MCA-527

Digital Multi-Channel Analyzer

Description of the MCA527 Firmware Commands

GBS Elektronik GmbH
Bautzner Landstraße 22
01454 Großerkmannsdorf
Germany

Tel.: 0049 (0)351 21 70 07 - 0
Fax: 0049 (0)351 21 70 07 - 21
E-Mail: kontakt@gbs-elektronik.de
Website: www.gbs-elektronik.de

Exclusion of liability

The information in this document has been carefully reviewed and is believed to be accurate and reliable. However, the GBS Elektronik GmbH assumes no liabilities for inaccuracies in this manual. This manual is subject to change without notice.

Last update: 2021-09-20

Address:
GBS-Elektronik GmbH
Bautzner Landstraße 22
01454 Großerkmannsdorf
Tel.: (0351) 217007-0
Fax: (0351) 217007-21

Internet: http://www.gbs-elektronik.de
Email to: kontakt@gbs-elektronik.de

ATTENTION! This description is subject to change. The current one refers to MCA527 firmware version 21.00.

Introduction

The MCA527 provides all MCA166 firmware functions. They are identical or else at least compatible to the MCA166 firmware functions. But there are also entirely new firmware functions. Older software applications that use only the MCA166 firmware functions are still usable to operate the MCA527. However, they waste the new capabilities of the MCA527.

Since the MCA527 can be operated by different interfaces (RS232, RS485, USB or Ethernet) concurrently, the MCA527 grants an execution right for commands which must not be executable by different applications at the same time.

The execution right is granted to that application that calls first such a command. An application keeps the granted execution right as long as it communicates continuously with the MCA527. It loses the execution right when it does not communicate for more than 15 seconds.

The execution right is valid only for one communication path. For example, if an application owns the execution right for the RS232 interface, it has to release that execution right by interrupting the communication for at least 15 seconds before it or another application can obtain the execution right for the Ethernet.

The communication between sender (computer or microprocessor) and MCA527 runs in the following way. The sender sends a 12 bytes long command. Each command consists of the preamble (0xA5, 0x5A), the command number (2 bytes), the parameters (6 bytes) and the end flag (0xB9, 0x9B). The MCA527 usually responses immediately, but for some commands the response time is up to 1 second. The result data array is embedded between the preamble and the end flag. The preamble is always 0xA5, 0x5A. The end flag informs about the result of the command according the following table.

0xB9,	0x9B:	successful
0xA4	0xAA	timeout[1]
0xA5	0xAA	different baud rates between sender and MCA527[2]
0xA6	0xAA	invalid preamble or end flag[3]
0xA7	0xAA	µSD memory card error
0xA8	0xAA	file writing is presently in process, but this is forbidden for this command
0xA9,	0xAA:	not handled by this firmware version (either generally or customized)
0xAA,	0xAA:	invalid parameter
0xAB,	0xAA:	unknown command
0xAC,	0xAA:	measurement is running, but stopped measurement is required for this command
0xAD,	0xAA:	execution right violation
0xAE,	0xAA:	measurement is stopped, but running measurement is required for this command
0xAF,	0xAA:	wrong mode for using this command (with this parameters)

[1] Since firmware version 15.00. The timeout for the 12 command bytes is 4 milliseconds. If the bytes are not received within this period, the MCA527 returns this error flag. The same happens if the MCA527 receives too many or too few bytes. In previous firmware versions, the MCA527 had returned 'invalid parameter' for this error.

[2] Since firmware version 15.00. Each time the MCA527 detects different baud rates between sender and MCA527, it swaps the baud rate and returns this error flag with the new baud rate. In previous firmware versions, the MCA527 had returned 'invalid parameter' for this error.

[3] Since firmware version 15.00, an invalid preamble or end flag returns this flag. In previous firmware versions, the MCA527 had returned 'invalid parameter' for this error.

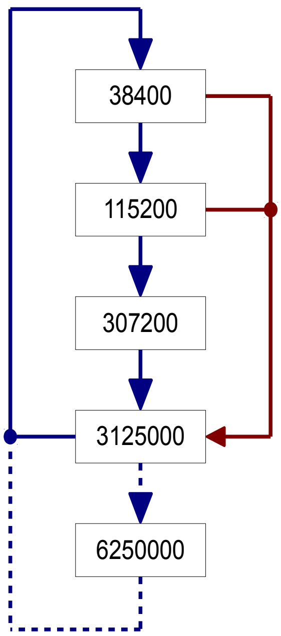
For most commands, 136 bytes are returned (preamble = 2 bytes, result data array = 132 bytes, end flag = 2 bytes). But there are also commands with a larger result data array. However, if a command is unsuccessful, then always only 136 bytes are returned.

If the end flag indicates a successful command, the result data array contains the 8 sent data bytes (without preamble and end flag) at byte offset 106 and a two byte checksum at byte offset 126 of the result data array. These both byte offsets apply for almost all commands (including the commands where the result data array is not described because it contains no further data).

For commands that return more than 136 bytes, the sent data bytes and the checksum are returned at another byte offsets within the result data array. See the description for these commands. The checksum is a simple 16-bit sum over all other returned 2-byte words (without preamble and end flag).

For almost all other commands, the checksum is a simple 16-bit sum over all other returned 2-byte words (including preamble and end flag).

The commands CMD_QUERY_SPECTRA and CMD_QUERY_SPECTRA_EX work according to a different rule. They do not return the 8 sent data bytes, and the checksum is the 16-bit sum over all sent and returned 2-byte words. The checksum is returned at byte offset 130 of the result data array. (This different rule originates from the MCA166. Because all commands of the MCA166 return always 136 bytes, there is no space for returning the 8 sent data bytes.)



The RS232 / USB interface offers the following baud rates: 38 400, 115 200, 307 200, 3 125 000[4] or 6 250 000 (MCA527nano only). The further parameters are 8 data bits, 1 stop bit, no parity, no flow control.

In order to set the baud rate on the MCA527, the sender (computer or microprocessor) starts the communication with the desired baud rate. As long as the MCA527 detects different baud rates between sender and itself, it swaps to the next baud rate according the scheme on the left and responds with the new baud rate.

The MCA527 uses two ways to detect different baud rates between the sender and itself. If it detects a frame error, it swaps to the next baud rate along the blue arrow. If the current baud rate on the MCA527 is 38 400 or 115 200 and the baud rate on the sender is 3 125 000 or 6 250 000, no frame error is triggered. In such a case, however, the MCA527 apparently receives too few bytes. Because of that, it assumes a baud rate difference and swaps to the next baud rate along the red arrow.

If the MCA527 is operated within a bus system (e.g. RS485), other rules are applied. About the special functional principle of the serial interface within bus systems, read the document “Description of the MCA527 Serial Interface”.

The Ethernet communication uses the UDP protocol with the destination port 50 000. The preamble of the returned data are prefixed two additional bytes (0xA5, 0x5A) for intern byte alignment within the MCA527.

The MCA527 can be configured to use a fixed customized IP address or to obtain the IP address automatically from a DHCP server or if not available, to use Zero Configuration Networking (also named Automatic Private IP Addressing).

While the MCA527 is writing a file, only the query commands and CMD_STOP command are executable. All other commands are ignored and response with an error value. The file writing state is returned by CMD_QUERY_STATE527_EX.

[4] The USB circuits built within the MCA527 allows only up to 3 000 000 baud. However, this is not a problem. The MCA527 also communicates properly with 3 125 000 baud if the counterpart does it with 3 000 000 baud.

MCA Reset Command:

Command name	**CMD_INIT**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	All MCA Parameters are reset to their initial state and the spectra are cleared. The measurement is aborted. The preamplifier power and the high voltage are turned off! The command is intended for use at the start of a session, but not for permanently use during the session.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	41	00	00	00	00	00	00	00	B9	9B

MCA Clear Command:

Command name	**CMD_CLEAR**											
Compatibility	Differs to the MCA166 command.											
Execution right	Necessary											
Description	Depending on the parameter: CLEAR_MEASUREMENT_DATA = 0 or 1 CLEAR_ROI = 2 CLEAR_ALL = 3 The command clears the measurement data, the ROI limits exclusively or all together. Before the measurement data are cleared, a running measurement is immediately stopped.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		clear		0				end flag	
Byte string (HEX)	A5	5A	44	00	clear	00	00	00	00	00	B9	9B

MCA Save State Command:

Command name	**CMD_SAVE_MCA_STATE**											
Compatibility	New MCA527 command (since firmware version 15.07).											
Execution right	Necessary											
Description	Depending on the option: = 0: Deletes only the previously saved MCA state. ≠ 0: Saves the current MCA state. The command is only available for the OEM, Micro and Nano version. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		option		0				end flag	
Byte string (HEX)	A5	5A	37	01	option_l	option_h	00	00	00	00	B9	9B
Remarks	In Full and Lite version, the current MCA state is saved on the internal EEPROM at shutdown. In OEM (without power module[5]) and Micro version, an orderly shutdown is not possible because the power supply may be disconnect suddenly and without warning. For this reason, this command has been added. If the MCA state needs to be saved for the next session, the command has to be executed before the MCA527 is disconnected from the power supply. The MCA state remains saved within the EEPROM until it is explicitly deleted or overwritten. Even if the MCA527 is reset by CMD_INIT, the EEPROM is not erased.											

[5] Since firmware version 18.03, the OEM version is able to operate an optional power module that allows an orderly shutdown. If the MCA527 OEM is assembled with a power module, it works like the Full and Lite version and this command is not handled.

MCA Mode Commands:

Command name	**CMD_SET_GENERAL_MODE**											
Compatibility	New MCA527 command											
Execution right	Necessary											
Description	The MCA527 is not only usable as a MCA. It can also alternatively be used as an oscilloscope, as a transient recorder or in a list mode. Depending on the parameter, the MCA527 serves as: 0 = MCA 1 = Transient recorder (records ADC raw data[6]) 2 = Oscilloscope 3 = List mode 1: Time stamp recorder (level triggered signals)[7,8] 4 = List mode 2: Time stamp recorder (edge triggered signals)[8] 5 = List mode 3: Time stamp recorder (AHRC[9] = analog high rate counting)[8] 6 = List mode 4[10]: List with spectroscopic and time information 7 = List mode 5[11]: List with spectroscopic data pairs measured with two different flat top times for the evaluation filter The command sets the MCA527 to the required general mode. Previously acquired data are cleared. The command is ignored and responded with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		mode		0				end flag	
Byte string (HEX)	A5	5A	05	01	mode	00	00	00	00	00	B9	9B
Remarks	The oscilloscope mode is described in the document "Description of the MCA527 Oscilloscope Mode".											

Command name	**CMD_SET_MODE**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	Depending on the parameter: MODE_MCA = 0 or MODE_MCS = 1, the command sets the MCA527 to the required mode by activating the previously specified setup parameters. The command is ignored and responds with an error value if a measurement is still running or if it is tried to set the mode to MODE_MCS when the gating mode 'sort by time' (see CMD_SET_GATING) is set.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		mode		0				end flag	
Byte string (HEX)	A5	5A	45	00	mode	00	00	00	00	00	B9	9B

[6] Including the overflow (bit 14) and the gate input (bit 15).

[7] Note, there are hardware modifications without offset DAC and without hardware-based coarse gain. In such a case, the general mode 3 works only properly if the input signal is adjusted by default for this mode.

[8] The time stamp recorders are no standard components of all firmware versions. The availability of the time stamp recorders is indicated by the corresponding flag within the parameter "MCA features" (byte offset 8) returned by CMD_QUERY_STATE527.

[9] See the remarks on CMD_SET_AHRC_PARAM for more information.

[10] Since firmware version 20.00. The list mode 4 is no standard component of all firmware versions. The availability of it is indicated by the corresponding flag within the parameter "MCA features" (byte offset 8) returned by CMD_QUERY_STATE527.

[11] Since firmware version 16.00. The list mode 5 is no standard component of all firmware versions. The availability of it is indicated by the corresponding flag within the parameter "MCA features" (byte offset 8) returned by CMD_QUERY_STATE527.

MCA Acquire Commands:

Command name	**CMD_START**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The MCA527 distinguishes different general modes (see CMD_SET_GENERAL_MODE). The command behaves accordingly to the general mode. MCA: The data acquisition is started depending on the flags. In repeat modes, the previous data are always cleared and the passed start time is used for the first sweep and is incremented internally by firmware once a second to be used for the following sweeps. Flags: 0 = Continues the previous data acquisition. The start time is ignored. 1 = All previous data are cleared and a new data acquisition is started. 2 = Repeat mode 1[12] (MCA/MCS) 3 = Repeat mode 2[13] (MCA) 4 = Repeat mode 3[14] (MCA) 5 = Repeat mode 4[15] (MCS) 6 = Repeat mode 5[16] (MCA/MCS) 7 = Repeat mode 6[17] (MCA) 8 = Repeat mode 7[18] (MCS) Bit 15: trigger1: 0 = inactive, 1 = active [19] [20] Bit 14: trigger2: 0 = inactive, 1 = active Bit 13: trigger3: 0 = inactive, 1 = active Bit 12: repeated triggering[21]: 0 = inactive, 1 = active Start time: Seconds since December 31, 1969, 16:00:00 GMT[22] Other general modes: The command starts the data acquisition if it is not already running. The flags are ignored. Except for the oscilloscope mode, the start time is taken over.												
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		flags		start time				end flag	
Byte string (HEX)	A5	5A	42	00	flags$_l$	flags$_h$	st$_l$	..	..	st$_h$	B9	9B
Remarks	Repeat modes are only allowed in MCS mode or in MCA mode with measurement stop condition equal to PRESET_REAL or PRESET_REAL_MILLISECONDS (see CMD_SET_PRESETS). If the conditions are not kept, the command is ignored and responded with an error value											

[12] Measurement will be stopped if buffer overruns. In MCS mode a differential amplitude spectrum per sweep is collected.

[13] Measurement will be suspended if buffer overruns, and will be resumed, after the buffer is read out.

[14] MCA166: Buffer will be overwritten, even if not read out.
MCA527: Buffer will be overwritten, if it is unlocked, otherwise the MCA will be suspended.

[15] MCS Repeat Mode with one integral amplitude spectrum for all MCS sweeps. (MCA166: not for MCS Input TTL)

[16] Similar to repeat mode 1, but the data are saved on the intern microSD card instead of the buffer.

[17] Similar to repeat mode 2, but the data are saved on the intern microSD card instead of the buffer.

[18] Similar to repeat mode 4, but the data are saved on the intern microSD card instead of the buffer.

[19] The measurement can be triggered by external signals on the extension port pins. The extension port is available at the full, OEM and micro version of the MCA527. The pins have to be configured with CMD_SET_EXTENSION_PORT and CMD_SET_EXTENSION_POLARITY. The valid options for the extension port pins are 'Trigger' and 'Input'. If the extension port pin is misconfigured or even missing, the corresponding bits are ignored.

[20] The trigger bits are available since firmware version 17.00.

[21] In firmware repeat mode 2 and 6, each sub-measurement is discretely triggered if this bit is set.

[22] This parameter is usually used in this way, but it can be also used in any other way. The start time based on this parameter is returned by CMD_QUERY_STATE, byte offset 100.

Command name	**CMD_START_NEW_SWEEP**											
Compatibility	New MCA527 command (since firmware version 16.00).											
Execution right	Necessary											
Description	The command stops the current sweep, starts a new sweep without any time delay and returns the measuring data of the just stopped sweep. The command is an alternative to the repeat mode, which basically does the same at preset intervals. The command requires the following conditions: general mode = 'MCA', mode = 'MCA', automatic stop condition (preset) = 'none' and the measurement must be run. If any of the conditions is not met, the command is ignored and responded with an error value. It is essential to check the returned value 'Buffer state', because if the buffer is still occupied[23] by previous measuring data, it can not yet be used for the next one. If this happens, the command returns without stopping the current and starting a new sweep. In this case, the command must be repeated.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	39	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	unused						2 bytes					
Byte offset 2	Detected counts						48 bit integer					
Byte offset 8	unused						2 bytes					
Byte offset 10	Counts above the spectrum range						48 bit integer					
Byte offset 16	Start time						unsigned long					
Byte offset 20	Real time [s]						unsigned long					
Byte offset 24	Fractional digits of the real time [msec]						unsigned short					
Byte offset 26	unused						2 bytes					
Byte offset 28	Dead time [ms]						unsigned long					
Byte offset 32	Fast dead time [ms]						unsigned long					
Byte offset 36	PUR counter						unsigned long					
Byte offset 40	Battery current [mA]						unsigned long		At the MCA527Micro this data corresponds to the USB input voltage.			
Byte offset 44	HV primary current [mA]						unsigned long					
Byte offset 48	+12V primary current [mA]						unsigned long					
Byte offset 52	-12V primary current [mA]						unsigned long					
Byte offset 56	+24V primary current [mA]						unsigned long					
Byte offset 60	-24V primary current [mA]						unsigned long					
Byte offset 64	Battery voltage [mV]						unsigned long		At the MCA527Micro this data corresponds to the USB input voltage.			
Byte offset 68	HV [* 1.2 V]						unsigned long					

[23] The firmware internally deals with three memory buffers. One buffer is being used for the current measuring data, one buffer is containing the directly previous measuring data and one buffer is being erased to use it for the next measuring data. Occupied means the buffer for the next measuring data has not yet be fully erased.

Command name	**CMD_START_NEW_SWEEP** *(Continuation)*		
Result data array			
Byte offset 72	+12V actual value [* 0.0625 V]	unsigned char	
Byte offset 73	-12V actual value [* 0.0625 V]	unsigned char	
Byte offset 74	+24V actual value [* 0.125 V]	unsigned char	
Byte offset 75	-24V actual value [* 0.125 V]	unsigned char	
Byte offset 76	Voltage on SUB-D9 pin3 [* 0.3125 mV]	unsigned short	relevant only for MCA527 full version
Byte offset 78	Voltage on SUB-D9 pin5 [* 0.3125 mV]	unsigned short	relevant only for MCA527 full version
Byte offset 80	Charger current [mA]	unsigned long	
Byte offset 84	Extension port counter 1	unsigned long	
Byte offset 88	Extension port counter 2	unsigned long	
Byte offset 92	Extension port counter 3	unsigned long	
Byte offset 96	MCA temperature [* 0.007 812 5 °C]	short	0x8000 = not available
Byte offset 98	Detector temperature [* 0.007 812 5 °C][24]	short	0x8000 = not available
Byte offset 100	Power module temperature [* 0.007 812 5 °C]	short	0x8000 = not available
Byte offset 102	Additional temperature 1 [* 0.007 812 5 °C][25]	short	0x8000 = not available
Byte offset 104	Additional temperature 2 [* 0.007 812 5 °C][26]	short	0x8000 = not available
Byte offset 106	Command flag and parameters	8 bytes	

Byte offset 114	Buffer state	unsigned short	OCCUPIED = 0x2000
			OVERRUN = 0x4000
			FILLED = 0x8000

Byte offset 116	unused	10 bytes	
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	unused	2 bytes	
Remarks	Unlike the repeat mode, the command does not consider whether the data buffer is locked by CMD_QUERY_SPECTRA, CMD_QUERY_SPECTRA_EX, CMD_QUERY_SPECTRA_EX2 or CMD_QUERY_EXTENSION_RS232_RX.		

[24] *Full version:* This value comes from the one-wire interface if a corresponding temperature sensor is connected.
Micro and OEM version: The value from an additional external temperature sensor connected to the two-wire interface with the address 1001 0100 is assumed as the detector temperature. This sensor must be a TMP102 (Micro) or LM73 (OEM).

[25] Additional external temperature sensor at the two-wire interface with address 1001 0000. The sensor must be a TMP102 (Micro) or LM73 (OEM).

[26] Additional external temperature sensor at the two-wire interface with address 1001 0110 (Micro) or 1001 1000 (OEM). The sensor must be a TMP102 (Micro) or LM73 (OEM).

Command name	**CMD_STOP**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	In MCA mode the measurement is stopped at the next integer real time, otherwise immediately											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	43	00	00	00	00	00	00	00	B9	9B

MCA Setup Commands:

Command name	**CMD_SET_ADC_RES_DISCR**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the ADC resolution (res: 128, 256 … 16384), the LLD (Low level discriminator) and the ULD (Upper level discriminator) to the values of its three parameter. The maximum resolution is returned by CMD_QUERY_STATE527, byte offset 56. The LLD must be smaller than the ULD. The maximum ULD is equal ADC resolution minus 1. The command is ignored and responded with an error value if one or more parameter are invalid or a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		res		LLD		ULD		end flag	
Byte string (HEX)	A5	5A	46	00	res$_l$	res$_h$	LLD$_l$	LLD$_h$	ULD$_l$	ULD$_h$	B9	9B

Command name	**CMD_SET_PRESETS**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the automatic stop condition (pre) to PRESET_NONE = 0 (val is irrelevant) PRESET_REAL = 1 PRESET_LIVE = 2 (val ≤ 2 000 000)[27] PRESET_INT = 3 PRESET_AREA = 4 PRESET_REAL_MILLISECONDS[28] = 5 and value (val) of the preset.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		pre		val				end flag	
Byte string (HEX)	A5	5A	48	00	pre$_l$	pre$_h$	val$_l$	...	...	val$_h$	B9	9B
Remarks	Except for PRESET_LIVE[27] and PRESET_REAL_MILLISECONDS, the MCA527 stops the measurement always at integer real time. For the list modes 1 to 5 (general modes 3 to 7, see CMD_SET_GENERAL_MODE), only PRESET_REAL is supported. PRESET_LIVE, PRESET_INT and PRESET_AREA are without effect and mean the same like PRESET_NONE.											

[27] Up to firmware version 13.03, the measurement was stopped always at integer seconds real time. Since firmware version 13.04, if preset is equal to PRESET_LIVE, the measurement is stopped according to the preset live time and the resulting real time is no longer integer. The fractional digits are returned by CMD_QUERY_STATE527_EX, byte offset 80. Besides, in consequence of the more exact automatic stop, the maximum value for PRESET_LIVE was reduced to 65 535. Since firmware version 18.00, the maximum value for PRESET_LIVE is 2 000 000. This is practicable, but with dead times greater than 53 percent, it can be that the measurement is stopped prematurely.

[28] Since firmware version 14.03.

Command name	**CMD_SET_ROI**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the begin and end of the ROI for the preset integral and area. (LLD <= begin < end and LLD < end <= ULD)											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		beg		end				end flag	
Byte string (HEX)	A5	5A	49	00	beg_l	beg_h	end_l	end_h	00	00	B9	9B

Command name	**CMD_SET_REPEAT**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the number (rep: 0 ... 65 535) of sweeps for repetitive measurement. Rep = 0 means infinite repetitions. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		rep		0				end flag	
Byte string (HEX)	A5	5A	4A	00	rep_l	rep_h	00	00	00	00	B9	9B

Command name	**CMD_SET_MCS_CHANNEL**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the number of channels (ch: 1 ... 16 384) for MCS mode. The command is ignored and responds with an error value if a measurement is still running.											
Command syntax												
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		ch		0				end flag	
Byte string (HEX)	A5	5A	63	00	ch_l	ch_h	00	00	00	00	B9	9B

Command name	**CMD_SET_TIME_PER_CHANNEL**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the dwell time per channel (tpc: 1 ... 65 535) for the MCS mode. The dwell time is tpc * 10 ms. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		tpc		0				end flag	
Byte string (HEX)	A5	5A	4B	00	tpc_l	tpc_h	00	00	00	00	B9	9B
Remarks	See CMD_SET_TIME_PER_CHANNEL527 (next command).											

Command name	**CMD_SET_TIME_PER_CHANNEL527**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the dwell time per channel (tpc: 1 … 42 949 672[29]) for the MCS mode. The dwell time is tpc * 0.1 ms. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		long				integer		integer	
Parameter	preamble		command		tpc				0		end flag	
Byte string (HEX)	A5	5A	15	01	tpc$_l$	...	...	tpc$_h$	00	00	B9	9B
Remarks	This command has been added for setting the dwell time with higher resolution.											

Command name	**CMD_SET_GAIN**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the amplifier coarse(cg) and fine gain(fg). cg: 2, 5, 10, 20, 50, 100, 200, 500 or 1 000 fg: 5 000 ... 65 000 (corresponds to 0.5 ... 6.5)											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		cg		fg		0		end flag	
Byte string (HEX)	A5	5A	4C	00	cg$_l$	cg$_h$	fg$_l$	fg$_h$	00	00	B9	9B
Remarks	Not all coarse gain levels are realized by hardware. Some coarse gain levels are realized by firmware. The information that coarse gain levels are really hardware-based is returned by CMD_QUERY_STATE527, byte offset 125.											

Command name	**CMD_SET_OFFSET_DAC**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the offset DAC[30]. DAC: 0 … 16 383											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		dac		0				end flag	
Byte string (HEX)	A5	5A	0A	01	dac$_l$	dac$_h$	00	00	00	00	B9	9B

[29] As from firmware version 13.04, the upper bound has been reduced in favor of other requirements.

[30] There are hardware modifications without offset DAC. In such a case, the command has no effect.

Command name	**CMD_SET_INPUT_POLARITY**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the input polarity for the internal amplifier. ip = 0 positive input signals ip = 1 negative input signals The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		ip		0				end flag	
Byte string (HEX)	A5	5A	56	00	ip	00	00	00	00	00	B9	9B

Command name	**CMD_SET_MCA_INPUT**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the input to shaping or to direct. ip = 0 input shaping (equivalent to 'input amplifier' of the MCA166) ip = 3 input direct (0 … +3V, no PUR, obsolete, for compatibility with older applications) ip = 4 input direct (0 … -3V, no PUR, obsolete, for compatibility with older applications) ip = 5 input direct (**P**ulse **P**eak **A**nalysis)[31] [32] ip = 6 input direct (**P**ulse **I**ntegral **A**nalysis)[31] [33] The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		ip		0				end flag	
Byte string (HEX)	A5	5A	54	00	ip	00	00	00	00	00	B9	9B

Command name	**CMD_SET_MCS_INPUT**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the input for the Multi-Channel Scaler. ip = 0 external TTL signals[34] ip = 1 internal count rate signal ip = 2 counts only pulses with amplitude between LLD and ULD threshold The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		ip		0				end flag	
Byte string (HEX)	A5	5A	55	00	ip	00	00	00	00	00	B9	9B

[31] Polarity according to CMD_SET_INPUT_POLARITY, gain according to CMD_SET_GAIN, no PUR

[32] Previously only named “input direct”.

[33] Since firmware version 21.00.

[34] Note, there are hardware modifications without offset DAC and without hardware-based coarse gain. In such a case, this MCS input mode works only properly if the input signal is adjusted by default for this mode.

Command name	**CMD_SET_THRESHOLD**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	This command sets the threshold value (thr: 0 … 60 percent).											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		thr		0				end flag	
Byte string (HEX)	A5	5A	47	00	thr	00	00	00	00	00	B9	9B
Remarks	See next command.											

Command name	**CMD_SET_THRESHOLD_TENTHS**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	This command sets the threshold value (thr: 0 … 600 * 0.1 percent).											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		thr		0				end flag	
Byte string (HEX)	A5	5A	0D	01	thr_l	thr_h	00	00	00	00	B9	9B
Remarks	This command has been added for setting the threshold with higher resolution.											

Command name	**CMD_SET_SHAPING_TIME**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the shaping time (dtc). dtc = 1 shaping time low dtc = 3 shaping time high The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		dtc		0				end flag	
Byte string (HEX)	A5	5A	52	00	dtc	00	00	00	00	00	B9	9B
Remarks	The MCA527 provides more than two shaping times, but in order to keep compatibility with older software, this command has been retained. The command allows furthermore to chose the shaping time from a pair of shaping times. This pair can be set by CMD_SET_SHAPING_TIME_PAIR.											

Command name	**CMD_SET_SHAPING_TIME_PAIR**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the shaping time pair.(lst = 1 … 254 * 0.1 µs, hst = 2 … 255 * 0.1 µs, lst<hst). The highest allowed shaping time is returned by CMD_QUERY_STATE527, byte offset 121. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		lst		hst		0		end flag	
Byte string (HEX)	A5	5A	0C	01	lst	00	hst	00	00	00	B9	9B
Remarks	See also previous command.											

Command name	**CMD_SET_TRIGGER_FILTER**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the trigger filters used for low and high shaping time (tfl = 0 … 5, tfh = 0 … 5). 0 = (-1; +1) 1 = (-1; 0; +1) 2 = (+1; -2; +1) 3 = (+1; 0; -2; 0; +1) 4 = (4* -1; 12* 0; 4* +1) 5 = (4* +1; 4* 0; 4* -2; 4* 0; 4* +1) The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		tfl		tfh		0		end flag	
Byte string (HEX)	A5	5A	03	01	tfl	00	tfh	00	00	00	B9	9B
Remarks	The trigger filter availability flags[35] (see CMD_QUERY_STATE527_EX, byte offset 88) mark the available trigger filters. If a non-available trigger filter is tried to set, the command responds with an error value.											

Command name	**CMD_SET_TRIGGER_PARAM**											
Compatibility	New MCA527 command (since firmware version 13.00).											
Execution right	Necessary											
Description	param: parameter to be set 0 = Coefficient for automatic threshold calculation (see CMD_QUERY_STATE527, 78) 1 = Coefficient for automatic threshold calculation for direct input (see CMD_QUERY_STATE527_EX, 98) 2 = Fixed trigger threshold (see CMD_QUERY_STATE527, 116) 3 = Fixed trigger threshold for direct input[36] (see CMD_QUERY_STATE527_EX3, 0) 4 = Fixed baseline for direct input[36] (see CMD_QUERY_STATE527_EX3, 2) value: new value (for format and allowed range, see indicated data result array) The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		param		value				end flag	
Byte string (HEX)	A5	5A	06	01	p_l	p_h	value$_l$	...	...	value$_h$	B9	9B

[35] Since firmware version 12.00.
[36] Since firmware version 20.01.

Command name	**CMD_SET_EVAL_FILTER_TYPE**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the evaluation filter type (eft = 0 or 1) 0 = standard filter 1 = LF rejection filter 2 = slow rise filter[37] The command is not standardly handled. It is only handled if the flag FEATURES_LF_REJECTION is set within the parameter “MCA features” that is returned by CMD_QUERY_STATE527, byte offset 8, or if the flag FEATURES2_SLOW_RISE is set within the parameter “MCA features (2nd group)” that is returned by CMD_QUERY_STATE527_EX2, byte offset 120. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		eft		0				end flag	
Byte string (HEX)	A5	5A	14	01	eft	00	00	00	00	00	B9	9B

Command name	**CMD_SET_FLAT_TOP_TIME**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the flattop times (ftt = 0 … 255 * 0.1 µs, ftt2 = 0 … 255 * 0.1 µs, ftt2≤ftt). Until firmware version 13.06 the highest allowed flattop time was equal to 50. Since firmware version 13.07 the highest allowed flattop time is returned by CMD_QUERY_STATE527_EX, byte offset 33. The second flattop time has been added in firmware version 16.00. It is used for special optional evaluation routines. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		ftt		ftt2				end flag	
Byte string (HEX)	A5	5A	13	01	ftt	00	ftt2	00	00	00	B9	9B

Command name	**CMD_SET_JITTER_CORRECTION**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the jitter correction. jc = 0 turns jitter correction off jc ≠ 0 turns jitter correction on The command is not standardly handled. It is only handled if the flag FEATURES_JITTER_CORRECTION is set within the value “MCA features” that is returned by CMD_QUERY_STATE527, byte offset 8. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		jc		0				end flag	
Byte string (HEX)	A5	5A	16	01	jc	00	00	00	00	00	B9	9B

[37] Since firmware version 19.00

Command name	**CMD_SET_BASELINE_RESTORING**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the baseline restorer. blr = 0 off blr = 1 1/1 blr = 2 1/2 blr = 3 1/4 blr = 4 1/8 blr = 5 1/16 blr = 6 1/32 The command is not standardly handled. It is only handled if the flag FEATURES_ADJUSTABLE_BASELINE_RESTORER is set within the value “MCA features” that is returned by CMD_QUERY_STATE527, byte offset 8. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		blr		0				end flag	
Byte string (HEX)	A5	5A	18	01	blr	00	00	00	00	00	B9	9B

Command name	**CMD_SET_PUR**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the pile up rejection. pur = 0 turns PUR off pur ≠ 0 turns PUR on The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		pur		0				end flag	
Byte string (HEX)	A5	5A	53	00	pur	00	00	00	00	00	B9	9B

Command name	**CMD_SET_MINIMUM_EVENT_DISTANCE**											
Compatibility	New MCA527 command (since firmware version 19.03).											
Execution right	Necessary											
Description	This command sets the minimum event distance (0 ... 5000 *100 nsec, default 0). The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		med		0				end flag	
Byte string (HEX)	A5	5A	3B	01	med_l	med_h	00	00	00	00	B9	9B
Remarks	This parameter is only relevant if the pile up rejection (PUR) is on. There is always a time interval between two events. This means, subsequent events (signal impulses) that occur before the expiration of this time interval are not evaluated. Normally, this time interval is dependent on the shaping time and the flattop time (2 * shaping time plus flattop time). In some cases it can be useful to extend this time interval to improve the accuracy of the measurement.											

Command name	**CMD_SET_FAST**											
Compatibility	Compatible to the MCA166 command. The command is irrelevant for the MCA527. It has only been retained for compatibility with older software.											
Execution right	Necessary											
Description	This command sets the fast discriminator threshold (0 ... 2499, default 400).											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		fast		0				end flag	
Byte string (HEX)	A5	5A	50	00	fast_l	fast_h	00	00	00	00	B9	9B

Command name	**CMD_SET_SLOW**											
Compatibility	Compatible to the MCA166 command. The command is irrelevant for the MCA527. It has only been retained for compatibility with older software.											
Execution right	Necessary											
Description	This command sets the slow discriminator threshold (0 ... 2499, default 400).											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		slow		0				end flag	
Byte string (HEX)	A5	5A	51	00	slow_l	slow_h	00	00	00	00	B9	9B

Command name	**CMD_SET_MEASURE_PZC**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command performs the Pole Zero Cancellation. pm = 0 set PCZ only pm > 0 set PZC and measure PZC offset pv: 0 ... 2499 PZC control voltage The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		pm		pv		0		end flag	
Byte string (HEX)	A5	5A	58	00	pm_l	pm_h	pv_l	pv_h	00	00	B9	9B
Result data array												
Byte offset 0	unused						106 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						10 bytes					
Byte offset 124	Averaged negative offset of measured input pulses[38] [39]						short					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	Number of measured pulses						unsigned short					
Byte offset 130	Averaged negative offset of measured input pulses (compliant to the MCA166)[39]						short					
Remarks	This command takes at least 800 milliseconds to return, because it runs a measurement of this duration.											

[38] Since firmware version 12.05.

[39] The both values at byte offset 124 and 130 stand in principle for the same parameter. However, the value at byte offset 124 is 32 times larger than the value at byte offset 130. The value at byte offset 124 has been added to achieve a higher precision.

Command name	**CMD_SET_PZC_TIME_OFFSET**											
Compatibility	Compatible to the MCA166 command. The command is irrelevant for the MCA527. It has only been retained for compatibility with older software.											
Execution right	Necessary											
Description	The command sets the time offset for Pole Zero Cancellation. t1: 0 ... 31 Time offset for low shaping time t2: 0 ... 31 Time offset for high shaping time											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		t1		t2		0		end flag	
Byte string (HEX)	A5	5A	60	00	t1	00	t2	00	00	00	B9	9B

Command name	**CMD_SET_GATING**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the parameter (mode, signal and shift) for gating. mode: 0 = none 1 = discard 2 = sort by state 3 = sort by time[40] rejection / trigger signal: 0 = low (mode 1 and 2) / falling edge (mode 3) 1 = high (mode 1 and 2) / rising edge (mode 3) shift: 0 … 255 * 100 nsec (only relevant for mode 'sort by state') The command is ignored and responds with an error value if a measurement is still running or if there is any conflict with other settings (see remarks).											
Format	integer		integer		char	char	char	char	integer		integer	
Parameter	preamble		command		mode	signal	shift	0	0		end flag	
Byte string (HEX)	A5	5A	0F	01	mode	signal	shift	00	00	00	B9	9B
Remarks	According to the signal on the gate input the counts are sorted or discarded. In discard mode, the counts are just discarded. In 'sort by state' mode, the counts are sorted according to the current signal state either into the useful spectrum or into the so-called rejected spectrum. This rejected spectrum is usable for stabilization. This mode is intended for LED stabilized detectors that deliver a signal while the LED is on. In 'sort by time' mode, each (either rising or falling) signal edge restarts a timer. According to the time elapsed since the timer has been started, the counts are sorted into eight different MCA spectra which refer to eight time windows whose width can be adjusted with the command CMD_SET_GATING_TIME_WINDOW_WIDTH. Besides[41], the counts are sorted into one MCS spectrum which can be adjusted with the commands CMD_SET_MCS_CHANNEL and CMD_SET_GATING_TIME_PER_CHANNEL. Counts which occurs between the measurement start and the first relevant signal edge as well as counts which do not apply to any time window are discarded. This gating mode is in conflict with the MCS mode (see CMD_SET_MODE) and with the stabilization using the rejected spectrum (see CMD_SET_STABILISATION). See MCA features returned by the commands CMD_QUERY_STATE527 (byte offset 8) and CMD_QUERY_STATE527_EX2 (byte offset 120) for the availability of gating modes.											

[40] Since firmware version 14.02.

[41] Since firmware version 16.00.

Command name	**CMD_SET_GATING_TIME_WINDOW_WIDTH**											
Compatibility	New MCA527 command (since firmware version 14.02).											
Execution right	Necessary											
Description	The command sets the width of the time windows for the gating mode 'sort by time' (see CMD_SET_GATING). index: 0 … 7 width: 1 … 4 294 966 289 * 100 nsec[42] 0xFFFFFFFF = infinite (until the next gating signal) The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		index		width				end flag	
Byte string (HEX)	A5	5A	32	01	index$_l$	index$_h$	width$_l$	...	...	width$_h$	B9	9B

Command name	**CMD_SET_GATING_TIME_PER_CHANNEL**											
Compatibility	New MCA527 command (since firmware version 16.00).											
Execution right	Necessary											
Description	The command sets the time per channel of the MCS spectrum that is acquired at the gating mode 'sort by time' (see CMD_SET_GATING). Time per channel (tpc): 1 … 65 535 * 100 nsec[42] The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		tpc		0				end flag	
Byte string (HEX)	A5	5A	3A	01	tpc$_l$	tpc$_h$	0	0	0	0	B9	9B

[42] The time of 100 ns refers to the standard ADC sampling rate of 10 MHz. See also CMD_QUERY_SYSTEM_DATA, byte offset 130.

Command name	**CMD_SET_STABILISATION**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the peak stabilization. fl: 0 turns stabilization off 1 stabilization to the current centroid within peak ROI 2 stabilization to the current centroid of the highest peak within the spectrum rb+3<fl<re-3 stabilization to channel fl Bit 15 = 0 use normal spectrum Bit 15 = 1 use rejected spectrum (allowed only if gating mode is set to 'sort by state', see CMD_SET_GATING) rb: Peak ROI begin (LLD<= begin < end) re: Peak ROI end (begin < end <= ULD, (end-begin) < 250)											
Command syntax												
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		fl		rb		re		end flag	
Byte string (HEX)	A5	5A	4D	00	fl_l	fl_h	rb_l	rb_h	re_l	re_h	B9	9B

Command name	**CMD_SET_STAB_PARAM**											
Compatibility	Identical to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the stabilization parameters. st: time interval (1 … 32 767 sec., default 10 sec.) sa: area (default 25 000)											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		st		sa				end flag	
Byte string (HEX)	A5	5A	67	00	st_l	st_h	sa_l	...	...	sa_h	B9	9B

Command name	**CMD_SET_PREAMPLIFIER_POWER**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the preamplifier power (pp). 0X80 -24V on 0X40 +24V on 0X20 -12V on 0X10 +12V on 0XF0 all on 0X00 all off											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		pp		0				end flag	
Byte string (HEX)	A5	5A	4E	00	pp	00	00	00	00	00	B9	9B
Remarks	The Lite version provides no ± 24 V and the OEM and the Micro version provides no preamplifier power at all. The corresponding bits are ignored.											

Command name	**CMD_SET_BIAS**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the detector high voltage (hv) and controls the HV inhibit input (i). hv: 0 V ... maximum allowed high voltage (see CMD_QUERY_STATE527, byte offset 64) i = 0 Inhibit off i = 1 ”Canberra HPGe mode”, HV shut down if inhibit input < 0.5V[43] i = 2 “DSG HPGe mode”, HV shut down if inhibit input < 0.5V[43] i = -1 ”Ortec HPGe mode”, HV shut down if inhibit input ≥ 5V											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		hv		i				end flag	
Byte string (HEX)	A5	5A	4F	00	hv_l	hv_h	i_l	...	...	i_h	B9	9B
Remarks	Not all versions of the MCA527 supplies high voltage. For high voltage supply, the MCA527 must contain a power module. The parameter 'MCA features' (see CMD_QUERY_STATE527, byte offset 8) indicates the availability of a power module. If the command is sent to a MCA527 without power module, the command returns with an error. If the high voltage exceeds the 'maximum allowed high voltage' (see CMD_QUERY_STATE527, byte offset 64), the command returns with an error. Only the full version of the MCA527 provides HV inhibit input. All other MCA527s ignore this parameter. If the connected detector supplies detector information, the MCA527 compares the polarity of the detector with that of the power module. If the polarities are different, all settings are ignored.											

Command name	**CMD_SET_PIN5_CURRENT_SOURCE**											
Compatibility	New MCA527 command											
Execution right	Necessary											
Description	The command switches the current source (cs) on SUB-D9 pin5 on (= 1) or off (= 0).											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		cs		0				end flag	
Byte string (HEX)	A5	5A	19	01	cs	00	00	00	00	00	B9	9B
Remarks	The SUB-D9 pin5 is supported by the full version and partially by OEM version.											

[43] Mode 1 and 2 are identical. The difference is made by reasons of legacy.

Command name	**CMD_SET_TDF**											
Compatibility	Compatible to the MCA166 command. The command is irrelevant for the MCA527. It has only been retained for compatibility with older software.											
Execution right	Necessary											
Description	The command does nothing. It has only been retained for compatibility with older software.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		tdf		0				end flag	
Byte string (HEX)	A5	5A	61	00	tdf_l	tdf_h	0	0	0	0	B9	9B
Remarks	The command is irrelevant for the current firmware version. Possibly, future firmware will use the parameter again.											

Command name	**CMD_SET_UF6_ROIS**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command sets the begin and the end of a ROI used by the other UF6 commands. The name of the command refers to its original usage. r: ROI number (1, 2 or 3) b: ROI begin e: ROI end *Up to firmware version 14.01:* b < e < max. channels number, (e − b) < 512. *Since firmware version 14.02:* b < e < max. channels number If b = 0 and e = 0, the corresponding ROI is reset to *“not used”*.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		r		b		e		end flag	
Byte string (HEX)	A5	5A	64	00	r	00	b_l	b_h	e_l	e_h	B9	9B

Command name	**CMD_SET_USER_DATA**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Necessary											
Description	The command stores any 32 bit value (val) in the MCA memory (256 entries). e: 0 ... 255 entry number											
Format	integer		integer		integer		32 bit				integer	
Parameter	preamble		command		e		val				end flag	
Byte string (HEX)	A5	5A	57	00	e	00	…	...	...	...	B9	9B
Remarks	See also document “Use of MCA User Data Memory by Specific Applications”.											

Command name	**CMD_SET_TIME**									
Compatibility	New MCA527 command.									
Execution right	Necessary									
Description	The command sets the time (t) on the internal clock. Bit 31 … 17: days since January 1, 2008 Bit 16 … 12: hours (0 … 23) Bit 11 … 6: minutes (0 … 59) Bit 5 … 0: seconds (0 … 59)									
Format	integer		integer		32 bit				integer		integer	
Parameter	preamble		command		t				0		end flag	
Byte string (HEX)	A5	5A	04	01	t_l	...	…	t_h	00	00	B9	9B

Command name	**CMD_SET_IP_ADDRESS**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command allows to set a customized IP address. If the IP address is set to 0.0.0.0, the MCA is instructed to obtain the IP address from a DHCP server or if not available, to use Zero Configuration Networking (also named Automatic Private IP Addressing).											
Format	integer		integer		char	char	char	char	integer		integer	
Parameter	preamble		command		ip1	ip2	ip3	ip4	0		end flag	
Byte string (HEX)	A5	5A	0B	01	ip1	ip2	ip3	ip4	00	00	B9	9B

Command name	**CMD_SET_COMMON_MEMORY_FILL_STOP**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the fill stop for the common memory. The MCA acquires data until the preset number of bytes has been written. The common memory is used by the following general modes: 1 = Transient recorder (records ADC raw data) 3 = Time stamp recorder (records the timestamps of level triggered signals) 4 = Time stamp recorder (records the timestamps of edge triggered signals) 5 = Time stamp recorder (AHRC = analog high rate counting) stop: 0 … common memory size (see CMD_QUERY_STATE527_EX)											
Format	integer		integer		long				integer		integer	
Parameter	preamble		command		stop				0		end flag	
Byte string (HEX)	A5	5A	17	01	stop$_l$	...	...	stop$_h$	00	00	B9	9B
Remarks	For the time stamp recorders, this command is not the only way to preset an automatic stop condition. This can also be done with the command CMD_SET_PRESETS.											

Command name	**CMD_SET_TTL_LEVELS**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command allows to change the low level (ll) and the high level (hl) for TTL input signals. The low level must be lower than the high level. The default levels are 0.8V and 2.0V, which are the standard levels for TTL input signals. Micro version[44]: low level = 6 … 23 * 0.1V, high level = 7 … 24 * 0.1V Other versions: low level = 1 … 99 * 0.1V, high level = 2 … 100 * 0.1V The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		ll		hl		0		end flag	
Byte string (HEX)	A5	5A	27	01	ll	00	hl	00	00	00	B9	9B

Command name	**CMD_SET_OSCI_TRIGGER**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command allows to set source index (s), resolution index (r), threshold (t) and position (p) for the oscilloscope mode. s: 0 … 4 r: -5 … +16[45] t: 0 … 16 383 p: 0 … 499 The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.											
Format	integer		integer		char	char	integer		integer		integer	
Parameter	preamble		command		s	r	t		p		end flag	
Byte string (HEX)	A5	5A	11	01	s	r	t_l	t_h	p_l	p_h	B9	9B

Command name	**CMD_SET_CORE_CLOCK**											
Compatibility	New MCA527 command (since firmware version 13.04).											
Execution right	Necessary											
Description	The command allows to set the core clock of the MCA527 processor. Clk: (1 … 6) * 100 MHz Use a value between the minimally recommended and the maximally allowed core clock (see CMD_QUERY_SYSTEM_DATA, byte offset 124 and 125)[46].											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		clk		0		0		end flag	
Byte string (HEX)	A5	5A	2D	01	ckl_l	00	00	00	00	00	B9	9B
Remarks	The default core clock is sufficient in most cases. The change of the core clock is only necessary in exceptional cases. Note, a higher core clock consumes more power.											

[44] Since firmware version 14.05, the command considered the special characteristic of the micro version. If pin AIN- and pin VCM are tied together, a voltage from 0.5 to 2.5V can be applied between pin AIN+ and pin GND.

[45] Since firmware version 17.01. Before this firmware version: - 5 ... 13.

[46] The minimally recommended core clock and the maximally allowed core clock were added in firmware version 14.06 because of the growing number of hardware modifications.

Command name	**CMD_SET_AHRC_PARAM**											
Compatibility	New MCA527 command (since firmware version 14.00).											
Execution right	Necessary											
Description	The command allows to set the AHRC parameters. param: parameter to be set -1 = AHRC trigger threshold (allowed range = 0 … 65 535[47]) 0 … 9 = AHRC bin width [0 … 9] (allowed range = 1 …2 147 483 647[47]) value: new value corresponding to the allowed range The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		param		value				end flag	
Byte string (HEX)	A5	5A	30	01	p_l	p_h	value$_l$	...	...	value$_h$	B9	9B
Remarks	AHRC (Analog High Rate Counting) is a method which is intended for measuring arrangements with several detectors connected in parallel where temporally superposed events are possible. The events are detected with a fixed threshold. An event is recorded when the signal curve exceeds the threshold. From this point on, the differences between the current ADC value and the baseline are summed up until the signal curve falls below the threshold again. This sum is equivalent to the area under the signal curve. Assuming that there is a relation between the area and the number of events, the area is evaluated for detecting potential multiple events. The whole range is subdivided in bins. Bin[0] represents the signals those no valid event is assigned. Bin[1 ... 9] represent signals with the corresponding number of events. In practice, all bins except bin[0] are set to the same width. In order to find out the trigger threshold and the bin widths, an AHRC histogram has to be acquired with CMD_QUERY_AHRC_HISTOGRAM. At first, an applicable trigger threshold has to be determined. If the default value for the trigger threshold is inapplicable, the histogram routine is repeated with various values for the trigger threshold as long as an applicable value is found. If a proper histogram could be acquired, the bin widths can be set according to the visible peaks within the histogram.											

Command name	**CMD_SET_FAST_TRIGGER_INPUT**											
Compatibility	New MCA527 command (since firmware version 20.00)											
Execution right	Necessary											
Description	The command disables (=0) or enables (=1) the fast trigger input. The signal edge can be rising (=0) or falling (=1). The parameter 'edge" only exists since firmware version 20.01.											
Format	integer		integer		byte	byte	long				integer	
Parameter	preamble		command		enable	edge	0				end flag	
Byte string (HEX)	A5	5A	3C	01	enable	edge	00	00	00	00	B9	9B
Remarks	See MCA features returned by CMD_QUERY_STATE527_EX2 (byte offset 120) for the availability of the fast trigger input.											

[47] The given intervals for the parameters only prevent a firmware hangup. They do not guarantee a meaningful result.

MCA Extension Port Commands:

Command name	**CMD_SET_EXTENSION_PORT**
Compatibility	New MCA527 command.
Execution right	Necessary
Description	The command configures the parts of the extension port. It is only available if the MCA527 version supports the extension port (see “MCA features”, CMD_QUERY_STATE527, byte offset 8) otherwise the command will return “not handled”. The extension port consists of up to six configurable parts. The availability of the parts is depending on the MCA527 version. It is returned by CMD_QUERY_STATE527_EX, byte offset 30. The parts can be RS232 interfaces, pulsers, simple outputs, counters, trigger inputs, simple inputs or 5V power supply. This command only determines the purposes of the parts. The commands described below allow special settings for the determined purposes. The command accepts each combination of the following settings with one exception, part A cannot be configured at once with part B and/or part C as RS232 interface. Decoding:

```
#define EXT_PORT_OFF                          0
//                                                   | Full  |  OEM  | Micro | Nano  | Ref.
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_A_RS232                 4   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_A_RS232_BUFFER          5   // |   x   |   x   |   x   |   -   |  (1)
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_B_PULSER_COMMON_START   1   // |   x   |   x   |   x   |   x   |  (2)
#define EXT_PORT_PART_B_PULSER_SEPARATE_START 2   // |   x   |   x   |   x   |   x   |  (3)
#define EXT_PORT_PART_B_OUTPUT                3   // |   x   |   x   |   x   |   x   |
#define EXT_PORT_PART_B_RS232                 4   // |   x   |   x   |   -   |   -   |  (4)
#define EXT_PORT_PART_B_LOOP_THROUGH          4   // |   -   |   -   |   x   |   -   |  (4)
#define EXT_PORT_PART_B_COUNTER               5   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_B_TRIGGER               6   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_B_INPUT                 7   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_B_PSEUDO_PURPOSE        8   // |   -   |   -   |   x   |   -   |  (7)
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_C_COUNTER               1   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_C_TRIGGER               2   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_C_INPUT                 3   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_C_RS232                 4   // |   x   |   x   |   -   |   -   |
#define EXT_PORT_PART_C_RS232_BUFFER          5   // |   x   |   x   |   -   |   -   |  (1)
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_D_PULSER_COMMON_START   1   // |   x   |   x   |   x   |   x   |  (2)
#define EXT_PORT_PART_D_PULSER_SEPARATE_START 2   // |   x   |   x   |   x   |   x   |  (3)
#define EXT_PORT_PART_D_OUTPUT                3   // |   x   |   x   |   x   |   x   |
#define EXT_PORT_PART_D_COUNTER               4   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_D_TRIGGER               5   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_D_INPUT                 6   // |   -   |   -   |   -   |   x   |
#define EXT_PORT_PART_D_PSEUDO_PURPOSE        8   // |   -   |   -   |   x   |   x   |  (7)
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_E_COUNTER               1   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_E_TRIGGER               2   // |   x   |   x   |   x   |   -   |
#define EXT_PORT_PART_E_INPUT                 3   // |   x   |   x   |   x   |   -   |
//---------------------------------------------------+-------+-------+-------+-------+-------
#define EXT_PORT_PART_F_ON                    1   // |   x   |   x   |   -   |   -   |  (5)
#define EXT_PORT_PART_F_ON_AT_START_UP        2   // |   x   |   x   |   -   |   -   |  (6)
#define EXT_PORT_PART_F_PULSER_COMMON_START   6   // |   -   |   -   |   x   |   x   |  (2)
#define EXT_PORT_PART_F_PULSER_SEPARATE_START 7   // |   -   |   -   |   x   |   x   |
#define EXT_PORT_PART_F_OUTPUT                8   // |   -   |   -   |   x   |   x   |
#define EXT_PORT_PART_F_COUNTER               9   // |   -   |   -   |   x   |   x   |
#define EXT_PORT_PART_F_TRIGGER               10  // |   -   |   -   |   x   |   x   |
#define EXT_PORT_PART_F_INPUT                 11  // |   -   |   -   |   x   |   x   |
```

(1) The last 1024 data bytes that have been received up to the measurement stop are stored in the buffer.

(2) The pulser is started or stopped in common with the measurement but it is still possible to start or stop it with CMD_START_EXTENSION_PULSER or CMD_STOP_EXTENSION_PULSER.

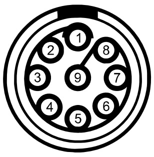
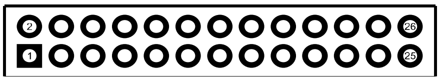
(3) The pulser can only be started or stopped with CMD_START_EXTENSION_PULSER or CMD_STOP_EXTENSION_PULSER.

(4) EXT_PORT_PART_B_RS232 and EXT_PORT_PART_B_LOOP_THROUGH have unfortunately the equal value. That has a historical reason. The meaning must be found out by the MCA527 version.

(5) The power output remains turned on as long as the MCA527 remains turned on or until it is turned off by this command.

(6) The power output will be automatically turned on at power-up of the MCA527.

(7) The pseudo purpose pretends that the pin is usable as extension port. In this way, the driver enable signal is suppressed. If you use this value with this command, corresponding pin is switched off.

Command name	**CMD_SET_EXTENSION_PORT** *(Continuation)*											
Format	integer		integer		char	char	char	char	char	char	integer	
Parameter	preamble		command		a	b	c	d	e	f	end flag	
Byte string (HEX)	A5	5A	1A	01	a	b	c	d	e	f	B9	9B
Remarks	The parts of the extension port are attached to the following pins. Full version OEM version   Part A: pin 1 pin 3 RS232 RX pin 2 pin 7 RS232 TX Part B: pin 4 pin 19 RS232 TX (TTL), pulser2 or output2 Part C: pin 5 pin 21 RS232 RX (TTL), counter2, trigger2 or input2 Part D: pin 7 pin 22 pulser1 or output1 Part E: pin 6 pin 25 counter1, trigger1 or input1 Part F: pin 8 pin 26 5V / 100mA power output Micro version MicroE version Part A: pin X4:RxD pin X4:RxD RS232 RX (TTL) pin X4:TxD pin X4:TxD RS232 TX (TTL) Part B: *not connected* pin X3:OUT1 pulser2 or output2[48] Part C: *not connected* pin X3:IN1 counter2, trigger2 or input2[49] Part D: *not connected* pin X3:OUT2 pulser1 or output1[49] Part E: *not connected* pin X9:IN2 counter1, trigger1 or input1[49] Part F: pin X4:GPIO pin X4:GPIO pulser3, output3, counter3, trigger3 or input3 Nano version Part B: pin X3:GPIO2 pulser2, output2, counter2, trigger2 or input2 Part D: pin X3:GPIO1 pulser1, output1, counter1, trigger1 or input1 Part F: pin X1:GPIO3[49] pulser3, output3, counter3, trigger3 or input3 At the Micro, MicroE or Nano version, the parts must be declared as available either by a potential power module or with the aid of the MCA-527 Basic State Configurator[50].											

[48] Attention! There is a confusion between hardware and firmware naming.

[49] This pin is coincident with fast trigger pin.

[50] The program can be downloaded from our software download page.

Command name	**CMD_SET_EXTENSION_PULSER_PERIOD**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the pulser period (p) for the parts (part) of the extension port. part: 1 = part B (pulser2) 3 = part D (pulser1) 5 = part F (pulser3) p: The pulser period must be larger than the pulser width (see next command). If the parameter does not meet this condition, the command will return “invalid parameter”. The setting range is between 2 … 4 294 967 295 * 10 ns. For the pulser 2[51] of the full, OEM and micro version, the setting range is between 2 … 4 294 967 * 10 µs. The command is ignored and responds with an error value if the corresponding pulser is still running. For the parts that can be used alternatively as a counter, the counter must also be deactivated.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		part		p				end flag	
Byte string (HEX)	A5	5A	1C	01	part	00	p_l	...	...	p_h	B9	9B

Command name	**CMD_SET_EXTENSION_PULSER_WIDTH**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the pulser width (w) for the parts (part) of the extension port. part: 1 = part B (pulser2) 3 = part D (pulser1) 5 = part F (pulser3) w: The pulser width must be smaller than the pulser period (see previous command). If the parameter does not meet this condition, the command will return “invalid parameter”. The setting range is between 1 … 4 294 967 294 * 10 ns. For the pulser 2[51] of the full, OEM and micro version, the setting range is between 1 … 4 294 966 * 10 µs. The command is ignored and responds with an error value if the corresponding pulser is still running. For the parts that can be used alternatively as a counter, the counter must also be deactivated.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		part		w				end flag	
Byte string (HEX)	A5	5A	1D	01	part	00	w_l	...	...	w_h	B9	9B

[51] For full, OEM and micro version, the pulsers work differently. Pulser 1 and 3 work very quickly and exactly because they are realized in hardware. Pulser 2, however, is realized in firmware, wherefore it is less quickly and exactly, and it occupies additional system resources. For this reasons pulser 1 and 3 should be preferred. (For nano version, all pulsers are realized in hardware.

Command name	**CMD_SET_EXTENSION_POLARITY**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the polarity (pol) for the parts (part) of the extension port. part: 1 = part B 2 = part C 3 = part D 4 = part E 5 = part F[52] pol: 0 = positive or rising edge 1 = negative or falling edge											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		part		pol		0		end flag	
Byte string (HEX)	A5	5A	1B	01	part	00	pol	00	00	00	B9	9B

Command name	**CMD_SET_EXTENSION_RS232**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command configures the RS232 interface of the extension port. div: 0x0001 … 0xFFFF (baud rate = 6 250 000 / div, see predefined constants in mca_comm.h) flags: bits 1, 0: 00 5-bit word 01 6-bit word 10 7-bit word 11 8-bit word bit 2: 0 1 stop bit 1 2 stop bits for non-5-bit word length or 1½ stop bits for 5-bit word length bit 3: 0 parity not transmitted or checked 1 transmit and check parity bit 4: 0 Odd parity 1 Even parity											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		div		flags		0		end flag	
Byte string (HEX)	A5	5A	1E	01	div$_l$	div$_h$	flags	00	00	00	B9	9B

Command name	**CMD_CLEAR_EXTENSION_RS232_TX**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command clears all bytes which has been written to the RS232 transfer buffer.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	1F	01	00	00	00	00	00	00	B9	9B

[52] Since firmware version 15.06. Micro version only.

<table>
<tr><td>Command name</td><td colspan="12"><b>CMD_WRITE_EXTENSION_RS232_TX_ASCII</b></td></tr>
<tr><td>Compatibility</td><td colspan="12">New MCA527 command.</td></tr>
<tr><td>Execution right</td><td colspan="12">Necessary</td></tr>
<tr><td>Description</td><td colspan="12">The command is intended for transferring strings via the RS232 interface. Each call writes up to 6 characters to the end of the RS232 transfer buffer. A zero character terminates the string and starts the transfer. The transfer buffer is 300 bytes long. If it is full, the transfer is started automatically.
Before starting a new string transfer, it is recommendable to check the RS232 transfer buffer byte count (see CMD_QUERY_STATE527_EX, byte offset 80). If the transfer buffer is not empty, it can be cleared with CMD_CLEAR_EXTENSION_RS232_TX. It is also possible to clear the transfer buffer preventively without checking the byte count.</td></tr>
<tr><td>Format</td><td colspan="2">integer</td><td colspan="2">integer</td><td>char</td><td>char</td><td>char</td><td>char</td><td>char</td><td>char</td><td colspan="2">integer</td></tr>
<tr><td>Parameter</td><td colspan="2">preamble</td><td colspan="2">command</td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td colspan="2">end flag</td></tr>
<tr><td>Byte string (HEX)</td><td>A5</td><td>5A</td><td>20</td><td>01</td><td>c1</td><td>c2</td><td>c3</td><td>c4</td><td>c5</td><td>c6</td><td>B9</td><td>9B</td></tr>
</table>

<table>
<tr><td>Command name</td><td colspan="12"><b>CMD_WRITE_EXTENSION_RS232_TX_BINARY</b></td></tr>
<tr><td>Compatibility</td><td colspan="12">New MCA527 command.</td></tr>
<tr><td>Execution right</td><td colspan="12">Necessary</td></tr>
<tr><td>Description</td><td colspan="12">The command is intended for transferring bytes via the RS232 interface. Each call writes up to 4 bytes to the end of the RS232 transfer buffer.
flags: bits 2...0: 000 write 0 bytes
001 write 1 byte
010 write 2 bytes
011 write 3 bytes
100 write 4 bytes
bit 7: 1 = start transfer
The transfer buffer is 300 bytes long. If it overruns, it will be cleared and the command will be return “invalid parameter”.
Before starting to fill the transfer buffer with new data, it is recommendable to check the RS232 transfer buffer byte count (see CMD_QUERY_STATE527_EX, byte offset 80). If the transfer buffer is not empty, it can be cleared with CMD_CLEAR_EXTENSION_RS232_TX. It is also possible to clear the transfer buffer preventively without checking the byte count.</td></tr>
<tr><td>Format</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td><td>byte</td><td>byte</td><td>byte</td><td>byte</td><td colspan="2">integer</td></tr>
<tr><td>Parameter</td><td colspan="2">preamble</td><td colspan="2">command</td><td colspan="2">flags</td><td>b1</td><td>b2</td><td>b3</td><td>b4</td><td colspan="2">end flag</td></tr>
<tr><td>Byte string (HEX)</td><td>A5</td><td>5A</td><td>21</td><td>01</td><td>flags</td><td>00</td><td>b1</td><td>b2</td><td>b3</td><td>b4</td><td>B9</td><td>9B</td></tr>
</table>

Command name	**CMD_START_EXTENSION_PULSER**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command starts the selected pulser. The command will only be executed successfully if the corresponding parts have been correctly configured (see CMD_SET_EXTENSION_PORT). part: 1 = part B (pulser2) 3 = part D (pulser1) 5 = part F (pulser3) 7 = part B (pulser2) and part D (pulser1) 9 = part B (pulser2) and part F (pulser3) 11 = part D (pulser1) and part F (pulser3) 255 = all parts											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		part		0				end flag	
Byte string (HEX)	A5	5A	22	01	part	00	00	00	00	00	B9	9B

Command name	**CMD_STOP_EXTENSION_PULSER**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command stops the selected pulser. The command will only be executed successfully if the corresponding parts have been correctly configured (see CMD_SET_EXTENSION_PORT). part: *see CMD_START_EXTENSION_PULSER*											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		part		0				end flag	
Byte string (HEX)	A5	5A	23	01	part	00	00	00	00	00	B9	9B

Command name	**CMD_SET_EXTENSION_OUTPUT**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command sets the selected output. The command will only be executed successfully if the corresponding parts have been correctly configured (see CMD_SET_EXTENSION_PORT). part: 1 = part B (output2) 3 = part D (output1) 5 = part F (output3) 7 = part B (output2) and part D (output1) 9 = part B (output2) and part F (output3) 11 = part D (output1) and part F (output3) 255 = all parts o1: 0 = off, ≠0 = on o2: 0 = off, ≠0 = on o3: 0 = off, ≠0 = on											
Format	integer		integer		integer		byte	byte	byte	byte	integer	
Parameter	preamble		command		part		o1	o3	o2		end flag	
Byte string (HEX)	A5	5A	24	01	part	00	o1	o3	o2	00	B9	9B

MCA FileSystem Command:

Command name	**CMD_WRITE_FILE**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command writes the current measurement into a file. The file name is created automatically from a consecutive number. Basis for this command is that a microSD card with enough free memory is plugged in. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		0		0		0		end flag	
Byte string (HEX)	A5	5A	28	01	00	00	00	00	00	00	B9	9B
Remarks	Call CMD_QUERY_STATE527_EX to get the needed and the free microSD memory size. The command returns immediately., so CMD_QUERY_STATE527_EX has to be called again to get the state and the result of the file writing process.											

Additional Module Command:

Command name	**CMD_COMM_ADD_ON_MODULE**											
Compatibility	New MCA527 command (since firmware version 20.02)											
Execution right	Necessary											
Description	The command serves to communicate with an add-on module that can be optionally connected with the MCA527[53]via the two-wire interface (I^2C) with the address 0100 0100. An add-on module can be used to control other components within a measuring system. With the help of the command, bytes are sent to and received from the module. The MCA527 only acts as intermediary. The caller of the command has to pass the correct bytes to send and has to interpret the received bytes. As add-on module are all modules allowed that support the stipulated communication protocol. The command that is sent to the add-on module consists of a length[54] byte (len), a command byte (cmd), up to four optional parameter bytes (p1, p2, p3, p4) and a checksum byte. The checksum byte is calculated by the MCA527 firmware. The MCA527 passes the bytes to the add-on module without knowing the meaning and receives always 32 bytes from it, no matter if they matter. The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		byte	byte	byte	byte	byte	byte	integer	
Parameter	preamble		command		len	cmd	p1	p2	p3	p4	end flag	
Byte string (HEX)	A5	5A	40	01	len	cmd	p1	p2	p3	p4	B9	9B
Result data array												
Byte offset 0	Result of the communication						unsigned short		successful ≠ 0 unsuccessful = 0			
Byte offset 2	Bytes received from the add-on module						32 bytes					
Byte offset 34	unused						72 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	Unused						4 bytes					
Remarks	See MCA features returned by CMD_QUERY_STATE527_EX2 (byte offset 120) for the availability of the add-on module. If the add-on module is available, CMD_QUERY_STATE527_EX3 returns firmware version, hardware version and hardware ID of the add-on module (byte offset 4, 5 and 6).											

[53] The MCA527 can be a MCA527OEM or MCA527Micro.

[54] The length is the number of all bytes, including the length byte itself and the checksum byte.

MCA Query Commands:

<table>
<tr><td>Command name</td><td colspan="12"><b>CMD_QUERY_POWER</b></td></tr>
<tr><td>Compatibility</td><td colspan="12">Compatible to the MCA166 command.</td></tr>
<tr><td>Execution right</td><td colspan="12">Not necessary</td></tr>
<tr><td>Description</td><td colspan="12">The command reads the power state.</td></tr>
<tr><td>Format</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="4">long</td><td colspan="2">integer</td></tr>
<tr><td>Parameter</td><td colspan="2">preamble</td><td colspan="2">command</td><td colspan="2">0</td><td colspan="4">0</td><td colspan="2">end flag</td></tr>
<tr><td>Byte string (HEX)</td><td>A5</td><td>5A</td><td>59</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>00</td><td>B9</td><td>9B</td></tr>
<tr><td>Result data array</td><td colspan="12"></td></tr>
<tr><td>Byte offset 0</td><td colspan="6">Battery current [mA]</td><td colspan="2">unsigned long</td><td colspan="4">At the MCA527Micro this data corresponds to the USB input current.</td></tr>
<tr><td>Byte offset 4</td><td colspan="6">HV primary current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 8</td><td colspan="6">+12 V primary current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 12</td><td colspan="6">-12 V primary current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 16</td><td colspan="6">+24 V primary current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 20</td><td colspan="6">-24 V primary current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 24</td><td colspan="6">Battery voltage [mV]</td><td colspan="2">unsigned long</td><td colspan="4">At the MCA527Micro this data corresponds to the USB input voltage.</td></tr>
<tr><td>Byte offset 28</td><td colspan="6">HV [* 1.2 V]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 32</td><td colspan="6">HV state</td><td colspan="2">unsigned long</td><td colspan="4">irrelevant for MCA527</td></tr>
<tr><td>Byte offset 36</td><td colspan="6">+ 12 V actual value [* 0.0625 V]</td><td colspan="2">unsigned char</td><td colspan="4"></td></tr>
<tr><td>Byte offset 37</td><td colspan="6">- 12 V actual value [* 0.0625 V]</td><td colspan="2">unsigned char</td><td colspan="4"></td></tr>
<tr><td>Byte offset 38</td><td colspan="6">+ 24 V actual value [* 0.125 V]</td><td colspan="2">unsigned char</td><td colspan="4"></td></tr>
<tr><td>Byte offset 39</td><td colspan="6">- 24 V actual value [* 0.125 V]</td><td colspan="2">unsigned char</td><td colspan="4"></td></tr>
<tr><td>Byte offset 40</td><td colspan="6">Current high voltage [V]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 44</td><td colspan="6">Voltage on SUB-D9 pin3 [* 0.3125 mV]</td><td colspan="2">short</td><td colspan="4">optional feature[55]</td></tr>
<tr><td>Byte offset 46</td><td colspan="6">Voltage on SUB-D9 pin5 [* 0.3125 mV]</td><td colspan="2">short</td><td colspan="4">optional feature[55]</td></tr>
<tr><td>Byte offset 48</td><td colspan="6">Power switches</td><td colspan="2">unsigned long</td><td colspan="4">0X80 -24V ON
0X40 +24V ON
0X20 -12V ON
0X10 +12V ON</td></tr>
<tr><td>Byte offset 52</td><td colspan="6">Charger current [mA]</td><td colspan="2">unsigned long</td><td colspan="4"></td></tr>
<tr><td>Byte offset 56</td><td colspan="6">Current source value on SUB-D9 pin5 [* 0.1 µA]</td><td colspan="2">unsigned short</td><td colspan="4">optional feature[55]</td></tr>
<tr><td>Byte offset 58</td><td colspan="6">Current source state on SUB-D9 pin5
(0 = off, 1 = on)</td><td colspan="2">unsigned short</td><td colspan="4">optional feature[55]</td></tr>
<tr><td>Byte offset 60</td><td colspan="6">Input resistance on SUB-D9 pin5 [kΩ]</td><td colspan="2">unsigned short</td><td colspan="4">optional feature[55]</td></tr>
</table>

[55] See the remarks of the commands CMD_QUERY_STATE527 and CMD_QUERY_STATE527_EX2 referring to the feature flags `FEATURES_ANALOG_VOLTAGES` and `FEATURES2_ANALOG_VOLTAGES`.

Command name	**CMD_QUERY_POWER** *(1st Continuation)*		
Result data array			
Byte offset 62	ADC correction offset on SUB-D9 pin5 [LSB]	char	optional feature[55]
Byte offset 63	Gain correction factor on SUB-D9 pin5 (factor = 0.001 * value + 1)	char	optional feature[55]
Byte offset 64	Battery current [mA] of the previous sweep[56]	unsigned long	At the MCA527Micro this data corresponds to the USB input voltage.
Byte offset 68	HV primary current [mA] of the previous sweep	unsigned long	
Byte offset 72	+12V primary current [mA] of the previous sweep	unsigned long	
Byte offset 76	-12V primary current [mA] of the previous sweep	unsigned long	
Byte offset 80	+24V primary current [mA] of the previous sweep	unsigned long	
Byte offset 84	-24V primary current [mA] of the previous sweep	unsigned long	
Byte offset 88	Battery voltage [mV] of the previous sweep	unsigned long	At the MCA527Micro this data corresponds to the USB input voltage.
Byte offset 92	HV [* 1.2 V] of the previous sweep	unsigned long	
Byte offset 96	ADC correction offset on SUB-D9 pin3 [LSB]	char	optional feature[55]
Byte offset 97	Gain correction factor on SUB-D9 pin3 (factor = 0.001 * value + 1)	char	optional feature[55]
Byte offset 98	HV control voltage [mV][57]	unsigned short	relevant for MCA527Micro with power module firmware version ≥ 0.5
Byte offset 100	+12V actual value [* 0.0625 V] of the previous sweep	unsigned char	
Byte offset 101	-12V actual value [* 0.0625 V] of the previous sweep	unsigned char	
Byte offset 102	+24V actual value [* 0.125 V] of the previous sweep	unsigned char	
Byte offset 103	-24V actual value [* 0.125 V] of the previous sweep	unsigned char	
Byte offset 104	Voltage on SUB-D9 pin3 [* 0.3125 mV] of the previous sweep	short	optional feature[55]
Byte offset 106	Command flag and parameters	8 bytes	
Byte offset 114	Voltage on SUB-D9 pin5 [* 0.3125 mV] of the previous sweep	short	optional feature[55]
Byte offset 116	Charger current [mA] of the previous sweep	unsigned long	
Byte offset 120	unused	4 bytes	

[56] Values which description ends with the string “of the previous sweep” refer either to the previously finished sweep during the repeat mode or to the last stopped measurement.

[57] Since firmware version 15.02.

Command name	**CMD_QUERY_POWER** *(2nd Continuation)*		
Result data array			
Byte offset 124	Power module detector info source (low nibble)[57] Power module DAC type (high nibble)	unsigned char	relevant for MCA527Micro with power module firmware version ≥ 0.5
Byte offset 125	Power module features[57]	unsigned char	relevant for MCA527Micro with power module firmware version ≥ 0.5
Byte offset 126	Checksum	unsigned short	

Byte offset 128	MCA state	integer value	STATE_READY = 1
			STATE_RUN = 2
			STATE_SUSPEND = 3
			STATE_FINISH = 4
			STATE_STOP = 5
			STATE_FAIL = 6
			STATE_WAIT_FOR_TRIGGER = 7

Byte offset 130	unused	2 bytes	

Command name	**CMD_QUERY_SYSTEM_DATA**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads the special MCA data.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	62	00	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	unused						10 bytes					
Byte offset 10	Detected counts						48 bit integer					
Byte offset 16	unused						2 bytes					
Byte offset 18	Counts above the spectrum range[58]						48 bit integer					
Byte offset 24	unused						2 bytes					
Byte offset 26	Counts above the spectrum range of the previous sweep[58] [59]						48 bit integer					
Byte offset 32	unused						4 bytes					
Byte offset 36	MMCA on time [s]						unsigned long					
Byte offset 40	Real time [s] of the previous sweep						unsigned long					
Byte offset 44	Dead time [ms] of the previous sweep						unsigned long					
Byte offset 48	Start time of the previous sweep						unsigned long					
Byte offset 52	Fast dead time [ms] of the previous sweep						unsigned long					
Byte offset 56	Elapsed sweeps (repeat mode)						unsigned long					
Byte offset 60	Busy time [ms] of the previous sweep						unsigned long		always 0 at MCA527			
Byte offset 64	Fractional digits[60] of the real time [msec] of the previous sweep						unsigned short					
Byte offset 66	unused						8 bytes					
Byte offset 74	Detected counts of the previous sweep						48 bit integer					
Byte offset 80	Counter of stabilization steps						unsigned long					
Byte offset 84	Current stabilization offset[61]						long					
Byte offset 88	Maximal negative stabilization offset						long					
Byte offset 92	Maximal positive stabilization offset						long					
Byte offset 96	Counter of received commands						unsigned long					

[58] Since firmware version 16.00.

[59] Values which description ends with the string “of the previous sweep” refer either to the previously finished sweep during the repeat mode or to the last stopped measurement.

[60] Since firmware version 14.03.

[61] The stabilization offset is the internal fine gain change that is needed to stabilize the peak. Normally, you would have to divide the value by 10 000 to get the float value. But because of compatibility with the MCA166, you have to divide value by 11 142 if it is negative and by 9 104 if it is positive.

Command name	**CMD_QUERY_SYSTEM_DATA** *(Continuation)*		
Result data array			
Byte offset 100	Counter of unsuccessful commands	unsigned long	
Byte offset 104	ADC pipeline latency[62]	unsigned char	
Byte offset 105	High sampling rate used[62]	unsigned char	
Byte offset 106	Command flag and parameters	8 bytes	

Byte offset 114	Buffer state	unsigned short	OCCUPIED = 0x2000
			OVERRUN = 0x4000
			FILLED = 0x8000

Byte offset 116	Stabilization area preset	unsigned long	
Byte offset 120	Stabilization time preset [sec]	unsigned short	
Byte offset 122	Low shaping time [* 0.1 µsec]	unsigned char	
Byte offset 123	High shaping time[* 0.1 µsec]	unsigned char	
Byte offset 124	Minimum recommended core clock [* 100 MHz][63]	unsigned char	
Byte offset 125	Maximum allowed core clock [* 100 MHz][63]	unsigned char	
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	Set ADC sampling rate [kilohertz][64]	unsigned short	

[62] Since firmware version 20.00

[63] Since firmware version 14.06.

[64] Since firmware version 16.00.

Command name	**CMD_QUERY_STATE**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads the state.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	5A	00	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	MCA acquire mode						unsigned short		MODE_MCA = 0 MODE_MCS = 1			
Byte offset 2	Preset (automatic stop condition) [65]						unsigned short		PRESET_NONE = 0 PRESET_REAL = 1 PRESET_LIVE = 2 PRESET_INT = 3 PRESET_AREA = 4			
Byte offset 4	Preset value						unsigned long					
Byte offset 8	MCA mode: elapsed preset MCS mode: elapsed MCS channels[66]						unsigned long					
Byte offset 12	Repeat value						unsigned short					
Byte offset 14	Elapsed sweeps						unsigned short					
Byte offset 16	MCS time per channel [* 10 msec]						unsigned short					
Byte offset 18	Elapsed time per channel [* 10 msec]						unsigned short					
Byte offset 20	Real time [sec]						unsigned long		The fractional digits of the real time are returned by CMD_QUERY_STATE527_EX, byte offset 82.			
Byte offset 24	MCA mode: counts per second MCS mode: counts per channel[67]						unsigned long					
Byte offset 28	Dead time [msec]						unsigned long					
Byte offset 32	Busy time [msec]						unsigned long		always 0 at MCA527			
Byte offset 36	MCA channels						unsigned short					
Byte offset 38	Threshold [%]						unsigned short		See CMD_QUERY_STATE527 and CMD_SET_THRESHOLD			
Byte offset 40	LLD						unsigned short					
Byte offset 42	ULD						unsigned short					
Byte offset 44	ROI begin (preset integral/area)						unsigned short					
Byte offset 46	ROI end (preset integral/area)						unsigned short					

[65] For general mode 3 and 4 (see CMD_SET_GENERAL_MODE), only PRESET_REAL is supported, PRESET_LIVE, PRESET_INT and PRESET_AREA are unsupported and mean the same like PRESET_NONE.

[66] Since firmware version 13.00.

[67] Older applications for the MCS mode used the counts per channel to calculate the counts per second. However, this is inexact. For this reason, since firmware version 13.00, the 'counts per second' for both modes are (additionally) returned at byte offset 116. Newer applications for the MCS mode should use this parameter.

Command name	**CMD_QUERY_STATE** *(1st Continuation)*		
Result data array			
Byte offset 48	Amplifier coarse gain	unsigned short	
Byte offset 50	Amplifier fine gain	unsigned short	
Byte offset 52	Slow discriminator value	unsigned short	irrelevant for MCA527
Byte offset 54	Fast discriminator value	unsigned short	irrelevant for MCA527
Byte offset 56	High voltage [V]	unsigned short	
Byte offset 58	High voltage polarity	unsigned short	0 = positive, 1 = negative
Byte offset 60	Power switches	unsigned short	0X80 -24V ON 0X40 +24V ON 0X20 -12V ON 0X10 +12V ON 0X02 HV ON
Byte offset 62	PZC value	unsigned short	
Byte offset 64	Time offset for PZC and DTC low	unsigned short	irrelevant for MCA527
Byte offset 66	Time offset for PZC and DTC high	unsigned short	irrelevant for MCA527
Byte offset 68	Stabilization state or channel	unsigned short	

Byte offset 70	Stabilization result	unsigned short	STAB_STARTED = 0
			STAB_ACTIVE = 1
			STAB_ERROR = 3

Byte offset 72	Stab. ROI begin	unsigned short	
Byte offset 74	Stab. ROI end	unsigned short	

Byte offset 76	ADC input	unsigned short	ADC_SHAPING = 0
			ADC_DIRECT_PPA_POS = 3[68]
			ADC_DIRECT_PPA_NEG = 4[68]
			ADC_DIRECT_PIA = 6[69]

Byte offset 78	ADC input polarity	unsigned short	INPUT_POLARITY_POS = 0
			INPUT_POLARITY_NEG = 1

Byte offset 80	Shaping time choice[70]	unsigned short	Low shaping time = 1 High shaping time = 3

Byte offset 82	Pile up rejection (PUR) state	unsigned short	PUR_OFF = 0
			PUR_ON = 1

Byte offset 84	MCS input	unsigned short	MCS_INPUT_TTL = 0
			MCS_INPUT_RATE = 1
			MCS_INPUT_DISCR = 2

[68] In previous document versions, ADC_DIRECT_PPA_POS and ADC_DIRECT_PPA_NEG were named ADC_DIRECT_POS and ADC_DIRECT_NEG, respectively. If the parameter passed to the command CMD_SET_MCA_INPUT is equal to 5 = input direct (**P**ulse **P**eak **A**nalysis), the returned value is still ADC_DIRECT_PPA_POS or ADC_DIRECT_PPA_NEG according to the ADC input polarity (byte offset 78). This behavior is based on the compatibility with the MCA166 and older programs.

[69] Since firmware version 21.00.

[70] In the MCA-166 User's Manual, this parameter is named DTC. You find therefore this parameter in the source code under this name.

Command name	**CMD_QUERY_STATE** *(2nd Continuation)*		
Result data array			
Byte offset 86	MCA serial number	unsigned short	
Byte offset 88	MCA hardware version	unsigned short	= 0xFFFF
Byte offset 90	MCA firmware version	unsigned short	= 0xFFFF
Byte offset 92	MCS channels	unsigned short	
Byte offset 94	Last power state	unsigned short	
Byte offset 96	Battery capacity [%]	unsigned short	
Byte offset 98	Battery life time	unsigned short	not supported
Byte offset 100	Start time	unsigned long	See CMD_START
Byte offset 104	TDF	unsigned short	always 0 at MCA527
Byte offset 106	Command flag and parameters	8 bytes	

Byte offset 114	Buffer state	unsigned short	OCCUPIED = 0x2000
			OVERRUN = 0x4000
			FILLED = 0x8000

Byte offset 116	Counts per second[71]	unsigned long	
Byte offset 120	Differential dead time [‰]	unsigned short	
Byte offset 122	HV inhibit mode	short	
Byte offset 124	HV inhibit state	unsigned short	
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	Start flag[72]	unsigned short	See CMD_START

[71] Since firmware version 13.00.

[72] Since firmware version 12.02.

Command name	**CMD_QUERY_STATE527**											
Compatibility	New MCA527 command.											
Execution right	Not necessary											
Description	The command reads the state.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	01	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	MCA hardware version (hexadecimal)						unsigned short		High byte: major version Low byte: minor version			
Byte offset 2	MCA firmware version (hexadecimal)						unsigned short		High byte: major version Low byte: minor version			
Byte offset 4	MCA hardware modification						unsigned short		Full version = 0 Lite version = 1 OEM version = 2 Micro version = 3 Nano version = 4			
Byte offset 6	MCA firmware modification						unsigned short					
Byte offset 8	MCA features						unsigned long		See remarks at the end of the table (2nd continuation)			
Byte offset 12	Time on internal clock						32 bits		See CMD_SET_TIME			
Byte offset 16	reserved						4 bytes					
Byte offset 20	Testing phase [sec]						unsigned long		0 = expired 0xFFFFFFFF = without other value = remaining			
Byte offset 24	MCA temperature [* 0.007 812 5 °C]						short		0x8000 = not available			
Byte offset 26	General MCA mode						unsigned short		See CMD_SET_GENERAL_MODE			
Byte offset 28	Discarded cycles						unsigned long		1 discarded cycle = 400 µsec			
Byte offset 32	Core clock [* 100 MHz]						unsigned short					
Byte offset 34	Trigger filter for low shaping time						unsigned char		See CMD_SET_TRIGGER_FILTER			
Byte offset 35	Trigger filter for high shaping time						unsigned char		See CMD_SET_TRIGGER_FILTER			
Byte offset 36	Expander flags						unsigned short					
Byte offset 38	Offset DAC						unsigned short					
Byte offset 40	Detector temperature [* 0.007 812 5 °C][73]						short		0x8000 = not available			
Byte offset 42	Power module temperature [* 0.007 812 5 °C]						short		0x8000 = not available			
Byte offset 44	MCA serial number						unsigned short					
Byte offset 46	Am I right holder?						short		0 = no, -1 = yes			
Byte offset 48	Right holder IP address						unsigned char [4]		0.0.0.0 = USB or RS232			
Byte offset 52	Right holder UDP port						unsigned short		0 = USB or RS232			

[73] *Full version:* The value comes from a temperature sensor that is connected to the one-wire interface.
OEM version: If the board is assembled with a power module, the value comes from a temperature sensor that is connected to the one-wire interface, otherwise the value from an additional external temperature sensor connected to the two-wire interface with the address 1001 0100 is assumed as the detector temperature. This sensor must be a LM73.
Micro version: The value from an additional external temperature sensor connected to the two-wire interface with the address 1001 0100 is assumed as the detector temperature. This sensor must be a TMP102.

Command name	**CMD_QUERY_STATE527** *(1st Continuation)*		
Result data array			
Byte offset 54	Execution right	short	-1 = not granted 0 = reserved 1 … 15 = granted
Byte offset 56	Maximum MCA channels provided by this MCA	unsigned short	
Byte offset 58	Power module firmware version	unsigned char	High nibble: major version Low nibble: minor version
Byte offset 59	Power module hardware version	unsigned char	High nibble: major version Low nibble: minor version
Byte offset 60	Power module serial number	unsigned short	
Byte offset 62	Power module ID	unsigned short	Full version[74] = 0 Lite version = 1 Micro/Nano version = 3
Byte offset 64	Maximum allowed high voltage[75] [V]	unsigned short	
Byte offset 66	Threshold [* 0.1 %]	unsigned short	See CMD_QUERY_STATE and CMD_SET_THRESHOLD_TENTHS
Byte offset 68	Fast dead time [msec]	unsigned long	
Byte offset 72	Evaluation filter type	unsigned short	
Byte offset 74	Flattop time [* 0.1 µsec]	unsigned short	See CMD_SET_FLAT_TOP_TIME
Byte offset 76	Evaluation filter size	unsigned short	
Byte offset 78	Coefficient for automatic threshold calculation [* 0.0625]	unsigned short	80 … 1600
Byte offset 80	MCA temperature [* 0.007 812 5 °C] of the previous sweep[76]	short	0x8000 = not available
Byte offset 82	Detector temperature [* 0.007 812 5 °C] of the previous sweep	short	0x8000 = not available
Byte offset 84	Customized IP address	unsigned char [4]	See CMD_SET_IP_ADDRESS
Byte offset 88	Actual IP address	unsigned char [4]	
Byte offset 92	MCS time per channel [* 0.1 msec]	unsigned long	
Byte offset 96	Elapsed time per channel [* 0.1 msec]	unsigned long	
Byte offset 100	Auto trigger threshold [* 0.000 061 035 156 25][77]	long	
Byte offset 104	Power module temp. [* 0.007 812 5 °C] of the previous sweep	short	0x8000 = not available
Byte offset 106	Command flag and parameters	8 bytes	
Byte offset 114	Jitter correction	unsigned char	0 = off, 1 = on
Byte offset 115	Baseline restoring	unsigned char	0 = off, 1 = 1/1, 2 = 1/2, 3 = 1/4, 4 = 1/8, 5 = 1/16, 6 = 1/32
Byte offset 116	Fixed trigger threshold [* 0.000 061 035 156 25]	long	0 … 268 435 455 0 = auto threshold calculation

[74] The OEM version is able to operate a power module that is compatible to that from the full version.

[75] The maximum allowed high voltage is determined by the power module, and in some system configurations if the detector is firmly connected, additionally by the maximum high voltage of the detector.

[76] Values which description ends with the string “of the previous sweep” refer either to the previously finished sweep during the repeat mode or to the last stopped measurement.

[77] Since firmware version 15.00, the auto trigger threshold is also returned for direct input.

Command name	**CMD_QUERY_STATE527** *(2nd Continuation)*		
Result data array			
Byte offset 120	Input mode	unsigned char	0 = alterable, 1 = fixed
Byte offset 121	Highest allowed shaping time [* 0.1 µsec]	unsigned char	
Byte offset 122	Gating mode	unsigned char	See CMD_SET_GATING
Byte offset 123	Gating signal	unsigned char	See CMD_SET_GATING
Byte offset 124	Gating shift	unsigned char	See CMD_SET_GATING
Byte offset 125	Hardware-based coarse gain levels[78] (All other coarse gain levels are realized by the firmware.)	unsigned char	Bit 0: 2, Bit 1: 5, Bit 2: 10, Bit 3: 20, Bit 4: 50, Bit 5: 100 Bit 6: 200, Bit 7: 500
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	Differential fast dead time [‰]	unsigned short	
Remarks	The different hardware and firmware versions of the MCA527 support different features. The parameter “MCA features” (byte offset 8) gives more detailed information about the supported features than the value “MCA hardware modification” (byte offset 4). **FEATURES_EXPANDER:** The expander is an imperative component of the full, lite and OEM version of the MCA527. If this flag is not set, these devices are defective. The micro version of the MCA527, however, does not contain the expander. **FEATURES_INTERN_TEMPERATURE:** This flag indicates that the MCA527 supplies the temperature of its circuit board. **FEATURES_EXTERN_TEMPERATURE:** In Full version, this flag indicates that temperature measuring via 1-wire interface is supplied. The flag does, however, not indicate whether a 1-wire temperature sensor is connected or not. In Micro version, this flag indicates that a temperature sensor with the address 1001 0100 is connected to the two-wire interface. In OEM version, this flag can indicate two different kinds of temperature measuring. If a power module is available, the temperature measuring via 1-wire interface is supplied, otherwise a temperature sensor with the address 1001 0100 is connected to the two-wire interface. **FEATURES_MAC_ADDRESS:** This flag indicates that the EEPROM that supplies the MAC address for the Ethernet has been detected. A valid MAC address is one of the requirements for connecting the MCA527 via Ethernet. **FEATURES_POWER_MODULE:** This flag indicates that a power module has been detected. In the full and lite version, the power module is an integral component. If the flag is not set, these devices are defective. In OEM and micro version, the power module is optional. If the flag is not set, the devices either have no power module or they are defective. **FEATURES_MSD_CARD:** This flag indicates that a micro SD memory card with FAT32 file system has been detected. **FEATURES_PMODULE_TEMPERATURE:** This flag indicates that the power module supplies the temperature of its circuit board. **FEATURES_ANALOG_VOLTAGES:** This flag indicates that the MCA527 supports analog voltage measuring on SUB-D9 pin3 and pin5. **FEATURES_GATING_INPUT:** This flag indicates that the MCA527 supports gating. **FEATURES_EXTENSION_PORT:** This flag indicates that the MCA527 supports the extension port. The availability of the single parts is indicated by “Extension port parts availability” (see CMD_QUERY_STATE527_EX, byte offset 30**).** It is possible to plug a bluetooth module in the extension port. If the MCA527 identifies a bluetooth module, it clears the flag to avoid that other processes try to access to the extension port.		

[78] Since firmware version 13.05.

Command name	**CMD_QUERY_STATE527** *(3rd Continuation)*
Remarks	**FEATURES_LF_REJECTION:** This flag indicates that the MCA527 supports an evaluation filter with low frequency rejection. If this flag is set, the evaluation filter can be chosen with CMD_SET_EVAL_FILTER_TYPE. **FEATURES_JITTER_CORRECTION:** This flag indicates that the MCA527 supports jitter correction. If not set, the jitter correction cannot be switched on with CMD_SET_JITTER_CORRECTION. The command will return “not handled”. **FEATURES_ADJUSTABLE_TRIGGER_FILTER:** This flag indicates that the MCA527 supports the change between different trigger filters. Since firmware version 12.00, this flag is only set if all four originally existing trigger filters are available. This is due to the compatibility of older programs. Newer programs should always test the value “Trigger filter availability flags” returned from CMD_QUERY_STATE527_EX (byte offset 88) that gives information about the trigger filters that are supported by the current firmware version. If more than one trigger filter is available, the command CMD_SET_TRIGGER_FILTER can be successfully executed otherwise the command will return “not handled”. **FEATURES_ADJUSTABLE_BASELINE_RESTORER:** This flag indicates that the MCA527 supports the change between different settings for the baseline restorer. If not set, only the default setting is available and the command CMD_SET_BASELINE_RESTORING will return “not handled”. **FEATURES_ADJUSTABLE_COARSE_GAIN:** This flag indicates that the MCA527 supports the change between different coarse gain levels. If not set, only the default coarse gain level is available and the command CMD_SET_GAIN will ignore the parameter “coarse gain”. **FEATURES_USB_CHARGING_DEFAULT_OFF:** This flag indicates that the USB charger will not be automatically switched on when the USB cable is plugged in. If the flag is not set, the USB charger will be switched on by default. This flag is only relevant if a power module is existent and the firmware version is higher than 12.01. **FEATURES_NO_OFFSET_DAC:** This flag indicates that the MCA527 contains no offset DAC. This applies to the micro version. In this case, the MCS TTL input mode and the general mode 3 work only properly if the input signal is adjusted by default for these modes. **FEATURES_SDRAM_OK:** The MCA527 runs a SDRAM memory test at booting[79]. If this flag is not set, the device is defective. **FEATURES_TIMESTAMP_RECORDER:** This flag indicates that the MCA527 supports the time stamp recorders (list modes 1, 2 and 3, respectively general modes 3, 4 and 5). **FEATURES_BLUETOOTH:** This flag indicates that the MCA527 has detected a Bluetooth module at the extension port. **FEATURES_GATING_BY_TIME:** This flag indicates that the MCA527 supports the gating mode 'sort by time'. **FEATURES_BOOT_PRESETS[80] [81]:** This flag indicates that the MCA527 is preset at booting with predefined parameters stored within the MCA527 by the manufacturer or a technician. **FEATURES_DETECTOR_INFO[81]:** This flag indicates that the connected detector supports detector information. However, only the flag FEATURES_DETECTOR_INFO_VALID indicates the validity. **FEATURES_DETECTOR_INFO_VALID[80] [81]:** This flag indicates that the detector information are valid. Some parameters of the detector information are used for presetting the MCA527 at booting. **FEATURES_ADDITIONAL_TEMPERATURE1[82]**: This flag indicates that an additional external temperature sensor with the corresponding address has been detected at the two-wire interface. The temperature is returned by CMD_QUERY_STATE527_EX2. See there for more information. **FEATURES_ADDITIONAL_TEMPERATURE2[82]:** See previous flag.

[79] Since firmware version 13.05.

[80] Since firmware version 14.03.

[81] The flags FEATURES_BOOT_PRESETS and FEATURES_DETECTOR_INFO_VALID indicate two competing methods for presetting the MCA527 at booting. If both flags are set, the firmware uses the parameters of both sources according to certain rules.

[82] Since firmware version 15.06. Micro and OEM version only.

Command name	**CMD_QUERY_STATE527** *(4th Continuation)*
Remarks	**FEATURES_FILTER2[83]:** This flag indicates that the MCA527 supports a second evaluation filter for special evaluation routines. **FEATURES_LISTMODE4[84]:** This flag indicates that the MCA527 supports the list mode 4 (general mode 6). **FEATURES_LISTMODE5[83]:**This flag indicates that the MCA527 supports the list mode 5 (general mode 7). **FEATURES_DEBUG_INFO:** This flag indicates that the MCA527 supplies additional debug information. The debug information is destined for exclusive use by the MCA527 firmware developer. For this reason it is not described more detailed. Decoding:

```
#define FEATURES_EXPANDER                    0x00000001
#define FEATURES_INTERN_TEMPERATURE          0x00000002
#define FEATURES_EXTERN_TEMPERATURE          0x00000004
#define FEATURES_MAC_ADDRESS                 0x00000008

#define FEATURES_POWER_MODULE                0x00000010
#define FEATURES_MSD_CARD                    0x00000020
#define FEATURES_PMODULE_TEMPERATURE         0x00000040
#define FEATURES_ANALOG_VOLTAGES             0x00000080

#define FEATURES_GATING_INPUT                0x00000100
#define FEATURES_EXTENSION_PORT              0x00000200
#define FEATURES_LF_REJECTION                0x00000400
#define FEATURES_JITTER_CORRECTION           0x00000800

#define FEATURES_ADJUSTABLE_TRIGGER_FILTER    0x00001000
#define FEATURES_ADJUSTABLE_BASELINE_RESTORER 0x00002000
#define FEATURES_ADJUSTABLE_COARSE_GAIN       0x00004000
#define FEATURES_USB_RS232                    0x00008000

#define FEATURES_USB_CHARGING_DEFAULT_OFF     0x00010000
#define FEATURES_NO_OFFSET_DAC                0x00020000
#define FEATURES_SDRAM_OK                     0x00040000
#define FEATURES_TIMESTAMP_RECORDERS          0x00080000

#define FEATURES_BLUETOOTH                    0x00100000
#define FEATURES_GATING_BY_TIME               0x00200000
#define FEATURES_BOOT_PRESETS                 0x00400000
#define FEATURES_DETECTOR_INFO                0x00800000

#define FEATURES_DETECTOR_INFO_VALID          0x01000000
#define FEATURES_RS485_BUS                    0x02000000
#define FEATURES_ADDITIONAL_TEMPERATURE1      0x04000000
#define FEATURES_ADDITIONAL_TEMPERATURE2      0x08000000

#define FEATURES_FILTER2                      0x10000000
#define FEATURES_LISTMODE4                    0x20000000
#define FEATURES_LISTMODE5                    0x40000000
#define FEATURES_DEBUG_INFO                   0x80000000
```

[83] Since firmware version 16.00.

[84] Since firmware version 20.00.

Command name	**CMD_QUERY_STATE527_EX**											
Compatibility	New MCA527 command.											
Execution right	Not necessary											
Description	The command reads the state.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	10	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Common memory size [bytes][85]						unsigned long					
Byte offset 4	Common memory fill stop [bytes]						unsigned long					
Byte offset 8	Common memory fill level [bytes]						unsigned long					
Byte offset 12	Oscilloscope time resolution						short					
Byte offset 14	Oscilloscope trigger source						unsigned short					
Byte offset 16	Oscilloscope trigger position						unsigned short					
Byte offset 18	Oscilloscope trigger threshold						unsigned short					
Byte offset 20	PUR counter						unsigned long					
Byte offset 24	Configured purpose of extension port part A						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 25	Configured purpose of extension port part B						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 26	Configured purpose of extension port part C						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 27	Configured purpose of extension port part D						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 28	Configured purpose of extension port part E						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 29	Configured purpose of extension port part F						unsigned char		See CMD_SET_EXTENSION_PORT			
Byte offset 30	Extension port parts availability						unsigned char		Bit (0 … 5): part (A … F) is available Bit 6: Input signal (part E) can be looped through to output pin (part B).			
Byte offset 31	Extension port state flags						unsigned char					
Byte offset 32	Extension port polarity flags						unsigned char					
Byte offset 33	Highest allowed flattop time [* 0.1 µsec][86]						unsigned char					
Byte offset 34	Size of the booting presets data structure[87]						unsigned short					
Byte offset 36	Extension port pulser 1 period						unsigned long					
Byte offset 40	Extension port pulser 2 period						unsigned long					
Byte offset 44	Extension port pulser 1 width						unsigned long					
Byte offset 48	Extension port pulser 2 width						unsigned long					
Byte offset 52	Extension port RS232 baud rate						unsigned short		6 250 000 / value			
Byte offset 54	Extension port RS232 flags						unsigned short					

[85] The common memory size is not only depending on the hardware and firmware version but also on other operating conditions. If a micro SD memory card is plugged in the MCA527, additional memory is needed to operate it. This reduces the available common memory.

[86] Since firmware version 13.07.

[87] Since firmware version 14.03.

Command name	**CMD_QUERY_STATE527_EX** *(1st Continuation)*		
Result data array			
Byte offset 56	Extension port counter 1	unsigned long	
Byte offset 60	Extension port counter 1 cps	unsigned long	
Byte offset 64	Extension port counter 1 of the previous sweep[88]	unsigned long	
Byte offset 68	Extension port counter 2	unsigned long	
Byte offset 72	Extension port counter 2 cps	unsigned long	
Byte offset 76	Extension port counter 2 of the previous sweep	unsigned long	
Byte offset 80	RS232 transfer buffer byte count	unsigned short	
Byte offset 82	Fractional digits[89] of the real time [msec]	unsigned short	The integer part of the real time is returned by CMD_QUERY_STATE, byte offset 20.
Byte offset 84	PUR counter of the previous sweep	unsigned long	
Byte offset 88	Trigger filter availability flags[90]	unsigned long	Bit *n = 1* means trigger filter *n is available.*
Byte offset 92	Trigger filter value1 [* 0.000 061 035 156 25]	short	According the selected trigger filter: (value2; value1) (value2; 0; value1) (value1; value2; value1) (value1; 0; value2; 0; value1)
Byte offset 94	Trigger filter value2 [* 0.000 061 035 156 25]	short	
Byte offset 96	TTL low level [* 0.1 V]	unsigned char	
Byte offset 97	TTL high level [* 0.1 V]	unsigned char	
Byte offset 98	Coefficient for automatic threshold calculation for direct input [* 0.0625]	unsigned short	80 … 1600
Byte offset 100	ADC overflows per second[91]	unsigned long	
Byte offset 104	Set ADC sampling rate [kilohertz][91]	unsigned short	
Byte offset 106	Command flag and parameters	8 bytes	
Byte offset 114	File size[92] corresponding to setup [kilobytes]	unsigned short	
Byte offset 116	Total microSD card memory size [kilobytes]	unsigned long	
Byte offset 120	Free microSD card memory size [kilobytes][93]	unsigned long	
Byte offset 124	File writing state	char	0 = idle, ≠0 = in process
Byte offset 125	Result of last file writing	char	-1 = not yet executed 0 = unsuccessful 1 = successful

[88] Values which description ends with the string “of the previous sweep” refer either to the previously finished sweep during the repeat mode or to the last stopped measurement.

[89] Since firmware version 13.04. If the value is returned during a running measurement, it should be treat with caution. Since the integer part of the real time is returned by another command, the user must consider the time difference between the commands.

[90] Since firmware version 12.00.

[91] Since firmware version 13.04.

[92] This is the needed size on the microSD card.

[93] This is the free microSD card memory size minus the size of one cluster that is reserved for a potential expansion of the directory.

Command name	**CMD_QUERY_STATE527_EX** *(2nd Continuation)*		
Result data array			
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	RS485 bus baud rate[94]	unsigned short	6 250 000 / value

[94] Since firmware version 15.03.

Command name	**CMD_QUERY_STATE527_EX2**											
Compatibility	New MCA527 command (since firmware version 14.00).											
Execution right	Not necessary											
Description	The command reads the state.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	2F	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	AHRC group 0 width						unsigned long		See CMD_SET_AHRC_PARAM			
Byte offset 4	AHRC group 1 width						unsigned long					
Byte offset 8	AHRC group 2 width						unsigned long					
Byte offset 12	AHRC group 3 width						unsigned long					
Byte offset 16	AHRC group 4 width						unsigned long					
Byte offset 20	AHRC group 5 width						unsigned long					
Byte offset 24	AHRC group 6 width						unsigned long					
Byte offset 28	AHRC group 7 width						unsigned long					
Byte offset 32	AHRC group 8 width						unsigned long					
Byte offset 36	AHRC group 9 width						unsigned long					
Byte offset 40	AHRC trigger threshold						unsigned short					
Byte offset 42	MCS time per channel for gating mode 'sort by time'[95]						unsigned short		See CMD_SET_GATING_TIME_PER_CHANNEL			
Byte offset 44	Time window 0 width for gating mode 'sort by time'[96]						unsigned long		See CMD_SET_GATING_TIME_WINDOW_WIDTH			
Byte offset 48	Time window 1 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 52	Time window 2 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 56	Time window 3 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 60	Time window 4 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 64	Time window 5 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 68	Time window 6 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 72	Time window 7 width for gating mode 'sort by time'[96]						unsigned long					
Byte offset 76	Extension port pulser 3 period[97]						unsigned long					
Byte offset 80	Extension port pulser 3 width[97]						unsigned long					
Byte offset 84	Extension port counter 3[97]						unsigned long					
Byte offset 88	Extension port counter 3 cps[97]						unsigned long					
Byte offset 92	Extension port counter 3 of the prev. sweep[97] [98]						unsigned long					

[95] Since firmware version 16.00.

[96] Since firmware version 14.02.

[97] Since firmware version 15.06. Micro version only.

[98] Values which description ends with the string “of the previous sweep” refer either to the previously finished sweep during the repeat mode or to the last stopped measurement.

Command name	**CMD_QUERY_STATE527_EX2** *(1st Continuation)*		
Result data array			
Byte offset 96	Add. temperature 1 [* 0.007 812 5 °C][99] [100]	short	0x8000 = not available
Byte offset 98	Add. temperature 1 [* 0.007 812 5 °C] of the previous sweep[99] [100]	short	0x8000 = not available
Byte offset 100	Add. temperature 2 [* 0.007 812 5 °C][99] [101]	short	0x8000 = not available
Byte offset 102	Add. temperature 2 [* 0.007 812 5 °C] of the previous sweep[99] [101]	short	0x8000 = not available
Byte offset 104	Flattop time 2 [* 0.1 µsec][102]	unsigned short	See CMD_SET_FLAT_TOP_TIME
Byte offset 106	Command flag and parameters	8 bytes	
Byte offset 114	Evaluation filter 2 size[102]	unsigned short	
Byte offset 116	Internal fine gain set by the stabilization routine[103]	unsigned long	
Byte offset 120	MCA features (2nd group)[103]	unsigned long	
Byte offset 124	Minimum event distance	unsigned short	See CMD_SET_MINIMUM_EVENT_DISTANCE
Byte offset 126	Checksum	unsigned short	
Byte offset 128	MCA state	unsigned short	See CMD_QUERY_POWER
Byte offset 130	Fast trigger input mode[104]	unsigned char	0 = disabled, 1 = enabled
Byte offset 131	Fast trigger input edge[105]	unsigned char	0 = rising, 1 = falling
Remarks	The different hardware and firmware versions of the MCA527 support different features. The value “MCA features (2nd group)” (byte offset 120) gives detailed information about the supported features. The first group of the MCA features is returned by CMD_QUERY_STATE527 (byte offset 8). **FEATURES2_MINIMUM_EVENT_DISTANCE[103]:** This flag indicates that the MCA527 supports CMD_SET_MINIMUM_EVENT_DISTANCE. **FEATURES2_SLOW_RISE[103]:** This flag indicates that the MCA527 supports an evaluation filter for slow rising signals. If set, this evaluation filter can be chosen with CMD_SET_EVAL_FILTER_TYPE. **FEATURES2_GATING_INPUT_DOWNGRADED[106]:** This flag indicates that the MCA527 has got a downgraded gating input. The gating mode “Discard” is not supported. The flag FEATURES_GATING_INPUT within the first group of the MCA features is not set. **FEATURES2_FAST_TRIGGER_INPUT:** This flag indicates that the MCA527 has got a fast trigger input. **FEATURES2_SECOND_SAMPLING_RATE:** This flag indicates that the ADC sampling rate of the MCA527 can be set to the double value. **FEATURES2_ADD_ON_MODULE[107]:** This flag indicates that an add-on module is connected with the MCA527 board.		

[99] Since firmware version 15.06. Micro and OEM version only.

[100] Additional external temperature sensor at the two-wire interface with address 1001 0000. The sensor must be a TMP102 (Micro) or LM73 (OEM).

[101] Additional external temperature sensor at the two-wire interface with address 1001 0110 (Micro) or 1001 1000 (OEM). The sensor must be a TMP102 (Micro) or LM73 (OEM).

[102] Since firmware version 16.00.

[103] Since firmware version 19.00.

[104] Since firmware version 20.00.

[105] Since firmware version 20.01.

[106] Since firmware version 20.00. Nano version only.

[107] Since firmware version 20.02.

Command name	**CMD_QUERY_STATE527_EX2** *(2nd Continuation)*

Remarks	**FEATURES2_ANALOG_VOLTAGES[108]:** This flag indicates that the MCA527 is connected with an additional circuit board that supports analog voltage measuring. The parameter “MCA features” returned by CMD_QUERY_STATE527 contains a flag with a similar meaning. The different between both variants is that the additional circuit board does not support the current source for resistance measurement.
	Decoding:
	`#define FEATURES2_MINIMUM_EVENT_DISTANCE      0x00000001`
	`#define FEATURES2_SLOW_RISE                   0x00000002`
	`#define FEATURES2_GATING_INPUT_DOWNGRADED     0x00000004`
	`#define FEATURES2_FAST_TRIGGER_INPUT          0x00000008`
	
	`#define FEATURES2_SECOND_SAMPLING_RATE        0x00000010`
	`#define FEATURES2_ADD_ON_MODULE               0x00000020`
	`#define FEATURES2_ANALOG_VOLTAGES             0x00000040`

[108] Since firmware version 20.03.

Command name	**CMD_QUERY_STATE527_EX3**											
Compatibility	New MCA527 command (since firmware version 20.01).											
Execution right	Not necessary											
Description	The command reads the state.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	3F	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Fixed trigger threshold for direct input						unsigned short		0 … 16383 0 = auto threshold calculation			
Byte offset 2	Fixed baseline for direct input						unsigned short		0 … 16383 0 = baseline restoring			
Byte offset 4	Firmware version of the add-on module[109]						unsigned char		The information about the add-on module is valid if the flag FEATURES2_ADD_ON_MODULE is set. See CMD_QUERY_STATE527_EX2			
Byte offset 5	Hardware version of the add-on module[109]						unsigned char					
Byte offset 6	Hardware ID of the add-on module[109]						unsigned char					
Byte offset 7	unused						1 byte					
Byte offset 8	Allowed purposes of extension port part A[110]						unsigned short		Bit 0 is always 1. Bit 1 ... 15 is 1 if the corresponding purpose is allowed. For the numbers of the purposes see CMD_SET_EXTENSION_PORT.			
Byte offset 10	Allowed purposes of extension port part B[110]						unsigned short					
Byte offset 12	Allowed purposes of extension port part C[110]						unsigned short					
Byte offset 14	Allowed purposes of extension port part D[110]						unsigned short					
Byte offset 16	Allowed purposes of extension port part E[110]						unsigned short					
Byte offset 18	Allowed purposes of extension port part F[110]						unsigned short					
Byte offset 20	unused						86 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	MCA state						unsigned short		See CMD_QUERY_POWER			
Byte offset 130	unused						2 bytes					

[109] Since firmware version 20.02.

[110] Since firmware version 20.03.

Command name	**CMD_QUERY_USER_DATA**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads sixteen 32 bit values from the MCA memory. e: 0 ... 255 start entry number											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		e		0				end flag	
Byte string (HEX)	A5	5A	5E	00	e	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Sixteen 32 bit values						32 bit [16]					
Byte offset 64	unused						42 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

Command name	**CMD_QUERY_SPECTRA**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Conditionally necessary (see buffer control and remarks)											
Description	The command reads the spectrum data. n: Number of first channel and buffer control (Bit 15 ... 12) c: Compress factor (1 ... 128)											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		n		c		0		end flag	
Byte string (HEX)	A5	5A	5B	00	n_l	n_h	c_l	c_h	00	00	B9	9B
Buffer control	See CMD_QUERY_SPECTRA_EX (next command).											
Result data array												
Byte offset 0	Channels counts						unsigned long [32]					
Byte offset 128	Buffer state						unsigned short		OCCUPIED = 0x2000 OVERRUN = 0x4000 FILLED = 0x8000			
Byte offset 130	Checksum[111]						unsigned short					
Remarks	The compress factor determines how many channels each are combined to one value. This value is the maximum of the combined channels. The transfer of combined channels can be used to reduce data transfer if the data are only intended for displaying the accumulation of the spectrum during a measurement. If the spectrum data are intended for evaluation, the whole spectrum with compress factor 1 has to be transferred. See also the remarks of CMD_QUERY_SPECTRA_EX (next command).											

[111] This checksum is calculated differently from the other commands. See the introduction at the beginning of the document.

<table>
<tr><td>Command name</td><td colspan="12"><b>CMD_QUERY_SPECTRA_EX</b></td></tr>
<tr><td>Compatibility</td><td colspan="12">New MCA527 command.</td></tr>
<tr><td>Execution right</td><td colspan="12">Conditionally necessary (see buffer control and remarks)</td></tr>
<tr><td>Description</td><td colspan="12">The command reads the spectrum data.
n: Number of first channel
c: Compress factor (1 ... 128)
b: Buffer control: item [bit 4…0], index [bit 8…5], reserved [bit 13…9], flags [bit 15…14]</td></tr>
<tr><td>Format</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td><td colspan="2">integer</td></tr>
<tr><td>Parameter</td><td colspan="2">preamble</td><td colspan="2">command</td><td colspan="2">n</td><td colspan="2">c</td><td colspan="2">b</td><td colspan="2">end flag</td></tr>
<tr><td>Byte string (HEX)</td><td>A5</td><td>5A</td><td>02</td><td>01</td><td>n_l</td><td>n_h</td><td>c_l</td><td>c_h</td><td>b_l</td><td>b_h</td><td>B9</td><td>9B</td></tr>
<tr><td rowspan="20">Buffer control (item)</td><td colspan="12">The buffer control applies to CMD_QUERY_SPECTRA and CMD_QUERY_SPECTRA_EX, but items higher than 15 are only usable with CMD_QUERY_SPECTRA_EX.</td></tr>
<tr><td>Item</td><td colspan="5">Meaning</td><td colspan="2">MCA mode</td><td colspan="2">MCS mode</td><td colspan="2">Execution right</td></tr>
<tr><td>0</td><td colspan="5">Read spectrum</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>1</td><td colspan="5">Read MCS amplitude spectrum</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>2</td><td colspan="5">Read spectrum buffer and unlock buffer</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>3</td><td colspan="5">Read spectrum buffer and lock buffer</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>6</td><td colspan="5">Read MCS amplitude spectrum buffer and unlock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>7</td><td colspan="5">Read MCS amplitude spectrum buffer and lock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>8</td><td colspan="5">Read rejected spectrum</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>9</td><td colspan="5">Read rejected MCS amplitude spectrum</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>10</td><td colspan="5">Read rejected spectrum buffer and unlock buffer</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>11</td><td colspan="5">Read rejected spectrum buffer and lock buffer</td><td colspan="2">available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>14</td><td colspan="5">Read rejected MCS amplitude spectrum buffer and unlock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>15</td><td colspan="5">Read rejected MCS amplitude spectrum buffer and lock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>17</td><td colspan="5">Read MCS counter1 spectrum</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>18</td><td colspan="5">Read MCS counter1 spectrum buffer and unlock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>19</td><td colspan="5">Read MCS counter1 spectrum buffer and lock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>21</td><td colspan="5">Read MCS counter2 spectrum</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">not necessary</td></tr>
<tr><td>22</td><td colspan="5">Read MCS counter2 spectrum buffer and unlock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
<tr><td>23</td><td colspan="5">Read MCS counter2 spectrum buffer and lock buffer</td><td colspan="2">not available</td><td colspan="2">available</td><td colspan="2">necessary</td></tr>
</table>

<table>
<tr><td>Command name</td><td colspan="5"><b>CMD_QUERY_SPECTRA_EX</b> <i>(Continuation)</i></td></tr>
<tr><td rowspan="4">Buffer control (item)</td><td>Item</td><td>Meaning</td><td>MCA mode</td><td>MCS mode</td><td>Execution right</td></tr>
<tr><td>25</td><td>Read MCS counter3 spectrum[112]</td><td>not available</td><td>available</td><td>not necessary</td></tr>
<tr><td>26</td><td>Read MCS counter3 spectrum buffer and unlock buffer[112]</td><td>not available</td><td>available</td><td>necessary</td></tr>
<tr><td>27</td><td>Read MCS counter3 spectrum buffer and lock buffer[112]</td><td>not available</td><td>available</td><td>necessary</td></tr>
<tr><td>Buffer control (index[113] [114])</td><td colspan="5">The index is only relevant for gating mode 'sort by time'. If the gating mode is not 'sort by time', the index should be zero. Indices 0 … 7 refer to the MCA spectra and index 15 refers to the MCS spectrum. The indices 8 … 14 are not allowed. See also command CMD_SET_GATING.</td></tr>
<tr><td>Buffer control (reserved[113])</td><td colspan="5">The reserved bits should by zero.</td></tr>
<tr><td>Buffer control (flags)</td><td colspan="5">Bit 15: 0 = maximum of the compressed channels
1 = sum of the compressed channels[115]
Bit 14: 0 = 32 channels [unsigned long]
1 = 64 channels [unsigned short][116]</td></tr>
<tr><td>Result data array</td><td colspan="5"></td></tr>
<tr><td>Byte offset 0</td><td colspan="2">Channels counts</td><td>unsigned long [32]
or
unsigned short [64]</td><td colspan="2">The data format is depending on bit 14 of buffer control.</td></tr>
<tr><td>Byte offset 128</td><td colspan="2">Buffer state</td><td>unsigned short</td><td colspan="2">OCCUPIED = 0x2000
OVERRUN = 0x4000
FILLED = 0x8000</td></tr>
<tr><td>Byte offset 130</td><td colspan="2">Checksum[117]</td><td>unsigned short</td><td colspan="2"></td></tr>
<tr><td>Remarks</td><td colspan="5">This command has been added because CMD_QUERY_SPECTRA does not support more than 4096 channels and its buffer control parameter is limited.
If the MCA527 measures in repeat mode, the execution right is necessary, no matter the value of buffer control.
The capability to lock and unlock the buffer is relevant if the MCA527 measures in repeat mode. It is also used with the command CMD_QUERY_EXTENSION_RS232_RX.
Locking the buffer makes sure that the buffer is not overwritten with the data of the next measuring cycle before the buffer is not unlocked again.
If the MCA527 measures in a repeat mode which writes to the microSD memory card, the access to the buffer is denied. In this case the buffer is exclusively controlled by the MCA527.
This remarks are also applicable for CMD_QUERY_SPECTRA.</td></tr>
</table>

[112] Since firmware version 15.06. Micro version only.

[113] Since firmware version 14.02

[114] Since firmware version 16.00, the index is 4 bits long.

[115] Since firmware version 12.03.

[116] Since firmware version 12.04.

[117] This checksum is calculated differently from the other commands. See the introduction at the beginning of the document.

Command name	**CMD_QUERY_SPECTRA_EX2**											
Compatibility	New MCA527 command (since firmware version 16.00).											
Execution right	Conditionally necessary (see buffer control and remarks of CMD_QUERY_SPECTRA_EX)											
Description	The command reads the spectrum data. The meaning of all parameters are identical to the parameters of CMD_QUERY_SPECTRA_EX. The difference between CMD_QUERY_SPECTRA_EX and CMD_QUERY_SPECTRA_EX2 is the number of the returned spectrum data bytes.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		n		c		b		end flag	
Byte string (HEX)	A5	5A	38	01	n_l	n_h	c_l	c_h	b_l	b_h	B9	9B
Result data array												
Byte offset 0	Channels counts						unsigned long [256] or unsigned short [512]		The data format is depending on bit 14 of buffer control.			
Byte offset 1024	Buffer state						unsigned short		OCCUPIED = 0x2000 OVERRUN = 0x4000 FILLED = 0x8000			
Byte offset 1026	Command flag and parameters						8 bytes					
Byte offset 1034	Checksum						unsigned short					
Remarks	See the remarks of CMD_QUERY_SPECTRA_EX (previous command).											

Name	**CMD_QUERY_EXTENSION_RS232_RX**											
Compatibility	New MCA527 command.											
Execution right	Conditionally necessary (see remarks)											
Description	The command returns the last 1024 bytes received by the RS232 interface of the extension port. b: Buffer control: 0 = current data 1 = buffered data 2 = buffered data, unlock buffer 3 = buffered data, lock buffer											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		b		0				end flag	
Byte string (HEX)	A5	5A	25	01	b_l	b_h	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Number of bytes received since the last RS232 data transmission.											
Byte offset 2	Data						1024 bytes					
Byte offset 1026	Command flag and parameters						8 bytes					
Byte offset 1034	Checksum						unsigned short					
Remarks	If the MCA527 measures in repeat mode, the execution right is necessary. The capability to lock and unlock the buffer is relevant if the MCA527 measures in repeat mode. It is also used with the commands CMD_QUERY_SPECTRA and CMD_QUERY_SPECTRA_EX. Locking the buffer makes sure that the buffer is not overwritten with the data of the next measuring cycle before the buffer is not unlocked again. If the MCA527 measures in a repeat mode which writes to the microSD memory card, the access to the buffer is denied. In this case the buffer is exclusively controlled by the MCA527.											

Name	**CMD_QUERY_HISTOGRAM**											
Compatibility	New MCA527 command.											
Execution right	Not necessary											
Description	The command causes the MCA to acquire a histogram from 500 000 ADC samples (= 50 ms). s: start (0 ... 16 384 – 256 * class interval width) c: class interval width (1, 2, 4 ... 64) The command is ignored and responds with an error value if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		s		c		0		end flag	
Byte string (HEX)	A5	5A	09	01	s_l	s_h	c	00	00	00	B9	9B
Result data array												
Byte offset 0	Sample counts within the 256 class intervals						unsigned long [256]					
Byte offset 1024	Command flag and parameters						8 bytes					
Byte offset 1032	Checksum						unsigned short					

Command name	**CMD_QUERY_VOLTAGE_CURRENT**											
Compatibility	Compatible to the MCA166 command. The command is irrelevant for the MCA527. It has only been retained for compatibility with older software.											
Execution right	Not necessary											
Description	The command reads the voltages and currents.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		0		0		0		end flag	
Byte string (HEX)	A5	5A	05	00	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	unused						106 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

Command name	**CMD_QUERY_CENTROID**											
Compatibility	Identical to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads the centroid of a ROI (beg<end, LLD<=beg, end<=ULD, end-beg<250).											
Format	integer		integer		integer		integer		integer		integer	
Parameter	preamble		command		beg		end		0		end flag	
Byte string (HEX)	A5	5A	5F	00	beg_l	beg_h	end_l	end_h	00	00	B9	9B
Result data array												
Byte offset 0	Peak centroid within the given ROI						float					
Byte offset 4	unused						102 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

Command name	**CMD_QUERY_ENRICHMENT**											
Compatibility	Identical to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads current dead time, real time and the integral of two ROIs. This is very useful during a running measurement because in this way there is no need for transferring parts of the spectrum data for calculating the ROI integrals itself. The name of the command refers to its original usage. The command is, of course, usable for all applications that need this information. r1b: ROI_1 begin r2b: ROI_2 begin r1l: ROI_1 length (8 ... 250 channels) r2l: ROI_2 length (8 ... 250 channels)											
Format	integer		integer		char	char	integer		integer		integer	
Parameter	Preamble		command		r1l	r2l	r1b		r2b		end flag	
Byte string (HEX)	A5	5A	5D	00	r1l	r2l	$r1b_l$	$r1b_h$	$r2b_l$	$r2b_h$	B9	9B
Result data array												
Byte offset 0	Dead time						unsigned long					
Byte offset 4	Real time						unsigned long					
Byte offset 8	Integral of the 1st ROI						unsigned long					
Byte offset 12	Integral of the 2nd ROI						unsigned long					
Byte offset 16	unused						90 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

Name	**CMD_QUERY_UF6_ROIS**											
Compatibility	Identical to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads begin and end of up to three ROIs that can be set by the command CMD_SET_UF6_ROIS. The name of the command refers to its original usage. See also the command CMD_QUERY_UF6_INFO.											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	65	00	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	1st ROI begin						unsigned long					
Byte offset 4	1st ROI end						unsigned long					
Byte offset 8	2nd ROI begin						unsigned long					
Byte offset 12	2nd ROI end						unsigned long					
Byte offset 16	3rd ROI begin						unsigned long					
Byte offset 20	3rd ROI end						unsigned long					
Byte offset 24	unused						82 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

Name	**CMD_QUERY_UF6_INFO**											
Compatibility	Compatible to the MCA166 command.											
Execution right	Not necessary											
Description	The command reads current dead time, real time, the integral, begin and end of up to three ROIs. This is very useful during a running measurement because in this way there is no need for transferring parts of the spectrum data for calculating the ROI integrals itself. The name of the command refers to its original usage. The command is, of course, usable for all applications that need this information.											
Format	integer		integer		integer		long				integer	
Parameter	preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	66	00	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Dead time [msec]						unsigned long					
Byte offset 4	Real time [sec]						unsigned long					
Byte offset 8	Integral of the 1st ROI						unsigned long					
Byte offset 12	Integral of the 2nd ROI						unsigned long					
Byte offset 16	Integral of the 3rd ROI						unsigned long					
Byte offset 20	1st ROI begin						unsigned long					
Byte offset 24	1st ROI end						unsigned long					
Byte offset 28	2nd ROI begin						unsigned long					
Byte offset 32	2nd ROI end						unsigned long					
Byte offset 36	3rd ROI begin						unsigned long					
Byte offset 40	3rd ROI end						unsigned long					
Byte offset 44	Fractional digits of the real time [msec][118]						unsigned long		See CMD_QUERY_STATE527_EX, byte offset 82			
Byte offset 48	Area of the 1st ROI[118]						unsigned long					
Byte offset 52	Area error of the 1st ROI[118]						unsigned long					
Byte offset 56	Area of the 2nd ROI[118]						unsigned long					
Byte offset 60	Area error of the 2nd ROI[118]						unsigned long					
Byte offset 64	Area of the 3rd ROI[118]						unsigned long					
Byte offset 68	Area error of the 3rd ROI[118]						unsigned long					
Byte offset 72	unused						34 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	unused						4 bytes					

[118] Since firmware version 14.02.

Name	**CMD_QUERY_COMMON_MEMORY**											
Compatibility	New MCA527 command.											
Execution right	Not necessary											
Description	The command reads 1440 bytes from the common memory. o: offset (0 ... common memory size / 2 – 720) The firmware multiplies the passed offset by two to get even memory addresses.											
Format	integer		integer		long				integer		integer	
Parameter	Preamble		command		o				0		end flag	
Byte string (HEX)	A5	5A	07	01	o_l	...	...	o_h	00	00	B9	9B
Result data array												
Byte offset 0	1440 data bytes						1440 bytes					
Byte offset 1440	Command flag and parameters						8 bytes					
Byte offset 1448	Checksum						unsigned short					
Remarks	The common memory is used for the general modes 1 = Transient recorder (records ADC raw data), 3 = List mode 1: Time stamp recorder (level triggered signals) 4 = List mode 2: Time stamp recorder (edge triggered signals) 5 = List mode 3: Time stamp recorder (AHRC = analog high rate counting) 6 = List mode 4: List with spectroscopic and time information 7 = List mode 5: List with spectroscopic data pairs measured with two different flat top times for the evaluation filter **In transient recorder mode,** the common memory contains a sequence of 2 bytes long ADC raw data. **In list modes 1 to 3,** the common memory contains a list of time intervals between two detected events. Each item is 1 to 4 bytes long. The bytes are arranged in big-endian order. The first byte always contains the code for the byte length of the value[119]. `1 byte:  XXXXXXXX                               0 …       191` `2 bytes: 11XXXXXX XXXXXXXX                    192 …     12479` `3 bytes: 1111XXXX XXXXXXXX XXXXXXXX         12480 …    798911` `4 bytes: 111111XX XXXXXXXX XXXXXXXX XXXXXXXX 798912 …  67907775` Each value represents the number of time units which are bygone since the previous event. Values of 67,907,775 means that no event has occurred for 67,907,775 time units. Only the next value less than 67,907,775 marks the next event. The time unit length can be calculated from the 'ADC sampling rate' (see CMD_QUERY_STATE527_EX, byte offset 104). **In list mode 4,** each list entry is an event information followed by a time information. The event information is either the calculated channel number within a 16k[120] spectrum or the information about events that do not result in a count within the spectrum. The time information is the time interval between two events If the event information is a channel number, it consists of two bytes: `0XXXXXXX XXXXXXXX`. The bytes are arranged in big-endian order. Bit 15 is always 0. Bit 14 has not been used so far. It should be 0. Bit 13 ... 0 stand for the calculated channel number.											

[119] For more detailed descriptions of the time information coding, see the document "MCA Binary Data Format", appendix A.

[120] The list mode 4 uses a 16k spectrum regardless of the setting at CMD_QUERY_STATE, byte offset 36.

Name	**CMD_QUERY_COMMON_MEMORY***(Continuation)*

Remarks	All other event information consists of one byte in the format `10XXXXXX` as listed following.
	`10000000` Event above the spectrum range.
	`10000001` Event below the spectrum range.
	`10000010` An event was triggered, but could not be evaluated because of pile up rejection.
	`10000011` An event was triggered, but could not be evaluated by the jitter correction algorithm.
	`10000100` An event was triggered, but could not be evaluated because of a subsequent event.
	`10000101` Beginning of an ADC overflow or underflow.
	`10000110` End of an ADC overflow or underflow[121].
	`10000111` Beginning of a discarded cycle[122].
	`10001000` Stop of the measurement because of reaching the preset real time.
	Besides, there is a single byte long non-event information in the format `11000000.` A non-event information stands for a time interval without events that cannot be coded with the used time coding method.
	The time information is stored with varying byte length. The bytes are arranged in big-endian order. The first byte always contains the code for the byte length of the value[119].
	`1 byte:  XXXXXXXX                                   0 …       191`
	`2 bytes: 11XXXXXX XXXXXXXX                        192 …     12479`
	`3 bytes: 1111XXXX XXXXXXXX XXXXXXXX             12480 …    798911`
	`4 bytes: 111111XX XXXXXXXX XXXXXXXX XXXXXXXX   798912 …  67907775`
	Each value represents the number of time units which are bygone since the previous event. Time intervals that exceeds the value of 67,907,775 are coded as non-event information. `11000000` means that no event has occurred for 67,907,776 time units. The time unit length can be calculated from the 'ADC sampling rate' (see CMD_QUERY_STATE527_EX, byte offset 104).
	In list mode 5, the common memory contains a list with spectroscopic data pairs. A data pair consist of two short values (little-endian). The first one is the result calculated with the evaluation filter with the first flat top time. The second one is the result calculated with the evaluation filter with the second flat top time.

[121] At the start of the measurement, the MCA527 firmware assumes an ADC overflow for the time before. Therefore, the list always begins with an 'end of ADC overflow' entry.

[122] If the processor is overloaded, it discards cycles of 4000 time units The time unit length can be calculated from the 'ADC sampling rate' (see CMD_QUERY_STATE527_EX, byte offset 104).

Name	**CMD_QUERY_OSCI_SCREEN**											
Compatibility	New MCA527 command.											
Execution right	Necessary if the resolution index that is set with CMD_SET_OSCI_TRIGGER is larger than 0.											
Description	The command reads the data for the oscilloscope screen. p: -1 or the previously returned position											
Format	integer		integer		long				integer		integer	
Parameter	Preamble		command		p				0		end flag	
Byte string (HEX)	A5	5A	12	01	p_l	...	...	p_h	00	00	B9	9B
Result data array												
Byte offset 0	Start position of the received data						unsigned long					
Byte offset 4	Start position of the subsequent data						unsigned long					
Byte offset 8	Data						unsigned short [500]					
Byte offset 1008	Command flag and parameters						8 bytes					
Byte offset 1016	Checksum						unsigned short					
Remarks	The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.											

Name	**CMD_QUERY_OSCI_SCREEN_EX**											
Compatibility	New MCA527 command.											
Execution right	Necessary if flags are set.											
Description	The command reads the data for the oscilloscope screen. Flags: Bit 0: convolve data with trigger filter Bit 1: convolve data with main filter											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		flags		0		0		end flag	
Byte string (HEX)	A5	5A	29	01	flags$_l$	flags$_h$	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Start position of the received data						unsigned long					
Byte offset 4	Data						unsigned short [720]					
Byte offset 1444	Command flag and parameters						8 bytes					
Byte offset 1452	Checksum						unsigned short					
Remarks	This command has been added to extend the features of the oscilloscope. Corresponding to the flags, the MCA527 convolves the data with the corresponding filters. The results can be subsequently read with the commands CMD_QUERY_OSCI_TRIGGER_FILTER_RESULTS and CMD_QUERY_OSCI_MAIN_FILTER_RESULTS. The command is only usable if the resolution index that is set with CMD_SET_OSCI_TRIGGER is not larger than 0. The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.											

Name	**CMD_QUERY_OSCI_TRIGGER_FILTER_RESULTS**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command reads data that are calculated from the oscilloscope data during the last call of the command CMD_QUERY_OSCI_SCREEN_EX if the corresponding flag is set. Range: 0: results [0 … 359] 1: results [360 … 719]											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		range		0		0		end flag	
Byte string (HEX)	A5	5A	2A	01	range$_l$	range$_h$	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Applied trigger threshold [* 0.00006103515625]						long					
Byte offset 4	Results [* 0.00006103515625]						long [360]					
Byte offset 1444	Command flag and parameters						8 bytes					
Byte offset 1452	Checksum						unsigned short					
Remarks	In order to read all data, the command has to be called twice. The first time with range=0 and the second time with range=1. The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.											

Name	**CMD_QUERY_OSCI_MAIN_FILTER_RESULTS**											
Compatibility	New MCA527 command.											
Execution right	Necessary											
Description	The command reads data that are calculated from the oscilloscope data during the last call of the command CMD_QUERY_OSCI_SCREEN_EX if the corresponding flag is set.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		0		0		0		end flag	
Byte string (HEX)	A5	5A	2B	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Results						short [720]					
Byte offset 1440	Jitter[123]						short					
Byte offset 1442	Command flag and parameters						8 bytes					
Byte offset 1450	Checksum						unsigned short					
Remarks	The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.											

[123] The value is up to now unused. It is always 0.

Name	**CMD_QUERY_AHRC_HISTOGRAM**											
Compatibility	New MCA527 command (since firmware version 13.08).											
Execution right	Necessary											
Description	The command causes the MCA to acquire a histogram from the areas of all detected events within an acquisition time of 0.8 seconds. The area is equivalent to the sum of the differences between the ADC value and the baseline from the point where the signal curve exceeds the threshold to the point where the signal curve falls below the threshold again. b: bin interval width (1, 2, 4 ... 32 768) The command is ignored and responds with an error value if the general mode is unequal to “Time stamp recorder (AHRC = analog high rate counting)” (= 5) or if a measurement is still running.											
Format	integer		integer		integer		integer		integer		integer	
Parameter	Preamble		command		c		0		0		end flag	
Byte string (HEX)	A5	5A	31	01	b_l	b_h	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Results						unsigned long [360]					
Byte offset 1440	Command flag and parameters						8 bytes					
Byte offset 1448	Checksum						unsigned short					
Remarks	The histogram is used to find the best values for the AHRC threshold and the AHRC group widths in an approximation procedure. The parameters can be set with CMD_SET_AHRC_PARAM. More about AHRC, see the remarks on this command.											

Name	**CMD_QUERY_DETECTOR_INFO**											
Compatibility	New MCA527 command (since firmware version 14.03).											
Execution right	Not necessary											
Description	The command returns 1024 bytes of the detector information. Range: 0: [0 … 1023] 1: [1024 … 2047]											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		range		0				end flag	
Byte string (HEX)	A5	5A	33	01	$range_l$	$range_h$	00	00	00	00	B9	9B
Result data array												
Byte offset 0	Data						1024 bytes					
Byte offset 1024	Command flag and parameters						8 bytes					
Byte offset 1032	Checksum						unsigned short					
Remarks	There are detectors that supply detector information that are stored in an integrated EEPROM with a maximum size of 2048 bytes. The structure of the detector information is described in an extra document. The flag FEATURES_DETECTOR_INFO_VALID of the parameter “MCA features” (byte offset 8) returned by the command CMD_QUERY_STATE527 indicates if valid detector information are available.											

Name	**CMD_QUERY_ADJUSTMENT_TABLE**											
Compatibility	New MCA527 command.											
Execution right	Not necessary											
Description	The command returns the ADC values that has been measured with different settings at open signal input. These data can be used to interpolate the ADC value which corresponds to the ground line. A value is valid if it is unequal to -1.											
Format	integer		integer		integer		long				integer	
Parameter	Preamble		command		0		0				end flag	
Byte string (HEX)	A5	5A	26	01	00	00	00	00	00	00	B9	9B
Result data array												
Byte offset 0	measured with coarse gain = 2, offset DAC = 1024						short					
Byte offset 2	measured with coarse gain = 2, offset DAC = 15360						short					
Byte offset 4	measured with coarse gain = 5, offset DAC = 1024						short					
Byte offset 6	measured with coarse gain = 5, offset DAC = 15360						short					
Byte offset 8	measured with coarse gain = 10, offset DAC = 1024						short					
Byte offset 10	measured with coarse gain = 10, offset DAC = 15360						short					
Byte offset 12	measured with coarse gain = 20, offset DAC = 1024						short					
Byte offset 14	measured with coarse gain = 20, offset DAC = 15360						short					
Byte offset 16	measured with coarse gain = 50, offset DAC = 1024						short					
Byte offset 18	measured with coarse gain = 50, offset DAC = 15360						short					
Byte offset 20	measured with coarse gain = 100, offset DAC = 1024						short					
Byte offset 22	measured with coarse gain = 100, offset DAC = 15360						short					
Byte offset 24	unused						82 bytes					
Byte offset 106	Command flag and parameters						8 bytes					
Byte offset 114	unused						12 bytes					
Byte offset 126	Checksum						unsigned short					
Byte offset 128	MCA state						unsigned short		See CMD_QUERY_POWER			
Byte offset 130	unused						2 bytes					