MCA-166, MCA-527

Multi-Channel Analyzers

Description of the MCA Communication DLL

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: 2023-03-01

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

Table of Contents

1 MS-Windows DLL for Communication with MCA166 and MCA527

The dynamic link library (mca32com.dll) is provided for Win32 applications. It must reside in the Windows system directory or in the same directory as the calling application. In order to include the DLL in your program, you have to use the file MCA32COM.BAS for Visual Basic or the files MCA_COMM.H and MCA32COM.LIB for C/C++. All functions and structures are declared in these files.

The description of the communication DLL refers to version 2.14.

2 Initializing and Closing the Communication Port

Before you can use any other function, the communication port must be initialized. There are three functions for initializing the communication port: **COMM_INIT_EX2** initializes a RS232 or USB communication port. **COMM_INIT_ETHERNET** initializes an Ethernet communication port. **COMM_INIT_SELECT** is the most recently added function and usable for all possible communication ports. It also offers the most comfort for the developer. For this reason, it is the most commendable function.

Before the application will be exited, the communication port has to be closed with **COMM_CLOSE**. Only one communication port can be used at the same time. If the communication port shall be changed, the current communication port has to be closed before the new one can be initialized.

Name	**COMM_INIT_EX2**
Description	The function initializes the RS232 or USB communication.

Declaration	`short CALLBACK COMM_INIT_EX2(LPSTR lpszInterface,clock_t lTimeOut,`
	`int nTryAgain,unsigned long ulBaudRate);`

`lpszInterface`	Interface name (e. g. “COM1”)
`lTimeOut`	Time out [milliseconds] for the communication. It should be set to about 1000 milliseconds, because there are commands that run a short measurement before they return. The parameter can be set to a higher value if needed.
`nTryAgain`	Number of tries to send the command
`ulBaudRate`	Baud rate (38.400, 115.200, 307.200 or 3.000.000)
Return value	If the function succeeds, the return value is nonzero.

Name	**COMM_INIT_ETHERNET**
Description	The function initializes the Ethernet communication.

Declaration	`short CALLBACK COMM_INIT_ETHERNET(unsigned short unNumberToFind,`
	`clock_t lTimeOut,int nTryAgain,HWND hWnd);`

`unNumberToFind`	Serial number of the MCA527 which shall be connected to the application. If the parameter is unequal 0, the function exclusively tries to connect to the MCA527 with this serial number. If the parameter is equal 0, the function searches for all available MCA527. If MCA527's are available, they are displayed in a list, from what the user can choose the desired one.
`lTimeOut`	Time out [milliseconds] for the communication. It should be set to about 1000 milliseconds, because there are commands that run a short measurement before they return. The parameter can be set to a higher value if needed.
`nTryAgain`	Number of tries to send the command
`hWnd`	Handle to the parent window
Return value	If the function succeeds, the return value is nonzero.

Name	**COMM_INIT_SELECT**
Description	The function initializes the communication with any MCA. The function detects all available MCA's.

Declaration	`short CALLBACK COMM_INIT_SELECT(MCA_COMM_INIT* init,clock_t lTimeOut,`
	`int nTryAgain,HWND hWnd);`

`init`	Pointer to a MCA_COMM_INIT structure.
`lTimeOut`	Time out [milliseconds] for the communication. It should be set to about 1000 milliseconds, because there are commands that run a short measurement before they return. The parameter can be set to a higher value if needed.
`nTryAgain`	Number of tries to send the command
`hWnd`	Handle to the parent window
Return value	If the function succeeds, the return value is nonzero.

Name	**MCA_COMM_INIT**
Description	A pointer of the structure has to be passed to the `COMM_INIT_SELECT` function.

Declaration	`struct MCA_COMM_INIT`
	`{ unsigned short  unSize;`
	`  unsigned short  unNumberToFind;`
	`  VB_BOOL         bMca166;`
	`  unsigned short  unMca166FwVersion;`
	`  VB_BOOL         bMca527;`
	`  unsigned short  unMca527FwVersion;`
	`  short           nShowLogo;`
	`  short           nUserInterface;`
	`  unsigned short  unInterfaceId;`
	`  unsigned long   ulBaudrate;`
	`  char            szHelpFile[MAX_PATH];`
	`  unsigned long   ulHelpId;`
	`};`

`unSize`	[In] Size of the structure.
`unNumberToFind`	[In] If this number is nonzero, the function tries to initializes the communication to the MCA with the corresponding serial number. If the communication with this MCA is possible, the communication is initialized and the function returns. If the corresponding MCA is not detected or not usable, all detected MCA are listed within a dialog window. If this number is zero, all detected MCA are always listed within a dialog window.
`bMca166`	[In] If this parameter is nonzero, the MCA-166 is assumed to be applicable.
`unMca166FwVersion`	[In] This parameter sets the earliest MCA-166 firmware version that is assumed to be applicable.
`bMca527`	[In] If this parameter is nonzero, the MCA-527 is assumed to be applicable.
`unMca527FwVersion`	[In] This parameter sets the earliest MCA-527 firmware version that is assumed to be applicable.
`nShowLogo`	[In] If this parameter is equal to 0, no logo is displayed. If this parameter is equal to -1, the GBS logo is displayed.
`nUserInterface`	[In] This parameter can be one of the following values: USER_INTERFACE_VERBOSE (= -1): The function displays message windows for signaling the work progress and asks for the serial number if necessary. USER_INTERFACE_RETICENT (= 0): The function asks for the serial number if necessary. USER_INTERFACE_MUTE (= 1): There is no interaction with the user. This value must not used if `unNumberToFind` is equal to 0.
`unInterfaceId`[1]	[In] If this parameter is unequal to 0xFFFF, it determines the communication port that shall be used. [Out] This parameter returns the ID of the used communication port.

[1] See structure `MCA_COMM_PARAM`

Name	**MCA_COMM_INIT** ***(Continuation)***
`ulBaudrate`	[Out] If the communication port is RS232, RS485 or USB, the parameter returns the current baud rate.
`szHelpFile`	[In] If this parameter contains the path and the name of a help file (*.hlp) and the help index is unequal to 0xFFFFFFFF, the help button within the dialog window is enabled. The existence of the help file is not checked.
`ulHelpId`	[In] If this parameter is unequal to 0xFFFFFFFF and the previous parameter does not contain an empty string, the help button within the dialog window is enabled.

Name	**COMM_CLOSE**
Description	The function closes the communication.
Declaration	`void CALLBACK COMM_CLOSE();`

Name	**GET_COMM_PARAM**
Description	The function returns the communication parameters.
Declaration	`void CALLBACK GET_COMM_PARAM(MCA_COMM_PARAM* param);`
`param`	Pointer to a MCA_COMM_PARAM structure.

Name	**MCA_COMM_PARAM**
Description	The structure contains the MCA communication parameters. A pointer of the structure is passed to the `GET_COMM_PARAM` function.

Declaration	`struct MCA_COMM_PARAM`
	`{ unsigned short  unSize;`
	`  unsigned short  unInterfaceId;`
	`  unsigned long   ulBaudrate;`
	`  unsigned short  unComPortNumber;`
	`};`

`unSize`	Size of the structure.
`unInterfaceId`	This parameter identifies the used communication interface. 0 = Ethernet 1 … 256 = RS232 (COM1 … COM256) 257 = USB, the MCA battery cannot be charged[2] 258 = USB, the MCA battery can be charged 259 … 514 = RS485 (COM1 … COM256) 515 = USB, RS485 adapter
`ulBaudrate`	The baud rate is valid with RS232 and USB interface.
`unComPortNumber`	The virtual port number (COM1 … COM256) of the USB interface.

Name	**SET_TIMEOUT**
Description	The function sets the time out for the communication.
Declaration	`void CALLBACK SET_TIMEOUT(clock_t lTimeOut);`
`lTimeOut`	Time out [milliseconds] for the communication. It should be set to about 1000 milliseconds, because there are commands that run a short measurement before they return. The parameter can be set to a higher value if needed.

[2] The MCA can only be charged over USB if the computer is connected directly with the USB connector and not only via a USB adapter with the RS232 connector.

3 Sending Firmware Commands to the MCA

For information about MCA firmware commands (CMD_XXX) see document "Description of the MCA527 Firmware Commands" or appendix "Description of the MCA166 firmware functions" within the MCA-166 User's Manual.

Name	**MCA_COMM, MCA_COMM6**
Description	These functions allow to send single firmware commands to the MCA. These function are only intended for testing. The function MCA_COMM6 has been added because new MCA527 commands need more than 4 parameters. For the real work with the MCA you should use the other functions (MMCA_ ...) described in chapter 4. Regardless of its type, each parameter of the firmware command is passed to one unsigned long parameter each. It will be cast within the function to the required type.

Declaration	`ERROR_FLAG CALLBACK MCA_COMM(LPSTR command,LPSTR rec_data,`
	`unsigned long param1=0,unsigned long param2=0,`
	`unsigned long param3=0,unsigned long param4=0);`
	`ERROR_FLAG CALLBACK MCA_COMM6(LPSTR command,LPSTR rec_data,`
	`unsigned long param1=0,unsigned long param2=0,`
	`unsigned long param3=0,unsigned long param4=0`
	`unsigned long param5=0,unsigned long param6=0);`

`command`	Command name (e. g. "CMD_START")
`rec_data`	Pointer to RECDATA or command specific structure in which the received data is returned.
`param1`	1st parameter if required
`param2`	2nd parameter if required
`param3`	3rd parameter if required
`param4`	4th parameter if required
`param5`	5th parameter if required
`param6`	6th parameter if required
Return value	ERROR_OK = 0 successful data transfer ERROR_INTERFACE = 1 communication port is not initialized ERROR_UNKNOWN_COMMAND = 2 unknown command ERROR_COMMUNICATION = 3 faulty data transfer ERROR_INVALID_PARAM = 4 invalid parameter ERROR_RUNNING_MEAS = 5 measuring is running, but stopped measurement is required for this command ERROR_VIOLATED_RIGHT = 6 execution right violation ERROR_STOPPED_MEAS = 7 measurement is stopped, but running measurement is required for this command ERROR_WRONG_MODE = 8 wrong mode for using this command ERROR_UNHANDLED_COMMAND = 9 not handled by this firmware version ERROR_FILE_WRITING_IN_PROCCESS = 10 file writing is in process; this command must not be called before the process is finished

4 Functions to work with the MCA

For the work with the MCA a lot of functions are available. The functions can be subdivided in five groups:

- Function for resetting the MCA (MMCA_RESET)
- Functions for starting and stopping the data acquisition (MMCA_..._AQUIRE)
- Functions for clearing (MMCA_CLEAR_...)
- Functions for setting (MMCA_SET_...)
- Functions for query (MMCA_QUERY_...)

The query functions are handed over a reference to a specific data structure. You can derive the name of the structure from the name of the function, by leaving out "MMCA_" from the function name. (e. g. function MMCA_QUERY_STATE requires the data structur QUERY_STATE). You can also find important information in the files mca_comm.h (C/C++) and mca32com.bas (Visual Basic).

For information about MCA firmware commands (CMD_XXX) see document "Description of the MCA527 Firmware Commands" or appendix "Description of the MCA166 firmware functions" within the MCA-166 User's Manual.

MCA Reset function:

Name	**MMCA_RESET**
Description	All MCA parameters will be reset to their initial state and all spectra are cleared. The measurement is aborted. MCA-166: The states of the preamplifier power and the value of the high voltages are unchanged! MCA-527: The preamplifier power and the high voltage are turned off!
Declaration	`ERROR_FLAG CALLBACK MMCA_RESET();`
Return value	See MCA_COMM function

MCA Acquire functions:

Name	**MMCA_START_ACQUIRE**
Description	The acquisition is started or continued with the current parameters (see CMD_START).

Declaration	`ERROR_FLAG CALLBACK MMCA_START_ACQUIRE(unsigned short unFlags,`
	`unsigned long ulStartTime);`

`unFlags`	Flags (see CMD_START)
`ulStartTime`	Start time (see CMD_START)
Return value	See MCA_COMM function
Remarks	Note, that the flags for MCA166 and MCA527 differ.

Name	**MMCA_START_NEW_SWEEP**
Description	See CMD_START_NEW_SWEEP.
Declaration	`ERROR_FLAG CALLBACK MMCA_START_NEW_SWEEP(LPSTR rec_data);`
`rec_data`	Pointer to a PREVIOUS_SWEEP_DATA data structure
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_START_NEW_SWEEP.

Name	**MMCA_STOP_ACQUIRE**
Description	The acquisition is stopped.
Declaration	`ERROR_FLAG CALLBACK MMCA_STOP_ACQUIRE();`
Return value	See MCA_COMM function

MCA Mode functions:

Name	**MMCA_SET_GENERAL_MODE**
Description	The function sets the general MCA527 mode.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_GENERAL_MODE(unsigned short unMode);`
`unMode`	General MCA527 mode (see CMD_SET_GENERAL)
Return value	See MCA_COMM function

Name	**MMCA_SET_MODE_MCA**
Description	The function sets the acquire mode to Multi-Channel Analyser.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MODE_MCA();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MODE_MCS**
Description	The function sets the acquire mode to Multi-Channel Scaler. The MCS input remains unchanged.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MODE_MCS();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MODE_MCS_TTL**
Description	The function sets the acquire mode to Multi-Channel Scaler and the MCS input to “extern TTL”.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MODE_MCS_TTL();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MODE_MCS_ICR**
Description	The function sets the acquire mode to Multi-Channel Scaler and the MCS input to “Input Rate”.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MODE_MCS_ICR();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MODE_MCS_DISCR**
Description	The function sets the acquire mode to Multi-Channel Scaler and the MCS input to “LLD/ULD”.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MODE_MCS_DISCR();`
Return value	See MCA_COMM function

MCA Clear functions:

Name	**MMCA_CLEAR_MEMORY**
Description	MCA166: The acquisition is stopped and the spectrum is cleared. MCA527: The acquisition is stopped and the spectrum, the real time and the dead time are cleared.
Declaration	`ERROR_FLAG CALLBACK MMCA_CLEAR_MEMORY();`
Return value	See MCA_COMM function

Name	**MMCA_CLEAR_TIME**
Description	MCA166: The acquisition is stopped and the real time and the dead time are cleared. MCA527: The acquisition is stopped and the spectrum, the real time and the dead time are cleared.
Declaration	`ERROR_FLAG CALLBACK MMCA_CLEAR_TIME();`
Return value	See MCA_COMM function

Name	**MMCA_CLEAR_ROI**
Description	The acquisition is stopped and the ROI limits are set to LLD and ULD
Declaration	`ERROR_FLAG CALLBACK MMCA_CLEAR_ROI();`
Return value	See MCA_COMM function

Name	**MMCA_CLEAR_ALL**
Description	The function is a combination of MMCA_CLEAR_ROI, MMCA_CLEAR_MEMORY and MMCA_CLEAR_TIME
Declaration	`ERROR_FLAG CALLBACK MMCA_CLEAR_ALL();`
Return value	See MCA_COMM function

MCA Setup functions:

Name	**MMCA_SET_PRESET_NONE**
Description	None automatic stop condition.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_NONE();`
Return value	See MCA_COMM function

Name	**MMCA_SET_PRESET_LIVE_TIME**
Description	The function sets the time for the automatic stop condition (dead time corrected).
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_LIVE_TIME(unsigned long ulLiveTime);`
`ulLiveTime`	Live time: MCA166: 1 ... 65535 sec MCA527: 1 ... 4294967295 sec
Return value	See MCA_COMM function

Name	**MMCA_SET_PRESET_REAL_TIME**
Description	The function sets the time for the automatic stop condition.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_REAL_TIME(unsigned long ulRealTime);`
`ulRealTime`	Real time: MCA166: 1 ... 65535 sec MCA527: 1 ... 4294967295 sec
Return value	See MCA_COMM function

Name	**MMCA_SET_PRESET_INTEGRAL**
Description	The function sets the value for the ROI integral
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_INTEGRAL(unsigned long ulIntegral);`
`ulIntegral`	Integral: 1 ... 4294967295
Return value	See MCA_COMM function

Name	**MMCA_SET_PRESET_AREA**
Description	The function sets the value for the ROI area
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_AREA(unsigned long ulArea);`
`ulArea`	Area: 1 ... 4294967295
Return value	See MCA_COMM function

Name	**MMCA_SET_PRESET_REAL_TIME_MILLISECONDS**
Description	The function sets the time for the automatic stop condition. The function is only usable for MCA527, firmware version 14.03 or higher.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PRESET_REAL_TIME_MILLISECONDS(unsigned long ulRealTime);`
`ulRealTime`	Real time: 1 ... 4294967295 msec
Return value	See MCA_COMM function

Name	**MMCA_SET_ADC_RES_DISCR**
Description	The function sets the ADC resolution and the software discriminator range.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_ADC_RES_DISCR(unsigned short unCountOfChannels,`
	`unsigned short unLLD,`
	`unsigned short unULD)`

`unCountOfChannels`	Channels: MCA166: 128, 256, 512, 1024, 2048 or 4096 MCA527: 128, … maximum channels count
`unLLD`	LLD: 0 <= LLD < ULD
`unULD`	ULD: MCA166: LLD < ULD < Channels - Channels/32 MCA527: LLD < ULD < Channels
Return value	See MCA_COMM function

Name	**MMCA_SET_ROI**
Description	The function sets the begin and end of the ROI for the preset integral and area.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_ROI(unsigned short unROIBegin,`
	`unsigned short unROIEnd);`

`unROIBegin`	ROI begin channel: LLD <= begin < end
`unROIEnd`	ROI end channel: LLD < end <= ULD
Return value	See MCA_COMM function

Name	**MMCA_SET_REPEAT**
Description	The function sets the number of sweeps for repetitive measurement.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_REPEAT(unsigned short unRepeat);`
`unRepeat`	Sweeps: 0 ... 65535 (0 = infinite)
Return value	See MCA_COMM function

Name	**MMCA_SET_MCS_CHANNEL**
Description	The function sets the number of channels for the MCS and Rate mode.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCS_CHANNEL(unsigned short unChannel);`
`unChannel`	MCS channels: MCA166: 128, 256, 512, 1024, 2048, 4096 MCA527: 1 … 16384
Return value	See MCA_COMM function

Name	**MMCA_SET_TIME_PER_CHANNEL**
Description	The function sets the dwell time per channel for the MCS mode.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TIME_PER_CHANNEL(unsigned short unTime);`
`unTime`	Real time per channel: 1 ... 65535 * 10 msec
Return value	See MCA_COMM function

Name	**MMCA_SET_TIME_PER_CHANNEL527**
Description	The function sets the dwell time per channel for the MCS mode.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TIME_PER_CHANNEL527(unsigned long ulTime);`
`ulTime`	Real time per channel: 1 ... 268435455 * 0.1 msec
Return value	See MCA_COMM function
Remark	This function is only usable for MCA527.

Name	**MMCA_SET_MCS_INPUT_ICR**
Description	The function selects the count rate signal as MCS input.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCS_INPUT_ICR();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCS_INPUT_TTL**
Description	The function selects the external TTL signals as MCS input.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCS_INPUT_TTL();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCS_INPUT_DISCR**
Description	The function selects the software discrimination of ADC-input pulses as MCS input.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCS_INPUT_DISCR();`
Return value	See MCA_COMM function

Name	**MMCA_SET_GAIN**
Description	The function sets the amplifier coarse and fine gain.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_GAIN(unsigned short unCoarseGain,`
	`unsigned short unFineGain);`

`unCoarseGain`	Coarse gain: 2, 5, 10, 20, 50, 100, 200, 500 or 1000
`unFineGain`	Fine gain: MCA166: 5000 ... 15000 * 0.0001 (if coarse gain=1000, max. fine gain=10000) MCA527: 5000 … 65000 * 0.0001
Return value	See MCA_COMM function

Name	**MMCA_SET_OFFSET_DAC**
Description	The function sets the offset DAC of the MCA527.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_OFFSET_DAC(unsigned short unValue);`
`unValue`	DAC value: 0 … 16383
Return value	See MCA_COMM function
Remarks	The function is only usable for MCA527.

Name	**MMCA_SET_MCA_INPUT_AMPLIFIER**
Description	The function sets the input to 'amplifier' (MCA166) or 'shaping' (MCA527).
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_AMPLIFIER();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCA_INPUT_AMPLIFIER_POS**
Description	The function sets the amplifier input polarity to positive.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_AMPLIFIER_POS();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCA_INPUT_AMPLIFIER_NEG**
Description	The function sets the amplifier input polarity to negative.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_AMPLIFIER_NEG();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCA_INPUT_DIRECT**
Description	The function sets the ADC input to direct.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_DIRECT();`
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. For MCA166 use exclusively MMCA_SET_MCA_INPUT_DIRECT_POS or MMCA_SET_MCA_INPUT_DIRECT_NEG.

Name	**MMCA_SET_MCA_INPUT_DIRECT_POS**
Description	The function sets the ADC input to direct (0 to +3V input range) and PUR to off.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_DIRECT_POS();`
Return value	See MCA_COMM function

Name	**MMCA_SET_MCA_INPUT_DIRECT_NEG**
Description	The function sets the ADC input to direct (0 to -3V input range) and PUR to off.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_MCA_INPUT_DIRECT_NEG();`
Return value	See MCA_COMM function

Name	**MMCA_SET_INPUT_POLARITY_POS**
Description	The function sets the input polarity to positive.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_INPUT_POLARITY_POS();`
Return value	See MCA_COMM function

Name	**MMCA_SET_INPUT_POLARITY_NEG**
Description	The function sets the ADC input polarity to negative.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_INPUT_POLARITY_NEG();`
Return value	See MCA_COMM function

Name	**MMCA_SET_THRESHOLD**
Description	The function sets the analog threshold .
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_THRESHOLD(unsigned short unThreshold);`
`unThreshold`	0 ... 60 %
Return value	See MCA_COMM function

Name	**MMCA_SET_THRESHOLD_TENTHS**
Description	The function sets the analog threshold .
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_THRESHOLD_TENTHS(unsigned short unThreshold);`
`unThreshold`	0 ... 600 * 0.1 %
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_SHAPING_TIME_LOW**
Description	The function sets the low shaping time. MCA166: ordinarily 1µs MCA527: value set by MMCA_SET_SHAPING_TIME_PAIR
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_SHAPING_TIME_LOW();`
Return value	See MCA_COMM function

Name	**MMCA_SET_SHAPING_TIME_HIGH**
Description	The function sets the high shaping time. MCA166: ordinarily 2µs MCA527: value set by MMCA_SET_SHAPING_TIME_PAIR
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_SHAPING_TIME_HIGH();`
Return value	See MCA_COMM function

Name	**MMCA_SET_SHAPING_TIME_PAIR**
Description	The function sets the values for the low and high shaping time of the MCA527. Afterwards, the shaping time is selected with MMCA_SET_SHAPING_TIME_LOW or MMCS_SET_SHAPING_TIME_HIGH. This practice seems to be circumstantial, but it considered the compatibility to the MCA166.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_SHAPING_TIME_PAIR(unsigned short unLow,`
	`unsigned short unHigh);`

`unLow`	Low shaping time: 1 … 254 (low shaping time < high shaping time)
`unHigh`	High shaping time: 2 … 255 (low shaping time < high shaping time)
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_TRIGGER_FILTER**
Description	The function sets the trigger filters used for low and high shaping time.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TRIGGER_FILTER(unsigned short unTriggerFilterLow,`
	`unsigned short unTriggerFilterHigh);`

`unTriggerFilterLow`	Trigger filter for low shaping time: 0 = (-1; +1), 1 = (-1; 0; +1), 2 = (+1; -2; +1), 3 = (+1; 0; -2; 0; +1)
`unTriggerFilterHigh`	Trigger filter for high shaping time: 0 = (-1; +1), 1 = (-1; 0; +1), 2 = (+1; -2; +1), 3 = (+1; 0; -2; 0; +1)
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_TRIGGER_PARAM**
Description	The function sets the trigger parameters..
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TRIGGER_PARAM(unsigned short unParam,unsigned long ulValue);`
`unParam`	Parameter to be set 0 = Trigger level for automatic threshold calculation 1 = Trigger level for automatic threshold calculation for direct input 2 = Set trigger threshold
`ulValue`	New value
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_EVAL_FILTER_TYPE**
Description	The function sets the evaluation filter type.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EVAL_FILTER_TYPE(unsigned short unFilterType);`
`unFilterType`	Evaluation filter type: 0 = standard filter, 1 = LF filter
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_FLAT_TOP_TIME, MMCA_SET_FLAT_TOP_TIME2**
Description	The function sets the flattop time.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_FLAT_TOP_TIME(unsigned short unFlatTopTime);`
	`ERROR_FLAG CALLBACK MMCA_SET_FLAT_TOP_TIME2(unsigned short unFlatTopTime`
	`unsigned short unFlatTopTime2);`

`unFlatTopTime`	Flattop time: 0 … maximal value * 0.1 µs
`unFlatTopTime2`	Flattop time2: 0 … maximal value * 0.1 µs
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_JITTER_CORRECTION**
Description	The function sets the jitter correction.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_JITTER_CORRECTION(unsigned short unOffOn);`
`unOffOn`	0 turns jitter correction off and ≠ 0 turns it on
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_BASELINE_RESTORING**
Description	The function sets the baseline restorer.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_BASELINE_RESTORING(unsigned char uchOption);`
`uchOption`	See CMD_SET_BASELINE_RESTORING
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_PILE_UP_REJECTION**
Description	The function sets the pile up rejection.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PILE_UP_REJECTION(unsigned short unOffOn);`
`unOffOn`	0 turns PUR off and ≠ 0 turns it on
Return value	See MCA_COMM function

Name	**MMCA_SET_FAST**
Description	The function sets the fast discriminator threshold.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_FAST(unsigned short unFast);`
`unFast`	0 ... 2499, default 400
Return value	See MCA_COMM function

Name	**MMCA_SET_SLOW**
Description	The function sets the slow discriminator threshold.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_SLOW(unsigned short unSlow);`
`unSlow`	0 ... 2499, default 400
Return value	See MCA_COMM function

Name	**MMCA_SET_PZC_MANUAL**
Description	The function sets the PZC value and returns the PZC offset.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PZC_MANUAL(unsigned short unPZC,LPSTR rec_data);`
`unPZC`	PZC value: 0 ... 2499
`rec_data`	Pointer to a SET_MEASURE_PZC structure. See CMD_SET_MEASURE_PZC, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	MCA166: Please note that this function may cause a firmware hang up if the following requirements are not meets: - statistical distributed input pulses with a count rate between 500 and 30000 Cps - the significant peak is located above the first quarter of the spectrum

Name	**MMCA_SET_PZC_VALUE**
Description	The function sets the PZC value.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PZC_VALUE(unsigned short unPZC,LPSTR rec_data);`
`unPZC`	PZC value: 0 ... 2499
`rec_data`	Pointer to a SET_MEASURE_PZC structure. However, the function has no effect on this structure. See CMD_SET_MEASURE_PZC, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	See remarks for MMCA_SET_PZC_MANUAL function.

Name	**MMCA_SET_PZC_TIME_OFFSET**
Description	The function sets the time offset for the PZC measurement.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PZC_TIME_OFFSET(unsigned short unOffset1,`
	`unsigned short unOffset3)`

`unOffset1`	Offset for low DTC: 0 ... 31, default 15
`unOffset3`	Offset for high DTC: 0 ... 31, default 15
Return value	See MCA_COMM function

Name	**MMCA_SET_GATING**
Description	The function sets the gating.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_GATING(unsigned char uchMode,`
	`unsigned char uchSignal,`
	`unsigned char uchShift);`

`uchMode`	Mode: 0 = none, 1 = discard, 2 = sort
`uchSignal`	Signal: 0 = low, 1 = high
`uchShift`	Time shift: 0 … 255 * 100 nsec
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_GATING_TIME_WINDOW_WIDTH**
Description	The function sets the width of the time windows for the gating mode 'sort by time' (see CMD_SET_GATING_TIMEWINDOW_WIDTH).

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_GATING_TIME_WINDOW_WIDTH(unsigned short unWindow,`
	`unsigned long ulWidth);`

`unWindow`	Index of the time window: 0 … 7
`ulWidth`	Time window width: 1 … 4 294 966 289
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_GATING_TIME_PER_CHANNEL**
Description	The function 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).
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_GATING_TIME_PER_CHANNEL(unsigned short unTimePerChannel);`
`unTimePerChannel`	Time per channel: 1 … 65 535
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_COMMON_MEMORY_FILL_STOP**
Description	The function sets the fill stop for the common memory. The common memory is used for acquiring raw data from the ADC (general mode = GMODE_TRANSFER_ADC_VALUES). The MCA acquires data until the fill stop is reached.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_COMMON_MEMORY_FILL_STOP(unsigned long ulFillStop);`
`ulFillStop`	0 … common memory size (see CMD_QUERY_STATE527_EX, mca_comm.h or mca32com.bas.)
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_TTL_LEVELS**
Description	The function sets the low and high level for TTL input signals.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TTL_LEVELS(unsigned short unLowLevel,`
	`unsigned short unHighLevel);`

`unLowLevel`	1 … 99 * 0.1 V
`unHighLevel`	2 … 100 * 0.1 V
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_CORE_CLOCK**
Description	The function sets the core clock of the MCA527 processor.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_CORE_CLOCK(unsigned short unClock);`
`unClock`	Clock: 1 … 6 [*100 MHz]
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_OSCI_TRIGGER**
Description	The function sets the parameters for the oscilloscope mode.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_OSCI_TRIGGER(unsigned char uchMode,`
	`char chTimeResolution,`
	`unsigned short unThreshold,`
	`unsigned short unPosition);`

`uchMode`	Mode: 0 … 4
`chTimeResolution`	Time resolution: -5 … +13
`unThreshold`	Threshold: 0 … 16383
`unPosition`	Position: 0 … 499
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

MCA Stabilization functions:

Name	**MMCA_SET_STABILISATION**
Description	The function sets the peak stabilization.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_STABILISATION(unsigned short unStateOrChannel,`
	`unsigned short unROIBegin,`
	`unsigned short unROIEnd);`

`unStateOrChannel`	Stabilization flags (see CMD_SET_STABILISATION)
`unROIBegin`	Peak ROI begin: LLD<= begin < end
`unROIEnd`	Peak ROI end: begin < end <= ULD, (end-begin) < 250
Return value	See MCA_COMM function

Name	**MMCA_SET_STAB_PARAM**
Description	The function sets the peak stabilization parameters.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_STAB_PARAM(unsigned short unStabTime,`
	`unsigned long ulStabArea);`

`unStabTime`	Stabilization time: default 10 sec interval
`ulStabArea`	Stabilization area: default 25000
Return value	See MCA_COMM function

MCA Detector Power functions:

Name	**MMCA_SET_PREAMPLIFIER_POWER**
Description	The function sets the preamplifier power.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PREAMPLIFIER_POWER(unsigned short unPreamplifierPower);`
`unPreamplifierPower`	Preamplifier switches: 0x80 -24V 0x40 +24V 0x20 -12V 0x10 +12V
Return value	See MCA_COMM function

Name	**MMCA_SET_HIGH_VOLTAGES**
Description	The function sets the detector high voltage and controls the HV-inhibit-signal.[3]

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_HIGH_VOLTAGES(unsigned short unHighVoltage,`
	`long lInhibit);`

`unHighVoltage`	High voltage: 0 ... 3600 V
`lInhibit`	High voltage inhibit: 0 = Inhibit off 1 = ”Canberra HPGe mode”, HV shut down if inhibit input < 0.5V[4] 2 = “DSG HPGe mode”, HV shut down if inhibit input < 0.5V[4] -1 = ”Ortec HPGe mode”, HV shut down if inhibit input ≥ 5V
Return value	See MCA_COMM function

[3] The high voltages will ramp up or down by the MMCA.

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

Name	**MMCA_SET_PIN5_CURRENT_SOURCE**
Description	The function switches the current source on SUB-D9 pin5 on or off.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_PIN5_CURRENT_SOURCE(unsigned char uchOption);`
`uchOption`	0 = off, 1 = on
Return value	See MCA_COMM function
Remarks	The function is only usable for MCA527.

MCA Data Setup functions:

Name	**MMCA_SET_TDF**
Description	The function sets the dead time correction parameter.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TDF(unsigned short unTdf);`
`unTdf`	100 ... 3000, default 800
Return value	See MCA_COMM function

Name	**MMCA_SET_USER_DATA**
Description	The function stores any 32 bit value in the MCA memory.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_USER_DATA(unsigned short unNumber,`
	`unsigned long ulUserData);`

`unNumber`	Number: MCA166: 0 ... 63 MCA527: 0 … 255
`ulUserData`	32 bit value
Return value	See MCA_COMM function
Remarks	Appendix 5 contains additional information about user data.

Name	**MMCA_SET_TIME**
Description	The function writes the current PC time to the internal MCA clock.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_TIME();`
Return value	See MCA_COMM function
Remarks	The function is only usable for MCA527.

Name	**MMCA_SET_UF6_ROIS**
Description	The function sets the begin and the end of a ROI used by the other UF6 functions.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_UF6_ROIS(unsigned short unNumber,`
	`unsigned short unBegin,`
	`unsigned short unEnd);`

`unNumber`	ROI number: 1 ... 3
`unBegin`	ROI begin: LLD<= begin < end
`unEnd`	ROI end: begin < end <= ULD, (end-begin) < 250
Return value	See MCA_COMM function

Name	**MMCA_SET_AHRC_PARAM**
Description	The function sets the parameters for analog high rate counting (AHRC).

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_AHRC_PARAM(unsigned short unThreshold,`
	`unsigned long ulGroupWidth);`

`unThreshold`	AHRC trigger threshold
`ulGroupWidth`	AHRC group width
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_AHRC_PARAM.

Name	**MMCA_SET_IP_ADDRESS**
Description	The function sets the IP address.
Declaration	`ERROR_FLAG CALLBACK MMCA_SET_IP_ADDRESS(unsigned char* lpuchAddress);`
`lpuchAddress`	Pointer to a 4 byte long IP address.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_IP_ADDRESS.

Name	**MMCA_COMM_ADD_ON_MODULE**
Description	The function allows to communicate with an optional add-on module.

Declaration	`ERROR_FLAG CALLBACK MMCA_COMM_ADD_ON_MODULE(unsigned char* lpuchBytes,`
	`LPSTR rec_data)`

`lpuchBytes`	Pointer to the bytes to send.
`rec_data`	Pointer to a COMM_ADD_ON_MODULE structure. See CMD_COMM_ADD_ON_MODULE, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

MCA Query functions:

Name	**MMCA_QUERY_USER_DATA**
Description	The function reads 32 user data (32 bit values) from the MCA parameter memory.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_USER_DATA(unsigned short unNumber,`
	`LPSTR rec_data);`

`unNumber`	Number of the first entry: MCA166: 0 ... 63 MCA527: 0 … 255
`rec_data`	Pointer to a QUERRY_USER_DATA structure. See CMD_QUERY_USER_DATA, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	Appendix 5 contains additional information about user data.

Name	**MMCA_QUERY_POWER**
Description	The function reads the MMCA power state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_POWER(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_POWER structure. See CMD_QUERY_POWER, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_STATE**
Description	The function reads the MCA state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_STATE(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_STATE structure. See CMD_QUERY_STATE, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_STATE527**
Description	The function reads the MCA527 state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_STATE527(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_STATE527 structure. See CMD_QUERY_STATE527, mca_comm.h or mca32com.bas. (This function converts the 32 bits value at byte offset 12 to year, month, day, hour, minute and second
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_STATE527_EX**
Description	The function reads the extended MCA527 state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_STATE527_EX(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_STATE527_EX structure. See CMD_QUERY_STATE527_EX, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_STATE527_EX2**
Description	The function reads the extended MCA527 state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_STATE527_EX2(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_STATE527_EX2 structure. See CMD_QUERY_STATE527_EX2, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_STATE527_EX3**
Description	The function reads the extended MCA527 state.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_STATE527_EX3(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_STATE527_EX3 structure. See CMD_QUERY_STATE527_EX3, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_SYSTEM_DATA**
Description	The function reads the MMCA system data.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_SYSTEM_DATA(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_SYSTEM_DATA structure. See CMD_QUERY_SYSTEM_DATA, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_SPECTRA**	
Description	The function read the MMCA spectrum data.	

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_SPECTRA(unsigned short unFirstChannel,`
	`unsigned long ulCompressFactor,`
	`LPSTR rec_data)`	

`unFirstChannel`	Number of first channel (bit 11 … 0): and buffer control (bit 15 … 12):	0 … channels count – 32 * compress factor – 1 0 … 15 (see CMD_QUERY_SPECTRA_EX).
`ulCompressFactor`	Compress factor	(MCA166: 1 ... 32, MCA527: 1 … 128)
`rec_data`	Pointer to a QUERRY_SPECTRA structure. See CMD_QUERY_SPECTRA, mca_comm.h or mca32com.bas.	
Return value	See MCA_COMM function	

Name	**MMCA_QUERY_SPECTRA_EX**
Description	The function read the MMCA spectrum data.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_SPECTRA_EX(unsigned short unFirstChannel,`
	`unsigned short unCompressFactor,`
	`unsigned short unBufferControl,`
	`LPSTR rec_data);`

`unFirstChannel`	Number of first channel: 0 … channels count – 32 * compress factor - 1
`unCompressFactor`	Compress factor: (MCA166: 1 ... 32, MCA527: 1 … 128)
`unBufferControl`	Buffer control (see CMD_QUERY_SPECTRA_EX).
`rec_data`	Pointer to a QUERRY_SPECTRA structure. See CMD_QUERY_SPECTRA, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function can be used for both MCA166 and MCA527. The function decides based on the parameters which firmware command (CMD_QUERY_SPECTRA or CMD_QUERY_SPECTRA_EX) has to be applied. The developer has to make sure that the parameters suit to the connected MCA type.

Name	**MMCA_QUERY_SPECTRA_EX2**
Description	The function read the MMCA spectrum data.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_SPECTRA_EX2(unsigned short unFirstChannel,`
	`unsigned short unCompressFactor,`
	`unsigned short unBufferControl,`
	`LPSTR rec_data);`

`unFirstChannel`	Number of first channel: 0 … channels count – 32 * compress factor - 1
`unCompressFactor`	Compress factor: (MCA166: 1 ... 32, MCA527: 1 … 128)
`unBufferControl`	Buffer control (see CMD_QUERY_SPECTRA_EX).
`rec_data`	Pointer to a QUERRY_SPECTRA_EX2 structure. See CMD_QUERY_SPECTRA_EX2, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function can only be used for MCA527.

Name	**MMCA_QUERY_IMAGE**
Description	The function reads the MCA166 spectrum image (120x256). It should only be used to display the spectrum in a 120 x 256 pixel window.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_IMAGE(unsigned short unFirstChannel,`
	`unsigned short unCompressFactor,`
	`unsigned short unVerticalFullScale,`
	`LPSTR rec_data);`

`unFirstChannel`	Number of first channel
`unCompressFactor`	Compress factor: 1 ... 32
`unVerticalFullScale`	Vertical full scale
`rec_data`	Pointer to a QUERRY_IMAGE structure. See mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA166.

Name	**MMCA_QUERY_ENRICHMENT**		
Description	The function reads current dead time, real time and the integral of two ROIs.		

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_ENRICHMENT(unsigned short unRoi1Begin,`
	`unsigned short unRoi1End,`
	`unsigned short unRoi2Begin,`
	`unsigned short unRoi2End,`
	`LPSTR rec_data);`		

`unRoi1Begin`	Peak begin	integer value	LLD<= begin < end
`unRoi1End,`	Peak end	integer value	begin < end <= ULD, (end-begin) < 250
`unRoi2Begin`	Background begin	integer value	LLD<= begin < end
`unRoi2End`	Background end	integer value	begin < end <= ULD, (end-begin) < 250
`rec_data`	Pointer to a QUERRY_ENRICHMENT structure. See CMD_QUERY_ENRICHMENT, mca_comm.h or mca32com.bas.		
Return value	See MCA_COMM function		

Name	**MMCA_QUERY_UF6_INFO**
Description	The function reads current dead time, real time the integral, begin and end of the UF6 ROIs
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_UF6_INFO(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_UF6_INFO structure. See CMD_QUERY_UF6_INFO, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_UF6_ROIS**
Description	The function reads the begin and end of the 3 ROIs
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_UF6_ROIS(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_UF6_ROIS structure. See CMD_QUERY_UF6_ROIS, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_CENTROID**
Description	The function reads peak centroid of the specified ROI

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_CENTROID(unsigned short unRoiBegin,`
	`unsigned short unRoiEnd,`
	`LPSTR rec_data)`

`unRoiBegin`	Peak ROI begin: LLD<= begin < end
`unRoiEnd`	Peak ROI end: begin < end <= ULD, (end-begin) < 250
`rec_data`	Pointer to a QUERRY_CENTROID structure. See CMD_QUERY_CENTROID, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_VOLTAGE_CURRENT**
Description	The function reads the voltages and currents from the MCA.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_VOLTAGE_CURRENT(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_VOLTAGE_CURRENT structure. See CMD_QUERY_VOLTAGE_CURRENT, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function

Name	**MMCA_QUERY_COMMON_MEMORY**
Description	The function reads the common memory from the MCA527.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_COMMON_MEMORY(unsigned long ulOffset,`
	`unsigned short* lpunData);`

`ulOffset`	Even-numbered offset to be read from.
`lpunData`	Pointer to an array of 720 unsigned short values. See CMD_QUERY_COMMON_MEMORY, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_HISTOGRAM**
Description	The function causes the MCA527 to acquire a histogram from 500.000 ADC samples (= 50 ms).

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_HISTOGRAM(unsigned short unOffset,`
	`unsigned short unCompressFactor,`
	`unsigned long* lpulData);`

`unOffset`	Offset to be read from (0 ... 16384 – 256 * class interval width).
`unCompressFactor`	Class interval width (1, 2, 4 ... 64).
`lpunData`	Pointer to an array of 256 unsigned long values.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_QUERY_HISTORY, mca_comm.h or mca32com.bas.

Name	**MMCA_QUERY_OSCI_SCREEN**
Description	The function reads the data for the oscilloscope screen.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_OSCI_SCREEN(unsigned long ulPosition,`
	`LPSTR rec_data);`

`ulPosition`	-1 or the previously returned position.
`rec_data`	Pointer to a QUERY_OSCI_SCREEN structure. See CMD_QUERY_OSCI_SCREEN or mca_comm.h.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.

Name	**MMCA_QUERY_OSCI_SCREEN_EX**
Description	The function reads the data for the oscilloscope screen. Besides, it initiates the convolution of the data with the trigger filter and the main filter. The results can be read afterwards with the functions MMCA_QUERY_OSCI_TRIGGER_FILTER_RESULTS and MMCA_QUERY_OSCI_MAIN_FILTER_RESULTS.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_OSCI_SCREEN_EX(unsigned short unFlags,`
	`LPSTR rec_data);`

`unFlags`	Bit 0: convolve data with trigger filter Bit 1: convolve data with main filter
`rec_data`	Pointer to a QUERY_OSCI_SCREEN_EX structure. See CMD_QUERY_OSCI_SCREEN_EX or mca_comm.h.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. The oscilloscope mode is described in document: “MCA527 Oscilloscope Mode”.

Name	**MMCA_QUERY_OSCI_TRIGGER_FILTER_RESULTS**
Description	The function returns the results of the convolution of the oscilloscope data with the trigger filter. The calculation is initiated by the function MMCA_QUERY_OSCI_SCREEN_EX.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_OSCI_TRIGGER_FILTER_RESULTS(unsigned short unFlag,`
	`LPSTR rec_data);`

`unFlag`	0 = read 360 values, ≠0 = read 720 values.
`rec_data`	Pointer to a QUERY_OSCI_TRIGGER_FILTER_RESULTS structure. See CMD_QUERY_OSCI_TRIGGER_FILTER_RESULTS or mca_comm.h.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_OSCI_MAIN_FILTER_RESULTS**
Description	The function returns the results of the convolution of the oscilloscope data with the main filter. The calculation is initiated by the function MMCA_QUERY_OSCI_SCREEN_EX.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_OSCI_MAIN_FILTER_RESULTS(LPSTR rec_data);`
`rec_data`	Pointer to a QUERY_OSCI_MAIN_FILTER_RESULTS structure. See CMD_QUERY_OSCI_MAIN_FILTER_RESULTS or mca_comm.h.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_AHRC_HISTOGRAM**
Description	The function causes the MCA527 to acquire a corresponding histogram.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_AHRC_HISTOGRAM(unsigned short unClassWidth,`
	`LPSTR rec_data);`

`unClassWidth`	Class interval width (1, 2, 4 ... 32768).
`rec_data`	Pointer to an array of 360 unsigned long values. See CMD_QUERY_AHRC_HISTOGRAM, mca_comm.h or mca32com.bas.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_ADJUSTMENT_TABLE**
Description	The function 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.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_ADJUSTMENT_TABLE(LPSTR rec_data);`
`rec_data`	Pointer to a QUERRY_ADJUSTMENT_TABLE structure. See CMD_QUERY_ADJUSTMENT_TABLE or mca_comm.h.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_QUERY_ON_LINE**
Description	The function checks the communication between the MCA and the PC.
Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_ON_LINE();`
Return value	See MCA_COMM function

MCA527 Extension Port functions:

Name	**MMCA_SET_EXTENSION_PORT**
Description	The function configures the parts of the extension port..

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_PORT(unsigned char uchA,`
	`unsigned char uchB,`
	`unsigned char uchC,`
	`unsigned char uchD,`
	`unsigned char uchE,`
	`unsigned char uchF)`

`uchA … uchF`	The parameters assign the parts their job.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_EXTENSION_PORT.

Name	**MMCA_SET_EXTENSION_POLARITY**
Description	The function sets the polarity for the parts of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_POLARITY(unsigned short unPart,`
	`unsigned short unPolarity);`

`unPart`	Part of the extension port.
`unPolarity`	Polarity
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_EXTENSION_POLARITY.

Name	**MMCA_SET_EXTENSION_PULSER_PERIOD**
Description	The function sets the pulser period for the parts of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_PULSER_PERIOD(unsigned short unPart,`
	`unsigned long ulPeriod);`

`unPart`	Part of the extension port.
`ulPeriod`	Pulser period
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_EXTENSION_PULSER_PERIOD.

Name	**MMCA_SET_EXTENSION_PULSER_WIDTH**
Description	The function sets the pulser width for the parts of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_PULSER_PERIOD(unsigned short unPart,`
	`unsigned long ulWidth);`

`unPart`	Part of the extension port.
`ulWidth`	Pulser width
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_EXTENSION_PULSER_WIDTH.

Name	**MMCA_SET_EXTENSION_RS232**
Description	The function configures the RS232 interface of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_RS232(unsigned short unBaudrate,`
	`unsigned char uchFlags);`

`unBaudrate`	Baud rate
`uchFlags`	Flags
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527. See CMD_SET_EXTENSION_RS232.

Name	**MMCA_WRITE_TO_EXTENSION**
Description	The function writes a string to the RS232 interface of the extension port.
Declaration	`ERROR_FLAG CALLBACK MMCA_WRITE_TO_EXTENSION(LPSTR lpszBuffer);`
`lpszBuffer`	Pointer to a null-terminated string.
Return value	See MCA_COMM function
Remarks	This function is only usable for MCA527.

Name	**MMCA_READ_FROM_EXTENSION, MMCA_READ_FROM_EXTENSION_EX**
Description	The function reads a string from the RS232 interface of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_READ_FROM_EXTENSION(LPSTR lpszBuffer,`
	`unsigned short unBufferSize);`
	`ERROR_FLAG CALLBACK MMCA_READ_FROM_EXTENSION_EX(unsigned short unOption,`
	`LPSTR lpszBuffer,`
	`unsigned short unBufferSize);`

`unOption`	0 = current data 1 = buffered data 2 = buffered data, unlock buffer 3 = buffered data, lock buffer
`lpszBuffer`	Pointer to a null-terminated string.
Return value	See MCA_COMM function,
Remarks	This function is only usable for MCA527. See CMD_QUERY_EXTENSION_RS232_RX.

Name	**MMCA_QUERY_EXTENSION_RS232_RX**
Description	The function reads bytes from the RS232 interface of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_QUERY_EXTENSION_RS232_RX(unsigned short unOption,`
	`LPSTR rec_data);`

`unOption`	0 = current data 1 = buffered data 2 = buffered data, unlock buffer 3 = buffered data, lock buffer
`rec_data`	Pointer to a QUERY_EXTENSION_RS232_RX structure. See CMD_QUERY_EXTENSION_RS232_RX or mca_comm.h.
Return value	See MCA_COMM function,
Remarks	This function is only usable for MCA527.

Name	**MMCA_SET_EXTENSION_OUTPUT, MMCA_SET_EXTENSION_OUTPUT_EX**
Description	The function sets the selected output of the extension port.

Declaration	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_OUTPUT(unsigned short unRelevantParts,`
	`unsigned short unPartB,`
	`unsigned short unPartD);`
	`ERROR_FLAG CALLBACK MMCA_SET_EXTENSION_OUTPUT_EX(unsigned short unRelevantParts,`
	`unsigned short unPartB,`
	`unsigned short unPartD,`
	`unsigned short unPartF);`

`unRelevantParts`	1 = part B (output2) 3 = part D (output1) 5 = part F (output3) (Do not use with MMCA_SET_EXTENSION_OUTPUT.) 7 = part B (output2) and part D (output1) 9 = part B (output2) and part F (output3) (Do not use with MMCA_SET_EXTENSION_OUTPUT.) 11 = part D (output1) and part F (output3) (Do not use with MMCA_SET_EXTENSION_OUTPUT.) 255 = all parts (Do not use with MMCA_SET_EXTENSION_OUTPUT.)

`unPartB`
`unPartD`
`unPartF`
	0 = off, ≠0 = on

Return value	See MCA_COMM function,
Remarks	These functions are only usable for MCA527. See CMD_SET_EXTENSION_OUTPUT.

5 Managing the USB port

There are special functions for the USB port. The function COMM_INIT_ETHERNET returns information about the communication port to the argument `init.unInterfaceId`. This parameter shows whether the communication port is an USB port and whether the MCA batteries are rechargeable over the USB interface.

Name	**GET_USB_CHARGER**
Description	The function gets information as to whether USB charging is switched on or off.
Declaration	`VB_BOOL CALLBACK GET_USB_CHARGER();`
Return value	If nonzero, the USB charging is switched on.

Name	**SET_USB_CHARGER**
Description	The function switches USB charging on or off.
Declaration	`void CALLBACK SET_USB_CHARGER(VB_BOOL bCharge);`
`bCharge`	If nonzero, the USB charging will be switched on otherwise off.

Name	**GET_USB_CHARGER_STATE**
Description	The function gets the USB charging state. The MCA batteries are recharged over USB if USB charging is switched on and no external charger is plugged.
Declaration	`VB_BOOL CALLBACK GET_USB_CHARGER();`
Return value	If nonzero, the MCA batteries are recharged over USB.

Name	**GET_CURRENT_LATENCY_TIMER**
Description	The function gets the latency time of the USB port.
Declaration	`VB_BOOL CALLBACK GET_CURRENT_LATENCY_TIMER(LPDWORD lpdwTime);`
`lpdwTime`	Pointer to the returned latency time.
Return value	If the function succeeds, the return value is nonzero.

Name	**SET_CURRENT_LATENCY_TIMER**
Description	The function sets the latency time of the USB port. Changing the latency time can speed up the communication.
Declaration	`VB_BOOL CALLBACK SET_CURRENT_LATENCY_TIMER(DWORD dwTime);`
`dwTime`	Latency time
Return value	If the function succeeds, the return value is nonzero.

6 MS-Windows DLL for the Communication with Several MCAs

There is a further dynamic link library (mca32cmx.dll) which allows applications to communicate with several MCAs simultaneously. In order to include the DLL in your program, you have to use the file MCA32CMX.BAS for Visual Basic or the files MCA_COMX.H and MCA32CMX.LIB for C/C++. All functions and structures are declared in these files.

The function names of mca32cmx.dll. are identical to the function names of mca32com.dll. However, not all functions are provided. The initialization functions COMM_INIT, COMM_INIT_EX, COMM_INIT_EX2, COMM_INIT_EX3, COMM_INIT_ETHERNET and COMM_INIT_ETHERNET_IP has been omitted.

Use the function COMM_INIT_SELECT for initialization. The function works like the corresponding function within mca32com.dll, with one exception, there is an additional parameter and the structure MCA_COMM_INIT has two additional members.

```
short CALLBACK COMM_INIT_SELECT(unsigned short* lpunIndex, MCA_COMM_INIT* init,
                                clock_t lTimeOut,int nTryAgain, HWND hWnd);
```

Name	**MCA_COMM_INIT**
Description	A pointer of the structure has to be passed to the `COMM_INIT_SELECT` function.

Declaration	`struct MCA_COMM_INIT`
	`{ unsigned short  unSize;`
	`  unsigned short  unNumberToFind;`
	`  VB_BOOL         bMca166;`
	`  unsigned short  unMca166FwVersion;`
	`  VB_BOOL         bMca527;`
	`  unsigned short  unMca527FwVersion;`
	`  short           nShowLogo;`
	`  short           nUserInterface;`
	`  unsigned short  unInterfaceId;`
	`  unsigned long   ulBaudrate;`
	`  char            szCaption[60];`
	`  VB_BOOL         bRedetect;`
	`  char            szHelpFile[MAX_PATH];`
	`  unsigned long   ulHelpId;`
	`  VB_BOOL         bRS485;`
	`};`

`unSize`	[In] Size of the structure.
`unNumberToFind`	[In] If this number is nonzero, the function tries to initializes the communication to the MCA with the corresponding serial number. If the communication with this MCA is possible, the communication is initialized and the function returns. If the corresponding MCA is not detected or not usable, all detected MCA are listed within a dialog window. If this number is zero, all detected MCA are always listed within a dialog window.
`bMca166`	[In] If this parameter is nonzero, the MCA-166 is assumed to be applicable.
`unMca166FwVersion`	[In] This parameter sets the earliest MCA-166 firmware version that is assumed to be applicable.
`bMca527`	[In] If this parameter is nonzero, the MCA-527 is assumed to be applicable.
`unMca527FwVersion`	[In] This parameter sets the earliest MCA-527 firmware version that is assumed to be applicable.
`nShowLogo`	[In] If this parameter is equal to 0, no logo is displayed. If this parameter is equal to -1, the GBS logo is displayed.
`nUserInterface`	[In] This parameter can be one of the following values: USER_INTERFACE_VERBOSE (= -1): The function displays message windows for signaling the work progress and asks for the serial number if necessary. USER_INTERFACE_RETICENT (= 0): The function asks for the serial number if necessary. USER_INTERFACE_MUTE (= 1): There is no interaction with the user. This value must not used if `unNumberToFind` is equal to 0.

Name	**MCA_COMM_INIT** ***(Continuation)***
`unInterfaceId`[5]	[In] If this parameter is unequal to 0xFFFF, it determines the communication port that shall be used. [Out] This parameter returns the ID of the used communication port.
`ulBaudrate`	[Out] If the communication port is RS232, RS485 or USB, the parameter returns the current baud rate.
`szCaption[60]`	[In] This null-terminated string sets the caption of the dialog window. If it is empty, the default caption is displayed.
`bRedetect`	[In] In order to connect several MCAs, the function has to be called several times. At the first function call, the function detects MCAs definitely. At subsequent function calls, the function redetects MCAs only if this parameter is nonzero, otherwise it uses the previously detected MCAs. If none previously detected MCAs are available, the function redetects MCAs regardless of this parameter.
`szHelpFile`	[In] If this parameter contains the path and the name of a help file (*.hlp) and the help index is unequal to 0xFFFFFFFF, the help button within the dialog window is enabled. The existence of the help file is not checked.
`ulHelpId`	[In] If this parameter is unequal to 0xFFFFFFFF and the previous parameter does not contain an empty string, the help button within the dialog window is enabled.
`bRS485`	[In] If this parameter is nonzero, the function searches exclusively for the MCA with the serial number that is passed in `unNumberToFind` at all already opened serial communication interfaces. This method is useful for detecting further MCAs within a RS485 bus system because it avoids redundant tasks.

If the function succeeds, the additional parameter `lpunIndex` returns the index for the communication path that has to been passed to all other functions. The index is always the first parameter. All functions are defined corresponding to the following template.

```
return_type CALLBACK function_name(unsigned short unIndex[,parameters_equal_to_mca32com.dll]);
```

The functions IS_USB_PORT, SET_LATENCY_TIMER and SET_LATENCY_TIMER_EX has also been omitted. All omitted functions are no more needed. There are alternative functions for it. Mca32com.dll provides these functions only for compatibility reasons.

[5] See structure `MCA_COMM_PARAM` (chapter 2)