Read this manual before installing, operating or maintaining this actuator. Failure to follow safety precautions and instructions could cause actuator failure and result in serious injury, death or property damage.

SKF is a registered trademark of the SKF Group.

© SKF 2010
The contents of this publication are the copyright of the publisher and may not be reproduced (even extracts) unless prior written permission is granted. Every care has been taken to ensure the accuracy of the information contained in this publication but no liability can be accepted for any loss or damage whether direct, indirect or consequential arising out of the use
of the information contained herein.

L5318,2971E . June 2010

Table of Contents

1. Introduction

This chapter contains information regarding the structure and the organization of the operation manual which simplifies use of the operation manual and makes it possible to obtain rapid access to desired information.

Content

This operation manual contains a description of the RS232 serial interface of the SCU control unit. Please note that the RS232 interface is an option with the SCU control unit and must be ordered on the basis of the type key.

Validity scope

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

- Manufacturer: SKF Actuation System (Liestal) AG
- Product name: SCU control unit with serial RS232 interface
- Type designation: SCUxx-xxxxx1-xxxx
- Year of manufacture: after 2007 with Firmwave version V2B0
- CE identification: in accordance with technical documentation

Target group

This manual is intended for development engineers who have the necessary professional knowledge to be able to develop control software for the operation of this product.

Presentation conventions

In this operation manual we employ certain abbreviations and markings to identify text sections or advice.

Safety advice

WARNING: Safety advice to notify of danger of irreparable damage to equipment and persons based on hazard analyses. This includes advice as regards protective measures and any required special training and personal protective gear.

Such advice is indicated as follows:

WARNING

The hazard source is indicated.

Description of possible consequences!

- **Measures that can be taken to prevent the hazard.**

CAUTION: Safety advice regarding remaining hazards that may still be present due to inadequate functioning of protective measures against damage to equipment and persons. Advice regarding any required special training and personal protective gear.

Such advice is indicated as follows:

CAUTION

The hazard source is indicated.

Description of possible consequences!

- **Measures that can be taken to prevent the hazard.**

Other advice
Advice regarding important and/or useful additional information to be taken into consideration during maintenance work.

Such advice is indicated as follows:

Advice: Advice text is identified.

Code examples

The code examples given in the manual are in C++ and serve as clarification.
The code examples are set off using normal software formatting:

```
unsigned short HelloWorld()

{

//@todo

}
```

Cross-references

Cross-references to sections in other areas of the operation manual are bracketed. They contain the corresponding header text and page number.

Cross-references are indicated as follows:

(→ Cross-references, page 6)

Referencing of diagram details

Details in diagrams are sequentially lettered clockwise and correspondingly referenced in the text.

2. Safety

Safety advice in this manual is differentiated according to applicability as follows.

- General safety advice
  Such safety advice applies in general and is to be taken into consideration on replacement of any assembly group. They are given in the section General Safety Advice .
- Special safety advice
  Such safety advice is only relevant for some assembly groups. This type of advice is found in the replacement description for the assembly group concerned.

General safety advice

With maintenance work please take the following safety advice into consideration:

WARNING

Maintenance work with live units.

Electrical shock!

- **Switch off the unit prior to carrying out any maintenance work and take out the mains plug.**

WARNING

Squashing of or damage to cables.

Electrical shock!

- **Please pay attention to correct cable strain relief and cable routing on installing assembly groups.**

CAUTION

Unintentional movement of work bench.

Damage to exposed device parts!

- **Prior to starting maintenance work set all locking brakes.**

CAUTION

Use of unsuitable tools or materials.

Damage / defective operation of the device!

- **Please only use original parts and the specified special tool.**

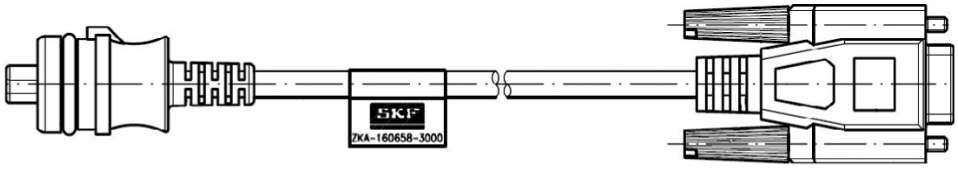
3. Technical overview

The basic technical characteristics of the serial interface are given in this chapter.

Note:
If the remote user does not provide a mains supply according to medical standards (safety according to ENE60601-1) the final application has to be grounded to ensure a correct operation of RS 232 interface.

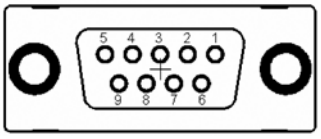
Connection cable

Recommended connection cable: ZKA-160658-3000



Physical layer

- Electrical characteristics in accordance with RS232 definition
- Half duplex
- Bi-directional
- Baud rate: With standard control units the baud rate is set to 38400.
  With customized control units the baud rate may be set to the following values:
  9600, 19200, 38400,
- Plug: 9-pole SUB-D (female)
- The control lines are not used. However, DTR and RTS must be switched on as permanently active because they supply the RS232 converter in the control unit. Instead of the DTR and RTS signals a separate power source of 5.5...15 VDC/30mA can be connected (+ on pin 4 DTR or pin 7 RTS and – on pin 5 GND)
- Connection allocation:



2. RxD
3. TxD
4. DTR
5. GND
7. RTS

Data link layer

- One start bit
- 8 data bits (LSB first)
- One stop bit
- No parity bit
- No handshake

Network Layer

- Point to point connection (only two participants)
- The control unit functions as slave and replies to the requests of the master (e.g. PC program)
- The slave replies to each request from the master
- Maximum request delay: 2000 ms
- Maximum delay between individual telegram bytes: 1000 ms
- When the control is operated with batteries and the parameterization is set to <Low Power> = Enabled and the controller is set at low power mode, the controller can be set to remote mode in holding the circuit RXD during min. 100 ms at status “space” (Level > +3 V) (from FW V2B1).
- When the control is operated with batteries and the parameterization is set to <Low Power> = Enabled and the remote mode is activated, but there is no command, the controller does not go to low power mode. If the communication is interrupted at this status the controller sets to low power mode (from FW V2B1).

Transport Layer

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 Kommando Nummer (1Byte)
<Y> :	Minor Kommando Nummer (1Byte)
<P1>...<Pn> :	Parameter Bytes (Intel Little Endian Format, LS Byte... MS Byte)
<C1> :	Low Byte der 16 Bit Telegramm Checksumme
<C2> :	High Byte of 16 Bit Telegramm Checksumme

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 start value is 0.

Each reply includes an ACK byte, which contains the device status. Many replies contain the parameter ctp in P1/P2. This defines the number of the following data bytes. Each reply that contains more than 1 data byte uses a ctp for the definition of data length.

A telegram can be described as follows:
Request: <X><Y>[<P1><P2>...<Pn>]<C1><C2>
Reply: <X><Y><ACK>[<<ctp1><ctp2>=n><P3><P4>...<Pn+2>]<C1><C2>

4. Communication protocol

Command set

The following commands are available after mains on or in battery operation:

- The remote function is activated with the RO command.
- The remote function is deactivated with the RA command.
- 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 RG, RT, RC, RE and RS commands are only available if the remote function is activated.

The control lines DTR and RTS must be permanently switched on so that the RS232 converter is supplied and communication with the SCU is possible.

Cmd <X>Y>	Name	Query parameter						Reply parameter					Description
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	Pn	
RG	Remote data get	data_ID [0]	data_ID [1]	–	–	–		ACK, **E	ctp1	ctp2	1. Data byte	n-2. Data byte	Data transfer from SCU P2 to Pn as reply only if P1 = ACK
RT	Remote data transfer	ctp1	ctp2	data_ID [0]	data_ID [1]	1. Data byte	n-4. Data byte	ACK, **E	–	–	–	–	Data transfer to SCU. Only data_Ids 3xxx are permitted (RemoteData)!
RC	Remote cyclic	ctp1	ctp2	Index of cyclicObj	1. Byte write data of cyclicObj	2. Byte write data of cyclicObj	n-3. Byte write data of cyclicObj	ACK, **E	ctp1	ctp2	1. Byte read data of cyclicObj	n-3. Byte read data of cyclicObj	Must be sent at least every 500 ms so that the SCU remains in Remote-Mode (WDT). With P3 = -1 no data are transferred, otherwise P3 is the index for cyclicObj, which defines the query/reply data P3=0: cyclicObj with dataID=3001 P2 to Pn is in the reply only if P1 = ACK
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. P2/3 are additional function parameters, note. Note the definitions of fnc_ID and para_ID.
RS	Remote stop function	fnc_ID	para_ID [0]	–	–	–	–	ACK, **E	–	–	–	–	Stops a function. P1 is the index in the function list. P2/3 are additional function parameters, note. 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 timeout (0.5 s) occurs, all movements are terminated, that is, no movement will be started. If a communication timeout occurs, only the RC, RA and RO commands are available. Safety_ID = 1: If a communication timeout (0.5 s) occurs, all movements are terminated. Safety ID = 2: If a communication timeout (0,5 s) occurs, only remote motion is terminated. (for FWV2B3)
RA	Remote mode abort	–	–	–	–	–	–	ACK, **E	–	–	–	–	Set the SCU is in normal mode (without reset).

Communication error and acknowledge codes

Code	Hex	Dec	Name	Description
ACK	06	6	Command acknowledged	Query 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 possible with current SCU mode/state

Abbreviations used

Code	Value	Description
ctp	Dyn	Number of following telegram bytes
data_ID	Dyn	Index in data list (see data list table)
fnc_id	Dyn	Index for function list (see function list table)
para_id	Dyn	Additional parameters depend on function (see function list table)

Data list

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

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000		0001	Firmware Info Name Version CS	STRING	size = 31 Byte
		0002	Configuration info Name Version CS	STRING	size = 36 Byte

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	0010	0011	Actual_Position Actuator 1	INT32	Unit: Encoder flank (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	
	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)
	0030	0031	Actual_State_Analogue_Input_1	UINT16	Data: 0...600 Resolution 0.01V Range: 0...6.00V
		0032	Actual_State_Analogue_Input_2	UINT16	
		0033	Actual_State_Analogue_Input_3	UINT16	
		0034	Actual_State_Analogue_Input_4	UINT16	
	0040		Actual_State_Keys	UINT32	Bit 0: K1 ... Bit 19: K20 Bit 20 ... Bit 31 not used (0 = open / 1 = closed)
	0060	0061	Number_cycle_off_on_off_Relay_in A1	UINT32	
		0062	Number_cycle_off_on_off_Relay_in A2	UINT32	
		0063	Number_cycle_off_on_off_Relay_in A3	UINT32	
		0064	Number_cycle_off_on_off_Relay_in A4	UINT32	
		0065	Number_cycle_off_on_off_Relay_in A5	UINT32	
		0066	Number_cycle_off_on_off_Relay_in	UINT32	

Communication protocol

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	0070	0071	Number_cycle_off_on_off_Relay_out A1	UINT32	
		0072	Number_cycle_off_on_off_Relay_out A2	UINT32	
		0073	Number_cycle_off_on_off_Relay_out A3	UINT32	
		0074	Number_cycle_off_on_off_Relay_out A4	UINT32	
		0075	Number_cycle_off_on_off_Relay_out A5	UINT32	
		0076	Number_cycle_off_on_off_Relay_out A6		
	0080	0081	Number_Actuator error A1	UINT32	count 2 byte: number of actuator error 1 byte: number of peak current occurrence 1 byte: number of short circuit occurence
		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	
	0090	0091	Cumulated_Stroke A1	UINT32	Unit: Encoder flank
		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	
	00A0	00A1	Current A1	UINT16	Data: 0...1000 Unit: fixed-point 0.1A Range: 0..100A
		00A2	Current A2	UINT16	
		00A3	Current A3	UINT16	
		00A4	Current A4	UINT16	
		00A5	Current A5	UINT16	
		00A6	Current A6	UINT16	

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	00B0	00B1	Max_Current A1	UINT16	Data: 0...1000 Unit: fixed-point 0.1A Range: 0..100A
		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_Rectifier_FET	UINT8	Unit: ADC value. 0...255
		00C1	Number_Over_Temp_Rectifier_FET	UINT32	
	00D0	00D1	Error_Code 1 (last recent)	UINT32	For structure see chapter SCU 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 5)	UINT32	
	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: collision on manual level Bit 4: collision on remote level Bit 5: emergency stop Bit 6 and 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	
	00F0	00F1	Speed A1	UINT16	If speed select relative: Unit: % Range: 0..100 If speed select absolute: Unit: Encoder flank/ s Range: 0..1000
		00F2	Speed A2	UINT16	
		00F3	Speed A3	UINT16	
		00F4	Speed A4	UINT16	
		00F5	Speed A5	UINT16	
		00F6	Speed A6	UINT16	
	0100		Battery Mains	UINT8	Bit 0 0/1: Mains not connected/ connected Bit 1 0/1: Battery disconnected /connected Bit 2 0/1: Charging control on/off 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

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
0000	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
		0140	Buzzer	UINT8	Bit 0 0/1: Buzzer off/on
		0150	Sensor Supply	UINT8	Bit 0 0/1: Sensor Supply off/on
		0162	Lock Status	UINT16	Bit 0 0/1: Funktion 0 unlocked/ locked Bit 1 0/1: Funktion 1 unlocked/ locked ... Bit 9 0/1: Funktion 10 unlocked/ locked
		0164	Battery voltage	UINT16	Unit: Fixed-point 0,1V Range: 0... 40,0 V
		0165	Locking Box detected	UINT8	0..2 locking box
		0166	User	UINT8	User 1..4
	0170	0171	Actuator Status 1 A 1	UINT8	Bit 0 0/1 drive unavailable/ drive available Bit 1 0/1: signal limit_in_out inactive/aktive Bit 2 0/1: signal switch 1 inactive/active Bit 3 0/1: signal switch 2 inactive/active Bit 4 0/1: motion inactive/active Bit 5 0/1: in position not reached/reached Bit 6 0/1: out position Bit 7 0/1 Stroke not done/done Bit 7 0/1 Stroke not done/done
		0172	Actuator Status 1 A 2	UINT8	
		0173	Actuator Status 1 A 3	UINT8	
		0174	Actuator Status 1 A 4	UINT8	
		0175	Actuator Status 1 A 5	UINT8	
		0176	Actuator Status 1 A 6	UINT8	
1000	1010	1011-1016	Conversion factor A 1-6	FLOAT	
2000	2001		UserPositionData A 1	STRUCT	Structure definition according to chapter → 7. Structure definitions, page 28: ACTUATOR_POSITIONS
	2002		UserPositionData A 2	STRUCT	
	2003		UserPositionData A 3	STRUCT	
	2004		UserPositionData A 4	STRUCT	
	2005		UserPositionData A 5	STRUCT	
	2006		UserPositionData A 6	STRUCT	

Primary collection index	Secondary collection index	Data index	Name	Data type	Comment
Remote data items Stored in volatile register. Initialized after reset with preset values.					
3000		3001	CyclicObj 1	UINT16[12]	With the CyclicObj definition the data transferred to and from the SCU with each RC command can be determined. The data indices set in the first 6 bytes (para[0..5] define the data to be sent to the SCU, (write data) and the data indices set in the last 6 bytes (para[6..11]) define the data that will be returned by the SCU (read data). A data index of –1 means no data transfer. All data can be read by the SCU, but only data with the 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 F1-10	UINT16	Default value: function speed from configuration. If speed select relative: Unit: % Range: 0..100 If speed select absolute: Unit: Encoder flank/ s Range: 0..1000
	3020	3021-3026	Remote Position A1-6	INT32	Default value: memory 1 / user 1 position of UserPositionData (DynamicConfiguration) Unit: Encoder flank

Function list

Func-ID	Value (dez.)	Used by command	Description	Para_ID[x]
F1...F10	0...9	RE, RS	Motion function (depends on parameterization)	Para_ID[0] according to Tab 4-1 Para_ID[1] = -1
F11	10	RE, RS	Buzzer (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1
F17	16	RE, RS	Binary Output 1 (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1
F18	17	RE, RS	Binary Output 2 (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1
F20	19	RE, RS	Emergency stop (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1
F21	20	RE, RS	Operating unit Led1 (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1
F22	21	RE, RS	Operating unit Led2 (from FW V2B1)	Para_ID[0] = -1 Para_ID[1] = -1

Tab. 4-1 Parameter depends on function:

Used for func_ID	Para_ID[1]	Value (dez.)	Description
F1-F10 (only with RE command)	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 RS command)	motion_stop	0-1	0: Fast Start/stop (start/stop ramp not considered) 1: Soft Start/stop (start/stop ramp considered)

The unused parameters in the telegram structure are to be set to –1 (unused_para).
The function of F1 – F10 is established in the control unit parameterization. A function can be assigned from one to six drives. If more than one drive is assigned to a function the drives may be coordinated among themselves:

- Simultaneous running in the same or opposite direction (simultaneous starting / stopping, but no position synchronization)
- Synchronized simultaneous running in the same direction or in the opposite direction (controlled position synchronization)

The second case can also be parameterized with a constant difference between the drives.

SCU Error code

Bit in error field	Cause	Condition for appearance	Reaction
Bit 1	CRC error with ROM test. Faulty ROM	–	Motions are stopped and the control unit carries out a reset.
Bit 2	Error with RAM test. Faulty RAM.	–	Motions are stopped and the control unit carries out a reset.
Bit 3	Error with CPU test. Faulty CPU.	–	Motions are stopped and the control unit carries out a reset.
Bit 4	STACK overrun detected.	–	Motions are stopped (fast stop) and the control unit carries out a reset..
Bit 5	Programm sequence error. Watchdog reset.	–	Motions are stopped (fast stop) and the control unit carries out a reset..
Bit 6	Error with hand switch test. Short detected in hand switch.	Only if hand switch is parameterized as “safe”	Motions are stopped (fast stop)
Bit 7	Error with binary inputs. Short detected between binary inputs	Only if binary inputs are parameterized as safe and no analogue input is parameterized.	Motions are stopped (fast stop)
Bit 8	Error with relay and FET tests. Faulty relay or FET.	Test performed at start of motion.	Motion not executed.
Bit 9	–	–	–
Bit 10	Error with communication with MoveEnable controller. No reply from MoveEnable controller.	–	Motions stopped (fast stop).
Bit 11	Error with MoveEnable output test. The MoveEnable controller output is incorrect.	–	Motions stopped (fast stop).
Bit 12	Overtemperature detected at rectifier or FET.	–	Motions stopped (fast stop).
Bit 13	Switching off due to excessive discharge of battery	–	Motions stopped. (Fast stop). Control unit switches itself off.
Bit 14	Total current is exceeded	If motion in process.	Motions stopped (fast stop). Bit reset in the next motion.

Bit in error field	Cause	Condition for appearance	Reaction
Bit 15: Drive 1 Bit 16: Drive 2 Bit 17: Drive 3 Bit 18: Drive 4 Bit 19: Drive 5 Bit 20: Drive 6	Error with drive	Peak current Short circuit current Sensor monitor Over current (if not limit position) Time out (if not limit position)	Drive stopped (fast stop). Bit reset on next motion.
Bit 21	Position difference between drives too great (synchronized parallel run)	Only if synchronized parallel run is parameterized.	Motion not started or if motion in progress the motion is stopped (fast stop). Bit reset on 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 safe parameterized	Motions not performed or stopped
Bit 25	RAM copy of EEPROM configuration data indicates incorrect CRC on	–	Motions not performed or stopped
Bit 26	RAM copy of EEPROM user data indicates incorrect CRC on	–	Motions not performed or stopped
Bit 27	EEPROM locking box data indicates incorrect CRC on	–	Motions not performed or stopped
Bit 28	RAM copy of EEPROM dynamic data indicate incorrect CRC on	–	Motions not performed or stopped
Bit 29	RAM copy of EEPROM calibration data indicate incorrect CRC on	–	Motions not performed or stopped
Bit 30	RAM copy of EEPROM HW settings indicates incorrect CRC on	–	Motions not performed or stopped
Bit 31	IO Test	Is performed if no motion is active.	Motions not performed
Bit 32	IDF operating system error	–	Motions not performed or stopped

Control of drives

Control of individual drives occurs via functions F1-F10. A function is activated via the RE command and thus one or more drives started. Each RE command must be stopped with an RS, even if the drive is stopped after reaching the end position.

Function definition

Please obtain the function definitions from the parameterization dokumentation for the control unit.

Setting of motion parameters

The motion parameters of speed and target position can be set via the indices 3011 to 301A or 3021 to 3026. The speed applies to the selected function, the target position is connected with individual drives. Motion is started with the RE command and parameter 9.

Speed is to be given in percentages (0-100%) or increments. This depends on the parameterization of the control unit. For standard control units the speed is set in percentages. The lower threshold on which a drive is set into motion depends on the type of drive and load. The speed can be changed during motion. The control unit adjusts the speed according to the soft start ramp.

Read-out of information

Operating states and information can be read from the control unit via the RG command. Values can be queried individually or blockwise.

Position data

The indices 0011 to 0016 will return current position. The grouping index 0010 returns the position of all 6 possible 6 drives. The position can be calculated in mm from the values
of end position and hub length.

5. Communication examples

Example: Move to position and read current position with SCP11 parameterization

With the SCP11 parameterization all drives are set for individual operation. Drive 1 is assigned to function 1, drive 2 to function 2 and so on. In this way the drives can be controlled individually using functions 1-6.

Routine:

- Communication mode open with RO (Safety ID)
- Set remote position of drive 1
- Start movement of drive 1
- Read status of drive 1. Check if movement is activated.
- Read current position of drive 1
- During the entire routine a cyclically repeated RC command communication must occur at least every 500 ms. The RC communication functions as a watchdog. If the RC communication should fail, the SCU will stop all drives in motion and deactivate the remote mode.
- Before the first command is sent to the SCU an RC communication must also have taken place (activation of remote mode)
- Communication mode closed with RA

Tab. 5-1 Periodic RC communication without any data transfer in this case (without CyclObj):

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

– – – Safe communication mode open

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

Tab. 5-2 Setting of remote speed of drive 1 to value 100h with RT command (data index 3011):

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

Tab. 5-3 Drive to the Remote Position with actuator 1 without start/stop ramp.
Starts with the RE command (Data index 0):

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

Tab. 5-4 Request status of drive 1 with RG command (Data index 0171):

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

Status bit 4 is set so long as the motion is active.

Tab. 5-5 Request current position of drive 1 with RG command (data index 0011):

Cmd	Name	Request parameter						Reply parameter						
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4	P5	P6	P7
RG	Remote data get	11	00	–	–	–	–	ACK, **E	04	00	1. Daten byte	2. Daten byte	3. Daten byte	4. Daten byte

Cmd	Name	Request parameter						Reply parameter			
		P1	P2	P3	P4	P5	Pn	P1	P2	P3	P4
RA	Remote Mode abord	–	–	-1	–	–		ACK, **E	–	–	–

Close communication mode

6. Code examples

Checksum calculation

The checksum is determined using the standard CCITT CRC16 algorithm. 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. In order to reduce this a CRC table should ideally be used.

Tab. 6-1 Code example 1: 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
};
```

Code example 2 is an example of CRC checksum determination using the table. The 2 bytes returned must be connected to the command.

Tab. 6-2 Code example 3: Calculation of checksum using the table

```
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;
}
```

Tab. 6-3 Code example 4: Check of checksum result

```
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

```
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];
```