EWELLIX

RS232 interface

EWELLIX SCU Control Unit

User Manual

We pioneer motion

SCHAEFFLER

Contents

1 About the manual

1.1 Information in this user manual

This manual provides important information on how to work with the device safely and efficiently.

The manual is part of the device, must always be kept in the device's direct proximity and should be available for personnel to read at any time. All personnel working with the device must read and understand this manual before starting any work. Strict compliance with all specified safety notes and instructions is a basic requirement for safety at work.

Moreover, the accident prevention guidelines and general safety regulations applicable at the place of use of the device must also be complied with.

Validity

The information in this manual applies to the serial interface for the SCU control unit with the following identification:

- manufacturer: Schaeffler
- product name: control unit with RS232 serial interface
- type designation: SCUxx-xxxxx1-xxxx
- year of manufacture: from 2007 with firmware version V2B0
- CE marking: according to technical documentation

Authorized personnel

This manual is intended for qualified personnel who are able to develop control software for operating this product.

1.2 Symbols

Safety precautions are identified by symbols and signal words as shown. The signal words indicate the severity of the hazard and the chance it could occur. Follow these safety precautions and act cautiously in order to avoid accidents, personal injury and damage to property.

The warning and hazard symbols are defined in accordance with ANSI Z535.6-2011.

1 Warning and hazard symbols

Signs and descriptions	
⚠ DANGER	In case of non-compliance, death or serious injury will occur.
⚠ WARNING	In case of non-compliance, death or serious injury may occur.
⚠ CAUTION	In case of non-compliance, minor or moderate injury may occur.
NOTICE	In case of non-compliance, damage or malfunctions in the product or the adjacent construction may occur.

1.3 Signs

The warning, prohibition, and mandatory signs are defined in accordance with DIN EN ISO 7010 or DIN 4844-2.

2 Warning, prohibition, and mandatory signs

Signs and descriptions	
	General warning
	Electrical voltage warning
	Observe the manual
	General mandatory sign
	Ground before use
	Disconnect mains plug from electrical outlet

1.4 Availability

A current version of this manual is available at:
https://www.schaeffler.de/std/2221

Ensure that this manual is always complete and legible and is available to all persons engaged in transporting, fitting, dismantling, commissioning, operating, or maintaining the product.

Keep the manual in a safe place for immediate reference.

1.5 Legal notices

The information in this manual reflects the status at the time of publication.

Unauthorized modifications to or improper use of the product are not permitted. Schaeffler accepts no liability in these cases.

1.6 Limitation of liability

All information and notes in this manual have been compiled in accordance with the applicable standards and regulations, the present status of technology and our many years of knowledge and experience.

The manufacturer is not liable for any damage resulting from:

- failure to observe this manual
- unintended use
- employment of untrained personnel
- unauthorized conversions
- technical changes
- tampering with or removal of screws
- use of unapproved spare parts

Where the device has been customized, the actual product delivered may differ from the description provided in this manual. In such cases, please contact Schaeffler, to obtain further instructions or information on safety precautions for these devices.

We reserve the right to make technical modifications to the device to improve usability.

1.7 Images

The images in this manual may be schematic representations and may differ from the delivered device.

2 General safety regulations

This section provides an overview of all essential safety aspects for optimum personal protection as well as safe and trouble-free operation. Failure to observe this manual and the safety instructions contained herein may result in significant hazards and potentially lead to serious injury or death.

The safety program from Schaeffler details authorized users and the responsibility of individual users. The product was designed and built in accordance with the latest technical standards and accepted safety regulations. EU conformity is documented within the technical documentation.

2.1 Hazards

This section lists the residual risks identified through the risk assessment.

The manufacturer has minimized the effects of existing hazards through design and protective measures. Pay attention to the residual hazards and potential countermeasures described in the following chapters, as well as to the warning notices.

Danger to life caused by electric current

Touching conductive parts poses an immediate danger to life. Damage to insulation or individual components may pose a danger to life. Therefore, observe the following:

- In an emergency, the device must be disconnected from the power supply.
- Applications in which the device is installed must have an emergency stop switch or be isolated from the mains supply on all conductors.
- Prevent the device from being exposed to water jets.
- If the insulation is damaged, immediately switch off the power supply and have the parts repaired.
- Work on the electrical system may only be carried out by electrically skilled persons.
- Before performing any work on the electrical system, disconnect the machine from the power supply.
- Before maintenance, cleaning, or repair work, disconnect the power supply and secure it against reconnection.
- Do not bypass or disable fuses. When replacing fuses, ensure the correct current rating is used.
- Keep moisture away from conductive parts to prevent short circuits.

Risk of injury due to moving components

Rotating or linearly moving components can cause serious injury. Uncontrollable, faulty, or automatic movements may be triggered by a component defect, rare errors in RAM or ROM, or a stuck button on the hand switch. Unintended movement may also occur due to battery operation, even when the mains power supply is switched off. Therefore, observe the following:

- Do not work on moving components.
- Keep your body, including hands and arms, away from moving components.
- Switch off the mains power supply and remove any batteries.
- If a button on the hand switch becomes stuck, stop the movement using the button for the opposite direction.

Risk of injury due to crushing

When colliding with solid objects, the force exerted can cause injuries.

- Make sure that there are no persons present in the danger zone during operation.
- Ensure that no objects or persons come into contact with the push or protection tube on the front and rear attachment.

Risk of injury from pinching

If the actuator runs into fixed objects, the driving force may cause personal injury.

- If the actuator is left unattended, ensure that the full stroke length is free of obstacles and that no persons are located in the stroke area.
- Alternatively, all cables can be disconnected from the mains power supply.

Risk of injury due to damaged housing

Injury due to cracks and related openings in the housing of the device or its accessories.

- If the housing is damaged due to cracks, breakage or heavy wear, stop using the device and follow the disassembly instructions.

Risk of injury and property damage due to incorrect operation

Incorrect operation can endanger persons and objects in the danger zone of the system.

- Secure the device against unintended operation.
- Before pressing any button on the operating device, make sure that you have selected the correct one.

Property damage due to incorrect cleaning

Incorrect cleaning using unsuitable liquids, a washing machine, a high-pressure cleaner, or a steam cleaner may chemically or mechanically damage and destroy the device.

- Use pH-neutral cleaning agents.
- Do not use a high-pressure cleaner, steam cleaner, washing machine, or similar cleaning equipment.
- Use only isopropyl alcohol for manual wipe disinfection.
- Inspect the housing every six months for mechanical damage, particularly cracks.

Property damage due to incorrect rechargeable battery

Using an incorrect rechargeable battery may result in malfunction or damage. The batteries used in the device are of a special type.

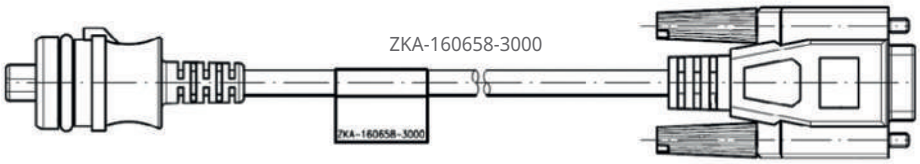
- Only use original rechargeable batteries from the manufacturer.
- Instruct maintenance personnel on how to open and close the battery compartment cover and replace the batteries.

3 Product description

This chapter describes the basic technical characteristics of the serial interface.

If the remote user does not provide a mains supply in accordance with medical standards (safety according to EN 60601–1), the end application must be grounded to ensure correct operation of the RS232 interface.

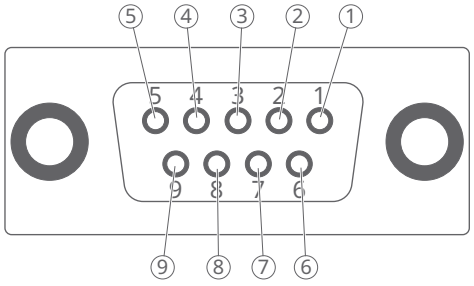
1 Recommended power cable



Physical layer

- electrical characteristics in accordance with the RS232 interface definition
- half-duplex
- bidirectional
- Baud rate: for standard control units, the baud rate is set to 38400. For customized control units, the baud rate can be set to the following data: 9600, 19200, 38400.
- plug: 9-pin sub-D (female)
- The control lines are not used. However, DTR and RTS must be permanently activated, as they supply the RS232 converter in the control unit. Instead of the DTR signals and RTS signals, a separate power source of DC 5.5 V ... DC 15 V at 30 mA can be connected (+ to pin 4 DTR or pin 7 RTS and – to Pin 5 GND)
  - + to pin 4 (DTR) or pin 7 (RTS)
  - – to pin 5 (GND)

2 Pin assignment (socket)



1	Not assigned	2	RxD
3	TxD	4	DTR
5	GND	6	Not assigned
7	RTS	8	Not assigned
9	Not assigned		

Data link layer

- 1 start bit
- 8 data bits (LSB first)
- 1 stop bit
- no parity bit
- no handshake

Network layer

- point-to-point connection (only 2 participants)
- control unit acts as slave and responds to requests from the master, e.g., PC program
- slave responds to every request from the master
- max. request delay: 2000 ms
- max. delay between individual telegram bytes: 1000 ms
- If the control is operated with batteries and the parameterization is set to **Low Power = Enabled**, the controller can be switched to remote mode by holding the RxD circuit in the **space** state (level > +3 V) for at least 100 ms (from firmware version V2B1).
- If the control is operated with batteries and the parameterization is set to **Low Power = Enabled**, and remote mode is activated but no command is present, the control does not switch to low-power mode. If communication is interrupted in this state, the controller switches to low-power mode (from firmware version V2B1).

Transport layer, telegram structure

The checksums are calculated using the standard algorithm CCITT CRC-16. The polynomial for the algorithm is CRC16 = $x^{16} + x^{12} + x^5 + 1$. The starting value is 0.

Each response includes an ACK byte that contains the device status. Some responses include the parameter ctp in P1 and P2. This defines the number of subsequent data bytes. Each response containing more than 1 B uses a ctp to define the data length.

3 Telegram structure

Request:	<X><Y>[<P1><P2> ... <Pn>]<C1><C2>
Reply:	<X><Y>[<P1><P2> ... <Pn>]<C1><C2>
Request:	<X><Y>[<P1><P2> ... <Pn>]<C1><C2>
Reply:	<X><Y>[<P1><P2> ... <Pn>]<C1><C2>

[ ]	Optional	<X>	Major command number (1 B)
<Y>	Minor command number (1 B)	<P1> ... <Pn>	Parameter bytes (Intel little-endian format, LS byte ... MS byte)
<C1>	Low byte of the 16 bit telegram checksum	<C2>	High byte of the 16 bit telegram checksum

4 Telegram description

Request:	<X><Y>[<P1><P2> ... <Pn>]<C1><C2>
Reply:	<X><Y><ACK>[<<ctp1><ctp2>=n><P><P4> ... <Pn+2>]<C1><C2>

4 Communication protocol

4.1 Command set

The following commands are available when the mains voltage is on or when operating on battery power:

- The RO command activates the remote function.
- The RA command terminates the remote function.
- To maintain the remote function, the RC command must be executed in a repeated cycle at least every 1000 ms. Each additional remote command (RG, RT, RC, RE, RS, except for RO, RA) must be executed in a repeated cycle at least every 500 ms.
- The commands RG, RT, RC, RE, and RS are only available when the remote function is activated.

The control lines DTR and RTS must be permanently switched on (active) to supply power to the RS232 converter and enable communication with the control unit.

3 Command set

Cmd <X> Y>	Name	Query parameters						Reply parameters					Description
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	Pn	
RG	Remote data get	data_ID [0]	data_ID [1]	–	–	–	–	ACK, **e	ctp 1	ctp2	1st byte	n-2nd byte	Data transmission from SCU P2 to Pn as response only if P1 = ACK
RT	Remote data transfer	ctp1	ctp2	data_ID [0]	data_ID [1]	1st byte	n-4th byte	ACK, **e	–	–	–	–	Data transmission to SCU. Only data_Ids 3xxx are permitted (RemoteData)!
RC	Remote cyclic	ctp1	ctp2	Index of cyclicObj	1st byte write data of cyclicObj	2nd byte write data of cyclicObj	n-3rd byte write data of cyclicObj	ACK, **e	ctp 1	ctp2	1st byte read data of cyclicObj	n-3rd byte read data of cyclicObj	Must be sent at least every 500 ms so that the SCU remains in remote mode (WDT). If P3 = –1, no data are transmitted; otherwise, P3 is the index for cyclicObj, which defines the query data/response data P3 = 0: cyclicObj with dataID = 3001 P2 to Pn are only included in the response if P1 = ACK
WK	RE Remote execute function	fnc_ID	para_ID [0]	para_ID [1]	–	–	–	ACK, **e	–	–	–	–	Execution of a function. P1 is the index in the function list. P2, P3 are additional function parameters, please observe. Note definitions of fnc_ID and para_ID.
SG	Remote stop function	fnc_ID	para_ID [0]	–	–	–	–	ACK, **e	–	–	–	–	Stops a function. P1 is the index in the function list. P2, P3 are additional function parameters, please observe. Note definitions of fnc_ID and para_ID.
RO	Remote mode open	Safety_ID	para_ID [0]	–	–	–	–	ACK, **e	–	–	–	–	Safety_ID = 0: If a communication time out (0.5 s) occurs, all movements are terminated, i.e., no movement is started. If a communication time out occurs, only the RC, RA, and RO commands are available. Safety_ID = 1: If a communication time out (0.5 s) occurs, all movements are terminated. Safety_ID = 2: If a communication timeout (0.5 s) occurs, only the remote movement is terminated. (for firmware version V2B3)
RA	Terminate remote mode	–	–	–	–	–	–	ACK, **e	–	–	–	–	Set the SCU to normal mode (without reset).

4.2 Communication errors and abbreviations

4 Communication errors

Code	Hex	Dec	Name	Description
ACK	06	6	Command acknowledged	Request accepted
CSE	80	128	Checksum error	Error in the telegram checksum
PDE	81	129	Parameter data error	Error in the telegram data bytes
PCE	82	130	Parameter count error	Incorrect counter level of the telegram data bytes
ICE	83	131	Invalid command error	Unknown command code
PE	84	132	Permission error	Command not permitted in the current SCU mode or SCU state

5 Abbreviations

Code	Data	Description
ctp	Dyn	Number of subsequent telegram data bytes
data_ID	Dyn	Index in data list
fnc_id	Dyn	Index for function list
para_id	Dyn	Additional parameter depending on the function

4.3 Data list

The specific settings and the status of the control can be queried via the data list (command RG). Both individual data and entire blocks can be queried. The data with collection index = 3000 can be described using the command RT.

6 Data list: data elements 0000 ... 2000

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	0000	0001	Firmware info Name Version CS	STRING	Size = 31 B
		0002	Configuration info Name Version CS	STRING	Size = 36 B
0000	0010	0011	Actual_Position Actuator 1	INT32	Unit: encoder edges (count)
		0012	Actual_Position Actuator 2	INT32	
		0013	Actual_Position Actuator 3	INT32	
		0014	Actual_Position Actuator 4	INT32	
		0015	Actual_Position Actuator 5	INT32	
		0016	Actual_Position Actuator 6	INT32	
0000	0020		Actual_State_Binary Inputs 1 ... 4	UINT8	Logic level Bit 0: binary input 1 (0 = not active/ 1 = active) Bit 1: binary input 2 (0 = not active/ 1 = active) Bit 2: binary input 3 (0 = not active/ 1 = active) Bit 3: binary input 4 (0 = not active/ 1 = active) Input level Bit 4: binary input 1 (0 = not active/ 1 = active) Bit 5: binary input 2 (0 = not active/ 1 = active) Bit 6: binary input 3 (0 = not active/ 1 = active) Bit 7: binary input 4 (0 = not active/ 1 = active)
0000	0030	0031	Actual_State_Analogue_Input_1	UINT16	Data: 0 ... 600 Resolution: 0.01 V Range: 0 ... 6.00 V
		0032	Actual_State_Analogue_Input_2	UINT16	
		0033	Actual_State_Analogue_Input_3	UINT16	
		0034	Actual_State_Analogue_Input_4	UINT16	
0000	0040		Actual_State_Keys	UINT32	Bit 0: K1 ... Bit 19: K20 Bit 20 ... Bit 31: not used (0 = open / 1 = closed)
0000	0060	0061	Number_cycle_of_on_of_Relay_in A1	UINT32	
		0062	Number_cycle_of_on_of_Relay_in A2	UINT32	
		0063	Number_cycle_of_on_of_Relay_in A3	UINT32	
		0064	Number_cycle_of_on_of_Relay_in A4	UINT32	
		0065	Number_cycle_of_on_of_Relay_in A5	UINT32	
		0066	Number_cycle_of_on_of_Relay_in A6	UINT32	
0000	0070	0071	Number_cycle_of_on_of_Relay_out A1	UINT32	
		0072	Number_cycle_of_on_of_Relay_out A2	UINT32	
		0073	Number_cycle_of_on_of_Relay_out A3	UINT32	
		0074	Number_cycle_of_on_of_Relay_out A4	UINT32	
		0075	Number_cycle_of_on_of_Relay_out A5	UINT32	
		0076	Number_cycle_of_on_of_Relay_out A6	UINT32	

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	0080	0081	Number_Actuator error A1	UINT32	Error counter: 2 B: linear actuator error 1 B: peak current reached 1 B: short detections
		0082	Number_Actuator error A2	UINT32	
		0083	Number_Actuator error A3	UINT32	
		0084	Number_Actuator error A4	UINT32	
		0085	Number_Actuator error A5	UINT32	
		0086	Number_Actuator error A6	UINT32	
		008F	Number_Total_Over_Current	UINT32	
0000	0090	0091	Cumulated_Stroke A1	UINT32	Unit: encoder edges
		0092	Cumulated_Stroke A2	UINT32	
		0093	Cumulated_Stroke A3	UINT32	
		0094	Cumulated_Stroke A4	UINT32	
		0095	Cumulated_Stroke A5	UINT32	
		0096	Cumulated_Stroke A6	UINT32	
0000	00A0	00A1	Current A1	UINT16	Data: 0 ... 1000 Unit: fixed-point 0.1 A Range: 0 ... 100 A
		00A2	Current A2	UINT16	
		00A3	Current A3	UINT16	
		00A4	Current A4	UINT16	
		00A5	Current A5	UINT16	
		00A6	Current A6	UINT16	
0000	00B0	00B1	Max_Current A1	UINT16	Data: 0 ... 1000 Unit: fixed-point 0.1 A Range: 0 ... 100 A
		00B2	Max_Current A2	UINT16	
		00B3	Max_Current A3	UINT16	
		00B4	Max_Current A4	UINT16	
		00B5	Max_Current A5	UINT16	
		00B6	Max_Current A6	UINT16	
		00BF	Max_Total_Current	UINT16	
		00C0	Max_Temp_Rectifer_FET	UINT8	Unit: ADC value 0 ... 255
		00C1	Number_Over_Temp_Rectifer_FET	UINT32	
0000	00D0	00D1	Error_Code 1 (last recent)	UINT32	see chapter Error code
		00D2	Error_Code 2 (History 1)	UINT32	
		00D3	Error_Code 3 (History 2)	UINT32	
		00D4	Error_Code 4 (History 3)	UINT32	
		00D5	Error_Code 5 (History 4)	UINT32	
0000	00E0	00E1	Actuator status 2 A1	UINT8	Bit 0: initialization (0 = not initialized / 1 = initialized) Bit 1: release flag for retraction (0 = no release / 1 = release) Bit 2: release flag for extension (0 = no release / 1 = release) Bit 3 ... Bit 7: not used
		00E2	Actuator status 2 A2	UINT8	
		00E3	Actuator status 2 A3	UINT8	
		00E4	Actuator status 2 A4	UINT8	
		00E5	Actuator status 2 A5	UINT8	
		00E6	Actuator status 2 A6	UINT8	
0000	00F0	00F1	Speed A1	UINT16	If speed select relative: Unit: % Range: 0 ... 100
		00F2	Speed A2	UINT16	
		00F3	Speed A3	UINT16	
		00F4	Speed A4	UINT16	
		00F5	Speed A5	UINT16	
		00F6	Speed A6	UINT16	

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	0100		Battery Mains	UINT8	Bit 0 (0/1: power supply not connected / connected) Bit 1 (0/1: battery not connected / connected) Bit 2 (0/1: charge controller off / on) Bit 3 (0/1: charging process inactive / active)
	0110		Binary Output Status	UINT8	Bit 0 (0/1: Binary Output 1 off/on) Bit 1 (0/1: Binary Output 1 off/on)
	0120		LED HS	UINT8	Bit 0 (0/1: LED1 hand switch off/on) Bit 1 (0/1: LED2 hand switch off/on)
	0130		LED LB	UINT8	Bit 0 (0/1: LED1 locking box off/on) Bit 1 (0/1: LED2 locking box off/on) ... Bit 7 (0/1: LED8 locking box off/on)
0000		0140	Buzzer	UINT8	Bit 0 (0/1: buzzer off/on)
0000		0150	Sensor Supply	UINT8	Bit 0 (0/1: sensor supply off/on)
0000		0162	Lock Status	UINT16	Bit 0 (0/1: **function 0** unlocked / locked) Bit 1 (0/1: **function 1** unlocked / locked) ... Bit 9 (0/1: **function 10** unlocked / locked)
0000		0164	Battery voltage	UINT16	Unit: fixed-point 0.1 V Range: 0 ... 40.0 V
0000		0165	Locking Box detected	UINT8	0 ... 2 locking box
		0166	User	UINT8	User 1 ... 4
0000	0170	0171	Actuator Status 1 A1	UINT8	Bit 0 (0/1: actuator not present / actuator present) Bit 1 (0/1: **signal limit_in_out** not active / active) Bit 2 (0/1: **signal switch 1** not active / active) Bit 3 (0/1: **signal switch 2** not active / active) Bit 4 (0/1: **motion** not active / active) Bit 5 (0/1: **in position** not reached / reached) Bit 6 (0/1: **out position** not reached / reached) Bit 7 (0/1: **stroke** not recorded / recorded)
		0172	Actuator Status 1 A2	UINT8	
		0173	Actuator Status 1 A3	UINT8	
		0174	Actuator Status 1 A4	UINT8	
		0175	Actuator Status 1 A5	UINT8	
		0176	Actuator Status 1 A6	UINT8	
1000	1010	1011 ... 1016	Conversion factor A1 ... A6	FLOAT	–
2000		2001	UserPositionData A1	STRUCT	Structure definitions: ACTUATOR_POSITIONS
		2002	UserPositionData A2	STRUCT	
		2003	UserPositionData A3	STRUCT	
		2004	UserPositionData A4	STRUCT	
		2005	UserPositionData A5	STRUCT	
		2006	UserPositionData A6	STRUCT	

Remote data elements: stored in volatile registers. Initialized after reset with preset data.

7 Data list: remote data elements 3000

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
3000		3001	CyclicObj 1	UINT16[12]	The CyclicObj definition specifies the data transferred to and from the control unit with each RC command. The data indices set in the first 6 B (para[0 ... 5]) define the data sent to the control unit (write data) and the data indices set in the last 6 B (para[6 ... 11]) define the data returned from the control unit (read data). A data index of −1 means: no data transfer. All data can be read from the control unit, but only data with indices 3xxx can be written. Default value: −1
		3002	CyclicObj 2	UINT16[12]	
		3003	CyclicObj 3	UINT16[12]	
		3004	CyclicObj 4	UINT16[12]	
		3005	CyclicObj 5	UINT16[12]	
	3010	3011 ... 301A	Remote Speed F1 ... F10	UINT16	Default value: function speed from configuration. If speed select relative: Unit: % Range: 0 ... 100 If speed select absolute: Unit: encoder edges/ s Range: 0 ... 1000
	3020	3021 ... 3026	Remote Position A1 ... A6	INT32	Default value: Memory 1 / User 1 position of UserPositionData (DynamicConfiguration) Unit: encoder edges

4.4 Function list

Set the unused parameters in the telegram structure to −1 (unused_para).

Functions F1 to F10 are defined in the control parameterization. A function can be assigned 1 to 6 actuators. If more than 1 actuator is assigned to a function, the actuators can be coordinated with each other:

- simultaneous running in the same direction or in opposite directions:
  - simultaneous start and stop, but no position synchronization
- synchronized simultaneous running in the same direction or in opposite directions:
  - controlled position synchronization
  - can also be parameterized with a constant offset between the actuators

8 Function list

func-ID	Data (dec.)	Used by command	Description	para_ID[x]
F1 ... F10	0 ... 9	RE, RS	Motion Function (depending on parameters)	para_ID[0] para_ID[1] = –1
F11	10	RE, RS	Buzzer (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1
F17	16	RE, RS	Binary Output 1 (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1
F18	17	RE, RS	Binary Output 2 (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1
F20	19	RE, RS	Emergency stop (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1
F21	20	RE, RS	Operating unit LED1 (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1
F22	21	RE, RS	Operating unit LED1 (from firmware version V2B1)	para_ID[0] = –1 para_ID[1] = –1

9 Parameter dependent on function

Used for func_ID	para_ID[1]	Data (dec.)	Description
F1 ... F10 (only with command RE)	motion_direction	0 ... 9	0: Undefined direction (no motion) 1: Move to position In 2: Move to position Out 3: Move to position Mem1 4: Move to position Mem2 5: Move to position Mem3 6: Move to position Mem4 7: Move to position Intermediate In 8: Move to position Intermediate Out 9: Move to Remote Position
F1 ... F10 (only with command RE) motion_ direction	motion_stop	0 ... 1	0: Fast start/stop (start/stop ramp not considered) 1: Soft start/stop (start/stop ramp considered)

4.5 Error codes

10 Error codes

Bit	Cause	Condition	Reaction
Bit 1	• CRC error during ROM test • Faulty ROM	–	• Motion is stopped • Control performs a reset
Bit 2	• RAM test failed • Faulty RAM	–	• Motion is stopped • Control performs a reset
Bit 3	• CPU test failed • Faulty CPU	–	• Motion is stopped • Control performs a reset
Bit 4	• STACK overrun detected	–	• Motion is stopped (fast stop) • Control performs a reset

Bit	Cause	Condition	Reaction
Bit 5	• Program sequence error • Watchdog reset	–	• Motion is stopped (fast stop) • Control performs a reset
Bit 6	• Hand switch test failed • Short in hand switch detected	Only if hand switch is parameterized as safe	• Motion is stopped (fast stop)
Bit 7	• Binary input test failed • Short between binary inputs detected	Only if binary inputs are parameterized as safe and no analogue input is parameterized	• Motion is stopped (fast stop)
Bit 8	• Relay and FET test failed • Faulty relay or FET	Test is performed at start of motion	• Motion is not executed
Bit 9	–	–	–
Bit 10	• Communication with MoveEnable controller failed	No response from MoveEnable controller	• Motion is stopped (fast stop)
Bit 11	• MoveEnable output check failed • Output from the MoveEnable controller is incorrect	–	• Motion is stopped (fast stop)
Bit 12	• Overtemperature at rectifier or FET detected	–	• Motion is stopped (fast stop)
Bit 13	• Shutdown due to deep discharge of battery	–	• Motion is stopped (fast stop) • Control unit switches off
Bit 14	• Total current exceeded	When in motion	• Motion is stopped (fast stop) • Bit is reset with next motion
Bit 15 ... Bit 20	• Actuator fault 1 ... 6	• Peak current • Short circuit current • Sensor monitor • Over current (if not limit position) • Time out (if not limit position)	• Actuator is stopped (fast stop) • Bit is reset with next motion
Bit 21	• Position difference between actuators too great (synchronized parallel run)	Only if synchronized parallel run is parameterized	• Motion is not started or stopped (fast stop) • Bit is reset with next motion
Bit 22	• Remote communication time out	–	Depending on the Safety_ID
Bit 23		–	
Bit 24	• Locking box I^2C communication error	Only if locking box is parameterized as safe	• Motion is not executed or is stopped
Bit 25	• RAM copy of EEPROM configuration data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 26	• RAM copy of EEPROM user data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 27	• EEPROM locking box data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 28	• RAM copy of EEPROM dynamic data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 29	• RAM copy of EEPROM calibration data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 30	• RAM copy of EEPROM calibration data has an invalid CRC	–	• Motion is not executed or is stopped
Bit 31	• IO test	Executed when no motion is active	• Motion is not executed
Bit 32	• Motions are not executed.	–	• Motion is not executed or is stopped

4.6 Controlling the actuators

The individual actuators are controlled via functions F1 to F10. A function is activated using the RE command, which starts one or more actuators. Each RE command must be stopped with RS, even if the actuator stops automatically upon reaching the end position.

Function definition

The function definition can be found in the documentation for control unit parameterization.

Setting motion parameters

The motion parameters of speed and target position can be set using data indices 3011 to 301A and 3021 to 3026. The speed applies to the selected function, while the target position relates to the individual actuators. The RE command with parameter 9 starts the motion.

Speed is given as a percentage (0 % to 100 %) or in increments. This depends on the parameterization of the control unit. For control units with standard parameterization, the speed is set in %.

The lower threshold at which an actuator is set in motion depends on the actuator type and load. The speed can be changed during motion. The control unit adjusts the speed while maintaining the soft start ramp.

Reading information

Operating states and information can be read from the control via the RG command. Data can be queried individually or in blocks.

Data indices 0011 to 0016 return the current position. Grouping index 0010 returns the position of all 6 possible actuators. The position in mm can be calculated from the end position data and the stroke length.

5 Communication examples

Move to position and read

With parameterization SCP11, all actuators are set for individual operation. Each actuator is assigned to a function:

- actuator 1 to function 1
- actuator 2 to function 2
- actuator n to function n

This allows the actuators to be controlled individually using functions 1 to 6.

During the entire routine, communication must take place in a repeated cycle at least every 500 ms using the RC command. Communication via RC works as a watchdog.

If RC communication fails, the control unit stops all actuators currently in motion and deactivates remote mode. Before the first command is sent to the control unit, communication via RC must already have taken place.

Routine:

1. open secure communication mode with RO (Safety_ID)
2. activate remote mode
3. set remote position of actuator 1
4. start motion of actuator 1
5. read status of actuator 1 and check whether motion is active
6. read back current position of actuator 1
7. close communication mode using RA

Periodic communication via RC without data transfer (without CyclObj)

11 Set parameters

Cmd	Name	Request parameters						Reply parameters			
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4
RO	Remote Mode open	00	–	–	–	–	–	ACK, **E	–	–	–

Activate remote mode

12 Set parameters

Cmd	Name	Request parameters						Reply parameters						
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	P5	P6	P7
RC	Remote cyclic	01	00	–1	–	–	–	ACK, **E	–	–	–	–	–	–

Set remote speed of actuator 1 to 100 h using the RT command (data index 3011).

13 Set parameters

Cmd	Name	Request parameters						Reply parameters				
		P1	P2	P3	P4	P5	P6	P1	P2	P3	P4	Pn
RT	Remote data transfer	04	00	11	30	01	00	ACK, **E	–	–	–	–

Movement to remote position with actuator 1 without start/stop. Start using the RE command (function index 0)

14 Set parameters

Cmd	Name	Request parameters						Reply parameters				
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	Pn
RE	RE Remote execute function	00	09	–1	–	–	–	ACK, **E	–	–	–	–

Query status of actuator 1 using the RG command (data index 0171). Bit 4 in the status is set as long as motion is active.

15 Set parameters

Cmd	Name	Request parameters						Reply parameters						
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	P5	P6	P7
RG	Remote data get	71	01	–	–	–	–	ACK, **E	01	00	Status	–	–	–

Query current position of actuator 1 using the RG command (data index 0011)

16 Set parameters

Cmd	Name	Request parameters						Reply parameters						
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	P5	P6	P7
RG	Remote data get	11	00	–	–	–	–	ACK, **E	04	00	1st byte	2nd byte	3rd byte	4th byte

Close communication mode.

17 Set parameters

Cmd	Name	Request parameters						Reply parameters			
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4
RA	Remote Mode abort	–	–	–	–	–	–	ACK, **E	–	–	–

6 Code examples

Checksum calculation

The checksum is calculated using the RC16 algorithm standardized according to CCITT. The polynomial is CRC16 = $x^{16} + x^{12} + x^5 + 1$, the starting value is 0.

The calculation of the CRC checksum makes heavy use of the processor. To reduce this effort, a CRC table should ideally be used.

5 Code example: CRC table

```
static const unsigned short CRC _ TABLE[256] = {
0x0000 0x1021 0x2042 0x3063 0x4084 0x50A5 0x60C6 0x70E7
0x8108 0x9129 0xA14A 0xB16B 0xC18C 0xD1AD 0xE1CE 0xF1EF
0x1231 0x0210 0x3273 0x2252 0x52B5 0x4294 0x72F7 0x62D6
0x9339 0x8318 0xB37B 0xA35A 0xD3BD 0xC39C 0xF3FF 0xE3DE
0x2462 0x3443 0x0420 0x1401 0x64E6 0x74C7 0x44A4 0x5485
0xA56A 0xB54B 0x8528 0x9509 0xE5EE 0xF5CF 0xC5AC 0xD58D
0x3653 0x2672 0x1611 0x0630 0x76D7 0x66F6 0x5695 0x46B4
0xB75B 0xA77A 0x9719 0x8738 0xF7DF 0xE7FE 0xD79D 0xC7BC
0x48C4 0x58E5 0x6886 0x78A7 0x0840 0x1861 0x2802 0x3823
0xC9CC 0xD9ED 0xE98E 0xF9AF 0x8948 0x9969 0xA90A 0xB92B
0x5AF5 0x4AD4 0x7AB7 0x6A96 0x1A71 0x0A50 0x3A33 0x2A12
0xDBFD 0xCBDC 0xFBBF 0xEB9E 0x9B79 0x8B58 0xBB3B 0xAB1A
0x6CA6 0x7C87 0x4CE4 0x5CC5 0x2C22 0x3C03 0x0C60 0x1C41
0xEDAE 0xFD8F 0xCDEC 0xDDCD 0xAD2A 0xBD0B 0x8D68 0x9D49
0x7E97 0x6EB6 0x5ED5 0x4EF4 0x3E13 0x2E32 0x1E51 0x0E70
0xFF9F 0xEFBE 0xDFDD 0xCFFC 0xBF1B 0xAF3A 0x9F59 0x8F78
0x9188 0x81A9 0xB1CA 0xA1EB 0xD10C 0xC12D 0xF14E 0xE16F
0x1080 0x00A1 0x30C2 0x20E3 0x5004 0x4025 0x7046 0x6067
0x83B9 0x9398 0xA3FB 0xB3DA 0xC33D 0xD31C 0xE37F 0xF35E
0x02B1 0x1290 0x22F3 0x32D2 0x4235 0x5214 0x6277 0x7256
0xB5EA 0xA5CB 0x95A8 0x8589 0xF56E 0xE54F 0xD52C 0xC50D
0x34E2 0x24C3 0x14A0 0x0481 0x7466 0x6447 0x5424 0x4405
0xA7DB 0xB7FA 0x8799 0x97B8 0xE75F 0xF77E 0xC71D 0xD73C
0x26D3 0x36F2 0x0691 0x16B0 0x6657 0x7676 0x4615 0x5634
0xD94C 0xC96D 0xF90E 0xE92F 0x99C8 0x89E9 0xB98A 0xA9AB
0x5844 0x4865 0x7806 0x6827 0x18C0 0x08E1 0x3882 0x28A3
0xCB7D 0xDB5C 0xEB3F 0xFB1E 0x8BF9 0x9BD8 0xABBB 0xBB9A
0x4A75 0x5A54 0x6A37 0x7A16 0x0AF1 0x1AD0 0x2AB3 0x3A92
0xFD2E 0xED0F 0xDD6C 0xCD4D 0xBDAA 0xAD8B 0x9DE8 0x8DC9
0x7C26 0x6C07 0x5C64 0x4C45 0x3CA2 0x2C83 0x1CE0 0x0CC1
0xEF1F 0xFF3E 0xCF5D 0xDF7C 0xAF9B 0xBFBA 0x8FD9 0x9FF8
0x6E17 0x7E36 0x4E55 0x5E74 0x2E93 0x3EB2 0x0ED1 0x1EF0
};
```

The following code is an example for calculating the CRC checksum using the CRC table. The returned 2 B (bytes) must be appended to the command.

6 Code example: CRC checksum determination

```
unsigned short CalculateChecksum (const unsigned char* pAdr, int len)
{
if (len < 0)
{
ASSERT(FALSE);
return 0;
}
unsigned short crc = 0;
while (len--)
{
crc = static _ cast<unsigned short>(CRC _ TABLE[((crc >> 8) ^ *pAdr++) & 0xFF] ^ (crc << 8));
}
return crc;
}
```

7 Code example: Checksum result verification

```
bool CheckResponseChecksum(const CArray<unsigned char>& responseData, bool
suppressTimeoutError)
{
CArray<unsigned char> tempData;
unsigned char crcByte1;
unsigned char crcByte2;
DWORD bytesRead;
tempData.Append(responseData);
if (!ReadFile(m _ hComm, &crcByte1, 1, &bytesRead, NULL) || (bytesRead !=1))
{
if(!GetLastError()) {
// case time out
if(!suppressTimeoutError)
AfxMessageBox(IDS _ READ _ ERROR _ CRC);
}
else {
Disconnect();
AfxMessageBox(IDS _ READ _ ERROR);
}
}
if (!ReadFile(m _ hComm, &crcByte2, 1, &bytesRead, NULL) || (bytesRead !=1))
{
if(!GetLastError()) {
// case time out
if(!suppressTimeoutError)
AfxMessageBox(IDS _ READ _ ERROR _ CRC);
}
else {
Disconnect();
AfxMessageBox(IDS _ READ _ ERROR);
}
}
tempData.Add(crcByte2);
tempData.Add(crcByte1);
if (CalculateChecksum(tempData.GetData(), static _ cast<int>(tempData.Get-Size())) != 0)
{
AfxMessageBox(IDS _ READ _ ERROR _ CRC _ INVALID);
return false;
}
else
{
return true;
}
}
```

7 Structure definitions

8 Structure definitions

```
struct ACTUATOR _ POSITIONSstruct {
INT32 Position _ Memory _ 1[USER _ 1;
INT32 Position _ Memory _ 2[USER _ 1];
INT32 Position _ Memory _ 3[USER _ 1];
INT32 Position _ Memory _ 4[USER _ 1];
INT32 Position _ Intermediate _ In[USER _ 1];
INT32 Position _ Intermediate _ Out[USER _ 1];
INT32 Position _ Memory _ 1[USER _ 2;
INT32 Position _ Memory _ 2[USER _ 2];
INT32 Position _ Memory _ 3[USER _ 2];
INT32 Position _ Memory _ 4[USER _ 2];
INT32 Position _ Intermediate _ In[USER _ 2];
INT32 Position _ Intermediate _ Out[USER _ 2];
INT32 Position _ Memory _ 1[USER _ 3;
INT32 Position _ Memory _ 2[USER _ 3];
INT32 Position _ Memory _ 3[USER _ 3];
INT32 Position _ Memory _ 4[USER _ 3];
INT32 Position _ Intermediate _ In[USER _ 3];
INT32 Position _ Intermediate _ Out[USER _ 3];
INT32 Position _ Memory _ 1[USER _ 4;
INT32 Position _ Memory _ 2[USER _ 4];
INT32 Position _ Memory _ 3[USER _ 4];
INT32 Position _ Memory _ 4[USER _ 4];
INT32 Position _ Intermediate _ In[USER _ 4];
INT32 Position _ Intermediate _ Out[USER _ 4];
INT32 Position _ Virtual _ Limit _ In;
INT32 Position _ Virtual _ Limit _ Out;
};
ACTUATOR _ POSITIONS positions[ACTUATOR _
COUNT];
```

Schaeffler Technologies AG & Co. KG
Georg-Schäfer-Straße 30
97421 Schweinfurt
Germany
www.schaeffler.de/en
info.de@schaeffler.com

In Germany:
Phone 0180 5003872
From other countries:
Phone +49 9721 91-0

BA 140-02 / 01 / en-US / 2026-05