maxon motor control		
Maxon motor ag Brünigstrasse 220 CH – 6072 Sachseln www.maxonmotor.com	**EPOS4: Basic rules and hints for commanding via RS232/UART**	Version: V1-11 Date: 2018-11-16 Author: ROMA

Contents

Documents

Scope of the document:

This document provides some additional information and limitations which are important to take care in case of any implementation of customized libraries or master's application code commanding EPOS4's UART/USB protocol.

This article addresses experienced software developers with some base knowledge about communication protocols and the need to implement the EPOS4 RS232 or USB protocol on their master system.

Hint:
If there is a PC (with Windows or Linux), Raspberry Pi, BegaleBone, or nVIDIA Jetson TX2 as a master system in use, it is possible to use the Epos Command Library, which has the required protocol implementation internally included and provides high level functions for motion control commanding.

If the Epos Command library can be in use, there is no need to investigate in low level RS232 or USB protocol implementation. The development of the master's application code can focus on the complex high level motion control function calls.

Additional documentation:

Please refer to the following document to find the required base information about the protocol's OpCodes, Data Link Layer, Frame Structure, etc.:

- "EPOS4 Communication Guide"

The document is located in the following default path on the PC's local drive after installation of EPOS Studio:

C:\Program Files (x86)\maxon motor ag\EPOS Positioning Controller\EPOS4\04 Programming

Command processing

Communication sequence

Please take care of the following strict order!
Check the corresponding subchapters (e.g. “Frame structure”, “Error Control”, “Character Stuffing”, “Transmission Byte Order”) of the chapter “2 USB & RS232 Communication” of the “EPOS4 Communication Guide” too.

1. **Setup the desired data stream**.
   - ➪ Low byte has to be transmitted first (i.e. high and low bytes have to be swapped).
   - ➪ Detailed frame structure is described in the “EPOS4 Communication Guide”

2. **Calculate CRC** and add it to the data frame.
   - ➪ Please ensure that the CRC is calculated correctly.
     If the CRC is not correct, the command will not be accepted and processed.
   - ➪ CRC calculation does not(!) include “character stuffing”.

3. **Add “character stuffing”** to the data frame.

4. **Transmit the data** stream.

5. **IMPORTANT: Wait until you receive the reply message** of the EPOS4
   - ➪ **The master’s application code must wait** for the reply of EPOS4
     before the next command is transmitted!
   - ➪ The protocol is based on the following simple principle:
     - ▪ The master sends a request ot the EPOS4.
     - ▪ The EPOS4 replies (or the master detects a timed-out communication).
   - ➪ EPOS4 cannot process data simultaneously!
     - ▪ Do not send any data before the reply (or a timeout) is present!

6. **IMPORTANT: Check the content of the EPOS4’s reply and included error code**!
   - ➪ The EPOS4’s reply holds two bytes representing an error code.
   - ➪ If the error code is unequal ‘0’, the command was not(!) be processed.
     - ▪ Check chapter “6 Communication Error Code Definition”
       of the “EPOS4 Communication Guide” about error details.
     - ▪ Fix the root cause of the error before you send the data frame again.

Trouble shooting: EPOS4 does not respond

1. Ensure that the **latest EPOS4 firmware** is installed
   - ⇨ EPOS Studio -> Wizards -> Firmware Update

2. Ensure that the **EPOS4's supply voltage is on** and plugged in.
   - ⇨ The green or red LED must be on (or blinking).
   - ⇨ Pure USB bus powering is not(!) sufficient to establish communication.

3. Ensure that the **identical bit rate is configured** by the EPOS4 and the master.
   - ⇨ Check EPOS4's "RS232 bit rate" by object 0x2002.

4. Ensure that the data frame and **its complete content is correct**.
   - ⇨ Typically a wrong CRC or missing "Character stuffing" is one cause of failing messages.

5. Ensure that the **motor phase cable are shielded**.
   - ⇨ If there is no shielding of the motor phase cable present and it is close (e.g. in the same cable channel) like the RS232 cable, there is a high risk of electromagnetic noise and disturbed RS232 communication (or other signal lines too like encoders and hallsensors).
   - ⇨ Ensure that the shielding of the motor cable is attached to broad Earth potential.

6. **Record a communication bus trace** if further analysis is required.
   - ⇨ If none of the measures and checkpoints mentioned above help to clarify the root cause of a communication problem, there is the need to record the bus communication.
   - ⇨ A bus trace can be recorded by a so-called Sniffer Software and a special adapter connecting the Rx and Tx signal lines.
     - ▪ Google for "RS232 sniffer" to find software and required cables or wiring diagrams.
   - ⇨ If there is a logic analyzer available, the signals of the Rx and TX lines can be also recorded.

7. **Submit a request to the maxon support**: http://support.maxonmotor.com.
   Please provide the following information for analyzing by the maxon experts:
   - ⇨ Communication bus trace and / or logic analyzer files.
   - ⇨ EPOS4 *.dcf configuration file.
   - ⇨ Diagrams of the communication cable's wiring plus photos of the wiring.
   - ⇨ A list of typically processed commands and the commanding cycle time.
   - ⇨ Some notes about the program structure or even program code extracts.

Additional information

The following information is mainly an extract of the information included in the "**EPOS4 Communication Guide**".
Please read the chapter "**2 USB & RS232 Communication**" of this manual too and check at least the subsections with identical titles like below.

Data Format (RS232)

Data are transmitted in an asynchronous way, thus each data byte is transmitted individually with its own start and stop bit. The format is

- 1 start bit,
- 8 data bits,
- no parity,
- 1 stop bit.

Most serial communication chips (SCI, UART) can generate such data format.

Character Stuffing

The sequence "DLE" and "STX" are reserved for frame start synchronization. If the character "DLE" appears at a position between "OpCode" and "CRC" and is not a starting character, the character mustbe doubled (character stuffing). Otherwise, the protocol begins to synchronize for a new frame. The character "STX" needs not to be doubled.

Examples:

Sending Data	0x21, **0x90**, 0x45
Stuffed Data	0x21, **0x90**, **0x90**, 0x45

Sending Data	0x21, **0x90, 0x02,** 0x45
Stuffed Data	0x21, **0x90**, **0x90**, **0x02**, 0x45

Sending Data	0x21, **0x90**, **0x90**, 0x45
Stuffed Data	0x21, **0x90**, **0x90**, **0x90**, **0x90**, 0x45

Important:

Character stuffing is used for all bytes in the frame, except the starting characters!

CRC Calculation

Important Note:

- The elements **“OpCode” to “Data[Len-1]” are included in CRC calculation.**
- **Add ZeroWord** (0x0000) at the end of the data stream **before the CRC is calculated.**
- The synchronization elements **“DLE” and “STX” are not(!) included.**
- **The CRC has to be calculated before “character stuffing” is inserted.**

If these rules above are not taken into account, the CRC’s calculation result will be wrong and the command is not accepted and processed by the EPOS4!

Packet M(x):	WORD DataArray[n]
Generator Polynom G(x):	10001000000100001 (= $x^{16}+x^{12}+x^{5}+x^{0}$)
DataArray[0]:	HighByte(Len) + LowByte(**OpCode**)
DataArray[1]:	**Data[0]**
DataArray[2]:	**Data[1]**
…	
DataArray[n-1]:	**0x0000 (ZeroWord)**

C++ sample code for CRC calculation

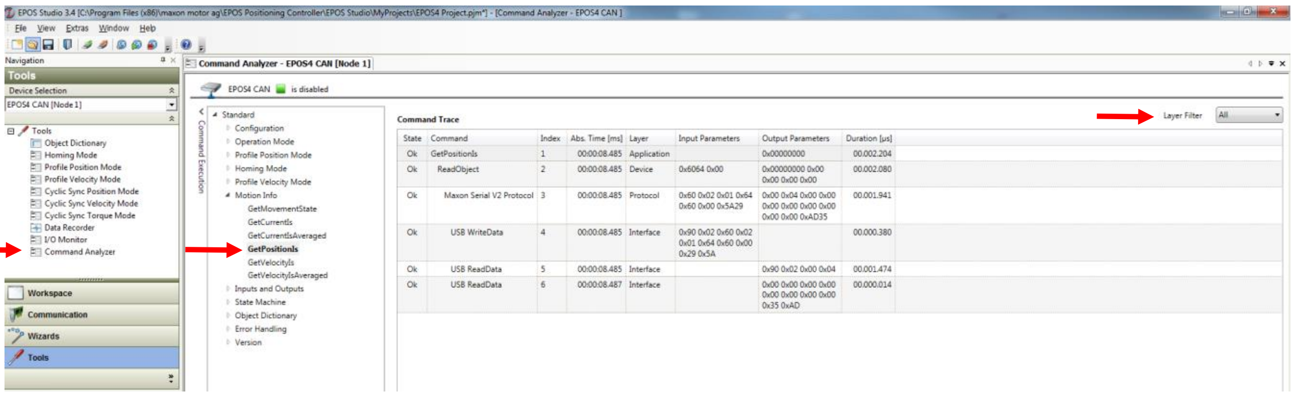
```
WORD CalcFieldCRC(WORD* pDataArray, WORD numberOfWords)
{
        WORD shifter, c;
        WORD carry;
        WORD CRC = 0;

        //Calculate pDataArray Word by Word
        while(numberOfWords--)
        {
                shifter = 0x8000;                    //Initialize BitX to Bit15
                c = *pDataArray++;                   //Copy next DataWord to c
                do
                {
                        carry = CRC & 0x8000;  //Check if Bit15 of CRC is set

                        CRC <<= 1;             //CRC = CRC * 2

                        if(c & shifter) CRC++; //CRC = CRC + 1, if BitX is set in c

                        if(carry) CRC ^= 0x1021;       //CRC = CRC XOR G(x), if carry is true

                        shifter >>= 1;         //Set BitX to next lower Bit, shifter = shifter/2

                } while(shifter);
        }
        return CRC
}
```

Command Analyzer / Frame Structure

The frame structure is described in the "EPOS4 Communication Guide" document.

EPOS Studio's "Command Analyzer" tool is a great help for investigation or comparison of a correct frame structure and data content. The EPOS4 has to be connected to the EPOS Studio via USB or RS232 because these both interfaces use the same protocol.

If a function (out of the function tree) is processed, the transmitted and received data stream is recorded depending on the "Layer filter" setting in the top right corner.



Command Trace

State	Command	Index	Abs. Time [ms]	Layer	Input Parameters	Output Parameters	Duration [µs]
Ok	GetPositionIs	1	00:00:08.485	Application		0x00000000	00.002.204
Ok	ReadObject	2	00:00:08.485	Device	0x6064 0x00	0x00000000 0x00 0x00 0x00 0x00	00.002.080
Ok	Maxon Serial V2 Protocol	3	00:00:08.485	Protocol	0x60 0x02 0x01 0x64 0x60 0x00 0x5A29	0x00 0x04 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xAD35	00.001.941
Ok	USB WriteData	4	00:00:08.485	Interface	0x90 0x02 0x60 0x02 0x01 0x64 0x60 0x00 0x29 0x5A		00.000.380
Ok	USB ReadData	5	00:00:08.485	Interface		0x90 0x02 0x00 0x04	00.001.474
Ok	USB ReadData	6	00:00:08.487	Interface		0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x35 0xAD	00.000.014