Version history

Revision	Release	Date	Description
0.1	V1.00	06.07.2009	Initial revision
0.2	V1.00	07.08.2009	- Correction of command descriptions factory / applications defaults - Get Serialnumber added. - Table of content added.
0.3	V1.00	31.08.2009	- WriteSingleBlock command description corrected.
0.4	V1.10	28.09.2009	- Added WriteMultiBlock command - extension of column Release in this table
0.5	V1.20	16.08.2010	- Now with integrated PDF-Bookmarks - Read- & WriteMultipleBlocksString added. - More descriptions on the Read/WriteBlockCommands. - Chapter I/O Handling added. - Error code definition update. - Basic TransponderProcessing handling added.

Table of content

1. Basic Commands

Some commands to get access and basic informations from the device.

1.1. Login to device

You have to login to the device for several commands, e.g. to write a variable.

Any changes start to take effect after logout from the device!

Command: Host => Device	
`[STX]sMN SetAccessMode <level> <passwordHashValue>[ETX]`	
Parameter	**Description**
level	Access mode level (e. g. AuthorizedClient = 3)
passwordHashValue	Hash value of the required password

Response: Device => Host	
`[STX]sAN SetAccessMode <success>[ETX]`	
Return value	**Description**
success	Login was successful = 1, else 0.

Example: Login as AuthorizedClient
`[STX]sMN SetAccessMode 3 7A99FDC6[ETX]`
`[STX]sAN SetAccessMode 1[ETX]`

1.2. Logout from device

Set device back to run level. After successful execution of this command, changes will take effect.

Command: Host => Device
`[STX]sMN Run[ETX]`

Response: Device => Host	
`[STX]sAN Run <success>[ETX]`	
Return value	**Description**
success	Logout was successful = 1, else 0.

Example: Logout / Set device into run level
`[STX]sMN Run[ETX]`
`[STX]sAN Run 1[ETX]`

1.3. Save parameters permanent

All parameters will be stored permanently.

Command: Host => Device
`[STX]sMN mEEwriteall[ETX]`

Response: Device => Host	
`[STX]sAN mEEwriteall <success>[ETX]`	
Return value	**Description**
success	Saving was successful = 1, else 0.

Example: Save all parameters permanent.
`[STX]sMN mEEwriteall[ETX]`
`[STX]sAN mEEwriteall 1[ETX]`

1.4. Load factory defaults in device

All parameters, including the communication settings, will be set to their default values.

Command: Host => Device
`[STX]sMN mSCloadfacdef[ETX]`

Response: Device => Host
`[STX]sAN mSCloadfacdef[ETX]`

Example: Set all parameters back to their defaults.
`[STX]sMN mSCloadfacdef[ETX]`
`[STX]sAN mSCloadfacdef[ETX]`

1.5. Load application defaults in device

All parameters, exclusive of the communication settings, will be set to their default values.

Command: Host => Device
`[STX]sMN mSCloadappdef[ETX]`

Response: Device => Host
`[STX]sAN mSCloadappdef[ETX]`

Example: Set all parameters back to their defaults.
`[STX]sMN mSCloadappdef[ETX]`
`[STX]sAN mSCloadappdef[ETX]`

1.6. Get device ident

Read the device identification.

Command: Host => Device
`[STX]sRN DeviceIdent[ETX]`

Response: Device => Host	
`[STX]sRA DeviceIdent <ln> <name> <lv> <version>[ETX]`	
Return value	**Description**
ln	length of name
name	Name of the device
lv	length of version
version	firmware version of the device

Example: Read device identification.
`[STX]sRN DeviceIdent[ETX]`
`[STX]sRA DeviceIdent 6 RFH620 10 V1.20-03.03.2010[ETX]`

1.7. Get device type

Read the type of the device.

Command: Host => Device
`[STX]sRN DItype[ETX]`

Response: Device => Host	
`[STX]sRA DItype <lt> <type>[ETX]`	
Return value	**Description**
lt	length of type
type	type of the device (see ATAB)

Example: Read device type
`[STX]sRN DItype[ETX]`
`[STX]sRA DItype E RFH620-1001201[ETX]`

1.8. Get serialnumber

Read the serial number from the device.

Command: Host => Device
`[STX]sRN SerialNumber[ETX]`

Response: Device => Host	
`[STX]sRA SerialNumber <ls> <serial>[ETX]`	
Return value	**Description**
ls	length of serial
serial	serial number of the device

Example: Read device identification.
`[STX]sRN SerialNumber[ETX]`
`[STX]sRA SerialNumber 8 08510010[ETX]`

2. Action Commands

2.1. Inventory / Get UID

Method to start an inventory to search for transponder. The method returns a FlexArray of tranponder information. For each tranponder the following four information are returned.

Command: Host => Device
`[STX]sMN CSGtUID[ETX]`

Response: Device => Host	
`[STX]sAN CSGtUID <n> {<err> <rssi> <dsfid> <uid>}[ETX]`	
Return value	**Description**
n	number of returned tranponder sets (0-32)
err	error code (0x00 => no error; see section 2.18)
rssi	RSSI RX value of this transponder (1byte)
dsfid	DSFID (1 byte)
uid	UID (eight byte, space separated)

Example 1: Start an inventory with one tranponder as result
`[STX]sMN CSGtUID[ETX]`
`[STX]sAN CSGtUID 1 0 3 0 F3 AB 16 8 0 1 4 E0[ETX]`

Example 2: Start an inventory with two tranponder as result
`[STX]sMN CSGtUID[ETX]`
`[STX]sAN CSGtUID 2 0 4 0 F3 AB 16 8 0 1 4 E0 0 4 0 FB AB 16 8 0 1 4 E0[ETX]`

Example 3: Start an inventory with no tranponder as result (0x22 = No response), see section 2.18
`[STX]sMN CSGtUID[ETX]`
`[STX]sAN CSGtUID 1 22 0 0 0 0 0 0 0 0 0 0[ETX]`

2.2. Stay quiet

Command: Host => Device	
`[STX]sMN CSStayQt <uid>[ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSStayQt <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Send StayQuiet command to transponder E0-04-01-00-08-16-AB-F3
`[STX]sMN CSStayQt F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSStayQt 0[ETX]`

2.3. Read single block

Command: Host => Device	
`[STX]sMN CSRdSnglBlck <uid> <bn>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of the block that should be read

Response: Device => Host	
`[STX]sAN CSRdSnglBlck <err> <lbc> <bc>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)
lbc	length of block content (hex) in byte
bc	block content, space separated

Example 1: Read block 10 (dec.) from transponder E0-04-01-00-08-16-AB-F3
`[STX]sMN CSRdSnglBlck F3 AB 16 8 0 1 4 E0 +10[ETX]`
`[STX]sAN CSRdSnglBlck 0 4 11 22 33 44[ETX]`

Example 2: Read block 17 (dec. / hex 0x11) non adressed.
`[STX]sMN CSRdSnglBlck 00 00 00 00 00 00 00 00 11[ETX]`
`[STX]sAN CSRdSnglBlck 0 4 AA BB CC DD[ETX]`

2.4. Write single block

Command: Host => Device	
`[STX]sMN CSWrtSnglBlck <uid> <bn> <lbc> <bc>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of the block that should be written
lbc	length of block content in byte
bc	blockcontent, space separated as HexByte

Response: Device => Host	
`[STX]sAN CSWrtSnglBlck <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example 1: Write block 10 (dec.) from transponder E0-04-01-00-08-16-AB-F3 with content 0x31 0x32 0x33 0x34.
`[STX]sMN CSWrtSnglBlck F3 AB 16 8 0 1 4 E0 +10 4 31 32 33 34[ETX]`
`[STX]sAN CSWrtSnglBlck 0[ETX]`

Example 2: Write block 17 (dec. / hex 0x11) non adressed with content 0x31 0x32 0x33 0x34.
`[STX]sMN CSWrtSnglBlck 00 00 00 00 00 00 00 00 11 4 31 32 33 34[ETX]`
`[STX]sAN CSWrtSnglBlck 0[ETX]`

2.5. Lock block

Command: Host => Device	
`[STX]sMN CSLckBlck <uid> <bn>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of the block that should be written.

Response: Device => Host	
`[STX]sAN CSLckBlck <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Lock block number 10 (dec.) from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSLckBlck F3 AB 16 8 0 1 4 E0 +10[ETX]`
`[STX]sAN CSLckBlck 0[ETX]`

2.6. Read multiple blocks

Command: Host => Device	
`[STX]sMN CSRdMltBlck <uid> <bn> <nb>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of first block that should be read
nb	number of blocks minus one

Response: Device => Host	
`[STX]sAN CSRdMltBlck <err> <lbc> <bc>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)
lbc	length of block content (hex) in byte
bc	block content, space separated as HexBytes

Example: Read blocks 10 and 11(dec.) from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSRdMltBlck F3 AB 16 8 0 1 4 E0 +10 1[ETX]`
`[STX]sAN CSRdMltBlck 0 8 31 32 33 34 35 36 37 38[ETX]`

2.7. Write multiple blocks

Note: The number of blocks and the length of the block content are used to determine the block size.

Command: Host => Device	
`[STX]sMN WrtMltBlck <uid> <bn> <nb> <lbc> <bc>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of first block that should be written
nb	number of blocks minus one
lbc	length of block content in byte
bc	block content as HexBytes

Response: Device => Host	
`[STX]sAN WrtMltBlck <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Write blocks 10 and 11(dec.) from transponder E0-04-01-00-08-16-AB-F3. Assume blocksize of 4 bytes and write the string ”RFIDtest“
`[STX]sMN WrtMltBlck F3 AB 16 8 0 1 4 E0 +10 1 8 52 46 49 44 74 65 73 74[ETX]`
`[STX]sAN WrtMltBlck 0[ETX]`

2.8. ReadMultipleBlocksString

Command: Host => Device	
`[STX]sMN RdMltBlckStr <uid> <bn> <nb>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of first block that should be read
nb	number of blocks minus one

Response: Device => Host	
`[STX]sAN RdMltBlckStr <err> <lbc> <bc>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)
lbc	length of block content (hex) in byte
bc	block content as string

Example: Read blocks 0 to 5 non-adressed.
`[STX]sMN RdMltBlckStr 0 0 0 0 0 0 0 0 0 5[ETX]`
`[STX]sAN RdMltBlckStr 0 18 Lorem ipsum dolor sit am[ETX]`

2.9. WriteMultipleBlocksString

Note: The number of blocks and the length of the block content are used to determine the block size.

Command: Host => Device	
`[STX]sMN WrtMltBlckStr <uid> <bn> <nb> <lbc> <bc>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of first block that should be written
nb	number of blocks minus one
lbc	length of block content
bc	block content as string

Response: Device => Host	
`[STX]sAN WrtMltBlckStr <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Write blocks 0 to 5 non-adressed. Assume blocksize of 4 bytes.
`[STX]sMN WrtMltBlckStr 0 0 0 0 0 0 0 0 0 5 18 Lorem ipsum dolor sit am[ETX]`
`[STX]sAN WrtMltBlckStr 0[ETX]`

2.10. Select state

Command: Host => Device	
`[STX]sMN CSSlct <uid> [ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSSlct <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Set transponder E0-04-01-00-08-16-AB-F3 to select-state.
`[STX]sMN CSSlct F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSSlct 0[ETX]`

2.11. Reset to ready

Command: Host => Device	
`[STX]sMN CSRstRdy <uid> [ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSRstRdy <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Set transponder E0-04-01-00-08-16-AB-F3 to ready-state.
`[STX]sMN CSRstRdy F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSRstRdy 0[ETX]`

2.12. Write AFI

Command: Host => Device	
`[STX]sMN CSWrtAFI <uid> <afi>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
afi	AFI that should be written to the transponder.

Response: Device => Host	
`[STX]sAN CSWrtAFI <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Write AFI 18 (dec.) from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSWrtAFI F3 AB 16 8 0 1 4 E0 +18[ETX]`
`[STX]sAN CSWrtAFI 0[ETX]`

2.13. Lock AFI

Command: Host => Device	
`[STX]sMN CSLckAFI <uid>[ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSLckAFI <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Lock AFI forever from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSLckAFI F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSLckAFI 0[ETX]`

2.14. Write DSFID

Command: Host => Device	
`[STX]sMN CSWrtDSFID <uid> <dsfid>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
dsfid	DSFID that should be written to the transponder.

Response: Device => Host	
`[STX]sAN CSWrtDSFID <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Write DSFID 18 (dec.) from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSWrtDSFID F3 AB 16 8 0 1 4 E0 +18[ETX]`
`[STX]sAN CSWrtDSFID 0[ETX]`

2.15. Lock DSFID

Command: Host => Device	
`[STX]sMN CSLckDSFID <uid>[ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSLckDSFID <err>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)

Example: Lock DSFID forever from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSLckDSFID F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSLckDSFID 0[ETX]`

2.16. Get transponder information

This command is also called „GetSystemInformation“.

Command: Host => Device	
`[STX]sMN CSGtTAGInf <uid>[ETX]`	
Parameter	**Description**
uid	UID of the transponder

Response: Device => Host	
`[STX]sAN CSGtTAGInf <err> <uid> <f-dsfid> <dsfid> <f-afi> <afi> <f-bn> <bn> <f-bs> <bs> <f-icr> <icr>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)
uid	UID of the transponder
`f-dsfid`	Flag indicates if corresponding value exists. (1 = existing; 0 = not existing)
`dsfid`	DSFID of the transponder.
`f-afi`	Flag indicates if corresponding value exists. (1 = existing; 0 = not existing)
`afi`	AFI of the transponder.
`f-bn`	Flag indicates if corresponding value exists. (1 = existing; 0 = not existing)
`bn`	Number of blocks minus one on this transponder.
`f-bs`	Flag indicates if corresponding value exists. (1 = existing; 0 = not existing)
`bs`	Size of a data block minus one on this transponder.
`f-icr`	Flag indicates if corresponding value exists. (1 = existing; 0 = not existing)
`icr`	IC type of this transponder.

Example: Get system information from transponder E0-04-01-00-08-16-AB-F3.
`[STX]sMN CSGtTAGInf F3 AB 16 8 0 1 4 E0[ETX]`
`[STX]sAN CSGtTAGInf 0 F3 AB 16 8 0 1 4 E0 1 12 1 12 1 1B 1 3 1 1[ETX]`

2.17. Get multiple blocks security information

Command: Host => Device	
`[STX]sMN CSGtBlckSecSt <uid> <bn> <nob>[ETX]`	
Parameter	**Description**
uid	UID of the transponder
bn	number of start block (starts at zero!)
nob	number of blocks minus one.

Response: Device => Host	
`[STX]sAN CSGtBlckSecSt <err> <length> <info>[ETX]`	
Return value	**Description**
err	error code (0x00 => no error; see section 2.18)
length	number of following bytes
info	security information for the required blocks.

Example: Get security information from transponder E0-04-01-00-08-16-AB-F3 of blocks 8 to 13. (Block 10 and 13 are locked)
`[STX]sMN CSGtBlckSecSt F3 AB 16 8 0 1 4 E0 8 5[ETX]`
`[STX]sAN CSGtBlckSecSt 0 6 0 0 1 0 0 1[ETX]`

2.18. Error code definition

Codes with VICC prefix are generated by the transponder, commands with VCD and MPC prefixes are generated by the device.

No dec.	No hex.	Name
0	0x00	NO_ERROR
1	0x01	VICC_CMD_NOT_SUPPORTED
2	0x02	VICC_CMD_NOT_RECOGNIZED
3	0x03	VICC_OPTION_NOT_SUPPORTED
15	0x0F	VICC_UNKNOWN_ERROR
16	0x10	VICC_BLCK_NOT_AVAILABLE
17	0x11	VICC_BLCK_ALRDY_LOCKED
19	0x13	VICC_BLCK_WRITE_ERROR
20	0x14	VICC_BLCK_LOCK_ERROR
30	0x1E	VCD_UNKNOWN_ERROR
31	0x1F	VCD_CRC_ERROR
32	0x20	VCD_PARITY_ERROR
33	0x21	VCD_TIMEOUT_ERROR
34	0x22	VCD_NO_RESP_ERROR
35	0x23	VCD_COLLISION_ERROR
36	0x24	VCD_CONTENT_CHECK_ERROR
37	0x25	VCD_FRAMING_ERROR
38	0x26	VCD_VERIFY_ERROR
39	0x27	VCD_TRANSMIT_ERROR
40	0x28	VCD_RECEIVE_ERROR
41	0x29	VCD_NON_ADDRESSED_ERROR
42	0x2A	VCD_TAGTYPE_SELECTION_ERROR
43	0x2B	MPC_MAX_BLOCK_COUNT_ERROR
44	0x2C	MPC_BLOCK_LENGTH_MISMATCH_ERROR
70	0x46	VCD_SLOT_DETECT_WARNING

3. HF Settings

3.1. Transmission modulation

Read TX modulation: Host => Device		
`[STX]sRN CSTxMod[ETX]`		
Response: Device => Host		
`[STX]sRA CSTxMod <val>[ETX]`		
Return value	**Description**	
val	setting of the TX modulation:	
	3	10% ASK
	5	20% ASK (Default)
	7	100% ASK

Write TX modulation: Host => Device	
`[STX]sWN CSTxMod <val>[ETX]`	
Parameter	**Description**
val	value that should be set
Response: Device => Host	
`[STX]sWA CSTxMod[ETX]`	

Example 1: Read the TX modulation setting.
`[STX]sRN CSTxMod[ETX]`
`[STX]sRA CSTxMod 5[ETX]`

Example 2: Write the TX modulation setting.
`[STX]sWN CSTxMod 7[ETX]`
`[STX]sWA CSTxMod[ETX]`

3.2. Anticollision

Read anti-collision-mode: Host => Device		
`[STX]sRN CSSlSlct[ETX]`		
Response: Device => Host		
`[STX]sRA CSSlSlct <val>[ETX]`		
Return value	**Description**	
val	setting of the anti-collision-mode:	
	0	Automatic (Default; first try single slot, on fault multi-slot)
	1	multi-slot mode
	2	single-slot-mode

Write anti-collision-mode: Host => Device	
`[STX]sWN CSSlSlct <val>[ETX]`	
Parameter	**Description**
val	value that should be set
Response: Device => Host	
`[STX]sWA CSSlSlct[ETX]`	

Example 1: Read the anti-collision-mode setting.
`[STX]sRN CSSlSlct[ETX]`
`[STX]sRA CSSlSlct 2[ETX]`

Example 2: Write the anti-collision-mode setting.
`[STX]sWN CSSlSlct 1[ETX]`
`[STX]sWA CSSlSlct[ETX]`

3.3. HF field

Read HF-field-mode: Host => Device		
`[STX]sRN CSHF[ETX]`		
Response: Device => Host		
`[STX]sRA CSHF <val>[ETX]`		
Return value	**Description**	
val	setting of the HF-field-mode:	
	0	Field is only during request active. (Default)
	1	Always active.

Write HF-field-mode: Host => Device	
`[STX]sWN CSHF <val>[ETX]`	
Parameter	**Description**
val	value that should be set
Response: Device => Host	
`[STX]sWA CSHF[ETX]`	

Example 1: Read the HF-field-mode setting.
`[STX]sRN CSHF[ETX]`
`[STX]sRA CSHF 0[ETX]`

Example 2: Write the HF-field-mode setting.
`[STX]sWN CSHF 1[ETX]`
`[STX]sWA CSHF[ETX]`

4. Trigger commands

4.1. External trigger command

Command: Gate ON Host => Device	
`[STX]sMN mTCgateon[ETX]`	
Response: Device => Host	
`[STX]sAN mTCgateon <success>[ETX]`	
Return value	**Description**
`success`	indicates if command was successfull (1 = success; 0 else)

Command: Gate OFF Host => Device	
`[STX]sMN mTCgateoff[ETX]`	
Response: Device => Host	
`[STX]sAN mTCgateoff <success>[ETX]`	
Return value	**Description**
`success`	indicates if command was successfull (1 = success; 0 else)

Example: One reading gate	
`[STX]sMN mTCgateon[ETX]`	Command: Gate ON Host => Device
`[STX]sAN mTCgateon 1[ETX]`	Response: Device => Host
`[STX]sMN mTCgateoff[ETX]`	Command: Gate OFF Host => Device
`[STX]sAN mTCgateoff 1[ETX]`	Response: Device => Host
`[STX]0[962ms];E00401000816ABF3[ETX]`	result output, userdefined.

4.2. Automatic self trigger / freewheel mode

Read freewheel-mode: Host => Device		
`[STX]sRN UCfrwhlActv[ETX]`		
Response: Device => Host		
`[STX]sRA UCfrwhlActv <val>[ETX]`		
Return value	**Description**	
val	setting of the freewheel-mode:	
	0	not active (Default)
	1	active

Write freewheel-mode: Host => Device	
`[STX]sWN UCfrwhlActv <val>[ETX]`	
Parameter	**Description**
val	value that should be set
Response: Device => Host	
`[STX]sWA UCfrwhlActv[ETX]`	

Example 1: Read the freewheel-mode setting.
`[STX]sRN UCfrwhlActv[ETX]`
`[STX]sRA UCfrwhlActv 0[ETX]`

Example 2: Write the freewheel-mode setting.
`[STX]sWN UCfrwhlActv 1[ETX]`
`[STX]sWA UCfrwhlActv[ETX]`

5. I/O Handling

5.1. Set result / output states

Beware of other possible signal sources for the outputs, e. g. GoodRead or DeviceReady.

Command: Activate output: Host => Device	
`[STX]sMN mDOSetOutput <output> <val>[ETX]`	
Parameter	**Description**
output	output that should be set (1= Result 1; 2= Result 2)
val	value that should be set
Response: Device => Host	
`[STX]sAN mDOSetOutput <success>[ETX]`	
Return value	**Description**
`success`	indicates if command was successfull success (1 = success; 0 else)

Example: Activate Output 2
`[STX]sMN mDOSetOutput 2 1[ETX]`
`[STX]sAN mDOSetOutput 1[ETX]`

5.2. Read sensor / input states

Command: Get input state: Host => Device	
`[STX]sMN mDIReadInput <output>[ETX]`	
Parameter	**Description**
output	output that should be set (1= Result 1; 2= Result 2)
Response: Device => Host	
`[STX]sAN mDIReadInput <state>[ETX]`	
Return value	**Description**
`state`	indicates if input is active or not (1 = active)

Example: Read State Of Sensor1 input. Answer is „Active“
`[STX]sMN mDIReadInput 1[ETX]`
`[STX]sAN mDIReadInput 1[ETX]`

6. Transponder processing

The transponder-processing is a definition of actions that is executed on each transponder, detected by the interrogator. The set of actions is stored in one variable called `UCRWCfg`.

The following actions / commands can be used:

command id	**description**
0	Read block
1	Write block
2	Activate „stay quiet“
3	Read tranponder information
4	Read multiple blocks
5	Write multiple blocks
6	Get UID

The variable can hold up to 50 commands.

They are organised as an array of commands with dynamic length.

6.1. Read/Write TransponderProcessing

Command: Host => Device
`[STX]sRN UCRWCfg[ETX]`

Response: Device => Host	
`[STX]sRA UCRWCfg noc c1-id c1-fix-byte c1-len c1-dyn-content c2-id ...[ETX]`	
Return value	**Description**
noc	number of following commands.
c1-id	id of the command
`c1-fix-byte`	first parameter, always on this position
`c1-len`	length byte of following optional part of command
`c1-dyn-content`	optional part of command with dynmaic length
`c2-id`	next command, if exists.

Example: Write config:

Recently there are two (2)commands defined:

4 => ReadMultiBlock StartBlock and NumberOfBlocks(- one).

6 => GetUID w/o any parameters.

nob => number of bytes that follows in this command

```
[STX]sWN UCRWCfg 2 4 A 1 1 6 0 0[ETX]
[STX]sWA UCRWCfg[ETX]
```

Example: Read config:

Recently there are two (2)commands defined:

5 => WriteMultiBlock StartBlock and NumberOfBlocks(- one) and the content (11 ... 88).

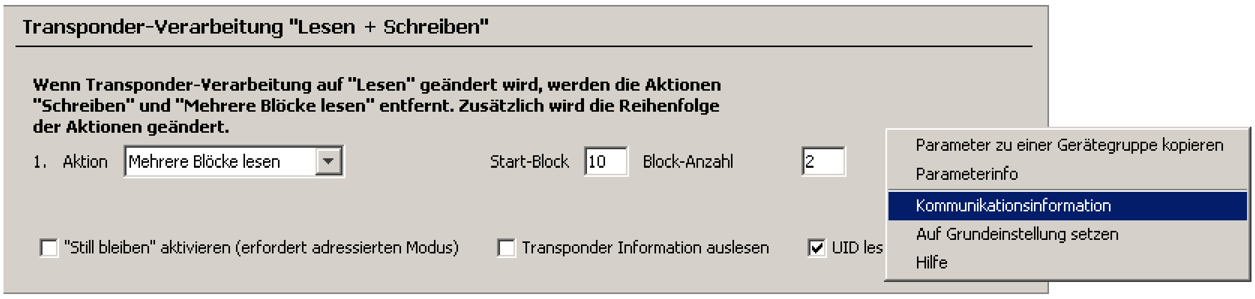
6 => GetUID w/o any parameters.

nob => number of bytes that follows in this command

```
[STX]sRN UCRWCfg[ETX]
[STX]sRA UCRWCfg 2 5 A 9 1 11 22 33 44 55 66 77 88 6 0 0[ETX]
```

6.2. Communicationsinformation in SopasET

Please use for first experiences the SopasTool and check the definitions of your config with the communicationsinformation in the contextmenue.



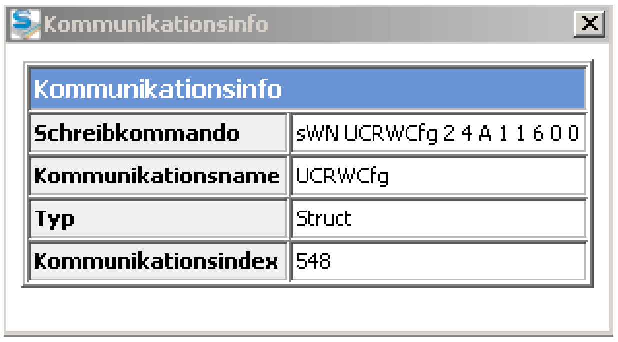
Transponder-Verarbeitung "Lesen + Schreiben"

Wenn Transponder-Verarbeitung auf "Lesen" geändert wird, werden die Aktionen "Schreiben" und "Mehrere Blöcke lesen" entfernt. Zusätzlich wird die Reihenfolge der Aktionen geändert.

1. Aktion: Mehrere Blöcke lesen | Start-Block: 10 | Block-Anzahl: 2

☐ "Still bleiben" aktivieren (erfordert adressierten Modus) ☐ Transponder Information auslesen ☑ UID les

Parameter zu einer Gerätegruppe kopieren
Parameterinfo
Kommunikationsinformation
Auf Grundeinstellung setzen
Hilfe



Kommunikationsinfo

Kommunikationsinfo	
Schreibkommando	sWN UCRWCfg 2 4 A 1 1 6 0 0
Kommunikationsname	UCRWCfg
Typ	Struct
Kommunikationsindex	548