Application Note
"RS232 to CANopen Gateway"

EPOS 24/1, EPOS 24/5, EPOS 70/10
Firmware version 2010h or higher

Introduction

The EPOS is a digital positioning system suitable for DC and EC (brushless) motors with incremental encoders in a modular package. The performance range of these compact positioning controllers ranges from a few watts up to 700 watts.

A variety of operating modes means that all kinds of drive and automation systems can be flexibly assembled using positioning, speed and current regulation. The in-built CANopen interface allows networking to multiple axis drives and online commanding by CAN bus master units.

For simple point to point communication, the EPOS also supports an RS232 interface. In order to access a network using the RS232 protocol, the EPOS includes RS232 to CANopen gateway functionality.

Objectives

This application note explains the functionality of the built-in communication gateway RS232 to CANopen. Advantages and disadvantages of this communication structure are discussed.

Required Tool

maxon motor EPOS Graphical User Interface GUI Version 1.10 or higher
Freely available at http://www.maxonmotor.com category «Service», subdirectory «Downloads», Order number 280937, 280938, 302267, 302287, 275512, 300583

References

maxon motor EPOS Communication Guide
maxon motor EPOS Firmware Specification
Freely available at http://www.maxonmotor.com category «Service», subdirectory «Downloads», Order number 280937, 280938, 302267, 302287, 275512, 300583

Communication Structure

Using the gateway functionality, the master can access all other EPOS devices connected to the CAN Bus via the RS232 port of the gateway device. Even other CANopen devices (i.e. I/O modules) supporting the CANopen standard CiA DS 301 can be accessed. The figure below shows the communication structure.

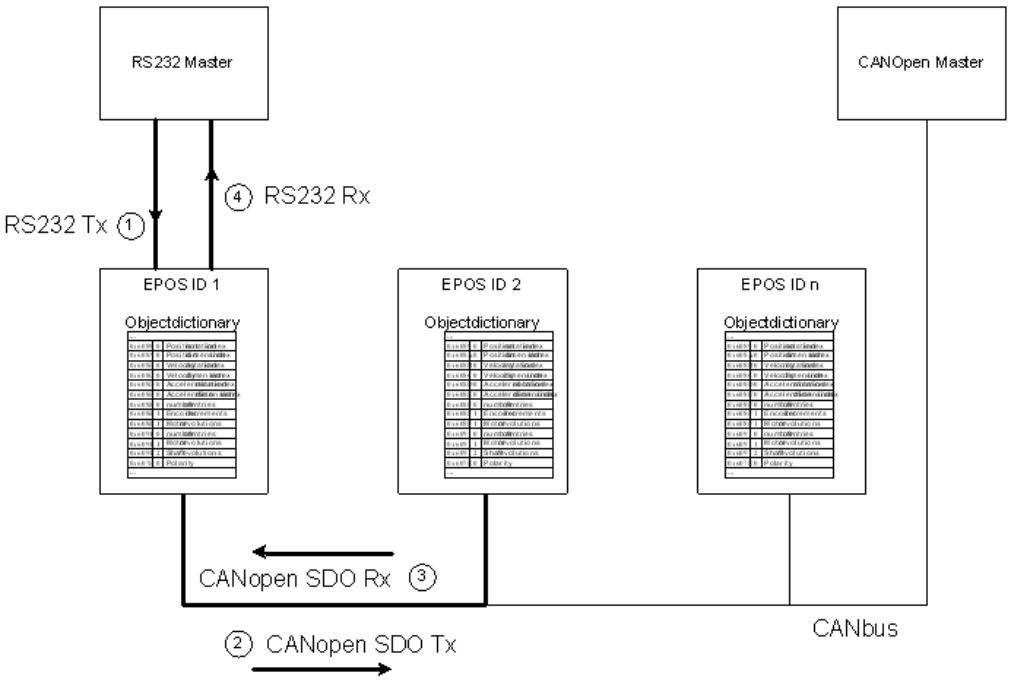

Figure 1: Communication Structure Gateway

Step	Protocol	Sender ⇨ Receiver	Description
Step 1	RS232, maxon specific	RS232 Master ↓ EPOS ID 1, Gateway	Command including the node ID is sent to the device working as a gateway. The gateway decides whether to execute the command or to translate and forward it to the CAN bus. **Criteria:** Node ID = 0 (Gateway) ➝ Execute Node ID = DIP-Switch ➝ Execute else ➝ Forward to CAN
Step 2	CANopen, SDO	EPOS ID 1, Gateway ↓ EPOS ID 2	The gateway is forwarding the command to the CAN network. The RS232 command is translated to a CANopen SDO service.
Step 3	CANopen, SDO	EPOS ID 2 ↓ EPOS ID 1, Gateway	The Epos ID 2 is executing the command and sending the corresponding CAN frame back to the gateway.
Step 4	RS232, maxon specific	EPOS ID 1, Gateway ↓ RS232 Master	The gateway is receiving the CAN frame corresponding to the SDO service. This CAN frame is translated back to the RS232 frame and sent back to the RS232 master.

The communication data is exchanged between the RS232 master and the gateway using the RS232 protocol. This protocol is maxon specific. Between the gateway and the addressed device the data is exchanged using the CANopen SDO protocol according to the CiA Standard DS 301.

Communication Example

In order to clarify the behaviour of the communication via gateway this section shows a simple example. The process of reading an 32-bit parameter from a device in the CAN network is shown.

Object: DeviceType, Index 0x1000, SubIndex 0x00
Node: 2
RS232 Command: ReadObject
CANopen Service: SDO Upload (Expedited Transfer)

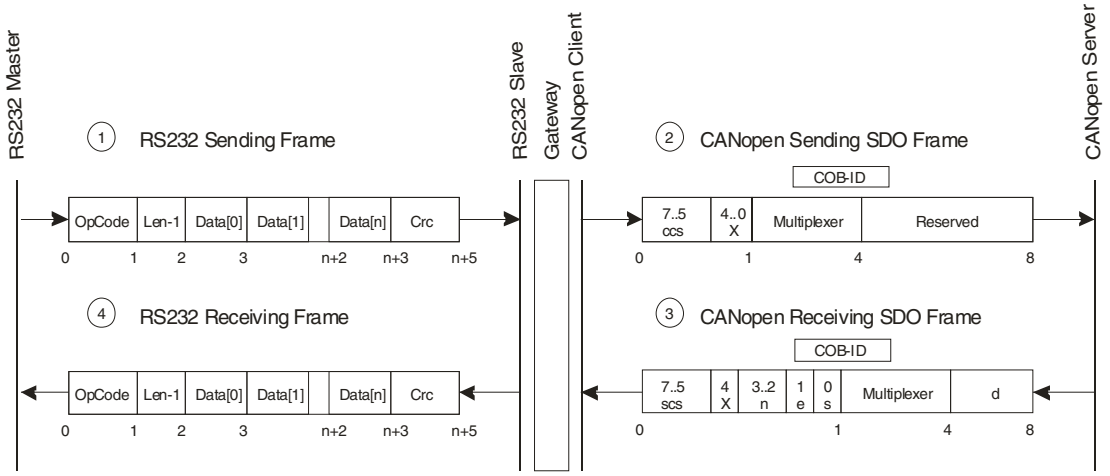

Figure 2: Communication Example

Step 1: RS232 Sending Frame		
OpCode	0x10	ReadObject command
Len-1	0x01	2 Data Words
Data[0]	0x00	Index Byte 0
Data[1]	0x10	Index Byte 1
Data[2]	0x00	SubIndex
Data[3]	0x02	Node Id
CRC[0]	0x10	Checksum Byte 0
CRC[1]	0xCD	Checksum Byte 1

Step 2: CANopen Sending SDO Frame		
COB-ID	0x602	0x600 + Node Id
Data[0]	0x40	ccs = 2
Data[1]	0x00	Index Byte 0
Data[2]	0x10	Index Byte 1
Data[3]	0x00	SubIndex
Data[4]	0x00	reserved
Data[5]	0x00	reserved
Data[6]	0x00	reserved
Data[7]	0x00	reserved

Step 4: RS232 Receiving Frame		
OpCode	0x00	Answer to ReadObject
Len-1	0x03	4 Data Words
Data[0]	0x00	ErrorCode Byte 0
Data[1]	0x00	ErrorCode Byte 1
Data[2]	0x00	ErrorCode Byte 2
Data[3]	0x00	ErrorCode Byte 3
Data[4]	0x92	DeviceType Byte 0
Data[5]	0x01	DeviceType Byte 1
Data[6]	0x02	DeviceType Byte 2
Data[7]	0x00	DeviceType Byte 3
CRC[0]	0xEB	Checksum Byte 0
CRC[1]	0x6D	Checksum Byte 1

Step 3: CANopen Receiving SDO Frame		
COB-ID	0x582	0x580 + Node Id
Data[0]	0x43	scs = 2, n = 0, e = 1, s = 1
Data[1]	0x00	Index LowByte
Data[2]	0x10	Index HighByte
Data[3]	0x00	SubIndex
Data[4]	0x92	DeviceType Byte 0
Data[5]	0x01	DeviceType Byte 1
Data[6]	0x02	DeviceType Byte 2
Data[7]	0x00	DeviceType Byte 3

Legend:
ccs: client command specifier (Bit 7 ... 5)
scs: server command specifier (Bit 7 ... 5)
n: Only valid if e = 1 and s = 1, otherwise 0.
If valid it indicates the number of bytes in Data [Byte 4 - 7] that do not contain data.
Bytes [8 - n, 7] do not contain segment data. (Bit 3 .. 2)
e: transfer type (0: normal transfer; 1: expedited transfer) (Bit 1)
s: size indicator (0: data set size is not indicated; 1: data set size is indicated) (Bit 0)

Command Translation

The RS232 command set is designed to be very close to the CANopen services. All RS232 commands have a directly corresponding service on the CAN network. This approach helps to simplify the gateway functionality. No data has to be stored and buffered between two following RS232 commands. The memory usage of the gateway can be minimized. All received data are directly forwarded to the CAN bus.

RS232 command		CANopen service
ReadObject	⇨	Initiate SDO Upload / Expedited Transfer
InitiateSegmentedRead	⇨	Initiate SDO Upload / Normal Transfer
SegmentRead	⇨	Upload SDO Segment
WriteObject	⇨	Initiate SDO Download / Expedited Transfer
InitiateSegmentedWrite	⇨	Initiate SDO Download / Normal Transfer
SegmentWrite	⇨	Download SDO Segment
SendNMTService	⇨	NMT Service

This kind of implementation gives a greater responsibility to the programmer of the RS232 protocol. The programmer has to decide whether to use the non-segmented commands or the segmented commands. This decision has to be made considering the number of data bytes to be transmitted. The non-segmented commands (ReadObject, WriteObject) support only up to 4 data bytes. More than 4 data bytes have to be transmitted using the segmented commands (InitiateSegmentedRead, SegmentRead, InitiateSegmentedWrite, SegmentWrite).

Limiting Factors

The number of segments has a big influence on the performance of the data exchange. Exchanging data directly with a device connected to RS232 (no gateway), a data segment can transfer up to 63 Bytes per command. This means that for 1kB of data, 17 commands have to be sent. In comparison, sending this data to a device addressable via gateway, 147 commands have to be sent. The CANopen services (Normal Transfer) allow only 7 bytes to be transferred in a segment. So the CANopen segment limits also the RS232 segment. Remember: the gateway is not able to buffer data and to split the data into several CANopen services.

Considering the segment size, the CANopen is the limiting factor for the communication performance. In considering the bitrate of the two field buses, the RS232 is the limiting factor. The communication via gateway can not profit from the high bitrate of the CAN bus. So the communication via gateway is limited by a slow bitrate (RS232) and a small segment size (CANopen).

	RS232 protocol	CANopen	RS232 to CANopen Gateway
Max. Bitrate	115.2 kBit/s	1 MBit/s	115.2 kBit/s
Max. Segment Size	63 Bytes	7 Bytes	7 Bytes
Conclusion	Slow Transfer Rate Big Segment Size	Fast Transfer Rate Small Segment Size	Slow Transfer Rate Small Segment Size

Figure 3: Limiting Factors

However, these limiting factors have to be put in perspective, because most of the objects in the object dictionary are 32-bit parameters or even smaller. So segmented transfer is very rarely used. Only for reading the data buffer of the data recorder or for a firmware download, does the segmented transfer need to be used.

Timing

The primary bottleneck of the communication via gateway is the RS232 bitrate. The maximum RS232 bitrate of 115.2 KBit/s is 10x smaller than the maximum CAN bitrate of 1 MBit/s. The duration of the communication depends more or less on the RS232 bitrate used. The timing example below shows the delaying of the communication addressing a device via the gateway.

Timing Example

Test Platform	Pentium 4, 2.66 GHz, Windows XP, EPOS_UserInterface
Command	ReadObject, 32-Bit Object
RS232 Bitrate	38400 Bit/s (Default)
CAN Bitrate	1 MBit/s (Default)
Time via Gateway	10.125 ms (measured)
Time without Gateway	9.995 ms (measured)
Delay	130 µs

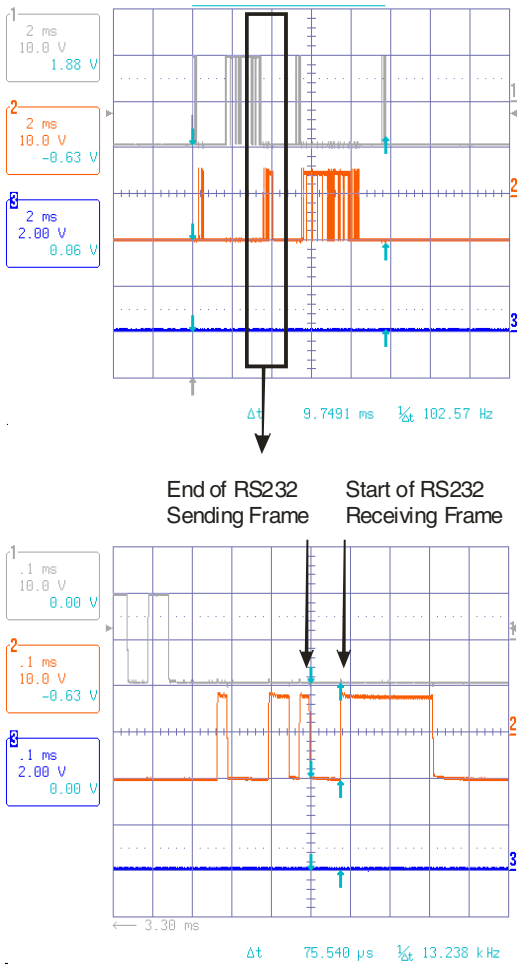

Figure 4: Timing Communication without Gateway

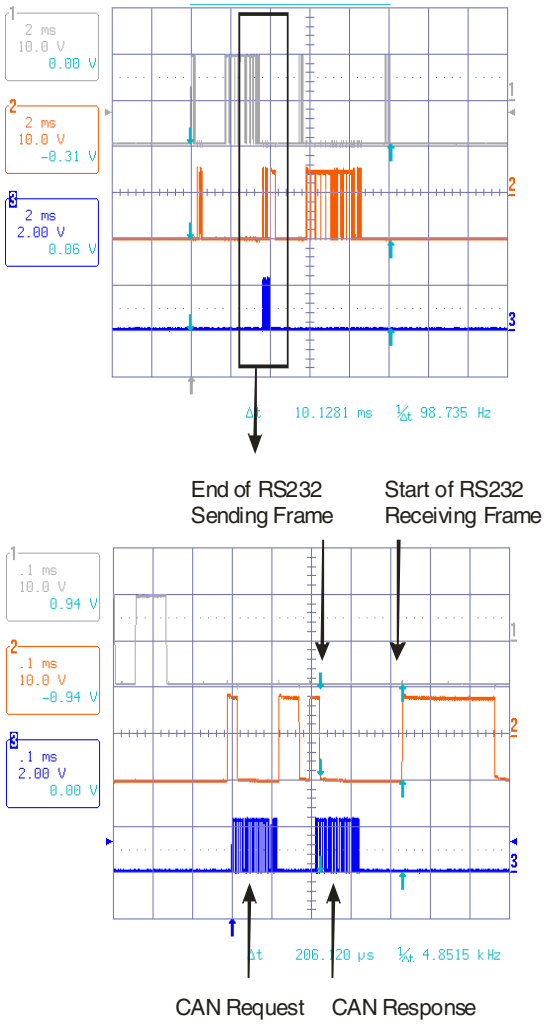

Figure 5: Timing Communication with Gateway

Legend: CH1: RS232 Tx
CH2: RS232 Rx
CH3: CAN High/Low

Theoretical Timing Values (Calculated)

Bitrate	Read/Write 8-bit / 16-bit / 32-bit Object (2 CAN frames @ 8 bytes)	Read/Write 1kByte Object (296 CAN frames @ 8 data bytes)
1 MBit/s	260 µs	38.48 ms
800 kBit/s	325 µs	48.1 ms
500 kBit/s	520 µs	76.96 ms
250 kBit/s	1040 µs = 1.04 ms	153.92 ms
125 kBit/s	2080 µs = 2.08 ms	307.84 ms
50 kBit/s	5200 µs = 5.2 ms	769.6 ms
20 kBit/s	13000 µs = 13 ms	1924 ms = 1.924 s

Figure 6: Theoretical Timing CAN Bus (CANopen SDO services)

Bitrate	Read/Write 8-bit / 16-bit / 32-bit object (2 RS232 frames @ 10/14 bytes)	Read/Write 1kByte object (296 RS232 frames @ 8/18 bytes)
115200 Bit/s	2.083 ms	0.33 s
57600 Bit/s	4.16 ms	0.66 s
38400 Bit/s	6.25 ms	0.99 s
19200 Bit/s	12.5 ms	1.99 s
14400 Bit/s	16.6 ms	2.66 s
9600 Bit/s	34.47 ms	3.99 s

Figure 7: Theoretical Timing RS232 (maxon specific protocol)

Monitoring Devices using the EPOS Graphical User Interface GUI

The EPOS devices on the network can be searched using the command ‘Search Nodes’ under the menu 'Communication’. For all detected devices the connection type is displayed. Either the device is found via RS232 or via RS232-CAN Gateway.

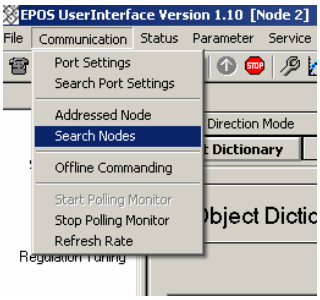

Figure 8: Menu Communication

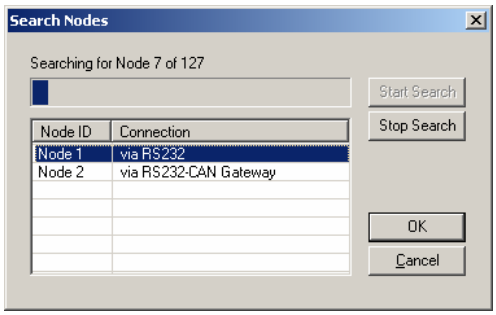

Figure 9: Search Nodes

The addressed EPOS device can then be selected and the whole functionality of the Graphical User Interface can be switched to one of the devices in the network.

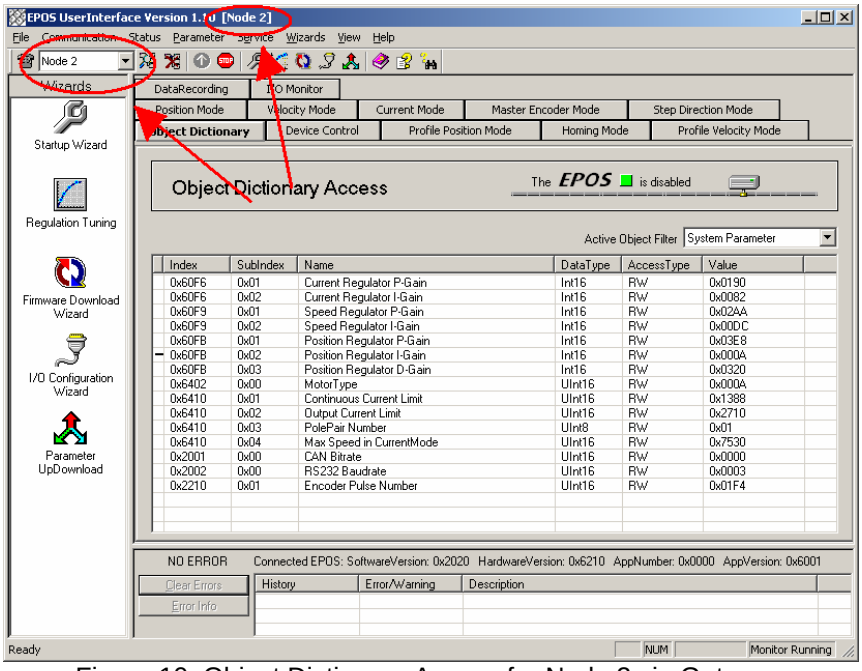

Figure 10: Object Dictionary Access for Node 2 via Gateway

Conclusion

The gateway functionality enables the user to easily connect to the CAN network. The user does not need a separate CAN interface card for monitoring a CAN network. Also, the wiring of the CAN network does not have to be altered. By just plugging in the RS232 cable to one of the EPOS devices, the user is able to control and monitor all EPOS devices in the network.

The communication delay for the CAN communication can be ignored compared to the duration for the RS232 baudrate. This means the gateway does not slow down the RS232 communication. It doesn’t make any difference whether the master is addressing a device directly via RS232 or via the gateway in the CAN network (Exception: segmented transfers).