HEART BEAT	**ASCII READ COMMAND**	#HEART BEAT<CR> #HB<CR> Result: #HB<CR>	ASCII	
	TX	#HEART BEAT<CR>		
	RX	N/A		
Sends an Heartbeat to test the communcation				
GET VERSION	**ASCII READ COMMAND**	#VERSION<CR> #VER<CR> Result: #VERSION:<VersionHi>,<VersionMed>,<VersionLo><CR>	ASCII	
	TX	#VERSION<CR>		
	RX	N/A		
		Current SW version:????		
Returns the version number of the module VersionHi: Version number high (1..255) VersionMed: Version number medium (1..255) VersionLo: Version number low (1..255)				
GET TYPE	**ASCII READ COMMAND**	#TYPE<CR> #TYP<CR> Result: #TYPE:<Type><CR>	ASCII	
	TX	#TYPE<CR>		
	RX	N/A		
		Current module type:????		
Returns the current module type				
GET FEATURES	**ASCII READ COMMAND**	#FEATURES<CR> #FTRS<CR> Result: #FTRS:<Type><CR>	ASCII	
	TX	#FEATURES<CR>		
	RX	N/A		
		Current module type:N/A		
		Number of digital inputs:N/A		
		Type of digital inputs:N/A		
Returns the current module type				
GET OWNER	**ASCII READ COMMAND**	#OWNER<CR> #OWN<CR> Result: #OWNER:<Owner><CR>	ASCII	
	TX	#OWNER<CR>		
	RX	N/A		
		Current owner:????		
Returns the current owner of the module				

GET CREATOR	**ASCII READ COMMAND**	#CREATOR<CR> #CRE<CR> Result: #CREATOR:<Creator><CR>	ASCII	
	TX	#CREATOR<CR>		
	RX	N/A		
		Current creator:????		
Returns the current creator of the module				
GET COPYRIGHT	**ASCII READ COMMAND**	#COPYRIGHT<CR> #COPY<CR> Result: #COPYRIGHT:<Copyright><CR>	ASCII	
	TX	#COPYRIGHT<CR>		
	RX	N/A		
		Current copyright:????		
Returns the current copyright of the module				
GET SERIAL NUMBER	**ASCII READ COMMAND**	#SERIAL NUMBER<CR> #SN<CR> Result: #SN:<Serial><CR>	ASCII	
	TX	#SERIAL NUMBER<CR>		
	RX	N/A		
		Current serial number:????		
Returns the current serial number of the module				
SET BOX NAME	**ASCII WRITE COMMAND**	#SET BOX NAME:<BOXNAME><CR> #SETBOXNAME:<BOXNAME><CR> Result: #OK<CR>	ASCII	YES
	BOXNAME	PBOX00005		
	TX	#SET BOX NAME:PBOX00005<CR>		
	RX	N/A		
Sets a new box name in the format TBOXddddd, where ddddd is a unique number between 00001 and 99999				
GET BOX NAME	**ASCII READ COMMAND**	#BOX NAME<CR> #BOXNAME<CR> Result: #BOXNAME:<BoxName><CR>	ASCII	
	TX	#BOX NAME<CR>		
	RX	N/A		
		Current box name:????		
Returns the current box name of the module. if no box name is defined, the value NONAME is returned				
GET INTERNAL STATUS	**ASCII READ COMMAND**	#INTERNAL STATUS<CR> #INTSTAT<CR> Result: #INTSTAT:<Status><CR>	ASCII	
	TX	#INTERNAL STATUS<CR>		

	RX	N/A		
		Current internal status:????		
Returns the device specific internal status				
GET DIP SWITCH	**ASCII READ COMMAND**	#GET DIP<CR> #GDIP<CR> Result: #GDIP:<DIPSwitchDec>,<DIPSwitchHex><CR>	ASCII	
	TX	#GET DIP<CR>		
	RX	N/A		
		Current DIP SWITCH settings:00000000		
Returns the current setting of the Dip switches as decimal number and as hexadecimal number. DIPSwitchDec DIPSwitchHex The current value of the DIP switches: Bit 0: DIP Switch 1 (=0:OFF, =1:ON) Bit 1: DIP Switch 2 (=0:OFF, =1:ON) Bit 2: DIP Switch 3 (=0:OFF, =1:ON) Bit 3: DIP Switch 4 (=0:OFF, =1:ON) Bit 4: DIP Switch 5 (=0:OFF, =1:ON) Bit 5: DIP Switch 6 (=0:OFF, =1:ON) Bit 6: DIP Switch 7(=0:OFF, =1:ON) Bit 7: DIP Switch 8 (=0:OFF, =1:ON)				
SYSTEM COMMANDS				
RESET	**ASCII WRITE COMMAND**	#RESET<CR> #RST<CR> Result: #OK<CR>	ASCII	NO
	TX	#RESET<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				
FACTORY RESET	**ASCII WRITE COMMAND**	#FACTORY RESET<CR> #FRST<CR> Result: #OK<CR>	ASCII	NO
	TX	#FACTORY RESET<CR>		
	RX	N/A		
Executes a factory reset of the module				
WADTCHDOG TIMER	**ASCII WRITE COMMAND**	#WATCHDOG:<WDTIME><CR> #WD:<WDTIME><CR> Result: #OK<CR>	ASCII	NO
	WDTIME	1		
	TX	#WATCHDOG:1<CR>		
	RX	N/A		
Enables or disables the WATCHDOG Timer for the Raspberry Pi module. WDTIME: 1..3600000: Time for Watchdog in Milliseconds (Maximum 60 Minutes) =0: No Watchdog is generated HINT: The Watchdog is internally handled every 10ms, so every value below 10 will reset immediately the Raspberry Pi computer.				

GET REAL TIME CLOCK	**ASCII READ COMMAND**	#GET RTC<CR> #GRTC<CR> Result: #GRTC:YMD,<YEAR>,<MONTH>,<DAY>,HMS,<HOUR>,<MINUTE>,<SECOND>,<WEEKDAY> ,DOK,<DATEOK>,TOK,<TIMEOK><CR>	ASCII	
	TX	#GET RTC<CR>		
	RX	#255,GRTC:YMD,20,1,1,HMS,4,47,20,WED,DOK,1,TOK,1<CR>		
		Current date DD.MM.YYYY:1.1.2020		
		Current time HH.MM.SS (24h):04:47:20		
		Current Weekday:WED		
		Battery buffered date is ok:YES		
		Battery buffered time is ok:YES		
Shows current RTC time of battery backup RTC on module				
GET UTC	**ASCII READ COMMAND**	#GET UTC<CR> #GUTC<CR> Result: #GUTC:YMD,<YEAR>,<MONTH>,<DAY>,HMS,<HOUR>,<MINUTE>,<SECOND>,<WEEKDAY>, SYNC,<SyncState><CR>	ASCII	
	TX	#GET UTC<CR>		
	RX	#255,GUTC:YMD,20,1,1,HMS,4,47,18,WED,SYNC,5<CR>		
		Current date DD.MM.YYYY:1.1.2020		
		Current time HH.MM.SS (24h):04:47:18		
		Current Weekday:WED		
		Sync state of clock:5		
Shows current RTC time of battery backup RTC on module				
ASCII COMMANDS:REAL TIME CLOCK				
SET REAL TIME CLOCK	**ASCII WRITE COMMAND**	#SET RTC:YMD,<YEAR>,<MONTH>,<DAY>,HMS,<HOUR>,<MINUTE>,<SECOND>, <WEEKDAY><CR> #SRTC:YMD,<YEAR>,<MONTH>,<DAY>,HMS,<HOUR>,<MINUTE>,<SECOND>, <WEEKDAY><CR> Result: #OK<CR>	ASCII	YES
	YEAR	2020		
	MONTH	4		
	DAY	23		
	HOUR	8		
	MINUTE	20		
	SECOND	23		
	WEEKDAY	THU		
	TX	#SET RTC:YMD,20,4,23,HMS,8,20,23,THU<CR>		
	RX	N/A		
Executes a software reset (Reboot) of the module.				

GET FRAM32	**ASCII READ COMMAND**	#GET FRAM32:<INDEX><CR> #GFRAM32:<INDEX><CR> Result: #GFRAM32:<INDEXDEC>,<VALUEDEC>,<INDEXHEX>,<VALUEHEX><CR> or #GFRAM32:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	0		
	TX	#GET FRAM32:0<CR>		
	RX	#255,GFRAM32:0,1380275017,0x00000000,0x52455349<CR>		
		FRAM Index in bytes:0		
		FRAM Value in decimal:1380275017		
Reads the actual value of FRAM memory <INDEX>				
GET FRAMDBL	**ASCII READ COMMAND**	#GET FRAMDBL:<INDEX><CR> #GFRAMDBL:<INDEX><CR> Result: #GFRAMDBL:<INDEXDEC>,<VALUEDBL>,<INDEXHEX>,<VALUEDBL><CR> or #GFRAMDBL:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	172		
	TX	#GET FRAMDBL:172<CR>		
	RX	#255,GFRAMDBL:172,1.3506e-78,0x000000ac,1.3506e-78<CR>		
		FRAM Index in bytes:172		
		FRAM Value in decimal:1.3506e-78		
Reads the actual value of FRAM memory <INDEX>				
ASCII COMMANDS:FRAM				
SET FRAM32	**ASCII WRITE COMMAND**	#SET FRAM32:<INDEX>,<VALUE><CR> #SFRAM32:<INDEX>,<VALUE><CR> Result: #SFRAM32:OK<CR> or #SFRAM32:ERR<CR>	ASCII	NO
	INDEX	0		
	VALUE	123456		
	TX	#SET FRAM32:0,123456<CR>		
	RX	N/A		
Writes a new value into FRAM memory <INDEX>				
SET FRAMDBL	**ASCII WRITE COMMAND**	#SET FRAMDBL:<INDEX>,<DOUBLEVALUE><CR> #SFRAMDBL:<INDEX>,<DOUBLEVALUE><CR> Result: #SFRAM32:OK<CR> or #SFRAM32:ERR<CR>	ASCII	YES
	INDEX	400		
	DOUBLEVALUE	3,1415926		
	TX	#SET FRAMDBL:400,3.1415926<CR>		
	RX	N/A		
Writes a new value into FRAM memory <INDEX>				

GET FRAMDBL	**ASCII READ COMMAND**	#GET FRAMDBL:<INDEX><CR> #GFRAMDBL:<INDEX><CR> Result: #GFRAMDBL:<INDEXDEC>,<VALUEDBL>,<INDEXHEX>,<VALUEDBL><CR> or #GFRAMDBL:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	400		
	TX	#GET FRAMDBL:400<CR>		
	RX	#255,GFRAMDBL:400,3.1416,0x00000190,3.1416<CR>		
		FRAM Index in bytes:400		
		FRAM Value in decimal:3.1416		
Reads the actual value of FRAM memory <INDEX>				

ASCII COMMANDS:FRAM				
SET FRAMDBL	**ASCII WRITE COMMAND**	#SET FRAMDBL:<INDEX>,<DOUBLEVALUE><CR> #SFRAMDBL:<INDEX>,<DOUBLEVALUE><CR> Result: #SFRAM32:OK<CR> or #SFRAM32:ERR<CR>	ASCII	YES
	INDEX	1000		
	DOUBLEVALUE	-3.1234e-37		
	TX	#SET FRAMDBL:1000,-3.1234e-37<CR>		
	RX	#255,SFRAMDBL:OK<CR>		
Writes a new value into FRAM memory <INDEX>				
GET FRAMDBL	**ASCII READ COMMAND**	#GET FRAMDBL:<INDEX><CR> #GFRAMDBL:<INDEX><CR> Result: #GFRAMDBL:<INDEXDEC>,<VALUEDBL>,<INDEXHEX>,<VALUEDBL><CR> or #GFRAMDBL:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	1000		
	TX	#GET FRAMDBL:1000<CR>		
	RX	#255,GFRAMDBL:1000,-3.1234e-37,0x000003e8,-3.1234e-37<CR>		
		FRAM Index in bytes:1000		
		FRAM Value in decimal:-3.1234e-37		
Reads the actual value of FRAM memory <INDEX>				
GET FRAM32	**ASCII READ COMMAND**	#GET FRAM32:<INDEX><CR> #GFRAM32:<INDEX><CR> Result: #GFRAM32:<INDEXDEC>,<VALUEDEC>,<INDEXHEX>,<VALUEHEX><CR> or #GFRAM32:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	20		
	TX	#GET FRAM32:20<CR>		
	RX	#255,GFRAM32:20,3046406868,0x00000014,0xb5947ad4<CR>		
		FRAM Index in bytes:20		
		FRAM Value in decimal:3046406868		
Reads the actual value of FRAM memory <INDEX>				
GET FRAM32	**ASCII READ COMMAND**	#GET FRAM32:<INDEX><CR> #GFRAM32:<INDEX><CR> Result: #GFRAM32:<INDEXDEC>,<VALUEDEC>,<INDEXHEX>,<VALUEHEX><CR> or #GFRAM32:<INDEXDEC>,ERR,<INDEXHEX>,ERR<CR>	ASCII	
	INDEX	24		
	TX	#GET FRAM32:24<CR>		
	RX	#255,GFRAM32:24,3241837596,0x00000018,0xc13a841c<CR>		
		FRAM Index in bytes:24		
		FRAM Value in decimal:3241837596		
Reads the actual value of FRAM memory <INDEX>				

ASCII COMMANDS:FRAM				
SET FRAMDBL	**ASCII WRITE COMMAND**	#SET FRAMDBL:<INDEX>,<DOUBLEVALUE><CR> #SFRAMDBL:<INDEX>,<DOUBLEVALUE><CR> Result: #SFRAM32:OK<CR> or #SFRAM32:ERR<CR>	ASCII	YES
	INDEX	1000		
	DOUBLEVALUE	3,1415926		
	TX	#SET FRAMDBL:1000,3.1415926<CR>		
	RX	???		
Writes a new value into FRAM memory <INDEX>				

GET LED1	**ASCII READ COMMAND**	#GET LED1<CR> #GLED1<CR> Result: #GLED1:<LEDMode>,<LEDStateDec>,<LEDStateHex><CR>	ASCII	
	TX	#GET LED1<CR>		
	RX	#255,GLED1:BLINK,0,0x0<CR>		
		Current LED state:BLINK LED ist currently 0		
Returns the current state of the LED1:GREEN on the cover of module				
LED COMMANDS:LED1:GREEN				
SET LED1 OFF	**ASCII WRITE COMMAND**	#SET LED1 OFF<CR> #SL1OFF<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED1 OFF<CR>		
	RX	N/A		
Sets the current state of the LED1:GREEN on the cover of module to OFF				
SET LED1 ON	**ASCII WRITE COMMAND**	#SET LED1 ON<CR> #SL1ON<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED1 ON<CR>		
	RX	N/A		
Sets the current state of the LED1:GREEN on the cover of module to ON				
SET LED1 INVERT	**ASCII WRITE COMMAND**	#SET LED1 INVERT<CR> #SL1INV<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED1 INVERT<CR>		
	RX	N/A		
Inverts the current state of the LED1:GREEN on the cover of module from ON to OFF or from OFF to ON				
SET LED1 PULSE	**ASCII WRITE COMMAND**	#SET LED1 PULSE:<PULSETIME><CR> #SL1PULSE:<PULSETIME><CR> Result: #OK<CR>	ASCII	YES
	PULSETIME	1000		
	TX	#SET LED1 PULSE:1000<CR>		
	RX	N/A		
Sets the current state of the LED1:GREEN on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED1 BLINK	**ASCII WRITE COMMAND**	#SET LED1 BLINK:<BLINKTIME><CR> #SL1BLINK:<BLINKTIME><CR> Result: #OK<CR>	ASCII	NO

	BLINKTIME	1000		
	TX	#SET LED1 BLINK:1000<CR>		
	RX	N/A		
Sets the current state of the LED1:GREEN on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED1 FLASH	**ASCII WRITE COMMAND**	#SET LED1 FLASH:<ONTIME>,<OFFTIME><CR> #SL1FLASH:<ONTIME>,<OFFTIME><CR> Result: #OK<CR>	ASCII	YES
	ONTIME	200		
	OFFTIME	3000		
	TX	#SET LED1 FLASH:200,3000<CR>		
	RX	N/A		
Sets the current state of the LED1:GREEN on the cover of module to FLASH and defines the on and off intervals in Milliseconds between 20 and 600000				

GET LED2	**ASCII READ COMMAND**	#GET LED2<CR> #GLED2<CR> Result: #GLED2:<LEDMode>,<LEDStateDec>,<LEDStateHex><CR>	ASCII	
	TX	#GET LED2<CR>		
	RX	#255,GLED2:BLINK,0,0x0<CR>		
		Current LED state:BLINK LED ist currently 0		
Returns the current state of the LED2:WHITE on the cover of module				
LED COMMANDS:LED2:WHITE				
SET LED2 OFF	**ASCII WRITE COMMAND**	#SET LED2 OFF<CR> #SL2OFF<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED2 OFF<CR>		
	RX	N/A		
Sets the current state of the LED2:WHITE on the cover of module to OFF				
SET LED2 ON	**ASCII WRITE COMMAND**	#SET LED2 ON<CR> #SL2ON<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED2 ON<CR>		
	RX	N/A		
Sets the current state of the LED2:WHITE on the cover of module to ON				
SET LED2 INVERT	**ASCII WRITE COMMAND**	#SET LED2 INVERT<CR> #SL2INV<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED2 INVERT<CR>		
	RX	N/A		
Inverts the current state of the LED2:WHITE on the cover of module from ON to OFF or from OFF to ON				
SET LED2 PULSE	**ASCII WRITE COMMAND**	#SET LED2 PULSE:<PULSETIME><CR> #SL2PULSE:<PULSETIME><CR> Result: #OK<CR>	ASCII	NO
	PULSETIME	1000		
	TX	#SET LED2 PULSE:1000<CR>		
	RX	N/A		
Sets the current state of the LED2:WHITE on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED2 BLINK	**ASCII WRITE COMMAND**	#SET LED2 BLINK:<BLINKTIME><CR> #SL2BLINK:<BLINKTIME><CR> Result: #OK<CR>	ASCII	NO

	BLINKTIME	1000		
	TX	#SET LED2 BLINK:1000<CR>		
	RX	N/A		
Sets the current state of the LED2:WHITE on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED2 FLASH	**ASCII WRITE COMMAND**	#SET LED2 FLASH:<ONTIME>,<OFFTIME><CR> #SL2FLASH:<ONTIME>,<OFFTIME><CR> Result: #OK<CR>	ASCII	YES
	ONTIME	200		
	OFFTIME	3000		
	TX	#SET LED2 FLASH:200,3000<CR>		
	RX	#255,OK<CR>		
Sets the current state of the LED2:WHITE on the cover of module to FLASH and defines the on and off intervals in Milliseconds between 20 and 600000				

GET LED3	**ASCII READ COMMAND**	#GET LED3<CR> #GLED3<CR> Result: #GLED3:<LEDMode>,<LEDStateDec>,<LEDStateHex><CR>	ASCII	
	TX	#GET LED3<CR>		
	RX	#255,GLED3:ON,1,0x1<CR>		
		Current LED state:ON LED ist currently 1		
Returns the current state of the LED3:RED on the cover of module				
LED COMMANDS:LED3:RED				
SET LED3 OFF	**ASCII WRITE COMMAND**	#SET LED3 OFF<CR> #SL3OFF<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED3 OFF<CR>		
	RX	N/A		
Sets the current state of the LED3:RED on the cover of module to OFF				
SET LED3 ON	**ASCII WRITE COMMAND**	#SET LED3 ON<CR> #SL3ON<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED3 ON<CR>		
	RX	N/A		
Sets the current state of the LED3:RED on the cover of module to ON				
SET LED3 INVERT	**ASCII WRITE COMMAND**	#SET LED3 INVERT<CR> #SL3INV<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED3 INVERT<CR>		
	RX	N/A		
Inverts the current state of the LED3:RED on the cover of module from ON to OFF or from OFF to ON				
SET LED3 PULSE	**ASCII WRITE COMMAND**	#SET LED3 PULSE:<PULSETIME><CR> #SL3PULSE:<PULSETIME><CR> Result: #OK<CR>	ASCII	NO
	PULSETIME	1000		
	TX	#SET LED3 PULSE:1000<CR>		
	RX	N/A		
Sets the current state of the LED3:RED on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED3 BLINK	**ASCII WRITE COMMAND**	#SET LED3 BLINK:<BLINKTIME><CR> #SL3BLINK:<BLINKTIME><CR> Result: #OK<CR>	ASCII	NO

	BLINKTIME	1000		
	TX	#SET LED3 BLINK:1000<CR>		
	RX	N/A		
Sets the current state of the LED3:RED on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED3 FLASH	**ASCII WRITE COMMAND**	#SET LED3 FLASH:<ONTIME>,<OFFTIME><CR> #SL3FLASH:<ONTIME>,<OFFTIME><CR> Result: #OK<CR>	ASCII	YES
	ONTIME	200		
	OFFTIME	3000		
	TX	#SET LED3 FLASH:200,3000<CR>		
	RX	#255,OK<CR>		
Sets the current state of the LED3:RED on the cover of module to FLASH and defines the on and off intervals in Milliseconds between 20 and 600000				

GET LED4	**ASCII READ COMMAND**	#GET LED4<CR> #GLED4<CR> Result: #GLED4:<LEDMode>,<LEDStateDec>,<LEDStateHex><CR>	ASCII	
	TX	#GET LED4<CR>		
	RX	#255,GLED4:OFF,0,0x0<CR>		
		Current LED state:OFF LED ist currently 0		
Returns the current state of the LED4:YELLOW on the cover of module				
LED COMMANDS:LED4:YELLOW				
SET LED4 OFF	**ASCII WRITE COMMAND**	#SET LED4 OFF<CR> #SL4OFF<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED4 OFF<CR>		
	RX	N/A		
Sets the current state of the LED4:YELLOW on the cover of module to OFF				
SET LED4 ON	**ASCII WRITE COMMAND**	#SET LED4 ON<CR> #SL4ON<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED4 ON<CR>		
	RX	N/A		
Sets the current state of the LED4:YELLOW on the cover of module to ON				
SET LED4 INVERT	**ASCII WRITE COMMAND**	#SET LED4 INVERT<CR> #SL4INV<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED4 INVERT<CR>		
	RX	N/A		
Inverts the current state of the LED4:YELLOW on the cover of module from ON to OFF or from OFF to ON				
SET LED4 PULSE	**ASCII WRITE COMMAND**	#SET LED4 PULSE:<PULSETIME><CR> #SL4PULSE:<PULSETIME><CR> Result: #OK<CR>	ASCII	NO
	PULSETIME	1000		
	TX	#SET LED4 PULSE:1000<CR>		
	RX	N/A		
Sets the current state of the LED4:YELLOW on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED4 BLINK	**ASCII WRITE COMMAND**	#SET LED4 BLINK:<BLINKTIME><CR> #SL4BLINK:<BLINKTIME><CR> Result: #OK<CR>	ASCII	YES

	BLINKTIME	1000		
	TX	#SET LED4 BLINK:1000<CR>		
	RX	#255,OK<CR>		
Sets the current state of the LED4:YELLOW on the cover of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED4 FLASH	**ASCII WRITE COMMAND**	#SET LED4 FLASH:<ONTIME>,<OFFTIME><CR> #SL4FLASH:<ONTIME>,<OFFTIME><CR> Result: #OK<CR>	ASCII	NO
	ONTIME	200		
	OFFTIME	3000		
	TX	#SET LED4 FLASH:200,3000<CR>		
	RX	N/A		
Sets the current state of the LED4:YELLOW on the cover of module to FLASH and defines the on and off intervals in Milliseconds between 20 and 600000				

GET LED5	**ASCII READ COMMAND**	#GET LED5<CR> #GLED5<CR> Result: #GLED5:<LEDMode>,<LEDStateDec>,<LEDStateHex><CR>	ASCII	
	TX	#GET LED5<CR>		
	RX	N/A		
		Current LED state:N/A LED ist currently N/A		
Returns the current state of the LED5:INTERNAL on the cover of module				
LED COMMANDS:LED5:YELLOW				
SET LED5 OFF	**ASCII WRITE COMMAND**	#SET LED5 OFF<CR> #SL5OFF<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED5 OFF<CR>		
	RX	N/A		
Sets the current state of the LED5:INTERNAL of module to OFF				
SET LED5 ON	**ASCII WRITE COMMAND**	#SET LED5 ON<CR> #SL5ON<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED5 ON<CR>		
	RX	N/A		
Sets the current state of the LED5:INTERNAL of module to ON				
SET LED5 INVERT	**ASCII WRITE COMMAND**	#SET LED5 INVERT<CR> #SL5INV<CR> Result: #OK<CR>	ASCII	NO
	TX	#SET LED5 INVERT<CR>		
	RX	N/A		
Inverts the current state of the LED5:INTERNAL of module from ON to OFF or from OFF to ON				
SET LED5 PULSE	**ASCII WRITE COMMAND**	#SET LED5 PULSE:<PULSETIME><CR> #SL5PULSE:<PULSETIME><CR> Result: #OK<CR>	ASCII	NO
	PULSETIME	1000		
	TX	#SET LED5 PULSE:1000<CR>		
	RX	N/A		
Sets the current state of the LED5:INTERNAL of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED5 BLINK	**ASCII WRITE COMMAND**	#SET LED5 BLINK:<BLINKTIME><CR> #SL5BLINK:<BLINKTIME><CR> Result: #OK<CR>	ASCII	YES

	BLINKTIME	2000		
	TX	#SET LED5 BLINK:2000<CR>		
	RX	N/A		
Sets the current state of the LED5:INTERNAL of module to PULSE and defines the one time pulse duration in Milliseconds between 1 and 60000				
SET LED5 FLASH	**ASCII WRITE COMMAND**	#SET LED5 FLASH:<ONTIME>,<OFFTIME><CR> #SL5FLASH:<ONTIME>,<OFFTIME><CR> Result: #OK<CR>	ASCII	NO
	ONTIME	200		
	OFFTIME	3000		
	TX	#SET LED5 FLASH:200,3000<CR>		
	RX	N/A		
Sets the current state of the LED5:INTERNAL of module to FLASH and defines the on and off intervals in Milliseconds between 20 and 600000				

PUSHBUTTONS				
GET PBS	**ASCII READ COMMAND**	#GET PBS<CR> #GPBS<CR> Result: #GPBS:<PBSDec>,<PBSHex><CR>	ASCII	
	TX	#GET PBS<CR>		
	RX	#255,GPBS:0,0x0<CR>		
		Current status of push buttons:0000.0000.0000.0000		
Returns the current state of all push buttons as decimal number and as hexadecimal number. PBDec, PBHex The current state of all pushbuttons: Bit 0: State of push button 1 (=0:RELEASED, =1:PRESSED) Bit 1: State of push button 2 (=0:RELEASED, =1:PRESSED) Bit 2-15: Always 0				
GET PBx	**ASCII READ COMMAND**	#GET PB<PBNR><CR> #GPB<PBNR><CR> Result: #GPB<PBNR>:<PBxDec>,<PBxHex><CR>	ASCII	
	PBNR	2		
	TX	#GET PB2<CR>		
	RX	#255,GPB2:0,0x0<CR>		
		Current status of push button 2:0=RELEASED		
Returns the current state of the push button PBx as decimal number and as hexadecimal number. X stands for the desired push button between 1 and 2. PBxDec, PBxHex: The current state of the push button x: =0: Push button is RELEASED =1: Push button is PRESSED				
GET ALL CHANGES PBS	**ASCII READ COMMAND**	#GET ALL CHANGES PBS<CR> #GACPBS<CR> Result: #GACPBS:<ChangesDec>,<ChangesHex><CR>	ASCII	
	TX	#GET ALL CHANGES PBS<CR>		
	RX	#255,GACPBS:30,0x1E<CR>		
		Current change counter:30		
Returns the counter for changes on all push buttons. As soon as the module detects a short keypress or long key press or long key release event, this counter is incremented by 1. If this values has changed sience the last polling request, the host knows, that at least one push button has changed its state.				
CHANGE ALL PBS	**ASCII READ COMMAND**	#CHANGE ALL PBS<CR> #CAPBS<CR> Result: #CAPBS:<ChangePB1Dec>,<ChangePB2Dec>,<ChangePB1Hex>,<ChangePB2Hex><CR>	ASCII	
	TX	#CHANGE ALL PBS<CR>		
	RX	#255,CAPBS:12,18,0xC,0x12<CR>		
		Current counter for changes on push button 1:12		
		Current counter for changes on push button 2:18		

Returns for each push button the counter for changes. As soon as the module detects a signal change on a push button, the change counter for the affected push button is incremented by 1. A signal change can be: Detection of a short keypress Detection of the start of a long keypress Detection of a release of a long keypress				
CHANGE PBx	**ASCII READ COMMAND**	#CHANGE PB<PBNR><CR> #CPB<PBNR><CR> Result: #CPB<PBNR>:<ChangesDec>,<ChangesHex><CR>	ASCII	
	PBNR	1		
	TX	#CHANGE PB1<CR>		
	RX	#255,CPB1:12,0xC<CR>		
		Current counter for changes on push button 1:12		
Returns for pushbutton <PBNR> the counter for signal changes. As soon as the module detects a signal change on a push button, the change counter for the affected push button is incremented by 1. A signal change can be: Detection of a short keypress Detection of the start of a long keypress Detection of a release of a long keypress				
SHORT KEY ALL PBS	**ASCII READ COMMAND**	#SHORT KEY ALL PBS<CR> #SKAPBS<CR> Result: #SKAPBS:<ShortKeyPB1Dec>,<ShortKeyPB2Dec>, <ShortKeyPB1Hex>,<ShortKeyPB2Hex><CR>	ASCII	
	TX	#SHORT KEY ALL PBS<CR>		
	RX	#255,SKAPBS:10,8,0xA,0x8<CR>		
		Current counter for short keypress events on push button 1:10		
		Current counter for short keypress events on push button 2:8		
Returns for each push button the counter for short keypress events. As soon as the module detects a short keypress on a push button, the counter for the affected digital input is incremented by 1.				
SHORT KEY PBx	**ASCII READ COMMAND**	#SHORT KEY PB<PBNR><CR> #SKPB<PBNR><CR> Result: #SKPB<PBNR>:<ShortKeyDec>,<ShortKeyHex><CR>	ASCII	
	PBNR	1		
	TX	#SHORT KEY PB1<CR>		
	RX	#255,SKPB1:10,0xA<CR>		
		Current counter for short keypress events on push button 1:10		
Returns for push button <PBNR> the counter for short keypress events. As soon as the module detects a short keypress on a push button, the counter for the affected push button is incremented by 1.				
LONG KEY START ALL PBS	**ASCII READ COMMAND**	#LONG KEY START ALL PBS<CR> #LKSAPBS<CR> Result: #LKSAPBS:<LongKeyStarPB1Dec>,<LongKeyStartPB2Dec>, <LongKeyStartPB1Hex>,<LongKeyStartPB2Hex><CR>	ASCII	
	TX	#LONG KEY START ALL PBS<CR>		
	RX	#255,LKSAPBS:1,5,0x1,0x5<CR>		
		Current counter for long keypress start events on push button 1:1		

		Current counter for long keypress start events on push button 2:5		
Returns for each push button the counter for long keypress start events. As soon as the module detects the start of a long keypress on a push button, the counter for the affected push button is incremented by 1.				
LONG KEY START PBx	**ASCII READ COMMAND**	#LONG KEY START PB<PBNR><CR> #LKSPB<PBNR><CR> Result: #LKSPB<PBNR>:<LongKeyStartDec>,<LongKeyStartHex><CR>	ASCII	
	PBNR	1		
	TX	#LONG KEY START PB1<CR>		
	RX	#255,LKSPB1:1,0x1<CR>		
		Current counter for long keypress start events on push button 1:1		
Returns for push button <PBNR> the counter for long keypress start events. As soon as the module detects the start of a long keypress on a push button, the counter for the affected push button is incremented by 1.				
LONG KEY END ALL PBS	**ASCII READ COMMAND**	#LONG KEY END ALL PBS<CR> #LKEAPBS<CR> Result: #LKEAPBS:<LongKeyEndPB1Dec>,<LongKeyEndPB2Dec>, <LongKeyEndPB1Hex>,<LongKeyEndPB2Hex><CR>	ASCII	
	TX	#LONG KEY END ALL PBS<CR>		
	RX	#255,LKEAPBS:1,5,0x1,0x5<CR>		
		Current counter for long keypress end events on push button 1:1		
		Current counter for long keypress end events on push button 2:5		
Returns for each push button the counter for long keypress end events. As soon as the module detects the end of a long keypress on a push button, the counter for the affected push button is incremented by 1.				
LONG KEY END PBx	**ASCII READ COMMAND**	#LONG KEY END PB<PBNR><CR> #LKEPB<PBNR><CR> Result: #LKEPB<PBNR>:<LongKeyEndDec>,<LongKeyEndHex><CR>	ASCII	
	PBNR	1		
	TX	#LONG KEY END PB1<CR>		
	RX	#255,LKEPB1:1,0x1<CR>		
		Current counter for long keypress end events on pusbh button 1:1		
Returns for push button <PBNR> the counter for long keypress end events. As soon as the module detects the end of a long keypress on a push button, the counter for the affected push button is incremented by 1.				
RISE ALL PBS	**ASCII READ COMMAND**	#RISE ALL PBS<CR> #RAPBS<CR> Result: #RAPBS:<RisePB1Dec>,<RisePB2Dec>,<RisePB1Hex>,<RisePB2Hex><CR>	ASCII	
	TX	#RISE ALL PBS<CR>		
	RX	#255,RAPBS:11,13,0xB,0xD<CR>		
		Current counter for rising edges on push button 1:11		
		Current counter for rising edges on push button 2:13		
Returns for each push button the counter for rising edges. As soon as the module detects a rising edge on a push button, the rising edge counter for the affected push button is incremented by 1.				
RISE PBx	**ASCII READ COMMAND**	#RISE PB<PBNR><CR> #RPB<PBNR><CR> Result: #RPB<PBNR>:<RiseDec>,<RiseHex><CR>	ASCII	
	PBNR	1		
	TX	#RISE PB1<CR>		

	RX	#255,RPB1:11,0xB<CR>		
		Current counter for rising edges on psuh button 1:11		
Returns for push button <PBNR> the counter for rising edges. As soon as the module detects a rising edge on a push button, the rising edge counter for the affected push button is incremented by 1.				
FALL ALL PBS	**ASCII READ COMMAND**	#FALL ALL PBS<CR> #FAPBS<CR> Result: #FAPBS:<FallPB1Dec>,<FallPB2Dec>,<FallPB1Hex>,<FallPB2Hex><CR>	ASCII	
	TX	#FALL ALL PBS<CR>		
	RX	#255,FAPBS:11,13,0xB,0xD<CR>		
		Current counter for falling edges on psuh button 1:11		
		Current counter for falling edges on psuh button 2:13		
Returns for each push button the counter for falling edges. As soon as the module detects a falling edge on a push button, the falling edge counter for the affected push button is incremented by 1.				
FALL PBx	**ASCII READ COMMAND**	#FALL PB<PBNR><CR> #FPB<PBNR><CR> Result: #FPB<PBNR>:<FallDec>,<FallHex><CR>	ASCII	
	PBNR	1		
	TX	#FALL PB1<CR>		
	RX	#255,FPB1:11,0xB<CR>		
		Current counter for falling edges on push button 1:11		
Returns for push button <PBNR> the counter for falling edges. As soon as the module detects a falling edge on a push button, the falling edge counter for the affected push button is incremented by 1.				
RESET COUNTERS PBS	**ASCII WRITE COMMAND**	#RESET COUNTERS PBS<CR> #RCPBS<CR> Result: #OK<CR>	ASCII	YES
	TX	#RESET COUNTERS PBS<CR>		
	RX	N/A		