Canon Controller

Control unit for Canon EF/EF-S lenses

Communication protocol

atesystem
FOCUSED ON DETAIL

ATEsystem s.r.o.
Studentská 6202/17
708 00 Ostrava-Poruba
Czech Republic

M +420 595 172 720
E produkty@atesystem.cz
W www.atesystem.cz

DOCUMENT INFORMATION

Revision No.	Author	Revision date	Description
0	Baroš J.	18/01/2018	Document creation
1	Navrátil J.	24/01/2018	Proofreading
-	Navrátil J.	13/12/2018	English version
2	Štefanský J.	15/02/2019	Adding example of communication

Appendixes

Notes

Contact

ATEsystem s.r.o.	T	+420 595 172 720
Studentská 6202/17	F	+420 595 170 100
708 00 Ostrava 8 – Poruba	E	produkty@atesystem.cz
Czech Republic	W	www.atesystem.cz

ATEsystem s.r.o. does not assume any guarantees for the content of this document and any incidental misprints.

Names of products and companies used in this document may be trademarks or registered trademarks of their respective owners.

TABLE OF CONTENTS

1 INTERFACE PARAMETERS

Communication with remote operations controller (hereafter only ROC) is accomplished via the RS-485 bus.

Parameter	Value
Communication speed	9600 bit/s
Data bits	8
Parity	No parity bit
Stop bit	1

Tab. 1 RS485 communication parameters

Superior device (such as PC), also called master, generates queries (commands), to which the ROC as a slave responses. ROC never initiates communication; when it is idle, the device is switched to receiving. When the ROC receives a query, it switches to transmitting mode, sends a response and then returns to receiving mode.

2 COMMUNICATION STRUCTURE

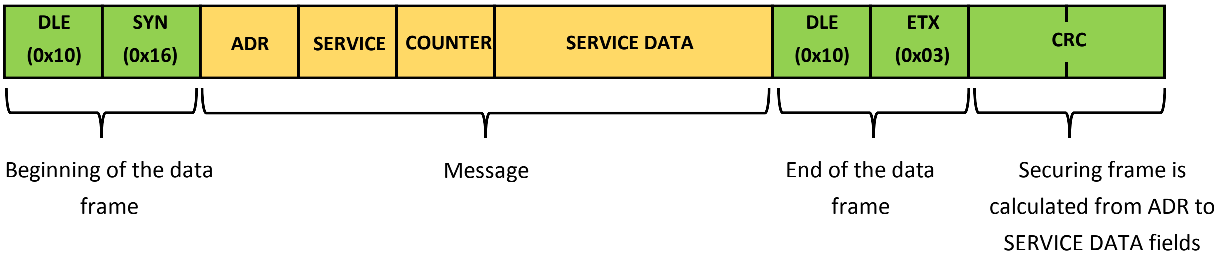

Fig. 1 – Structure of the communication frame

The beginning is unambiguously defined by the sequence of DLE and SYN characters. The sequence of DLE, ETX defines the end of the frame. After this sequence, there are two CRC bytes – a securing frame. If a DLE character appears in anywhere in the message (i.e. in fields such as ADR, SERVICE, COUNTER, and SERVICE DATA), it is automatically duplicated by the transmitting procedure. This redundant character is not included in the securing frame calculation, as well as the starting and ending sequence of the frame.

The receiving procedure then works as follows: the beginning of the frame is detected by the sequence of DLE and SYN characters. If a DLE character is detected while reading other frame's characters, it is ignored (dropped). If the following character is also a DLE, it is considered a normal valid byte of the frame. However, if a SYN character is received after the DLE character is dropped, the frame's beginning is received, and the receiver returns to the ADR character receiving status.

If ETX is received after the DLE character is dropped, it is the end of the frame, and the receiving process expects two CRC securing bytes. If another character other than DLE, SYN, or ETX is received after the DLE, it is considered an error, and the receiving procedure restarts the reception, i.e., it goes back to the state of searching beginning of the frame – the sequence of DLE, SYN characters.

The first byte of the message, the ADR character, represents the address of the slave station. The most significant bit (MSB) is reserved for the direction. Queries have this bit set to 1 and responses to 0. For example, if the address of the slave unit is 0x01, then the queries sent by the master unit have an address with the value of 0x81. Responses have the value of 0x01 in the ADR field. This applies to a slave unit with a hardware address of 0x01.

The value of SERVICE byte (bytes) defines the meaning of the message. It specifies how the slave unit should process received SERVICE DATA. CRC represents 16-bit security. All bytes of the message (i.e. ADR, SERVICE, READER, and SERVICE DATA fields) are included in the calculation. Bytes of the sequences that define beginning/end of the frame and redundant DLE characters are not included in the calculation. The calculation algorithm is further shown in C language and LabVIEW diagram.

The CRC calculation for *unsigned char *msg* message with the length of *msg_len* is as follows:

```
unsigned short crc = 0;
for(int i = 0; i < msg_len; i++)
{
        crc = crc16_table[ (crc >> 8) ^ (unsigned short)msg[i] ] ^ (crc << 8);
}
```

In the transmitted frame, the upper 8 bits of the calculated CRC are sent after the ETX character. The lowest 8 bits of the CRC form the last byte of the frame.

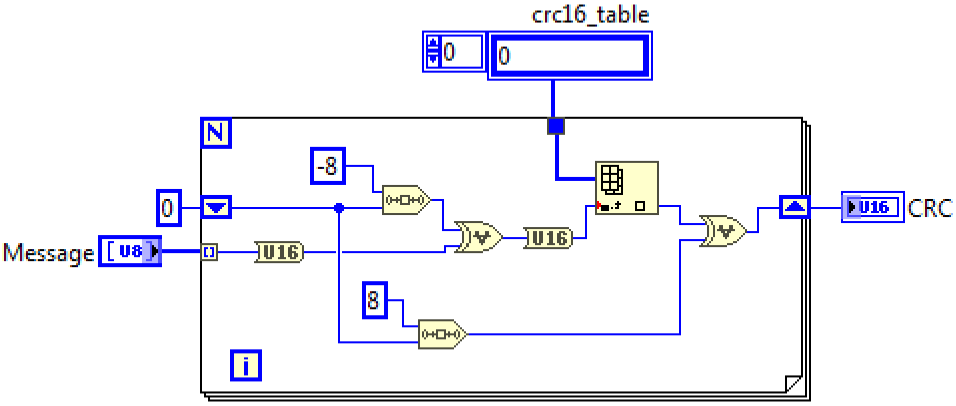

Fig. 2 – LabVIEW diagram for CRC Calculation

```
const unsigned short int crc16_table[256] =
{
0x0000, 0x8005, 0x800F, 0x000A, 0x801B, 0x001E, 0x0014, 0x8011,
0x8033, 0x0036, 0x003C, 0x8039, 0x0028, 0x802D, 0x8027, 0x0022,
0x8063, 0x0066, 0x006C, 0x8069, 0x0078, 0x807D, 0x8077, 0x0072,
0x0050, 0x8055, 0x805F, 0x005A, 0x804B, 0x004E, 0x0044, 0x8041,
0x80C3, 0x00C6, 0x00CC, 0x80C9, 0x00D8, 0x80DD, 0x80D7, 0x00D2,
0x00F0, 0x80F5, 0x80FF, 0x00FA, 0x80EB, 0x00EE, 0x00E4, 0x80E1,
0x00A0, 0x80A5, 0x80AF, 0x00AA, 0x80BB, 0x00BE, 0x00B4, 0x80B1,
0x8093, 0x0096, 0x009C, 0x8099, 0x0088, 0x808D, 0x8087, 0x0082,
0x8183, 0x0186, 0x018C, 0x8189, 0x0198, 0x819D, 0x8197, 0x0192,
0x01B0, 0x81B5, 0x81BF, 0x01BA, 0x81AB, 0x01AE, 0x01A4, 0x81A1,
0x01E0, 0x81E5, 0x81EF, 0x01EA, 0x81FB, 0x01FE, 0x01F4, 0x81F1,
0x81D3, 0x01D6, 0x01DC, 0x81D9, 0x01C8, 0x81CD, 0x81C7, 0x01C2,
0x0140, 0x8145, 0x814F, 0x014A, 0x815B, 0x015E, 0x0154, 0x8151,
0x8173, 0x0176, 0x017C, 0x8179, 0x0168, 0x816D, 0x8167, 0x0162,
0x8123, 0x0126, 0x012C, 0x8129, 0x0138, 0x813D, 0x8137, 0x0132,
0x0110, 0x8115, 0x811F, 0x011A, 0x810B, 0x010E, 0x0104, 0x8101,
0x8303, 0x0306, 0x030C, 0x8309, 0x0318, 0x831D, 0x8317, 0x0312,
0x0330, 0x8335, 0x833F, 0x033A, 0x832B, 0x032E, 0x0324, 0x8321,
0x0360, 0x8365, 0x836F, 0x036A, 0x837B, 0x037E, 0x0374, 0x8371,
0x8353, 0x0356, 0x035C, 0x8359, 0x0348, 0x834D, 0x8347, 0x0342,
0x03C0, 0x83C5, 0x83CF, 0x03CA, 0x83DB, 0x03DE, 0x03D4, 0x83D1,
0x83F3, 0x03F6, 0x03FC, 0x83F9, 0x03E8, 0x83ED, 0x83E7, 0x03E2,
0x83A3, 0x03A6, 0x03AC, 0x83A9, 0x03B8, 0x83BD, 0x83B7, 0x03B2,
0x0390, 0x8395, 0x839F, 0x039A, 0x838B, 0x038E, 0x0384, 0x8381,
0x0280, 0x8285, 0x828F, 0x028A, 0x829B, 0x029E, 0x0294, 0x8291,
0x82B3, 0x02B6, 0x02BC, 0x82B9, 0x02A8, 0x82AD, 0x82A7, 0x02A2,
0x82E3, 0x02E6, 0x02EC, 0x82E9, 0x02F8, 0x82FD, 0x82F7, 0x02F2,
0x02D0, 0x82D5, 0x82DF, 0x02DA, 0x82CB, 0x02CE, 0x02C4, 0x82C1,
0x8243, 0x0246, 0x024C, 0x8249, 0x0258, 0x825D, 0x8257, 0x0252,
0x0270, 0x8275, 0x827F, 0x027A, 0x826B, 0x026E, 0x0264, 0x8261,
0x0220, 0x8225, 0x822F, 0x022A, 0x823B, 0x023E, 0x0234, 0x8231,
0x8213, 0x0216, 0x021C, 0x8219, 0x0208, 0x820D, 0x8207, 0x0202
};
```

Tab. 2 – Value vector for CRC Calculation

3 COMMUNICATION PROTOCOL SERVICES

Multi-byte values are transmitted in little-endian format.

3.1 Service 0x00, 0x00 – Query about unit state

Query from MASTER

Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x00	First byte of the service
2	0x00	Second byte of the service
3	XX	Counter (if 0x00 is not used)

Response from SLAVE

Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x00	First byte of the service
2	0x00	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	Version of the HW unit – the whole part
6	XX	Version of the HW unit – decimal part
7	XX	Version of the FW unit – the whole part
8	XX	Version of the FW unit – decimal part
9	XX	Version of the Bootloader – the whole part
10	XX	Version of the Bootloader – decimal part
11	XX	Type of the HW unit – the whole part
12	XX	Type of the HW unit – decimal part
13	XX	0. Seconds counter byte since turning the unit on (since reset)
14	XX	1. Seconds counter byte since turning the unit on (since reset)
15	XX	2. Seconds counter byte since turning the unit on (since reset)
16	XX	3. Seconds counter byte since turning the unit on (since reset)
17	XX	Unit supply voltage – the whole part
18	XX	Unit supply voltage – decimal part
19	XX	0.Byte - The end position of the aperture - open
20	XX	1.Byte - The end position of the aperture - open
21	XX	0.Byte - current position of the aperture
22	XX	1.Byte - current position of the aperture
23	XX	0.Byte - The end position of the aperture - closed
24	XX	1.Byte - The end position of the aperture - closed

25	XX	0.Byte - The end position - distance focus
26	XX	1.Byte - The end position - distance focus
27	XX	0.Byte - current position of the focus
28	XX	1.Byte - current position of the focus
29	XX	0.Byte - The end position - close focus
30	XX	1.Byte - The end position - close focus
31	XX	0.Byte – Zoom min.
32	XX	1.Byte – Zoom min.
33	XX	0.Byte - current position of the zoom
34	XX	1.Byte - current position of the zoom
35	XX	0.Byte – Zoom max.
36	XX	1.Byte – Zoom max.
37	XX	Reserved for other possible parameters
38	XX	Reserved for other possible parameters
39	XX	Reserved for other possible parameters
40	XX	Reserved for other possible parameters
41	XX	Reserved for other possible parameters
42	XX	Reserved for other possible parameters
43	XX	Reserved for other possible parameters
44	XX	Reserved for other possible parameters
45	XX	Reserved for other possible parameters
46	XX	Reserved for other possible parameters
47	XX	Reserved for other possible parameters
48	XX	Reserved for other possible parameters
49	XX	Reserved for other possible parameters
50	XX	Reserved for other possible parameters
51	XX	Reserved for other possible parameters
52	XX	Reserved for other possible parameters
53	XX	0. Byte - Version of CANON Protocol
54	XX	1. Byte - Version of CANON Protocol
55	XX	Lens ID
56	XX	Lens type
57	XX	Status machine state
58	XX	Error register of communication with the lens
59	XX	Not used
60	XX	Status register
61	XX	Not used
62 - 102		ASCII Name of the attached lens

Error register (the meaning of setting to 1)	
Bit 0	Lens connection status (1 - not connected, 0 - connected)
Bit 1	The lens does not communicate
Bit 2	Focus exceeded the time limit
Bit 3	The lens cannot be initialized

Status register (the meaning of setting to 1)	
Bit 0	Allow reading ZOOM values (default is 0)

3.2 Service 0x46, 0x46 – Distance focus

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x46	First byte of the service
2	0x46	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	0. Byte for the number of steps
5	XX	1. Byte for the number of steps

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x46	First byte of the service
2	0x46	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	0x00 – Command received; 0x01 – Command not accepted – Lens is busy

3.3 Service 0x46, 0x4E – Close focus

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x46	First byte of the service
2	0x4E	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	0. Byte for the number of steps
5	XX	1. Byte for the number of steps

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x46	First byte of the service
2	0x4E	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	0x00 – Command received; 0x01 – Command not accepted – Lens is busy

3.4 Service 0x49, 0x43 - Closing the aperture

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x49	First byte of the service
2	0x43	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	0. Byte for the number of steps
5	XX	1. Byte for the number of steps

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x49	First byte of the service
2	0x43	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	0x00 – Command received; 0x01 – Command not accepted – Lens is busy

3.5 Service 0x49, 0x4F - Opening the aperture

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x49	First byte of the service
2	0x4F	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	0. Byte for the number of steps
5	XX	1. Byte for the number of steps

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x49	First byte of the service
2	0x4F	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	0x00 – Command received; 0x01 – Command not accepted – Lens is busy

3.6 Service 0x4C, 0x52 - Resetting lens / settings to the default position

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x4C	First byte of the service
2	0x52	Second byte of the service
3	XX	Counter (if 0x00 is not used)

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x4C	First byte of the service
2	0x52	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported
5	XX	0x00 – Command received; 0x01 – Command not accepted – Lens is busy

3.7 Service 0x55, 0x53 - ROC settings

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0x55	First byte of the service
2	0x53	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	0. Byte for settings

Response from SLAVE		
Byte		Meaning
0	address	Unit address (direction SLAVE -> MASTER)
1	0x55	First byte of the service
2	0x53	Second byte of the service
3	XX	Counter (if 0x00 is not used)
4	XX	ACK: 0x00 – OK, 0x01 – service is not supported

Byte for setting (meaning of setting to 1)	
Bit 0	Allow reading ZOOM values (default is 0)

3.8 Service 0xFF, 0xFF – ROC reset

Query from MASTER		
Byte		Meaning
0	0x80 + address	Unit address (7 bits) + 0x80 (direction MASTER -> SLAVE)
1	0xFF	First byte of the service
2	0xFF	Second byte of the service
3	XX	Counter (if 0x00 is not used)

The ROC unit **does not respond** to this service as the system is being reset.

4 EXAMPLE OF COMMUNICATION

Example of communication between PC (master) and ROC unit (slave) in text form.

Command	Message	Meaning
Query of unit state	**Query from MASTER:** 1016 8200 0000 1003 283C	direction MASTER -> SLAVE, address: 2, service: 0x00 0x00, counter is 0x00 and CRC is 0x28 0x3C
	Response from SLAVE: 1016 0200 0000 0000 0101 0000 0000 00A4 0100 0000 0053 0028 0028 007B 2E7B 2E10 1027 1200 1900 3700 0000 0000 0000 0000 0000 0000 0000 0000 0075 3491 2600 0001 0045 462D 5331 382D 3535 6D6D 2066 2F33 2E35 2D35 2E36 2049 5320 4949 0000 0000 0000 0000 0000 0000 0010 03D4 BD	direction SLAVE -> MASTER, address: 2, service: 0x00 0x00, counter is 0x00, following sequence of data is described in section 3.1 (Response from SLAVE), ACK: 0x00 – OK, 0x00 – command received, and CRC is 0xD4 0xBD
Close focus	**Query from MASTER:** 1016 8246 4E00 7002 1003 178C	direction MASTER -> SLAVE, address: 2, service: 0x46 0x4E, counter is 0x00, data for focus is 0x70 0x02 (assembled 0x0270 -> dec 624) a CRC is 0x17 0x8C
	Response from SLAVE: 1016 0246 4E00 0000 1003 370E	direction SLAVE -> MASTER, address: 2, service: 0x46 0x4E, counter is 0x00, ACK: 0x00 – OK, 0x00 – Command received, and CRC is 0x37 0x0E
Closing the aperture	**Query from MASTER:** 1016 8249 4300 2500 1003 8F2D	direction MASTER -> SLAVE, address: 2, service: 0x49 0x43, counter is 0x00, data for closing the aperture is 0x25 0x00 (assembled 0x0025 -> dec 37) a CRC je 0x8F 0x2D
	Response from SLAVE: 1016 0249 4300 0000 1003 51A5	direction SLAVE -> MASTER, address: 2, service: 0x49 0x43, counter is 0x00, ACK: 0x00 – OK, 0x00 – Command received, and CRC is 0x51 0xA5
Resetting lens	**Query from MASTER:** 1016 824C 5200 1003 41C9	direction MASTER -> SLAVE, address: 2, service: 0x4C 0x52, counter is 0x00 and CRC is 0x41 0xC9
	Response from SLAVE: 1016 024C 5200 0000 1003 8438	direction SLAVE -> MASTER, address: 2, service: 0x4C 0x52, counter is 0x00, ACK: 0x00 – OK, 0x00 – Command received, and CRC is 0x84 0x38

Tab. 3 Examples of communication