch Mode Selective
- Configure Node ID
- Configure Bit Timing Objects
- Activate Bit Timing Objects
- Store Configured Objects

9.4.5.1 EXAMPLE LSS CONFIGURE BIT TIMING

The LSS master initiates a node change with the COB-ID of 0x7E5. The first byte Command Specifier is 0x13. The second byte corresponds a table selector (0x00 = the standard CIA bit timing table). The third byte corresponds to a table index.

COB-ID	DATA (in Hex)	
0x7E5	13 00 03 00 00 00 00 00	Set bit rate to 250kBaud
0x7E5	13 00 04 00 00 00 00 00	Set bit rate to 125kBaud
0x7E5	13 00 FF 00 00 00 00 00	Set bit rate auto-baud

Valid indices to the CANopen baud rate table are:

Index	Baud rate (kBaud)
0	1000
1	800
2	500
3	250
4	125
5	Reserved (100)
6	50
7	20
8	10

Table 9-6 LSS baud rate definitions

The LSS slave CAN module shall also respond with Command Specifier 0x13. The second and third bytes correspond to error codes. An error code “ee” = 0x00 corresponds to successful completion.

COB-ID	DATA (in Hex)	
0x7E4	ee ## 00 00 00 00 00 00	Set bit rate

9.4.5.2 EXAMPLE LSS SWITCH MODE GLOBAL

The LSS master initiates a switch mode global with COB-ID 0x7E5. The first byte Command Specifier is 0x04. The second byte corresponds to the operating mode.

COB-ID	DATA (in Hex)	
0x7E5	04 00 00 00 00 00 00 00	Operate Mode
0x7E5	04 01 00 00 00 00 00 00	Config Mode

9.4.5.3 EXAMPLE LSS SET NODE-ID

Valid node-IDs are 1 to 127.

The LSS master initiates a node change with the COB-ID of 0x7E5. The first byte Command Specifier is 0x11. The second byte corresponds to the Node-id.

COB-ID	DATA (in Hex)	
0x7E5	11 0A 00 00 00 00 00 00	Set Node ID to 0x0A
0x7E5	11 0B 00 00 00 00 00 00	Set Node ID to 0x0B

The LSS slave CAN module shall also respond with Command Specifier 0x11. The second and third bytes correspond to error codes. An error code “ee” = 0x00 corresponds to successful completion.

COB-ID	DATA (in Hex)	
0x7E4	ee ## 00 00 00 00 00 00	Set Node ID to 0x0A

9.4.5.4 EXAMPLE LSS STORE CONFIGURATION

Any change of the baud rate or node-ID without saving the changes causes the CAN module to come-up with the last successfully saved values again after power-on.

The LSS master initiates a node change with the COB-ID of 0x7E5. The first byte Command Specifier is 0x17.

COB-ID	DATA (in Hex)	
0x7E5	17 00 00 00 00 00 00 00	Store Configuration

The LSS slave CAN module shall also respond with Command Specifier 0x17. The second and third bytes correspond to error codes. An error code “ee” = 0x00 corresponds to successful completion.

COB-ID	DATA (in Hex)	
0x7E4	ee ## 00 00 00 00 00 00	

9.5 ERROR HANDLING

The base module is responsible for handling (forward) expansion module errors

9.5.1 SWITCH CONTACT VALUE

When an invalid state of a switch is detected, the reported value is indicated by the value “not available or error”. See Table 7-1 ROCKER SWITCH CONTACT DATA.

If a communication error occurs with an expansion module, the switch contact data value for the expansion module shall be replaced with the value “not available or error”.

9.5.2 ENCODER VALUE

When and invalid quadrature transition occurs, the firmware shall transmit an error message. The position value shall remain at the last known valid value.

9.5.3 ERROR OBJECT

In the Error Register (0x1001), only bits 0, 4 and 7 are used for indicating error states. The bits are set when an emergency is transmitted. If a bit is set once, it shall never be reset while the CANopen CAN module application is in operational state. After power-on or upon a Reset Communication or Reset Application command it shall be reset.

The pre-defined error list (0x1003) has a size of 10 entries. The contents are stored in a non-volatile memory (EEPROM). The value of an entry of the pre-defined error list is a 32-bit value and composed in the following way:

MSB	**LSB**
0x0000	Emergency Error Code

Device situation / Error condition	**Set bits in object 0x1001**	**Emergency Error Code**	**Manufacturer specific error field** (coded fields)
CAN Overrun (Messages lost)	0 and 4	0x8110	'C','N','O','V','R'
Life Guard or Heartbeat error	0 and 4	0x8130	'L','T','H','B','T'
PDO not processed due to length error	0 and 4	0x8210	'P','D','O','P','E'
PDO length exceeded	0 and 4	0x8220	'P','D','O','L','E'
E2P error	0 and 7	0xFF00	'E','2','P','E','R'
IO Initialization error	0 and 7	0xFF01	'I','I','N','I','T'
Encoder error	0 and 7	0xFF0E	'E','N','C','E','R'
Expansion module 1 error	0 and 7	0xFF11	'E','X','P','A','1'
Expansion module 2 error	0 and 7	0xFF12	'E','X','P','A','2'
Expansion module 3 error	0 and 7	0xFF13	'E','X','P','A','3'
Expansion module 4 error	0 and 7	0xFF14	''E','X','P','A','4'
Expansion module 5 error	0 and 7	0xFF15	'E','X','P','A','5'
Expansion module 6 error	0 and 7	0xFF16	'E','X','P','A','6'
Expansion module 7 error	0 and 7	0xFF17	'E','X','P','A','7'

Table 9-7 CANopen error messages

Object 0x1029 specifies the error behavior of the device. Its value is constant and always equal to 0x01. This signals that device does not change its node state if an error occurs.

9.5.4 ERROR MESSAGE

An emergency message (EMCY) shall be transmitted when any of the device situations listed in Table 9-7 CANopen error messages occur. The COB-ID of the emergency message is stored in object 0x1014. The data bytes are defined by CANopen DS301. The data is defined as follows:

Byte	0	1	2	3	4	5	6	7
	Emergency error code See Table 9-7 CANopen error messages		Error register (Object 0x1001)	Manufacturer specific error field				

9.5.5 EXPANSION MODULE ERRORS

The CAN base module firmware may encounter a timeout waiting for a periodic transmission from an expansion module. See section 7.1 SWITCH CONTACT READ 2-BIT DATA. Should a timeout occur, error objects 0x1001 and 0x1003 shall be created. An error message shall be transmitted. The firmware shall continually retry the request. Until a successful transfer, no more error objects or error messages shall be created.

10.0 LED DIMMING

LED dimming level is set by writing to object dictionary entry (LED_DIM) of the base module.

- Since there is only one RDPO (used by WRITE_LED_DATA), changing LED dimming values is accomplished through SDO for object 0x2200.
- For base modules, if (LED_DIM) sub-index 0x01 is 1:
  - The module firmware shall automatically dim all LEDs within the module according the formula:
    - The modules shall proportionally scale the three RGB LED values for each LED.
    - (Current red color level * (LED_DIM)) / 255 = Red data sent to RGB LED (0 to 255).
    - (Current green color level * (LED_DIM)) / 255 = Green data sent to RGB LED (0 to 255).
    - (Current blue color level * (LED_DIM)) / 255 = Blue data sent to RGB LED (0 to 255)
- For base modules, if (LED_DIM) sub-index 0x01 is 0:
  - The module shall ignore any dimming setting.

11.0 LED FLASH

LEDs within a CAN base module and its expansion modules shall flash simultaneously. LEDs flash is set by writing to object dictionary entry (LED_FLASH) of the base module.

First, the desired LED indicators and icons need to be enabled for flashing. See section 7.2LED WRITE 4 BYTE DATA J1939 AND CANOPEN for information regarding configuration messages. See Table 7-3 ROCKER MODULE LED WRITE DATA OBJECT 0x2000 to enable flashing.

- LED flashing by the customer application is accomplished by writing LED data per section 7.2 LED WRITE 4 BYTE DATA J1939 AND CANOPEN with byte 0 bit 6 set to a 1.
  - The ON and OFF flashing intervals for the module are located in object 0x2300. Since there is only one RDPO (used by WRITE_LED_DATA), changing these values is accomplished through SDO.
- There is no mechanism to synchronize the flash between different CAN base modules.

12.0 SLEEP MODE

All LEDs within the modules shall be turned off while in sleep mode. While in sleep mode, both periodic and event driven transmit messages will be sent. The processor does not enter deep sleep mode where internal peripherals are shut down.

12.1 CAN ROCKER MODULE

When the CAN base module enters sleep mode, it shall place the connected expansion modules in sleep mode.

The CAN base module shall enter sleep mode when:

- No CAN messages are received for four minutes.

12.1.1 CANopen

To set all devices on the bus the sleep, the PM (power management) master first sends the message "Query sleep object" with COBID 0x691 and data [0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00].

After 1000mS, the PM master sends the message "Set sleep" with COBID 0x691 and data [0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00].

To wake up all devices on the bus, the PM master sends message "Wake-up" with COBID 0x691 and no data. This is repeated every 250mS until all devices are detected.

Any device may ask the PM master to wake up (e.g. due to a push button press). Pressing a switch, the module sends a bootup message (0x700 + node ID; data 0x00), followed by the wakeup message (0x690 with no data). The wakeup message is repeated every 250mS until the PM master wakeup message (0x691 with no data) is received.

12.1.2 J1939

Sleep mode is entered by sending a proprietary PGN1 special sleep message. The address is 0x??FE00?? where "?" = do not care (PGN 65024). The data shall be [0x37, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0xEE, 0xE6].

The CAN base module shall exit sleep mode when:

- Any CAN LED data message is received addressed to the module.
- The module receives the "Wake-up" message.

12.2 EXPANSION MODULE

The expansion modules shall support a low power sleep mode. The CAN base module sets the connected expansion modules into a bus sleep mode.

The expansion module shall exit sleep mode when:

- Signaled by the CAN base module.

13.0 CUSTOMER CONFIGURATION SOFTWARE-590342-0500

Software is available for the customer to configure the module prior to installation within the application. The software also performs basic operation tests. The software has been verified to run on Windows 10.

- The customer application does not support application firmware download.
- The application can save configuration files for the customer to recall.
- There is no support for more than one module connection to the test system. Only the test tool and the module to be programmed shall be connected to the network.
- Only Kvaser hardware is supported by the customer configuration software.

It is also possible for the customer to use CAN software and hardware from any manufacturer, by creating individual CAN configuration messages. The OTTO customer configuration software simplifies the configuration process.

13.1 CAN HARDWARE INSTALLATION

To utilize the OTTO customer rocker configuration software, only CAN hardware from Kvaser shall be used.

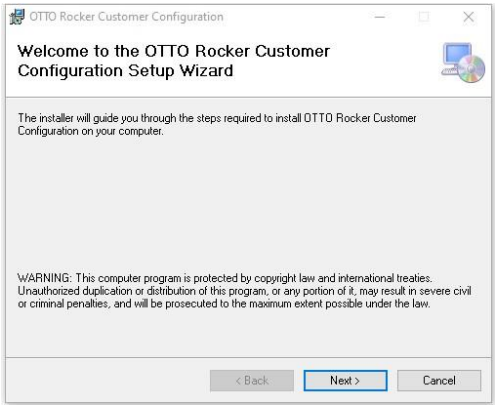
If not already installed, drivers for the CAN hardware are available from the Kvaser web site. Follow instructions for the driver installation for the hardware to be used.

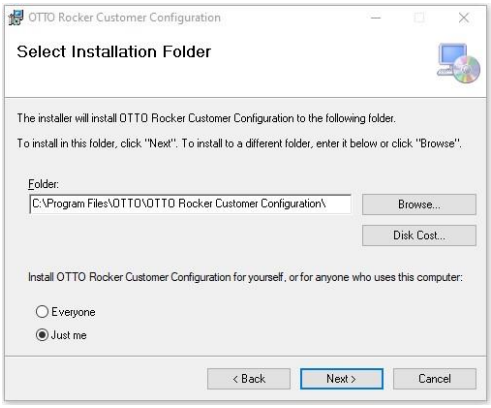
13.2 SOFTWARE INSTALLATION

This software is free. No license agreements are necessary.

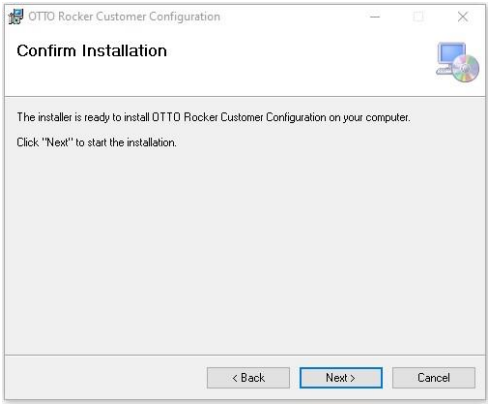
Copy the zip file contents of the 590342-0500#.zip file into a convenient directory on the computer. The "#" suffix is the revision of the OTTO installation package. Extract the zip file contents. Any appropriate zip file extractor may be used or right click the file and select Extract All.

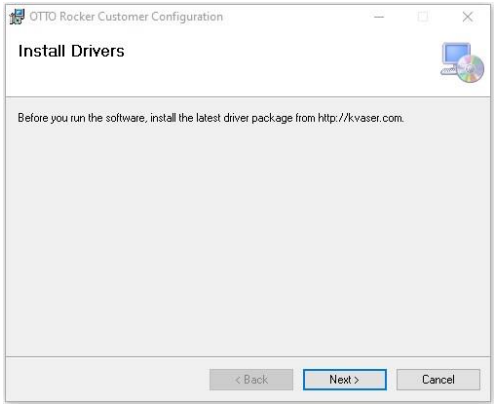
Locate and double click the "setup.exe" under the CustomerRocker directory. The "Welcome…" page will display. Click "Next". The "Select Installation Folder" page will display. You may change the installation options and file location here. Click "Next".



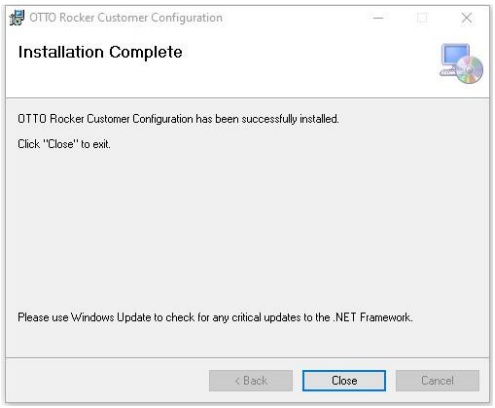


The Confirm installation page will be shown. Click "Next". You may see a pop-up regarding the SetupCustomerRocker.msi. Click "Yes". There will be an informational pop-up reminding you to install the Kvaser drivers. Click "Next".





There will be a pop-up stating the installation is complete. Click “Close”.



The program will be available using the start button (Windows taskbar) under the “OTTO” group. You may drag it to the desktop for convenience.

The installation directory contains a file “CustomerRocker.exe.config”. You may alter the contents of this text file to set the default directory for loading and saving files. Change the <add key="CONFIG_DIRECTORY" value="C:\tmp\otto" /> value within the file.

13.3 CONFIGURATION SOFTWARE

Double click the “Customer Configuration Rocker” icon on the desktop. The “Customer Configuration Rocker 590342-0500 main form will be displayed. This software allows the customer to configure and test the rocker module. Once the CAN Status displays “Connected”, testing can commence. Note that unpredictable behavior may result if the program’s loaded profile parameters do not match the values stored inside of the module. Testing should be done after sending the settings to the module.

13.3.1 File

Load Profile: Load a previously saved customer configuration or sample configuration. A file browser window will pop up looking for XML resource files (*resx).

Create CANopen Profile: Start a new CANopen configuration *.resx file. This file contains values for the configurable parameters within the rocker.

Create J1939 Profile: Start a new CANopen configuration *.resx file. This file contains values for the configurable parameters within the rocker.

Save Profile: This menu choice is used after setting the configurable parameters from Create CANopen profile or Create J1939 Profile. A file browser pop-up will be displayed to save the configuration to the user's computer.

Send settings to module: This is the only menu choice requiring the use of the Kvaser hardware. When selected, the current settings will be downloaded to the rocker.

After the settings are sent to the module, testing the module may be performed.

- Depressing a button on the rocker will change the gray circle icon to green.
- Left clicking on any one of the square icons within a button will turn on the associated LED. The LED sequence is red, green, blue, then off.
- Left clicking on the square icon within the encoder will turn the white backlight.

Quit: Exit the Customer Configuration Rocker program.

13.3.2 Help

The help menu contains information regarding hardware setup and hints for setting some configurable parameters.

13.3.3 Current baud rate

Set this dropdown to the baud rate the module is currently communicating with. It may be necessary to select different rates if communications to the module cannot be established when sending settings to the module.

13.4 CONFIGURING A ROCKER

Select "Create J1939 Profile" from the File menu. A representative rocker image will be shown and another drop down below CAN status will be shown. The new (initially blank) drop down allows different features to be configured within the rocker.

13.4.1 Object dictionary CANopen

See 9.4.1 CANOPEN OBJECT DICTIONARY

The object dictionary is a table to set the behavior of the CAN rocker on the customer network. Some objects are read only. The object dictionary allows the customer to build the CAN messages (COB) they desire for communication over the network.

When installed into a customer network, object dictionary entries may be individually read or modified using a SDO message. This SDO is inefficient since it contains four data bytes of overhead within each CAN message. LED dimming and flashing is accomplished with the SDO message. Setting LED colors using the configurable RX-PDO (0x1400 and 0x1600) is the preferred method. See 8.6.2.2 LED COLOR AND INTENSITY.

The preferred method for switch contact communication is to build TX-PDOs. Up to eight TX-PDO messages may be configured for use within the customer network. Each TX-PDO CAN address, data and transmit rate (event timer) is configurable. There is a limitation regarding the switch content data. The minimum resolution is one STATUS data byte. See 7.1.1 ROCKER SWITCH CONTACT STATUS BYTES.

13.4.2 Object dictionary J1939

See 8.6.1 J1939 OBJECT DICTIONARY.

The object dictionary is a table to set the behavior of the CAN rocker on the customer network. Some objects are read only. The object dictionary allows the customer to build the PGN they desire for communication over the network.

When installed into a customer network, object dictionary entries may be individually read or modified using the PDU1 (destination specific) proprietary message. This PDU1 is inefficient since it contains four data bytes of overhead within each CAN message. LED dimming and flashing is accomplished with the PDU1 proprietary message. Setting LED colors using the configurable RX-PDU (0x1400 and 0x1600) is the preferred method. See 8.6.2.2 LED COLOR AND INTENSITY.

The preferred method for switch contact communication is to build TX-PDUs. Up to eight TX-PDU messages may be configured for use within the customer network. Each TX-PDU CAN address, data and transmit rate (event timer) is configurable. There is a limitation regarding the switch content data. The minimum resolution is one STATUS data byte. See7.1.1 ROCKER SWITCH CONTACT STATUS BYTES.

13.4.3 Additional settings

Most of the data on the page is informational. It is recommended to set the “Expansion setting” drop down to the desired module operation before changing entries within the object dictionary. Entries in the object dictionary may be read only, determined by the “Expansion setting”.

13.4.4 LSS settings

The desired Node-ID (J1939 Source Address) and baud rate is entered on this page.

13.5 CHANGING TRANSMIT MAPPING PARAMETERS

This is an outline for the steps to configure CAN messages for the customer network. This section reflects using the customer configuration rocker software. There are eight transmit CAN messages that may be used. Each message supports up to 8 bytes of data.

Note that communication object 0x1800 is paired with mapping object 0x1A00. 0x1801 is paired with 0x1A01, and so on.

13.5.1 CAN ADDRESS

- Select the “Object dictionary J1939” or “Object dictionary CANopen” dropdown after loading or creating profile.
- Change object 0x1800:01 to the desired CAN address.
- Optionally set 0x1800:05 to the periodic transmit rate (in milliseconds).

13.5.2 CAN DATA

- Select the “Object dictionary J1939” or “Object dictionary CANopen” dropdown after loading or creating profile.
- Only mappable objects may be transmitted. For example, object 0x4000. See the object dictionary table.
- Up to 8 bytes of data may be transmitted. Pay attention when mapping data objects to count the number of bytes.
- Fill in mapping object 0x1A0# sub index 0x01 through 0x08 with the object index(es) of the data to be transmitted. For example, 0x40000108 (Object 0x4000, sub-index 0x01, 0x08 bits of data).
- Fill in mapping object 0x1A0# sub-index 0x00 with the number of mapped objects. Do not enter the number of bytes transmitted. Note that the Customer Configuration Rocker software may automatically do this for you.